

Θεωρία Αποφάσεων

Εργαστηριακή Άσκηση

Θέμα: "Σχεδιασμός Ταξινομητή για Κατηγοριοποίηση και Πρόβλεψη Καρκίνου του Μαστού"

Εισαγωγή

Στην άσκηση αυτή χρησιμοποιώντας βιοϊατρικές μετρήσεις από 106 ασθενείς, καθώς και τη πραγματική ασθένεια που τελικά αυτοί είχαν, θα δημιουργήσουμε ταξινομητές οι οποίοι θα εκπαιδεύονται με την υπάρχουσα γνώση και θα μπορούν να κάνουν μία πρόβλεψη για έναν νέο ασθενή που δεν έχουν δει ξανά.

Για την υλοποίηση χρησιμοποιήθηκε η γλώσσα προγραμματισμού python σε συνδυασμό με τη βιβλιοθήκη numpy.

Ερώτημα 1

Για τη προεπεξεργασία δεδομένων αρχικά θέλουμε να μεταφέρουμε τα δεδομένα που βρίσκονται στο xls αρχείο στο πρόγραμμά μας. Αυτό υλοποιήθηκε με τη χρήση της βιβλιοθήκης openpyxl στο script: *get_data.py*, αφού πρώτα έγινε μετατροπή του xls αρχείου σε xlsx μιας και η βιβλιοθήκη αυτή υποστηρίζει μόνο αυτή τη μορφή. Η συνάρτηση *get_data_from_excel()* διαβάζονται όλες οι στήλες των δεδομένων, εκτός από τη στήλη όπου φαίνεται το id (case #) του κάθε ασθενή, το οποίο δε μας χρειάζεται κάπου.

Στη συνάρτηση *get_normalized_data()* του ίδιου script παίρνουμε τη κανονικοποιημένη τιμή στο διάστημα [-1,1] των μετρήσεων και αναπαριστούμε τις κλάσεις με αριθμούς. Για τη κανονικοποιημένη τιμή δημιουργείται ένα dictionary για κάθε βιοϊατρική μέτρηση στο οποίο υπολογίζεται (με τη χρήση της numpy) η μέγιστη, η ελάχιστη και η μέση τιμή. Έχοντας αυτές τις τιμές, υπολογίζουμε τη κανονικοποιημένη τιμή με τη χρήση του τύπου:

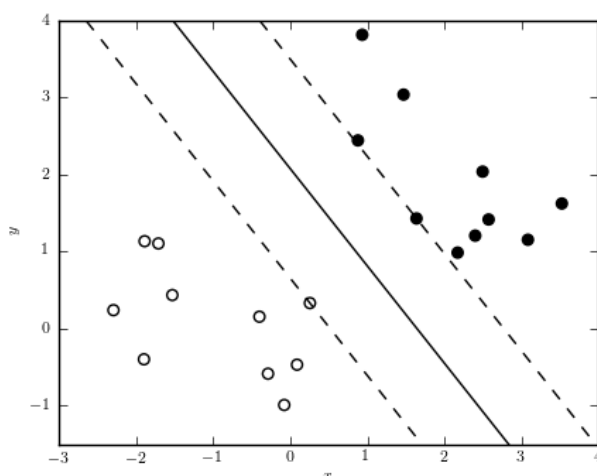
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Η συνάρτηση επιστρέφει μία λίστα λιστών, όπου η πρώτη υπολίστα είναι τα ονόματα των κλάσεων (συμβολιζόμενα πλέον με αριθμούς από το 1 έως το 6), και οι υπόλοιπες είναι οι υπόλοιπες μετρήσεις σε κανονικοποιημένη μορφή. Δηλαδή:

classes,I0,PA500,HFS,DA,Area,ADA,MaxIP,DR,P

Ερώτημα 2 (SVM)

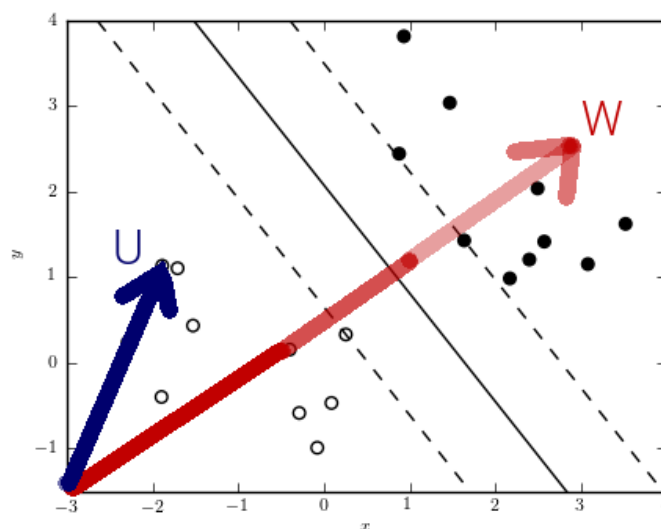
Ο ταξινομητής Support Vector Machine (SVM) είναι ένας ταξινομητής εποπτευόμενης μάθησης. Δέχεται σαν είσοδο δεδομένα με ετικέτες (labels) και δημιουργεί ένα υπερεπίπεδο έτσι ώστε να τα ταξινομήσει σε δύο κατηγορίες. Ο αλγόριθμος προσπαθεί να βρει το υπερεπίπεδο το οποίο απέχει τη μέγιστη δυνατή απόσταση από τα προφανή όρια της κάθε κλάσης τα οποία ονομάζονται support vectors.



Εικόνα 1: Κατηγοριοποίηση SVM

Έχοντας υπολογίσει τη υπερεπίπεδο μπορούμε να βρούμε μία φόρμα για τον υπολογισμό του ορίου απόφασης (decision boundary) χρησιμοποιώντας διανύσματα.

Συγκεκριμένα ορίζουμε το διάνυσμα \vec{w} το οποίο είναι ένα διάνυσμα κάθετο στο όριο, όπως φαίνεται στην εικόνα 2. Το μέγεθος του διανύσματος αυτού δεν είναι ακόμα γνωστό. Έχουμε ένα δείγμα για δοκιμή (test data) μπορούμε να φέρουμε το διάνυσμα \vec{u} προς αυτό.



Εικόνα 2: Το διάνυσμα u σε κάποια τιμή και το διάνυσμα w .

Για να βρούμε σε ποια κλάση ανήκει το u (ονομάζουμε τη δεξιά κλήση θετική και την αριστερή αρνητική), αρκεί να πάρουμε τη προβολή του \vec{u} προς το \vec{w} και να δούμε που βρίσκεται σε σχέση με το όριο. Στην ουσία θέλουμε:

$$\vec{w} * \vec{u} \geq c$$

ή ισοδύναμα:

$$\vec{w} * \vec{u} + b \geq 0 \quad (1)$$

Εάν ισχύει αυτό βρισκόμαστε στη θετική κλάση. Εάν δεν ισχύει βρισκόμαστε στην αρνητική.

Η τιμή αυτή κυμαίνεται ανάμεσα στο 0 και στο 1 όταν βρισκόμαστε μέσα στη περιοχή που σχηματίζει το decision boundary και το support vector των θετικών τιμών και ανέμεσα στο -1 και το 0 για αυτά που βρίσκονται ανάμεσα στο support vector των αρνητικών και στο decision boundary. Αν έχουμε ένα δείγμα που ξέρουμε από πριν ότι είναι στα θετικά για παράδειγμα (έχει γίνει η ταξινόμηση με αυτό το δείγμα μέσα στο training data), τότε γνωρίζουμε ότι:

$$\vec{w} * \vec{x}_{\text{θετικό}} \geq 1 \quad \text{και} \quad \vec{w} * \vec{x}_{\text{αρνητικό}} \leq -1 \quad (2)$$

Αν έχουν τις οριακές τιμές (-1,1) σημαίνει ότι βρίσκονται πάνω στα support vector.

Για διευκόλυνση θέτουμε τη μεταβλητή y_i η οποία έχει τη τιμή +1 εάν η κλάση είναι θετική και τη τιμή -1 αν η κλάση είναι αρνητική.

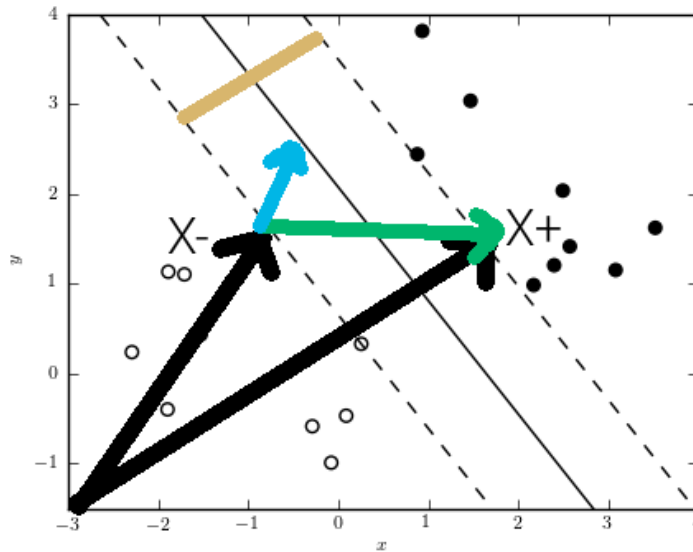
Αν πολλαπλασιάσουμε τα δύο μέλη των πάνω ανισώσεων με το y_i και το αντικαταστήσουμε στο δεξί μέλος με τη τιμή +1 ή -1 ανάλογα με τη κλάση που αναφέρεται (+1 στη πρώτη ανίσωση, -1 στη δεύτερη) προκύπτει και για τις δύο ότι:

$$y_i (\vec{x}_i \cdot \vec{w}) - 1 \geq 0 \quad (3)$$

όπου η ισότητα ισχύει όταν βρισκόμαστε πάνω στα support vectors.

Εάν τώρα σχεδιάσουμε το X_+ που είναι το διάνυσμα προς ένα θετικό στοιχείο πάνω στο support vector και στο αντίστοιχο αρνητικό (X_-) με τη βοήθεια του διανύσματος της διαφοράς τους και ενός μοναδιαίου διανύσματος μπορούμε να υπολογίσουμε την απόσταση ανάμεσα στα δύο support vectors. Για μοναδιαίο διάνυσμα μπορούμε να χρησιμοποιήσουμε το \vec{w} του οποίου ακόμα δε γνωρίσουμε το μήκος. Το κανονικοποιημένο w ισούται με:

$$\vec{w}_{normalized} = \frac{\vec{w}}{\|\vec{w}\|}$$



Εικόνα 3: Υπολογισμός της απόστασης ανάμεσα στα SVs

Έτσι, προκύπτει ότι η απόσταση αυτή είναι:

$$width = (x_{plus} - x_{minus}) \frac{\vec{w}}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|} \quad (4)$$

Σκοπός μας είναι να μεγιστοποιήσουμε το width δηλαδή να ελαχιστοποιήσουμε το μέτρο του $\|\vec{w}\|$ και να μεγιστοποιήσουμε το b. Το πρόβλημα έτσι μετατρέπεται με πρόβλημα βελτιστοποίησης (optimization problem).

Το πρόβλημα αποδεικνύεται ότι είναι convex, δηλαδή ότι δεν υπάρχουν τοπικά ελάχιστα στα οποία ο optimizer θα μπορούσε να κολλήσει. Συνεπώς, αυτό που πρέπει να κάνουμε είναι ξεκινώντας από κάποια τιμή του $\|\vec{w}\|$ να προσπαθούμε να μειώνουμε τη τιμή του αρχικά με μεγάλα βήματα και έπειτα με μικρότερα μέχρι να βρούμε την ελάχιστη τιμή για την οποία φυσικά ικανοποιείται η συνθήκη (3) για όλα τα δεδομένα \vec{x} . Επίσης προσπαθούμε μεγιστοποιήσουμε τη τιμή του b παίρνοντας όμως μεγάλα βήματα μια και δεν είναι τόσο σημαντική η ακρίβεια του b σε σχέση με το w. Και μιας και προσπαθούμε να ελαχιστοποιήσουμε το $\|\vec{w}\|$ πρέπει να λάβουμε υπ' όψιν τα διανύσματα που έχουν την ίδια νόρμα, αλλά αντίθετες κατευθύνσεις.

Αφού λοιπόν δοκιμάσουμε πολλά \vec{w} (στα οποία το $\|\vec{w}\|$ μειώνεται) μαζί με όλα τα \vec{w} που έχουν το ίδιο μέτρο και πολλά b που αυξάνονται κρατάμε αυτά που ικανοποιούν τη συνθήκη (3), δηλαδή αυτά που χωρίζουν τα training data σε 2 κλάσεις. Από αυτά τώρα, επιλέγουμε αυτό που έχει τη μικρότερη νόρμα και το αντίστοιχο b του, το οποίο έχουμε φροντίσει να αποθηκεύσει σε ένα dictionary. Αφού η διαδικασία επαναληφθεί και με μικρότερα βήματα στο τέλος κρατάμε το ελάχιστο \vec{w} και το μέγιστο b που ικανοποιούν τη συνθήκη και συνεπώς χωρίζουν τα δεδομένα μας με τον καλύτερο δυνατό τρόπο, δηλαδή το \vec{w} και το b που μεγιστοποιούν την απόσταση ανάμεσα στα 2 support vectors. Όλα αυτά λειτουργούν για γραμμικά προβλήματα. Εάν το πρόβλημα είναι μη γραμμικό, τότε χρησιμοποιούνται οι kernels. Οι kernels μεταφέρουν το πρόβλημα σε μία μεγαλύτερη διάσταση, στην οποία ίσως και να μπορεί να λυθεί. Υπάρχουν διάφοροι kernels, όπως ο RBF και ο linear που χρησιμοποιούμε εδώ. Κάθε συνδυασμός σημείων περνάει από τη συνάρτηση kernel έτσι ώστε να δημιουργηθούν πλέον νέα σημεία τα οποία ανήκουν σε μεγαλύτερη διάσταση. Τότε η ταξινόμηση γίνεται με αυτά.

Επειδή ο SVM μπορεί να χωρίσει τα δεδομένα σε μόνο 2 κλάσεις, χρησιμοποιούνται διάφορες τεχνικές για να τον κάνουν να δουλεύει και με πολλές. Ένας τρόπος είναι ο αλγόριθμος One versus All, στον οποίο αρχικά μία κλάση θεωρείται σαν κλάση 1 και όλες οι άλλες σαν κλάση -1. Υπολογίζεται το υπερεπίπεδο για αυτές τις δύο και στη συνέχεια η διαδικασία επαναλαμβάνεται για κάθε άλλη κλάση.

Πηγές SVM:

https://www.youtube.com/watch?v=_PwhiWxHK8o

https://en.wikipedia.org/wiki/Support_vector_machine

(NaiveBayes)

Ο Naive Bayes είναι ένας ταξινομητής ο οποίος βασίζεται στο θεώρημα του Bayes. Είναι μια τεχνική που εκτιμά την πιθανοφάνεια μιας ιδιότητας παίρνοντας το σύνολο των δεδομένων σαν απόδειξη ή σαν είσοδο. Δοθέντος ενός συνόλου εκπαίδευσης, ο αλγόριθμος εκτιμά την εκ των προτέρων πιθανότητα για κάθε κατηγορία μετρώντας το πόσο συχνά κάθε κατηγορία εμφανίζεται στα δεδομένα εκπαίδευσης. Επίσης υπολογίζεται η πιθανότητα για κάθε γνώρισμα.

Έχοντας αυτά τα στοιχεία για κάθε τιμή νέα καταχώρηση που θέλουμε να ταξινομήσουμε μπορούμε να υπολογίσουμε την εκ των υστέρων πιθανότητα $P(\text{νέα_πλειάδα}|\text{κλάση})$ και τη πιθανοφάνεια. Έχοντας αυτά μπορούμε να υπολογίσουμε τη πιθανότητα $P(\text{κλάση}|\text{νέα_πλειάδα})$ για κάθε κλάση που έχουμε χρησιμοποιώντας το νόμο του Bayes:

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

Όταν υπολογίσουμε αυτή τη πιθανότητα για κάθε κλάση που έχουμε μπορούμε να πούμε ότι η πλειάδα x κατηγοριοποιείται στη κλάση που έχει τη μεγαλύτερη πιθανότητα.

Πηγές Naive Bayes:

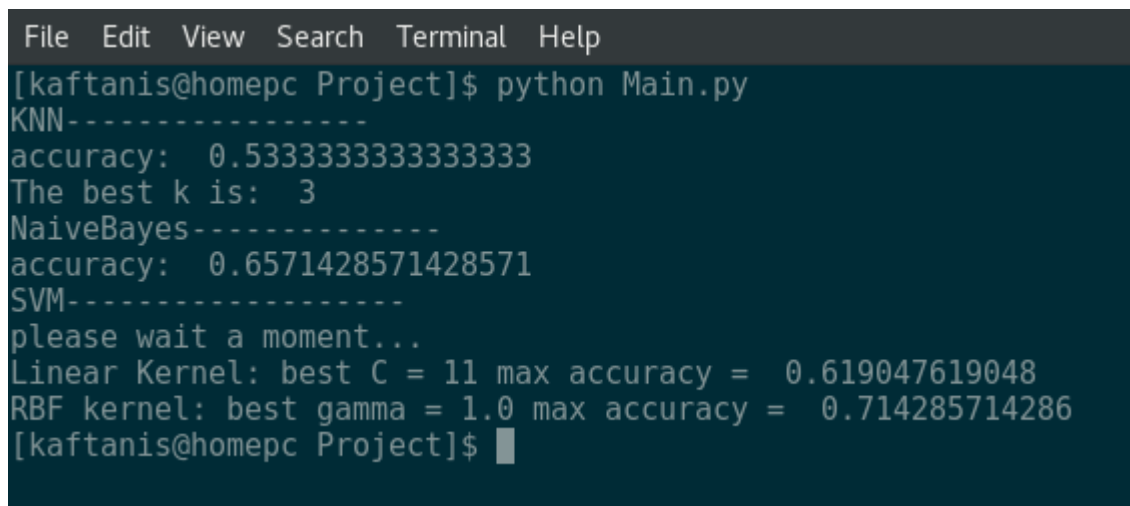
https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Data Mining – Margaret H. Dunham

Ερώτημα 3

Με τη χρήση Python και των βιβλιοθηκών NumPy για τις πράξεις με μητρώα και διανύσματα, της CVXOPT για τη λύση του προβλήματος βελτιστοποίησης στο SVM και της openpyxl για το διάβασμα των δεδομένων από το excel αρχείο, δημιουργήθηκαν οι ταξινομητές Support Vector Machine, KNN και Naive Bayes.

Από το script Main.py καλούνται καλούνται όλα τα υπόλοιπα scripts για να εμφανίσουν το accuracy του κάθε αλγόριθμου. Δύο εκτελέσεις του προγράμματος φαίνονται στα παρακάτω screenshots:



```
File Edit View Search Terminal Help
[kaftanis@homepc Project]$ python Main.py
KNN-----
accuracy:  0.5333333333333333
The best k is:  3
NaiveBayes-----
accuracy:  0.6571428571428571
SVM-----
please wait a moment...
Linear Kernel: best C = 11 max accuracy =  0.619047619048
RBF kernel: best gamma = 1.0 max accuracy =  0.714285714286
[kaftanis@homepc Project]$
```

Εικόνα 4: Εκτέλεση εφαρμογής 1

```
File Edit View Search Terminal Help
[kaftanis@homepc Project]$ python Main.py
KNN-----
accuracy: 0.5238095238095238
The best k is: 3
NaiveBayes-----
accuracy: 0.7047619047619047
SVM-----
please wait a moment...
Linear Kernel: best C = 96 max accuracy = 0.714285714286
RBF kernel: best gamma = 2.5 max accuracy = 0.714285714286
[kaftanis@homepc Project]$
```

Εικόνα 5: Εκτέλεση εφαρμογής 2

Το accuracy αντιπροσωπεύει το ποσοστό των test data τα οποία ταξινομήθηκαν σωστά από τον εκάστοτε ταξινομητή. Στο KNN μπορούμε να δούμε το k που έδωσε τη καλύτερη ακρίβεια μετά από τη δοκιμή πολλών k , ενώ στον SVM φαίνονται οι βέλτιστοι παράμετροι C και γ , οι οποίες βρέθηκαν μετά από αντίστοιχες δοκιμές. Παρατηρούμε ότι οι ακρίβειες που δίνουν οι αλγόριθμοι δεν είναι ιδιαίτερα εντυπωσιακές. Αυτό συμβαίνει κυρίως επειδή το τα δεδομένα που έχουμε στη διάθεσή μας είναι αρκετά λίγα (106). Στον πραγματικό κόσμο, χρησιμοποιούνται χιλιάδες ή δεκάδες χιλιάδες καταχωρήσεις έτσι ώστε να επιτύχουμε μία άκρως ικανοποιητική accuracy.

Στον αλγόριθμο KNN το k δηλώνει το πλήθος των κοντινότερων σημείων που θα ελέγξει ο αλγόριθμος. Το k που δίνει την καλύτερη accuracy διαφέρει από εκτέλεση σε εκτέλεση, μιας και το ανακάτεμα των δεδομένων γίνεται με τυχαίο τρόπο, αλλά συνήθως είναι μικρό, πράγμα που σημαίνει ότι για τα μεγάλα k συμβαίνει υπερεκπαίδευση και το μοντέλο δεν είναι σε θέση να ταξινομήσει καινούρια δεδομένα.

Στον αλγόριθμο SVM, έχουμε τις παραμέτρους C και γ . Η παράμετρος C αναφέρεται στο πόση "αξία" δίνουμε στο slack, δηλαδή στο κατά πόσο τιμωρούμε τις παραβιάσεις. Πιο συγκεκριμένα όσο μεγαλώνουμε το C , τόσο περισσότερη αξία έχει το slack, άρα τόσο περισσότερο τιμωρούμε τις παραβιάσεις. Τιμωρώντας τις παραβιάσεις το υπερεπίπεδο το οποίο χωρίζει τις δύο κλάσεις θα αφήνει όσο το δυνατό λιγότερα στοιχεία της μίας να βρίσκονται στη περιοχή της άλλης. Αυτό, όπως είναι εμφανές μπορεί να προκαλέσει υπερεκπαίδευση αν το C μεγαλώσει αρκετά, μιας και το υπερεπίπεδο θα προσαρμόζεται ακριβώς στα training data. Αυτό ονομάζεται hard margin. Στην αντίθετη περίπτωση (soft

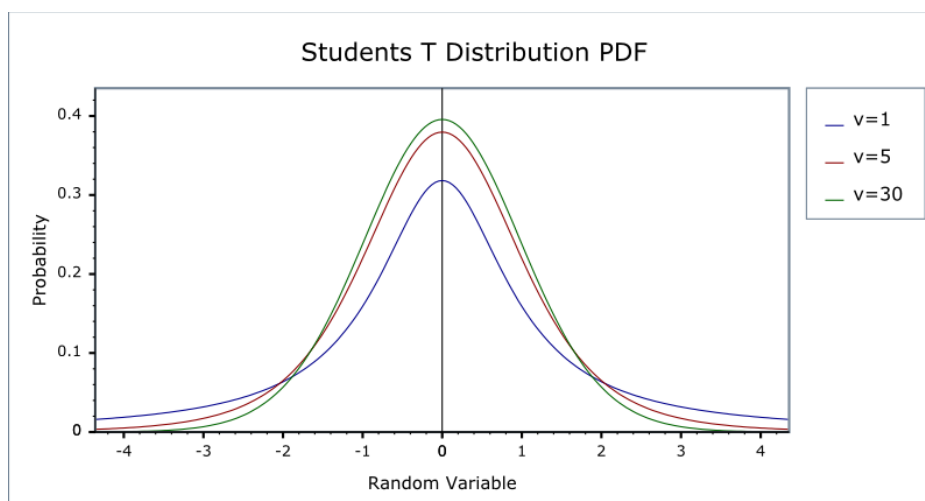
margin) όπου το C είναι μικρό δείχνουμε περισσότερη ανέχεια σε λάθη.

Οι kernel, όπως αναφέρθηκε και νωρίτερα μεταφέρουν το πρόβλημα σε μεγαλύτερη διάσταση έτσι ώστε να βρεθεί εκεί κάποιος γραμμικός διαχωρισμός των δεδομένων. Ο RBF kernel, ο οποίος θεωρητικά μεταφέρει το πρόβλημα σε άπειρες διαστάσεις χρησιμοποιεί τον συντελεστή ελευθερίας γ , για τον οποίο επίσης μετράμε την απόδοση του αλγορίθμου χρησιμοποιώντας το καλύτερο C που βρέθηκε σε κάθε εκτέλεση.

Κάθε μέθοδος δίνει διαφορετικά αποτελέσματα επειδή ο τρόπος που αντιμετωπίζει το πρόβλημα είναι εντελώς διαφορετικός. Η απλή θεώρηση του KNN απέχει πολύ πολύ από την πολύπλοκη διαδικασία του SVM. Η καλύτερη μέθοδος σε απόδοση είναι εμφανώς το SVM με την επιλογή του kernel να μην είναι *a priori*, αλλά να εξαρτάται από τα δεδομένα. Ο SVM διασφαλίζει ότι το υπερπίπεδο το οποίο θα χωρίσει τα δεδομένα θα είναι αρκετά επαρκές μιας και όλος ο αλγόριθμος βασίζεται στη μεγιστοποίηση της απόστασής του από τα δεδομένα. Παρ' όλα αυτά ο αλγόριθμος αυτός είναι αρκετά πιο αργός σε χρόνο από τους άλλους δύο, πράγμα που είναι σημαντικό μειονέκτημα που προέρχεται κυρίως από τη δυσκολία του προβλήματος βελτιστοποίησης αυτού καθ' αυτού. Μία *optimal* λύση προφανώς θα είναι αρκετά καλύτερη σε χρόνο. Συνεπώς το SVM είναι η μέθοδος που προτείνεται.

Ερώτημα 4

Η Student t-test είναι μία στατιστική μέθοδος, η οποία βασιζόμενη στη t-κατανομή μπορεί να εντοπίσει εάν δύο σετ δεδομένων έχουν κάποια πρακτική στατιστική διαφορά ή στην ουσία είναι παρόμοια δείγματα.



Εικόνα 6: Η κατανομή πυκνότητας πιθανότητας της T

Αρχικά ξεκινάμε με τη λεγόμενη null υπόθεση, ότι δεν υπάρχει κάποια διάφορα ανάμεσα στα δύο δείγματα και προσπαθούμε να δούμε εάν θα απορρίψουμε ή θα δεχτούμε αυτή την υπόθεση.

Αρχικά πρέπει να βρούμε τη τιμή t (t value) η οποία προέρχεται από το τύπο:

$$t_{value} = \frac{|\bar{x}_1 - \bar{x}_2|}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

όπου x_1 και x_2 οι μέσες τιμές του δείγματος 1 και 2 αντίστοιχα, s_1^2 και s_2^2 οι αντίστοιχες διασπορές και n_1 και n_2 το πλήθος του κάθε δείγματος.

Στη συνέχεια πρέπει να συγκρίνουμε τη τιμή αυτή με τη critical value της κατανομής t . Η τιμή αυτό προέρχεται από τη pdf της κατανομής t και μπορεί να βρεθεί είτε από τους πίνακες της κατανομής είτε κατευθείαν από την εξίσωσή της. Για να πάρουμε τη critical value χρειαζόμαστε δύο παραμέτρους: τη two-tailed p-value και το βαθμό ελευθερίας. Η two-tailed p-value είναι στην ουσία η πιθανότητα να δεχτούμε ως αληθές τη null υπόθεση (συνήθως ορίζουμε όποια θέλουμε) και οι βαθμοί ελευθερίας είναι το μέγεθος του πρώτου δείγματος συν το μέγεθος του δεύτερου μείον 2.

Στο τέλος εάν η t value είναι μεγαλύτερη της critical value τότε δεν δεχόμαστε ως αληθή την υπόθεση. Διαφορετικά δε δεχόμαστε. Όταν δε δεχόμαστε την αρχική υπόθεση σημαίνει ότι τα δείγματα έχουν κάποια σχέση μεταξύ τους και επομένως ίσως δε θα πρέπει να χρησιμοποιηθούν όλα σαν training δεδομένα στη ταξινόμηση.

Πηγές:

<https://www.youtube.com/watch?v=pTmLQvMM-1M>

https://en.wikipedia.org/wiki/Student's_t-test

Ερώτημα 5

Στο Main.py υλοποιήθηκε η συνάρτηση `t_test(a,b)` η οποία δέχεται δύο διανύσματα και επιστρέφει True εάν υπάρχει στατιστική σχέση ανάμεσά τους ή False εάν δεν υπάρχει. Εκτελώντας το πρόγραμμα, για όλους τους συνδυασμούς των features εμφανίζεται οι σχέσεις μεταξύ τους. Αφού κρατήσουμε τα 4 χαρακτηριστικά που έχουν τις λιγότερες σχέσεις με άλλα, δηλαδή τα 4 σημαντικότερα τρέχουμε πάλι το SVM και παρατηρούμε ότι το accuracy είναι λίγο βελτιωμένο. Έχοντας αφαιρέσει τα πιο ασήμαντα χαρακτηριστικά έχουμε μειώσει τη πολυπλοκότητα κάνοντας την εκτέλεση γρηγορότερη. Τα χαρακτηριστικά που αφαιρέθηκαν ουσιαστικά “προσέφεραν” θόρυβο στο πρόβλημα κάνοντας την ακρίβεια του ταξινομητή μικρότερη. Η διαφορά όμως είναι αρκετά μικρή και αυτό έχει να κάνει πιθανόν στο ότι τα ιατρικά δεδομένα δεν είναι άσχετα με την ασθένεια από την οποία πάσχει ο ασθενής, απλώς σχετίζονται με κάποιες από τις “σημαντικές” μετρήσεις.

```
[kaftanis@homepc Project]$ /usr/bin/python3 Main.py
KNN-----
accuracy: 0.6095238095238096
The best k is: 7
NaiveBayes-----
accuracy: 0.6571428571428571
SVM-----
please wait a moment...
Linear Kernel: best C = 51 max accuracy = 0.666666666667
RBF kernel: best gamma = 1.5 max accuracy = 0.666666666667
υπάρχει σχέση στο: 1 με το 4
υπάρχει σχέση στο: 1 με το 5
υπάρχει σχέση στο: 1 με το 7
υπάρχει σχέση στο: 2 με το 8
υπάρχει σχέση στο: 3 με το 5
υπάρχει σχέση στο: 4 με το 7
υπάρχει σχέση στο: 4 με το 8
υπάρχει σχέση στο: 6 με το 7
υπάρχει σχέση στο: 6 με το 8
υπάρχει σχέση στο: 6 με το 9
υπάρχει σχέση στο: 7 με το 8
υπάρχει σχέση στο: 7 με το 9

SVM μετά το t-test
SVM-----
please wait a moment...
Linear Kernel: best C = 56 max accuracy = 0.619047619048
RBF kernel: best gamma = 5.0 max accuracy = 0.714285714286
[kaftanis@homepc Project]$
```

Εικόνα 7: Εκτέλεση script μαζί με το t-test και την εκτέλεση του SVM με τα σημαντικότερα δεδομένα

Για την εκτέλεση του script αρκεί να τρέξετε το Main.py ενώ έχετε τα υπόλοιπα script και το .xlsx στον ίδιο φάκελο και τις απαραίτητες βιβλιοθήκες εγκατεστημένες.