ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΤΡΕΑΣ ΠΑΛΑΜΑΣ Α.Μ.: 1049789

ΜΙΧΑΛΗΣ ΝΙΚΟΛΑΟΥ Α.Μ : 1049788

Ως ομάδα επιλέξαμε να υλοποιήσουμε το θέμα¨Task force 1 ΄΄το οποίο έχει να κάνει με ένα σοβαρό θέμα που ταλανίζει την υγεία κάθε ανθρώπου, η καρδιοπάθεια. Στο πρόβλημα που επιλέξαμε χρησιμοποιούμε το dataset που υπάρχει στη βάση δεδομένων του Κλίβελαντ που μέχρι σήμερα είναι η μόνη που έχει χρησιμοποιηθεί από ερευνητές ML. Το συγκεκριμένο σύνολο δεδομένων έχει χρησιμοποιηθεί για την πρόβλεψη του 10ετους κινδύνου της Στεφανιαίας Νόσου. Σε αυτό το dataset που μας δόθηκε υπάρχουν κάποιες εξαρτημένες μεταβλητές οι οποίες αποτελούν παράγοντες κινδύνου για καρδιακές παθήσεις. Επιπλέον έχουμε μια ανεξάρτητη μεταβλητή η οποία είναι ο 10ετής κίνδυνος της ΣΝ. Το πρόβλημα μας είναι ένα πρόβλημα δυαδικής ταξινόμησης και η μεταβλητή target μπορεί να πάρει την τιμή 0 που σημαίνει ότι ο ασθενής δεν ανέπτυξε ποτέ καρδιακή νόσο και 1 για ασθενή που εμφάνισε. Οι μεταβλητές που παίζουν ρόλο στο αν κάποιος ασθενής εμφάνισε καρδιακή πάθηση και περιέχονται στο dataset μας είναι οι παρακάτω:

Age: Η ηλικία είναι ο σημαντικότερος παράγοντας κινδύνου για την ανάπτυξη καρδιαγγειακών ή καρδιακών παθήσεων, με περίπου τριπλασιασμό του κινδύνου σε κάθε δεκαετία της ζωής. Οι στεφανιαίες λιπαρές ραβδώσεις μπορεί να αρχίσουν να σχηματίζονται στην εφηβεία. Υπολογίζεται ότι το 82 τοις εκατό των ανθρώπων που πεθαίνουν από στεφανιαία νόσο είναι 65 ετών και άνω. Ταυτόχρονα, ο κίνδυνος εγκεφαλικού διπλασιάζεται κάθε δεκαετία μετά την ηλικία των 55 ετών.

Sex: Οι άνδρες διατρέχουν μεγαλύτερο κίνδυνο καρδιακών παθήσεων από τις προεμμηνοπαυσιακές γυναίκες. Μόλις περάσει η εμμηνόπαυση, έχει υποστηριχθεί ότι ο κίνδυνος μιας γυναίκας είναι παρόμοιος με τον κίνδυνο ενός άνδρα. Στο dataset μας η συγκεκριμένη μεταβλητή παίρνει τιμές 0 και 1. Το 0 αντιπροσωπεύει την γυναίκα και το 1 αντιπροσωπεύει τον άντρα.

Chest Pain (cp): Πόνος στο στήθος ή δυσφορία που προκαλείται όταν ο καρδιακός σας μυς δεν λαμβάνει αρκετό αίμα πλούσιο σε οξυγόνο. Μπορεί να αισθάνεστε σαν πίεση ή συμπίεση στο στήθος σας. Η ενόχληση μπορεί επίσης να εμφανιστεί στους ώμους, τα χέρια, το λαιμό, το σαγόνι ή την πλάτη σας. Ο πόνος στη στηθάγχη μπορεί ακόμη και να μοιάζει με δυσπεψία. Στο dataset μας παίρνει τιμές από 0 μέχρι 3. Το 0 αντιπροσωπεύει τον τυπικό πόνο στο στήθος, το 1 αντιπροσωπεύει τον άτυπο πόνο στο στήθος , το 2 αντιπροσωπεύει τον μη πόνο στο στήθος και το 3 αντιπροσωπεύει τον ασυμπτοματικό.

Resting Blood Pressure (trestbps): Με την πάροδο του χρόνου, η υψηλή αρτηριακή πίεση μπορεί να βλάψει τις αρτηρίες που τροφοδοτούν την καρδιά σας. Η υψηλή αρτηριακή πίεση που εμφανίζεται με άλλες καταστάσεις, όπως η παχυσαρκία, η υψηλή χοληστερόλη ή ο διαβήτης, αυξάνει τον κίνδυνο ακόμη περισσότερο. Στο dataset μας η συγκεκριμένη μεταβλητή παίρνει μοναδικές τιμές.

Serum Cholesterol (chol): Ένα υψηλό επίπεδο χοληστερόλης λιποπρωτεϊνών χαμηλής πυκνότητας (LDL) (η «κακή» χοληστερόλη) είναι πιο πιθανό να στενέψει τις αρτηρίες. Ένα υψηλό επίπεδο τριγλυκεριδίων, ένας τύπος λίπους στο αίμα που σχετίζεται με τη διατροφή σας, αυξάνει επίσης τον κίνδυνο καρδιακής προσβολής. Ωστόσο, ένα υψηλό επίπεδο λιποπρωτεΐνης υψηλής πυκνότητας (HDL) χοληστερόλης (η «καλή» χοληστερόλη) μειώνει τον κίνδυνο καρδιακής προσβολής. Στο dataset μας παίρνει μοναδικές τιμές.

Fasting Blood Sugar (fbs): Η μη παραγωγή αρκετής ορμόνης που εκκρίνεται από το πάγκρεας (ινσουλίνη) ή η μη σωστή ανταπόκριση στην ινσουλίνη προκαλεί αύξηση των επιπέδων

σακχάρου στο αίμα του σώματός σας, αυξάνοντας τον κίνδυνο καρδιακής προσβολής. Στο dataset μας παίρνει τιμές 0 ή 1. Το 1 σημαίνει ότι αν το fasting blood sugar είναι μεγαλύτερο του 120mg/dl και το 0 αν δεν είναι.

Resting ECG (restecg): Για άτομα με χαμηλό κίνδυνο καρδιαγγειακής νόσου, το USPSTF καταλήγει με μέτρια βεβαιότητα ότι οι πιθανές βλάβες του προσυμπτωματικού ελέγχου με ΗΚΓ ανάπαυσης ή άσκησης είναι ίσες ή υπερβαίνουν τα πιθανά οφέλη. Για τα άτομα μεσαίου έως υψηλού κινδύνου, τα τρέχοντα στοιχεία είναι ανεπαρκή για την αξιολόγηση της ισορροπίας των οφελών και των βλαβών του προσυμπτωματικού ελέγχου. Στο dataset μας παίρνει τιμές 0 ή 1 ή 2. Το 0 σημαίνει ότι είναι κανονικό, το 1 σημαίνει ότι έχει ανωμαλία κύματος ST-T και το 2 σημαίνει ότι υπάρχει υπερτροφία της αριστερής κοιλίας.

Max heart rate achieved (thalach): Η αύξηση του καρδιαγγειακού κινδύνου, που σχετίζεται με την επιτάχυνση του καρδιακού ρυθμού, ήταν συγκρίσιμη με την αύξηση του κινδύνου που παρατηρήθηκε με την υψηλή αρτηριακή πίεση. Έχει αποδειχθεί ότι η αύξηση του καρδιακού ρυθμού κατά 10 παλμούς ανά λεπτό συσχετίστηκε με αύξηση του κινδύνου καρδιακού θανάτου κατά τουλάχιστον 20%, και αυτή η αύξηση του κινδύνου είναι παρόμοια με αυτή που παρατηρήθηκε με αύξηση του συστολικού αίματος πίεση κατά 10 mm Hg. Στο dataset μας παίρνει μοναδικές τιμές για κάθε ασθενή.

Exercise induced angina (exang): Ο πόνος ή η ενόχληση που σχετίζεται με τη στηθάγχη συνήθως αισθάνεται σφιχτή, πιάσιμο ή συμπίεση και μπορεί να ποικίλλει από ήπιο έως σοβαρό. Η στηθάγχη γίνεται συνήθως αισθητή στο κέντρο του στήθους σας, αλλά μπορεί να εξαπλωθεί σε έναν ή και στους δύο ώμους σας, ή στην πλάτη, το λαιμό, το σαγόνι ή το χέρι σας. Μπορεί να γίνει αισθητό ακόμα και στα χέρια σας. ο Τύποι στηθάγχης α. Σταθερή στηθάγχη / Στηθάγχη β. Ασταθής στηθάγχη γ. Παραλλαγή (Prinzmetal) Στηθάγχη d. Μικροαγγειακή στηθάγχη. Στο dataset μας αυτή η μεταβλητή παίρνει τιμές 0 ή 1. Το 1 σημαίνει ότι υπάρχει αυτό το πρόβλημα και το 0 ότι δεν υπάρχει.

ST depression induced by exercise relative to rest (oldpeak): Μια δοκιμασία καταπόνησης ΗΚΓ σε διάδρομο θεωρείται μη φυσιολογική όταν υπάρχει οριζόντια ή με κλίση κατάθλιψη του τμήματος ST $\geq$ 1 mm στα 60–80 ms μετά το σημείο J. Τα ΗΚΓ άσκησης με ανοδικές καταθλίψεις του τμήματος ST αναφέρονται συνήθως ως «αμφίβολη» εξέταση. Γενικά, η εμφάνιση οριζόντιας ή κατηφορικής κατάθλιψης του τμήματος ST σε χαμηλότερο φόρτο εργασίας (υπολογισμένο σε MET) ή καρδιακό ρυθμό υποδηλώνει χειρότερη πρόγνωση και μεγαλύτερη πιθανότητα πολυαγγειακής νόσου. Η διάρκεια της κατάθλιψης του τμήματος ST είναι επίσης σημαντική, καθώς η παρατεταμένη ανάκαμψη μετά το μέγιστο στρες συνάδει με ένα θετικό τεστ καταπόνησης ΗΚΓ σε διάδρομο. Ένα άλλο εύρημα που είναι ιδιαίτερα ενδεικτικό σημαντικής ΣΝ είναι η εμφάνιση ανύψωσης του τμήματος ST > 1 mm (συχνά υποδηλώνει διατοιχωματική ισχαιμία). Αυτοί οι ασθενείς συχνά παραπέμπονται επειγόντως για στεφανιογραφία. Στο dataset μας παίρνει float τιμές.

Peak exercise ST segment (slope): Μας δείχνει την μέγιστη άσκηση τμήματος ST που κάνει ο ασθενής. Στο dataset μας παίρνει τιμές 0 ή 1 ή 2. Το 0 μας λέει ότι είναι upslopping το 1 μας λέει ότι flat και το 2 μας λέει downsloping.

Number of major vessels (0–3) colored by flourosopy (ca): Μας δείχνει τον αριθμό μεγάλων αγγείων χρωματισμένα με ακτινοσκόπηση. Στο dataset μας παίρνει τιμές 0 ή 1 ή 2 ή 3. Αυτοί οι ακέραιοι αριθμοί μας δείχνει το πλήθος αυξητικά.

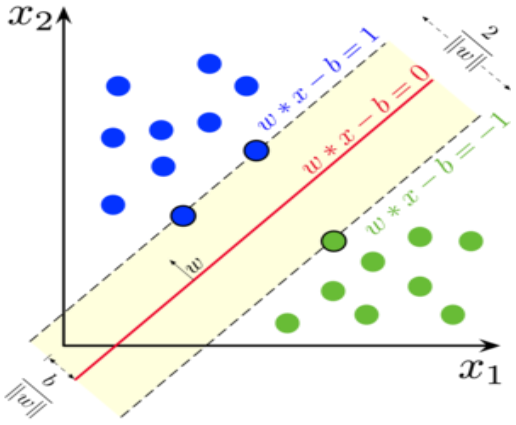
Thal (thal): Μας δείχνει κατά πόσο ο ασθενής έχει θαλασανεμία. Στο dataset μας παίρνει τιμές 1 ή 2 ή 3. Το 1 μας δείχνει αν είναι σε κανονικά επίπεδα, το 1 μας δείχνει αν το ελάττωμα διορθώθηκε και το 3 μας δείχνει αν το ελάττωμα είναι αναστρέψιμο.

Diagnosis of heart disease (target): Μας δείχνει αν ο ασθενής υποφέρει από καρδιακή πάθηση ή όχι. Στο dataset μας παίρνει τιμές 0 ή 1. Το 0 μας δείχνει ότι ο ασθενής δεν υποφέρει από καρδιακές παθήσεις και το 1 μας δείχνει ότι ο ασθενής υποφέρει από καρδιακές παθήσεις.

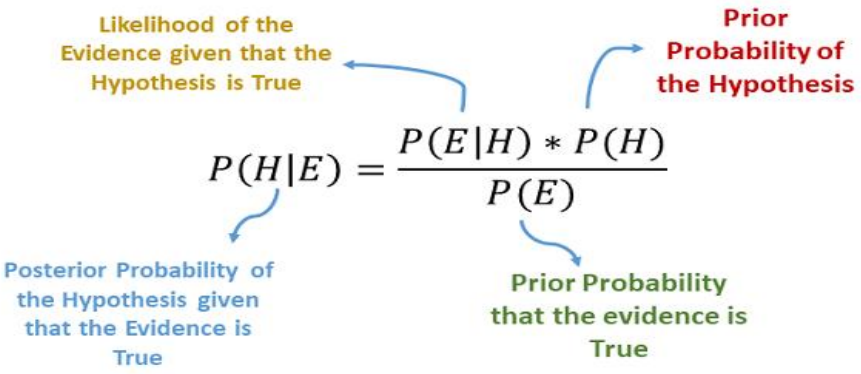
Εξήγηση αλγορίθμου Μηχανικής Μάθησης:

Στη συνέχεια της εργασίας μας , καθώς έχουμε να αντιμετωπίσουμε ένα dataset το οποίο είναι αρκετά ανισορροπημένο για το λόγο ότι πολλοί από τους ασθενής σε αυτό το dataset δεν εμφάνισαν καρδιακή νόσο, το αντιμετωπίσαμε με τις κατάλληλες τεχνικές και αλγορίθμους έτσι ώστε να μην καταλήξουμε σε ένα μοντέλο το οποίο απλώς προβλέπει την πλειοψηφική κατηγορία για κάθε σημείο δεδομένων και να μπορεί να εντοπίσει ασθενείς που ανέπτυξαν καρδιοπάθεια. Όπως φαίνεται και στον κώδικα που ανεβάσαμε στο repository του github μας για κάθε αλγόριθμο δημιουργήσαμε μια συνάρτηση όπου μέσα υλοποιούμε τον κάθε αλγόριθμο και αυτές τις συναρτήσεις τις καλούμε μέσα στη συνάρτηση main. Για το σκοπό του προβλήματος μας χρησιμοποιήσαμε τους πιο κάτω αλγόριθμους ταξινόμησης για Μηχανική Μάθηση:

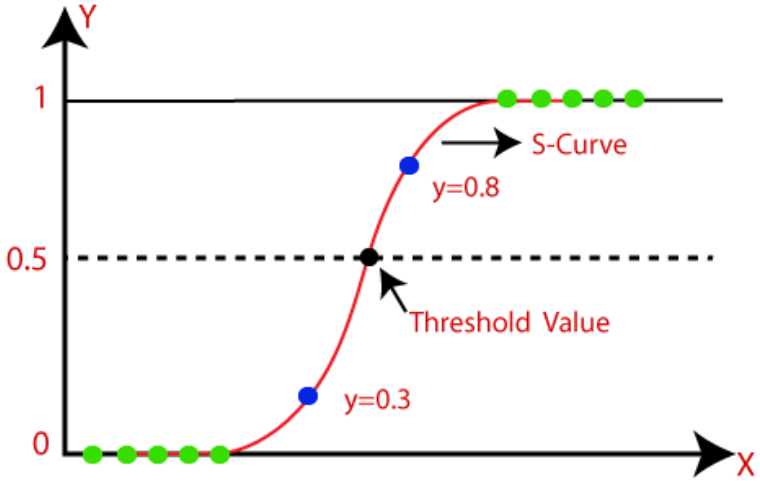
1. **Support Vector Machine**: Ο SVM είναι ένας supervised αλγόριθμος μηχανικής μάθησης που μπορεί να χρησιμοποιηθεί τόσο για προκλήσεις ταξινόμησης όσο και για παλινδρόμηση. Ωστόσο, χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης. Στον αλγόριθμο SVM, σχεδιάζουμε κάθε στοιχείο δεδομένων ως ένα σημείο σε n-διάστατο χώρο (όπου n είναι ένας αριθμός χαρακτηριστικών που έχουμε) με την τιμή κάθε χαρακτηριστικού να είναι η τιμή μιας συγκεκριμένης συντεταγμένης. Στη συνέχεια, πραγματοποιούμε ταξινόμηση βρίσκοντας το υπερεπίπεδο που διαφοροποιεί πολύ καλά τις δύο κατηγορίες. Τα διανύσματα υποστήριξης είναι απλώς οι συντεταγμένες της μεμονωμένης παρατήρησης. Ο ταξινομητής SVM είναι ένα σύνορο που διαχωρίζει καλύτερα τις δύο κατηγορίες.



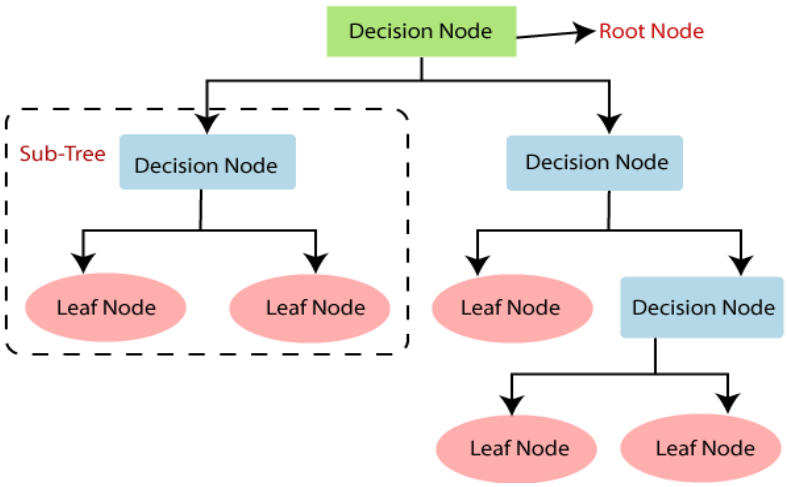
2. **Naïve Bayes:** Είναι μια τεχνική ταξινόμησης που βασίζεται στο θεώρημα του Bayes με μια υπόθεση ανεξαρτησίας μεταξύ των προγνωστικών. Με απλά λόγια, ένας ταξινομητής Naive Bayes υποθέτει ότι η παρουσία ενός συγκεκριμένου χαρακτηριστικού σε μια κλάση δεν σχετίζεται με την παρουσία οποιουδήποτε άλλου χαρακτηριστικού .



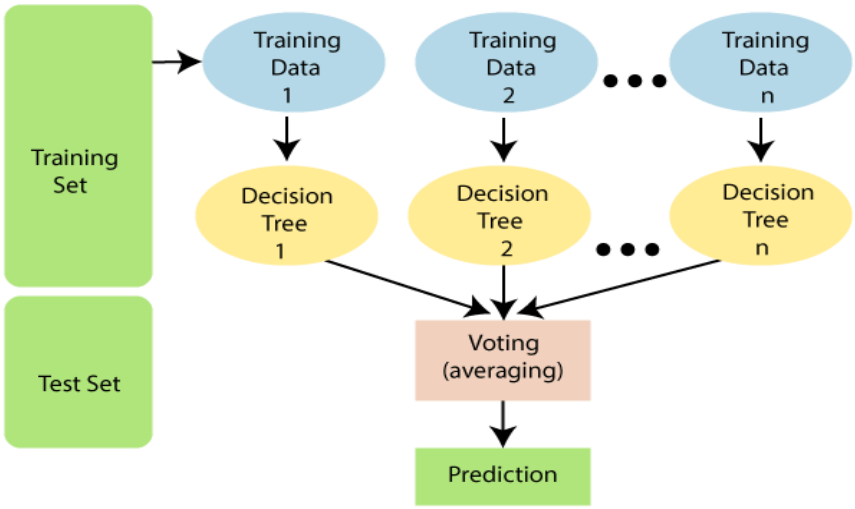
3. **Logistic Regression**: Η λογιστική παλινδρόμηση χρησιμοποιείται γενικά όπου πρέπει να ταξινομήσουμε τα δεδομένα σε δύο ή περισσότερες κλάσεις. Το ένα είναι δυαδικό και το άλλο είναι πολυκλάση λογιστική παλινδρόμηση. Μπορούμε να ονομάσουμε μια λογιστική παλινδρόμηση μοντέλο γραμμικής παλινδρόμησης, αλλά η λογιστική παλινδρόμηση χρησιμοποιεί μια πιο σύνθετη συνάρτηση κόστους, αυτή η συνάρτηση κόστους μπορεί να οριστεί ως «Σιγμοειδής συνάρτηση» ή επίσης γνωστή ως «λογιστική συνάρτηση» αντί για γραμμική συνάρτηση.



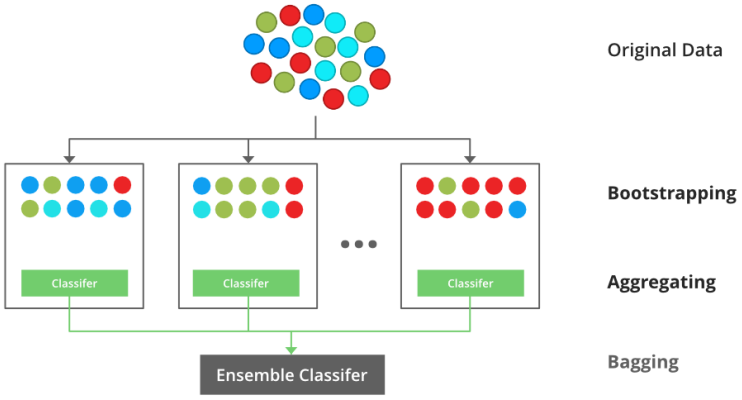
4. **Decision Tree**: Τα δέντρα αποφάσεων χρησιμοποιούν πολλαπλούς αλγόριθμους για να αποφασίσουν να χωρίσουν έναν κόμβο σε δύο ή περισσότερους υπο-κόμβους. Η δημιουργία υπο-κόμβων αυξάνει την ομοιογένεια των υπο-κόμβων που προκύπτουν. Με άλλα λόγια, μπορούμε να πούμε ότι η καθαρότητα του κόμβου αυξάνεται σε σχέση με τη μεταβλητή στόχο. Το δέντρο απόφασης χωρίζει τους κόμβους σε όλες τις διαθέσιμες μεταβλητές και στη συνέχεια επιλέγει τη διαίρεση που οδηγεί στους περισσότερους ομοιογενείς υπο-κόμβους.



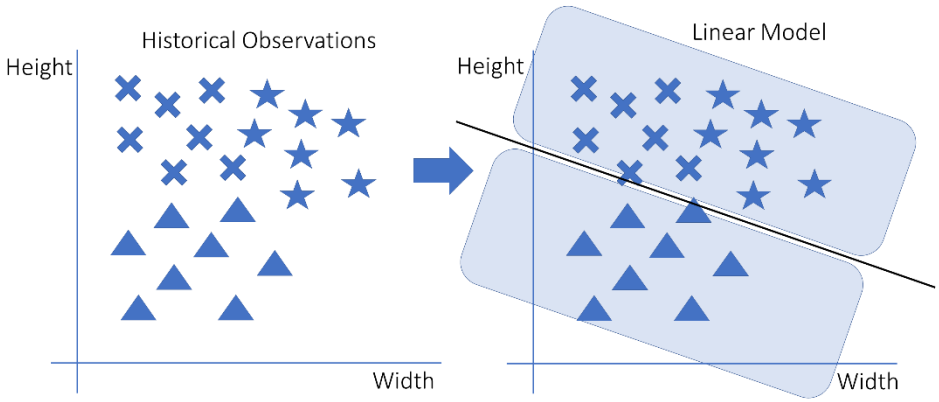
5. **<u>Random Forest</u>**: Το Random Forest είναι ένας ταξινομητής που περιέχει έναν αριθμό δέντρων αποφάσεων σε διάφορα υποσύνολα του δεδομένου συνόλου και παίρνει τον μέσο όρο για να βελτιώσει την προγνωστική ακρίβεια αυτού του συνόλου." Αντί να βασίζεται σε ένα δέντρο απόφασης, το τυχαίο δάσος παίρνει την πρόβλεψη από κάθε δέντρο και με βάση την πλειοψηφία των ψήφων προβλέψεων, και προβλέπει την τελική παραγωγή. Οσο μεγαλύτερος αριθμός δέντρων στο άσος τόσο μεγαλύτερη ακρίβεια και αποτρέπει το πρόβλημα της υπερπροσαρμογής.



6. **<u>XGBoost:</u>** Ο XGBoost είναι ένας αλγόριθμος μηχανικής μάθησης που βασίζεται σε δέντρα αποφάσεων που χρησιμοποιεί ένα πλαίσιο ενίσχυσης κλίσης. Σε προβλήματα πρόβλεψης που αφορούν μη δομημένα δεδομένα (εικόνες, κείμενο, κ.λπ.), τα τεχνητά νευρωνικά δίκτυα τείνουν να έχουν καλύτερη απόδοση από όλους τους άλλους αλγόριθμους ή πλαίσια. Ωστόσο, όταν πρόκειται για μικρά έως μεσαία δομημένα δεδομένα, οι αλγόριθμοι που βασίζονται σε δέντρα αποφάσεων θεωρούνται οι καλύτεροι στην κατηγορία.



7. **<u>K Nearest Neighbours:</u>** Το K-Nearest Neighbours είναι ένας από τους πιο βασικούς αλλά ουσιαστικούς αλγόριθμους ταξινόμησης στη Μηχανική Μάθηση. Ανήκει στον τομέα της supervised μάθησης και βρίσκει έντονη εφαρμογή στην αναγνώριση προτύπων, την εξόρυξη δεδομένων και την ανίχνευση εισβολής. Είναι ευρέως αναλώσιμο σε σενάρια πραγματικής ζωής, καθώς είναι μη παραμετρικό, που σημαίνει ότι δεν κάνει υποκείμενες υποθέσεις σχετικά με τη διανομή των δεδομένων.



Γραφικές παραστάσεις που εξάγουμε κατά την εκτέλεση του κώδικα:

Κατά την ανάλυση των δεδομένων και κατά την εκτέλεση των αλγορίθμων μηχανικής μάθησης που χρησιμοποιήσαμε εξάγαμε κάποιες γραφικές παραστάσεις που θεωρούσαμε αναγκαίες για την περεταίρω ανάλυση και κατανόηση του προβλήματος που κληθήκαμε να αντιμετωπίσουμε. Αρχικά στην πιο κάτω εικόνα παρουσιάζομε πως αναλύσαμε τα αρχικά μας δεδομένα από το υπάρχον dataset έτσι ώστε να μπορούμε να οπτικοποιούμε τα δεδομένα μας με γραφικές παραστάσεις:

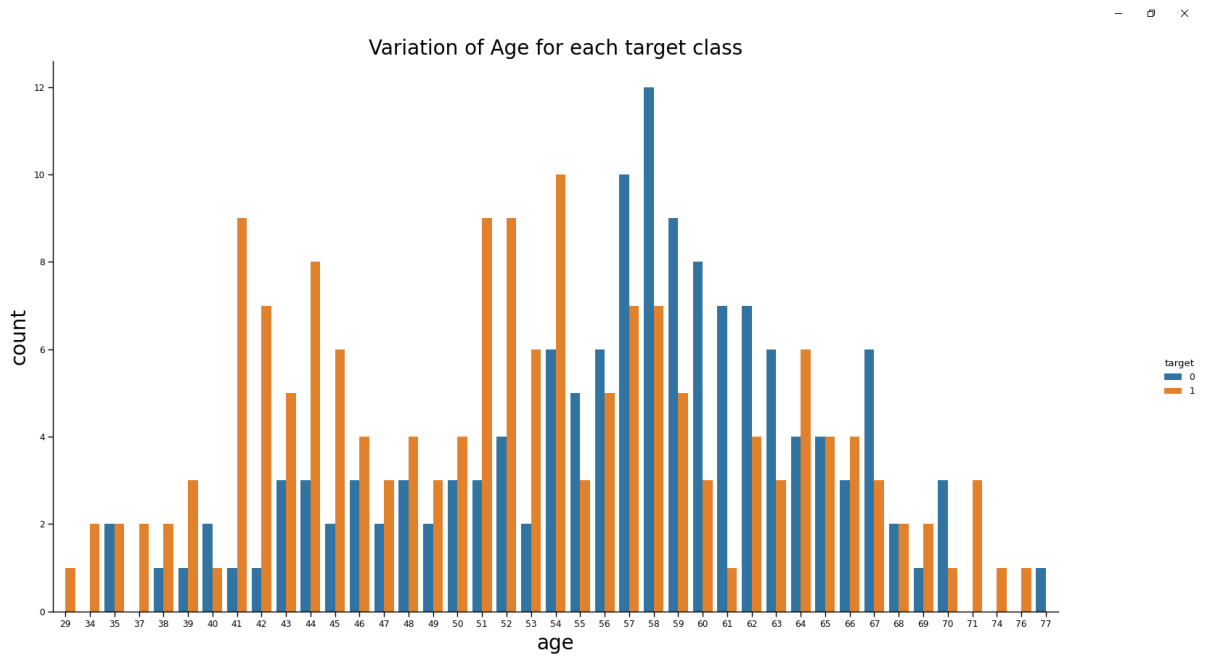
```
df = pd.read_csv("Dataset 1.csv", delimiter=",") #read the dataset
df.columns = ['age', 'sex', 'cp', 'trestbps', 'chol',
              'fbs', 'restecg', 'thalach', 'exang',
              'oldpeak', 'slope', 'ca', 'thal', 'target'] #we define the labales of each column
print(df.isnull().sum())#returns the number of missing values in the dataset.


df['sex'] = df.sex.map({0: 'female', 1: 'male'})#we map 0 as female and 1 as male
df['thal'] = df.thal.fillna(df.thal.mean()) #replace each NaN value with the mean of thal
df['ca'] = df.ca.fillna(df.ca.mean()) #replace each NaN value with the mean of ca
```

Αρχικά εντολή pd.read_csv διαβάζουμε το Dataset 1 το οποίο αποτελεί το dataset που μας δόθηκε. Στη συνέχεια με την εντολή df.columns ορίζουμε τα labels για κάθε στήλη στο dataset μας που αποτελούν και τις μεταβλητές (παράγοντες) για το αν κάποιος ασθενής υποφέρει από καρδιακή νόσο η όχι. Στη συνέχεια με την εντολή df.isnull().sum() επιστρέφουμε το πλήθος των στοιχείων που λείπουν για κάθε στήλη. Μετέπειτα με την εντολή df.sex.map({0: ‘female’, 1: ‘male’}) ορίζουμε ότι οπού υπάρχει στο πεδίο sex 0 είναι female και ότι όπου υπάρχει 1 είναι male.

Στη συνέχεια υλοποιήσαμε τον παρακάτω κώδικα έτσι ώστε να δημιουργήσουμε τις γραφικές που θα παραθέσουμε πιο κάτω:

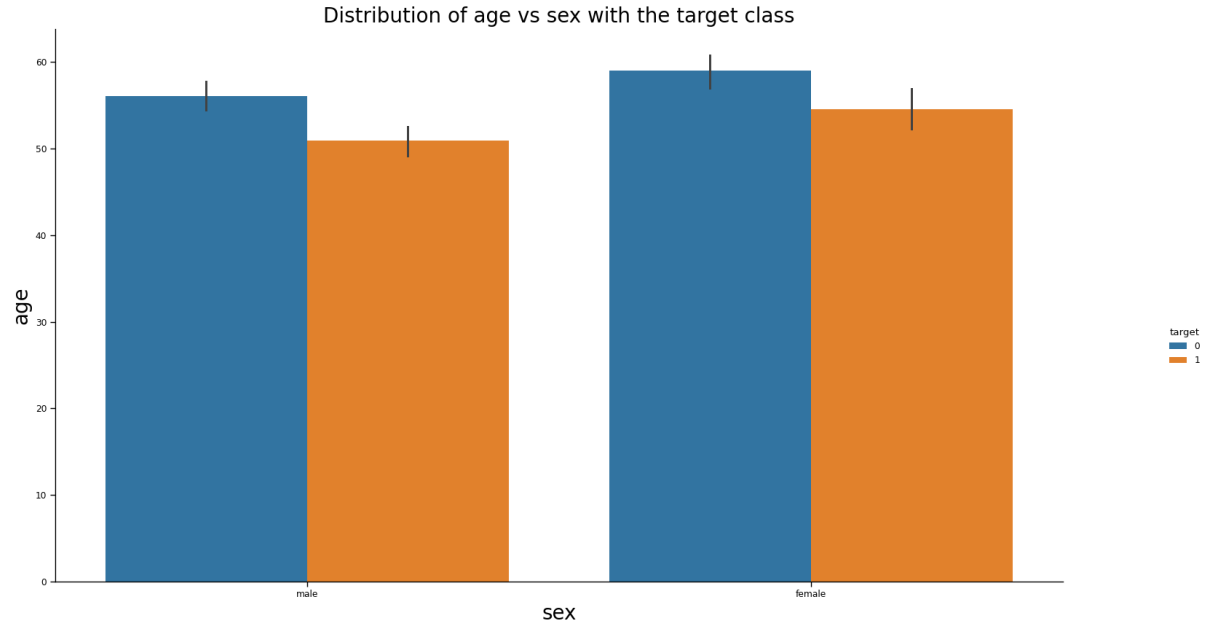
```
df = pd.read_csv("Dataset 1.csv", delimiter=",") #read the dataset
df.columns = ['age', 'sex', 'cp', 'trestbps', 'chol',
              'fbs', 'restecg', 'thalach', 'exang',
              'oldpeak', 'slope', 'ca', 'thal', 'target'] #we define the labales of each column
print(df.isnull().sum())#returns the number of missing values in the dataset.


df['sex'] = df.sex.map({0: 'female', 1: 'male'})#we map 0 as female and 1 as male
```

Ο πιο πάνω κώδικας εξάγει τις παρακάτω γραφικές παραστάσεις:



Όπως μπορούμε να παρατηρήσουμε από τη γραφική παράσταση, οι περισσότεροι άνθρωποι που πάσχουν από καρδιακή νόσο είναι 54 χρόνων και μετά ακολουθούν εκείνοι που είναι 41,51 και 52 χρονών.

Πιο κάτω παραθέτουμε τη γραφική παράσταση η οποία μας δείχνει την κατανομή της ηλικίας και του φύλου για κάθε target class:



Όπως μπορούμε να διακρίνουμε οι γυναίκες υποφέρουν σε μεγαλύτερη ηλικία από καρδιακές παθήσεις σε σχέση με τους άντρες.

Στη συνέχεια της εκτέλεσης του κώδικα, μέσα σε κάθε συνάρτηση που υλοποιήσαμε για κάθε ένα αλγόριθμο που αναφέραμε πιο πάνω , υλοποιήσαμε ένα κομμάτι κώδικα στο οποίο δημιουργείτε το confusion matrix για κάθε αλγόριθμο από το οποίον μπορούμε να

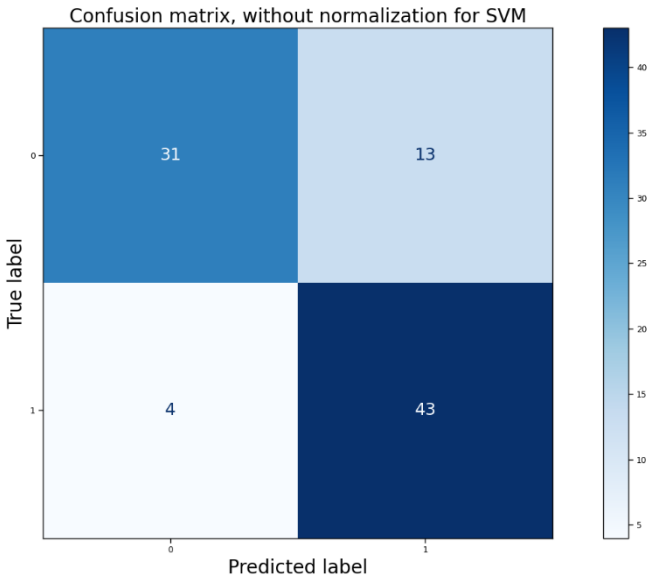
υπολογίσουμε το accuracy για το test κάθε αλγορίθμου. Το κομμάτι κώδικα που υλοποιήσαμε σε κάθε συνάρτηση για την δημιουργία του confusion matrix είναι το παρακάτω:

```
title = "Confusion matrix, without normalization for SVM" #define the title of confusion matrix
disp = ConfusionMatrixDisplay.from_estimator( #we call ConfusionMatrixDisplay function with some arguments to create the confusion matrix
        svm_classifier,
        X_test,
        y_test,
        cmap=plt.cm.Blues #we set color blue for the confusion matrix
    )
disp.ax_.set_title(title)

print(title)#display the title of confusion matrix
print(disp.confusion_matrix)
plt.show() #show the confusion matrix
```

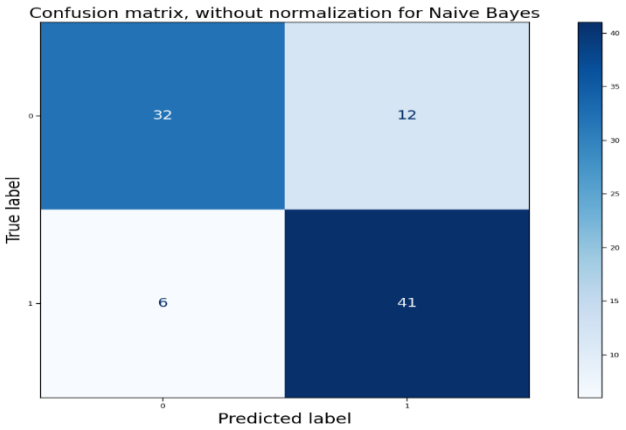
Παραθέτουμε μόνο το κομμάτι που υλοποιήσαμε στην συνάρτηση για τον αλγόριθμο SVM καθώς ο κώδικας σε όλες τις συναρτήσεις είναι ο ίδιος με την μονή διαφορά ότι σε κάθε συνάρτηση παίρνει το κατάλληλο όρισμα (π.χ στην περίπτωση του SVM παίρνει το svm_classifier, στην περίπτωση του Navie Bayes παίρνει το nb_classifier και ουτο καθεξής). Αρχικά όπως βλέπουμε στον κώδικα ορίζουμε τον τίτλο του confusion matrix. Στη συνέχεια μέσα στην μεταβλητή disp καλούμε τη συνάρτηση ConfusionMatrixDisplay όπου παίρνει σαν ορίσματα το svm_classifier το X_test, το y_test και το cmap όπου στην ουσία μας χρωματίζει το confusion matrix σε αποχρώσεις του μπλέ. Μετέπειτα τυπώνουμε το τον τίτλο του confusion matrix και εμφανίζουμε τον confusion matrix.

Πιο κάτω παραθέτουμε τα confusion matrix για κάθε αλγόριθμο:

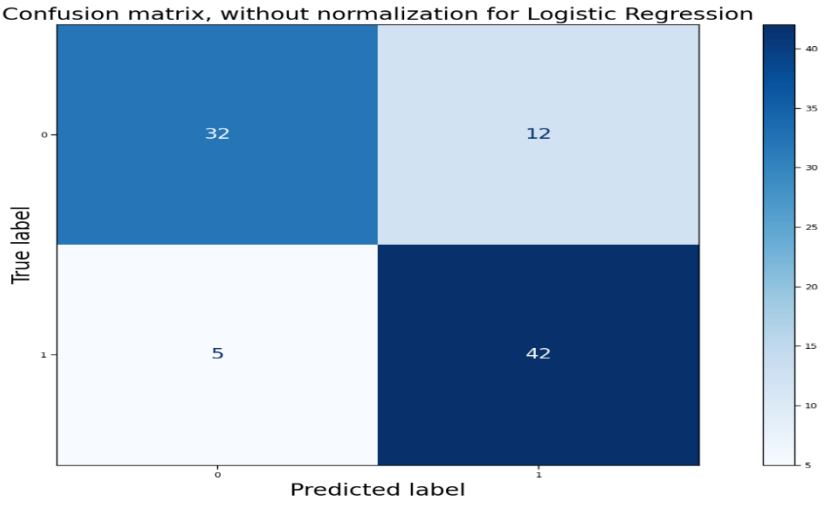
1. **<u>Confusion Matrix for SVM:</u>**



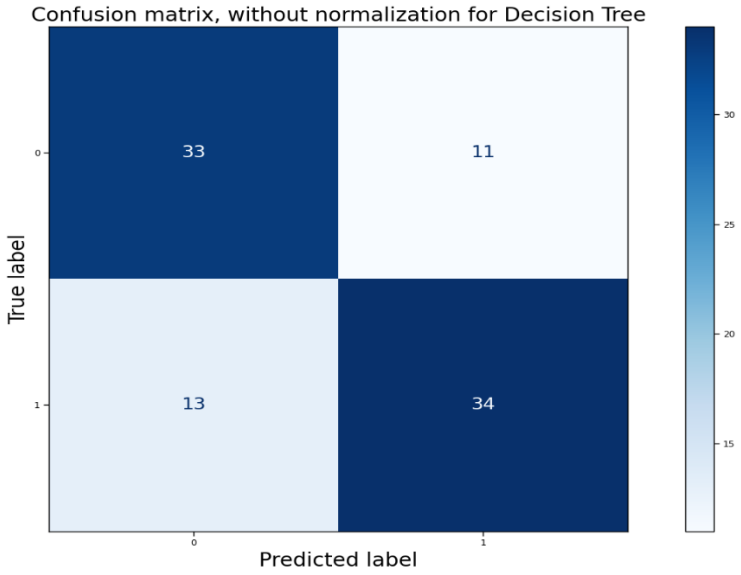
2. Confusion Matrix for Naive Bayes:



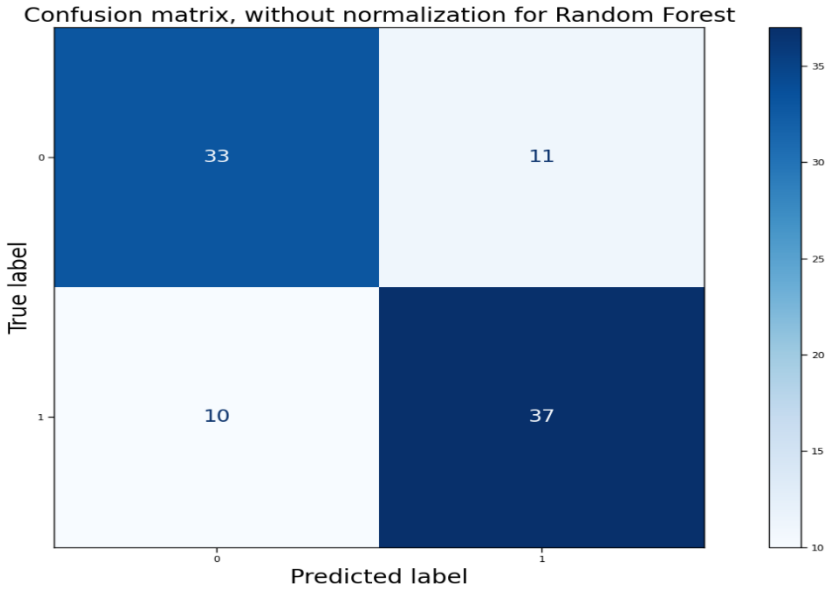
3. Confusion Matrix for Logistic Regression:



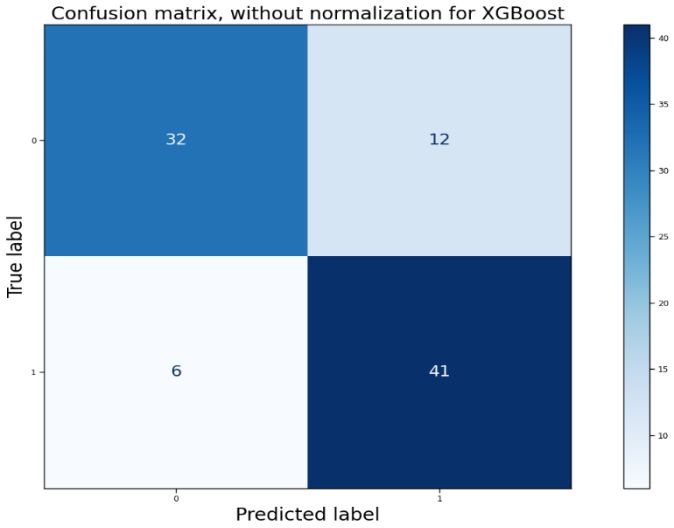
4. Confusion Matrix for Decision Tree:



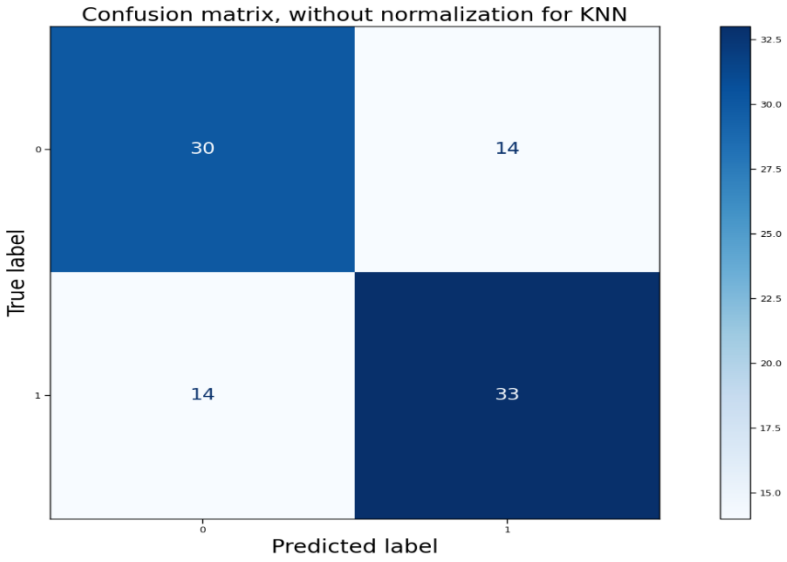
5. Confusion Matrix for Random Forest:



6. Confusion Matrix for XGBoost:



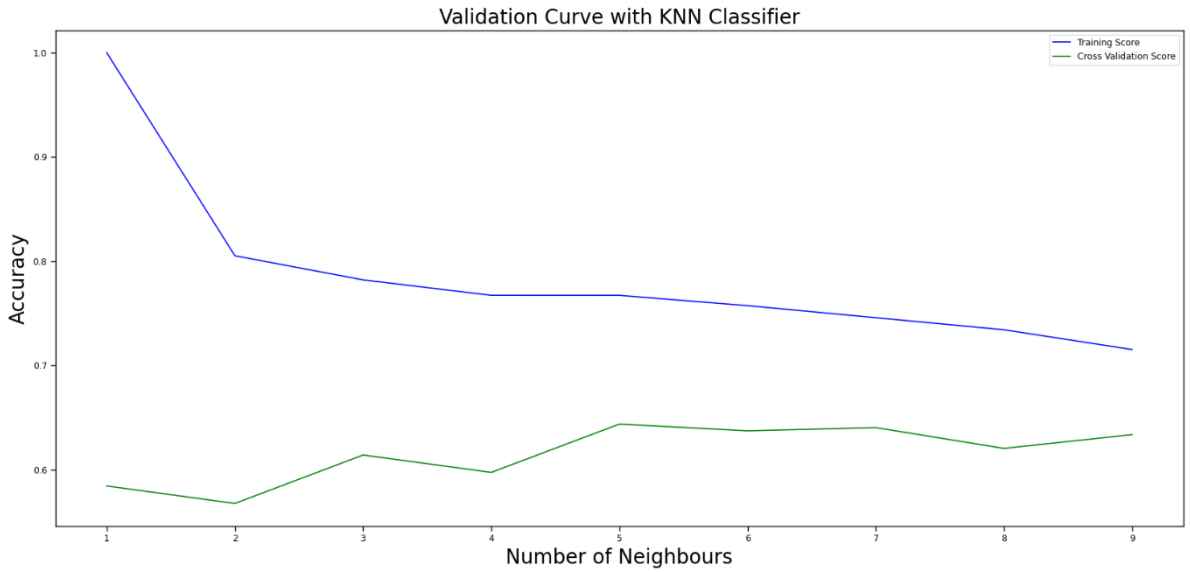
7. Confusion Matrix for K nearest Neighbours:



Επιπλέον στην περίπτωση του αλγορίθμου K nearest Neighbours υλοποιήσαμε μέσα στην συνάρτηση που υλοποιήσαμε τον αλγόριθμο ένα κομμάτι κώδικα στο οποίο σχεδιάζουμε μια γραφική παράσταση Validation Curve. Ο κώδικας που υλοποιήσαμε είναι ο παρακάτω:

```
# Calculate accuracy on training and test set using the
# gamma parameter with 5-fold cross validation
train_score, test_score = validation_curve(KNeighborsClassifier(), X, y,
                                           param_name = "n_neighbors",
                                           param_range = parameter_range,
                                           cv = 5, scoring = "accuracy")
# Calculating mean and standard deviation of training score
mean_train_score = np.mean(train_score, axis = 1)
std_train_score = np.std(train_score, axis = 1)

# Calculating mean and standard deviation of testing score
mean_test_score = np.mean(test_score, axis = 1)
std_test_score = np.std(test_score, axis = 1)

# Plot mean accuracy scores for training and testing scores
plt.plot(parameter_range, mean_train_score,
     label = "Training Score", color = 'b')
plt.plot(parameter_range, mean_test_score,
     label = "Cross Validation Score", color = 'g')

# Creating the plot
plt.title("Validation Curve with KNN Classifier")
plt.xlabel("Number of Neighbours")
plt.ylabel("Accuracy")
plt.tight_layout()
plt.legend(loc = 'best')
plt.show()
```

Όπως φαίνεται ο κώδικας εξηγείται από τα σχόλια που παραθέσαμε σε αυτόν. Πιο κάτω παραθέτουμε την γραφική παράστασή που εξάγει:



Πιο συγκεκριμένα στην παραπάνω γραφική στον άξονα x έχουμε 9 γείτονες όπως ορίσαμε στον κώδικα και στον άξονα y έχουμε το accuracy. Όπως βλέπουμε η γραφική παράσταση μας δείχνει την πορεία του πώς εξελίσσεται το accuracy για το training και το test για το συγκεκριμένο πλήθος γειτόνων που του ορίσαμε.

Πιο κάτω παραθέτουμε τα αποτελέσματα κάθε αλγορίθμου. Στη ουσία δείχνουμε τα accuracies για δεδομένα όταν γίνονται train και τα accuracies για δεδομένα όταν γίνονται test και τις μετρικές Mean Absolute Error, Mean Squared Error και Root Mean Squared Error για να αξιολογήσουμε καλύτερα κάθε αλγόριθμο καλύτερα. Πιο κάτω παραθέτουμε screenshots από τα αποτελέσματα:

Accuracies και μετρικές για τον αλγόριθμο Support Vector Machine:

```
Mean Absolute Error: 0.18681318681318682
Mean Squared Error: 0.18681318681318682
Root Mean Squared Error: 0.4322189107537832

Accuracy for training set for svm = 0.910377358490566
Accuracy for test set for svm = 0.8131868131868132
```

Accuracies και μετρικές για τον αλγόριθμο Naive Bayes:

```
Mean Absolute Error: 0.1978021978021978
Mean Squared Error: 0.1978021978021978
Root Mean Squared Error: 0.4447495899966607

Accuracy for training set for Naive Bayes = 0.8443396226415094
Accuracy for test set for Naive Bayes = 0.8021978021978022
```

Accuracies και μετρικές για τον αλγόριθμο Logistic Regression:

```
Mean Absolute Error: 0.18681318681318682
Mean Squared Error: 0.18681318681318682
Root Mean Squared Error: 0.4322189107537832

Accuracy for training set for Logistic Regression = 0.8537735849056604
Accuracy for test set for Logistic Regression = 0.8131868131868132
```

Accuracies και μετρικές για τον αλγόριθμο Decision Tree:

```
Mean Absolute Error: 0.2857142857142857
Mean Squared Error: 0.2857142857142857
Root Mean Squared Error: 0.5345224838248488

Accuracy for training set for Decision Tree = 1.0
Accuracy for test set for Decision Tree = 0.7142857142857143
```

Accuracies και μετρικές για τον αλγόριθμο Random Forest:

```
Mean Absolute Error: 0.2087912087912088
Mean Squared Error: 0.2087912087912088
Root Mean Squared Error: 0.4569367667316877

Accuracy for training set for Random Forest = 0.9858490566037735
Accuracy for test set for Random Forest = 0.7912087912087912
```

Accuracies και μετρικές για τον αλγόριθμο XGBoost:

```
Mean Absolute Error: 0.1978021978021978
Mean Squared Error: 0.1978021978021978
Root Mean Squared Error: 0.4447495899966607

Accuracy for training set for XGBoost = 1.0
Accuracy for test set for XGBoost = 0.8021978021978022
```

Accuracies και μετρικές για τον αλγόριθμο K Nearest Neighbours:

```
Mean Absolute Error: 0.3076923076923077
Mean Squared Error: 0.3076923076923077
Root Mean Squared Error: 0.5547001962252291

Accuracy for training set for Kneighbors = 0.7311320754716981
Accuracy for test set for Kneighbors = 0.6923076923076923
```

Εξήγηση μετρικών:

- Η μετρική **Mean Absolute Error** πρόκειται για το μέσο απόλυτο σφάλμα του μοντέλου. Αναφέρεται στη μέση τιμή των απόλυτων τιμών κάθε σφάλματος πρόβλεψης σε όλες τις τιμές του συνόλου δεδομένων δοκιμής. Το MAE δίνεται από τον παρακάτω τύπο:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\mathrm{x}_{\mathrm{predict}} - \mathrm{x}_{\mathrm{actual}}\right|$$

  όπου n είναι ο συνολικός αριθμός σφαλμάτων και |xpredict - xactual| το απόλυτο σφάλμα.

- Η μετρική **RMSE** πρόκειται για την ρίζα του μέσου τετραγωνικού σφάλματος. Όσο πιο μεγάλη τιμή πέρνει τόσο περισσότερα σφάλματα υπάρχουν. Το RMSE δίνεται από τον παρακάτω τύπο:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\left|x_{\text{predict}} - x_{\text{actual}}\right|\right)^2}$$

όπου n είναι ο συνολικός αριθμός σφαλμάτων και |xpredict - xactual| το απόλυτο σφάλμα.

- Το μέσο τετράγωνο σφάλμα (MSE) είναι ένα μέτρο του πόσο κοντά είναι μια προσαρμοσμένη γραμμή στα σημεία δεδομένων. Για κάθε σημείο δεδομένων, παίρνετε την απόσταση κάθετα από το σημείο μέχρι την αντίστοιχη τιμή y στην προσαρμογή της καμπύλης (το σφάλμα) και τετραγωνίζετε την τιμή. Ορίζεται από τον πιο κάτω τύπο:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

όπου n είναι ο συνολικός αριθμός σφαλμάτων και |xpredict - xactual| το απόλυτο σφάλμα.

Παρατήρηση: Από τα πιο πάνω αποτελέσματα βλέπουμε ότι το καλύτερο accuracy για το test set το βγάζει ο αλγόριθμος μηχανικής μάθηση Support Vector Machine και ο Logistic regression και το χειρότερο accuracy το βγάζει ο αλγόριθμος K Nearest Neighbours για το test set. Ο αλγόριθμος K nearest Neighbours έχει το χαμηλότερο accuracy για το λόγο ότι κάθε χαρακτηριστικό της μεθόδου έχει το ίδιο αποτέλεσμα στον υπολογισμό της απόστασης. Επιπλέον βλέπουμε από τις μετρικές σφαλμάτων που ορίσαμε ότι ο αλγόριθμος K Nearest Neighbours βγάζει τα μεγαλύτερα σφάλματα σε σύγκριση με του υπολοίπους αλγορίθμους που υλοποιήσαμε οι οποίοι σχεδόν έχουν τα ίδια μεγέθη σφαλμάτων. Ακόμη παρατηρούμε ότι στην περίπτωση του προβλήματος που αντιμετωπίζουμε τα χαμηλότερα σφάλματα τα εξάγει ο αλγόριθμος Logistic Regression.

Στο τέλος για να ελέγξουμε την ορθότητα των αλγορίθμων μας δοκιμάσαμε 5 περιπτώσεις test βάζοντας δικές μας τιμές σε κάθε μεταβλητή που είχαμε στο dataset και τυπώνουμε τη μεταβλητή target που είναι είτε 0 (ο ασθενής δεν υποφέρει από καρδιοπάθεια) είτε 1 (ο ασθενής υποφέρει από καρδιοπάθεια). Πιο κάτω είναι ο κώδικας με τον οποίο κάναμε αυτά τα test:

```
test = [[65, 1, 0, 200, 250, 1, 1, 180, 1, 2.5, 2, 3, 3],
        [100, 1, 0, 200, 300, 1, 2, 200, 1, 2.5, 2, 3, 3],
        [52, 1, 0, 200, 150, 1, 0, 150, 1, 2.3, 0, 1, 3],
        [45, 0, 3, 130, 200, 1, 2, 180,0 ,2.1, 2, 1 ,3],
        [55, 1, 3, 180, 140, 0, 2, 180,0 ,2.5, 2, 3 ,3]]

test = sc.fit_transform(test)
#Test models for 5 cases

for i in range(len(test)): #print target for each algorithm that tell us if a patient suffers from heart disease
    print(f"Patient {i} is: \n")
    print(f"\t SVM: {svm_classifier.predict([test[i]])}" )
    print(f"\t Naive Bayes: {nv_classifier.predict([test[i]])}")
    print(f"\t Logistic Regression: {lr_classifier.predict([test[i]])}")
    print(f"\t Decision Tree: {dt_classifier.predict([test[i]])}")
    print(f"\t Random Forest: {rf_classifier.predict([test[i]])}")
    print(f"\t XGBoost: {xgb_classifier.predict([test[i]])}")
    print(f"\t KNeighbors: {kn_classifier.predict([test[i]])}")
    print()
```

Όπως φαίνεται και στην πιο πάνω εικόνα ορίσαμε μια μεταβλητή test όπου είναι ένας πίνακας ο οποίος έχει άλλους πίνακες μέσα. Στους δύο πίνακες βάλαμε δικές μας τιμές για κάθε μεταβλητή με τη σειρά με την οποία βρίσκονται στο dataset μας. Σε αυτές τις τιμές δεν συμπεριλάβαμε την μεταβλητή target για την οποία περιμένουμε να μας επιστραφεί για κάθε αλγόριθμο στις πιο κάτω γραμμές κώδικα όπου μέσα σε ένα for loop (for i in range(len(test))) για κάθε αλγόριθμο καλούμε τη συνάρτηση predict με όρισμα το test[i] όπου την πρώτη φορά παίρνει τον πρώτο πίνακα, την δεύτερη φορά τον άλλο και ούτω καθεξής. Αυτοί οι δύο πίνακες περιέχουν τιμές για τις μεταβλητές για κάθε ασθενή. Εκτελώντας τον πιο πάνω κώδικα μας παρουσιάζονται τα ακόλουθα αποτελέσματα:

```
Patient 0 is:

     SVM: [0]
     Naive Bayes: [0]
     Logistic Regression: [0]
     Decision Tree: [0]
     Random Forest: [0]
     XGBoost: [0]
     KNeighbors: [0]

Patient 1 is:

     SVM: [0]
     Naive Bayes: [0]
     Logistic Regression: [0]
     Decision Tree: [0]
     Random Forest: [0]
     XGBoost: [0]
     KNeighbors: [0]

Patient 2 is:

     SVM: [0]
     Naive Bayes: [0]
     Logistic Regression: [1]
     Decision Tree: [1]
     Random Forest: [0]
     XGBoost: [1]
     KNeighbors: [0]
```

```
Patient 3 is:

     SVM: [1]
     Naive Bayes: [1]
     Logistic Regression: [1]
     Decision Tree: [1]
     Random Forest: [1]
     XGBoost: [1]
     KNeighbors: [0]

Patient 4 is:

     SVM: [0]
     Naive Bayes: [0]
     Logistic Regression: [1]
     Decision Tree: [1]
     Random Forest: [0]
     XGBoost: [1]
     KNeighbors: [0]
```

Όπως βλέπουμε κάθε για τον ασθενή 0, βάση τα δεδομένα που δώσαμε, ο SVM επιστρέφει target 0 που σημαίνει ότι ο ασθενής δεν πάσχει από καρδιοπάθεια, ο Naive Bayes το ίδιο και αυτός, ο Logistic Regression επιστρέφει target 1 που σημαίνει ότι προβλέπει ότι ο ασθενής πάσχει από καρδιακά προβλήματα, ο Decision και ο Random Forest επιστρέφουν επίσης target 1, ενώ ο XGBoost και K nearest Neighbours επιστρέφουν target 0. Για τους υπόλοιπους ασθενείς πάλι κάθε αλγόριθμος επιστρέφει μια τιμή για το target που δηλώνει την πρόβλεψη του αν ο ασθενής υποφέρει από καρδιακά προβλήματα ή όχι.

Link για το Github όπου είναι ανεβασμένος ο κώδικας μας:

https://github.com/Apalms/Project_Decision_Theory