ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Ανάλυση της συσχέτισης γονιδίων με τον καρκίνο του ήπατος χρησιμοποιώντας μεθόδους Μηχανικής Μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΓΓΕΛΟΥ ΚΟΥΡΟΥΠΕΤΡΟΓΛΟΥ

Επιβλέπων : **Κωνσταντίνος Διαμαντάρας**
Καθηγητής, ΑΤΕΙ Θεσσαλονίκης

Θεσσαλονίκη, Ιούνιος 2018

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEBINTELLIGENCE

Ανάλυση της συσχέτισης γονιδίων με τον καρκίνο του ήπατος χρησιμοποιώντας μεθόδους Μηχανικής Μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΓΓΕΛΟΥ ΚΟΥΡΟΥΠΕΤΡΟΓΛΟΥ

Επιβλέπων : **Κωνσταντίνος Διαμαντάρας**
Καθηγητής, ΑΤΕΙ Θεσσαλονίκης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις Choose a date.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο	Όνομα Επώνυμο	Όνομα Επώνυμο
Choose an item.Α.Τ.Ε.Ι.Θ.	Choose an item.Α.Τ.Ε.Ι.Θ.	Choose an item.Α.Τ.Ε.Ι.Θ.

Θεσσαλονίκη, Ιούνιος 2018

(Υπογραφή)

.....

Άγγελος Κουρουπέτρογλου

Μηχανικός Πληροφορικής Α.Τ.Ε.Ι.Θ.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα Καθηγητή μου, κ. Διαμαντάρα Κωνσταντίνο, για την εμπιστοσύνη, την πολύτιμη καθοδήγηση, την υπομονή και την επαγγελματική του στάση κατά την διάρκεια εκπόνησης αυτής της διπλωματικής εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω την Επίκουρη Καθηγήτρια του Τμήματος Βιολογίας του Α.Π.Θ., κα. Ντάφου Δήμητρα και τον υποψήφιο Διδάκτορα του Τμήματος Βιολογίας του Α.Π.Θ., κ. Μώκο Παναγιώτη για την βοήθεια τους στην κατανόηση του συνόλου δεδομένων που χρησιμοποιήσαμε. Ευχαριστώ επίσης τον Διαχειριστή των Συστημάτων του Ινστιτούτου Εφαρμοσμένων Βιοεπιστημών, κ. Χουβαρδά Βασίλειο για την τεχνική του υποστήριξη και την διάθεση του απαραίτητου εξοπλισμού για την εκτέλεση των πειραμάτων.

Τέλος, ευχαριστώ την οικογένειά μου, τους γονείς μου Νικόλαο και Ελένη, και τον αδελφό μου Χρήστο, για τη συνεχή και πολύτιμη ενθάρρυνση και υποστήριξη που μου παρείχαν κατά τη διάρκεια του κύκλου σπουδών μου.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι ο εντοπισμός ενός μικρού αριθμού γονιδίων, με την χρήση μεθόδων Μηχανικής Μάθησης, τα οποία περιέχουν ικανή πληροφορία για την κατασκευή ενός ταξινομητή που θα διαθέτει την ικανότητα της γενίκευσης. Το σύνολο δεδομένων που εξετάσαμε αφορά τον καρκίνο του ήπατος και προέρχεται από την βάση δεδομένων του TCGA (The Cancer Genome Atlas). Οι μεταβλητές στόχοι είναι ο βαθμός του καρκίνου του ήπατος (Grades) και το στάδιο του καρκίνου του ήπατος (Stages). Οι τιμές των γονιδίων αποτελούν τιμές έκφρασης μετασχηματισμένες από αλληλούχιση RNA (RNA-seq). Αρχικά, με την χρήση των μεθόδων μείωσης διαστάσεων PCA, Fisher Score, Mutual Information, Kolmogorov Smirnov 2 Samples, ReliefF και mRMR MIQ αξιολογήσαμε τα γονίδια και τα ταξινομήσαμε βάση της σημαντικότητάς τους. Έπειτα, επιλέξαμε τα πρώτα 500 σημαντικά γονίδια της κάθε μεθόδου αξιολόγησης χαρακτηριστικών. Χρησιμοποιώντας τα πρώτα 10 σημαντικά γονίδια εκτελέσαμε εξαντλητική αναζήτηση (Grid Search) με χρήση των εκτιμητών SVM Linear και SVM RBF για την εύρεση των παραμέτρων που ο κάθε ταξινομητής πετυχαίνει την καλύτερη τιμή μέσης ακρίβειας (20 Fold Cross Validation). Στην συνέχεια επαναλάβαμε την παραπάνω διαδικασία προσθέτοντας κάθε φορά τα επόμενα 10 σημαντικά γονίδια έως και τα 500. Η ίδια διαδικασία ακολουθήθηκε και για τον έλεγχο όλων των υποσυνόλων τομών και ενώσεων που προκύπτουν από τον συνδυασμό σε ζεύγη όλων των μεθόδων επιλογής χαρακτηριστικών. Τα αποτελέσματα των πειραμάτων έδειξαν ότι το κριτήριο mRMR MIQ υπερέχει των υπόλοιπων κριτηρίων αξιολόγησης που εξετάσαμε. Σε σχέση και με τις δύο μεταβλητές στόχους, το κριτήριο mRMR MIQ κατάφερε να εντοπίσει τον αντίστοιχο μικρότερο αριθμό σημαντικών γονιδίων (χαρακτηριστικών) τα οποία περιέχουν πληροφορία ικανή για την κατασκευή ενός ταξινομητή SVM RBF ο οποίος διαθέτει καλύτερη ικανότητα γενίκευσης έναντι άλλων υποσυνόλων σημαντικών γονιδίων που προήλθαν από τα υπόλοιπα κριτήρια αξιολόγησης που εξετάσαμε.

Λέξεις Κλειδιά: Μηχανική Μάθηση, Μέθοδοι Μείωσης Διαστάσεων, PCA, Fisher Score, Mutual Information, Kolmogorov-Smirnov 2 Samples, ReliefF, mRMR MIQ, SVM, Καρκίνος του Ήπατος, Γονίδια, Αλληλούχιση RNA (RNA-seq), Επιλογή Γονιδίων

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The purpose of this thesis is to identify a small number of genes using Machine Learning methods, which contain sufficient information for the construction of a classifier capable of generalization. The dataset we examined concerns liver cancer and is from the TCGA database (The Cancer Genome Atlas). The target variables were the grade (Grades) and the stage of liver cancer (Stages). Gene expression data are transformed by RNA sequencing (RNA-seq). Initially, using the dimensional reduction methods PCA, Fisher Score, Mutual Information, Kolmogorov Smirnov 2 Samples, ReliefF, and mRMR MIQ, we evaluated the genes and ranked them based on their significance. Then, we chose the first 500 significant genes of each feature selection method and using the first 10 significant genes, we performed a Grid Search using the SVM Linear and SVM RBF classifiers to find the parameters that each classifier achieves the best mean accuracy value (20 Fold Cross Validation). We then repeated the above procedure by adding the next 10 significant genes each time up to 500. This process was also followed to examine the union and intersection subsets between all combinations of feature selection methods. The results of the experiments showed that the mRMR MIQ criterion is better than the other validation criteria we examined. With respect to both target variables, the mRMR MIQ criterion was able to identify the corresponding smaller number of significant genes that contain information capable of constructing an SVM RBF classifier that has the best generalization capability of all other subsets of important genes examined.

Keywords: Machine Learning, Dimensionality Reduction, PCA, Fisher Score, Mutual Information, Kolmogorov-Smirnov 2 Samples, ReliefF, mRMR MIQ, SVM, Liver Cancer, Genes, RNA-seq Expression, Gene Selection

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Μηχανική Μάθηση στην Βιοπληροφορική

Οι επιστήμες της Πληροφορικής και της Βιολογίας, κατά την διάρκεια των τελευταίων δεκαετιών έχουν χαρακτηριστεί από σημαντικές εξελίξεις. Στην επιστήμη της Πληροφορικής, και συγκεκριμένα στο ευρύτερο επιστημονικό πεδίο της Τεχνητής Νοημοσύνης, οι σημαντικές εξελίξεις αντικατοπτρίζονται από την ανάπτυξη τεχνικών, μεθόδων και αλγορίθμων για την αυτοματοποίηση δραστηριοτήτων που σχετίζονται με την ανθρώπινη σκέψη, όπως η λήψη αποφάσεων, η επίλυση προβλημάτων και η μάθηση. Η Μηχανική Μάθηση αποτελεί έναν κλάδο της Τεχνητής Νοημοσύνης όπου υπήρξε αλματώδης ανάπτυξη μεθόδων και αλγορίθμων που έχουν ως σκοπό να καταστήσουν μια μηχανή (αλγόριθμο) έμπειρη, βελτιώνοντας την αποτελεσματικότητά της για μία συγκεκριμένη λειτουργία.

Στην επιστήμη της Βιολογίας, οι σημαντικές εξελίξεις αντικατοπτρίζονται από την επιτυχή διεξαγωγή του Προγράμματος Αποκρυπτογράφησης του Ανθρώπινου Γονιδιώματος[1] (The Human Genome Project, HGP) το οποίο αποτελεί ένα από τα μεγαλύτερα ερευνητικά προγράμματα της σύγχρονης επιστήμης. Το Πρόγραμμα Αποκωδικοποίησης του Ανθρώπινου Γονιδιώματος είναι ένα διεθνές επιστημονικό ερευνητικό πρόγραμμα που είχε ως στόχο τον καθορισμό της ακολουθίας των ζευγών βάσεων που αποτελούν το ανθρώπινο DNA καθώς επίσης και την εύρεση των γονιδίων του ανθρώπινου γονιδιώματος. Το

[1] https://web.ornl.gov/sci/techresources/Human_Genome/index.shtml

πρόγραμμα ξεκίνησε το 1990, η πρώτη καταγραφή του γονιδιώματος δημοσιεύτηκε το 2000 και η επίσημη δημοσίευση έγινε το 2003.

Οι παραπάνω σημαντικές εξελίξεις οδήγησαν τις δύο επιστήμες στην ανάγκη συνεργασίας η οποία διαρκώς αυξάνεται, όσο οι δύο επιστήμες συνεχίζουν να εξελίσσονται. Η εξέλιξη της Πληροφορικής προσφέρει στην επιστήμη της Βιολογίας τα μέσα για την διαχείριση και την ανάλυση των βιολογικών δεδομένων με αποτέλεσμα την εξέλιξη της. Από την άλλη η εξέλιξη της Βιολογίας έχει ως αποτέλεσμα την διαρκή δημιουργία ερωτημάτων τα οποία απαιτούν την εξέλιξη της Πληροφορικής για να απαντηθούν. Η Βιοπληροφορική αποτελεί την διεπιστημονική ερευνητική περιοχή, η οποία προήλθε από την ανάγκη συνεργασίας των επιστημών της Πληροφορικής και της Βιολογίας.

Η Μηχανική Μάθηση χρησιμοποιείται σε αρκετές εφαρμογές της Βιοπληροφορικής, καθώς αποτελεί ένα από τα πολυτιμότερα εργαλεία για την διαχείριση του τεράστιου όγκου δεδομένων που προκύπτουν από βιολογικές αναλύσεις, με απώτερο σκοπό την εξόρυξη γνώσης από αυτά. Συγκεκριμένα, μέθοδοι και αλγόριθμοι Μηχανικής Μάθησης έχουν χρησιμοποιηθεί για την διάγνωση και πρόγνωση ασθενειών, για την υποστήριξη λήψης ιατρικών αποφάσεων, για την διαχείριση, την επεξεργασία και την ανάλυση βιολογικών και κυτταρικών δεδομένων με σκοπό την εξαγωγή χρήσιμων συμπερασμάτων.

Ένα από τα μεγαλύτερα προβλήματα στην ανάπτυξη υπολογιστικών μοντέλων για την ανάλυση βιολογικών δεδομένων και εξόρυξη γνώσης από αυτά, είναι ο πολύ μεγάλος αριθμός μεταβλητών (χαρακτηριστικών) σε σχέση με τον μικρό αριθμό των δειγμάτων που είναι διαθέσιμα. Το πρόβλημα αυτό, γνωστό ως το φαινόμενο της ‘κατάρας των πολλών διαστάσεων’ στην Μηχανική Μάθηση, αντιμετωπίζεται με την επιλογή μεταβλητών (χαρακτηριστικών). Κατά την επιλογή χαρακτηριστικών, τα χαρακτηριστικά μειώνονται σε ένα μικρότερο υποσύνολο χαρακτηριστικών, επιλέγοντας τα περισσότερο σημαντικά και σχετικά με την εκάστοτε εφαρμογή.

Για παράδειγμα, αν τα βιολογικά δεδομένα εκφράζονται μέσω ενός τύπου γονιδιακής έκφρασης (π.χ. αλληλούχιση RNA), λόγω του μεγάλου αριθμού γονιδίων (χαρακτηριστικών), εφαρμόζονται τεχνικές επιλογής γονιδίων σε σχέση με την ύπαρξη μιας συγκεκριμένης κατάστασης (μεταβλητή στόχος). Ο σκοπός της επιλογής υποψηφίων γονιδίων (χαρακτηριστικών) είναι η χρήση τους σε μοντέλα μηχανικής μάθησης, ώστε να ελεγχθεί αν διαθέτουν ικανή πληροφορία ώστε να μπορούν να χρησιμοποιηθούν ως ενδείξεις για την ύπαρξη μιας συγκεκριμένης κατάστασης.

Δηλαδή, στην περίπτωση που η εξεταζόμενη κατάσταση της εφαρμογής αποτελείται από μία δίτιμη μεταβλητή, τότε εξετάζεται αν η κατασκευή ενός μοντέλου μηχανικής μάθησης με την χρήση ενός υποψήφιου υποσυνόλου γονιδίων (χαρακτηριστικών) θα αποκτά την ικανότητα γενίκευσης. Δηλαδή την ικανότητα να προβλέπει την τιμή της δίτιμης μεταβλητής με χρήση

δεδομένων που δεν έχει δει κατά την εκπαίδευση του. Συνεπώς, το υποσύνολο των γονιδίων το οποίο συμμετέχει ως είσοδος στην κατασκευή ενός αποδοτικού μοντέλου μηχανικής μάθησης με μεγάλη ικανότητα γενίκευσης, θεωρείται καθοριστικό για την πρόβλεψη της εξεταζόμενης κατάστασης.

Ο εντοπισμός των σημαντικών (καθοριστικών) γονιδίων τα οποία διαθέτουν την ικανή πληροφορία για την κατασκευή ενός μοντέλου μηχανικής μάθησης που διαθέτει την ικανότητα γενίκευσης αποτελεί ένα από τα σημαντικότερα ερευνητικά πεδία της Βιοπληροφορικής.

1.2 Αντικείμενο Διπλωματικής

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάλυση και η αξιολόγηση 12604 γονιδίων με χρήση μεθόδων μηχανικής μάθησης (μέθοδοι μείωσης διαστάσεων), σε σχέση με 2 δίτιμες καταστάσεις (2 δίτιμες μεταβλητές στόχοι) που αφορούν τον καρκίνο του ήπατος. Ο απώτερος στόχος, για κάθε δίτιμη κατάσταση, είναι ο εντοπισμός και η χρήση του κατάλληλου (καθοριστικού) υποσυνόλου γονιδίων, για την κατασκευή ενός αντίστοιχου αποδοτικού ταξινομητή. Δηλαδή, για κάθε δίτιμη κατάσταση, θα πρέπει να καταλήξουμε σε ένα καθοριστικό υποσύνολο γονιδίων που περιέχει ικανή πληροφορία για την κατασκευή ενός ταξινομητή ο οποίος θα διαθέτει την όσο το δυνατόν καλύτερη ικανότητα γενίκευσης.

Οι τιμές των γονιδίων του συνόλου δεδομένων που εξετάζουμε, αποτελούν τιμές έκφρασης μετασχηματισμένες από αλληλούχιση RNA (RNA-seq). Η αλληλούχιση RNA αποτελεί μία μέθοδο ποσοτικοποίησης η οποία εφαρμόζεται σε βιολογικά δεδομένα για να διερευνηθεί η διαφορική έκφραση γονιδίων.

Οι δίτιμες καταστάσεις στόχοι (μεταβλητές στόχοι) που αφορούν τον καρκίνο του ήπατος είναι ο βαθμός του καρκίνου του ήπατος (Grades) και το στάδιο του καρκίνου του ήπατος (Stages). Ο βαθμός του καρκίνου δηλώνει το κατά πόσο ο καρκίνος που έχει διαγνωστεί σε έναν ασθενή, σε μία δεδομένη χρονική στιγμή, μπορεί εύκολα να θεραπευτεί (αντιμετωπιστεί). Οι τιμές της κατάστασης (μεταβλητή στόχος) που αφορά τον βαθμό του καρκίνου είναι Early_Grade και Late_Grade. Το στάδιο του καρκίνου δηλώνει το πόσο «προχωρημένος» είναι ο καρκίνος που διαγνώστηκε σε μία δεδομένη χρονική στιγμή. Με την έννοια «προχωρημένος», νοείται το μέγεθος του καρκίνου, το πόσο έχει εξαπλωθεί στο ήπαρ, αν ο καρκίνος έχει κάνει μετάσταση, το στάδιο της μετάστασης και το εύρος αυτής. Οι τιμές της κατάστασης (μεταβλητή στόχος) που αφορά το στάδιο του καρκίνου είναι Early_Stage και Late_Stage.

Οι τεχνικές δυσκολίες που αντιμετωπίσαμε κατά την εκτέλεση των πειραμάτων ήταν οι απαιτήσεις σε υπολογιστική δύναμη. Συνυπολογίζοντας τον μεγάλο όγκο των δεδομένων που

εξετάσαμε σε συνδυασμό με την εξαντλητική αναζήτηση που εφαρμόσαμε για τον εντοπισμό των καλύτερων παραμέτρων των ταξινομητών, είναι εύκολα κατανοητό το μέγεθος των απαιτήσεων σε υπολογιστική δύναμη. Η συγκεκριμένη τεχνική δυσκολία ξεπεράστηκε με την διάθεση των κατάλληλων συστημάτων μεγάλης υπολογιστικής δύναμης, για την εκτέλεση των πειραμάτων μας, από το Ινστιτούτο Εφαρμοσμένων Βιοεπιστημών (INAB).

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήσαμε και εφαρμόσαμε στα 12604 γονίδια σε σχέση με τις 2 μεταβλητές στόχους Grades και Stages τις μεθόδους μείωσης διαστάσεων: Principal Component Analysis (PCA), Fisher Score, Kolmogorov-Smirnov 2 Samples, Information Gain, ReliefF και mRMR MIQ.
2. Για κάθε μεταβλητή στόχος (Grdes, Stages) αξιολογήσαμε και ταξινομήσαμε τα γονίδια (χαρακτηριστικά), βάση σημαντικότητας σύμφωνα με τις παραπάνω μεθόδους, για την επιλογή υποψήφιων καθοριστικών υποσυνόλων γονιδίων. Επίσης για κάθε ζεύγος υποσυνόλων που προήλθαν από διαφορετική μέθοδο, εντοπίσαμε τα σημαντικά κοινά (τομή υποσυνόλων) και τα σημαντικά μοναδικά (ένωση υποσυνόλων) γονίδια.
3. Για κάθε υποψήφιο υποσύνολο γονιδίων εφαρμόσαμε εξαντλητική αναζήτηση (Grid Search) με χρήση των εκτιμητών SVM Linear και SVM RBF για τον εντοπισμό των παραμέτρων που έχουν ως αποτέλεσμα την μέγιστη ικανότητα γενίκευσης των εκτιμητών (τιμή ακρίβειας).
4. Συγκρίναμε όλους τους ταξινομητές με βάση την τιμή της ακρίβειας (ικανότητα γενίκευσης), τις τιμές των παραμέτρων τους, τον αριθμό των σημαντικών γονιδίων που χρησιμοποιήθηκαν και περιέχουν ικανή πληροφορία για τον διαχωρισμό της μεταβλητής στόχου και την μέθοδο από την οποία προήλθαν.
5. Εντοπίσαμε για κάθε μεταβλητή στόχος το καθοριστικό υποσύνολο γονιδίων (το οποίο περιέχει την ικανή πληροφορία για τον διαχωρισμό της κάθε μεταβλητής στόχος), με χρήση του οποίου κατασκευάστηκε ο αντίστοιχος ταξινομητής που διαθέτει την καλύτερη ικανότητα γενίκευσης (καλύτερη τιμή ακρίβειας).

1.3 Οργάνωση Κειμένου

Αρχικά, στο Κεφάλαιο 2 παρουσιάζονται εργασίες σχετικές με το αντικείμενο της παρούσας διπλωματικής εργασίας. Στην συνέχεια στο Κεφάλαιο 3, αφού πρώτα γίνεται μία αναφορά σε βασικές έννοιες της Μηχανικής Μάθησης, στην συνέχεια πραγματοποιείται λεπτομερή

περιγραφή των Μεθόδων Μείωσης Διαστάσεων και των Μηχανών Διανυσμάτων Υποστήριξης (SVM) που χρησιμοποιήθηκαν για την εκτέλεση των πειραμάτων. Στο Κεφάλαιο 4 περιγράφεται το σύνολο δεδομένων που χρησιμοποιήσαμε και γίνεται λεπτομερή περιγραφή της εφαρμογής των Μεθόδων Μείωσης Διαστάσεων και της εφαρμογής των Μηχανών Διανυσμάτων Υποστήριξης με γραμμικό πυρήνα και με πυρήνα RBF. Τα αποτελέσματα όλων των πειραμάτων που εκτελέσαμε καθώς και η μεθοδολογία που ακολουθήσαμε παρουσιάζονται και σχολιάζονται στο Κεφάλαιο 5, το οποίο ολοκληρώνεται με την τελική σύνοψη των συμπερασμάτων μας. Στο Κεφάλαιο 6 αναφέρονται οι πλατφόρμες, τα προγραμματιστικά εργαλεία και το λογισμικό που χρησιμοποιήσαμε για την εκτέλεση των πειραμάτων. Τέλος, στο Κεφάλαιο 7 καταγράφουμε τα ευρήματά μας και προτείνουμε ιδέες για μελλοντικές επεκτάσεις.

2

Σχετικές Εργασίες

Κατά την διάρκεια της έρευνας που διεξήχθη για την εύρεση εργασιών σχετικών με την συσχέτιση γονιδίων με διάφορα είδη καρκίνου με την χρήση της μηχανικής μάθησης, διαπιστώθηκε ότι έχει διενεργηθεί πληθώρα ερευνητικών πειραμάτων που αφορά το συγκεκριμένο πεδίο. Η κοινή δυσκολία η οποία εμφανίζεται σε όλες τις εργασίες που μελετήθηκαν είναι ο σχετικά μικρός αριθμός δειγμάτων ασθενών (μικρός αριθμός προτύπων) και ο πολύ μεγάλος αριθμός γονιδίων (χαρακτηριστικών). Ο πολύ μεγάλος αριθμός γονιδίων έχει ως αποτέλεσμα την εμφάνιση του φαινομένου γνωστού ως 'κατάρα των πολλών διαστάσεων', το οποίο δημιουργεί προβλήματα αποδοτικότητας στις περισσότερες μεθόδους της μηχανικής μάθησης και αναλύεται σε παρακάτω υποκεφάλαιο που αφορά την μείωση των διαστάσεων.

Ο κοινός στόχος αυτών των ερευνητικών πειραμάτων είναι η ανίχνευση 'σημαντικών' γονιδίων με την χρήση αλγορίθμων και μεθόδων της μηχανικής μάθησης ώστε να καταλήξουν στην εκπαίδευση ενός μοντέλου, με την χρήση μόνο των 'σημαντικών' γονιδίων (δηλαδή μειώνοντας τον αριθμό των χαρακτηριστικών), το οποίο θα είναι ικανό να προβλέψει την αντίστοιχη περίπτωση κλάσης του εκάστοτε τύπου καρκίνου. Με τον χαρακτηρισμό 'σημαντικά γονίδια' νοούνται εκείνα τα γονίδια τα οποία περιέχουν την περισσότερη πληροφορία η οποία λειτουργεί καθοριστικά για την κατάταξη των δειγμάτων (προτύπων) στις αντίστοιχες κλάσεις.

Για την εύρεση των υποψήφιων 'σημαντικών' γονιδίων γίνεται χρήση μεθόδων μείωσης διαστάσεων, οι οποίες βασίζονται στην στατιστική και κατηγοριοποιούνται βάση των τύπων

των συσχετίσεων που υπολογίζουν και λαμβάνουν υπόψη για την ‘βαθμολόγηση’ των εξεταζόμενων γονιδίων. Βάση αυτής της βαθμολογίας που λαμβάνουν τα γονίδια, επιλέγονται αυτά με την θεωρούμενη ως καλύτερη βαθμολογία ανάλογα με την μέθοδο που χρησιμοποιήθηκε. Έπειτα, χρησιμοποιώντας ως εισόδους τα ‘σημαντικά’ γονίδια, εκπαιδεύουν διάφορα μοντέλα μηχανικής μάθησης.

Συνεπώς, αναζητώντας τον καλύτερο συνδυασμό μεθόδων μείωσης διαστάσεων και μοντέλων μάθησης, προσπαθούν να καταλήξουν σε ένα μοντέλο το οποίο βασισμένο στα ‘σημαντικά’ γονίδια θα εκπαιδεύεται με ένα συγκεκριμένο μοντέλο μάθησης, το οποίο στην συνέχεια θα μπορεί να προβλέπει σωστά την αντίστοιχη περίπτωση κλάσης για δείγματα που δεν χρησιμοποιήθηκαν κατά την εκπαίδευση. Παρακάτω γίνεται μία παρουσίαση τριών ενδεικτικών πειραματικών ερευνών που πραγματοποιήθηκαν εντός της τελευταίας τριετίας.

2.1 Μια προσέγγιση με Βαθιά Μάθηση για την Ανίχνευση Καρκίνου και των Σχετικών Γονιδίων

Οι Danaee, P., Ghaeini R., κ.α. [1] παρουσιάζουν μια προσέγγιση βαθιάς μάθησης για την ανίχνευση του καρκίνου και για την ταυτοποίηση των γονιδίων που είναι κρίσιμα για τη διάγνωση του καρκίνου του μαστού. Το σύνολο δεδομένων που χρησιμοποίησαν στα πειράματα τους ήταν δεδομένα έκφρασης RNA-seq (τιμές έκφρασης μετασχηματισμένες από αλληλούχιση RNA) από την Βάση TCGA (The Cancer Genome Atlas) που αφορούν δείγματα εμφάνισης καρκίνου του μαστού. Συγκεκριμένα αποτελούνται από 1097 δείγματα καρκίνου του μαστού και 113 υγιή δείγματα. Για να ξεπεραστεί η ταξική ανισορροπία των δεδομένων, χρησιμοποιήσαν την τεχνική SMOTE (Synthetic Minority Over-sampling Technique) για να μετατρέψουν τα δεδομένα σε ένα πιο ισορροπημένο σύνολο. Επιπλέον, αφαιρέσαν όλα τα γονίδια που είχαν μηδενική έκφραση σε όλα τα δείγματα.

Αρχικά, χρησιμοποιώντας πολλούς Denoising Autoencoders, στοιβαγμένους ο ένας μετά τον άλλο σχηματίζοντας μία αρχιτεκτονική με το όνομα Stacked Denoising Autoencoders (SDAE) [2], εξήχθησαν βαθιά λειτουργικά χαρακτηριστικά από δεδομένα (προφίλ έκφρασης) μεγάλων διαστάσεων με θόρυβο. Επιπλέον, πραγματοποίησαν εξαγωγή χαρακτηριστικών με την χρήση PCA και Kernel PCA με πυρήνα RBF. Επίσης εντοπίσανε ένα σύνολο πολύ διαδραστικών γονιδίων αναλύοντας τους πίνακες συνδεσιμότητας SDAE τα οποία αποκαλούν DCGs (Deeply Connected Genes). Η κάθε μία από τις παραπάνω μεθόδους πραγματοποίησε μείωση των αρχικού αριθμού των χαρακτηριστικών σε 500.

Όταν μιλάμε για Αυτοκωδικοποιητές (Autoencoders), αναφερόμαστε σε τεχνητά νευρωνικά δίκτυα, τα οποία εκπαιδεύονται έτσι ώστε να μπορούν να ανακατασκευάσουν την είσοδο τους μέσω μιας ενδιάμεσης αναπαράστασης, τυπικά μικρότερης διάστασης από την αρχική. Ο

Denoising Autoencoder αποτελεί μια παραλλαγή του απλού αυτοκωδικοποιητή (autoencoder) που έχει ως βασική ιδέα την προσθήκη θορύβου στα αρχικά δεδομένα και την προσπάθεια ανακατασκευής τους χωρίς το θόρυβο. Διαισθητικά, η προσέγγιση αυτή βγάζει νόημα, μιας και οι ίδιοι σαν άνθρωποι μπορούμε να αναγνωρίσουμε για παράδειγμα τι απεικονίζει μια εικόνα ακόμη και αν έχει διαφθαρεί από αρκετό θόρυβο. Από την άποψη βελτιστοποίησης, η προσθήκη θορύβου σε κάποια χαρακτηριστικά, αναγκάζει το νευρωνικό δίκτυο να ανακαλύψει πιο εύρωστα χαρακτηριστικά.[2]

Στη συνέχεια, αξιολογήσανε τα τέσσερα σύνολα χαρακτηριστικών που προέκυψαν μέσω εποπτευόμενων μοντέλων ταξινόμησης για να επαληθεύσουνε τη χρησιμότητα των νέων χαρακτηριστικών στην ανίχνευση καρκίνου. Συγκεκριμένα χρησιμοποίησαν τεχνητά νευρωνικά δίκτυα ενός στρώματος (single layer ANN), μηχανές διανυσμάτων υποστήριξης με γραμμικό πυρήνα (Linear SVM) και με πυρήνα RBF (SVM-RBF).

Μέθοδος Μείωσης Διαστάσεων	**Μοντέλο Μηχανικής Μάθησης**	**Ακρίβεια % (Accuracy)**
SDAE	ANN	96.95
	SVM	98.04
	SVM-RBF	**98.26**
PCA	ANN	98.38
	SVM	94.58
	SVM-RBF	83.31
KPCA	ANN	96.02
	SVM	96.38
	SVM-RBF	89.92
DCGs	ANN	91.74
	SVM	91.74
	SVM-RBF	**94.78**

Πίνακας 1. *Αποτελέσματα Πειραμάτων*

Τα αποτελέσματα (πίνακας 1) που παρουσιάζονται μας δείχνουν ότι η βαθιά αρχιτεκτονική, SDAE, για την εξαγωγή σημαντικών χαρακτηριστικών από δεδομένα γονιδιακής έκφρασης, αποτελεί μία πολύ καλή μέθοδο επιλογής χαρακτηριστικών για μία επιτυχή και αποδοτική ταξινόμηση (SDAE, SVM-RBF, ακρίβεια: 98.26%). Επίσης με την χρήση των DCGs ως χαρακτηριστικά για τα μοντέλα μηχανικής μάθησης πέτυχαν ακρίβεια 94,78% (SVM-RBF). Το πλεονέκτημα αυτών των χαρακτηριστικών είναι η πιο εύκολη ερμηνεία τους. Ένας περιορισμός των προσεγγίσεων βαθιάς μάθησης είναι η απαίτηση για μεγάλα σύνολα δεδομένων, τα οποία μπορεί να μην είναι διαθέσιμα για καρκινικούς ιστούς. Βέβαια, καθώς

θα διατίθενται περισσότερα δεδομένα γονιδιακής έκφρασης, αυτό το μοντέλο μπορεί να βελτιώσει την απόδοση του.

2.2 Μέθοδοι Μηχανικής Μάθησης για την Ταξινόμηση του Καρκίνου με χρήση Δεδομένων Γονιδιακής Έκφρασης

Οι Bhola, A. και Tiwari, A. K. [3] χρησιμοποίησαν για ανάλυση συνολικά πέντε σύνολα δεδομένων γονιδιακής έκφρασης τα οποία προέρχονται από το Εργαστήριο Τεχνητής Νοημοσύνης της Λιουμπλιάνα (Artificial Intelligence Lab, Ljubljana). Τα τέσσερα από αυτά διαχωρίζονται σε δύο κλάσεις στόχους και το ένα σε τρεις κλάσεις στόχους. Τα τέσσερα σύνολα δεδομένων που κατηγοριοποιούνται σε δύο κλάσεις είναι τα παρακάτω.

- 72 δείγματα που αφορούν την λευχαιμία. Από αυτά τα 47 αφορούν την οξεία λεμφοβλαστική λευχαιμία και τα 25 την οξεία μυελογενή λευχαιμία. Το κάθε δείγμα περιγράφεται από 5147 γονίδια.
- 102 δείγματα που αφορούν τον καρκίνο του προστάτη. Από αυτά τα 50 δείγματα αφορούν φυσιολογικούς ιστούς και 52 δείγματα που αφορούν τον όγκο του προστάτη. Το κάθε δείγμα περιγράφεται από 12533 γονίδια.
- 24 δείγματα που αφορούν τον καρκίνο του μαστού. Από αυτά τα 14 δείγματα είναι ανθεκτικά στην θεραπεία με docetaxel και 10 δείγματα είναι ευαίσθητα στην θεραπεία με docetaxel. Το κάθε δείγμα περιγράφεται από 12625 γονίδια.
- 77 δείγματα που αφορούν τον καρκίνο του λεμφώματος. 58 δείγματα αφορούν διάχυτα μεγάλα λεμφώματα Β-κυττάρων και τα υπόλοιπα 19 δείγματα αφορούν το λέμφωμα των θυλακίων. Το κάθε δείγμα περιγράφεται από 7070 γονίδια.

Για την επιλογή των κρίσιμων γονιδίων χρησιμοποιήθηκαν πέντε μέθοδοι επιλογής χαρακτηριστικών. Συγκεκριμένα χρησιμοποιήθηκαν το Κέρδος Πληροφορίας (Information Gain, IG), ο ReliefF, Μηχανές διανυσμάτων υποστήριξης με επαναληπτική εξάλειψη χαρακτηριστικών (Support vector machine Recursive feature elimination, SVMRFE), η βελτιστοποίηση σμήνους σωματιδίων (Particle swarm optimization, PSO) και η επιλογή χαρακτηριστικών βάσει γρήγορης συσχέτισης (Fast correlation based feature selection, FCBF). Οι ταξινομητές που εφαρμόστηκαν είναι οι εξής: Naive Bayesian, SVM, K-Nearest Neighbour (k-NN), Adaboost, Random Forest (RF) και Bagging. Η μετρική που χρησιμοποιήθηκε για την σύγκριση των αποτελεσμάτων είναι η ακρίβεια (accuracy).

Σύνολο Δεδομένων Λευχαιμίας - Ακρίβεια (%)						**Σύνολο Δεδομένων Καρκίνου του Προστάτη - Ακρίβεια (%)**					
	IG	ReliefF	SVMRFE	PSO	FCFB		IG	ReliefF	SVMRFE	PSO	FCFB
NB	97.2	98.6	100.0	97.2	100.0	NB	92.2	93.1	92.2	62.7	93.1
k-NN	97.2	95.8	90.3	91.7	98.6	k-NN	91.2	92.2	97.1	87.3	94.1
RF	97.2	98.6	95.8	97.2	98.6	RF	86.3	92.2	95.1	90.2	95.1
SVM	97.2	98.6	98.6	98.6	94.4	SVM	93.1	94.1	95.1	93.1	96.1
Bagging	91.7	93.1	93.1	91.7	93.1	Bagging	85.3	86.3	88.2	84.3	87.3
AdaBoost	97.2	97.2	97.2	97.2	98.6	AdaBoost	89.2	92.2	92.2	93.1	98.0
Σύνολο Δεδομένων Καρκίνου του Μαστού - Ακρίβεια (%)						**Σύνολο Δεδομένων Καρκίνου του Λεμφώματος - Ακρίβεια (%)**					
	IG	ReliefF	SVMRFE	PSO	FCFB		IG	ReliefF	SVMRFE	PSO	FCFB
NB	94.1	92.1	97.0	96.1	95.1	NB	88.3	89.6	96.1	80.5	96.1
k-NN	96.1	93.1	97.5	89.7	94.6	k-NN	97.4	96.1	100.0	85.7	94.8
RF	91.6	90.1	93.6	88.2	92.6	RF	90.9	93.5	96.1	87.0	96.1
SVM	68.5	68.5	94.6	92.6	94.1	SVM	75.3	75.3	100.0	93.5	98.7
Bagging	94.1	92.6	92.1	88.7	93.6	Bagging	87.0	90.9	90.9	87.0	87.0
AdaBoost	78.3	78.3	77.3	72.4	75.4	AdaBoost	92.2	88.3	88.3	93.5	97.4

Πίνακας 2. *Αποτελέσματα Πειραμάτων για κάθε σύνολο δεδομένων*

Με βάση τα αποτελέσματα που παρουσιάζονται στον παραπάνω πίνακα, φαίνεται σε κάθε σύνολο δεδομένων να αποδίδει καλύτερα ένας διαφορετικός συνδυασμός μεθόδου επιλογής χαρακτηριστικών και ταξινομητή. Ο ταξινομητής K-Nearest Neighbour συμμετέχει σε δύο βέλτιστους συνδυασμούς. Και στους δύο συνδυασμούς η μέθοδος επιλογής χαρακτηριστικών που χρησιμοποιείται είναι η επαναληπτική εξάλειψη χαρακτηριστικών (SVMRFE) και τα σύνολα δεδομένων στα οποία επιτυγχάνεται η βέλτιστη απόδοση είναι το σύνολο δεδομένων που αφορά τον καρκίνο του λεμφώματος και το σύνολο δεδομένων που αφορά την λευχαιμία. Επίσης, η επαναληπτική εξάλειψη χαρακτηριστικών (SVMRFE) δίνει πολύ καλά αποτελέσματα σε συνδυασμό με τον ταξινομητή SVM στο σύνολο δεδομένων που αφορά τον καρκίνο του λεμφώματος.

2.3 Ένας Αλγόριθμος Επιλογής Καρκινικών Γονιδίων Βασισμένος στο KS-Test & στο CFS

Οι Su, Q., Wang, Y., κ.α. [4] χρησιμοποίησαν στα ερευνητικά τους πειράματα συνολικά πέντε σύνολα δεδομένων γονιδιακής έκφρασης. Το κάθε ένα από τα σύνολα δεδομένων διαχωρίζεται σε δύο κλάσεις στόχους, όπου η μία κλάση αντιπροσωπεύει τα υγιή δείγματα και η άλλη κλάση δείγματα με καρκίνο του εκάστοτε συνόλου δεδομένων. Συγκεκριμένα, τα πέντε σύνολα δεδομένων γονιδιακής έκφρασης που χρησιμοποιήθηκαν είναι τα παρακάτω.

- 97 δείγματα που αφορούν τον καρκίνο του μαστού. Το κάθε δείγμα περιγράφεται από 24481 γονίδια.
- 181 δείγματα που αφορούν τον καρκίνο του πνεύμονα. Το κάθε δείγμα περιγράφεται από 12533 γονίδια.
- 62 δείγματα που αφορούν τον καρκίνο του παχέος εντέρου. Το κάθε δείγμα περιγράφεται από 2000 γονίδια.
- 253 δείγματα που αφορούν τον καρκίνο των ωοθηκών. Το κάθε δείγμα περιγράφεται από 15154 γονίδια.
- 72 δείγματα που αφορούν την λευχαιμία. Το κάθε δείγμα περιγράφεται από 7129 γονίδια.

Στην συγκεκριμένη πειραματική έρευνα παρουσιάζονται συνολικά τρεις συγκρίσεις μεταξύ μεθόδων επιλογής χαρακτηριστικών. Η τρίτη σύγκριση εμπεριέχει τις μεθόδους επιλογής χαρακτηριστικών που συγκρίνονται στην δεύτερη (ακολουθείται η ίδια μεθοδολογία), οπότε παρακάτω θα αναφερθούμε στην πρώτη και στην τρίτη σύγκριση της έρευνας. Ο ταξινομητής που χρησιμοποιείται σε όλα τα πειράματα είναι οι Μηχανές διανυσμάτων υποστήριξης (SVM) με γραμμικό πυρήνα (Linear SVM) δίνοντας στην παράμετρο C την τιμή 1 (όπου C το βάρος του κόστους των λάθος ταξινομήσεων. Για C = 1, αγνοούνται όχι τελείως, αλλά αρκετά οι παράμετροι χαλαρότητας και οι λάθος ταξινομήσεις δεν μας ενδιαφέρουν).

Αρχικά, πραγματοποιείται σύγκριση μεταξύ τριών μεθόδων επιλογής χαρακτηριστικών. Συγκεκριμένα, μεταξύ των KS-Test (Kolmogorov-Smirnov Test), T-Test και του Wilcoxon-Test. Το κριτήριο KS-Test είναι ένα μη παραμετρικό τεστ και χρησιμοποιείται για τον έλεγχο καλής προσαρμογής ενός τυχαίου δείγματος σε μία δεδομένη συνεχή κατανομή.

Το T-Test προσφέρεται για τον έλεγχο της συσχέτισης μιας συνεχούς μεταβλητής με μια δίτιμη μεταβλητή κλάσης, στην οποία η κάθε κλάση αφορά μετρήσεις που γίνονται σε διαφορετικά δείγματα. Ουσιαστικά η υπόθεση που εξετάζεται είναι η ισότητα των μέσων των

δύο ανεξάρτητων ομάδων. Εάν η υπόθεση της ισότητας δεν απορριφθεί τότε θα συνεπάγεται ότι οι δύο μέσοι είναι ίδιοι στις δύο ομάδες, συνεπώς η κατηγοριοποίηση δεν οδηγεί σε διαφορετικά αποτελέσματα, ή με άλλα λόγια η μεταβλητή κλάσης δεν σχετίζεται με τη συνεχή μεταβλητή. Το Wilcoxon-Test είναι ένας έλεγχος μη παραμετρικός ισοδύναμος του T-test.

Μετά την εφαρμογή των παραπάνω κριτηρίων σε όλα τα σύνολα δεδομένων, παρουσιάζουν σε έναν πίνακα τον αριθμό των γονιδίων που επιλέχθηκαν από κάθε κριτήριο. Συγκεκριμένα, για κάθε κριτήριο παρουσιάζουν τα υποσύνολα που επιλέχθηκαν θέτοντας ως επίπεδο σημαντικότητας των αποτελεσμάτων τις τιμές 0.05, 0.01, 0.005 και 0.001. Στην συνέχεια χρησιμοποίησαν ως ταξινομητή τις μηχανές διανυσμάτων υποστήριξης (SVM) όπου για την εκπαίδευση και την επικύρωση των αποτελεσμάτων χρησιμοποίησαν την μέθοδο διασταύρωσης δέκα τμημάτων (10 fold cross-validation).

Τα αποτελέσματα έδειξαν ότι το KS-Test είναι ένας πολύ αποτελεσματικός αλγόριθμος μέτρησης των κρίσιμων γονιδίων. Συγκεκριμένα, με την χρήση του KS-Test και με επίπεδο σημαντικότητας $\alpha = 0.001$, επιλέχθηκαν τα μικρότερα υποσύνολα γονιδίων στα τέσσερα από τα πέντε σύνολα δεδομένων (καρκίνος του πνεύμονα / 1300 γονίδια, καρκίνος του παχέος εντέρου / 44 γονίδια, καρκίνος των ωοθηκών / 268 γονίδια, λευχαιμία / 524 γονίδια). Τα τρία από αυτά τα σύνολα δεδομένων έδωσαν την καλύτερη μέση ακρίβεια ταξινόμησης (καρκίνος του παχέος εντέρου / 83,2%, καρκίνος των ωοθηκών / 96,5%, λευχαιμία / 85,6%). Η μέση ακρίβεια ταξινόμησης στο σύνολο δεδομένων που αφορά τον καρκίνο του μαστού (83,2%), είναι λίγο χειρότερη από την καλύτερη που παρατηρήθηκε ενώ στο σύνολο δεδομένων που αφορά τον καρκίνο του πνεύμονα η μέση ακρίβεια ταξινόμησης είναι 91,6%.

Η τρίτη σύγκριση πραγματοποιείται μεταξύ τεσσάρων μεθόδων επιλογής χαρακτηριστικών και ενός συνδυασμού δύο μεθόδων επιλογής χαρακτηριστικών. Συγκεκριμένα, μεταξύ των Correlation based Feature Selection (CFS), του κριτήριου του ελάχιστου πλεονασμού - μέγιστης συσχέτισης (Minimum Redundancy Maximum Relevance, mRMR), της ReliefF, του KS-Test και του συνδυασμού των KS-Test με την CFS.

Η μέθοδος CFS αξιολογεί υποσύνολα χαρακτηριστικών σύμφωνα με την ακόλουθη υπόθεση: τα υποσύνολα των κρίσιμων χαρακτηριστικών περιέχουν χαρακτηριστικά που συσχετίζονται σε μεγάλο βαθμό με την κλάση ταξινόμησης και παράλληλα δεν συσχετίζονται μεταξύ τους. Τα πλεονάζοντα χαρακτηριστικά αφαιρούνται. Η αποδοχή ενός χαρακτηριστικού εξαρτάται από το βαθμό στον οποίο προβλέπει την τιμή της κλάσης σε περιοχές του χώρου παρουσίασης που δεν έχουν προβλεφθεί από άλλα χαρακτηριστικά.

Η μέθοδος του ελάχιστου πλεονασμού - μέγιστης συσχέτισης (mRMR) βασίζεται πάνω στην αμοιβαία πληροφορία και το υποψήφιο υποσύνολο θα πρέπει να πληροί ταυτόχρονα δύο συνθήκες. Συγκεκριμένα, θα πρέπει τα χαρακτηριστικά του υποσυνόλου να έχουν όσο το

δυνατό μεγαλύτερη συνάφεια με την κλάση (μέγιστη συσχέτιση - maximum relevance) και παράλληλα να είναι όσο το δυνατόν ανόμοια μεταξύ τους (ελάχιστος πλεονασμός - minimum redudancy).

Η μέθοδος ReliefF αποτελεί προέκταση της Relief. Μπορεί χρησιμοποιηθεί και σε προβλήματα με περισσότερες από δύο κλάσεις. Η βασική ιδέα της μεθόδου ReliefF έγκειται στην επιλογή χαρακτηριστικών βάσει της διαχωριστικής ικανότητας των τιμών τους πάνω σε κοντινά πρότυπα.

Για την επιλογή υποσυνόλου χαρακτηριστικών με τον mRMR και την ReliefF, αρχικά επιλέχθηκαν τα πρώτα καλύτερα 50 γονίδια. Στην συνέχεια με την χρήση ενός προς τα εμπρός (ή αλλιώς αυξητικής αναζήτησης) αλγόριθμου επιλογής χαρακτηριστικών, επιλέχθηκαν τα κατάλληλα υποσύνολα χαρακτηριστικών από το κάθε υποσύνολο των 50 γονιδίων. Το υποσύνολα των γονιδίων που επιλέχθηκαν για τους KS-Test και για τον CFS ήταν το υποσύνολο που προέκυψε με επίπεδο σημαντικότητας $\alpha = 0.01$.

Το υποσύνολο χαρακτηριστικών από τον συνδυασμό των KS-Test και του CFS προέκυψε από την εξής διαδικασία. Αρχικά, επιλέχθηκε το υποσύνολο των χαρακτηριστικών σύμφωνα με τον KS-Test με επίπεδο σημαντικότητας $\alpha = 0.01$. Στην συνέχεια σε αυτό το υποσύνολο χαρακτηριστικών εφαρμόστηκε η μέθοδος CFS από όπου και προέκυψε το τελικό υποσύνολο χαρακτηριστικών που χρησιμοποιήθηκε για την συνέχεια του πειράματος.

Όπως προαναφέρθηκε στο πείραμα χρησιμοποίησαν ως ταξινομητή τις μηχανές διανυσμάτων υποστήριξης (SVM) όπου για να αξιολογήσουν την απόδοση των ταξινομητών στα επιλεγμένα υποσύνολα γονιδίων χρησιμοποίησαν την μέθοδο διασταύρωσης δέκα τμημάτων (10 fold cross-validation). Για τη λήψη στατιστικά σημαντικών πειραματικών αποτελεσμάτων, τα δείγματα του συνόλου δεδομένων ανακατεύονταν τυχαία, η διαδικασία επαναλήφθηκε 10 φορές και καταγράφηκαν και συγκρίθηκαν ο μέσος όρος των 10 επαναλήψεων. Τα αποτελέσματα των πειραμάτων παρουσιάζονται στον πίνακα 3.

Το συμπέρασμα στο οποίο κατέληξαν σύμφωνα με τον παρακάτω πίνακα αποτελεσμάτων, είναι ότι ο συνδυασμός των μεθόδων επιλογής χαρακτηριστικών KS-Test και CFS (KS-Test CFS) πέτυχε την καλύτερη απόδοση από τους άλλους αλγορίθμους επιλογής γονιδίων σχεδόν σε όλα τα σύνολα δεδομένων. Βέβαια διαπιστώθηκε ότι ο χρόνος εκτέλεσης του KS-Test CFS δεν ήταν καλύτερος έναντι των υπόλοιπων αλγορίθμων και για αυτό τον λόγο θέτουν ως επόμενο βήμα, την βελτιστοποίηση του χρόνου εκτέλεσης του KS-Test CFS.

Σύνολα Δεδομένων	Μέθοδος Μείωσης Διαστάσεων	Αριθμός Γονιδίων (*μέσος όρος επαναλήψεων*)	Ακρίβεια % (*μέσος όρος επαναλήψεων*)
Καρκίνος του μαστού	**k-S test - CFS**	**11.7**	**87.4**
	CFS	19.6	80.5
	k-S test	22.5	78.8
	mRMR	21.8	82.4
	ReliefF	15.9	59.4
Καρκίνος του πνεύμονα	**k-S test - CFS**	**23**	**91.6**
	CFS	27.3	88.9
	k-S test	33.4	80.6
	mRMR	289	89.8
	ReliefF	33.6	84.7
Καρκίνος του παχέος εντέρου	**k-S test - CFS**	**10.7**	**90.1**
	CFS	6.8	89.7
	k-S test	19.4	84.5
	mRMR	5.9	89.7
	ReliefF	15	74.9
Καρκίνος των ωοθηκών	**k-S test - CFS**	**33.2**	**98.5**
	CFS	31.6	95.3
	k-S test	46	78.9
	mRMR	32.7	95.2
	ReliefF	39.6	90.6
Λευχαιμία	**k-S test - CFS**	**25.2**	**79.6**
	CFS	33.3	78.9
	k-S test	38.7	72.7
	mRMR	28.6	75.7
	ReliefF	36.4	77.6

Πίνακας 3. *Αποτελέσματα Πειραμάτων για κάθε σύνολο δεδομένων*

3

Θεωρητικό Υπόβαθρο

Στόχος της εργασίας είναι η ανάλυση της συσχέτισης γονιδίων με τον καρκίνο του ήπατος ώστε να κατασκευάσουμε έναν ταξινομητή ο οποίος θα διαθέτει την ικανότητα της γενίκευσης. Αρχικά χρησιμοποιήσαμε κριτήρια και μεθόδους μηχανικής μάθησης για τον εντοπισμό και την αξιολόγηση των συσχετίσεων των γονιδίων με τον καρκίνο του ήπατος και την επιλογή των κρίσιμων γονιδίων βάση αυτών των συσχετίσεων (μείωση διαστάσεων). Στην συνέχεια, βάσει των επιλεγμένων υποψήφιων υποσυνόλων, πραγματοποιήσαμε μία σειρά πειραμάτων με την χρήση της μεθόδου των Διανυσμάτων Υποστήριξης (Support Vector Machine, SVM) για τον εντοπισμό του βέλτιστου υποσυνόλου του οποίου τα γονίδια κατέχουν καθοριστικό ρόλο για την κατασκευή ενός αποδοτικού ταξινομητή.

Στο παρόν κεφάλαιο, αρχικά γίνεται μία αναφορά στις βασικές έννοιες της Μηχανικής Μάθησης, στο πρόβλημα της Ταξινόμησης, στα Κριτήρια επίδοσης των ταξινομητών και σε κάποιες δημοφιλή μεθόδους Διακριτοποίησης. Στην συνέχεια αναλύονται η μέθοδος των Διανυσμάτων Υποστήριξης (Support Vector Machine, SVM) και τα κριτήρια που χρησιμοποιήθηκαν για την μείωση των διαστάσεων του συνόλου δεδομένων, τα οποία είναι η Ανάλυση Κυρίων Συνιστωσών (Principal Component Analysis, PCA), το κριτήριο Fisher (Fisher Score), η Αμοιβαία Πληροφορία (Mutual Information), το κριτήριο Kolmogorov-Smirnov για δύο δείγματα (KS-2samples Test), η μέθοδος ReliefF και το κριτήριο του ελάχιστου πλεονασμού - μέγιστης συσχέτισης (Minimum Redundancy Maximum Relevance, mRMR).

3.1 Μηχανική Μάθηση (Machine Learning)

Η μηχανική μάθηση (machine learning) είναι ένας κλάδος της τεχνητής νοημοσύνης που αποτελείται από μεθόδους και αλγόριθμους με τους οποίους βελτιώνεται η αποδοτικότητα μιας μηχανής στην εκτέλεση ευφυών εργασιών. Πιο συγκεκριμένα, σκοπός της μηχανικής μάθησης είναι να καταστήσει μια μηχανή (αλγόριθμο) έμπειρη, βελτιώνοντας την αποτελεσματικότητα της για μία συγκεκριμένη λειτουργία. Ένας πιο τυπικός και αυστηρός όρος της μηχανικής μάθησης είναι ο εξής: Ένα πρόγραμμα υπολογιστή μαθαίνει από την εμπειρία E, την οποία αποκτά κατά την εκτέλεση ενός συνόλου εργασιών T, εφόσον η απόδοση του P βελτιώνεται με την αξιοποίηση της εμπειρίας E. [5]

Οι βασικές κατηγορίες ευφυών εργασιών που εκτελούν οι μέθοδοι και οι αλγόριθμοι της μηχανικής μάθησης είναι η αναγνώριση προτύπων, η εξαγωγή χαρακτηριστικών, η πρόβλεψη και η παλινδρόμηση, η ομαδοποίηση και η γενίκευση. Στην παρούσα εργασία αναλύονται και εφαρμόζονται η αναγνώριση προτύπων, με την χρήση ενός δυαδικού ταξινομητή και διάφοροι μέθοδοι επιλογής χαρακτηριστικών.

Ένας ταξινομητής αποτελεί έναν αλγόριθμο αντιστοίχισης πραγματικών αντικειμένων, εικόνων ή χρονοσειρών σε κλάσεις. Το σύνολο δεδομένων που δέχεται ένας ταξινομητής θα πρέπει να αποτελείται από αριθμητικά δεδομένα. Το πρόβλημα της ταξινόμησης που καλείτε να αντιμετωπίσει ένας ταξινομητής καθώς και η μεθοδολογία που χρησιμοποιείται αναλύονται στο επόμενο υποκεφάλαιο.

Γενικά, υπάρχουν διάφοροι τρόποι κατηγοριοποίησης των μεθόδων μηχανικής μάθησης. Ο πιο διαδομένος διαχωρισμός διακρίνει τις μεθόδους στα παρακάτω τρία είδη μάθησης:

Μάθηση με επίβλεψη (supervised learning): Ο αλγόριθμος για την εκπαίδευση του χρησιμοποιεί τα πρότυπα εισόδου μαζί με την επιθυμητή τιμή εξόδου του κάθε προτύπου (τιμή της κλάσης).

Μάθηση χωρίς επίβλεψη (unsupervised learning): Ο αλγόριθμος για την εκπαίδευση του χρησιμοποιεί μόνο τα πρότυπα εισόδου χωρίς να χρησιμοποιεί στόχους.

Μάθηση με ενίσχυση (reinforcement learning): Ο αλγόριθμος για την εκπαίδευση του χρησιμοποιεί μόνο τα πρότυπα εισόδου χωρίς να γνωρίζει αν πάει καλά ή όχι, παρά μόνο στο τέλος. Τότε μόνο γνωρίζει αν πέτυχε τον στόχο του και χρησιμοποιεί αυτή την πληροφορία για να βελτιωθεί την επόμενη φορά.

3.2 Ταξινόμηση (Classification)

Η ταξινόμηση ή κατηγοριοποίηση (Classification) προσδιορίζεται ως το πρόβλημα ανάθεσης ενός προτύπου (pattern) σε μία ή περισσότερες προκαθορισμένες κλάσεις (classes). Στην

παρούσα εργασία, όπως προαναφέρθηκε, θα ασχοληθούμε με την ταξινόμηση ενός συνόλου προτύπων ανάμεσα σε δύο κλάσεις. Ως κλάση ορίζεται ένα σύνολο ομοειδών αντικειμένων και ως πρότυπο ορίζεται ένα διάνυσμα που περιέχει αριθμητικά χαρακτηριστικά ενός αντικειμένου από μία κλάση.

Η περίπτωση της δυαδικής ταξινόμησης έγκειται στην διαδικασία κατά την οποία δοσμένου ενός συνόλου προτύπων $X_i = \{x_1, x_2, ..., x_v\}$ και ενός συνόλου δύο κλάσεων $C = \{C_0, C_1\}$, πρέπει να καθοριστεί σε ποια από τις δύο κλάσεις ανήκει κάθε ένα από τα πρότυπα X. Η επίτευξη της παραπάνω διαδικασίας βασίζεται στην εύρεση μίας συνάρτησης στόχου ή διαχωρισμού (target / discriminant function) *f*, που απεικονίζει το κάθε σύνολο τιμών ενός αντικειμένου X σε μία από τις δύο προκαθορισμένες κλάσεις, ώστε να είναι δυνατή η ταξινόμηση μελλοντικών προτύπων.

$$y = f(\boldsymbol{x}; \boldsymbol{w}) \quad (1)$$

Η διαδικασία που ακολουθείται για την ταξινόμηση προτύπων στην μηχανική μάθηση αποτελείται από το στάδιο κατασκευής του μοντέλου ταξινόμησης (εκμάθηση), τον έλεγχο του μοντέλου ταξινόμησης (επικύρωση ή ανάκληση) και τέλος την εφαρμογή του μοντέλου. Αρχικά, τα δεδομένα (τα πρότυπα) χωρίζονται σε δύο σύνολα, στο σύνολο εκπαίδευσης (training set) και στο σύνολο ελέγχου (test set).

Στο πρώτο στάδιο, εισάγονται τα δεδομένα εκπαίδευσης και η αντίστοιχη κλάση (target) στην οποία ανήκει το κάθε πρότυπο (supervised learning). Στο στάδιο αυτό αναλύονται τα δεδομένα εκπαίδευσης ώστε να ανακαλυφθούν οι σχέσεις που συνδέουν την κάθε κλάση με τα αντίστοιχα πρότυπα. Από την ανάλυση των δεδομένων εκπαίδευσης προκύπτει το αντίστοιχο μοντέλο για την ταξινόμηση τους. Είναι σημαντικό να σημειωθεί ότι η επιλογή του συνόλου εκπαίδευσης είναι καθοριστική για την κατασκευή του μοντέλου ταξινόμησης, καθώς η αστοχία κατά την επιλογή του μπορεί να έχει ως αποτέλεσμα την κατασκευή ενός μεροληπτικού μοντέλου ταξινόμησης.

Στο δεύτερο στάδιο πραγματοποιείται έλεγχος του μοντέλου ταξινόμησης με την χρήση του συνόλου ελέγχου. Ο έλεγχος του μοντέλου ταξινόμησης είναι απαραίτητος καθώς κρίνει την καταλληλόλητα (απόδοση) του μοντέλου για την σωστή ταξινόμηση προτύπων χωρίς να γνωρίζει την κλάση στην οποία ανήκουν. Κατά το στάδιο του ελέγχου εισάγονται τα δεδομένα του συνόλου ελέγχου και αρχικά το μοντέλο ταξινόμησης προβλέπει την κλάση στην οποία ανήκει το κάθε πρότυπο. Στην συνέχεια συγκρίνοντας για κάθε πρότυπο την προβλεπόμενη κλάση με την πραγματική κλάση στην οποία ανήκει, υπολογίζεται η ακρίβεια της απόδοσης του μοντέλου. Ο στόχος ενός ταξινομητή είναι να διαθέτει την ικανότητα της γενίκευσης. Δηλαδή την ικανότητα να εκτιμά την σωστή κλάση (έξοδο) για πρότυπα (είσοδος) που δεν έχει δει κατά την εκπαίδευση.

Για την επίτευξη του στόχου της γενίκευσης είναι πολύ σημαντικό ο έλεγχος καταλληλόλητας του μοντέλου να πραγματοποιηθεί με δεδομένα που δεν συμμετείχαν στην εκπαίδευση του. Επίσης, είναι πολύ σημαντικό να αναφερθεί ότι για την κατασκευή ενός αποδοτικού ταξινομητή έχει μεγάλη βαρύτητα η αναλογία του αριθμού των προτύπων και του αριθμού των χαρακτηριστικών των προτύπων. Στην περίπτωση που ο αριθμός των προτύπων εκπαίδευσης είναι μικρός σε σχέση με τον αριθμό των χαρακτηριστικών τους, τότε για την κατασκευή ενός αποδοτικού και αποτελεσματικού ταξινομητή θα πρέπει να μειωθούν τα χαρακτηριστικά των προτύπων. Η διαδικασία της μείωσης των χαρακτηριστικών ή διαφορετικά της μείωσης των διαστάσεων ενός συνόλου δεδομένων, αποτελεί μία από τις σημαντικότερες προκλήσεις σε ερευνητικούς τομείς όπου ο αριθμός των χαρακτηριστικών των συνόλων δεδομένων που ερευνώνται είναι πολύ μεγάλος. Οι μέθοδοι και τα κριτήρια που χρησιμοποιούνται για την μείωση των διαστάσεων αναλύονται παρακάτω στο αντίστοιχο υποκεφάλαιο του κεφαλαίου 3.

Σε περίπτωση που δεν ληφθούν υπόψη τα παραπάνω μπορεί να οδηγηθούμε στην κατασκευή ενός μοντέλου το οποίο θα παρουσιάζει το φαινόμενο της υπερπροσαρμογής (over-fitting). Με τον όρο υπερπροσαρμογή ορίζεται το φαινόμενο κατά το οποίο η κατασκευή του μοντέλου ταξινόμησης αρκείται σε ένα είδος απομνημόνευσης των χαρακτηριστικών των προτύπων με αποτέλεσμα την κατασκευή ενός πολύπλοκου μοντέλου το οποίο δεν γενικεύει σωστά μακριά από τις τιμές των χαρακτηριστικών των προτύπων. Το αντίθετο φαινόμενο της υπερπροσαρμογής είναι η υποπροσαρμογή (under-fitting), κατά το οποίο καταλήγουμε σε ένα μοντέλο ταξινόμησης πολύ απλό με αποτέλεσμα να μην μπορεί να γενικεύει σωστά μακριά από τις τιμές των χαρακτηριστικών των προτύπων.

Για την αποφυγή των παραπάνω φαινομένων και για την βελτίωση της ικανότητας του μοντέλου ταξινόμησης να γενικεύει σωστά, μπορεί να χρησιμοποιηθεί για την εκπαίδευσή του και την επικύρωσή του η μέθοδος της διασταύρωσης (cross-validation). Οι πιο κοινές εφαρμογές της μεθόδου διασταύρωσης είναι οι παρακάτω:

K-fold cross-validation: Το αρχικό σύνολο δεδομένων χωρίζεται σε k υποσύνολα. Το ένα χρησιμοποιείται για έλεγχο και τα υπόλοιπα (k-1) υποσύνολα χρησιμοποιούνται για την εκπαίδευση. Αναλυτικά τα βήματα που ακολουθεί η συγκεκριμένη εφαρμογή της μεθόδου διασταύρωσης, η οποία χρησιμοποιήθηκε στην παρούσα εργασία, είναι τα εξής:

1) Τεμαχίζει το σύνολο δεδομένων σε k ισομεγέθη τμήματα.
2) Χρησιμοποιεί τα k-1 τμήματα για την εκπαίδευση του μοντέλου και αφήνει 1 για έλεγχο.
3) Υπολογίζει το σφάλμα και την ακρίβεια ελέγχου για κάθε πείραμα.
4) Επαναλαμβάνει το πείραμα k φορές, αφήνοντας κάθε φορά άλλο τμήμα για έλεγχο και υπολογίζει το σφάλμα και την ακρίβεια ελέγχου.

5) Υπολογίζει τον μέσο όρο των σφάλματων και της ακρίβειας ελέγχου από όλα τα k πειράματα.
6) Επαναλαμβάνει ξανά όλη την παραπάνω διαδικασία για ένα άλλο δίκτυο με περισσότερους ή λιγότερους νευρώνες.
7) Τυπώνει την καμπύλη του μέσου σφάλματος ή της μέσης ακρίβειας ελέγχου σε συνάρτηση με το πλήθος των νευρώνων.
8) Επιλέγει το δίκτυο με το μικρότερο μέσο σφάλμα ελέγχου και την μεγαλύτερη μέση ακρίβεια ελέγχου.

Το πλεονέκτημα αυτής της εφαρμογής της μεθόδου διασταύρωσης είναι ότι όλα τα πρότυπα του συνόλου δεδομένων χρησιμοποιούνται και για εκπαίδευση αλλά και για έλεγχο.

Leave-one-out cross-validation (LOOCV): Χρησιμοποιεί ένα μόνο πρότυπο από το αρχικό σύνολο δεδομένων για έλεγχο και όλα τα υπόλοιπα πρότυπα χρησιμοποιούνται για εκπαίδευση. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να χρησιμοποιηθούν όλα τα πρότυπα από μία τουλάχιστον φορά για έλεγχο. Η μέθοδος leave-one-out συνήθως δίνει τα καλύτερα αποτελέσματα, αλλά έχει μεγάλο κόστος σε υπολογιστική ισχύ λόγω των πολλών επαναλήψεων που απαιτούνται για την ολοκλήρωση της εκπαίδευσης.

3.3 Κριτήρια Επίδοσης Ταξινομητών

Όπως προαναφέρθηκε στο προηγούμενο υποκεφάλαιο, ο στόχος ενός ταξινομητή είναι να διαθέτει την ικανότητα της γενίκευσης. Δηλαδή την ικανότητα να εκτιμά την σωστή κλάση (έξοδο) για πρότυπα (είσοδος) που δεν έχει δει κατά την εκπαίδευση. Για την έγκυρη αξιολόγηση των ταξινομητών απαιτείται η χρήση διάφορων μέτρων και διαδικασιών αξιολόγησης.

Για την κατανόηση των μέτρων αξιολόγησης, θεωρούμε ένα πρόβλημα ταξινόμησης δύο κλάσεων, όπου η μία κλάση (C_1) περιέχει έναν αριθμό προτύπων για τα οποία ισχύει μία κατάσταση X, ενώ η άλλη κλάση (C_0) περιέχει ένα αριθμό προτύπων για την οποία δεν ισχύει η κατάσταση X. Ουσιαστικά η ετικέτα κλάσης αποτελεί μία boolean μεταβλητή t = 0/1 ή διαφορετικά t = false/true, η οποία αντιστοιχεί στην ύπαρξη ή στην μη-ύπαρξη της κατάστασης X.

Ο αντίστοιχος ταξινομητής δημιουργεί μια συνάρτηση διαχωρισμού $yi=f(\mathbf{x}i;\boldsymbol{\theta})$ η οποία παραμετροποιείται από το διάνυσμα $\boldsymbol{\theta}$ και το οποίο μαθαίνει κατά την εκπαίδευσή του. Η συνάρτηση $f()$ είναι ένα είδος τεστ στο οποίο υποβάλλουμε το πρότυπο $\mathbf{x}i$:

- αν το πρότυπο βγει θετικό (y_i=1) τότε λέμε ότι το πρότυπο ταξινομήθηκε στην Κλάση C_1

- αν το πρότυπο βγει αρνητικό (y_i=0) τότε λέμε ότι το πρότυπο ταξινομήθηκε στην Κλάση C_0

Στην ιδανική περίπτωση, για κάθε πρότυπο $\mathbf{x}_i$, το τεστ θα πρέπει να δίνει έξοδο y_i που να ταυτίζεται με την ετικέτα t_i της κλάσης στην οποία ανήκει το πρότυπο. Αυτό όμως δεν συμβαίνει πάντα. Οι περιπτώσεις που μπορεί να προκύψουν είναι συνολικά τέσσερις και είναι οι παρακάτω.

1) True Negatives (πραγματικά αρνητικά): Λέμε τα πρότυπα τα οποία ο ταξινομτής τα πρόβλεψε ότι ανήκουν στην κλάση C_0 (δηλαδή $y_i = 0$) και όντως ανήκουν στην κλάση C_0, δηλαδή $t_i = 0$ (ή διαφορετικά t_i = false).
2) False Negatives (εσφαλμένα αρνητικά): Λέμε τα πρότυπα τα οποία ο ταξινομτής τα πρόβλεψε ότι ανήκουν στην κλάση C_0 (δηλαδή $y_i = 0$) αλλά ανήκουν στην κλάση C_1 όπου $t_i = 1$ (ή διαφορετικά t_i = true).
3) False Positives (εσφαλμένα θετικά): Λέμε τα πρότυπα τα οποία ο ταξινομτής τα πρόβλεψε ότι ανήκουν στην κλάση C_1 (δηλαδή $y_i = 1$) αλλά ανήκουν στην κλάση C_0 όπου $t_i = 0$ (ή διαφορετικά t_i = false).
4) True Positives (πραγματικά θετικά): Λέμε τα πρότυπα τα οποία ο ταξινομτής τα πρόβλεψε ότι ανήκουν στην κλάση C_1 (δηλαδή $y_i = 1$) και όντως ανήκουν στην κλάση C_1 όπου $t_i = 1$ (ή διαφορετικά t_i = true).

Θεωρώντας ότι σύνολο προτύπων και των δύο κλάσεων είναι $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, αξιολογούμε την επίδοση ενός ταξινομητή χρησιμοποιώντας τον παρακάτω πίνακα ο οποίος αποκαλείται πίνακας σύγχυσης (confusion matrix).

	Ταξινομήθηκαν στην Κλάση C_0 ($y = 0$)	Ταξινομήθηκαν στην Κλάση C_1 ($y = 1$)
Ανήκουν στην Κλάση C_0 ($t = 0$)	True Negatives (TN)	False Positives (FP)
Ανήκουν στην Κλάση C_1 ($t = 1$)	False Negatives (FN)	True Positives (TP)

Πίνακας 4. *Πίνακας Σύγχυσης*

Σύμφωνα με τον παραπάνω πίνακα, η ιδανική περίπτωση, όπως προαναφέρθηκε είναι ο αριθμός των False Positives και ο αριθμός των False Negatives να είναι 0.

Η επίδοση ενός ταξινομητή συνήθως μετράτε από την ακρίβεια (accuracy) η οποία δίνεται από τον παρακάτω λόγο.

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{2}$$

Σύμφωνα με τον παραπάνω τύπο, προφανώς 0 ≤ Accuracy ≤ 1 και στην ιδανική περίπτωση όπου FP = FN = 0, τότε η ακρίβεια είναι ίση με 1. Το μειονέκτημα του κριτηρίου Accuracy είναι η γενικότητα του. Συγκεκριμένα, σε περίπτωση που δεν υπάρχει μία σχετική ισορροπία μεταξύ των δύο συνόλων προτύπων των δύο κλάσεων, τότε είναι πολύ πιθανόν να μας δοθεί ως Accuracy μία αρκετά υψηλή τιμή, η οποία όμως στην πραγματικότητα είναι παραπλανητική. Με άλλα λόγια η τιμή του κριτηρίου Accuracy σε μη ισορροπημένα σύνολα προτύπων δεν αρκεί για να εκτιμηθεί έγκυρα η επίδοση ενός ταξινομητή. Για αυτόν τον λόγο έχουν ορισθεί τα δύο παρακάτω κριτήρια.

$$Precision = \frac{TP}{POSITIVE} = \frac{TP}{TP + FN} \tag{3}$$

$$Recall = \frac{TP}{Class\ C_1} = \frac{TP}{TP + FP} \tag{4}$$

Το κριτήριο Precision αποτελεί το ποσοστό των προτύπων που ταξινομήθηκαν ως θετικά και ανήκουν όντως στην Κλάση C_1. Το κριτήριο Recall αποτελεί το ποσοστό των προτύπων που ανήκουν στην κλάση C_1 και ταξινομούνται ως θετικά. Προφανώς, οι τιμές και των δύο κριτηρίων κυμαίνονται μεταξύ 0 και 1 και στην ιδανική περίπτωση που FP = FN = 0, τότε θα έχουμε Precision = Recall = 1, το οποίο όμως δεν είναι πάντα εφικτό. Επιπλέον είναι δυνατόν κάποιος ταξινομητής να έχει καλή επίδοση στο κριτήριο Precision αλλά όχι και στο Recall, και αντίστροφα.

Τα κριτήρια Precision και Recall δεν αρκούν από μόνα τους για την συνολική εκτίμηση της επίδοσης ενός ταξινομητή. Για αυτόν τον λόγο συνδυάζονται συνήθως με το κριτήριο F-measure (ή αλλιώς F1-score) που δίνεται από τον παρακάτω τύπο.

$$Fmeasure = \frac{Precision \cdot Recall}{(Precision + Recall)/2} \tag{5}$$

Σύμφωνα με τον παραπάνω τύπο και τους τύπους των κριτηρίου Precision και Recall, αποδεικνύεται ότι το εύρος τιμών του κριτηρίου F-measure είναι επίσης [0,1] και επιτυγχάνει την μέγιστη τιμή αν και μόνο αν Precision = Recall = 1.

Ένα μειονέκτημα των κριτηρίων Precision και Recall είναι ότι εστιάζουν αποκλειστικά στην κλάση C_1. Επιπλέον, ο αριθμός των TN δεν εμπλέκεται πουθενά στον ορισμό και των δύο κριτηρίων. Αν και αυτό μπορεί να αρκεί στην περίπτωση που είναι πολύ σημαντική η κλάση C_1, πολλές φορές είναι εξίσου σημαντική και η σωστή ταξινόμηση στην κλάση C_0. Για αυτό έχουν οριστεί δύο άλλα κριτήρια επίδοσης τα οποία δίνουν ίση βαρύτητα και στις δύο κλάσεις.

$$Sensitivity = \frac{TP}{Class\ C_1} = \frac{TP}{TP + FN}\ (= Recall = True\ Positive\ Rate) \qquad (6)$$

$$Specificity = \frac{TN}{Class\ C_0} = \frac{TN}{TN + FP}\ (= True\ Negative\ Rate) \qquad (7)$$

Το κριτήριο Sensitivity (γνωστό και ως True-Positive Rate ή TPR) είναι ουσιαστικά το ίδιο κριτήριο με το κριτήριο Recall. Το κριτήριο Specificity (γνωστό και ως True-Negative Rate ή TNR) μας δείχνει σε τι ποσοστό το τεστ ταξινομεί σωστά τα πρότυπα της κλάσης C_0. Τα κριτήρια είναι ουσιαστικά ίδια στον μαθηματικό ορισμό τους με εξαίρεση την κλάση στην οποία εστιάζουν. Οι τιμές και των δύο κριτηρίων επίσης κυμαίνονται μεταξύ 0 και 1 και στην ιδανική περίπτωση παίρνουν την τιμή 1.

Όπως και στην περίπτωση του Precision και του Recall, τα κριτήρια Specificity και Sensitivity από μόνα τους δεν μπορούν να περιγράψουν πλήρως την επίδοση ενός ταξινομητή. Για αυτό το λόγο υπάρχει ένα συνδυαστικό κριτήριο με βάση αυτά. Αυτό το συνδυαστικό κριτήριο βασίζεται στο γράφημα Sensitivity - Specificity που αποκαλείται Receiver Operating Characteristic (ROC) (εικόνα 1). Συγκεκριμένα, η καμπύλη ROC αποτελεί το γράφημα του Sensitivity σε σχέση με το (1 - Specificity). Το (1 - Specificity) καλείται επίσης και False-Positive Rate.

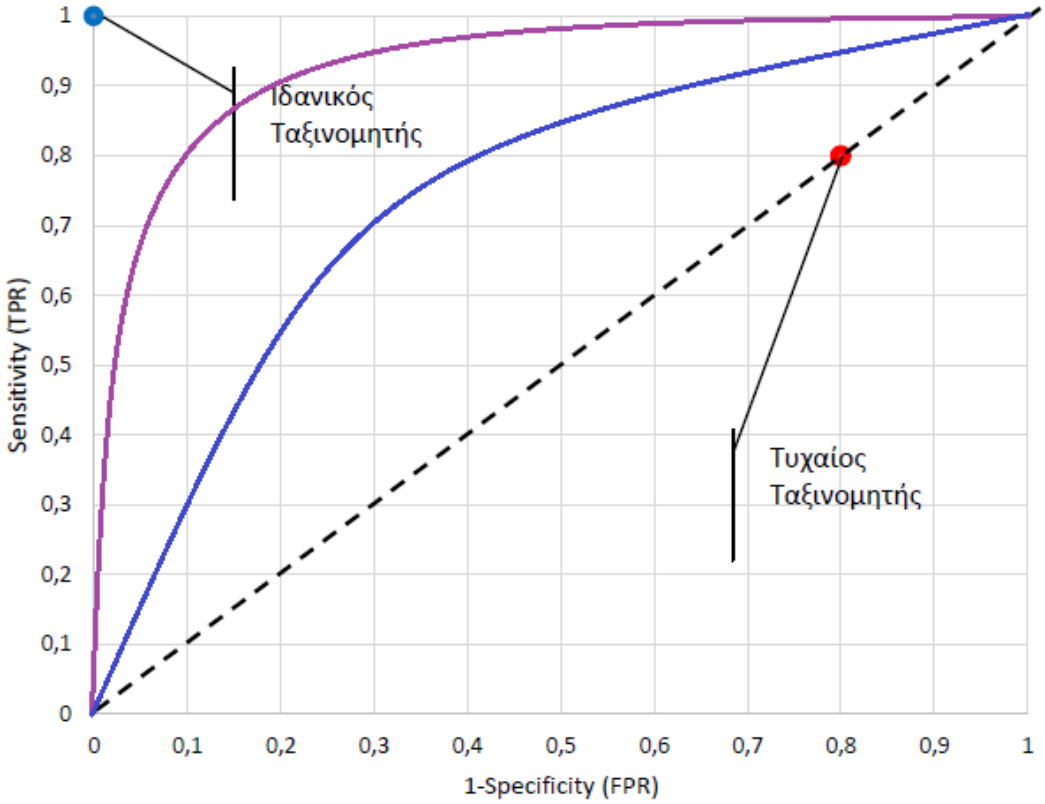
Ένας στοιχειώδης ταξινομητής είναι αυτός που αποφασίζει εντελώς τυχαία με κάποια πιθανότητα p αν το πρότυπο ανήκει στην κλάση 1 (κλάση C_1) και άρα με πιθανότητα $(1 - p)$ στην κλάση 0 (κλάση C_0). Σε αυτή την περίπτωση, το ποσοστό των προτύπων που ταξινομούνται θετικά ή αρνητικά είναι ανεξάρτητο από την κλάση στην οποία ανήκουν. Συνεπώς:

$$Sensitivity = \frac{TP}{Class\ C_1} = P(y = 1 \mid t = 1) = P(y = 1) = p \tag{8}$$

$$Specificity = \frac{TN}{Class\ C_0} = P(y = 0 \mid t = 0) = P(y = 0) = 1 - p \tag{9}$$

Οπότε ισχύει

$$1 - Specificity = p \tag{10}$$



Εικόνα 1. *Receiver Operating Characteristic (ROC).*

Στην εικόνα 1, πάνω στην μαύρη διακεκομμένη διαγώνιο βρίσκεται το σημείο λειτουργίας του τυχαίου ταξινομητή. Στην ίδια εικόνα, ο ιδανικός ταξινομητής βρίσκεται στο σημείο 1 – Specificity = 0 και Sensitivity = 1. Το πάνω αριστερά τρίγωνο είναι η «καλή» περιοχή λειτουργίας οποιουδήποτε ταξινομητή. Συνεπώς, όσο πιο κοντά στην πάνω αριστερή γωνία, τόσο καλύτερα.

Το μέτρο αξιολόγησης των ταξινομητών που προκύπτει από την καμπύλη ROC είναι η επιφάνεια κάτω (και δεξιά) από την καμπύλη ROC. Όσο πιο μεγάλη είναι αυτή η περιοχή

(κάτω δεξιά της καμπύλης), τόσο καλύτερος ο ταξινομητής. Προφανώς η μέγιστη τιμή του εμβαδού είναι 1.

Οπότε στην εικόνα 1, αν θεωρήσουμε ότι η μπλε καμπύλη προέρχεται από τον ταξινομητή Α και η μωβ καμπύλη από τον ταξινομητή Β, τότε ο ταξινομητής Β είναι καλύτερος από τον Α καθώς η καμπύλη του Β συγκλίνει πιο πολύ στην πάνω αριστερή γωνία και (συνεπώς) η περιοχή κάτω από την καμπύλη Β είναι μεγαλύτερη από την αντίστοιχη περιοχή του Α.

3.4 Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines)

Οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines) προτάθηκαν από το Vladimir Vapnik το 1995 [6] και είναι μέθοδοι μάθησης με επίβλεψη που χρησιμοποιούνται σε προβλήματα ταξινόμησης και παλινδρόμησης. Στηρίζονται στη θεωρία στατιστικής μάθησης και στα νευρωνικά δίκτυα τύπου Perceptron.

Ο αλγόριθμος των μηχανών διανυσμάτων υποστήριξης προσπαθεί να εντοπίσει το βέλτιστο υπερ-επίπεδο στο χώρο των χαρακτηριστικών το οποίο να διαχωρίζει τα θετικά από τα αρνητικά παραδείγματα. Η επιλογή του βέλτιστου υπερ-επίπεδου πραγματοποιείται με τέτοιο τρόπο ώστε να απέχει όσο το δυνατόν περισσότερο από τα πιο κοντινά θετικά και αρνητικά παραδείγματα (maximum margin hyperplane) [7]. Τα παραδείγματα με την πιο μικρή απόσταση από το επιλεγμένο υπερ-επίπεδο ονομάζονται διανύσματα υποστήριξης (support vectors). Το αποτέλεσμα ενός SVM είναι μία αριθμητική τιμή στο διάστημα [-1,1].

Οι μηχανές διανυσμάτων υποστήριξης εκτός από την εφαρμογή τους σε γραμμικώς διαχωρίσιμα προβλήματα, μπορούν να εφαρμοστούν και σε περιπτώσεις που τα θετικά και αρνητικά παραδείγματα δεν είναι γραμμικώς διαχωρίσιμα. Στις περιπτώσεις των μη-γραμμικά διαχωρίσιμων προβλημάτων χρησιμοποιούνται οι λεγόμενες συναρτήσεις πυρήνα (kernel functions) για τον εντοπισμό του βέλτιστου μη-γραμμικού υπερ-επίπεδου σε ένα νέο μετασχηματισμένο χώρο χαρακτηριστικών.

Τα SVMs αποτελούν σήμερα μία από τις πιο διαδεδομένες και δημοφιλέστερες μεθόδους γραμμικής και μη, ταξινόμησης. Συγκεκριμένα αποτελούν μία από τις βέλτιστες επιλογές για εφαρμογές όπως η ταξινόμηση κειμένων, η ταξινόμηση δεδομένων έκφρασης γονιδίων κ.α..

Κάποιοι από τους λόγους που συνετέλεσαν στην διάδοση τους και στην εφαρμογή τους σε αρκετούς ερευνητικούς τομείς, είναι η αποτελεσματικότητα και η ταχύτητα που επιδεικνύουν, αλλά και της ικανότητά τους να παράγουν μη-γραμμικά υπερ-επίπεδα, καθιστώντας με αυτό τον τρόπο υπολογιστικά εφικτή την επίλυση ενός μεγάλου αριθμού πρακτικών προβλημάτων μάθησης που δεν μπορούν να αντιμετωπιστούν από γραμμικά μοντέλα.

Στην συνέχεια αναλύονται τα βασικά σημεία της θεωρίας των SVMs, στην απλή περίπτωση ενός προβλήματος ταξινόμησης δύο γραμμικά διαχωρίσιμων κλάσεων και δύο μη-γραμμικά διαχωρίσιμων κλάσεων.

3.4.1 Γραμμικά Διαχωρίσιμα Προβλήματα

Υποθέτουμε ότι αντιμετωπίζουμε το πρόβλημα ταξινόμησης δύο κλάσεων C_0 και C_1, οι οποίες είναι γραμμικά διαχωρίσιμες. Οπότε υπάρχει ένα διάνυσμα **w** και ένα κατώφλι w0 τέτοια ώστε:

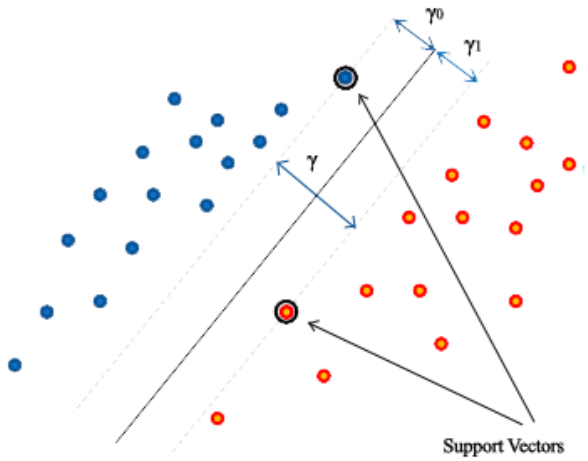
$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = \begin{cases} < 0 & \alpha\nu\ \boldsymbol{x} \in C_0 \\ > 0 & \alpha\nu\ \boldsymbol{x} \in C_1 \end{cases} \tag{11}$$

Επειδή υπάρχουν άπειρες λύσεις, δηλαδή άπειρα ζεύγη (**w**,w_0) για τον διαχωρισμό των κλάσεων, θεωρούμε κάποιο κριτήριο αξιολόγησης των λύσεων. Το κριτήριο αυτό είναι το περιθώριο ταξινόμησης γ (margin) μεταξύ των δύο κλάσεων το οποίο ορίζεται ως το άθροισμα $\gamma = \gamma_0 + \gamma_1$ μεταξύ των δύο παρακάτω περιθωρίων (το γ_0 για την κλάση C_0 και το γ_1 για την κλάση C_1).

$$\gamma_0 = \min_{x \epsilon C_o} \frac{-(\boldsymbol{w}^T\boldsymbol{x} + w_0)}{\| \boldsymbol{w} \|}$$

$$\gamma_1 = \min_{x \epsilon C_1} \frac{(\boldsymbol{w}^T\boldsymbol{x} + w_0)}{\| \boldsymbol{w} \|} \tag{12}$$

Για την ερμηνεία του παραπάνω κριτηρίου αρκεί να παρατηρήσουμε ότι η τιμή $|\mathbf{w}^T\mathbf{x}+w_0| / \|w\|$ είναι ευθέως ανάλογη με την απόσταση του διανύσματος **x** από την γραμμική διαχωριστική επιφάνεια (διαχωριστικό υπερ-επίπεδο). Οπότε οι τιμές γ_0 και γ_1 είναι θετικές και δηλώνουν την απόσταση του κοντινότερου προτύπου της κάθε κλάσης από την διαχωριστική επιφάνεια. Όσο οι τιμές γ_0 και γ_1 πλησιάζουν κοντά στο μηδέν, τόσο πιο οριακή είναι η σωστή ταξινόμηση των προτύπων της κλάσης C_0 και C_1 αντίστοιχα. Τα πρότυπα **x** της κλάσης C_0 και τα πρότυπα **x'** της κλάσης C_1, για τα οποία επιτυγχάνεται η ελάχιστη απόσταση (η ελάχιστη απόλυτη τιμή των γ_0 και γ_1), αποκαλούνται διανύσματα υποστήριξης (support vectors). Συνεπώς, το συνολικό περιθώριο ταξινόμησης γ αποτελεί ένα μέτρο εγγύτητας μεταξύ των κλάσεων C_0 και C_1.



Εικόνα 2. *Βέλτιστο διαχωριστικό υπερεπιπέδο, περιθώριο γ = γ1 + γ2 και support vectors.*

Θεωρούμε στην συνέχεια το «κανονικό διαχωριστικό υπερεπίπεδο» για το οποίο:

i. Το κατώφλι w_0 τοποθετείται ακριβώς στη μέση, μεταξύ των δύο κλάσεων. Δηλαδή, $\gamma_0 = \gamma_1$.

ii. Η κλιμάκωση των **w** και w_0 είναι τέτοια ώστε:

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = \begin{cases} \leq -1 & \alpha\nu\ \boldsymbol{x} \in C_0 \\ \geq 1 & \alpha\nu\ \boldsymbol{x} \in C_1 \end{cases} \tag{13}$$

Συνεπώς, σύμφωνα με την (12) προκύπτει:

$$\gamma_0 = \gamma_1 = \frac{1}{\| \boldsymbol{w} \|} \quad \kappa\alpha\iota \quad \gamma = \frac{2}{\| \boldsymbol{w} \|} \tag{14}$$

Στην συνέχεια, θεωρούμε ότι διαθέτουμε P ζεύγη προτύπων - στόχων $(\mathbf{x}_1, d_1), \ldots, (\mathbf{x}_p, d_p)$. Οπότε οι ανισότητες της (13) που περιγράφουν το «κανονικό διαχωριστικό υπερεπίπεδο» απλοποιούνται ως εξής:

$$d_i(\boldsymbol{w}^T\boldsymbol{x_i} + w_0) \geq 1, \quad i = 1, \ldots, P \tag{15}$$

Μπορούμε πλέον να ορίσουμε ως καταλληλότερη λύση του προβλήματος, το «κανονικό διαχωριστικό υπερεπίπεδο» της παραπάνω μορφής. Το πρόβλημα συνεπώς μετατρέπεται στον

προσδιορισμό του ελαχίστου της παρακάτω συνάρτησης υπό τους περιορισμούς των P ανισοτήτων της (15).

$$J(\boldsymbol{w}, w_0) = \frac{1}{2} \| \boldsymbol{w} \|^2 \tag{16}$$

Το παραπάνω πρόβλημα βελτιστοποίησης συναντάται στη βιβλιογραφία ως πρόβλημα τετραγωνικής βελτιστοποίησης με περιορισμούς [8] και αποδεικνύεται ότι το διάνυσμα λύσης **w** είναι ένας θετικός γραμμικός συνδυασμός των διανυσμάτων υποστήριξης.

$$\boldsymbol{w} = \sum_{i \in I_{SV}} \lambda_i d_i \boldsymbol{x}_i \tag{17}$$

όπου I_{SV} είναι το σύνολο που αποτελείται από τα διανύσματα υποστήριξης, και λ_i είναι οι βέλτιστοι πολλαπλασιαστές Lagrange [8].

Η βέλτιστη διαχωριστική επιφάνεια είναι η ακόλουθη:

$$g^*(x) = \sum_{i \in I_{SV}} \lambda_i d_i \boldsymbol{x}_i^T \boldsymbol{x} + w_0 \tag{18}$$

Και το κατώφλι υπολογίζεται ως εξής:

$$w_0 = \frac{1}{|I_{SV}|} \sum_{i \in I_{SV}} \left(\frac{1}{d_i} - \boldsymbol{w}^T \boldsymbol{x}_i \right) \tag{19}$$

3.4.2 Μη Γραμμικά Διαχωρίσιμα Προβλήματα

Στην περίπτωση των μη γραμμικά διαχωρίσιμων προβλημάτων, δηλαδή στα προβλήματα όπου οι κλάσεις δεν είναι γραμμικά διαχωρίσιμες, το πρόβλημα μετατρέπεται στον προσδιορισμό του ελαχίστου της παρακάτω συνάρτησης

$$J_{ns}(\boldsymbol{w}, w_0) = \frac{1}{2} \| \boldsymbol{w} \|^2 + C \sum_{i=1}^{P} \xi_i \tag{20}$$

υπό τους περιορισμούς των παρακάτω P ανισοτήτων

$$d_i(\boldsymbol{w}^T \boldsymbol{x_i} + w_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{21}$$

όπου i = 1, …, P, η μεταβλητή ξ_i η οποία αποκαλείται μεταβλητή χαλαρότητας και η παράμετρος C η οποία επιλέγεται από το χρήστη και είναι το βάρος του κόστους των λάθος ταξινομήσεων. Όταν C = 0, αγνοούνται τελείως οι παράμετροι χαλαρότητας και οι λάθος ταξινομήσεις δεν μας ενδιαφέρουν. Στην αντίθετη περίπτωση, όπου το C έχει μεγάλη τιμή, τότε ο ταξινομητής γίνεται πιο 'αυστηρός' και δίνει μεγάλη σημασία στην σωστή ταξινόμηση των προτύπων.

Το παραπάνω πρόβλημα αντιμετωπίζεται όπως και στα γραμμικά διαχωρίσιμα προβλήματα και το διάνυσμα λύσης βρίσκεται ότι είναι ίδιο όπως στην (17). Όπως είναι κατανοητό, επειδή χρησιμοποιούνται γραμμικές συναρτήσεις για τον διαχωρισμό μη γραμμικών κλάσεων, είναι πολύ πιθανόν αρκετά πρότυπα να μην ταξινομηθούν σωστά.

3.4.3 Χρήση Συναρτήσεων Πυρήνα (Kernel functions)

Όπως προαναφέρθηκε στα προηγούμενα υποκεφάλαια, οι μηχανές διανυσμάτων υποστήριξης μπορούν να κατασκευάσουν γραμμικά διαχωριστικά υπερεπίπεδα για τον καλύτερο δυνατό διαχωρισμό δύο κλάσεων οι οποίες μπορεί να είναι είτε γραμμικά διαχωρίσιμες είτε όχι. Βέβαια, υπάρχουν και περιπτώσεις όπου μπορεί να επιτευχθεί πολύ καλύτερος διαχωρισμός με μη-γραμμικά διαχωριστικά υπερεπίπεδα, για την αποφυγή λανθασμένων ταξινομήσεων που είναι πολύ πιθανόν να προκύψουν όπως προαναφέρθηκε στο προηγούμενο υποκεφάλαιο.

Για τον σκοπό αυτό, χρησιμοποιείται κάποιος κατάλληλος μη-γραμμικός μετασχηματισμός $\Phi(\cdot)$, αντιστοιχίζοντας το διάνυσμα εισόδου σε ένα χώρο χαρακτηριστικών μεγαλύτερης διάστασης στον οποίο τα πρότυπα $\Phi(x_1)$, …, $\Phi(x_P)$ γίνονται γραμμικά διαχωρίσιμα. Στη συνέχεια κατασκευάζεται στο χώρο αυτό το βέλτιστο διαχωριστικό υπερεπίπεδο, χρησιμοποιώντας τα μετασχηματισμένα πρότυπα.

Σε αυτήν την περίπτωση, διατηρώντας τα αποτελέσματα των προηγούμενων υποκεφαλαίων, αρκεί να αντικατασταθεί το κάθε πρότυπο x_i με το $\Phi(x_i)$, οπότε πλέον το βέλτιστο διαχωριστικό υπερεπίπεδο θα είναι το εξής

$$g^*(x) = \sum_{i=1}^{P} \lambda_i d_i \, \Phi(\boldsymbol{x}_i)^T \, \Phi(\boldsymbol{x}) + w_0 \tag{22}$$

Αντίστοιχα το κατώφλι θα είναι το εξής

$$w_0 = \frac{1}{|I_{SV}|} \sum_{i \in I_{SV}} \left(\frac{1}{d_i} - \sum_{j=1}^{P} \lambda_j d_j \, \Phi(\boldsymbol{x}_j)^T \, \Phi(\boldsymbol{x}_i) \right) \tag{23}$$

Παρατηρώντας στις παραπάνω εξισώσεις ότι εμφανίζονται εσωτερικά γινόμενα της μορφής $\Phi(\mathbf{x})^T\Phi(\mathbf{y})$ και συνυπολογίζοντας ότι η συνάρτηση μετασχηματισμού δεν εμφανίζεται πουθενά μόνη της, ορίζεται η παρακάτω συνάρτηση:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{y}) \tag{24}$$

Αντικαθιστώντας τα εσωτερικά γινόμενα της μορφής $\Phi(\mathbf{x})^T\Phi(\mathbf{y})$ στις (22) και (23) σύμφωνα με την (24), μετατρέπονται ως εξής

$$g^*(x) = \sum_{i=1}^{P} \lambda_i d_i \, k(\boldsymbol{x}_i, \boldsymbol{x}) + w_0 \tag{25}$$

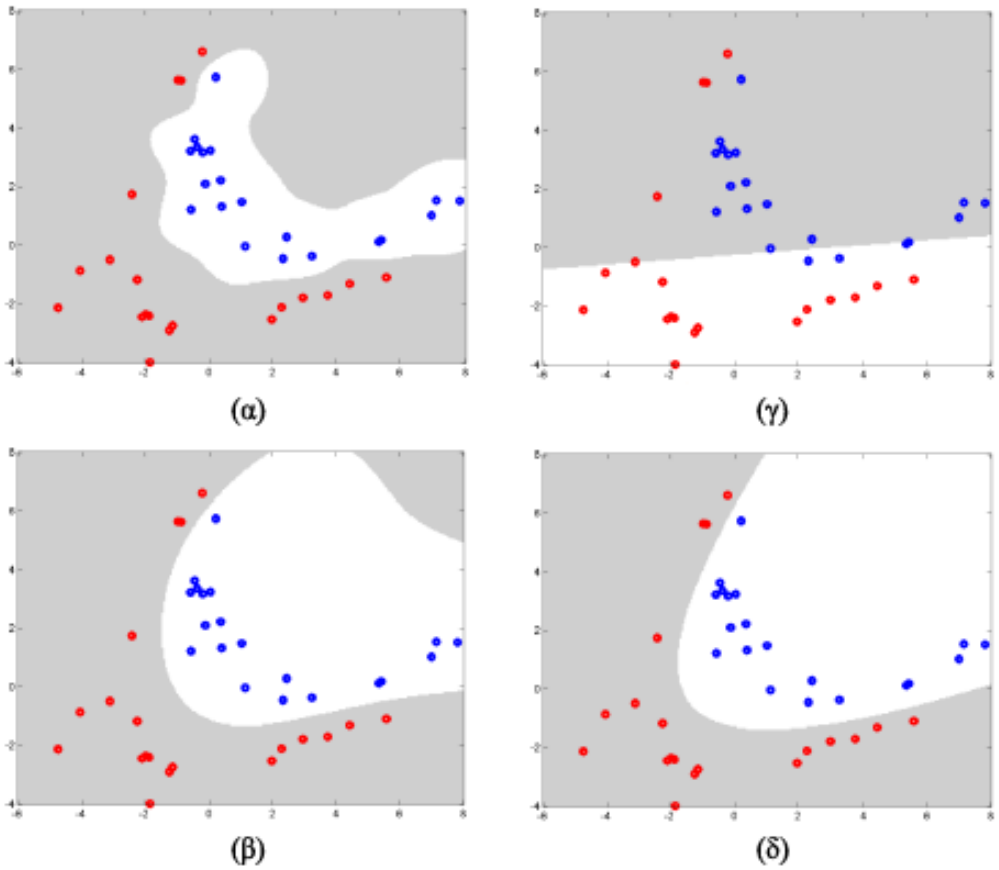
$$w_0 = \frac{1}{|I_{SV}|} \sum_{i \in I_{SV}} \left(\frac{1}{d_i} - \sum_{j=1}^{P} \lambda_j d_j \, k(\boldsymbol{x}_j, \boldsymbol{x}_i) \right) \tag{26}$$

Η συνάρτηση k(**.,.**) αποκαλείται συνάρτηση πυρήνα (kernel function) και με την χρήση της μπορούν να απλοποιηθούν οι πράξεις με αποτέλεσμα την εξοικονόμηση χρόνου. Για παράδειγμα, υπάρχουν περιπτώσεις όπου τα διανύσματα $\Phi(\mathbf{x})$ και $\Phi(\mathbf{y})$ μπορεί να έχουν αρκετά μεγάλη διάσταση και το k(x,y) μπορεί να υπολογιστεί με μικρό αριθμό πράξεων. Οι πιο συνηθισμένες συναρτήσεις πυρήνα που χρησιμοποιούνται είναι οι παρακάτω.

$e^{-\|x-y\|^2/(2\sigma^2)}$	Γκαουσσιανή (RBF)
$[\boldsymbol{x}^T\boldsymbol{y} + \theta]^p$	Πολυωνομική
$\tanh(\alpha\, \boldsymbol{x}^T\mathbf{y} + \theta)$	Σιγμοειδής
$\frac{1}{\sqrt{\| \boldsymbol{x} - \boldsymbol{y} \|^2 + c^2}}$	Αντίστροφη πολυτετραγωνική

Πίνακας 5. *Συναρτήσεις Πυρήνα k(x,y)*

Στην εικόνα 3 απεικονίζονται οι προσπάθειες επίλυσης του ίδιου προβλήματος με χρήση διαφορετικών πυρήνων.



Εικόνα 3. *Προσπάθειες επίλυσης του ίδιου προβλήματος με χρήση διαφορετικών πυρήνων (α)Γκαουσιανός $\sigma^2=1$ (β)Γκαουσιανός $\sigma^2=10$ (γ)Γραμμικός (δ)Πολυωνυμικός (τετραγωνικός)*

3.5 Διακριτοποίηση (Discretization)

Η διακριτοποίηση είναι η μέθοδος που ακολουθείται σε ένα συνεχές διάστημα τιμών για την μετατροπή του σε έναν αριθμό διακριτών τιμών. Κάποιες από τις πιο δημοφιλή μεθόδους διακριτοποίησης είναι η διακριτοποίηση με διαχωρισμό ίσων διαστημάτων (equidistant binning), με διαχωρισμό ίσης συχνότητας (equiprobable binning) και με διαχωρισμό βάσει της Εντροπίας.

Κατά την διακριτοποίηση με διαχωρισμό ίσων διαστημάτων, το συνεχές διάστημα τιμών διαιρείται σε έναν προκαθορισμένο αριθμό ίσων διαστημάτων (bins) με τιμή έναρξης του πρώτου διαστήματος την μικρότερη τιμή του συνεχούς διαστήματος τιμών και με άνω τιμή του τελευταίου διαστήματος την μεγαλύτερη τιμή του συνεχούς διαστήματος τιμών. Στην συνέχεια ανατίθεται σε κάθε διάστημα μία τιμή όπως π.χ. ο μέσος όρος των 'τοιχωμάτων' του κάθε διαστήματος. Η παρούσα διαδικασία μπορεί να υλοποιηθεί με την δημιουργία ενός

ιστογράμματος και τον ορισμό του μέσου όρου κάθε διαστήματος. Τέλος για την ολοκλήρωση της διακριτοποίησης, η κάθε τιμή από το συνεχές διάστημα τιμών αντικαθίσταται με τον μέσο όρο του αντίστοιχου διαστήματος στο οποίο ανήκει.

Στην διακριτοποίηση με διαχωρισμό ίσης συχνότητας, τα διαστήματα ορίζονται με βάση τον αριθμό εμφάνισης των τιμών του συνεχούς διαστήματος σε κάθε διάστημα. Δηλαδή ορίζονται τα διαστήματα, έτσι ώστε όλα να περιέχουν τον ίδιο αριθμό τιμών. Στην παρούσα μέθοδο δεν λαμβάνεται υπόψη το εύρος των διαστημάτων.

3.6 Μείωση Διαστάσεων (Dimensionality Reduction)

Ο μεγάλος αριθμός διαστάσεων που εμφανίζεται σε σύνολα δεδομένων τόσο στον τομέα της Βιοπληροφορικής όσο και σε άλλους τομείς αποτελεί μία από τις σημαντικότερες προκλήσεις. Όπως έχει προαναφερθεί, μπορεί να δημιουργήσει προβλήματα αποδοτικότητας και αποτελεσματικότητας στις μεθόδους μάθησης. Αρκετές μέθοδοι δεν κλιμακώνονται καλά καθώς αυξάνεται ο αριθμός των χαρακτηριστικών. Το μέγεθος του προβλήματος που μπορεί να προκαλέσει το μεγάλο πλήθος των διαστάσεων, μπορεί να περιγράφει από την χαρακτηριστική έκφραση ‘κατάρα των πολλών διαστάσεων’ (‘curse of dimensionality’). Για αυτόν τον λόγο η μείωση των διαστάσεων και η κατασκευή ενός υποσυνόλου αποτελεί ένα από τα βασικά βήματα που ακολουθείται σε εφαρμογές ταξινόμησης (κατηγοριοποίησης) και συσταδοποίησης δεδομένων.

Κατά την διαδικασία της εκπαίδευσης ενός ταξινομητή, συχνά υπάρχουν πολλά χαρακτηριστικά των οποίων η γνώση δεν είναι απαραίτητη για την σωστή ταξινόμηση των στιγμιότυπων στις αντίστοιχες κλάσεις. Τα χαρακτηριστικά αυτά μπορεί να περιέχουν πληροφορία η οποία είναι είτε άσχετη με το πρόβλημα της ταξινόμησης, είτε περιττή επειδή η ίδια πληροφορία περιέχεται και σε άλλα χαρακτηριστικά. Ο στόχος κατά την επιλογή των χαρακτηριστικών είναι ο εντοπισμός των χαρακτηριστικών τα οποία περιέχουν την απαραίτητη πληροφορία για την σωστή ταξινόμηση των στιγμιότυπων στις αντίστοιχες κλάσεις. Ουσιαστικά, ο σκοπός είναι να χρησιμοποιηθούν υποσύνολα από το σύνολο των χαρακτηριστικών, των οποίων τα χαρακτηριστικά είναι τα πιο σχετικά με τη μεταβλητή κλάσης, και τα λιγότερο πλεονάζοντα μεταξύ τους [9][10][11].

Εφόσον αγνοηθούν μόνο τα άσχετα και τα περιττά χαρακτηριστικά, μπορεί να κατασκευαστεί ένας ταξινομητής η απόδοση του οποίου σίγουρα δεν θα είναι χειρότερη από την απόδοση ενός ταξινομητή ο οποίος θα χρησιμοποιούσε όλο το σύνολο των χαρακτηριστικών. Τις περισσότερες φορές η απόδοση ενός ταξινομητή που χρησιμοποιεί μόνο τα χαρακτηριστικά που περιέχουν σημαντική πληροφορία, δεν είναι απλώς χειρότερη αλλά αντιθέτως βελτιώνει σημαντικά τα αποτελέσματα που επιτυγχάνει ο ταξινομητής. Συνυπολογίζοντας και την μείωση της πιθανότητας εμφάνισης του φαινομένου της υπερεκπαίδευσης (overfitting) λόγω

των μειωμένων χαρακτηριστικών, η επιλογή χαρακτηριστικών αποδεικνύει ότι κατέχει έναν από τους κυριότερους πυλώνες για την κατασκευή ενός αποδοτικού ταξινομητή.

Τα οφέλη που προκύπτουν σε εφαρμογές κατηγοριοποίησης (ταξινόμησης) οι οποίες βασίζονται σε δείγματα που περιγράφονται από διανύσματα χαρακτηριστικών είναι πολλά. Αρχικά, όπως προαναφέρθηκε, η μείωση των διανυσμάτων επηρεάζει θετικά την απόδοση των αλγορίθμων μηχανικής μάθησης λόγω της μείωσης του κινδύνου εμφάνισης υπερεκπαίδευσης. Στην περίπτωση χρήσης διανυσμάτων μεγάλων διαστάσεων για την μάθηση με επίβλεψη, ο κίνδυνος εμφάνισης υπερεκπαίδευσης είναι αρκετά αυξημένος.

Ένα επιπλέον πλεονέκτημα είναι ότι μία απεικόνιση που ορίζεται με βάση έναν μικρό αριθμό χαρακτηριστικών είναι πιο εύκολα κατανοητή και ερμηνεύσιμη από μία απεικόνιση που ορίζεται με βάση πολλά χαρακτηριστικά. Επίσης, με την επιλογή των χαρακτηριστικών που είναι πιο σημαντικά για την διαδικασία παραγωγής των αποτελεσμάτων, δίνεται η δυνατότητα στους ειδικούς του αντίστοιχου τομέα να αποκτήσουν μία καλύτερη διαίσθηση του προβλήματος. Εντοπίζοντας οι ειδικοί τα πραγματικά αίτια του προβλήματος, μπορούν να το αντιμετωπίσουν πιο αποτελεσματικά. Συγκεκριμένα στον τομέα της Βιοπληροφορικής για παράδειγμα δίνεται η δυνατότητα να αναγνωριστούν τα γονίδια που σχετίζονται με την εμφάνιση διάφορων νόσων.

Ένα ακόμα σημαντικό όφελος που προκύπτει από την επιλογή χαρακτηριστικών είναι η μείωση του υπολογιστικού κόστους κατά την εκπαίδευση ενός ταξινομητή καθώς και η γρηγορότερη ολοκλήρωση της διαδικασίας εκπαίδευσης. Είναι προφανές ότι το κέρδος σε υπολογιστικό κόστος και σε απαιτήσεις αποθηκευτικού χώρου, αυξάνονται όσο μειώνεται ο αριθμός των χαρακτηριστικών που χρησιμοποιούνται κατά την διαδικασία της εκπαίδευσης ενός ταξινομητή[12].

Οι βασικοί τρόποι που χρησιμοποιούνται για την μείωση της διάστασης των χαρακτηριστικών είναι οι παρακάτω:

- Η Εξαγωγή Χαρακτηριστικών (Feature Extraction, FE)
- Η Επιλογή ενός Υποσυνόλου Χαρακτηριστικών (Subset Feature Selection, SFS)

3.6.1 Εξαγωγή Χαρακτηριστικών (Feature Extraction, FE)

Κατά την εξαγωγή χαρακτηριστικών, παράγονται νέα χαρακτηριστικά από τις προβολές του αρχικού συνόλου όλων των χαρακτηριστικών, κάποια από τα οποία πολύ πιθανόν να συσχετίζονται μεταξύ τους. Αυτός ο μετασχηματισμός του αρχικού συνόλου των χαρακτηριστικών έχει ως αποτέλεσμα την απώλεια της πληροφορίας που περιέχει το κάθε χαρακτηριστικό. Τα νέα χαρακτηριστικά που παράγονται είναι ασυσχέτιστα μεταξύ τους και λιγότερα από τα αρχικά. Η ανάλυση κυρίων συνιστωσών (principal component analysis, PCA) αποτελεί την πιο διαδομένη μέθοδο εξαγωγής χαρακτηριστικών [10].

3.6.2 Επιλογή Υποσυνόλου Χαρακτηριστικών (Subset Feature Selection, SFS)

Αντιθέτως, κατά την επιλογή ενός υποσυνόλου χαρακτηριστικών, το κάθε χαρακτηριστικό διατηρεί την πληροφορία που περιέχει και στο αρχικό σύνολο. Για την δημιουργία του υποσυνόλου απορρίπτονται τα χαρακτηριστικά που περιέχουν πληροφορία η οποία είναι είτε άσχετη με το πρόβλημα της ταξινόμησης, είτε περιττή επειδή η ίδια πληροφορία περιέχεται και σε άλλα χαρακτηριστικά. Το τελικό υποσύνολο των χαρακτηριστικών που επιλέγεται από τα αρχικά χαρακτηριστικά, είναι αυτό που κατά την εκπαίδευση και επικύρωση του εκάστοτε ταξινομητή δίνει τα βέλτιστα αποτελέσματα.

Για την επιλογή υποσυνόλου χαρακτηριστικών, οι δύο από τις βασικές προσεγγίσεις είναι οι παρακάτω:

- Η προσέγγιση του ‘περιτυλίγματος’ (wrapper approach)
- Η προσέγγιση του ‘φίλτρου’ (filter approach)

Η προσέγγιση του ‘περιτυλίγματος’ χρησιμοποιεί κάποιον ταξινομητή κατά τη διαδικασία επιλογής του βέλτιστου υποσυνόλου χαρακτηριστικών, σε αντίθεση με την προσέγγιση του ‘φίλτρου’ η οποία βασίζεται αποκλειστικά στην κατάταξη των χαρακτηριστικών βάση του εκάστοτε κριτηρίου αξιολόγησης [11].

3.6.2.1 Προσέγγιση του ‘περιτυλίγματος’ (Wrapper Approach)

Στην προσέγγιση του ‘περιτυλίγματος’ η επιλογή του υποσυνόλου των χαρακτηριστικών είναι προσκολλημένη στην διαδικασία εκπαίδευσης του ταξινομητή, όπου γίνεται χρήση της απόδοσης του ως κριτήριο για να εκτιμηθεί και να καθοριστεί το υποσύνολο των χαρακτηριστικών που θα επιλεγεί. Δηλαδή από την εφαρμογή ενός ταξινομητή, για κάθε υποψήφιο υποσύνολο χαρακτηριστικών, μετράτε η ακρίβεια της κατηγοριοποίησης (accuracy) και στην συνέχεια πραγματοποιείται σύγκριση μεταξύ των υποσυνόλων. Το υποσύνολο με την μεγαλύτερη ακρίβεια ταξινόμησης είναι αυτό που θα επιλεγεί ως το βέλτιστο υποσύνολο. Εκτός από την ακρίβεια της κατηγοριοποίησης του ταξινομητή ως κριτήριο αξιολόγησης, χρησιμοποιείται επίσης και ο ρυθμός σφάλματος της κατηγοριοποίησης (error rate) [9]. Οι ταξινομητές που χρησιμοποιούνται συνήθως είναι οι μηχανές διανυσμάτων υποστήριξης (SVM), τα νευρωνικά δίκτυα κ.α.

Η προσέγγιση του ‘περιτυλίγματος’ συνήθως είναι πιο αποτελεσματική και πιο ακριβής επειδή τα χαρακτηριστικά αξιολογούνται ως μέρη ενός υποσυνόλου και όχι μεμονωμένα. Με αυτόν τον τρόπο δίνεται η δυνατότητα να ανακαλυφθούν τυχόν αλληλεπιδράσεις μεταξύ των χαρακτηριστικών στην περίπτωση που αυτά τα χαρακτηριστικά τύχει να βρεθούν στο ίδιο υποψήφιο υποσύνολο προς αξιολόγηση. Επειδή ο απώτερος στόχος της επιλογής του

κατάλληλου υποσυνόλου είναι η βέλτιστη απόδοση του ταξινομητή, η χρήση της ακρίβειας του ταξινομητή ως κριτήριο θεωρείται και πιο σωστή ως αντιμετώπιση [9].

Λαμβάνοντας βέβαια υπόψιν ότι ο κάθε ταξινομητής που μπορεί να χρησιμοποιηθεί έχει τα δικά του χαρακτηριστικά, μπορεί το βέλτιστο υποσύνολο βάση ενός συγκεκριμένου ταξινομητή να μην αποτελεί το ίδιο αποδοτικό υποσύνολο για έναν άλλο ταξινομητή. Η ακρίβεια ταξινόμησης αποτελεί το πιο αξιόπιστο κριτήριο για την εύρεση του βέλτιστου υποσυνόλου αλλά μόνο σε συνδυασμό με έναν ταξινομητή [9].

Ως βασικό μειονέκτημα της συγκεκριμένης προσέγγισης αποτελεί το μεγάλο υπολογιστικό κόστος. Καθώς η απόδοση του ταξινομητή αποτελεί το κριτήριο αξιολόγησης, απαιτείται η κατασκευή του ταξινομητή για κάθε υποψήφιο υποσύνολο χαρακτηριστικών που ελέγχεται. Αυτή η διαδικασία έχει ως αρνητικό επακόλουθο την αύξηση του υπολογιστικού κόστους και του χρόνου εκτέλεσης σε σχέση με την προσέγγιση του 'φίλτρου' [9]. Επιπλέον, οι προσεγγίσεις 'περιτυλίγματος' πάσχουν από έλλειψη γενίκευσης επειδή το αποτέλεσμα τους, όπως προαναφέρθηκε, βασίζεται σε έναν συγκεκριμένο ταξινομητή [9][13].

3.6.2.2 Προσέγγιση του 'φίλτρου' (Filter Approach)

Στην προσέγγιση του 'φίλτρου' ανήκουν αλγόριθμοι οι οποίοι βασίζονται σε στατιστικά μέτρα. Σε αντίθεση με την προσέγγιση του 'περιτυλίγματος', όπου γίνεται χρήση ενός ταξινομητή για την εκτίμηση της απόδοσης του κάθε υποψήφιου υποσυνόλου, η προσέγγιση του 'φίλτρου' βασίζεται στην έννοια της συνάφειας μεταξύ των χαρακτηριστικών και των αντίστοιχων κλάσεων χρησιμοποιώντας στατιστικά κριτήρια αξιολόγησης. Η συγκεκριμένη προσέγγιση διαχωρίζεται σε δύο βασικές κατηγορίες μεθόδων, στις μονοπαραγοντικές (univariate) μεθόδους και στις πολυπαραγοντικές (multivariate) μεθόδους. [14][15]

- **Μονοπαραγοντικές μέθοδοι (Univariate Methods)**

Στις μονοπαραγοντικές μεθόδους το κάθε χαρακτηριστικό αξιολογείται μεμονωμένα με βάση την συσχέτιση του με την κλάση. Όσο μεγαλύτερη είναι η συσχέτιση του με την κλάση τόσο πιο σημαντικό θεωρείται το χαρακτηριστικό. Έπειτα από την αξιολόγηση του κάθε χαρακτηριστικού, επιλέγονται τα N πιο σημαντικά χαρακτηριστικά. Η επιλογή του αριθμού N των χαρακτηριστικών καθορίζεται εμπειρικά αναλόγως την περίσταση.[14][15]

Τα πλεονεκτήματα των μονοπαραγοντικών μεθόδων είναι ότι μπορούν εύκολα να κλιμακωθούν σε σύνολα δεδομένων πολύ υψηλών διαστάσεων, είναι υπολογιστικά απλά και γρήγορα. Επίσης, λόγω της υλοποίησης τους χωρίς την χρήση κάποιου ταξινομητή, είναι ανεξάρτητοι από το είδος του ταξινομητή που χρησιμοποιείται στην συνέχεια. Η επιλογή του καλύτερου υποσυνόλου χαρακτηριστικών γίνεται μία φορά και στην συνέχεια μπορούν να εφαρμοστούν και να αξιολογηθούν σε αυτό διαφορετικοί ταξινομητές. Το επιλεγμένο

υποσύνολο χαρακτηριστικών μπορεί να επιτύχει αποδοτικά αποτελέσματα σε παραπάνω του ενός ταξινομητή.[14]

Το πιο βασικό μειονέκτημα τους είναι ο πλεονασμός κατά την επιλογή των χαρακτηριστικών. Επειδή, όπως προαναφέρθηκε το κάθε χαρακτηριστικό αξιολογείται μεμονωμένα, δεν λαμβάνονται υπόψιν οι εξαρτήσεις μεταξύ των χαρακτηριστικών. Αυτό έχει ως αποτέλεσμα την επιλογή περιττών χαρακτηριστικών. Δηλαδή, επιλέγονται χαρακτηριστικά τα οποία περιέχουν όμοια πληροφορία, με αποτέλεσμα η συνύπαρξη τους στο επιλεγμένο υποσύνολο να μην προσφέρει πολύ περισσότερη πληροφορία, από αυτή που θα προσέφερε το κάθε χαρακτηριστικό από μόνο του. Προκειμένου να ξεπεραστεί το πρόβλημα του πλεονασμού, εισήχθησαν οι πολυπαραγοντικές μέθοδοι με στόχο την ενσωμάτωση εξαρτήσεων μεταξύ των χαρακτηριστικών κατά την αξιολόγηση τους. Επίσης ένα ακόμα μειονέκτημα των μονοπαραγοντικών μεθόδων είναι ότι αγνοούν την αλληλεπίδραση με τον ταξινομητή που θα χρησιμοποιηθεί στην συνέχεια. [14]

Για την μέτρηση και την αξιολόγηση της συσχέτισης των χαρακτηριστικών με τις αντίστοιχες κλάσεις υπάρχουν διάφορα κριτήρια όπως το κριτήριο του Fisher, η μέτρηση της αμοιβαίας πληροφορίας και το κριτήριο Kolmogorov Smirnov, τα οποία αναλύονται παρακάτω καθώς χρησιμοποιήθηκαν για την αξιολόγηση των χαρακτηριστικών στην παρούσα έρευνα. Επίσης κάποια επιπλέον κριτήρια αξιολόγησης που ανήκουν στις μονοπαραγοντικές μεθόδους είναι το Wilcoxon, το t-test και το ANOVA [14][15]

- **Πολυπαραγοντικές μέθοδοι (Multivariate Methods)**

Οι πολυπαραγοντικές μέθοδοι, όπως προαναφέρθηκε, εισήχθησαν για να ξεπεραστεί το πρόβλημα του πλεονασμού. Η επιλογή του υποσυνόλου των χαρακτηριστικών επικεντρώνεται ταυτόχρονα σε δύο στόχους. Πρώτος στόχος είναι όπως και στις μονοπαραγοντικές μεθόδους η μεγάλη συσχέτιση του κάθε χαρακτηριστικού με την αντίστοιχη κλάση. Ο δεύτερος στόχος είναι τα χαρακτηριστικά με την μεγαλύτερη συσχέτιση με την κλάση να είναι ταυτόχρονα και ανόμοια μεταξύ τους. Δηλαδή τα χαρακτηριστικά του υποσυνόλου να περιέχουν όσο το δυνατόν διαφορετική πληροφορία.

Στα πλεονεκτήματα των πολυπαραγοντικών μεθόδων συγκαταλέγεται προφανώς η ανομοιότητα των χαρακτηριστικών του επιλεγμένου υποσυνόλου η οποία τείνει να αφανίσει το φαινόμενο του πλεονασμού. Επίσης, σε σχέση με τους αλγόριθμους της προσέγγισης του ‘περιτυλίγματος’, είναι μειωμένη η υπολογιστική τους πολυπλοκότητα. Τέλος, όπως και στις μονοπαραγοντικές μεθόδους, υπάρχει ανεξαρτησία σε σχέση με τον ταξινομητή που μπορεί να χρησιμοποιηθεί στην συνέχεια.

Σε σχέση με τις μονοπαραγοντικές μεθόδους, οι πολυπαραγοντικές μέθοδοι είναι υπολογιστικά πιο πολύπλοκες, πιο αργές και λιγότερο κλιμακώσιμες σε σύνολα δεδομένων

μεγάλων διαστάσεων. Το μειονέκτημα που συνεχίζει να συντηρείται και στις πολυπαραγαροντικές μεθόδους είναι η άγνοια της αλληλεπίδρασης με τον ταξινομητή.

Για την επίτευξη των δύο στόχων των πολυπαραγοντικών μεθόδων υπάρχουν διάφοροι μέθοδοι αξιολόγησης των χαρακτηριστικών όπως η ReliefF. Επίσης μία από τις πιο βασικές μεθόδους είναι η mRMR, η οποία για την επίτευξη του πρώτου στόχου βασίζεται στο κριτήριο της αμοιβαίας πληροφορίας. Παρακάτω γίνεται εκτενής αναφορά στις μεθόδους Relief-F και mRMR, καθώς χρησιμοποιήθηκαν στην παρούσα έρευνα για την εύρεση του πιο σημαντικού υποσυνόλου χαρακτηριστικών. Μία ακόμα πολυραγοντική μέθοδος αξιολόγησης είναι το φίλτρο Markov blanket filter (MBF).

3.7 Principal Component Analysis (PCA)

Η Ανάλυση Κυρίων Συνιστωσών (Principal Component Analysis, PCA) αποτελεί μία μέθοδο γραμμικής συμπίεσης δεδομένων, κατά την οποία μπορεί να πραγματοποιηθεί μείωση του πλήθους των διαστάσεων ενός συνόλου δεδομένων μέσω του επαναπροσδιορισμού των συντεταγμένων τους. Συγκεκριμένα, υποθέτουμε ότι έχουμε ένα σύνολο δεδομένων με n γραμμές (δείγματα) και k στήλες (διαστάσεις, χαρακτηριστικά). Η μέθοδος PCA βρίσκει ένα σύστημα m κάθετων διανυσμάτων, όπου $m < k$ και προβάλει τα δεδομένα στον νέο χώρο m διαστάσεων. Με αυτήν την διαδικασία δημιουργεί γραμμικούς συνδυασμούς (principal components) των αρχικών μεταβλητών οι οποίοι διαθέτουν τις παρακάτω ιδιότητες [16]:

- Είναι ασυσχέτιστοι μεταξύ τους.
- Περιέχουν το δυνατό μεγαλύτερο μέρος της διακύμανσης των αρχικών μεταβλητών.
- Διαθέτουν φθίνουσα σειρά όσο αναφορά την τιμή της διακύμανσης με αποτέλεσμα ο πρώτος γραμμικός συνδυασμός (principal component) να διατηρεί περισσότερες πληροφορίες δεδομένων σε σχέση με τον δεύτερο, ο οποίος δεν διατηρεί πληροφορίες που έχουν εισέλθει νωρίτερα.

Ο συνολικός αριθμός των συνιστωσών (principal components) είναι ίσος με την ποσότητα των αρχικών μεταβλητών και διαθέτει το ίδιο μέγεθος πληροφορίας. Επειδή όμως οι πρώτες συνιστώσες διατηρούν περισσότερο από το 90% της συνολικής πληροφορίας από το αρχικό σύνολο δεδομένων, η μέθοδος ουσιαστικά επιτρέπει την μείωση του συνόλου των συνιστωσών [16]. Τα βήματα που ακολουθούνται για την μέθοδο PCA είναι τα παρακάτω.

Βήμα 1ο

Αρχικά, για την ορθή χρήση της μεθόδου, για κάθε μεταβλητή (στήλη) υπολογίζεται η μέση τιμή των τιμών της και αφαιρείται από όλες τις τιμές της. Το αποτέλεσμα αυτής της διαδικασίας είναι η μέση τιμή της κάθε μεταβλητής να γίνεται ίση με μηδέν.

Βήμα 2ο

Υπολογίζεται ο πίνακας συνδιασποράς του πίνακα που δημιουργήθηκε στο πρώτο βήμα. Για την κατανόηση της διαδικασίας δημιουργίας του πίνακα συνδιασποράς, ας υποθέσουμε ένα σύνολο δεδομένων με n γραμμές (δείγματα) και k=3 στήλες (διαστάσεις, χαρακτηριστικά) όπου οι τιμές των διαστάσεων απεικονίζονται σε έναν τρισδιάστατο χώρο με άξονες X, Y και Z. Η συνδιασπορά (covariance) μεταξύ των διαστάσεων (μεταβλητών) X και Y ορίζεται από την παρακάτω εξίσωση.

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{(n-1)} \tag{27}$$

Η συνδιασπορά αποτελεί στατιστικό μέτρο το οποίο δηλώνει πως μεταβάλλεται μία μεταβλητή σε σχέση με μία άλλη και υπολογίζεται πάντα μεταξύ δύο μεταβλητών. Όταν οι τιμές των δύο μεταβλητών αυξάνονται ταυτόχρονα, τότε η τιμή της συνδιασποράς είναι θετική. Αντίστοιχα, όταν οι τιμές της μίας μεταβλητής αυξάνονται και παράλληλα οι τιμές της άλλης μεταβλητής μειώνονται, τότε η τιμή της συνδιασποράς είναι αρνητική. Επίσης, η συνδιασπορά μιας μεταβλητής με τον εαυτό της είναι η διασπορά της.

Υπολογίζοντας την τιμή συνδιασποράς για το κάθε ζευγάρι μεταξύ των μεταβλητών X, Y και Z, διαμορφώνεται ο παρακάτω πίνακας συνδιασποράς (covariance matrix).

$$\mathrm{C} = \begin{pmatrix} cov(X,X) & cov(X,Y) & cov(X,Z) \\ cov(Y,X) & cov(Y,Y) & cov(Y,Z) \\ cov(Z,X) & cov(Z,Y) & cov(Z,Z) \end{pmatrix} \tag{28}$$

Ο πίνακας συνδιασποράς είναι πάντα συμμετρικός ως προς την διαγώνιο του, επειδή cov(X,Y) = cov(Y,X). Επίσης, όπως είναι κατανοητό, οι διαστάσεις του πίνακα συνδιασποράς ενός συνόλου δεδομένων με k στήλες (διαστάσεις), θα είναι πάντα k x k.

Βήμα 3ο

Υπολογίζονται τα ιδιοδιανύσματα (eigenvectors) και οι ιδιοτιμές (eigenvalues) του πίνακα συνδιασποράς. Για την κατανόηση του υπολογισμού των ζευγών ιδιοδιανυσμάτων και των ιδιοτιμών τους, ας εξετάσουμε τον παρακάτω πολλαπλασιασμό μεταξύ ενός δισδιάστατου πίνακα με ένα διάνυσμα.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{29}$$

Ας θεωρήσουμε ότι ο δισδιάστατος πίνακας είναι ένας πίνακας μετασχηματισμού του διανύσματος που συμμετέχει στον πολλαπλασιασμό. Το αποτέλεσμα του πολλαπλασιασμού, όπως φαίνεται παραπάνω, είναι ένα διάνυσμα το οποίο αποτελεί πολλαπλάσιο του αρχικού διανύσματος. Ουσιαστικά πρόκειται για ένα διάνυσμα ίδιας διεύθυνσης αλλά διαφορετικού μήκους. Αυτό το διάνυσμα, που προκύπτει από τον πολλαπλασιασμό, είναι ένα ιδιοδιάνυσμα (eigenvector) του πίνακα μετασχηματισμού. Η τιμή πολλαπλασιασμού του ιδιοδιανύσματος αποτελεί την ιδιοτιμή (eigenvalue). Ο υπολογισμός των k ζευγών ιδιοδιανυσμάτων και ιδιοτιμών απαιτεί την παρουσία ενός πίνακα διαστάσεων k x k, χωρίς όμως αυτό να σημαίνει ότι όλοι οι πίνακες με διάσταση k x k έχουν ιδιοδιάνυσματα και ιδιοτιμές. Η σχέση στην οποία καταλήγουμε σύμφωνα με τα παραπάνω είναι η εξής:

$$A \times u = \lambda \times u \tag{30}$$

όπου A ο πίνακας μετασχηματισμού (με διαστάσεις u x u), u το ιδιοδιάνυσμα και λ η ιδιοτιμή.

Τα ιδιοδυανύσματα αποτελούν τις κύριες συνιστώσες (principal componets). Οι βασικές συνιστώσες είναι πάντα ορθογώνιες μεταξύ τους και μπορούν να αποτελέσουν το νέο σύστημα αξόνων για την αναπαράσταση των δεδομένων [16].

Βήμα 4ο

Ταξινομούνται τα ιδιοδιανύσματα με βάση τις ιδιοτιμές τους καθώς οι ιδιοτιμές αποτελούν το μέτρο σημαντικότητας τους. Όσο μεγαλύτερη είναι η ιδιοτιμή ενός ιδιοδιανύσματος τόσο περισσότερη πληροφορία περιέχει το ιδιοδιάνυσμα ως συνιστώσα σε σχέση με τα υπόλοιπα ιδιοδιανύσματα. Αντίστοιχα όσο μικρότερη είναι η ιδιοτιμή ενός ιδιοδιανύσματος τόσο λιγότερη πληροφορία περιέχει το διάνυσμα ως συνιστώσα. Αυτό σημαίνει ότι στην συνέχεια μπορούν να αφαιρεθούν οι λιγότερο σημαντικές συνιστώσες χωρίς να χαθεί σημαντική ποσότητα πληροφορίας. Συνεπώς, ο λόγος που τα ιδιοδιανύσματα ταξινομούνται σε φθίνουσας σειρά, είναι ότι αν ένας μικρός αριθμός q ιδιοδιανυσμάτων αρκεί για να καλυφθεί ένα μεγάλο ποσοστό της μεταβλητότητας του δείγματος, τότε τα αρχικά διανύσματα μπορούν να συμπιεστούν από k σε q ιδιοδιανύσματα (k>q), με μικρό σφάλμα (μικρή απώλεια πληροφορίας) [16].

Βήμα 5ο

Επανακαθορισμός δεδομένων στο νέο σύστημα αξόνων. Όπως προαναφέρθηκε, οι λιγότερο σημαντικές συνιστώσες μπορούν να αφαιρεθούν εφόσον η απώλεια της πληροφορίας θα είναι μικρή. Συνεπώς, αν υπάρχουν k ιδιοτιμές και ιδιοδιανύσματα και επιλεχθούν οι πρώτες p ιδιοτιμές, καταλήγουμε σε ένα σύνολο δεδομένων με μόνο p διαστάσεις. Στην συνέχεια με

την χρήση των p διαστάσεων μπορεί να δημιουργηθεί στο νέο σύστημα αξόνων μία ικανοποιητική προσέγγιση των αρχικών δεδομένων από άποψη ποσότητας πληροφορίας σε σχέση με τα αρχικά δεδομένα.

Όπως προαναφέρθηκε, κάθε μία από τις κύριες συνιστώσες PC_1, PC_2, ..., PC_k, είναι ένας γραμμικός συνδυασμός των μεταβλητών (χαρακτηριστικών) X_1, X_2, ..., X_k των αρχικών διανυσμάτων. Συνεπώς η i-οστή συνιστώσα PC_i θα έχει την παρακάτω μορφή:

$$PC_i = a_1X_1,\ a_2X_2, \dots, a_kX_k \tag{31}$$

Οι τιμές α_1, α_2, ..., α_k καθορίζουν τον βαθμό στον οποίο η κάθε αρχική μεταβλητή επηρεάζει την κάθε συνιστώσα και ονομάζονται παράγοντες βαρύτητας [16].

Η μέθοδος PCA αν και αποτελεί μέθοδο μείωσης των χαρακτηριστικών (διαστάσεων) μέσω της εξαγωγής νέων διαστάσεων, υπάρχουν και άλλες τεχνικές χρήσης της μεθόδου για την μείωση των χαρακτηριστικών. Για παράδειγμα, αξιοποιώντας τους παράγοντες βαρύτητας των αρχικών μεταβλητών στις κύριες συνιστώσες, γίνεται χρήση της μεθόδου PCA ως μεθόδου επιλογής χαρακτηριστικών. Σύμφωνα με αυτή την εκδοχή, θα μπορούσε να επιλεχθεί μόνο η πρώτη κύρια συνιστώσα, να ταξινομηθούν οι αρχικές μεταβλητές σύμφωνα με τους παράγοντες βαρύτητας και να επιλεχθούν τα χαρακτηριστικά που διαθέτουν την μεγαλύτερη βαρύτητα.

3.8 Fisher Score

Το κριτήριο Fisher είναι ένα από τα πιο ευρέως διαδεδομένα κριτήρια για την επιλογή χαρακτηριστικών με επίβλεψη. Η βασική ιδέα του κριτηρίου Fisher είναι να βρει ένα υποσύνολο χαρακτηριστικών, έτσι ώστε στο χώρο δεδομένων που εκτείνονται τα επιλεγμένα χαρακτηριστικά, οι αποστάσεις μεταξύ των σημείων δεδομένων που ανήκουν σε διαφορετικές κλάσεις να είναι όσο το δυνατόν μεγαλύτερες, ενώ οι αποστάσεις μεταξύ των σημείων δεδομένων που ανήκουν στην ίδια κλάση να είναι όσο το δυνατόν μικρότερες [17].

Το σκορ του κριτηρίου Fisher [17] για κάθε χαρακτηριστικό για δύο κλάσεις δίνεται από τον παρακάτω τύπο

$$w_i = \frac{(\mu_{i1} - \mu_{i2})^2}{\sigma_{i1}^2 + \sigma_{i2}^2} \tag{32}$$

Τα μ_{i1} και μ_{i2} αντιστοιχούν στις μέσες τιμές του i-οστού χαρακτηριστικού για τα δείγματα της πρώτης και της δεύτερης κλάσης αντίστοιχα. Τα σ_{i1} και σ_{i2} αποτελούν τις τυπικές αποκλίσεις

του i-οστού χαρακτηριστικού για τα δείγματα της πρώτης και της δεύτερης κλάσης αντίστοιχα.

Υπολογίζοντας την βαθμολογία Fisher για κάθε χαρακτηριστικό σύμφωνα με τον παραπάνω τύπο είναι προφανές ότι τα χαρακτηριστικά των οποίων η πληροφορία διαφοροποιείται πιο πολύ μεταξύ των δύο κλάσεων, λαμβάνει υψηλότερο σκορ σε σχέση με τα χαρακτηριστικά που η πληροφορία διαφοροποιείται λιγότερο μεταξύ των δύο κλάσεων. Στην περίπτωση που ένα χαρακτηριστικό περιέχει την ίδια πληροφορία και στις δύο καταστάσεις της κλάσης λαμβάνει την μικρότερη βαθμολογία η οποία είναι μηδενική.

Ταξινομώντας τα χαρακτηριστικά σε φθίνουσα σειρά σύμφωνα με την βαθμολογία Fisher, τα πρώτα N χαρακτηριστικά τα οποία έχουν την ισχυρότερη συσχέτιση με τις αντίστοιχες κλάσεις αποτελούν το επιλεγμένο υποσύνολο το οποίο χρησιμοποιείται στην συνέχεια για την εφαρμογή του εκάστοτε ταξινομητή.

3.9 Mutual Information (MI)

Η Αμοιβαία Πληροφορία (Mutual Information) και το Κέρδος της Πληροφορίας (Information Gain) αποτελούν από τις βασικότερες μετρικές στον χώρο της θεωρίας της πληροφορίας. Η Αμοιβαία Πληροφορία σχετίζεται με την εντροπία (Entropy) της πληροφορίας. Η εντροπία της πληροφορίας αποτελεί μέτρο της αβεβαιότητας για την τιμή μιας τυχαίας μεταβλητής και βασίζεται αποκλειστικά στην κατανομή της πιθανότητας της μεταβλητής. Αν υποθέσουμε ότι X είναι μια τυχαία μεταβλητή, τότε η εντροπία της μεταβλητής X ορίζεται ως εξής:

$$H(X) = -\sum_{i=0}^{n} p_i log_2(p_i) \tag{33}$$

Όσο πιο μεγάλη είναι η τιμή της εντροπίας μίας τυχαίας μεταβλητής, τόσο μεγαλύτερη είναι και η αβεβαιότητα που υπάρχει για αυτή. Δηλαδή, όσο μεγαλύτερη είναι η ποσότητα της εντροπίας μιας τυχαίας μεταβλητής, τόσο λιγότερο καλές μπορεί να είναι και οι προβλέψεις μας για αυτή [18]. Η ποσότητα H(X) μετράτε σε bits. Επίσης καλείται και ως εντροπία του Shannon επειδή εισήχθη ως έννοια από τον Claude E. Shannon [19].

Όταν η τιμή μίας τυχαίας μεταβλητής X εξαρτάται από την τιμή μίας άλλης τυχαίας μεταβλητής Y, τότε μπορεί να οριστεί η εντροπία υπό συνθήκη (Conditional Entropy) ή Δεσμευμένη Εντροπία. Η εντροπία υπό συνθήκη αποτελεί την ποσότητα της αβεβαιότητας που μένει για την X, όταν η Y είναι ήδη γνωστή και ορίζεται ως εξής [18]:

$$H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} p(X = x_i, Y = y_i) log_2(p(X = x_i|Y = y_i)) \quad (34)$$

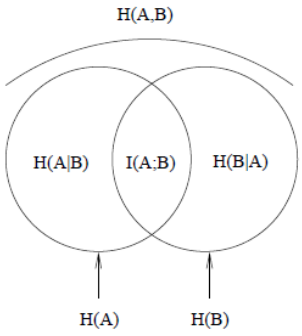
Ουσιαστικά η υπό συνθήκη εντροπία μας δίνει την πληροφορία που μπορούμε να πάρουμε από μία μεταβλητή X δεδομένου ότι είναι γνωστή η μεταβλητή Y.

Η πιθανότητα $p(X = x_i, Y = y_i)$ αποκαλείται από κοινού πιθανότητα και αποτελεί την πιθανότητα οι μεταβλητές X και Y να πάρουν ταυτόχρονα τις τιμές x_i και y_i αντίστοιχα. Η πιθανότητα $p(X = x_i \mid Y = y_i)$ αποκαλείται δεσμευμένη πιθανότητα και αποτελεί την πιθανότητα η μεταβλητή X να πάρει την τιμή x_i, δεδομένου ότι η Y παίρνει την τιμή y_i. Η υπό συνθήκη εντροπία H(X|Y) είναι πάντα ίση ή μικρότερη από την εντροπία H(X).

Η Αμοιβαία Πληροφορία όπως προαναφέρθηκε σχετίζεται με την εντροπία. Ως Αμοιβαία Πληροφορία I(X,Y) μεταξύ δύο τυχαίων μεταβλητών X και Y ορίζεται η διαφορά της αβεβαιότητας που υπάρχει για το X μείον της αβεβαιότητας που υπάρχει για το X με δεδομένη την μεταβλητή Y και δίνεται από την παρακάτω σχέση [18]:

$$I(X,Y) = H(X) - H(X|Y) \quad (35)$$

Ουσιαστικά η παραπάνω σχέση μας δίνει το κέρδος της πληροφορίας (information gain) που υπάρχει για την μεταβλητή X. Όπως προαναφέρθηκε η υπό συνθήκη εντροπία H(X|Y) είναι πάντα ίση ή μικρότερη από την εντροπία H(X). Άρα η αμοιβαία πληροφορία είναι πάντα θετική ποσότητα. Επίσης η αμοιβαία πληροφορία έχει μηδενική τιμή όταν οι μεταβλητές X, Y είναι στατιστικά ανεξάρτητες, όπου η γνώση της μεταβλητής Y δεν μας δίνει καμία πληροφορία για την X. Το ίδιο ισχύει αντίστοιχα και στην αντίστροφη περίπτωση. Στην παρακάτω εικόνα παρουσιάζεται μία γραφική αναπαράσταση των μεγεθών των παραπάνω σχέσεων για δύο τυχαίες μεταβλητές A και B.



Εικόνα 4. *Γραφική απεικόνιση των σχέσεων μεταξύ της εντροπίας, της υπο συνθήκης εντροπίας και της αμοιβαίας πληροφορίας για δύο τυχαίες μεταβλητές A και B.*

3.10 Kolmogorov-Smirnov 2 Samples Test (KS 2Samples Test)

Το κριτήριο Kolmogorov-Smirnov για δύο δείγματα είναι ένα μη-παραμετρικό τεστ το οποίο συγκρίνει τις σωρευτικές κατανομές (cumulative distributions) δύο συνόλων δεδομένων. Συγκεκριμένα, μπορεί να χρησιμοποιηθεί για να ελέγξουμε αν δύο δείγματα αποτελούμενα από συνεχείς τιμές προέρχονται από την ίδια κατανομή [20] [21].

Ας θεωρήσουμε ότι X_1, X_2, …, X_N και x_1, x_2, ..., x_n είναι δύο τυχαία δείγματα (π.χ. οι τιμές ενός χαρακτηριστικού για τις δύο διαφορετικές κλάσεις) από τις κατανομές F_1 και F_2 αντίστοιχα και θέλουμε να ελέγξουμε την παρακάτω υπόθεση.

$$H_0: F_1 = F_2 \ \text{έναντι της} \ H_1: F_1 \neq F_2 \tag{36}$$

Για την σύγκριση των δύο παραπάνω κατανομών, το κριτήριο Kolmogorov-Smirnov χρησιμοποιεί τις σωρευτικές κατανομές των δύο δειγμάτων. Υποθέτοντας λοιπόν ότι οι συναρτήσεις σωρευτικής κατανομής των δύο παραπάνω δειγμάτων είναι $S_1(x)$ και $S_2(x)$, τότε η στατιστική τιμή D του κριτηρίου Kolmogorov-Smirnov είναι η μεγαλύτερη απόλυτη απόκλιση (διαφορά) μεταξύ των δύο σωρευτικών κατανομών, ανεξάρτητα από την κατεύθυνσης της διαφοράς [20] [21].

$$D_{N,n} = max\,|S_1(x) - S_2(x)| \tag{37}$$

Σύμφωνα με την θεωρία του κριτηρίου Kolmogorov-Smirnov για δύο δείγματα, όσο πιο μεγάλη είναι η στατιστική τιμή D, τα δύο δείγματα δείχνουν να μην προέρχονται από την ίδια κατανομή. Όσο πιο μικρή είναι η στατιστική τιμή D, τα δύο δείγματα δείχνουν να προέρχονται από την ίδια κατανομή [20].

Μία ακόμα στατιστική τιμή του κριτηρίου Kolmogorov-Smirnov είναι το p-value των δύο δειγμάτων. Το p-value απαντάει στην εξής ερώτηση: Εάν τα δύο τυχαία δείγματα προέρχονται από ίδιους πληθυσμούς, ποια είναι η πιθανότητα οι δύο σωρευτικές κατανομές να απέχουν όσο παρατηρείται; Πιο συγκεκριμένα, ποια είναι η πιθανότητα η τιμή της στατιστικής τιμής D του Komogorov-Smirnov να είναι τόσο μεγάλη ή μεγαλύτερη από αυτή που παρατηρήθηκε [21].

$$p - value = P\big(D_{N,n} > d|H_0\big) \tag{38}$$

Εάν η τιμή p-value είναι μικρή, καταλήγουμε στο συμπέρασμα ότι τα δύο δείγματα προέρχονται από πληθυσμούς με διαφορετικές κατανομές. Οι πληθυσμοί μπορεί να διαφέρουν ως προς τη μέση τιμή, τη μεταβλητότητα ή το σχήμα της κατανομής.

3.11 Relief & ReliefF

Η πρώτη έκδοση του αλγόριθμου Relief μπορεί να χρησιμοποιηθεί μόνο σε δυαδικά προβλήματα και μπορεί να εφαρμοστεί σε διακριτά και σε συνεχή χαρακτηριστικά. Βέβαια, υπάρχουν και οι επεκτάσεις του αρχικού αλγόριθμου Relief, οι οποίες δίνουν την δυνατότητα χρήσης του και σε προβλήματα με περισσότερες από δύο κλάσεις. [22]

Η βασική ιδέα της μεθόδου Relief έγκειται στον υπολογισμό χαρακτηριστικών βάσει της διαχωριστικής ικανότητας των τιμών τους πάνω σε κοντινά πρότυπα. Στον αλγόριθμο της Relief που δίνεται παρακάτω, αποτυπώνεται η βασική της ιδέα. Συγκεκριμένα, για ένα συγκεκριμένο πρότυπο, αναζητά ένα πρότυπο γείτονα από την ίδια κλάση τον οποίο ονομάζει ως 'πλησιέστερη επιτυχία' (nearest hit, H) και ένα πρότυπο γείτονα από την άλλη κλάση τον οποίο ονομάζει ως 'πλησιέστερη αποτυχία' (nearest miss, M). [22][23][24][25]

Input: for each training instance a vector of attribute values and the class value

Output: the vector W of estimations of the qualities of attributes

1. set all weights W[A] := 0.0;
2. for i := 1 to m do begin
3. randomly select an instance R_i ;
4. find nearest hit H and nearest miss M;
5. for A := 1 to a do
6. W[A] := W[A] − diff(A, R_i , H)/m + diff(A, R_i , M)/m;
7. end;

Ψευδοκώδικας Αλγόριθμου Relief

Η μετρική (βάρος) Relief ενός χαρακτηριστικού A (W[A]) είναι μία προσέγγιση της διαφοράς μεταξύ των παρακάτω πιθανοτήτων.

$$W[A] = P(different\ value\ of\ A \mid nearest\ instance\ from\ different\ class) - P(different\ value\ of\ A \mid nearest\ instance\ from\ same\ class) \quad (39)$$

Σύμφωνα με την παραπάνω σχέση, ένα καλό (σχετικό) χαρακτηριστικό, δηλαδή ένα χαρακτηριστικό με μεγάλη διακριτική ισχύ, πρέπει να λαμβάνει διαφορετικές τιμές μεταξύ προτύπων που ανήκουν σε διαφορετικές κλάσεις και πρέπει να έχει την ίδια τιμή μεταξύ προτύπων της ίδιας κλάσης.[22] Σκεπτικό αρκετά λογικό, αν σκεφτούμε ότι ένα

χαρακτηριστικό σχετικό με το εξεταζόμενο πρόβλημα πρέπει να διαθέτει διαφορετική συμπεριφορά (δηλαδή διαφορετική τιμή) για κάθε κλάση του προβλήματος.

Συνεπώς, η μεγάλη τιμή του W[A] μεταφράζεται ως ένα χαρακτηριστικό με ισχυρή διακριτική ικανότητα, εφόσον είτε θα υπάρχει μεγάλη πιθανότητα για ένα πρότυπο, η 'πλησιέστερη αποτυχία' του (nearest miss, M) να εμφανίζει διαφορετική τιμή για το χαρακτηριστικό A, είτε θα υπάρχει μικρή πιθανότητα η 'πλησιέστερη επιτυχία' του (nearest hit, H) να εμφανίζει διαφορετική τιμή για το χαρακτηριστικό A. [22]

Παρατηρώντας τον αλγόριθμο της Relief, θεωρούμε ότι έχουμε ένα σύνολο προτύπων και επιλέγουμε m φορές από αυτό πρότυπα με επανάθεση (δηλαδή ένα πρότυπο κάθε φορά, γραμμή αλγόριθμου 2). Για κάθε χαρακτηριστικό A, η πιθανότητα να έχει διαφορετική τιμή για την 'πλησιέστερη αποτυχία' του τρέχοντος προτύπου ισούται με τον αριθμό των φορών που παρατηρήθηκε κάποια διαφορά στην τιμή του χαρακτηριστικού A ανάμεσα στο τρέχον πρότυπο και στην 'πλησιέστερη αποτυχία' του προς το συνολικό αριθμό m των φορών που επαναλαμβάνεται ο αλγόριθμος. Αντίστοιχα, για κάθε χαρακτηριστικό A, η πιθανότητα να διαθέτει διαφορετική τιμή για την 'πλησιέστερη επιτυχία' του τρέχοντος προτύπου ισούται με τον αριθμό των φορών που παρατηρήθηκε κάποια διαφορά στην τιμή του χαρακτηριστικού A ανάμεσα στο τρέχον πρότυπο και στην 'πλησιέστερη επιτυχία' του προς το συνολικό αριθμό m των φορών που επαναλήφθηκε ο αλγόριθμος. Ουσιαστικά στις παραπάνω περιπτώσεις υπολογίζεται ο μέσος όρος των διαφορών στις τιμές του χαρακτηριστικού A ανάμεσα σε δύο πρότυπα στο σύνολο των m επαναλήψεων.

Ορίζεται ότι η διαφορά diff στις τιμές ενός διακριτού χαρακτηριστικού A μεταξύ δύο προτύπων I1 και I2, είναι 1 αν είναι διαφορετικές οι τιμές του χαρακτηριστικού και 0 αν οι τιμές είναι ίδιες.[22][23]

$$diff(A, I_1, I_2) = \begin{cases} 0 : value(A, I_1) = value(A, I_2) \\ 1 : otherwise \end{cases} \quad (40)$$

Για συνεχείς χαρακτηριστικά, η πραγματική διαφορά κανονικοποιείται στο διάστημα [0, 1]. Στην περίπτωση λοιπόν των συνεχών χαρακτηριστικών, ορίζεται ως διαφορά diff, η διαφορά στις τιμές του χαρακτηριστικού A ανάμεσα σε δύο πρότυπα I1 και I2 προς το εύρος τιμών του συγκεκριμένου χαρακτηριστικού στο σύνολο των προτύπων. [22][23]

$$diff(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{\max(A) - \min(A)} \quad (41)$$

όπου min(A) και max(A) αποτελούν αντίστοιχα την ελάχιστη και την μέγιστη τιμή του χαρακτηριστικού A στο σύνολο των προτύπων. Όσο το αποτέλεσμα της diff πλησιάζει το 1, τότε η διαφορά μεγαλώνει ενώ αντίστοιχα όσο το αποτέλεσμα diff πλησιάζει το 0, η διαφορά μικραίνει. Η διαφορά diff χρησιμοποιείται επίσης και για την εύρεση των πλησιέστερων γειτόνων όπου η συνολική απόσταση είναι το άθροισμα των διαφορών όλων των χαρακτηριστικών.[22]

Συνεχίζοντας από το σημείο όπου ορίστηκε ως W[A] η διαφορά των πιθανοτήτων, αναφέροντας στην συνέχεια ότι είναι ο μέσος όρος των διαφορών στις τιμές του χαρακτηριστικού A ανάμεσα σε δύο πρότυπα στο σύνολο των m επαναλήψεων, καταλήγουμε στις δύο παρακάτω σχέσεις.

$$P(different\ value\ of\ A|nearest\ instance\ from\ different\ class) = \frac{\sum_{i=1}^{m} diff(A,R,M)}{m} \quad (42)$$

$$P(different\ value\ of\ A|nearest\ instance\ from\ same\ class) = \frac{\sum_{i=1}^{m} diff(A,R,H)}{m} \quad (43)$$

Όπου R το τυχαία επιλεγμένο πρότυπο, M η ‘πλησιέστερη αποτυχία’ (42), H η ‘πλησιέστερη επιτυχία’ (43) και m ο αριθμός των επαναλήψεων.

Μετά το τέλος των m επαναλήψεων του αλγόριθμου, όπου για κάθε χαρακτηριστικό (για κάθε επανάληψη) που επιλέγεται εκτιμιούνται τα βάρη των χαρακτηριστικών, ο αλγόριθμος μας επιστρέφει ως έξοδο το διάνυσμα **W** των εκτιμήσεων της διακριτικής ισχύς (ποιότητας) των χαρακτηριστικών. Τέλος, είναι κατανοητό ότι ο αριθμός των επαναλήψεων m δεν πρέπει να ξεπερνά τον αριθμό των προτύπων που διαθέτουμε αλλά ούτε και να είναι πολύ μικρότερος καθώς όσο ο αριθμός των επαναλήψεων m μεγαλώνει τόσο καλύτερα προσεγγίζονται οι πιθανότητες.[22]

Η πρακτική που ακολουθεί ο αλγόριθμος Relief, επιλέγοντας κάθε φορά για ένα συγκεκριμένο πρότυπο έναν πλησιέστερο γείτονα (πρότυπο) από την ίδια κλάση και έναν πλησιέστερο γείτονα (πρότυπο) από την άλλη κλάση μπορεί να οδηγήσει σε αναξιόπιστα αποτελέσματα. Συγκεκριμένα, στην περίπτωση που στο σύνολο δεδομένων υπάρχουν περιττά (redundant) και θορυβώδη (noisy) δεδομένα (με την έννοια ότι είναι πιθανόν να “πολώσουν” την αναζήτηση). Για αυτόν τον λόγο, δημιουργήθηκε η ανάγκη επέκτασης του αλγορίθμου. Μία από τις επεκτάσεις του αλγορίθμου Relief είναι ο ReliefF [22], ο οποίος δίνεται παρακάτω.

Input: for each training instance a vector of attribute values and the class value

Output: the vector W of estimations of the qualities of attributes

1. set all weights W[A] := 0.0;

2. for i := 1 **to** m **do begin**

3. randomly select an instance R_i ;

4. find k nearest hits H_j ;

5. **for** each class C ≠ class(R_i) **do**

6. from class C find k nearest misses M_j (C);

7. for A := 1 **to** a **do**

8. $W[A] := W[A] - \sum_{j=1}^{k} \frac{diff(A, R_i, H_j)}{m*k} +$

9. $\sum_{C \neq class(R_i)} \left[\frac{P(C)}{1-P(class(R_i))} \sum_{j=1}^{k} diff\left(A, R_i, M_j(C)\right) \right] / (m*k);$

10. end;

Ψευδοκώδικας Αλγόριθμου ReliefF

Ο αλγόριθμος ReliefF, αναζητά τις k πλησιέστερες ‘επιτυχίες’ και ‘αποτυχίες’ και υπολογίζει τον μέσο όρο της συνεισφοράς τους. Συγκεκριμένα, για κάθε πρότυπο, αναζητά k πρότυπα γείτονες από την ίδια κλάση τα οποία τα ονομάζει ως ‘πλησιέστερες επιτυχίες’ (nearest hits, Hj) και k πρότυπα γείτονες, από τα οποία το κάθε ένα προέρχεται από διαφορετική κλάση και τα ονομάζει ως ‘πλησιέστερες αποτυχίες’ (nearest misses, Mj(C)). Σύμφωνα με την βιβλιογραφία, μία καλή επιλογή του αριθμού k των πλησιέστερων γειτόνων είναι το 10. [22]

Επομένως, η μετρική (βάρος) ReliefF, W[A], είναι μία προσέγγιση της διαφοράς μεταξύ των παρακάτω πιθανοτήτων (αντί της (39) που ισχύει στην Relief).

$$W[A] = P(different\ value\ of\ A \mid different\ class) - P(different\ value\ of\ A \mid same\ class) \quad (44)$$

Δηλαδή εντοπίζονται τα k πλησιέστερα πρότυπα που ανήκουν σε διαφορετική κλάση, έπειτα τα k πλησιέστερα πρότυπα που ανήκουν στην ίδια κλάση και εξετάζονται στην συνέχεια οι τιμές του χαρακτηριστικού A σε αυτά.

Ο μέσος όρος της συνεισφοράς των k ‘πλησιέστερων επιτυχιών’ στο βάρος του χαρακτηριστικού A υπολογίζεται από τον τύπο

$$\frac{\sum_{j=1}^{k} diff\ (A, R_i, H_i)}{k} \tag{45}$$

και ο μέσος όρος αυτής της συνεισφοράς στο σύνολο των m επαναλήψεων που ισούται με την πιθανότητα *P(different value of A | same class)*, δίνεται από τον παρακάτω τύπο

$$\frac{\sum_{j=1}^{k} diff\ (A, R_i, H_i)}{m \cdot k} \tag{46}$$

Αντίστοιχα, μετά από υπολογισμούς σύμφωνα με την βιβλιογραφία [22], η πιθανότητα *P(different value of A | different class)* ισούται με

$$\sum_{C \neq class(R_i)} \left[\frac{P(C)}{1 - P(class(R_i))} \cdot \frac{\sum_{j=1}^{k} diff(A, R_i, H_j)}{m \cdot k} \right] \tag{47}$$

Όπως και ο πρωτότυπος Relief, ο ReliefF μετά το τέλος των m επαναλήψεων του αλγόριθμου, όπου για κάθε χαρακτηριστικό (για κάθε επανάληψη) που επιλέγεται εκτιμούνται τα βάρη των χαρακτηριστικών, μας επιστρέφει ως έξοδο το διάνυσμα **W** των εκτιμήσεων της διακριτικής ισχύς (ποιότητας) των χαρακτηριστικών. Τέλος, ο αλγόριθμος ReliefF, εκτός από την ικανότητα του να διαχειριστεί θορυβώδη δεδομένα όπως προαναφέρθηκε, παρουσιάζει μεγαλύτερη ευρωστία, μπορεί να διαχειριστεί ελλιπή δεδομένα και να εφαρμοστεί, όπως περιεγράφηκε, σε προβλήματα με περισσότερες από δύο κλάσεις.

3.12 Minimum Redundancy Maximum Relevance (mRMR)

Το κριτήριο του ελάχιστου πλεονασμού - μέγιστης συσχέτισης (mRMR) βασίζεται πάνω στην αμοιβαία πληροφορία. Έχει χρησιμοποιηθεί σε αρκετά προβλήματα βιοπληροφορικής με σκοπό την επιλογή των γονιδίων που σχετίζονται με την εκδήλωση διαφόρων νόσων [26]. Ο στόχος της μεθόδου mRMR διαχωρίζεται σε δύο επιμέρους συνθήκες οι οποίες πρέπει να ικανοποιούνται ταυτόχρονα από το υποσύνολο των χαρακτηριστικών. Οι συνθήκες αυτές είναι οι εξής [26][27]:

i. Τα χαρακτηριστικά του υποσυνόλου πρέπει να έχουν όσο το δυνατό μεγαλύτερη συνάφεια με την κλάση (μέγιστη συσχέτιση - maximum relevance).
ii. Τα χαρακτηριστικά του υποσυνόλου πρέπει να είναι όσο το δυνατόν ανόμοια μεταξύ τους (ελάχιστος πλεονασμός - minimum redudancy).

Πριν αναλύσουμε τις δύο παραπάνω συνθήκες του mRMR, πρώτα θα εξετάσουμε το κριτήριο μέγιστης εξάρτησης, το οποίο χρησιμοποιείται για την επιλογή του υποσυνόλου των χαρακτηριστικών με την όσο δυνατή μεγαλύτερη συνάφεια με την κλάση. Το κριτήριο της μέγιστης εξάρτησης αποτελεί την 'βάση του προβλήματος' από την οποία προήλθε η μέθοδος mRMR.

Σύμφωνα λοιπόν με τους όρους από την θεωρία της πληροφορίας, το ιδανικό υποσύνολο χαρακτηριστικών $S = \{X_1, X_2, \ldots, X_N\}$ είναι αυτό που έχει την μεγαλύτερη αμοιβαία πληροφορία I(S,Y) με την κλάση Y που εξετάζεται. Δηλαδή το υποσύνολο από το οποίο η κλάση έχει την μεγαλύτερη εξάρτηση. Οπότε το υποσύνολο που θα επιλεγεί θα πρέπει να μεγιστοποιεί την παρακάτω ποσότητα της αμοιβαίας πληροφορίας:

$$I(S,Y) = \sum_{s\in S}\sum_{y\in Y} p(S=s,Y=y)\log\frac{p(S=s,Y=y)}{p(S=s)p(Y=y)} \tag{48}$$

όπου p(S) η από κοινού κατανομή των $X_1, X_2, \ldots, X_N$ και p(S,Y) η από κοινού κατανομή των $X_1, X_2, \ldots, X_N$, Y. [27]

Αν και το κριτήριο της μέγιστης εξάρτησης διαθέτει μεγάλη θεωρητική αξία, δυστυχώς είναι δύσκολο να εκτιμηθούν με αξιόπιστο τρόπο οι ποσότητες p(s) και p(s,y) και επομένως και η αμοιβαία πληροφορία I(S,Y). Ο λόγος της δυσκολίας που υπάρχει για τον προσδιορισμό τον από κοινού κατανομών p(s) και p(s,y) είναι γιατί ο αριθμός παραδειγμάτων που απαιτείται για την εκτίμηση τους αυξάνεται εκθετικά ως προς τον αριθμό των χαρακτηριστικών του υποσυνόλου S.

Υποθέτουμε για παράδειγμα ότι ζητείται η εκτίμηση της κατανομής p(s) και ότι τα $X_1, X_2, \ldots, X_N$ είναι διακριτά χαρακτηριστικά με δυαδικό πεδίο ορισμού. Αυτό σημαίνει ότι συνολικά υπάρχουν 2^N διαφορετικές τιμές που μπορούν να περιέχονται στο S. Ως επακόλουθο χρειάζεται ο υπολογισμός 2^N πιθανοτήτων. Αυτός ο αριθμός θα είναι πολύ μεγαλύτερος από το πλήθος των διαθέσιμων παραδειγμάτων εκτός από την περίπτωση που το N είναι πολύ μικρό.

Επίσης ένα ακόμα σημαντικό μειονέκτημα του κριτηρίου της μέγιστης εξάρτησης αποτελεί το μεγάλο χρονικό διάστημα που χρειάζεται για να γίνουν όλοι οι υπολογισμοί. Το αποτέλεσμα αυτής της χρονοβόρας διαδικασίας τον καθιστά ως ένα από τα ιδιαίτερα αργά κριτήρια.

Λόγω των παραπάνω δυσκολιών που υπάρχουν στον υπολογισμό της αμοιβαία πληροφορίας I(S,Y), χρησιμοποιείται μία διαφορετική προσέγγιση ώστε να βρεθεί και να επιλεγεί το υποσύνολο S από το οποίο η κλάση θα έχει την μεγαλύτερη δυνατή εξάρτηση. Η

μονοπαραγοντική προσέγγιση στο παραπάνω πρόβλημα έγκειται στον υπολογισμό της εξάρτησης της κλάσης με το κάθε χαρακτηριστικό ξεχωριστά. Στην συνέχεια επιλέγονται τα Μ χαρακτηριστικά από τα οποία υπάρχει μεγαλύτερη εξάρτηση με την κλάση. Οπότε η παραπάνω προσέγγιση για τον προσδιορισμό του υποσυνόλου S μετατρέπεται σε μία πιο εύκολη προσέγγιση για την οποία ισχύει [27]:

$$S = \underset{S \subset F, |S|=M}{\arg\max} \left\{ \sum_{X_i \in S} I(X_i, Y) \right\} \tag{49}$$

Το μειονέκτημα της παραπάνω μονοπαραγοντικής προσέγγισης είναι ότι στα Μ χαρακτηριστικά που επιλέγονται στο υποσύνολο S, επιλέγονται και πολλά περιττά χαρακτηριστικά. Δηλαδή αν αρχικά ένα χαρακτηριστικό επιλεχθεί επειδή έχει μεγάλη αμοιβαία πληροφορία με την κλάση, αντίστοιχα επιλέγονται και άλλα χαρακτηριστικά τα οποία είναι όμοια με αυτό και άρα έχουν και αυτά μεγάλη αμοιβαία πληροφορία. Η επιλογή όμως ενός υποσυνόλου με αρκετά όμοια χαρακτηριστικά έχει ως αποτέλεσμα την προσφορά πολύ λιγότερης πληροφορίας για την κλάση σε σχέση με την επιλογή ενός υποσυνόλου χαρακτηριστικών που δεν περιέχει όμοια χαρακτηριστικά. Ουσιαστικά δεν προσφέρει κανένα κέρδος σε πληροφορία η γνώση των υπόλοιπων όμοιων χαρακτηριστικών.

Οπότε η επιλογή του καλύτερου υποσυνόλου S μπορεί να προκύψει αν επιλέγονται χαρακτηριστικά που έχουν όσο το δυνατόν μεγαλύτερη εξάρτηση με την κλάση αλλά ταυτόχρονα είναι μεταξύ τους όσο το δυνατόν ανόμοια. Πάνω σε αυτήν την ιδέα βασίζονται και οι δύο συνθήκες που πρέπει να ικανοποιούνται ταυτόχρονα για την επιλογή του βέλτιστου υποσυνόλου χαρακτηριστικών σύμφωνα με την μέθοδο mRMR.

Για την ικανοποίηση της πρώτης συνθήκης, γίνεται χρήση της μονοπαραγοντικής προσέγγισης που προαναφέρθηκε. Οπότε αν το υποσύνολο των χαρακτηριστικών είναι S με X_1, X_2, …, X_N τα χαρακτηριστικά και η κλάση Y, τότε η μέτρηση της συνάφειας μεταξύ των χαρακτηριστικών του υποσυνόλου S και της κλάσης Y προέρχεται από [27]:

$$\max D(S, Y), \;\; D = \frac{1}{|S|} \sum_{x_i \in S} I(X_i, Y) \tag{50}$$

Για την ελαχιστοποίηση των όμοιων χαρακτηριστικών, γίνεται χρήση της αμοιβαίας πληροφορίας. Συγκεκριμένα για την μέτρηση της ομοιότητας μεταξύ δύο χαρακτηριστικών

X_i και X_j υπολογίζεται η αμοιβαία πληροφορία $I(X_i, X_j)$, ελαχιστοποιώντας την ποσότητα minR(S) η οποία δίνεται από την παρακάτω σχέση [27]:

$$\min R(S), \;\; R = \frac{1}{|S|^2} \sum_{x_i, x_{j,} \in S} I(X_i, X_j) \tag{51}$$

Ο συνδυασμός των δύο παραπάνω μετρήσεων αποτελεί ουσιαστικά την μέθοδο mRMR. Όπως προαναφέρθηκε, ο στόχος της είναι η επιλογή ενός υποσυνόλου Φ(D,R) το οποίο να μεγιστοποιεί την συνάφεια D(S) και ταυτόχρονα να ελαχιστοποιεί την περιττή πληροφορία R(S) [26][27]. Για την επιλογή λοιπόν του βέλτιστου υποσυνόλου συνδυάζονται τα δύο κριτήρια σε μία συνάρτηση αξιολόγησης η οποία είναι η παρακάτω:

$$\Phi = D - R \tag{52}$$

Μία ακόμη συνάρτηση αξιολόγησης που προτάθηκε [26] και συνδυάζει τα κριτήρια D(S) και R(S) και δίνει μεγαλύτερη έμφαση στην μείωση της περιττής πληροφορίας είναι η παρακάτω:

$$\Phi = \frac{D}{R} \tag{53}$$

Προκειμένου η διαδικασία αναζήτησης να γίνει υπολογιστικά εφικτή, αρχικά επιλέγεται το χαρακτηριστικό που έχει την μεγαλύτερη συνάφεια με την κλάση. Δηλαδή το χαρακτηριστικό που έχει την μεγαλύτερη αμοιβαία πληροφορία με την κλάση. Στην συνέχεια σε κάθε επανάληψη επιλέγεται ένα χαρακτηριστικό που έχει μεγάλη συνάφεια με την κλάση αλλά ταυτόχρονα και μικρή ομοιότητα με τα χαρακτηριστικά που ήδη έχουν επιλεγεί.

Τροποποιώντας την συνάρτηση αξιολόγησης Φ = D – R (52*)* που αξιολογεί υποσύνολα χαρακτηριστικών και υποθέτοντας ότι F είναι το σύνολο όλων των αρχικών χαρακτηριστικών, καταλήγουμε στην παρακάτω συνάρτηση αξιολόγησης η οποία αξιολογεί και επιλέγει μεμονωμένα χαρακτηριστικά.

$$X' = \arg\max_{X \subset F - S} \left\{ I(X_i, Y) - \frac{1}{|S|} \sum_{X_i \in S} I(X_i, X) \right\} \tag{54}$$

Τροποποιώντας αντίστοιχα και την συνάρτηση αξιολόγησης Φ = D / R (53) καταλήγουμε στην παρακάτω συνάρτηση αξιολόγησης μεμονωμένων χαρακτηριστικών.

$$X' = \arg\max_{X \subset F-S} \left\{ \frac{I(X_i, Y)}{\frac{1}{|S|} \sum_{X_i \in S} I(X_i, X)} \right\} \tag{55}$$

Η μέθοδος mRMR μπορεί να χρησιμοποιηθεί και σε συνεχή χαρακτηριστικά. Ο υπολογισμός της αμοιβαίας πληροφορίας ανάμεσα σε δύο μεταβλητές X και Y όταν είναι συνεχείς, αντιμετωπίζει δυσκολίες. Με την χρήση όμως άλλων κριτηρίων συσχέτισης που μπορούν να χειριστούν συνεχή χαρακτηριστικά μπορεί να αποφύγει η χρήση της αμοιβαίας πληροφορίας [26].

Συγκεκριμένα, για την επίτευξη της μέγιστης εξάρτησης μεταξύ των χαρακτηριστικών και των κλάσεων, μπορεί να χρησιμοποιηθεί το κριτήριο Fisher όταν υπάρχουν δύο κλάσεις. Στην περίπτωση ύπαρξης περισσότερων των δύο κλάσεων μπορεί να γίνει χρήση του κριτηρίου F-test. Για την ικανοποίηση του δεύτερου κριτηρίου του mRMR, δηλαδή της ελαχιστοποίησης της περιττής πληροφορίας, μπορεί να γίνει χρήση του συντελεστή συσχέτισης Pearson. Το κριτήριο F-test δεν μπορεί να χρησιμοποιηθεί στην προκειμένη περίπτωση καθώς προϋποθέτει την ύπαρξη μίας διακριτής μεταβλητής μεταξύ των δύο [26].

Μία επιπλέον προσέγγιση για τον χειρισμό των συνεχών χαρακτηριστικών, είναι η διακριτοποίηση τους. Η μετατροπή των συνεχών τιμών σε διακριτές τιμές φαίνεται να ταιριάζει στα προβλήματα βιοπληροφορικής καθώς αποφέρει βελτίωση της απόδοσης του ταξινομητή όταν του παρέχονται χαρακτηριστικά με διακριτές τιμές [26]. Ο τρόπος της διακριτοποίησης των τιμών των χαρακτηριστικών μπορεί να επιτευχθεί με διάφορες μεθόδους όπως οι δύο μέθοδοι που αναφέρθηκαν σε προηγούμενη ενότητα.

4

Ανάλυση της Συσχέτισης Γονιδίων με τον Καρκίνο του Ήπατος με χρήση μεθόδων Μηχανικής Μάθησης

Σε αυτό το κεφάλαιο αρχικά γίνεται μία λεπτομερής περιγραφή του συνόλου δεδομένων που χρησιμοποιήθηκε στην παρούσα εργασία και αφορά τον καρκίνο του ήπατος. Στην συνέχεια, στο υποκεφάλαιο 4.2 πραγματοποιείται περιγραφή της εφαρμογής των μεθόδων και των κριτηρίων που χρησιμοποιήθηκαν για τον εντοπισμό των συσχετίσεων μεταξύ των χαρακτηριστικών (γονιδίων) και του καρκίνου του ήπατος (αξιολόγηση των γονιδίων) με σκοπό την επιλογή των υποψήφιων υποσυνόλων για την κατασκευή ενός αποδοτικού ταξινομητή. Συγκεκριμένα σε κάθε ενότητα του 4.2 περιγράφεται η εφαρμογή μίας μεθόδου μείωσης διαστάσεων από τις συνολικά 6 που εφαρμόσαμε. Επιπλέον, περιγράφεται η διαδικασία που ακολουθήσαμε για την δημιουργία επιπλέον υποψήφιων υποσυνόλων από όλους τους δυνατούς συνδυασμούς μεταξύ των 6 μεθόδων μείωσης διαστάσεων.

Στο υποκεφάλαιο 4.3 γίνεται περιγραφή των παραμέτρων της εφαρμογής του SVM με γραμμικό πυρήνα (SVM Linear), του SVM με πυρήνα RBF (SVM RBF) και της τεχνικής της αναζήτησης πλέγματος (Grid Search) με χρήση των υποψηφίων υποσυνόλων χαρακτηριστικών (γονιδίων).

Η γλώσσα που επιλέχθηκε για την υλοποίηση των παραπάνω μεθόδων μηχανικής μάθησης είναι η Python. Συγκεκριμένα σε όλες τις εφαρμογές χρησιμοποιήσαμε την Python 2.7, εκτός από την εφαρμογή της μεθόδου mRMR, όπου λόγω απαιτήσεων του αντίστοιχου πακέτου εγκαταστήσαμε την Python 3.

Οι λόγοι που μας οδήγησαν σε αυτή την επιλογή είναι ότι υπάρχει πληθώρα πολύ καλών και ενημερωμένων βιβλιοθηκών για την εφαρμογή των μεθόδων μηχανικής μάθησης (Scikit-learn), για την στατιστική ανάλυση (Analytics) και για επιστημονικούς υπολογισμούς (Scientific Computing) (Numpy, Scipy, Pandas) και για την οπτικοποίηση δεδομένων μέσω διαγραμμάτων (matplotlib). Επιπλέον η Python αποτελεί μία σύγχρονη γλώσσα, η οποία προσφέρει ικανοποιητική ταχύτητα και με την οποία μπορείς να εκτελέσεις ποικίλες λειτουργίες σε επαρκή βαθμό.

4.1 Περιγραφή Συνόλου Δεδομένων

Το σύνολο δεδομένων που χρησιμοποιήσαμε στην παρούσα εργασία αφορά τον καρκίνο του ήπατος και προέρχεται από την βάση δεδομένων του TCGA (The Cancer Genome Atlas). Η βάση δεδομένων του TCGA αποτελεί μια συνεργασία μεταξύ του Εθνικού Ινστιτούτου Καρκίνου (National Cancer Institute, NCI) και του Εθνικού Ινστιτούτου Ερευνών Ανθρώπινου Γονιδιώματος (National Human Genome Research Institute, NHGRI) όπου έχουν δημιουργήσει ολοκληρωμένους πολυδιάστατους χάρτες των βασικών γονιδιωματικών αλλαγών σε 33 τύπους καρκίνου. Τα σύνολα δεδομένων TCGA, τα οποία περιλαμβάνουν περισσότερα από δύο petabytes γονιδιωματικών δεδομένων, έχουν δημοσιοποιηθεί και βοηθάνε την ερευνητική κοινότητα του καρκίνου να βελτιώσει την πρόληψη, τη διάγνωση και τη θεραπεία του καρκίνου.[28]

Συγκεκριμένα, το προς ανάλυση σύνολο δεδομένων μας δόθηκε από το Ινστιτούτο Εφαρμοσμένων Βιοεπιστημών (INAB)[2] του Εθνικού Κέντρου Έρευνας και Τεχνολογικής Ανάπτυξης (ΕΚΕΤΑ)[3], αφού πρώτα υπέστη την απαραίτητη προεπεξεργασία (filtering, normalization, transformation), προκειμένου να είναι κατάλληλο για την εφαρμογή μεθόδων και αλγορίθμων μηχανικής μάθησης.

Τα δεδομένα που αναλύσαμε περιέχονται σε δύο αρχεία δεδομένων. Το ένα αρχείο (Liver_Cancer_RNA_seq_data.txt**)** περιέχει έναν πίνακα του οποίου οι στήλες αποτελούν τα γονίδια (χαρακτηριστικά) και οι γραμμές αποτελούν τα δείγματα (πρότυπα) ασθενών με ηπατοκυτταρικό καρκίνωμα. Οι τιμές των γονιδίων αποτελούν τιμές έκφρασης μετασχηματισμένες από αλληλούχιση RNA (RNA-seq). Η αλληλούχιση RNA αποτελεί μία

[2] Ιστότοπος του Ινστιτούτου Εφαρμοσμένων Βιοεπιστημών (INAB): www2.inab.certh.gr

[3] Ιστότοπος Εθνικού Κέντρου Έρευνας και Τεχνολογικής Ανάπτυξης (ΕΚΕΤΑ): www.certh.gr

μέθοδο ποσοτικοποίησης η οποία εφαρμόζεται σε βιολογικά δεδομένα για να διερευνηθεί η διαφορική έκφραση γονιδίων. Ο συνολικός αριθμός των δειγμάτων (προτύπων) είναι 317 και ο αριθμός των γονιδίων (χαρακτηριστικών) είναι 12604. Οι τιμές του συνόλου των γονιδίων (χαρακτηριστικών) περιγράφονται σε ένα συνεχές διάστημα τιμών με μικρότερη τιμή την 3.92734020987 και μεγαλύτερη την 23.3670292757. Δηλαδή, αν G είναι το σύνολο των τιμών των ανεξάρτητων μεταβλητών x (χαρακτηριστικών) σε όλα τα δείγματα (πρότυπα), τότε G = [3.92734020987, 23.3670292757].

Το δεύτερο αρχείο δεδομένων (Clinicopathological_data.txt**)** αποτελείται από 5 στήλες και 317 γραμμές. Οι 317 γραμμές αντιστοιχούν στα δείγματα (πρότυπα) των ασθενών με ηπατοκυτταρικό καρκίνωμα από το πρώτο αρχείο δεδομένων. Η πρώτη στήλη αποτελεί ένα μοναδικό αλφαριθμητικό για κάθε δείγμα (index). Οι υπόλοιπες 4 στήλες περιγράφονται από τις εξής επικεφαλίδες: event, time, Grades και Stages. Στην παρούσα εργασία θα μας απασχολήσουν οι στήλες Grades και Stages. Η κάθε μία από αυτές αποτελεί μία δίτιμη ποιοτική μεταβλητή.

Οι τιμές της στήλης Grades είναι «Early Grade» και «Late Grade». Οπότε το ένα από τα δύο προβλήματα ταξινόμησης που μελετήσαμε στην παρούσα εργασία είναι ένα πρόβλημα ταξινόμησης μεταξύ δύο κλάσεων. Συγκεκριμένα μεταξύ της κλάσης «Early Grade», για την οποία στα πειράματα μας ορίστηκε ως ετικέτα κλάσης η τιμή 0 (Early Grade = 0) και της κλάσης «Late Grade» για την οποία ορίστηκε ως ετικέτα κλάσης η τιμή 1 (Late Grade = 1).

Η στήλη Grade μας περιγράφει τον βαθμό του καρκίνου. Δηλαδή, το κατά πόσο ο καρκίνος που έχει διαγνωστεί σε έναν ασθενή, σε μία δεδομένη χρονική στιγμή, μπορεί εύκολα να θεραπευτεί (αντιμετωπιστεί). Σε αυτό το σημείο αξίζει να σημειωθεί ότι οι τιμές που αρχικά έπαιρνε η ποιοτική μεταβλητή της στήλης Grades ήταν τέσσερις (4). Δηλαδή, οι ασθενείς του δείγματος ήταν κατηγοριοποιημένοι σε τέσσερις (4) κατηγορίες (κλάσεις) οι οποίες ήταν οι εξής [29]:

1) Πολύ καλά διαφοροποιημένο ηπατοκυτταρικό καρκίνωμα (Very well differentiated hepatocellular carcinoma)
2) Καλά διαφοροποιημένο ηπατοκυτταρικό καρκίνωμα (Well differentiated hepatocellular carcinoma)
3) Μερικώς διαφοροποιημένο ηπατοκυτταρικό καρκίνωμα (Moderately differentiated hepatocellular carcinoma)
4) Κακώς διαφοροποιημένο ηπατοκυτταρικό καρκίνωμα (Poorly differentiated hepatocellular carcinoma)

Οι δύο κατηγορίες (κλάσεις) της στήλης Grades προήλθαν μετά από κατάλληλη επεξεργασία που πραγματοποίησε το INEB για πρακτικούς λόγους διευκόλυνσης και για την

εξισορρόπηση των προτύπων μεταξύ των κλάσεων. Συγκεκριμένα η κλάση «Early Grade» προήλθε από την συγχώνευση της 1ης και της 2ης κλάσης (κατηγορίας) και η κλάση «Late Grade» προήλθε από την ένωση της 3ης και της 4ης κλάσης (κατηγορίας). Έτσι, η κλάση «Early Grade» αντιστοιχεί συνολικά σε 195 πρότυπα (ασθενείς) και η κλάση «Late Grade» αντιστοιχεί συνολικά σε 122 πρότυπα (ασθενείς).

Η στήλη Stage μας περιγράφει το πόσο «προχωρημένος» είναι ο καρκίνος που διαγνώστηκε σε μία δεδομένη χρονική στιγμή. Με την έννοια «προχωρημένος», νοείται το μέγεθος του καρκίνου, το πόσο έχει εξαπλωθεί στο ήπαρ, αν ο καρκίνος έχει κάνει μετάσταση, το στάδιο της μετάστασης και το εύρος αυτής. Αντίστοιχα και οι τιμές που αρχικά έπαιρνε η ποιοτική μεταβλητή της στήλης Stages ήταν τέσσερις (4). Συγκεκριμένα, οι ασθενείς του δείγματος ήταν κατηγοριοποιημένοι σε τέσσερις (4) κατηγορίες (κλάσεις), Stage I, Stage II, Stage III και Stage IV, βάσει μετρήσεων οι οποίες αφορούν τα παραπάνω «χαρακτηριστικά» που αναφέρθηκαν [30].

Οι δύο κατηγορίες (κλάσεις) της στήλης Stages προήλθαν και αυτές μετά από κατάλληλη επεξεργασία που πραγματοποίησε το INEB για τους λόγους που προαναφέρθηκαν και στα Grades. Οπότε, στην στήλη Stages, η κλάση «Early Stage» προήλθε από την συγχώνευση της 1ης και της 2ης κλάσης (κατηγορίας) και η κλάση «Late Stage» προήλθε από την ένωση της 3ης και της 4ης κλάσης (κατηγορίας). Η κλάση «Early Stage» αντιστοιχεί συνολικά σε 234 πρότυπα (ασθενείς) και η κλάση «Late Stage» αντιστοιχεί συνολικά σε 83 πρότυπα (ασθενείς).

Σε αυτό το σημείο είναι σημαντικό να αναφερθεί ότι κατά την εφαρμογή των μεθόδων μηχανικής μάθησης στο σύνολο δεδομένων, η αρίθμηση των χαρακτηριστικών (γονιδίων) ακολουθεί την αρίθμηση θέσεων ενός πίνακα (δείκτης θέσης), όπου δοσμένου ενός πίνακα Ν θέσεων (Ν χαρακτηριστικών), ο πίνακας αριθμείται ξεκινώντας από το 0 και όχι από το 1, με τον δείκτη θέσης του τελευταίου χαρακτηριστικού να είναι Ν-1. Συνεπώς, όταν για παράδειγμα αναφερόμαστε στο γονίδιο 870, νοείται το γονίδιο 871 του συνόλου δεδομένων που μας δόθηκε στο αρχείο Liver_Cancer_RNA_seq_data.txt.

Αριθμός Χαρακτ/κών (γονίδια)	**Αριθμός Προτύπων (ασθενείς)**	**Πρόβλ. Ταξ/σης**	**Κλάση Ετικέτα Κλάσης**	**Αριθμός Προτύπων ανά Κλάση**
12604	317	**Grades**	Early Grade / 0	195
			Late Grade / 1	122
12604	317	**Stages**	Early Stage / 0	234
			Late Stage / 1	83

Πίνακας 6. *Σύνολο δεδομένων που αφορά τον καρκίνο του ήπατος*

Συνοψίζοντας, στον πίνακα 6 παρουσιάζονται τα 2 προβλήματα δυαδικής ταξινόμησης που μελετήσαμε στην παρούσα εργασία, όπου για το κάθε ένα ορίσαμε ως ετικέτες κλάσης τις τιμές 0 και 1.

4.2 Κριτήρια Αξιολόγησης Χαρακτηριστικών για Μείωση Διαστάσεων Συνόλου Δεδομένων

4.2.1 Principal Component Analysis (PCA)

Η μέθοδος PCA αν και αποτελεί μέθοδο μείωσης των χαρακτηριστικών (διαστάσεων) μέσω της εξαγωγής νέων χαρακτηριστικών, στην παρούσα εργασία την χρησιμοποιήσαμε για την επιλογή υποσυνόλου χαρακτηριστικών. Επιλέξαμε μόνο την πρώτη κύρια συνιστώσα και αξιοποιήσαμε τους παράγοντες βαρύτητας των αρχικών μεταβλητών (χαρακτηριστικών). Δηλαδή, ταξινομήσαμε τα χαρακτηριστικά (γονίδια) σύμφωνα με τους παράγοντες βαρύτητας που έλαβαν στην πρώτη κύρια συνιστώσα.

Ο κώδικας που συντάχθηκε για την αξιολόγηση των γονιδίων με PCA είναι μέρος του παραρτήματος 9.1.1. Αρχικά, με την χρήση των βιβλιοθηκών pandas και numpy φορτώσαμε το σύνολο δεδομένων με τις τιμές των χαρακτηριστικών (γονιδίων) για κάθε πρότυπο (δείγμα - διάνυσμα) στον πίνακα data. Στις γραμμές του κώδικα 22-27, για την ορθή χρήση της μεθόδου, για κάθε χαρακτηριστικό (γονίδιο) υπολογίσαμε την μέση τιμή των τιμών της και την αφαιρέσαμε από όλες τις τιμές της. Το αποτέλεσμα αυτής της διαδικασίας είναι η μέση τιμή της κάθε μεταβλητής να γίνει ίση με μηδέν.

```
22 # Extract vector with mean values for every column
23 mean_data = np.mean(data, axis=0)
24
25 # Remove mean from each column
26 for icol in range(data.shape[1]):
27    data[:,icol] = data[:,icol] - mean_data[icol]
```

Απόσπασμα Κώδικα 1: *Απόσπασμα Παραρτήματος 9.1.1, γραμμές κώδικα 22-27 (PCA & (Grades) SVM Linear)*

Στην συνέχεια, χρησιμοποιήσαμε την υλοποιημένη μέθοδο sklearn.decomposition.PCA()[4] της βιβλιοθήκης Scikit-learn για την εξαγωγή της τιμής της πρώτης κύριας συνιστώσας για κάθε πρότυπο. Συγκεκριμένα, δώσαμε ως παράμετρο στην μέθοδο τον αριθμό των συνιστωσών που θέλουμε να εξάγουμε (γραμμή κώδικα 29) και έπειτα εισήγαμε στο μοντέλο τον πίνακα

[4] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

(data) με τις τιμές των μεταβλητών για κάθε πρότυπο (δείγμα - διάνυσμα) για την μείωση των διαστάσεων (μεταβλητών) και την εξαγωγή της τιμής της κύριας συνιστώσας για κάθε πρότυπο (γραμμή κώδικα 30).

```
29 pca = PCA(n_components = 1)
30 data_pca = pca.fit_transform(data)
31
32 comp = pca.components_
33
34 features_idx_max = np.argsort(-comp)
35 features_idx_max_w = np.transpose(features_idx_max)
36 features_idx_max_w = features_idx_max_w.flatten()
```

Απόσπασμα Κώδικα 2: *Απόσπασμα Παραρτήματος 9.1.1, γραμμές κώδικα 29-36 (PCA & (Grades) SVM Linear)*

Στην συνέχεια, λάβαμε τους παράγοντες βαρύτητας (γραμμή κώδικα 32) του γραμμικού συνδυασμού των αρχικών μεταβλητών, οι οποίοι καθορίζουν τον βαθμό στον οποίο η κάθε αρχική μεταβλητή επηρεάζει κάθε συνιστώσα (στην παρούσα περίπτωση την πρώτη συνιστώσα). Έπειτα, ταξινομήσαμε τις αρχικές μεταβλητές (χαρακτηριστικά - γονίδια) με βάση τις τιμές των παραγόντων βαρύτητας, σε φθίνουσα σειρά (γραμμές κώδικα 34-36). Οπότε καταλήξαμε σε έναν πίνακα (features_idx_max_w) με μία στήλη η οποία περιέχει τους δείκτες θέσης των αρχικών μεταβλητών (χαρακτηριστικών - γονιδίων), οι οποίες ταξινομήθηκαν με βάση την τιμή της βαρύτητας που έλαβαν κατά την εξαγωγή της πρώτης κύριας συνιστώσας.

Το ίδιο κομμάτι κώδικα συμπεριλαμβάνεται και στο παράρτημα 9.1.2.

4.2.2 Fisher Score

Το κριτήριο Fisher δεν το βρήκαμε υλοποιημένο σε Python, οπότε αναπτύξαμε τον απαιτούμενο κώδικα για την αξιολόγηση των χαρακτηριστικών (γονιδίων) σύμφωνα με τις κλάσεις της στήλης Grades και σύμφωνα με τις κλάσεις της στήλης Stages.

Ο κώδικας που συντάχθηκε για την αξιολόγηση των γονιδίων με το κριτήριο Fisher Score είναι μέρος του παραρτήματος 9.2.1. Παρακάτω περιγράφουμε την εφαρμογή του Fisher Score σύμφωνα με τις κλάσεις της στήλης Grades. Η αντίστοιχη διαδικασία ακολουθήθηκε και για την εφαρμογή του Fisher Score σύμφωνα με τις κλάσεις της στήλης Stages.

Αρχικά, με την χρήση των βιβλιοθηκών pandas και numpy φορτώσαμε το σύνολο δεδομένων με τις τιμές των χαρακτηριστικών (γονιδίων) για κάθε πρότυπο (δείγμα - διάνυσμα) στον πίνακα data2. Επίσης φορτώσαμε το σύνολο δεδομένων που περιέχει τις κλάσεις των δύο προβλημάτων ταξινόμησης (data1) και δημιουργήσαμε τον πίνακα grades, ο οποίος περιέχει

την τιμή κλάσης της στήλης Grades για κάθε πρότυπο. Συγκεκριμένα, όπου Early_Grade δώσαμε ως ετικέτα κλάσης την τιμή 0 (αντικαταστήσαμε τα Early_Grade με την τιμή 0) και όπου Late_Grade δώσαμε ως ετικέτα κλάσης την τιμή 1 (αντικαταστήσαμε τα Late_Grade με την τιμή 1) (γραμμές κώδικα 21-25).

```
21 # targets for Grades (0-1)
22 map_dict = {'Early_Grade':0, 'Late_Grade':1}
23 targets_grades = np.array([map_dict[key] for key in
   data1['Grades'].values])
24
25 grades = targets_grades.astype('float64')
```

Απόσπασμα Κώδικα 3: *Απόσπασμα Παραρτήματος 9.2.1, γραμμές κώδικα 21-25 (Grades: Fisher Score & SVM Linear)*

Στην συνέχεια, μέσα σε έναν βρόχο επανάληψης, για κάθε χαρακτηριστικό (γονίδιο) δημιουργήσαμε έναν πίνακα με τις τιμές του χαρακτηριστικού όπου η ετικέτα κλάσης είναι 0 και έναν πίνακα με τις τιμές του χαρακτηριστικού όπου η ετικέτα κλάσης είναι 1 (γραμμές κώδικα 29-37). Χρησιμοποιώντας για κάθε χαρακτηριστικό τους δύο παραπάνω πίνακες, όπου περιέχουν τις τιμές του για κάθε κλάση, υλοποιήσαμε τον τύπο του Fisher Score στην γραμμή του κώδικα 39. Συγκεκριμένα υλοποιήσαμε το πηλίκο, όπου στον αριθμητή έχουμε την διαφορά των μέσων τιμών (np.mean) μεταξύ των τιμών του χαρακτηριστικού στις δύο κλάσεις, εις το τετράγωνο και στον παρονομαστή έχουμε το άθροισμα των διακυμάνσεων (np.var) μεταξύ των τιμών του χαρακτηριστικού στις δύο κλάσεις.

Σε κάθε επανάληψη κρατούσαμε στον πίνακα fisher_score_grades την βαθμολογία του κάθε χαρακτηριστικού, οπότε μετά από την ολοκλήρωση του επαναληπτικού βρόχου, συγκεντρώσαμε στον πίνακα fisher_score_grades τις βαθμολογίες όλων των χαρακτηριστικών (γραμμή κώδικα 40). Τέλος δημιουργήσαμε τον πίνακα grades_max ο οποίος περιέχει τους δείκτες θέσης των χαρακτηριστικών (γονιδίων), ταξινομώντας τα χαρακτηριστικά βάση της βαθμολογίας που έλαβαν σε φθίνουσα σειρά (γραμμή κώδικα 42).

Όσο μεγαλύτερη είναι η βαθμολογία του Fisher Score για ένα χαρακτηριστικό (γονίδιο), σημαίνει ότι τόσο περισσότερο διαφοροποιείται μεταξύ των δύο κλάσεων, οπότε θεωρείται σημαντικό για την κατασκευή ενός αποδοτικού ταξινομητή.

```
27 ### GRADES FISHER SCORE ###################################
28 fisher_score_grades = np.array([])
29 for icol in range(data2.shape[1]):
30     early_grade_array = np.array([]) #0
31     late_grade_array = np.array([])  #1
32
```

```
33      for irow in range(data2.shape[0]):
34          if grades[irow] == 0:
35              early_grade_array =  np.append(early_grade_array,
    [data2[irow,icol]], axis=0)
36          if grades[irow] == 1:
37              late_grade_array =  np.append(late_grade_array,
    [data2[irow,icol]], axis=0)
38
39      fs = ((np.mean(early_grade_array) - np.mean(late_grade_array))**2) /
    (np.var(early_grade_array) + np.var(late_grade_array))
40      fisher_score_grades =  np.append(fisher_score_grades, [fs], axis=0)
41
42  grades_max = np.argsort(-fisher_score_grades)
43  np.savetxt("fisher-score_GRADES.csv", fisher_score_grades,
    delimiter=",",fmt='%.12f')
44  ### END GRADES FISHER SCORE ################################
```

Απόσπασμα Κώδικα 4: *Απόσπασμα Παραρτήματος 9.2.1, γραμμές κώδικα 27-44 (Grades: Fisher Score & SVM Linear)*

Το αντίστοιχο κομμάτι κώδικα που αφορά την εφαρμογή του κριτηρίου Fisher για την μεταβλητή στόχος Stages, συμπεριλαμβάνεται στο παραρτήματα 9.2.2.

4.2.3 Mutual Information (MI)

Για την εφαρμογή της Αμοιβαίας Πληροφορίας (Mutual Information, MI) και την αξιολόγηση των χαρακτηριστικών (γονιδίων) βάσει του Κέρδους της Πληροφορίας (Information Gain, IG), χρησιμοποιήσαμε την υλοποιημένη συνάρτηση sklearn.metrics.normalized_mutual_info_score()[5] της βιβλιοθήκης Scikit-learn. Η συνάρτηση της Κανονικοποιημένης (normalized) Αμοιβαίας Πληροφορίας μεταξύ δύο συστάδων είναι μια ομαλοποίηση της βαθμολογίας της Αμοιβαίας Πληροφορίας για την κλιμάκωση των αποτελεσμάτων μεταξύ 0 (κανένα κέρδος σε πληροφορία) και 1 (τέλεια συσχέτιση - μέγιστο κέρδος σε πληροφορία).

Για την αξιολόγηση των χαρακτηριστικών με την Αμοιβαία Πληροφορία, αρχικά πραγματοποιήσαμε διακριτοποίηση των τιμών των χαρακτηριστικών για κάθε πρότυπο με διαχωρισμό ίσων διαστημάτων (equidistant binning) (παράρτημα 9.3.1, γραμμές κώδικα 28-50), καθώς όπως προαναφέρθηκε οι τιμές των χαρακτηριστικών για κάθε πρότυπο αποτελούν ένα συνεχές διάστημα τιμών με μικρότερη τιμή την 3.92734020987 και μεγαλύτερη την 23.3670292757.

Για την διακριτοποίηση των τιμών των χαρακτηριστικών για κάθε πρότυπο ακολουθήσαμε την εξής διαδικασία. Αρχικά, στο παράρτημα 9.3.1, με την χρήση των βιβλιοθηκών pandas

[5] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html

και numpy φορτώσαμε στον πίνακα data2 το σύνολο των τιμών των χαρακτηριστικών. Με την χρήση της numpy.histogram()[6] (γραμμή κώδικα 29), χωρίσαμε το σύνολο τιμών (data2) σε 20 τμήματα ίσων διαστημάτων (bins=20) με εύρος τιμών από την τιμή 3 έως και την τιμή 24 (range=(3,24)). Για κάθε τμήμα (bin) υπολογίσαμε τον μέσο όρο των τιμών των 'τοιχωμάτων' του (γραμμές κώδικα 31-36). Στην συνέχεια, σε ένα αντίγραφο του πίνακα data2 (data2_copy), αντικαταστήσαμε τις τιμές των χαρακτηριστικών για κάθε πρότυπο με τον αριθμό του τμήματος (bin) στο οποίο ανήκει η κάθε μία τιμή (γραμμές κώδικα 38-40). Τέλος, στον πίνακα data2_copy αντικαταστήσαμε τον κάθε αριθμό τμήματος με τον αντίστοιχο μέσο όρο των 'τοιχωμάτων' του (γραμμές κώδικα 42-49).

```
28 ### Discretization ########################################
29 frequencies_all, bin_edges_all = np.histogram(data2, bins=20, range=(3,
   24))
```

Απόσπασμα Κώδικα 5: *Απόσπασμα Παραρτήματος 9.3.1, γραμμές κώδικα 28-29 (Grades: Mutual Information & SVM Linear)*

Τον πίνακα data2_copy, με τις διακριτές τιμές των χαρακτηριστικών για κάθε πρότυπο, τον χρησιμοποιήσαμε στην συνέχεια για την βαθμολόγηση των χαρακτηριστικών βάση του Κέρδους της Πληροφορίας.

Έπειτα από την διακριτοποίηση των τιμών των χαρακτηριστικών, εισάγαμε στην normalized_mutual_info_score() τις απαιτούμενες παραμέτρους για τον υπολογισμό του κέρδους της πληροφορίας του κάθε χαρακτηριστικού. Οι παράμετροι που δώσαμε είναι οι εξής:

- **labels_true[samples]**: Πίνακας με τις S διακριτές τιμές μιας τυχαίας μεταβλητής X (στην περίπτωση μας οι διακριτές τιμές του χαρακτηριστικού στα S πρότυπα (δείγματα)).
- **labels_pred[samples]**: Πίνακας με τις S διακριτές τιμές μιας τυχαίας μεταβλητής Y (στην περίπτωση μας οι διακριτές τιμές (0 για Early_Grade και 1 για Late_Grade) της στήλης Grades στα S πρότυπα (δείγματα)).

Στις γραμμές του κώδικα 54-57, εφαρμόσαμε την παραπάνω συνάρτηση εντός ενός επαναληπτικού βρόχου, όπου σε κάθε επανάληψη δίνουμε τις διακριτές τιμές του χαρακτηριστικού X_i στα S πρότυπα (τυχαία μεταβλητή X) και τις διακριτές τιμές της στήλης Grades (0 για Early_Grade και 1 για Late_Grade) στα S πρότυπα. Σε κάθε επανάληψη κρατάμε στον πίνακα ing_grades την τιμή της αμοιβαίας πληροφορίας του εξεταζόμενου χαρακτηριστικού. Τέλος δημιουργήσαμε τον πίνακα grades_max_ig ο οποίος περιέχει τους

[6] https://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html

δείκτες θέσης των χαρακτηριστικών (γονιδίων), ταξινομώντας τα χαρακτηριστικά βάση της βαθμολογίας που έλαβαν σε φθίνουσα σειρά (γραμμή κώδικα 60).

```
52 ### Information Gain ######################################
53 ing_grades = np.array([])
54 for icol in range(data2_copy.shape[1]):
55     # information gain for grades
56     info_gain_grades =
   normalized_mutual_info_score(data2_copy[:,icol],targets_grades)
57     ing_grades = np.append(ing_grades, [info_gain_grades], axis=0)
58
59 # Sorted Genes according to Information Gain
60 grades_max_ig = np.argsort(-ing_grades)
61 ### End Information Gain ####################################
```

Απόσπασμα Κώδικα 6: *Απόσπασμα Παραρτήματος 9.3.1, γραμμές κώδικα 52-61 (Grades: Mutual Information & SVM Linear)*

Όσο μεγαλύτερη είναι η αμοιβαία πληροφορία ενός χαρακτηριστικού (γονίδιο), τόσο περισσότερο είναι το κέρδος σε πληροφορία που υπάρχει για αυτό το χαρακτηριστικό. Συνεπώς αυτό το χαρακτηριστικό μπορεί να θεωρηθεί σημαντικό για την κατασκευή ενός αποδοτικού ταξινομητή.

Το αντίστοιχο κομμάτι κώδικα που αφορά τον υπολογισμό της Αμοιβαίας Πληροφορίας για την μεταβλητή στόχος Stages, συμπεριλαμβάνεται στο παραρτήματα 9.3.2.

4.2.4 Kolmogorov-Smirnov 2 Samples Test (KS 2Samples Test)

Για την εφαρμογή του κριτηρίου Kolmogorov-Smirnov για δύο δείγματα χρησιμοποιήσαμε την συνάρτηση scipy.stats.ks_2samp()[7] της βιβλιοθήκης Scipy. Οι παράμετροι που δέχεται η συνάρτηση είναι οι εξής:

- **Data1:** Αποτελεί έναν πίνακα με τις τιμές του πρώτου δείγματος προς σύγκριση (1st sample). Στην περίπτωση μας είναι ένας πίνακας με τιμές του χαρακτηριστικού X_i που αντιστοιχούν στην κλάση Early_Grade της στήλης Grades. Όπου Early_Grade στην στήλη Grade έχουμε αντικαταστήσει με την τιμή 0.
- **Data2:** Αποτελεί έναν πίνακα με τις τιμές του δεύτερου δείγματος προς σύγκριση (2nd sample). Στην περίπτωση μας είναι ένας πίνακας με τιμές του χαρακτηριστικού X_i που αντιστοιχούν στην κλάση Late_Grade της στήλης Grades. Όπου Late_Grade στην στήλη Grade έχουμε αντικαταστήσει με την τιμή 1.

[7] https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.ks_2samp.html

Ουσιαστικά τα data1 και data2 είναι δύο συστοιχίες παρατηρήσεων (τιμών) του ίδιου χαρακτηριστικού όπου οι τιμές της κάθε μίας αντιστοιχούν στην μία από τις δύο κλάσεις της στήλης Grades. Τα μεγέθη των πινάκων data1 και data2 μπορεί να είναι διαφορετικά (όπου στην περίπτωση μας είναι διαφορετικά).

Ο κώδικας που συντάχθηκε για την εφαρμογή του κριτηρίου Kolmogorov-Smirnov για 2 δείγματα είναι μέρος του παραρτήματος 9.4.1. Αρχικά, με την χρήση των βιβλιοθηκών pandas και numpy φορτώσαμε το σύνολο δεδομένων με τις τιμές των χαρακτηριστικών (γονιδίων) για κάθε πρότυπο (δείγμα - διάνυσμα) στον πίνακα data2. Επίσης φορτώσαμε το σύνολο δεδομένων που περιέχει τις κλάσεις των δύο προβλημάτων ταξινόμησης (data1) (γραμμές κώδικα 14-20).

Έπειτα, ενώσαμε τις στήλες των πινάκων data1 και data2 και δημιουργήσαμε τον πίνακα data_all (γραμμή κώδικα 24). Από τον πίνακα data_all επιλέξαμε και αποθηκεύσαμε στον πίνακα final_early_grade, όλα τα πρότυπα (γραμμές) που είχαν στην στήλη Grades την τιμή Early_Grade και στην συνέχεια αφαιρέσαμε (διαγράψαμε) από τον πίνακα final_early_grade τις 4 στήλες που προέρχονταν από τον πίνακα data1 (γραμμές κώδικα 25-28). Οπότε ο πίνακας final_early_grade περιέχει τις τιμές των χαρακτηριστικών για τα πρότυπα που έχουν ως τιμή κλάσης Early_Grade. Με την αντίστοιχη διαδικασία δημιουργήσαμε τον πίνακα final_late_grade (γραμμές κώδικα 29-32).

```
22 ### KS 2Sample ##########################################
23 # merge data1_matrix & data2
24 data_all = np.concatenate((data1_matrix,data2), axis=1)
25 # instances with target class Early Grade
26 data_early_grade = data_all[data_all[:,2] == 'Early_Grade']
27 final_early_grade = np.delete(data_early_grade, (0,1,2,3), axis=1)
28 final_early_grade = final_early_grade.astype('float64')
29 # instances with target class Late Grade
30 data_late_grade = data_all[data_all[:,2] == 'Late_Grade']
31 final_late_grade = np.delete(data_late_grade, (0,1,2,3), axis=1)
32 final_late_grade = final_late_grade.astype('float64')
33
34 p_value_grade_array = np.array([])
35 # for every gene -->
36 # 1st Sample: values with target class Early Grade
37 # 2nd Sample: values with target class Late Grade
38 for icol in range(final_early_grade.shape[1]):
39     gene_early_grade = np.array([])
40     gene_late_grade = np.array([])
41     gene_early_grade = final_early_grade[:,icol]
42     gene_late_grade =  final_late_grade[:,icol]
43     ks_statistic_grade, pvalue_grade = stats.ks_2samp(gene_early_grade,
   gene_late_grade)
```

```
44      p_value_grade_array =  np.append(p_value_grade_array,
    [pvalue_grade], axis=0)
45
46 # Sorted Genes according to p-value (information gain)
47 grades_min = np.argsort(p_value_grade_array)
48 ### end KS 2Sample ########################################
```

Απόσπασμα Κώδικα 7: *Απόσπασμα Παραρτήματος 9.4.1, γραμμές κώδικα 22-48 (Grades: KS 2 Samples & SVM Linear)*

Στην συνέχεια, εντός ενός επαναληπτικού βρόχου, για κάθε χαρακτηριστικό παίρνουμε τις τιμές του που αντιστοιχούν στην κλάση Early_Grade (από τον πίνακα final_early_grade) και τις τιμές του που αντιστοιχούν στην κλάση Late_Grade (από τον πίνακα final_late_grade). Οι πίνακες που προκύπτουν gene_early_grade και gene_late_grade σε κάθε επανάληψη αποτελούν τους δύο πίνακες που δέχεται ως παράμετρο η συνάρτηση stats.ks_2samp() (γραμμές κώδικα 35-43).

Η έξοδος της συνάρτησης stats.ks_2samp() για κάθε χαρακτηριστικό είναι η ks_statistic_grade (D) και η pvalue_grade (p-value). Η ks_statistic_grade είναι η στατιστική τιμή D του Kolmogorov-Smirnov για 2 δείγματα η οποία αποτελεί την μεγαλύτερη απόλυτη απόκλιση (διαφορά) μεταξύ των δύο σωρευτικών κατανομών των 2 δειγμάτων, ανεξάρτητα από την κατεύθυνσης της διαφοράς. Η pvalue_grade (p-value) είναι η πιθανότητα η τιμή της στατιστικής τιμής D του Komogorov-Smirnov να είναι τόσο μεγάλη ή μεγαλύτερη από αυτή που παρατηρήθηκε. Οπότε εάν η τιμή p-value είναι μικρή, καταλήγουμε στο συμπέρασμα ότι τα δύο δείγματα προέρχονται από πληθυσμούς με διαφορετικές κατανομές.

Δηλαδή στην περίπτωση μας, το χαρακτηριστικό με την μικρότερη p-value σημαίνει ότι μπορεί να θεωρηθεί σημαντικό για την κατασκευή ενός αποδοτικού ταξινομητή, επειδή οι τιμές του για κάθε πρότυπο στις δύο κλάσεις δείχνουν να προέρχονται από διαφορετικές κατανομές. Με άλλα λόγια, το χαρακτηριστικό δείχνει να διαφοροποιείται αρκετά από την μία κλάση στην άλλη.

Στην γραμμή του κώδικα 44 αποθηκεύουμε στον πίνακα p_value_grade_array την pvalue_grade (p-value) του κάθε χαρακτηριστικού. Τέλος, στην γραμμή του κώδικα 47 δημιουργούμε τον πίνακα grades_min ο οποίος περιέχει τους δείκτες θέσης των χαρακτηριστικών (γονιδίων), ταξινομώντας τα χαρακτηριστικά βάση της τιμής pvalue_grade (p-value) που έλαβαν σε αύξουσα σειρά.

Το αντίστοιχο κομμάτι κώδικα που αφορά την εφαρμογή του κριτηρίου Kolmogorov-Smirnov για 2 δείγματα για την μεταβλητή στόχος Stages, συμπεριλαμβάνεται στο παραρτήματα 9.4.2.

4.2.5 ReliefF

Για την εφαρμογή του αλγορίθμου της ReliefF χρησιμοποιήσαμε το πακέτο ReliefF 0.1.2[8], το οποίο βρήκαμε στο ευρετήριο πακέτων της Python[9] (Python Package Index, PyPI). Το ευρετήριο πακέτων της Python είναι ένα αποθετήριο λογισμικού (repository of software) για την γλώσσα προγραμματισμού Python. Στο παράρτημα 9.5.1 υπάρχει ο κώδικας που συντάχθηκε για την εφαρμογή της ReliefF για την μεταβλητή στόχος Grades και για την Stages.

Αρχικά, με την χρήση των βιβλιοθηκών pandas και numpy φορτώσαμε το σύνολο δεδομένων με τις τιμές των χαρακτηριστικών (γονιδίων) για κάθε πρότυπο (δείγμα - διάνυσμα) στον πίνακα data2. Επίσης φορτώσαμε το σύνολο δεδομένων που περιέχει τις κλάσεις των δύο προβλημάτων ταξινόμησης (data1) και δημιουργήσαμε τον πίνακα grades, ο οποίος περιέχει την τιμή κλάσης της στήλης Grades για κάθε πρότυπο. Συγκεκριμένα, όπου Early_Grade δώσαμε ως ετικέτα κλάσης την τιμή 0 (αντικαταστήσαμε τα Early_Grade με την τιμή 0) και όπου Late_Grade δώσαμε ως ετικέτα κλάσης την τιμή 1 (αντικαταστήσαμε τα Late_Grade με την τιμή 1). Η αντίστοιχη διαδικασία ακολουθήθηκε και για την μεταβλητή κλάσης Stages. (γραμμές κώδικα 20-28).

```
30 ### ReliefF for Grades ######################################
31 results_idx_grades = np.zeros(shape=(500,50))
32 csv_col = -1
33
34 for nf in range(10, 510, 10):
35
36     X_train, X_test, y_train, y_test = train_test_split(data2, grades)
37     y_train = y_train.flatten()
38     y_test = y_test.flatten()
39
40     fs = ReliefF(n_neighbors=10, n_features_to_keep = nf)
41     X_train = fs.fit_transform(X_train, y_train)
42     X_test_subset = fs.transform(X_test)
43     print(X_test.shape, X_test_subset.shape)
44
45     trX_test = np.transpose(X_test)
46     trX_test_subset = np.transpose(X_test_subset)
47
48     trX_test = {tuple(row):i for i,row in enumerate(trX_test)}
49     answer = []
50     for i,row in enumerate(trX_test_subset):
```

[8] https://pypi.org/project/ReliefF/0.1.2/

[9] https://pypi.org/

```
51          if tuple(row) in trX_test:
52             answer.append((trX_test[tuple(row)], i))
53
54      npa = array( answer )
55
56      csv_col = csv_col + 1
57
58      for row in range(0, nf, 1):
59          results_idx_grades[row,csv_col] = npa[row,0]
60
61  np.savetxt("GRADES_relief_idx.csv", results_idx_grades,
    delimiter=",",fmt='%.0f')
62
63  ### ReliefF for Stages ###################################
```

Απόσπασμα Κώδικα 8: *Απόσπασμα Παραρτήματος 9.5.1, γραμμές κώδικα 30-63 (Grades & Stages: Εφαρμογή ReliefF)*

Στην συνέχεια εφαρμόσαμε την ReliefF, σύμφωνα με το αντίστοιχο παράδειγμα[10], για την μεταβλητή κλάσης Grades και για την Stages. Οι παράμετροι που δέχεται ο αλγόριθμος της ReliefF είναι οι εξής:

- **n_neighbors:** Αποτελεί έναν ακέραιο αριθμό ο οποίος δηλώνει τον αριθμό των πλησιέστερων γειτόνων που ανήκουν στην ίδια κλάση με το τυχαία επιλεγμένο πρότυπο και τον αριθμό των πλησιέστερων γειτόνων που ανήκουν σε διαφορετική κλάση από το τυχαία επιλεγμένο πρότυπο.
- **n_features_to_keep:** Αποτελεί έναν ακέραιο αριθμό ο οποίος δηλώνει τον αριθμό των χαρακτηριστικών με την μεγαλύτερη διακριτική ισχύ που θέλουμε να επιλεγεί από τον αλγόριθμο.

Η τιμή που δώσαμε στην παράμετρο n_neighbors, όπως προτείνεται και στην βιβλιογραφία ως μία καλή επιλογή [22], είναι 10. Εντός ενός επαναληπτικού βρόχου εφαρμόσαμε τον αλγόριθμο της ReliefF, όπου σε κάθε επανάληψη αυξάνουμε την τιμή της παραμέτρου n_features_to_keep, ξεκινώντας από την τιμή 10 έως την τιμή 500 με βήμα 10. Επιπλέον σε κάθε επανάληψη κρατάμε τους δείχτες θέσης των επιλεγμένων χαρακτηριστικών, που μας επιστρέφει ο αλγόριθμος ως έξοδο, στον πίνακα results_idx_grades (γραμμές κώδικα 34-59).

Μετά την ολοκλήρωση της εφαρμογής της ReliefF, για την μεταβλητή στόχος Grades και για την Stages, αποθηκεύσαμε στα αντίστοιχα αρχεία (csv) τους δείκτες θέσης των χαρακτηριστικών που επιλέχθηκαν.

[10] https://libraries.io/pypi/ReliefF/0.1.2

4.2.6 Minimum Redundancy Maximum Relevance (mRMR)

Για την εφαρμογή του κριτηρίου του ελάχιστου πλεονασμού - μέγιστης συσχέτισης (mRMR) χρησιμοποιήσαμε το πακέτο λογισμικού mRMR του Peng, H.[11]. Η υλοποίηση του πακέτου PymRMR[12] σε Python 3 έχει πραγματοποιηθεί από τον Brundu, F. με την συνεισφορά του Peng, H.. και είναι βασισμένη στο επιστημονικό άρθρο των Peng, H., Long, F., κ.α. με τίτλο "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy" [27]. Για την εγκατάσταση και την εφαρμογή του πακέτου PymRMR εγκαταστήσαμε την έκδοση 3 της Python και μία λίστα συγκεκριμένων βιβλιοθηκών[13].

Μία από τις παραμέτρους εισόδου του PymRMR πρέπει να είναι ένα pandas.DataFrame που περιέχει διακριτοποιημένο το σύνολο δεδομένων εισόδου. Οι σειρές δεδομένων πρέπει να είναι τα διαφορετικά πρότυπα. Η πρώτη στήλη πρέπει να είναι η μεταβλητή ταξινόμησης (στόχος) για κάθε πρότυπο. Οι υπόλοιπες στήλες πρέπει να είναι τα χαρακτηριστικά (μεταβλητές) που μπορούν να επιλεγούν από τον αλγόριθμο. Επιπλέον, τα ονόματα των στηλών (ονόματα χαρακτηριστικών - headers) πρέπει να είναι τύπου συμβολοσειράς.

Στο παράρτημα 9.6.1 υπάρχει ο κώδικας που συντάχθηκε για την δημιουργία του διακριτοποιημένου συνόλου δεδομένων με διαχωρισμό ίσων διαστημάτων (equidistant binning). Αρχικά, με την χρήση των βιβλιοθηκών pandas και numpy φορτώσαμε στον πίνακα data2 το σύνολο των τιμών των χαρακτηριστικών και στον πίνακα data1 το σύνολο δεδομένων που περιέχει τις μεταβλητές ταξινόμησης (στόχοι) Grades και Stages (γραμμές κώδικα 11-17). Έπειτα, δημιουργήσαμε τον πίνακα targets_grades από τον πίνακα data1, ο οποίος περιέχει την κλάση (στόχο) της μεταβλητής Grades για κάθε πρότυπο. Συγκεκριμένα, αντικαταστήσαμε όπου Early_Grade με την τιμή 0 και όπου Late_Grade με την τιμή 1. Αντίστοιχα δημιουργήσαμε και τον πίνακα targets_stages (γραμμές κώδικα 19-24).

Στην συνέχεια, με την χρήση της numpy.histogram()[14], χωρίσαμε το σύνολο τιμών των χαρακτηριστικών (data2) σε 20 τμήματα ίσων διαστημάτων (bins=20) με εύρος τιμών από την τιμή 3 έως και την τιμή 24 (range=(3,24)). Τέλος, αντικαταστήσαμε τις τιμές των χαρακτηριστικών για κάθε πρότυπο με τον αριθμό του τμήματος (bin) στο οποίο ανήκει η κάθε μία τιμή (γραμμές κώδικα 26-35).

Στις γραμμές του κώδικα 37-41 δημιουργήσαμε το απαιτούμενο dataFrame (df_grades), ενώνοντας κατά στήλη τον πίνακα με τις τιμές της μεταβλητής στόχου Grades με τον πίνακα

[11] http://home.penglab.com/proj/mRMR/

[12] https://github.com/fbrundu/pymrmr

[13] https://github.com/fbrundu/pymrmr/blob/master/requirements_dev.txt

[14] https://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html

που περιέχει τις διακριτοποιημένες τιμές των χαρακτηριστικών. Αντίστοιχα δημιουργήσαμε και το απαιτούμενο dataFrame df_Stages. Ως ονόματα στις στήλες των χαρακτηριστικών δώσαμε τον αύξοντα αριθμό της κάθε στήλης (ως τύπο συμβολοσειράς).

Στο παραρτήματα 9.6.2 και 9.6.3 υπάρχει ο κώδικας που συντάχθηκε για την εφαρμογή του PymRMR για την μεταβλητή στόχος Grades και για την μεταβλητή στόχος Stages αντίστοιχα. Η μέθοδος PymRMR απαιτεί τις 3 παρακάτω παραμέτρους εισόδου (γραμμή κώδικα 11).

- Η πρώτη είναι ένα pandas.DataFrame με το περιεχόμενο που περιεγράφηκε παραπάνω.
- Η δεύτερη είναι μια συμβολοσειρά που καθορίζει την συνάρτηση αξιολόγησης που θέλουμε να χρησιμοποιηθεί. Οι πιθανές τιμές της είναι "MIQ" (σχέση 55) ή "MID" (σχέση 54). Η συνάρτηση αξιολόγησης που χρησιμοποιήσαμε στην παρούσα εφαρμογή είναι η "MIQ", καθώς δίνει μεγαλύτερη έμφαση στην μείωση της περιττής πληροφορίας.
- Η τρίτη παράμετρος εισόδου είναι ο αριθμός των χαρακτηριστικών που θέλουμε να επιλεγεί.

```
10 df_grades = pd.read_csv('grades_mrmr.csv')
11 li = pymrmr.mRMR(df_grades, 'MIQ', 100)
12
13 grades_mrmr_miq = np.asarray(li)
14 grades_mrmr_miq = grades_mrmr_miq.astype('int64')
15
16 np.savetxt("final_grades_mrmr_MIQ.csv", grades_mrmr_miq,
   delimiter=",",fmt='%.0f')
```

Απόσπασμα Κώδικα 9: *Απόσπασμα Παραρτήματος 9.6.2, γραμμές κώδικα 10-16 (Grades: Εφαρμογή mRMR)*

Η μέθοδος δίνει ως έξοδο τις επικεφαλίδες (headers από το DataFrame εισόδου) των χαρακτηριστικών που επιλέχθηκαν, ταξινομημένες σε φθίνουσα σειρά σύμφωνα με την τιμή της συνάρτησης αξιολόγησης. Το χαρακτηριστικό με την μεγαλύτερη τιμή μεταφράζεται ως το χαρακτηριστικό που έχει την μέγιστη συνάφεια με την κλάση (μέγιστη συσχέτιση) και παράλληλα είναι όσο το δυνατό ανόμοιο με τα υπόλοιπα χαρακτηριστικά που επιλέχθηκαν (ελάχιστος πλεονασμός). Στον πίνακα grades_mrmr_miq κρατάμε τις επικεφαλίδες (αύξοντας αριθμός χαρακτηριστικών/στηλών) των χαρακτηριστικών με την σειρά ταξινόμησης που μας επέστρεψε η μέθοδος (γραμμή κώδικα 13-14).

4.2.7 Συνδυασμοί Μεθόδων Μείωσης Διαστάσεων

Μετά από την ολοκλήρωση της αξιολόγησης των χαρακτηριστικών με κάθε ένα από τα παραπάνω κριτήρια (μεθόδους μείωσης διαστάσεων), συγκεντρώσαμε ταξινομημένους τους δείχτες θέσεις των χαρακτηριστικών για κάθε κριτήριο. Συγκεκριμένα, μετά την εφαρμογή των παραπάνω κριτηρίων, συγκεντρώσαμε τα παρακάτω αρχεία .csv:

- PCA_idx.csv
- GRADES_idx_fisher-max.csv
- GRADES_idx_infogain-max.csv
- GRADES_idx_ks2sample-min.csv
- GRADES_relief_idx.csv
- GRADES_idx_mrmrMIQ_max.csv
- STAGES_idx_fisher-max.csv
- STAGES _idx_infogain-max.csv
- STAGES _idx_ks2sample-min.csv
- STAGES _relief_idx.csv
- STAGES _idx_mrmrMIQ_max.csv

Το κάθε ένα από αυτά τα αρχεία περιέχει ταξινομημένους τους δείκτες θέσης των χαρακτηριστικών, σύμφωνα με το εκάστοτε κριτήριο. Όσο πιο ψηλά στην κατάταξη βρίσκεται ένα χαρακτηριστικό, τόσο πιο κρίσιμο (σημαντικό) θεωρείται για την κατασκευή ενός αποδοτικού ταξινομητή. Δηλαδή ενός ταξινομητή που διαθέτει την ικανότητα γενίκευσης. Ο χαρακτηρισμός ‘κρίσιμο’ (σημαντικό) χαρακτηριστικό μεταφράζεται, στις υποενότητες που προηγήθηκαν, με διαφορετικό τρόπο για κάθε κριτήριο αξιολόγησης (μέθοδο μείωσης διαστάσεων).

Έχοντας συγκεντρώσει τα παραπάνω αποτελέσματα, για τα πρώτα 500 χαρακτηριστικά του κάθε αρχείου, υπολογίσαμε τον αριθμό των κοινών χαρακτηριστικών (τομή υποσυνόλων) μεταξύ όλων των δυνατών συνδυασμών (για Grades και για Stages). Συνολικά, οι δυνατοί συνδυασμοί είναι 15 συνδυασμοί για την κλάση στόχος Grades και 15 συνδυασμοί για την κλάση στόχος Stages. Συγκεκριμένα, για κάθε συνδυασμό, υπολογίσαμε τον αριθμό των κοινών χαρακτηριστικών ξεκινώντας από την σύγκριση των πρώτων 10 σημαντικών χαρακτηριστικών έως και των πρώτων 500, αυξάνοντας κάθε φορά κατά 10.

4.3 Εφαρμογή Μηχανών Διανυσμάτων Υποστήριξης (SVM) με Χρήση Αναζήτησης Πλέγματος (Grid Search)

4.3.1 Αναζήτηση Πλέγματος (Grid Search)

Για την εφαρμογή του SVM χρησιμοποιήσαμε την συνάρτηση svm.SVC()[15] της βιβλιοθήκης Scikit-learn. Για να εντοπίσουμε τον κατάλληλο συνδυασμό των τιμών των παραμέτρων που βελτιστοποιούν την ικανότητα γενίκευσης του SVM με γραμμικό πυρήνα και με πυρήνα RBF, εφαρμόσαμε εξαντλητική αναζήτηση (αναζήτηση πλέγματος - grid search). Συγκεκριμένα, χρησιμοποιήσαμε την συνάρτηση GridSearchCV()[16] της βιβλιοθήκης Scikit-learn.

Για την εφαρμογή της αναζήτησης πλέγματος (grid search) για τον ταξινομητή SVM θέσαμε μία σειρά από παραμέτρους οι οποίες είναι οι παρακάτω.

```
110    clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
111    clf.fit(grades_data, grades_targets)
```

Απόσπασμα Κώδικα 10: *Απόσπασμα Παραρτήματος 9.3.1, γραμμές κώδικα 110-111 (Grades: KS 2 Samples & SVM Linear)*

- **estimator:** Είναι η συνάρτηση του εκτιμητή που θα χρησιμοποιηθεί για την εκτέλεση της αναζήτησης πλέγματος.

  Η συνάρτηση που θέσαμε, όπως προαναφέρθηκε, είναι η svm.SVC().
- **param_grid:** Αποτελεί μία λίστα από λεξικά με τις τιμές των παραμέτρων για την εφαρμογή του αντίστοιχου εκτιμητή.

  Οι τιμές των παραμέτρων που θέσαμε για την εφαρμογή του SVM με γραμμικό πυρήνα και με πυρήνα RBF περιγράφονται στις δύο επόμενες υποενότητες.
- **n_jobs:** Είναι ο αριθμός των διεργασιών που μπορεί να εκτελείται παράλληλα.

  Θέσαμε την τιμή -1 για να εκτελεστεί ο μέγιστος αριθμός παράλληλων διεργασιών.
- **cv:** Είναι ο αριθμός των k τμημάτων (folds) ίσου μεγέθους που θα διαχωριστούν τα πρότυπα (δείγματα) για την εφαρμογή της μεθόδου διασταύρωσης k τμημάτων (k-Fold Cross-Validation).

  Θέσαμε την τιμή 20.

[15] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[16] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

- **refit:** Είναι μία boolean μεταβλητή με την οποία δίνουμε την εντολή να επισκευαστεί - ξαναχτιστεί (refit) ο εκτιμητής με βάση τις καλύτερες τιμές παραμέτρων που εντοπίστηκαν σε όλο το σύνολο δεδομένων.

  Θέσαμε την τιμή False.
- **return_train_score:** Είναι μία boolean μεταβλητή με την οποία δηλώνουμε αν θέλουμε να συμπεριλαμβάνονται στα αποτελέσματα τα score των τμημάτων εκπαίδευσης (training folds).

  Θέσαμε την τιμή True.

Μετά από τον ορισμό των παραπάνω παραμέτρων, εισάγαμε στο μοντέλο τον πίνακα με τις τιμές των προτύπων (δειγμάτων) για κάθε χαρακτηριστικό του επιλεγμένου υποσυνόλου χαρακτηριστικών (πίνακας διανυσμάτων) και τον πίνακα με τις τιμές της αντίστοιχης μεταβλητής κλάσης (Grades / Stages).

Στις δύο παρακάτω υποενότητες περιγράφουμε τις παραμέτρους που δώσαμε για την εφαρμογή της αναζήτησης πλέγματος για τον ταξινομητή SVM με γραμμικό πυρήνα και για τον ταξινομητή SVM με πυρήνα RBF. Ουσιαστικά περιγράφεται το περιεχόμενο της παραμέτρου param_grid της συνάρτησης αναζήτησης πλέγματος GridSearchCV()**.**

Ο κώδικας που συντάχθηκε για την εφαρμογή της αναζήτησης πλέγματος για τον εκτιμητή svm.SVC(), συμπεριλαμβάνεται σε όλα τα παραρτήματα (εκτός των 9.5.1, 9.6.1, 9.6.2, 9.6.3). Το κάθε ένα από αυτά τα παραρτήματα αποτελεί αυτοτελή κομμάτι κώδικα στο οποίο συμπεριλαμβάνονται τα εξής:

- η εφαρμογή του εκάστοτε κριτηρίου (μέθοδος μείωσης διαστάσεων) για την αξιολόγηση των χαρακτηριστικών (βάση της μεταβλητής κλάσης Grades και βάση της μεταβλητής κλάσης Stages)
- η εφαρμογή αναζήτησης πλέγματος για τον εκτιμητή svm.SVC() (με γραμμικό πυρήνα και με πυρήνα RBF)

4.3.2 SVM με Γραμμικό Πυρήνα

Οι παράμετροι που θέσαμε για την εφαρμογή της συνάρτησης svm.SVC() με χρήση γραμμικού πυρήνα εντός της συνάρτησης της αναζήτησης πλέγματος είναι οι παρακάτω.

- **kernel:** string, optional (default='rbf')

  Καθορίζει τον τύπο του πυρήνα που θα χρησιμοποιήσει η συνάρτηση svm.SVC(). Οι τιμές που δέχεται είναι μία από τις προκαθορισμένες 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'. Επίσης, μπορεί να γίνει κλήση μίας μεθόδου που θα δημιουργήσει ο χρήστης. Εάν δεν δοθεί τιμή τότε χρησιμοποιείται η προκαθορισμένη "rbf".

Η τιμή που θέσαμε στην παράμετρο kernel για την χρήση γραμμικού πυρήνα είναι 'linear'.

- **C:** float, optional (default=1.0)

  Είναι η παράμετρος που προσδιορίζει το βάρος του κόστους των λάθος ταξινομήσεων (παράμετρος ποινής για βέλτιστη ταξινόμηση των κλάσεων). Εάν δεν δοθεί τιμή, τότε χρησιμοποιείται η προκαθορισμένη τιμή 1.

  Οι τιμές που θέσαμε στην παράμετρο C είναι οι παρακάτω:

  C: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000

```
107    parameter_candidates = [
108            {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
   'kernel': ['linear']}]
109
110    clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
111    clf.fit(grades_data, grades_targets)
```

Απόσπασμα Κώδικα 11: *Απόσπασμα Παραρτήματος 9.3.1, γραμμές κώδικα 107-111 (Grades: KS 2 Samples & SVM Linear)*

4.3.3 SVM με Πυρήνα RBF

Οι παράμετροι που θέσαμε για την εφαρμογή της συνάρτησης svm.SVC() με χρήση γραμμικού πυρήνα εντός της συνάρτησης της αναζήτησης πλέγματος είναι οι παρακάτω.

- **kernel:** string, optional (default='rbf')

  Καθορίζει τον τύπο του πυρήνα που θα χρησιμοποιήσει η συνάρτηση svm.SVC(). Οι τιμές που δέχεται είναι μία από τις προκαθορισμένες 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'. Επίσης, μπορεί να γίνει κλήση μίας μεθόδου που θα δημιουργήσει ο χρήστης. Εάν δεν δοθεί τιμή τότε χρησιμοποιείται η προκαθορισμένη "rbf".

  Η τιμή που θέσαμε στην παράμετρο kernel για την χρήση RBF πυρήνα είναι 'rbf'.

- **C:** float, optional (default=1.0)

  Είναι η παράμετρος που προσδιορίζει το βάρος του κόστους των λάθος ταξινομήσεων (παράμετρος ποινής για βέλτιστη ταξινόμηση των κλάσεων). Εάν δεν δοθεί τιμή, τότε χρησιμοποιείται η προκαθορισμένη τιμή 1.

  Οι τιμές που θέσαμε στην παράμετρο C είναι οι παρακάτω:

  C: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000

- **gamma:** float, optional (default='auto')

Είναι η παράμετρος που προσδιορίζει τον συντελεστή των πυρήνων. Συγκεκριμένα, η τιμή της gamma (gamma > 0) ελέγχει την ακτίνα της Γκαουσιανής συνάρτησης. Εάν δεν δοθεί τιμή, τότε χρησιμοποιείται η προκαθορισμένη τιμή 'auto' η οποία θέτει την τιμή 1/N όπου N ο αριθμός των χαρακτηριστικών (1/n_features).

Οι τιμές που θέσαμε στην παράμετρο gamma είναι οι παρακάτω:

gamma: 1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e+0, 1e+1, 1e+2, 1e+3, 1e+4

- **tol:** float, optional (default=1e-3)
  Είναι η τιμή που ορίζει το όριο ανοχής για το κριτήριο τερματισμού**.** Εάν δεν δοθεί τιμή, τότε χρησιμοποιείται η προκαθορισμένη τιμή '1e-3'.
  Η τιμή που θέσαμε στην παράμετρο tol είναι 1e-10 (επειδή η μικρότερη τιμή που θέσαμε στην παράμετρο gamma ήταν 1e-10)

```
107     parameter_candidates = [
108             {'kernel': ['rbf'],
109              'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
   1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
110              'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
111              'tol': [1e-10]}]
112
113     clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
114     clf.fit(grades_data, grades_targets)
```

Απόσπασμα Κώδικα 12: *Απόσπασμα Παραρτήματος 9.3.2, γραμμές κώδικα 107-114 (Stages: Mutual Information & SVM RBF)*

5

Αξιολόγηση

Σε αυτό το κεφάλαιο, αρχικά γίνεται μία αναφορά στις μετρικές που χρησιμοποιήθηκαν για την αξιολόγηση των χαρακτηριστικών (γονιδίων) σε σχέση με τις μεταβλητές στόχους Grades και Stages. Επιπλέον αναφέρονται οι μετρικές που χρησιμοποιήθηκαν για την αξιολόγηση και την σύγκριση της απόδοσης των ταξινομητών SVM με γραμμικό πυρήνα και με πυρήνα RBF. Στην συνέχεια περιγράφεται η μεθοδολογία που ακολουθήσαμε για την εκτέλεση των πειραμάτων. Έπειτα, παρουσιάζονται αναλυτικά τα αποτελέσματα της αξιολόγησης των χαρακτηριστικών (γονιδίων) μετά από την εφαρμογή του εκάστοτε κριτηρίου, σε σχέση με τις μεταβλητές στόχους Grades και Stages. Τέλος, παρουσιάζονται αναλυτικά και συγκρίνονται τα αποτελέσματα της αξιολόγησης της απόδοσης των ταξινομητών SVM με γραμμικό πυρήνα και με πυρήνα RBF για τις μεταβλητές στόχους Grades και Stages.

5.1 Παράμετροι Αξιολόγησης

Οι μέθοδοι μείωσης διαστάσεων (μέθοδοι επιλογής υποσυνόλων) που χρησιμοποιήθηκαν και οι αντίστοιχες μετρικές τους για την αξιολόγηση των χαρακτηριστικών σε σχέση με τις μεταβλητές στόχους Grades και Stages περιγράφονται αναλυτικά στις υποενότητες 3.7 έως και 3.12. Επιπλέον η περιγραφή της εφαρμογής τους υπάρχει στις υποενότητες 4.2.1 έως και 4.2.7. Επιγραμματικά, οι μέθοδοι που χρησιμοποιήθηκαν και οι αντίστοιχες μετρικές τους είναι οι παρακάτω:

Μέθοδοι Εξαγωγής Χαρακτηριστικών

- **Principal Component Analysis (PCA):** Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των χαρακτηριστικών είναι το βάρος των παραγόντων βαρύτητας της 1^{ης} Κύριας Συνιστώσας (σχέση 31).

Μέθοδοι Επιλογής Υποσυνόλου Χαρακτηριστικών (Μονοπαραγοντικές)

- **Fisher:** Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των χαρακτηριστικών είναι το Fisher Score (σχέση 32).
- **Mutual Information (MI):** Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των χαρακτηριστικών είναι το Κέρδος της Πληροφορίας (Information Gain) (σχέση 35).
- **Kolmogorov-Smirnov 2 Samples Test (KS 2Samples Test):** Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των χαρακτηριστικών είναι η στατιστική τιμή του κριτηρίου KS 2 Samples, p-value (σχέση 38).

Μέθοδοι Επιλογής Υποσυνόλου Χαρακτηριστικών (Πολυπαραγοντικές)

- **ReliefF:** Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των χαρακτηριστικών είναι η εκτίμηση της διακριτικής ισχύς (ποιότητας) του κάθε χαρακτηριστικού. Σε αυτό το σημείο είναι σημαντικό να αναφέρουμε ότι το πακέτο λογισμικού ReliefF 0.1.2 που χρησιμοποιήσαμε για την εφαρμογή της αντίστοιχης μεθόδου, επιστρέφει τους δείκτες θέσεις των N ζητούμενων χαρακτηριστικών με την υψηλότερη τιμή εκτίμησης της διακριτικής ισχύς (ποιότητας).
- **mRMR MIQ:** Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των χαρακτηριστικών αποτελεί τον συνδυασμό ικανοποίησης δύο υποθέσεων. Συγκεκριμένα το κάθε χαρακτηριστικό πρέπει να έχει όσο το δυνατό μεγαλύτερη συνάφεια με την κλάση (μέγιστη συσχέτιση - maximum relevance) και παράλληλα πρέπει να είναι όσο το δυνατόν ανόμοιο με τα υπόλοιπα χαρακτηριστικά του επιλεγμένου υποσυνόλου (ελάχιστος πλεονασμός - minimum redudancy). Η μετρική για την αξιολόγηση των χαρακτηριστικών βασίζεται στην Αμοιβαία Πληροφορία και δίνεται από την σχέση 55. Σε αυτό το σημείο, αρχικά είναι σημαντικό να αναφέρουμε ότι το πακέτο λογισμικού pymRMR που χρησιμοποιήσαμε για την εφαρμογή της αντίστοιχης μεθόδου, επιστρέφει τους δείκτες θέσεις των N ζητούμενων χαρακτηριστικών με την υψηλότερη τιμή της μετρικής (σχέση 55) που ικανοποιεί τις δύο υποθέσεις που προαναφέρθηκαν. Επιπλέον, λόγω του μεγάλου χρονικού διαστήματος που απαιτείται για την εκτέλεση και ολοκλήρωση της παρούσας μεθόδου, καταφέραμε να συλλέξουμε τα 100 χαρακτηριστικά με την υψηλότερη τιμή της μετρικής που προαναφέρθηκε.

Σύμφωνα λοιπόν με τις παραπάνω μετρικές, αξιολογήθηκαν και ταξινομήθηκαν τα χαρακτηριστικά (γονίδια). Όπως είναι κατανοητό, η κάθε μία από τις παραπάνω μετρικές έχει την δική της σημασία και βάση αυτής ταξινομήθηκαν αντίστοιχα τα χαρακτηριστικά (γονίδια).

Για την αξιολόγηση και την σύγκριση των ταξινομητών SVM με γραμμικό πυρήνα και με πυρήνα RBF χρησιμοποιήσαμε από τα κριτήρια επίδοσης ταξινομητών την ακρίβεια (Accuracy, σχέση 2), η οποία περιγράφεται αναλυτικά στην υποενότητα 3.3. Επιπλέον, για την σύγκριση των ταξινομητών λάβαμε υπόψιν μας, τις αντίστοιχες τιμές των παραμέτρων τους (π.χ. C, gamma) και τον αριθμό των χαρακτηριστικών του υποσυνόλου που χρησιμοποιήθηκε.

Ο αριθμός των χαρακτηριστικών του υποσυνόλου που χρησιμοποιείται για την κατασκευή ενός αποδοτικού ταξινομητή είναι σημαντικό να είναι αρκετά μικρός σε σχέση με τα 12604 χαρακτηριστικά του συνόλου δεδομένων. Ο σχετικά μικρός αριθμός των χαρακτηριστικών του επιλεγμένου υποσυνόλου αρχικά μειώνει την πιθανότητα εμφάνισης του φαινομένου της ‘κατάρας των πολλών διαστάσεων’ και επιπλέον δίνει την δυνατότητα (πλεονέκτημα) εύκολης ερμηνείας, από βιολογικής άποψης, των καθοριστικών γονιδίων (επιλεγμένων χαρακτηριστικών).

5.2 Οργάνωση Πειραμάτων & Αξιολόγηση

Η μεθοδολογία που ακολουθήσαμε για την εκτέλεση των πειραμάτων αποτελείται από 2 στάδια. Τα βήματα που ακολουθήσαμε στα 2 στάδια απεικονίζονται στην εικόνα 5.

Στο πρώτο στάδιο πραγματοποιήσαμε την αξιολόγηση των χαρακτηριστικών σύμφωνα με το κάθε κριτήριο αξιολόγησης, σε σχέση με την μεταβλητή στόχος Grades και την μεταβλητή στόχος Stages. Στο δεύτερο στάδιο, βάση της αξιολόγησης των χαρακτηριστικών σε σχέση με την μεταβλητή στόχος Grades και Stages, επιλέξαμε τα υποψήφια υποσύνολα χαρακτηριστικών και εκτελέσαμε εξαντλητική αναζήτηση (Grid Search) για τους εκτιμητές (ταξινομητές) SVM με γραμμικό πυρήνα και με πυρήνα RBF.

Τα βήματα από 1 έως και 4 της εικόνας 5 αποτελούν το πρώτο στάδιο της μεθοδολογίας που ακολουθήσαμε. Τα βήματα από 5 έως και 8 αποτελούν το δεύτερο στάδιο. Αναλυτικά τα βήματα αυτά είναι τα εξής:

Βήμα 1

Εισαγωγή του συνόλου δεδομένων (τιμές χαρακτηριστικών για κάθε πρότυπο) και της αντίστοιχης μεταβλητής στόχος Grades και Stages. Αντικατάσταση των τιμών της ποιοτικής μεταβλητής Grades με 0 όπου Early_Grade και 1 όπου Late_Grade. Η αντίστοιχη αντικατάσταση έγινε και για την ποιοτική μεταβλητή Stages.

Βήμα 2

Προεπεξεργασία του συνόλου δεδομένων για την εφαρμογή του εκάστοτε κριτηρίου αξιολόγησης. Για παράδειγμα, σε αυτό το βήμα περιέχονται διαδικασίες όπως η διακριτοποίηση των τιμών του συνόλου δεδομένων για την εφαρμογή της Αμοιβαίας Πληροφορίας. Αντίστοιχα υπάρχουν και κριτήρια αξιολόγησης για τα οποία δεν απαιτούνταν κάποια προεπεξεργασία του συνόλου δεδομένων (π.χ. Fisher Score)

Βήμα 3

Εφαρμογή του εκάστοτε κριτηρίου αξιολόγησης για την μεταβλητή στόχος Grades και για την μεταβλητή στόχος Stages.

Βήμα 4

Ταξινόμηση των χαρακτηριστικών σύμφωνα με την βαθμολογία του εκάστοτε κριτηρίου αξιολόγησης σε σχέση με την αντίστοιχη μεταβλητή στόχος (Grades / Stages). Αποθήκευση της τιμής της μετρικής του κριτηρίου αξιολόγησης και του αντίστοιχου δείκτη θέσης του κάθε χαρακτηριστικού σε αρχείο τύπου .csv.

Βήμα 5.1 - 5.2

Επιλογή των υποψήφιων υποσυνόλων χαρακτηριστικών για την κατασκευή ταξινομητών για την μεταβλητή στόχος Grades και για την μεταβλητή στόχος Stages. Δημιουργία ενός επαναληπτικού βρόχου από N = 10 έως και N = 500 με βήμα 10. Όπου N ο αριθμός των πρώτων N ταξινομημένων χαρακτηριστικών σύμφωνα με το εκάστοτε κριτήριο αξιολόγησης και την αντίστοιχη μεταβλητή στόχος. Το σύνολο των υποψήφιων υποσυνόλων που εξετάστηκε είναι 50 (500/10) για κάθε ταξινομητή.

Βήμα 6.1 - 6.2

Για κάθε υποσύνολο χαρακτηριστικών, εκτέλεση εξαντλητικής αναζήτησης (Grid Search) για την εύρεση των παραμέτρων των ταξινομητών SVM Linear και SVM RBF, οι οποίοι δίνουν ως αποτέλεσμα τον μέγιστο μέσο όρο ακρίβειας κατά την ταξινόμηση των συνόλων ελέγχου (K-fold cross-validation).

Για την βελτίωση της ικανότητας των μοντέλων ταξινόμησης να γενικεύουν σωστά, χρησιμοποιήθηκε για την εκπαίδευσή τους και την επικύρωσή τους η μέθοδος της διασταύρωσης με χρήση 20 τμημάτων (20 Fold Cross-Validation).

Οι τιμές της παραμέτρου C για τον ταξινομητή SVM Linear, που δόθηκαν για την εξαντλητική αναζήτηση είναι οι παρακάτω.

- **C: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000**

Οι τιμές των παραμέτρων C και gamma για τον ταξινομητή SVM RBF, που δόθηκαν για την εξαντλητική αναζήτηση είναι οι παρακάτω.

- **C: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000**
- **gamma: 1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e+0, 1e+1, 1e+2, 1e+3, 1e+4**

Βήμα 7.1 - 7.2

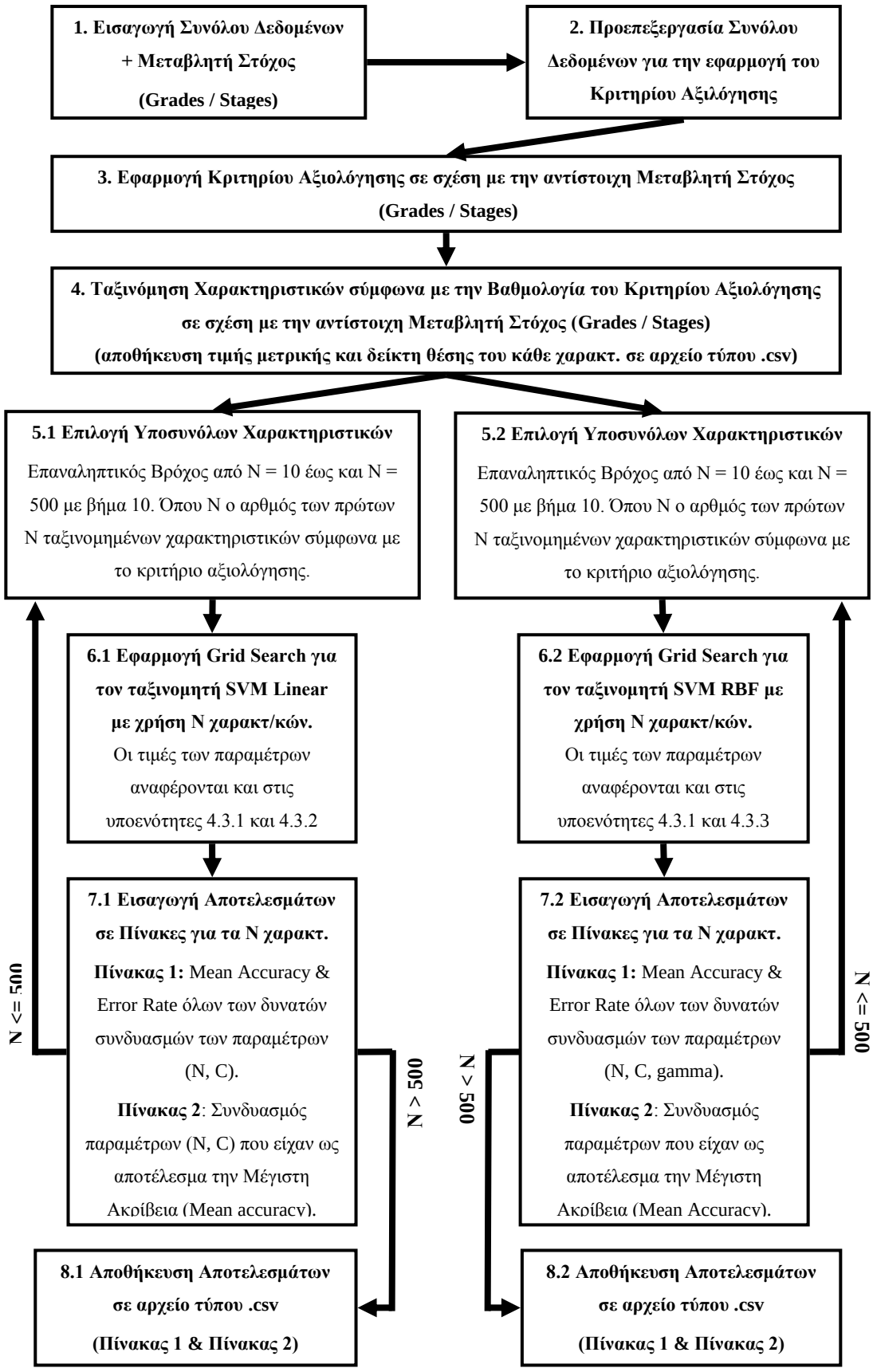
Για κάθε υποσύνολο χαρακτηριστικών, εισαγωγή αποτελεσμάτων σε δύο πίνακες. Στον πρώτο πίνακα εισάγεται ο μέσος όρος της ακρίβειας (mean accuracy) των 20 συνόλων ελέγχου και ο μέσος όρος του βαθμού σφάλματος (error rate) των 20 συνόλων ελέγχου, για όλους τους συνδυασμούς των παραμέτρων. Στον δεύτερο πίνακα εισάγεται ο υψηλότερος μέσος όρος της ακρίβειας (mean accuracy) των 20 συνόλων ελέγχου και ο αντίστοιχος μέσος όρος του βαθμού σφάλματος (error rate) των 20 συνόλων ελέγχου.

Βήμα 8.1 - 8.2

Μετά από την ολοκλήρωση του επαναληπτικού βρόχου, αποθήκευση όλων των αποτελεσμάτων, που περιέχονται στους δύο πίνακες του βήματος 7, σε αρχεία τύπου .csv.

Η παραπάνω μεθοδολογία αποτυπώνεται στον κώδικα που συντάχθηκε και υπάρχει στα παραρτήματα. Σε αυτό το σημείο είναι σημαντικό να αναφέρουμε ότι στα παραρτήματα για κάθε κριτήριο αξιολόγησης των χαρακτηριστικών, υπάρχει ο κώδικας που συντάχθηκε για την εφαρμογή του 1^{ου} σταδίου και του 2^{ου} σταδίου της μεθοδολογίας με χρήση του εκτιμητή SVM Linear για την μεταβλητή στόχος Grades (π.χ. παράρτημα 9.2.1). Αντίστοιχα, υπάρχει ο κώδικας που συντάχθηκε για την εφαρμογή του 1^{ου} σταδίου και του 2^{ου} σταδίου της μεθοδολογίας με χρήση του εκτιμητή SVM RBF για την μεταβλητή στόχος Stages (π.χ. παράρτημα 9.2.2). Οι συνδυασμοί των 2 σταδίων των 2 παραπάνω παραρτημάτων (π.χ. 9.2.1 και 9.2.2) αποτελούν τον κώδικα για την εφαρμογή του εκτιμητή SVM Linear για την μεταβλητή στόχος Stages και για την εφαρμογή του εκτιμητή SVM RBF για την μεταβλητή στόχος Grades.

Η μοναδική διαφοροποίηση που υπάρχει στην μεθοδολογία της εικόνας 5, είναι στην εκτέλεση του Βήματος 5 (5.1 - 5.2) για τα αποτελέσματα του κριτηρίου αξιολόγησης mRMR MIQ. Ο επαναληπτικός βρόχος σε αυτή την περίπτωση εκτελείται από N = 5 έως και N = 100 με βήμα 5, καθώς όπως προαναφέρθηκε στην προηγούμενη υποενότητα, καταφέραμε να συλλέξουμε τα 100 χαρακτηριστικά με την καλύτερη αξιολόγηση.



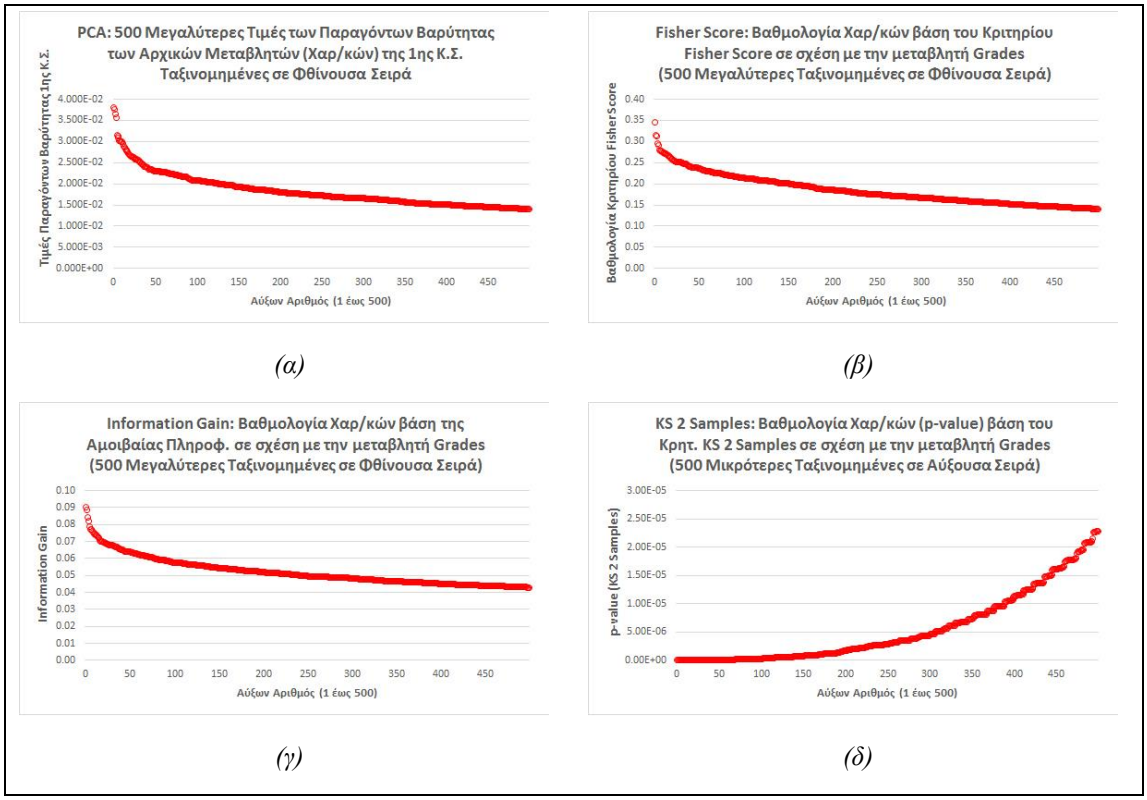
Εικόνα 5. *Μεθοδολογία εκτέλεσης πειραμάτων*

5.3 Αποτελέσματα

5.3.1 Αξιολόγηση Χαρακτηριστικών (Γονιδίων)

5.3.1.1 Grades

Μετά από την ολοκλήρωση της αξιολόγησης των χαρακτηριστικών σε σχέση με την μεταβλητή στόχος Grades, ταξινομήσαμε τα χαρακτηριστικά (γονίδια) σύμφωνα με την βαθμολογία που λάβανε για κάθε κριτήριο αξιολόγησης. Στα διαγράμματα της εικόνας 6 απεικονίζονται οι πρώτες 500 καλύτερες βαθμολογίες χαρακτηριστικών για το κάθε κριτήριο αξιολόγησης. Αντίστοιχα, στον πίνακα 7 παρουσιάζονται ενδεικτικά οι πρώτες 20 καλύτερες βαθμολογίες και η 500η, για την διευκόλυνση της κατανόησης των αντίστοιχων διαγραμμάτων.



Εικόνα 6. *500 καλύτερες τιμές αξιολόγησης χαρακτηριστικών (γονιδίων) σε σχέση με την μεταβλητή στόχος Grades. (α) PCA: τιμές των παραγόντων βαρύτητας των αρχικών μεταβλητών της 1ης Κ.Σ. (β) Fisher: τιμές κριτηρίου Fisher Score (γ) Αμοιβαία Πληροφορία: τιμές του Κέρδους της Πληροφορίας (δ) Kolmogorov-Smirnov 2 Samples: τιμές της στατιστικής τιμής του κριτηρίου p-value*

A/A	PCA	Fisher Score	Inform. Gain	KS 2 Samples
	weight	**f-score**	**IG**	**p-value**
1	3.808E-02	0.3463	0.0900	1.147E-12
2	3.756E-02	0.3144	0.0886	1.067E-11
3	3.661E-02	0.3136	0.0840	7.265E-11
4	3.564E-02	0.2956	0.0822	2.485E-10
5	3.152E-02	0.2903	0.0785	4.043E-10
6	3.126E-02	0.2804	0.0772	4.103E-10
7	3.082E-02	0.2787	0.0771	4.596E-10
8	3.026E-02	0.2780	0.0760	7.466E-10
9	3.009E-02	0.2761	0.0755	8.189E-10
10	3.005E-02	0.2731	0.0742	1.704E-09
11	2.994E-02	0.2725	0.0741	2.173E-09
12	2.947E-02	0.2722	0.0740	2.400E-09
13	2.882E-02	0.2718	0.0736	2.664E-09
14	2.860E-02	0.2705	0.0732	2.997E-09
15	2.831E-02	0.2698	0.0726	5.849E-09
16	2.795E-02	0.2678	0.0723	6.627E-09
17	2.768E-02	0.2664	0.0708	9.272E-09
18	2.756E-02	0.2624	0.0702	1.054E-08
19	2.720E-02	0.2615	0.0698	1.059E-08
20	2.695E-02	0.2580	0.0698	1.155E-08
…	…	…	…	…
…	…	…	…	…
500	1.397E-02	0.1408	0.0429	2.275E-05

Πίνακας 7. *Καλύτερες τιμές αξιολόγησης χαρακτηριστικών (γονιδίων) σε σχέση με την μεταβλητή στόχος Grades.*

Στην περίπτωση αξιολόγησης των χαρακτηριστικών σύμφωνα με τις τιμές των παραγόντων βαρύτητας των αρχικών μεταβλητών (χαρακτηριστικών) της 1ης Κ.Σ. (PCA), καλύτερες τιμές θεωρούνται οι μεγαλύτερες. Όσο μεγαλύτερη είναι η τιμή ενός παράγοντα βαρύτητας, τόσο περισσότερο το αντίστοιχο χαρακτηριστικό (αρχική μεταβλητή) επηρεάζει την εξαγωγή της 1ης Κ.Σ. Η υψηλότερη τιμή που παρατηρείται στους παράγοντες βαρύτητας είναι 3.808E-02. Στο αντίστοιχο διάγραμμα (6α), παρατηρούμε ότι η καμπύλη των τιμών εμφανίζει αρχικά μία απότομη κλίση προς τα κάτω έως περίπου και την 50η τιμή και στην συνέχεια η καμπύλη συνεχίζει την καθοδική της πορεία αλλά με εμφανή μικρότερη κλίση. Η κάθε τιμή αντιστοιχεί και σε ένα χαρακτηριστικό (γονίδιο), οπότε περίπου 50 από τα 12604 χαρακτηριστικά δείχνουν να επηρεάζουν περισσότερο την εξαγωγή της 1ης Κ.Σ.. Αυτό συνεπάγεται ότι αυτά τα χαρακτηριστικά διαθέτουν σχετικά περισσότερη πληροφορία, οπότε ίσως να είναι καθοριστικά για τον προσδιορισμό των τιμών της μεταβλητής στόχου Grades και των τιμών της μεταβλητής στόχου Stages.

Στην αξιολόγηση των χαρακτηριστικών με το κριτήριο Fisher, καλύτερες τιμές είναι οι υψηλότερες. Όσο πιο μεγάλη είναι η τιμή του Fisher Score ενός χαρακτηριστικού, σημαίνει οτι η πληροφορία που περιέχει διαφοροποιείται πιο πολύ μεταξύ των δύο κλάσεων σε σχέση με ένα χαρακτηριστικό που έχει μικρότερη τιμή. Παρατηρώντας το αντίστοιχο διάγραμμα (6β), φαίνεται να ξεχωρίζουν οι πρώτες 20 με 25 τιμές, οι οποίες είναι μεγαλύτερες από 0.25, με την υψηλότερη να είναι 0.3464. Στην συνέχεια μέχρι και την 500η τιμή παρατηρείται μία καθοδική τάση των τιμών φτάνοντας έως 0,1408. Τα 20 με 25 χαρακτηριστικά που αντιστοιχούν στις πρώτες 20 με 25 υψηλότερες τιμές δείχνουν να περιέχουν πληροφορία η οποία διαφοροποιείται πιο πολύ μεταξύ των δύο κλάσεων σε σχέση με τα υπόλοιπα χαρακτηριστικά. Αυτό συνεπάγεται με την ύπαρξη μεγάλης πιθανότητας αυτά τα χαρακτηριστικά να καθορίζουν σε μεγάλο βαθμό την τιμή της μεταβλητής στόχου Grades.

Καλύτερες τιμές του κέρδους της πληροφορίας είναι οι υψηλότερες. Οι τιμές κέρδους της πληροφορίας προέρχονται από την ασάφεια που υπάρχει για ένα χαρακτηριστικό μείον της ασάφειας του χαρακτηριστικού με δεδομένη την τιμή της μεταβλητής κλάσης Grades. Παρατηρώντας το αντίστοιχο διάγραμμα (6γ) και τις τιμές του πίνακα 7 φαίνεται να ξεχωρίζουν οι πρώτες 18 τιμές, οι οποίες είναι μεγαλύτερες από 0.07, με την υψηλότερη να είναι 0.09. Στην συνέχεια μέχρι και την 500η τιμή συνεχίζεται η καθοδική τάση των τιμών με μικρότερη διαβάθμιση φτάνοντας έως 0.0429. Συνεπώς, τα 18 χαρακτηριστικά που αντιστοιχούν στις πρώτες 18 υψηλότερες τιμές μπορεί να καθορίζουν την τιμή της μεταβλητής Grades.

Στην αξιολόγηση των χαρακτηριστικών με το κριτήριο Kolmogorov-Smirnov 2 Samples, καλύτερες είναι οι μικρότερες τιμές της στατιστικής τιμής p-value. Όσο μικρότερη είναι η τιμή του p-value σημαίνει ότι τα δύο δείγματα (1ο δείγμα: τιμές χαρακτηριστικού για την κλάση Early_Grade, 2ο δείγμα: τιμές του χαρακτηριστικού για την κλάση Late_Grade) προέρχονται από πληθυσμούς με διαφορετικές κατανομές. Συνεπώς, το χαρακτηριστικό με μικρή p-value διαφοροποιείται αρκετά στις δύο τιμές της μεταβλητής στόχου Grades. Παρατηρώντας το αντίστοιχο διάγραμμα (6δ) και τον πίνακα 7, ξεκινώντας από τιμή (μικρότερη) 1.147E-12, φαίνεται να ξεχωρίζουν οι πρώτες 140 τιμές, οι οποίες συγκλίνουν αρκετά στο 0 σε σχέση με τις υπόλοιπες. Συνεπώς, για το υποσύνολο των χαρακτηριστικών που αντιστοιχούν σε αυτές τις τιμές, υπάρχει η πιθανότητα να διαθέτουν την ικανότητα διαχωρισμού της μεταβλητής κλάσης Grades.

Για τα κριτήρια αξιολόγησης ReliefF και mRMR MIQ, τα αντίστοιχα πακέτα λογισμικού που χρησιμοποιήσαμε επιστρέφουν μόνο τους δείκτες θέσης των χαρακτηριστικών, ταξινομημένα με την καλύτερη αντίστοιχη βαθμολογία.

A/A	PCA	Fisher Score	Inform. Gain	KS 2 Samples	ReliefF	mRMR MIQ
1	10263	3678	1706	1706	3046	1706
2	260	10401	100	10401	3877	8858
3	10255	86	7084	86	3834	3962
4	2096	10241	7442	3678	10121	5333
5	5383	120	3962	2668	1752	8853
6	6336	1706	5014	3743	2038	7259
7	10274	7084	4483	100	4077	3480
8	11487	1731	1731	7084	2316	10820
9	8031	5621	2897	10311	6055	1907
10	6923	1158	10401	2334	10039	6229
11	4302	2897	8890	10241	6898	12293
12	6652	1918	811	1731	4929	7084
13	10736	3669	3678	1740	9806	811
14	9533	9790	7503	2897	7149	3786
15	11316	7838	10311	5014	2315	9065
16	2778	7442	2853	9910	9335	77
17	8663	1365	3676	140	345	2930
18	7650	119	10894	3669	7347	3573
19	3444	4688	6638	9994	7623	9458
20	9844	3957	1326	5008	7553	350
…	…	…	…	…	…	…
…	…	…	…	…	…	…
500	7546	1335	2105	404	4553	--

Πίνακας 8. *Δείκτες θέσης χαρακτηριστικών (γονιδίων) ταξινομημένοι σύμφωνα με την καλύτερη βαθμολογία του κάθε κριτηρίου σε σχέση με την μεταβλητή στόχος Grades*

Στον πίνακα 8 παρουσιάζονται ενδεικτικά οι δείκτες θέσης των 20 χαρακτηριστικών (γονιδίων) ταξινομημένοι σύμφωνα με την καλύτερη βαθμολογία του εκάστοτε κριτηρίου αξιολόγησης, σε σχέση με την μεταβλητή στόχος Grades. Επιπλέον στον πίνακα 9 παρουσιάζεται ο αριθμός των κοινών χαρακτηριστικών ανά ζεύγη κριτηρίων αξιολόγησης (σε σχέση με την μεταβλητή στόχος Grades) μεταξύ των πρώτων ‘καλύτερων’ χαρακτηριστικών.

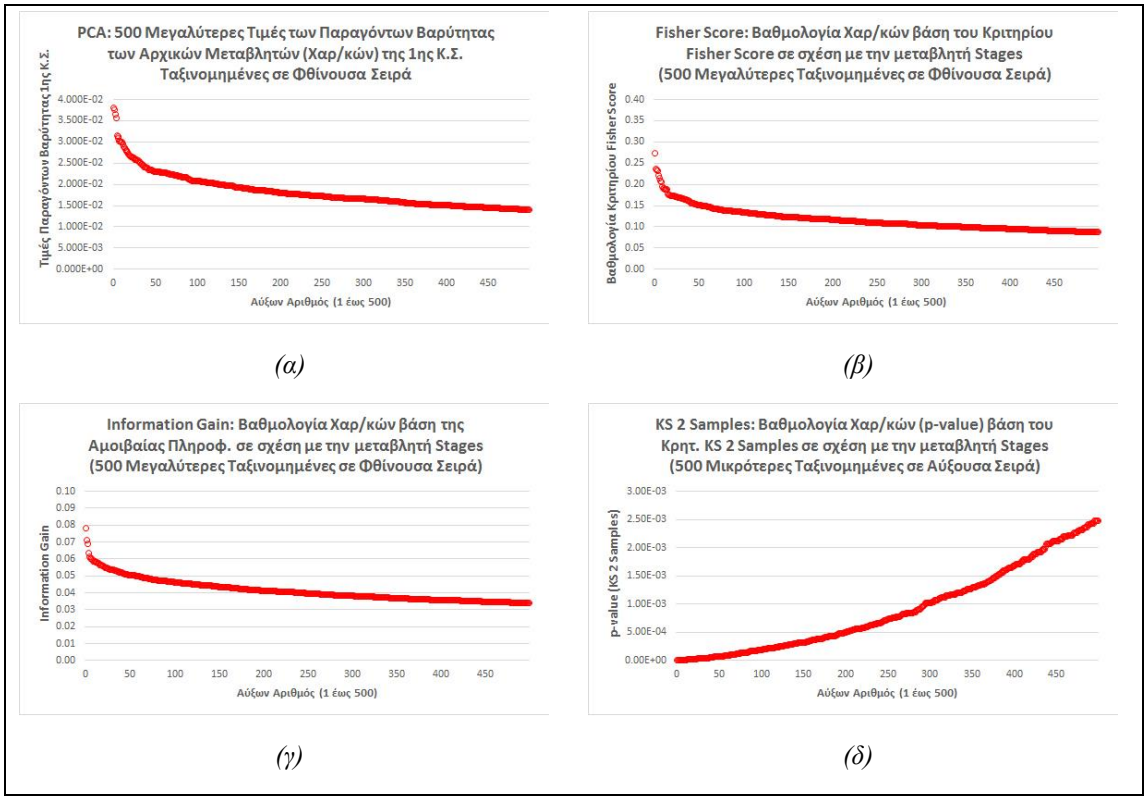
Ο μεγαλύτερος αριθμός κοινών χαρακτηριστικών παρατηρείται σε 3 ζεύγη (κόκκινα κελιά του πίνακα 9). Συγκεκριμένα στα ζεύγη των κριτηρίων αξιολόγησης Kolmogorv-Smirnov 2 Samples & Information Gain, Kolmogorv-Smirnov 2 Samples & Fisher Score και στο ζεύγος Information Gain & Fisher Score. Το ζεύγος των κριτηρίων αξιολόγησης Kolmogorv-Smirnov 2 Samples & Fisher Score έχει μεγαλύτερο αριθμό κοινών χαρακτηριστικών μεταξύ των περισσότερων υποσυνόλων. Επιπλέον παρατηρούμε ότι το χαρακτηριστικό με δείκτη θέσης 1706, βρίσκεται στην πρώτη εξάδα σε 4 κριτήρια αξιολόγησης, από τα οποία στα 3 από αυτά κατέχει την πρώτη θέση.

Αριθμός Γονιδίων	1. KS 2 Samples & Inform. Gain	2. KS 2 Samples & Fisher Score	3. KS 2 Samples & PCA	4. KS 2 Samples & ReliefF	5. Inform. Gain & Fisher Score	6. Inform. Gain & PCA	7. Inform. Gain & ReliefF	8. Fisher Score & PCA	9. Fisher Score & ReliefF	10. PCA & ReliefF	11. mRMR MIQ & PCA	12. mRMR MIQ & Fisher Score	13. mRMR MIQ & Inform. Gain	14. mRMR MIQ & KS 2 Samples	15. mRMR MIQ & ReliefF
10	4	5	0	0	4	0	0	0	0	0	0	1	2	1	0
20	9	9	0	0	7	0	0	0	0	0	0	2	4	2	0
30	14	12	0	0	8	0	0	0	0	1	0	2	4	2	0
40	16	16	0	1	17	0	0	1	0	0	0	2	5	3	0
50	22	23	0	1	23	1	1	1	0	0	0	5	7	7	0
60	27	29	0	2	28	1	2	2	1	0	0	9	10	10	0
70	31	31	1	2	32	1	1	3	0	0	0	10	11	11	0
80	41	36	2	2	38	1	3	3	1	0	0	10	12	13	0
90	47	44	2	1	43	1	3	4	1	0	0	12	16	15	0
100	52	51	2	1	50	5	2	6	0	1	0	12	20	16	0
110	56	58	2	2	56	7	3	8	3	0					
120	64	66	2	1	65	8	3	11	1	1					
130	70	73	3	1	74	11	3	15	2	0					
140	74	78	3	1	78	11	1	17	3	1					
150	77	86	5	3	86	12	2	18	3	3					
160	83	92	7	3	89	12	4	19	5	3					
170	90	99	7	0	96	15	1	22	1	8					
180	94	105	10	2	102	17	4	26	4	6					
190	102	113	12	3	111	18	3	28	4	7					
200	108	121	14	4	117	23	3	30	5	7					

Πίνακας 9. *Αριθμός κοινών χαρακτηριστικών (γονιδίων) ανά ζεύγη κριτηρίων αξιολόγησης (σε σχέση με την μεταβλητή στόχος Grades) μεταξύ των πρώτων 'καλύτερων' χαρακτηριστικών. (κόκκινα κελιά: μεγαλύτερος αριθμός κοιών γονιδίων)*

5.3.1.2 Stages

Αντίστοιχα, μετά από την ολοκλήρωση της αξιολόγησης των χαρακτηριστικών σε σχέση με την μεταβλητή στόχος Stages, ταξινομήσαμε τα χαρακτηριστικά (γονίδια) σύμφωνα με την βαθμολογία που λάβανε για κάθε κριτήριο αξιολόγησης. Στα διαγράμματα της εικόνας 7 απεικονίζονται οι πρώτες 500 καλύτερες βαθμολογίες χαρακτηριστικών για το κάθε κριτήριο αξιολόγησης. Στον πίνακα 10 παρουσιάζονται ενδεικτικά οι πρώτες 20 καλύτερες βαθμολογίες και η 500^{η}, για την διευκόλυνση της κατανόησης των αντίστοιχων διαγραμμάτων.



Εικόνα 7. *500 καλύτερες τιμές αξιολόγησης χαρακτηριστικών (γονιδίων) σε σχέση με την μεταβλητή στόχος Stages. (α) PCA: τιμές των παραγόντων βαρύτητας των αρχικών μεταβλητών της 1ης Κ.Σ. (β) Fisher: τιμές κριτηρίου Fisher Score (γ) Αμοιβαία Πληροφορία: τιμές του Κέρδους της Πληροφορίας (δ) Kolmogorov-Smirnov 2 Samples: τιμές της στατιστικής τιμής του κριτηρίου p-value*

Η αξιολόγηση των χαρακτηριστικών σύμφωνα με τις τιμές των παραγόντων βαρύτητας των αρχικών μεταβλητών (χαρακτηριστικών) της 1ης Κ.Σ. (PCA), είναι η ίδια με την αξιολόγηση των χαρακτηριστικών σε σχέση με την μεταβλητή στόχος Grades, καθώς η μέθοδος PCA δεν χρησιμοποιεί την μεταβλητή στόχος για την εξαγωγή χαρακτηριστικών.

Στην αξιολόγηση των χαρακτηριστικών με το κριτήριο Fisher, όπως προαναφέρθηκε στην προηγούμενη ενότητα καλύτερες τιμές είναι οι υψηλότερες. Παρατηρώντας το αντίστοιχο διάγραμμα (7β), φαίνεται να ξεχωρίζουν οι πρώτες 25 τιμές, οι οποίες είναι μεγαλύτερες από 0.17, με την υψηλότερη να είναι 0.273. Στην συνέχεια μέχρι και την 500η τιμή παρατηρείται μία καθοδική τάση των τιμών φτάνοντας έως 0,0873. Τα 25 χαρακτηριστικά που αντιστοιχούν στις πρώτες 25 υψηλότερες τιμές δείχνουν να περιέχουν πληροφορία η οποία διαφοροποιείται πιο πολύ μεταξύ των δύο κλάσεων σε σχέση με τα υπόλοιπα χαρακτηριστικά. Αυτό συνεπάγεται με την ύπαρξη μεγάλης πιθανότητας αυτά τα χαρακτηριστικά να καθορίζουν σε μεγάλο βαθμό την τιμή της μεταβλητής στόχου Stages.

Όπως προαναφέρθηκε, καλύτερες τιμές του κέρδους της πληροφορίας είναι οι υψηλότερες. Παρατηρώντας το αντίστοιχο διάγραμμα (7γ) και τις τιμές του πίνακα 10 φαίνεται να ξεχωρίζουν οι πρώτες 7 τιμές, οι οποίες είναι μεγαλύτερες από 0.06, με την υψηλότερη να

είναι 0.078. Στην συνέχεια μέχρι και την 500η τιμή συνεχίζεται η καθοδική τάση των τιμών με μικρότερη διαβάθμιση φτάνοντας έως 0.0339. Συνεπώς, για το υποσύνολο των 7 χαρακτηριστικών που αντιστοιχούν στις πρώτες 7 υψηλότερες τιμές, υπάρχει η πιθανότητα να διαθέτουν την ικανότητα καθορισμού της τιμής της μεταβλητής Stages.

Στην αξιολόγηση των χαρακτηριστικών με το κριτήριο Kolmogorov-Smirnov 2 Samples, όπως προαναφέρθηκε, καλύτερες είναι οι μικρότερες τιμές της στατιστικής τιμής p-value. Παρατηρώντας το αντίστοιχο διάγραμμα (7δ) και τον πίνακα 10, ξεκινώντας από τιμή (μικρότερη) 2.303E-07, φαίνεται να ξεχωρίζουν οι πρώτες 40 τιμές, οι οποίες συγκλίνουν αρκετά στο 0 σε σχέση με τις υπόλοιπες. Συνεπώς, για το υποσύνολο των χαρακτηριστικών που αντιστοιχούν στις 40 μικρότερες τιμές της p-value, υπάρχει η πιθανότητα να διαθέτουν την ικανότητα διαχωρισμού της μεταβλητής κλάσης Stages.

A/A	**PCA**	**Fisher Score**	**Inform. Gain**	**KS 2 Samples**
	weight	**f-score**	**IG**	**p-value**
1	3.808E-02	0.2730	0.0780	2.303E-07
2	3.756E-02	0.2359	0.0713	2.574E-07
3	3.661E-02	0.2349	0.0687	1.399E-06
4	3.564E-02	0.2310	0.0634	2.351E-06
5	3.152E-02	0.2210	0.0612	2.433E-06
6	3.126E-02	0.2145	0.0601	4.764E-06
7	3.082E-02	0.2078	0.0600	4.784E-06
8	3.026E-02	0.2069	0.0599	5.767E-06
9	3.009E-02	0.1952	0.0587	6.291E-06
10	3.005E-02	0.1913	0.0584	7.147E-06
11	2.994E-02	0.1902	0.0584	9.169E-06
12	2.947E-02	0.1888	0.0582	1.036E-05
13	2.882E-02	0.1885	0.0581	1.318E-05
14	2.860E-02	0.1884	0.0578	1.440E-05
15	2.831E-02	0.1875	0.0576	1.578E-05
16	2.795E-02	0.1781	0.0575	1.722E-05
17	2.768E-02	0.1750	0.0566	1.841E-05
18	2.756E-02	0.1743	0.0566	1.841E-05
19	2.720E-02	0.1739	0.0563	1.893E-05
20	2.695E-02	0.1735	0.0558	2.016E-05
…	…	…	…	…
…	…	…	…	…
500	1.397E-02	0.0873	0.0339	2.469E-03

Πίνακας 10. *Καλύτερες τιμές αξιολόγησης χαρακτηριστικών (γονιδίων) σε σχέση με την μεταβλητή στόχος Stages.*

Στον πίνακα 11 παρουσιάζονται ενδεικτικά οι δείκτες θέσης των 20 χαρακτηριστικών (γονιδίων) ταξινομημένοι σύμφωνα με την καλύτερη βαθμολογία του εκάστοτε κριτηρίου αξιολόγησης, σε σχέση με την μεταβλητή στόχος Stages. Επιπλέον στον πίνακα 12 παρουσιάζεται ο αριθμός των κοινών χαρακτηριστικών ανά ζεύγη κριτηρίων αξιολόγησης (σε σχέση με την μεταβλητή στόχος Stages) μεταξύ των πρώτων 'καλύτερων' χαρακτηριστικών.

Ο μεγαλύτερος αριθμός κοινών χαρακτηριστικών (κόκκινα κελιά πίνακα 12) παρατηρείται στο ζεύγος των κριτηρίων αξιολόγησης Kolmogorv-Smirnov 2 Samples & Fisher Score. Τους αμέσως επόμενους μεγαλύτερους αριθμούς κοινών χαρακτηριστικών έχουν τα ζεύγη των κριτηρίων αξιολόγησης Kolmogorv-Smirnov 2 Samples & Information Gain και Information Gain & Fisher Score. Τέλος, παρατηρούμε ότι το χαρακτηριστικό με δείκτη θέσης 7707, βρίσκεται στην πρώτη τριάδα σε 4 κριτήρια αξιολόγησης, από τα οποία στα 3 από αυτά κατέχει την πρώτη θέση.

A/A	PCA	Fisher Score	Inform. Gain	KS 2 Samples	ReliefF	mRMR MIQ
1	10263	7707	7707	7707	3877	2890
2	260	2314	1034	9613	1807	11023
3	10255	10495	765	9806	1752	7707
4	2096	9847	3739	10390	10121	6852
5	5383	10390	8243	369	11604	11464
6	6336	369	7974	2894	6055	1014
7	10274	1034	2317	4529	3834	11997
8	11487	3105	1641	1316	327	10700
9	8031	7243	9608	6502	10039	8029
10	6923	2312	2193	3105	10860	9644
11	4302	5217	2288	5143	1420	1581
12	6652	4529	6852	100	31	7505
13	10736	9806	0	1034	2040	10792
14	9533	10584	8890	9744	59	6288
15	11316	10494	2662	4586	582	4339
16	2778	5014	1014	10495	117	232
17	8663	8265	573	9472	1471	4779
18	7650	2313	3733	7243	5804	11299
19	3444	10019	4529	2288	10262	2267
20	9844	2315	8031	7149	12481	12
…	…	…	…	…	…	…
…	…	…	…	…	…	…
500	7546	9603	144	8104	4563	--

Πίνακας 11. *Δείκτες θέσης χαρακτηριστικών (γονιδίων) ταξινομημένοι σύμφωνα με την καλύτερη βαθμολογία του κάθε κριτηρίου σε σχέση με την μεταβλητή στόχος Stages.*

Αριθμός Γονιδίων	1. KS 2 Samples & Inform. Gain	2. KS 2 Samples & Fisher Score	3. KS 2 Samples & PCA	4. KS 2 Samples & ReliefF	5. Inform. Gain & Fisher Score	6. Inform. Gain & PCA	7. Inform. Gain & ReliefF	8. Fisher Score & PCA	9. Fisher Score & ReliefF	10. PCA & ReliefF	11. mRMR MIQ & PCA	12. mRMR MIQ & Fisher Score	13. mRMR MIQ & Inform. Gain	14. mRMR MIQ & KS 2 Samples	15. mRMR MIQ & ReliefF
10	1	4	0	0	2	0	0	0	0	0	0	1	1	1	0
20	4	9	0	0	3	1	0	0	0	0	0	1	3	1	0
30	4	13	0	1	5	1	0	0	0	1	0	1	5	1	0
40	8	20	1	2	8	1	0	0	2	0	0	1	7	1	0
50	9	25	1	1	13	1	1	0	1	1	0	1	10	2	0
60	15	28	2	2	17	1	1	0	4	0	0	2	11	2	1
70	18	34	4	2	21	2	2	1	2	1	0	3	13	3	1
80	19	35	4	1	24	2	2	1	1	1	0	3	20	4	0
90	22	40	4	1	28	2	1	1	2	1	0	5	23	4	1
100	26	44	4	2	33	4	2	2	2	2	0	6	25	6	1
110	32	50	5	3	37	4	4	2	4	1					
120	35	54	7	4	37	4	5	2	3	0					
130	38	61	7	4	43	5	7	4	3	1					
140	41	68	8	4	50	5	5	4	4	3					
150	47	78	10	2	57	6	4	4	1	4					
160	52	83	10	6	61	6	10	4	2	3					
170	56	90	11	4	69	6	8	4	2	5					
180	62	98	11	3	77	6	7	4	4	5					
190	65	103	13	5	81	7	11	4	5	6					
200	66	109	14	5	85	7	11	5	5	7					

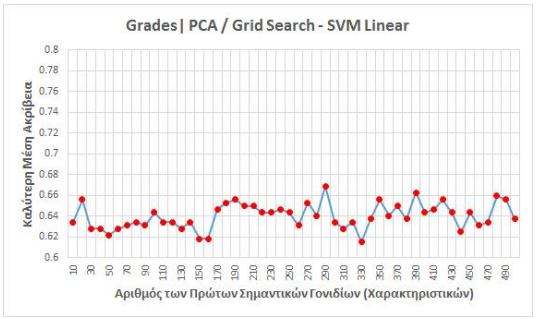
Πίνακας 12. *Αριθμός κοινών χαρακτηριστικών (γονιδίων) ανά ζεύγη κριτηρίων αξιολόγησης (σε σχέση με την μεταβλητή στόχος Stages) μεταξύ των πρώτων 'καλύτερων' χαρακτηριστικών. (κόκκινα κελιά: μεγαλύτερος αριθμός κοιών γονιδίων)*

5.3.2 SVM

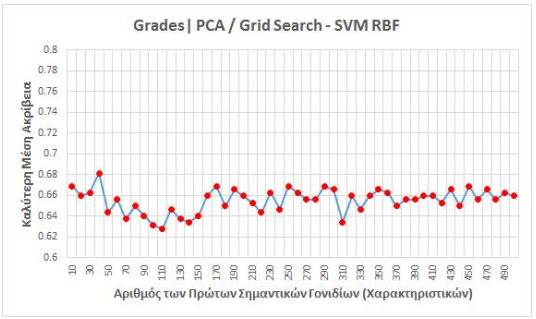
5.3.2.1 Grades

Σύμφωνα με την μεθοδολογία που ακολουθήσαμε, στην εικόνα 8 απεικονίζονται σε διαγράμματα τα τελικά αποτελέσματα της εξαντλητικής αναζήτησης για τους ταξινομητές SVM με γραμμικό πυρήνα και με πυρήνα RBF με μεταβλητή στόχο την Grades. Συγκεκριμένα σε κάθε διάγραμμα παρουσιάζονται για κάθε επιλεγμένο υποσύνολο (το οποίο προέρχεται από το εκάστοτε κριτήριο αξιολόγησης), οι υψηλότερες τιμές της μέσης ακρίβειας

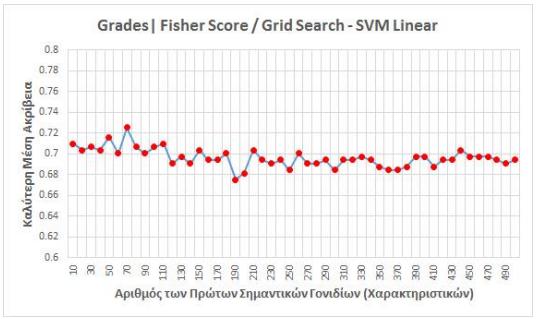
των 20 συνόλων ελέγχου, οι οποίες προέρχονται από διαφορετικούς συνδυασμούς παραμέτρων (C, gamma). Επιπλέον, στον πίνακα 13 παρουσιάζονται για κάθε διάγραμμα της εικόνας 8, η υψηλότερη μέση ακρίβεια των 20 συνόλων ελέγχου και οι αντίστοιχοι παράμετροι με τις οποίες επιτεύχθηκε αυτή.



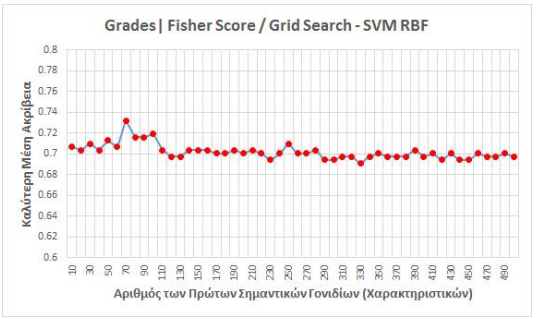
(α1)



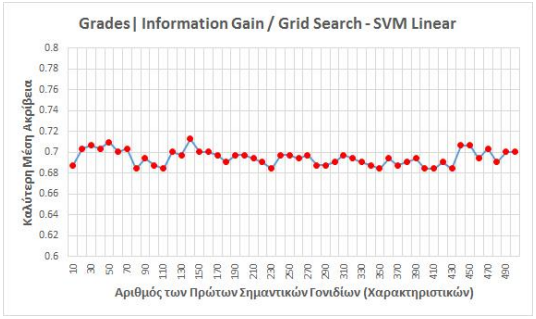
(β1)



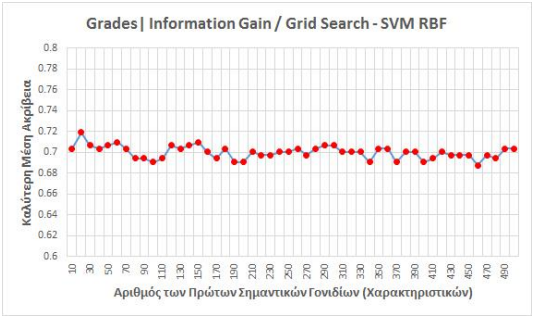
(α2)



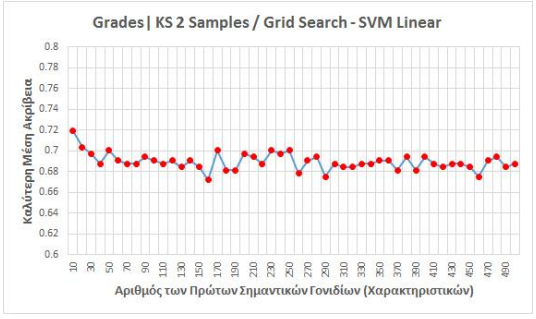
(β2)



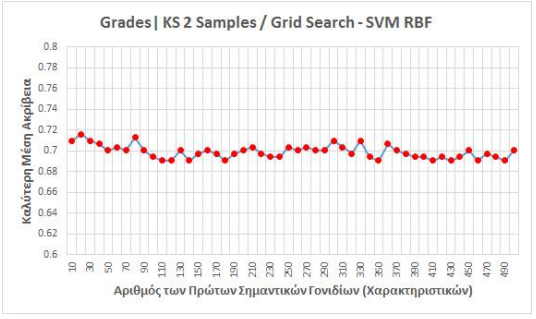
(α3)



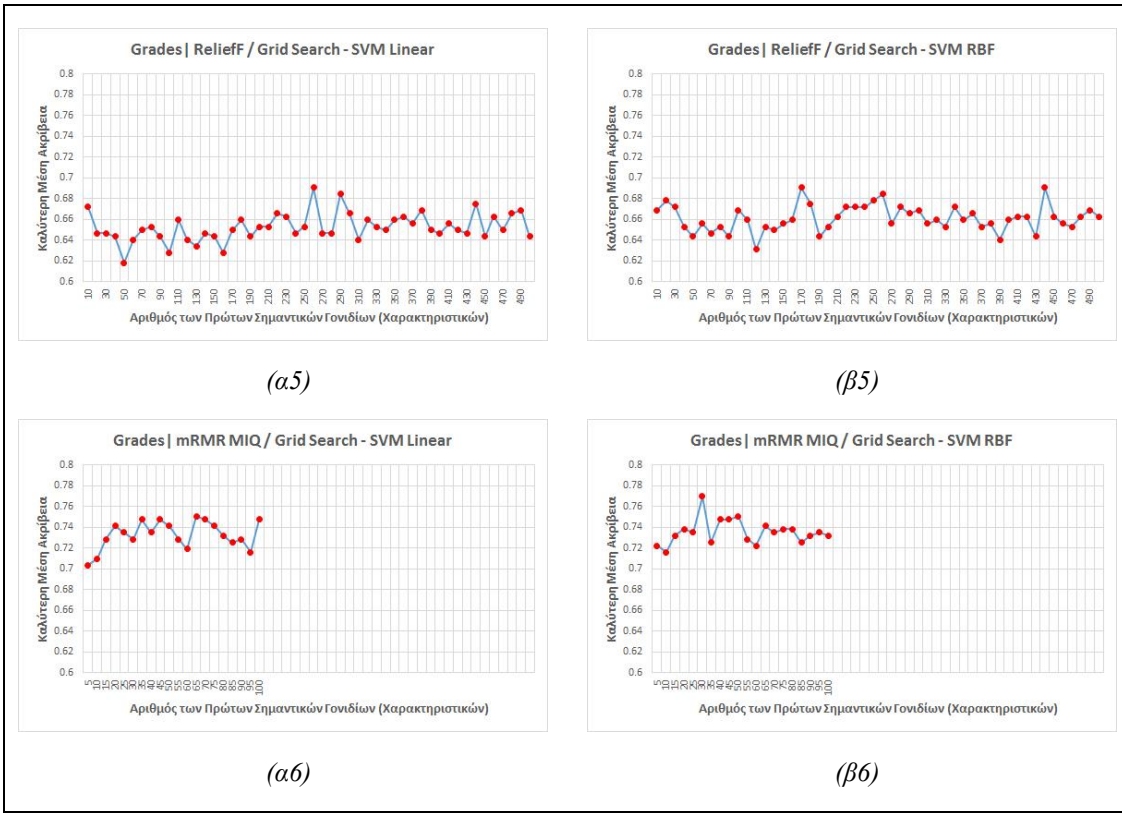
(β3)



(α4)



(β4)



Εικόνα 8. *Υψηλότερες τιμές (%) Μέσης Ακρίβειας (20 συνόλων ελέγχου) από την εκτέλεση Εξαντλητικής Αναζήτησης (Grid Search) με χρήση των ταξινομητών (α) SVM Linear και (β) SVM RBF για την μεταβλητή στόχος Grades. Τα υποσύνολα των χαρακτηριστικών (γονιδίων) προήλθαν από την αξιολόγηση των χαρακτηριστικών με βάση τα κριτήρια (α1, β1) PCA, (α2, β2) Fisher Score, (α3, β3) Mutual Information (Inform. Gain), (α4, β4) KS 2Samples (p-value), (α5, β5) ReliefF, (α6, β6) mRMR MIQ*

Διάγ/μα Εικ. 8	**Κριτήριο**	**Ταξινομητής**	**Μέση Ακρίβεια**	**Παράμετροι**		**Αριθμός Γονιδίων**
				C	**gamma**	
α1	PCA	SVM Linear	66.88%	0.01	-	290
α2	Fisher Score		72.56%	0.1	-	70
α3	Inform. Gain		71.29%	0.1	-	140
α4	KS 2 Samples		71.92%	1000	-	10
α5	ReliefF		69.09%	0.01	-	260
α6	**mRMR MIQ**		**75.08%**	**0.1**	**-**	**65**
β1	PCA	SVM RBF	68.14%	1	0.01	40
β2	Fisher Score		73.19%	10000	0.00001	70
β3	Inform. Gain		71.92%	1	0.01	20
β4	KS 2 Samples		71.61%	1	0.01	20
β5	ReliefF		69.09%	100	0.0001	170
				1000	0.00001	440
β6	**mRMR MIQ**		**76.97%**	**10000**	**0.001**	**30**

Πίνακας 13. *Για κάθε διάγραμμα της Εικόνας 8, η υψηλότερη τιμή Μέσης Ακρίβειας (20 συνόλων ελέγχου) & οι αντίστοιχοι παράμετροι από τις οποίες προήλθε αυτή. (μεταβλητή στόχος Grades)*

Στα διαγράμματα της εικόνας 8, παρατηρούμε ότι οι τιμές της μέσης ακρίβειας των ταξινομητών με χρήση υποσυνόλων χαρακτηριστικών (γονιδίων) που προήλθαν από τα κριτήρια αξιολόγησης PCA (βάρη παραγόντων βαρύτητας 1ης Κ.Σ.) (Εικ.8 (α1)(β1)) και ReliefF (Εικ.8 (α5)(β5)), κυμαίνονται στις χαμηλότερες τιμές. Συγκεκριμένα, οι τιμές της μέσης ακρίβειας των ταξινομητών SVM Linear και SVM RBF (Εικ.8 (α1)(β1)) για τα υποσύνολα των χαρακτηριστικών που προήλθαν από τα βάρη των παραγόντων βαρύτητας της 1ης Κ.Σ.(PCA), κυμαίνονται μεταξύ 61,00% - 68,14%. Αντίστοιχα οι τιμές της μέσης ακρίβειας των ταξινομητών SVM Linear και SVM RBF (Εικ.8 (α5)(β5)) για τα υποσύνολα των χαρακτηριστικών που προήλθαν από το κριτήριο ReliefF κυμαίνονται μεταξύ 61,50% - 69,09%.

Οι τιμές της μέσης ακρίβειας των ταξινομητών SVM Linear και SVM RBF για τα υποσύνολα χαρακτηριστικών που προήλθαν από τα κριτήρια αξιολόγησης Fisher Score (Εικ.8 (α2)(β2)), Mutual Information (Inform. Gain) (Εικ.8 (α3)(β3)) και Kolmogorov-Smirnov 2 Samples (Εικ.8 (α4)(β4)), δείχνουν να κυμαίνονται στις ίδιες περίπου τιμές. Συνολικά (και στις 3 περιπτώσεις), παρατηρούμε ότι η μέση ακρίβεια κυμαίνεται από 67,00% έως και 73.19%. Την υψηλότερη μέση ακρίβεια (73.19%) την επιτυγχάνει ο ταξινομητής SVM RBF με παραμέτρους C = 10000 και gamma = 0.00001 χρησιμοποιώντας τα πρώτα 70 ‘καλύτερα’ χαρακτηριστικά (γονίδια) σύμφωνα με το κριτήριο Fisher Score. Επιπλέον, με την χρήση του ίδιου υποσυνόλου χαρακτηριστικών (70 χαρακτ.) που προήλθαν από το κριτήριο Fisher Score, ο ταξινομητής SVM Linear με την παράμετρο C = 0.1, επιτυγχάνει την δεύτερη υψηλότερη μέση ακρίβεια (72,56%) μεταξύ των υπόλοιπων ταξινομητών που χρησιμοποιούν υποσύνολα χαρακτηριστικών βάση των κριτηρίων αξιολόγησης Mutual Information και KS 2 Samples.

Οι ταξινομητές των οποίων η μέση ακρίβεια υπερέχει, χρησιμοποιούν υποσύνολα χαρακτηριστικών βάση του κριτηρίου mRMR MIQ. Συγκεκριμένα, ο ταξινομητής SVM RBF με παραμέτρους C = 10000 και gamma = 0.001, χρησιμοποιώντας μόνο τα 30 ‘καλύτερα’ χαρακτηριστικά (γονίδια), επιτυγχάνει την συνολικά υψηλότερη μέση ακρίβεια 76.97%. Η δεύτερη συνολικά υψηλότερη μέση ακρίβεια (75,08), επιτυγχάνεται από τον ταξινομητή SVM Linear με παράμετρο C = 0.1, χρησιμοποιώντας τα 65 ‘καλύτερα’ χαρακτηριστικά (γονίδια) σύμφωνα και πάλι με το κριτήριο αξιολόγησης χαρακτηριστικών mRMR. Επίσης, είναι σημαντικό να αναφερθεί οτι οι χαμηλότερες τιμές της μέσης ακρίβειας των ταξινομητών SVM Linear (70.00%) και SVM RBF (71.00%) με χρήση υποσυνόλων χαρακτηριστικών βάση του κριτηρίου αξιολόγησης mRMR MIQ, είναι στις περισσότερες περιπτώσεις πολύ κοντά στις υψηλότερες τιμές μέσης ακρίβειας των ταξινομητών που χρησιμοποιούν υποσύνολα χαρακτηριστικών βάση των υπόλοιπων κριτηρίων αξιολόγησης.

Συνδυασμός Κριτηρίων (Τομή)	Ταξ/τής	Μέση Ακρίβεια	Παράμετροι		Αριθμός Κοινών Γονιδίων (Τομή Υποσυνόλων)	Αριθμ. Γονιδίων (Υποσύνολα από τα οποία προήλθε η Τομή)
			C	gamma		
KS 2 Samples & Inform. Gain	SVM Linear	71.92%	0.01	-	16	40
KS 2 Samples & Fisher Score		72.56%	10000	-	5	10
Inform. Gain & Fisher Score		71.92%	10000	-	8	30
KS 2 Samples & Inform. Gain	SVM RBF	71.61%	0.1	0.01	31	70
KS 2 Samples & Fisher Score		**73.82%**	**10000**	**0.0001**	**29**	**60**
Inform. Gain & Fisher Score		71.61%	1000	0.001	4	10

Πίνακας 14. *Υψηλότερη τιμή Μέσης Ακρίβειας (20 συνόλων ελέγχου) & οι αντίστοιχοι παράμετροι των ταξινομητών, από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh), με χρήση των κοινών χαρακτηριστικών (γονιδίων) ανά ζεύγος κριτηρίων αξιολόγησης (τομή υποσυνόλων). (μεταβλητή στόχος Grades)*

Στον πίνακα 14 παρουσιάζονται οι υψηλότερες τιμές μέσης ακρίβειας (20 συνόλων ελέγχου) από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh) για τους ταξινομητές SVM Linear και SVM RBF, με χρήση των κοινών χαρακτηριστικών (τομή υποσυνόλων) ανά ζεύγος κριτηρίων αξιολόγησης. Συγκεκριμένα, παρουσιάζονται τα ζεύγη των οποίων τα κοινά χαρακτηριστικά στα πρώτα 10 ‘καλύτερα’ χαρακτηριστικά είναι διαφορετικός του μηδενός και η τιμή της μέσης ακρίβειας ξεπερνά το 70.00%.

Αντίστοιχα, στον πίνακα 15 παρουσιάζονται οι υψηλότερες τιμές μέσης ακρίβειας (20 συνόλων ελέγχου) από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh) για τους ταξινομητές SVM Linear και SVM RBF, με χρήση των μοναδικών χαρακτηριστικών (ένωση υποσυνόλων) ανά ζεύγος κριτηρίων αξιολόγησης.

Η μόνη αξιοσημείωτη τιμή της μέσης ακρίβειας στον πίνακα 14 είναι η 73.82%, η οποία προήλθε από τον ταξινομητή SVM RBF με χρήση των 29 κοινών χαρακτηριστικών (γονιδίων) μεταξύ των ‘καλύτερων’ 60 χαρακτηριστικών των κριτηρίων αξιολόγησης KS 2 Samples & Fisher Score. Συγκρίνοντας τον συγκεκριμένο ταξινομητή με τα αποτελέσματα του πίνακα 13 και συνυπολογίζοντας τις παραμέτρους που χρησιμοποιήθηκαν και τον αριθμό των χαρακτηριστικών (γονιδίων), θα μπορούσαμε να τον κατατάξουμε ως τον τρίτο ταξινομητή με την καλύτερη ικανότητα γενίκευσης.

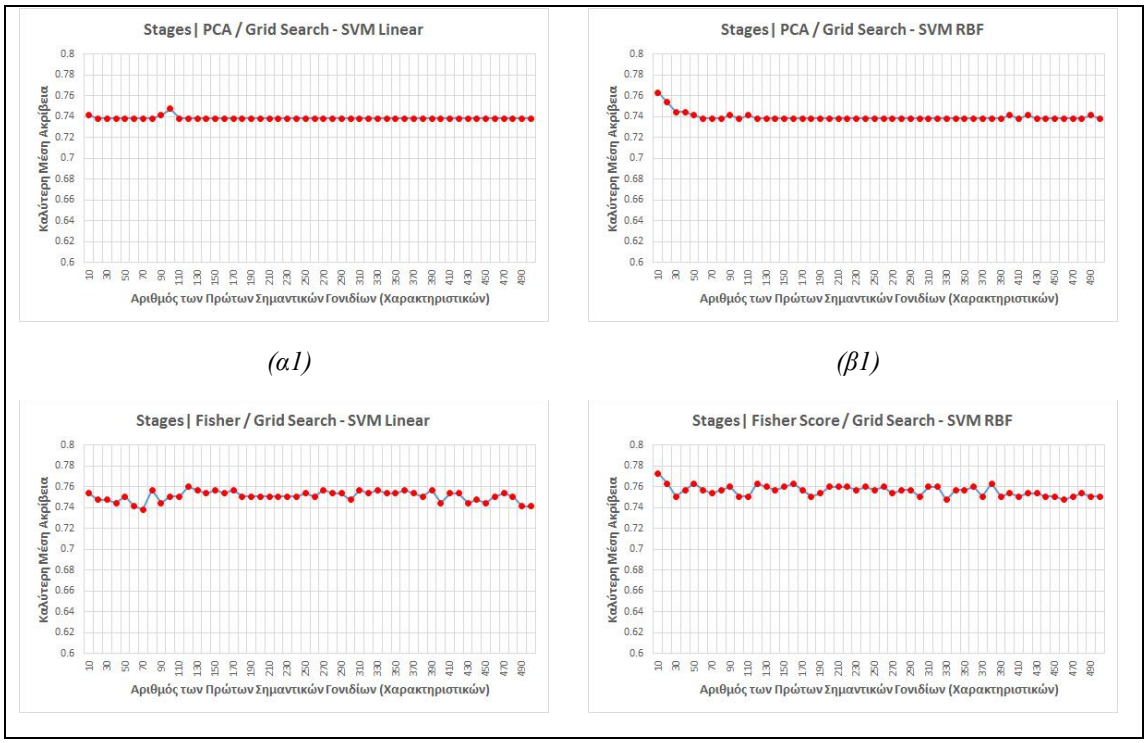
Συνδυασμός Κριτηρίων (Ένωση)	Ταξ/τής	Μέση Ακρίβεια	Παράμετροι		Αριθμ. Μοναδικών Γονιδίων (Ένωση Υποσυνόλων)	Αριθμ. Γονιδίων (Υποσύνολα από τα οποία προήλθε η ένωση)
			C	gamma		
KS 2 Samples & Inform. Gain	SVM Linear	71.61%	0.01	-	16	10
KS 2 Samples & Fisher Score		71.92%	0.01	-	15	10
KS 2 Samples & PCA		70.98%	0.1	-	20	10
KS 2 Samples & ReliefF		70.66%	0.001	-	608	310
Inform. Gain & Fisher Score		71.61%	0.01	-	33	20
			0.01	-	122	80
			0.01	-	175	120
Inform. Gain & PCA		69.40%	0.001	-	99	50
			0.001	-	362	190
			0.001	-	377	200
			0.01	-	397	210
			0.01	-	911	500
Inform. Gain & ReliefF		**72.87%**	**0.01**	**-**	**706**	**360**
Fisher Score & PCA		70.66%	1	-	20	10
			1	-	60	30
Fisher Score & ReliefF		71.61%	0.01	-	20	10
PCA & ReliefF		68.45%	0.1	-	897	460
KS 2 Samples & Inform. Gain	SVM RBF	**72.56%**	**10**	**0.001**	**206**	**140**
KS 2 Samples & Fisher Score		71.92%	0.1	0.01	31	20
KS 2 Samples & PCA		71.92%	1000	0.0001	585	310
KS 2 Samples & ReliefF		70.66%	1	0.001	40	20
			1000	0.00001	118	60
Inform. Gain & Fisher Score		**72.24%**	**10**	**0.001**	**150**	**100**
Inform. Gain & PCA		70.98%	100	0.00001	625	340
Inform. Gain & ReliefF		**72.24%**	**100**	**0.001**	**20**	**10**
Fisher Score & PCA		71.92%	10	0.001	20	10
Fisher Score & ReliefF		71.61%	100	0.00001	533	270
PCA & ReliefF		68.77%	100	0.0001	761	390

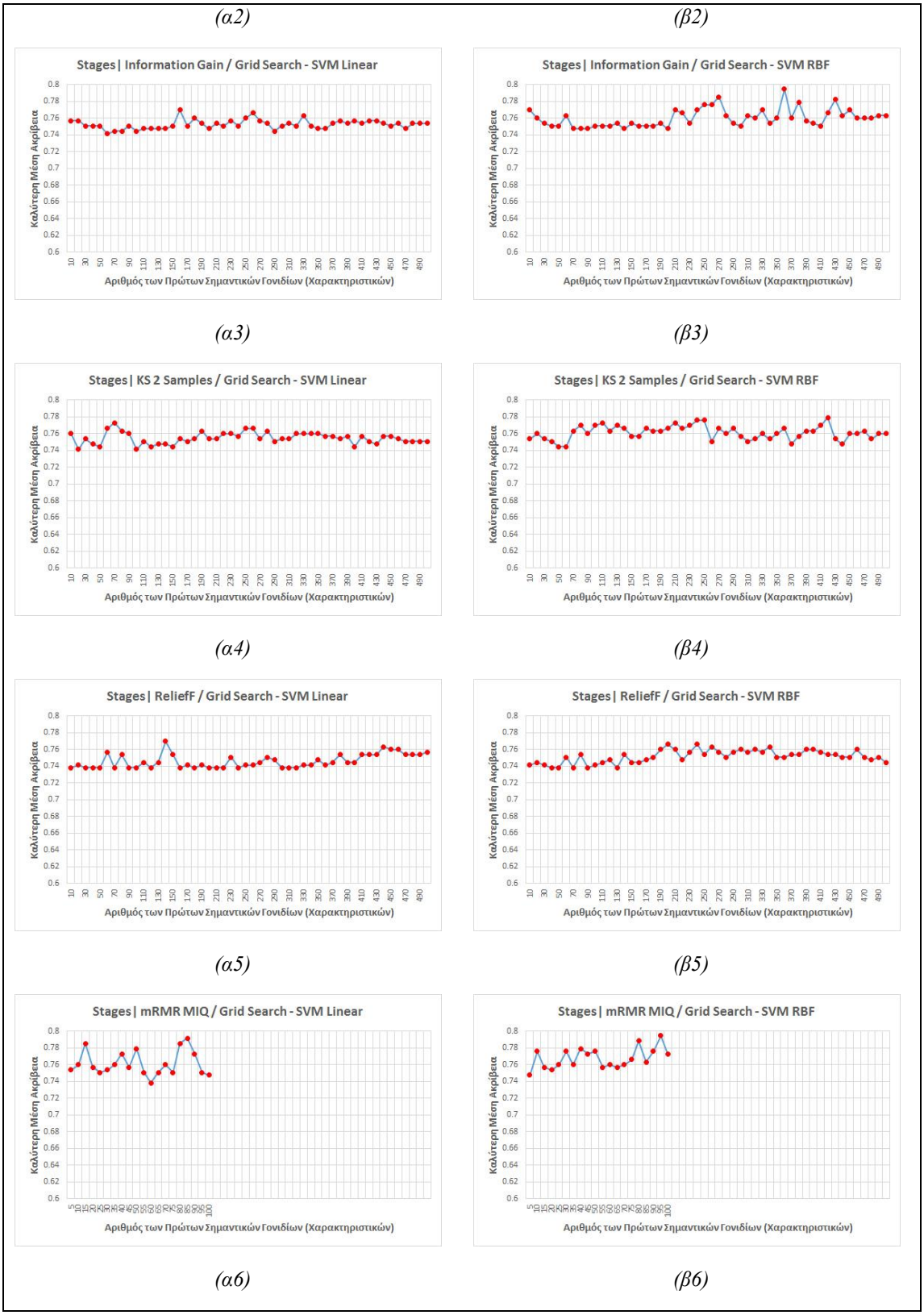
Πίνακας 15. *Υψηλότερη τιμή Μέσης Ακρίβειας (20 συνόλων ελέγχου) & οι αντίστοιχοι παράμετροι των ταξινομητών, από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh), με χρήση των μοναδικών χαρακτηριστικών (γονιδίων) ανά ζεύγος κριτηρίων αξιολόγησης (ένωση υποσυνόλων). (μεταβλητή στόχος Grades)*

Στον πίνακα 15, παρατηρούμε 4 τιμές της μέσης ακρίβειας των ταξινομητών να είναι μεγαλύτερες του 72.00%. Οι τιμές αυτές προέρχονται από ταξινομητές οι οποίοι χρησιμοποίησαν τα μοναδικά χαρακτηριστικά μεταξύ των κριτηρίων αξιολόγησης Mutual Information (IG) & ReliefF (SVM Linear & SVM RBF), KS 2 Samples & Mutual Information (IG) (SVM RBF) και Mutual Information (IG) & Fisher Score (SVM RBF). Η υψηλότερη μέση ακρίβεια μεταξύ των παραπάνω περιπτώσεων προέρχεται από τον ταξινομητή SVM Linear με χρήση των 706 μοναδικών χαρακτηριστικών του ζεύγους Mutual Information (IG) & ReliefF και είναι 72,87%. Συγκρίνοντας την μέση ακρίβεια του συγκεκριμένου ταξινομητή με τα αποτελέσματα του πίνακα 13 και συνυπολογίζοντας τις παραμέτρους που χρησιμοποιήθηκαν και τον αριθμό των χαρακτηριστικών (γονιδίων), φαίνεται ότι δεν επηρεάζει την κατάταξη της πρώτης τετράδας των ταξινομητών με την καλύτερη ικανότητα γενίκευσης.

5.3.2.2 Stages

Στην εικόνα 9 απεικονίζονται σε διαγράμματα τα τελικά αποτελέσματα της εξαντλητικής αναζήτησης για τους ταξινομητές SVM με γραμμικό πυρήνα και με πυρήνα RBF με μεταβλητή στόχο την Stages. Συγκεκριμένα σε κάθε διάγραμμα παρουσιάζονται για κάθε επιλεγμένο υποσύνολο (το οποίο προέρχεται από το εκάστοτε κριτήριο αξιολόγησης), οι υψηλότερες τιμές της μέσης ακρίβειας των 20 συνόλων ελέγχου, οι οποίες προέρχονται από διαφορετικούς συνδυασμούς παραμέτρων (C, gamma). Επιπλέον, στον πίνακα 16 παρουσιάζονται για κάθε διάγραμμα της εικόνας 9, η υψηλότερη μέση ακρίβεια των 20 συνόλων ελέγχου και οι αντίστοιχοι παράμετροι με τις οποίες επιτεύχθηκε αυτή.





Εικόνα 9. *Υψηλότερες τιμές (%) Μέσης Ακρίβειας (20 συνόλων ελέγχου) από την εκτέλεση Εξαντλητικής Αναζήτησης (Grid Search) με χρήση των ταξινομητών (α) SVM Linear και (β) SVM RBF για την μεταβλητή στόχος Stages. Τα υποσύνολα των χαρακτηριστικών (γονιδίων) προήλθαν από την αξιολόγηση των χαρακτηριστικών με βάση τα κριτήρια (α1, β1) PCA, (α2, β2) Fisher Score, (α3, β3) Mutual Information (Inform. Gain), (α4, β4) KS 2Samples (p-value), (α5, β5) ReliefF, (α6, β6) mRMR MIQ*

Διάγ/μα	Κριτήριο	Ταξινομητής	Μέση Ακρίβεια	Παράμετροι		Αριθμός Γονιδίων
				C	gamma	
α1	PCA	SVM Linear	74.76%	0.01	-	100
α2	Fisher Score		76.03%	0.001	-	120
α3	Inform. Gain		76.97%	0.01	-	160
α4	KS 2 Samples		77.29%	0.1	-	70
α5	ReliefF		76.97%	0.01	-	140
α6	**mRMR MIQ**		**79.18%**	**0.1**	**-**	**85**
β1	PCA	SVM RBF	76.34%	10000	0.0001	10
β2	Fisher Score		77.29%	100	0.01	10
β3	**Inform. Gain**		**79.50%**	**100**	**0.0001**	**360**
β4	KS 2 Samples		77.92%	100	0.00001	420
β5	ReliefF		76.66%	100	0.00001	200
				100	0.00001	240
β6	**mRMR MIQ**		**79.50%**	**10**	**0.01**	**95**

Πίνακας 16. *Για κάθε διάγραμμα της Εικόνας 9, η υψηλότερη τιμή Μέσης Ακρίβειας (20 συνόλων ελέγχου) & οι αντίστοιχοι παράμετροι από τις οποίες προήλθε αυτή. (μεταβλητή στόχος Stages)*

Στα διαγράμματα της εικόνας 9, παρατηρούμε ότι οι μικρότερες τιμές της μέσης ακρίβειας των ταξινομητών με χρήση υποσυνόλων χαρακτηριστικών (γονιδίων) των αντίστοιχων κριτηρίων αξιολόγησης, κυμαίνονται κοντά στο 74.00%.

Παρατηρώντας τον πίνακα 16, η υψηλότερη μέση ακρίβεια που επιτυγχάνουν 5 ταξινομητές κυμαίνεται από 76.00% έως και 77.00%. Συγκεκριμένα, οι ταξινομητές αυτοί είναι οι SVM Linear με χρήση υποσυνόλου χαρακτηριστικών το οποίο προήλθε από τα κριτήρια αξιολόγησης χαρακτηριστικών (γονιδίων) Fisher Score (Εικ.9 (α2)), Mutual Information (IG) (Εικ.9 (α3)) και ReliefF (Εικ.9 (α5)). Οι άλλοι δύο ταξινομητές είναι οι SVM RBF με χρήση υποσυνόλου χαρακτηριστικών το οποίο προήλθε από τα κριτήρια αξιολόγησης χαρακτηριστικών (γονιδίων) PCA (βάρη παραγόντων βαρύτητας 1ης Κ.Σ.) (Εικ.9 (β1)) και ReliefF (Εικ.9 (β5)).

Οι αμέσως επόμενες υψηλότερες τιμές της μέσης ακρίβειας επιτυγχάνονται από τους ταξινομητές SVM Linear και SVM RBF με χρήση υποσυνόλων χαρακτηριστικών (γονιδίων) που προέρχονται από το κριτήριο αξιολόγησης χαρακτηριστικών (γονιδίων) Kolmogorov-Smirnov 2 Samples. Συγκεκριμένα, ο ταξινομητής SVM Linear (Εικ.9 (α4)) επιτυγχάνει με χρήση 70 χαρακτηριστικών (γονιδίων) μέση ακρίβεια 77.29% με την παράμετρο C = 0.1. Ο ταξινομητής SVM RBF (Εικ.9 (β4)) επιτυγχάνει με χρήση 420 χαρακτηριστικών (γονιδίων) μέση ακρίβεια 77.92% με την παράμετρο C = 100 και gamma = 0.00001. Επιπλέον μεταξύ των ποσοστών 77.00% και 78.00% βρίσκεται και η μέση ακρίβεια (77.29%) που επιτυγχάνει

ο ταξινομητής SVM RBF (Εικ.9 (β2)) με χρήση 120 χαρακτηριστικών (γονιδίων) τα οποία προέρχονται από το κριτήριο αξιολόγησης χαρακτηριστικών (γονιδίων) Fisher Score.

Οι δύο από τις τρεις υψηλότερες τιμές της μέσης ακρίβειας στο σύνολο όλων των ταξινομητών προέρχονται από χρήση υποσυνόλων χαρακτηριστικών (γονιδίων) τα οποία επιλέχθηκαν με το κριτήριο αξιολόγησης χαρακτηριστικών (γονιδίων) mRMR MIQ. Συγκεκριμένα, ο ταξινομητής SVM Linear (Εικ.9 (α6)) επιτυγχάνει με χρήση 85 χαρακτηριστικών (γονιδίων) μέση ακρίβεια 79.18% με την παράμετρο C = 0.1. Ο ταξινομητής SVM RBF (Εικ.9 (β6)) επιτυγχάνει με χρήση 95 χαρακτηριστικών (γονιδίων) μέση ακρίβεια 79.50% με την παράμετρο C = 10 και gamma = 0.01.

Βέβαια, την συνολικά υψηλότερη τιμή ακρίβειας 97,50% την επιτυγχάνει και ο ταξινομητής SVM RBF (Εικ.9 (β3)) με παραμέτρους C = 100 και gamma = 0.0001, με χρήση πολύ περισσότερων χαρακτηριστικών (360 γονιδίων) τα οποία επιλέχθηκαν από το κριτήριο αξιολόγησης χαρακτηριστικών (γονιδίων) Mutual Information (IG).

Ο ταξινομητής SVM RBF με χρήση χαρακτηριστικών που επιλέχθηκαν από το κριτήριο αξιολόγησης χαρακτηριστικών mRMR MIQ είναι λίγο περισσότερο ανεκτικός (C = 10) στις λάθος ταξινομήσεις σε σχέση με τον ταξινομητή SVM RBF (C = 100) με χρήση χαρακτηριστικών που επιλέχθηκαν από το κριτήριο Mutual Information (IG). Από την άλλη όμως ο αριθμός των χαρακτηριστικών (γονιδίων) που χρησιμοποιήθηκαν είναι αρκετά μικρότερος (95 γονίδια mRMR MIQ / 360 γονίδια Mutual Information). Συνεπώς, η χρήση του κριτηρίου mRMR MIQ για την επιλογή χαρακτηριστικών δείχνει να είναι καλύτερη, καθώς με την επιλογή πολύ μικρότερου αριθμού χαρακτηριστικών, ο ταξινομητής SVM RBF επιτυγχάνει την ίδια μέση ακρίβεια. Συνυπολογίζοντας και το πλεονέκτημα της πιο εύκολης ερμηνείας των καθοριστικών χαρακτηριστικών (γονιδίων) από βιολογικής άποψης, όταν ο αριθμός τους είναι μικρός, συγκλίνουμε ακόμα περισσότερο στην χρήση του κριτηρίου mRMR MIQ.

Στον παρακάτω πίνακα (πίνακας 17) παρουσιάζονται οι υψηλότερες τιμές μέσης ακρίβειας (20 συνόλων ελέγχου) από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh) για τους ταξινομητές SVM Linear και SVM RBF, με χρήση των κοινών χαρακτηριστικών (τομή υποσυνόλων) ανά ζεύγος κριτηρίων αξιολόγησης. Συγκεκριμένα, παρουσιάζονται τα ζεύγη των οποίων τα κοινά χαρακτηριστικά στα πρώτα 10 ‘καλύτερα’ χαρακτηριστικά είναι διαφορετικός του μηδενός και η τιμή της μέσης ακρίβειας ξεπερνά το 76.00%.

Αντίστοιχα, στον πίνακα 18 παρουσιάζονται οι υψηλότερες τιμές μέσης ακρίβειας (20 συνόλων ελέγχου) από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh) για τους ταξινομητές SVM Linear και SVM RBF, με χρήση των μοναδικών χαρακτηριστικών (ένωση υποσυνόλων) ανά ζεύγος κριτηρίων αξιολόγησης.

Συνδυασμός Κριτηρίων (Τομή)	Ταξ/τής	Μέση Ακρίβεια	Παράμετροι		Αριθμός Κοινών Γονιδίων (Τομή Υποσυνόλων)	Αριθμ. Γονιδίων (Υποσύνολα από τα οποία προήλθε η Τομή)
			C	gamma		
KS 2 Samples & Inform. Gain	SVM Linear	76.03%	0.001	-	189	470
KS 2 Samples & Fisher Score		76.66%	0.001	-	268	450
Inform. Gain & Fisher Score		76.66%	0.01	-	21	70
			0.01	-	28	90
			0.01	-	161	360
KS 2 Samples & Inform. Gain	SVM RBF	**78.23%**	**100**	**0.0001**	**133**	**360**
KS 2 Samples & Fisher Score		76.34%	1000	0.01	4	10
			10000	0.00001	44	100
			100	0.00001	290	480
Inform. Gain & Fisher Score		**78.23%**	**100**	**0.0001**	**93**	**220**
			10000	**0.000001**	**110**	**260**
			100	**0.0001**	**130**	**290**

Πίνακας 17. *Υψηλότερη τιμή Μέσης Ακρίβειας (20 συνόλων ελέγχου) & οι αντίστοιχοι παράμετροι των ταξινομητών, από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh), με χρήση των κοινών χαρακτηριστικών (γονιδίων) ανά ζεύγος κριτηρίων αξιολόγησης (τομή υποσυνόλων). (μεταβλητή στόχος Stages)*

Στον πίνακα 17, παρατηρούμε ότι οι 2 υψηλότερες τιμές της μέσης ακρίβειας των ταξινομητών είναι 78.23%. Οι τιμές αυτές προέρχονται από ταξινομητές οι οποίοι χρησιμοποίησαν τα κοινά χαρακτηριστικά μεταξύ των κριτηρίων αξιολόγησης χαρακτηριστικών (γονιδίων) KS 2 Samples & Mutual Information (IG) (SVM RBF) και μεταξύ των Mutual Information (IG) & Fisher Score (SVM RBF). Η μέση ακρίβεια και στις δύο παραπάνω περιπτώσεις προέρχεται από χρήση περίπου 100 (95 με 133) μοναδικών χαρακτηριστικών (γονιδίων). Συγκρίνοντας την μέση ακρίβεια των παραπάνω ταξινομητών με τα αποτελέσματα του πίνακα 16 και συνυπολογίζοντας τις παραμέτρους που χρησιμοποιήθηκαν και τον αριθμό των χαρακτηριστικών (γονιδίων), φαίνεται ότι δεν επηρεάζουν την κατάταξη της πρώτης τριάδας των ταξινομητών με την καλύτερη ικανότητα γενίκευσης.

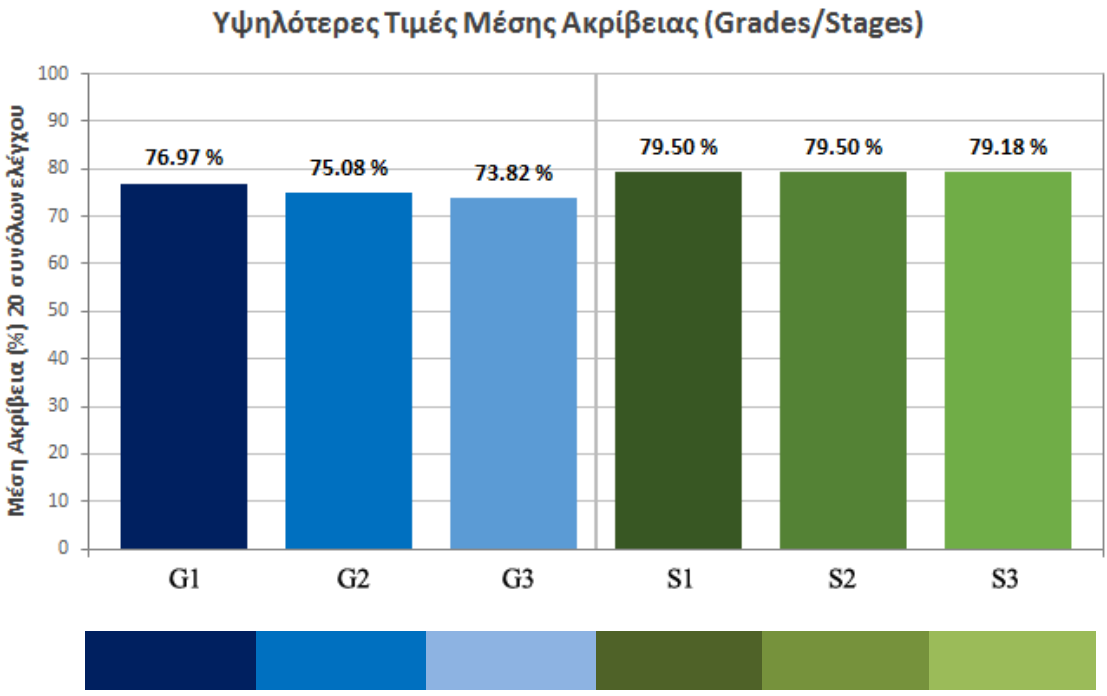
Η καλύτερη τιμή μέσης ακρίβειας που επιτυγχάνεται στον πίνακα 18, είναι επίσης 78,23% και προέρχεται από τον ταξινομητή SVM RBF με χρήση 425 μοναδικών χαρακτηριστικών του ζεύγους KS 2 Samples & Mutual Information (IG). Συνεπώς, συγκρίνοντας και αυτόν τον ταξινομητή με τα αποτελέσματα του πίνακα 16, καταλήγουμε στο ότι δεν επηρεάζει την κατάταξη της πρώτης τριάδας των ταξινομητών με την καλύτερη ικανότητα γενίκευσης.

Συνδυασμός Κριτηρίων (Ένωση)	Ταξ/τής	Μέση Ακρίβεια	Παράμετροι		Αριθμ. Μοναδικών Γονιδίων (Ένωση Υποσυνόλων)	Αριθμ. Γονιδίων (Υποσύνολα από τα οποία προήλθε η ένωση)
			C	gamma		
KS 2 Samples & Inform. Gain	SVM Linear	77.29%	0.01	-	366	220
KS 2 Samples & Fisher Score		77.29%	10	-	60	40
KS 2 Samples & PCA		76.03%	0.001	-	714	370
KS 2 Samples & ReliefF		76.03%	10000	-	20	10
			0.001	-	510	260
			0.001	-	783	400
			0.001	-	918	470
Inform. Gain & Fisher Score		76.34%	0.01	-	18	10
			0.01	-	72	40
Inform. Gain & PCA		76.66%	0.1	-	59	30
Inform. Gain & ReliefF		76.97%	0.001	-	945	490
Fisher Score & PCA		76.34%	1000	-	20	10
Fisher Score & ReliefF		76.03%	0.001	-	416	210
PCA & ReliefF		75.71%	0.001	-	472	240
KS 2 Samples & Inform. Gain	SVM RBF	**78.23%**	**100**	**0.0001**	**425**	**260**
KS 2 Samples & Fisher Score		77.60%	100	0.001	31	20
			100	0.00001	525	370
			100	0.00001	566	400
			1000	0.000001	670	480
KS 2 Samples & PCA		77.29%	10	0.001	60	30
KS 2 Samples & ReliefF		76.97%	10	0.0001	766	390
Inform. Gain & Fisher Score		77.29%	1000	0.0001	18	10
Inform. Gain & PCA		76.66%	100	0.001	20	10
Inform. Gain & ReliefF		75.71%	10000	0.0001	20	10
			10000	0.0000001	40	20
			10	0.0001	235	120
			10	0.0001	834	430
Fisher Score & PCA		77.29%	1000	0.0001	20	10
Fisher Score & ReliefF		76.03%	1000	0.000001	216	110
PCA & ReliefF		75.39%	10	0.0001	139	70

Πίνακας 18. *Υψηλότερη τιμή Μέσης Ακρίβειας (20 συνόλων ελέγχου) & οι αντίστοιχοι παράμετροι των ταξινομητών, από την εκτέλεση εξαντλητικής αναζήτησης (Grid Searh), με χρήση των μοναδικών χαρακτηριστικών (γονιδίων) ανά ζεύγος κριτηρίων αξιολόγησης (ένωση υποσυνόλων). (μεταβλητή στόχος Stages)*

5.4 Σύνοψη Συμπερασμάτων Αξιολόγησης

Μετά από την παρουσίαση και την ανάλυση της αξιολόγησης των χαρακτηριστικών (γονιδίων) (υποενότητα 5.3.1) και την σύγκριση της επίδοσης των αντίστοιχων ταξινομητών (υποενότητα 5.3.2) για κάθε μεταβλητή στόχος (Grades και Stages), στην εικόνα 10 παραθέτουμε ένα συγκεντρωτικό διάγραμμα (συνοδευόμενο από τον αντίστοιχο πίνακα) με τα καλύτερα αποτελέσματα.



Αριθμός Σημαντικών Γονιδίων	30	65	29	95	360	85
Κριτήριο Αξιολόγησης	mRMR MIQ	mRMR MIQ	KS 2Samples & Fisher Sc. (Τομή Υποσ. 60 γονιδ.)	mRMR MIQ	Mutual Information (Inf. Gain)	mRMR MIQ
Ταξ/τής	SVM RBF	SVM Linear	SVM RBF	SVM RBF	SVM RBF	SVM Linear
C	10000	0.1	10000	10	100	0.1
gamma	0.001	-	0.0001	0.01	0.0001	-

Εικόνα 10. *Συγκεντρωτικό διάγραμμα των καλύτερων αποτελεσμάτων συνοδευόμενο από τον αντίστοιχο πίνακα.*

Τα συμπεράσματα μας σύμφωνα με τα καλύτερα αποτελέσματα που αφορούν την μεταβλητή στόχος Grades, συνυπολογίζοντας ότι η αναλογία των προτύπων ως προς τις τιμές της μεταβλητής στόχος είναι 61% Early Grade και 39% Late Grade, είναι τα εξής:

- Η τομή υποσυνόλων σημαντικών χαρακτηριστικών (29 γονίδια) που προέρχονται από τα κριτήρια αξιολόγησης Fisher Score και Kolmogorov-Smirnov 2 Samples πετυχαίνει την κατασκευή ενός ταξινομητή SVM με πυρήνα RBF ο οποίος διαθέτει την τρίτη καλύτερη ικανότητα γενίκευσης για την πρόβλεψη της μεταβλητής στόχος Grades.
- Η αξιολόγηση των χαρακτηριστικών με το κριτήριο mRMR MIQ και η επιλογή ενός μικρού αριθμού σημαντικών χαρακτηριστικών (γονιδίων) για την κατασκευή ενός ταξινομητή SVM είτε με πυρήνα RBF είτε με γραμμικό πυρήνα, έχει ως αποτέλεσμα δύο ταξινομητές οι οποίοι διαθέτουν τις δύο πρώτες καλύτερες ικανότητες γενίκευσης για την πρόβλεψη της μεταβλητής στόχος Grades.

Τα συμπεράσματα μας σύμφωνα με τα καλύτερα αποτελέσματα που αφορούν την μεταβλητή στόχος Stages, συνυπολογίζοντας ότι η αναλογία των προτύπων ως προς τις τιμές της μεταβλητής στόχος είναι 73% Early Stage και 27% Late Stage, είναι τα εξής:

- Η αξιολόγηση των χαρακτηριστικών με το κριτήριο της Αμοιβαία Πληροφορίας αλλά και με την χρήση του κριτηρίου mRMR MIQ και η επιλογή των αντίστοιχων υποσυνόλων για την κατασκευή των αντίστοιχων ταξινομητών SVM με πυρήνα RBF, έχει ως αποτέλεσμα δύο ταξινομητές οι οποίοι διαθέτουν την ίδια καλύτερη ικανότητα γενίκευσης για την πρόβλεψη της μεταβλητής στόχος Stages. Βέβαια ο αριθμός των σημαντικών χαρακτηριστικών (γονιδίων) που χρησιμοποιείται και προέρχεται από το κριτήριο της Αμοιβαίας Πληροφορίας είναι πολύ μεγαλύτερος από τον αριθμό των σημαντικών χαρακτηριστικών (γονιδίων) που χρησιμοποιούνται και προέρχονται από το κριτήριο αξιολόγησης mRMR MIQ.

Συνοψίζοντας τα καλύτερα αποτελέσματα που αφορούν και τις δύο μεταβλητές στόχους (Grades και Stages), καταλήγουμε στα εξής συμπεράσματα:

1. Ο ταξινομητής που μπορεί να γενικεύσει καλύτερα με την χρήση του κατάλληλου υποσυνόλου σημαντικών χαρακτηριστικών (γονιδίων) δίνοντας μεγάλη σημασία στην σωστή ταξινόμηση των προτύπων, είναι ο SVM με πυρήνα RBF.

2. Η αξιολόγηση των χαρακτηριστικών (σε σχέση με κάθε μεταβλητή στόχο) με το κριτήριο mRMR MIQ και η επιλογή ενός μικρού αριθμού σημαντικών χαρακτηριστικών για την κατασκευή ενός ταξινομητή SVM με πυρήνα RBF, έχει ως αποτέλεσμα και στις δυο περιπτώσεις (Grades/Stages) έναν ταξινομητή ο οποίος διαθέτει την καλύτερη ικανότητα γενίκευσης. Με τον χαρακτηρισμό 'σημαντικά' εννοούμε τα χαρακτηριστικά που περιέχουν πληροφορία ικανή για την κατασκευή ενός ταξινομητή που θα διαθέτει την ικανότητα της γενίκευσης. Συνεπώς η χρήση της μεθόδου επιλογής χαρακτηριστικών mRMR MIQ δείχνει να υπερέχει των υπόλοιπων μεθόδων, καθώς με την χρήση ενός μικρού αριθμού σημαντικών χαρακτηριστικών καταφέρνει την κατασκευή ενός ταξινομητή SVM RBF ο οποίος διαθέτει την καλύτερη ικανότητα γενίκευσης.

6

Τεχνικές Λεπτομέρειες

Σε αυτό το κεφάλαιο καταγράφονται οι πλατφόρμες, τα προγραμματιστικά εργαλεία και τα πακέτα λογισμικού που χρησιμοποιήθηκαν για την εκπόνηση της παρούσας διπλωματικής εργασίας.

6.1 Πλατφόρμες και Προγραμματιστικά Εργαλεία

Το μεγαλύτερο ποσοστό των πειραμάτων της διπλωματικής εργασίας εκτελέστηκε σε μηχάνημα με λειτουργικό σύστημα Linux του Ινστιτούτου Εφαρμοσμένων Βιοεπιστημών (INAB) το οποίο διέθετε την απαραίτητη υπολογιστική δύναμη για την ολοκλήρωση του συνόλου των πειραμάτων σε εύλογο χρονικό διάστημα. Παράλληλα χρησιμοποιήθηκε και ένα μηχάνημα με λειτουργικό σύστημα Windows για την σύνταξη του κώδικα των πειραμάτων.

Για την εκπόνηση της παρούσας εργασίας χρησιμοποιήσαμε την γλώσσα προγραμματισμού Python 2.7. Η μόνη διαφοροποίηση στην έκδοση της Python που χρησιμοποιήσαμε, ήταν για την εκτέλεση συγκεκριμένου πακέτου λογισμικού (pymRMR) το οποίο απαιτούσε την εγκατάσταση της έκδοσης 3.6.

Για την εγκατάσταση της Python 2.7 και των βασικών βιβλιοθηκών, χρησιμοποιήθηκε η πλατφόρμα Anaconda2[17]. Οι βιβλιοθήκες και οι κυριότερες συναρτήσεις που χρησιμοποιήθηκαν και εμπεριέχονται στην πλατφόρμα Anaconda2 είναι οι παρακάτω:

[17] https://www.anaconda.com/

- **Spyder[18]:** Ολοκληρωμένο περιβάλλον ανάπτυξης για τη σύνταξη προγραμμάτων (Integrated Development Environment - IDE).
- **Scikit-learn[19]:** Βιβλιοθήκη για την εφαρμογή μεθόδων μηχανικής μάθησης. Οι κυριότερες συναρτήσεις που χρησιμοποιήθηκαν είναι οι εξής:
    - sklearn.decomposition.PCA()[4]
    - sklearn.metrics.normalized_mutual_info_score()[5]
    - sklearn.svm.SVC()[15]
    - sklearn.model_selection.GridSearchCV()[16]
- **Numpy[20], Scipy[21], Pandas[22]:** Βιβλιοθήκες για στατιστική ανάλυση (Analytics) και για επιστημονικούς υπολογισμούς (Scientific Computing). Οι κυριότερες συναρτήσεις που χρησιμοποιήθηκαν είναι οι εξής:
    - scipy.stats.ks_2samp()[7]
- **Matplotlib[23]:** Βιβλιοθήκη για την οπτικοποίηση δεδομένων μέσω διαγραμμάτων.

Επιπλέον χρησιμοποιήθηκαν και τα παρακάτω πακέτα λογισμικού:

- **ReliefF 0.1.2**[8]
- **pymRMR**[12]

[18] https://pythonhosted.org/spyder/index.html

[19] http://scikit-learn.org/stable/index.html

[20] http://www.numpy.org/

[21] https://docs.scipy.org/doc/scipy-0.15.1/reference/index.html

[22] https://pandas.pydata.org/

[23] https://matplotlib.org/

7

Επίλογος

Σε αυτό το κεφάλαιο συγκεντρώνονται τα σημαντικότερα συμπεράσματα που προέκυψαν από την εκπόνηση της παρούσας διπλωματικής εργασίας και παρουσιάζονται κάποιες ιδέες για μελλοντικές επεκτάσεις.

7.1 Σύνοψη και Συμπεράσματα

Σκοπός αυτής της διπλωματικής εργασίας είναι να καταλήξουμε σε έναν μικρό αριθμό γονιδίων (σε σχέση με τα 12604 γονίδια) τα οποία περιέχουν ικανή πληροφορία για την κατασκευή ενός ταξινομητή που θα διαθέτει την ικανότητα της γενίκευσης. Οι μεταβλητές στόχοι που εξετάσαμε και αφορούν τον καρκίνο του ήπατος είναι ο βαθμός του καρκίνου του ήπατος (Grades) και το στάδιο του καρκίνου του ήπατος (Stages). Οι τιμές των γονιδίων του συνόλου δεδομένων που εξετάσαμε αποτελούν τιμές έκφρασης μετασχηματισμένες από αλληλούχιση RNA (RNA-seq).

Συνοψίζοντας τα αποτελέσματα της παρούσας διπλωματικής εργασίας, καταλήγουμε στο συμπέρασμα ότι το κριτήριο επιλογής χαρακτηριστικών (γονιδίων) mRMR MIQ υπερέχει των υπόλοιπων κριτηρίων αξιολόγησης που εξετάσαμε κατά την επιλογή των σημαντικών (καθοριστικών) χαρακτηριστικών (γονιδίων). Τόσο σε σχέση με την μεταβλητή στόχος Grades αλλά και σε σχέση με την μεταβλητή στόχος Stages, το κριτήριο αξιολόγησης χαρακτηριστικών mRMR MIQ κατάφερε να εντοπίσει τον αντίστοιχο μικρότερο αριθμό σημαντικών χαρακτηριστικών (Grades: 30 σημαντικά γονίδια, Stages: 95 σημαντικά γονίδια).

Αυτά τα σημαντικά χαρακτηριστικά (γονίδια) περιέχουν πληροφορία ικανή για την κατασκευή ενός ταξινομητή SVM RBF ο οποίος διαθέτει καλύτερη ικανότητα γενίκευσης έναντι άλλων υποσυνόλων σημαντικών γονιδίων που προήλθαν από τα υπόλοιπα κριτήρια αξιολόγησης που εξετάσαμε.

Συγκεκριμένα, με την χρήση 30 σημαντικών γονιδίων σε σχέση με την μεταβλητή στόχος Grades, ο ταξινομητής SVM RBF κατάφερε να επιτύχει μέσο όρο ακρίβειας (από τα 20 σύνολα ελέγχου) 76.97%. Αντίστοιχα, με την χρήση 95 σημαντικών γονιδίων σε σχέση με την μεταβλητή στόχος Stages, ο ταξινομητής SVM RBF κατάφερε να επιτύχει μέσο όρο ακρίβειας (από τα 20 σύνολα ελέγχου) 79.50%.

Βέβαια, είναι σημαντικό να αναφερθεί ότι η μέθοδος επιλογής χαρακτηριστικών mRMR MIQ απαιτεί πολύ περισσότερο χρόνο για την εκτέλεση της και τον εντοπισμό των N σημαντικών χαρακτηριστικών σε σχέση με τις υπόλοιπες μεθόδους που εξετάσαμε. Για την κατανόηση του μεγάλου χρόνου που απαιτεί για την εκτέλεσή της, αρκεί να αναλογιστούμε ότι το μέγιστο χρονικό διάστημα που χρειάστηκε κάθε μία από τις υπόλοιπες μεθόδους για την αξιολόγηση όλων τα χαρακτηριστικών (12604 γονίδια) δεν ξεπερνούσε το ένα 12ωρο σε αντίθεση με την μέθοδο mRMR MIQ η οποία χρειάστηκε τον δεκαπλάσιο χρόνο για την επιλογή των 100 σημαντικών χαρακτηριστικών (γονιδίων). Συνυπολογίζοντας, ότι όλοι οι μέθοδοι επιλογής χαρακτηριστικών εκτελέστηκαν με χρήση της ίδιας υπολογιστικής δύναμης, συμπεραίνουμε τις αρκετά μεγαλύτερες απαιτήσεις σε υπολογιστική δύναμη για την εκτέλεση του κριτηρίου mRMR MIQ.

7.2 Μελλοντικές Επεκτάσεις

Σύμφωνα με τα αποτελέσματα και τα συμπεράσματα της παρούσας εργασίας, υπάρχει χώρος για μελλοντική έρευνα. Συγκεκριμένα, τα σημεία στα οποία μπορεί να επεκταθεί η παρούσα έρευνα είναι τα εξής:

1. Εφαρμογή του κριτηρίου αξιολόγησης mRMR με χρήση της συνάρτησης αξιολόγησης MID (σχέση 54), ώστε να υπάρχει ισορροπία μεταξύ των δύο συνθηκών που πρέπει να ικανοποιούνται.
2. Διακριτοποίηση του συνόλου δεδομένων για εφαρμογή του mRMR, σε μικρό αριθμό τμημάτων (bins), καθώς το κριτήριο έχει δείξει να λειτουργεί καλύτερα με μικρό αριθμό διακριτών τιμών κατά την εφαρμογή του σε κάποια σύνολα δεδομένων.
3. Συνδυασμός του κριτηρίου mRMR με άλλα κριτήρια αξιολόγησης χαρακτηριστικών. Για παράδειγμα, εφαρμογή του κριτηρίου Fisher Score ή Kolmogorov-Smirnov 2 Samples (τα οποία κατείχαν τα επόμενα καλύτερα αποτελέσματα στην έρευνά μας) στο υποσύνολο των N χαρακτηριστικών που προκύπτουν από το κριτήριο mRMR.

8

Βιβλιογραφία

[1] Danaee, P., Ghaeini, R., & Hendrix, D. A. (2017). A deep learning approach for cancer detection and relevant gene identification. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017* (pp. 219-229).

[2] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103). ACM.

[3] Bhola, A., & Tiwari, A. K. (2015). Machine Learning Based Approaches for Cancer Classification Using Gene Expression Data. *Machine Learning and Applications: An International Journal*, *2*(3/4), 01-12.

[4] Su, Q., Wang, Y., Jiang, X., Chen, F., & Lu, W. C. (2017). A Cancer Gene Selection Algorithm Based on the KS Test and CFS. *BioMed research international*, *2017*.

[5] Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill International Editions.

[6] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

[7] Brereton, R. G., & Lloyd, G. R. (2010). Support vector machines for

classification and regression. *Analyst*, *135*(2), 230-267.

[8] Διαμαντάρας, Κ. (2007). *Τεχνητά Νευρωνικά Δίκτυα*. Εκδόσεις Κλειδάριθμος.

[9] Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273-324.

[10] Duda, R. O., Hart, P. E., & Stork, D. G. (1973). *Pattern classification* (Vol. 2). New York: Wiley.

[11] Liu, H., & Motoda, H. (2008). Less is more. *Computational Methods of Feature Selection. Chapman & Hall/CRC*, 3-17.

[12] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157-1182.

[13] Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, *97*(1-2), 245-271.

[14] Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, *23*(19), 2507-2517.

[15] Hira, Z. M., & Gillies, D. F. (2015). A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics*, *2015*.

[16] Smith, L. I. (2002). *A tutorial on principal components analysis*.

[17] Gu, Q., Li, Z., & Han, J. (2012). Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.

[18] Jakulin, A. (2005). *Machine learning based on attribute interactions* (Doctoral dissertation, Univerza v Ljubljani).

[19] Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, *5*(1), 3-55.

[20] Berger, V. W., & Zhou, Y. (2014). Kolmogorov–smirnov test: Overview. *Wiley StatsRef: Statistics Reference Online*.

[21] Itl.nist.gov. *Kolmogorov-Smirnov 2-Sample Goodness of Fit Test*. [online] Available at: https://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/ks2samp.htm [Accessed 16 Oct. 2017].

[22] Kononenko, I. (1994, April). Estimating attributes: analysis and extensions of RELIEF. In *European conference on machine learning* (pp. 171-182). Springer, Berlin, Heidelberg.

[23] Robnik-Šikonja, M., & Kononenko, I. (1997, July). An adaptation of Relief for attribute estimation in regression. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)* (pp. 296-304).

[24] Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. In *Machine Learning Proceedings 1992* (pp. 249-256).

[25] Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine learning*, *53*(1-2), 23-69.

[26] Ding, C., & Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, *3*(02), 185-205.

[27] Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, *27*(8), 1226-1238.

[28] The Cancer Genome Atlas - National Cancer Institute. *About TCGA*. [online] Available at: https://cancergenome.nih.gov/abouttcga [Accessed 26 Mar. 2018].

[29] Ally, A., Balasundaram, M., Carlsen, R., Chuah, E., Clarke, A., Dhalla, N., ... & Marra, M. A. (2017). Comprehensive and integrative genomic characterization of hepatocellular carcinoma. *Cell*, *169*(7), 1327-1341.

[30] Chan, A. C., Fan, S. T., Poon, R. T., Cheung, T. T., Chok, K. S., Chan, S. C., & Lo, C. M. (2013). Evaluation of the seventh edition of the American Joint Committee on Cancer tumour–node–metastasis (TNM) staging system for patients undergoing curative resection of hepatocellular carcinoma: implications for the development of a refined staging system. *HPB*, *15*(6), 439-448.

9

Παράρτημα

9.1 PCA & SVM Linear / SVM RBF

9.1.1 PCA & (Grades) SVM Linear

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή PCA
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με τα βάρη του 1ου principal
   compnent
05 Εφαρμογή SVM Linear για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 GRADES: Κλάσεις Στόχοι -> Early Grade = 0 και Late Grade = 1
07 """
08
09 from pandas import read_csv
10 import numpy as np
11 from sklearn.decomposition import PCA
12 from sklearn import svm
13 from sklearn.model_selection import GridSearchCV
14
15 # Read data - features
16 data = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
17 data = np.array(data)
18 data2 = np.array(data)
19
20 ### PCA ####################################################
```

```
21
22 # Extract vector with mean values for every column
23 mean_data = np.mean(data, axis=0)
24
25 # Remove mean from each column
26 for icol in range(data.shape[1]):
27     data[:,icol] = data[:,icol] - mean_data[icol]
28
29 pca = PCA(n_components = 1)
30 data_pca = pca.fit_transform(data)
31
32 comp = pca.components_
33
34 features_idx_max = np.argsort(-comp)
35 features_idx_max_w = np.transpose(features_idx_max)
36 features_idx_max_w = features_idx_max_w.flatten()
37
38 np.savetxt("PCA_idx_now.csv", features_idx_max_w,
   delimiter=",",fmt='%.0f')
39
40 ### END PCA ##################################################
41
42 ### PREPARE DATA FOR SVM #####################################
43
44 # Read targets
45 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
46 data1_matrix = data1.as_matrix()
47
48 # merge data + targets (data1_matrix & data2)
49 data_all = np.concatenate((data1_matrix,data2), axis=1)
50
51 # Prepare Grades (targets) + data (genes) for SVM
52 data_all_grades = np.array([])
53 data_all_grades = data_all
54
55 # delete columns that we don't want
56 data_all_grades = np.delete(data_all_grades, (0,1,3), axis=1)
57
58 # targets for Grades (0-1) + data --> create data_all_grades
59 for irow in range(data_all_grades.shape[0]):
60     # Early_Grade = 0
61     if data_all_grades[irow,0]=='Early_Grade':
62         data_all_grades[irow,0] = 0
63     # Late_Grade = 1
64     if data_all_grades[irow,0]=='Late_Grade':
65         data_all_grades[irow,0] = 1
66
67 data_rows = data_all_grades.shape[0]  # number of rows
68 data_cols = data_all_grades.shape[1]  # numer of cols
```

```
69 data_all_grades = data_all_grades.astype('float64')
70
71 # sorted array according to features' weight from first PCA component
   (Grades (0-1) + data)
72 # create data_grades
73 data_grades = np.empty([data_rows,data_cols]);
74 for icol in range(data_cols):
75     if icol==0:
76         data_grades[:,icol] = data_all_grades[:,icol]
77     if icol>0:
78         grades_max_row = features_idx_max_w[icol-1]
79         grades_max_row = grades_max_row + 1
80         data_grades[:,icol] = data_all_grades[:,grades_max_row]
81
82 ### END - PREPARE DATA FOR SVM ############################
83
84 ### SVM LINEAR GRADES #####################################
85
86 # array with all final results - for save in csv
87 final_grades_svmlinear = np.zeros(shape=(500,4))
88 csv_row = -1
89 # array with best final results - for save in csv
90 final_grades_svmlinear_best = np.zeros(shape=(50,4))
91 csv_row_best = -1
92
93 # NUMBER OF GENES
94 for number_of_genes in range(11,511,10):
95
96     print 'GRADES ------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
97     data_for_svm_linear = np.empty([data_rows, number_of_genes]);
98
99     # array with targets + first 'number_of_genes' genes #create
   data_for_svm_linear
100    for icol in range(number_of_genes):
101        data_for_svm_linear[:,icol] = data_grades[:,icol]
102
103    shuffle_data_targets = np.random.permutation(data_for_svm_linear)
104    grades_data = np.delete(shuffle_data_targets, (0), axis=1)
105    grades_targets = shuffle_data_targets[:,0]
106
107    parameter_candidates = [
108            {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
   'kernel': ['linear']}]
109
110    clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
111    clf.fit(grades_data, grades_targets)
```

```
112
113     print(clf.best_params_)
114     bestC = clf.best_params_['C']
115     print " "
116
117     trmeans = clf.cv_results_['mean_train_score']
118     trstds = clf.cv_results_['std_train_score']
119     means = clf.cv_results_['mean_test_score']
120     stds = clf.cv_results_['std_test_score']
121     c_para = clf.cv_results_['param_C']
122
123     # accuracy scores & Best accuracy scores
124     for c_par, trmean, trstd, mean, std, params in zip(c_para, trmeans,
    trstds, means, stds, clf.cv_results_['params']):
125         print("GRADES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
126               % (trmean, trstd * 2, mean, std * 2, params))
127
128         csv_row = csv_row + 1
129
130         final_grades_svmlinear[csv_row,0] = number_of_genes-1
131         final_grades_svmlinear[csv_row,1] = c_par
132         final_grades_svmlinear[csv_row,2] = mean
133         final_grades_svmlinear[csv_row,3] = std * 2
134
135         if (c_par == bestC):
136             csv_row_best = csv_row_best + 1
137             final_grades_svmlinear_best[csv_row_best,0] =
    number_of_genes-1
138             final_grades_svmlinear_best[csv_row_best,1] = bestC
139             final_grades_svmlinear_best[csv_row_best,2] = mean
140             final_grades_svmlinear_best[csv_row_best,3] = std * 2
141
142 np.savetxt("GRADES_PCA_SVM-Linear_all.csv", final_grades_svmlinear,
    delimiter=",",fmt='%.5f')
143 np.savetxt("GRADES_PCA_SVM-Linear_best.csv",
    final_grades_svmlinear_best, delimiter=",",fmt='%.5f')
144
145 print 'END GRADES!'
```

9.1.2 PCA & (Stages) SVM RBF

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή PCA
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με τα βάρη του 1ου principal
   compnent
05 Εφαρμογή SVM RBF για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 STAGES: Κλάσεις Στόχοι -> Early Stage = 0 και Late Stage = 1
```

```
"""

from pandas import read_csv
import numpy as np
from sklearn.decomposition import PCA
from sklearn import svm
from sklearn.model_selection import GridSearchCV

# Read data - features
data = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
data = np.array(data)
data2 = np.array(data) #for SVM

### PCA ###################################################

# Extract vector with mean values for every column
mean_data = np.mean(data, axis=0)

# Remove mean from each column
for icol in range(data.shape[1]):
    data[:,icol] = data[:,icol] - mean_data[icol]

pca = PCA(n_components = 1)
data_pca = pca.fit_transform(data)

comp = pca.components_

features_idx_max = np.argsort(-comp)
features_idx_max_w = np.transpose(features_idx_max)
features_idx_max_w = features_idx_max_w.flatten()

np.savetxt("PCA_idx.csv", features_idx_max_w, delimiter=",",fmt='%.0f')

### END PCA ###############################################

### PREPARE DATA FOR SVM ##################################

# Read targets
data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
data1_matrix = data1.as_matrix()

# merge data + targets (data1_matrix & data2)
data_all = np.concatenate((data1_matrix,data2), axis=1)

# Prepare Stages (targets) + data (genes) for SVM
data_all_stages = np.array([])
data_all_stages = data_all

# delete columns that we don't want
```

```
56 data_all_stages = np.delete(data_all_stages, (0,1,2), axis=1)
57
58 # targets for Stages (0-1) + data --> create data_all_stages
59 for irow in range(data_all_stages.shape[0]):
60     # Early_Stage = 0
61     if data_all_stages[irow,0]=='Early_Stage':
62         data_all_stages[irow,0] = 0
63     # Late_Stage = 1
64     if data_all_stages[irow,0]=='Late_Stage':
65         data_all_stages[irow,0] = 1
66
67 data_rows = data_all_stages.shape[0] # number of rows
68 data_cols = data_all_stages.shape[1] # numer of cols
69 data_all_stages = data_all_stages.astype('float64')
70
71 # sorted array according to features' weight from first PCA component
   (Stages (0-1) + data)
72 # create data_stages
73 data_stages = np.empty([data_rows,data_cols]);
74 for icol in range(data_cols):
75     if icol==0:
76         data_stages[:,icol] = data_all_stages[:,icol]
77     if icol>0:
78         stages_max_row = features_idx_max_w[icol-1]
79         stages_max_row = stages_max_row + 1
80         data_stages[:,icol] = data_all_stages[:,stages_max_row]
81
82 ### END - PREPARE DATA FOR SVM ############################
83
84 ### SVM RBF GRADES ########################################
85
86 # array with all final results - for save in csv
87 final_stages_svmrbf = np.zeros(shape=(6800,5))
88 csv_row = -1
89 # array with best final results - for save in csv
90 final_stages_svmrbf_best = np.zeros(shape=(50,5))
91 csv_row_best = -1
92
93 # NUMBER OF GENES
94 for number_of_genes in range(11,511,10):
95
96     print 'STAGES -------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
97     data_for_svm_rbf = np.empty([data_rows, number_of_genes]);
98
99     # array with targets + first 'number_of_genes' genes #create
   data_for_svm_rbf
100    for icol in range(number_of_genes):
101        data_for_svm_rbf[:,icol] = data_stages[:,icol]
```

```
102
103     shuffle_data_targets = np.random.permutation(data_for_svm_rbf)
104     stages_data = np.delete(shuffle_data_targets, (0), axis=1)
105     stages_targets = shuffle_data_targets[:,0]
106
107     parameter_candidates = [
108             {'kernel': ['rbf'],
109              'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
    1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
110              'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
111              'tol': [1e-10]}]
112
113     clf = GridSearchCV(estimator=svm.SVC(),
    param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
    return_train_score=True)
114     clf.fit(stages_data, stages_targets)
115
116     print(clf.best_params_)
117     bestC = clf.best_params_['C']
118     bestGamma = clf.best_params_['gamma']
119     print " "
120
121     trmeans = clf.cv_results_['mean_train_score']
122     trstds = clf.cv_results_['std_train_score']
123     means = clf.cv_results_['mean_test_score']
124     stds = clf.cv_results_['std_test_score']
125     c_para = clf.cv_results_['param_C']
126     gamma_para = clf.cv_results_['param_gamma']
127
128     # accuracy scores & Best accuracy scores
129     for c_par, g_par, trmean, trstd, mean, std, params in zip(c_para,
    gamma_para, trmeans, trstds, means, stds, clf.cv_results_['params']):
130         print("STAGES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
131               % (trmean, trstd * 2, mean, std * 2, params))
132
133         csv_row = csv_row + 1
134
135         final_stages_svmrbf[csv_row,0] = number_of_genes-1
136         final_stages_svmrbf[csv_row,1] = c_par
137         final_stages_svmrbf[csv_row,2] = g_par
138         final_stages_svmrbf[csv_row,3] = mean
139         final_stages_svmrbf[csv_row,4] = std * 2
140
141         if ((c_par == bestC) and (g_par == bestGamma)):
142             csv_row_best = csv_row_best + 1
143             final_stages_svmrbf_best[csv_row_best,0] = number_of_genes-1
144             final_stages_svmrbf_best[csv_row_best,1] = bestC
145             final_stages_svmrbf_best[csv_row_best,2] = bestGamma
```

```
146             final_stages_svmrbf_best[csv_row_best,3] = mean
147             final_stages_svmrbf_best[csv_row_best,4] = std * 2
148
149 np.savetxt("STAGES_PCA_SVM-RBF_all.csv", final_stages_svmrbf,
    delimiter=",",fmt='%.12f')
150 np.savetxt("STAGES_PCA_SVM-RBF_best.csv", final_stages_svmrbf_best,
    delimiter=",",fmt='%.12f')
151
152 print 'STAGES END!'
```

9.2 Fisher Score & SVM Linear / SVM RBF

9.2.1 Grades: Fisher Score & SVM Linear

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή Fisher Score (Grades)
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με την βαθμολογία του Fisher
   Score
05 Εφαρμογή SVM Linear για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 GRADES: Κλάσεις Στόχοι -> Early Grade = 0 και Late Grade = 1
07 """
08 from pandas import read_csv
09 import numpy as np
10 from sklearn import svm
11 from sklearn.model_selection import GridSearchCV
12
13 # Read data Clinicopathological_data.txt
14 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
15 data1_matrix = data1.as_matrix()
16
17 # Read data Liver_Cancer_RNA_seq_data.txt
18 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
19 data2 = np.array(data2)
20
21 # targets for Grades (0-1)
22 map_dict = {'Early_Grade':0, 'Late_Grade':1}
23 targets_grades = np.array([map_dict[key] for key in
   data1['Grades'].values])
24
25 grades = targets_grades.astype('float64')
26
27 ### GRADES FISHER SCORE ###################################
28 fisher_score_grades = np.array([])
29 for icol in range(data2.shape[1]):
30     early_grade_array = np.array([]) #0
31     late_grade_array = np.array([])  #1
32
```

```
33      for irow in range(data2.shape[0]):
34          if grades[irow] == 0:
35              early_grade_array =  np.append(early_grade_array,
    [data2[irow,icol]], axis=0)
36          if grades[irow] == 1:
37              late_grade_array =  np.append(late_grade_array,
    [data2[irow,icol]], axis=0)
38
39      fs = ((np.mean(early_grade_array) - np.mean(late_grade_array))**2) /
    (np.var(early_grade_array) + np.var(late_grade_array))
40      fisher_score_grades =  np.append(fisher_score_grades, [fs], axis=0)
41
42  grades_max = np.argsort(-fisher_score_grades)
43  np.savetxt("fisher-score_GRADES.csv", fisher_score_grades,
    delimiter=",",fmt='%.12f')
44  ### END GRADES FISHER SCORE ################################
45
46  ### PREPARE DATA FOR SVM ###################################
47  # merge targets-grades & data2
48  data_all = np.column_stack((grades,data2))
49  data_all_grades = np.array([])
50  data_all_grades = data_all
51  data_rows = data_all_grades.shape[0] # number of rows
52  data_cols = data_all_grades.shape[1] # numer of cols
53
54  # sorted array according to fisher score (Grades (0-1) + data)
55  # create data_grades
56  data_grades = np.empty([data_rows,data_cols]);
57  for icol in range(data_cols):
58      if icol==0:
59          data_grades[:,icol] = data_all_grades[:,icol]
60      if icol>0:
61          grades_max_row = grades_max[icol-1]
62          grades_max_row = grades_max_row + 1
63          data_grades[:,icol] = data_all_grades[:,grades_max_row]
64
65  ### END - PREPARE DATA FOR SVM #############################
66
67  ### SVM LINEAR GRADES ######################################
68  # array with all final results - for save in csv
69  final_grades_svmlinear = np.zeros(shape=(500,4))
70  csv_row = -1
71  # array with best final results - for save in csv
72  final_grades_svmlinear_best = np.zeros(shape=(50,4))
73  csv_row_best = -1
74
75  # NUMBER OF GENES
76  for number_of_genes in range(11,511,10):
77
```

```
78      print 'GRADES -------------------- NUMBER OF GENES =
    {0}'.format(number_of_genes-1)
79      data_for_svm_linear = np.empty([data_rows, number_of_genes]);
80
81      # array with targets + first 'number_of_genes' genes
82      # create data_for_svm_linear
83      for icol in range(number_of_genes):
84          data_for_svm_linear[:,icol] = data_grades[:,icol]
85
86      shuffle_data_targets = np.random.permutation(data_for_svm_linear)
87      grades_data = np.delete(shuffle_data_targets, (0), axis=1)
88      grades_targets = shuffle_data_targets[:,0]
89
90      parameter_candidates = [
91              {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
    'kernel': ['linear']}]
92
93      clf = GridSearchCV(estimator=svm.SVC(),
    param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
    return_train_score=True)
94      clf.fit(grades_data, grades_targets)
95
96      print(clf.best_params_)
97      bestC = clf.best_params_['C']
98      print " "
99
100     trmeans = clf.cv_results_['mean_train_score']
101     trstds = clf.cv_results_['std_train_score']
102     means = clf.cv_results_['mean_test_score']
103     stds = clf.cv_results_['std_test_score']
104     c_para = clf.cv_results_['param_C']
105
106     # accuracy scores & Best accuracy scores
107     for c_par, trmean, trstd, mean, std, params in zip(c_para, trmeans,
    trstds, means, stds, clf.cv_results_['params']):
108         print("GRADES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
109               % (trmean, trstd * 2, mean, std * 2, params))
110
111         csv_row = csv_row + 1
112
113         final_grades_svmlinear[csv_row,0] = number_of_genes-1
114         final_grades_svmlinear[csv_row,1] = c_par
115         final_grades_svmlinear[csv_row,2] = mean
116         final_grades_svmlinear[csv_row,3] = std * 2
117
118         if (c_par == bestC):
119             csv_row_best = csv_row_best + 1
```

```
120             final_grades_svmlinear_best[csv_row_best,0] =
    number_of_genes-1
121             final_grades_svmlinear_best[csv_row_best,1] = bestC
122             final_grades_svmlinear_best[csv_row_best,2] = mean
123             final_grades_svmlinear_best[csv_row_best,3] = std * 2
124
125 np.savetxt("GRADES_fisher_SVM-Linear_all.csv", final_grades_svmlinear,
    delimiter=",",fmt='%.12f')
126 np.savetxt("GRADES_fisher_SVM-Linear_best.csv",
    final_grades_svmlinear_best, delimiter=",",fmt='%.12f')
127
128 print 'END GRADES!'
```

9.2.2 Stages: Fisher Score & SVM RBF

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή Fisher Score (Stages)
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με την βαθμολογία του Fisher
    Score
05 Εφαρμογή SVM RBF για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 STAGES: Κλάσεις Στόχοι -> Early Stage = 0 και Late Stage = 1
07 """
08 from pandas import read_csv
09 import numpy as np
10 from sklearn import svm
11 from sklearn.model_selection import GridSearchCV
12
13 # Read data Clinicopathological_data.txt
14 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
15 data1_matrix = data1.as_matrix()
16
17 # Read data Liver_Cancer_RNA_seq_data.txt
18 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
19 data2 = np.array(data2)
20
21 # targets for Stages (0-1)
22 map_dict = {'Early_Stage':0, 'Late_Stage':1}
23 targets_stages = np.array([map_dict[key] for key in
    data1['Stages'].values])
24
25 stages = targets_stages.astype('float64')
26
27 ### STAGES FISHER SCORE ###################################
28 fisher_score_stages = np.array([])
29 for icol in range(data2.shape[1]):
30     early_stage_array = np.array([]) #0
31     late_stage_array = np.array([])  #1
32
```

```
33      for irow in range(data2.shape[0]):
34          if stages[irow] == 0:
35              early_stage_array =  np.append(early_stage_array,
    [data2[irow,icol]], axis=0)
36          if stages[irow] == 1:
37              late_stage_array =  np.append(late_stage_array,
    [data2[irow,icol]], axis=0)
38
39      fs = ((np.mean(early_stage_array) - np.mean(late_stage_array))**2) /
    (np.var(early_stage_array) + np.var(late_stage_array))
40      fisher_score_stages =  np.append(fisher_score_stages, [fs], axis=0)
41
42  stages_max = np.argsort(-fisher_score_stages)
43  np.savetxt("fisher-score_STAGES.csv", fisher_score_stages,
    delimiter=",",fmt='%.12f')
44  ### END STAGES FISHER SCORE ################################
45
46  ### PREPARE DATA FOR SVM ###################################
47  # merge targets-stages & data2
48  data_all = np.column_stack((stages,data2))
49  data_all_stages = np.array([])
50  data_all_stages = data_all
51  data_rows = data_all_stages.shape[0] #number of rows
52  data_cols = data_all_stages.shape[1] #numer of cols
53
54  # sorted array according to fisher score (Stages (0-1) + data)
55  # create data_stages
56  data_stages = np.empty([data_rows,data_cols]);
57  for icol in range(data_cols):
58      if icol==0:
59          data_stages[:,icol] = data_all_stages[:,icol]
60      if icol>0:
61          stages_max_row = stages_max[icol-1]
62          stages_max_row = stages_max_row + 1
63          data_stages[:,icol] = data_all_stages[:,stages_max_row]
64
65  ### END - PREPARE DATA FOR SVM #############################
66
67  ### SVM LINEAR STAGES ######################################
68  # array with all final results - for save in csv
69  final_stages_svmrbf = np.zeros(shape=(6800,5))
70  csv_row = -1
71  # array with best final results - for save in csv
72  final_stages_svmrbf_best = np.zeros(shape=(50,5))
73  csv_row_best = -1
74
75  # NUMBER OF GENES
76  for number_of_genes in range(11,511,10):
77
```

```
78      print 'STAGES -------------------- NUMBER OF GENES =
    {0}'.format(number_of_genes-1)
79      data_for_svm_rbf = np.empty([data_rows, number_of_genes]);
80
81      # array with targets + first 'number_of_genes' genes
82      # create data_for_svm_rbf
83      for icol in range(number_of_genes):
84          data_for_svm_rbf[:,icol] = data_stages[:,icol]
85
86      shuffle_data_targets = np.random.permutation(data_for_svm_rbf)
87      stages_data = np.delete(shuffle_data_targets, (0), axis=1)
88      stages_targets = shuffle_data_targets[:,0]
89
90      parameter_candidates = [
91              {'kernel': ['rbf'],
92               'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
    1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
93               'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
94               'tol': [1e-10]}]
95
96      clf = GridSearchCV(estimator=svm.SVC(),
    param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
    return_train_score=True)
97      clf.fit(stages_data, stages_targets)
98
99      print(clf.best_params_)
100     bestC = clf.best_params_['C']
101     bestGamma = clf.best_params_['gamma']
102     print " "
103
104     trmeans = clf.cv_results_['mean_train_score']
105     trstds = clf.cv_results_['std_train_score']
106     means = clf.cv_results_['mean_test_score']
107     stds = clf.cv_results_['std_test_score']
108     c_para = clf.cv_results_['param_C']
109     gamma_para = clf.cv_results_['param_gamma']
110
111     # accuracy scores & Best accuracy scores
112     for c_par, g_par, trmean, trstd, mean, std, params in zip(c_para,
    gamma_para, trmeans, trstds, means, stds, clf.cv_results_['params']):
113         print("STAGES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
114               % (trmean, trstd * 2, mean, std * 2, params))
115
116         csv_row = csv_row + 1
117
118         final_stages_svmrbf[csv_row,0] = number_of_genes-1
119         final_stages_svmrbf[csv_row,1] = c_par
120         final_stages_svmrbf[csv_row,2] = g_par
```

```
121         final_stages_svmrbf[csv_row,3] = mean
122         final_stages_svmrbf[csv_row,4] = std * 2
123 
124         if ((c_par == bestC) and (g_par == bestGamma)):
125             csv_row_best = csv_row_best + 1
126             final_stages_svmrbf_best[csv_row_best,0] = number_of_genes-1
127             final_stages_svmrbf_best[csv_row_best,1] = bestC
128             final_stages_svmrbf_best[csv_row_best,2] = bestGamma
129             final_stages_svmrbf_best[csv_row_best,3] = mean
130             final_stages_svmrbf_best[csv_row_best,4] = std * 2
131 
132 np.savetxt("STAGES_fisher_SVM-RBF_all.csv", final_stages_svmrbf,
    delimiter=",",fmt='%.12f')
133 np.savetxt("STAGES_fisher_SVM-RBF_best.csv", final_stages_svmrbf_best,
    delimiter=",",fmt='%.12f')
134 
135 print 'STAGES END!'
```

9.3 Mutual Information (MI) & SVM Linear / SVM RBF

9.3.1 Grades: Mutual Information & SVM Linear

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή Mutual Information (Information Gain) (Grades)
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με την βαθμολογία του
   Information Gain
05 Εφαρμογή SVM Linear για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 GRADES: Κλάσεις Στόχοι -> Early Grade = 0 και Late Grade = 1
07 """
08 from pandas import read_csv
09 import numpy as np
10 from sklearn.metrics.cluster import normalized_mutual_info_score
11 from sklearn import svm
12 from sklearn.model_selection import GridSearchCV
13 
14 # Read data Clinicopathological_data.txt
15 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
16 data1_matrix = data1.as_matrix()
17 
18 # Read data Liver_Cancer_RNA_seq_data.txt
19 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
20 data2 = np.array(data2)
21 # data2_copy for Discretization & Information Gain
22 data2_copy = np.array(data2)
23 
24 # targets Grades (0-1)
25 map_dict = {'Early_Grade':0, 'Late_Grade':1}
```

```
targets_grades = np.array([map_dict[key] for key in
data1['Grades'].values])

### Discretization ########################################
frequencies_all, bin_edges_all = np.histogram(data2, bins=20, range=(3,
24))

# bins average values
bin_edges_all_avg = np.array([])
for irow in range(bin_edges_all.shape[0]-1):
    # print irow
    avg = (bin_edges_all[irow] + bin_edges_all[irow+1]) / 2
    bin_edges_all_avg = np.append(bin_edges_all_avg, [avg], axis=0)

# replace gene values with the number of the bin group
for icol in range(data2.shape[1]):
    data2_copy[:,icol] = np.digitize(data2_copy[:,icol],bin_edges_all)

# replace the bin group with the avg value of the bin group
data_int = np.array([])
data_int = data2_copy.astype('int64')
for icol in range(data_int.shape[1]):
    for irow in range(data_int.shape[0]):
        bin_cell = data_int[irow,icol]
        bin_cell = bin_cell - 1
        data2_copy[irow,icol] = bin_edges_all_avg[bin_cell]
### End Discretization ####################################

### Information Gain ######################################
ing_grades = np.array([])
for icol in range(data2_copy.shape[1]):
    # information gain for grades
    info_gain_grades =
normalized_mutual_info_score(data2_copy[:,icol],targets_grades)
    ing_grades = np.append(ing_grades, [info_gain_grades], axis=0)

# Sorted Genes according to Information Gain
grades_max_ig = np.argsort(-ing_grades)
### End Information Gain ##################################

### PREPARE DATA FOR SVM ##################################
# targets Grades (0-1) + data
data_all_grades = np.array([])
targets_grades_f = targets_grades.astype('float64')
data_all_grades = np.column_stack((targets_grades_f,data2))
data_rows = data_all_grades.shape[0] # number of rows
data_cols = data_all_grades.shape[1] # numer of cols

# sorted array according to information gain score (Grades (0-1) + data)
```

```
72 # create data_grades
73 data_grades = np.empty([data_rows,data_cols]);
74 for icol in range(data_cols):
75     if icol==0:
76         data_grades[:,icol] = data_all_grades[:,icol]
77     if icol>0:
78         grades_max_row = grades_max_ig[icol-1]
79         grades_max_row = grades_max_row + 1
80         data_grades[:,icol] = data_all_grades[:,grades_max_row]
81 
82 ### END - PREPARE DATA FOR SVM ############################
83 
84 ### SVM LINEAR GRADES #####################################
85 # array with all final results - for save in csv
86 final_grades_svmlinear = np.zeros(shape=(500,4))
87 csv_row = -1
88 # array with best final results - for save in csv
89 final_grades_svmlinear_best = np.zeros(shape=(50,4))
90 csv_row_best = -1
91 
92 # NUMBER OF GENES
93 for number_of_genes in range(11,511,10):
94 
95     print 'GRADES -------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
96     data_for_svm_linear = np.empty([data_rows, number_of_genes]);
97 
98     # array with targets + first 'number_of_genes' genes
99     # create data_for_svm_linear
100    for icol in range(number_of_genes):
101        data_for_svm_linear[:,icol] = data_grades[:,icol]
102 
103    shuffle_data_targets = np.random.permutation(data_for_svm_linear)
104    grades_data = np.delete(shuffle_data_targets, (0), axis=1)
105    grades_targets = shuffle_data_targets[:,0]
106 
107    parameter_candidates = [
108            {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
   'kernel': ['linear']}]
109 
110    clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
111    clf.fit(grades_data, grades_targets)
112 
113    print(clf.best_params_)
114    bestC = clf.best_params_['C']
115    print " "
116 
```

```
117     trmeans = clf.cv_results_['mean_train_score']
118     trstds = clf.cv_results_['std_train_score']
119     means = clf.cv_results_['mean_test_score']
120     stds = clf.cv_results_['std_test_score']
121     c_para = clf.cv_results_['param_C']
122 
123     # accuracy scores & Best accuracy scores
124     for c_par, trmean, trstd, mean, std, params in zip(c_para, trmeans,
    trstds, means, stds, clf.cv_results_['params']):
125         print("GRADES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
126               % (trmean, trstd * 2, mean, std * 2, params))
127 
128         csv_row = csv_row + 1
129 
130         final_grades_svmlinear[csv_row,0] = number_of_genes-1
131         final_grades_svmlinear[csv_row,1] = c_par
132         final_grades_svmlinear[csv_row,2] = mean
133         final_grades_svmlinear[csv_row,3] = std * 2
134 
135         if (c_par == bestC):
136             csv_row_best = csv_row_best + 1
137             final_grades_svmlinear_best[csv_row_best,0] =
    number_of_genes-1
138             final_grades_svmlinear_best[csv_row_best,1] = bestC
139             final_grades_svmlinear_best[csv_row_best,2] = mean
140             final_grades_svmlinear_best[csv_row_best,3] = std * 2
141 
142 np.savetxt("GRADES_infogain_SVM-Linear_all.csv", final_grades_svmlinear,
    delimiter=",",fmt='%.12f')
143 np.savetxt("GRADES_infogain_SVM-Linear_best.csv",
    final_grades_svmlinear_best, delimiter=",",fmt='%.12f')
144 
145 print 'END GRADES!'
```

9.3.2 Stages: Mutual Information & SVM RBF

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή Mutual Information (Information Gain) (Stages)
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με την βαθμολογία του
   Information Gain
05 Εφαρμογή SVM RBF για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 STAGES: Κλάσεις Στόχοι -> Early Stage = 0 και Late Stage = 1
07 """
08 from pandas import read_csv
09 import numpy as np
10 from sklearn.metrics.cluster import normalized_mutual_info_score
11 from sklearn import svm
```

```
from sklearn.model_selection import GridSearchCV

# Read data Clinicopathological_data.txt
data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
data1_matrix = data1.as_matrix()

# Read data Liver_Cancer_RNA_seq_data.txt
data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
data2 = np.array(data2)
# data2_copy for Discretization & Information Gain
data2_copy = np.array(data2)

# targets Stages (0-1)
map_dict = {'Early_Stage':0, 'Late_Stage':1}
targets_stages = np.array([map_dict[key] for key in
data1['Stages'].values])

### Discretization ########################################
frequencies_all, bin_edges_all = np.histogram(data2, bins=20, range=(3,
24))

# bins average values
bin_edges_all_avg = np.array([])
for irow in range(bin_edges_all.shape[0]-1):
    avg = (bin_edges_all[irow] + bin_edges_all[irow+1]) / 2
    bin_edges_all_avg = np.append(bin_edges_all_avg, [avg], axis=0)

# replace gene values with the number of the bin group
for icol in range(data2.shape[1]):
    data2_copy[:,icol] = np.digitize(data2_copy[:,icol],bin_edges_all)

# replace the bin group with the avg value of the bin group
data_int = np.array([])
data_int = data2_copy.astype('int64')
for icol in range(data_int.shape[1]):
    for irow in range(data_int.shape[0]):
        bin_cell = data_int[irow,icol]
        bin_cell = bin_cell - 1
        data2_copy[irow,icol] = bin_edges_all_avg[bin_cell]
### End Discretization ####################################

### Information Gain ######################################
ing_stages = np.array([])
for icol in range(data2_copy.shape[1]):
    # information gain for stages
    info_gain_stages =
normalized_mutual_info_score(data2_copy[:,icol],targets_stages)
    ing_stages = np.append(ing_stages, [info_gain_stages], axis=0)

```

```
58 # Sorted Genes according to Information Gain
59 stages_max_ig = np.argsort(-ing_stages)
60 ### End Information Gain ###################################
61
62 ### PREPARE DATA FOR SVM ###################################
63 # targets Stages (0-1) + data
64 data_all_stages = np.array([])
65 targets_stages_f = targets_stages.astype('float64')
66 data_all_stages = np.column_stack((targets_stages_f,data2))
67 data_rows = data_all_stages.shape[0] # number of rows
68 data_cols = data_all_stages.shape[1] # numer of cols
69
70 # sorted array according to information gain score (Stages (0-1) + data)
71 # create data_stages
72 data_stages = np.empty([data_rows,data_cols]);
73 for icol in range(data_cols):
74     if icol==0:
75         data_stages[:,icol] = data_all_stages[:,icol]
76     if icol>0:
77         stages_max_row = stages_max_ig[icol-1]
78         stages_max_row = stages_max_row + 1
79         data_stages[:,icol] = data_all_stages[:,stages_max_row]
80
81 ### END - PREPARE DATA FOR SVM ##############################
82
83 ### SVM RBF STAGES ##########################################
84 # array with all final results - for save in csv
85 final_stages_svmrbf = np.zeros(shape=(6800,5))
86 csv_row = -1
87 # array with best final results - for save in csv
88 final_stages_svmrbf_best = np.zeros(shape=(50,5))
89 csv_row_best = -1
90
91 # NUMBER OF GENES
92 for number_of_genes in range(11,511,10):
93
94     print 'STAGES -------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
95     data_for_svm_rbf = np.empty([data_rows, number_of_genes]);
96
97     # array with targets + first 'number_of_genes' genes
98     # create data_for_svm_rbf
99     for icol in range(number_of_genes):
100        data_for_svm_rbf[:,icol] = data_stages[:,icol]
101
102    shuffle_data_targets = np.random.permutation(data_for_svm_rbf)
103    stages_data = np.delete(shuffle_data_targets, (0), axis=1)
104    stages_targets = shuffle_data_targets[:,0]
105
```

```
106     parameter_candidates = [
107             {'kernel': ['rbf'],
108              'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
    1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
109              'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
110              'tol': [1e-10]}]
111
112     clf = GridSearchCV(estimator=svm.SVC(),
    param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
    return_train_score=True)
113     clf.fit(stages_data, stages_targets)
114
115     print(clf.best_params_)
116     bestC = clf.best_params_['C']
117     bestGamma = clf.best_params_['gamma']
118     print " "
119
120     trmeans = clf.cv_results_['mean_train_score']
121     trstds = clf.cv_results_['std_train_score']
122     means = clf.cv_results_['mean_test_score']
123     stds = clf.cv_results_['std_test_score']
124     c_para = clf.cv_results_['param_C']
125     gamma_para = clf.cv_results_['param_gamma']
126
127     # accuracy scores & Best accuracy scores
128     for c_par, g_par, trmean, trstd, mean, std, params in zip(c_para,
    gamma_para, trmeans, trstds, means, stds, clf.cv_results_['params']):
129         print("STAGES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
130               % (trmean, trstd * 2, mean, std * 2, params))
131
132         csv_row = csv_row + 1
133
134         final_stages_svmrbf[csv_row,0] = number_of_genes-1
135         final_stages_svmrbf[csv_row,1] = c_par
136         final_stages_svmrbf[csv_row,2] = g_par
137         final_stages_svmrbf[csv_row,3] = mean
138         final_stages_svmrbf[csv_row,4] = std * 2
139
140         if ((c_par == bestC) and (g_par == bestGamma)):
141             csv_row_best = csv_row_best + 1
142             final_stages_svmrbf_best[csv_row_best,0] = number_of_genes-1
143             final_stages_svmrbf_best[csv_row_best,1] = bestC
144             final_stages_svmrbf_best[csv_row_best,2] = bestGamma
145             final_stages_svmrbf_best[csv_row_best,3] = mean
146             final_stages_svmrbf_best[csv_row_best,4] = std * 2
147
148 np.savetxt("STAGES_infogain_SVM-RBF_all.csv", final_stages_svmrbf,
    delimiter=",",fmt='%.12f')
```

```
149 np.savetxt("STAGES_infogain_SVM-RBF_best.csv", final_stages_svmrbf_best,
    delimiter=",",fmt='%.12f')
150
151 print 'STAGES END!'
```

9.4 KS 2 Samples & SVM Linear / SVM RBF

9.4.1 Grades: KS 2 Samples & SVM Linear

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή Kolmogorov Smirnov 2 Samples Test (Grades)
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με την βαθμολογία του KS
   2Samples (p-value)
05 Εφαρμογή SVM Linear για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 GRADES: Κλάσεις Στόχοι -> Early Grade = 0 και Late Grade = 1
07 """
08 from pandas import read_csv
09 import numpy as np
10 from scipy import stats
11 from sklearn import svm
12 from sklearn.model_selection import GridSearchCV
13
14 # Read data Clinicopathological_data.txt
15 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
16 data1_matrix = data1.as_matrix()
17
18 # Read data Liver_Cancer_RNA_seq_data.txt
19 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
20 data2 = np.array(data2)
21
22 ### KS 2Sample ##############################################
23 # merge data1_matrix & data2
24 data_all = np.concatenate((data1_matrix,data2), axis=1)
25 # instances with target class Early Grade
26 data_early_grade = data_all[data_all[:,2] == 'Early_Grade']
27 final_early_grade = np.delete(data_early_grade, (0,1,2,3), axis=1)
28 final_early_grade = final_early_grade.astype('float64')
29 # instances with target class Late Grade
30 data_late_grade = data_all[data_all[:,2] == 'Late_Grade']
31 final_late_grade = np.delete(data_late_grade, (0,1,2,3), axis=1)
32 final_late_grade = final_late_grade.astype('float64')
33
34 p_value_grade_array = np.array([])
35 # for every gene -->
36 # 1st Sample: values with target class Early Grade
37 # 2nd Sample: values with target class Late Grade
38 for icol in range(final_early_grade.shape[1]):
```

```
39      gene_early_grade = np.array([])
40      gene_late_grade = np.array([])
41      gene_early_grade = final_early_grade[:,icol]
42      gene_late_grade =  final_late_grade[:,icol]
43      ks_statistic_grade, pvalue_grade = stats.ks_2samp(gene_early_grade,
    gene_late_grade)
44      p_value_grade_array =  np.append(p_value_grade_array,
    [pvalue_grade], axis=0)
45
46  # Sorted Genes according to p-value (information gain)
47  grades_min = np.argsort(p_value_grade_array)
48  ### end KS 2Sample ########################################
49
50  ### PREPARE DATA FOR SVM ##################################
51  # targets Grades (0-1)
52  map_dict = {'Early_Grade':0, 'Late_Grade':1}
53  targets_grades = np.array([map_dict[key] for key in
    data1['Grades'].values])
54  # targets Grades (0-1) + data
55  data_all_grades = np.array([])
56  targets_grades_f = targets_grades.astype('float64')
57  data_all_grades = np.column_stack((targets_grades_f,data2))
58  data_rows = data_all_grades.shape[0] # number of rows
59  data_cols = data_all_grades.shape[1] # numer of cols
60
61  # sorted array according to p-value (ks-2sample) (Grades (0-1) + data)
62  # create data_grades
63  data_grades = np.empty([data_rows,data_cols]);
64  for icol in range(data_cols):
65      if icol==0:
66          data_grades[:,icol] = data_all_grades[:,icol]
67      if icol>0:
68          grades_min_row = grades_min[icol-1]
69          grades_min_row = grades_min_row + 1
70          data_grades[:,icol] = data_all_grades[:,grades_min_row]
71
72  ### END - PREPARE DATA FOR SVM ############################
73
74  ### SVM LINEAR GRADES #####################################
75  # array with all final results - for save in csv
76  final_grades_svmlinear = np.zeros(shape=(500,4))
77  csv_row = -1
78  # array with best final results - for save in csv
79  final_grades_svmlinear_best = np.zeros(shape=(50,4))
80  csv_row_best = -1
81
82  # NUMBER OF GENES
83  for number_of_genes in range(11,511,10):
84
```

```
85      print 'GRADES ------------------- NUMBER OF GENES =
    {0}'.format(number_of_genes-1)
86      data_for_svm_linear = np.empty([data_rows, number_of_genes]);
87
88      # array with targets + first 'number_of_genes' genes
89      # create data_for_svm_linear
90      for icol in range(number_of_genes):
91          data_for_svm_linear[:,icol] = data_grades[:,icol]
92
93      shuffle_data_targets = np.random.permutation(data_for_svm_linear)
94      grades_data = np.delete(shuffle_data_targets, (0), axis=1)
95      grades_targets = shuffle_data_targets[:,0]
96
97      parameter_candidates = [
98              {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
    'kernel': ['linear']}]
99
100     clf = GridSearchCV(estimator=svm.SVC(),
    param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
    return_train_score=True)
101     clf.fit(grades_data, grades_targets)
102
103     print(clf.best_params_)
104     bestC = clf.best_params_['C']
105     print " "
106
107     trmeans = clf.cv_results_['mean_train_score']
108     trstds = clf.cv_results_['std_train_score']
109     means = clf.cv_results_['mean_test_score']
110     stds = clf.cv_results_['std_test_score']
111     c_para = clf.cv_results_['param_C']
112
113     # accuracy scores & Best accuracy scores
114     for c_par, trmean, trstd, mean, std, params in zip(c_para, trmeans,
    trstds, means, stds, clf.cv_results_['params']):
115         print("GRADES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
116               % (trmean, trstd * 2, mean, std * 2, params))
117
118         csv_row = csv_row + 1
119
120         final_grades_svmlinear[csv_row,0] = number_of_genes-1
121         final_grades_svmlinear[csv_row,1] = c_par
122         final_grades_svmlinear[csv_row,2] = mean
123         final_grades_svmlinear[csv_row,3] = std * 2
124
125         if (c_par == bestC):
126             csv_row_best = csv_row_best + 1
```

```
127             final_grades_svmlinear_best[csv_row_best,0] =
    number_of_genes-1
128             final_grades_svmlinear_best[csv_row_best,1] = bestC
129             final_grades_svmlinear_best[csv_row_best,2] = mean
130             final_grades_svmlinear_best[csv_row_best,3] = std * 2
131
132 np.savetxt("GRADES_ks2sample_SVM-Linear_all.csv",
    final_grades_svmlinear, delimiter=",",fmt='%.5f')
133 np.savetxt("GRADES_ks2sample_SVM-Linear_best.csv",
    final_grades_svmlinear_best, delimiter=",",fmt='%.5f')
134
135 print 'END GRADES!'
```

9.4.2 Stages: KS 2 Samples & SVM RBF

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή Kolmogorov Smirnov 2 Samples Test (Stages)
04 Αξιολόγηση και Ταξινόμηση Γονιδίων σύμφωνα με την βαθμολογία του KS
   2Samples (p-value)
05 Εφαρμογή SVM RBF για τα πρώτα καλύτερα 500 γονίδια με βήμα 10
06 STAGES: Κλάσεις Στόχοι -> Early Stage = 0 και Late Stage = 1
07 """
08 from pandas import read_csv
09 import numpy as np
10 from scipy import stats
11 from sklearn import svm
12 from sklearn.model_selection import GridSearchCV
13
14 # Read data Clinicopathological_data.txt
15 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
16 data1_matrix = data1.as_matrix()
17
18 # Read data Liver_Cancer_RNA_seq_data.txt
19 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
20 data2 = np.array(data2)
21
22 ### KS 2Sample ##############################################
23 # merge data1_matrix & data2
24 data_all = np.concatenate((data1_matrix,data2), axis=1)
25 # instances with target class Early Stage
26 data_early_stage = data_all[data_all[:,3] == 'Early_Stage']
27 final_early_stage = np.delete(data_early_stage, (0,1,2,3), axis=1)
28 final_early_stage = final_early_stage.astype('float64')
29 # instances with target class Late Stage
30 data_late_stage = data_all[data_all[:,3] == 'Late_Stage']
31 final_late_stage = np.delete(data_late_stage, (0,1,2,3), axis=1)
32 final_late_stage = final_late_stage.astype('float64')
33
```

```
p_value_stage_array = np.array([])
# for every gene -->
# 1st Sample: values with target class Early Stage
# 2nd Sample: values with target class Late Stage
for icol in range(final_early_stage.shape[1]):
    gene_early_stage = np.array([])
    gene_late_stage = np.array([])
    gene_early_stage = final_early_stage[:,icol]
    gene_late_stage =  final_late_stage[:,icol]
    ks_statistic_stage, pvalue_stage = stats.ks_2samp(gene_early_stage,
gene_late_stage)
    p_value_stage_array =  np.append(p_value_stage_array,
[pvalue_stage], axis=0)

# Sorted Genes according to p-value (information gain)
stages_min = np.argsort(p_value_stage_array)
### end KS 2Sample ########################################

### PREPARE DATA FOR SVM ##################################
# targets Stages (0-1)
map_dict = {'Early_Stage':0, 'Late_Stage':1}
targets_stages = np.array([map_dict[key] for key in
data1['Stages'].values])
# targets Stages (0-1) + data
data_all_stages = np.array([])
targets_stages_f = targets_stages.astype('float64')
data_all_stages = np.column_stack((targets_stages_f,data2))
data_rows = data_all_stages.shape[0] # number of rows
data_cols = data_all_stages.shape[1] # numer of cols

# sorted array according to p-value (ks-2sample) (Stages (0-1) + data)
# create data_stages
data_stages = np.empty([data_rows,data_cols]);
for icol in range(data_cols):
    if icol==0:
        data_stages[:,icol] = data_all_stages[:,icol]
    if icol>0:
        stages_min_row = stages_min[icol-1]
        stages_min_row = stages_min_row + 1
        data_stages[:,icol] = data_all_stages[:,stages_min_row]

### END - PREPARE DATA FOR SVM ############################

### SVM RBF STAGES ########################################
# array with all final results - for save in csv
final_stages_svmrbf = np.zeros(shape=(6800,5))
csv_row = -1
# array with best final results - for save in csv
final_stages_svmrbf_best = np.zeros(shape=(50,5))
```

```
csv_row_best = -1

# NUMBER OF GENES
for number_of_genes in range(11,511,10):

    print 'STAGES -------------------- NUMBER OF GENES =
{0}'.format(number_of_genes-1)
    data_for_svm_rbf = np.empty([data_rows, number_of_genes]);

    # array with targets + first 'number_of_genes' genes
    # create data_for_svm_rbf
    for icol in range(number_of_genes):
        data_for_svm_rbf[:,icol] = data_stages[:,icol]

    shuffle_data_targets = np.random.permutation(data_for_svm_rbf)
    stages_data = np.delete(shuffle_data_targets, (0), axis=1)
    stages_targets = shuffle_data_targets[:,0]

    parameter_candidates = [
            {'kernel': ['rbf'],
             'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
             'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
             'tol': [1e-10]}]

    clf = GridSearchCV(estimator=svm.SVC(),
param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
return_train_score=True)
    clf.fit(stages_data, stages_targets)

    print(clf.best_params_)
    bestC = clf.best_params_['C']
    bestGamma = clf.best_params_['gamma']
    print " "

    trmeans = clf.cv_results_['mean_train_score']
    trstds = clf.cv_results_['std_train_score']
    means = clf.cv_results_['mean_test_score']
    stds = clf.cv_results_['std_test_score']
    c_para = clf.cv_results_['param_C']
    gamma_para = clf.cv_results_['param_gamma']

    # accuracy scores & Best accuracy scores
    for c_par, g_par, trmean, trstd, mean, std, params in zip(c_para,
gamma_para, trmeans, trstds, means, stds, clf.cv_results_['params']):
        print("STAGES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
%0.03f) for %r"
              % (trmean, trstd * 2, mean, std * 2, params))

```

```
123         csv_row = csv_row + 1
124
125         final_stages_svmrbf[csv_row,0] = number_of_genes-1
126         final_stages_svmrbf[csv_row,1] = c_par
127         final_stages_svmrbf[csv_row,2] = g_par
128         final_stages_svmrbf[csv_row,3] = mean
129         final_stages_svmrbf[csv_row,4] = std * 2
130
131         if ((c_par == bestC) and (g_par == bestGamma)):
132             csv_row_best = csv_row_best + 1
133             final_stages_svmrbf_best[csv_row_best,0] = number_of_genes-1
134             final_stages_svmrbf_best[csv_row_best,1] = bestC
135             final_stages_svmrbf_best[csv_row_best,2] = bestGamma
136             final_stages_svmrbf_best[csv_row_best,3] = mean
137             final_stages_svmrbf_best[csv_row_best,4] = std * 2
138
139 np.savetxt("STAGES_ks2sample_SVM-RBF_all.csv", final_stages_svmrbf,
    delimiter=",",fmt='%.12f')
140 np.savetxt("STAGES_ks2sample_SVM-RBF_best.csv",
    final_stages_svmrbf_best, delimiter=",",fmt='%.12f')
141
142 print 'STAGES END!'
```

9.5 ReliefF & SVM Linear / SVM RBF

9.5.1 Grades & Stages: Εφαρμογή ReliefF

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή ReliefF (GRADES & STAGES)
04 https://libraries.io/pypi/ReliefF/0.1.2
05 """
06 from sklearn.cross_validation import train_test_split
07 from ReliefF import ReliefF
08 from pandas import read_csv
09 import numpy as np
10 from numpy import array
11
12 # Read data Clinicopathological_data.txt
13 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
14 data1_matrix = data1.as_matrix()
15
16 # Read data Liver_Cancer_RNA_seq_data.txt
17 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
18 data2 = np.array(data2)
19
20 # targets Grades (0-1)
21 map_dict = {'Early_Grade':0, 'Late_Grade':1}
```

```
grades = np.array([map_dict[key] for key in data1['Grades'].values])
grades = grades.astype('int')

# targets Stages (0-1)
map_dict = {'Early_Stage':0, 'Late_Stage':1}
stages = np.array([map_dict[key] for key in data1['Stages'].values])
stages = stages.astype('int')

### ReliefF for Grades ####################################
results_idx_grades = np.zeros(shape=(500,50))
csv_col = -1

for nf in range(10, 510, 10):

    X_train, X_test, y_train, y_test = train_test_split(data2, grades)
    y_train = y_train.flatten()
    y_test = y_test.flatten()

    fs = ReliefF(n_neighbors=10, n_features_to_keep = nf)
    X_train = fs.fit_transform(X_train, y_train)
    X_test_subset = fs.transform(X_test)
    print(X_test.shape, X_test_subset.shape)

    trX_test = np.transpose(X_test)
    trX_test_subset = np.transpose(X_test_subset)

    trX_test = {tuple(row):i for i,row in enumerate(trX_test)}
    answer = []
    for i,row in enumerate(trX_test_subset):
        if tuple(row) in trX_test:
           answer.append((trX_test[tuple(row)], i))

    npa = array( answer )

    csv_col = csv_col + 1

    for row in range(0, nf, 1):
        results_idx_grades[row,csv_col] = npa[row,0]

np.savetxt("GRADES_relief_idx.csv", results_idx_grades,
delimiter=",",fmt='%.0f')

### ReliefF for Stages ####################################
results_idx_stages = np.zeros(shape=(500,50))
csv_col = -1

for nf in range(10, 510, 10):

    X_train, X_test, y_train, y_test = train_test_split(data2, stages)
```

```
    y_train = y_train.flatten()
    y_test = y_test.flatten()

    fs = ReliefF(n_neighbors=10, n_features_to_keep = nf)
    X_train = fs.fit_transform(X_train, y_train)
    X_test_subset = fs.transform(X_test)
    print(X_test.shape, X_test_subset.shape)

    trX_test = np.transpose(X_test)
    trX_test_subset = np.transpose(X_test_subset)

    trX_test = {tuple(row):i for i,row in enumerate(trX_test)}
    answer = []
    for i,row in enumerate(trX_test_subset):
        if tuple(row) in trX_test:
           answer.append((trX_test[tuple(row)], i))

    npa = array( answer )

    csv_col = csv_col + 1

    for row in range(0, nf, 1):
        results_idx_stages[row,csv_col] = npa[row,0]

np.savetxt("STAGES_relief_idx.csv", results_idx_stages,
delimiter=",",fmt='%.0f')
```

9.5.2 Grades (ReliefF): SVM Linear

```
# -*- coding: utf-8 -*-
"""
Εφαρμογή SVM Linear για τα πρώτα καλύτερα 500 γονίδια (ReliefF) με βήμα
10
GRADES: Κλάσεις Στόχοι -> Early Grade = 0 και Late Grade = 1
"""
from pandas import read_csv
from numpy import genfromtxt
import numpy as np
from sklearn import svm
from sklearn.model_selection import GridSearchCV

# Read data Clinicopathological_data.txt
data1 = read_csv('../../data/Clinicopathological_data.txt',  sep="\t")
data1_matrix = data1.as_matrix()

# Read data Liver_Cancer_RNA_seq_data.txt
data2 = read_csv("../../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
data2 = np.array(data2)

```

```
# Read reliefF idx results
results_idx_grades = genfromtxt('GRADES_relief_idx.csv', delimiter=',')
results_idx_grades = results_idx_grades.astype('int')

### PREPARE DATA FOR SVM ##################################
# targets Grades (0-1) + data
map_dict = {'Early_Grade':0, 'Late_Grade':1}
targets_grades = np.array([map_dict[key] for key in
data1['Grades'].values])

data_all_grades = np.array([])
targets_grades_f = targets_grades.astype('float64')

data_all_grades = np.column_stack((targets_grades_f,data2))
data_rows = data_all_grades.shape[0] # number of rows
data_cols = data_all_grades.shape[1] # numer of cols
### END - PREPARE DATA FOR SVM ############################

### SVM LINEAR GRADES #####################################
# array with all final results - for save in csv
final_grades_svmlinear = np.zeros(shape=(500,4))
csv_row = -1
#array with best final results - for save in csv
final_grades_svmlinear_best = np.zeros(shape=(50,4))
csv_row_best = -1

# NUMBER OF GENES
idx_col = 0
for number_of_genes in range(11,511,10):

    print 'GRADES ------------------- NUMBER OF GENES =
{0}'.format(number_of_genes-1)

    # GRADES + DATA FOR SVM with the best 'number_of_genes-1' genes
    idx_for_svm =  np.zeros(shape=(number_of_genes-1,1))

    # get column from results_idx_grades
    # (col=0 10genes, col=1 20genes, ..., col=49 500genes)
    for irow in range(0, number_of_genes-1, 1):
        idx_for_svm[irow,0] = results_idx_grades[irow,idx_col]

    # next column for the next loop
    idx_col = idx_col + 1

    # create array 'data_for_svm_linear' = targets + data
    data_for_svm_linear = np.empty([data_rows, number_of_genes]);
    data_for_svm_linear = data_for_svm_linear.astype('float64')

    for icol_svm in range(0, number_of_genes, 1):
```

```
67          # column 0 = targets
68          if icol_svm==0:
69              data_for_svm_linear[:,icol_svm] =
    data_all_grades[:,icol_svm]
70          # from column 1 to column 'number_of_genes-1' = data
71          if icol_svm>0:
72              idx = idx_for_svm[icol_svm-1,0]
73              idx=idx.astype('int')
74              data_for_svm_linear[:,icol_svm] = data2[:,idx]
75
76      shuffle_data_targets = np.random.permutation(data_for_svm_linear)
77      grades_data = np.delete(shuffle_data_targets, (0), axis=1)
78      grades_targets = shuffle_data_targets[:,0]
79
80      parameter_candidates = [
81              {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
    'kernel': ['linear']}]
82
83      clf = GridSearchCV(estimator=svm.SVC(),
    param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
    return_train_score=True)
84      clf.fit(grades_data, grades_targets)
85
86      print(clf.best_params_)
87      bestC = clf.best_params_['C']
88      print " "
89
90      trmeans = clf.cv_results_['mean_train_score']
91      trstds = clf.cv_results_['std_train_score']
92      means = clf.cv_results_['mean_test_score']
93      stds = clf.cv_results_['std_test_score']
94      c_para = clf.cv_results_['param_C']
95
96      # accuracy scores & Best accuracy scores
97      for c_par, trmean, trstd, mean, std, params in zip(c_para, trmeans,
    trstds, means, stds, clf.cv_results_['params']):
98          print("GRADES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
99                % (trmean, trstd * 2, mean, std * 2, params))
100
101         csv_row = csv_row + 1
102
103         final_grades_svmlinear[csv_row,0] = number_of_genes-1
104         final_grades_svmlinear[csv_row,1] = c_par
105         final_grades_svmlinear[csv_row,2] = mean
106         final_grades_svmlinear[csv_row,3] = std * 2
107
108         if (c_par == bestC):
109             csv_row_best = csv_row_best + 1
```

```
110             final_grades_svmlinear_best[csv_row_best,0] =
    number_of_genes-1
111             final_grades_svmlinear_best[csv_row_best,1] = bestC
112             final_grades_svmlinear_best[csv_row_best,2] = mean
113             final_grades_svmlinear_best[csv_row_best,3] = std * 2
114
115 np.savetxt("GRADES_Relief_SVM-Linear_all.csv", final_grades_svmlinear,
    delimiter=",",fmt='%.12f')
116 np.savetxt("GRADES_Relief_SVM-Linear_best.csv",
    final_grades_svmlinear_best, delimiter=",",fmt='%.12f')
117
118 print 'END GRADES!'
```

9.5.3 Stages (ReliefF): SVM RBF

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή SVM RBF για τα πρώτα καλύτερα 500 γονίδια (ReliefF) με βήμα 10
04 STAGES: Κλάσεις Στόχοι -> Early Stage = 0 και Late Stage = 1
05 """
06 from pandas import read_csv
07 from numpy import genfromtxt
08 import numpy as np
09 from sklearn import svm
10 from sklearn.model_selection import GridSearchCV
11
12 # Read data Clinicopathological_data.txt
13 data1 = read_csv('../../data/Clinicopathological_data.txt',  sep="\t")
14 data1_matrix = data1.as_matrix()
15
16 # Read data Liver_Cancer_RNA_seq_data.txt
17 data2 = read_csv("../../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
18 data2 = np.array(data2)
19
20 # Read relief idx results
21 results_idx_stages = genfromtxt('STAGES_relief_idx.csv', delimiter=',')
22 results_idx_stages = results_idx_stages.astype('int')
23
24 ### PREPARE DATA FOR SVM ##################################
25 # targets Stages (0-1) + data
26 map_dict = {'Early_Stage':0, 'Late_Stage':1}
27 targets_stages = np.array([map_dict[key] for key in
    data1['Stages'].values])
28
29 data_all_stages = np.array([])
30 targets_stages_f = targets_stages.astype('float64')
31
32 data_all_stages = np.column_stack((targets_stages_f,data2))
33 data_rows = data_all_stages.shape[0] # number of rows
```

```
34 data_cols = data_all_stages.shape[1] # numer of cols
35 ### END - PREPARE DATA FOR SVM ############################
36
37 ### SVM RBF STAGES ########################################
38 # array with all final results - for save in csv
39 final_stages_svmrbf = np.zeros(shape=(6800,5))
40 csv_row = -1
41 # array with best final results - for save in csv
42 final_stages_svmrbf_best = np.zeros(shape=(50,5))
43 csv_row_best = -1
44
45 # NUMBER OF GENES
46 idx_col = 0
47 for number_of_genes in range(11,511,10):
48
49     print 'STAGES ------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
50
51     # STAGES + DATA FOR SVM with the best 'number_of_genes-1' genes
52     idx_for_svm =  np.zeros(shape=(number_of_genes-1,1))
53
54     # get column from results_idx_stages
55     # (col=0 10genes, col=1 20genes, ..., col=49 500genes)
56     for irow in range(0, number_of_genes-1, 1):
57         idx_for_svm[irow,0] = results_idx_stages[irow,idx_col]
58
59     # next column for the next loop
60     idx_col = idx_col + 1
61
62     # create array 'data_for_svm_rbf' = targets + data
63     data_for_svm_rbf = np.empty([data_rows, number_of_genes]);
64     data_for_svm_rbf = data_for_svm_rbf.astype('float64')
65
66     for icol_svm in range(0, number_of_genes, 1):
67         # column 0 = targets
68         if icol_svm==0:
69             data_for_svm_rbf[:,icol_svm] = data_all_stages[:,icol_svm]
70         # from column 1 to column 'number_of_genes-1' = data
71         if icol_svm>0:
72             idx = idx_for_svm[icol_svm-1,0]
73             idx=idx.astype('int')
74             data_for_svm_rbf[:,icol_svm] = data2[:,idx]
75
76     shuffle_data_targets = np.random.permutation(data_for_svm_rbf)
77     stages_data = np.delete(shuffle_data_targets, (0), axis=1)
78     stages_targets = shuffle_data_targets[:,0]
79
80     parameter_candidates = [
81             {'kernel': ['rbf'],
```

```
82              'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
   1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
83              'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
84              'tol': [1e-10]}]
85
86      clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
87      clf.fit(stages_data, stages_targets)
88
89      print(clf.best_params_)
90      bestC = clf.best_params_['C']
91      bestGamma = clf.best_params_['gamma']
92      print " "
93
94      trmeans = clf.cv_results_['mean_train_score']
95      trstds = clf.cv_results_['std_train_score']
96      means = clf.cv_results_['mean_test_score']
97      stds = clf.cv_results_['std_test_score']
98      c_para = clf.cv_results_['param_C']
99      gamma_para = clf.cv_results_['param_gamma']
100
101     # accuracy scores & Best accuracy scores
102     for c_par, g_par, trmean, trstd, mean, std, params in zip(c_para,
   gamma_para, trmeans, trstds, means, stds, clf.cv_results_['params']):
103         print("STAGES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
   %0.03f) for %r"
104               % (trmean, trstd * 2, mean, std * 2, params))
105
106         csv_row = csv_row + 1
107
108         final_stages_svmrbf[csv_row,0] = number_of_genes-1
109         final_stages_svmrbf[csv_row,1] = c_par
110         final_stages_svmrbf[csv_row,2] = g_par
111         final_stages_svmrbf[csv_row,3] = mean
112         final_stages_svmrbf[csv_row,4] = std * 2
113
114         if ((c_par == bestC) and (g_par == bestGamma)):
115             csv_row_best = csv_row_best + 1
116             final_stages_svmrbf_best[csv_row_best,0] = number_of_genes-1
117             final_stages_svmrbf_best[csv_row_best,1] = bestC
118             final_stages_svmrbf_best[csv_row_best,2] = bestGamma
119             final_stages_svmrbf_best[csv_row_best,3] = mean
120             final_stages_svmrbf_best[csv_row_best,4] = std * 2
121
122 np.savetxt("STAGES_Relief_SVM-RBF_all.csv", final_stages_svmrbf,
   delimiter=",",fmt='%.12f')
123 np.savetxt("STAGES_Relief_SVM-RBF_best.csv", final_stages_svmrbf_best,
   delimiter=",",fmt='%.12f')
```

```
124
125 print 'STAGES END!'
```

9.6 mRMR & SVM Linear / SVM RBF

9.6.1 Grades & Stages: Προεπεξεργασία Δεδομένων για εφαρμογή του mRMR

```
01 # -*- coding: utf-8 -*-
02 """
03 Προεπεξεργασία δεδομένων για GRADES & STAGES για εφαρμογή του mRMR
04 df_grades = pandas.DataFrame(final_grades)
05 df_stages = pandas.DataFrame(final_stages)
06 """
07 from pandas import read_csv
08 import pandas
09 import numpy as np
10
11 # Read data Clinicopathological_data.txt
12 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
13 data1_matrix = data1.as_matrix()
14
15 # Read data Liver_Cancer_RNA_seq_data.txt
16 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
17 data2 = np.array(data2)
18
19 # targets Grades (0-1)
20 map_dict = {'Early_Grade':0, 'Late_Grade':1}
21 targets_grades = np.array([map_dict[key] for key in
   data1['Grades'].values])
22 # targets Stages (0-1)
23 map_dict = {'Early_Stage':0, 'Late_Stage':1}
24 targets_stages = np.array([map_dict[key] for key in
   data1['Stages'].values])
25
26 ### Discretization ########################################
27 frequencies_all, bin_edges_all = np.histogram(data2, bins=20, range=(3,
   24))
28
29 # replace gene values with the number of the bin group
30 for icol in range(data2.shape[1]):
31     data2[:,icol] = np.digitize(data2[:,icol],bin_edges_all)
32
33 data_int = np.array([])
34 data_int = data2.astype('int64')
35 ### End Discretization ####################################
36
37 final_grades = np.column_stack((targets_grades,data_int))
38 final_stages = np.column_stack((targets_stages,data_int))
```

```
39
40 df_grades = pandas.DataFrame(final_grades)
41 df_stages = pandas.DataFrame(final_stages)
42
43 ###############################################
44 ### INPUT FOR mRMR df_grades & df_stages ###
45 ###############################################
```

9.6.2 Grades: Εφαρμογή mRMR

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή mRMR MIQ (GRADES)
04 https://github.com/fbrundu/pymrmr
05 """
06 import pandas as pd
07 import numpy as np
08 import pymrmr
09
10 df_grades = pd.read_csv('grades_mrmr.csv')
11 li = pymrmr.mRMR(df_grades, 'MIQ', 100)
12
13 grades_mrmr_miq = np.asarray(li)
14 grades_mrmr_miq = grades_mrmr_miq.astype('int64')
15
16 np.savetxt("final_grades_mrmr_MIQ.csv", grades_mrmr_miq,
   delimiter=",",fmt='%.0f')
```

9.6.3 Stages: Εφαρμογή mRMR

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή mRMR MIQ (STAGES)
04 https://github.com/fbrundu/pymrmr
05 """
06 import pandas as pd
07 import numpy as np
08 import pymrmr
09
10 df_stages = pd.read_csv('stages_mrmr.csv')
11 li = pymrmr.mRMR(df_stages, 'MIQ', 100)
12
13 stages_mrmr_miq = np.asarray(li)
14 stages_mrmr_miq = stages_mrmr_miq.astype('int64')
15
16 np.savetxt("final_stages_mrmr_MIQ.csv", stages_mrmr_miq,
   delimiter=",",fmt='%.0f')
```

9.6.4 Grades (mRMR): SVM Linear

```
01 # -*- coding: utf-8 -*-
02 """
03 Εφαρμογή SVM Linear για τα πρώτα καλύτερα 100 γονίδια (mRMR MIQ) με βήμα
   5
04 GRADES: Κλάσεις Στόχοι -> Early Grade = 0 και Late Grade = 1
05 """
06 from pandas import read_csv
07 import numpy as np
08 from sklearn import svm
09 from sklearn.model_selection import GridSearchCV
10
11 # Read data Clinicopathological_data.txt
12 data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
13 data1_matrix = data1.as_matrix()
14
15 # Read data Liver_Cancer_RNA_seq_data.txt
16 data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
17 data2 = np.array(data2)
18
19 ### PREPARE DATA FOR SVM ##################################
20 # mrmr MIQ results - Best 100 Genes
21 # from df_grades
22 mrmr_MIQ_results = read_csv("final_mrmrMIQ_grades_for_SVM.csv")
23 mrmr_MIQ_results = np.array(mrmr_MIQ_results)
24 mrmr_MIQ_results = np.transpose(mrmr_MIQ_results)
25
26 # data from (genes) mrmr results
27 data_for_svm = np.empty([317,100]);
28 for icol in range(mrmr_MIQ_results.shape[1]):
29     gene = mrmr_MIQ_results[0,icol]
30     data_for_svm[:,icol] = data2[:,gene]
31
32 # targets Grades (0-1)
33 map_dict = {'Early_Grade':0, 'Late_Grade':1}
34 targets_grades = np.array([map_dict[key] for key in
   data1['Grades'].values])
35
36 data_grades = np.column_stack((targets_grades,data_for_svm))
37 data_rows = data_grades.shape[0]  # number of rows
38
39 ### SVM LINEAR GRADES #####################################
40 # array with all final results - for save in csv
41 final_grades_svmlinear = np.zeros(shape=(180,4))
42 csv_row = -1
43 # array with best final results - for save in csv
44 final_grades_svmlinear_best = np.zeros(shape=(20,4))
45 csv_row_best = -1
```

```
46
47 # NUMBER OF GENES
48 for number_of_genes in range(6,106,5):
49
50     print 'GRADES -------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
51     data_for_svm_linear = np.empty([data_rows, number_of_genes]);
52
53     # array with targets + first 'number_of_genes' genes
54     # create data_for_svm_linear
55     for icol in range(number_of_genes):
56         data_for_svm_linear[:,icol] = data_grades[:,icol]
57
58     shuffle_data_targets = np.random.permutation(data_for_svm_linear)
59     grades_data = np.delete(shuffle_data_targets, (0), axis=1)
60     grades_targets = shuffle_data_targets[:,0]
61
62     parameter_candidates = [
63             {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
   'kernel': ['linear']}]
64
65     clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
66     clf.fit(grades_data, grades_targets)
67
68     print(clf.best_params_)
69     bestC = clf.best_params_['C']
70     print " "
71
72     trmeans = clf.cv_results_['mean_train_score']
73     trstds = clf.cv_results_['std_train_score']
74     means = clf.cv_results_['mean_test_score']
75     stds = clf.cv_results_['std_test_score']
76     c_para = clf.cv_results_['param_C']
77
78     # accuracy scores & Best accuracy scores
79     for c_par, trmean, trstd, mean, std, params in zip(c_para, trmeans,
   trstds, means, stds, clf.cv_results_['params']):
80         print("GRADES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
   %0.03f) for %r"
81               % (trmean, trstd * 2, mean, std * 2, params))
82
83         csv_row = csv_row + 1
84
85         final_grades_svmlinear[csv_row,0] = number_of_genes-1
86         final_grades_svmlinear[csv_row,1] = c_par
87         final_grades_svmlinear[csv_row,2] = mean
88         final_grades_svmlinear[csv_row,3] = std * 2
```

```
89
90          if (c_par == bestC):
91              csv_row_best = csv_row_best + 1
92              final_grades_svmlinear_best[csv_row_best,0] =
    number_of_genes-1
93              final_grades_svmlinear_best[csv_row_best,1] = bestC
94              final_grades_svmlinear_best[csv_row_best,2] = mean
95              final_grades_svmlinear_best[csv_row_best,3] = std * 2
96
97  np.savetxt("GRADES_mrmrMIQ_SVM-Linear_all.csv", final_grades_svmlinear,
    delimiter=",",fmt='%.5f')
98  np.savetxt("GRADES_mrmrMIQ_SVM-Linear_best.csv",
    final_grades_svmlinear_best, delimiter=",",fmt='%.5f')
99
100 print 'END GRADES!'
```

9.6.5 Stages (mRMR): SVM RBF

```
01  # -*- coding: utf-8 -*-
02  """
03  Εφαρμογή SVM RBF για τα πρώτα καλύτερα 100 γονίδια (mRMR MIQ) με βήμα 5
04  STAGES: Κλάσεις Στόχοι -> Early Stage = 0 και Late Stage = 1
05  """
06  from pandas import read_csv
07  import numpy as np
08  from sklearn import svm
09  from sklearn.model_selection import GridSearchCV
10
11  # Read data Clinicopathological_data.txt
12  data1 = read_csv('../data/Clinicopathological_data.txt',  sep="\t")
13  data1_matrix = data1.as_matrix()
14
15  # Read data Liver_Cancer_RNA_seq_data.txt
16  data2 = read_csv("../data/Liver_Cancer_RNA_seq_data.txt", sep="\t")
17  data2 = np.array(data2)
18
19  ### PREPARE DATA FOR SVM ##################################
20  # mrmr MIQ results - Best 100 Genes
21  # from df_stages
22  mrmr_MIQ_results = read_csv("final_mrmrMIQ_stages_for_SVM.csv")
23  mrmr_MIQ_results = np.array(mrmr_MIQ_results)
24  mrmr_MIQ_results = np.transpose(mrmr_MIQ_results)
25
26  # data from (genes) mrmr results
27  data_for_svm = np.empty([317,100]);
28  for icol in range(mrmr_MIQ_results.shape[1]):
29      gene = mrmr_MIQ_results[0,icol]
30      data_for_svm[:,icol] = data2[:,gene]
31
```

```
32 # targets Stages (0-1)
33 map_dict = {'Early_Stage':0, 'Late_Stage':1}
34 targets_stages = np.array([map_dict[key] for key in
   data1['Stages'].values])
35
36 data_stages = np.column_stack((targets_stages,data_for_svm))
37 data_rows = data_stages.shape[0] # number of rows
38
39 ### SVM RBF STAGES ########################################
40 # array with all final results - for save in csv
41 final_stages_svmrbf = np.zeros(shape=(2700,5))
42 csv_row = -1
43 # array with best final results - for save in csv
44 final_stages_svmrbf_best = np.zeros(shape=(20,5))
45 csv_row_best = -1
46
47 # NUMBER OF GENES
48 for number_of_genes in range(6,106,5):
49
50     print 'STAGES-------------------- NUMBER OF GENES =
   {0}'.format(number_of_genes-1)
51     data_for_svm_rbf = np.empty([data_rows, number_of_genes]);
52
53     # array with targets + first 'number_of_genes' genes
54     # create data_for_svm_rbf
55     for icol in range(number_of_genes):
56         data_for_svm_rbf[:,icol] = data_stages[:,icol]
57
58     shuffle_data_targets = np.random.permutation(data_for_svm_rbf)
59     stages_data = np.delete(shuffle_data_targets, (0), axis=1)
60     stages_targets = shuffle_data_targets[:,0]
61
62     parameter_candidates = [
63             {'kernel': ['rbf'],
64              'gamma': [1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3,
   1e-2, 1e-1, 1e+0, 1e+1,  1e+2,  1e+3,  1e+4],
65              'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
66              'tol': [1e-10]}]
67
68     clf = GridSearchCV(estimator=svm.SVC(),
   param_grid=parameter_candidates, cv=20, n_jobs=-1, refit=False,
   return_train_score=True)
69     clf.fit(stages_data, stages_targets)
70
71     print(clf.best_params_)
72     bestC = clf.best_params_['C']
73     bestGamma = clf.best_params_['gamma']
74     print " "
75
```

```
76      trmeans = clf.cv_results_['mean_train_score']
77      trstds = clf.cv_results_['std_train_score']
78      means = clf.cv_results_['mean_test_score']
79      stds = clf.cv_results_['std_test_score']
80      c_para = clf.cv_results_['param_C']
81      gamma_para = clf.cv_results_['param_gamma']
82
83      # accuracy scores & Best accuracy scores
84      for c_par, g_par, trmean, trstd, mean, std, params in zip(c_para,
    gamma_para, trmeans, trstds, means, stds, clf.cv_results_['params']):
85          print("STAGES | Train: %0.3f (+/-%0.03f) | Test: %0.3f (+/-
    %0.03f) for %r"
86                % (trmean, trstd * 2, mean, std * 2, params))
87
88          csv_row = csv_row + 1
89
90          final_stages_svmrbf[csv_row,0] = number_of_genes-1
91          final_stages_svmrbf[csv_row,1] = c_par
92          final_stages_svmrbf[csv_row,2] = g_par
93          final_stages_svmrbf[csv_row,3] = mean
94          final_stages_svmrbf[csv_row,4] = std * 2
95
96          if ((c_par == bestC) and (g_par == bestGamma)):
97              csv_row_best = csv_row_best + 1
98              final_stages_svmrbf_best[csv_row_best,0] = number_of_genes-1
99              final_stages_svmrbf_best[csv_row_best,1] = bestC
100             final_stages_svmrbf_best[csv_row_best,2] = bestGamma
101             final_stages_svmrbf_best[csv_row_best,3] = mean
102             final_stages_svmrbf_best[csv_row_best,4] = std * 2
103
104 np.savetxt("STAGES_mrmrMIQ_SVM-RBF_all.csv", final_stages_svmrbf,
    delimiter=",",fmt='%.12f')
105 np.savetxt("STAGES_mrmrMIQ_SVM-RBF_best.csv", final_stages_svmrbf_best,
    delimiter=",",fmt='%.12f')
106
107 print 'STAGES END!'
```