ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΡΟΙΤΟΡ ΚΑΤΑΡΤΖΙΟΥ ΙΩΑΝ Π21077

ΑΠΑΛΛΑΚΤΙΚΗ ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ ΜΑΘΗΜΑΤΟΣ ΑΝΑΛΥΤΙΚΗ ΔΕΔΟΜΕΝΩΝ

ΠΕΙΡΑΙΑΣ

Σεπτέμβριος 2024

ΠΡΟΛΟΓΟΣ

Η παρούσα εργασία αναπτύχθηκε ως μέρος του μαθήματος Αναλυτική Δεδομένων, με κύριο στόχο την εξοικείωση με ένα πραγματικό σύνολο δεδομένων και η εφαρμογή τεχνικών Αναλυτικής Δεδομένων πάνω σε αυτό.

ΕΚΦΩΝΗΣΗ

Σκοπός της εργασίας είναι η εξοικείωση με ένα πραγματικό σύνολο δεδομένων και η εφαρμογή τεχνικών Αναλυτικής Δεδομένων πάνω σε αυτό. Για τον σκοπό αυτό θα επιλέξετε το παρακάτω σύνολο δεδομένων:

- Bank Marketing Data Set (https://archive.ics.uci.edu/ml/datasets/bank+marketing). Αποτελείται από περίπου 45 χιλιάδες εγγραφές, και περιέχει πληροφορίες σχετικά με εκστρατείες άμεσου μάρκετινγκ (τηλεφωνικές κλήσεις) ενός Πορτογαλικού τραπεζικού ιδρύματος.

Βήμα 1: Προπαρασκευή δεδομένων (Data proprocessing)

Το πρώτο είναι η εξοικείωση του ερευνητή με τα δεδομένα του. Αφού κατεβάσετε το παραπάνω σύνολο δεδομένων, προχωρήστε σε όποια προπαρασκευαστική εργασία (επιλογή, οπτικοποίηση, καθαρισμό, μετασχηματισμό, δειγματοληψία, κλπ.) θεωρείτε απαραίτητη ώστε: α) να «καθαρίσετε» τα δεδομένα από ελλιπείς ή εσφαλμένες τιμές, εάν υπάρχουν (π.χ., συμπλήρωση κενών πεδίων, απαλοιφή ακραίων τιμών), β) να κανονικοποιήσετε – διακριτοποιήσετε τα δεδομένα (π.χ. για αντιμετώπιση των συνεχών πεδίων τιμών), γ) να μειώσετε τον όγκο των δεδομένων (π.χ. μείωση διαστάσεων). Επίσης θα πρέπει να κάνετε μια απλή στατιστική ανάλυση, σε μορφή ιστογραμμάτων, box plots κλπ., των πιο βασικών (κατά τη γνώμη σας) χαρακτηριστικών του συνόλου δεδομένων.

Βήμα 2: Συσταδοποίηση (Clustering)

Έχοντας εξοικειωθεί με το σύνολο δεδομένων, το επόμενο βήμα της πειραματικής σας διαδικασίας είναι η χρήση τεχνικών συσταδοποίησης, προκειμένου να ανακαλύψετε ιδιότητες του συνόλου και πρότυπα που δεν είναι προφανή με μια απλή στατιστική ανάλυση. Σε αυτό το στάδιο, σημαντικό ρόλο παίζει η μοντελοποίηση του προβλήματος (τι ακριβώς ψάχνετε να εντοπίσετε). Διαδικαστικά, αφού επιλέξετε (α) τα χαρακτηριστικά του συνόλου δεδομένων τα οποία θα αποφασίσετε να εξετάσετε και (β) μια κατάλληλη μετρική απόστασης/ομοιότητας, χρησιμοποιήστε μέσω του εργαλείου Scikit-Learn δύο διαφορετικές τεχνικές συσταδοποίησης (K-means, DBSCAN), συζητήστε τα αποτελέσματα και την επίπτωση των παραμέτρων των μεθόδων σε αυτά, και συγκρίνετέ τα ως προς την ποιότητα/αποτελεσματικότητα της συσταδοποίησης (π.χ. scatter plots, clustering metrics).

Βήμα 3: Ταξινόμηση (Classification)

Τελευταίο βήμα της πειραματικής σας διαδικασίας είναι η χρήση μοντέλων ταξινόμησης με στόχο την ανάθεση ενός αντικειμένου σε προκαθορισμένες κατηγορίες (κλάσεις). Όπως πριν, και σε αυτό το στάδιο, σημαντικό ρόλο παίζει η μοντελοποίηση του προβλήματος (τι ακριβώς ψάχνετε να εντοπίσετε). Διαδικαστικά, αφού μετασχηματίσετε κατάλληλα το dataset στη μορφή (<Feature(s)>, <Label(s)>), δημιουργήστε μέσω του API Scikit-Learn δύο ταξινομητές

(π.χ., Bayesian, LS-SVM, Neural Networks) και - όπως και στο προηγούμενο βήμα - συγκρίνετε τις επιδόσεις τους (π.χ., υπολογίζοντας confusion matrix, ROC-AUC curve). Παρόμοια, κατασκευάστε δυο νευρωνικά δίκτυα μέσω του API TensorFlow/Keras, και δοκιμάστε να εκπαιδεύσετε το ένα ως έχει, και το άλλο μέσω transfer learning και συγκρίνετέ τα με τα προηγούμενα μοντέλα. Τι παρατηρείτε; Βελτιώθηκε η επίδοση των μοντέλων και γιατί;

Βήμα 4: Σύνοψη

Λαμβάνοντας υπόψη τα αποτελέσματα των προηγούμενων βημάτων, καταγράψτε τις παρατηρήσεις σας και 2-3 βασικά συμπεράσματα (“take-home messages”) αναφορικά με το σύνολο δεδομένων με κατάλληλη απεικόνιση/οπτικοποίηση – με άλλα λόγια, “πείτε μια ιστορία” με τα δεδομένα σας (“data story telling”).

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 ΣΤΟΧΟΙ ΕΡΓΑΣΙΑΣ

Βασικό στόχος της εργασίας αποτελεί η εξοικείωση με ένα πραγματικό σύνολο δεδομένων και η εφαρμογή τεχνικών Αναλυτικής Δεδομένων πάνω σε αυτό. **Σημαντική σημείωση είναι κατά κύριο λόγο στην εργασία ακολουθήθηκαν μέθοδοι που διδάχθηκαν στα εργαστήρια του μαθήματος. Η τελική απάντηση για το βήμα 4 γράφτηκε σε αυτήν ενότητα.**

2. ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

2.1 ΠΑΡΟΥΣΙΑΣΗ ΑΡΧΙΚΗΣ ΣΚΕΨΗΣ

2.1.1 Βήμα 1

Το πρώτο βήμα αφορά την προπαρασκευή των δεδομένων και την εξοικείωση με αυτά. Αρχικά , λοιπόν χρησιμοποιήθηκαν εντολές για τη φόρτωση του αρχείου, αλλά και εντολές για την εξοικείωση με την δομή και το περιεχόμενο του αρχείου. Ύστερα, ακολούθησαν εντολές και τεχνικές εύρεσης και αντιμετώπισης ελλιπών τιμών, που φαίνεται λεπτομερώς παρακάτω. Ομοίως έγινε και για τις ακραίες τιμές του συνόλου δεδομένων, ενώ στο τέλος έγινε και η κανονικοποίηση, αλλά και το ‘encoding’ ορισμένων στηλών όπως θα εξηγηθεί παρακάτω.

2.1.2 Βήμα 2

Το δεύτερο βήμα αφορά την συσταδοποίηση των δεδομένων. Αφού, επιλέχθηκαν τα κατάλληλα χαρακτηριστικά όπως θα αναλυθεί παρακάτω, εφαρμόστηκε συσταδοποίηση μέσω KMeans και μέσω DBSCAN, οι οποίες ύστερα συγκρίθηκαν.

2.1.3 Βήμα 3

Το τρίτο ερώτημα του προβλήματος αφορούσε την χρήση μοντέλων ταξινόμησης με στόχο την ανάθεση ενός αντικείμενου (μία νέα εγγραφή) σε προκαθορισμένες κατηγορίες/κλάσεις. Αρχικά χρησιμοποιήθηκαν δύο απλοί ταξινομητές αφού προετοιμάστηκαν κατάλληλα τα δεδομένα και ύστερα χρησιμοποιήθηκαν δύο νευρωνικά δίκτυα για την εκπαίδευση.

2.1.4 Βήμα 4

Σκοπό αυτής της ανάλυσης αποτελεί η πρόβλεψη της απάντησης των χρηστών όσον αφορά της εκστρατείας τους, ‘term deposit’, μέσου άμεσου μάρκετινγκ (τηλεφωνικές κλήσεις) ενός Πορτογαλικού τραπεζικού ιδρύματος, καθώς και η εξοικείωση με τα δεδομένα αυτά. Κάποια από τα βασικά συμπεράσματα είναι πως ορισμένα από τα χαρακτηριστικά που περιέχονται στο σύνολο δεδομένων δεν χρειάζονται να συμπεριληφθούν στην ανάλυση τόσο λόγω της φύσεώς τους, π.χ. poutcome αφού το μεγαλύτερο ποσοστό του είναι άγνωστο και duration για το classification task όπως εξηγήθηκε πιο πάνω, όσο και λόγω της μη συσχέτισής τους με την τελική μεταβλητή στόχο y, όπως το default και το contact. Επιπλέον, κάποια

χαρακτηριστικά έχουν μεγαλύτερη βαρύτητα από όλα στον προσδιορισμό της μεταβλητής στόχου αφού έχουν και μεγαλύτερη συσχέτιση με αυτήν.

Όσον αφορά της συσταδοποίηση, η συσταδοποίηση με K-means αποκάλυψε 14 ομάδες, ενώ η DBSCAN βρήκε 3 ομάδες τελικά, δείχνοντας την αποτελεσματικότητα της επιλογής της τεχνικής συσταδοποίησης, αλλά και η διαφορά που κάνει η επιλογή ορισμένων χαρακτηριστικών έναντι άλλων. Στη ταξινόμηση, το νευρωνικό δίκτυο με transfer learning παρουσίασε την καλύτερη ακρίβεια, με ελάχιστη αύξηση σε σχέση με το αρχικό μοντέλο, σε μικρότερο σύνολο δεδομένων και λιγότερες εποχές. Η κανονικοποίηση και η μείωση διαστάσεων βελτίωσαν τη γενίκευση των μοντέλων, μειώνοντας τον θόρυβο και βελτιώνοντας την ταχύτητα εκπαίδευσης.

2.2 ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για την ανάπτυξη των προγραμμάτων, χρησιμοποιήθηκε η Pyhton3. Αρχικά θα δοθούν κάποιες βασικές πληροφορίες και στη συνέχεια θα αναλυθούν τα κυρίως προγράμματα. **Τα screenshot με τις λύσεις και τα αποτελέσματα, περιέχονται στο τρίτο μέρος του εγγράφου, 'Επίδειξη της λύσης'.**

2.2.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ

Οι βασικές βιβλιοθήκες που χρησιμοποιήθηκαν ήταν διάφορες και φαίνονται αναλυτικά στο notebook.

2.3.1 ΑΝΑΛΥΣΗ ΒΑΣΙΚΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ

2.3.1.1 preprocessing.ipynb

Το αρχείο αυτό περιέχει τον κώδικα για την υλοποίηση των ζητούμενων του 1ου βήματος.

Μετά την φόρτωση των δεδομένων, ακολούθησαν κάποιες εντολές για την προβολή τους, όπως *data, data.shape, data.columns* οι οποίες αποκαλύπτουν πληροφορίες για το μέγεθος και την δομή των δεδομένων, αλλά και το *data.describe* το οποίο αποκαλύπτει στατιστικά στοιχεία για τα εν λόγω δεδομένα. Συνεχίζοντας, εκτελέστηκε η εντολή για την διαγραφή των διπλοτύπων, η οποία δεν είχε κάποια επιρροή αφού δεν υπήρξαν διπλότυπα.

Στη συνέχεια, χρειάστηκε να αντιμετωπιστεί το ζήτημα των ελλιπών τιμών, που εκ πρώτης όψεως φαίνεται να μην υπάρχουν. Εκτελώντας την εντολή *data.isna().sum()* παρατηρείται πως καμία στήλη δεν έχει ελλιπές τιμές. Παρόλα αυτά, μετά από μία πιο προσεκτική ματιά είναι φανερό πως ορισμένα κατηγορικά δεδομένα περιέχουν μία στήλη 'unknown' η οποία υπονοεί πως σε όποια εγγραφή της αποδόθηκε αυτή η τιμή, αυτό έγινε διότι δεν γνώριζαν (για οποιοδήποτε λόγο) ποια είναι η τιμή που αντιστοιχεί. Βλέποντας, λοιπόν, τον αριθμό των εγγραφών που περιέχουν το unknown αυτός είναι αρκετά μεγάλος στις στήλες education, contact και poutcome (στην οποία οι περισσότερες εγγραφές είναι unknown), ενώ στην στήλη job, ο αριθμός είναι αρκετά μικρός (288) επομένως αποφασίστηκε να αφαιρεθούν όλες οι εγγραφές που περιέχουν unknown στη στήλη αυτή. Όσον αφορά τη στήλη education και επειδή στην περίπτωση αυτή οι εγγραφές είναι περισσότερες θα χρησιμοποιηθεί η μέθοδος 'Simple Impute' από την Scikit-Learn, ώστε να συμπληρωθούν οι στήλες που περιέχουν unknown, αντικαθιστώντας με την πιο συχνή τιμή του πεδίου (secondary). Συνεχίζοντας στο contact, εδώ παρατηρείται πως οι τιμές unknown αποτελούν ένα μεγάλο ποσοστό (12909) και για αυτό θα χρησιμοποιηθεί και πάλι η μέθοδος 'Simple Impute'. Τέλος για την στήλη poutcome, επειδή σχεδόν όλες οι εγγραφές της ήταν unknown, αποφασίστηκε να αφαιρεθεί τελείως η στήλη.

Μετά, ακολούθησε η αφαίρεση των ακραίων τιμών (outliers), αφού όπως παρατηρείται και από τα διαγράμματα κατανομών, οι κατανομές δεν είναι κανονικές. Το pdays θα αγνοηθεί επειδή περιέχει την τιμή -1, η οποία σημαίνει πως δεν έχει υπάρξη επικοινωνία με τον πελάτη στο παρελθόν. Ξεκινώντας, παρατηρώντας τα στατιστικά του age, μπορεί να εφαρμοστεί η μέθοδος IQR (0.25-0.75) και ύστερα να αφαιρεθούν οι outliers. Επιλέχθηκε το 75, 25 καθώς παρατηρείται από τα διαγράμματα πως η συγκέντρωση των τιμών βρίσκεται γύρω από τις ηλικίες 25-75. Συνεχίζοντας, στην περίπτωση του balance φαίνεται ότι υπάρχουν κάποιες πολύ ακραίες τιμές, τόσο αρνητικές όσο και θετικές, αφού

mean=1342.482236 με std=2982.923889 και max=102127.000000, min=-8019.000000. Άρα, πρώτα θα εφαρμοστεί IQR (97-3) ώστε να αφαιρεθούν οι πολύ ακραίες τιμές (191) και μετά θα εφαρμοστεί IQR (15-85) ώστε να αφαιρεθούν και οι υπόλοιπες ακραίες τιμές (1819). Ακολουθεί το day πάνω στο οποίο δε θα εφαρμοστεί κάτι, διότι το χαρακτηριστικό μέρα δεν είναι δυνατό να έχει ακραίες τιμές αφού απλώς αντιπροσωπεύουν μία ημερομηνία. Φτάνοντας στο duration, αρχικά είναι χρήσιμο να αφαιρεθούν εκείνες οι εγγραφές που έχουν το χαρακτηριστικό αυτό μικρότερο του 2, αφού αυτό θα σημαίνει πως η κλήση απέτυχε ή υπήρξε κάποιο πρόβλημα με αυτήν για να είναι τόσο μικρή. Ύστερα θα εφαρμοστεί IQR (25-75), βάση του ιστογράμματος, ώστε να αφαιρεθούν οι ακραίες τιμές (3044). Όσον αφορά το campaign, αυτό φαίνεται από το ιστόγραμμα πως περιέχει πολύ ακραίες τιμές που τείνουν να σχηματίζουν λογαριθμική συνάρτηση. Αποφασίστηκε να εφαρμοστεί IQR (85-15) και ύστερα να αφαιρεθούν οι ακραίες τιμές που προκύπτουν (1076). Τέλος, το previous φαίνεται να έχει μία υπερβολικά ακραία τιμή η οποία αφαιρέθηκε, ενώ εφαρμόστηκε iqr (85-15),ομοίως με το campaign.

Ως επόμενο βήμα, αποτέλεσε η οπτικοποίηση των χαρακτηριστικών του συνόλου δεδομένων, σε μορφή ιστογραμμάτων για τα αριθμητικά χαρακτηριστικά, και σε μορφή ‘pie chart’ τα μη αριθμητικά. Επιπλέον εμφανίζεται το correlation και το covariance των χαρακτηριστικών. Σε αυτό το βήμα χρησιμοποιείται η μέθοδος Cramer για να δημιουργηθεί μία συσχέτιση μεταξύ των κατηγορικών μεταβλητών και της μεταβλητής στόχου, και για αυτό επιλέγεται να αφαιρεθούν τελείως τα χαρακτηριστικά contact και default αφού δεν έχουν παρά μόνο 0.01 και 0.02 αντίστοιχα συσχέτιση, σύμφωνα με την μέθοδο.

Επιπροσθέτως, ακολούθησε ο μετασχηματισμός των δεδομένων, δηλαδή η κωδικοποίηση (encoding) και η κανονικοποίηση (normalization). Στο encoding, στο χαρακτηριστικό pdays αποφασίστηκε να αναπαρασταθεί ως 1 αν έχει γίνει επικοινωνία στο παρελθόν και ως 0 αν δεν έχει γίνει επικοινωνία, ενώ για τα χαρακτηριστικά των housing, loan, y χρησιμοποιήθηκε το *fit_transfrom()* του *LabelEncoder()*. Ύστερα, έγιναν κάποιες ενέργειες για την καλύτερη κατανόηση της επιρροής ορισμένων μεταβλητών πάνω στην τελική μεταβλητή y, δείχνοντας στατιστικά του τύπου κατά πόσο τις εκατό η συγκεκριμένη τιμή της μεταβλητής previous θα καταλήξει σε τιμή 0 ή 1 αντίστοιχα στην μεταβλητή στόχο. Συνεχίζοντας, έγινε κανονικοποίηση των αριθμητικών μεταβλητών μέσω του *MinMaxScaler()* και encoding των υπόλοιπων κατηγορικών χαρακτηριστικών μέσω του *OneHotEncoder()*. Στο τέλος της προπαρασκευής, οι εγγραφές που παρέμειναν στο σύνολο δεδομένων ήταν 34919.

2.3.1.2 clustering.ipynb

Το αρχείο αυτό περιέχει τον κώδικα για την υλοποίηση των ζητούμενων του 2ου βήματος.

Ερχόμενοι στο δεύτερο σκέλος, αφού φορτωθούν τα δεδομένα, θα πρέπει να γίνει επιλογή των χαρακτηριστικών που θα εξεταστούν. Εδώ χρησιμοποιήθηκε η επιλογή των χαρακτηριστικών βάση της μεθόδου *SelectKBest* αφού πρώτα αφαιρέθηκε η στήλη previous επειδή σύμφωνα με το correlation heatmap αυτό είχε πολύ μεγάλο correlation (>0.9). Χρησιμοποιήθηκαν οι συναρτήσεις *mutual_info_classif* και *f_classif* ως παράμετροι και επιλέχθηκαν τα 15 χαρακτηριστικά με μεγαλύτερο score και στα δύο.

Στη συνέχεια για να βρεθεί το βέλτιστο k για τον KMeans, εφαρμόστηκε Silhouette ανάλυση, με βάση την οποία το βέλτιστο k αναδείχθηκε το 14. Ύστερα, εφαρμόστηκε ο KMeans, ενώ φανερώθηκε και η κατανομή των μεγεθών cluster που ήταν αρκετά ομοιόμορφη, με silhouette score ίσο με περίπου 0.28.

Συνεχίζοντας στον DBSCAN, στην αρχή εφαρμόστηκε ώστε να βρεθούν ο αρχικός αριθμός των συστάδων ο οποίος ήταν 147. Έπειτα χρησιμοποιήθηκε η μέθοδος *NearestNeighbors* με k=15, ώστε να βρεθεί το βέλτιστο EPS που βρέθηκε ίσο με 1. Στη συνέχεια, επιλέχθηκε min_samples=6 μετά από πειραματισμό, αφού έφερνε το υψηλότερο silhouette score και εφαρμόστηκε εκ νέου ο DBSCAN, καταλήγοντας σε 136 συστάδες και silhouette score ίσο περίπου με 0.3. Επιπλέον, παρατηρώντας τα δύο scatter plot, είναι φανερό πως υπήρξε καλύτερη επίδοση στη περίπτωση του KMeans.

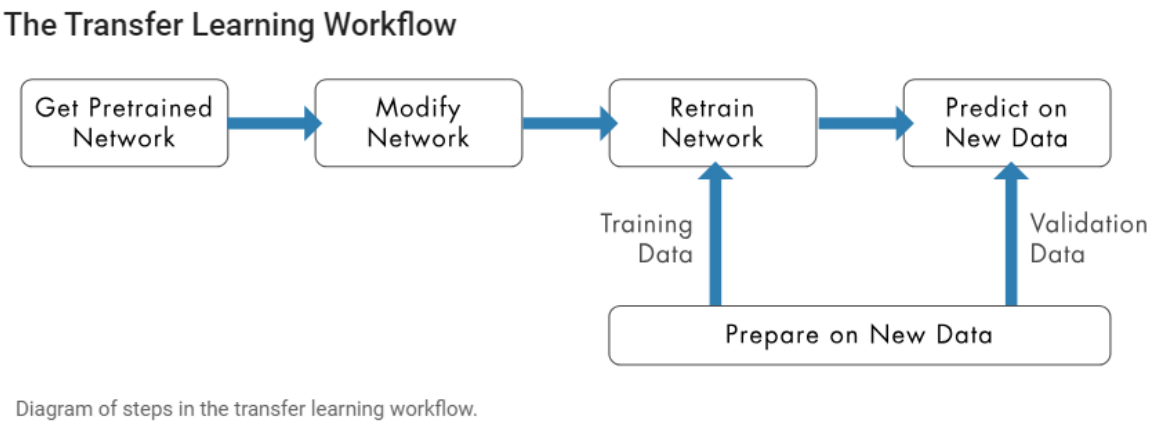
Παρόλα αυτά, ο DBSCAN κατέληξε σε 136 clusters που είναι πάρα πολλά, πράγμα που οδήγησε στην αναθεώρηση της λύσης. Έτσι, μετά από πειραματισμό, συμπεράθηκε ότι πιθανότερο να ευθύνεται η επιλογή των features που έγινε αρχικά. Οπότε στη συνέχεια του αρχείου εφαρμόσαμε ακριβώς την ίδια διαδικασία με επιλογή διαφορετικών χαρακτηριστικών. Συγκεκριμένα, χρησιμοποιήθηκε η μέθοδος *VarianceThreshold* στη οποία αφαιρούνται τα χαρακτηριστικά με χαμηλή διασπορά. Ορίστηκε threshold=0.1, οπότε συμπεριλήφθηκαν 15 με διασπορά>0.1. Εκτελώντας, ομοίως, το silhouette analysis, βρέθηκε και πάλι k=14 και μετά την εφαρμογή του KMeans τα σκορ είχαν ως εξής: Silhouette Score περίπου 0.26, Calinski-Harabasz Score περίπου 3404.96 και Davies-Bouldin Score περίπου 1.57, ενώ η κατανομή των συστάδων ήταν μερικώς ομοιόμορφη. Ύστερα ο DBSCAN αρχικά κατέληξε σε 336 συστάδες ενώ μετά ορίστηκε eps=0.1 και min_samples=6 και κατέληξε σε 3 συστάδες (πολύ λιγότερες σε σχέση με την προηγούμενη επιλογή χαρακτηριστικών). Συγκρίνοντας, μάλιστα, στα scatter plots, παρατηρείται ότι ο KMeans τυα πήγε καλύτερα στο Silhouette Score αλλά χειρότερα στα Calinski-Harabasz και Davies-Bouldin. Επομένως, η υπόθεση που έγινε ότι θα είχε καλύτερη απόδοση ο DBSCAN, όσον αφορά τον αριθμό τον clusters (λιγότερα), αποδείχθηκε ορθή. Τα σκορ είχαν ως εξής: Silhouette Score περίπου 0.24, Calinski-Harabasz Score περίπου 6519.18 και Davies-Bouldin Score περίπου 1.63, ενώ η κατανομή των συστάδων ήταν μερικώς ομοιόμορφη

2.3.1.3 classification.ipynb

Το αρχείο αυτό περιέχει τον κώδικα για την υλοποίηση των ζητούμενων του 3ου βήματος. Αρχικά, αφαιρέθηκε το χαρακτηριστικό duration, αφού ο σκοπός του ερωτήματος είναι να προβλεφθεί βάση των δεδομένων αν το χαρακτηριστικό y θα είναι yes ή no, επομένως το duration δεν θα προσφέρει κάτι στην πρόβλεψη αφού είναι ένα χαρακτηριστικό που καταγράφεται αφού έχει ήδη τελειώσει η συμφωνία και άρα δεν παίζει κάποιο ρόλο. Συνεχίζοντας παρατηρήθηκε πως η κατανομή της μεταβλητής-στόχος (y) δεν είναι καθόλου ομοιόμορφη (92.7%-no και 7.3%-yes), άρα χρησιμοποιήθηκε η μέθοδος *SMOTE()* για να κατανεμηθούν ομοιόμορφα οι τιμές της y, μέθοδος η οποία παράγει συνθετικά δεδομένα ώστε να ισοφαρίσει της πλεονάζουσα κλάση (σε αυτήν την περίπτωση πλεονάζουσα είναι κλάση no), επιλέγοντας του k κοντινότερους γείτονες για κάθε στιγμιότυπο της κλάσης σε μειονότητα, καταλήγοντας σε μία αναλογία 50-50. Συνεχίζοντας, εφαρμόζουμε το *train_test_split* και επιλέγονται το Naïve Bayes Classifier και Random Forest Classifier ως ταξινομητές, με το πρώτο να έχει ένα test accuracy περίπου 0.7 και ROC AUC score περίπου 0.78, ενώ το δεύτερο ένα test accuracy περίπου 0.95 και ROC AUC score περίπου 0.99. Το

ROC AUC score είναι μία μετρική για να αξιολογηθεί η απόδοση ενός binary classification μοντέλου με το ROC (Receiver Operating Characteristic) να δείχνει πόσο συχνά το μοντέλο αναγνωρίζει σωστά τα θετικά σε σχέση με το πόσο συχνά αναγνωρίζει λανθασμένα τα αρνητικά ως θετικά και η AUC (Area Under the Curve) είναι ένας αριθμός που δείχνει πόσο καλή είναι συνολικά η καμπύλη ROC. Προφανώς, είχε πολύ καλύτερη επίδοση το Random Forest Classifier και αυτό επειδή ο Naïve Bayes κάνει εξ ορισμού την παραδοχή ότι όλες οι μεταβλητές είναι ανεξάρτητες μεταξύ τους πράγμα που οδηγεί σε χειρότερη απόδοση αφού υπάρχουν σχέσεις εξάρτησης μεταξύ ορισμένων μεταβλητών. Από την άλλη το Random Forest μπορεί να διαχειριστεί περίπλοκες σχέσεις και συσχετίσεις μεταξύ των μεταβλητών λόγω της δομής των δένδρων απόφασης. Επιπλέον, μπορεί να ευθύνεται και η φύση του συνόλου δεδομένων, η οποία δεν διευκολύνει το Naïve Bayes, αλλά το Random Forest.

Συνεχίζοντας, θα γίνει χρήση Νευρωνικών Δικτύων για την εκπαίδευση. Χρησιμοποιήθηκε οι βιβλιοθήκη TensorFlow/Keras και δημιουργήθηκε ένα *Sequential* μοντέλο, με 1 *Dense* layer εισόδου, 4 εσωτερικά (hidden) *Dense* layers, και ένα τελευταίο *Dense* classifier layer. Σε όλα τα layers εφαρμόστηκε relu activation, πέρα από το τελευταίο στο οποίο χρησιμοποιήθηκε sigmoid. Επιπλέον, χρησιμοποιήθηκε *Dropout(x),* που σημαίνει ότι το x% των νευρώνων στο layer δεν θα χρησιμοποιηθεί στο forward και backward pass, αφού μειώνει την υπερπροσαρμογή και βοηθάει στο generalization του μοντέλου σε ξένα δεδομένα. Ως optimizer χρησιμοποιήθηκε ο adam, ως loss το binary_crossentropy και ως μετρική το accuracy. Το μοντέλο εκπαιδεύτηκε σε 30 εποχές με batch_size=32, πετυχαίνοντας ένα test_accuracy ίσο με περίπου 0.89. Για την εκπαίδευση χρησιμοποιήθηκε το X_train1, y_train1 (56% του συνόλου δεδομένων), για το test set το X_test, y_test (10% του συνόλου δεδομένων), ενώ για ως validation set χρησιμοποιήθηκε το X_val, y_val (10% του συνόλου δεδομένων). Στη συνέχεια, χρειάστηκε να γίνει save το μοντέλο μέσω της *model.save()* και ύστερα αυτό φορτώθηκε μέσω της *load_model()*. Αυτό έγινε για να υλοποιηθεί το transfer learning στο οποίο το trained μοντέλο που ήδη υπάρχει αποθηκεύεται και ξαναχρησιμοποιείται για να γίνει train ξανά σε διαφορετικά δεδομένα με ίδιο test set και validation set, όμως.



Εικόνα. Transfer Learning

Τα βήματα για την επίτευξη του transfer learning είναι τα εξής:

1. Επιλογή pretrained μοντέλου
2. Αντικατάσταση των final layers έτσι ώστε να προσαρμοστεί στο νέο dataset για να παράγει τόσες προβλέψεις όσες και νέος αριθμός classes που έχει το νέο task

3. Προαιρετικά, freeze weights, θέτοντας το learning rate των layers ίσο με 0, ώστε οι παράμετροι αυτών των layers να μην γίνονται update, αυξάνοντας τόσο την ταχύτητα, αλλά και αποτροπή overfitting σε περίπτωση που το dataset είναι μικρό
4. Επανεκπαίδευση μοντέλου
5. Πρόβλεψη

Ακολουθώντας αυτήν την διαδικασία, επιτεύχθηκε ένα accuracy της τάξεως του περίπου 89%. Αυτό έγινε χρησιμοποιώντας 1 εποχή , batch_size=32, ενώ έγινε compile με ακριβώς της ίδιες παραμέτρους όπως πριν. Σημαντικό είναι να σημειωθεί ότι αυτή τη φορά, για την εκπαίδευση, έγινε χρήση του X_train1, y_train1 (24% του συνόλου δεδομένων), για το test set το X_test, y_test (10% του συνόλου δεδομένων), ενώ για ως validation set χρησιμοποιήθηκε το X_val, y_val (10% του συνόλου δεδομένων), δηλαδή τα δύο τελευταία ήταν ίδια με πριν. Επομένως παρατηρείται πως χρησιμοποιώντας λιγότερο από το μισό, αναλογικά, σύνολο δεδομένων, και μόλις 1 εποχή, που αποτελεί περίπου το 3.3% των εποχών που χρησιμοποιήθηκαν προηγουμένως πετυχαίνουμε ίσο και μεγαλύτερο accuracy (0.8923 vs 0.8948) στο περίπου 3.28% του χρόνου που έκανε το πρώτο train (61s vs 2s). Έτσι, αναδεικνύεται η χρησιμότητα του transfer learning.

3. ΕΠΙΔΕΙΞΗ ΤΗΣ ΛΥΣΗΣ

Για να αναδείξουμε την λειτουργία των προγραμμάτων θα τρέξουν με τη σειρά τα προγράμματα προαναφερόμενα προγράμματα και, ενδεικτικά, θα δειχθούν ορισμένα screenshot.

3.1.1 preprocessing.ipynb

```
data
```
✓ 0.0s

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

45211 rows × 17 columns

```
# size & structure of data
*data.shape,\
data.columns
```

```
(45211,
 17,
 Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
        'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
        'previous', 'poutcome', 'y'],
       dtype='object'))
```

Statistics

```
data.describe().round(2) # for numerical data
```
[52] ✓ 0.0s

	age	balance	day	duration	campaign	pdays	previous
count	45211.00	45211.00	45211.00	45211.00	45211.00	45211.00	45211.00
mean	40.94	1362.27	15.81	258.16	2.76	40.20	0.58
std	10.62	3044.77	8.32	257.53	3.10	100.13	2.30
min	18.00	-8019.00	1.00	0.00	1.00	-1.00	0.00
25%	33.00	72.00	8.00	103.00	1.00	-1.00	0.00
50%	39.00	448.00	16.00	180.00	2.00	-1.00	0.00
75%	48.00	1428.00	21.00	319.00	3.00	-1.00	0.00
max	95.00	102127.00	31.00	4918.00	63.00	871.00	275.00

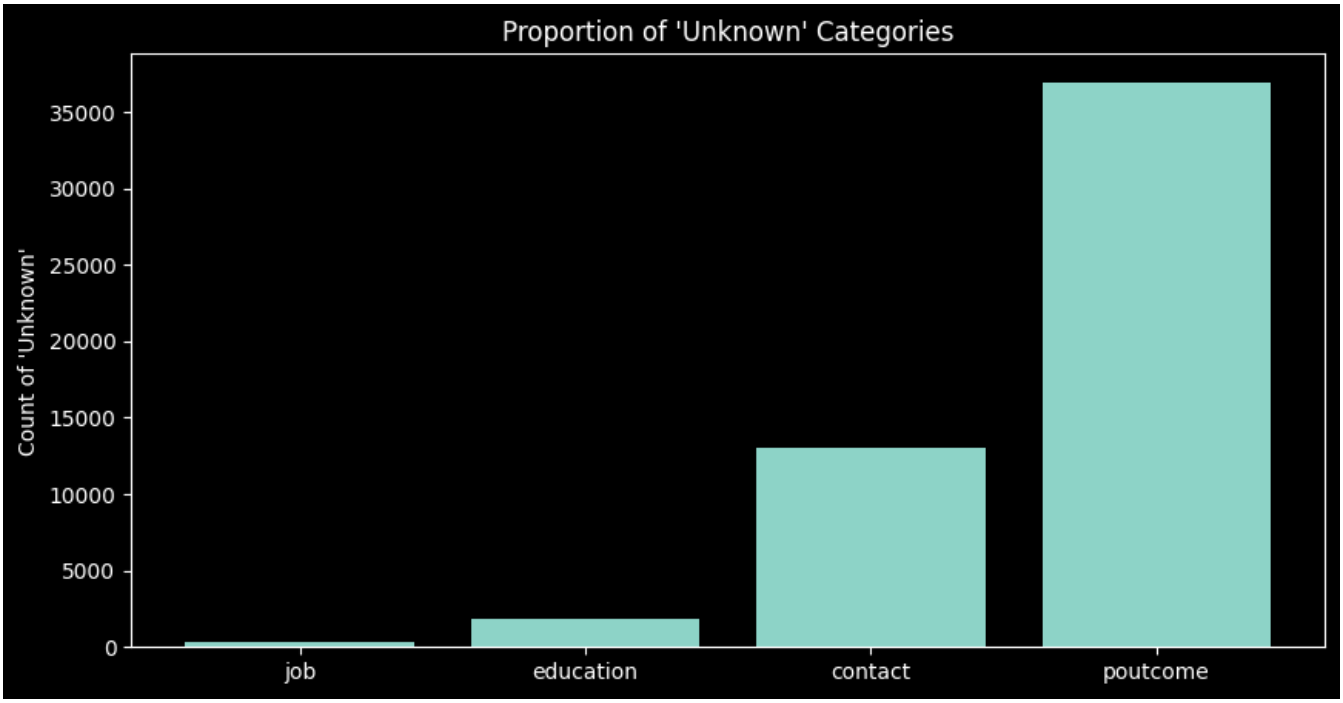
Εικόνα 1. Δομή, περιεχόμενο και στατιστικά δεδομένων

```
# missing values per column
data.isna().sum()
```

✓ 0.0s

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

Εικόνα 2. Ελλιπές τιμές ανά στήλη



Εικόνα 3. Αριθμός unknown κατηγορίας ανά στήλη

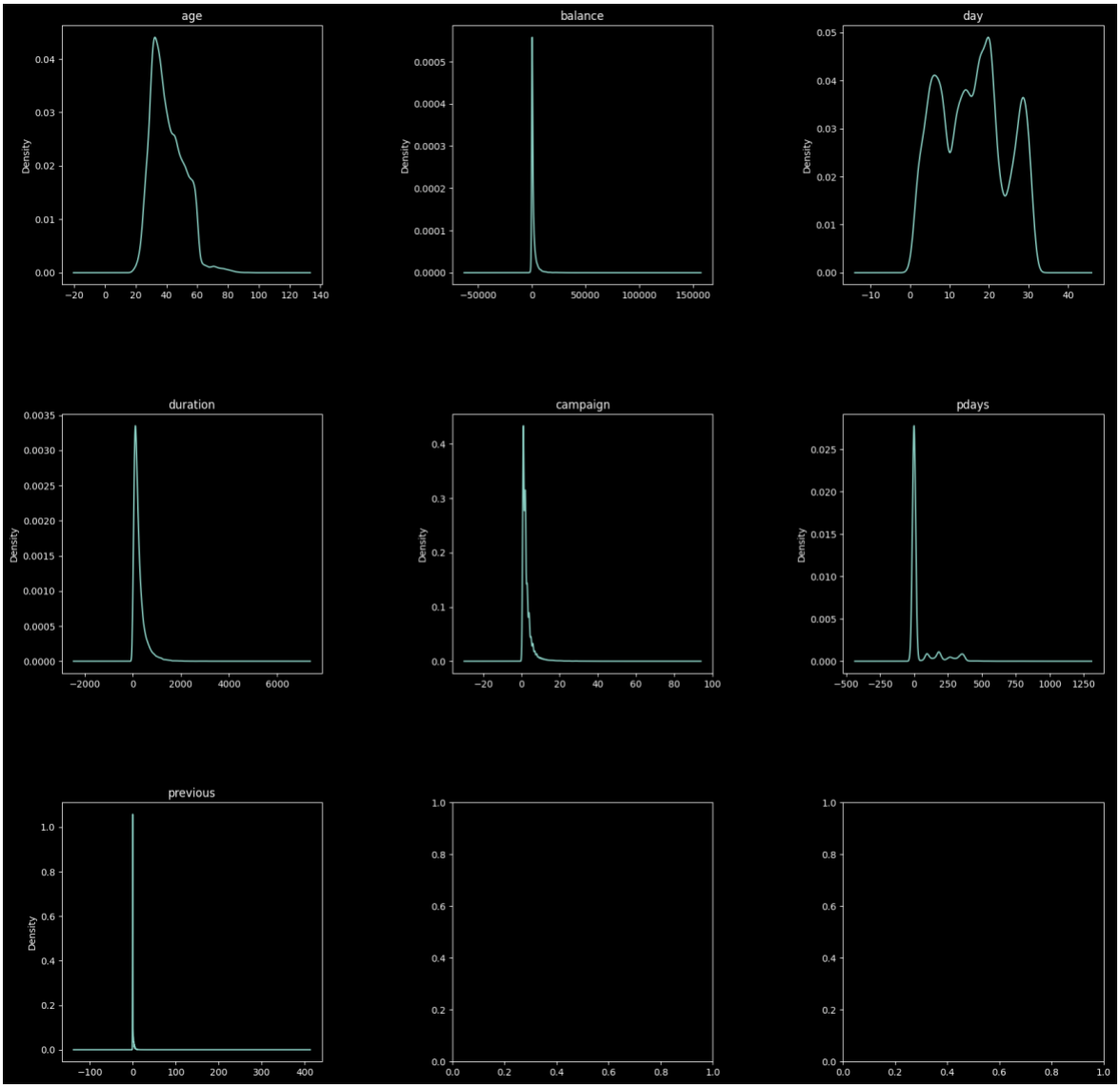
```
data.job.value_counts()
✓ 0.0s

job
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: count, dtype: int64
```

Εικόνα 4. Αριθμός τιμών των διαφορετικών χαρακτηριστικών της στήλης job

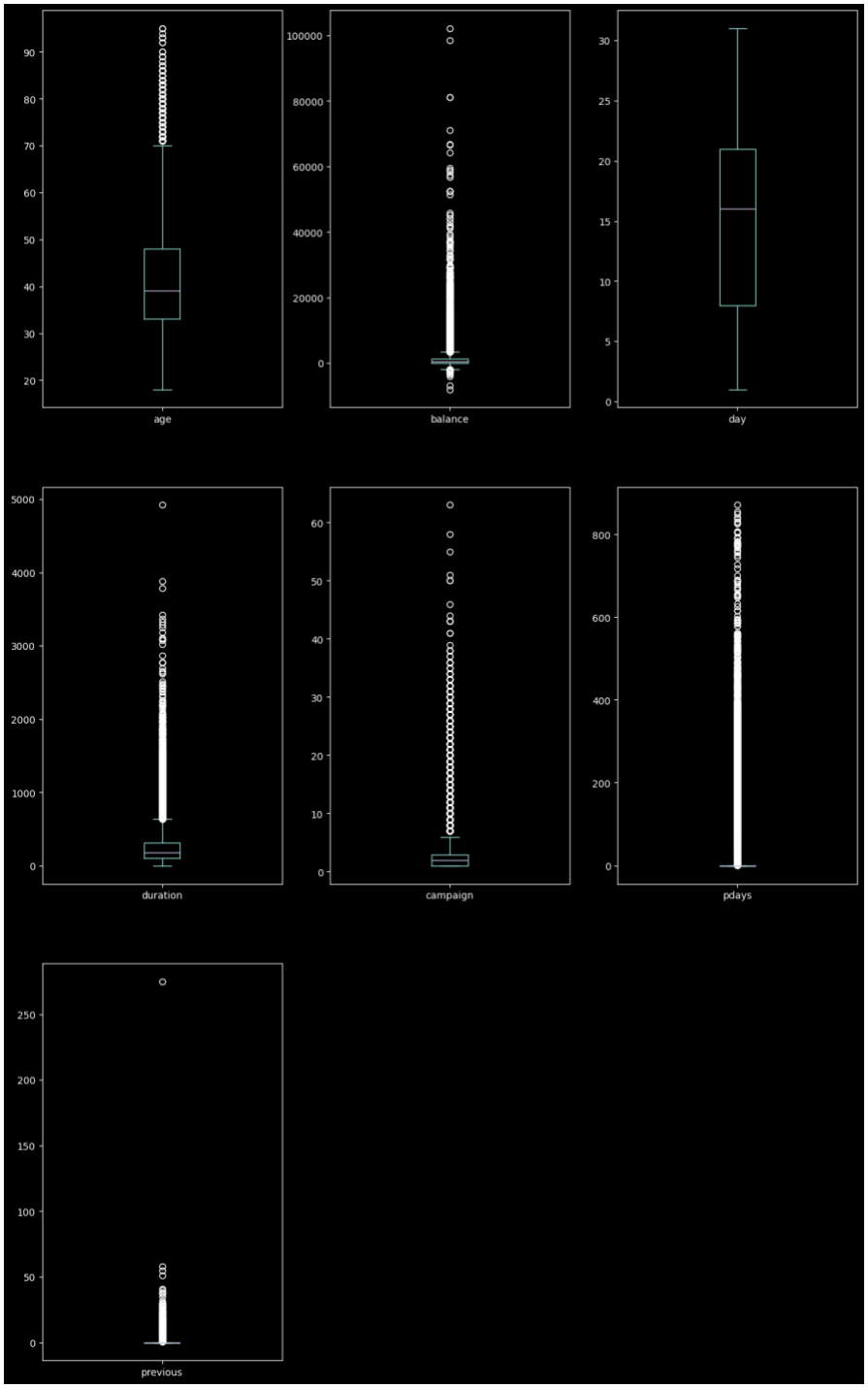
```
# remove job 'unknown' since only 288 unknown

data = data[data['job'] != 'unknown']
data.job.value_counts(sort=True)
✓ 0.0s

from sklearn.impute import SimpleImputer

data['education'] = data['education'].replace('unknown', np.nan)
imputer = SimpleImputer(strategy='most_frequent')
data['education'] = imputer.fit_transform(data[['education']]).ravel()
✓ 0.1s
```

Εικόνα 5. Ενδεικτικοί τρόποι που χρησιμοποιήθηκαν για τη στήλη job και education

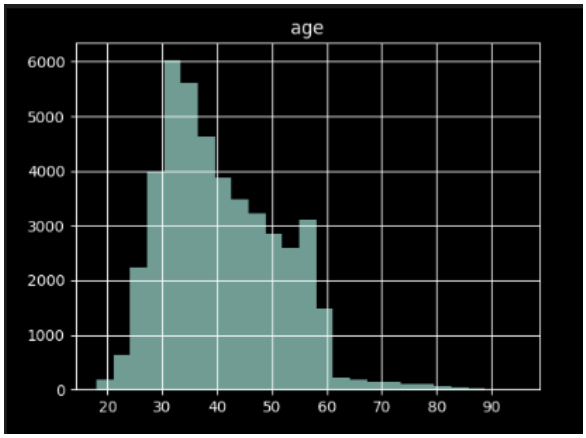
Εικόνα 6. Ολική απαλοιφή unknown



Εικόνα 7. Κατανομές των αριθμητικών δεδομένων

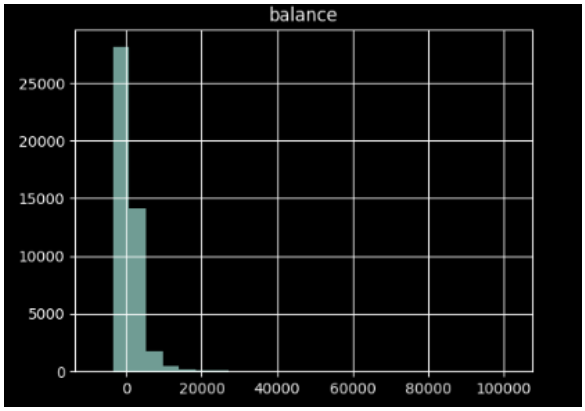


Εικόνα 8. Ακραίες τιμές



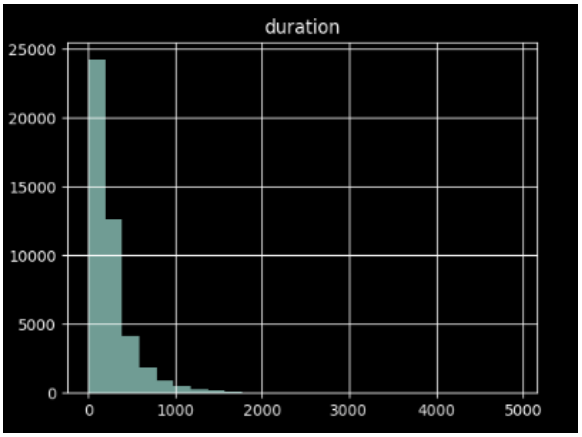

```
data.age.describe()
✓ 0.0s
count    44923.000000
mean        40.893529
std         10.604399
min         18.000000
25%         33.000000
50%         39.000000
75%         48.000000
max         95.000000
Name: age, dtype: float64
```

Εικόνα 9. Κατανομή και στατιστικά age



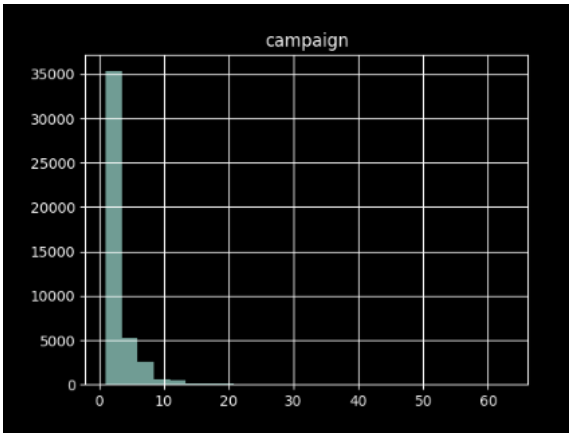

```
data.balance.describe()
✓ 0.0s
count     44443.000000
mean       1342.482236
std        2982.923889
min       -8019.000000
25%          70.000000
50%         441.000000
75%        1403.000000
max      102127.000000
Name: balance, dtype: float64
```

Εικόνα 10. Κατανομή και στατιστικά balance



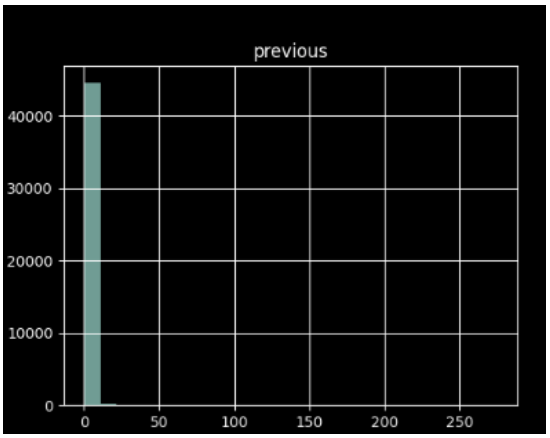

```
data.duration.describe()
✓ 0.0s
count    42433.000000
mean       256.782881
std        256.206147
min          0.000000
25%        103.000000
50%        179.000000
75%        317.000000
max       3881.000000
Name: duration, dtype: float64
```

Εικόνα 11. Κατανομή και στατιστικά duration




```
data.campaign.describe()
✓ 0.0s

count    39382.000000
mean         2.781042
std          3.132114
min          1.000000
25%          1.000000
50%          2.000000
75%          3.000000
max         58.000000
Name: campaign, dtype: float64
```

Εικόνα 12. Κατανομή και στατιστικά campaign

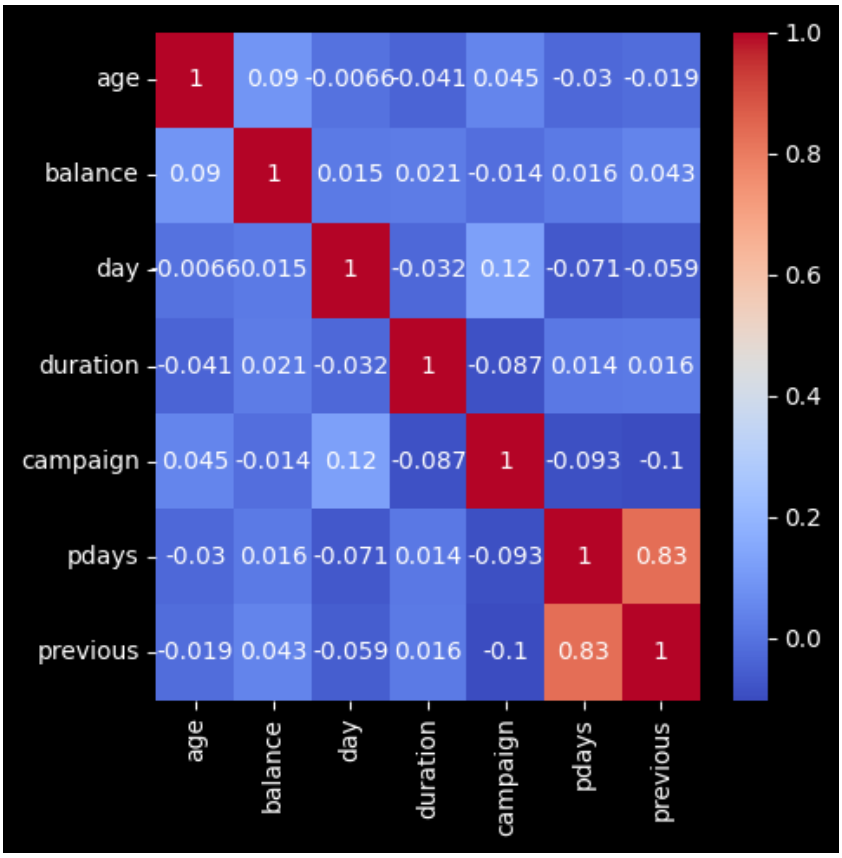



```
data.previous.describe()
✓ 0.0s

count    37793.000000
mean         0.595190
std          2.376533
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max        275.000000
Name: previous, dtype: float64
```

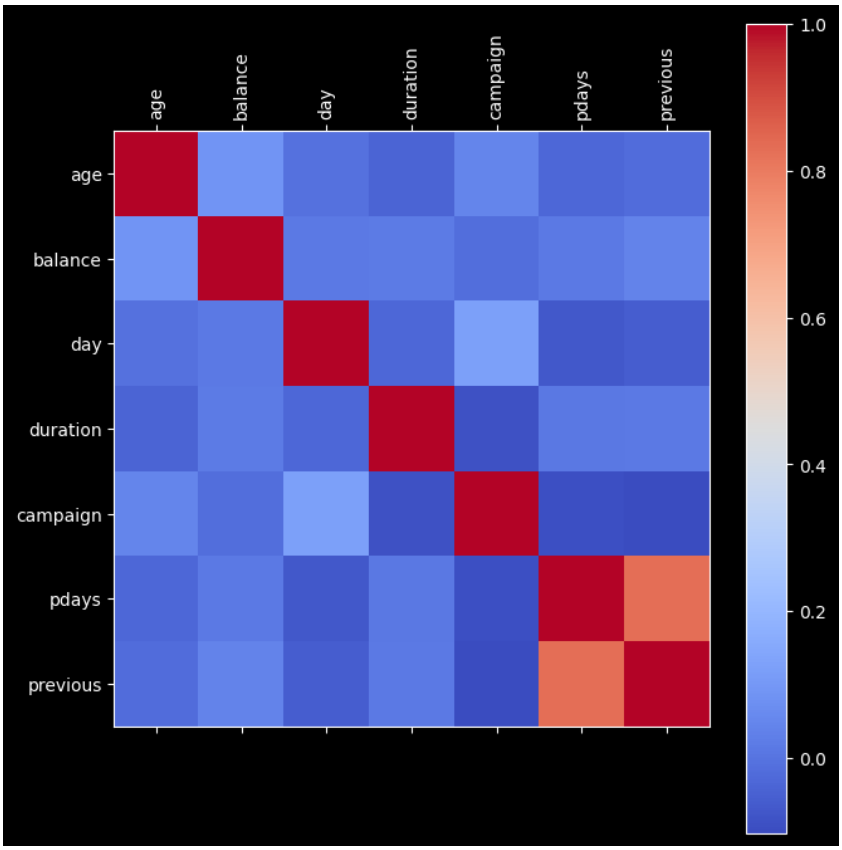
Εικόνα 13. Κατανομή και στατιστικά previous

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.089989	-0.006636	-0.040774	0.044884	-0.029772	-0.018609
balance	0.089989	1.000000	0.014649	0.020562	-0.014032	0.015707	0.043189
day	-0.006636	0.014649	1.000000	-0.031978	0.123322	-0.070787	-0.059054
duration	-0.040774	0.020562	-0.031978	1.000000	-0.087099	0.013673	0.016189
campaign	0.044884	-0.014032	0.123322	-0.087099	1.000000	-0.092575	-0.102391
pdays	-0.029772	0.015707	-0.070787	0.013673	-0.092575	1.000000	0.828538
previous	-0.018609	0.043189	-0.059054	0.016189	-0.102391	0.828538	1.000000

Εικόνα 14. Correlation



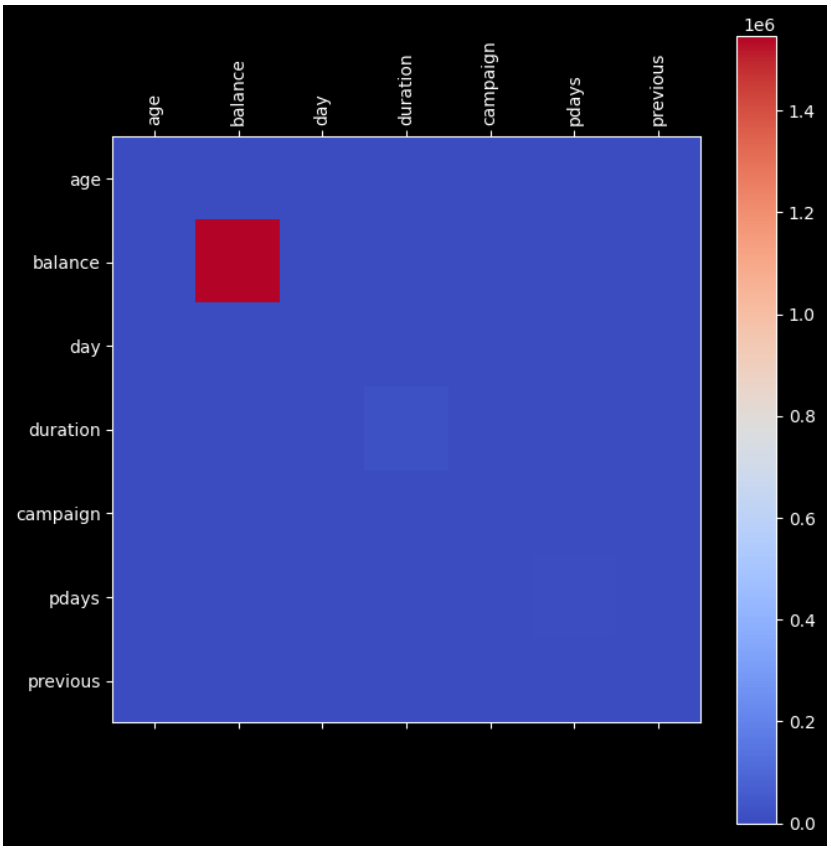
Εικόνα 15. Correlation Heatmap 1



Εικόνα 16. Correlation Heatmap 2

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.089989	-0.006636	-0.040774	0.044884	-0.029772	-0.018609
balance	0.089989	1.000000	0.014649	0.020562	-0.014032	0.015707	0.043189
day	-0.006636	0.014649	1.000000	-0.031978	0.123322	-0.070787	-0.059054
duration	-0.040774	0.020562	-0.031978	1.000000	-0.087099	0.013673	0.016189
campaign	0.044884	-0.014032	0.123322	-0.087099	1.000000	-0.092575	-0.102391
pdays	-0.029772	0.015707	-0.070787	0.013673	-0.092575	1.000000	0.828538
previous	-0.018609	0.043189	-0.059054	0.016189	-0.102391	0.828538	1.000000

Εικόνα 17. Covariance



Εικόνα 18. Covariance Heatmap

```python
# https://stackoverflow.com/questions/46498455/categorical-features-correlation/46498792#46498792
import scipy.stats as ss


def cramers_v(confusion_matrix):
    """ calculate Cramers V statistic for categorial-categorial association.
        uses correction from Bergsma and Wicher,
        Journal of the Korean Statistical Society 42 (2013): 323-328
    """
    chi2 = ss.chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    phi2corr = max(0, phi2 - ((k - 1) * (r - 1)) / (n - 1))
    rcorr = r - ((r - 1) ** 2) / (n - 1)
    kcorr = k - ((k - 1) ** 2) / (n - 1)
    return np.sqrt(phi2corr / min((kcorr - 1), (rcorr - 1)))

# https://medium.com/@ktoprakucar/how-to-calculate-the-correlation-between-categorical-and-continuous-values-dcb7abf79406
```

✓ 0.0s

```python
for categorical_variable in categorical_columns[:-2]:
    confusion_matrix = pd.crosstab(data[categorical_variable], data['y'])
    print(f' Feature {categorical_variable}: Association with Target {round(cramers_v(confusion_matrix.values), 2)}')
```

✓ 0.0s

```
Feature job: Association with Target 0.13
Feature marital: Association with Target 0.08
Feature education: Association with Target 0.09
Feature default: Association with Target 0.02
Feature housing: Association with Target 0.15
Feature loan: Association with Target 0.07
Feature contact: Association with Target 0.01
```

Εικόνα 19. Covariance μέσω Crammer's V Statistic

```python
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, MinMaxScaler

label_encoder = LabelEncoder()

data['pdays'] = data['pdays'].apply(lambda x: 0 if x == -1 else 1)
data.loc[:, 'housing'] = label_encoder.fit_transform(data['housing'])
data.loc[:, 'loan'] = label_encoder.fit_transform(data['loan'])
data.loc[:, 'y'] = label_encoder.fit_transform(data['y'])
```

✓ 0.0s

Εικόνα 20. Encoding των pdays, housing, loan, y

```
counts = data.groupby('pdays')['y'].value_counts().unstack(fill_value=0)

counts['Percentage(%) for y=1'] = (counts[1] / (counts[0] + counts[1])) * 100
counts['Percentage(%) for y=0'] = (counts[0] / (counts[0] + counts[1])) * 100

print(counts)
```

Εικόνα 21. Εύρεση στατιστικών στοιχείων για το pdays σε σχέση με το y

```
# normalize
columns_to_normalize = ['age', 'balance', 'day', 'duration', 'previous', 'campaign']

data_normalized = data.copy()
scaler = MinMaxScaler()
data_normalized[columns_to_normalize] = scaler.fit_transform(data_normalized[columns_to_normalize])

data = data_normalized
✓ 0.0s
```

Εικόνα 22. Κανονικοποίηση των αριθμητικών μεταβλητών μέσω MinMaxScaler()

```
# one-hot encoding for categorical columns
categorical_columns = ['job', 'marital', 'education', 'month']

ohe = OneHotEncoder(sparse_output=False)

encoded_array = ohe.fit_transform(data[categorical_columns])
encoded_data = pd.DataFrame(encoded_array, columns=ohe.get_feature_names_out(categorical_columns), index=data.index)

data.drop(columns=categorical_columns, inplace=True)

data = pd.concat([data, encoded_data], axis=1)
```

Εικόνα 23. Encoding των κατηγορικών μεταβλητών μέσω OneHotEncoder()

```
data.to_csv('../data/data_cleaned.csv', index=False)
✓ 0.3s
```

Εικόνα 24. Αποθήκευση του 'καθαρού' συνόλου δεδομένων

3.1.2 clustering.ipynb

```
corr_matrix = data.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool_))
to_drop = [column for column in upper.columns if any(upper[column] > 0.9)]
data.drop(columns=to_drop, inplace=True)
to_drop
✓ 0.1s
['previous']
```

Εικόνα 25. Χαρακτηριστικό με correlation>0.9

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif

skb = SelectKBest(mutual_info_classif, k='all').fit(data.drop(['y'], axis=1), data['y'])
✓ 2.6s
```

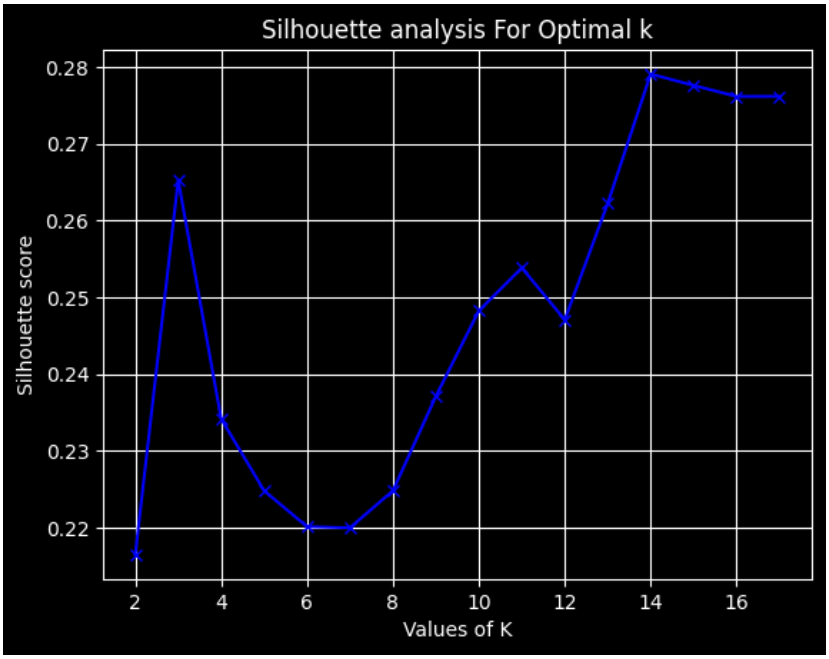
Εικόνα 26. Select KBest με mutual_info_clasif

```
from sklearn.feature_selection import SelectKBest, f_classif

skb = SelectKBest(f_classif, k='all').fit(data.drop(['y'], axis=1), data['y'])

pd.Series(skb.scores_, index=skb.get_feature_names_out()).sort_values(ascending=False)
✓ 0.0s
```

Εικόνα 27. Select KBest με f_classif

```
feats = ['duration', 'housing', 'balance', 'marital_married', 'month_may',
         'job_blue-collar', 'pdays', 'education_primary',
         'month_mar', 'education_tertiary', 'education_secondary', 'loan',
         'day', 'campaign']

X = data[feats].copy()
✓ 0.0s
```

Εικόνα 28. Επιλεγμένα feats



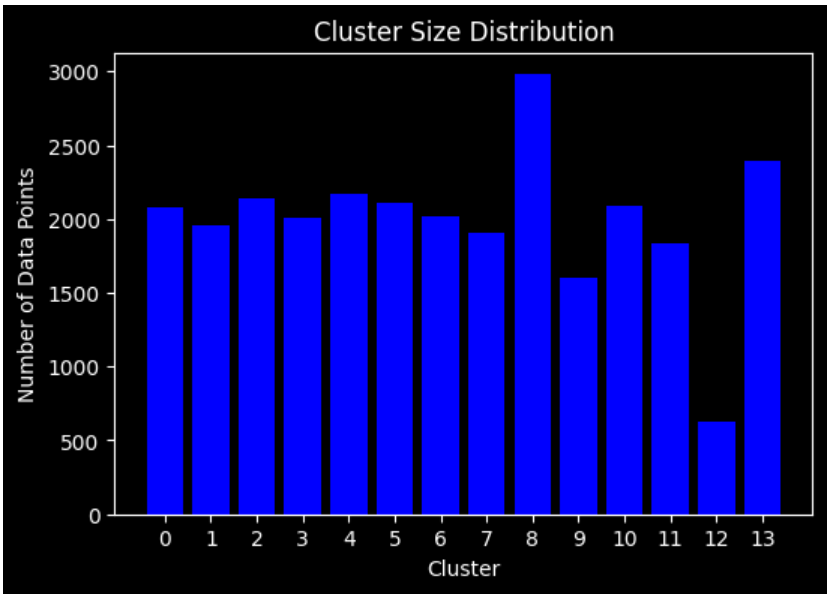
Εικόνα 29. Silhouette analysis

```
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score

kmeans = KMeans(n_clusters=14, random_state=42, n_init = 10)
kmeans.fit(X_train)
kmeans_labels = kmeans.labels_

print("K-Means Silhouette Score:", silhouette_score(X_train, kmeans_labels))
print("K-Means Calinski-Harabasz Score:", calinski_harabasz_score(X_train, kmeans_labels))
print("K-Means Davies-Bouldin Score:", davies_bouldin_score(X_train, kmeans_labels))
```

✓ 6.1s

```
K-Means Silhouette Score: 0.27905096869191937
K-Means Calinski-Harabasz Score: 4133.921862747297
K-Means Davies-Bouldin Score: 1.4667161551150119
```

Εικόνα 30. Calinski-Harabasz και Davies-Bouldin Score για το clustering



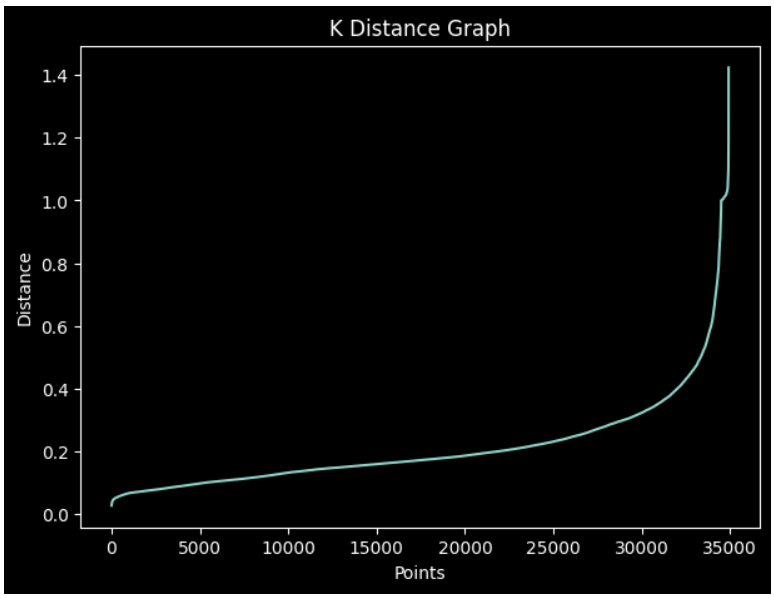
Εικόνα 31. Cluster size

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(n_jobs=-1)
dbscan.fit(X_train)
dbscan_labels = dbscan.labels_

n_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
print(f"Number of clusters found: {n_clusters}")
```

✓ 0.5s

```
Number of clusters found: 147
```

Εικόνα 32. Αριθμός cluster DBSCAN



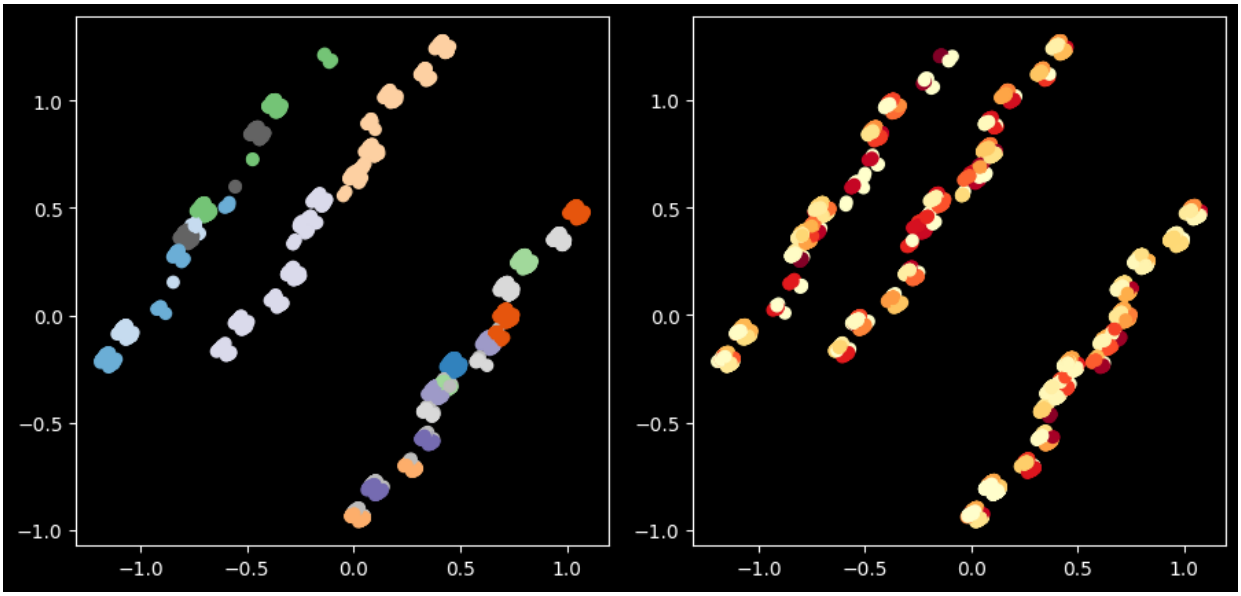
Εικόνα 33.Γράφημα κ απόστασης

```
dbscan = DBSCAN(eps=1, min_samples=6)
dbscan.fit(X_train)
dbscan_labels = dbscan.labels_

n_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
print(f"Number of clusters found: {n_clusters}")

print("DBSCAN Silhouette Score:", silhouette_score(X_train, dbscan_labels))
print("DBSCAN Calinski-Harabasz Score:", calinski_harabasz_score(X_train, dbscan_labels))
print("DBSCAN Davies-Bouldin Score:", davies_bouldin_score(X_train, dbscan_labels))
```

✓ 8.5s

```
Number of clusters found: 136
DBSCAN Silhouette Score: 0.3041390232366628
DBSCAN Calinski-Harabasz Score: 731.4336590307088
DBSCAN Davies-Bouldin Score: 1.2876720822569223
```

Εικόνα 34. Νέος αριθμός συστάδων και αντίστοιχα score



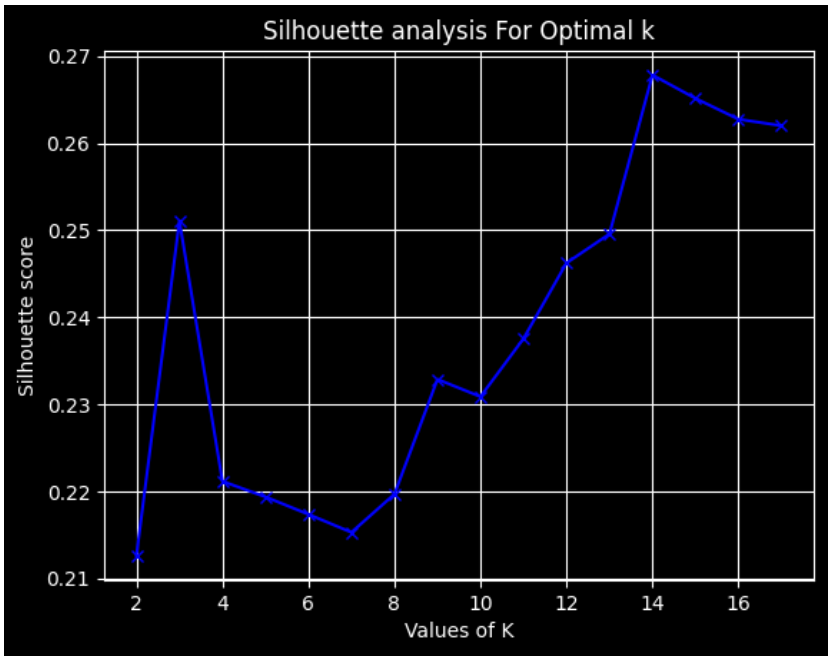
Εικόνα 35. Scatter plot KMeans και DBSCAN

```
from sklearn.feature_selection import VarianceThreshold

threshold = 0.1

selector = VarianceThreshold(threshold=threshold)
selector.fit(data)
selected_features = data.columns[selector.get_support()]

print("\nSelected features based on variance:")
print(selected_features)
```

✓ 0.0s

```
Selected features based on variance:
Index(['housing', 'loan', 'pdays', 'job_admin.', 'job_blue-collar',
       'job_management', 'job_technician', 'marital_divorced',
       'marital_married', 'marital_single', 'education_primary',
       'education_secondary', 'education_tertiary', 'month_aug', 'month_jul',
       'month_jun', 'month_may'],
      dtype='object')
```

Εικόνα 36. Επιλεγμένα feats 2



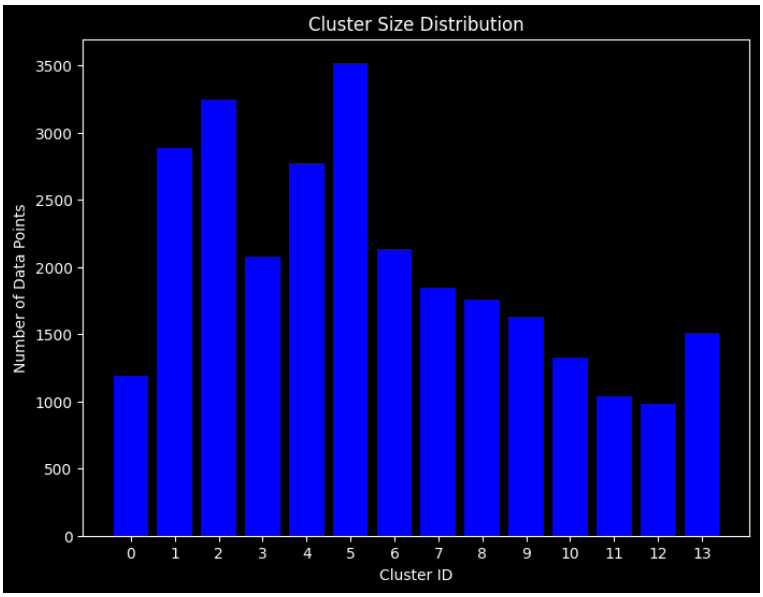
Εικόνα 37. Silhouette analysis 2

```
kmeans = KMeans(n_clusters=14, random_state=42, n_init = 10)
kmeans.fit(X_train)
kmeans_labels = kmeans.labels_

print("K-Means Silhouette Score:", silhouette_score(X_train, kmeans_labels))
print("K-Means Calinski-Harabasz Score:", calinski_harabasz_score(X_train, kmeans_labels))
print("K-Means Davies-Bouldin Score:", davies_bouldin_score(X_train, kmeans_labels))
```

✓ 4.9s

```
K-Means Silhouette Score: 0.26782863428154163
K-Means Calinski-Harabasz Score: 3404.9637848394664
K-Means Davies-Bouldin Score: 1.5746250878234227
```

Εικόνα 38. Calinski-Harabasz και Davies-Bouldin Score για το clustering 2



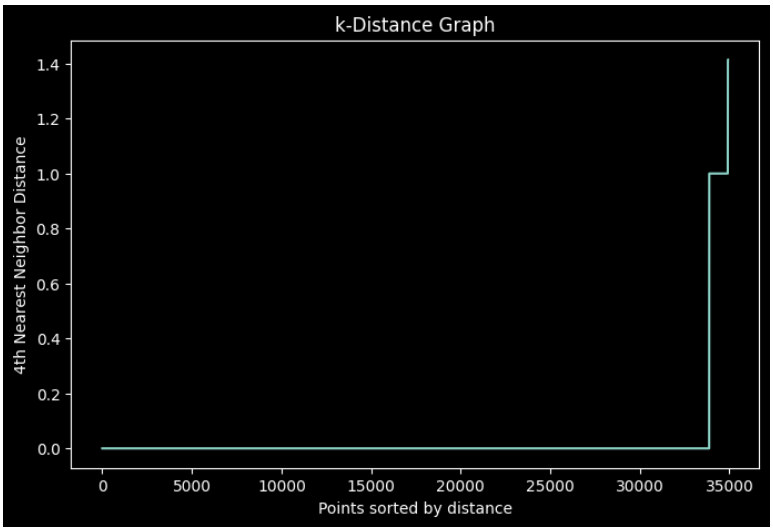
Εικόνα 39. Cluster size 2

```
dbscan = DBSCAN(n_jobs=-1)
dbscan.fit(X_train)
dbscan_labels = dbscan.labels_

n_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
print(f"Number of clusters found: {n_clusters}")
```

✓ 0.4s

```
Number of clusters found: 336
```

Εικόνα 40. Αριθμός cluster DBSCAN 2



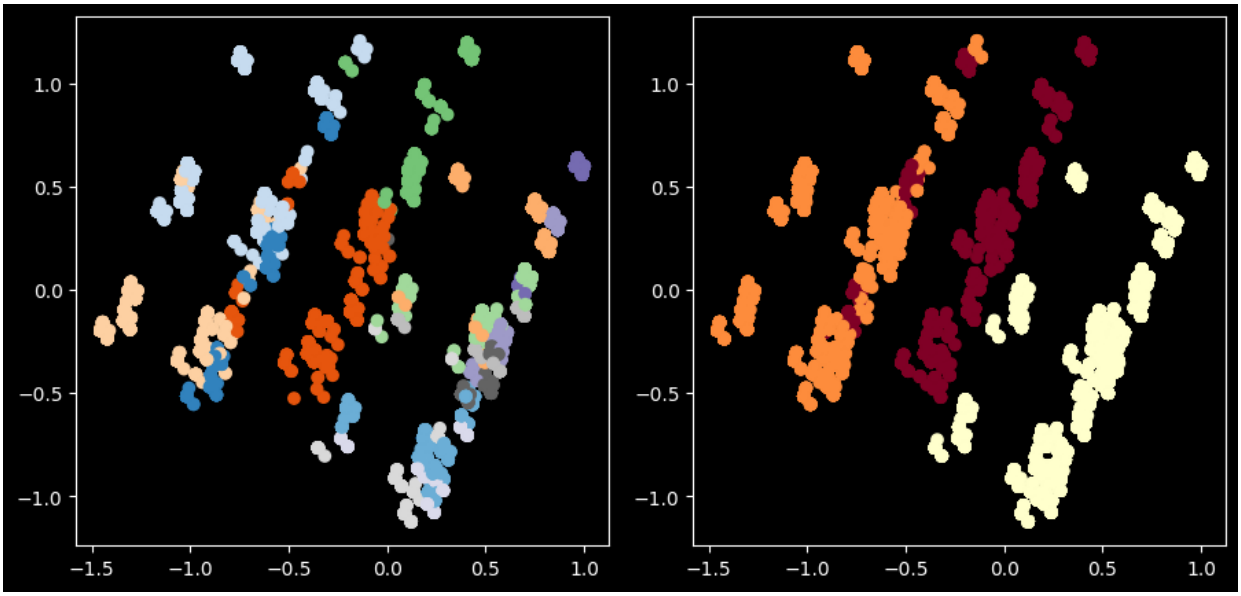
Εικόνα 41.Γράφημα κ απόστασης

```
dbscan = DBSCAN(eps=1, min_samples=6)
dbscan.fit(X_train)
dbscan_labels = dbscan.labels_

n_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
print(f"Number of clusters found: {n_clusters}")

print("DBSCAN Silhouette Score:", silhouette_score(X_train, dbscan_labels))
print("DBSCAN Calinski-Harabasz Score:", calinski_harabasz_score(X_train, dbscan_labels))
print("DBSCAN Davies-Bouldin Score:", davies_bouldin_score(X_train, dbscan_labels))
```

✓ 11.8s

```
Number of clusters found: 3
DBSCAN Silhouette Score: 0.2510685224149017
DBSCAN Calinski-Harabasz Score: 6206.0261389647185
DBSCAN Davies-Bouldin Score: 1.61839220203375
```

Εικόνα 42. Νέος αριθμός συστάδων και αντίστοιχα score 2



Εικόνα 43. Scatter plot KMeans και DBSCAN 2

3.1.3 classification.ipynb

```
# calculate distribution of y

value_counts = data['y'].value_counts()
total = value_counts.sum()
percentages = (value_counts / total) * 100
print("Counts:\n", value_counts)
print("\nPercentages:\n", percentages)
```

✓ 0.0s

```
Counts:
 y
0    32380
1     2539
Name: count, dtype: int64

Percentages:
 y
0    92.728887
1     7.271113
Name: count, dtype: float64
```

Εικόνα 44. Αναλογίες yes-no της μεταβλητής στόχος y

```
# uniform the y distribution
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(data.drop(columns='y'),data['y'])
```

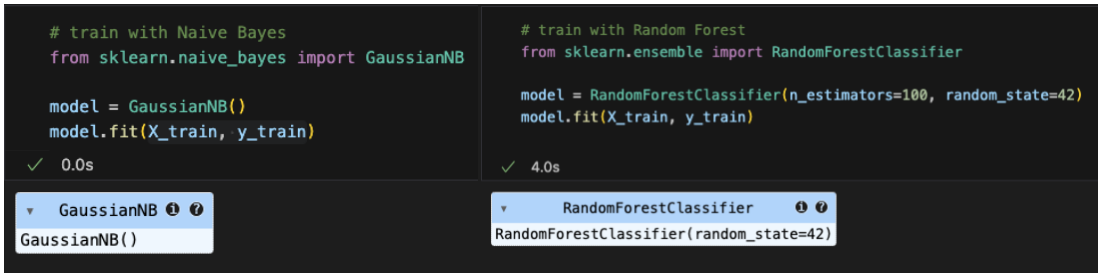
✓ 0.2s

```
value_counts = y_balanced.value_counts()
total = value_counts.sum()
percentages = (value_counts / total) * 100
print("Counts:\n", value_counts)
print("\nPercentages:\n", percentages)
```

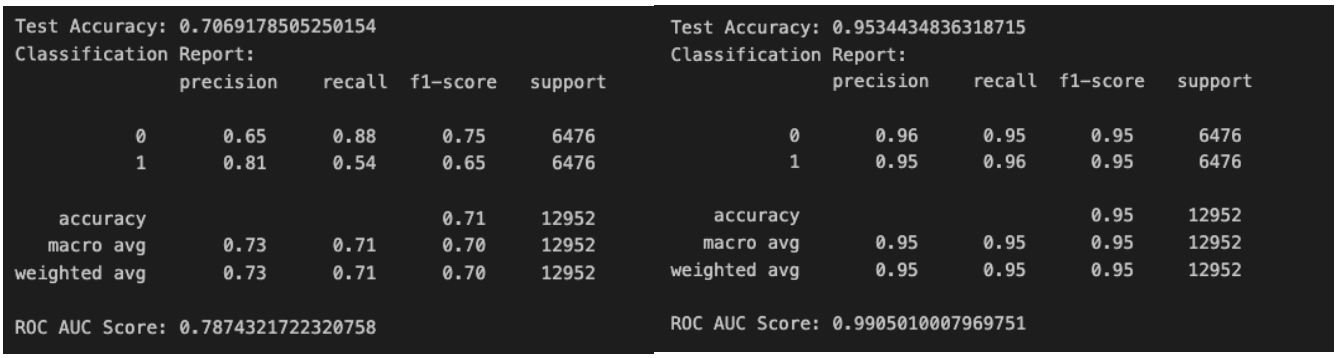
✓ 0.0s

```
Counts:
 y
0    32380
1    32380
Name: count, dtype: int64

Percentages:
 y
0    50.0
1    50.0
Name: count, dtype: float64
```

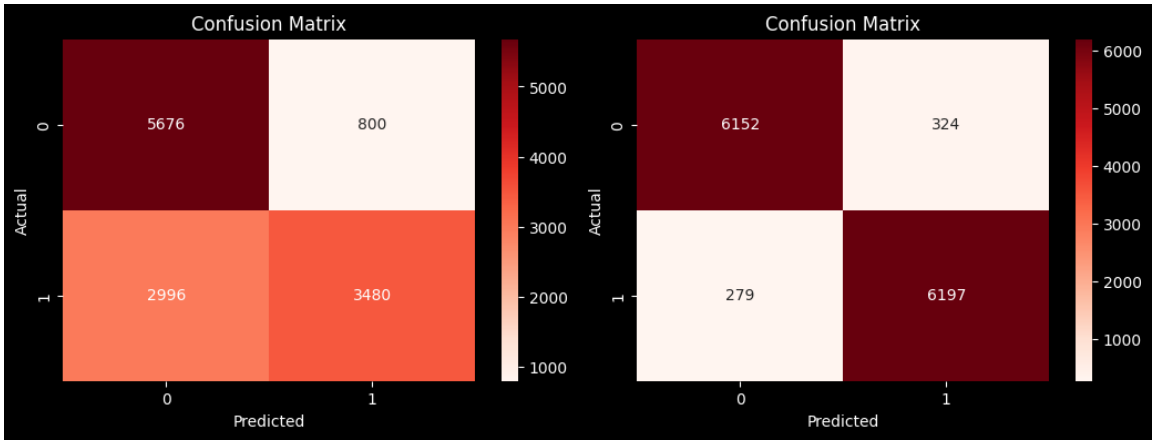
Εικόνα 45. Εφαρμογή SMOTE και αναλογίες yes-no της μεταβλητής στόχος y



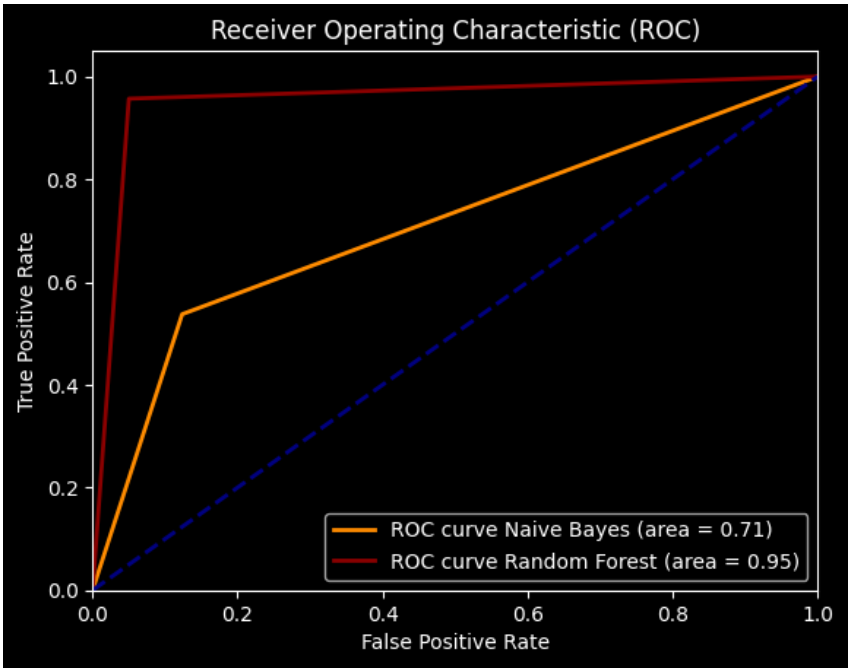
Εικόνα 46. Naïve Bayes Classifier και Random Forest Classifier



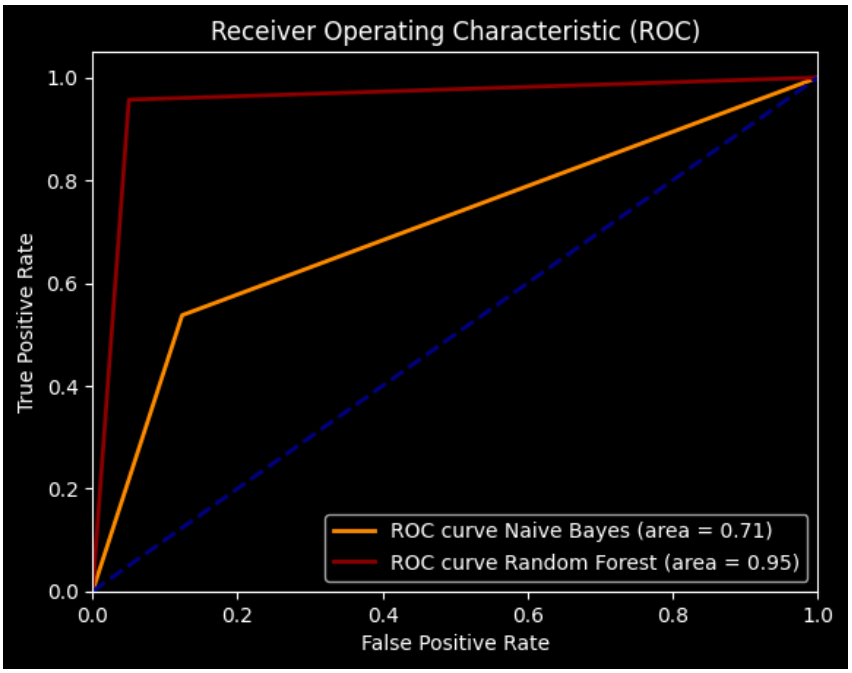
Εικόνα 47. Αποτελέσματα Naïve Bayes Classifier και Random Forest Classifier



Εικόνα 48. Confusion Matrix Naïve Bayes Classifier και Random Forest Classifier



Εικόνα 49. ROC Curve with AUC για Naïve Bayes Classifier και Random Forest Classifier



Εικόνα 50. Δημιουργία μοντέλου

```python
# compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

✓ 0.0s

Εικόνα 51. Compile μοντέλου

```python
# train the model
history = model.fit(X_train1, y_train1,
                    epochs=30,
                    batch_size=32,
                    validation_data=(X_val, y_val))
```

✓ 1m 27.7s Python

```
Epoch 1/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7208 - loss: 0.5574 - val_accuracy: 0.7796 - va
Epoch 2/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.7866 - loss: 0.4694 - val_accuracy: 0.7952 - va
Epoch 3/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8074 - loss: 0.4359 - val_accuracy: 0.8091 - va
Epoch 4/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8205 - loss: 0.4060 - val_accuracy: 0.8232 - va
Epoch 5/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8260 - loss: 0.3915 - val_accuracy: 0.8326 - va
Epoch 6/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8372 - loss: 0.3689 - val_accuracy: 0.8346 - va
Epoch 7/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8440 - loss: 0.3535 - val_accuracy: 0.8423 - va
Epoch 8/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8488 - loss: 0.3397 - val_accuracy: 0.8426 - va
Epoch 9/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8531 - loss: 0.3316 - val_accuracy: 0.8527 - va
Epoch 10/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8596 - loss: 0.3204 - val_accuracy: 0.8547 - va
Epoch 11/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8627 - loss: 0.3155 - val_accuracy: 0.8606 - va
Epoch 12/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8675 - loss: 0.3070 - val_accuracy: 0.8638 - va
Epoch 13/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8727 - loss: 0.2936 - val_accuracy: 0.8579 - va
Epoch 14/30
1134/1134 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8724 - loss: 0.2953 - val_accuracy: 0.8686 - va
Epoch 15/30
```

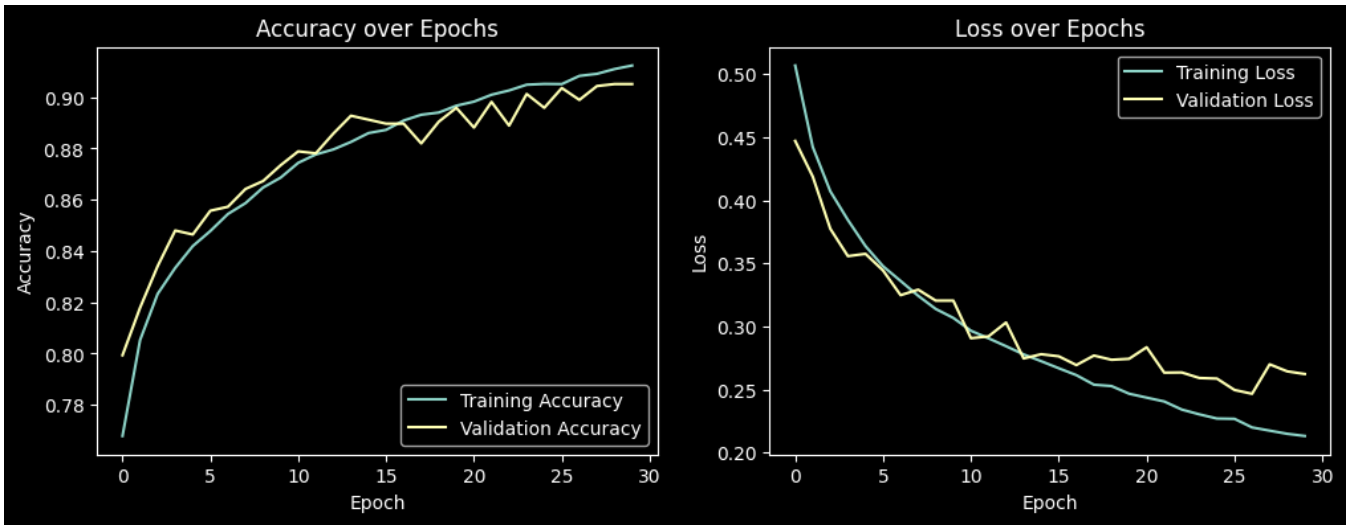
Εικόνα 52. Εκπαίδευση μοντέλου

```
# evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test accuracy: {test_acc}')
```

✓ 0.1s

```
203/203 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - accuracy: 0.8888 - loss: 0.2887
Test accuracy: 0.8923718333244324
```

Εικόνα 53. Test accuracy μοντέλου



Εικόνα 54. Διαγράμματα accuracy-loss μοντέλου

```
# save the model
model.save('model.h5')
```

✓ 0.0s

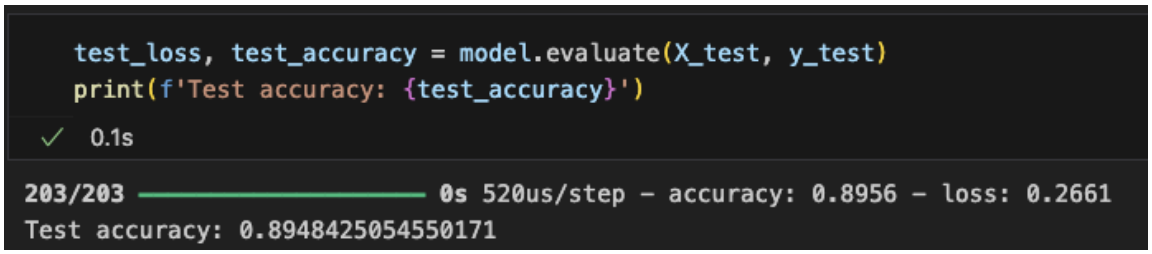
```
()` or `keras.saving.save_model(model)`. This file format is consi
```

```
# load the model
from tensorflow.keras.models import load_model

model = load_model('model.h5')
```

✓ 0.0s

```
WARNING:absl:Compiled the loaded model, but the compiled metrics
```

Εικόνα 55. Αποθήκευση και φόρτωση μοντέλου



Εικόνα 56. Test accuracy transfer learning μοντέλου

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] https://blog.dailydoseofds.com/p/missforest-and-knn-imputation-for (MissForest and kNN Imputation for Data Missing at Random)

[2] https://stackoverflow.com/questions/46498455/categorical-features-correlation/46498792#46498792 (Cramer)

[3] https://medium.com/@ktoprakucar/how-to-calculate-the-correlation-between-categorical-and-continuous-values-dcb7abf79406 (How to Calculate the Correlation Between Categorical and Continuous Values)

[4] https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48#1cb1 (Select KBest)

[5] https://www.kaggle.com/code/tanmaymane18/nearestneighbors-to-find-optimal-eps-in-dbscan (Optimal eps)

[6] https://machinelearningknowledge.ai/tutorial-for-dbscan-clustering-in-python-sklearn/ (Optimal eps)

[7] https://scikit-learn.org/stable/modules/feature_selection.html (Variance threshold)

[8] https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (SMOTE)

[9] https://builtin.com/data-science/transfer-learning (transfer learning)