ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΠΟΛΟΓΙΣΤΙΚΗ ΣΥΓΚΡΙΣΗ ΔΙΑΦΟΡΩΝ ΑΛΓΟΡΙΘΜΩΝ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗΣ ΣΕ ΠΟΛΥΜΕΤΑΒΛΗΤΑ ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ ΡΥΤΗΟΝ

Διπλωματική Εργασία

Ονοματεπώνυμο: Τρίχας Θεόφιλος

ΑΕΜ: mai19076

Επιβλέπων καθηγητής: Σαμαράς Νικόλαος

Θεσσαλονίκη, Ιούνιος 2021

ΥΠΟΛΟΓΙΣΤΙΚΗ ΣΥΓΚΡΙΣΗ ΔΙΑΦΟΡΩΝ ΑΛΓΟΡΙΘΜΩΝ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗΣ ΣΕ ΠΟΛΥΜΕΤΑΒΛΗΤΑ ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ ΡΥΤΗΟΝ

Τρίχας Θεόφιλος

Πτυχίο Οικονομικών, Πανεπιστήμιο Μακεδονίας, 2014

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής

Σαμαράς Νικόλαος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29/06/2021

Ονοματεπώνυμο 1	Ονοματεπώνυμο 2	Ονοματεπώνυμο 3
.....	.....	.....

Τρίχας Θεόφιλος

.....

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των σπουδών μου για την απόκτηση μεταπτυχιακού τίτλου σπουδών στο Τμήμα Εφαρμοσμένης Πληροφορικής, της σχολής Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας.

Το θέμα της παρούσας εργασίας είναι η «Υπολογιστική σύγκριση διαφόρων αλγορίθμων κατηγοριοποίησης σε πολυμεταβλητά σύνολα δεδομένων με τη χρήση της Python».

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της εργασίας, κύριο Σαμαρά Νικόλαο, που με βοήθησε τόσο στην επιλογή του θέματος της εργασίας όσο και στην διεκπεραίωσή της. Τον ευχαριστώ θερμά για την υποστήριξή του, την άμεση ανταπόκριση και τις πολύτιμες συμβουλές που μου έδωσε.

Τέλος, θα αναφερθώ στην αγαπημένη μου οικογένεια. Τους ευχαριστώ που είναι δίπλα μου σε κάθε βήμα της ζωής μου και μου παρείχαν τη δυνατότητα να ολοκληρώσω τις σπουδές μου.

Τρίχας Θεόφιλος, 29/06/2021

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας είναι η μελέτη των βασικότερων αλγορίθμων κατηγοριοποίησης, η κατασκευή ενός προβλεπτικού μοντέλου για κάθε αλγόριθμο με την βοήθεια της γλώσσας προγραμματισμού Python και βιβλιοθηκών μηχανικής μάθησης (numpy, matplotlib, pandas), η ανάλυση τους και η σύγκριση της απόδοσης αυτών των μοντέλων σε διαφορετικά σύνολα δεδομένων παρουσιάζοντας τα αποτελέσματα με βάση την ακρίβεια τους.

Οι αλγόριθμοι που θα μελετηθούν για την Κατηγοριοποίηση (Classification) των δεδομένων είναι:

1. K-Nearest Neighbors (K-NN)
2. Naive Bayes
3. Logistic Regression
4. Decision Tree (CART)

Παράλληλα χρησιμοποιήθηκε και ένας αλγόριθμος Συσταδοποίησης (Clustering) ο οποίος χρειάστηκε για περεταίρω ανάλυση των δεδομένων. Αυτός ο αλγόριθμος είναι ο:

1. K-Means model

Για την κατασκευή ενός μοντέλου για κάθε αλγόριθμο, θα χρησιμοποιηθούν σύνολα δεδομένων (data sets) από το UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php).

Η μετάφραση ορισμένων όρων έχει βασιστεί σε διαδεδομένες λέξεις για τις αντίστοιχες αγγλικές. Προκειμένου να αποφευχθούν συγχύσεις αλλά και για την διευκόλυνση των αναγνωστών πολλοί όροι συμπεριλαμβάνονται σε παρενθέσεις με την αγγλική τους ονομασία.

Λέξεις Κλειδιά:

Machine Learning, Logistic Regression, K-Nearest Neighbors (K-NN), Decision Trees, Naïve Bayes, K-Means model, *k*-Fold Validation, Feature Scaling

Abstract

The purpose of this research is the study of the variety of basic Classification methods, building model for each algorithm with the use of Python and Machine Learning libraries (numpy, matplotlib, pandas), their analysis and comparison of the performance of these models in different data sets presenting the results based on their accuracy.

The algorithms that are going to be studied for the classification if the data sets are:

1. K-Nearest Neighbors (K-NN)
2. Naive Bayes
3. Logistic Regression
4. Decision Tree (CART)

At the same time, a clustering algorithm was used which was needed for further analysis of the data sets. This algorithm is:

1. K-Means model

To build a generator for each algorithm, you will use sets from the UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php).

The translation of some terms has been based on common English words. In order to avoid confusion but also for the convenience of readers many terms are included in parentheses with their English name.

Keywords:

Machine Learning, Logistic Regression, K-Nearest Neighbors (K-NN), Decision Trees, Naïve Bayes, K-Means model, *k*-Fold Validation, Feature Scaling

Περιεχόμενα

Λίστα Πινάκων **9**

1. Εισαγωγή **10**

1.1. Ορισμός και στόχοι του προβλήματος 10

1.2. Δομή εργασίας 11

2. Κατηγοριοποίηση (Classification) **12**

2.1. K-Nearest Neighbors (K-NN) 12

2.2. Naive Bayes 14

2.3. Logistic Regression 17

2.4. Decision Tree (CART) 21

3. Συσταδοποίηση (Clustering) **25**

3.1. K-Means model 25

4. Data Sets **29**

4.1. Introduction 29

4.2. Internet Firewall Data Set 29

4.2.1. Ανάλυση χαρακτηριστικών του Data Set 30

4.2.2. Εφαρμογή του Data Set σε διάφορα άρθρα 32

4.3. Teaching Assistant Evaluation Data Set 33

4.3.1. Ανάλυση χαρακτηριστικών του Data Set 34

4.3.2. Εφαρμογή του Data Set σε διάφορα άρθρα 34

4.4. Car Evaluation Data Set 35

4.4.1 Ανάλυση χαρακτηριστικών του Data Set 36

4.4.2. Εφαρμογή του Data Set σε διάφορα άρθρα 37

4.5. Electrical Grid Stability Simulated Data Set 40

4.5.1. Ανάλυση χαρακτηριστικών του Data Set 41

4.5.2. Εφαρμογή του Data Set σε διάφορα άρθρα 41

5. Αποτελέσματα Μεθόδων Μηχανικής Μάθησης **43**

5.1. Προ επεξεργασία στο Internet Firewall Data Set 43

5.1.1. Σύγκριση Αποτελεσμάτων του IF Data Set 47

5.2. Προ επεξεργασία στο Teaching Assistant Evaluation Data Set 48

5.2.1. Σύγκριση Αποτελεσμάτων του TAE Data Set 54

5.3. Προ επεξεργασία στο Car Evaluation Data Set 56

5.3.1. Σύγκριση Αποτελεσμάτων του CE Data Set 59

5.4. Προ επεξεργασία στο Electrical Grid Stability Data Set 59

5.4.1. Σύγκριση Αποτελεσμάτων του EGSS Data Set 62

6. Συμπεράσματα 64

Βιβλιογραφία 66

ΠΑΡΑΡΤΗΜΑ 68

Λίστα Σχημάτων

Εικόνα 1: Παράδειγμα K-Nearest Neighbors για 3-NN και 6-NN……13

Εικόνα 2: Παράδειγμα Naïve Bayes για εισόδημα και ηλικία……16

Εικόνα 3: Παράδειγμα ενός logistic regression……18

Εικόνα 4: Παράδειγμα ενός decision tree……22

Εικόνα 5: Παράδειγμα ενός K-means μοντέλου……...26

Εικόνα 6: Παράδειγμα ενός Support Vector Machine μοντέλου……..32

Εικόνα 7: Απλή μορφή Naïve-Bayes αλγόριθμου……38

Εικόνα 8: Απλή μορφή TAN αλγόριθμου……..…38

Εικόνα 9: Απλή μορφή BAN αλγόριθμου……39

Εικόνα 10: Απλή μορφή GBN αλγόριθμου……..39

Εικόνα 11: Decision tree για το σύστημά τους…….……....42

Εικόνα 12: Σύγκριση αποτελεσμάτων……..48

Εικόνα 13: The Elbow Method……52

Εικόνα 14: Accuracy for TAE Data Set……...55

Εικόνα 15: Accuracy for CE Data Set……..59

Εικόνα 16: Accuracy for EGSS Data Set……62

Λίστα Πινάκων

Πίνακας 1: Ψευδοκώδικας k-Nearest Neighbor……13

Πίνακας 2: Ψευδοκώδικας Naïve Bayes……...15

Πίνακας 3: Ψευδοκώδικας Logistic Regression……19

Πίνακας 4: Ψευδοκώδικας Decision Tree (Cart)……..22

Πίνακας 5: Ψευδοκώδικας K-Means model……26

Πίνακας 6: Χαρακτηριστικά του Data set……30

Πίνακας 7: Ενέργειες της εξαρτημένης μεταβλητής……31

Πίνακας 8: Αποτελέσματα SVM (IF)……..…33

Πίνακας 9: Χαρακτηριστικά του data set (TAE)……33

Πίνακας 10: Χαρακτηριστικά του data set (CE)……..36

Πίνακας 11: Χαρακτηριστικά του data set (EGSS)……40

Πίνακας 12: Συνολικά Αποτελέσματα IF Data Set……..47

Πίνακας 13: Συνολικά Αποτελέσματα Error Rates για το TAE Data Set……....54

Πίνακας 14: Συνολικά Error Rates για το TAE Data Set με K-Means……55

Πίνακας 15: Συνολικά Αποτελέσματα Accuracy για CE Data Set……..59

Πίνακας 16: Συνολικά Αποτελέσματα Accuracy για EGSS Data Set……62

1. Εισαγωγή

Η μηχανική μάθηση (Machine Learning) είναι ένας τομέας στην Επιστήμη των Υπολογιστών που υπάρχει από το 1950. Είναι μία μελέτη αλγορίθμων υπολογιστών η οποία βελτιώνεται μέσα από την χρήση δεδομένων. Οι αλγόριθμοι αυτοί δημιουργούν μοντέλα βασισμένα σε δείγματα δεδομένων, γνωστά και ως "δεδομένα εκπαίδευσης (training data)" προκειμένου να λαμβάνουν προβλέψεις και αποφάσεις. Οι αλγόριθμοι μηχανικής εκμάθησης χρησιμοποιούνται σε πάρα πολλές εφαρμογές όπως για παράδειγμα στο φιλτράρισμα των emails το οποίο είναι πολύ δύσκολο να πραγματοποιηθεί μέσω συμβατών αλγορίθμων.

Οι οικογένειες αλγορίθμων που μας ενδιαφέρουν σε αυτήν την διπλωματική εργασία είναι: η Κατηγοριοποίηση (Classification), η οποία σχετίζεται στενά με την Κατηγοριοποίηση (Categorization) και σε κάποιες συγκεκριμένες περιπτώσεις η ταυτόχρονη χρήση της Συσταδοποίησης (Clustering). Για την κατηγοριοποίηση στόχος μας είναι να προβλέψουμε την τιμή ενός χαρακτηριστικού με μια συγκεκριμένη ακρίβεια βάσει των χαρακτηριστικών που προσφέρει ο κάθε αλγόριθμος.

1.1. Ορισμός και στόχοι του προβλήματος

Υπάρχει μία μεγάλη κατηγορία αλγορίθμων που μπορούν να χρησιμοποιηθούν για την κατηγοριοποίηση, όπως Δέντρα Αποφάσεων (Decision Trees) ή Νευρωνικά Δίκτυα (Neural Networks). Λόγω των διαφορετικών λειτουργιών κάθε αλγορίθμου, δεν υπάρχει καμιά οδηγία για το ποιος αλγόριθμος ταιριάζει καλύτερα στα δεδομένα που έχει κάποιος, προκειμένου να παράγει το καλύτερο μοντέλο. Πρέπει να εφαρμοστούν και να δοκιμαστούν ώστε να δει κάποιος τα αποτελέσματα και να κάνει συγκρίσεις.

Ο σκοπός αυτής της έρευνας είναι να συγκρίνει τους βασικότερους αλγόριθμους κατηγοριοποίησης σε σύνολα δεδομένων διαφόρων πολυπλοκότητας. Για να προσδιορίσουμε την αποτελεσματικότητα κάθε αλγορίθμου θα χρησιμοποιήσουμε την ακρίβεια του μοντέλου που δημιουργήθηκε από κάθε έναν και σε κάποιους το ποσοστό σφάλματος που έχει ο καθένας με βάση τα δεδομένα.

1.2. Δομή εργασίας

Αρχικά, δίνεται μια σύντομη περιγραφή των αλγορίθμων κατηγοριοποίησης (Classification) για να βοηθήσει τον αναγνώστη να κατανοήσει τα βασικά του τρόπου λειτουργίας κάθε αλγορίθμου. Υπάρχει, επίσης, μια σύνοψη των αλγορίθμων που χρησιμοποιούνται ως μέρος της προ επεξεργασίας για ορισμένα σύνολα δεδομένων. Στην συνέχεια δίνεται περιγραφή για τον αλγόριθμο της συσταδοποίησης (Clustering) που ενσωματώθηκε κι αυτός για να βοηθήσει στην ανάπτυξη των αποτελεσμάτων

Στα επόμενα κεφάλαια, τέταρτο έως πέμπτο, μελετάμε τα σύνολα δεδομένων που δωρίστηκαν στο UCI. Σε κάθε κεφάλαιο υπάρχει μια σύντομη περιγραφή του συνόλου δεδομένων και τα χαρακτηριστικά που περιέχουν με σκοπό την καλύτερη κατανόηση αυτών. Στη συνέχεια, εξηγείται η μεθοδολογία επεξεργασίας που χρησιμοποιήθηκε και τέλος παρουσιάζονται τα αποτελέσματα κάθε αλγορίθμου (ακρίβεια και ποσοστά σφάλματος).

Τέλος στο κεφάλαιο 6 είναι τα συμπεράσματα που βασίζονται στα συνολικά αποτελέσματα όλων των αλγορίθμων.

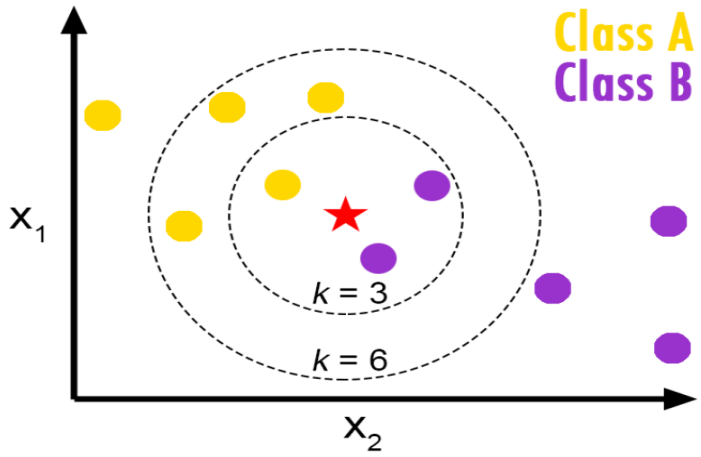
2. Κατηγοριοποίηση (Classification)

Σε αυτήν την ενότητα, εξηγούνται εν συντομία όλοι οι αλγόριθμοι που πρόκειται να χρησιμοποιηθούν, προκειμένου κάθε αναγνώστης να μπορέσει με ευκολία να κατανοήσει τα βασικά τους μέρη ώστε στην συνέχεια να μπορέσει να κατανοήσει και τον τρόπο με τον οποίο αναπαράχθηκαν τα αποτελέσματα στον κάθε αλγόριθμο. Οι συγκεκριμένοι αλγόριθμοι χρησιμοποιήθηκαν για την κατηγοριοποίηση των δεδομένων ώστε να βγάλουμε τα επιθυμητά αποτελέσματα στις ακρίβειες. Θα γίνει μία περιγραφή στο πως λειτουργούν, την βασική τους μορφή, θα αναλυθούν κάποιοι μαθηματικοί όροι και θα παρουσιαστούν οι ψευδοκώδικες του καθενός αντίστοιχα. Τέλος, θα αναλυθούν οι εντολές με βάση της `sklearn` βιβλιοθήκης που χρησιμοποιήθηκε για τους αλγορίθμους στην Python και τι παραμέτρους μπορεί να πάρει ο καθένας αναλόγως με το τι σύνολα δεδομένων έχει στην κατοχή του.

2.1. K-Nearest Neighbors (K-NN)

Ο αλγόριθμος K-NN είναι μια μη παραμετρική (non-parametric) μέθοδος που χρησιμοποιείται για την ταξινόμηση και είναι από τους πιο δημοφιλείς αλγορίθμους ταξινόμησης και ταυτόχρονα από τους πιο απλούς. Χρησιμοποιείται για να αντιμετωπίσει προβλήματα που έχουν άγνωστες και μη κανονικές κατανομές. Ωστόσο, ο συγκεκριμένος αλγόριθμος χρειάζεται πάντα ένα μεγάλο σύνολο δεδομένων για να αποδώσει το μέγιστο δυνατό.

Λαμβάνοντας υπόψη ένα σύνολο δεδομένων εκπαίδευσης και μια προσαρμοσμένη μέτρηση απόστασης (συνήθως χρησιμοποιείται η Ευκλείδεια απόσταση) μπορούμε να προβλέψουμε σε ποια κλάση θα ενσωματωθεί το νέο data point αναλόγως την πλειοψηφία από τους γείτονές της. Χρησιμοποιώντας τη μέτρηση της απόστασης που παρέχουμε, αναζητούμε τους πλησιέστερους γείτονές της και αναλόγως ποια κλάση έχει τους περισσότερους, εκεί θα ενσωματωθεί το νέο data point έχοντας δηλώσει εξαρχής των αριθμό των k.



Εικόνα 1: Παράδειγμα K-Nearest Neighbors για 3-NN και 6-NN

Ανακτήθηκε από: https://equipintelligence.medium.com/k-nearest-neighbor-classifier-knn-machine-learning-algorithms-ed62feb86582

Η καλύτερη επιλογή του k εξαρτάται από τα δεδομένα που θα έχει κάποιος στην κατοχή του. Γενικά, όσο μεγαλύτερο είναι το k τόσο μειώνονται τα επίπεδα θορύβου που έχει το σύνολο δεδομένων. Ωστόσο, τα όρια μεταξύ των τάξεων γίνονται λιγότερο διακριτά.

Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου με βάση όσα ειπώθηκαν προηγουμένως:

Πίνακας 1: Ψευδοκώδικας k-Nearest Neighbor

```
k-Nearest Neighbor
Classify (X,Y,x) // X: training data, Y: class labels of
X, x: Unknown sample
for i = 1 to m do
    Compute distance d(X_i,x)
end for
Compute set I containing indices for the k smallest
distances d(X_i,x)
return majority label for {Y_i where i € I}
```

Η γενική μορφή του αλγορίθμου στην sklearn βιβλιοθήκη η οποία χρησιμοποιήθηκε στην δικιά μας έρευνα στην γλώσσα Python είναι της κλάσης:

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs)
```

Όπου βλέποντας την κλάση καταλαβαίνουμε και τις παραμέτρους τις οποίες μπορεί να πάρει ο συγκεκριμένος αλγόριθμος. Ο 'n_neighbors' μας δίνει τον αριθμό των γειτόνων που θα χρησιμοποιηθεί ώστε να διαλέγει κάθε φορά τόσους πλησιέστερους όσους δηλώνετε σε αυτήν την παράμετρο. Προκαθορισμένο αριθμό έχει τον αριθμό 5. Στην συνέχεια η παράμετρος weights είναι η συνάρτηση βάρους που χρησιμοποιείται στην πρόβλεψη και μπορεί να πάρει είτε την τιμή uniform, η οποία είναι και η προκαθορισμένη τιμή, έχει ομοιόμορφα βάρη και όλα τα σημεία σε κάθε γειτονιά σταθμίζονται εξίσου όμοια, είτε την τιμή distance όπου εδώ τα σημεία βάρους είναι αντίστροφα της απόστασής τους και εδώ οι πιο κοντινοί γείτονες έχουν μεγαλύτερη επίδραση από τους γείτονες σε πιο μακρινές αποστάσεις. Η παράμετρος 'algorithm' χρησιμοποιείται για να υπολογίζει τους κοντινότερους γείτονες και μπορεί να πάρει τιμές 'ball_tree', 'kd_tree', 'brute' και 'auto'. Το 'auto' είναι η προκαθορισμένη τιμή και θα προσπαθήσει από μόνο του να διαλέξει τον καταλληλότερο αλγόριθμο με βάσει τις τιμές. Η παράμετρος 'leaf size' προσαρμόζει το μέγεθος των φύλλων και επηρεάζει την ταχύτητα της κατασκευής του προβλήματος. Η παράμετρος 'p' χρησιμοποιείται για την μετρική Minkowski και παίρνει τιμές 1 για αποστάσεις Manhattan και 2 και Ευκλείδειες αποστάσεις. Τέλος, η παράμετρος 'n_jobs' δίνει τον αριθμό των παράλληλων εργασιών που θα εκτελούνται για την αναζήτηση των γειτόνων.

Έτσι, μπορεί πολύ εύκολα να καλεστεί με την παρακάτω εντολή στην python κάνοντας χρήση της παραπάνω βιβλιοθήκης:

```
« from sklearn.neighbors import KNeighborsClassifier »
```

2.2. Naive Bayes

Το Naïve Bayes είναι ένα μοντέλο πιθανότητας υπό όρους που χρησιμοποιεί το Bayes Θεώρημα για να υπολογίσει την πιθανότητα για κάθε ετικέτα κατηγορίας C_k.

Με δεδομένο ένα training σύνολο N χαρακτηριστικών και ενός x παραδείγματος που αντιπροσωπεύεται από ένα διάνυσμα $x = \{x1,\ldots, xN\}$ χρησιμοποιούμε τον τύπο:

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

Μας ενδιαφέρει να υπολογιστεί ο αριθμητής του κλάσματός μας που είναι ίσος με το μοντέλο κοινών πιθανοτήτων $p\ (C_k, x1,\ldots, xN)$. Χρησιμοποιώντας τον κανόνα της αλυσίδας μπορούμε να ξαναγράψουμε αυτήν την πιθανότητα ως εξής:

$p(C_k,x1,\ldots,xN) = p(x1|x2,\ldots,xN,C_k)p(x2|x3,\ldots,xN,C_k)\ldots p(xN-1|xN, C_k)p(xN|C_k)$

Αυτός ο ταξινομητής ονομάζεται "naive" επειδή όλα τα χαρακτηριστικά είναι αμοιβαία ανεξάρτητα και εξαρτώνται από την κατηγορία C_k.

Έτσι, έχοντας αυτήν την υπόθεση μπορεί να γίνει η παρακάτω προσέγγιση:

$p(x_i\,|\,x_i\,|+1,\ldots,xN,C_k) = p(x_i\,|\,C_k)$.

Μετά από αυτό, το μοντέλο μπορεί να εκφραστεί ως:

$p(C_k,x1,\ldots,xN) = \frac{1}{T}p(C_k)\prod_{i=1}^{N} p(x_i\,|\,C_k)$ όπου το T είναι ένας παράγοντας κλιμάκωσης.

Αφού δοθεί ένα μοντέλο, μπορεί να υπολογιστεί η πιθανότητα να ανήκει σε μία κλάση. Στη συνέχεια, εφαρμόζεται ένας κανόνας απόφασης για να βρεθεί η κλάση αυτού του στοιχείου. Ο πιο συνηθισμένος κανόνας απόφασης είναι γνωστός ως “maximum a posteriori” κανόνας, που σημαίνει ότι εκχωρείται στην ετικέτα κλάσης με την υψηλότερη πιθανότητα.

Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου με βάση όσα ειπώθηκαν προηγουμένως:

Πίνακας 2: Ψευδοκώδικας Naïve Bayes

```
Naïve Bayes
Input:
    Training Data Set T
    C=(c1, c2,…, cn)     // value of the predictor variable
in testing dataset
Output:
```

A class of testing dataset

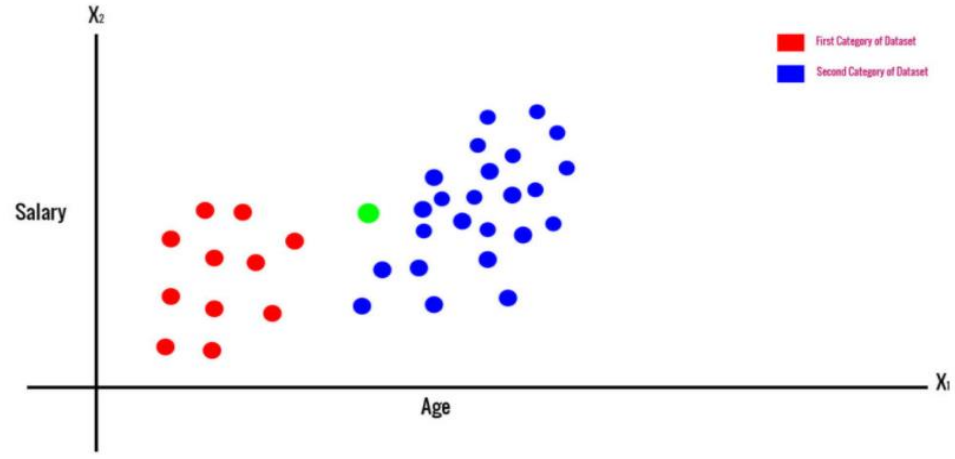
Step:

1. Read the training dataset T;
2. Calculate the mean and standard deviation of the predictor variables in each class;
3. Repeat

   Calculate the probability of c_i using the gauss density equation in each class

   Until the probability of all predictor variables (c_1, c_2, c_3,…, c_n) has been calculated
4. Calculate the probability for each class;
5. Get the greatest probability

Έτσι με βάση όσα σχολιάστηκαν πιο πάνω μπορούμε να δούμε το επόμενο παράδειγμα. Έχοντας παραδείγματος χάριν έναν άξονα όπου στον κάθετο έχουμε το μισθό κάποιου και στον οριζόντιο την ηλικία, οι κόκκινες κουκίδες είναι η πρώτη κατηγορία η οποία είναι όσοι πάνε με τα πόδια στην δουλειά και τις μπλε κουκίδες όπου είναι αυτοί που πάνε με το αυτοκίνητο.



Εικόνα 2: Παράδειγμα Naïve Bayes για εισόδημα και ηλικία

Ανακτήθηκε από: https://www.codershood.info/2019/01/14/naive-bayes-classifier-using-python-with-example/

Στην ουσία θέλουμε να υπολογίσουμε την πιθανότητα της τυχαίας πράσινης κουκίδας να ανήκει στην πρώτη ή στην δεύτερη κλάση κατηγορίας. Με το παραπάνω παράδειγμα επιδιώκουμε να υπολογίσουμε την πιθανότητα που έχει κάποιο τυχαίο άτομο να πάει με τα πόδια ή με το αυτοκίνητο στην δουλειά του αναλόγως πού θα είναι τοποθετημένος ανάμεσα στους δύο άξονες.

Η γενική μορφή του αλγορίθμου στην sklearn βιβλιοθήκη η οποία χρησιμοποιήθηκε στην δικιά μας έρευνα στην γλώσσα Python είναι της κλάσης:

```
class sklearn.naive_bayes.GaussianNB(*, priors=None, var_smoothing=1e-09)
```

Ο συγκεκριμένος αλγόριθμος παίρνει δύο παραμέτρους όπως βλέπουμε από την παραπάνω κλάση. Η πρώτη παράμετρος είναι η `'prios'` και αν δηλωθεί από τον χρήστη δείχνει τις προηγούμενες πιθανότητες των κλάσεων. Αν ο χρήστης την καθορίσει οι πιθανότητες αυτές δεν προσαρμόζονται σύμφωνα με τα δεδομένα. Η προκαθορισμένη τιμή της παραμέτρου ισούται με `None`. Η δεύτερη παράμετρος είναι η `'var smoothing'` και καθορίζει το τμήμα της μεγαλύτερης απόδοσης όλων των χαρακτηριστικών για να υπάρξει σταθερότητα υπολογισμού. Εδώ η προκαθορισμένη τιμή είναι η `1e-09`.

Έτσι, μπορεί πολύ εύκολα να καλεστεί με την παρακάτω εντολή στην python κάνοντας χρήση της παραπάνω βιβλιοθήκης:

« `from sklearn.naive_bayes import GaussianNB` »

2.3. Logistic Regression

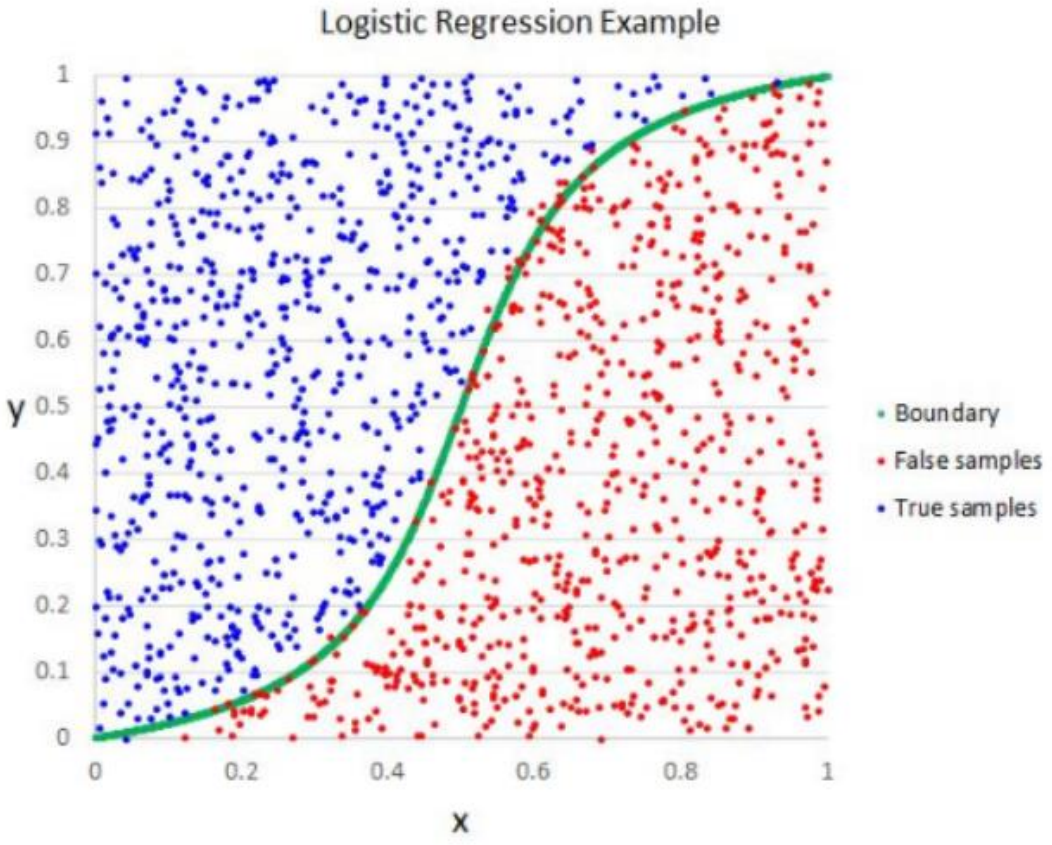
Στην στατιστική, το logistic μοντέλο χρησιμοποιείται για την μοντελοποίηση μίας συγκεκριμένης κλάσης ή ενός γεγονότος (πχ True ή False). Στην ουσία είναι σαν να δίνει πιθανότητα μεταξύ του 0 και 1. Το logistic regression είναι ένα στατιστικό μοντέλο που μας δίνει μία τέτοια πιθανότητα παίρνοντας ένα μοντέλο και χρησιμοποιώντας μία λογιστική συνάρτηση, μοντελοποιεί μία δυαδική μεταβλητή. Ωστόσο, μπορεί να επεκταθεί και σε πιο περίπλοκα προβλήματα.

Δεδομένου ότι έχουμε ένα μοντέλο με δύο προβλέψεις x_1 και x_2, μία δυαδική μεταβλητή Y την οποία δηλώνουμε p = P (Y=1) και έχοντας μία γραμμική σχέση μεταξύ των μεταβλητών πρόβλεψης και των log-odds αυτή η γραμμική σχέση μπορεί να οριστεί ως εξής:

$$l = \log_b \frac{p}{p-1} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

- l είναι το log-odds
- b είναι η βάση του λογάριθμου
- β_i είναι οι παράμετροι του μοντέλου

Η παραπάνω σχέση, μας δημιουργεί την πράσινη γραμμή στους παρακάτω άξονες η οποία μπορεί να περιγράψει ακριβώς την διαδικασία που ελέγχει ο αλγόριθμος:



Εικόνα 3: Παράδειγμα ενός logistic regression

Ανακτήθηκε από: https://www.datasciencecentral.com/profiles/blogs/why-logistic-regression-should-be-the-last-thing-you-learn-when-b

Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου με βάση όσα ειπώθηκαν προηγουμένως και ο οποίος όταν έχει πολλαπλές κλάσης χρησιμοποιεί το 1 vs all αλγόριθμο. Πρόκειται για μία στρατηγική κατηγοριοποίησης πολλαπλών κλάσεων η οποία έχει ως σκοπό την εκπαίδευση ενός απλού classifier για κάθε κλάση παίρνοντας δείγματα από κάθε κλάση ως θετικά και τα υπόλοιπα που μένουν ως αρνητικά. Ο αλγόριθμος εκπαίδευσης για όλα αυτά χρησιμοποιεί το logistic regression μοντέλο και ο οποίος κατασκευάζεται από διάφορους δυαδικούς classifiers:

Πίνακας 3: Ψευδοκώδικας Logistic Regression

Logistic Regression

Input:

- Training algorithm L (logistic regression)
- Sample matrix X
- Labels vector y = [1,…K]

Main:

For i = 1:K

Create a new binary vector y_i for each label where y_i = 1 if it belong to . the label and y_i = 0 if it does not belong.

Apply L to X to find θ_i

Output:

θ_i Parameters vector for each regressor

Η γενική μορφή του αλγορίθμου στην sklearn βιβλιοθήκη η οποία χρησιμοποιήθηκε στην δικιά μας έρευνα στην γλώσσα Python είναι της κλάσης:

```
class sklearn.linear_model.LogisticRegression(penalty='
l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True
, intercept_scaling=1, class_weight=None, random_state=No
ne, solver='lbfgs', max_iter=100, multi_class='auto', ver
bose=0, warm_start=False, n_jobs=None, l1_ratio=None)
```

Ο συγκεκριμένος αλγόριθμος όπως μπορούμε να δούμε παίρνει αρκετούς παραμέτρους οι οποίοι θα αναλυθούν. Η παράμετρος 'penalty' χρησιμοποιείται για να καθορίσει την ποινή και μπορεί να πάρει τρεις τιμές 'l1', 'l2', 'elasticnet' και 'none'. Προκαθορισμένη τιμή είναι η l2. Η παράμετρος 'dual' έχει προκαθορισμένη τιμή false και μπορεί να πάρει είτε αυτήν είτε true. Εφαρμόζεται πάνω στην ποινή και προτιμάται dual = False όταν τα n_samples είναι περισσότερα από τα n_features. Στην συνέχεια, η παράμετρος 'tol' παίρνει μία float τιμή για να ορίσει τα κριτήρια διακοπής και έχει ως προκαθορισμένη τιμή την 1e-4. Η παράμετρος 'C' καθορίζει την αντίστροφη ισχύς κανονικοποίησης και πρέπει να είναι ένας θετικός float αριθμός (η default τιμή του εδώ είναι το 1.0). Η παράμετρος 'fit_intercept' είναι μία boolean μεταβλητή και ορίζει αν στην συνάρτηση πρέπει να προσθέσουμε μία σταθερά. Το 'intercept_scaling' με default τιμή ίση με 1 είναι χρήσιμο να την δηλώσουμε όταν το fit_intercept είναι ίσο με True. Τότε το x γίνεται [x, intercept_scaling] και έτσι προστίθεται ένα χαρακτηριστικό στο διάνυσμα με τιμή ίση με το intercept_scaling. Η παράμετρος 'class_weight' με default τιμή None μπορεί να πάρει τιμές dict ή balanced και καθορίζει το βάρος που σχετίζεται με τις κλάσης.

Στην συνέχεια, η παράμετρος 'random_state' παίρνει τιμές integer και κάνει ανάμιξη στα δεδομένα. Η παράμετρος 'solver' χρησιμοποιείται για την βελτιστοποίηση του προβλήματος και μπορεί να πάρει τιμές 'newton-cg', 'lbfgs', "liblinear", "sag", 'saga' όπου η liblinear τιμή είναι καλύτερη λύση για μικρά σύνολα δεδομένων ενώ οι άλλες είναι πιο γρήγορες για μεγάλα σύνολα δεδομένων. Η 'max_iter' παίρνει integer τιμές και ορίζει το μέγιστο αριθμό επαναλήψεων. Η παράμετρος 'warm_start' παίρνει boolean τιμές όπου όταν είναι True ξανά χρησιμοποιεί τη λύση της προηγούμενης κλήσης ως αρχικοποίηση στην επόμενη ενώ όταν είναι False απλά διαγράφει την προηγούμενη. Τέλος, η παράμετρος 'n_jobs' με default τιμή None παίρνει integer τιμές και δηλώνει τον αριθμό των πυρήνων της CPU που θα χρησιμοποιηθούν και η 'l1_ratio' είναι μία παράμετρος ανάμιξης του Elastic-net που χρησιμοποιείται στο penalty και παίρνει μόνο float τιμές με την default τιμή να είναι ίση με None.

Έτσι, κάποιος μπορεί να καλέσει αυτόν τον αλγόριθμο με την παρακάτω εντολή στην python κάνοντας χρήση της παραπάνω βιβλιοθήκης:

« `from sklearn.linear_model import LogisticRegression` »

2.4. Decision Tree (CART)

Η εκμάθηση δέντρων αποφάσεων είναι μία από τις τεχνικές μοντελοποίησης που χρησιμοποιούνται τόσο στην στατιστική όσο και στην εξόρυξη δεδομένων καθώς και στην μηχανική μάθηση. Η συγκεκριμένη τεχνική είναι από τις πιο παλιές τεχνικές στον χώρο της μηχανικής μάθησης. Αυτή που θα χρησιμεύσει στην δικιά μας έρευνα και θα μας βοηθήσει να βγάλουμε τα αποτελέσματα είναι το decision tree το οποίο ονομάζεται Classification and Regression Tree (C.A.R.T) (Breiman, Friedman, Olshen, & Stone, 1984).

Η διαφορά του Classification tree με το Regression tree είναι ότι το πρώτο μας βοηθάει να ταξινομήσουμε τα δεδομένα μας, ενώ το δεύτερο μας προβλέπει ένα αληθινό αριθμό όπως παραδείγματος χάριν, μπορεί να προβλέψει το εισόδημα ενός ατόμου. Στην δικιά μας έρευνα αυτό που θα χρειαστούμε είναι το Classification tree.

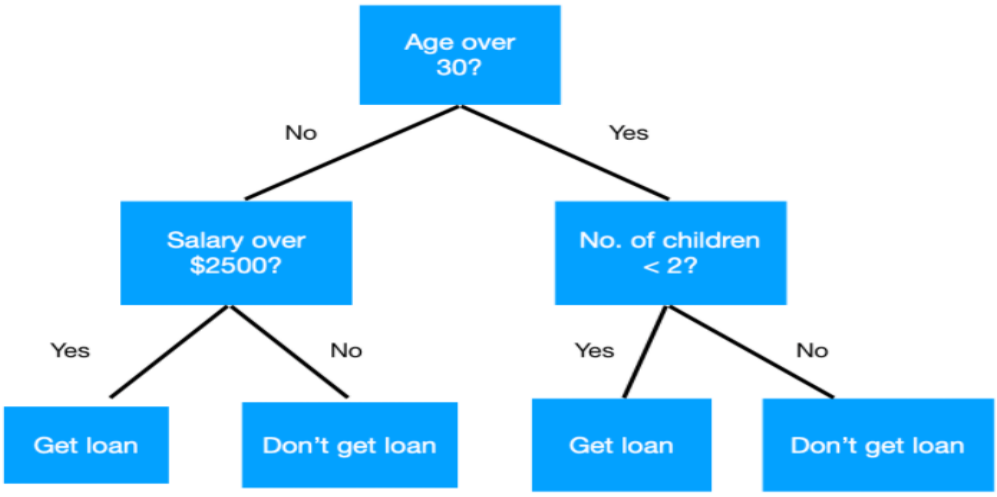
Ας υποθέσουμε ότι μας δίνεται ένα training σύνολο διανυσμάτων S = {s1… sl} ταξινομημένο σε ένα σύνολο $s_i \in R^n$ και ένα διάνυσμα ετικέτας y. Σε κάθε κόμβο N, για κάθε χαρακτηριστικό j και με δεδομένο ένα όριο t_N , χωρίζουμε τα δεδομένα σε Q_{right} ({j, t_N}) και Q_{left} ({j, t_N}) όπου Q_{right} ({j, t_N}) = (x, y) | $x_j > t_N$ και Q_{left} ({j, t_N}) = $Q - Q_{right}$.

Η ποιότητα όπου γίνεται ο διαχωρισμός ονομάζεται ακαθαρσία και μετριέται χρησιμοποιώντας μια συνάρτηση H (). Τα μέτρα ποιότητας που χρησιμοποιούνται είναι συνήθως τα μέτρα entropy ή ο δείκτης Gini. Η επιλογή της λειτουργίας μέτρησης εξαρτάται από τον τρόπο επίλυσης του προβλήματος. Είτε Classification είτε Regression.

Στην συνέχεια, υπολογίζουμε τον δείκτη Gini για τον κόμβο N και χωρίζουμε τα δεδομένα με βάση το χαρακτηριστικό j:

$$G(N,\{j,t_N\}) = \frac{n_{left}}{N_N} H(Q_{left}(\{j,t_N\})) + \frac{n_{right}}{N_N} H(Q_{right}(\{j,t_N\}))$$

Διαχωρίζουμε τα δεδομένα μας στη δυνατότητα που ελαχιστοποιεί το $G\,(N, \{j, t_N\})$ και συνεχίζουμε την ίδια διαδικασία αντίστοιχα για Q_{left} και Q_{right} έως ότου δεν υπάρχουν άλλα κριτήρια διαχωρισμού των δεδομένων.



Εικόνα 4: Παράδειγμα ενός decision tree

Ανακτήθηκε από: https://www.codershood.info/2019/01/14/naive-bayes-classifier-using-python-with-example/

Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου με βάση όσα ειπώθηκαν προηγουμένως:

Πίνακας 4: Ψευδοκώδικας Decision Tree (Cart)

```
Decision Tree (Cart)
d = 0
Note(0), Node(1) = 1, Node(2) = 0
While endtree < 1
            if
Node(2^d − 1) + Node(2^d) + … + Node(2^(d+1) − 2) = 2 - 2^(d+1)
                              endtree = 1
            else
                     do i = 2^d − 1, 2^d, …, 2^(d+1) − 2
                              if Node(i) > -1
                                        Split tree
                              else
                                        Node(2i + 1) = -1
                                        Node(2i + 2) = -1
                              end if
```

```
                    end do
            end if
d = d + 1
end while
```

Η γενική μορφή του αλγορίθμου στην sklearn βιβλιοθήκη η οποία χρησιμοποιήθηκε στην δικιά μας έρευνα στην γλώσσα Python είναι της κλάσης:

class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

Παρακάτω αναλύονται οι παράμετροι του αλγορίθμου. Ο συγκεκριμένος αλγόριθμος ξεκινάει δηλώνοντας τον δείκτη 'criterion' αν είναι Gini ή Entropy. Αυτός ο δείκτης είναι μία συνάρτηση η οποία μετράει την ποιότητα του split που θα γίνει στα δεδομένα. Το Gini χρησιμοποιείται για την Gini ακαθαρσία των δεδομένων και το Entropy για την απόκτηση πληροφοριών αυτών. Δεύτερη παράμετρος στην συνέχεια είναι η 'splitter' η οποία δηλώνει την επιλογή του τρόπου διαχωρισμού κάθε κόμβου. Μπορεί να πάρει δύο τιμές, best και random όπου η μία επιλέγει το καλύτερο split και η άλλη το καλύτερο τυχαίο split αντίστοιχα. Η παράμετρος 'max_depth' μπορεί να πάρει integer τιμές και ορίζει το βάθος που θα έχει το δέντρο. Εάν δεν δηλωθεί καθόλου τότε το δέντρο επεκτείνεται μέχρι τα φύλλα να είναι κενά και να μην μπορούν να παρθούν άλλες αποφάσεις.

Στην συνέχεια, η παράμετρος 'min_samples_split' δηλώνεται και αυτήν με μία τιμή integer και δείχνει τον ελάχιστο αριθμό δειγμάτων που χρειάζονται ώστε να γίνει διάσπαση στον εσωτερικό κόμβο όπως και από την άλλη η παράμετρος 'min_samples_leaf' όπου δηλώνει τον ελάχιστο αριθμό δειγμάτων που πρέπει να βρίσκονται σε έναν κόμβο φύλλων. Η 'max_features' δηλώνει τον μέγιστο αριθμό χαρακτηριστικών όταν κάποιος ψάχνει να βρει το καλύτερο split και μπορεί να πάρει τιμές όπως integer, float, sqrt, log2 και auto. Από την άλλη η παράμετρος 'random_state' παίρνει integer τιμές και μπορεί να ελέγξει την τυχαιότητα του εκτιμητή. Η παράμετρος 'max_leaf_nodes' μπορεί να δηλώσει τους κόμβους. Σε περίπτωση που δεν δηλωθεί κανένας τότε ο αριθμός των κόμβων των

φύλλων είναι απεριόριστος, ενώ αν δηλωθεί διαλέγονται οι καλύτεροι κόμβοι σε αυτούς ώστε να μειώσουν την ακαθαρσία που υπάρχει στα φύλλα.

Σχετικά με την 'min_impurity_decrease' παράμετρο η οποία δηλώνει μία τιμή από την οποία εξαρτάται αν ένας κόμβος θα γίνει split και αυτό θα εξαρτηθεί από το αν ο διαχωρισμός θα προκαλέσει μείωση της τιμής της ακαθαρσίας μεγαλύτερη ή ίση με αυτήν την τιμή. Η παράμετρος δέχεται μόνο float τιμές. Η παράμετρος 'min_impurity_split' δέχεται και αυτήν μόνο float τιμές και ορίζει αν η ανάπτυξη του δέντρου πρέπει να σταματήσει πρόωρα. Τέλος, άλλες δύο παραμέτρους που μπορεί να πάρει ο συγκεκριμένος αλγόριθμος είναι η 'class_weight' και η 'ccp_aplha' όπου η πρώτη δηλώνει το βάρος που θα έχει μία κλάση και η δεύτερη με μία τιμή float δηλώνει την παράμετρο πολυπλοκότητας που χρησιμοποιείται για κλάδεμα του ελαχίστου κόστους. Αν η τιμή είναι η προκαθορισμένη δεν πραγματοποιείται κανένα κλάδεμα.

Έτσι, και αυτόν όπως και τους προηγούμενους αλγορίθμους μπορεί κάποιος να τον καλέσει με την παρακάτω εντολή στην python κάνοντας χρήση της παραπάνω βιβλιοθήκης:

« from sklearn.tree import DecisionTreeClassifier »

3. Συσταδοποίηση (Clustering)

Σε αυτήν την ενότητα θα μελετηθεί ένας αλγόριθμος Συσταδοποίησης (Clustering) που χρησιμοποιήθηκε στην συγκεκριμένη έρευνα και μας βοήθησε στην καλύτερη ανάλυση κάποιων δεδομένων. Η συσταδοποίηση βοηθάει στο να χωριστούν τα δεδομένα μεταξύ τους σε κλάσεις, όπου στην κάθε κλάση τα αντικείμενα θα έχουν κάποιες ομοιότητες μεταξύ τους και θα την κάνει να ξεχωρίζει από τα υπόλοιπα σύνολα. Μία μέθοδος συσταδοποίησης είναι καλή αν οι συστάδες που παράγει είναι καλής ποιότητας, δηλαδή έχουν σημαντική ομοιότητα μέσα στην συστάδα και σημαντικά μικρή ομοιότητα ανάμεσα στις συστάδες.

Η συσταδοποίηση μπορεί να γίνει με τους εξής τρόπους:

- **Αποκλειστικά ή μη αποκλειστικά**, όπου τα αντικείμενα μπορούν να ανήκουν σε περισσότερος από μία ομάδες.
- **Μερική ή ολική**, σε περίπτωση που ο ομαδοποίηση γίνει σε ορισμένα από τα δεδομένα.
- **Ετερογενή ή ομογενή,** όπου η ομαδοποίηση μπορεί να γίνει από διαφορετικά μεγέθη και σχήματα.

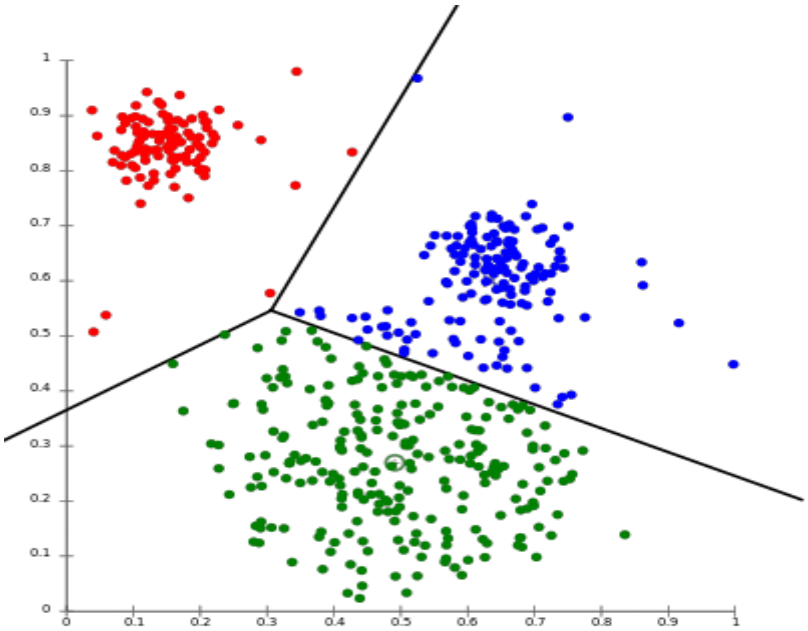
3.1. K-Means model

Ο αλγόριθμος που χρησιμοποιήθηκε στην συγκεκριμένη έρευνα και μας βοήθησε στην διαδικασία της συσταδοποίησης είναι ο K-Means αλγόριθμος ο οποίος για πρώτη φορά προτάθηκε από τον Stuart Lloyd το 1957. Ο K-Means αλγόριθμος προσπαθεί και στοχεύει να χωρίσει ένα σύνολο n παρατηρήσεων $(x_1, x_2, \dots x_n)$ σε k σύνολα S = $\{S_1, S_2, \dots, S_k\}$ όπου το k <= n έτσι ώστε να ελαχιστοποιηθεί το άθροισμα των τετραγώνων εντός τις συστάδας:

$$arg_s min \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 = arg_s min \sum_{i=1}^{k} |S_i| \; Var \; S_i$$

Όπου το μ_i είναι ο μέσος όρος των σημείων στο S_i. Έτσι, έχουμε:

$$arg_s min \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{x,y \in S_i} ||x - y||^2$$



Εικόνα 5: Παράδειγμα ενός K-means μοντέλου

Ανακτήθηκε από: https://medium.com/analytics-vidhya/comparative-study-of-the-clustering-algorithms-54d1ed9ea732

Μόλις χωριστούν τα αντικείμενα σε συστάδες, η τελική μορφή είναι αυτή που προκύπτει στο παραπάνω σχήμα. Ωστόσο, κάτι που πρέπει να τονιστεί σχετικά με τον συγκεκριμένο αλγόριθμο, είναι ότι τα αρχικά κεντρικά σημεία συνήθως επιλέγονται τυχαία και στη συνέχεια, πως η εγγύτητα των σημείων υπολογίζεται με βάση κάποια απόσταση που εξαρτάται από το είδος των σημείων.

Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου με βάση όσα ειπώθηκαν προηγουμένως:

Πίνακας 5: Ψευδοκώδικας K-Means model

K-Means model
Input:
D = {t_1, t_2, … T_n} // Set of elements

```
     K    // Number of clusters
Output:
     K   // Set of clusters
K-Means algorithm:
     Assign initial values for m_1, m_2, … m_k
     repeat
       Assign each item t_i to the clusters which has the
closest mean;
       Calculate new mean for each cluster;
     until convergence criteria is met;
```

Η γενική μορφή του αλγορίθμου στην sklearn βιβλιοθήκη η οποία χρησιμοποιήθηκε στην δικιά μας έρευνα στην γλώσσα Python είναι της μορφής:

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='kme
ans++', n_init=10, max_iter=300, tol=0.0001, precompute_d
istances='deprecated', verbose=0, random_state=None, copy
_x=True, n_jobs='deprecated', algorithm='auto')
```

Αναλύοντας τώρα τις παραπάνω παραμέτρους βλέπουμε πως και αυτός μπορεί να χαρακτηριστεί από πολλές διαφορετικές τιμές, όπως και οι αλγόριθμοι που αναλύθηκαν στην ενότητα της κατηγοριοποίησης. Με την 'n_clusters' παράμετρο δηλώνεται ο αριθμός των συστάδων και ταυτόχρονα και ο αριθμός των centroids που θα δημιουργηθούν. Η παράμετρος 'init' μπορεί να δεχθεί δύο τιμές, είτε την τιμή k-means++ είτε την random. Η πρώτη επιλέγει τα κέντρα με τέτοιο έξυπνο τρόπο ώστε να επιταχύνει τη διαδικασία, ενώ η δεύτερη παίρνει τυχαία τα κέντρα με βάση τα αρχικά centroids.

Στην συνέχεια, υπάρχει η παράμετρος 'n_init' στην οποία ορίζεται ο χρόνος εκτέλεσης του αλγορίθμου k-means και δηλώνεται με μία μεταβλητή integer. Με την 'max_iter' κάποιος μπορεί να δηλώσει τον αριθμό των μέγιστων επαναλήψεων για μία μόνο εκτέλεση του αλγορίθμου. Η παράμετρος 'precompute_distances' προϋπολογίζει της αποστάσεις για ταχύτερη διαδικασία, χρησιμοποιώντας ωστόσο περισσότερη μνήμη. Η 'random_state' ορίζει την δημιουργία τυχαίων αριθμών για την αρχικοποίηση των centroids. Τέλος, μέσω της παραμέτρου 'n_jobs'

ορίζεται με μία integer τιμή ο αριθμός των OpenMp threads που θα βοηθήσουν στον υπολογισμό και η παράμετρος `'algorithm'` που δηλώνει τον αλγόριθμο που θα χρησιμοποιηθεί. Με την τιμή `full` χρησιμοποιείται ο κλασσικός αλγόριθμος ενώ με την τιμή `elkan` χρησιμοποιείται μία παραλλαγή του, η οποία είναι πιο αποτελεσματική σε δεδομένα με ομάδες οι οποίες είναι καλά χωρισμένες, χρησιμοποιώντας ωστόσο περισσότερη μνήμη.

Έτσι, μπορεί κάποιος να τον καλέσει με την παρακάτω εντολή στην python κάνοντας χρήση της παραπάνω βιβλιοθήκης:

« `from sklearn.cluster import KMeans` »

4. Data Sets

4.1. Introduction

Σε αυτήν την ενότητα θα αναλυθούν τα σύνολα δεδομένων (data sets) που χρησιμοποιήθηκαν στην συγκεκριμένη διπλωματική εργασία από το UC Irvine Machine Learning Repository[1]. Θα παρουσιαστεί αναλυτικά το κάθε data set ξεχωριστά ως προς την δομή που ακολουθεί και τις διάφορες τιμές που χρησιμοποιεί. Μετέπειτα, θα γίνει ανάλυση αυτών των data sets στα διάφορα papers που αναφέρθηκαν με σκοπό να παρουσιαστούν τα αποτελέσματα που δημοσιεύθηκαν από τους ίδιους, οι ακρίβειές τους, τα ποσοστά σφάλματος και οι χρόνοι τους, αναλόγως με τους αλγορίθμους που χρησιμοποιήθηκαν σε κάθε έρευνα. Όλα αυτά θα γίνουν ώστε να μπορέσουμε να τα συγκρίνουμε στην επόμενη ενότητα με τα αποτελέσματα των δικών μας αλγορίθμων για να οδηγηθούμε στο συμπέρασμα αν ξεπεράσαμε τις συγκεκριμένες ακρίβειες και αν υπάρχει χώρος βελτίωσης των συγκεκριμένων data sets.

Τα data sets που χρησιμοποιήθηκαν είναι:

1. Internet Firewall Data Set
2. Teaching Assistant Evaluation Data set
3. Car Evaluation Data Set
4. Electrical Grid Stability Simulated Data Set

4.2. Internet Firewall Data Set

Το συγκεκριμένο Data Set συλλέχθηκε από τα αρχεία κίνησης στο Διαδίκτυο του τείχους προστασίας του πανεπιστημίου Firat. Αποτελείται από 65.532 εγγραφές οι οποίες περιγράφουν την κίνηση που υπάρχει στο Διαδίκτυο. Το τείχος προστασίας είναι πολύ σημαντικό για την σωστή και ασφαλή επικοινωνία ανάμεσα στο δίκτυο επικοινωνίας, έχοντας σκοπό να επιτρέψει ή να απαγορεύσει την είσοδο αυτών των δεδομένων. Το data set χωρίζει αυτές τις εγγραφές σε 12 χαρακτηριστικά τα οποία αναλύονται παρακάτω:

[1] https://archive.ics.uci.edu/ml/datasets.php

Πίνακας 6: Χαρακτηριστικά του data set (IF)

Χαρακτηριστικά	Περιγραφή
Source Port	Θύρα του client
Destination Port	Θύρα προορισμού
NAT Source Port	Θύρα πηγής της διεύθυνσης του δικτύου
NAT Destination Port	Θύρα προορισμού της διεύθυνσης δικτύου
Elapsed Time (sec)	Χρόνος που πέρασε για την ροή
Bytes	Συνολικά Bytes
Bytes Sent	Bytes που στάλθηκαν
Bytes Received	Bytes που ελήφθησαν
Packets	Συνολικά πακέτα
Pkts_sent	Πακέτα που στάλθηκαν
Pkts_received	Πακέτα που ελήφθησαν
Action	Ενέργεια που έγινε από το τοίχος

4.2.1. Ανάλυση χαρακτηριστικών του Data Set

Βλέποντας τον παραπάνω πίνακα παρατηρείται ότι οι στήλες του data set είναι 12, εκ των οποίων οι 11 είναι οι ανεξάρτητες μεταβλητές στο data set. Εν αντιθέσει με αυτές, η δωδέκατη στήλη “action” είναι η εξαρτημένη μας μεταβλητή. Για τα διαστήματα των τιμών χρησιμοποιήθηκαν οι παρακάτω εντολές στην python μέσω της pandas βιβλιοθήκης για να βρεθεί η ελάχιστη και η μέγιστη τιμή:

```
«(dataset[dataset['column_name']==dataset['column
name'].min()])»
```

```
«(dataset[dataset['column_name']==dataset['column
name'].max()]) »
```

Έτσι, όσον αφορά τις ανεξάρτητες μεταβλητές θα αναλύσουμε τις τιμές που περιέχονται στις εγγραφές.

- Source Port: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 65534].
- Destination Port: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 65535].

- Nat Source Port: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 65535].
- Nat Destination Port: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 65535].
- Elapsed Time: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 10824].
- Bytes: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [60, 1269359015].
- Bytes Sent: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [60, 948477220].
- Bytes Received: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 320881795].
- Packets: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 1036116].
- Packets Sect: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [1, 747520].
- Packets Received: Είναι αριθμητική μεταβλητή, με συνεχείς αριθμούς ανάμεσα στο διάστημα [0, 327208].

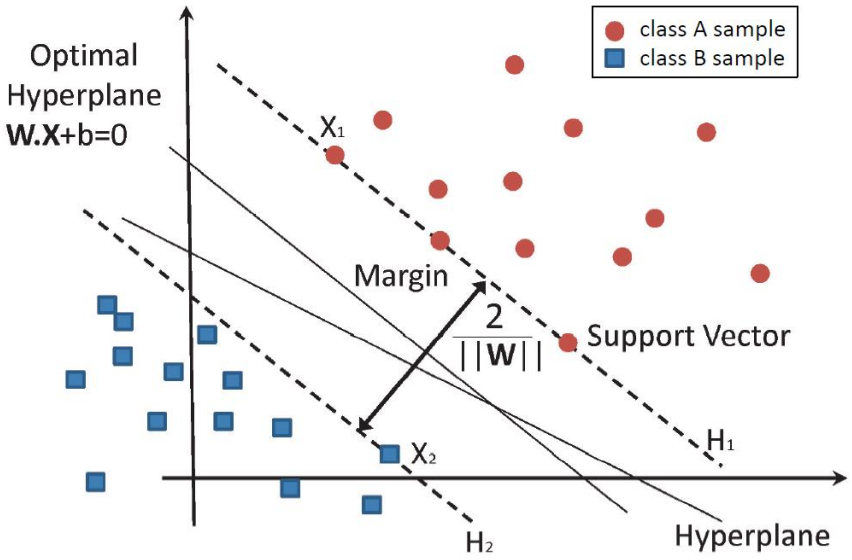
Όσον αφορά την εξαρτημένη μεταβλητή του data set, την στήλη “action”, είναι διακριτή κατηγορική μεταβλητή και μας δείχνει την ενέργεια που κάνει στο τέλος το τείχος προστασίας και παίρνει τέσσερις τιμές. (Allow, Deny, Drop, Reset-Both). Ο παρακάτω πίνακας δείχνει αναλυτικά τις ενέργειες:

Πίνακας 7: Ενέργειες της εξαρτημένης μεταβλητής

Ενέργειες	Περιγραφή
Allow	Επιτρέπει την κίνηση στο Δίκτυο
Deny	Αποτρέπει την κίνηση στο Δίκτυο
Drop	Ρίχνει την κίνηση και αντικαθιστά την προεπιλεγμένη ενέργεια άρνησης της εφαρμογής. Η επαναφορά του TCP δεν αποστέλλεται στην εφαρμογή
Reset-Both	Στέλνει την επαναφορά του TCP στις συσκευές και του client και του server

4.2.2. Εφαρμογή του Data Set σε διάφορα άρθρα

Σύμφωνα με τους Fetih Ertam και Mustafa Kaya στο άρθρο τους "Classification of Firewall Log Files with Multiclass Support Vector Machine", όπου χρησιμοποίησαν το συγκεκριμένο data set, είναι πολύ σημαντική η ανάλυση των συγκεκριμένων αρχείων καταγραφής στις διάφορες συσκευές του τείχους προστασίας. Οι ίδιοι στην έρευνά τους για την Κατηγοριοποίηση (Classification) των δεδομένων χρησιμοποίησαν τον αλγόριθμο Support Vector Machine. Ο συγκεκριμένος αλγόριθμος πραγματοποιεί ταξινόμηση υπολογίζοντας το hyperplane (μια απλή γραμμή σε δύο διαστάσεις) που διαφοροποιεί καλύτερα τις κλάσεις του συνόλου δεδομένων. Από έναν μεγάλο αριθμό hyperplanes, το SVM υπολογίζει το βέλτιστο χρησιμοποιώντας ένα υποσύνολο εκπαιδευτικών δειγμάτων, τους φορείς υποστήριξης (support vectors).



Εικόνα 6: Παράδειγμα ενός Support Vector Machine μοντέλου

Ανακτήθηκε από: https://www.researchgate.net/figure/Classification-of-data-by-support-vector-machine-SVM_fig8_304611323

Στην έρευνά τους, την ακρίβεια των αποτελεσμάτων την μέτρησαν μέσω της τιμής F_1 Score, της Precision τιμής και της τιμής Recal. Η precision είναι ο αριθμός των πραγματικών θετικών (TP) διά των αριθμό των πραγματικών θετικών, συν τον αριθμό των ψευδών θετικών (FP). Η recall, η οποία ορίζει πόσες πληροφορίες ανακτώνται, υπολογίζεται από τον αριθμό των πραγματικών θετικών διά τον αριθμό των πραγματικών θετικών συν των πραγματικών αρνητικών. Τέλος, η τιμή F_1 Score, είναι ο μέσος όρος των τιμών της ακρίβειας (P) και ανάκλησης (R) :

$$P = \frac{TP}{TP+FP}, \qquad R = \frac{TP}{TP+FN}, \qquad F_1 \text{ Score} = 2\frac{P*R}{P+R}$$

Η μέτρηση αυτών των τριών τιμών έγινε ανάμεσα σε τέσσερις διαφορετικούς αλγορίθμους σχετικά με τον κατηγοριοποιητή Support Vector Machine και τα αποτελέσματα φαίνονται στον παρακάτω πίνακα:

Πίνακας 8: Αποτελέσματα SVM

Αλγόριθμος	F_1 Score	Precision	Recall
SVM Linear	75,4	67,5	85,3
SVM Polynomial	53,6	61,8	47,4
SVM RBF	76,4	63,0	97,1
SVM Sigmoid	74,8	60,3	98,5

4.3. Teaching Assistant Evaluation Data Set

Το συγκεκριμένο data set αποτελείται από αξιολογήσεις της διδακτικής απόδοσης σε πέντε εξάμηνα στο Τμήμα Στατιστικής του Πανεπιστημίου του Wisconsin-Madison. Αποτελείται από 151 εγγραφές (είναι 151 teaching assistant assignments), οι οποίες είναι χωρισμένες σε έξι χαρακτηριστικά. Τα εξάμηνα αποτελούνται από τρία κανονικά και δύο καλοκαιρινά εξάμηνα. Αυτό που έχει ως σκοπό το συγκεκριμένο data set είναι να αναδείξει ανάμεσα σε τρεις τιμές την βαθμολογία και το χαρακτηριστικό κάθε τάξης:

Πίνακας 9: Χαρακτηριστικά του data set (TAE)

Χαρακτηριστικά	Περιγραφή
Native English	Αν είναι μητρική γλώσσα τα Αγγλικά
Course Instructor	Ο εκπαιδευτής του μαθήματος
Course	Το μάθημα
Semester	Το εξάμηνο
Class Size	Ο αριθμός ατόμων της τάξης
Class Attribute	Χαρακτηριστικό της τάξης

4.3.1. Ανάλυση χαρακτηριστικών του Data Set

Αναλύοντας τον πίνακα 9 οι πέντε πρώτες στήλες είναι οι ανεξάρτητες μεταβλητές και η τελευταία η “class attribute” η εξαρτημένη. Παρακάτω θα δούμε τις τιμές που δέχεται η κάθε στήλη:

- Native English: Παίρνει binary τιμές 1 και 2. 1 όταν τα Αγγλικά είναι μητρική γλώσσα και 2 για όταν δεν είναι.
- Course Instructor: Είναι αριθμητική κατηγορική μεταβλητή, αποτελείτε από 25 κατηγορίες, με διάστημα τιμών [1,25]
- Course: Είναι αριθμητική κατηγορική μεταβλητή, αποτελείται από 26 κατηγορίες, με διάστημα τιμών [1,26]
- Semester: Παίρνει binary τιμές 1 και 2. 1 όταν είναι καλοκαιρινό εξάμηνο και 2 όταν είναι κανονικό
- Class Size: Είναι μία αριθμητική μεταβλητή με συνεχείς αριθμούς οι οποίοι ανήκουν στο διάστημα [3, 66]

Όσον αφορά την εξαρτημένη μεταβλητή του data set, την στήλη “class attribute” είναι διακριτή κατηγορική μεταβλητή και μας δείχνει το χαρακτηριστικό κάθε τάξης, η οποία μπορεί να είναι low, medium ή high. Στο συγκεκριμένο data set δεν λείπει καμία τιμή ώστε να χρειαστεί ενέργειες για να την συμπληρώσουμε.

4.3.2. Εφαρμογή του Data Set σε διάφορα άρθρα

Οι Wei-Yin Loh και Yu-Shan Shih με βάση το άρθρο τους “A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms ”, χρησιμοποίησαν 33 αλγορίθμους πάνω σε 32 data sets ένα εκ των οποίων είναι και το “Teaching Assistant Evaluation Data Set”, όπου εργαστήκαμε και εμείς στην δική μας διπλωματική εργασία.

Χρησιμοποίησαν 22 decision tree αλγόριθμους, 9 στατιστικούς (statistical) και 2 νευρικά δίκτυα (neural network). Επιγραμματικά οι αλγόριθμοι στους οποίους δούλεψαν είναι:

- Δέντρα Αποφάσεων (Trees and rules): CART, S-plus, Tree, C4.5, FACT QUEST, IND, OC1, LMDT, CALS, T1

- Στατιστικοί αλγόριθμοι (statistical): LDA, QDA, NN, LOG, FDA, PDA, MDA, POL
- Νευρικά Δίκτυα (Neural Network): LVQ, RBF

Τα αποτελέσματά τους τα σύγκριναν με βάση το accuracy, το training time, και τον αριθμό των φύλλων των δέντρων όπου, σε περιπτώσεις που είχαν ίδιο είδος δοκιμών ανάμεσα σε δύο δέντρα και ίδια ακρίβεια διάλεγαν συνήθως αυτόν με τα λιγότερα φύλλα. Στην συγκεκριμένη έρευνα το accuracy το υπολόγιζαν με βάση το ποσοστό σφάλματος. Για λόγους σύγκρισης και με τα αποτελέσματα της διπλωματικής εργασίας, το ποσοστό σφάλματος υπολογίζεται με την συνάρτηση:

Error rate = 1 – Accuracy

Έτσι, μετά από ανάλυση των αποτελεσμάτων τους και χρησιμοποιώντας στην ανάλυση του data set 10-fold cross validation το ελάχιστο και μέγιστο ποσοστό σφάλματος (με αυτό που θα γίνει και σύγκριση πάνω στα δικά μας αποτελέσματα) ήταν:

	Error rates
Ελάχιστο όριο	0,33
Μέγιστο όριο	0,66

4.4. Car Evaluation Data Set

Το Car Evaluation data set είναι μία βάση δεδομένων ώστε με κάποια χαρακτηριστικά να μπορεί να αξιολογηθεί ένα αυτοκίνητο. Η βάση δεδομένων αξιολόγησης αυτοκινήτου προήλθε αρχικά από ένα απλό μοντέλο απόφασης το οποίο αναπτύχθηκε από τους M. Bohanec και V. Rajkivic και αξιολογεί τα αυτοκίνητα ως εξής:

CAR	Αποδοχή αυτοκινήτου
.PRICE	Συνολική τιμή
. . buying	Τιμή αγοράς
. . maint	Τιμή συντήρησης
.TECH	Τεχνικά χαρακτηριστικά

. . COMFORT	Άνεση
. . . doors	Αριθμός από πόρτες
. . . persons	Χωρητικότητα ατόμων
. . . lug_boot	Μέγεθος αποσκευών
. . safety	Ασφάλεια του αυτοκινήτου

Το συγκεκριμένο data set αποτελείται από 1728 εγγραφές και ο αριθμός των χαρακτηριστικών που χρησιμοποιήθηκαν εδώ είναι εφτά. Από το σύνολο δεδομένων δεν έχουμε κάποιο στοιχείο που να λείπει οπότε δεν χρειάζεται κάποια ενέργεια συμπλήρωσης δεδομένων και ως δομή ακολουθεί τα χαρακτηριστικά του παραπάνω πίνακα:

Πίνακας 10: Χαρακτηριστικά του data set (CE)

Χαρακτηριστικά	Περιγραφή
Buying	Τιμή αγοράς
Maint	Τιμή συντήρησης
Doors	Αριθμός από πόρτες
Persons	Χωρητικότητα ατόμων
Lug boot	Μέγεθος αποσκευών
Safety	Ασφάλεια του αυτοκινήτου
Class	Η κλάση του αυτοκινήτου

4.4.1 Ανάλυση χαρακτηριστικών του Data Set

Αναλύοντας τον πίνακα 10 οι έξι πρώτες στήλες είναι οι ανεξάρτητες μεταβλητές και η τελευταία η “Class” η εξαρτημένη. Παρακάτω θα δούμε τις τιμές που δέχεται η κάθε στήλη:

- Buying: Είναι διακριτή κατηγορική μεταβλητή και παίρνει τιμές vhigh, high, med, low.
- Maint: Είναι διακριτή κατηγορική μεταβλητή και παίρνει τιμές vhigh, high, med, low.
- Doors: Είναι κατηγορική μεταβλητή και παίρνει τιμές 2,3,4 ή 5more
- Persons: Είναι κατηγορική μεταβλητή και παίρνει τιμές 2,4 ή more

- Lug boot: Είναι διακριτή κατηγορική μεταβλητή και παίρνει τιμές small, med, big.
- Safety: Είναι διακριτή κατηγορική μεταβλητή και παίρνει τιμές low, med, high.

Όσον αφορά την εξαρτημένη μεταβλητή του data set, την στήλη "class" είναι διακριτή κατηγορική μεταβλητή και μας δείχνει την κλάση του αυτοκινήτου η οποία μπορεί να είναι unacc, acc, good, v-good. Παρακάτω φαίνονται και πόσες εγγραφές ανήκουν σε κάθε κλάση στο αρχικό data set:

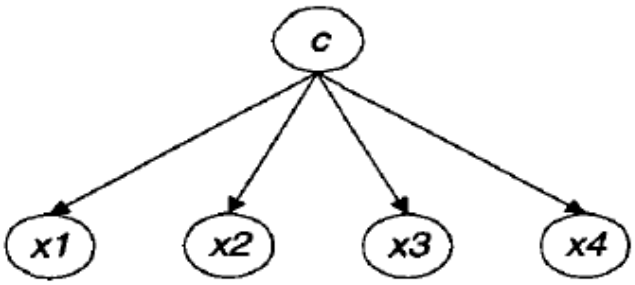
Class	**N**	**N[%]**
Unacc	1210	70,023%
Acc	384	22,222%
Good	69	3,993%
V-good	65	3,762%

4.4.2. Εφαρμογή του Data Set σε διάφορα άρθρα

Οι Jie Cheng και Russell Greiner κάνουν ανάλυση στο άρθρο τους "Comparing Bayesian Network Classifiers" το συγκεκριμένο data set. Ο αλγόριθμος πάνω στον οποίο προσομοίωσαν το σύνολο δεδομένων είναι ο Naïve-Bayes. Χρησιμοποίησαν τέσσερις διαφορετικούς τύπους κατηγοριοποίησης Bayesian Network (BN):

- Naïve-Bayes
- Tree Augmented Naïve-Bayes (TAN)
- BN Augmented Naïve-Bayes (BAN)
- General BNs (GBN)

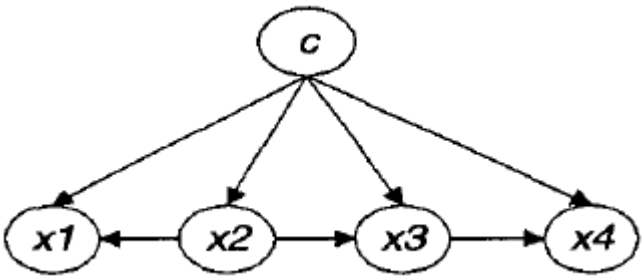
Ο κλασσικός Naïve-Bayes έχει μία απλή δομή, η οποία έχει τον classification κόμβο ως κεντρικό κόμβο όλων των άλλων. Σαν αλγόριθμος είναι εύκολος να κατασκευαστεί, διότι η δομή του δίνεται εκ των προτέρων και είναι πολύ αποτελεσματική. Όλα τα χαρακτηριστικά του είναι ανεξάρτητα μεταξύ τους. Η διαδικασία εκμάθησης είναι να μένει ο classification κόμβος ως κεντρικός γονέας όλων, να μαθαίνει τις παραμέτρους και τέλος να κάνει output τον Naïve-Bayes BN.



Εικόνα 7: Απλή μορφή Naïve-Bayes αλγόριθμου

Ανακτήθηκε από: https://arxiv.org/ftp/arxiv/papers/1301/1301.6684.pdf

Ο Tree Augmented Naïve-Bayes (TAN) ακολουθεί μία άλλη δομή. Εδώ έχουμε ένα σύνολο X = $\{x_1, \ldots x_n, c\}$ όπου το X είναι ένα σύνολο κόμβων και c ο classification κόμβος. Πρώτα, μαθαίνει μία δομή δέντρου πάνω στο X \ (c) και μετά προσθέτει έναν σύνδεσμο από κάθε classification κόμβο προς κάθε κόμβο χαρακτηριστικών. Εδώ, η διαδικασία εκμάθησης είναι να παίρνει το training set και το X \ (c) ως input, να καλεί τον αλγόριθμο Chow-Liu[2], να προσθέτει τον c ως γονέα κάθε x, να μαθαίνει της παραμέτρους και να κάνει output τον TAN.

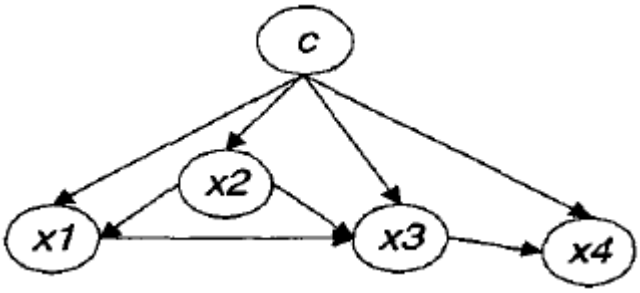


Εικόνα 8: Απλή μορφή TAN αλγόριθμου

Ανακτήθηκε από: https://arxiv.org/ftp/arxiv/papers/1301/1301.6684.pdf

Από την άλλη, ο BN Augmented Naïve-Bayes (BAN) είναι μία επέκταση του κατηγοριοποιητή TAN αφήνοντας τα χαρακτηριστικά να σχηματίσουν ένα αυθαίρετο γράφημα. Η μόνη διαφορά με τον TAN είναι ότι αντί για τον tree learning αλγόριθμο αυτός καλεί έναν unrestricted BN-Iearning αλγόριθμο τον τροποποιημένο CBLl.

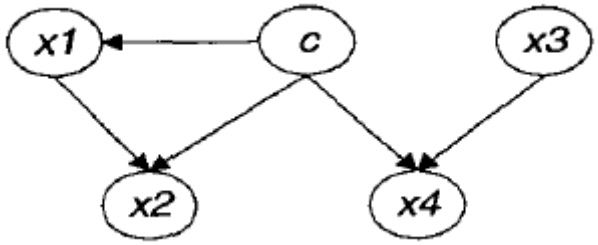
[2] tree learning αλγόριθμος



Εικόνα 9: Απλή μορφή BAN αλγόριθμου

Ανακτήθηκε από: https://arxiv.org/ftp/arxiv/papers/1301/1301.6684.pdf

Τέλος, ο General BNs (GBN) αντιμετωπίζει τους classification κόμβους ως κοινούς. Η διαδικασία εκ μάθησης εδώ είναι να παίρνει σαν input το training set και το set χαρακτηριστικών, να καλεί τον μη τροποποιημένο CBLl αλγόριθμο, να βρίσκει το Markov blanket του classification κόμβου, να διαγράφει όλους τους κόμβους που είναι έξω από το Markov blanket, να μαθαίνει τις παραμέτρους και να κάνει output τον GBN.



Εικόνα 10: Απλή μορφή GBN αλγόριθμου

Ανακτήθηκε από: https://arxiv.org/ftp/arxiv/papers/1301/1301.6684.pdf

Έτσι, αφού αναλύθηκαν οι αλγόριθμοι πάνω στους οποίους εργάστηκαν παρακάτω παρουσιάζονται και τα αποτελέσματα που είχαν σε κάθε αλγόριθμο ξεχωριστά:

	Accuracy			
Dataset/Algorithm	**Naïve-Byes**	**BAN**	**TAN**	**GBN**
Car Evaluation	86,58	94,04	94,10	86,11

Την καλύτερη ακρίβεια, βλέποντας τον πίνακα, την είχε ο TAN αλγόριθμος με μία μικρή απόκλιση από τον BAN. Ο GBN και Naïve-Bayes είχαν πολύ μικρότερη ακρίβεια από τους άλλους δύο ωστόσο παρέμειναν και αυτοί πολύ υψηλοί.

4.5. Electrical Grid Stability Simulated Data Set

Στο συγκεκριμένο data set αναλύεται η σταθερότητα της ηλεκτρικής ενέργειας. Για να είναι σταθερά τα ηλεκτρικά δίκτυα πρέπει να υπάρχει ισορροπία μεταξύ προσφοράς και ζήτησης ηλεκτρικής ενέργειας. Το data set αποτελείται από 10000 εγγραφές και έχει ως σκοπό μέσα από τους χρόνους αντίδρασης του κάθε συμμετέχοντα, τις αξίες ηλεκτρικής ενέργειας, την ισχύ κατανάλωσης και τις τιμές της παραγωγής της να δείξει την σταθερότητα του συστήματος. Αν είναι σταθερό ή ασταθές. Παρακάτω παρουσιάζεται ο πίνακας με τα χαρακτηριστικά του data set:

Πίνακας 11: Χαρακτηριστικά του data set (EGSS)

Χαρακτηριστικά	Περιγραφή
tau1	Χρόνος απόκρισης γεννήτριας 1
tau2	Χρόνος απόκρισης γεννήτριας 2
tau3	Χρόνος απόκρισης γεννήτριας 3
tau4	Χρόνος απόκρισης γεννήτριας 4
p1	Είναι η ονομαστική ισχύς δικτύου 1
p2	Είναι η ονομαστική ισχύς δικτύου 2
p3	Είναι η ονομαστική ισχύς δικτύου 3
p4	Είναι η ονομαστική ισχύς δικτύου 4
g1	Συντελεστής γ και είναι ανάλογος της ελαστικότητας της τιμής 1 (δείχνει βαθμό συσχέτισης μεταξύ των δεδομένων
g2	Συντελεστής γ και είναι ανάλογος της ελαστικότητας της τιμής 2 (δείχνει βαθμό συσχέτισης μεταξύ των δεδομένων
g3	Συντελεστής γ και είναι ανάλογος της ελαστικότητας της τιμής 3 (δείχνει βαθμό συσχέτισης μεταξύ των δεδομένων
g4	Συντελεστής γ και είναι ανάλογος της ελαστικότητας της τιμής 4 (δείχνει βαθμό συσχέτισης μεταξύ των δεδομένων
Stab	Μέγιστο πραγματικό μέρος της ρίζας εξίσωσης
stabf	Ετικέτα σταθερότητας συστήματος

Το σύστημα αποτελείται από τέσσερα διαφορετικά δίκτυα και οι προσαρμογές των τιμών γίνονται πάνω σε αυτά.

4.5.1. Ανάλυση χαρακτηριστικών του Data Set

Αναλύοντας τον πίνακα 11 οι δεκατρείς στήλες είναι οι ανεξάρτητες μεταβλητές και η τελευταία η "stabf" είναι η εξαρτημένη. Παρακάτω θα δούμε τις τιμές που δέχεται η κάθε στήλη:

- Tau(x): Είναι αριθμητική μεταβλητή και παίρνει τιμές ανάμεσα στο διάστημα [0.5, 10]
- P(x): Είναι αριθμητική μεταβλητή και παίρνει τιμές ανάμεσα στο διάστημα [-0.5, -2]$s^{\wedge}$-2. Αν είναι αρνητική τότε υπάρχει κατανάλωση ισχύς ενώ αν είναι θετική παραγωγή ισχύς.
- G(x): Είναι αριθμητική μεταβλητή και παίρνει τιμές ανάμεσα στο διάστημα [0,05, 1]$s^{\wedge}$-1
- stab: Είναι αριθμητική μεταβλητή και παίρνει τιμές ανάμεσα στο διάστημα (-1, 1). Αν είναι θετικό το σύστημα είναι γραμμικά ασταθές αλλίως είναι σταθερό.

Όσον αφορά την εξαρτημένη μεταβλητή του data set, την στήλη "stabf" είναι διακριτή κατηγορική μεταβλητή και μας δείχνει την σταθερότητα του συστήματος και παίρνει δύο τιμές, stable και unstable για σταθερό και ασταθές σύστημα αντίστοιχα.

4.5.2. Εφαρμογή του Data Set σε διάφορα άρθρα

Οι Vadim Arzamasov, Klemens Bohm και Patrick Jochem με βάση το άρθρο τους "Towards Concise Models of Grid Stability", χρησιμοποίησαν το συγκεκριμένο data set για να αναλύσουν την σταθερότητα της ηλεκτρικής ενέργειας στο Δίκτυο μέσα από τη συμπεριφορά των συμμετεχόντων και αν η απόκρισή τους στις αλλαγές τιμών αποσταθεροποιεί αυτό το σύστημα.

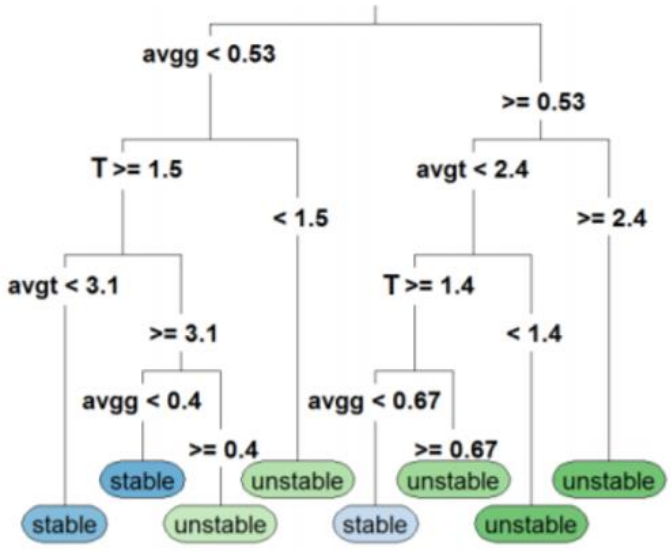
Οι τύποι σταθερότητας ενός συστήματος με βάση τους συγγραφείς του άρθρου είναι οι:

1) Σταθερότητα έναντι μεμονωμένων διαταραχών: είναι η ικανότητα του συστήματος να φτάσει σε ισορροπία μετά από κάποια διαταραχή.

2) Σταθερότητα λεκάνης: ορίζονται κάποιες πιθανές διαταραχές και προσομοιώνεται το σύστημα για ένα σύνολο διαταραχών που έχουν επιλεγεί τυχαία από αυτές.
3) Ανάλυση τοπικής σταθερότητας: εδώ διερευνάται η δυναμική σταθερότητα γύρω από τη λειτουργία σταθερής κατάστασης του δικτύου.

Στην συγκεκριμένη έρευνα χρησιμοποιούν τον τρίτο τύπο, δηλαδή την ανάλυση τοπικής σταθερότητας, γιατί δεν παρέχονται επιπλέον βαθμοί ελευθερίας σε ένα ήδη πολύπλοκο σύστημα. Αυτό το σύστημα συνίσταται στην εύρεση ριζών της εξίσωσης, η οποία είναι της μορφής: $\det A = 0$, όπου A είναι ένας πίνακας $2N \times 2N$ που προέρχεται από εξισώσεις κίνησης. Η εξίσωση έχει απεριόριστες λύσεις, αλλά μόνο ένας πεπερασμένος αριθμός λύσεων μπορεί να έχει ένα θετικό πραγματικό μέρος όπου και καθορίζουν την αστάθεια του συστήματος. Η λύση αυτών των ριζών βρίσκεται από την επίλυση ενός προβλήματος αριθμητικής βελτιστοποίησης.

Από εκεί και πέρα για την έρευνά τους δούλεψαν πάνω στον αλγόριθμο CART, χρησιμοποιώντας τον σχεδιασμό γεμίσματος χώρου LHS. Οι διαδρομές των φύλλων του δέντρου που παράγονται από τον αλγόριθμο καθορίζουν τις σταθερές ή ασταθείς περιοχές. Η ακρίβεια που είχαν στα πειράματά τους ήταν 80%.



Εικόνα 11: Decision tree για το σύστημά τους

Ανακτήθηκε από: https://ieeexplore.ieee.org/abstract/document/8587498

5. Αποτελέσματα Μεθόδων Μηχανικής Μάθησης

Σε αυτήν την ενότητα θα αναλυθούν οι δικοί μας μέθοδοι που χρησιμοποιήθηκαν πάνω στα data sets για τα οποία έγινε λεπτομερής περιγραφή στο τέταρτο κεφάλαιο. Οι αλγόριθμοι για το classification που χρησιμοποιήθηκαν σε κάθε data set είναι τέσσερις όπως αναφέρθηκε και στην αρχή (K-NN, Naïve-Byes, Logistic Regression, CART) και ένας αλγόριθμος (K-Means model) που χρησιμοποιήθηκε μόνο στο data set " Teaching Assistant Evaluation Data Set" όπου και χρειάστηκε η διαδικασία συσταδοποίησης σε μία κλάση. Επιπλέον, θα αναλυθούν οι παράμετροι που δόθηκαν σε κάθε αλγόριθμο, τα αποτελέσματα που εμφάνισαν στο κάθε data set και η σύγκρισή τους με τα αποτελέσματα των υπόλοιπων ερευνών που έγιναν και αναλύθηκαν κι αυτά στο τέταρτο κεφάλαιο.

Σε όλα τα data set η έρευνα έγινε με την γλώσσα προγραμματισμού `Python` και χρήση της βιβλιοθήκης `sklearn`, `numpy` και της βιβλιοθήκης `pandas`. Τέλος, σε όλες τις δοκιμές χρησιμοποιήθηκε *10*-`Fold cross validation` και σε όλους τους αλγορίθμους ένας σταθερός, λόγω της εργασίας, `random_state` ώστε τα αποτελέσματα να είναι για όλους τα ίδια αν και οι αυξομειώσεις ήταν πολύ μικρές και χωρίς αυτήν την παράμετρο.

5.1. Προ επεξεργασία στο Internet Firewall Data Set

Όπως αναφέρθηκε προηγουμένως, στο συγκεκριμένο data set στόχος είναι ένα τείχος προστασίας το οποίο μπορεί να επιτρέψει, ή όχι, ένα σύνολο δεδομένων το οποίο αποτελεί μία κίνηση δεδομένων στο Διαδίκτυο. Το data set αποτελείται από δώδεκα στήλες εκ των οποίων η τέταρτη στήλη είναι η εξαρτημένη μεταβλητή και ορίζει αν επιτρέπεται ή όχι η είσοδος.

Αν και τα δεδομένα στις υπόλοιπες έντεκα στήλες είναι αριθμητικά όταν γινόταν ενέργεια διαβάσματος του data set, το σύστημα τα αναγνώριζε ως string δεδομένα. Έτσι, σαν πρώτη ενέργεια, αφού διαβάζαμε το αρχείο excel ήταν να μετατρέψουμε αυτά τα δεδομένα σε αριθμητικά και συγκεκριμένα σε float αριθμούς. Αυτό γιατί, θα χρειαζόντουσαν σε επόμενο βήμα όπου το data set θα χρειαστεί να γίνει feature scaling. Η παραπάνω ενέργεια πραγματοποιήθηκε μέσω της εντολής:

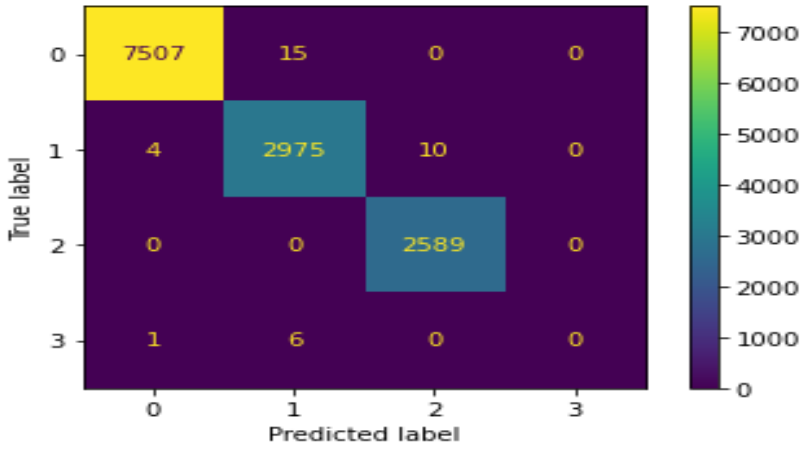
- `dataset = pd.read_csv('file_name.csv')`, για ανάγνωση του αρχείου δεδομένων
- `dataset['Column_name']=pd.to_numeric(dataset['Column_name'],downcast='float')`, για μετατροπή σε float αριθμητικά δεδομένα

Επόμενο βήμα είναι η μετατροπή της εξαρτημένης μεταβλητής, η οποία παίρνει τέσσερις διαφορετικές string τιμές. Έτσι, βασικό στοιχείο ήταν να μετατρέψουμε αυτές τις τιμές σε αριθμητικές, ώστε να παίρνει τιμές από 0 μέχρι 3 αντίστοιχα για καλύτερη κατηγοριοποίηση των δεδομένων. Στην συνέχεια κάναμε split το data set σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής. Εδώ, ο χωρισμός έγινε με `'test_size = 0.20'` όπου για δοκιμή πήγαινε το 20% των δεδομένων μας και το 80% για εκπαίδευση.

Στο συγκεκριμένο data set χρειάστηκε να γίνει feature scaling στα δεδομένα ώστε να χωριστούν σε μικρά διαστήματα με τιμές ανάμεσα στο διάστημα (-4,4) γιατί οι τιμές που περιείχαν ήταν σε πολύ μεγάλα άσχετα μεταξύ τους διαστήματα.

Μετέπειτα, γίνεται χρήση του κάθε αλγορίθμου ώστε ο καθένας με τους δικούς του παραμέτρους να εκπαιδεύσει τα δεδομένα και να μας δώσει τα εξής confusion matrix και τις εξής ακρίβειες:

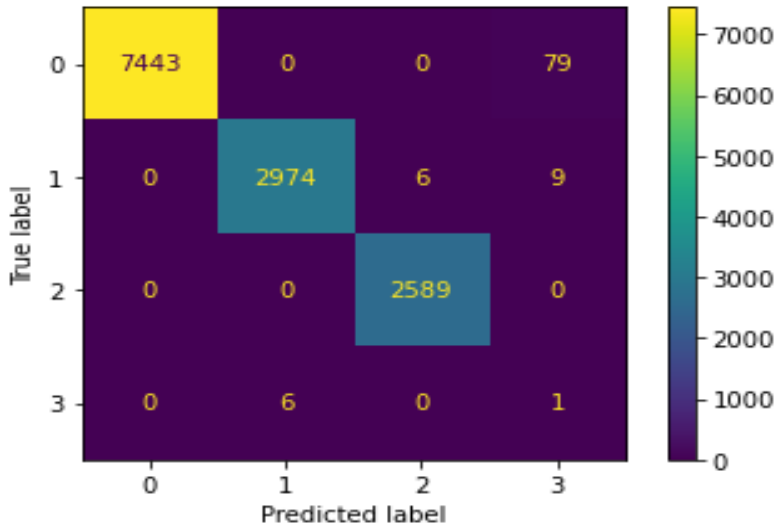
1) Για τον K-NN αλγόριθμο χρησιμοποιήθηκε η `'n_neighbors'` μεταβλητή να είναι ίση με 5 η οποία μας δίνει τον αριθμό των γειτόνων που θα χρησιμοποιηθεί και η παράμετρος `'p'` για μετρική `Minkowski` με τιμή 2 για να έχουμε Ευκλείδειες αποστάσεις. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Βλέποντας το confusion matrix του αλγορίθμου παρατηρούμε ότι οι προβλέψεις ήταν σχεδόν απόλυτα σωστές και έχουμε ένα πίνακα 4x4 αφού οι τιμές της εξαρτημένης μεταβλητής ήταν τέσσερις. Από τον πίνακα φαίνεται πως για την τιμή allow από τα 7522 δεδομένα που στάλθηκαν για εκπαίδευση, τα 7507 τα μάντεψε σωστά. Για τιμή deny, από τα 2989 μάντεψε σωστά τα 2975 ενώ για την τιμή drop, ήταν σωστά και τα 2589. Για την τιμή reset_both, και τα 7 δεδομένα τα μάντεψε λάθος.

Η ακρίβεια που εμφάνισε το data set σε αυτόν τον αλγόριθμο ήταν 99,67% με standard deviation 0% και το F1 score 75% και precision 99,7%.

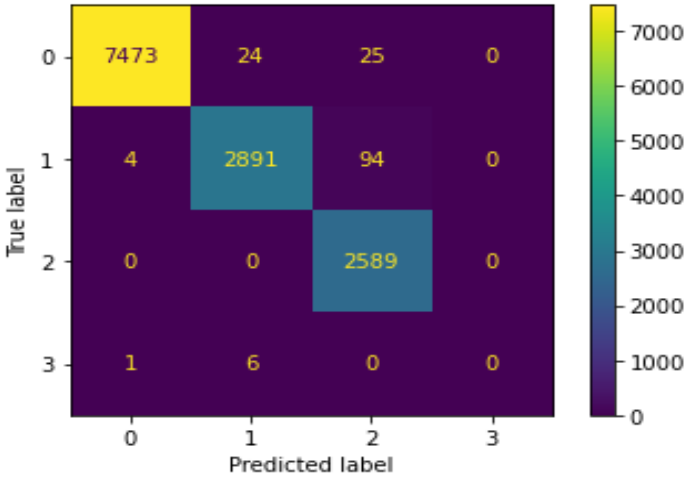
2) Για τον Naïve-Bayes χρησιμοποιήθηκε ο απλός GaussianNB αλγόριθμος. . Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Βλέποντας το confusion matrix του αλγορίθμου παρατηρούμε ότι οι προβλέψεις για την τιμή allow, από τα 7522 δεδομένα που στάλθηκαν για εκπαίδευση, τα 7443 τα μάντεψε σωστά ενώ για τιμή deny, από τα 2989 μάντεψε σωστά τα 2974. Για την τιμή drop, από τα 2589 σωστά ήταν και τα 2589 και, τέλος, για την τιμή reset_both, από τα 7 δεδομένα το 1 μόνο μάντεψε σωστά.

Η ακρίβεια που εμφάνισε το data set σε αυτόν τον αλγόριθμο ήταν 99,1% με standard deviation 0% και το F1 score 75,2% και precision 75,1%.

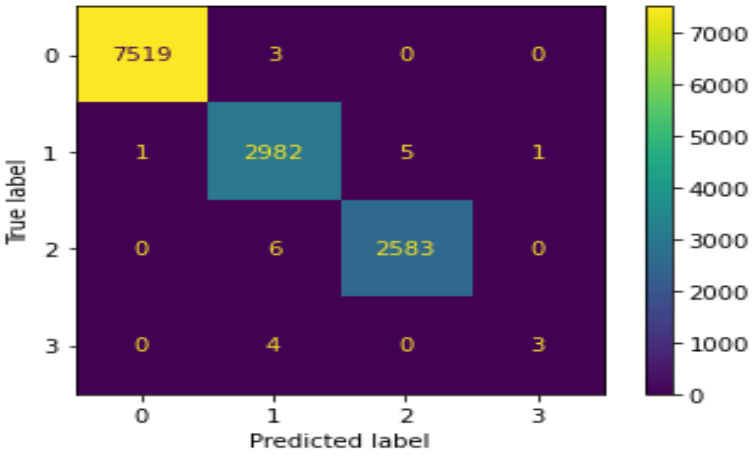
3) Για τον Logistic Regression η μόνη παράμετρος που χρησιμοποιήθηκε είναι η `max_iter=1000` για τον μέγιστο αριθμό επαναλήψεων. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Βλέποντας το confusion matrix του αλγορίθμου παρατηρούμε ότι οι προβλέψεις για την τιμή allow, από τα 7522 δεδομένα που στάλθηκαν για εκπαίδευση, τα 7473 τα μάντεψε σωστά. Σχετικά με την τιμή deny, από τα 2989 μάντεψε σωστά τα 2891, ενώ για την τιμή, drop από τα 2589 ο αλγόριθμος τα μάντεψε όλα σωστά. Δε συνέβη το ίδιο με την τιμή reset_both, όπου και τα 7 δεδομένα που στάλθηκαν τα μάντεψε λάθος.

Η ακρίβεια που εμφάνισε το data set σε αυτόν τον αλγόριθμο ήταν 98,5% με standard deviation 0% και το F1 score 73,8% και precision 99,7%.

4) Για τον CART χρησιμοποιήθηκε η παράμετρος `'criterion'` να είναι ίση με την τιμή `Entropy`. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Βλέποντας το confusion matrix του αλγορίθμου παρατηρούμε τις προβλέψεις για την τιμή allow, όπου από τα 7522 δεδομένα που στάλθηκαν για εκπαίδευση, τα 7519 τα μάντεψε σωστά. Επιπλέον, για τιμή deny, από τα 2999 μάντεψε σωστά τα 2982 ενώ για την τιμή drop, από τα 2589 σωστά ήταν τα 2583. Τέλος, για την τιμή reset_both από τα 7 δεδομένα μάντεψε σωστά μόνο τα 3.

Η ακρίβεια που εμφάνισε το data set σε αυτόν τον αλγόριθμο ήταν 99,7% με standard deviation 0% και το F1 score 88,4% και precision 93,5%.

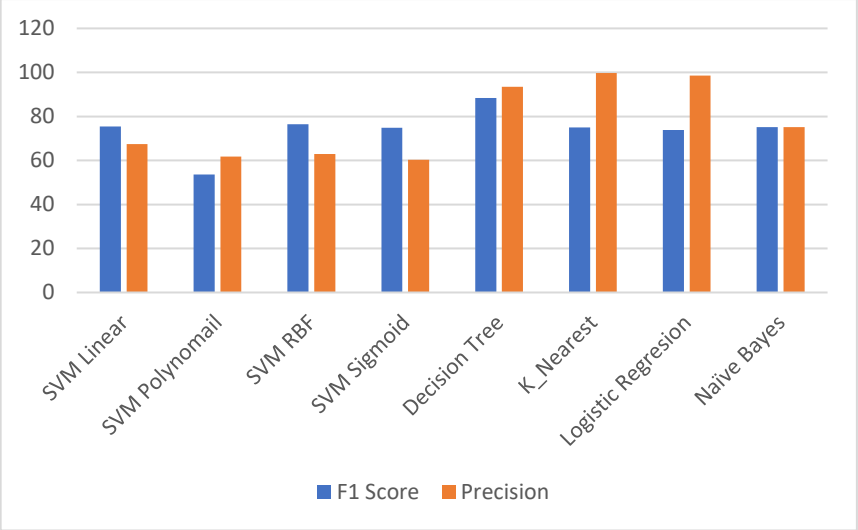
5.1.1. Σύγκριση Αποτελεσμάτων του IF Data Set

Στο κομμάτι αυτό θα γίνει μία σύγκριση των δικών μας αποτελεσμάτων με τα αποτελέσματα των Fetih Ertam και Mustafa Kaya που παρουσιάστηκαν στο άρθρο τους “Classification of Firewall Log Files with Multiclass Support Vector Machine” και αναφέρθηκαν στην τέταρτη ενότητα.

Πίνακας 12: Συνολικά Αποτελέσματα IF Data Set

ΑΛΓΌΡΙΘΜΟΙ	F1 SCORE	PRECISION	ACCURACY
SVM Linear	75,4	67,5	
SVM Polynomial	53,6	61,8	
SVM RBF	76,4	63,0	
SVM Sigmoid	74,8	60,3	
K-Nearest	75,0	99,7	99,67
Naïve Bayes	75,2	75, 1	99,1
Logistic Regresion	73,8	98,6	98,5
Decision Tree	88,4	93,5	99,7

Παρατηρώντας τον πίνακα 12 με τα αποτελέσματα βλέπουμε πως το accuracy και στου τέσσερις αλγορίθμους που χρησιμοποιήσαμε στην έρευνά μας είναι πολύ υψηλό. Καταλαβαίνουμε πως έγινε καλή εκπαίδευση στα δεδομένα και μεγάλο ρόλο ίσως έπαιξε πως το μέγεθος του συνόλου δεδομένων που ήταν τεράστιο.



Εικόνα 12: Σύγκριση αποτελεσμάτων

Όπως αναφέρθηκε και στο τέταρτο κεφάλαιο οι μετρήσεις των Fetih Ertam και Mustafa Kaya έγιναν πάνω στην ακρίβεια του F1 Score και του Precision. Συγκρίνοντας τώρα όλους τους αλγορίθμους καταλαβαίνουμε πως ο πιο αποδοτικός ήταν ο decision tree αλγόριθμος όπου το F1 Score συγκριτικά με τους υπόλοιπους είχε αρκετή διαφορά της τάξης του 13%. Ωστόσο, παρατηρούμε επίσης ότι στην τιμή του precision και οι τέσσερις αλγόριθμοι, στους οποίους δουλέψαμε εμείς, ήταν πιο αποδοτικοί εν αντιθέσει με τον SVM αλγόριθμο.

5.2. Προ επεξεργασία στο Teaching Assistant Evaluation Data Set

Το συγκεκριμένο data set αποτελείται από αξιολογήσεις της διδακτικής απόδοσης σε πέντε εξάμηνα στο Τμήμα Στατιστικής ενός Πανεπιστημίου. Αποτελείται από 151 εγγραφές όπως προ αναφέρθηκε, οι οποίες είναι χωρισμένες σε έξι χαρακτηριστικά με την έκτη στήλη του data set να είναι η εξαρτημένη μεταβλητή. Μετράει το χαρακτηριστικό της κάθε τάξης και την ταξινομεί σε μία κλίμακα με τιμές low, medium ή high.

Όλα τα δεδομένα στις στήλες είναι αριθμητικά και είναι ομοιόμορφα κατηγοριοποιημένα, εκτός από την πέμπτη στήλη η οποία μετράει το μέγεθος της τάξης. Εδώ τα δεδομένα είναι μεν αριθμητικά, αλλά οι αριθμοί μεταξύ τους δεν έχουν κάποια ομοιομορφία αφού η μεταβλητή παίρνει τιμές ανάμεσα στο διάστημα [3,66]. Εδώ, σαν πρώτη ενέργεια, ήταν να χωρίσουμε την συγκεκριμένη στήλη (Class size) σε

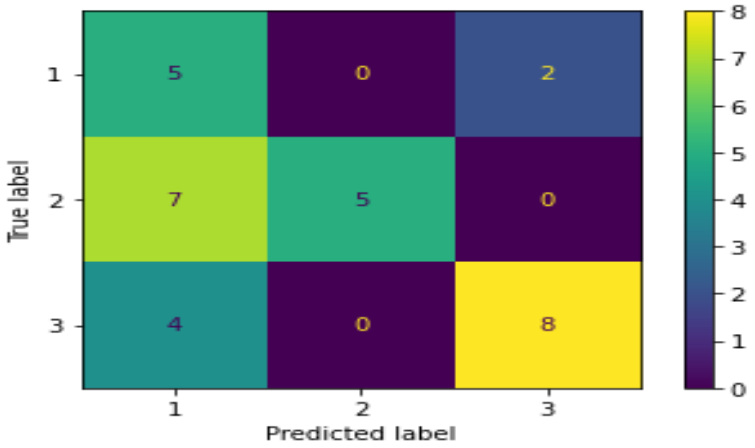
έξι κλάσεις ώστε να έχουν καλύτερη ομοιομορφία και η κατηγοριοποίηση στον αλγόριθμο να δώσει καλύτερα αποτελέσματα:

- Τιμές που ήταν στο διάστημα <=11 πήραν τιμή 1
- Τιμές που ήταν στο διάστημα (11, 22] πήραν τιμή 2
- Τιμές που ήταν στο διάστημα (22, 33] πήραν τιμή 3
- Τιμές που ήταν στο διάστημα (33, 44] πήραν τιμή 4
- Τιμές που ήταν στο διάστημα (44, 55] πήραν τιμή 5
- Τιμές που ήταν στο διάστημα (55, 66] πήραν τιμή 6

Επόμενο βήμα είναι η μετατροπή της εξαρτημένης μεταβλητής, η οποία παίρνει τρεις διαφορετικές string τιμές. Έτσι, βασικό στοιχείο ήταν να μετατρέψουμε αυτές τις τιμές σε αριθμητικές ώστε να παίρνει τιμές από 0 μέχρι 2 αντίστοιχα για καλύτερη κατηγοριοποίηση των δεδομένων. Στην συνέχεια, κάναμε split το data set σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής. Εδώ ο χωρισμός έγινε με `'test_size = 0.20'` όπου για δοκιμή πήγαινε το 20% των δεδομένων μας και το 80% για εκπαίδευση.

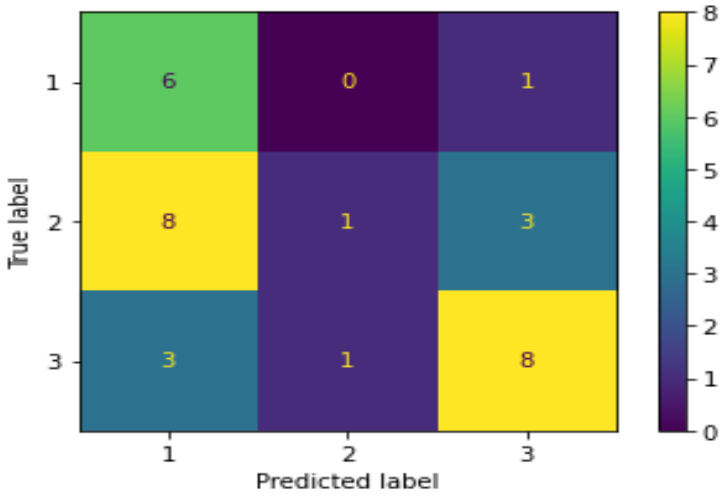
Στην συνέχεια, γίνεται χρήση του κάθε αλγορίθμου ώστε να πάρουμε πάλι τα confusion matrix και τις ακρίβειες αντίστοιχα. Στο συγκεκριμένο data set θα μετρήσουμε την ακρίβεια και ως ποσοστό σφάλματος για να μπορέσει να γίνει σύγκριση με τα αποτελέσματα τις τέταρτης ενότητας:

1) Για τον K-NN αλγόριθμο και εδώ χρησιμοποιήθηκε η `'n_neighbors'` μεταβλητή να είναι ίση με 5 η οποία μας δίνει τον αριθμό των γειτόνων που θα χρησιμοποιηθεί και η παράμετρος `'p'` για μετρική `Minkowski` με τιμή 2 για να έχουμε Ευκλείδειες αποστάσεις. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



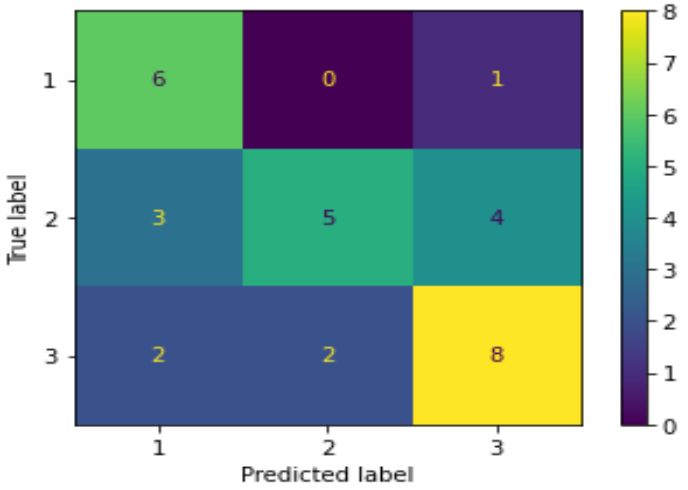
Τα αποτελέσματα εδώ δεν ήταν όπως θα τα περιμέναμε. Η ακρίβεια ήταν αρκετά χαμηλή και ήταν ίση με 42% με την τιμή του standard deviation στο 0,12% . Το μέσο ποσοστό σφάλματος εδώ ήταν 55%.

2) Για τον Naïve-Bayes χρησιμοποιήθηκε ο απλός GaussianNB αλγόριθμος. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



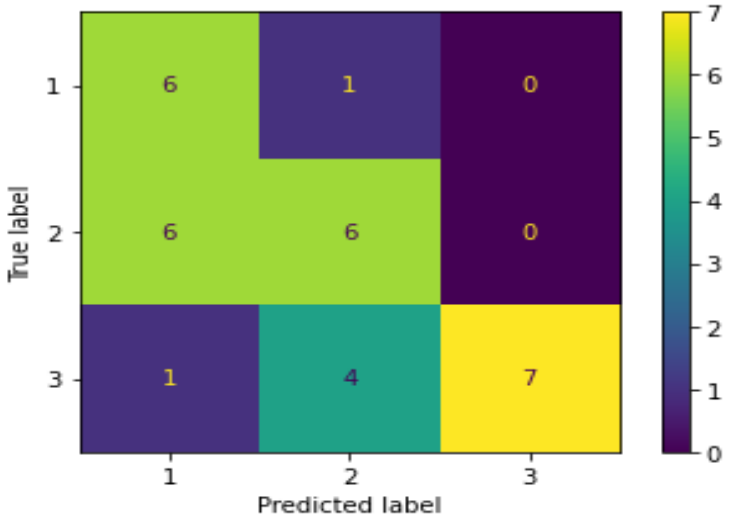
Τα αποτελέσματα κι εδώ δεν ήταν τόσο καλά έχοντας μόνο μία μικρή αύξηση σε σχέση με τον K-NN αλγόριθμο. Η ακρίβεια ήταν ίση με 47,5% με την τιμή του standard deviation στο 0,15% . Το μέσο ποσοστό σφάλματος εδώ ήταν 52,5%.

3) Για τον Logistic Regression η μόνη παράμετρος που χρησιμοποιήθηκε είναι η `max_iter=1000` για τον μέγιστο αριθμό επαναλήψεων. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Τα αποτελέσματα και σε αυτόν τον αλγόριθμο συνέχισαν στο ίδιο επίπεδο. Η ακρίβεια ήταν ίση με 46,67% με την τιμή του standard deviation στο 0,11% . Το μέσο ποσοστό σφάλματος εδώ ήταν 53,5%.

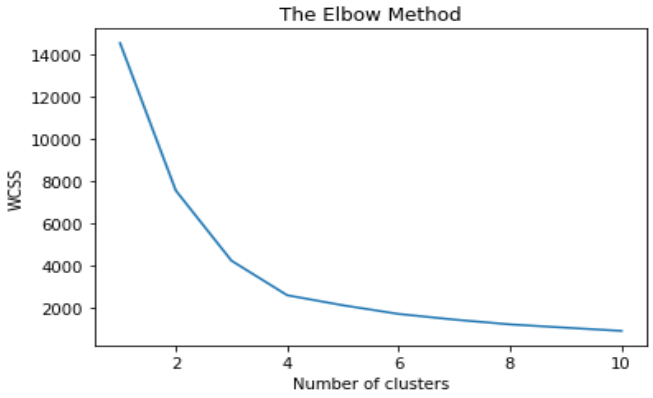
4) Για τον CART χρησιμοποιήθηκε πάλι η παράμετρος `'criterion'` να είναι ίση με την τιμή `Entropy`. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Η ακρίβεια στον CART αλγόριθμου είχε μία αύξηση και ήταν ίση με 58,33% με την τιμή του standard deviation στο 0,16% . Το μέσο ποσοστό σφάλματος εδώ ήταν 53,5%. Και σε αυτό το data set βλέπουμε πως ο CART αλγόριθμος είχε καλύτερα αποτελέσματα σε σχέση με τους άλλους.

Ωστόσο, τα αποτελέσματα δεν ήταν πολύ καλά σε κανέναν αλγόριθμο, γι' αυτό και δοκιμάστηκε μία καινούργια μέθοδος πάνω στο σύνολο δεδομένων. Αντί να κάνουμε εμείς τον διαχωρισμό την μεταβλητής class size σε κλάσεις, χρησιμοποιήσαμε τον αλγόριθμο συσταδοποίησης (clustering) K-Means, προκειμένου να χωρίσει αυτός τα δεδομένα όσο καλύτερα γίνεται. Αυτό που κάναμε είναι να βάλουμε όλα τα δεδομένα εκτός της στήλης class size ώστε να εκπαιδευτούν με τον αλγόριθμο για να μας δοθεί μία καινούργια στήλη new_class_size.

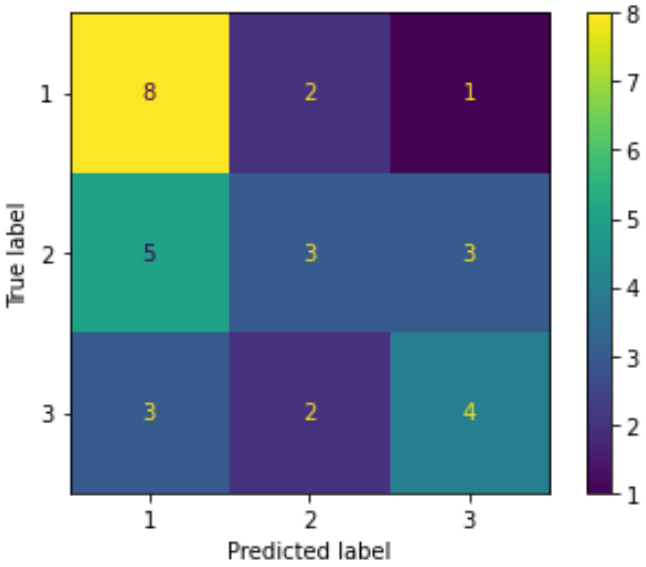
Μέσω της elbow μεθόδου τρέξαμε τα δεδομένα για να βρούμε σε πόσες συστάδες είναι το καλύτερο να τα χωρίσει ο αλγόριθμος και πόσα centroids να έχει:



Εικόνα 13: The Elbow Method

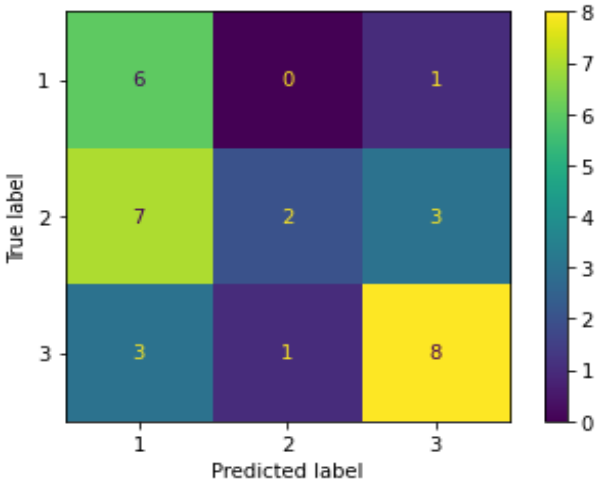
Έτσι, βλέπουμε ότι το καλύτερο αποτέλεσμα θα μας το δώσει αν χρησιμοποιήσουμε 6 συστάδες. Οι παράμετροι που χρησιμοποιήθηκαν στον K-Means αλγόριθμο είναι όπως είπαμε `'n_clusters'` ίσο με 6 και η `'init'` παράμετρος ίση με `'k-means++'` ώστε να επιλέξει τα κέντρα με έξυπνο τρόπο για να επιταχύνει τη διαδικασία. Αφού ολοκληρώθηκε η διαδικασία της συσταδοποίησης χρησιμοποιήθηκαν τα δεδομένα, πλέον με την καινούργια στήλη και στους τέσσερις αλγόριθμους όπως και πριν με τις ίδιες παραμέτρους. Τα αποτελέσματα είναι τα εξής:

1) Για τον K-NN αλγόριθμο έχουμε το confusion matrix:



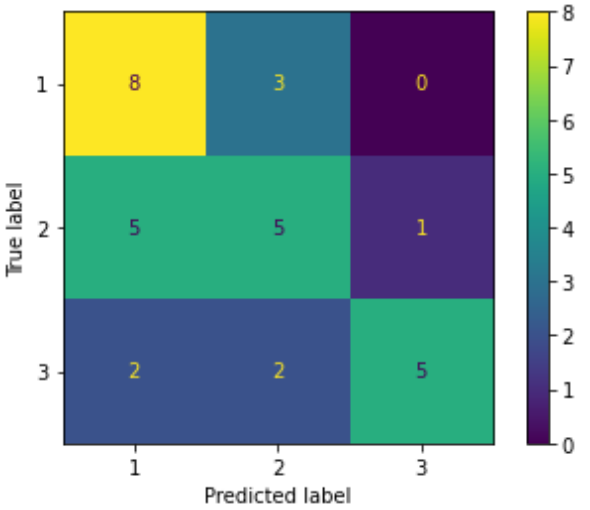
Τα αποτελέσματα ούτε τώρα άλλαξαν ιδιαίτερα. Η ακρίβεια είχε μία πολύ μικρή αύξηση σε σχέση με πριν και ήταν ίση με 45% με την τιμή του standard deviation στο 0,08% . Το μέσο ποσοστό σφάλματος εδώ ήταν 55%.

2) Για τον Naïve-Bayes αλγόριθμο έχουμε το confusion matrix:



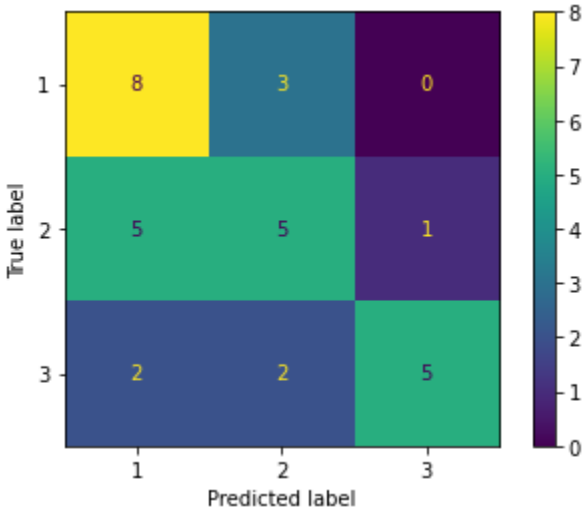
Το ίδιο ισχύει και γι' αυτόν τον αλγόριθμο. Η ακρίβεια αυξήθηκε πολύ λίγο και ήταν ίση με 51,6% με την τιμή του standard deviation στο 0,16% . Το μέσο ποσοστό σφάλματος εδώ ήταν 48,4%.

3) Για τον Logistic Regression αλγόριθμο έχουμε το confusion matrix:



Στο ίδιο μοτίβο κινήθηκε και αυτός ο αλγόριθμος. Η ακρίβεια ήταν ίση με 50% με την τιμή του standard deviation στο 0,13% . Το μέσο ποσοστό σφάλματος εδώ ήταν 50%.

4) Για τον CART αλγόριθμο έχουμε το confusion matrix:



Η ακρίβεια στον CART αλγόριθμου είχε μία μικρή πτώση σε σχέση με την χρήση του data set χωρίς τον K-Means αλγόριθμο και ήταν ίση με 54,17% με την τιμή του standard deviation στο 0,12% . Το μέσο ποσοστό σφάλματος εδώ ήταν 45,83%.

5.2.1. Σύγκριση Αποτελεσμάτων του TAE Data Set

Βλέπουμε πως στο συγκεκριμένο data set ό,τι και να δοκιμάσαμε στους αλγορίθμους δεν άλλαξαν και πολλά. Αυτό ίσως γίνεται γιατί, το σύνολο δεδομένων στο συγκεκριμένο data set, είναι πολύ μικρό και δεν υπάρχει χώρος για καλή εκπαίδευση αυτών. Με την χρήση του K-Means μοντέλου για την συσταδοποίηση των δεδομένων στην αρχή, παρατηρούμε ότι η αύξηση της ακρίβειας ήταν ελάχιστη σε όλους τους αλγορίθμους εκτός από τον CART αλγόριθμο όπου η ακρίβεια του μειώθηκε για λίγο ποσοστό. Παρακάτω, παρουσιάζονται συγκριτικά τα αποτελέσματα και για τα 10 *k*-Fold του κάθε αλγορίθμου για τα Error Rates τους:

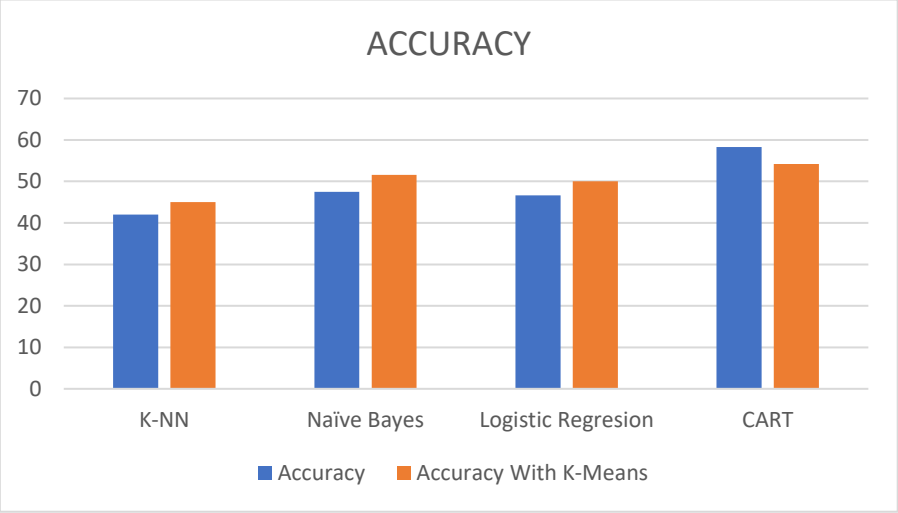
Πίνακας 13: Συνολικά Αποτελέσματα Error Rates για το TAE Data Set

ALG/FOLD	1	2	3	4	5	6	7	8	9	10	MIN	MAX
K-NN	0,41	0,66	0,75	0,33	0,58	0,58	0,50	0,66	0,58	0,41	0,33	0,75
Naïve	0,66	0,66	0,50	0,50	0,66	0,75	0,41	0,41	0,41	0,25	0,25	0,75
Logistic	0,50	0,66	0,50	0,41	0,66	0,66	0,33	0,50	0,58	0,50	0,33	0,66
CART	0,33	0,33	0,66	0,33	0,71	0,45	0,25	0,41	0,41	0,25	0,25	0,66
ΑΡΘΡΟ											0,33	0,66

Πίνακας 14: Συνολικά Αποτελέσματα Error Rates για το TAE Data Set με εφαρμογή K-Means

ALG/FOLD	1	2	3	4	5	6	7	8	9	10	MIN	MAX
K-NN	0,50	0,66	0,75	0,41	0,83	0,58	0,58	0,41	0,33	0,41	0,33	0,83
Naïve	0,58	0,66	0,50	0,41	0,66	0,66	0,41	0,33	0,41	0,16	0,16	0,66
Logistic	0,50	0,41	0,41	0,58	0,66	0,75	0,50	0,41	0,25	0,50	0,25	0,75
CART	0,66	0,41	0,33	0,41	0,58	0,58	0,25	0,50	0,41	0,41	0,25	0,66
ΑΡΘΡΟ											0,33	0,66

Παρατηρώντας τον πίνακα 13 και πίνακα 14 βλέπουμε πως και με τους δύο τρόπους καταφέρνουμε μικρότερα ποσοστά error rates από αυτά που είχαν πετύχει οι Wei-Yin Loh και Yu-Shan Shih με βάση αυτά που παρουσιάστηκαν στο άρθρο τους "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms ", όπου και χρησιμοποίησαν 33 αλγορίθμους. Αν και οι αλγόριθμοι τους ήταν 33, παρουσίασαν μόνο τον ελάχιστο και μέγιστο error rate που κατάφεραν, γι' αυτό και η σύγκριση γίνεται πάνω σε αυτά. Συγκεκριμένα, το μικρότερο ποσοστό error rate στην δικιά μας έρευνα το πετυχαίνει ο naïve bayes αλγόριθμος αν και μεταξύ τους τα αποτελέσματα δεν είχαν και πολλές διαφορές. Παρακάτω παρουσιάζονται και οι ακρίβειες των αλγορίθμων σε γράφημα όπου και σε αυτό το data set τις υψηλότερες τις είχε ο CART αλγόριθμος:



Εικόνα 14: Accuracy for TAE Data Set

5.3. Προ επεξεργασία στο Car Evaluation Data Set

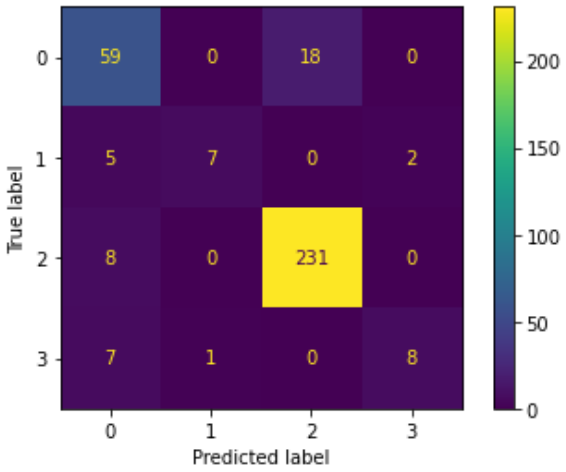
Το Car Evaluation data set, όπως αναφέρθηκε, είναι μία βάση δεδομένων ώστε με κάποια χαρακτηριστικά να μπορεί να αξιολογηθεί ένα αυτοκίνητο. Αποτελείται από 1728 εγγραφές. Οι έξι πρώτες στήλες του data set είναι οι ανεξάρτητες μεταβλητές και η τελευταία 'Class value' είναι η εξαρτημένη μεταβλητή η οποία χαρακτηρίζει και ένα αυτοκίνητο ως unacc, acc, good, v-good.

Εδώ όλες οι ανεξάρτητες μεταβλητές ήταν διακριτές κατηγορικές οπότε σαν πρώτη ενέργεια ήταν να της μετατρέψουμε σε αριθμητικές. Αυτό που κάναμε ήταν να χρησιμοποιήσουμε την `OneHotEncoder` εντολή της `sklearn` βιβλιοθήκης. Στην ουσία, μετατρέπει την κάθε στήλη σε τόσες καινούργιες στήλες όσες και οι ξεχωριστές τιμές της κάθε μεταβλητής. Έτσι, κάθε κατηγορική τιμή την μετατρέπει σε ένα ξεχωριστό διάνυσμα με τιμές 0 και 1. Για παράδειγμα στην πρώτη στήλη που υπάρχουν τέσσερις ξεχωριστές τιμές θα δημιουργηθούν τέσσερις καινούργιες στήλες, δηλαδή την τιμή vhigh την μετέτρεψε σε τιμή τύπου 0.0 |0.0| 0.1| 0.0. Το `OneHotEncoder` δίνει καλύτερο ποσοστό βαρύτητας στις προβλέψεις απ' ότι μία απλή μετατροπή της μεταβλητής σε τιμές 0,1,2,3.

Επόμενο βήμα είναι η μετατροπή της εξαρτημένης μεταβλητής, η οποία παίρνει τέσσερις διαφορετικές string τιμές. Έτσι, βασικό στοιχείο ήταν να μετατρέψουμε αυτές τις τιμές σε αριθμητικές ώστε να παίρνει τιμές από 0 μέχρι 3 αντίστοιχα για καλύτερη κατηγοριοποίηση των δεδομένων. Στην συνέχεια, κάναμε split το data set σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής. Και εδώ ο χωρισμός έγινε με `'test_size = 0.20'` όπου για δοκιμή πήγαινε το 20% των δεδομένων μας και το 80% για εκπαίδευση.

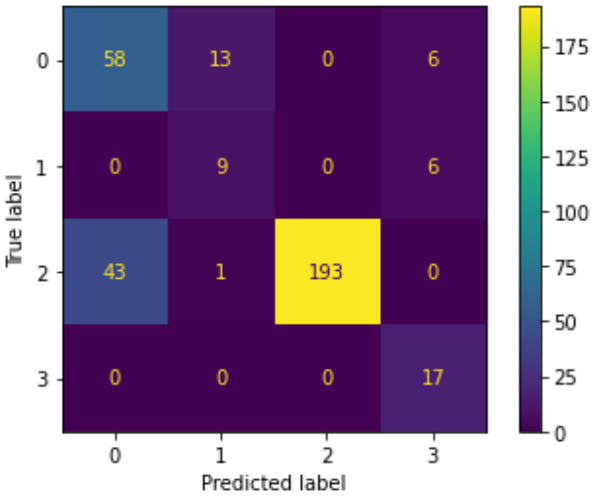
Τέλος, τρέξαμε σε όλες τους Classification αλγόριθμους το νέο διαμορφωμένο data set και είχαμε τα εξής confusion matrix και αποτελέμσατα στις ακρίβειες:

1) Για τον K-NN αλγόριθμο η χρήση των παραμέτρων ήταν η ίδια. Η `'n_neighbors'` μεταβλητή ίση με 5 η οποία μας δίνει τον αριθμό των γειτόνων που θα χρησιμοποιηθεί και η παράμετρος `'p'` για μετρική `Minkowski` με τιμή 2 για να έχουμε Ευκλείδειες αποστάσεις. Το `'random_state'` εδώ ήταν ίσο με 0. Το confusion matrix:



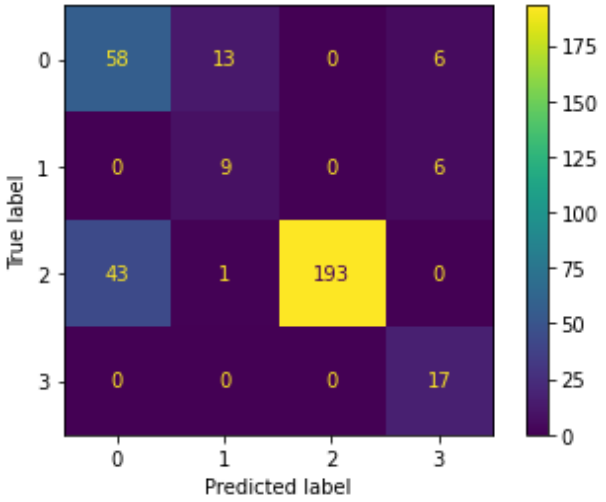
Βλέποντας το confusion matrix του αλγορίθμου παρατηρούμε ότι ο αλγόριθμος πρόβλεψε σχεδόν σωστά όλα τα δεδομένα του που έστειλε για test αφού η ακρίβεια που εμφάνισε το data set ήταν 90,73% με standard deviation 0,02%.

2) Για τον Naïve-Bayes χρησιμοποιήθηκε ο απλός GaussianNB αλγόριθμος. . Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



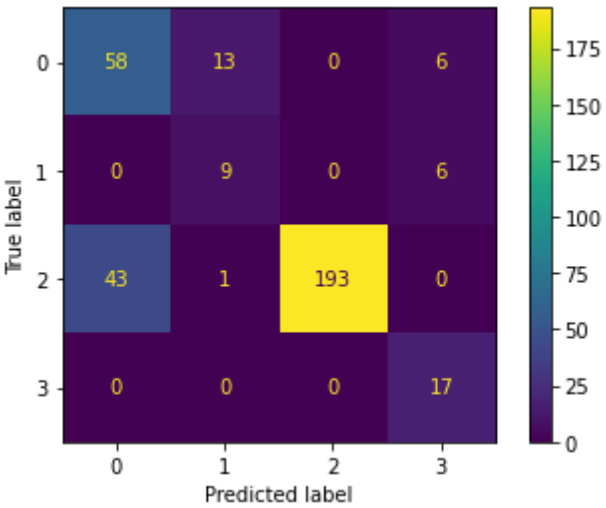
Κι εδώ, στο confusion matrix, παρατηρούμε ότι ο αλγόριθμος έκανε καλές προβλέψεις στα δεδομένα που έστειλε για test, αφού η ακρίβεια που εμφάνισε το data set ήταν 80,30% με standard deviation 0,04%.

3) Για τον Logistic Regression η μόνη παράμετρος που χρησιμοποιήθηκε είναι η `max_iter=1000` για τον μέγιστο αριθμό επαναλήψεων. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Το confusion matrix του αλγορίθμου ακολουθεί το ίδιο μοτίβο αφού τα δεδομένα που έστειλε για test είχαν ακρίβεια 90,8% με standard deviation 0,03%.

4) Για τον CART χρησιμοποιήθηκε η παράμετρος `'criterion'` να είναι ίση με την τιμή `Entropy`. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



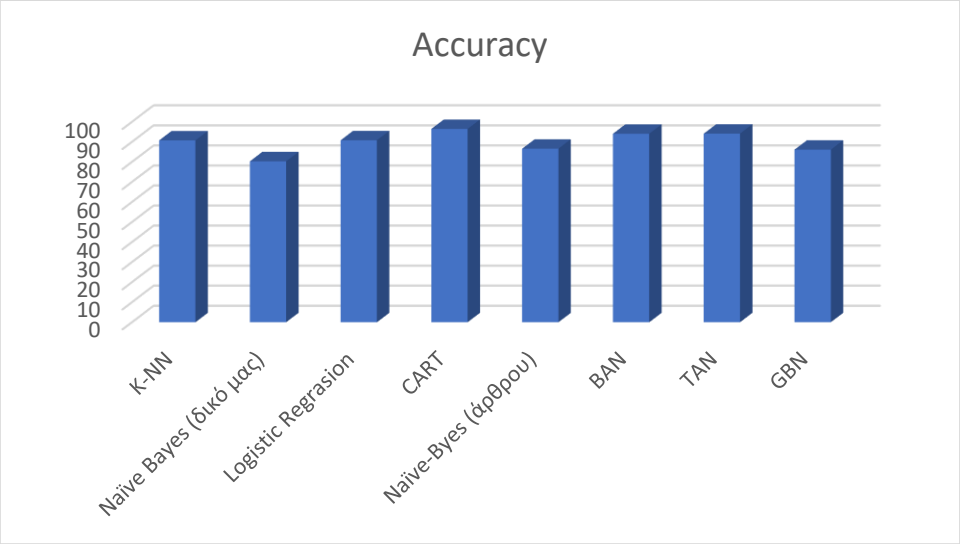
Εδώ τα αποτελέσματα ήταν τα καλύτερα και από τους τέσσερις αλγορίθμους αφού η ακρίβεια που εμφάνισε το data set ήταν 96,45% με standard deviation 0,01%.

5.3.1. Σύγκριση Αποτελεσμάτων του CE Data Set

Μετά από όλα τα αποτελέσματα που συλλέξαμε, παρακάτω παρουσιάζεται ένας πίνακας και ένα σχεδιάγραμμα με την συγκριτική ακρίβεια όλων τον αλγορίθμων για να δούμε ποιός ήταν ο πιο αποτελεσματικός στο συγκεκριμένο data set:

Πίνακας 15: Συνολικά Αποτελέσματα Accuracy για CE Data Set

	Accuracy							
Dataset/Algorithm	**K-NN**	**Naïve (mine)**	**Logistic**	**CART**	**Naïve (άρθρου)**	**BAN**	**TAN**	**GBN**
Car Evaluation	90,73	80,30	90,80	96,45	86,58	94,04	94,10	86,11



Εικόνα 15: Accuracy for CE Data Set

Παρατηρώντας τον πίνακα 15 και το σχεδιάγραμμα στην εικόνα 13 βλέπουμε πως τα καλύτερα αποτελέσματα έχει για ακόμη μια φορά ο αλγόριθμος CART. Αν και τα αποτελέσματα ήταν πολύ κοντά σε όλου τους αλγορίθμους, με τον CART αλγόριθμο καταφέραμε να τους ξεπεράσουμε όλους και να βρούμε την καλύτερη ακρίβεια.

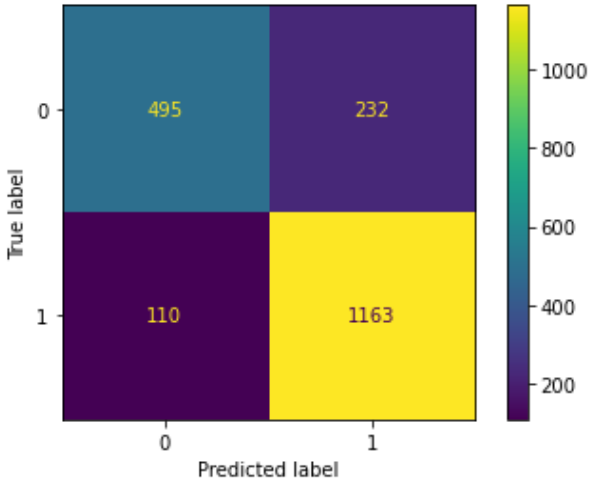
5.4. Προ επεξεργασία στο Electrical Grid Stability Data Set

Το συγκεκριμένο data set αναλύει, όπως αναφέρθηκε στο κεφάλαιο 4, την σταθερότητα της ηλεκτρικής ενέργειας. Το data set αποτελείται από 10000 εγγραφές

και έχει ως σκοπό να δείξει την σταθερότητα ενός συστήματος. Δηλαδή, αν είναι σταθερό ή ασταθές. Αυτό μας το δείχνει η τελευταία στήλη του data set, η εξαρτημένη μεταβλητή ‘stabf’ όπου παίρνει δύο τιμές stable και unstable.

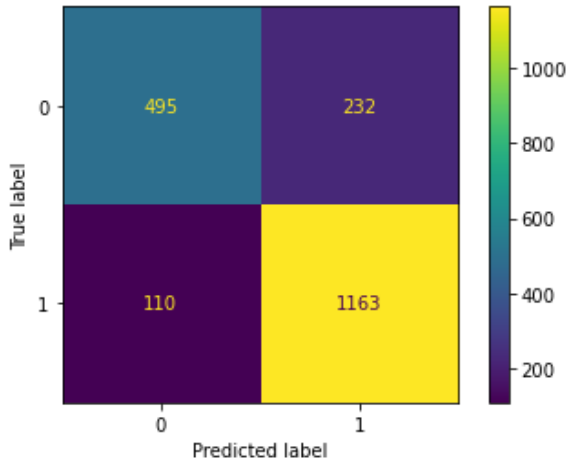
Στις υπόλοιπες 13 στήλες, οι οποίες αποτελούν ένα σύστημα με τέσσερις γεννήτριες, χρόνους αντίδρασης των ατόμων και τιμές παραγωγής και κατανάλωσης ενέργειας, δεν χρειάστηκαν και πολλές ενέργειες καθώς τα δεδομένα ήταν ομοιόμορφα κατανεμημένα και ήταν όλα αριθμητικά. Το μόνο πράγμα που χρειάστηκε εδώ ήταν πάλι στην εξαρτημένη μεταβλητή, η οποία αποτελείτε από δύο string τιμές, να μετατραπεί σε αριθμητική. Για τιμή stable δόθηκε να είναι ίση με 0 και για την τιμή unstable να είναι ίση με 1. Τα δεδομένα χρησιμοποιήθηκαν πάλι από τους τέσσερις classification αλγόριθμους και έτσι έχουμε τα παρακάτω αποτελέσματα στα confusion matrix και τα accuracy αντίστοιχα:

1) Για τον K-NN αλγόριθμο η χρήση των παραμέτρων ήταν στην `'n_neighbors'` μεταβλητή να ίση με 7 η οποία μας δίνει τον αριθμό των γειτόνων που θα χρησιμοποιηθεί και η παράμετρος `'p'` για μετρική `Minkowski` με τιμή 1 για να έχουμε Μανχάταν αποστάσεις. Το `'random_state'` εδώ ήταν ίσο με 0. Το confusion matrix:



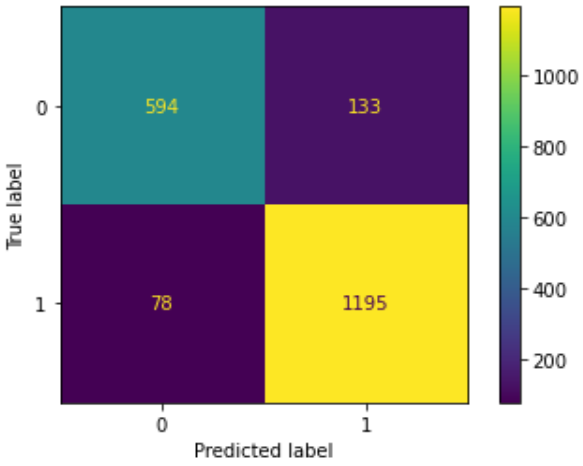
Βλέποντας το confusion matrix του αλγορίθμου παρατηρούμε ότι ο αλγόριθμος πρόβλεψε καλά τα δεδομένα του που έστειλε για test αφού η ακρίβεια που εμφάνισε το data set ήταν 82,15% με standard deviation 0,01%.

2) Για τον Naïve-Bayes χρησιμοποιήθηκε ο απλός GaussianNB αλγόριθμος. . Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



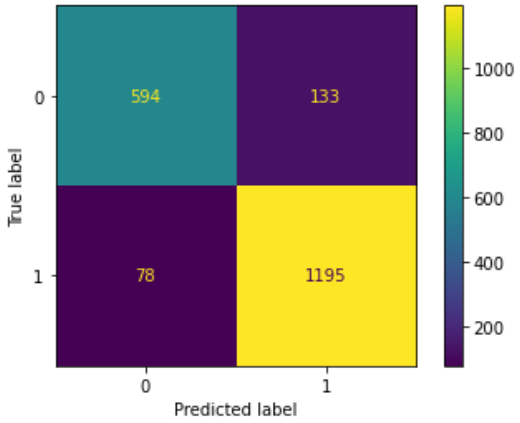
Εδώ στο confusion matrix παρατηρούμε ότι ο αλγόριθμος πρόβλεψε σχεδόν σωστά όλα τα δεδομένα που έστειλε για test αφού η ακρίβεια που εμφάνισε το data set ήταν 97,92% με standard deviation 0,01%.

3) Για τον Logistic Regression η μόνη παράμετρος που χρησιμοποιήθηκε είναι η `max_iter=1000` για τον μέγιστο αριθμό επαναλήψεων. Το `'random_state'` εδώ ήταν ίσο με 0. Τα αποτελέσματα είναι τα εξής:



Το confusion matrix του αλγορίθμου ακολουθεί το ίδιο μοτίβο αφού τα data sets που έστειλε για test είχαν ακρίβεια 89,14% με standard deviation 0,02%.

4) Για τον CART χρησιμοποιήθηκε η παράμετρος `'criterion'` να είναι ίση με την τιμή `Entropy`. Το `'random_state'` εδώ ήταν ίσο με 42. Τα αποτελέσματα είναι τα εξής:



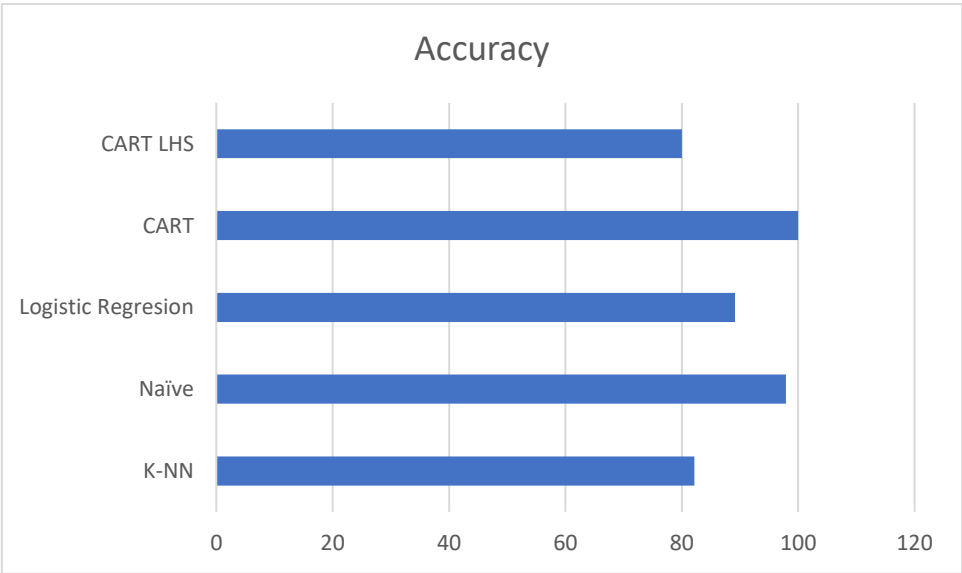
Για άλλη μία φορά εδώ τα αποτελέσματα ήταν τα καλύτερα και από τους τέσσερις αλγορίθμους αφού η ακρίβεια που εμφάνισε το data set ήταν 99,98%.

5.4.1. Σύγκριση Αποτελεσμάτων του EGSS Data Set

Μετά από όλα τα αποτελέσματα που συλλέξαμε, παρακάτω παρουσιάζεται πάλι ένας πίνακας και ένα σχεδιάγραμμα με την συγκριτική ακρίβεια όλων τον αλγορίθμων για να δούμε ποιος ήταν ο πιο αποτελεσματικός στο συγκεκριμένο data set:

Πίνακας 16: Συνολικά Αποτελέσματα Accuracy για EGSS Data Set

	Accuracy				
Data Set/Algorithm	**K-NN**	**Naïve**	**Logistic**	**CART**	**CART LHS**
Electrical Grid Stability	82,15	97,92	89,14	99,98	80,00



Εικόνα 16: Accuracy for EGSS Data Set

Παρατηρώντας τον πίνακα 15 και το σχεδιάγραμμα στην εικόνα 13 βλέπουμε τα αποτελέσματα του κάθε αλγορίθμου. Σε αυτό το data set καταφέραμε να αποδώσουμε με όλους τους αλγορίθμους αρκετά υψηλότερες ακρίβειες σε σχέση με τις ακρίβειες του αλγορίθμου CART LHS που παρουσίασαν οι Vadim Arzamasov, Klemens Bohm και Patrick Jochem στο άρθρο τους "Towards Concise Models of Grid Stability". Και εδώ ο CART αλγόριθμος δούλεψε πιο αποδοτικά από τους υπόλοιπους με τον Naïve Bayes αλγόριθμο να ακολουθεί κι αυτός με μία αρκετά υψηλή ακρίβεια.

6. Συμπεράσματα

Στα προηγούμενα κεφάλαια έγινε μία εμπεριστατωμένη μελέτη της θεωρητικής ανάπτυξης αλλά και της πρακτικής εφαρμογής διάφορων αλγορίθμων για classification αλλά και για clustering πάνω σε κάποια data set . Τα data set είναι από το UC Irvine Machine Learning Repository[3].

Αρχικά, δίνεται μια σύντομη περιγραφή των αλγορίθμων κατηγοριοποίησης (Classification) και μία σύνοψη αυτών που χρησιμοποιούνται ως μέρος της προ επεξεργασίας για ορισμένα σύνολα δεδομένων. Στην συνέχεια, δίνεται περιγραφή για τον αλγόριθμο της συσταδοποίησης (Clustering) που ενσωματώθηκε κι αυτός για να βοηθήσει στην ανάπτυξη των αποτελεσμάτων, ο οποίος συγκεκριμένα χρησιμοποιήθηκε στο 'Teaching Assistant Evaluation' Data Set.

Στο τέταρτο κεφάλαιο γίνεται ανάλυση όλων των data sets που εφαρμόστηκαν στους αλγόριθμους κάνοντας μία ανάλυση ως προς το είδος των μεταβλητών, ανεξάρτητες και εξαρτημένες, και τί τιμές περιέχουν. Μάλιστα, γίνεται και αναφορά σε διάφορα άρθρα που έχουν εφαρμόσει τα συγκεκριμένα data sets και τί αποτελέσματα είχαν στις ακρίβειες είτε αυτές ήταν μετρημένες σε τιμές accuracy, είτε βάση ποσοστού error rates, είτε σε precision και F_1 Score.

Στο πέμπτο, και τελευταίο, κεφάλαιο αναφέρονται τα αποτελέσματα της δικιάς μας έρευνας με εφαρμογή των data sets πάνω στους αλγορίθμους K-NN, Naïve Bayes, Logistic Regresion, CART και στον αλγόριθμο συσταδοποίησης K-Means. Τέλος, γίνεται σύγκριση των δικών μας αποτελεσμάτων με αυτά που παρουσιάστηκαν στο τέταρτο κεφάλαιο μέσω συγκεντρωτικών πινάκων και σχεδιαγραμμάτων.

Αυτό που καταφέραμε να κάνουμε ήταν να βγάλουμε καλύτερα αποτελέσματα από αυτά που είχαν παρουσιαστεί σε διάφορα άρθρα πέραν του 'Teaching Assistant Evaluation' Data Set όπου εκεί δεν δούλεψαν όλα όπως τα περιμέναμε με το συμπέρασμά μας να αποδίδεται στο μικρό μέγεθος του data set. Όπως είδαμε, τα καλύτερα αποτελέσματα τα είχε σε όλα τα data sets ο CART αλγόριθμος και ήταν αυτός που έκανε την καλύτερη εκπαίδευση των δεδομένων κάθε φορά και πρόβλεψη αυτών, χωρίς βέβαια οι υπόλοιποι να μην τα πάνε εξίσου καλά.

[3] https://archive.ics.uci.edu/ml/datasets.php

Αυτό που έχει βέβαια σημασία είναι πως υπάρχει ακόμα χώρος για βελτίωση αυτών τον αποτελεσμάτων δεδομένου ότι μπορεί να υπάρχουν άλλοι αλγόριθμοι που να έχουν καλύτερη απόδοση σε αυτά τα σύνολα δεδομένων που χρησιμοποιήθηκαν. Κάθε αναλυτής δεδομένων σίγουρα έχει όνειρο να είναι σε θέση να χρησιμοποιήσει έναν αλγόριθμο που να έχει την καλύτερη απόδοση όσον αφορά την ακρίβεια και το χρόνο εκτέλεσης, ανεξάρτητα από το σύνολο δεδομένων που τον δοκιμάζει. Ωστόσο, λόγω των διαφορετικών λειτουργιών κάθε αλγορίθμου, δεν υπάρχει καμιά οδηγία για το ποιος αλγόριθμος ταιριάζει καλύτερα στα δεδομένα που έχει κάποιος, προκειμένου να παράγει το καλύτερο μοντέλο. Πρέπει να εφαρμοστούν και να δοκιμαστούν ώστε να δει κάποιος τα αποτελέσματα και να κάνει συγκρίσεις.

Computers are able to see, hear and learn. Welcome to the future.

~*Dave Waters*

Βιβλιογραφία

Angel, L., Viola, J., Vega, M., & Restrepo, R. (2016, August). Sterilization Process Stages Estimation for an Autoclave Using Logistic Regression Models, *IEEE Xplore*. doi:10.1109/STSIVA.2016.7743337.

Arzamasov, V., Bohm, K., & Jochem, P. (2018). Towards Concise Models of Grid Stability, *IEEE Xplore,* doi:10.1109/SmartGridComm.2018.8587498.

Bohanec, M. & Rajkovic, V. (1990). DEX: An Expert System Shell for Decision Support, *Sistemica, no.1,* pp. 145-157.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). Classification and Regression Trees, *New York.*

Cheng, J., Bell, D.A., & Liu, W. (1997). An Algorithm For Bayesian Belief Network Construction From Data, *Proceedings Of AI,* pp.83-90.

Cheng, J., Greiner, R. (1999). Comparing Bayesian Network Classifiers, *UAI,* 101-108.

Christopher Sibona, J.B. (2012). A Statistical Comparison Of Classification Algorithm On A Single Data Set, *AMCIS.*

Ertam, F., Kaya, M. (2018, March). Classification of Firewall Log Files with Multiclass Support Vector Machine, *IEEE Xplore,* doi:10.1109/ISDFS.2018.8355382.

Garsia-Gonzalo, E., Fernaandez-Muniz, Z., Nieto, P.J.G., Sanchez, A.B., & Fernandez, M.M. (2016, June) Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers, *Materials, no.9* (vol.531), doi:10.3390/ma9070531.

Guo, H., Nguyen, H., Vu, D.A., & Bui, X.N. (2019, August). Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach, *Resources Policy*. doi: 10.1016/j.resourpol.2019.101474.

Jordan, M.I., & Mitchell, T.M. (2015, July). Machine learning: Trends, perspectives, and prospects, *Science Mag,* (vol.349), pp.255-260

Lim, T.S., Loh, W.Y., & Shih, Y.S. (2000). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms, *Machine Learning, no.40,* pp.203-229.

Lizotte, D.J., Madani, O., & Greiner, R. (2003). Budgeted Learning of Naive-Bayes Classifiers, *UAI,* pp.378-385.

Loh, W.Y., & Shih, Y.S. (1997). Split Selection Methods For Classification Trees, *Statistica Sinica, no.7,* pp.815-840.

Mohd, W.M.W., Beg, A.H., Herawan, T. & Rabbi, K.F. (2012, January). MaxD K-means: A clustering algorithm for Auto-generation of centroids and distance of data points in clusters, *Communications in Computer and Information Science,* doi:10.1007/978-3-642-34289-9_22.

Niazi, A., Dai, J.S., Balabani, S., Seneviratne, L., (2006). Product cost estimation: technique classification and methodology review. *J. Manuf. Sci. Eng. No.128,* pp.563–575.

Pan, J.S., Qiao, Y.L., & Sun, S.H. (2004, April). A Fast k Nearest Neighbors Classification Algorithm, *IEICE Trans. Fundamentals, no.4* (vol.E87).

Quinlan, J.R. (1986). Induction Of Decision Trees, *Machine Learning,* (vol.1), pp.81-106, doi:10.1007/bf00116251

Saptura, M.F.A., Widiyaningtyas, T. & Wibawa, A.P. (2018). Illiteracy Classification Using K Means-Naïve Bayes Algorithm, *International Journal On Informatics Visualization, no.3* (vol.2).

Sebban, M., Nock, R., & Lallich, S. (2002). Stopping Criterion for Boosting-Based Data Reduction Techniques: from Binary to Multiclass Problems, *Journal Of Machine Learning Research, no.3,* pp.863-885

Tay, B., Hyun, J.K., Oh, S. (2014). A Machine Learning Approach for Specification of Spinal Cord Injuries Using Fractional Anisotropy Values Obtained from Diffusion Tensor Images, *Computational and Mathematical Methods in Medicine,* (vol.2014), doi:10.1155/2014/276589.

Zhang, M.L, & Zhou, Z.H. (2005). A k-nearest neighbor based algorithm for multi-label classification, *IEEE Xplore,* (vol.2), doi:10.1109/GRC.2005.1547385.

ΠΑΡΑΡΤΗΜΑ

```
# K-Nearest Neighbors (K-NN)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Firewall.csv')

# Convert dataset to numeric because it is as float
dataset['Source_Port'] = pd.to_numeric(dataset['Source_Port'],
downcast='float')
dataset['Destination_Port'] =
pd.to_numeric(dataset['Destination_Port'], downcast='float')
dataset['NAT_Source_Port'] =
pd.to_numeric(dataset['NAT_Source_Port'], downcast='float')
dataset['NAT_Destination_Port'] =
pd.to_numeric(dataset['NAT_Destination_Port'], downcast='float')
dataset['Bytes'] = pd.to_numeric(dataset['Bytes'],
downcast='float')
dataset['Bytes_Sent'] = pd.to_numeric(dataset['Bytes_Sent'],
downcast='float')
dataset['Bytes_Received'] =
pd.to_numeric(dataset['Bytes_Received'], downcast='float')
dataset['Packets'] = pd.to_numeric(dataset['Packets'],
downcast='float')
dataset['Elapsed_Time_(sec)'] =
pd.to_numeric(dataset['Elapsed_Time_(sec)'], downcast='float')
dataset['Pkts_sent'] = pd.to_numeric(dataset['Pkts_sent'],
downcast='float')
dataset['Pkts_received'] = pd.to_numeric(dataset['Pkts_received'],
downcast='float')

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,5,6,7,8,9,10,11]].values
y = dataset.iloc[:, 4].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)


# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.fit_transform(X_train[:,
[0,1,2,3,4,5,6,7,8,9,10]])
X_test[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.transform(X_test[:,
[0,1,2,3,4,5,6,7,8,9,10]])
```

```
# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 01: K-NN στο IF Data Set

```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Firewall.csv')

# Convert dataset to numeric because it is as float
dataset['Source_Port'] = pd.to_numeric(dataset['Source_Port'],
downcast='float')
dataset['Destination_Port'] =
pd.to_numeric(dataset['Destination_Port'], downcast='float')
dataset['NAT_Source_Port'] =
pd.to_numeric(dataset['NAT_Source_Port'], downcast='float')
dataset['NAT_Destination_Port'] =
pd.to_numeric(dataset['NAT_Destination_Port'], downcast='float')
dataset['Bytes'] = pd.to_numeric(dataset['Bytes'],
downcast='float')
dataset['Bytes_Sent'] = pd.to_numeric(dataset['Bytes_Sent'],
downcast='float')
```

```
dataset['Bytes_Received'] =
pd.to_numeric(dataset['Bytes_Received'], downcast='float')
dataset['Packets'] = pd.to_numeric(dataset['Packets'],
downcast='float')
dataset['Elapsed_Time_(sec)'] =
pd.to_numeric(dataset['Elapsed_Time_(sec)'], downcast='float')
dataset['Pkts_sent'] = pd.to_numeric(dataset['Pkts_sent'],
downcast='float')
dataset['Pkts_received'] = pd.to_numeric(dataset['Pkts_received'],
downcast='float')

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,5,6,7,8,9,10,11]].values
y = dataset.iloc[:, 4].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.fit_transform(X_train[:,
[0,1,2,3,4,5,6,7,8,9,10]])
X_test[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.transform(X_test[:,
[0,1,2,3,4,5,6,7,8,9,10]])

# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))
```

```
# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 02: Naïve Bayes στο IF Data Se

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Firewall.csv')

# Convert dataset to numeric because it is as float
dataset['Source_Port'] = pd.to_numeric(dataset['Source_Port'],
downcast='float')
dataset['Destination_Port'] =
pd.to_numeric(dataset['Destination_Port'], downcast='float')
dataset['NAT_Source_Port'] =
pd.to_numeric(dataset['NAT_Source_Port'], downcast='float')
dataset['NAT_Destination_Port'] =
pd.to_numeric(dataset['NAT_Destination_Port'], downcast='float')
dataset['Bytes'] = pd.to_numeric(dataset['Bytes'],
downcast='float')
dataset['Bytes_Sent'] = pd.to_numeric(dataset['Bytes_Sent'],
downcast='float')
dataset['Bytes_Received'] =
pd.to_numeric(dataset['Bytes_Received'], downcast='float')
dataset['Packets'] = pd.to_numeric(dataset['Packets'],
downcast='float')
dataset['Elapsed_Time_(sec)'] =
pd.to_numeric(dataset['Elapsed_Time_(sec)'], downcast='float')
dataset['Pkts_sent'] = pd.to_numeric(dataset['Pkts_sent'],
downcast='float')
dataset['Pkts_received'] = pd.to_numeric(dataset['Pkts_received'],
downcast='float')

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,5,6,7,8,9,10,11]].values
y = dataset.iloc[:, 4].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.fit_transform(X_train[:,
[0,1,2,3,4,5,6,7,8,9,10]])
```

```
X_test[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.transform(X_test[:,
[0,1,2,3,4,5,6,7,8,9,10]])

# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 03: Logistic Regresion στο IF Data Set

```
# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Firewall.csv')

# Convert dataset to numeric because it is as float
dataset['Source_Port'] = pd.to_numeric(dataset['Source_Port'],
downcast='float')
dataset['Destination_Port'] =
pd.to_numeric(dataset['Destination_Port'], downcast='float')
dataset['NAT_Source_Port'] =
pd.to_numeric(dataset['NAT_Source_Port'], downcast='float')
dataset['NAT_Destination_Port'] =
pd.to_numeric(dataset['NAT_Destination_Port'], downcast='float')
dataset['Bytes'] = pd.to_numeric(dataset['Bytes'],
downcast='float')
```

```
dataset['Bytes_Sent'] = pd.to_numeric(dataset['Bytes_Sent'],
downcast='float')
dataset['Bytes_Received'] =
pd.to_numeric(dataset['Bytes_Received'], downcast='float')
dataset['Packets'] = pd.to_numeric(dataset['Packets'],
downcast='float')
dataset['Elapsed_Time_(sec)'] =
pd.to_numeric(dataset['Elapsed_Time_(sec)'], downcast='float')
dataset['Pkts_sent'] = pd.to_numeric(dataset['Pkts_sent'],
downcast='float')
dataset['Pkts_received'] = pd.to_numeric(dataset['Pkts_received'],
downcast='float')

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,5,6,7,8,9,10,11]].values
y = dataset.iloc[:, 4].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.fit_transform(X_train[:,
[0,1,2,3,4,5,6,7,8,9,10]])
X_test[:, [0,1,2,3,4,5,6,7,8,9,10]] = sc.transform(X_test[:,
[0,1,2,3,4,5,6,7,8,9,10]])

# Training the Decision Tree Classification model on the Training
set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
```

```
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 04: CART στο IF Data Set

```
# K-Nearest Neighbors (K-NN)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')

# Split the column 'Class size' in six spaces
dataset.loc[dataset['Class size'] <= 11, 'Size'] = '1'
dataset.loc[(dataset['Class size'] >11) & (dataset['Class size']
<=22), 'Size'] = '2'
dataset.loc[(dataset['Class size'] >22) & (dataset['Class size']
<=33), 'Size'] = '3'
dataset.loc[(dataset['Class size'] >33) & (dataset['Class size']
<=44), 'Size'] = '4'
dataset.loc[(dataset['Class size'] >44) & (dataset['Class size']
<=55), 'Size'] = '5'
dataset.loc[(dataset['Class size'] >55) & (dataset['Class size']
<=67), 'Size'] = '6'

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 6, metric =
'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
```

```
#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 05: K-NN στο TEA Data Set

```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')

# Split the column 'Class size' in six spaces
dataset.loc[dataset['Class size'] <= 11, 'Size'] = '1'
dataset.loc[(dataset['Class size'] >11) & (dataset['Class size']
<=22), 'Size'] = '2'
dataset.loc[(dataset['Class size'] >22) & (dataset['Class size']
<=33), 'Size'] = '3'
dataset.loc[(dataset['Class size'] >33) & (dataset['Class size']
<=44), 'Size'] = '4'
dataset.loc[(dataset['Class size'] >44) & (dataset['Class size']
<=55), 'Size'] = '5'
dataset.loc[(dataset['Class size'] >55) & (dataset['Class size']
<=67), 'Size'] = '6'

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 06: Naïve Bayes στο TEA Data Set

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')

# Split the column 'Class size' in six spaces
dataset.loc[dataset['Class size'] <= 11, 'Size'] = '1'
dataset.loc[(dataset['Class size'] >11) & (dataset['Class size']
<=22), 'Size'] = '2'
dataset.loc[(dataset['Class size'] >22) & (dataset['Class size']
<=33), 'Size'] = '3'
dataset.loc[(dataset['Class size'] >33) & (dataset['Class size']
<=44), 'Size'] = '4'
dataset.loc[(dataset['Class size'] >44) & (dataset['Class size']
<=55), 'Size'] = '5'
dataset.loc[(dataset['Class size'] >55) & (dataset['Class size']
<=67), 'Size'] = '6'

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
```

```
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter = 300, random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 07: Logistic Regresion στο TEA Data Set

```
# Decision Tree Classification
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')

# Split the column 'Class size' in six spaces
dataset.loc[dataset['Class size'] <= 11, 'Size'] = '1'
dataset.loc[(dataset['Class size'] >11) & (dataset['Class size']
<=22), 'Size'] = '2'
```

```
dataset.loc[(dataset['Class size'] >22) & (dataset['Class size']
<=33), 'Size'] = '3'
dataset.loc[(dataset['Class size'] >33) & (dataset['Class size']
<=44), 'Size'] = '4'
dataset.loc[(dataset['Class size'] >44) & (dataset['Class size']
<=55), 'Size'] = '5'
dataset.loc[(dataset['Class size'] >55) & (dataset['Class size']
<=67), 'Size'] = '6'

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Decision Tree Classification model on the Training
set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 08: CART στο TEA Data Set

```
# K-Nearest Neighbors (K-NN)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')
T = dataset.iloc[:, [0,1,2,3,5]].values

# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',)
    kmeans.fit(T)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Training the K-Means model on the dataset
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state =
42)
y_kmeans = kmeans.fit_predict(T)
print(y_kmeans)

# Making new class by ‘class size’ column
dataset['new_class_size'] = y_kmeans

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 42)

# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
```

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 09: K-NN στο TEA Data Set με χρήση K-Means Model

```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')
T = dataset.iloc[:, [0,1,2,3,5]].values

# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',)
    kmeans.fit(T)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Training the K-Means model on the dataset
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state =
0)
y_kmeans = kmeans.fit_predict(T)
print(y_kmeans)

# Making new class by ‘class size’ column
dataset['new_class_size'] = y_kmeans
```

```
# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 10: Naïve Bayes στο TEA Data Set με χρήση K-Means Model

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')
T = dataset.iloc[:, [0,1,2,3,5]].values
```

```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',)
    kmeans.fit(T)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Training the K-Means model on the dataset
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state =
42)
y_kmeans = kmeans.fit_predict(T)
print(y_kmeans)

# Making new class by ‘class size’ column
dataset['new_class_size'] = y_kmeans

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 42)

# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter = 300, random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accurancy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
```

```
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 11: Logistic Regresion στο TEA Data Set με χρήση K-Means Model

```
# CART

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Teaching.csv')
T = dataset.iloc[:, [0,1,2,3,5]].values

# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',)
    kmeans.fit(T)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Training the K-Means model on the dataset
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 6, init = 'k-means++', random_state =
42)
y_kmeans = kmeans.fit_predict(T)
print(y_kmeans)

# Making new class by ‘class size’ column
dataset['new_class_size'] = y_kmeans

# Insert columns of dataset in two variables
X = dataset.iloc[:, [0,1,2,3,6]].values
y = dataset.iloc[:, 5].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 42)

# Training the Decision Tree Classification model on the Training
set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 42)
```

```
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Find the error rate to compare the datasets
error_rate = 1 - accuracies
print(error_rate)
A=0
for i in error_rate:
    A = A + i
A = A/10
print('Error rate: {:.2f} %'.format(A.mean()*100))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 12: CART στο TEA Data Set με χρήση K-Means Model

```
# K-Nearest Neighbors (K-NN)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Car.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[0])], remainder='passthrough')
ct1 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[4])], remainder='passthrough')
```

```
ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[8])], remainder='passthrough')
ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[12])], remainder='passthrough')
ct4 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[15])], remainder='passthrough')
ct5 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[18])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
X = np.array(ct1.fit_transform(X))
X = np.array(ct2.fit_transform(X))
X = np.array(ct3.fit_transform(X))
X = np.array(ct4.fit_transform(X))
X = np.array(ct5.fit_transform(X))
print(X)

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 13: K-NN στο Car Data Set

```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Car.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[0])], remainder='passthrough')
ct1 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[4])], remainder='passthrough')
ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[8])], remainder='passthrough')
ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[12])], remainder='passthrough')
ct4 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[15])], remainder='passthrough')
ct5 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[18])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
X = np.array(ct1.fit_transform(X))
X = np.array(ct2.fit_transform(X))
X = np.array(ct3.fit_transform(X))
X = np.array(ct4.fit_transform(X))
X = np.array(ct5.fit_transform(X))
print(X)

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 42)

# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 14: Naïve Bayes στο Car Data Set

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Car.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[0])], remainder='passthrough')
ct1 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[4])], remainder='passthrough')
ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[8])], remainder='passthrough')
ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[12])], remainder='passthrough')
ct4 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[15])], remainder='passthrough')
ct5 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[18])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
X = np.array(ct1.fit_transform(X))
X = np.array(ct2.fit_transform(X))
X = np.array(ct3.fit_transform(X))
X = np.array(ct4.fit_transform(X))
X = np.array(ct5.fit_transform(X))
print(X)

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 15: Logistic Regresion στο Car Data Set

```
# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Car.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[0])], remainder='passthrough')
ct1 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[4])], remainder='passthrough')
```

```
ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[8])], remainder='passthrough')
ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[12])], remainder='passthrough')
ct4 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[15])], remainder='passthrough')
ct5 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[18])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
X = np.array(ct1.fit_transform(X))
X = np.array(ct2.fit_transform(X))
X = np.array(ct3.fit_transform(X))
X = np.array(ct4.fit_transform(X))
X = np.array(ct5.fit_transform(X))
print(X)

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Decision Tree Classification model on the Training
set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 16: Cart στο Car Data Set

```
# K-Nearest Neighbors (K-NN)

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_for_UCI_named.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 7, metric =
'minkowski', p = 1)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 17: K-NN στο EGSS Data Set

```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_for_UCI_named.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 18: Naïve Bayes στο EGSS Data Set

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_for_UCI_named.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 0)

# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

# kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 19: Logistic Regresion στο EGSS Data Set

```
# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Data_for_UCI_named.csv')

# Insert columns of dataset in two variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Encoding the Dependent Variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.20, random_state = 42 )

# Training the Decision Tree Classification model on the Training
set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy')
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score, precision_score, f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred, average='macro'))
print(precision_score(y_test, y_pred, average='macro',
zero_division=1))
print(f1_score(y_test, y_pred, average='macro'))

#kFold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y
= y_train, cv = 10)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard Deviation: {:.2f} %'.format(accuracies.std()))

# Print confusion matrix
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

Κώδικας 20: CART στο EGSS Data Set