ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΝΕΑ ΜΕΘΟΔΟΛΟΓΙΑ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΙΚΩΝ ΕΠΙΒΛΕΠΟΜΕΝΗΣ ΚΑΙ ΜΗ ΕΠΙΒΛΕΠΟΜΕΝΗΣ ΕΚΜΑΘΗΣΗΣ

Διπλωματική Εργασία

του

Μιχαλόπουλου Μάριου

Θεσσαλονίκη, Ιανουάριος 2020

ΝΕΑ ΜΕΘΟΔΟΛΟΓΙΑ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΙΚΩΝ ΕΠΙΒΛΕΠΟΜΕΝΗΣ ΚΑΙ ΜΗ ΕΠΙΒΛΕΠΟΜΕΝΗ ΕΚΜΑΘΗΣΗΣ

Μιχαλόπουλος Μάριος

Πτυχίο Στρατιωτικής Σχολής Ευελπίδων, ΣΣΕ, 2008

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 26/02/2020

Σαμαράς Νικόλαος	Ευαγγελίδης Γεώργιος	Ρεφανίδης Ιωάννης
.....	.....	.....

Μιχαλόπουλος Μάριος

.....

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας είναι η κατασκευή μοντέλων πρόβλεψης βασισμένα σε τέσσερα διαφορετικά σύνολα δεδομένων με την χρήση διάφορων αλγορίθμων κατηγοριοποίησης και παλινδρόμησης, που έχουν υλοποιηθεί με την γλώσσα προγραμματισμού Python και με τη βιβλιοθήκη μηχανικής μάθησης Scikit-learn. Παράλληλα γίνεται σύγκριση των παραπάνω μοντέλων, με μια προτεινόμενη μεθοδολογία που συνδυάζει αλγορίθμους εποπτευόμενης και μη εποπτευόμενης μάθησης μαζί με μια μέθοδο επιλογής χαρακτηριστικών. Η αλγόριθμοι που θα χρησιμοποιηθούν είναι:

1. Νευρωνικά δίκτυα (Multilayer Perceptron),
2. Δέντρα απόφασης (Decision Trees)
3. Λογιστική παλινδρόμηση (Logistic Regression)
4. Γραμμική παλινδρόμηση (Linear Regression)
5. Αλγόριθμος K-μέσων (K-Means)

Όλα τα σύνολα δεδομένων που θα χρησιμοποιηθούν, μπορούν να βρεθούν στην σελίδα του UC Irvine Machine Learning Repository. Η έκδοση της βιβλιοθήκης Scikit-learn που θα χρησιμοποιηθεί στο σύνολο της εργασίας είναι η 0.22.1 .

Λέξεις Κλειδιά: *Νευρωνικά δίκτυα, Δέντρα απόφασης, Λογιστική παλινδρόμηση, Γραμμική παλινδρόμηση, Αλγόριθμος K-μέσων*

Abstract

The purpose of this thesis is to develop prediction models based on four different datasets using various categorization and regression algorithms implemented with the Python programming language and the Scikit-learn machine learning library. At the same time, the above models are compared with a proposed methodology that combines supervised and non-supervised learning algorithms along with feature selection technic. The algorithms to be used are:

1. Neural Networks (Multilayer Perceptron),
2. Decision Trees (CART)
3. Logistic Regression
4. Linear Regression
5. K-Means

The data sets that will be used are taken from UC Irvine Machine Learning Repository and are available online. The version of Scikit-learn library that we will use for the entire thesis is 0.22.1 .

Keywords: *Linear Regression, K-Means, Logistic Regression, Decision Trees, Neural Network, Multilayer Perceptron*

Περιεχόμενα

1 Εισαγωγή 1
1.1 Πρόβλημα 1
1.2 Στόχοι 1
1.3 Διάρθρωση της μελέτης 2
2 Θεωρητικό Υπόβαθρο 3
2.1 Γραμμική Παλινδρόμηση (Linear Regression) 3
2.2 Δένδρα Απόφασης (Decision Trees) 4
2.3 Λογιστική Παλινδρόμηση (Logistic Regression) 5
2.4 Νευρωνικά Δίκτυα (Neural Networks) 6
2.5 Αλγόριθμος Κ-μέσων 8
2.6 Μετρικές μέθοδοι 9
2.6.1 Μετρικές μέθοδοι κατά την παλινδρόμηση 9
2.6.2 Μετρικές μέθοδοι κατά την κατηγοριοποίηση 11
2.6.3 Πίνακας σύγχυσης 14
2.6.4 Συγκριτική αξιολόγηση απόδοσης μοντέλων με ανάλυση της διακύμανσης 15
3 Χρήση του αλγορίθμου Κ-μέσων κατά την κατηγοριοποίηση 17
3.1 Εισαγωγή 17
3.2 Επιθυμητά αποτελέσματα 17
3.3 Προτεινόμενη Μεθοδολογία 17
3.3.1 Επιλογή χαρακτηριστικών 18
3.3.2 Ομαδοποίηση 18
3.3.3 Ταξινόμηση 18
3.4 Πειραματική διάταξη 19
4 Μελέτη συνόλου δεδομένων Auto MPG 21
4.1 Εισαγωγή 21
4.2 Ανάλυση δεδομένων 21
4.2.1 Εξέταση χαρακτηριστικών 21
4.2.2 Διάγραμμα συσχετίσεων 26
4.3 Επεξεργασία δεδομένων 26
4.3.1 Καθαρισμός δεδομένων 26
4.3.2 Μετασχηματισμός δεδομένων 27

4.4 Αποτελέσματα 27
5 Μελέτη συνόλου δεδομένων Car Evaluation 29
5.1 Εισαγωγή 29
5.2 Ανάλυση δεδομένων 29
5.3 Επεξεργασία δεδομένων 30
5.3.1 Καθαρισμός δεδομένων 30
5.3.2 Μετασχηματισμός δεδομένων 30
5.3.3 Μη ισορροπημένες κλάσεις 30
5.4 Αποτελέσματα 31
6 Μελέτη συνόλου δεδομένων Bank Marketing 36
6.1 Εισαγωγή 36
6.2 Ανάλυση δεδομένων 36
6.2.1 Ανάλυση χαρακτηριστικών 36
6.2.2 Πίνακας συσχετίσεων 38
6.3 Επεξεργασία 39
6.3.1 Καθαρισμός δεδομένων 39
6.3.2 Μετασχηματισμός δεδομένων 40
6.3.3 Μη ισορροπημένες κλάσεις 40
6.4 Αποτελέσματα 41
7 Μελέτη συνόλου δεδομένων Default of credit card clients 45
7.1 Εισαγωγή 45
7.2 Ανάλυση δεδομένων 45
7.2.1 Πίνακας συσχετίσεων 47
7.3 Επεξεργασία δεδομένων 48
7.3.1 Καθαρισμός δεδομένων 48
7.3.2 Μετασχηματισμός δεδομένων 48
7.4 Αποτελέσματα 48
8 Επίλογος 53
8.1 Σύνοψη και συμπεράσματα 53
8.2 Όρια και περιορισμοί της έρευνας 53
8.3 Μελλοντικές Επεκτάσεις 55
Παράρτημα Α- Κώδικας σε Python 59

Κατάλογος Εικόνων

Εικόνα 2-1: Λογιστική καμπύλη (Πετρίδης, 2015)..... 5
Εικόνα 2-2: Δίκτυο πολυεπίπεδων νευρώνων (scikit-learn Machine Learning in Python) 7
Εικόνα 2-5: Διάγραμμα καμπύλης ROC. 13
Εικόνα 2-6: Περιοχή κάτω από την καμπύλη ROC. 14
Εικόνα 2-7: Πίνακας σύγχυσης για πρόβλημα ταξινόμησης δύο κλάσεων. 15
Εικόνα 3-1: Διάγραμμα ροής προτεινόμενης μεθοδολογίας. 19
Εικόνα 4-1: Γραφική αναπαράσταση σχέσεων μεταξύ των ποσοτικών χαρακτηριστικών του συνόλου δεδομένων Auto MPG. 22
Εικόνα 4-2: Πλήθος οχημάτων ανά αριθμό κυλίνδρων. 22
Εικόνα 4-3: Πλήθος οχημάτων ανά έτος κατασκευής. 23
Εικόνα 4-4: Αριθμός οχημάτων ανά περιοχή προέλευσης..... 23
Εικόνα 4-5: Διάγραμμα αυτονομίας καυσίμου σε σχέση με την περιοχή προέλευσης.... 24
Εικόνα 4-6: Διάγραμμα αυτονομίας καυσίμου ανά αριθμό κυλίνδρων. 25
Εικόνα 4-7: Διάγραμμα αυτονομίας καυσίμου ανά έτος κατασκευής. 25
Εικόνα 4-8: Πίνακας συσχετίσεων Pearson για το σύνολο δεδομένων Auto MPG..... 26
Εικόνα 5-1: Πλήθος οχημάτων για κάθε χαρακτηριστικό. 29
Εικόνα 5-2: Πλήθος οχημάτων ανά κατηγορία. 30
Εικόνα 5-3: Αποτελέσματα SMOTE ανά αριθμό πλησιέστερων γειτόνων για το σύνολο δεδομένων Car Evaluation. 31
Εικόνα 5-4: Αποτελέσματα λογιστικής παλινδρόμησης με την χρήση του αλγορίθμου K-μέσων και 18 χαρακτηριστικών για το σύνολο δεδομένων Car Evaluation. 32
Εικόνα 5-5: Πίνακας σύγχυσης για την λογιστική παλινδρόμηση με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Car Evaluation. 33
Εικόνα 5-6: Πίνακας σύγχυσης για το δέντρο απόφασης με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Car Evaluation. ... 34
Εικόνα 5-7: Πίνακας σύγχυσης για το νευρωνικό δίκτυο με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Car Evaluation. ... 34
Εικόνα 6-1: Πλήθος παρατηρήσεων για το χαρακτηριστικό στόχος του συνόλου δεδομένων Bank Marketing. 36
Εικόνα 6-2: Κατηγορικά χαρακτηριστικά του συνόλου δεδομένων Bank Marketing..... 37

Εικόνα 6-3: Κατανομή ποσοτικών χαρακτηριστικών του συνόλου δεδομένων Bank Marketing. 38
Εικόνα 6-4: Πίνακας συσχετίσεων Pearson για τα χαρακτηριστικά με συνεχής τιμές για το σύνολο δεδομένων Bank Marketing. 39
Εικόνα 6-5: Αποτελέσματα SMOTE ανά αριθμό πλησιέστερων γειτόνων με την χρήση λογιστικής παλινδρόμησης για το σύνολο δεδομένων Bank Marketing. 40
Εικόνα 6-6: Αποτελέσματα λογιστικής παλινδρόμησης με την χρήση του αλγορίθμου K-μέσων και 14 χαρακτηριστικά για το σύνολο δεδομένων Bank Marketing. 41
Εικόνα 6-7: Πίνακας σύγχυσης για την λογιστική παλινδρόμηση με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Bank Marketing. 43
Εικόνα 6-8: Πίνακας σύγχυσης για την δέντρου απόφασης με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Bank Marketing. 43
Εικόνα 6-9: Πίνακας σύγχυσης για το νευρωνικό δίκτυο με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Bank Marketing. . 43
Εικόνα 7-1: Πλήθος παρατηρήσεων για το χαρακτηριστικό στόχος για το σύνολο δεδομένων Default of credit card clients. 46
Εικόνα 7-2: Κατανομές κατηγορικών χαρακτηριστικών σε σχέση με το χαρακτηριστικό στόχος για το σύνολο δεδομένων Default of credit card clients. 46
Εικόνα 7-3: Πίνακας συσχετίσεων Pearson των χαρακτηριστικών με συνεχείς τιμές για το σύνολο δεδομένων Default of credit card clients. 47
Εικόνα 7-4: Αποτελέσματα λογιστικής παλινδρόμησης με την χρήση του αλγορίθμου K-μέσων για το σύνολο δεδομένων Default of credit card clients. 49
Εικόνα 7-5: Πίνακας σύγχυσης για την λογιστική παλινδρόμηση με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Default of credit card clients. 50
Εικόνα 7-6: Πίνακας σύγχυσης για το δέντρο απόφασης με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Default of credit card clients. 51
Εικόνα 7-7: Πίνακας σύγχυσης για το νευρωνικό δίκτυο με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Default of credit card clients. 51

Κατάλογος Πινάκων

Πίνακας 3-1: Πλήθος χαρακτηριστικών πριν και κατά την χρήση του αλγορίθμου K-μέσων..... 20
Πίνακας 4-1: Μέτρα θέσης και διασποράς για τα ποσοτικά χαρακτηριστικά του συνόλου δεδομένων Auto MPG..... 21
Πίνακας 4-2: Αποτελέσματα επιδόσεων για το σύνολο Auto MPG..... 27
Πίνακας 4-3: Αποτελέσματα ανάλυσης της διακύμανσης για το σύνολο δεδομένων Auto MPG..... 28
Πίνακας 5-1: Αποτελέσματα για το υποσύνολο δοκιμής για το σύνολο δεδομένων Car Evaluation..... 32
Πίνακας 5-2: Αποτελέσματα πολλαπλής επικύρωσης 10 φορών για το σύνολο δεδομένων Car Evaluation..... 33
Πίνακας 5-3: Ανάλυση διακύμανσης προβλέψεων χωρίς την χρήση K-Means για το σύνολο δεδομένων Car Evaluation..... 34
Πίνακας 5-4: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Car Evaluation..... 35
Πίνακας 6-1: Αποτελέσματα για το υποσύνολο δοκιμής για το σύνολο δεδομένων Bank Marketing..... 42
Πίνακας 6-2: Αποτελέσματα πολλαπλής επικύρωσης 10 φορών για το σύνολο δεδομένων Bank Marketing..... 42
Πίνακας 6-3: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Bank Marketing..... 44
Πίνακας 6-4: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Bank Marketing..... 44
Πίνακας 7-1: Αποτελέσματα για το υποσύνολο δοκιμής για το σύνολο δεδομένων Default of credit card clients..... 49
Πίνακας 7-2: Αποτελέσματα πολλαπλής επικύρωσης 10 φορών για το σύνολο δεδομένων Default of credit card clients..... 50
Πίνακας 7-3: Ανάλυση διακύμανσης προβλέψεων χωρίς την χρήση K-Means για το σύνολο δεδομένων Default of credit card clients..... 51
Πίνακας 7-4: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Default of credit card clients..... 52

Πίνακας 8-1: Σύνοψη αποτελεσμάτων για την ευστοχία κατά την πολλαπλή επικύρωση 10 φορών. 53

1 Εισαγωγή

Η μηχανική μάθηση (Machine Learning) είναι ένας κλάδος της Πληροφορικής που εμφανίστηκε περίπου την δεκαετία του 1950. Σαν όρος αναφέρθηκε για πρώτη φορά από τον Άρθουρ Σάμιουελ (Samuel, 1959), σε μια μελέτη του για την δημιουργία ενός προγράμματος που θα μάθαινε μόνο του να παίζει ντάμα. Από τότε και έπειτα, άρχισε ολοένα και περισσότερο να τραβάει την προσοχή σαν πεδίο μελέτης, ενώ από την δεκαετία του '90, άρχισε να χρησιμοποιείται σιγά σιγά και πέρα από το ερευνητικό επίπεδο, για την κατασκευή προγραμμάτων που μπορούν να μαθαίνουν από τα δεδομένα, καθώς και για την απλοποίηση και βελτιστοποίηση διαδικασιών.

Η μηχανική μάθηση πλέον χρησιμοποιείται σε μια πληθώρα εφαρμογών όπως πχ. στην κατηγοριοποίηση αντικειμένων του πραγματικού κόσμου, στο φιλτράρισμα email, στην οπτική αναγνώριση χαρακτήρων (OCR), στην οικονομία κ.α. Επιπλέον, μπορεί να χρησιμοποιηθεί και για την ανακάλυψή και εξαγωγή χρήσιμων προτύπων που κρύβονται μέσα σε μια συλλογή από δεδομένα, μέσω μιας διαδικασίας που ονομάζεται εξόρυξη δεδομένων (data mining) (North, 2012).

1.1 Πρόβλημα

Η μηχανική μάθηση έχει πλέον στην φαρέτρα της πάρα πολλούς αλγόριθμους κατηγοριοποίησης και παλινδρόμησης, όπως για παράδειγμα η λογιστική παλινδρόμηση, τα δέντρα απόφασης, τα νευρωνικά δίκτυα, ο αλγόριθμος Κ πλησιέστερων γειτόνων και πολλοί άλλοι. Οι αλγόριθμοι αυτοί, ορισμένες φορές, μπορούν να δώσουν περιορισμένα αποτελέσματα, όταν χρησιμοποιηθούν μόνοι τους. Έτσι, έχει προκύψει η ιδέα, ότι ο συνδυασμός περισσοτέρων από μίας μεθόδου κατηγοριοποίησης μπορούν να πετύχουν καλύτερα αποτελέσματα. Ο συνδυασμός αυτός μπορεί να περιλαμβάνει τόσο αλγορίθμους εποπτευόμενης μάθησης όσο και αλγορίθμους μη εποπτευόμενης μάθησης. Ωστόσο, δεν έχει δημιουργηθεί ένα σχήμα που να λειτουργεί πάντα καλά για κάθε είδους δεδομένα.

1.2 Στόχοι

Η εργασία αυτή έχει ως στόχο να δημιουργήσει τρία μοντέλα κατηγοριοποίησης για κάθε σύνολο δεδομένων με την χρήση των αλγορίθμων της λογιστικής παλινδρόμησης, δέντρου απόφασης και νευρωνικού δικτύου. Τα αποτελέσματα αυτών

των μοντέλων θα αποτελέσουν την βάση σύγκρισης των αποτελεσμάτων που θα προκύψουν από την προτεινόμενη μεθοδολογία με τον συνδυασμό του αλγορίθμου K-μέσων και των τριών προαναφερθέντων αλγορίθμων, που θα περιγραφεί στην συνέχεια, για τα ίδια σύνολα δεδομένων.

1.3 Διάρθρωση της μελέτης

Στο κεφάλαιο 2, θα γίνει μια σύντομη περιγραφή των αλγορίθμων κατηγοριοποίησης και παλινδρόμησης που θα χρησιμοποιηθούν, έτσι ώστε ο αναγνώστης να καταλάβει τα βασικά της λειτουργίας του κάθε αλγορίθμου. Παράλληλα θα περιγραφούν και κάποιες μετρικές μέθοδοι που θα χρησιμοποιηθούν για την αξιολόγηση της απόδοσης του κάθε αλγορίθμου. Στο κεφάλαιο 3 θα περιγραφεί ο τρόπος λειτουργίας της προτεινόμενης μεθοδολογίας. Στα κεφάλαια 4 έως και 7, θα μελετηθούν 4 συλλογές δεδομένων από το αποθετήριο μηχανικής μάθησης UCI Machine Learning Repository. Σε κάθε ένα από αυτά τα κεφάλαια, θα δοθεί μια σύντομη περιγραφή της συλλογής δεδομένων και των χαρακτηριστικών που περιλαμβάνουν. Στη συνέχεια μετά την περιγραφή της μεθοδολογίας που ακολουθήθηκε κατά την επεξεργασία των δεδομένων, θα παραθέτονται τα αποτελέσματα επιδόσεων του κάθε αλγορίθμου, ενώ ταυτόχρονα θα γίνεται και σύγκριση με τα αποτελέσματα της μεθοδολογίας που προτάθηκε. Στο κεφάλαιο 8 θα αναφερθούν διάφορα συμπεράσματα που θα προκύψουν περιορισμοί και όρια, ενώ θα προταθούν μελλοντικές μελέτες για έρευνα. Τέλος στο παράρτημα Α παρατίθεται ο κώδικας σε Python που χρησιμοποιήθηκε κατά την έρευνα, καθώς και οδηγίες για την αναπαραγωγή των αποτελεσμάτων της έρευνας.

2 Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο, θα περιγραφούν περιληπτικά οι αλγόριθμοι κατηγοριοποίησης και παλινδρόμησης, καθώς και οι διάφορες μετρικές μέθοδοι που θα χρησιμοποιηθούν, ώστε ο αναγνώστης να γνωρίσει τις εσωτερικές λειτουργίες τους.

2.1 Γραμμική Παλινδρόμηση (Linear Regression)

Η απλή γραμμική παλινδρόμηση είναι ένα μοντέλο με μια μόνο ανεξάρτητη μεταβλητή x, η οποία έχει μια σχέση με μια εξαρτημένη μεταβλητή y σε ευθεία γραμμή. Το μοντέλο αυτό μπορεί να περιγραφεί από την εξίσωση:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

όπου β_0 το σημείο τομής της ευθείας με τον άξονα y, β_1 η κλίση της ευθείας και ε μια τυχαία μεταβλητή λάθους. Τα λάθη θεωρούμε ότι έχουν μέσο όρο 0 και διακύμανση σ^2 που είναι θεωρητικά άγνωστη. Επιπλέον, θεωρούμε ότι τα λάθη δεν σχετίζονται μεταξύ τους, δηλαδή ότι η τιμή του ενός λάθους δεν σχετίζεται με την τιμή κάποιου άλλου. (MONTGOMERY, PECK, & VINING, 2012)

Οι παράμετροι β_0 και β_1 συνήθως καλούνται συντελεστές παλινδρόμησης και έχουν απλή και χρήσιμη ερμηνεία. Η κλίση β_1 είναι η αλλαγή στον μέσο όρο της κατανομής της εξαρτημένης μεταβλητης y για κάθε μοναδιαία αλλαγή της ανεξάρτητης μεταβλητής x. Αν οι τιμές του x περιλαμβάνουν το μηδέν, τότε η β_0 αντικατοπτρίζει τον μεσό όρο της κατανομής της μεταβλητής y όταν $x = 0$. Αν οι τιμές του x δεν περιλαμβάνουν το μηδέν τότε η β_0 δεν έχει κάποιά πρακτική ερμηνεία.

Ένα μοντέλο παλινδρόμησης το οποίο περιλαμβάνει περισσότερες από μία μεταβλητές, καλείται πολλαπλή γραμμική παλινδρόμηση. Γενικά μπορεί να περιγραφεί από την παρακάτω εξίσωση:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_\kappa x_\kappa + \varepsilon$$

με κ το πλήθος των ανεξάρτητων μεταβλητών και $\beta_j, με\ j = 0,1, \dots, \kappa$, να αποτελούν τους συντελεστές πολλαπλής παλινδρόμησης. Το μοντέλο αυτό περιγράφει ένα υπερεπίπεδο σε ένα χώρο κ διαστάσεων που δημιουργείται από κάθε μια μεταβλητή x_j.

Στα περισσότερα προβλήματα του πραγματικού κόσμου, οι τιμές των συντελεστών της παλινδρόμησης (οι τιμές β_j) και η διακύμανση σφάλματος σ^2 δεν είναι γνωστές και πρέπει να εκτιμηθούν από κάποιο δείγμα δεδομένων. Η ακριβής εξίσωση ή μοντέλο παλινδρόμησης, τυπικά χρησιμοποιείται για την πρόβλεψη μελλοντικών παρατηρήσεων της μεταβλητής απόκρισης y ή για την εκτίμηση του μέσου όρου απάντησης σε συγκεκριμένα επίπεδα εμπιστοσύνης του y.

2.2 Δένδρα Απόφασης (Decision Trees)

Οι αλγόριθμοι δέντρων αποφάσεων ID3 (Quinlan J.R., 1986) και C4.5 (Quinlan J. R., 2014) εφευρέθηκαν από τον John Ross Quinlan και οι δύο οικοδομούν ένα δέντρο από ένα σύνολο δεδομένων χρησιμοποιώντας την έννοια του κέρδους της πληροφορίας. Παράλληλα σχεδόν με τον Quinlan, ο Breiman δημιούργησε μια δική του εκδοχή των δέντρων απόφασης, που ονομάζονται Δέντρα Ταξινόμησης και παλινδρόμησης (C.A.R.T) (Breiman, Friedman, Stone, & Olshen, 1984) και είναι πολύ παρόμοια με αυτά που παράγει ο αλγόριθμος C4.5. Μια βασική διαφορά μεταξύ του αλγορίθμου C.A.R.T και C4.5 είναι ότι ο πρώτος υποστηρίζει και ποσοτικές μεταβλητές που μπορούν να υπολογιστούν (παλινδρόμηση) και δεν υπολογίζει σύνολα κανόνων. Η βιβλιοθήκη Skicit-learn χρησιμοποιεί μια βελτιστοποιημένη έκδοση του αλγόριθμου C.A.R.T.

Ας υποθέσουμε ένα σύνολο διανυσμάτων εκπαίδευσης $\mathcal{V} = \{v_1, v_2, \dots, v_k\}$ για ένα δείγμα διανυσμάτων κατηγοριοποίησης v_i με $v_i \epsilon R^n$ και ένα διάνυσμα στόχου y. Σε κάθε κόμβο M, για κάθε χαρακτηριστικό j και για κάθε όριο t_M, χωρίζουμε τα δεδομένα σε $L_{right}(\{j, t_M\})$ και $L_{left}(\{j, t_M\})$, όπου $L_{right}(\{j, t_M\}) = (x, y)|x_j > t_M$ και $L_{left}(\{j, t_M\}) = L - L_{right}$.

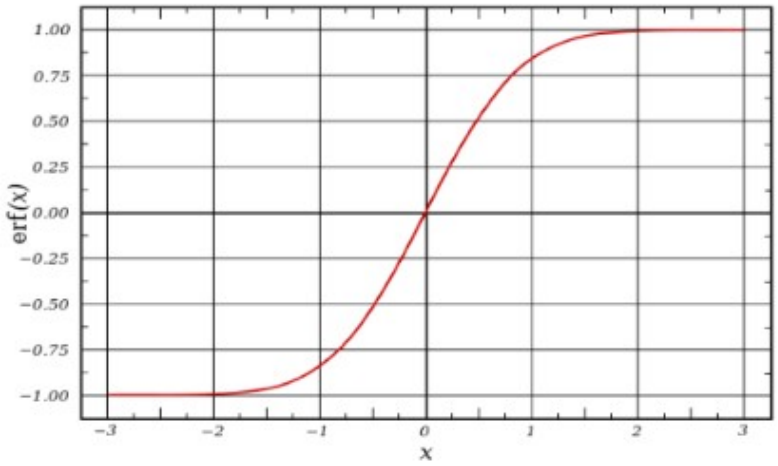
Η ποιότητα του διαχωρισμού μετράται μέσω μιας συνάρτησης $H()$, με τα μέτρα ποιότητας που χρησιμοποιούνται ως έξοδος της συνάρτησης αυτής, είναι συνήθως η εντροπία ή ο δείκτης Gini. Η επιλογή του μέτρου ποιότητας εξαρτάται από το είδος της εργασίας προς επίλυση, (ταξινόμηση ή παλινδρόμηση). Ο δείκτης Gini για τον κόμβο M μπορεί να υπολογιστεί ως εξής:

$$G(M,(j,t_M\})) = \frac{n_{left}}{M_M} H(L_{left}(\{j,t_M\})) + \frac{n_{right}}{M_M} H(L_{right}(\{j,t_M\}))$$

Με βάση τα παραπάνω, διαχωρίζουμε τα δεδομένα μας έτσι ώστε να ελαχιστοποιείται το $G(M,(j,t_M\}))$ και συνεχίζουμε την ίδια διαδικασία αναδρομικά για L_{left} και L_{right}, έως ότου ικανοποιηθούν τα κριτήρια πέρατος εκτέλεσης του αλγορίθμου. (scikit-learn Machine Learning in Python)

2.3 Λογιστική Παλινδρόμηση (Logistic Regression)

Η λογιστική παλινδρόμηση, παρά το όνομά της, είναι ένα γραμμικό μοντέλο για ταξινόμηση/κατηγοριοποίηση και όχι για παλινδρόμηση. Η πιο σημαντική διαφορά μεταξύ λογιστικής και γραμμικής παλινδρόμησης είναι το είδος της μεταβλητής απόκρισης, η οποία για την πρώτη μπορεί να είναι κατηγορική, (διακριτή ή ονομαστική), ενώ στη δεύτερη αποκλειστικά ποσοτική. Η λογιστική παλινδρόμηση ταξινομεί τις τιμές μιας μεταβλητής απόκρισης y, χρησιμοποιώντας τη θεωρία των πιθανοτήτων. Στο μοντέλο αυτό, όπου η μεταβλητή y συνήθως έχει δυαδικό χαρακτήρα (λαμβάνει δύο τιμές 0 ή 1), αν και υπάρχουν περιπτώσεις που η μεταβλητή y μπορεί να πάρει παραπάνω από δύο τιμές. Το αποτέλεσμα αυτής υπολογίζεται από ένα πλήθος μεταβλητών (διάνυσμα χαρακτηριστικών) που μπορεί να είναι κατηγορικές, διακριτές ή ποσοτικές. Στη στατιστική, η λογιστική παλινδρόμηση χρησιμοποιείται για την πρόβλεψη της πιθανότητας εμφάνισης ενός γεγονότος προσαρμόζοντας τις τιμές των ανεξάρτητων μεταβλητών πάνω στην εξίσωση της λογιστικής καμπύλης. (Πετρίδης, 2015)



Εικόνα 2-1: Λογιστική καμπύλη (Πετρίδης, 2015)

Η καμπύλη αυτή, όπως φαίνεται και στην παραπάνω εικόνα, έχει χαρακτηριστική σιγμοειδή μορφή. Η δυαδική λογιστική παλινδρόμηση έχει τη μορφή:

$$f(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

με z να είναι η μεταβλητή εισόδου. Ένα από τα πλεονεκτήματα της είναι το γεγονός ότι η μεταβλητή εισόδου μπορεί να λάβει οποιαδήποτε τιμή εντός του εύρους $\pm\infty$, με το αποτέλεσμα αυτής να περιορίζεται σε εύρος τιμών μεταξύ 0 και 1. Η μεταβλητή z εκφράζει επίσης το μέτρο της συνεισφοράς όλων των ανεξάρτητων μεταβλητών που συμμετέχουν στο μοντέλο και ορίζεται ως:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_\kappa X_\kappa\,, \qquad \mu\varepsilon\ i = 1,2,3,\dots,k$$

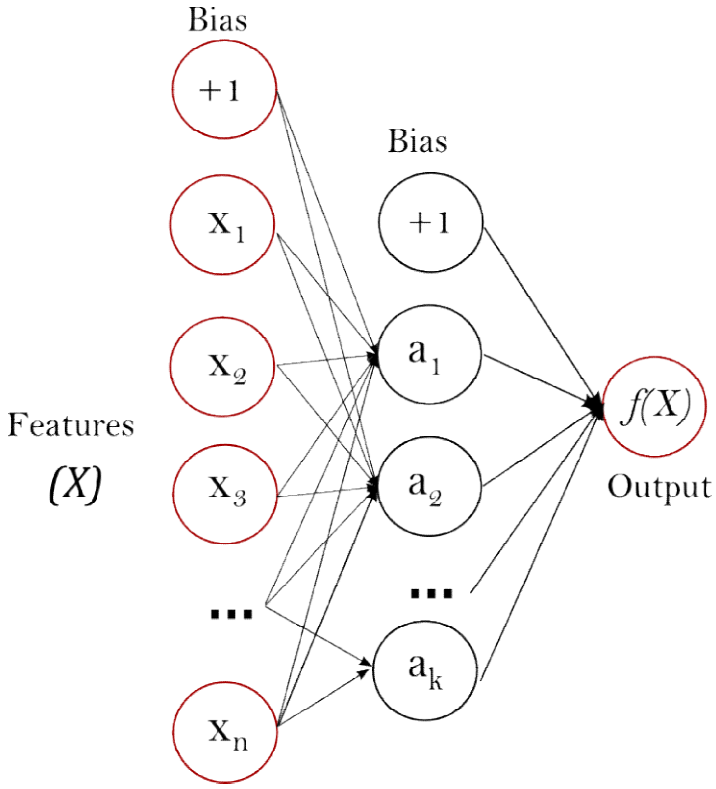
όπου β_0 είναι το ύψος της κλίσης της γραμμής παλινδρόμησης και ισούται με την τιμή z όταν οι τιμές όλων των ανεξάρτητων μεταβλητών X_κ ισούνται με 0, ενώ β_i είναι οι συντελεστές παλινδρόμησης που ο καθένας εκφράζει το μέγεθος συνεισφοράς της αντίστοιχης μεταβλητής. Έτσι ένα θετικό πρόσημο ενός συντελεστή μιας μεταβλητής, αντιστοιχεί σε αύξηση της πιθανότητας να συμβεί το γεγονός (1 ή θετική κλάση), ενώ ένα αρνητικό πρόσημο δηλώνει ότι η αντίστοιχη μεταβλητή μειώνει την πιθανότητα να συμβεί το γεγονός (0 ή αρνητική κλάση).

2.4 Νευρωνικά Δίκτυα (Neural Networks)

Τα Τεχνητά Νευρωνικά Δίκτυα είναι ένα υπολογιστικό μοντέλο εμπνευσμένο από τη φύση και το τρόπο που λειτουργούν τα βιολογικά νευρωνικά δίκτυα στον ανθρώπινο εγκέφαλο. Αποτελείται από μια συλλογή από κόμβους που ονομάζονται τεχνητοί νευρώνες. Για κάθε είσοδο δίνεται ένα βάρος για κάθε χαρακτηριστικό αυτής της εισόδου και συμπεριλαμβάνεται και το «σήμα» το οποίο έχει σταθερή τιμή (συνήθως -1 ή 1) και το οποίο αποτελεί είσοδο για όλους τους νευρώνες (bias).

Η μάθηση γίνεται εντός του νευρώνα αλλάζοντας τα βάρη σύνδεσης μετά από την επεξεργασία ενός τμήματος δεδομένων, με βάση την τιμή σφάλματος στην έξοδο σε σύγκριση με το αναμενόμενο αποτέλεσμα. Στη συνέχεια, το άθροισμα όλων των εισόδων πολλαπλασιάζεται με το βάρος τους και περνούν ως είσοδο σε μια συνάρτηση ενεργοποίησης που παράγει μία μόνο έξοδο, η οποία αποτελεί και την τελική έξοδο του μοντέλου. (Artificial Intelligence & Intelligent Systems group)

Μετά από έρευνα των (Minsky & Papert, 1969), αποδείχτηκε πως ένας μεμονωμένος νευρώνας είναι ικανός να λύσει μόνο ορισμένα γραμμικά προβλήματα. Αυτό οδήγησε στην χρήση πολλών επιπέδων που περιέχουν περισσότερους από έναν νευρώνες σε κάθε επίπεδο.



Εικόνα 2-2: Δίκτυο πολυεπίπεδων νευρώνων (scikit-learn Machine Learning in Python)

Το πιο αριστερό επίπεδο (όπως φαίνεται στην παραπάνω εικόνα) ονομάζεται επίπεδο εισόδου και αποτελείται από ένα σύνολο νευρώνων $\{x_i|x_1, x_2, \dots, x_n\}$ που αναπαριστούν ένα διάνυσμα χαρακτηριστικών εισόδου. Κάθε νευρώνας στο ενδιάμεσο κρυφό επίπεδο, μετασχηματίζει τις τιμές από το προηγούμενο επίπεδο σε ένα άθροισμα των επιμέρους εισόδων, μετά τον πολλαπλασιασμό τους με τους συντελεστές βαρύτητας όπως παρακάτω:

$$\sum_j w_j x_{j,k} = w_1 x_1 + w_2 x_2 + \cdots + w_k x_k + w_0$$

Στη συνέχεια το άθροισμα αυτό περνά μέσα από μια μη γραμμική συνάρτηση ενεργοποίησης $g(\cdot): R^n \to R^m$, όπως πχ. βηματική συνάρτηση, συνάρτηση προσήμου, σιγμοειδής συνάρτηση, γραμμική συνάρτηση κα. Το επίπεδο εξόδου λαμβάνει τις τιμές από το τελευταίο κρυφό επίπεδο και τις μετασχηματίζει στις τιμές εξόδου του δικτύου.

Τα πολυεπίπεδα νευρωνικά δίκτυα εκπαιδεύονται με εποπτευόμενο τρόπο χρησιμοποιώντας τον αλγόριθμο οπίσθιας διάδοσης (Li, Cheng, Shi, & Huang, 2012), ο οποίος βασίζεται στον κανόνα μάθησης διόρθωσης του λάθους. Η διαδικασία της πίσω διάδοσης του λάθους αποτελείται από δύο περάσματα διαμέσου των διαφορετικών επιπέδων του δικτύου, ένα προς τα εμπρός και ένα προς τα πίσω.

Στο εμπρός πέρασμα, ένα διάνυσμα εισόδου εφαρμόζεται στους νευρώνες εισόδου του δικτύου. Η επίδραση του, διαδίδεται δια μήκος του δικτύου από επίπεδο σε επίπεδο και παράγεται ένα σύνολο από τις εξόδους ως η πραγματική απόκριση του δικτύου. Κατά τη διάρκεια του εμπρός περάσματος τα βάρη του δικτύου είναι σταθερά. Κατά την διάδοση προς τα πίσω, η πραγματική απόκριση του δικτύου αφαιρείται από την επιθυμητή απόκριση για την παραγωγή ενός τιμής λάθους. Η τιμή του λάθους κινείται προς τα πίσω στο δίκτυο και τα βάρη προσαρμόζονται ανάλογα.

2.5 Αλγόριθμος K-μέσων

Ο αλγόριθμος K-μέσων συγκεντρώνει τα δεδομένα προσπαθώντας να διαχωρίσει τα δείγματα σε ομάδες (συμπλέγματα) K ίσων διακυμάνσεων, ελαχιστοποιώντας ένα κριτήριο που είναι γνωστό ως αδράνεια ή άθροισμα τετραγώνων εντός της ομάδας. Αυτός ο αλγόριθμος απαιτεί από πριν τον προσδιορισμό από τον χρήστη του αριθμού των ομάδων. Μπορεί να χρησιμοποιηθεί για δεδομένα που περιέχουν πολύ μεγάλο αριθμό δειγμάτων και έχει χρησιμοποιηθεί σε μεγάλο εύρος εφαρμογών σε πολλούς διαφορετικούς τομείς.

Ο αλγόριθμος αυτός, διαιρεί ένα σύνολο N δειγμάτων σε K ομάδες, καθεμία από τις οποίες περιγράφεται από τον μέσο όρο των δειγμάτων στο σύμπλεγμα. Οι μέσοι όροι των δειγμάτων στο σύμπλεγμα ονομάζονται κεντροειδή, τα οποία δεν είναι απαραίτητο να είναι σημεία που ανήκουν στο N, αν και ζουν στον ίδιο χώρο. Ο αλγόριθμος K-μέσων έχει ως στόχο να επιλέξει τα κεντροειδή που ελαχιστοποιούν ένα κριτήριο αδράνειας ή το σύνολο των τετραγώνων μεταξύ των ομάδων:

$$\sum_{i=0}^{n} \min_{\mu_j \epsilon\, C} (\left|x_i - \mu_j\right|^2)$$

Η αδράνεια μπορεί να θεωρηθεί ως ένα μέτρο του πώς είναι εσωτερικά τα συμπλέγματα. Ωστόσο εμφανίζει διάφορα μειονεκτήματα όπως:

- Η αδράνεια κάνει την υπόθεση ότι οι συστάδες είναι κυρτές και ισοτροπικές, κάτι που δεν συμβαίνει πάντα. Ανταποκρίνεται ανεπαρκώς σε επιμήκεις συστάδες ή σε πολλαπλές χωρικές τοπολογίες με ακανόνιστα σχήματα.
- Η αδράνεια δεν είναι μια κανονικοποιημένη μετρική μέθοδος: μπορούμε να πούμε ότι οι χαμηλότερες τιμές αδράνειας είναι καλύτερες, με το μηδέν να είναι η βέλτιστη τιμή, εντούτοις σε χώρους πολύ μεγάλων διαστάσεων, οι ευκλείδειες αποστάσεις τείνουν να διογκώνονται. Η εκτέλεση ενός αλγόριθμου μείωσης του πλήθους των διαστάσεων πριν από την ομαδοποίηση K-μέσων, μπορεί να μετριάσει αυτό το πρόβλημα και να επιταχύνει τους υπολογισμούς (scikit-learn Machine Learning in Python).

2.6 Μετρικές μέθοδοι

Η μηχανική μάθηση γενικά μπορεί να θεωρηθεί ένα πρόβλημα βελτιστοποίησης. Στόχος της, με την ευρεία έννοια, είναι να ελαχιστοποιήσει το σφάλμα που προκύπτει από μια γενική μέθοδο επίλυσης ενός προβλήματος. Το γεγονός αυτό αποτελεί τον ακρογωνιαίο λίθο της μηχανικής μάθησης και την βάση για την ανάπτυξη καλύτερων και πιο αποδοτικών αλγορίθμων. Οι αλγόριθμοι που θα χρησιμοποιήσουμε σε αυτή την εργασία ανήκουν σε 2 μεγάλες οικογένειες προβλημάτων: την παλινδρόμηση και την κατηγοριοποίηση. Κάθε πρόβλημα έχει τις δικές του μετρικές μεθόδους ποσοτικοποίησης της απόδοσης και του σφάλματος. Παρακάτω θα κάνουμε μια επισκόπηση των μετρικών μεθόδων που θα χρησιμοποιηθούν στην εργασία αυτή.

2.6.1 Μετρικές μέθοδοι κατά την παλινδρόμηση

2.6.1.1 Μέσο Απόλυτο Σφάλμα (ΜΑΣ)

Το Μέσο Απόλυτο Σφάλμα (Mean Absolute Error - MAE) είναι το πιο απλό μετρικό σφάλμα παλινδρόμησης στο να το κατανοήσουμε. Για κάθε τιμή που προβλέπει ένα μοντέλο παλινδρόμησης, θα υπολογίσουμε την διαφορά ανάμεσα της και στην αντίστοιχη πραγματική τιμή για κάθε σημείο δεδομένων, λαμβάνοντας μόνο την απόλυτη τιμή του καθενός έτσι ώστε οι αρνητικές και θετικές διαφορές να μην αλληλοακυρώνονται. Στη συνέχεια λαμβάνουμε τον μέσο όρο όλων αυτών των διαφορών (Mishra). Στην πράξη, το ΜΑΣ περιγράφει το τυπικό (μέσο) μέγεθος των διαφορών. Η εξίσωση υπολογισμού του ΜΑΣ φαίνεται παρακάτω:

$$MA\Sigma = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2.6.1.2 Μέσο Τετραγωνικό Σφάλμα (ΜΤΣ)

Το μέσο τετραγωνικό σφάλμα (Mean Squared Error - MSE) είναι ακριβώς όπως το ΜΑΕ, αλλά αντί να χρησιμοποιήσουμε την απόλυτη τιμή της διαφοράς, χρησιμοποιούμε το τετράγωνο της διαφοράς πριν από την άθροιση τους. Μπορούμε να δούμε αυτή τη διαφορά στην παρακάτω εξίσωση:

$$MT\Sigma = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Επειδή τετραγωνίζουμε την διαφορά, το ΜΤΣ θα είναι σχεδόν πάντα μεγαλύτερο από το ΜΑΣ. Για το λόγο αυτό, δεν μπορούμε να συγκρίνουμε άμεσα το ΜΤΣ με το ΜΑΣ . Η επίδραση του τετραγωνικού όρου στην εξίσωση ΜΤΣ είναι πιο εμφανής με την ύπαρξη υπερβολικών τιμών στα δεδομένα μας. Ενώ κάθε υπολειπόμενο σε ΜΑΕ συνεισφέρει αναλογικά στο συνολικό σφάλμα, το σφάλμα αυξάνεται σε τετραγωνικό ρυθμό στο ΜΤΣ. Αυτό σημαίνει ότι οι υπερβολικά υψηλές τιμές στα δεδομένα, συμβάλλουν σε πολύ υψηλότερο βαθμό στο συνολικό σφάλμα του ΜΤΣ από ότι στο ΜΑΕ. Παρομοίως, το μοντέλο, θα «τιμωρηθεί» περισσότερο για τις προβλέψεις που διαφέρουν σημαντικά από την αντίστοιχη πραγματική τιμή. Αυτό σημαίνει ότι οι μεγάλες διαφορές μεταξύ πραγματικών τιμών και των τιμών που υπολογίζει το μοντέλο, τιμωρούνται περισσότερο στο ΜΤΣ από ότι στο ΜΑΕ (Mishra).

2.6.1.3 Τετραγωνική ρίζα Μέσου Τετραγωνικού Σφάλματος

Μια άλλη μέτρηση σφάλματος είναι η τετραγωνική ρίζα μέσου τετραγωνικού σφάλματος (Root Mean Squared Error - RMSE). Όπως υποδηλώνει το όνομα, είναι η τετραγωνική ρίζα του ΜΤΣ. Επειδή το ΜΤΣ είναι υψωμένο στο τετράγωνο, οι μονάδες του δεν ταιριάζουν με αυτές του ΜΑΕ. Για τον λόγο αυτό χρησιμοποιούμε συχνά το ΡΜΤΣ για να μετατρέψουμε τη μέτρηση λάθους ξανά σε παρόμοιες μονάδες με το ΜΑΣ, διευκολύνοντας έτσι την ερμηνεία του. Δεδομένου ότι τα ΜΤΣ και ΡΜΤΣ έχουν υψωμένες στο τετράγωνο τις διαφορές από τις πραγματικές τιμές, επηρεάζονται παρομοίως από τα υπερβολικά υψηλά χαρακτηριστικά των δεδομένων (Mishra).

2.6.2 Μετρικές μέθοδοι κατά την κατηγοριοποίηση

2.6.2.1 Ευστοχία (Accuracy)

Η ευστοχία είναι μία μέτρηση για την αξιολόγηση μοντέλων κατηγοριοποίησης. Γενικά μιλώντας, η ευστοχία είναι το κλάσμα του αριθμού των σωστών προβλέψεων που έλαβε το μοντέλο μας προς το σύνολο των προβλέψεων. Πιο συγκεκριμένα, η ευστοχία έχει τον ακόλουθο ορισμό:

$$Ευστοχία = \frac{Αριθμός\ Σωστών\ Προβλέψεων}{Συνολικός\ Αριθμός\ Προβλέψεων}$$

Για τη δυαδική ταξινόμηση, η ευστοχία μπορεί επίσης να υπολογιστεί με όρους θετικών και αρνητικών ταξινομήσεων ως εξής:

$$Ευστοχία = \frac{TP + TN}{TP + TN + FP + FN}$$

όπου TP = Αληθινά Θετικά, TN = Αληθινά Αρνητικά, FP = Ψευδώς Θετικά και FN = Ψευδώς Αρνητικά (Machine Learning Crash Course).

2.6.2.2 Ακρίβεια (Precision)

Η ακρίβεια (Precision) επιχειρεί να απαντήσει στο εξής ερώτημα: *Ποιο ποσοστό των θετικών ταξινομήσεων ήταν σωστό;* Αυτού του είδους η ακρίβεια καθορίζεται από την παρακάτω σχέση (Machine Learning Crash Course) :

$$Ακρίβεια\ = \frac{TP}{TP + FP}$$

2.6.2.3 Ανάκληση (Recall)

Η ανάκληση σε αντίθεση με την ακρίβεια (precision) απαντά στο ερώτημα: *Ποιό ποσοστό των πραγματικών θετικών ταξινομήθηκε σωστά ως θετικό;* Η ανάκληση μπορεί να υπολογιστεί με τον εξής τρόπο (Machine Learning Crash Course):

$$\text{Ανάκληση}\ = \frac{TP}{TP + FN}$$

2.6.2.4 F_1

Για να αξιολογήσουμε πλήρως την αποτελεσματικότητα ενός μοντέλου, πρέπει να εξετάσουμε τόσο την ακρίβεια όσο και την ανάκληση. Δυστυχώς, η ακρίβεια και η ανάκληση έρχονται συχνά σε αντίθεση, δηλαδή η βελτίωση της ακρίβειας συνήθως μειώνει την ανάκληση και αντίστροφα. Γενικά, ένα μοντέλο που ξεπερνάει ένα άλλο μοντέλο σε ακρίβεια και ανάκληση είναι πιθανά το καλύτερο μοντέλο. Προφανώς, θα πρέπει να διασφαλίσουμε ότι η σύγκριση γίνεται σε ένα σημείο ακρίβειας / ανάκλησης που είναι χρήσιμο στην πράξη για να έχει νόημα αυτό (Shung).

Η βαθμολογία F_1 είναι ο αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης, με την καλύτερη τιμή της στο 1 (άριστη ακρίβεια και ανάκληση) και χειρότερη στο 0. Δίνεται από την σχέση:

$$F_1 = 2 * \frac{Ακρίβεια * \text{Ανάκληση}}{Ακρίβεια + \text{Ανάκληση}}$$

Χρησιμοποιείται όταν θέλουμε να συγχωνεύσουμε την ακρίβεια και την ανάκληση σε μια μετρική μέθοδο για να συγκρίνουμε 2 ή περισσότερα μοντέλα. Παράλληλα μπορεί να χρησιμοποιηθεί όταν επιθυμούμε μια ισορροπία ανάμεσα στην ακρίβεια και την ανάκληση πράγμα που είναι συχνό σε περιπτώσεις ταξινομήσεων όπου τα ποσοστά των κλάσεων 0 και 1 είναι σε ανισορροπία.

2.6.2.5 Καμπύλη ROC

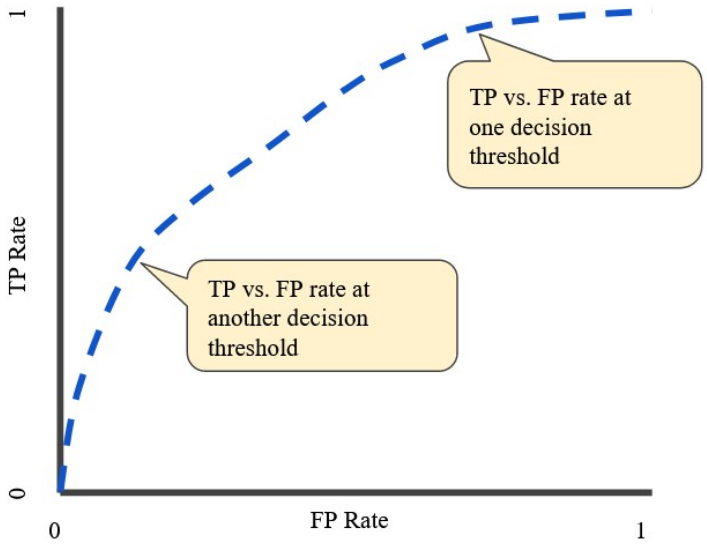
Η καμπύλη ROC (χαρακτηριστική καμπύλη λειτουργίας του δέκτη) είναι ένα γράφημα που δείχνει την απόδοση ενός μοντέλου ταξινόμησης σε όλα τα όρια ταξινόμησης (Machine Learning Crash Course). Αυτή η καμπύλη ορίζει δύο παραμέτρους:

- Αληθινό θετικό ποσοστό (True Positive Rate)
- Ψευδώς θετικό ποσοστό (False Positive Rate)

Ο πραγματικός θετικός ρυθμός (TPR) είναι συνώνυμο της ανάκλησης και ως εκ τούτου ορίζεται όπως είδαμε παραπάνω. Ο ψευδώς θετικός ρυθμός ορίζεται ως εξής:

$$F\text{PR} = \frac{FP}{FP + TN}$$

Μία καμπύλη ROC απεικονίζει το TPR έναντι του FPR σε όλα τα όρια ταξινόμησης. Η μείωση του ορίου ταξινόμησης κατατάσσει περισσότερα στοιχεία ως θετικά, αυξάνοντας έτσι τόσο τα ψευδώς θετικά όσο και τα αληθώς θετικά. Το παρακάτω σχήμα δείχνει μια τυπική καμπύλη ROC.

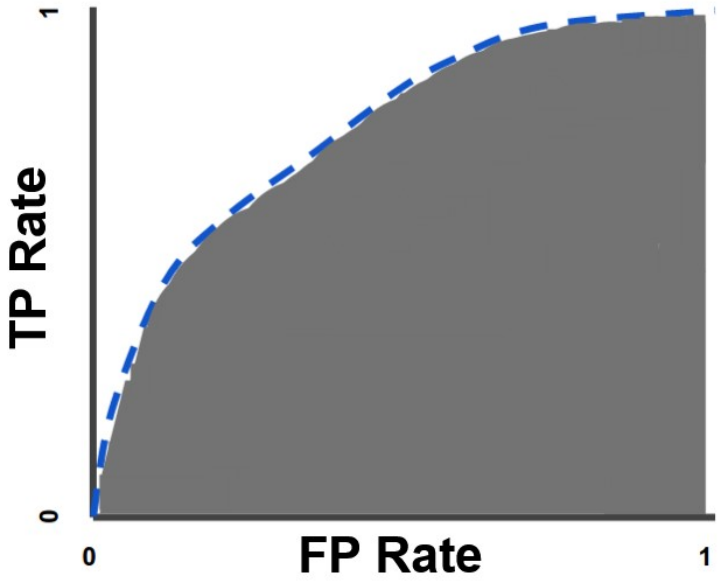


Εικόνα 2-3: Διάγραμμα καμπύλης ROC.

Για να υπολογίσουμε τα σημεία σε μια καμπύλη ROC, θα μπορούσαμε να αξιολογήσουμε ένα μοντέλο λογιστικής παλινδρόμησης πολλές φορές με διαφορετικά κατώτατα όρια ταξινόμησης, αλλά αυτό θα ήταν αναποτελεσματικό. Ευτυχώς, υπάρχει ένας αποτελεσματικός αλγόριθμος με βάση τη ταξινόμηση που μπορεί να παρέχει αυτές τις πληροφορίες για εμάς, που ονομάζεται AUC.

2.6.2.6 AUC: Περιοχή κάτω από την καμπύλη ROC

Το AUC σημαίνει "Περιοχή κάτω από την καμπύλη ROC". Δηλαδή, η AUC μετρά ολόκληρη την δισδιάστατη περιοχή κάτω από ολόκληρη την καμπύλη ROC (είναι το ολοκλήρωμα της) από (0,0) έως (1,1) (Machine Learning Crash Course).



Εικόνα 2-4: Περιοχή κάτω από την καμπύλη ROC.

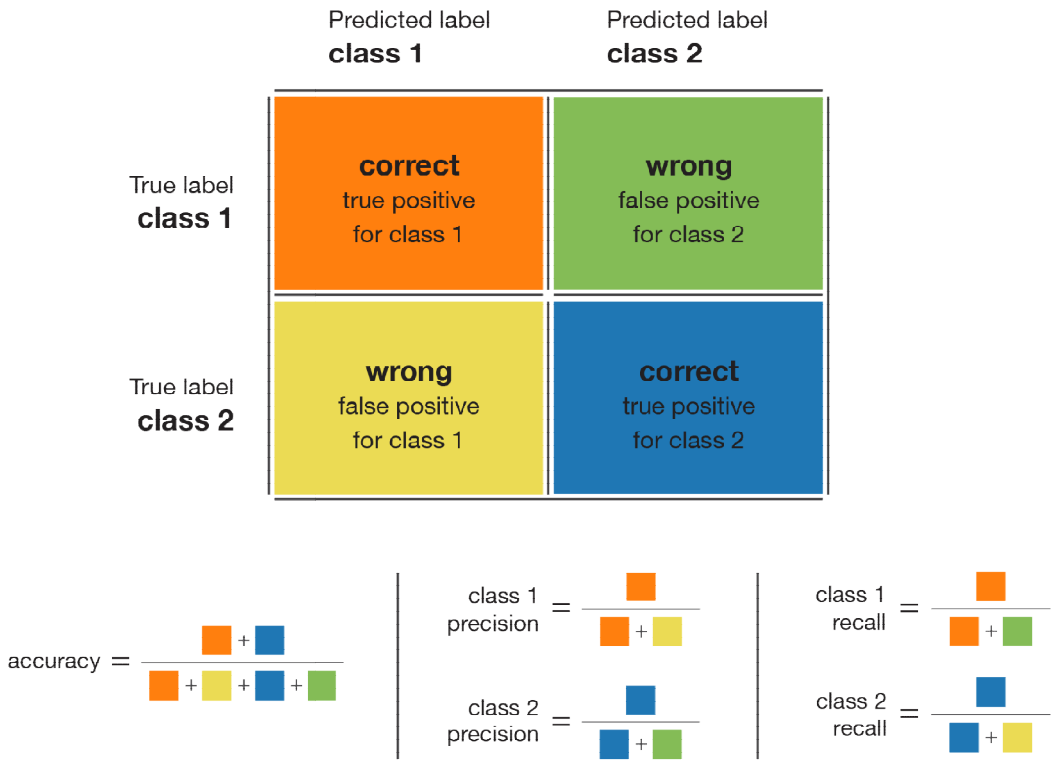
Η AUC παρέχει ένα συνολικό μέτρο απόδοσης σε όλα τα πιθανά όρια ταξινόμησης. Ένας τρόπος ερμηνείας της AUC είναι η πιθανότητα ότι το μοντέλο κατατάσσει ένα τυχαίο θετικό παράδειγμα περισσότερο από ένα τυχαίο αρνητικό παράδειγμα. Η περιοχή τιμών AUC κυμαίνεται από 0 έως 1. Ένα μοντέλο του οποίου οι προβλέψεις είναι 100% λανθασμένες έχει AUC 0, ενώ ενός του οποίου οι προβλέψεις είναι 100% σωστές, έχει AUC 1. Η AUC είναι επιθυμητή για τους ακόλουθους δύο λόγους:

- Η AUC είναι ανεξάρτητη με την κλίμακα των κλάσεων. Μετράει πόσο καλά ταξινομούνται οι προβλέψεις παρά τις απόλυτες τιμές τους.
- Η AUC είναι ανεξάρτητη από την ταξινόμηση. Μετρά την ποιότητα των προβλέψεων του μοντέλου ανεξάρτητα από το ποιο είναι το όριο ταξινόμησης.

2.6.3 Πίνακας σύγχυσης

Γενικά, μια καλή και συγχρόνως απλή μέτρηση που πρέπει πάντα να χρησιμοποιείται όταν αντιμετωπίζουμε ένα πρόβλημα ταξινόμησης είναι ο πίνακας σύγχυσης (scikit-learn Machine Learning in Python). Αυτή η μέτρηση δίνει μια ενδιαφέρουσα εικόνα για το πόσο καλά ταξινομεί παρατηρήσεις διαφόρων κλάσεων ένα μοντέλο. Έτσι, είναι ένα καλό σημείο εκκίνησης για οποιαδήποτε αξιολόγηση μοντέλου ταξινόμησης. Στην παρακάτω εικόνα (Εικόνα 2-5) φαίνεται ένας πίνακας σύγχυσης

καθώς και οι μετρικές που μπορούν να εξαχθούν από αυτόν, όπως τις είδαμε και πιο πάνω.



Εικόνα 2-5: Πίνακας σύγχυσης για πρόβλημα ταξινόμησης δύο κλάσεων.

2.6.4 Συγκριτική αξιολόγηση απόδοσης μοντέλων με ανάλυση της διακύμανσης

Πέρα από τους παραπάνω τρόπους υπολογισμού της απόδοσης του κάθε μοντέλου, μπορούμε να εφαρμόσουμε και μια συγκριτική μέθοδο η οποία εξετάζει αν υπάρχει στατιστικά σημαντική διαφορά ανάμεσα σε δύο μοντέλα που δοκιμάζονται στο ίδιο σύνολο δοκιμής. Πιο συγκεκριμένα, εφόσον όλα τα μοντέλα θα εκπαιδευτούν στο να εκτελούν προβλέψεις με το ίδιο υποσύνολο εκπαίδευσης για κάθε συλλογή δεδομένων, μπορούμε να εξετάσουμε το πόσο διαφορετικές προβλέψεις κάνουν (Refanidis & Samaras). Έτσι λοιπόν, αν υποθέσουμε ότι:

- H_0: Δεν υπάρχει διαφορά στατιστικά σημαντική στο σφάλμα πρόβλεψης δύο μοντέλων M_1 και M_2 που εκπαιδεύτηκαν με το ίδιο υποσύνολο δεδομένων .
- H_1: Υπάρχει διαφορά στατιστικά σημαντική στο σφάλμα πρόβλεψης δύο μοντέλων M_1 και M_2 που εκπαιδεύτηκαν με το ίδιο υποσύνολο δεδομένων.

τότε ορίζουμε :

- A το υποσύνολο δεδομένων δοκιμής με $card(A)=n$ το πλήθος στοιχείων του.
- mae_1 το ΜΑΣ για το μοντέλο M_1 στο υποσύνολο δοκιμής A.
- mae_2 το ΜΑΣ για το μοντέλο M_2 στο υποσύνολο δοκιμής A.
- Var_1 η διακύμανση των προβλέψεων για το μοντέλο M_1 στο υποσύνολο δοκιμής A.
- Var_2 η διακύμανση των προβλέψεων για το μοντέλο M_2 στο υποσύνολο δοκιμής A.

υπολογίζουμε το κλάσμα :

$$P = \frac{|mae_1 - mae_2|}{\sqrt{\frac{(Var_1 + Var_2)}{n}}}$$

Εφόσον $P \geq 2$, τότε η υπόθεση H_1 ισχύει. Αν όχι, τότε ισχύει η υπόθεση H_0. Οι παραπάνω υπολογισμοί ισχύουν για τα μοντέλα παλινδρόμησης. Αντίστοιχοι υπολογισμοί μπορούν να εφαρμοσθούν και σε μοντέλα ταξινόμησης χρησιμοποιώντας τις ίδιες υποθέσεις. Σε μια τέτοια περίπτωση, η παραπάνω σχέση μεταβάλλεται στην εξής:

$$P = \frac{|E_1 - E_2|}{\sqrt{\frac{(Var_1 + Var_2)}{n}}}$$

όπου η διακύμανση των προβλέψεων για το κάθε μοντέλο ταξινόμησης δίνεται από την σχέση:

$$Var_i = E_i * (1 - E_i)$$

με E_i το ποσοστό ευστοχίας των προβλέψεων του εκάστοτε μοντέλου στο υποσύνολο δοκιμής n πλήθους στοιχείων.

3 Χρήση του αλγορίθμου Κ-μέσων κατά την κατηγοριοποίηση

3.1 Εισαγωγή

Οι αλγόριθμοι ταξινόμησης και ομαδοποίησης έχουν αποδειχθεί επιτυχείς μεμονωμένα σε διαφορετικά πλαίσια. Και οι δύο έχουν τα δικά τους πλεονεκτήματα και περιορισμούς. Για παράδειγμα, αν και οι αλγόριθμοι ταξινόμησης είναι πιο ισχυροί από τις μεθόδους ομαδοποίησης στην πρόβλεψη των ετικετών κατηγορίας ενός δείγματος, δεν λειτουργούν καλά όταν υπάρχει έλλειψη επαρκών και αξιόπιστων δεδομένων για επισήμανση της κατηγορίας. Από την άλλη, παρόλο που οι αλγόριθμοι ομαδοποίησης δεν παράγουν περαιτέρω πληροφορίες παρά μόνο την ετικέτα των αντικείμενων που ομαδοποιούν, η πληροφορία αυτή μπορεί να χρησιμοποιηθεί για την πρόβλεψη ενός συνόλου άγνωστων αντικειμένων μαζί με ένα αλγόριθμο ταξινόμησης. Ως εκ τούτου, η συστηματική χρήση και των δύο αυτών τύπων αλγορίθμων μαζί μπορεί να οδηγήσει σε καλύτερη απόδοση πρόβλεψης (Alapati, 2016).

3.2 Επιθυμητά αποτελέσματα

Στόχος της μεθοδολογίας είναι, χρησιμοποιώντας τους παραπάνω αλγορίθμους, να πετύχουμε αποτελέσματα καλύτερα από αυτά που θα είχαμε αν χρησιμοποιούσαμε μόνο ένα αλγόριθμο ταξινόμησης ενώ ταυτόχρονα να επιλέξουμε τα καλύτερα και πιο αντιπροσωπευτικά χαρακτηριστικά από τα δεδομένα μας, μειώνοντας την πολυπλοκότητα και τον χρόνο εκπαίδευσης του μοντέλου που θα δημιουργήσουμε.

3.3 Προτεινόμενη Μεθοδολογία

Η προτεινόμενη μεθοδολογία χρησιμοποιεί τρεις αλγορίθμους:

1. Έναν αλγόριθμο επιλογής χαρακτηριστικών.
2. Τον αλγόριθμό ομαδοποίησης Κ-μέσων.
3. Τον αλγόριθμο λογιστικής παλινδρόμησης.

Πριν την έναρξη της μεθοδολογίας, τα δεδομένα προ-επεξεργάζονται και αφαιρείται το χαρακτηριστικό στόχος.

3.3.1 Επιλογή χαρακτηριστικών

Η επιλογή χαρακτηριστικών σε ένα σύνολο δεδομένων είναι μια διαδικασία αναγνώρισης των ποιο χρήσιμών χαρακτηριστικών τα οποία, αν χρησιμοποιηθούν δημιουργούν τα ίδια αποτελέσματα με αυτά του αρχικού συνόλου. Αρκετές φορές και ειδικά σε σύνολα δεδομένων με μεγάλο αριθμό χαρακτηριστικών, πολλά από αυτά μπορούν να θεωρηθούν ως άσχετα ή χαμηλής χρηστικότητας ή απλά ως θόρυβος, με αποτέλεσμα να μην παράγουν ικανοποιητικά αποτελέσματα κατά την ταξινόμηση αν χρησιμοποιηθούν. Γενικά τα χαρακτηριστικά που επιλέγονται θα πρέπει να μεγιστοποιούν την ικανότητα ενός ταξινομητή να δίνει τα καλύτερα αποτελέσματα.

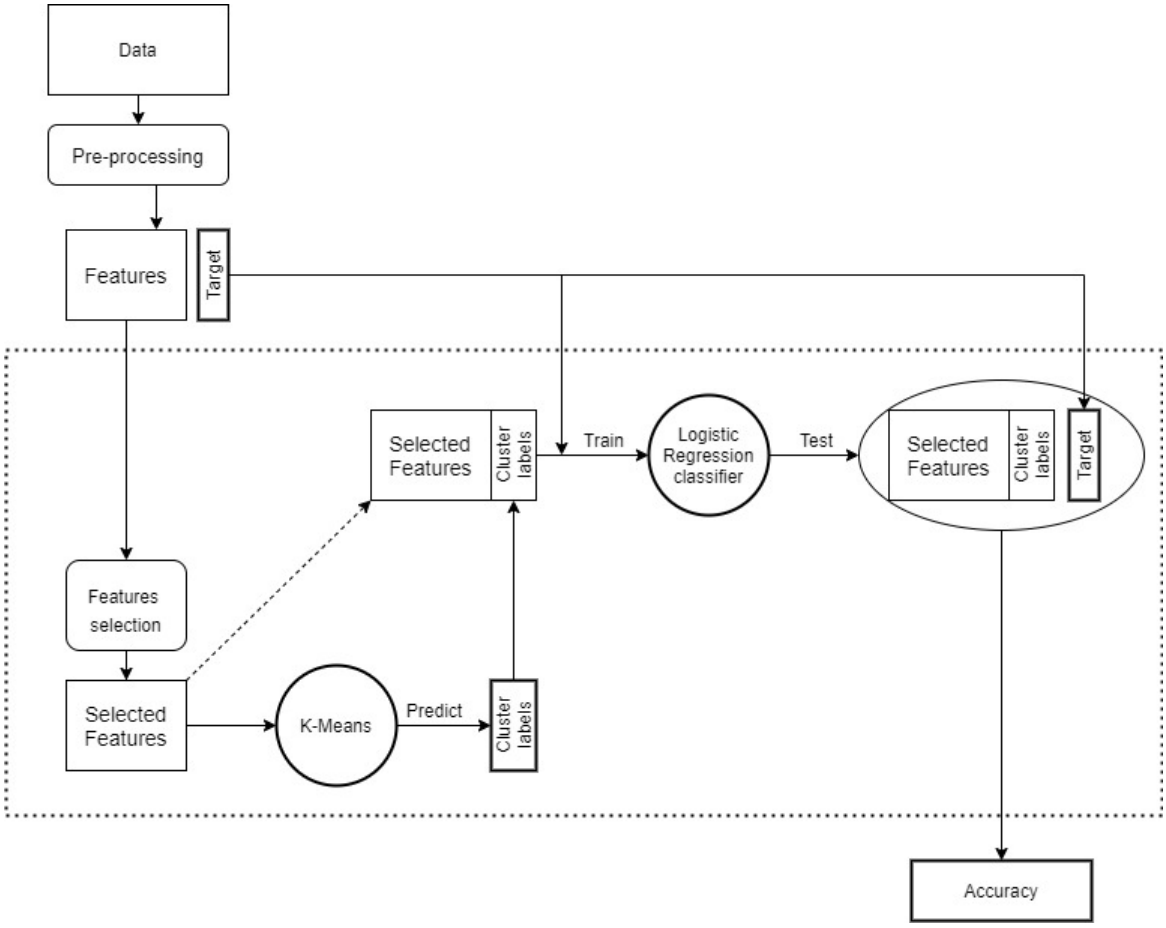
Κατά την μεθοδολογία, το κριτήριο επιλογής χαρακτηριστικών είναι το τεστ ανάλυσης της διακύμανσης (Ανάλυση διακύμανσης, 2019) κατά το οποίο, πραγματοποιείται έλεγχος υποθέσεων με στόχο να ανιχνευθούν εάν υπάρχουν διαφορές στις μέσες τιμές σε περισσότερα από δύο χαρακτηριστικά του συνόλου δεδομένων.

3.3.2 Ομαδοποίηση

Μετά την επιλογή των χαρακτηριστικών, εφαρμόζουμε τον αλγόριθμο ομαδοποίησης στο σύνολο των χαρακτηριστικών που έχουν επιλεχτεί από πριν. Για το πλήθος Κ των κεντροειδών που θέλουμε να δημιουργήσει ο αλγόριθμος Κ-μέσων, χρησιμοποιούμε το πλήθος των διαφορετικών (μοναδικών) τιμών που περιέχει το χαρακτηριστικό στόχος για το κάθε σύνολο δεδομένων. Όταν ολοκληρωθεί η ομαδοποίηση, η ετικέτα της ομάδος στην οποία ανήκει η κάθε εγγραφή, εισάγεται ως ένα νέο χαρακτηριστικό στα ήδη επιλεγμένα χαρακτηριστικά του προηγούμενου βήματος.

3.3.3 Ταξινόμηση

Τα ομαδοποιημένα δεδομένα του προηγούμενου βήματος χρησιμοποιούνται για την εκπαίδευση ενός ταξινομητή λογιστικής παλινδρόμησης. Για την μέτρηση της επίδοσης του ταξινομητή χρησιμοποιείται η ακρίβεια με την μέθοδο της επικύρωσης 10 φορών.



Εικόνα 3-1: Διάγραμμα ροής προτεινόμενης μεθοδολογίας.

3.4 Πειραματική διάταξη

Η εκτέλεση της παραπάνω μεθοδολογίας έγινε στα 3 από τις 4 σύνολα δεδομένων (ταξινόμηση) με επαναληπτικό τρόπο. Πιο συγκεκριμένα, δημιουργήθηκε ένας βρόγχος επανάληψης οποίος σε κάθε βήμα αύξανε κατά ένα το πλήθος των n καλύτερων χαρακτηριστικών που επιλεγόντουσαν και στο τέλος κατέγραφε την ακρίβεια που πετύχαινε ο ταξινομητής. Με την έξοδο από τον βρόγχο επανάληψης, προέκυπταν το πλήθος των κ καλύτερων χαρακτηριστικών μαζί με το καλύτερο σκορ. Τα κ αυτά χαρακτηριστικά χρησιμοποιούνταν κατά την εκπαίδευση και δοκιμή των μοντέλων και την εξαγωγή αποτελεσμάτων.

Για να μειωθεί ο χρόνος εκτέλεσης, ώστε να μην χρειαστεί να ελέγξουμε την ευστοχία των μοντέλων για όλο το πλήθος των χαρακτηριστικών, χρησιμοποιήθηκαν δύο παράμετροι, που συναντιούνται συνήθως στο πρώιμο σταμάτημα στα νευρωνικά δίκτυα. Η πρώτη παράμετρος αφορούσε το όριο αύξησης του καλύτερου σκορ σε κάθε γύρο. Η δεύτερη παράμετρος αφορούσε τις πόσες συνεχόμενες φορές το σκορ της κάθε

επανάληψης απέτυχε να ξεπεράσει το καλύτερο σκορ που καταγράφηκε στους προηγούμενους γύρους. Για την πρώτη παράμετρο το όριο αύξησης του σκορ ορίστηκε στο 0,001 ενώ για την δεύτερη παράμετρο το όριο πλήθους αποτυχημένων φορών να ξεπεραστεί το καλύτερο σκορ ήταν οι 10.

Οι παραπάνω τιμές επιλέχθηκαν μετά από αρκετούς πειραματισμούς και γενικά δίνουν αρκετά καλά και συνεπή αποτελέσματα και για τα 3 σύνολα δεδομένων, ενώ ταυτόχρονα καταλήγουν στο βέλτιστο αποτέλεσμα σχετικά γρήγορα και με σχετική σιγουριά ότι σχεδόν όλα τα χρήσιμα χαρακτηριστικά έχουν χρησιμοποιηθεί. Στον παρακάτω πίνακα φαίνονται το πλήθος των χαρακτηριστικών μετά τον μετασχηματισμό των δεδομένων και των αντίστοιχων χαρακτηριστικών που επιλέχθηκαν, όπως περιγράφτηκε παραπάνω.

Σύνολο Δεδομένων	**Πλήθος χαρακτηριστικών μετά τον μετασχηματισμό**	**Πλήθος επιλεγμένων χαρακτηριστικών από την μεθοδολογία (χωρίς τις ετικέτες ομαδοποίησης)**
Car Evaluation	21	18
Bank Marketing	81	14
Default of credit card clients	87	15

Πίνακας 3-1: Πλήθος χαρακτηριστικών πριν και κατά την χρήση του αλγορίθμου Κ-μέσων.

4 Μελέτη συνόλου δεδομένων Auto MPG

4.1 Εισαγωγή

Το σύνολο δεδομένων Auto mpg αφορά την αυτονομία καυσίμου για αυτοκίνητα που κινούνται εντός πόλης. Περιέχει δεδομένα οχημάτων που κατασκευάστηκαν ανάμεσα στο 1970 και το 1982 σε Αμερική, Ευρώπη και Ιαπωνία. Το σύνολο για πρώτη φορά χρησιμοποιήθηκε στην Έκθεση Αμερικανικής Στατιστικής Ένωσης (American Statistical Association Exposition) το 1983. Η πρώτη μελέτη του συνόλου διεξάχθηκε από τον Quinlan (Quinlan R. , 1993) κάνοντας χρήση διαφορών μεθόδων πρόβλεψης της αυτονομίας σε καύσιμο, όπως γραμμικής παλινδρόμησης, δέντρων απόφασης και νευρωνικών δικτύων.

4.2 Ανάλυση δεδομένων

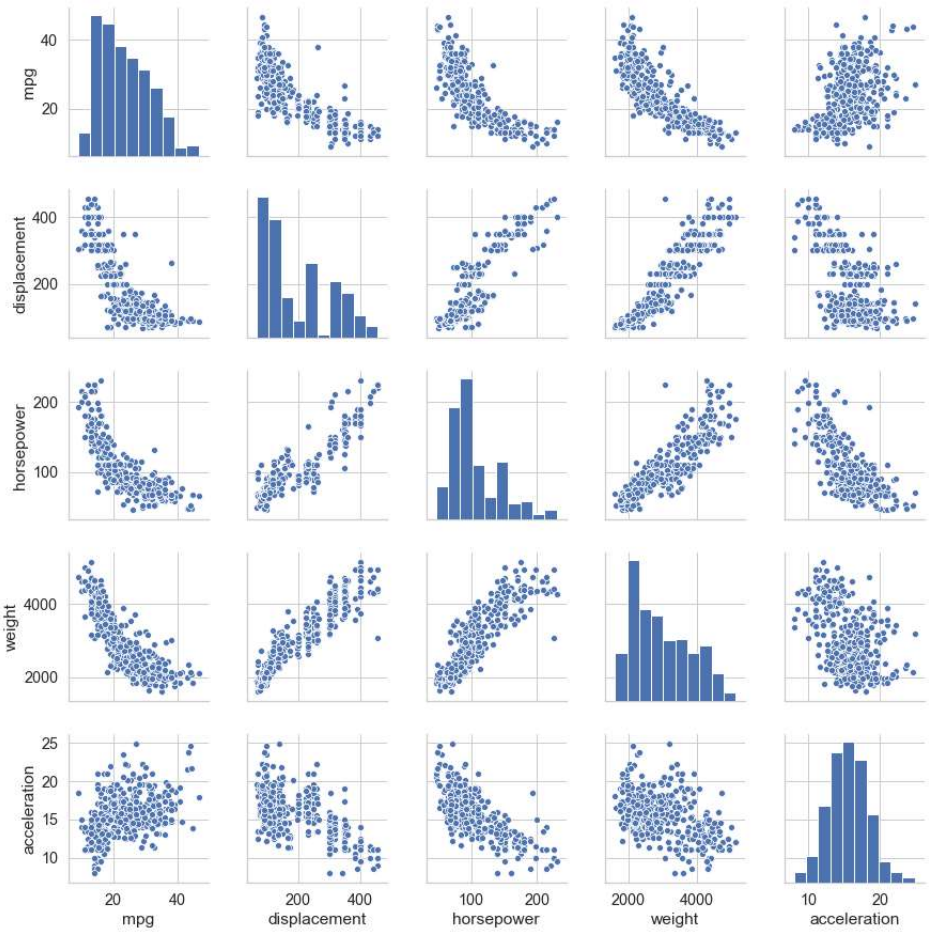
4.2.1 Εξέταση χαρακτηριστικών

Στον παρακάτω πίνακα μπορούμε να δούμε διάφορα μέτρα θέσης και διασποράς για τα χαρακτηριστικά των οχημάτων που έχουν συνεχείς τιμές.

	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
mpg	398.0	23.514573	7.815984	9.0	17.500	23.0	29.000	46.6
displacement	398.0	193.425879	104.269838	68.0	104.250	148.5	262.000	455.0
horsepower	398.0	104.478631	38.199261	46.0	76.000	95.0	125.000	230.0
weight	398.0	2970.424623	846.841774	1613.0	2223.750	2803.5	3608.000	5140.0
acceleration	398.0	15.568090	2.757689	8.0	13.825	15.5	17.175	24.8

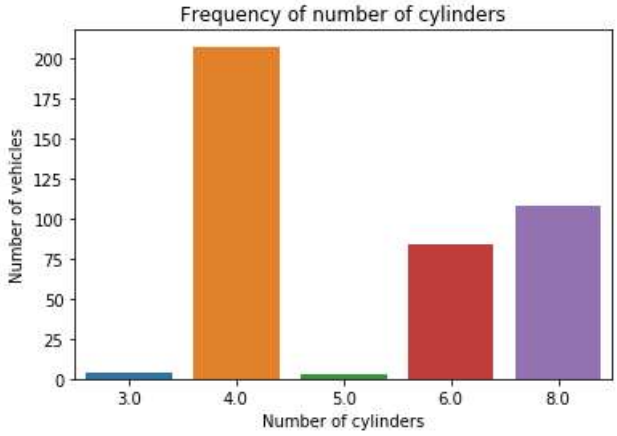
Πίνακας 4-1: Μέτρα θέσης και διασποράς για τα ποσοτικά χαρακτηριστικά του συνόλου δεδομένων Auto MPG.

Για τα ίδια χαρακτηριστικά, μπορούμε να δούμε και το πώς σχετίζονται μεταξύ τους, καθώς και την σχέση τους με το χαρακτηριστικό στόχος.(Εικόνα 4-1)



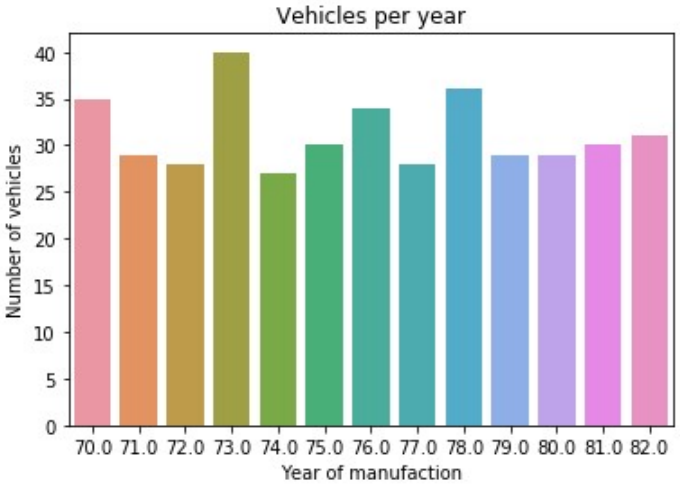
Εικόνα 4-1: Γραφική αναπαράσταση σχέσεων μεταξύ των ποσοτικών χαρακτηριστικών του συνόλου δεδομένων Auto MPG.

Για τα υπόλοιπα 3 χαρακτηριστικά με διακριτές τιμές, μπορούμε να χρησιμοποιήσουμε διαγράμματα με μπάρες για να τα κατανοήσουμε καλύτερα. Αρχικά μπορούμε να δούμε το πλήθος οχημάτων ανά αριθμό κυλίνδρων. (Εικόνα 4-2)



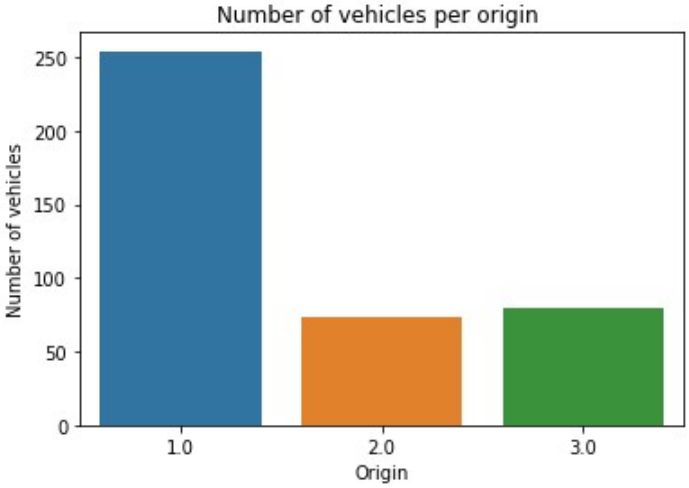
Εικόνα 4-2: Πλήθος οχημάτων ανά αριθμό κυλίνδρων.

Η πλειοψηφία των οχημάτων έχει 4 κυλίνδρους και ακολουθούν σε αριθμό τα οχήματα με 8 και 6 κυλίνδρους. Τα οχήματα με 3 ή 5 κυλίνδρους αποτελούν την μειοψηφία και είναι ελάχιστα σε αριθμό.



Εικόνα 4-3: Πλήθος οχημάτων ανά έτος κατασκευής.

Το έτος κατασκευής των οχημάτων είναι ανάμεσα στο 1970 και το 1982. Τα περισσότερα αυτοκίνητα κατασκευάστηκαν κατά σειρά τα έτη 1973, 1978, 1970 και 1976, με τα υπόλοιπα έτη να περιέχουν περίπου ίδιο αριθμό οχημάτων. (Εικόνα 4-3)

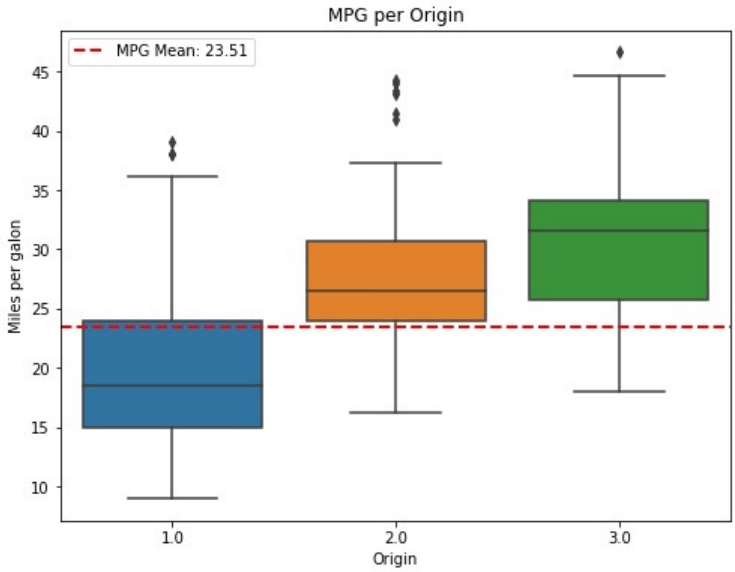


Εικόνα 4-4: Αριθμός οχημάτων ανά περιοχή προέλευσης.

Στο παραπάνω διάγραμμα βλέπουμε των αριθμό οχημάτων ανά περιοχή κατασκευής (Εικόνα 4-4). Η κωδικοποίηση της κάθε περιοχής είναι η εξής: το 1 αντιστοιχεί στην Αμερική, το 2 στην Ευρώπη και το 3 στην Ιαπωνία. Η πλειοψηφία των οχημάτων κατασκευάστηκαν στην Αμερική, ενώ ο υπόλοιπος αριθμός οχημάτων

μοιράζεται σε Ευρώπη και Ιαπωνία, με την τελευταία να υπερτερεί ελάχιστα σε αριθμό οχημάτων.

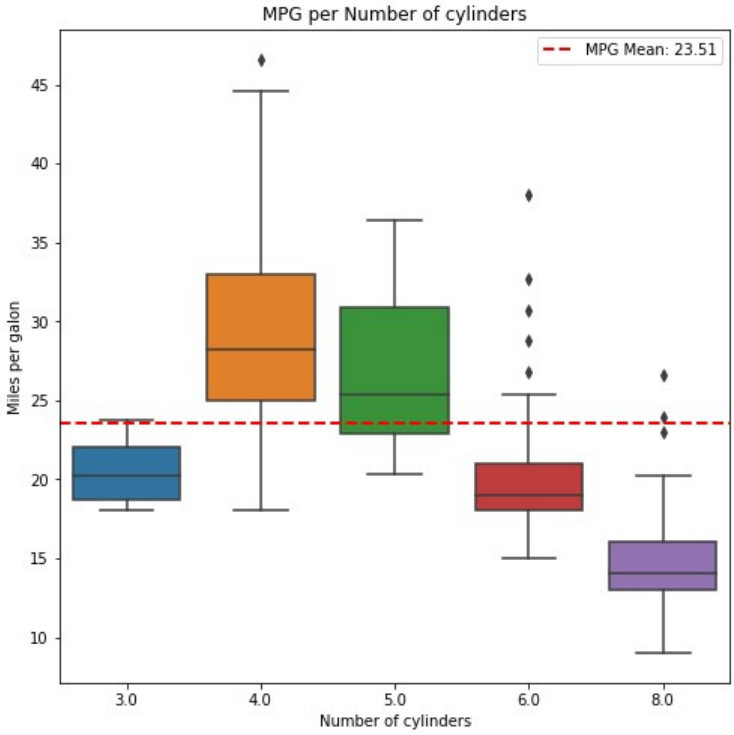
Έχοντας αναλύσει τα χαρακτηριστικά των οχημάτων με διακριτές τιμές, μπορούμε να εξετάσουμε πως αυτά αλληλεπιδρούν με το χαρακτηριστικό στόχος (*mpg*) το οποίο και επιθυμούμε να προβλέψουμε.



Εικόνα 4-5: Διάγραμμα αυτονομίας καυσίμου σε σχέση με την περιοχή προέλευσης.

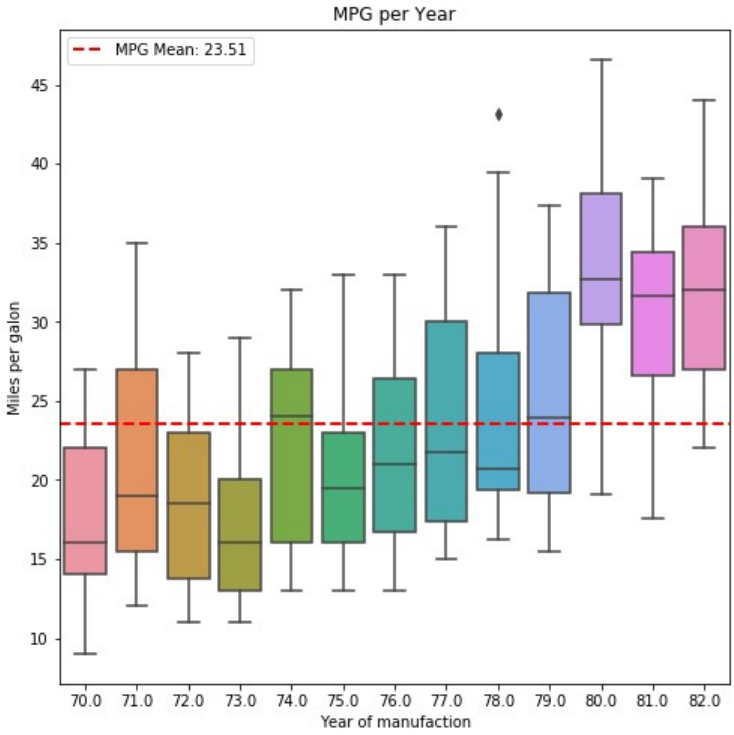
Είναι εμφανές, πως τα αυτοκίνητα που προέρχονται από την Αμερική έχουν την μεγαλύτερη κατανάλωση και άρα την μικρότερη αυτονομία σε σχέση με αυτά που προέρχονται από την Ευρώπη και την Ιαπωνία. Μάλιστα, περίπου το 75% των οχημάτων της Αμερικής έχουν κατανάλωση μεγαλύτερη από τον ολικό μέσο όρο όλων των οχημάτων.(Εικόνα 4-5)

Σε σχέση με τους κυλίνδρους, τα αυτοκίνητα τα οποία έχουν την καλύτερη αυτονομία είναι αυτά που έχουν 4 ή 5 κυλίνδρους κατά κύριο λόγο, κάτι που δείχνει να είναι λογικό αφού οι 3 κύλινδροι είναι λίγοι και απαιτούν περισσότερο καύσιμο για να έχουν ικανοποιητικές επιδόσεις, ενώ οι 6 και 8 κύλινδροι καταναλώνουν αθροιστικά περισσότερο καύσιμο. (Εικόνα 4-6)



Εικόνα 4-6: Διάγραμμα αυτονομίας καυσίμου ανά αριθμό κυλίνδρων.

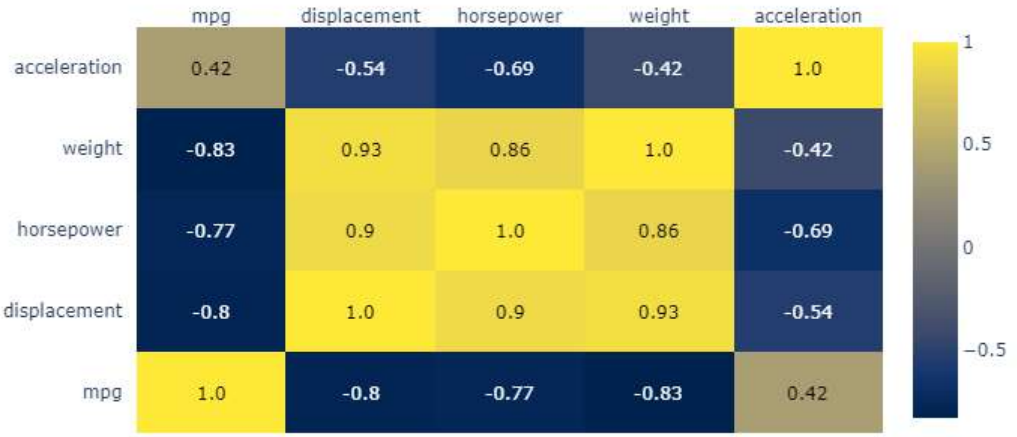
Όσον αφορά την εξέλιξη της αυτονομίας των οχημάτων με τα χρόνια, μπορούμε να εντοπίσουμε μια αυξητική τάση από το 1979 και μετά, λόγω της κρίσης πετρελαίου του 1979. (1979 oil crisis, 2020)



Εικόνα 4-7: Διάγραμμα αυτονομίας καυσίμου ανά έτος κατασκευής.

4.2.2 Διάγραμμα συσχετίσεων

Στο παρακάτω διάγραμμα μπορούμε να δούμε τις συσχετίσεις για κάθε χαρακτηριστικό με συνεχείς τιμές σε συνάρτηση με το χαρακτηριστικό στόχος αλλά και για κάθε ζεύγος χαρακτηριστικών.



Εικόνα 4-8: Πίνακας συσχετίσεων Pearson για το σύνολο δεδομένων Auto MPG.

Οι συσχετίσεις Pearson μπορούν να πάρουν τιμές από -1 έως 1 για κάθε ζεύγος χαρακτηριστικών, με την τιμή 1 να εκφράζει την πλήρη θετική γραμμική συσχέτιση ανάμεσα στα δύο χαρακτηριστικά, η τιμή -1 την πλήρη αρνητική γραμμική συσχέτιση ανάμεσα στα δύο χαρακτηριστικά και η τιμή 0 να εκφράζει την μη γραμμική συσχέτιση του ζεύγους χαρακτηριστικών.

Σε ένα πίνακα συσχετίσεων το επιθυμητό είναι τα χαρακτηριστικά να έχουν υψηλό κατά απόλυτη τιμή βαθμό συσχέτισης με το χαρακτηριστικό στόχος, ενώ παράλληλα να έχουν μηδενικό ή πολύ μικρό κατά απόλυτη τιμή βαθμό συσχέτισης μεταξύ τους.

4.3 Επεξεργασία δεδομένων

4.3.1 Καθαρισμός δεδομένων

Το σύνολο δεδομένων Auto MPG περιέχει 8 χαρακτηριστικά 398 οχημάτων μαζί με τα ονόματα των μοντέλων τους. Από τα 398 οχήματα, 8 δεν έχουν τιμές για το χαρακτηριστικό στόχος *mpg*, ενώ άλλα 6 δεν έχουν τιμή για το χαρακτηριστικό *horsepower*. Τα οχήματα χωρίς τιμές στο χαρακτηριστικό στόχος αφαιρέθηκαν από το σύνολο δεδομένων, ενώ για τα 6 οχήματα χωρίς τιμές στο χαρακτηριστικό *horsepower*,

αυτές συμπληρώθηκαν με τον μέσο όρο των υπολοίπων οχημάτων για το χαρακτηριστικό αυτό. Τέλος, δεν υπάρχουν διπλοεγγραφές στο σύνολο δεδομένων.

4.3.2 Μετασχηματισμός δεδομένων

Αρχικά έγινε η επιλογή των χαρακτηριστικών που θα συμμετέχουν στην εκπαίδευση των μοντέλων. Τα χαρακτηριστικά *horsepower* και *displacement* δεν θα χρησιμοποιηθούν, λόγω υψηλής συσχέτισης με τα χαρακτηριστικό *weight*. Το χαρακτηριστικό *origin* θα κωδικοποιηθεί σε 0 και 1 δημιουργώντας μία στήλη για κάθε μία μοναδική τιμή του χαρακτηριστικού αυτού. Ύστερα, όλα τα δεδομένα κανονικοποιούνται, ώστε κάθε χαρακτηριστικό να έχει μέσο όρο τιμών ίσο με μηδέν και τυπική απόκλιση ίση με ένα.

4.4 Αποτελέσματα

Τα δεδομένα χωρίστηκαν με τυχαίο τρόπο σε υποσύνολο εκπαίδευσης, που αποτελείται από το 70% των δεδομένων και σε υποσύνολο δοκιμής που θα περιέχει το υπόλοιπο 30% των δεδομένων. Κάθε αλγόριθμος εκπαιδεύτηκε με το ίδιο υποσύνολο εκπαίδευσης και δοκιμάστηκε τόσο έναντι του υποσυνόλου δόκιμης, όσο και με την μέθοδο της πολλαπλής επικύρωσης 10 φορών για το ΜΑΣ, ΜΤΣ ΡΜΤΣ. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

Models	MAE	MSE	RMSE	CV MAE	CV MSE	CV RMSE
Linear Regression	2.533	11.211	3.348	2.748	12.804	3.578
Decision Tree	2.407	12.784	3.576	2.733	14.801	3.847
Neural Network	1.924	6.794	2.607	2.451	11.271	3.357

Πίνακας 4-2: Αποτελέσματα επιδόσεων για το σύνολο Auto MPG.

Μπορούμε να παρατηρήσουμε ότι το νευρωνικό δίκτυο έχει πετύχει τα καλύτερα αποτελέσματα σε σχέση με τα άλλα δύο μοντέλα, τόσο στο υποσύνολο δοκιμής, όσο και κατά την πολλαπλή επικύρωση 10 φορών.

Αναλύοντας την διακύμανση των προβλέψεων που παράγουν και τα τρία μοντέλα, δεν εμφανίζονται στατιστικά σημαντικές διαφορές μεταξύ τους στις προβλέψεις που κάνουν , όπως μπορούμε να συμπεράνουμε από τον παρακάτω πίνακα:

Models	Linear Regression	Decision Tree	Neural Network
Linear Regression	-	0.017	0.706
Decision Tree	0.017	-	0.648
Neural Network	0.706	0.648	-

Πίνακας 4-3: Αποτελέσματα ανάλυσης της διακύμανσης για το σύνολο δεδομένων Auto MPG.

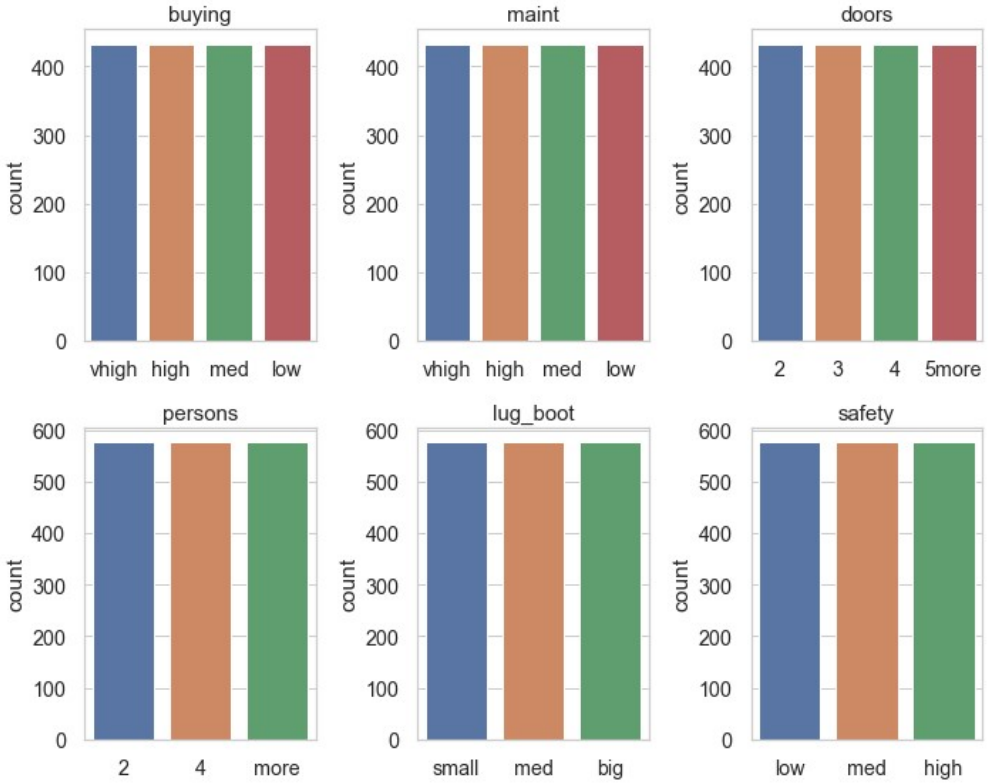
5 Μελέτη συνόλου δεδομένων Car Evaluation

5.1 Εισαγωγή

Το σύνολο δεδομένων Car Evaluation έχει προέλθει από ένα απλό ιεραρχικό μοντέλο απόφασης το όποιο δημιουργήθηκε από τους M.Bohanec, και V.Rajkovic. (Bohanec, Žnidaršič, Rajkovič, Bratko, & Zupan, 2013). Το μοντέλο αυτό κατατάσσει τα αυτοκίνητα σε 4 κατηγορίες, με βάση έξι ποιοτικά χαρακτηριστικά τους.

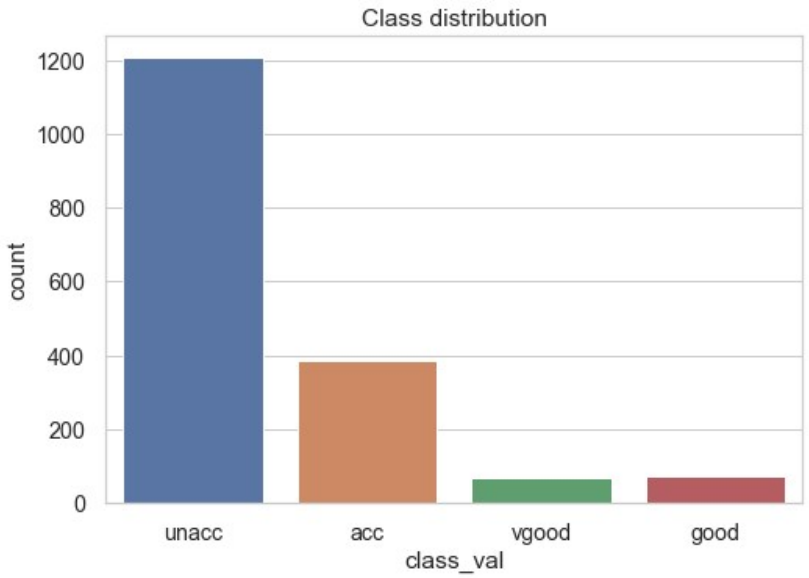
5.2 Ανάλυση δεδομένων

Όλα τα χαρακτηριστικά των οχημάτων είναι κατηγορικά. Κάθε χαρακτηριστικό περιγράφει για κάθε μοναδική τιμή του, ίδιο ποσοστό αυτοκινήτων. (Εικόνα 5-1)



Εικόνα 5-1: Πλήθος οχημάτων για κάθε χαρακτηριστικό.

Για το χαρακτηριστικό στόχος *class_val* (Εικόνα 5-2), το 70% (1210) των οχημάτων του συνόλου, ανήκουν στην χαμηλότερη ποιότητας κατηγορία (*unacc*), με την αμέσως πιο καλή κατηγορία (*acc*) να περιέχει το 22,2% (384) των οχημάτων. Οι δύο καλύτερες κατηγορίες (*good* και *vgood*) περιέχουν σχεδόν τον ίδιο αριθμό αυτοκινήτων με ποσοστά 3,99% (69) και 3,76% (65) αντίστοιχα και αποτελούν την μειοψηφία των οχημάτων.



Εικόνα 5-2: Πλήθος οχημάτων ανά κατηγορία.

5.3 Επεξεργασία δεδομένων

5.3.1 Καθαρισμός δεδομένων

Τα συγκεκριμένα δεδομένα αποτελούνται από 7 στήλες και 1728 γραμμές. Όλα τα πεδία χαρακτηριστικών, μαζί και το χαρακτηριστικό στόχος, έχουν τιμές σε όλα τα πεδία τους, ενώ δεν υπάρχουν διπλοεγγραφές.

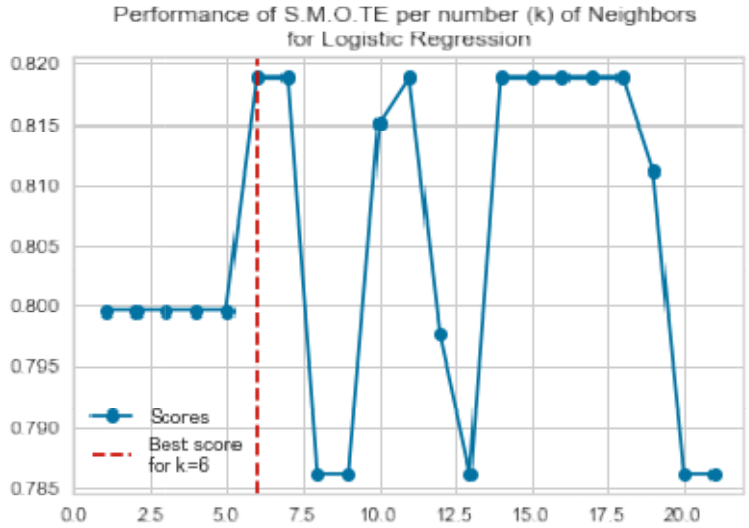
5.3.2 Μετασχηματισμός δεδομένων

Για να εκτελέσουμε κωδικοποίηση των κατηγορικών τιμών, θα χρησιμοποιήσουμε τη μέθοδο, κατά την οποία, για κάθε μια μοναδική τιμή των έξι χαρακτηριστικών δημιουργεί μια καινούρια στήλη σε κωδικοποίηση 0 και 1 που αντιστοιχεί στην τιμή αυτή. Μετά την κωδικοποίηση, το πλήθος των χαρακτηριστικών (στηλών) αυξάνεται στα 21. Για το χαρακτηριστικό στόχος, η κωδικοποίηση θα είναι με τις τιμές από 0 έως 3, για κάθε μια από τις τέσσερις κατηγορίες. Παράλληλα, όλα τα δεδομένα θα κανονικοποιηθούν στον μέσο όρο τους και με μοναδιαία τυπική απόκλιση.

5.3.3 Μη ισορροπημένες κλάσεις

Για την σωστότερη εκπαίδευση των μοντέλων, θα χρησιμοποιήσουμε τη μέθοδο SMOTE (Synthetic Minority Over-sampling Technique) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), η οποία θα δημιουργήσει τεχνητά δεδομένα στο χαρακτηριστικό *class_val*, για τις 3 κατηγορίες με τα λιγότερα οχήματα, ώστε όλες να έχουν τον ίδιο αριθμό οχημάτων. Γενικά, στα προβλήματα κατηγοριοποίησης, είναι επιθυμητό τα

μοντέλα να εκπαιδεύονται με δείγματα που περιέχουν περίπου ίσο αριθμό εγγραφών από κάθε κατηγορία, ώστε να μην δημιουργηθούν αποτελέσματα που δεν θα είναι αντικειμενικά.

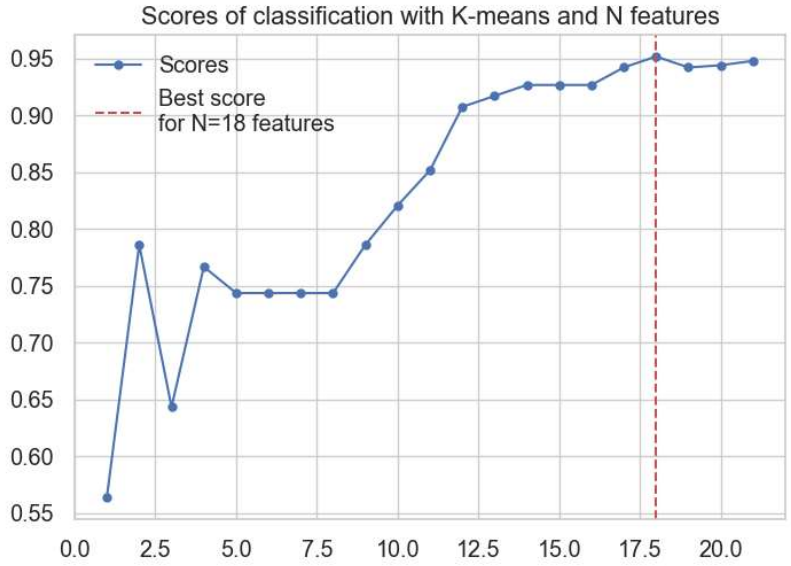


Εικόνα 5-3: Αποτελέσματα SMOTE ανά αριθμό πλησιέστερων γειτόνων για το σύνολο δεδομένων Car Evaluation.

Με την χρήση του SMOTE κατά την λογιστική παλινδρόμηση, τα καλύτερα αποτελέσματα επιτεύχθηκαν όταν δημιουργήθηκαν τεχνητά δεδομένα με βάση τους 6 πλησιέστερους γείτονες.

5.4 Αποτελέσματα

Για όλα τα μοντέλα χρησιμοποιούμε για εκπαίδευση το 70% των δεδομένων και για δοκιμή το υπόλοιπο 30%. Κάθε αλγόριθμος εκπαιδεύτηκε με το ίδιο υποσύνολο εκπαίδευσης και δοκιμάστηκε τόσο στο ίδιο υποσύνολο δόκιμης, όσο και με την μέθοδο της πολλαπλής επικύρωσης 10 φορών για τις μεθόδους ευστοχίας, ακρίβειας, ανάκλησης, F_1 και περιοχής κάτω από την καμπύλη ROC. Κατά την χρήση του αλγορίθμου Κ-μέσων, χρησιμοποιήθηκαν τα 18 καλύτερα χαρακτηριστικά από το σύνολο δεδομένων, με τα οποία επιτεύχθηκε το καλύτερο ποσοστό ευστοχίας. (Εικόνα 5-4)



Εικόνα 5-4: Αποτελέσματα λογιστικής παλινδρόμησης με την χρήση του αλγορίθμου Κ-μέσων και 18 χαρακτηριστικών για το σύνολο δεδομένων Car Evaluation.

Τα αποτελέσματα τόσο για το υποσύνολο δοκιμής, όσο και για την πολλαπλή επικύρωση φαίνονται στους δύο παρακάτω πίνακες.

Models	Accuracy	Precision	Recall	F_1	AUC
Logistic Regression	0.792	0.381	0.455	0.397	0.949
Logistic Regression + Kmeans	0.931	0.802	0.916	0.849	0.990
Decision Tree	0.715	0.530	0.679	0.528	0.924
Decision Tree + Kmeans	0.805	0.615	0.803	0.671	0.936
Neural Network	0.898	0.790	0.803	0.796	0.983
Neural Network + Kmeans	0.960	0.879	0.934	0.904	0.996

Πίνακας 5-1: Αποτελέσματα για το υποσύνολο δοκιμής για το σύνολο δεδομένων Car Evaluation.

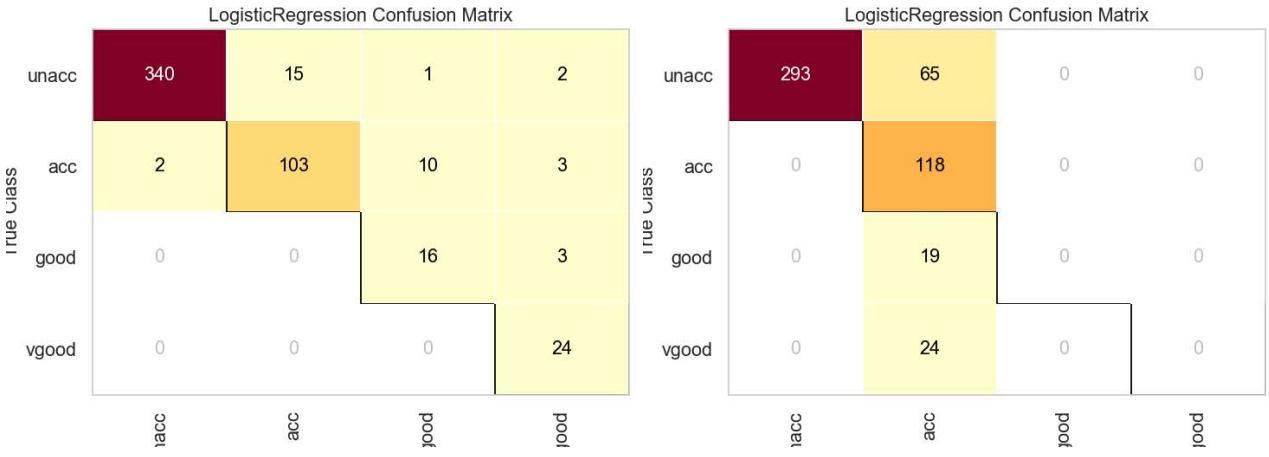
Εύκολα παρατηρούμε ότι, η χρήση του αλγορίθμου Κ-μέσων παρήγαγε καλύτερα αποτελέσματα σε σχέση με τα απλά μοντέλα, για όλες τις μετρικές μεθόδους για το υποσύνολο δοκιμής (Πίνακας 5-1). Ωστόσο, κατά την πολλαπλή επικύρωση, η λογιστική παλινδρόμηση και το δέντρο απόφασης επωφελήθηκαν της προτεινόμενης μεθοδολογίας πετυχαίνοντας καλύτερα αποτελέσματα, ενώ το νευρωνικό δίκτυο με την χρήση του

αλγορίθμου Κ-μέσων, δεν κατάφερε να ξεπεράσει τα αντίστοιχα αποτελέσματα του «απλού» νευρωνικού δικτύου, για καμία μετρική μέθοδο (Πίνακας 5-2).

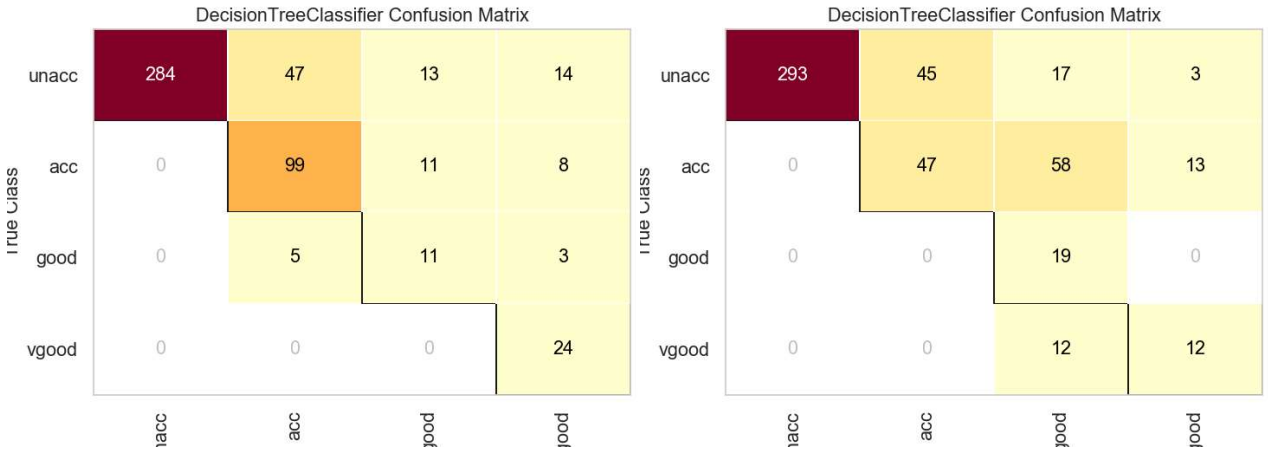
Models	**CV Accuracy**	**CV Precision**	**CV Recall**	**CV F_1**	**CV AUC**
Logistic Regression	0.833	0.684	0.680	0.653	0.974
Logistic Regression + Kmeans	0.844	0.771	0.740	0.714	0.977
Decision Tree	0.760	0.434	0.475	0.426	0.861
Decision Tree + Kmeans	0.790	0.564	0.558	0.524	0.921
Neural Network	0.894	0.877	0.882	0.861	0.977
Neural Network + Kmeans	0.873	0.816	0.834	0.797	0.966

Πίνακας 5-2: Αποτελέσματα πολλαπλής επικύρωσης 10 φορών για το σύνολο δεδομένων Car Evaluation.

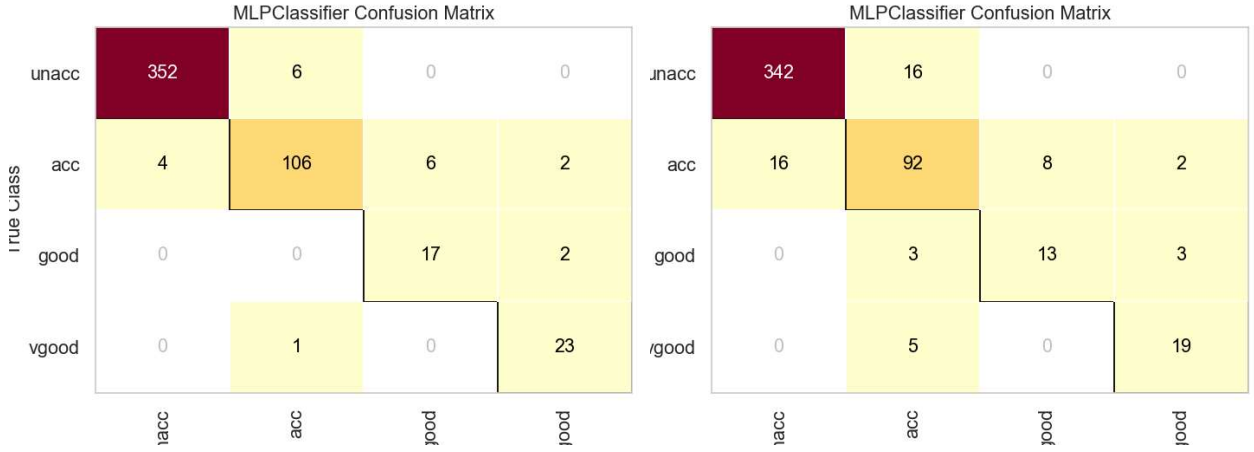
Παράλληλα με τις παραπάνω μετρικές μεθόδους, μπορούμε να χρησιμοποιήσουμε ένα πινάκα σύγχυσης για να καταλάβουμε καλυτέρα πως τα παραπάνω μοντέλα ταξινομούν τις παρατηρήσεις του υποσυνόλου δοκιμής, με και χωρίς την χρήση του αλγορίθμου Κ-μέσων.



Εικόνα 5-5: Πίνακας σύγχυσης για την λογιστική παλινδρόμηση με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου Κ-μέσων για το σύνολο δεδομένων Car Evaluation.



Εικόνα 5-6: Πίνακας σύγχυσης για το δέντρο απόφασης με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Car Evaluation.



Εικόνα 5-7: Πίνακας σύγχυσης για το νευρωνικό δίκτυο με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Car Evaluation.

Κατά την εκτέλεση της δοκιμής ANOVA για τα μοντέλα χωρίς την χρήση του αλγορίθμου K-μέσων και τα τρία μοντέλα παράγουν προβλέψεις με σημαντικές διαφορές στην διακύμανση, όπως φαίνεται παρακάτω:

Models	Logistic Regression	Decision Tree	Neural Network
Logistic Regression	-	2.892	4.767
Decision Tree	2.892	-	7.671
Neural Network	4.767	7.671	-

Πίνακας 5-3: Ανάλυση διακύμανσης προβλέψεων χωρίς την χρήση K-Means για το σύνολο δεδομένων Car Evaluation.

Αντίστοιχη εικόνα προκύπτει και με την χρήση του αλγορίθμου Κ-μέσων και της προτεινόμενης μεθοδολογίας για όλα τα μοντέλα, με την λογιστική παλινδρόμηση και το νευρωνικό δίκτυο να παράγουν προβλέψεις με αρκετά όμοιες διακυμάνσεις.

Models	Logistic Regression w/ K-means	Decision Tree w/ K-means	Neural Network w/ K-means
Logistic Regression w/ K-means	-	6.065	2.048
Decision Tree w/ K-means	6.065	-	7.941
Neural Network w/ K-means	2.048	7.941	-

Πίνακας 5-4: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Car Evaluation.

6 Μελέτη συνόλου δεδομένων Bank Marketing

6.1 Εισαγωγή

Η σύνολο δεδομένων Bank Marketing περιλαμβάνει πραγματικά δεδομένα που συλλέχθηκαν από πορτογαλική τράπεζα, από το Μάιο του 2008 έως τον Νοέμβριο του 2010, με συνολικά 41188 τηλεφωνικές επαφές. Αυτή η προωθητική ενέργεια από την τράπεζα επικεντρώνεται στη στόχευση πελατών μέσω τηλεφωνικών κλήσεων για την πώληση μακροχρόνιων, προθεσμιακών καταθέσεων. Το χαρακτηριστικό στόχος αφορά την αποδοχή ή μη αποδοχή από τον πελάτη του τηλεφωνικού προϊόντος.

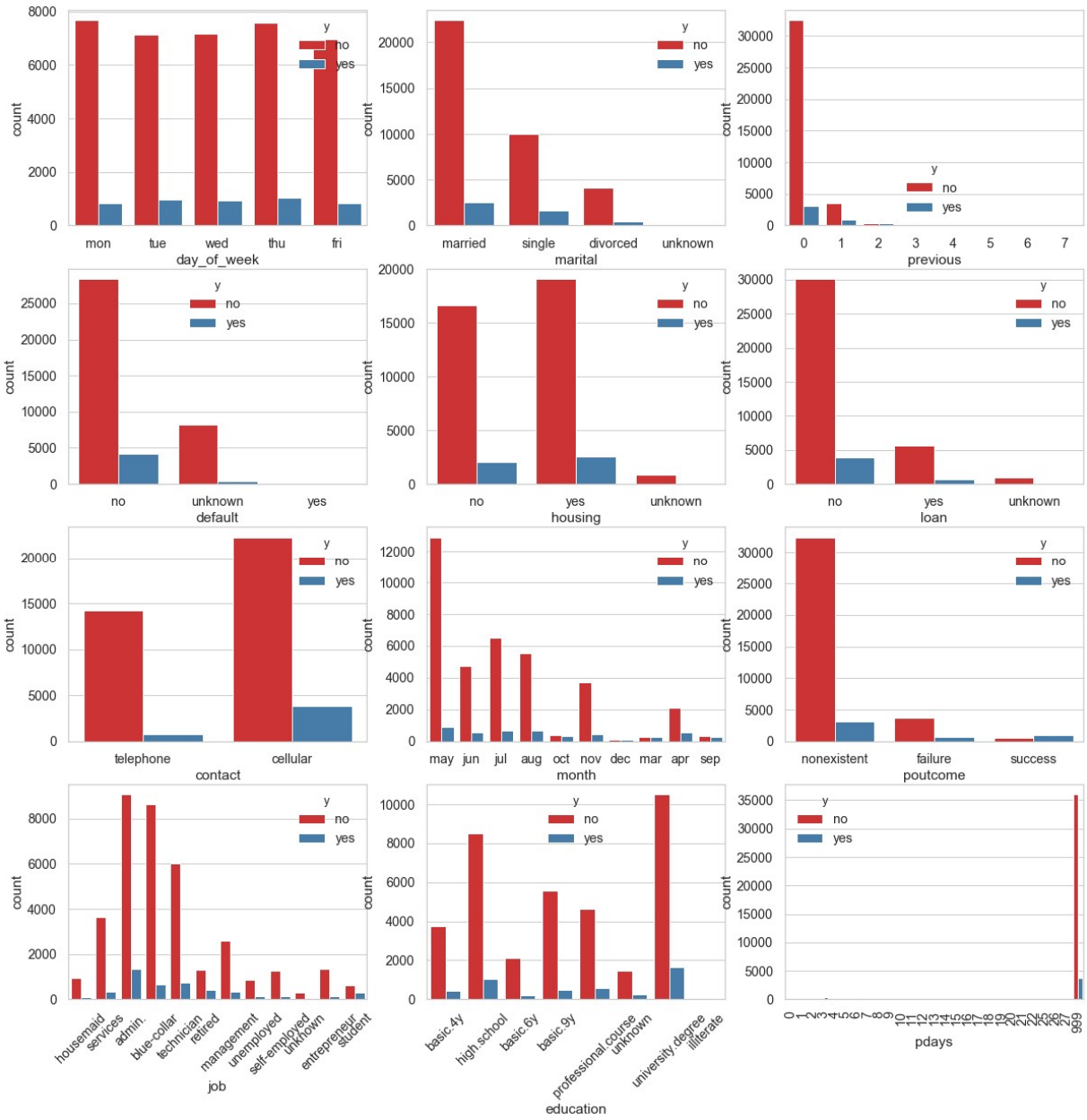
6.2 Ανάλυση δεδομένων

6.2.1 Ανάλυση χαρακτηριστικών

Το χαρακτηριστικό στόχος αποτελείται από δύο μη ισορροπημένες κλάσεις. Η θετική κλάση περιλαμβάνει 4640 (11,26%) παρατηρήσεις ενώ η αρνητική 36548 (88,74%) παρατηρήσεις.

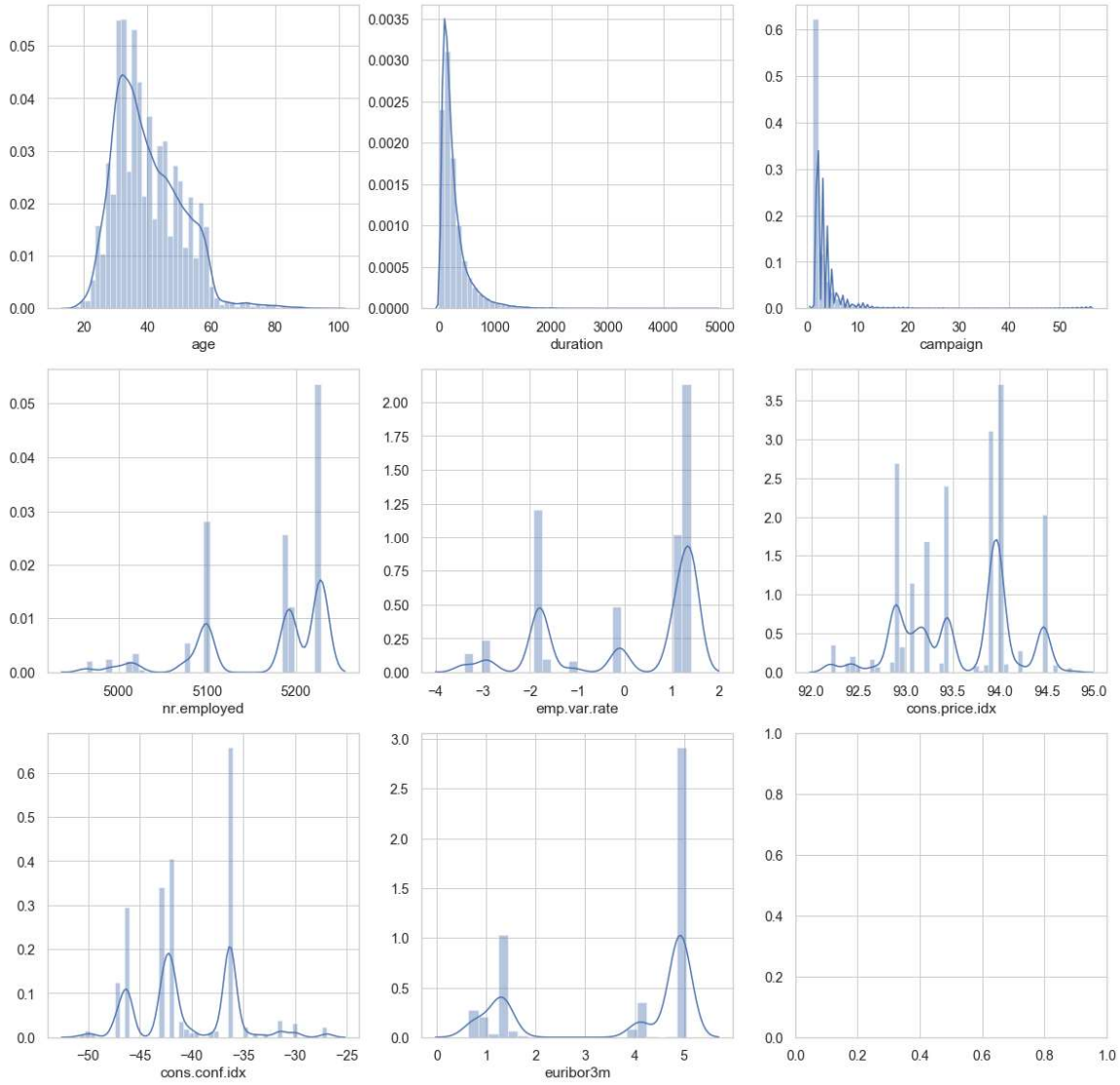
Εικόνα 6-1: Πλήθος παρατηρήσεων για το χαρακτηριστικό στόχος του συνόλου δεδομένων Bank Marketing.

Παρακάτω μπορούμε να δούμε πως κατανέμονται οι κατηγορίες των κατηγορικών χαρακτηριστικών για το σύνολο δεδομένων, σε συνδυασμό με το χαρακτηριστικό στόχος.



Εικόνα 6-2: Κατηγορικά χαρακτηριστικά του συνόλου δεδομένων Bank Marketing.

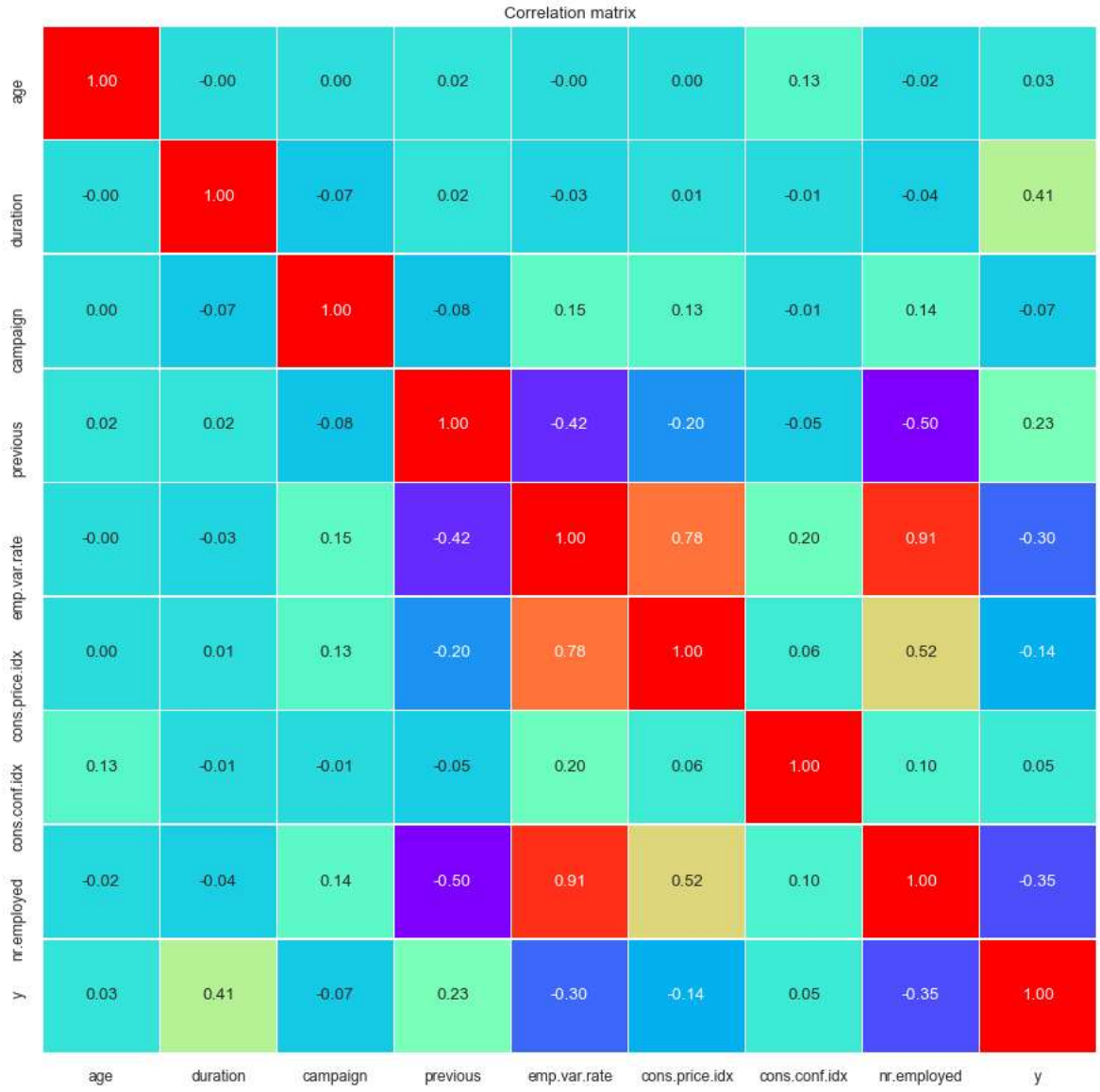
Για τα χαρακτηριστικά με συνεχείς τιμές οι αντίστοιχες κατανομές φαίνονται στην παρακάτω εικόνα.



Εικόνα 6-3: Κατανομή ποσοτικών χαρακτηριστικών του συνόλου δεδομένων Bank Marketing.

6.2.2 Πίνακας συσχετίσεων

Στο παρακάτω διάγραμμα, μπορούμε να δούμε τους συντελεστές των γραμμικών συσχετίσεων για κάθε χαρακτηριστικό με συνεχείς τιμές σε συνάρτηση με το χαρακτηριστικό στόχος, αλλά και σε σχέση με κάθε άλλο ποσοτικό χαρακτηριστικό (Εικόνα 6-4). Παρατηρούμε, πως το χαρακτηριστικό *emp.var.rate* πολύ υψηλό συντελεστή συσχέτισης με το χαρακτηριστικό *nr.employed* (0.91) και κατά συνέπεια το *nr.employed* δεν θα χρησιμοποιηθεί κατά την εκπαίδευση των μοντέλων



Εικόνα 6-4: Πίνακας συσχετίσεων Pearson για τα χαρακτηριστικά με συνεχής τιμές για το σύνολο δεδομένων Bank Marketing.

6.3 Επεξεργασία

6.3.1 Καθαρισμός δεδομένων

Το σύνολο δεδομένων αποτελείται από 21 χαρακτηριστικά. Από αυτά, τα 12 είναι κατηγορικά, ενώ τα υπόλοιπα είναι συνεχών τιμών. Περιέχει επίσης 12 διπλοεγγραφές, οι οποίες θα αφαιρεθούν πριν την δημιουργία των μοντέλων, αφού δεν προσφέρουν κάποια νέα πληροφορία. Κάποια κατηγορικά χαρακτηριστικά όπως τα *job, marital, education, default, housing, loan* έχουν μια κατηγορία *uknown,* που σύμφωνα με την περιγραφή του συνόλου δεδομένων αντιστοιχούν σε άγνωστες ή σε πεδία χωρίς

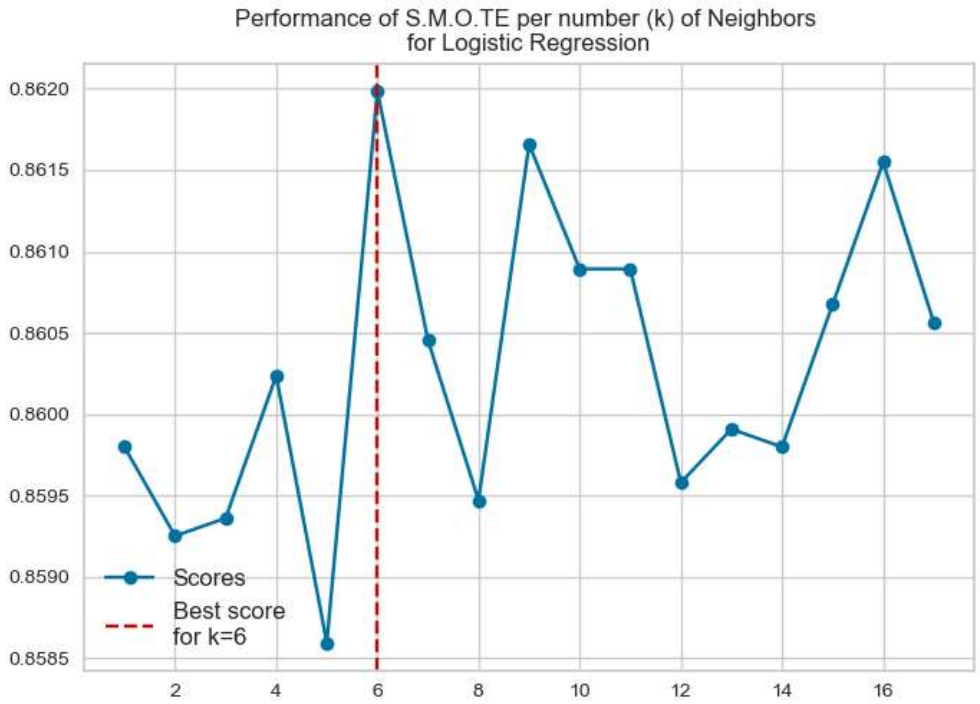
τιμές. Επειδή δεν μπορούμε να ξέρουμε την πιθανή τιμή για το κάθε χαρακτηριστικό, οι τιμές αυτές θα αφαιρεθούν πριν την εκπαίδευση των μοντέλων.

6.3.2 Μετασχηματισμός δεδομένων

Το χαρακτηριστικό *nr.employed* αφαιρείται λόγω υψηλής συσχέτισης με το χαρακτηριστικό *emp.var.rate* . Τα υπόλοιπα χαρακτηριστικά με κατηγορικές τιμές θα κωδικοποιηθούν σε 0 και 1, δημιουργώντας εικονικά χαρακτηριστικά-στήλες για κάθε μοναδική τιμή. Μετά την κωδικοποίηση αυτή, τα δεδομένα θα έχουν συνολικά 81 χαρακτηριστικά, μη συμπεριλαμβανομένου του χαρακτηριστικού στόχος. Τέλος, για όλα τα χαρακτηριστικά έγινε κανονικοποίηση σε μέσο όρο 0 και σε μοναδιαία τυπική απόκλιση.

6.3.3 Μη ισορροπημένες κλάσεις

Το χαρακτηριστικό στόχος μετά τη αφαίρεση των άγνωστων τιμών περιλαμβάνει 26620 (87,34%) εγγραφές για την αρνητική κλάση και 3858 (12,65%) για την θετική. Εδώ θα χρησιμοποιήσουμε πάλι το SMOTE, δημιουργώντας τεχνητά δεδομένα για την θετική κλάση για να την ισορροπήσουμε σε σχέση με την αρνητική.

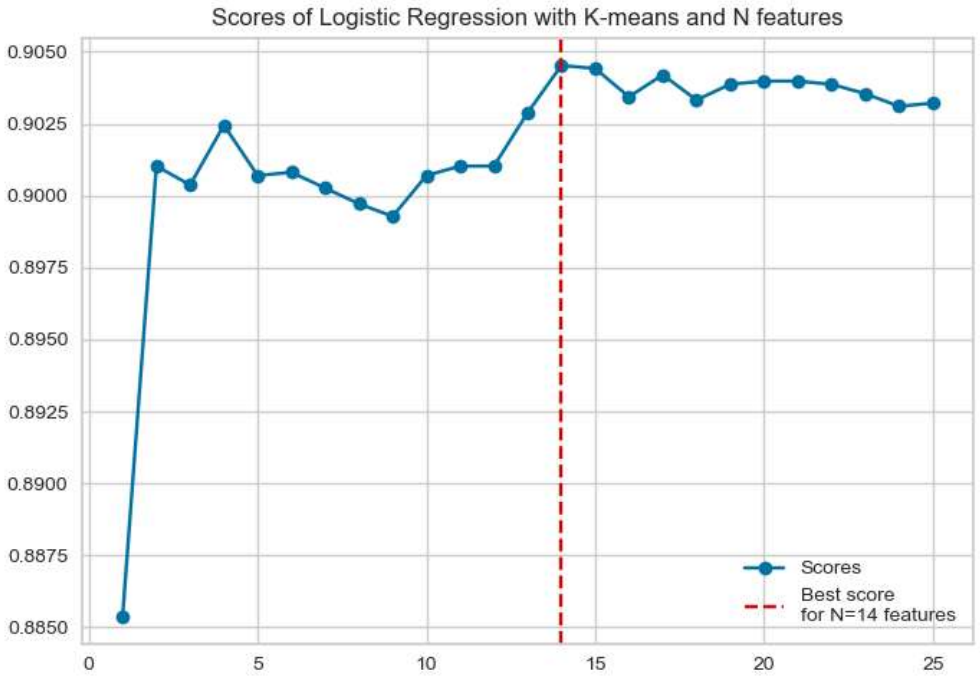


Εικόνα 6-5: Αποτελέσματα SMOTE ανά αριθμό πλησιέστερων γειτόνων με την χρήση λογιστικής παλινδρόμησης για το σύνολο δεδομένων Bank Marketing.

Όπως φαίνεται και στο παραπάνω διάγραμμα, τα καλύτερα αποτελέσματα επιτεύχθηκαν όταν δημιουργήθηκαν τεχνητά δεδομένα με βάση τους 6 πλησιέστερους γείτονες.

6.4 Αποτελέσματα

Για όλα τα μοντέλα χρησιμοποιούμε για εκπαίδευση το 70% των δεδομένων και για δοκιμή το υπόλοιπο 30%. Κάθε αλγόριθμος εκπαιδεύτηκε με το ίδιο υποσύνολο εκπαίδευσης και δοκιμάστηκε τόσο στο υποσύνολο δόκιμης, όσο και με την μέθοδο της πολλαπλής επικύρωσης 10 φορών για τις μεθόδους ευστοχίας, ακρίβειας, ανάκλησης, F_1 και περιοχής κάτω από την καμπύλη ROC. Κατά την χρήση του αλγορίθμου K-μέσων χρησιμοποιήθηκαν τα 14 καλύτερα χαρακτηριστικά από το σύνολο δεδομένων, με τα οποία επιτεύχθηκε το καλύτερο ποσοστό ευστοχίας.



Εικόνα 6-6: Αποτελέσματα λογιστικής παλινδρόμησης με την χρήση του αλγορίθμου K-μέσων και 14 χαρακτηριστικά για το σύνολο δεδομένων Bank Marketing.

Τα αποτελέσματα τόσο για το υποσύνολο δοκιμής, όσο και για την πολλαπλή επικύρωση φαίνονται στους δύο παρακάτω πίνακες.

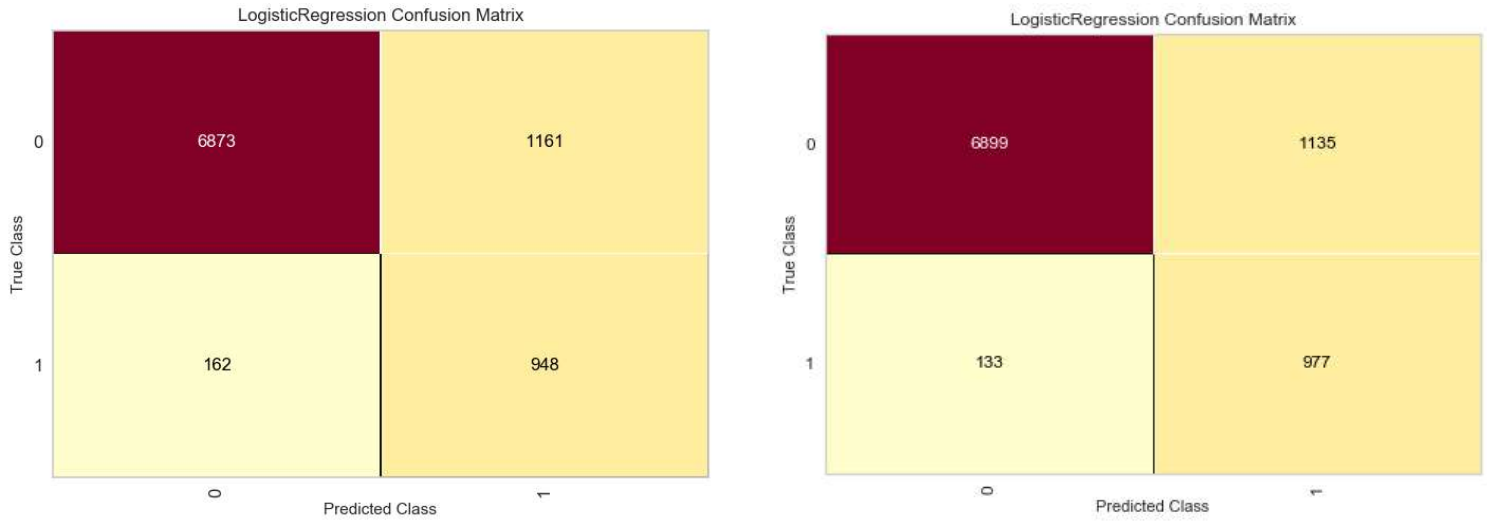
Models	Accuracy	Precision	Recall	F_1	AUC
Logistic Regression	0.862	0.464	0.885	0.609	0.937
Logistic Regression + Kmeans	0.855	0.450	0.854	0.589	0.931
Decision Tree	0.837	0.411	0.805	0.544	0.889
Decision Tree + Kmeans	0.840	0.419	0.827	0.556	0.881
Neural Network	0.887	0.527	0.718	0.608	0.928
Neural Network + Kmeans	0.846	0.434	0.899	0.586	0.936

Πίνακας 6-1: Αποτελέσματα για το υποσύνολο δοκιμής για το σύνολο δεδομένων Bank Marketing.

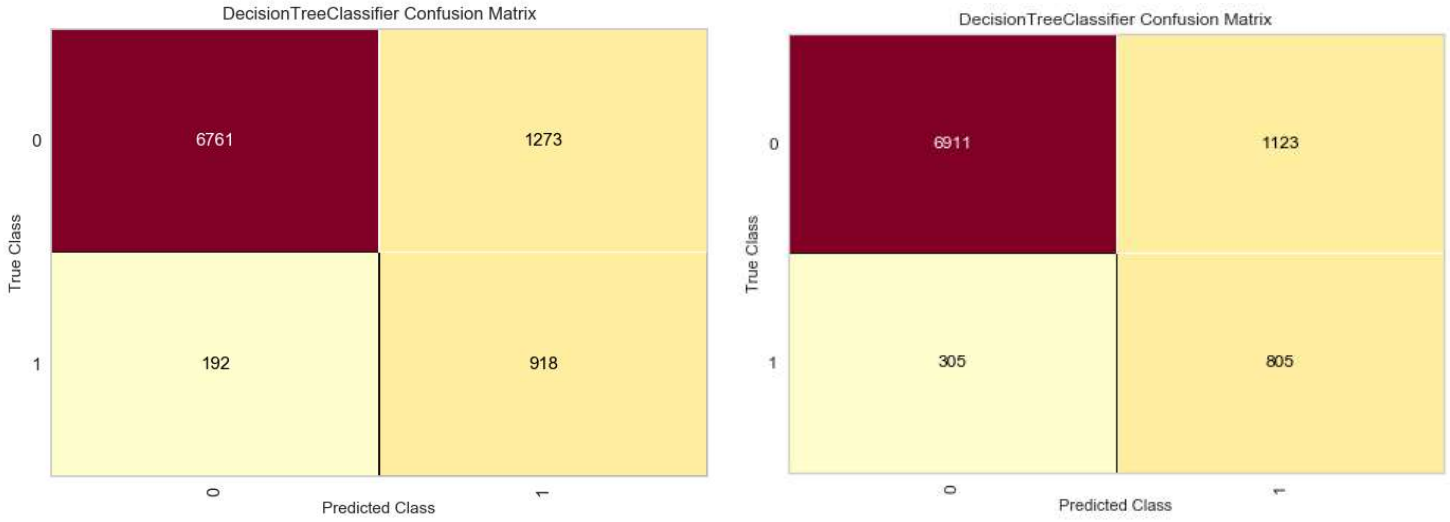
Models	CV Accuracy	CV Precision	CV Recall	CV F_1	CV AUC
Logistic Regression	0.822	0.636	0.303	0.327	0.862
Logistic Regression + Kmeans	0.849	0.663	0.341	0.362	0.913
Decision Tree	0.775	0.225	0.295	0.190	0.630
Decision Tree + Kmeans	0.781	0.359	0.340	0.258	0.658
Neural Network	0.764	0.365	0.252	0.203	0.823
Neural Network + Kmeans	0.788	0.363	0.260	0.209	0.849

Πίνακας 6-2: Αποτελέσματα πολλαπλής επικύρωσης 10 φορών για το σύνολο δεδομένων Bank Marketing.

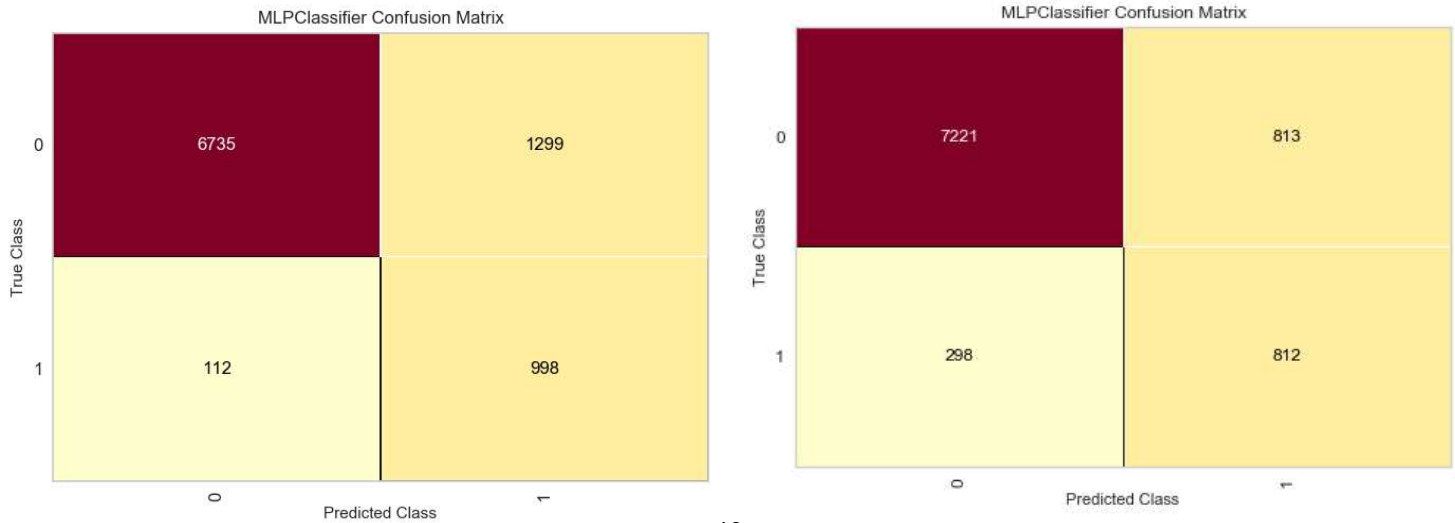
Γενικά, η προτεινόμενη μεθοδολογία στην περίπτωση του υποσυνόλου δοκιμής δεν κατάφερε (πλην του δέντρου απόφασης) να ξεπεράσει τα αρχικά μοντέλα από πλευρά ευστοχίας. Εντούτοις, κατά την πολλαπλή επικύρωση πέτυχε καλύτερες επιδόσεις σε σύγκριση με όλα τα αρχικά μοντέλα και για όλες τις μετρικές μεθόδους. Παρακάτω παραθέτονται όλοι οι πίνακες σύγχυσης για όλα τα μοντέλα.



Εικόνα 6-7: Πίνακας σύγχυσης για την λογιστική παλινδρόμηση με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Bank Marketing.



Εικόνα 6-8: Πίνακας σύγχυσης για την δέντρου απόφασης με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Bank Marketing.



Εικόνα 6-9: Πίνακας σύγχυσης για το νευρωνικό δίκτυο με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Bank Marketing.

Για τα αρχικά μοντέλα (χωρίς την χρήση του αλγορίθμου Κ-μέσων) το τεστ ανάλυσης της διακύμανσης δείχνει ότι οι προβλέψεις όλων των μοντέλων, έχουν στατιστικά σημαντικές διαφορές μεταξύ τους.

Models	Logistic Regression	Decision Tree	Neural Network
Logistic Regression	-	4.818	5.209
Decision Tree	4.818	-	10.018
Neural Network	5.209	10.018	-

Πίνακας 6-3: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Bank Marketing.

Όταν χρησιμοποιηθεί ο αλγόριθμος Κ-μέσων οι προβλέψεις που παράγονται από το δέντρο απόφασης και το νευρωνικό δίκτυο δεν έχουν σημντικές διαφορές στην διακύμανσή τους, σε αντίθεση με την λογιστική παλινδρόμηση που παράγει προβλέψεις με στατιστικά σημαντικές διαφορές στην διακύμανση σε σχέση με τα δύο άλλα μοντέλα, όπως φαίνεται στον παρακάτω πίνακα.

Models	Logistic Regression w/ K-means	Decision Tree w/ K-means	Neural Network w/ K-means
Logistic Regression w/ K-means	-	2.922	1.825
Decision Tree w/ K-means	2.922	-	1.097
Neural Network w/ K-means	1.825	1.097	-

Πίνακας 6-4: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Bank Marketing.

7 Μελέτη συνόλου δεδομένων Default of credit card clients

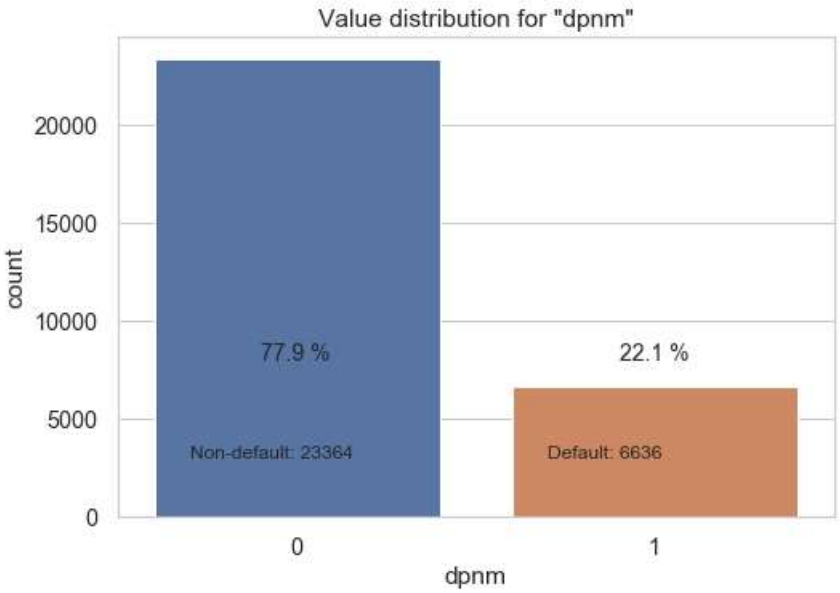
7.1 Εισαγωγή

Το σύνολο δεδομένων Default of credit card clients περιέχει δεδομένα πελατών ταϊβανέζικων τραπεζών, οι οποίοι είχαν στην κατοχή τους και χρησιμοποιούσαν πιστωτικές κάρτες (Yeh, 2009). Θέλοντας να αυξήσουν το μερίδιο τους στην αγορά, οι τράπεζες της Ταϊβάν εξέδωσαν ανεξέλεγκτα, μεγάλο αριθμό πιστωτικών καρτών σε πελάτες οι οποίοι δεν ήταν αξιόπιστοι και δεν πληρούσαν τα ορθά πιστωτικά κριτήρια. Από την πλευρά τους, οι κάτοχοι των πιστωτικών καρτών τις υπερφόρτωσαν, χωρίς να έχουν την δυνατότητα να αποπληρώσουν τα χρέος τους. Αυτό είχε ως αποτέλεσμα να δημιουργηθεί μεγάλη έλλειψη εμπιστοσύνης τόσο από τις τράπεζες προς του πελάτες όσο και το αντίστροφο.

Με την χρήση των παραπάνω δεδομένων, έγινε προσπάθεια να συνδεθεί η πιθανότητα να πληρώσει ένας πελάτης την δόση της πιστωτικής του κάρτας για τον επόμενο μήνα, με την «πραγματική» πιθανότητα αυτός ο πελάτης να είναι αξιόπιστος για έκδοση και κατοχή πιστωτικής κάρτας. Οι δύο αυτές πιθανότητες δεν συνδέονται απαραίτητα μεταξύ τους και αποτελούν ένα ανοιχτό πεδίο μελέτης.

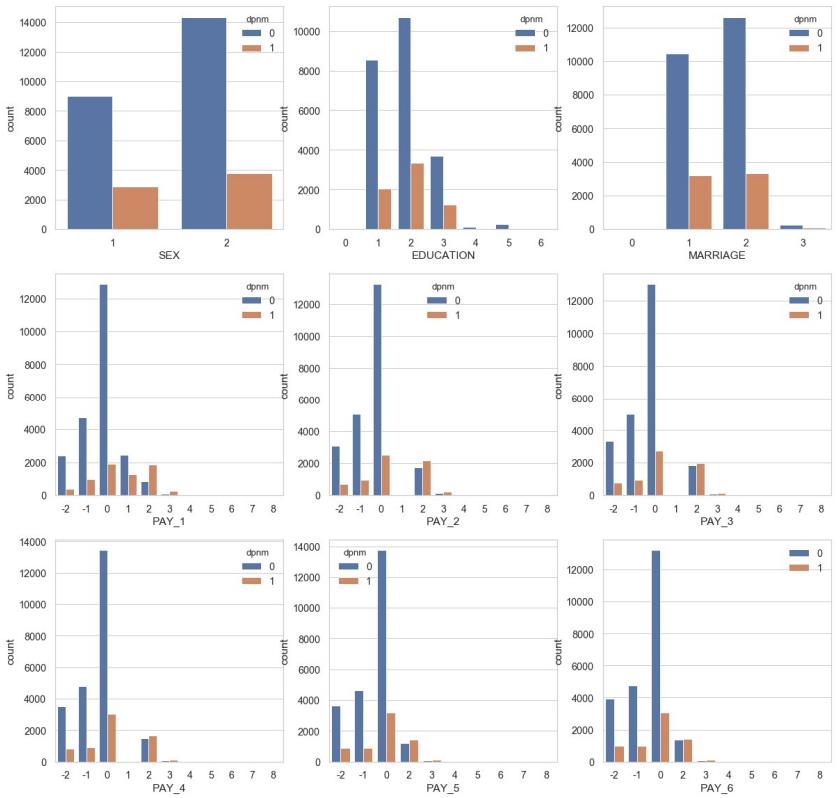
7.2 Ανάλυση δεδομένων

Το σύνολο δεδομένων περιέχει 30000 εγγραφές και 25 χαρακτηριστικά. Από αυτά, τα 9 είναι κατηγορικά ενώ τα υπόλοιπα έχουν συνεχείς τιμές. Το χαρακτηριστικό στόχος *dpnm* είναι κωδικοποιημένο σε 0 (πελάτες που θα πληρώσουν την δόση του επόμενου μήνα) και σε 1 (πελάτες που δεν θα πληρώσουν την δόση του επόμενου μήνα). Οι κατανομές των δύο αυτών τιμών, φαίνονται στο παρακάτω διάγραμμα.



Εικόνα 7-1: Πλήθος παρατηρήσεων για το χαρακτηριστικό στόχος για το σύνολο δεδομένων Default of credit card clients.

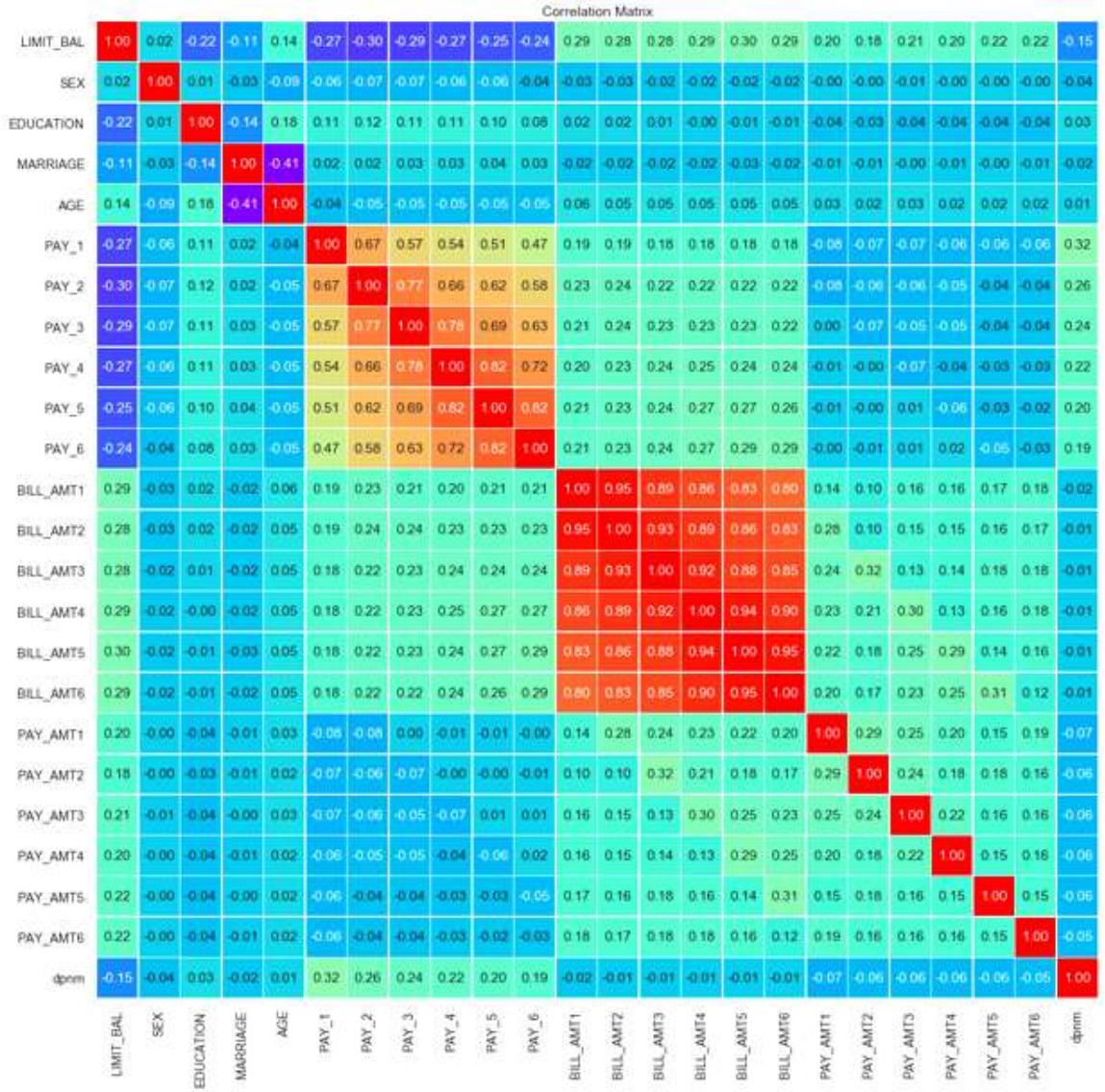
Για τα υπόλοιπα χαρακτηριστικά με κατηγορικές τιμές, οι κατανομές τους σε συνδυασμό με το χαρακτηριστικό στόχος φαίνονται παρακάτω.



Εικόνα 7-2: Κατανομές κατηγορικών χαρακτηριστικών σε σχέση με το χαρακτηριστικό στόχος για το σύνολο δεδομένων Default of credit card clients.

7.2.1 Πίνακας συσχετίσεων

Από τον πίνακα συσχετίσεων μπορούμε να δούμε τους συντελεστές συσχέτισης κατά Pearson των υπολοίπων χαρακτηριστικών με το χαρακτηριστικό στόχος. Μπορούμε να παρατηρήσουμε ότι τα χαρακτηριστικά *PAY_1* έως *PAY_6* καθώς και το *LIMIT_BAL* έχουν την υψηλότερη προβλεπτικότητα για το χαρακτηριστικό στόχος σε σύγκριση με όλα τα άλλα χαρακτηριστικά. Παράλληλα, διαπιστώνεται και η εξαιρετικά υψηλή συσχέτιση (0.8 και άνω) ανάμεσα στα χαρακτηριστικά *BILL_AMT1* έως *BILL_AMT6* γεγονός που υποδεικνύει ότι μπορούμε να χρησιμοποιήσουμε μονό το χαρακτηριστικό *BILL_AMT1* στα μοντέλα μας χωρίς να χάσουμε σημαντικό ποσοστό πληροφορίας για το χαρακτηριστικό *dpnm*.



Εικόνα 7-3: Πίνακας συσχετίσεων Pearson των χαρακτηριστικών με συνεχείς τιμές για το σύνολο δεδομένων Default of credit card clients.

7.3 Επεξεργασία δεδομένων

7.3.1 Καθαρισμός δεδομένων

Από τα 25 χαρακτηριστικά του συνόλου δεδομένων, το χαρακτηριστικό *ID* είναι ο αύξων αριθμός της κάθε εγγραφής και άρα θα αφαιρεθεί αφού δεν μας προσφέρει κάποια ουσιαστική πληροφορία. Σύμφωνα με την περιγραφή του συνόλου δεδομένων στο UCI Machine Learning Repository, το χαρακτηριστικό *MARRIAGE* έχει κωδικοποιηθεί με τρείς διακριτές τιμές (1 = married; 2 = single; 3 = others). Ωστόσο, αν αναλύσουμε το χαρακτηριστικό στα δεδομένα μας, θα δούμε ότι αυτό περιέχει τέσσερις διακριτές τιμές. Αντίστοιχη περίπτωση συναντάμε και για το χαρακτηριστικό *EDUCATION,* το οποίο σύμφωνα με την περιγραφή έχει τέσσερις διακριτές τιμές (1 = graduate school; 2 = university; 3 = high school; 4 = others), ενώ τα δεδομένα περιέχουν έξι διακριτές τιμές. Εφόσον και τα δύο χαρακτηριστικά έχουν κωδικοποιήσει την κατηγορία others ως διακριτή τιμή, οι παραπάνω κατηγορίες και για τα δύο χαρακτηριστικά θα κωδικοποιηθούν ως others, ώστε ο αριθμός των κατηγοριών τους να ταιριάζει με την επίσημη περιγραφή του συνόλου. Τέλος, το σύνολο δεν περιέχει εγγραφές με κενά πεδία τιμών, εντούτοις μετά την αφαίρεση του χαρακτηριστικό *ID* προέκυψαν 35 διπλοεγγραφές, οι οποίες και θα αφαιρεθούν.

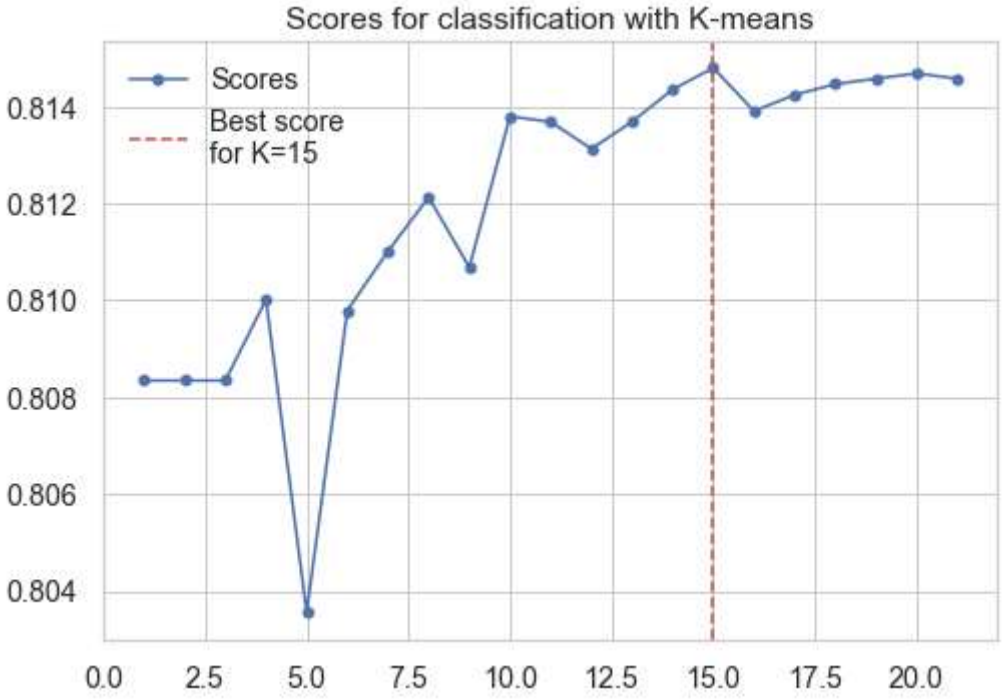
7.3.2 Μετασχηματισμός δεδομένων

Όπως αναφέραμε και ποιο πάνω, λόγω υψηλής συσχέτισης μεταξύ τους τα χαρακτηριστικά *BILL_AMT2* έως και *BILL_AMT6,* δεν θα χρησιμοποιηθούν κατά την εκπαίδευση των μοντέλων. Παράλληλα τα χαρακτηριστικά *EDUCATION, MARRIAGE, SEX, PAY_1, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6* θα κωδικοποιηθούν σε 0 και 1 με την δημιουργία εικονικών χαρακτηριστικών-στηλών για κάθε μοναδική τιμή τους. Μετά την κωδικοποίηση με αυτό τον τρόπο, το πλήθος των χαρακτηριστικών αυξήθηκε στα 82. Τέλος όλα χαρακτηριστικά θα κανονικοποιηθούν σε μηδενικό μέσο όρο και μοναδιαία τυπική απόκλιση.

7.4 Αποτελέσματα

Για όλα τα μοντέλα χρησιμοποιούμε για εκπαίδευση το 70% των δεδομένων και για δοκιμή το υπόλοιπο 30%. Κάθε αλγόριθμος εκπαιδεύτηκε με το ίδιο υποσύνολο εκπαίδευσης και δοκιμάστηκε τόσο στο υποσύνολο δόκιμης, όσο και με την μέθοδο της πολλαπλής επικύρωσης 10 φορών για τις μεθόδους ευστοχίας, ακρίβειας, ανάκλησης, F_1

και περιοχής κάτω από την καμπύλη ROC. Κατά την χρήση του αλγορίθμου Κ-μέσων χρησιμοποιήθηκαν τα 15 καλύτερα χαρακτηριστικά από το σύνολο δεδομένων, με τα οποία επιτεύχθηκε το καλύτερο ποσοστό ευστοχίας.



Εικόνα 7-4: Αποτελέσματα λογιστικής παλινδρόμησης με την χρήση του αλγορίθμου Κ-μέσων για το σύνολο δεδομένων Default of credit card clients.

Τα αποτελέσματα τόσο για το υποσύνολο δοκιμής, όσο και για την πολλαπλή επικύρωση φαίνονται στους δύο παρακάτω πίνακες.

Models	**Accuracy**	**Precision**	**Recall**	**F_1**	**AUC**
Logistic Regression	0.817	0.676	0.353	0.463	0.77
Logistic Regression + Kmeans	0.815	0.672	0.341	0.452	0.77
Decision Tree	0.814	0.669	0.337	0.448	0.74
Decision Tree + Kmeans	0.815	0.674	0.340	0.452	0.72
Neural Network	0.814	0.650	0.371	0.472	0.77
Neural Network + Kmeans	0.817	0.668	0.371	0.477	0.76

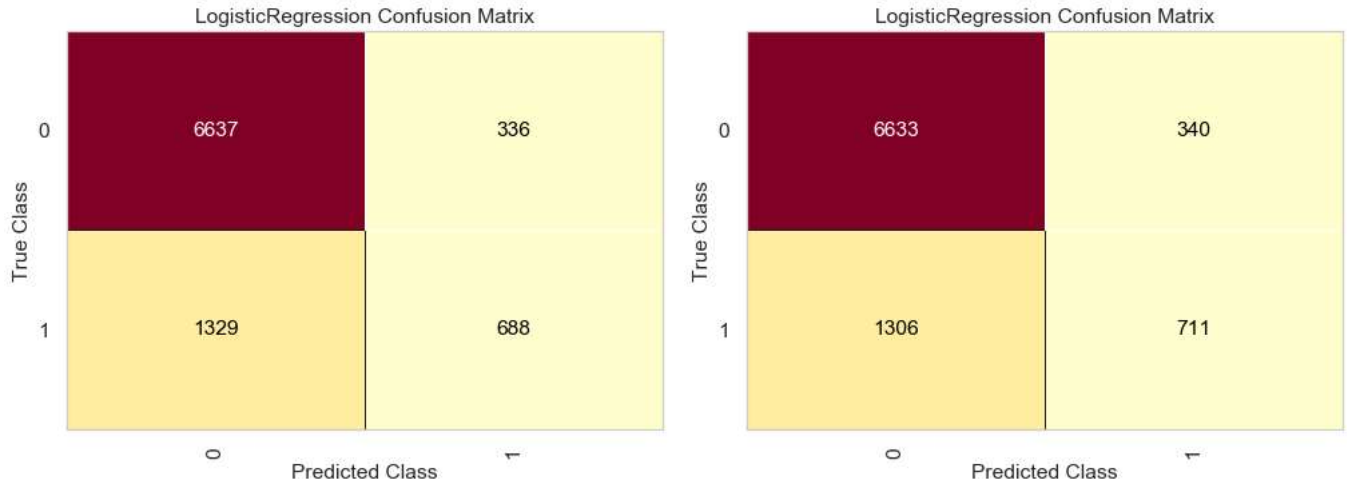
Πίνακας 7-1: Αποτελέσματα για το υποσύνολο δοκιμής για το σύνολο δεδομένων Default of credit card clients.

Models	CV Accuracy	CV Precision	CV Recall	CV F_1	CV AUC
Logistic Regression	0.779	0	0	0	0.639
Logistic Regression+Kmeans	0.818	0.674	0.345	0.456	0.757
Decision Tree	0.818	0.681	0.334	0.447	0.748
Decision Tree + Kmeans	0.820	0.679	0.354	0.464	0.723
Neural Network	0.761	0.546	0.204	0.226	0.657
Neural Network + Kmeans	0.819	0.674	0.355	0.464	0.763

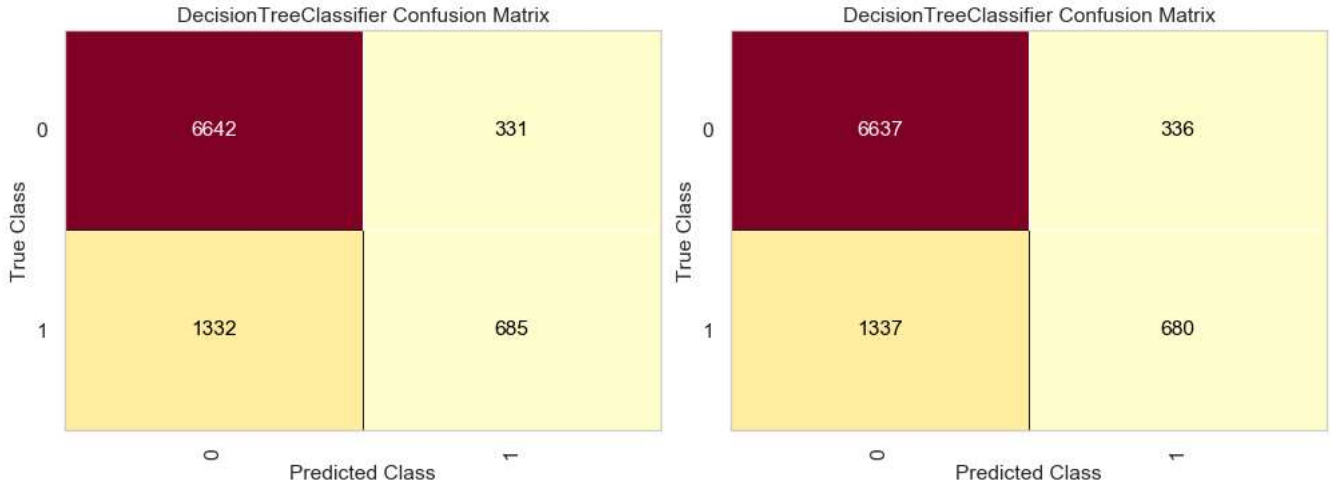
Πίνακας 7-2: Αποτελέσματα πολλαπλής επικύρωσης 10 φορών για το σύνολο δεδομένων Default of credit card clients.

Για το υποσύνολο δοκιμής, ο αλγόριθμος Κ-μέσων δεν δημιούργησε κάποια ουσιαστική διαφορά έναντι των απλών μοντέλων. Ωστόσο, είδαμε βελτίωση στα αποτελέσματα της πολλαπλής επικύρωσης, η οποία κατάφερε να πετύχει αποτελέσματα το ίδιο ικανοποιητικά με αυτά του υποσυνόλου δοκιμής και για τα τρία μοντέλα.

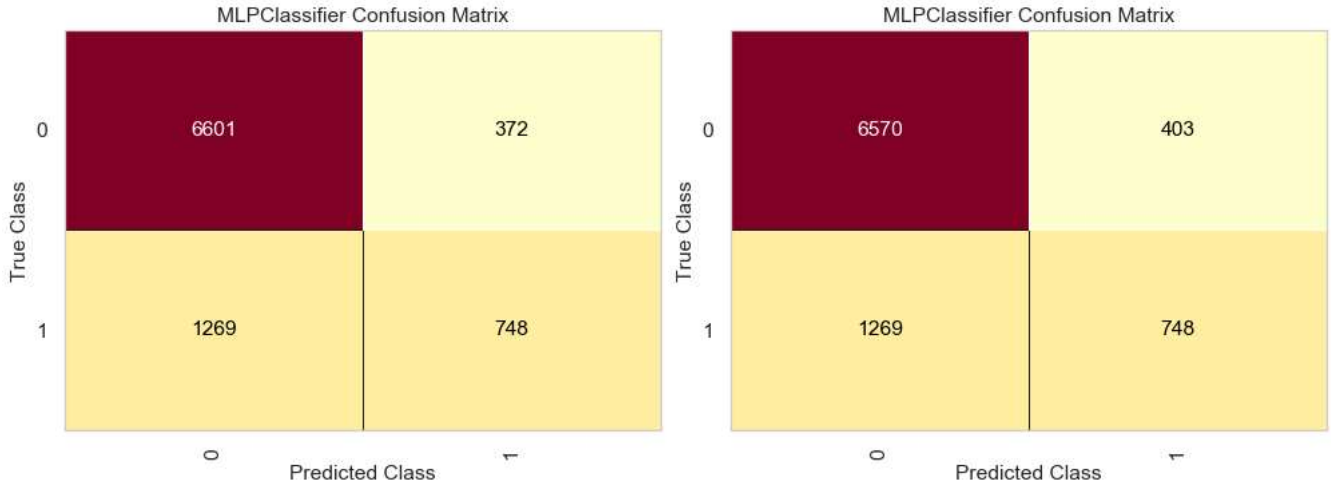
Παρακάτω παραθέτονται οι πίνακες σύγχυσης που παρήγαγαν τα εξεταζόμενα μοντέλα για το υποσύνολο δοκιμής.



Εικόνα 7-5: Πίνακας σύγχυσης για την λογιστική παλινδρόμηση με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου Κ-μέσων για το σύνολο δεδομένων Default of credit card clients.



Εικόνα 7-6: Πίνακας σύγχυσης για το δέντρο απόφασης με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Default of credit card clients.



Εικόνα 7-7: Πίνακας σύγχυσης για το νευρωνικό δίκτυο με (αριστερά) και χωρίς (δεξιά) την χρήση του αλγόριθμου K-μέσων για το σύνολο δεδομένων Default of credit card clients.

Εξετάζοντας τα αποτελέσματα της ανάλυσης της διακύμανσης για το υποσύνολο δοκιμής, διαπιστώνουμε πως και με την χρήση αλλά και χωρίς την χρήση της προτεινόμενης μεθοδολογίας, τα εξεταζόμενα μοντέλα παράγουν προβλέψεις χωρίς σημαντικές διαφορές στην διακύμανσή τους.

Models	Logistic Regression	Decision Tree	Neural Network
Logistic Regression	-	0.519	0.5
Decision Tree	0.519	-	0.019
Neural Network	0.5	0.019	-

Πίνακας 7-3: Ανάλυση διακύμανσης προβλέψεων χωρίς την χρήση K-Means για το σύνολο δεδομένων Default of credit card clients.

Models	Logistic Regression w/ K-means	Decision Tree w/ K-means	Neural Network w/ K-means
Logistic Regression w/ K-means	-	0.038	0.462
Decision Tree w/ K-means	0.038	-	0.424
Neural Network w/ K-means	0.462	0.424	-

Πίνακας 7-4: Ανάλυση διακύμανσης προβλέψεων με την χρήση K-Means για το σύνολο δεδομένων Default of credit card clients.

8 Επίλογος

8.1 Σύνοψη και συμπεράσματα

Σε αύτη την εργασία αναλύσαμε τέσσερα σύνολα δεδομένων και προτείναμε μια μεθοδολογία που χρησιμοποιεί τον αλγόριθμο Κ-μέσων, για να ενισχύσουμε τα αποτελέσματα των προβλημάτων κατηγοριοποίησης. Στα τρία από τα τέσσερα σύνολα που χρησιμοποιήθηκε η προτεινόμενη μεθοδολογία, είδαμε καλύτερα αποτελέσματα κατά την πολλαπλή επικύρωση 10 φορών στην ευστοχία των μοντέλων σε σύγκριση με το να μην την είχαμε χρησιμοποιήσει. Στον ακόλουθο πίνακα συνοψίζονται τα εν λόγω αποτελέσματα.

Models	Data sets			Average
	Car Evaluation	Bank Marketing	Default of credit card clients	
Logistic Regression	0.833	0.822	0.779	0.811
Logistic Regression + Kmeans	0.844	0.849	0.818	0.837
Decision tree	0.760	0.775	0.818	0.784
Decision tree + Kmeans	0.790	0.781	0.820	0.797
Neural Network	0.894	0.764	0.761	0.806
Neural Network + Kmeans	0.873	0.788	0.819	0.826

Πίνακας 8-1: Σύνοψη αποτελεσμάτων για την ευστοχία κατά την πολλαπλή επικύρωση 10 φορών.

8.2 Όρια και περιορισμοί της έρευνας

Κατά την διερεύνηση των μοντέλων για το κάθε σύνολο δεδομένων, επικεντρωθήκαμε κυρίως στην επίδραση που έχει ο αλγόριθμος Κ-μέσων στις προβλέψεις ενός αλγορίθμου κατηγοριοποίησης. Απώτερος στόχος δεν ήταν τόσο να πετύχουμε τα καλύτερα δυνατά αποτελέσματα για το κάθε μοντέλο του κάθε συνόλου, όσο να εξετάσουμε αν βελτιώνονται τα αποτελέσματα ενός μοντέλου ταξινόμησης με την χρήση ενός αλγορίθμου μη εποπτευόμενης μάθησης. Για τον λόγο αυτό δεν ασχοληθήκαμε καθόλου με την αναζήτηση των καλύτερων παραμέτρων για το κάθε

μοντέλο (με ή χωρίς την χρήση του αλγορίθμου Κ-μέσων) ώστε να πετύχει τα βέλτιστα αποτελέσματα.

Παράλληλα, κατά την αναζήτηση του βέλτιστου αριθμού χαρακτηριστικών που θα χρησιμοποιούμε στον αλγόριθμο Κ-μέσων, χρησιμοποιήθηκε μόνο η επιλογή των Ν καλύτερων χαρακτηριστικών με κριτήριο το τεστ ανάλυσης της διακύμανσης, που προσφέρει η βιβλιοθήκη Scikit-learn. Ενδεχομένως, κάποια άλλη μέθοδος, όπως η PCA (Principal Component Analysis) ή η RFE (Recursive Feature Elimination), να είχαν δώσει διαφορετικά αποτελέσματα. Η μεν πρώτη δεν προτιμήθηκε, ώστε το κάθε τελικό μοντέλο να περιέχει χαρακτηριστικά κατανοητά στον χρήστη και να είναι εύκολα ερμηνεύσιμο, η δε δεύτερη μετά από δοκιμή, χρειαζόταν αρκετό χρόνο για να καταλήξει σε αποτελέσματα.

Μια ακόμη παράμετρος που δεν διερευνήθηκε, είναι και ο αριθμός των κεντροείδων που δημιουργεί ο αλγόριθμος Κ-μέσων. Και για τα τρία σύνολα δεδομένων, στα οποία χρησιμοποιήθηκε ο παραπάνω αλγόριθμος, ορίστηκε το πλήθος των μοναδικών τιμών του χαρακτηριστικού στόχος, ως ο αριθμός των παραγόμενων κεντροειδών. Σε συνδυασμό με την επιλογή χαρακτηριστικών, το διαφορετικό πλήθος των κεντροιδών σε σχέση με το πλήθος των μοναδικών τιμών του χαρακτηριστικού στόχος, ενδεχομένως να προσδίδει μια πιο ισχυρή προβλεπτική ικανότητα στο εκάστοτε μοντέλο ταξινόμησης.

Εκτός από τον αλγόριθμο Κ-μέσων, θα μπορούσαμε να χρησιμοποιήσουμε και κάποιον άλλο αλγόριθμο ομαδοποίησης, όπως ενδεικτικά τον αλγόριθμο ομαδοποίησης κινούμενου μέσου όρου (Mean-Shift Clustering), τον αλγοριθμο ομαδοποίησης με βάση την πυκνότητα για δεδομένα με θόρυβο (Density-Based Spatial Clustering of Applications with Noise - DBSCAN) ή την ομαδοποίηση με έναν ιεραρχικό αλγόριθμο (Agglomerative Hierarchical Clustering). Στην περίπτωσή της εργασίας, προτιμήθηκε ο αλγόριθμος Κ-μέσων λόγω ταχύτητας και απλότητας.

Τέλος, η χρήση του πρώιμου σταματήματος κατά την αναζήτηση των Ν καλύτερων χαρακτηριστικών για το κάθε σύνολο δεδομένων, επιλέγει τα όποια χαρακτηριστικά εξετάζοντας την αύξηση του ποσοστού ευστοχίας και τον αριθμό των φορών της μη αύξησης της ευστοχίας πάνω από ένα όριο. Αυτό έχει ως αποτέλεσμα, να μην ελέγχει το σκορ για όλα τα χαρακτηριστικά του κάθε συνόλου, αλλά για κάποιο υποσύνολο C αυτών. Έτσι επιλέγει τον αριθμό των καλύτερων χαρακτηριστικών, μέσα

από το C, χωρίς να λαμβάνει υπόψη την πιθανότητα ένα διαφορετικό υποσύνολο χαρακτηριστικών πλήθους A με A>C, να πετυχαίνει καλύτερα αποτελέσματα.

8.3 Μελλοντικές Επεκτάσεις

Όπως διαπιστώσαμε παραπάνω, η χρήση του αλγορίθμου K-μέσων μαζί με έναν αλγόριθμο κατηγοριοποίησης, μπορεί να επηρεάσει θετικά τα αποτελέσματα που προκύπτουν. Η μεθοδολογία αυτή, ίσως έχει την δυνατότητα να δώσει καλύτερα αποτελέσματα από αυτά που προέκυψαν στη εργασία αυτή , αν χρησιμοποιηθεί κάποιος άλλος αλγόριθμος ομαδοποίησης όπως για παράδειγμα ο αλγόριθμος ομαδοποίησης κινούμενου μέσου όρου (Mean-Shift Clustering), ο αλγόριθμος ομαδοποίησης με βάση την πυκνότητα για δεδομένα με θόρυβο (Density-Based Spatial Clustering of Applications with Noise - DBSCAN) ή ομαδοποίηση με έναν ιεραρχικό αλγόριθμο (Agglomerative Hierarchical Clustering).

Παράλληλα με αυτό μπορούν να εξερευνηθεί το πώς επηρεάζουν τα αποτελέσματα διαφορετικοί μέθοδοι επιλογής χαρακτηριστικών όπως PCA ή RFE, ενώ μπορούν χρησιμοποιηθούν και διαφορετικοί αλγόριθμοι ταξινόμησης που γενικά έχουν παρατηρηθεί να δίνουν καλύτερα αποτελέσματα όπως SVM (Support Vector Machine), Random Forests ή XGboost.

Βιβλιογραφία

1979 oil crisis. (2020, 01 25). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/1979_oil_crisis

Alapati, Y. K. (2016). Combining Clustering with Classification:A Technique to Improve Classification Accuracy. *International Journal of Computer Science Engineering (IJCSE) , 5* (6), pp. 336-338.

Artificial Intelligence & Intelligent Systems group. (n.d.). Retrieved from University of Macedonia: http://ai.uom.gr/Courses/AdvancedNeuralNetworks/Material/NeuralNetworksAll.pdf

Auto MPG Data Set. (n.d.). Retrieved from UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Auto+MPG

Bank Marketing Data Set. (n.d.). Retrieved from UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Bohanec, M., Žnidaršič, M., Rajkovič, V., Bratko, I., & Zupan, B. (2013). DEX Methodology: Three Decades of Qualitative Multi-Attribute Modeling. *Informatica* , pp. 49-54.

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

Car Evaluation Data Set. (n.d.). Retrieved from UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, P. W. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.*

Default of credit card clients Data Set. (n.d.). Retrieved from UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Hearty, J. (2016). *Advanced Machine Learning with Python.* Birmingham: Packt Publishing.

Krzysztof, C. J., Pedrycz, W., Swiniarski, R. W., & Kurgan, L. A. (2007). *Data Mining A Knowledge Discovery Approach.* New York: Springer Science+Business Media, LLC.

Li, J., Cheng, J.-h., Shi, J.-y., & Huang, F. (2012). Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. *Advances in CSIE* , pp. 553–558.

Machine Learning Crash Course. (n.d.). Retrieved from Courses: https://developers.google.com/machine-learning/crash-course/classification/accuracy

Minsky, M., & Papert, S. (1969). *Perceptrons.* M.I.T. Press.

Mishra, D. (n.d.). *Regression: An Explanation of Regression Metrics And What Can Go Wrong.* Retrieved from https://towardsdatascience.com/: https://towardsdatascience.com/regression-an-explanation-of-regression-metrics-and-what-can-go-wrong-a39a9793d914

MONTGOMERY, D. C., PECK, E. A., & VINING, G. G. (2012). *INTRODUCTION TO LINEAR REGRESSION ANALYSIS.* A JOHN WILEY & SONS, INC.

North, D. M. (2012). *Data Mining for the Masses.* Creative Commons Attribution 3.0 License .

Quinlan, J. R. (2014). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Quinlan, R. (1993). *Combining Instance-Based and Model-Based Learning.* Amherst: Morgan Kaufmann.

Rajkovic, M. B. (1988). Knowledge acquisition and explanation for multi-attribute decision making. *In 8th Intl Workshop on Expert Systems and their Applications*, (pp. 59-78). Avignon, France.

Raschka, S. (2015). *Python Machine Learning.* Birmingham: Packt Publishing.

Refanidis, I., & Samaras, N. (n.d.). *University of Macedonia.* Retrieved from compus.uom.gr.

S. Moro, P. C. (2014, June). A Data-Driven Approach to Predict the Success of Bank Telemarketing. *Decision Support Systems* (62), pp. 22-31.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development.* (44), pp. 206–226.

scikit-learn Machine Learning in Python. (n.d.). Retrieved from https://scikit-learn.org/stable/index.html

Shung, K. P. (n.d.). *Accuracy, Precision, Recall or F1?* Retrieved from https://towardsdatascience.com/: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Witten, I. H., & Eibe, F. (2005). *Data Mining Practical Machine Learning Tools and Techniques.* San Francisco: Elsevier Inc.

Yeh, I. C. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications , 36* (2), pp. 2473-2480.

Ανάλυση διακύμανσης. (2019, 10 7). Retrieved from Wikipedia: https://el.wikipedia.org/wiki/Ανάλυση_διακύμανσης

Πετρίδης, Δ. (2015). ΛΟΓΙΣΤΙΚΗ ΠΑΛΙΝΔΡΟΜΗΣΗ. In Δ. Πετρίδης, *Ανάλυση πολυμεταβλητών τεχνικών.* (pp. 89-125). Αθήνα.

Παράρτημα Α - **Κώδικας σε Python**

Παρακάτω παραθέτεται ο κώδικας σε Python με τον οποίο προέκυψαν τα αποτελέσματα της εργασίας. Για το σύνολο δεδομένων Auto MPG, ο κώδικας εμπεριέχεται σε ένα αρχείο, ενώ για τα υπόλοιπα τρία σύνολα δεδομένων σε δύο αρχεία, ένα χωρίς την χρήση του αλγορίθμου Κ-μέσων και το δεύτερο με την χρήση του αλγορίθμου Κ-μέσων.

Όλοι οι κώδικες έτρεξαν εντός της πλατφόρμας Jupyter Notebook και εντός ενός εικονικού περιβάλλοντος με τη χρήση της διανομής Anaconda της Python. Ο χρήστης που επιθυμεί να αναπαράγει τα αποτελέσματα της εργασίας, θα πρέπει να εγκαταστήσει/ενημερώσει τις απαιτούμενες βιβλιοθήκες μέσω της εντολής *pip ή* της εντολής *conda* αν χρησιμοποιεί την διανομή Anaconda. Οι εν λόγω βιβλιοθήκες, καθώς και η έκδοση αυτών αναφέρονται παρακάτω:

- imbalanced-learn==0.5.0
- jupyterlab==1.2.6
- matplotlib==3.1.2
- numpy==1.18.1
- pandas==1.0.0
- plotly==4.5.0
- scipy==1.3.2
- seaborn==0.9.0
- yellowbrick==1.0.1
- scikit-learn==0.21.1

Ο κώδικας αναλαμβάνει να κατεβάσει τα δεδομένα του κάθε συνόλου από το UCI Machine Learning Repository και τα αποθηκεύει στην μνήμη του υπολογιστή, κάθε φορά που ο χρήστης τρέχει το κάθε αρχείο. Ωστόσο προτείνεται να αποθηκευτούν τα αντίστοιχα δεδομένα τοπικά, μηδενίζοντας έτσι τον χρόνο αναμονής για κατέβασμα των δεδομένων, καθώς και την μετέπειτα χρήση τους, εφόσον ο χρήστης επιθυμεί να πειραματιστεί με αυτά.

Σύνολο δεδομένων Auto MPG

```
# Import libraries
import warnings
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.figure_factory as ff
import plotly.graph_objects as go
import seaborn as sns
from sklearn import metrics, tree
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
warnings.filterwarnings('ignore')

# Read the data.
ds = pd.read_csv(
    "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-
mpg/auto-mpg.data-original", names=['mpg', 'cylinders', 'displacement',
'horsepower', 'weight', 'acceleration', 'model year', 'origin', 'car
name'], delim_whitespace=True)

# Examine the data types.
print(ds.info(), '\n')

# Statistics for the data set.
print(ds.describe().transpose(), '\n')

# Check for duplicate rows.
print(
    f"There are {ds.duplicated().sum()} duplicate rows in the data
set.", '\n')

# Number of null values for each attribute.
print(ds.isnull().sum(), '\n')

# Rows with null value at the "horsepower" column.
print('=====Number of null values per attribute=====', '\n')
print(ds[ds['horsepower'].isnull()], '\n')

# Fill the null values of "horsepower" with the mean of "horsepower".
ds['horsepower'] = ds['horsepower'].fillna(ds['horsepower'].mode())

# Pairplot
sns.pairplot(
    ds.drop(['cylinders', 'origin', 'model year', 'car name'], axis=1))

# Plotting a correlation heatmap for the data set.
corr = ds[['mpg', 'displacement', 'horsepower', 'weight',
```

```
'acceleration']].corr()
plt.figure(figsize=(8, 8))
sns.set(font_scale=1.1)
sns.heatmap(data=corr, annot=True, cbar=False, cmap='rainbow',
linewidth=0.5)
plt.title('Correlation Matrix')
plt.show()

# Barplot for cylinders
sns.countplot(x="cylinders",  data=ds)
plt.xlabel('Number of cylinders')
plt.ylabel('Number of vehicles')
plt.title('Cylinders')
plt.show()

# Barplot for model_year
sns.countplot(x="model year", data=ds)
plt.xlabel('Year of manufaction')
plt.ylabel('Number of vehicles')
plt.title('Vehicles per year')
plt.show()

# Barplot for model_year
sns.countplot(x="origin", data=ds)
plt.xlabel('Origin')
plt.ylabel('Number of vehicles')
plt.title('Number of vehicles per origin')
plt.show()

# MPG per Origin
plt.figure(figsize=(8, 6))
fig = sns.boxplot(x='origin', y="mpg", data=ds)
plt.axhline(ds.mpg.mean(), color='r', linestyle='dashed',
            linewidth=2, label=f"MPG Mean:
{ds['mpg'].mean().round(2)}")
plt.xlabel('Origin')
plt.ylabel('Miles per galon')
plt.title('MPG per Origin')
plt.legend()
plt.show()

# MPG per Year
plt.figure(figsize=(8, 8))
fig = sns.boxplot(x='model year', y="mpg", data=ds)
plt.axhline(ds.mpg.mean(), color='r', linestyle='dashed',
            linewidth=2, label=f"MPG Mean:
{ds['mpg'].mean().round(2)}")
plt.xlabel('Year of manufaction')
plt.ylabel('Miles per galon')
plt.title('MPG per Year')
plt.legend()
plt.show()
```

```
# MPG per Number of cylinders
plt.figure(figsize=(8, 8))
fig = sns.boxplot(x='cylinders', y="mpg", data=ds)
plt.axhline(ds.mpg.mean(), color='r', linestyle='dashed',
            linewidth=2, label=f"MPG Mean:
{ds['mpg'].mean().round(2)}")
plt.xlabel('Number of cylinders')
plt.ylabel('Miles per galon')
plt.title('MPG per Number of cylinders')
plt.legend()
plt.show()

# Remove rows from the data set with a null values.
new_ds = ds.dropna()

# Encoding 'cylinders','model year','origin' with get_dummies.
new_ds = pd.get_dummies(new_ds, columns=['origin'])

# Distinguish attribute columns and target column.
X = new_ds.drop(
    columns=['car name', 'mpg', 'horsepower', 'displacement'])
y = new_ds['mpg']

# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=25)

# Standardization
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Score metrics
scores = ['neg_mean_absolute_error', 'neg_mean_squared_error']
# Initialize a Logistic Regression estimator.
linreg = LinearRegression(n_jobs=-1)

# Train the estimator.
linreg.fit(X_train, y_train)

# Make predictions.
lin_pred = linreg.predict(X_test)

# Calculate CV.
cv_lin_reg = cross_validate(linreg, X, y, scoring=scores, cv=10,
n_jobs=-1)

# Initialize a decision tree estimator.
tr = tree.DecisionTreeRegressor(max_depth=5, random_state=25)

# Train the estimator.
tr.fit(X_train, y_train)
```

```
# Make predictions.
tr_pred = tr.predict(X_test)

# Calculate CV scores.
cv_tr_reg = cross_validate(tr, X, y, scoring=scores, cv=10, n_jobs=-1)


# Initialize a Multi-layer Perceptron classifier.
mlp = MLPRegressor(max_iter=1000, hidden_layer_sizes=(100),
                   random_state=25, shuffle=True, verbose=False)

# Train the classifier.
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# Calculate CV
cv_mlp_reg = cross_validate(mlp, X, y, scoring=scores, cv=10, n_jobs=-
1)

# Results
d = {
    'Models': ['Linear Regression', 'Decision Tree', 'Neural Network
(MLP)'],
    'MAE': [mean_absolute_error(y_test, lin_pred),
mean_absolute_error(y_test, tr_pred), mean_absolute_error(y_test,
mlp_pred)],
    'MSE': [mean_squared_error(y_test, lin_pred),
mean_squared_error(y_test, tr_pred), mean_squared_error(y_test,
mlp_pred)],
    'RMSE': np.sqrt([mean_squared_error(y_test, lin_pred),
mean_squared_error(y_test, tr_pred), mean_squared_error(y_test,
mlp_pred)]),
    'CV MAE': [abs(cv_lin_reg['test_neg_mean_absolute_error'].mean()),
abs(cv_tr_reg['test_neg_mean_absolute_error'].mean()),
abs(cv_mlp_reg['test_neg_mean_absolute_error'].mean())],
    'CV MSE': [abs(cv_lin_reg['test_neg_mean_squared_error'].mean()),
abs(cv_tr_reg['test_neg_mean_squared_error'].mean()),
abs(cv_mlp_reg['test_neg_mean_squared_error'].mean())],
    'CV RMSE':
np.sqrt([abs(cv_lin_reg['test_neg_mean_squared_error'].mean()),
abs(cv_tr_reg['test_neg_mean_squared_error'].mean()),
abs(cv_mlp_reg['test_neg_mean_squared_error'].mean())])
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')


# Dictionary with all the Regressors.
```

```
models = {'LinearRegression': linreg,
          'DecisionTreeRegressor': tr,
          'MLPRegressor': mlp
          }


def evaluating_performance(models=None):
    if models == None:
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            mae1 = mean_absolute_error(
                y_test, models[model[0]].predict(X_test))
            mae2 = mean_absolute_error(
                y_test, models[model[1]].predict(X_test))
            var1 = np.var(models[model[0]].predict(X_test))
            var2 = np.var(models[model[1]].predict(X_test))
            n = len(X_test)
            P = abs(mae1-mae2)/np.sqrt((var1+var2)/n)
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'The models {model[0]} and {model[1]} are
significant different')

# ANOVA test
evaluating_performance(models)
```

Σύνολο δεδομένων Car Evaluation χωρίς την χρήση του αλγορίθμου K-μέσων

```
# Import Libraries
import warnings
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn import metrics, tree
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, f1_score,
precision_score, recall_score, roc_auc_score, roc_curve)
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.classifier import ConfusionMatrix
warnings.filterwarnings('ignore')

# Load the data.
file = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/car/car.data'
data = pd.read_csv(file, names=['buying', 'maint', 'doors', 'persons',
'lug_boot', 'safety', 'class_val'])

# Information
print(data.info(), '\n')

# Check for duplicate rows.
print('Duplicate values:', data.duplicated().any(), '\n')

def distploting(df):
    col_value = df.columns.values.tolist()
    sns.set(context='notebook', style='whitegrid',
            font='sans-serif', font_scale=1.3)
    fig, axes = plt.subplots(nrows=2, ncols=3, constrained_layout=True)
    count = 0
    for i in range(2):
        for j in range(3):
            s = col_value[count+j]
            sns.countplot(df[s].values, ax=axes[i][j])
            axes[i][j].set_title(s, fontsize=15)
            fig = plt.gcf()
            fig.set_size_inches(10, 8)
            plt.tight_layout()
        count = count+j+1
    plt.show()

distploting(data)  # .drop(columns=['class_val'])

# Plot class distribution
```

```
sns.countplot(data['class_val'])
plt.title('Class distribution')
plt.show()

# Choose attribute columns and class column.
X = data[data.columns[:-1]]
y = data['class_val']

# OneHot encoder for all the features.
X = pd.get_dummies(X)

# Label encoder for 'class_val'
y = y.replace({'unacc': 0, 'acc': 1, 'good': 2, 'vgood': 3})

# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)

def find_best_neighbors(X, y, tol=0.001, n_iter_no_change=10,
verbose=False):
    scores = []
    n_features = len(X.columns)
    no_improvement_counter = 0
    best_validation_score_ = -np.inf

 # Main loop
    for kappa in range(1, n_features+1):
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=25)
        smt = SMOTE(k_neighbors=kappa, random_state=0)
        X_train, y_train = smt.fit_sample(X_train, y_train)
        scaler = StandardScaler()
        scaler.fit(X_train)
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
    # Initialize a Logistic Regression estimator.
        logreg = LogisticRegression(
            multi_class='auto', random_state=25, n_jobs=-1)
    # Train the estimator.
        logreg.fit(X_train, y_train)
        kappa_score = logreg.score(X_test, y_test)
        scores.append(kappa_score)
        last_valid_score = scores[-1]

        if verbose:
            print(
                f'K={kappa}, No improvement for
{no_improvement_counter} rounds, Best score: {best_validation_score_}')

        if last_valid_score < (best_validation_score_ + tol):
            no_improvement_counter += 1
        else:
            no_improvement_counter = 0
```

```
        if no_improvement_counter > n_iter_no_change:
            break

        if last_valid_score > best_validation_score_:
            best_validation_score_ = last_valid_score

    fig, ax = plt.subplots()
    ax.plot([k for k in range(1, len(scores)+1)],
            scores, marker='o', label="Scores")
    ax.axvline(x=scores.index(best_validation_score_)+1, ymin=0,
ymax=1, color='r', ls='--',
               label=f'Best score \nfor
K={scores.index(best_validation_score_)+1}')
    plt.legend()
    plt.title(
        f'Best score with SMOTE \nfor
K={scores.index(best_validation_score_)+1} neighbors')
    plt.show()


find_best_neighbors(X, y)


# Perform SMOTE
smt = SMOTE(k_neighbors=6, random_state=0)
X_train, y_train = smt.fit_sample(X_train, y_train)

# Standardization
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)

scoring = ['accuracy', 'f1_macro',
           'precision_macro', 'recall_macro', 'roc_auc_ovr']

# Initialize a Logistic Regression classifier.
logreg = LogisticRegression(
    solver='saga', multi_class='auto', random_state=42, n_jobs=-1)

# Train the classifier.
logreg.fit(X_train, y_train)

# Make predictions.
log_pred = logreg.predict(X_test)

# CV score
logreg_cv = cross_validate(logreg, X, y, cv=10, scoring=scoring,
n_jobs=-1)

# Confusion Matrix for Logistic Regression
cm = ConfusionMatrix(logreg, classes=['unacc', 'acc', 'good', 'vgood'],
```

```
                      label_encoder={0: 'unacc', 1: 'acc', 2: 'good', 3:
'vgood'}, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a decision tree estimator.
tr = tree.DecisionTreeClassifier(
    max_depth=5, criterion='gini', random_state=42)

# Train the estimator.
tr.fit(X_train, y_train)

# Make predictions.
tr_pred = tr.predict(X_test)

# CV score
tr_cv = cross_validate(tr, X, y, cv=10, scoring=scoring, n_jobs=-1)

# Print confusion matrix for Decision tree.
cm = ConfusionMatrix(tr, classes=['unacc', 'acc', 'good', 'vgood'],
                      label_encoder={0: 'unacc', 1: 'acc', 2: 'good', 3:
'vgood'}, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a Multi-layer Perceptron classifier.
mlp = MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=1000,
                    random_state=42, shuffle=True, verbose=False)

# Train the classifier.
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# CV score
mlp_cv = cross_validate(mlp, X, y, cv=10, scoring=scoring, n_jobs=-1)

# Plot confusion matrix for MLP.
cm = ConfusionMatrix(mlp, classes=['unacc', 'acc', 'good', 'vgood'],
                      label_encoder={0: 'unacc', 1: 'acc', 2: 'good', 3:
'vgood'}, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()


d = {
    'Models': ['Logistic Regression', 'Decision Tree', 'Neural
Network'],
    'Accuracy': [logreg.score(X_test, y_test), tr.score(X_test,
y_test), mlp.score(X_test, y_test)],
    'CV Accuracy': [logreg_cv['test_accuracy'].mean(),
tr_cv['test_accuracy'].mean(), mlp_cv['test_accuracy'].mean()],
```

```
    'Precision': [precision_score(y_test, log_pred, average='macro'),
precision_score(y_test, tr_pred, average='macro'),
precision_score(y_test, mlp_pred, average='macro')],
    'CV Precision': [logreg_cv['test_precision_macro'].mean(),
tr_cv['test_precision_macro'].mean(),
mlp_cv['test_precision_macro'].mean()],
    'Recall': [recall_score(y_test, log_pred, average='macro'),
recall_score(y_test, tr_pred, average='macro'), recall_score(y_test,
mlp_pred, average='macro')],
    'CV Recall': [logreg_cv['test_recall_macro'].mean(),
tr_cv['test_recall_macro'].mean(), mlp_cv['test_recall_macro'].mean()],
    'F1': [f1_score(y_test, log_pred, average='macro'),
f1_score(y_test, tr_pred, average='macro'), f1_score(y_test, mlp_pred,
average='macro')],
    'CV F1': [logreg_cv['test_f1_macro'].mean(),
tr_cv['test_f1_macro'].mean(), mlp_cv['test_f1_macro'].mean()],
    'AUC': [roc_auc_score(y_test, logreg.predict_proba(X_test),
multi_class='ovr'), roc_auc_score(y_test, tr.predict_proba(X_test),
multi_class='ovr'), roc_auc_score(y_test, mlp.predict_proba(X_test),
multi_class='ovr')],
    'CV AUC': [logreg_cv['test_roc_auc_ovr'].mean(),
tr_cv['test_roc_auc_ovr'].mean(), mlp_cv['test_roc_auc_ovr'].mean()]
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')


# Dictionary with all the classifiers.
models = {'LogisticRegression': logreg,
          'DecisionTreeClassifier': tr,
          'MLPClassifier': mlp
          }


def evaluating_performance2(models=None):
    if models == None:
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            E1 = models[model[0]].score(X_test, y_test)
            E2 = models[model[1]].score(X_test, y_test)
            var1 = E1*(1-E1)
            var2 = E2*(1-E2)
            n = len(X_test)
            P = abs(E1-E2)/np.sqrt((var1+var2)/n)
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
```

```
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'==>The models {model[0]} and {model[1]} are
significant different<==\n')


evaluating_performance2(models)
```

Σύνολο δεδομένων Car Evaluation με την χρήση του αλγορίθμου K-μέσων

```
# Import Libraries
import warnings
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn import metrics, tree
from sklearn.cluster import KMeans
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, f1_score,
precision_score, recall_score, roc_auc_score, roc_curve)
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.classifier import ConfusionMatrix
warnings.filterwarnings('ignore')


# Load the data.
file = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/car/car.data'
data = pd.read_csv(file, names=['buying', 'maint', 'doors', 'persons',
'lug_boot', 'safety', 'class_val'])

# Information
print(data.info(), '\n')

# Check for duplicate rows.
print('Duplicate values:', data.duplicated().any(), '\n')


def distploting(df):
    col_value = df.columns.values.tolist()
    sns.set(context='notebook', style='whitegrid',
            font='sans-serif', font_scale=1.3)

    fig, axes = plt.subplots(nrows=2, ncols=3, constrained_layout=True)
    count = 0
    for i in range(2):
        for j in range(3):
            s = col_value[count+j]
            sns.countplot(df[s].values, ax=axes[i][j])
            axes[i][j].set_title(s, fontsize=15)
            fig = plt.gcf()
            fig.set_size_inches(10, 8)
            plt.tight_layout()
        count = count+j+1
```

```
    plt.show()


distploting(data)  # .drop(columns=['class_val'])

# Plot class distribution
sns.countplot(data['class_val'])
plt.title('Class distribution')
plt.show()

# Choose attribute columns and class column.
X = data[data.columns[:-1]]
y = data['class_val']

# OneHot encoder for all the features.
X = pd.get_dummies(X)

# Label encoder for 'class_val'
y = y.replace({'unacc': 0, 'acc': 1, 'good': 2, 'vgood': 3})


def find_best_n(X, y, tol=0.001, n_iter_no_change=10, n_clusters=2,
verbose=False):
    from sklearn.feature_selection import SelectKBest
    scores = []
    n_features = len(X.columns)
    no_improvement_counter = 0
    best_validation_score_ = -np.inf

 # Main loop
    for kappa in range(1, n_features+1):
        X_new = SelectKBest(k=kappa).fit_transform(X, y)
        sc = StandardScaler()
        sc.fit(X_new)
        X_new = sc.transform(X_new)
        X_new = pd.DataFrame(X_new)
        model = KMeans(n_clusters=2, random_state=25).fit(X_new)
        X_new['cluster'] = model.labels_
        X_train, X_test, y_train, y_test = train_test_split(
            X_new, y, test_size=0.3, random_state=25)
        smt = SMOTE(k_neighbors=6, random_state=0)
        X_train, y_train = smt.fit_sample(X_train, y_train)
        scaler = StandardScaler()
        scaler.fit(X_train)
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
    # Initialize a Logistic Regression estimator.
        logreg = LogisticRegression(
            multi_class='auto', random_state=25, n_jobs=-1)
    # Train the estimator.
        logreg.fit(X_train, y_train)
        kappa_score = logreg.score(X_test, y_test)
        scores.append(kappa_score)
```

```
        last_valid_score = scores[-1]

        if verbose:
            print(
                f'K={kappa}, No improvement for
{no_improvement_counter} rounds, Best score: {best_validation_score_}')

        if last_valid_score < (best_validation_score_ + tol):
            no_improvement_counter += 1
        else:
            no_improvement_counter = 0

        if no_improvement_counter > n_iter_no_change:
            break

        if last_valid_score > best_validation_score_:
            best_validation_score_ = last_valid_score

    fig, ax = plt.subplots()
    ax.plot([k for k in range(1, len(scores)+1)],
            scores, marker='o', label="Scores")
    ax.axvline(x=scores.index(best_validation_score_)+1, ymin=0,
ymax=1, color='r', ls='--',
               label=f'Best score\nfor
N={scores.index(best_validation_score_)+1} features')
    plt.legend()
    plt.title(
        f'Scores of classification with K-means and N features')
    plt.show()


find_best_n(X, y)

X = SelectKBest(k=18).fit_transform(X, y)

# Create a k-means model with k=4.
model = KMeans(n_clusters=4, random_state=25)

# Standardization
sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)
X = pd.DataFrame(X)

X['cluster'] = model.fit_predict(X)

# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
# Perform SMOTE
smt = SMOTE(k_neighbors=6, random_state=0)
X_train, y_train = smt.fit_sample(X_train, y_train)
```

```
# Standardization
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)

scoring = ['accuracy', 'f1_macro',
           'precision_macro', 'recall_macro', 'roc_auc_ovr']

# Initialize a Logistic Regression classifier.
logreg = LogisticRegression(
    solver='saga', multi_class='auto', random_state=42, n_jobs=-1)

# Train the classifier.
logreg.fit(X_train, y_train)

# Make predictions.
log_pred = logreg.predict(X_test)

# CV score
logreg_cv = cross_validate(logreg, X, y, cv=10, scoring=scoring,
n_jobs=-1)

# Confusion Matrix for Logistic Regression
cm = ConfusionMatrix(logreg, classes=['unacc', 'acc', 'good', 'vgood'],
                     label_encoder={0: 'unacc', 1: 'acc', 2: 'good', 3:
'vgood'}, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a decision tree estimator.
tr = tree.DecisionTreeClassifier(
    max_depth=5, criterion='gini', random_state=42)

# Train the estimator.
tr.fit(X_train, y_train)

# Make predictions.
tr_pred = tr.predict(X_test)

# CV score
tr_cv = cross_validate(tr, X, y, cv=10, scoring=scoring, n_jobs=-1)

# Print confusion matrix for Decision tree.
cm = ConfusionMatrix(tr, classes=['unacc', 'acc', 'good', 'vgood'],
                     label_encoder={0: 'unacc', 1: 'acc', 2: 'good', 3:
'vgood'}, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a Multi-layer Perceptron classifier.
mlp = MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=1000,
                    random_state=42, shuffle=True, verbose=False)
```

```
# Train the classifier.
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# CV score
mlp_cv = cross_validate(mlp, X, y, cv=10, scoring=scoring, n_jobs=-1)

# Plot confusion matrix for MLP.
cm = ConfusionMatrix(mlp, classes=['unacc', 'acc', 'good', 'vgood'],
                     label_encoder={0: 'unacc', 1: 'acc', 2: 'good', 3:
'vgood'}, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()


d = {
    'Models': ['Logistic Regression', 'Decision Tree', 'Neural
Network'],
    'Accuracy': [logreg.score(X_test, y_test), tr.score(X_test,
y_test), mlp.score(X_test, y_test)],
    'CV Accuracy': [logreg_cv['test_accuracy'].mean(),
tr_cv['test_accuracy'].mean(), mlp_cv['test_accuracy'].mean()],
    'Precision': [precision_score(y_test, log_pred, average='macro'),
precision_score(y_test, tr_pred, average='macro'),
precision_score(y_test, mlp_pred, average='macro')],
    'CV Precision': [logreg_cv['test_precision_macro'].mean(),
tr_cv['test_precision_macro'].mean(),
mlp_cv['test_precision_macro'].mean()],
    'Recall': [recall_score(y_test, log_pred, average='macro'),
recall_score(y_test, tr_pred, average='macro'), recall_score(y_test,
mlp_pred, average='macro')],
    'CV Recall': [logreg_cv['test_recall_macro'].mean(),
tr_cv['test_recall_macro'].mean(), mlp_cv['test_recall_macro'].mean()],
    'F1': [f1_score(y_test, log_pred, average='macro'),
f1_score(y_test, tr_pred, average='macro'), f1_score(y_test, mlp_pred,
average='macro')],
    'CV F1': [logreg_cv['test_f1_macro'].mean(),
tr_cv['test_f1_macro'].mean(), mlp_cv['test_f1_macro'].mean()],
    'AUC': [roc_auc_score(y_test, logreg.predict_proba(X_test),
multi_class='ovr'), roc_auc_score(y_test, tr.predict_proba(X_test),
multi_class='ovr'), roc_auc_score(y_test, mlp.predict_proba(X_test),
multi_class='ovr')],
    'CV AUC': [logreg_cv['test_roc_auc_ovr'].mean(),
tr_cv['test_roc_auc_ovr'].mean(), mlp_cv['test_roc_auc_ovr'].mean()]
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')
```

```
# Dictionary with all the classifiers.
models = {'LogisticRegression': logreg,
          'DecisionTreeClassifier': tr,
          'MLPClassifier': mlp
          }


def evaluating_performance2(models=None):
    if models == None:
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            E1 = models[model[0]].score(X_test, y_test)
            E2 = models[model[1]].score(X_test, y_test)
            var1 = E1*(1-E1)
            var2 = E2*(1-E2)
            n = len(X_test)
            P = abs(E1-E2)/np.sqrt((var1+var2)/n)
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'==>The models {model[0]} and {model[1]} are
significant different<==\n')


evaluating_performance2(models)
```

Σύνολο δεδομένων Bank Marketing χωρίς την χρήση του αλγόριθμου K-μέσων

```
# Import libraries
import os
import urllib.request
import warnings
import zipfile
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn import metrics, tree
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, f1_score,
precision_score, recall_score, roc_auc_score)
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.classifier import ConfusionMatrix
warnings.filterwarnings('ignore')

# # Download zip file
urllib.request.urlretrieve(
    "https://archive.ics.uci.edu/ml/machine-learning-
databases/00222/bank-additional.zip", 'bank-additional.zip')

# # Extract files.
zip_ref = zipfile.ZipFile('bank-additional.zip', 'r')
zip_ref.extractall('.')
zip_ref.close()

# Load the data.
data = pd.read_csv('bank-additional/bank-additional-full.csv',
sep=';')

# Information
print(data.info())


f, axes = plt.subplots(4, 3, facecolor='white')
ax1 = sns.countplot(data['job'], hue=data['y'],  ax=axes[3, 0])
ax2 = sns.countplot(data['marital'], hue=data['y'],
                    ax=axes[0, 1])
ax3 = sns.countplot(data['education'], hue=data['y'],
                    ax=axes[3, 1])
ax4 = sns.countplot(data['default'], hue=data['y'],
                    ax=axes[1, 0])
ax5 = sns.countplot(data['housing'], hue=data['y'],
                    ax=axes[1, 1])
ax6 = sns.countplot(data['loan'], hue=data['y'],  ax=axes[1, 2])
ax7 = sns.countplot(data['contact'],  hue=data['y'],
```

```
                      ax=axes[2, 0])
ax8 = sns.countplot(data['month'], hue=data['y'],
                      ax=axes[2, 1])
ax9 = sns.countplot(data['poutcome'],  hue=data['y'],
                      ax=axes[2, 2])
ax10 = sns.countplot(data['day_of_week'], hue=data['y'],
                       ax=axes[0, 0])
ax11 = sns.countplot(data['previous'], hue=data['y'],
                       ax=axes[0, 2])
ax12 = sns.countplot(data['pdays'], hue=data['y'],
                       ax=axes[3, 2])
ax1.set_xticklabels(ax1.get_xticklabels(), rotation=45)
ax3.set_xticklabels(ax3.get_xticklabels(), rotation=45)
ax12.set_xticklabels(ax12.get_xticklabels(), rotation=90)
plt.show()

num_columns = ['age', 'duration', 'campaign', 'previous',
'emp.var.rate',
               'cons.price.idx', 'cons.conf.idx', 'euribor3m',
'nr.employed']


f, axes = plt.subplots(3, 3, facecolor='white')
ax1 = sns.distplot(data['age'], kde=True, ax=axes[0, 0])
ax2 = sns.distplot(data['duration'], ax=axes[0, 1])
ax3 = sns.distplot(data['campaign'],  ax=axes[0, 2])
ax5 = sns.distplot(data['emp.var.rate'], ax=axes[1, 1])
ax6 = sns.distplot(data['cons.price.idx'], ax=axes[1, 2])
ax7 = sns.distplot(data['cons.conf.idx'], ax=axes[2, 0])
ax8 = sns.distplot(data['euribor3m'],  ax=axes[2, 1])
ax10 = sns.distplot(data['nr.employed'], ax=axes[1, 0])
plt.show()

# Correlation matrix.
sns.heatmap(data[['age', 'duration', 'campaign', 'previous',
'emp.var.rate', 'cons.price.idx',
                  'cons.conf.idx', 'nr.employed', 'y']].corr(),
annot=True, cbar=False, cmap='rainbow', linewidth=0.5, fmt='.2f')
plt.title('Correlation matrix')
plt.show()

# Replace 'unknown' values from categorical data.
for i in data.columns[data.dtypes == 'object']:
    data[i] = data[i].replace('unknown', np.nan)

# Check for duplicate rows.
print(f"There are {data.duplicated().sum()} duplicate rows in the data
set.")

# Remove duplicate rows.
data = data.drop_duplicates()
print("The duplicate rows were removed.")
```

```
# Remove na values
data = data.dropna()

# OneHotEncoder for categorical data + 'pdays'
features = ['job', 'marital', 'education', 'default', 'housing',
'loan', 'contact', 'month', 'day_of_week',
            'pdays', 'poutcome']
data = pd.get_dummies(data, columns=features)

# Label encoding on column "y".
le = LabelEncoder()
data['y'] = le.fit_transform(data['y'])

# Distinguish attribute columns and class column.
X = data.drop(columns=['nr.employed', 'y'])
y = data['y']


def find_best_neighbor(X, y, tol=0.001, n_iter_no_change=10,
verbose=False):
    scores = []
    n_features = len(X.columns)
    no_improvement_counter = 0
    best_validation_score_ = -np.inf
# Main loop
    for kappa in range(1, 100):
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=25)
        smt = SMOTE(k_neighbors=kappa, random_state=0)
        X_train, y_train = smt.fit_sample(X_train, y_train)
        sc = StandardScaler()
        sc.fit(X_train)
        X_train = sc.transform(X_train)
        X_test = sc.transform(X_test)
    # Initialize a Logistic Regression estimator.
        logreg = LogisticRegression(
            multi_class='auto', random_state=25, n_jobs=-1)
    # Train the estimator.
        logreg.fit(X_train, y_train)
        kappa_score = logreg.score(X_test, y_test)
        scores.append(kappa_score)
        last_valid_score = scores[-1]

        if verbose:
            print(
                f'K={kappa}, No improvement for
{no_improvement_counter} rounds, Best score: {best_validation_score_}')

        if last_valid_score < (best_validation_score_ + tol):
            no_improvement_counter += 1
        else:
            no_improvement_counter = 0
```

```
            if no_improvement_counter > n_iter_no_change:
                break

            if last_valid_score > best_validation_score_:
                best_validation_score_ = last_valid_score

        fig, ax = plt.subplots()
        ax.plot([k for k in range(1, len(scores)+1)],
                scores, marker='o', label="Scores")
        ax.axvline(x=scores.index(best_validation_score_)+1, ymin=0,
ymax=1, color='r', ls='--',
                   label=f'Best score \nfor
k={scores.index(best_validation_score_)+1}')
        plt.legend(loc=3, fontsize='medium')
        plt.title(
            'Performance of S.M.O.TE per number (k) of Neighbors \nfor
Logistic Regression')
        plt.show()


find_best_neighbor(X, y,)

# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=25)

# Perform SMOTE
smt = SMOTE(k_neighbors=6, random_state=0)
X_train, y_train = smt.fit_sample(X_train, y_train)

# Standardization
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

scoring = ['accuracy', 'f1', 'precision', 'recall', 'roc_auc']
# Initialize a Logistic Regression estimator.
logreg = LogisticRegression(multi_class='auto', random_state=25,
n_jobs=-1)

# Train the estimator.
logreg.fit(X_train, y_train)

# Make predictions.
log_pred = logreg.predict(X_test)

# CV score for MLP
cv_logreg = cross_validate(logreg, X, y, scoring=scoring, cv=10)

# Print confusion matrix for Logistic Regression.
cm = ConfusionMatrix(logreg, is_fitted=True)
cm.score(X_test, y_test)
```

```
cm.poof()

# Initialize a decision tree estimator.
tr = tree.DecisionTreeClassifier(max_depth=3, random_state=25)

# Train the estimator.
tr.fit(X_train, y_train)

# Make predictions.
tr_pred = tr.predict(X_test)

# CV scores for Decision tree
cv_tr = cross_validate(tr, X, y, scoring=scoring, cv=10)

# Print confusion matrix for Decision tree.
cm = ConfusionMatrix(tr, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a Multi-layer Perceptron classifier.
mlp = MLPClassifier(hidden_layer_sizes=(10), max_iter=1000,
                    random_state=25, shuffle=True)

# Train the classifier.
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# CV score for MLP
cv_mlp = cross_validate(mlp, X, y, cv=10, scoring=scoring, n_jobs=-1)

# Print confusion matrix for Decision tree.
cm = ConfusionMatrix(mlp, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

d = {
    'Models': ['Logistic Regression', 'Decision Tree', 'Neural
Network'],
    'Accuracy': [logreg.score(X_test, y_test), tr.score(X_test,
y_test), mlp.score(X_test, y_test)],
    'CV Accuracy': [cv_logreg['test_accuracy'].mean(),
cv_tr['test_accuracy'].mean(), cv_mlp['test_accuracy'].mean()],
    'Precision': [precision_score(y_test, log_pred),
precision_score(y_test, tr_pred), precision_score(y_test, mlp_pred)],
    'CV Precision': [cv_logreg['test_precision'].mean(),
cv_tr['test_precision'].mean(), cv_mlp['test_precision'].mean()],
    'Recall': [recall_score(y_test, log_pred), recall_score(y_test,
tr_pred), recall_score(y_test, mlp_pred)],
    'CV Recall': [cv_logreg['test_recall'].mean(),
cv_tr['test_recall'].mean(), cv_mlp['test_recall'].mean()],
    'F1': [f1_score(y_test, log_pred), f1_score(y_test, tr_pred),
```

```
f1_score(y_test, mlp_pred)],
    'CV F1': [cv_logreg['test_f1'].mean(), cv_tr['test_f1'].mean(),
cv_mlp['test_f1'].mean()],
    'AUC': [roc_auc_score(y_test, logreg.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, tr.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, mlp.predict_proba(X_test)[:, 1])],
    'CV AUC': [cv_logreg['test_roc_auc'].mean(),
cv_tr['test_roc_auc'].mean(), cv_mlp['test_roc_auc'].mean()]
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')


# Dictionary with all the classifiers.
models = {'LogisticRegression': logreg,
          'DecisionTreeClassifier': tr,
          'MLPClassifier': mlp
          }


def evaluating_performance2(models=None):
    if models == None:
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            E1 = models[model[0]].score(X_test, y_test)
            E2 = models[model[1]].score(X_test, y_test)
            var1 = E1*(1-E1)
            var2 = E2*(1-E2)
            n = len(X_test)
            P = abs(E1-E2)/np.sqrt((var1+var2)/n)
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'==>The models {model[0]} and {model[1]} are
significant different<==\n')


evaluating_performance2(models)
```

Σύνολο δεδομένων Bank Marketing με την χρήση του αλγόριθμου K-μέσων

```
# Import Libraries
import os
import urllib.request
import warnings
import zipfile
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn import metrics, tree
from sklearn.cluster import KMeans
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, f1_score,
precision_score, recall_score, roc_auc_score)
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.classifier import ConfusionMatrix
warnings.filterwarnings('ignore')

# Download zip file
urllib.request.urlretrieve(
    "https://archive.ics.uci.edu/ml/machine-learning-
databases/00222/bank-additional.zip", 'bank-additional.zip')

# Extract files.
zip_ref = zipfile.ZipFile('bank-additional.zip', 'r')
zip_ref.extractall('.')
zip_ref.close()

# Load the data.
data = pd.read_csv('bank-additional/bank-additional-full.csv',
sep=';')

# Information
print(data.info())

f, axes = plt.subplots(4, 3, facecolor='white')
ax1 = sns.countplot(data['job'], hue=data['y'],  ax=axes[3, 0])
ax2 = sns.countplot(data['marital'], hue=data['y'],
                    ax=axes[0, 1])
ax3 = sns.countplot(data['education'], hue=data['y'],
                    ax=axes[3, 1])
ax4 = sns.countplot(data['default'], hue=data['y'],
                    ax=axes[1, 0])
ax5 = sns.countplot(data['housing'], hue=data['y'],
                    ax=axes[1, 1])
ax6 = sns.countplot(data['loan'], hue=data['y'],  ax=axes[1, 2])
```

```
ax7 = sns.countplot(data['contact'],  hue=data['y'],
                    ax=axes[2, 0])
ax8 = sns.countplot(data['month'], hue=data['y'],
                    ax=axes[2, 1])
ax9 = sns.countplot(data['poutcome'],  hue=data['y'],
                    ax=axes[2, 2])
ax10 = sns.countplot(data['day_of_week'], hue=data['y'],
                     ax=axes[0, 0])
ax11 = sns.countplot(data['previous'], hue=data['y'],
                     ax=axes[0, 2])
ax12 = sns.countplot(data['pdays'], hue=data['y'],
                     ax=axes[3, 2])
ax1.set_xticklabels(ax1.get_xticklabels(), rotation=45)
ax3.set_xticklabels(ax3.get_xticklabels(), rotation=45)
ax12.set_xticklabels(ax12.get_xticklabels(), rotation=90)
plt.show()

num_columns = ['age', 'duration', 'campaign', 'previous',
'emp.var.rate',
               'cons.price.idx', 'cons.conf.idx', 'euribor3m',
'nr.employed']


f, axes = plt.subplots(3, 3, facecolor='white')
ax1 = sns.distplot(data['age'], kde=True, ax=axes[0, 0])
ax2 = sns.distplot(data['duration'], ax=axes[0, 1])
ax3 = sns.distplot(data['campaign'],  ax=axes[0, 2])
ax5 = sns.distplot(data['emp.var.rate'], ax=axes[1, 1])
ax6 = sns.distplot(data['cons.price.idx'], ax=axes[1, 2])
ax7 = sns.distplot(data['cons.conf.idx'], ax=axes[2, 0])
ax8 = sns.distplot(data['euribor3m'],  ax=axes[2, 1])
ax10 = sns.distplot(data['nr.employed'], ax=axes[1, 0])
plt.show()

# Correlation matrix.
sns.heatmap(data[['age', 'duration', 'campaign', 'previous',
'emp.var.rate', 'cons.price.idx',
                  'cons.conf.idx', 'nr.employed', 'y']].corr(),
annot=True, cbar=False, cmap='rainbow', linewidth=0.5, fmt='.2f')
plt.title('Correlation matrix')
plt.show()

# Replace 'unknown' values from categorical data.
for i in data.columns[data.dtypes == 'object']:
    data[i] = data[i].replace('unknown', np.nan)

# Check for duplicate rows.
print(f"There are {data.duplicated().sum()} duplicate rows in the data
set.")

# Remove duplicate rows.
data = data.drop_duplicates()
print("The duplicate rows were removed.")
```

```
# Remove na values
data = data.dropna()

# OneHotEncoder for categorical data + 'pdays'
features = ['job', 'marital', 'education', 'default', 'housing',
'loan', 'contact', 'month', 'day_of_week',
            'pdays', 'poutcome']
data = pd.get_dummies(data, columns=features)

# Label encoding on column "y".
le = LabelEncoder()
data['y'] = le.fit_transform(data['y'])

# Distinguish attribute columns and class column.
X = data.drop(columns=['nr.employed', 'y'])
y = data['y']


def find_best_k(X, y, tol=0.001, n_iter_no_change=10, n_clusters=2,
verbose=False):
    from sklearn.feature_selection import SelectKBest
    scores = []
    n_features = len(X.columns)
    no_improvement_counter = 0
    best_validation_score_ = -np.inf

 # Main Loop
    for kappa in range(1, n_features+1):
        X_new = SelectKBest(k=kappa).fit_transform(X, y)
        sc = StandardScaler()
        sc.fit(X_new)
        X_new = sc.transform(X_new)
        X_new = pd.DataFrame(X_new)
        model = KMeans(n_clusters=2, random_state=25).fit(X_new)
        X_new['cluster'] = model.labels_
        X_train, X_test, y_train, y_test = train_test_split(
            X_new, y, test_size=0.3, random_state=25)
        scaler = StandardScaler()
        scaler.fit(X_train)
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
    # Initialize a Logistic Regression estimator.
        logreg = LogisticRegression(
            multi_class='auto', random_state=25, n_jobs=-1)
    # Train the estimator.
        logreg.fit(X_train, y_train)
        kappa_score = logreg.score(X_test, y_test)
        scores.append(kappa_score)
        last_valid_score = scores[-1]

        if verbose:
            print(
```

```
                f'K={kappa}, No improvement for
{no_improvement_counter} rounds, Best score: {best_validation_score_}')

        if last_valid_score < (best_validation_score_ + tol):
            no_improvement_counter += 1
        else:
            no_improvement_counter = 0

        if no_improvement_counter > n_iter_no_change:
            break

        if last_valid_score > best_validation_score_:
            best_validation_score_ = last_valid_score

    fig, ax = plt.subplots()
    ax.plot([k for k in range(1, len(scores)+1)],
            scores, marker='o', label="Scores")
    ax.axvline(x=scores.index(best_validation_score_)+1, ymin=0,
ymax=1, color='r', ls='--',
               label=f'Best score\nfor
N={scores.index(best_validation_score_)+1} features')
    plt.legend()
    plt.title('Scores of Logistic Regression with K-means and N
features')
    plt.show()


find_best_k(X, y)

X = SelectKBest(k=14).fit_transform(X, y)

# Create a k-means model with k=2
model = KMeans(n_clusters=2, random_state=25)

# Standardization
sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)
X = pd.DataFrame(X)

X['cluster'] = model.fit_predict(X)

# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=25)

# Perform SMOTE
smt = SMOTE(k_neighbors=6, random_state=0)
X_train, y_train = smt.fit_sample(X_train, y_train)

# Standardization
scaler = StandardScaler()
scaler.fit(X_train)
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

scoring = ['accuracy', 'f1', 'precision', 'recall', 'roc_auc']
# Initialize a Logistic Regression estimator.
logreg = LogisticRegression(multi_class='auto', random_state=25,
n_jobs=-1)

# Train the estimator.
logreg.fit(X_train, y_train)

# Make predictions.
log_pred = logreg.predict(X_test)

# CV score for MLP
cv_logreg = cross_validate(logreg, X, y, scoring=scoring, cv=10)

# Print confusion matrix for Logistic Regression.
cm = ConfusionMatrix(logreg, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a decision tree estimator.
tr = tree.DecisionTreeClassifier(max_depth=3, random_state=25)

# Train the estimator.
tr.fit(X_train, y_train)

# Make predictions.
tr_pred = tr.predict(X_test)

# CV scores for Decision tree
cv_tr = cross_validate(tr, X, y, scoring=scoring, cv=10)

# Print confusion matrix for Decision tree.
cm = ConfusionMatrix(tr, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a Multi-layer Perceptron classifier.
mlp = MLPClassifier(hidden_layer_sizes=(10), max_iter=1000,
                    random_state=25, shuffle=True)

# Train the classifier.
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# CV score for MLP
cv_mlp = cross_validate(mlp, X, y, cv=10, scoring=scoring, n_jobs=-1)

# Print confusion matrix for Decision tree.
```

```
cm = ConfusionMatrix(mlp, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

d = {
    'Models': ['Logistic Regression', 'Decision Tree', 'Neural
Network'],
    'Accuracy': [logreg.score(X_test, y_test), tr.score(X_test,
y_test), mlp.score(X_test, y_test)],
    'CV Accuracy': [cv_logreg['test_accuracy'].mean(),
cv_tr['test_accuracy'].mean(), cv_mlp['test_accuracy'].mean()],
    'Precision': [precision_score(y_test, log_pred),
precision_score(y_test, tr_pred), precision_score(y_test, mlp_pred)],
    'CV Precision': [cv_logreg['test_precision'].mean(),
cv_tr['test_precision'].mean(), cv_mlp['test_precision'].mean()],
    'Recall': [recall_score(y_test, log_pred), recall_score(y_test,
tr_pred), recall_score(y_test, mlp_pred)],
    'CV Recall': [cv_logreg['test_recall'].mean(),
cv_tr['test_recall'].mean(), cv_mlp['test_recall'].mean()],
    'F1': [f1_score(y_test, log_pred), f1_score(y_test, tr_pred),
f1_score(y_test, mlp_pred)],
    'CV F1': [cv_logreg['test_f1'].mean(), cv_tr['test_f1'].mean(),
cv_mlp['test_f1'].mean()],
    'AUC': [roc_auc_score(y_test, logreg.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, tr.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, mlp.predict_proba(X_test)[:, 1])],
    'CV AUC': [cv_logreg['test_roc_auc'].mean(),
cv_tr['test_roc_auc'].mean(), cv_mlp['test_roc_auc'].mean()]
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')


# Dictionary with all the classifiers.
models = {'LogisticRegression': logreg,
          'DecisionTreeClassifier': tr,
          'MLPClassifier': mlp
          }


def evaluating_performance2(models=None):
    if models == None:
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            E1 = models[model[0]].score(X_test, y_test)
            E2 = models[model[1]].score(X_test, y_test)
            var1 = E1*(1-E1)
            var2 = E2*(1-E2)
            n = len(X_test)
            P = abs(E1-E2)/np.sqrt((var1+var2)/n)
```

```
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'==>The models {model[0]} and {model[1]} are
significant different<==\n')


evaluating_performance2(models)
```

Σύνολο δεδομένων Default of credit card clients χωρίς την χρήση του αλγόριθμου Κ-μέσων

```
#Import libraries
import os
import warnings
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics, tree
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, f1_score,
precision_score, recall_score, roc_auc_score)
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.classifier import ConfusionMatrix
warnings.filterwarnings('ignore')

# Load the data.
data = pd.read_excel(
    'https://archive.ics.uci.edu/ml/machine-learning-
databases/00350/default%20of%20credit%20card%20clients.xls', header=1,
index_col=0)


data = data.rename(
    columns={'PAY_0': 'PAY_1', 'default payment next month': 'dpnm'})

# Information
print(data.info())

# Replace value '0' with value '3'.
data['MARRIAGE'] = data['MARRIAGE'].replace(0, 3)

# Replace values '0','5' and '6' with value '4'.
data['EDUCATION'] = data['EDUCATION'].replace([0, 5, 6], 4)

# The frequency of defaults
yes = data.dpnm.sum()
no = len(data)-yes

# Percentage
yes_perc = round(yes/len(data)*100, 1)
no_perc = round(no/len(data)*100, 1)

sns.countplot(data['dpnm'])
plt.annotate('Non-default: {}'.format(no),
             xy=(-0.3, 15000), xytext=(-0.3, 3000), size=12)
plt.annotate('Default: {}'.format(yes), xy=(
```

```
    0.7, 15000), xytext=(0.7, 3000), size=12)
plt.annotate(str(no_perc)+" %", xy=(-0.3, 15000), xytext=(-0.1, 8000),
size=14)
plt.annotate(str(yes_perc)+" %", xy=(0.7, 15000), xytext=(0.9, 8000),
size=14)
plt.title('Value distribution for "dpnm"')
plt.show()

# Creating a new dataframe with categorical variables
subset = data[['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_1', 'PAY_2',
'PAY_3', 'PAY_4',
               'PAY_5', 'PAY_6', 'dpnm']]

f, axes = plt.subplots(3, 3, figsize=(
    20, 15), facecolor='white', constrained_layout=False)
ax1 = sns.countplot(x="SEX", hue="dpnm", data=data,  ax=axes[0, 0])
ax2 = sns.countplot(x="EDUCATION", hue="dpnm", data=data, ax=axes[0,
1])
ax3 = sns.countplot(x="MARRIAGE", hue="dpnm", data=data, ax=axes[0, 2])
ax4 = sns.countplot(x="PAY_1", hue="dpnm", data=data,  ax=axes[1, 0])
ax5 = sns.countplot(x="PAY_2", hue="dpnm", data=data,  ax=axes[1, 1])
ax6 = sns.countplot(x="PAY_3", hue="dpnm", data=data,  ax=axes[1, 2])
ax7 = sns.countplot(x="PAY_4", hue="dpnm", data=data,  ax=axes[2, 0])
ax8 = sns.countplot(x="PAY_5", hue="dpnm", data=data,  ax=axes[2, 1])
ax9 = sns.countplot(x="PAY_6", hue="dpnm", data=data, ax=axes[2, 2])
plt.show()

# Correlation matrix
plt.figure(figsize=(20, 20))
sns.heatmap(data.corr(), annot=True, cmap='rainbow', linewidth=0.5,
fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Check for duplicate rows.
print(f"There are {data.duplicated().sum()} duplicate rows in the data
set.")

# Remove duplicate rows.
data = data.drop_duplicates()
print("The duplicate rows were removed.")

# OneHot encoder for columns 'EDUCATION','MARRIAGE','SEX','PAY_1',
'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6' .
data = pd.get_dummies(
    data, columns=['EDUCATION', 'MARRIAGE', 'SEX', 'PAY_1', 'PAY_2',
'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'])

# Select feature and class column.
X = data.drop(columns=['dpnm', 'BILL_AMT2', 'BILL_AMT3',
                       'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6'])
y = data['dpnm']
```

```
# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=25)

# Standardization
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

scoring = ['accuracy', 'f1', 'precision', 'recall', 'roc_auc']

# Initialize a Logistic Regression estimator.
logreg = LogisticRegression(
    multi_class='auto', random_state=25, solver='sag', n_jobs=-1)

# Train the estimator.
logreg.fit(X_train, y_train)

# Make predictions.
log_pred = logreg.predict(X_test)

# CV accuracy
cv_logreg = cross_validate(logreg, X, y, scoring=scoring, cv=10)

# Confusion matrix for Logitisc Regression
cm = ConfusionMatrix(logreg, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a decision tree estimator.
tr = tree.DecisionTreeClassifier(
    max_depth=5, criterion='gini', random_state=25)

# Train the estimator.
tr.fit(X_train, y_train)

# Make predictions.
tr_pred = tr.predict(X_test)

# CV accuracy.
cv_tr = cross_validate(tr, X, y, scoring=scoring, cv=10)

# Confusion matrix for Decision Tree
cm = ConfusionMatrix(tr, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a Multi-Layer Perceptron classifier.
mlp = MLPClassifier(hidden_layer_sizes=(12, 5), max_iter=1000,
                    random_state=25, shuffle=True, verbose=False)

# Train the classifier.
```

```
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# CV accuracy
cv_mlp = cross_validate(mlp, X, y, scoring=scoring, cv=10)

# Confusion matrix for MLP.
cm = ConfusionMatrix(mlp, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

d = {
    'Models': ['Logistic Regression', 'Decision Tree', 'Neural
Network'],
    'Accuracy': [logreg.score(X_test, y_test), tr.score(X_test,
y_test), mlp.score(X_test, y_test)],
    'CV Accuracy': [cv_logreg['test_accuracy'].mean(),
cv_tr['test_accuracy'].mean(), cv_mlp['test_accuracy'].mean()],
    'Precision': [precision_score(y_test, log_pred),
precision_score(y_test, tr_pred), precision_score(y_test, mlp_pred)],
    'CV Precision': [cv_logreg['test_precision'].mean(),
cv_tr['test_precision'].mean(), cv_mlp['test_precision'].mean()],
    'Recall': [recall_score(y_test, log_pred), recall_score(y_test,
tr_pred), recall_score(y_test, mlp_pred)],
    'CV Recall': [cv_logreg['test_recall'].mean(),
cv_tr['test_recall'].mean(), cv_mlp['test_recall'].mean()],
    'F1': [f1_score(y_test, log_pred), f1_score(y_test, tr_pred),
f1_score(y_test, mlp_pred)],
    'CV F1': [cv_logreg['test_f1'].mean(), cv_tr['test_f1'].mean(),
cv_mlp['test_f1'].mean()],
    'AUC': [roc_auc_score(y_test, logreg.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, tr.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, mlp.predict_proba(X_test)[:, 1])],
    'CV AUC': [cv_logreg['test_roc_auc'].mean(),
cv_tr['test_roc_auc'].mean(), cv_mlp['test_roc_auc'].mean()]
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')


# Dictionary with all the classifiers.
models = {'LogisticRegression': logreg,
          'DecisionTreeClassifier': tr,
          'MLPClassifier': mlp
          }


def evaluating_performance2(models=None):
    if models == None:
```

```
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            E1 = models[model[0]].score(X_test, y_test)
            E2 = models[model[1]].score(X_test, y_test)
            var1 = E1*(1-E1)
            var2 = E2*(1-E2)
            n = len(X_test)
            P = abs(E1-E2)/np.sqrt((var1+var2)/n)
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'==>The models {model[0]} and {model[1]} are
significant different<==\n')


evaluating_performance2(models)
```

Σύνολο δεδομένων Default of credit card clients με την χρήση του αλγόριθμου K-μέσων

```
#Import libraries
import os
import warnings
from itertools import combinations
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics, tree
from sklearn.cluster import KMeans
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, f1_score,
precision_score, recall_score, roc_auc_score)
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.classifier import ConfusionMatrix
warnings.filterwarnings('ignore')

# Load the data.
data = pd.read_excel(
    'https://archive.ics.uci.edu/ml/machine-learning-
databases/00350/default%20of%20credit%20card%20clients.xls', header=1,
index_col=0)


data = data.rename(
    columns={'PAY_0': 'PAY_1', 'default payment next month': 'dpnm'})

# Information
print(data.info())

# Replace value '0' with value '3'.
data['MARRIAGE'] = data['MARRIAGE'].replace(0, 3)

# Replace values '0','5' and '6' with value '4'.
data['EDUCATION'] = data['EDUCATION'].replace([0, 5, 6], 4)

# The frequency of defaults
yes = data.dpnm.sum()
no = len(data)-yes

# Percentage
yes_perc = round(yes/len(data)*100, 1)
no_perc = round(no/len(data)*100, 1)

sns.countplot(data['dpnm'])
plt.annotate('Non-default: {}'.format(no),
             xy=(-0.3, 15000), xytext=(-0.3, 3000), size=12)
```

```
plt.annotate('Default: {}'.format(yes), xy=(
    0.7, 15000), xytext=(0.7, 3000), size=12)
plt.annotate(str(no_perc)+" %", xy=(-0.3, 15000), xytext=(-0.1, 8000),
size=14)
plt.annotate(str(yes_perc)+" %", xy=(0.7, 15000), xytext=(0.9, 8000),
size=14)
plt.title('Value distribution for "dpnm"')
plt.show()

# Creating a new dataframe with categorical variables
subset = data[['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_1', 'PAY_2',
'PAY_3', 'PAY_4',
               'PAY_5', 'PAY_6', 'dpnm']]

f, axes = plt.subplots(3, 3, figsize=(
    20, 15), facecolor='white', constrained_layout=False)
ax1 = sns.countplot(x="SEX", hue="dpnm", data=data,  ax=axes[0, 0])
ax2 = sns.countplot(x="EDUCATION", hue="dpnm", data=data, ax=axes[0,
1])
ax3 = sns.countplot(x="MARRIAGE", hue="dpnm", data=data, ax=axes[0, 2])
ax4 = sns.countplot(x="PAY_1", hue="dpnm", data=data,  ax=axes[1, 0])
ax5 = sns.countplot(x="PAY_2", hue="dpnm", data=data,  ax=axes[1, 1])
ax6 = sns.countplot(x="PAY_3", hue="dpnm", data=data,  ax=axes[1, 2])
ax7 = sns.countplot(x="PAY_4", hue="dpnm", data=data,  ax=axes[2, 0])
ax8 = sns.countplot(x="PAY_5", hue="dpnm", data=data,  ax=axes[2, 1])
ax9 = sns.countplot(x="PAY_6", hue="dpnm", data=data, ax=axes[2, 2])
plt.show()

# Correlation matrix
plt.figure(figsize=(20, 20))
sns.heatmap(data.corr(), annot=True, cmap='rainbow', linewidth=0.5,
fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Check for duplicate rows.
print(f"There are {data.duplicated().sum()} duplicate rows in the data
set.")

# Remove duplicate rows.
data = data.drop_duplicates()
print("The duplicate rows were removed.")

# OneHot encoder for columns 'EDUCATION','MARRIAGE','SEX','PAY_1',
'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6' .
data = pd.get_dummies(
    data, columns=['EDUCATION', 'MARRIAGE', 'SEX', 'PAY_1', 'PAY_2',
'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'])

# Select feature and class column.
X = data.drop(columns=['dpnm', 'BILL_AMT2', 'BILL_AMT3',
                       'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6'])
y = data['dpnm']
```

```
def find_best_k(X, y, tol=0.001, n_iter_no_change=10, n_clusters=2,
verbose=False):
    from sklearn.feature_selection import SelectKBest
    scores = []
    n_features = len(X.columns)
    no_improvement_counter = 0
    best_validation_score_ = -np.inf

 # Main loop
    for kappa in range(1, n_features+1):
        X_new = SelectKBest(k=kappa).fit_transform(X, y)
        sc = StandardScaler()
        sc.fit(X_new)
        X_new = sc.transform(X_new)
        X_new = pd.DataFrame(X_new)
        model = KMeans(n_clusters=2, random_state=25).fit(X_new)
        X_new['cluster'] = model.labels_
        X_train, X_test, y_train, y_test = train_test_split(
            X_new, y, test_size=0.3, random_state=25)
        scaler = StandardScaler()
        scaler.fit(X_train)
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
    # Initialize a Logistic Regression estimator.
        logreg = LogisticRegression(
            multi_class='auto', random_state=25, n_jobs=-1)
    # Train the estimator.
        logreg.fit(X_train, y_train)
        kappa_score = logreg.score(X_test, y_test)
#         cross_val_score(logreg, X_new, y, cv=10).mean()
        scores.append(kappa_score)
        last_valid_score = scores[-1]

        if verbose:
            print(
                f'K={kappa}, No improvement for
{no_improvement_counter} rounds, Best score: {best_validation_score_}')

        if last_valid_score < (best_validation_score_ + tol):
            no_improvement_counter += 1
        else:
            no_improvement_counter = 0

        if no_improvement_counter > n_iter_no_change:
            break

        if last_valid_score > best_validation_score_:
            best_validation_score_ = last_valid_score

    fig, ax = plt.subplots()
    ax.plot([k for k in range(1, len(scores)+1)],
```

```
            scores, marker='o', label="Scores")
    ax.axvline(x=scores.index(best_validation_score_)+1, ymin=0,
ymax=1, color='r', ls='--',
               label=f'Best score\nfor
N={scores.index(best_validation_score_)+1} features')
    plt.legend()
    plt.title(
        f'Scores for Logistic Regression with K-means and N features')
    plt.show()


find_best_k(X, y)

X = SelectKBest(k=15).fit_transform(X, y)

# Standardization
sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)
X = pd.DataFrame(X)

model = KMeans(n_clusters=2, random_state=25).fit(X)
X['cluster'] = model.fit_predict(X)

# Split to train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=25)

# Standardization
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

scoring = ['accuracy', 'f1', 'precision', 'recall', 'roc_auc']

# Initialize a Logistic Regression estimator.
logreg = LogisticRegression(
    multi_class='auto', random_state=25, solver='sag', n_jobs=-1)

# Train the estimator.
logreg.fit(X_train, y_train)

# Make predictions.
log_pred = logreg.predict(X_test)

# CV accuracy
cv_logreg = cross_validate(logreg, X, y, scoring=scoring, cv=10)

# Confusion matrix for Logitisc Regression
cm = ConfusionMatrix(logreg, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()
```

```
# Initialize a decision tree estimator.
tr = tree.DecisionTreeClassifier(
    max_depth=5, criterion='gini', random_state=25)

# Train the estimator.
tr.fit(X_train, y_train)

# Make predictions.
tr_pred = tr.predict(X_test)

# CV accuracy.
cv_tr = cross_validate(tr, X, y, scoring=scoring, cv=10)

# Confusion matrix for Decision Tree
cm = ConfusionMatrix(tr, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

# Initialize a Multi-layer Perceptron classifier.
mlp = MLPClassifier(hidden_layer_sizes=(12, 5), max_iter=1000,
                    random_state=25, shuffle=True, verbose=False)

# Train the classifier.
mlp.fit(X_train, y_train)

# Make predictions.
mlp_pred = mlp.predict(X_test)

# CV accuracy
cv_mlp = cross_validate(mlp, X, y, scoring=scoring, cv=10)

# Confusion matrix for MLP.
cm = ConfusionMatrix(mlp, is_fitted=True)
cm.score(X_test, y_test)
cm.poof()

d = {
    'Models': ['Logistic Regression', 'Decision Tree', 'Neural
Network'],
    'Accuracy': [logreg.score(X_test, y_test), tr.score(X_test,
y_test), mlp.score(X_test, y_test)],
    'CV Accuracy': [cv_logreg['test_accuracy'].mean(),
cv_tr['test_accuracy'].mean(), cv_mlp['test_accuracy'].mean()],
    'Precision': [precision_score(y_test, log_pred),
precision_score(y_test, tr_pred), precision_score(y_test, mlp_pred)],
    'CV Precision': [cv_logreg['test_precision'].mean(),
cv_tr['test_precision'].mean(), cv_mlp['test_precision'].mean()],
    'Recall': [recall_score(y_test, log_pred), recall_score(y_test,
tr_pred), recall_score(y_test, mlp_pred)],
    'CV Recall': [cv_logreg['test_recall'].mean(),
cv_tr['test_recall'].mean(), cv_mlp['test_recall'].mean()],
    'F1': [f1_score(y_test, log_pred), f1_score(y_test, tr_pred),
```

```
f1_score(y_test, mlp_pred)],
    'CV F1': [cv_logreg['test_f1'].mean(), cv_tr['test_f1'].mean(),
cv_mlp['test_f1'].mean()],
    'AUC': [roc_auc_score(y_test, logreg.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, tr.predict_proba(X_test)[:, 1]),
roc_auc_score(y_test, mlp.predict_proba(X_test)[:, 1])],
    'CV AUC': [cv_logreg['test_roc_auc'].mean(),
cv_tr['test_roc_auc'].mean(), cv_mlp['test_roc_auc'].mean()]
}

results = pd.DataFrame(data=d).round(3).set_index('Models')
print('\n', '========RESULTS========', '\n')
print(results, '\n')


# Dictionary with all the classifiers.
models = {'LogisticRegression': logreg,
          'DecisionTreeClassifier': tr,
          'MLPClassifier': mlp
          }


def evaluating_performance2(models=None):
    if models == None:
        return 'No models to compare'
    else:
        for model in combinations(models.keys(), 2):
            E1 = models[model[0]].score(X_test, y_test)
            E2 = models[model[1]].score(X_test, y_test)
            var1 = E1*(1-E1)
            var2 = E2*(1-E2)
            n = len(X_test)
            P = abs(E1-E2)/np.sqrt((var1+var2)/n)
            print(
                f'Comparing Performance between {model[0]} and
{model[1]}:', P.round(3))
            if P < 2:
                print(
                    f'There is no significant diferrence between
{model[0]} and {model[1]}\n')
            else:
                print(
                    f'==>The models {model[0]} and {model[1]} are
significant different<==\n')


evaluating_performance2(models)
```