ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

Αναλυτική και Μηχανική Μάθηση σε δεδομένα πωλήσεων

της

Πολύζου Μαρία

Υποβλήθηκε ως προαπαιτούμενο για την απόκτηση του Μεταπτυχιακού Διπλώματος ειδίκευσης στα Πληροφοριακά Συστήματα.

Επιβλέπων Καθηγητής

Κωνσταντίνος Ταραμπάνης

kat@uom.edu.gr

Σεπτέμβριος 2022

Ευχαριστίες

Η παρούσα εργασία αποτελεί διπλωματική εργασία στα πλαίσια του διατμηματικού προγράμματος μεταπτυχιακών σπουδών με ειδίκευση στα Πληροφοριακά Συστήματα.

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου κ. Κωνσταντίνο Ταραμπάνη για την πολύτιμη καθοδήγηση και υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Ευάγγελο Καλαμπόκη για την άμεση βοήθεια και καθοδήγηση που μου παρείχε κατά την υλοποίηση του τεχνικού μέρους της εργασίας.

ΠΕΡΙΛΗΨΗ

Η ανάπτυξη της τεχνολογίας έχει οδηγήσει τα τελευταία χρόνια στην απότομη αύξηση του όγκου δεδομένων. Σε αυτό έχει συμβάλλει η επιστήμη δεδομένων που απαρτίζεται από ένα μεγάλο σύνολο τεχνικών ανάλυσης και παίζει σημαντικό ρόλο σε πολλούς τομείς. Η παρούσα εργασία ασχολείται με την αναλυτική και μηχανική μάθηση δεδομένων πωλήσεων της διαδικτυακής πλατφόρμας Wish.com. που εξειδικεύεται στο ηλεκτρονικό εμπόριο. Τα δεδομένα παρέχουν πληροφορίες για προϊόντα που είναι προς πώληση συγκεκριμένο χρονικό διάστημα. Πάνω σε αυτά εκτελούνται όλα τα απαραίτητα βήματα εξερεύνησης και επεξεργασίας που χρειάζονται με σκοπό την εμφάνιση σημαντικών επιχειρηματικών συμπερασμάτων γύρω από την επιτυχία πωλήσεων του ιστότοπου. Στη μελέτη παρουσιάζονται 16 διαφορετικά ερωτήματα που διερευνούν τη καταναλωτική συμπεριφορά των χρηστών του Wish.com και τους παράγοντες που επηρεάζονται οι πωλήσεις. Απαντώνται με τη δημιουργία γραφημάτων – οπτικοποιήσεων. Επιπλέον αναπτύσσονται 7 διαφορετικοί αλγόριθμοι Επιβλεπόμενης Μάθησης σε πρόβλημα δυαδικής ταξινόμησης με σκοπό τη πρόβλεψη επιτυχίας ή μη των πωλήσεων. Τα μοντέλα που εκπαιδεύονται είναι τα Νευρωνικά Δίκτυα, KNN, Decision Tree Classifier, Logistic Regression, Gaussian NB, Random Forest Classifier και XG Boost Classifier. Οι μετρικές απόδοσης που επιλέγονται για την αξιολόγηση των μοντέλων είναι οι Classification Report, Confusion Matrix και AUC Score . Για την επιλογή του καλύτερου ταξινομητή που είναι το XGBoost συμβάλλει το F1 score και AUC score με ποσοστό 92% και 99% αντίστοιχα.

ABSTRACT

The development of technology has led in recent years to a sharp increase in the volume of data. Contributing to this is data science, which consists of a large set of analysis techniques and plays an important role in many fields. One of these areas is e-commerce and retail, since through the appropriate choice of analysis it leads to important business decisions for a company. This paper deals with the analytical and machine learning of sales data of the online platform Wish.com. which specializes in e-commerce. The data provides information about products that are for sale during a certain period. On these are performed all the necessary exploration and processing steps needed to show important business conclusions around the sales success of the website. The study presents 16 different questions that explore the consumer behavior of Wish.com users and the factors that influence sales. They are answered by creating graphs - visualizations. In addition, 7 different Supervised Learning algorithms are developed in a binary classification problem to predict sales success or failure. The models trained are Neural Networks, KNN, Decision Tree Classifier, Logistic Regression, Gaussian NB, Random Forest Classifier and XG Boost Classifier. The performance metrics chosen to evaluate the models are Classification Report, Confusion Matrix and AUC Score. For the selection of the best classifier which is XGBoost, the F1 score and AUC score contribute with a percentage of 92% and 99% respectively.

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

1 Εισαγωγή

1.1 Κίνητρο

Η εφαρμογή της επιστήμης των δεδομένων στην εποχή μας παίζει σπουδαίο ρόλο στη βιομηχανία του ηλεκτρονικού και λιανικού εμπορίου. Μέσα από τα δεδομένα οι εταιρείες μπορούν να κατανοήσουν τις επιδόσεις τους και να λάβουν σημαντικές επιχειρηματικές αποφάσεις που θα οδηγήσουν σε κέρδος. Ένα παράδειγμα είναι η ανάλυση και πρόβλεψη της καταναλωτικής συμπεριφοράς πελατών και η επιτυχία πωλήσεων των προϊόντων. Οι εταιρείες συνηθίζουν να δημιουργούν ένα προφίλ πελατών ώστε να αντιλαμβάνονται τις προτιμήσεις τους μέσα από διάφορους παράγοντες κατά τη δράση τους σε έναν ιστότοπο λιανικών πωλήσεων. Έτσι προωθούν σωστά τις πληροφορίες των προϊόντων και ωθούν τον καταναλωτή προς αγορά. (‘Εφαρμογή Επιστήμης Δεδομένων _ Παραδείγματα εφαρμογών επιστήμης δεδομένων’, no date)

Τα παραπάνω γίνονται με την δημιουργία κατάλληλων οπτικοποιήσεων – γραφημάτων που προβάλουν μια ποιοτική εικόνα σε μία εταιρεία καθώς και με τη δημιουργία μοντέλων μηχανικής μάθησης.

Ο τεράστιος όγκος δεδομένων που λαμβάνει καθημερινά μία εταιρεία πρέπει να αξιοποιηθεί σωστά από το τμήμα μάρκετινγκ και πωλήσεων. Για να μεταφραστούν, να συσχετιστούν και να συγκριθούν τα δεδομένα πρέπει να απεικονίζονται σε οπτική μορφή που ονομάζεται οπτικοποίηση δεδομένων. Ο ανθρώπινος εγκέφαλος προσαρμόζεται καλύτερα σε μία εικόνα απ’ ότι σε αριθμούς. Έτσι γίνεται πιο εύκολη η παρακολούθηση στόχων και η αύξηση παραγωγικότητας και πωλήσεων.(‘What Can Data Visualization Do For Your Sales Department _ CustomerThink’, no date)

Η μηχανική μάθηση θεωρείται μία απ’ τις μεγαλύτερες τεχνολογικές επαναστάσεις τα τελευταία χρόνια στο τομέα των πωλήσεων. Βελτιώνει την εμπειρία των πελατών και την αποτελεσματική αύξηση κέρδους στις επιχειρήσεις. Εντάσσεται στην κατηγορία τεχνητής νοημοσύνης και η βασική της λειτουργία είναι να σχεδιάζει, αναλύει και αναπτύσσει αλγόριθμους. Μέσα από την αναγνώριση προτύπων και μοτίβων μετατρέπει τα δεδομένα σε χρήσιμες πληροφορίες που οδηγούν σε χρήσιμα

επιχειρηματικά συμπεράσματα. Κάποια από αυτά είναι ο εντοπισμός πιθανών πελατών, η βελτίωση παραγωγικότητας πωλήσεων και η ενσωμάτωση νέων διαφημιστικών ωθήσεων. (‘Five ways machine learning can improve your sales - CatalogPlayer’, no date)

1.2 Στόχος μελέτης

Στόχος της εργασίας είναι η παρουσίαση εργαλείων ανάλυσης δεδομένων για την απόκτηση αποτελεσματικών πληροφοριών με σκοπό την βέλτιστη ανταγωνιστικότητα ενός ιστότοπου λιανικών πωλήσεων. Συγκεκριμένα διερευνώνται ποιοι παράγοντες επηρεάζουν την επιτυχία πωλήσεων της πλατφόρμας wish. com με χρήση κατάλληλων οπτικοποιήσεων και στη συνέχεια γίνεται η πρόβλεψη της επιτυχίας των πωληθέντων προϊόντων με τη χρήση μοντέλων μηχανικής μάθησης. Ο αναγνώστης θα μπορεί να αντιληφθεί ποια είναι τα βήματα που συντελούν ώστε να φτάσει σε σημαντικά επιχειρηματικά συμπεράσματα και να κρίνει με βάση ποιες μετρικές απόδοσης επιλέγει το καλύτερο μοντέλο.

1.3 Περιεχόμενο μελέτης

Το κεφάλαιο 2 αναφέρεται στη μεθοδολογία που ακολουθείται για την εργασία. Χωρίζεται σε υπό ενότητές που περιγράφουν το κάθε βήμα που θα ακολουθήσει έπειτα στην υλοποίηση. Αρχικά γίνεται αναφορά στη πηγή και περιγραφή δεδομένων, στις τεχνολογίες και βιβλιοθήκες που γίνεται χρήση, στην εξερεύνηση των δεδομένων, στην προ επεξεργασία των δεδομένων και τέλος στη δημιουργία των μοντέλων μηχανικής μάθησης.

Στο κεφάλαιο 3 αναφέρεται η υλοποίηση της εργασίας. Γίνεται εκτενέστερη αναφορά του τρόπου που εκτελείται κάθε βήμα. Στο 1ο βήμα μελέτης δεδομένων, περιγράφεται τί αντιπροσωπεύει κάθε στήλη του dataset, της πληροφορίες που αντλούνται μέσω περιγραφικής στατιστικής, τον έλεγχο για λάθος τιμές των δεδομένων και τέλος 16 ερωτήματα που απαντώνται σε μορφή οπτικοποιήσεων σχετικά με τους παράγοντες που επηρεάζουν την επιτυχία πωλήσεων προϊόντων συγκεκριμένης χρονικής περιόδου. Στο 2ο βήμα προ επεξεργασίας δεδομένων γίνεται ο χειρισμός των

λανθασμένων – ελλιπών και ακραίων τιμών, η δημιουργία νέων στηλών και η κωδικοποίηση των μεταβλητών. Τέλος, στο 3ο βήμα εφαρμόζεται το σκέλος της μηχανικής μάθησης. Πριν τη δημιουργία μοντέλων γίνεται αναφορά στον ορισμό της επιβλεπόμενης μηχανικής μάθησης και του προβλήματος ταξινόμησης αφού και η εκπαίδευση-πρόβλεψη των μοντέλων πάνω στην επιλεγμένη εξαρτημένη μεταβλητή του συνόλου δεδομένων της πλατφόρμας Wish.com αφορά αυτό το είδος προβλήματος. Κατά την εκπαίδευση εφαρμόζεται η μέθοδος SMOTE λόγω της ανισορροπίας των δεδομένων εκπαίδευσης της εξαρτημένης μεταβλητής και ταυτόχρονα με τις τεχνολογίες που συντελούν για την εκπαίδευση των αλγορίθμων περιγράφεται και η λειτουργία τους με βιβλιογραφικές αναφορές. Επιπλέον αναφέρεται και ο ορισμός κάθε μοντέλου καθώς και οι υπερπαράμετροι που επιλέγονται για την εκπαίδευση τους.

Στο κεφάλαιο 4 αναφέρονται τα αποτελέσματα των μοντέλων με και χωρίς oversampling μαζί με τις μετρικές – σκορ απόδοσης και bar plots με τη σημαντικότητα των μεταβλητών.

Στο κεφάλαιο 5 γίνεται η σύγκριση των αποτελεσμάτων σχετικά με τα μοντέλα που έχουν τις υψηλότερες τιμές σε κάθε μετρική και οι περιγραφές αυτών.

Τέλος στο κεφάλαιο 6 αναφέρονται τα συμπεράσματα για το κομμάτι οπτικοποιήσεων και μηχανικής μάθησης. Στο τελευταίο κομμάτι αναφέρεται το καλύτερο μοντέλο και για ποιο λόγο επιλέγεται αυτό σύμφωνα με τη σημαντικότητα των μετρικών καθώς και οι μεταβλητές που είναι πιο σημαντικές στο παραπάνω μοντέλο. Επιπλέον αναφέρονται μελλοντικά εργαλεία που θα μπορούσαν να χρησιμοποιηθούν στο κομμάτι της ανάλυσης όπως και παραπάνω ενέργειες που θα μπορούσαν να γίνουν κατά την εκπαίδευση των αλγορίθμων.

2 Μεθοδολογία

2.1 Πηγή δεδομένων

Για την εκπόνηση της εργασίας επιλέγονται δεδομένα που προέρχονται από την πλατφόρμα Wish.com μια αμερικάνικη διαδικτυακή πλατφόρμα ηλεκτρονικού

εμπορίου όπου ένας χρήστης μπορεί να κάνει ηλεκτρονικές αγορές σε προϊόντα που επιθυμεί από μία τεράστια ποικιλία πωλητών. Η μεγαλύτερη επιτυχία της εφαρμογής στηρίζεται στην ευελιξία χρήσης της στα κινητά. Είναι μία από τις κορυφαίες επιλογές στο Google Play και Apple App Store και ιδρύθηκε το 2010 στο Σαν Φρανσίσκο από τους Peter Szulczewski και Danny Zhang. (‘Wish (company) - Wikipedia’, no date)

Το σύνολο δεδομένων αποτελείται από πληροφορίες που σχετίζονται με καλοκαιρινά διαθέσιμα προϊόντα προς πώληση από τον Ιούλιο του 2020. Περιέχουν καταχωρήσεις και αξιολογήσεις προϊόντων καθώς και την αντίστοιχη απόδοση πωλήσεων. Τα προϊόντα προκύπτουν αν ο χρήστης πληκτρολογήσει «καλοκαίρι» στο πεδίο αναζήτησης της πλατφόρμας. Το Wish.com πρόσφερε ένα κομμάτι δεδομένων στην πλατφόρμα Kaggle στα πλαίσια διαγωνισμού. Ύστερα από μία εκτενή αναζήτηση, επιλέχτηκε το παραπάνω σύνολο δεδομένων. Η ανάλυσή του θα γίνει με περισσότερες λεπτομέρειες στα επόμενα κεφάλαια.

2.2 Εργαλεία

Κατά την διάρκεια της εργασίας χρησιμοποιούνται εργαλεία και τεχνολογίες που χρειάζεται να αναφερθούν για καλύτερη εικόνα από τον αναγνώστη. Όπως αναφέρθηκε και πιο πάνω καθοριστικό παράγοντα για την ανεύρεση των δεδομένων έπαιξε η πλατφόρμα Kaggle. Ωστόσο το μεγαλύτερο κομμάτι της εργασίας καλύφθηκε από την γλώσσα Python με σκοπό την ανάλυση δεδομένων και την δημιουργία και εκπαίδευση μοντέλων. Για να μπορέσει να εκτελεστεί ο κώδικας αξιοποιήθηκε το εργαλείο JupyterLab και Jupyter Notebook.

Η πλατφόρμα Kaggle, θυγατρική της Google, είναι μία διαδικτυακή κοινότητα επιστημόνων δεδομένων και μηχανικής μάθησης που επιτρέπει στους χρήστες να δημοσιεύουν σύνολα δεδομένων με σκοπό την εξερεύνησή τους και την εκπαίδευση μοντέλων. Πλέον η Kaggle θεωρείται μία δημόσια πλατφόρμα δεδομένων . (‘Kaggle - Wikipedia’, no date)

Το Jupyter Lab είναι ένα δια δραστικό περιβάλλον ανάπτυξης που στηρίζεται στο διαδίκτυο για σημειωματάρια, κώδικα και δεδομένα. Επιτρέπει τη δημιουργία

εγγράφων που περιέχουν κώδικα, οπτικοποιήσεις και κείμενο. Με την τεχνολογία Jupyter Lab χρησιμοποιείται το Jupyter Notebook που επιτρέπει τη χρήση κώδικα μέσω της Python σε μορφή κελιών. (Project Jupyter, 2019)

Για την εκτέλεση της ανάλυσης γίνεται χρήση σχεδόν σε όλο το κομμάτι της εργασίας η γλώσσα Python. Σημαντικός παράγοντας για την λειτουργικότητα του κώδικα είναι η χρήση βιβλιοθηκών που παρέχουν συναρτήσεις και αλγόριθμους. Παρακάτω γίνεται μία αναφορά των βιβλιοθηκών που συντελούν στην υλοποίηση του κώδικα.

- Pandas : χειρίζεται δεδομένα σε μορφή Dataframe. Είναι ένα γρήγορο και ευέλικτο πακέτο ανάλυσης δεδομένων που παρέχει έτοιμες συναρτήσεις με πολλές λειτουργικότητες όπως μετατροπή, καθάρισμα και φιλτράρισμα αυτών. (McKinney and Team, 2015). Χρησιμοποιείται κυρίως στο κομμάτι της εξερεύνησης και προ επεξεργασίας των δεδομένων.
- Numpy : χρησιμοποιείται για τον χειρισμό πινάκων. Παρέχει ένα σύνολο έτοιμων υπολογιστικών συναρτήσεων. (‘What is NumPy_ — NumPy v1’, no date)
- Matplotlib : είναι μία ολοκληρωμένη βιβλιοθήκη στατιστικών απεικονίσεων. (Hunter *et al.*, 2020)
- Seaborn : είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων που βασίζεται στο matplotlib. Προσφέρει μια διεπαφή υψηλού επιπέδου για τη σχεδίαση ελκυστικών και ενημερωτικών στατιστικών γραφικών. (Waskom, 2021)

- Plotly : είναι μια διαδραστική βιβλιοθήκη ανοιχτού κώδικα που υποστηρίζει οπτικοποιήσεις που καλύπτουν ένα ευρύ φάσμα στατιστικών, οικονομικών, γεωγραφικών, επιστημονικών και τρισδιάστατων περιπτώσεων χρήσης. (Plotly, 2020)
- Sklearn : είναι από τα πιο γνωστά εργαλεία μηχανικής μάθησης της Python. Χρησιμοποιείται για να χτίσει μοντέλα και παρέχει όλες τις λειτουργίες για κάθε στάδιο προγνωστικής ανάλυσης των δεδομένων. (kunal, 2015)
- Imblearn : είναι ένα πακέτο της Python που χρειάζεται όταν υπάρχει ανισορροπία μεταξύ των κλάσεων της μεταβλητής στόχου. Προσφέρει ένα

σύνολο τεχνικών oversampling και undersampling. (Lemaitre and Aridas, no date)

- Collections : περιέχει κοντέινερς για την αποθήκευση συλλογών (collection) δεδομένων. Πολύ συχνή η χρήση της συνάρτησης counter για γρήγορες μετρήσεις. (‘Introduction to Python’s Collections Module’, no date)
- Xgboost : είναι μία βιβλιοθήκη σχεδιασμένη να τρέχει αλγόριθμους μηχανικής μάθησης στο πλαίσιο Gradient Boosting με εξαιρετικά αποδοτικό και ευέλικτο τρόπο. (XGBoost, 2020)

Σχετικά με τα μοντέλα που εκπαιδεύονται και τις τεχνικές που εκτελούνται στο τελευταίο στάδιο του Machine Learning θα γίνει εκτενέστερη αναφορά στο κεφάλαιο της υλοποίησης.

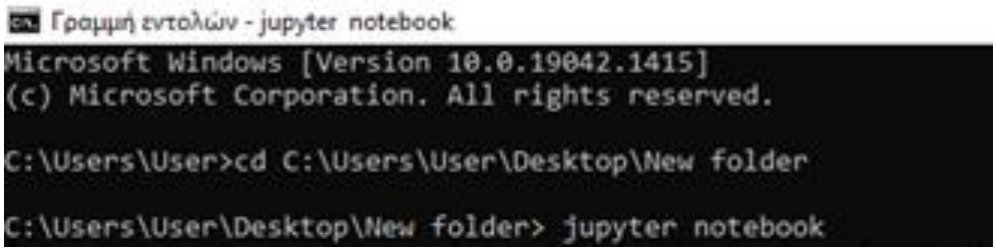
2.3 Βήματα

2.3.1 Εξερεύνηση δεδομένων

Στο πρώτο στάδιο αυτού του βήματος αφού επιλέγεται το αρχείο που θα γίνει η ανάλυση, γίνεται μελέτη των στηλών για το τί αντιπροσωπεύει κάθε μία. Ο ιστότοπος Wish παρέχει τρία CSV αρχεία. Το μεγαλύτερο κομμάτι της ανάλυσης γίνεται στο κεντρικό dataset και για τη δημιουργία οπτικοποιήσεων επιλέγονται και τα άλλα δύο. Τα 3 αρχεία σε μορφή csv είναι τα εξής :

- Πίνακας “ summer_products_with_rating_and_perfomance_2020-08.
- Πίνακας “unique_categories”
- Πίνακας “unique_categories.sorted_by_count”

Τα παραπάνω datasets φορτώνονται τοπικά στο κεντρικό υπολογιστή σε φάκελο με την ονομασία “New folder" ώστε να περαστούν στο Jupyter Notebook για να ξεκινήσει η ανάλυση. Ο 1ος πίνακας για λόγους διευκόλυνσης μετονομάζεται σε “sales_data”. Η εφαρμογή Jupyter ανοίγει με τη γραμμή εντολών όπου διαβάζει τον φάκελο με τα αρχεία με την εντολή “cd (φάκελος αρχείων)” και ύστερα γίνεται κλήση του “jupyter notebook”.



Εικόνα 1 Γραμμή εντολών για άνοιγμα της εφαρμογής Jupyter notebook

Με τη βοήθεια του Jupyter Notebook και της python γίνεται η φόρτωση των αρχείων μέσω της .read_csv() συνάρτησης η οποία ανήκει στο Pandas πακέτο και διαβάζοντας τα δεδομένα επιστρέφει ένα Data frame. Στη συνέχεια επιλέγεται η συνάρτηση .head() που εμφανίζει τις πρώτες πέντε σειρές του πίνακα και δίνει μια ολοκληρωμένη εικόνα των πληροφοριών που περιέχει η κάθε στήλη.

Στο δεύτερο στάδιο, γίνεται έλεγχος του τύπου των μεταβλητών, των ελλιπών τιμών και το πλήθος αυτών, της συσχέτισης που έχει το κάθε feature με το target πρόβλεψης και ορισμένα στατιστικά αποτελέσματα κάθε στήλης. Επιπλέον γίνεται έλεγχος για διπλότυπες εγγραφές, για το πλήθος μοναδικών τιμών σε κάθε στήλη καθώς και για την ύπαρξη ακραίων τιμών στα δεδομένα.

Τέλος το κομμάτι του exploratory ολοκληρώνεται με τη δημιουργία κατάλληλων οπτικοποιήσεων για να γίνει εμβάθυνση της καταναλωτικής συμπεριφοράς- προτιμήσεων των πελατών και των πωλήσεων του Wish.com. Παρουσιάζονται 16 διαφορετικά ερωτήματα με τα αντίστοιχα γραφήματα που δείχνουν σημαντικές πληροφορίες για την διερεύνηση της επιτυχίας του ιστότοπου. Το είδος των διαγραμμάτων θα αναφερθεί στο επόμενο κεφάλαιο.

2.3.2 Προ επεξεργασία δεδομένων

Σε αυτό το βήμα ό,τι έλεγχος και πληροφορία αντλείται απ' την εξερεύνηση των δεδομένων θα χρησιμοποιηθεί ώστε να γίνει κατάλληλη επεξεργασία και ετοιμασία του dataset για το κομμάτι του Machine Learning. Το βήμα αυτό δημιουργείται σε ένα δεύτερο notebook.

Αρχικά αφαιρούνται οι διπλότυπες εγγραφές που προκαλούν σύγχυση και περιττή πληροφορία στα μοντέλα. Γίνεται λεπτομερή ανάλυση των κατηγορικών και αριθμητικών αντίστοιχα μεταβλητών με σκοπό να εντοπιστούν, να αφαιρεθούν και να επεξεργαστούν ελλιπείς τιμές καθώς και τιμές που επαναλαμβάνονται με διαφορετικό τρόπο. Επίσης αποκαθίσταται το πρόβλημα που υπάρχει με τις ακραίες τιμές σε ορισμένες στήλες και γίνονται νέες στήλες με πιο ουσιαστικές πληροφορίες. Μετά την ανάλυση του κάθε feature του dataset αφαιρούνται μη απαραίτητες στήλες που δεν έχουν τόσο υψηλή συσχέτιση με το target του δείγματος. Τέλος ακολουθεί το τελικό στάδιο της αποκωδικοποίησης των μεταβλητών που είναι απαραίτητο καθώς τα μοντέλα αναγνωρίζουν μόνο αριθμητικές μεταβλητές. Τέλος γίνεται έλεγχος του target πρόβλεψης και μία επεξεργασία στις τιμές του καθώς η τεχνική μηχανικής μάθησης είναι πρόβλημα classification-ταξινόμησης με δυαδικές κλάσεις. Πιο λεπτομερής ανάλυση αυτού του θέματος θα γίνει στο κεφάλαιο της υλοποίησης. Για να συνεχισθεί το επόμενο βήμα, το νέο dataset αποθηκεύεται σε μορφή csv στο jupyter lab.

2.3.3 Δημιουργία μοντέλων

Σε αυτό το βήμα χρησιμοποιούνται πολλές τεχνικές όπως και ορισμοί της μηχανικής μάθησης, του προβλήματος ταξινόμησης, των μοντέλων εκπαίδευσης και των μετρικών απόδοσης αυτών που αναφέρονται στο κεφάλαιο της υλοποίησης αναλυτικότερα. Ωστόσο θα γίνει μία γενική αναφορά για καλύτερη κατανόηση των διαδικασιών μετέπειτα.

Αφού ολοκληρώνεται η επεξεργασία των δεδομένων δημιουργείται ένα τρίτο notebook με το νέο dataset. Αρχικά το σύνολο δεδομένων χωρίζεται σε δύο μέρη X και y όπου το τελευταίο είναι η εξαρτημένη μεταβλητή που θα γίνει η πρόβλεψη και το πρώτο οι ανεξάρτητες μεταβλητές. Στη συνέχεια γίνεται ένα split - διαχωρισμός των δεδομένων σε train και test. Μετά τη μέθοδο SMOTE, γίνεται εκπαίδευση 7 διαφορετικών μοντέλων με και χωρίς χρήση της τεχνικής υπεδειγματοληψίας για να γίνει και σύγκριση των αποτελεσμάτων στο τέλος.

2.3.4 Συμπεράσματα

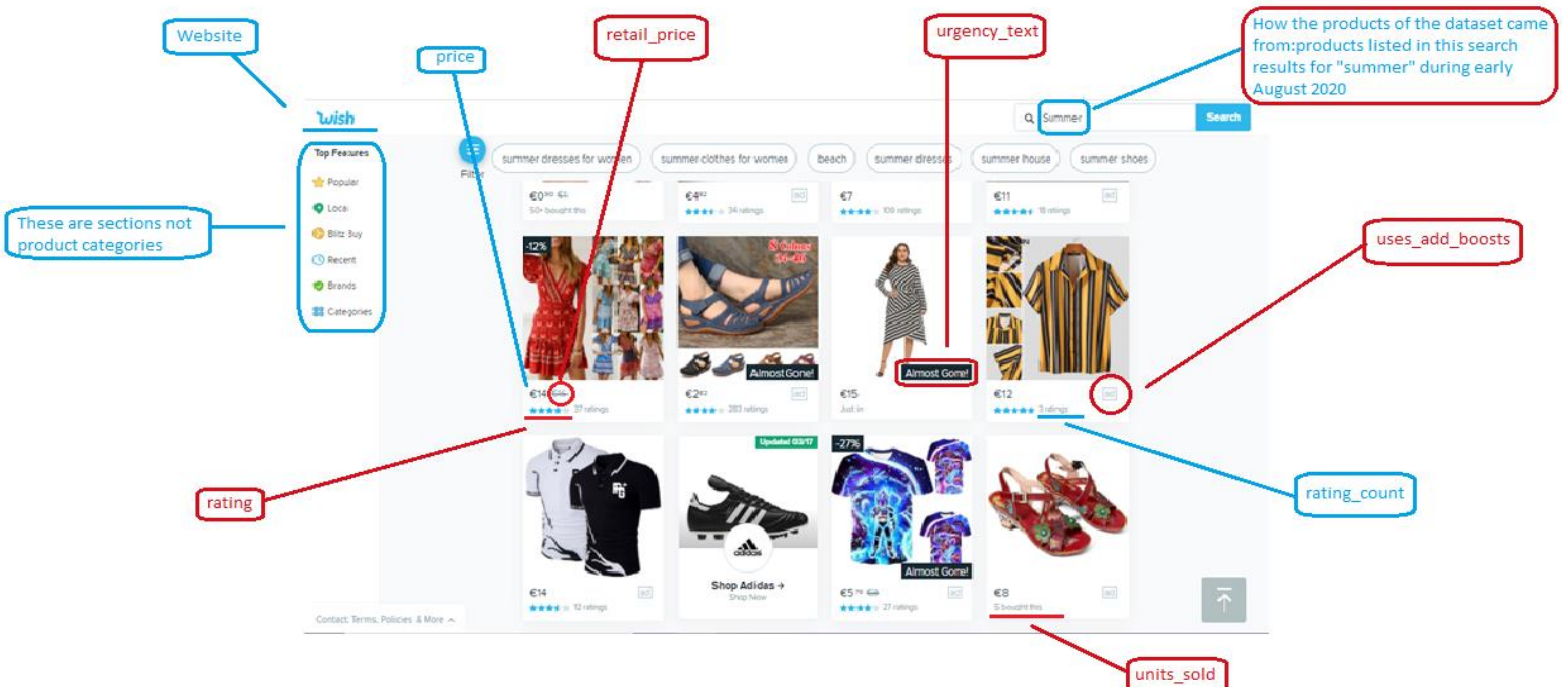
Αρχικά παρουσιάζονται τα συμπεράσματα των οπτικοποιήσεων των δεδομένων της πλατφόρμας Wish.com που εμβαθύνουν στον προσδιορισμό βασικών σχέσεων σχετικά με την επιτυχία πώλησης ενός προϊόντος. Τέλος, παρουσιάζονται τα αποτελέσματα των μοντέλων που έχουν εκπαιδευτεί και γίνεται σύγκριση μεταξύ τους για την καλύτερη πρόβλεψη σύμφωνα με τα σκορ που εμφανίζονται.

3 Υλοποίηση

3.1 Εισαγωγή και Exploratory (Μελέτη Δεδομένων)

Σε αυτό το κεφάλαιο γίνεται αναλυτική περιγραφή των ενεργειών που εκτελούνται με σκοπό την ολοκλήρωση του έργου. Εκτός από την υλοποίηση των βημάτων, γίνεται και μία βιβλιογραφική αναφορά τεχνολογιών που χρησιμοποιούνται με στόχο την καλύτερη κατανόηση του αναγνώστη.

Το κύριο σύνολο δεδομένων που επιλέγεται για την ανάλυση ονομάζεται “summer_products_with rating_and_performance_2020-08 ” και αποτελείται από 43 στήλες και 1573 παρατηρήσεις. Παρακάτω παρουσιάζεται μία αντιπροσωπευτική εικόνα του website όπου μπορεί κανείς να δει και το περιεχόμενο κάποιων στηλών για καλύτερη κατανόηση.



Εικόνα 2 Website Wish.com

Για την ανάλυση των δεδομένων γίνεται εγκατάσταση βιβλιοθηκών για την χρήση κατάλληλων συναρτήσεων. Έχει γίνει αναφορά των βιβλιοθηκών στην υπό ενότητα 2.2 “Εργαλεία”.

```
In [124]: # Εγκατάσταση βιβλιοθηκών
          !pip install seaborn
          !pip install plotly
          !pip install matplotlib_venn
          !pip install  WordCloud
          import pandas as pd
          import numpy as np
          import seaborn as sns
          %matplotlib inline
          import plotly
          import plotly.graph_objects as go
          import plotly.express as px
          import matplotlib.pyplot as plt
          from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
          from matplotlib_venn import venn3, venn3_circles
          from plotly.subplots import make_subplots
          from plotly.offline import iplot, init_notebook_mode
          init_notebook_mode(connected = True)
          !pip install cufflinks plotly
          from plotly.offline import iplot, init_notebook_mode
          import cufflinks
          cufflinks.go_offline(connected=True)
          init_notebook_mode(connected=True)
```

Εικόνα 3 Εγκατάσταση Βιβλιοθηκών

Το κύριο dataset μετονομάζεται σε “sales_data” για διευκόλυνση. Στη συνέχεια φορτώνεται στο jupyter notebook με την χρήση .read_csv() συνάρτησης, η οποία περιέχεται στο Pandas πακέτο. Αφού η συνάρτηση .read_csv() διαβάζει τα δεδομένα, επιστρέφεται το ακόλουθο Data Frame .

```
In [125]: sales_data=pd.read_csv("summer_products_with_rating_and_performance_2020_08.csv")
          sales_data
```

Out[125]:

	title	title_orig	price	retail_price	currency_buyer	units_sold	uses_ad_boosts	rating	rating_count	rating_five_count	rating_four_count	rating_th
0	2020 Summer Vintage Flamingo Print Pajamas Se...	2020 Summer Vintage Flamingo Print Pajamas Se...	16.00	14	EUR	100	0	3.76	54	26.0	8.0	
1	SSHOUSE Summer Casual Sleeveless Soirée Party ...	Women's Casual Summer Sleeveless Sexy Mini Dress	8.00	22	EUR	20000	1	3.45	6135	2269.0	1027.0	
2	2020 Nouvelle Arrivée Femmes Printemps et Été ...	2020 New Arrival Women Spring and Summer Beach...	8.00	43	EUR	100	0	3.57	14	5.0	4.0	
	Hot Summer	Hot Summer Cool T										

Εικόνα 4 Σύνολο δεδομένων sales_data

Με τη χρήση της συνάρτησης .head() εμφανίζονται οι πρώτες 5 παρατηρήσεις του πίνακα.

```
In [126]: pd.set_option('display.max_columns', None)
          sales_data.head()
          # Δείχω όλες τις στήλες
```

Out[126]:

	title	title_orig	price	retail_price	currency_buyer	units_sold	uses_ad_boosts	rating	rating_count	rating_five_count	rating_four_count	rating_three
0	2020 Summer Vintage Flamingo Print Pajamas Se...	2020 Summer Vintage Flamingo Print Pajamas Se...	16.00	14	EUR	100	0	3.76	54	26.0	8.0	
1	SSHOUSE Summer Casual Sleeveless Soirée Party ...	Women's Casual Summer Sleeveless Sexy Mini Dress	8.00	22	EUR	20000	1	3.45	6135	2269.0	1027.0	
2	2020 Nouvelle Arrivée Femmes Printemps et Été ...	2020 New Arrival Women Spring and Summer Beach...	8.00	43	EUR	100	0	3.57	14	5.0	4.0	
3	Hot Summer Cool T-shirt pour les	Hot Summer Cool T Shirt for Women	8.00	8	EUR	5000	1	4.03	579	295.0	119.0	

Εικόνα 5 Σύνολο Δεδομένων sales_data

3.1.1 Κατανόηση τί αντιπροσωπεύει κάθε στήλη

Οι πληροφορίες σχετικά με το τί αντιπροσωπεύει η κάθε στήλη του συνόλου δεδομένων βρίσκονται στο Kaggle πληκτρολογώντας στην αναζήτηση " Sales of summer clothes in E-commerce Wish ". Παρακάτω αναφέρονται οι ονομασίες και οι πληροφορίες από τα features των τριών πινάκων.

Πίνακας " summer_products_with rating_and_perfomance_2020-08.

- **"title"**: Τοπικός τίτλος προϊόντος
- **"origin_title"**: Τίτλος προϊόντος στα αγγλικά
- **"price"** : Τιμή πώλησης προϊόντος
- **"retail_price"** : Λιανική τιμή προϊόντος για άλλα καταστήματα. Χρήση από πωλητή για να δείξει μια κανονική τιμή
- **"currency_buyer"** : είδος νομίσματος
- **"units_sold"** : Αριθμός πωληθέντων προϊόντων
- **"uses_add_boosts"** : Αν ο πωλητής πληρώνει για να ενισχύσει το προϊόν του μέσα στη πλατφόρμα
- **"rating"** : Μέση βαθμολογία προϊόντος
- **"rating_count"** : Συνολικός αριθμός αξιολογήσεων
- **"rating_five_count, rating_four_count, rating_three_count, rating_two_count, rating_one_count":** Αριθμός αξιολογήσεων 5,4,3,2,1 αντίστοιχα αστεριών
- **"badges_count"** : Αριθμός σημάτων που διαθέτει το προϊόν/πωλητής
- **"badges_local_count"** : ένα σήμα που δηλώνει αν το προϊόν είναι τοπικό
- **"badges_product_quality"** : Δόθηκε σήμα όταν οι αγοραστές έδωσαν σταθερά καλές αξιολογήσεις
- **"badges_fast_shipping"** : Σήμα που απονέμεται όταν η παραγγελία φτάνει γρήγορα
- **" tags"** : ετικέτες που ορίζονται από τον πωλητή
- **"product_color":** χρώμα προϊόντος
- **"product_variation_size_id":** μεγέθη προϊόντος
- **"product_variation_inventory":** αποθέματα πωλητή (Μέγιστη ποσότητα 50)
- **"shipping_option_name":** όνομα αποστολής
- **"shipping_option_price":** τιμή αποστολής
- **"shipping_is_express":** 1-->express, 0-->not
- **"countries_shipped_to"** : Αριθμός χωρών που αποστέλλεται το προϊόν. Οι πωλητές μπορούν να περιορίσουν πού στέλνουν ένα προϊόν

- **"inventory_total"** : Συνολικό απόθεμα για όλες τις παραλλαγές προϊόντος
- **"has_urgency_banner"** : αν υπάρχει επείγον πανό ή όχι (1,0)
- **"urgency_text"** : banner κειμένου που εμφανίζεται σε ορισμένα προϊόντα στα αποτελέσματα αναζήτησης
- **"origin_country"** : χώρα προέλευσης προϊόντος
- **"merchant_title"** : όνομα εμπόρου (εμφανίζεται στην διεπαφή χρήστη ως όνομα καταστήματος πωλητή)
- **"merchant_name"** : Κανονικό όνομα εμπόρου
- **"merchant_info_subtitle"** : Το κείμενο όπως φαίνεται στην ενότητα πληροφοριών πωλητή στο χρήστη. Δίνει μια επισκόπηση των στατιστικών του πωλητή στον χρήστη. Συνήθως αποτελείται από "%< positive feedback>"
- **"merchant_rating-count"** : Αριθμός αξιολογήσεων του πωλητή
- **"merchant_rating":** Βαθμολογία εμπόρου
- **"merchant_id":** Το id εμπόρου
- **"merchant_has_profile_picture":** boolean τιμή που αναφέρει αν υπάρχει url "merchant_profile_picture"
- "**merchant_profile_picture"** : Εικόνα προφίλ του πωλητή σε μορφή url
- **"product_url":** Το url προϊόντος
- **"product_profile_picture":** Η φωτογραφία του προϊόντος
- **"product_id"** : Το id προϊόντος
- **"theme"** : Ο όρος αναζήτησης που χρησιμοποιείται στον ιστότοπο για την λήψη αυτών των αποτελεσμάτων ("summer")
- **"crawl_month"** : Μόνο για πληροφορίες

Πίνακας "unique_categories"

- **"tag" :** κατηγορίες ετικετών που βρέθηκαν στο κύριο αρχείο του συνόλου δεδομένων (" summer_products_with rating_and_perfomance_2020-08") κάτω από τη στήλη "tag".

Πίνακας "unique_categories.sorted_by_count"

- **"count" :** Ο αριθμός εμφάνισης της λέξης-κλειδί

- **"keyword" :** Επισήμανση λέξης-κλειδί που εμφανίζεται στη στήλη " tag " του κύριου αρχείου.

3.1.2 Λήψη πληροφοριών μέσω περιγραφικής στατιστικής

Το κομμάτι αυτό κρίνεται απαραίτητο για την καλύτερη κατανόηση των μεταβλητών με σκοπό την σωστή εκτέλεση της μετέπειτα προ επεξεργασίας των δεδομένων και της προετοιμασίας του τελικού dataset για την εκπαίδευση των μοντέλων.

Αρχικά γίνεται μία διερεύνηση των αριθμητικών τιμών σχετικά με το τί τύπο έχουν, αν υπάρχουν ελλιπείς τιμές και πόσες είναι αυτές, τί συσχέτιση έχουν με την εξαρτημένη μεταβλητή πρόβλεψης (“unis_sold”) , ποια είναι η μέση, μέγιστη και ελάχιστη τιμή, η διάμεσος, και ποια είναι η συχνότερη τιμή που εμφανίζεται. Το ίδιο γίνεται για τις κατηγορικές μεταβλητές . Παρακάτω εμφανίζονται τα αποτελέσματα των τιμών.

ColumnName	DataType	HasMissing	NumberOfMissingCells	CorrelationWithTarget	Mean	Median	Mode	MinValue	MaxValue
units_sold	int64	False	0	1.000000	4339.005086	1000.00000	100	1.000000	100000.0
rating_count	int64	False	0	0.899464	889.659250	150.00000	0	0.000000	20744.0
rating_three_count	float64	True	45	0.894243	134.549738	24.00000	0	0.000000	3658.0
rating_four_count	float64	True	45	0.891116	179.599476	31.50000	0	0.000000	4152.0
rating_five_count	float64	True	45	0.876231	442.263743	79.00000	1	0.000000	11548.0
rating_two_count	float64	True	45	0.866685	63.711387	11.00000	0	0.000000	2003.0
rating_one_count	float64	True	45	0.832816	95.735602	20.00000	0	0.000000	2789.0
merchant_rating_count	int64	False	0	0.272897	26495.832804	7936.00000	32168	0.000000	2174765.0
merchant_has_profile_picture	binary	False	0	0.143529	NaN	NaN	0	NaN	NaN
product_variation_inventory	int64	False	0	0.133846	33.081373	50.00000	50	1.000000	50.0
merchant_rating	float64	False	0	0.122504	4.032345	4.04065	3.88454	2.333333	5.0
badge_product_quality	binary	False	0	0.063187	NaN	NaN	0	NaN	NaN
badges_count	int64	False	0	0.045402	0.105531	0.00000	0	0.000000	3.0
rating	float64	False	0	0.039478	3.820896	3.85000	5	1.000000	5.0
retail_price	int64	False	0	0.012638	23.288620	10.00000	7	1.000000	252.0
inventory_total	int64	False	0	0.005608	49.821360	50.00000	50	1.000000	50.0
badge_fast_shipping	binary	False	0	-0.000898	NaN	NaN	0	NaN	NaN
badge_local_product	binary	False	0	-0.007544	NaN	NaN	0	NaN	NaN
shipping_is_express	binary	False	0	-0.008308	NaN	NaN	0	NaN	NaN
countries_shipped_to	int64	False	0	-0.013553	40.456453	40.00000	41	6.000000	140.0
uses_ad_boosts	binary	False	0	-0.016055	NaN	NaN	0	NaN	NaN
price	float64	False	0	-0.024815	8.325372	8.00000	8	1.000000	49.0
shipping_option_price	int64	False	0	-0.030987	2.345200	2.00000	2	1.000000	12.0
has_urgency_banner	binary	True	1100	NaN	NaN	NaN	1	NaN	NaN

Εικόνα 6 Στατιστικά στοιχεία αριθμητικών μεταβλητών

ColumnName	DataType	HasMissing	NumberOfMissingCells	Mode
title	object	False	0	Nouvelle mode d'été femmes robe décontractée c...
title_orig	object	False	0	New Fashion Summer Women Casual Dress Round Ne...
currency_buyer	object	False	0	EUR
tags	object	False	0	Summer,Fashion,Necks,Skirts,Dress,Loose,Women'...
product_color	object	True	41	black
product_variation_size_id	object	True	14	S
shipping_option_name	object	False	0	Livraison standard
urgency_text	object	True	1100	Quantité limitée !
origin_country	object	True	17	CN
merchant_title	object	False	0	guangzhouweishiweifushiyouxiangongsi
merchant_name	object	True	4	广州唯适唯服饰有限公司
merchant_info_subtitle	object	True	1	83 % avis positifs (32,168 notes)
merchant_id	object	False	0	558c2cdc89d53c4005ea2920
merchant_profile_picture	object	True	1347	https://s3-us-west-1.amazonaws.com/sweeper-pro...
product_url	object	False	0	https://www.wish.com/c/5c80e8a150c63d28c67b8f14
product_picture	object	False	0	https://contestimg.wish.com/api/webimage/5c80e...
product_id	object	False	0	5c80e8a150c63d28c67b8f14
theme	object	False	0	summer
crawl_month	object	False	0	2020-08

Εικόνα 7 Στατιστικά στοιχεία κατηγορικών μεταβλητών

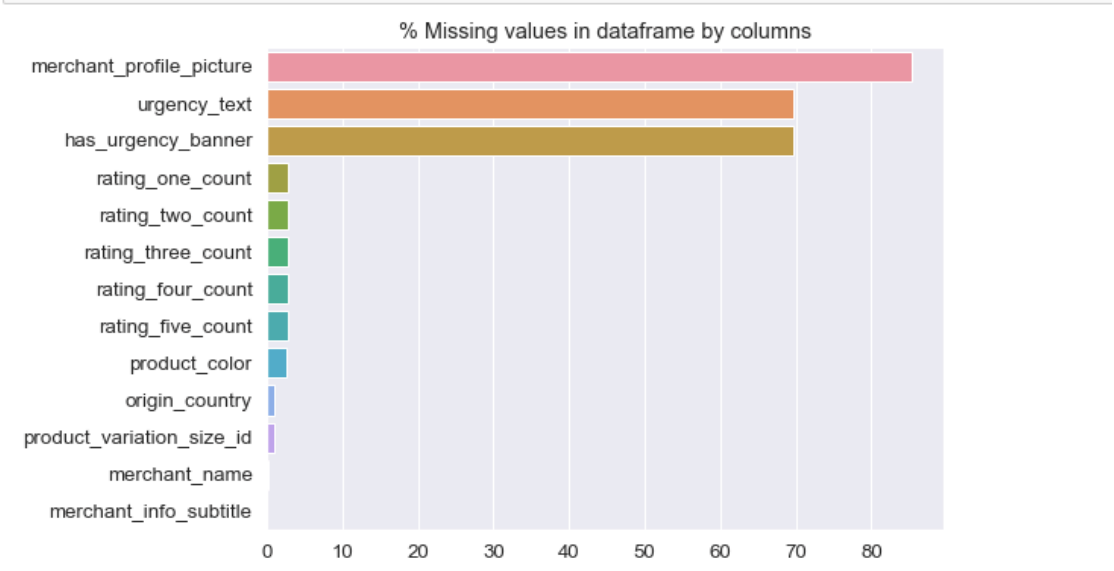
Επίσης θεωρείται σημαντικό να ελεγχθούν οι τιμές της στήλης “units_sold” αφού είναι και η εξαρτημένη μεταβλητή πρόβλεψης. Παρατηρείται ότι οι τιμές του “units_sold” είναι διακριτές και όχι συνεχείς και μάλιστα είναι εύρη. Συγκεκριμένα παίρνει τις τιμές 6,2,3,7,1,8,100000,50000,10,50,20000,10000,5000,1000,100 όπως φαίνεται και στην παρακάτω εικόνα. Με τη συνάρτηση .value_counts () υπολογίζεται πόσες φορές εμφανίζονται οι τιμές μιας μεταβλητής και είναι από το πακέτο Pandas.

```
In [134]: sales_data["units_sold"].value_counts()

Out[134]: 100       493
          1000      403
          5000      216
          10000     176
          20000     103
          50         68
          10         44
          50000      17
          100000      6
          8           3
          1           3
          7           2
          3           2
          2           2
          6           1
          Name: units_sold, dtype: int64
```

Εικόνα 8 Συχνότητα εμφάνισης μοναδικών τιμών εξαρτημένης μεταβλητής "units_sold"

Επειδή το πλήθος των ελλιπών τιμών είναι αρκετό σε πολλές στήλες, παρακάτω παρουσιάζεται ένα barplot για να υπάρχει μία καλύτερη εικόνα ποσοστιαία των κενών τιμών.



Εικόνα 9 Barplot ελλιπών τιμών

Τέλος ελέγχονται πόσες μοναδικές τιμές υπάρχουν σε κάθε στήλη. Αυτό γίνεται με τη βοήθεια της συνάρτησης .unique() από το πακέτο Pandas.

```
column                          unique values
title                           1201
title_orig                      1203
price                           127
retail_price                    104
currency_buyer                  1
        : ['EUR']
units_sold                      15
        : [   100  20000   5000     10  50000   1000  10000 100000     50      1
      7      2      3      8      6]
uses_ad_boosts                  2
        : [0 1]
rating                          192
rating_count                    761
rating_five_count               606
rating_four_count               441
rating_three_count              385
rating_two_count                263
rating_one_count                331
badges_count                    4
        : [0 1 2 3]
badge_local_product             2
        : [0 1]
badge_product_quality           2
        : [0 1]
badge_fast_shipping             2
        : [0 1]
tags                            1230
product_color                   102
product_variation_size_id       107
```

Εικόνα 10 Πλήθος μοναδικών τιμών των μεταβλητών

```
product_variation_inventory     48
shipping_option_name            15
        : ['Livraison standard' 'Envio Padrão' 'Standart Gönderi' 'Standardversand'
 'Envío normal' 'Standardowa wysyłka' 'الشحن القياسي' 'Expediere Standard'
 'การส่งสินค้ามาตรฐาน' 'Standard Shipping' 'ការដឹកជញ្ជូនតាមស្តង់ដារ'
 'Livraison Express' 'Стандартная доставка' 'Spedizione standard'
 'Ekspresowa wysyłka']
shipping_option_price           8
        : [ 4  2  3  1  6  5  7 12]
shipping_is_express             2
        : [0 1]
countries_shipped_to            94
inventory_total                 10
        : [50 40 36  1 30  2  9 24 37 38]
has_urgency_banner              2
        : [ 1. nan]
urgency_text                    3
        : ['Quantité limitée !' nan 'Réduction sur les achats en gros']
origin_country                  7
        : ['CN' 'US' nan 'VE' 'AT' 'SG' 'GB']
merchant_title                  958
merchant_name                   958
merchant_info_subtitle          1059
merchant_rating_count           917
merchant_rating                 952
merchant_id                     958
merchant_has_profile_picture    2
        : [0 1]
merchant_profile_picture        126
 product_url                    1341
 product_picture                1341
 product_id                     1341
 theme                          1
        : ['summer']
 crawl_month                    1
        : ['2020-08']
```

Εικόνα 11 Πλήθος μοναδικών τιμών των μεταβλητών

Τα παραπάνω δείχνουν ότι :

- 1201 μοναδικά προϊόντα από τα 1539 που είναι συνολικά οι εγγραφές αναφέρονται σε 958 μοναδικούς πωλητές.
- Μόνο 2 είδη urgency_text υπάρχουν που εμφανίζονται στα προϊόντα.
- Τα προϊόντα είναι διαθέσιμα από 6 διαφορετικές χώρες.
- 45 προϊόντα δεν έχουν αξιολογήσεις
- Οι περισσότεροι έμποροι δεν έχουν φωτογραφία προφίλ. Συγκεκριμένα 1413 από τους 1539 που είναι συνολικά.

3.1.3 Έλεγχος για διπλότυπες εγγραφές

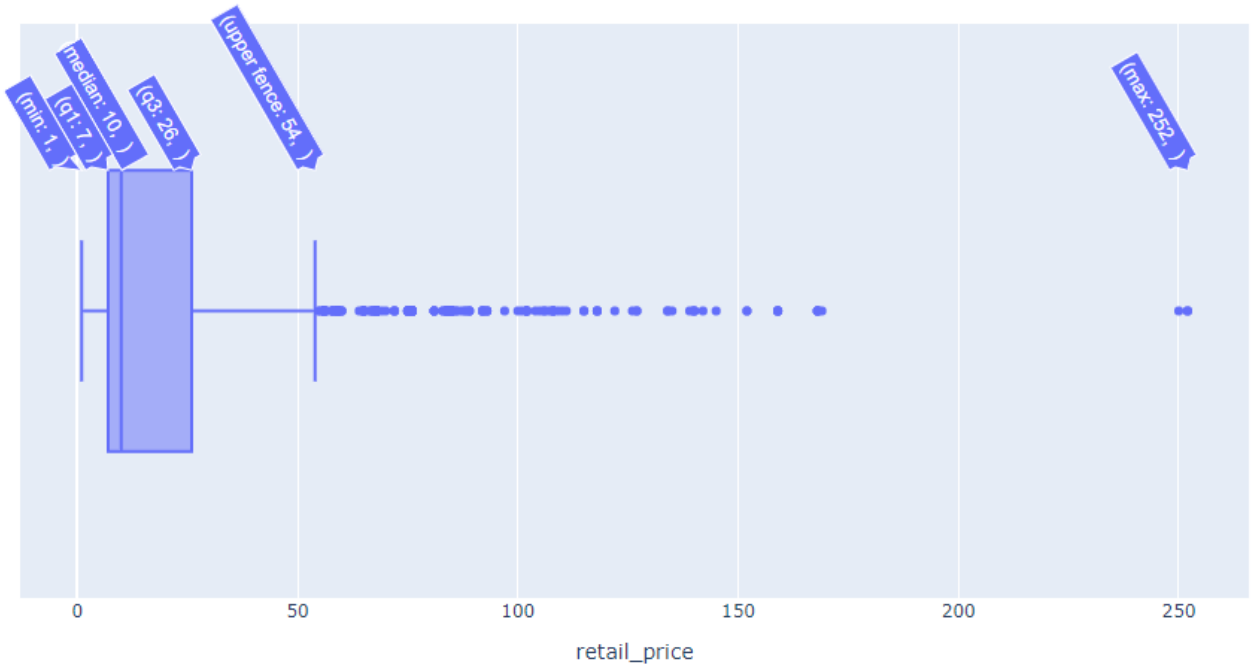
Στο σύνολο δεδομένων υπάρχουν 1573 εγγραφές. Μετά από διαγραφή διπλότυπων εγγραφών οι παρατηρήσεις φτάνουν στις 1539. Ωστόσο με τον έλεγχο του πλήθους διπλοεγγραφών της στήλης “ product_id ” παρατηρείται ότι υπάρχουν επιπλέον 198 εγγραφές. Αυτό συμβαίνει γιατί το ίδιο προϊόν μπορεί να πωληθεί από διαφορετικούς εμπόρους σε διαφορετική τιμή. Ως εκ τούτου αυτό φαίνεται σε μία σαφή ένδειξη ότι το “merchant_id” επηρεάζει την τιμή του προϊόντος.

3.1.4 Έλεγχος για ακραίες τιμές

Σε 2 στήλες του dataset παρατηρούνται κάποιες ακραίες τιμές. Αυτό γίνεται εμφανές με τα ακόλουθα boxplot.

```
In [140]: import plotly.express as px
          df =sales_data
          fig = px.box(df, x="retail_price")
          fig.show()
```

Εικόνα 12 Κώδικας Γραφήματος Boxplot "retail_price"

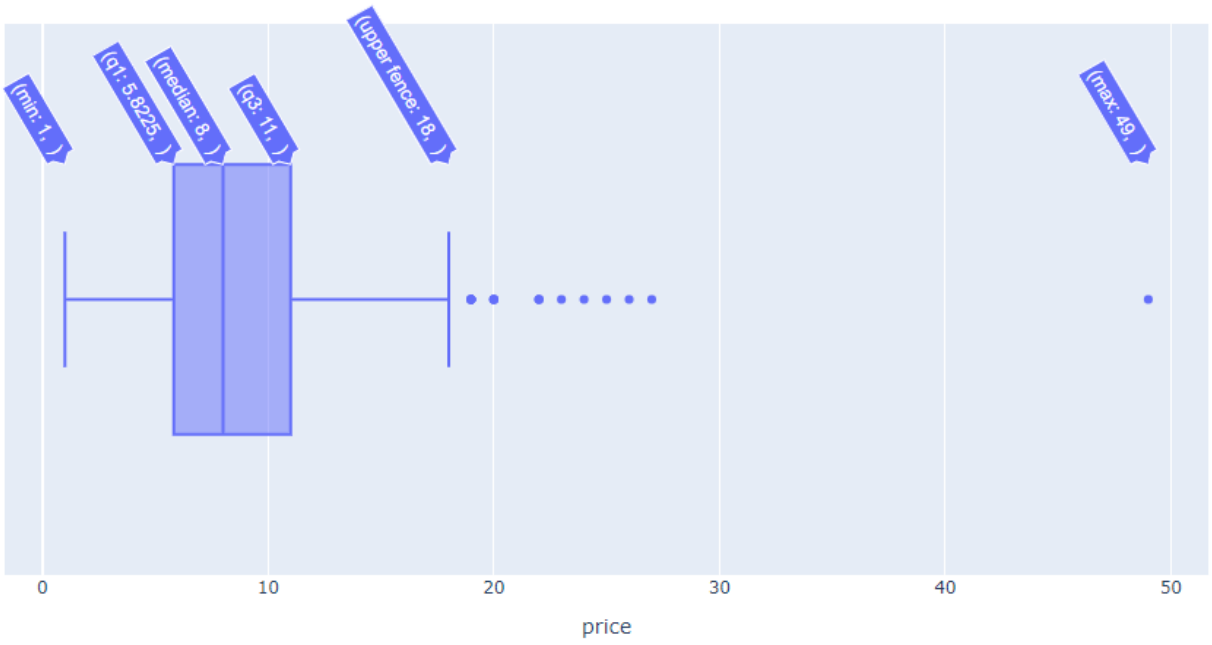


Εικόνα 13 Boxplot "retail_price"

Από το παραπάνω διάγραμμα παρατηρείται ότι τα δεδομένα είναι πιο απλωμένα και υπάρχουν περισσότερες ακραίες τιμές. Υπάρχει μεγάλη διαφορά του άνω φράχτη και του μέγιστου σημείου που είναι 252.

```
In [141]: import plotly.express as px
          df =sales_data
          fig = px.box(df, x="price")
          fig.show()
```

Εικόνα 14 Κώδικας Boxplot "price"



Εικόνα 15 Boxplot "price"

Από το παραπάνω διάγραμμα για την μεταβλητή “price” παρατηρείται ότι ο ανώτερος φράχτης είναι 18. Άρα τα περισσότερα δεδομένα έχουν τιμή μικρότερη από 18. Υπάρχει ένα στοιχείο με τιμή 49. Οι τιμές είναι πιο συγκεντρωμένες.

3.1.5 Οπτικοποιήσεις

Σε αυτό το κομμάτι, μετά τη μελέτη του συνόλου δεδομένων με τις πληροφορίες που παρέχουν οι μεταβλητές και την περιγραφική στατιστική , αναπτύσσονται ερωτήματα όπου οι απαντήσεις προκύπτουν από διάφορα διαγράμματα. Για λόγους διευκόλυνσης παρουσιάζεται ένα μικρό κομμάτι του κώδικά Python κατά την εκτέλεση γραφημάτων ενώ τα υπόλοιπα βρίσκονται στο τέλος της εργασίας ως παραρτήματα. Σε αυτό το σημείο θα γίνει εμβάθυνση στο προσδιορισμό βασικών σχέσεων σχετικά με την επιτυχία πώλησης ενός προϊόντος (“ units_sold”). Η στήλη “ units_sold ” περιγράφει τον αριθμό των πωληθέντων μονάδων του κάθε προϊόντος που υπάρχει στον ιστότοπο Wish.com. Θεωρείται ότι όσο μεγαλύτερος είναι αυτός ο αριθμός τόσο μεγαλύτερη επιτυχία πώλησης έχουν τα προϊόντα. Επομένως οι οπτικοποιήσεις θα γίνουν με βάση τους παράγοντες που επηρεάζουν τις πωλήσεις των προϊόντων.

Επειδή κάτω από 10 πωλήσεις υπάρχουν λίγες ποσότητες προϊόντων, όλες οι πωλήσεις κάτω των 10 συγκεντρώνονται στη τιμή 10 με την παρακάτω εντολή.

```
sales_data['units_sold'] = [10 if x < 10 else x for x in sales_data['units_sold']]
```

Εικόνα 16 Κώδικας επεξεργασίας μεταβλητής "units_sold"

3.1.5.1 Επίδραση τιμής πώλησης – λιανικής τιμής και η διαφορά των δύο αυτών τιμών στις πωλήσεις προϊόντων

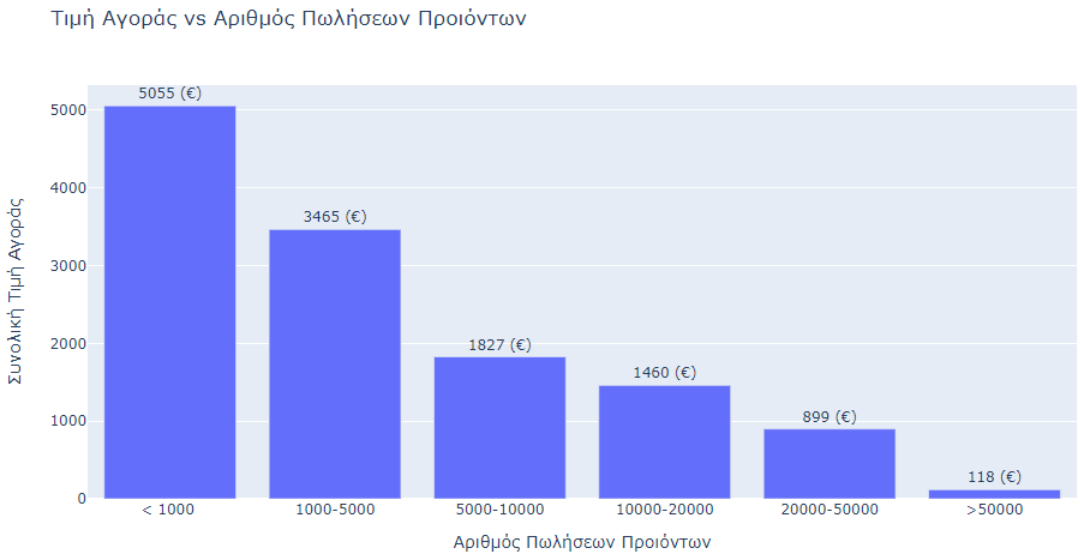
Αρχικά δημιουργείται ένα Violin διάγραμμα από το πακέτο Plotly Express το οποίο είναι παρόμοιο με το γνωστό boxplot. Η διαφορά τους είναι ότι στο πρώτο απεικονίζεται η πλήρη κατανομή των δεδομένων ενώ στο δεύτερο εμφανίζονται μόνο συνοπτικά στατιστικά στοιχεία όπως η μέση τιμή, η διάμεσος και τα δια τεταρτημόρια. (‘Violin Plots in Python’, no date). Σε αυτό το διάγραμμα γίνεται σύγκριση μεταξύ “price” που είναι η τελική τιμή αγοράς και “retail_price” που είναι η γενική τιμή αγοράς σε άλλα καταστήματα και εμφανίζεται σαν αρχική τιμή του προϊόντος.

Εικόνα 17 Violin Plot Σύγκρισης "price" και "retail_price"

Η διαφορά μεταξύ price και retail_price είναι αρκετά μεγάλη. Οι τιμές του price είναι πιο μαζεμένες ενώ του retail_price έχουν περισσότερες ακραίες τιμές. Αυτό θα μπορούσε να θεωρείται μια καλή στρατηγική πωλήσεων.

Στην επόμενη απεικόνιση γίνεται η χρήση του γνωστού Barplot που χρησιμεύει για οπτικοποίηση κατηγορικών δεδομένων . Γίνεται κλήση της βιβλιοθήκης plotly με το πακέτο graph_objects το οποίο μας επιτρέπει την αναπαράσταση πιο περίπλοκων

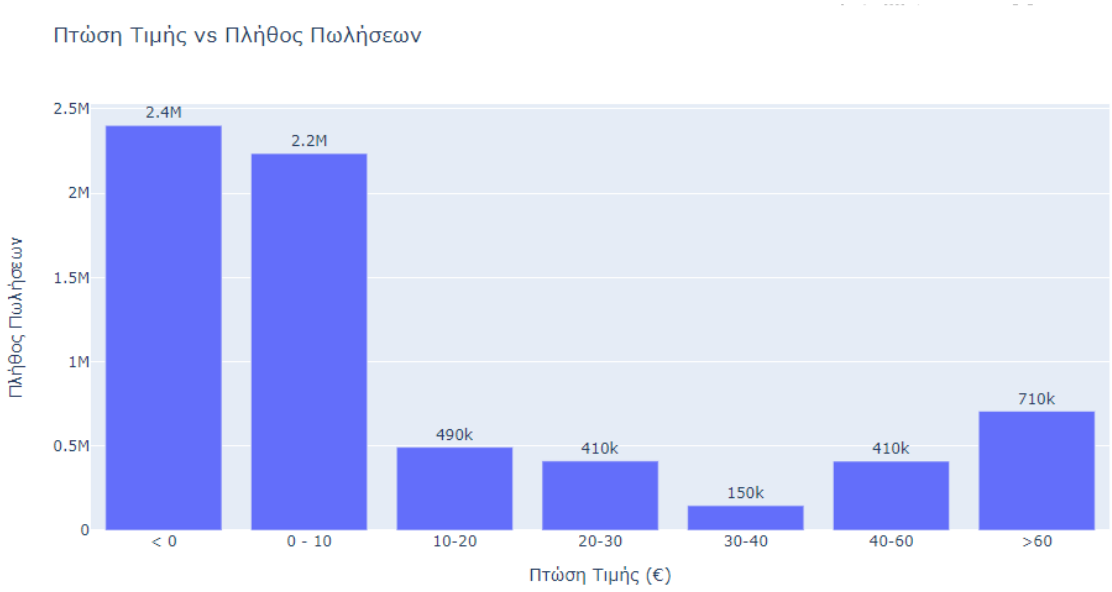
γραφημάτων σε σχέση με άλλες βιβλιοθήκες όπως Matplotlib. (Sblendorio, 2019). Σε αυτό το διάγραμμα το “ units_sold” μετατρέπεται σε συγκεκριμένο εύρος τιμών και για κάθε εύρος παρουσιάζονται αθροιστικά οι τιμές αγοράς των προϊόντων.



Εικόνα 18 Barplot Τιμή Αγοράς - Αριθμός πωλήσεων προϊόντων

Παρατηρείται ότι όσο αυξάνεται ο αριθμός πωλήσεων προϊόντων τόσο μειώνεται η τιμή αγοράς αυτών. Αυτό ίσως δείχνει ότι οι καταναλωτές προτιμούν χαμηλές τιμές πώλησης για την αγορά των προϊόντων.

Τέλος, διερευνάται κατά πόσο η πτώση τιμών επηρεάζει θετικά τις πωλήσεις. Για την πτώση τιμής αφαιρείται η αρχική τιμή πώλησης με την τελική και τοποθετείται σε εύρη. Για κάθε εύρος υπολογίζεται αθροιστικά το πλήθος των πωληθέντων μονάδων των προϊόντων .



Εικόνα 19 Barplot Πτώση Τιμών - Πλήθος Πωλήσεων

Παρατηρείται ότι οι μεγάλες πτώσεις τιμών δεν οδηγούν απαραίτητα στην επιτυχία πωλήσεων των προϊόντων. Αυτό μπορεί να συμβαίνει ίσως λόγω ποιότητας, κακής αξιολόγησης ή άλλων παραγόντων που θα αναλυθούν παρακάτω.

3.1.5.2 Αξιολογήσεις Προϊόντων

Σε αυτό το πεδίο γίνεται μία διερεύνηση για το πώς οι αξιολογήσεις των πελατών επηρεάζουν το πλήθος πωλήσεων των προϊόντων του ιστότοπου Wish.com ή αλλιώς την επιτυχία πωλήσεων αυτών.

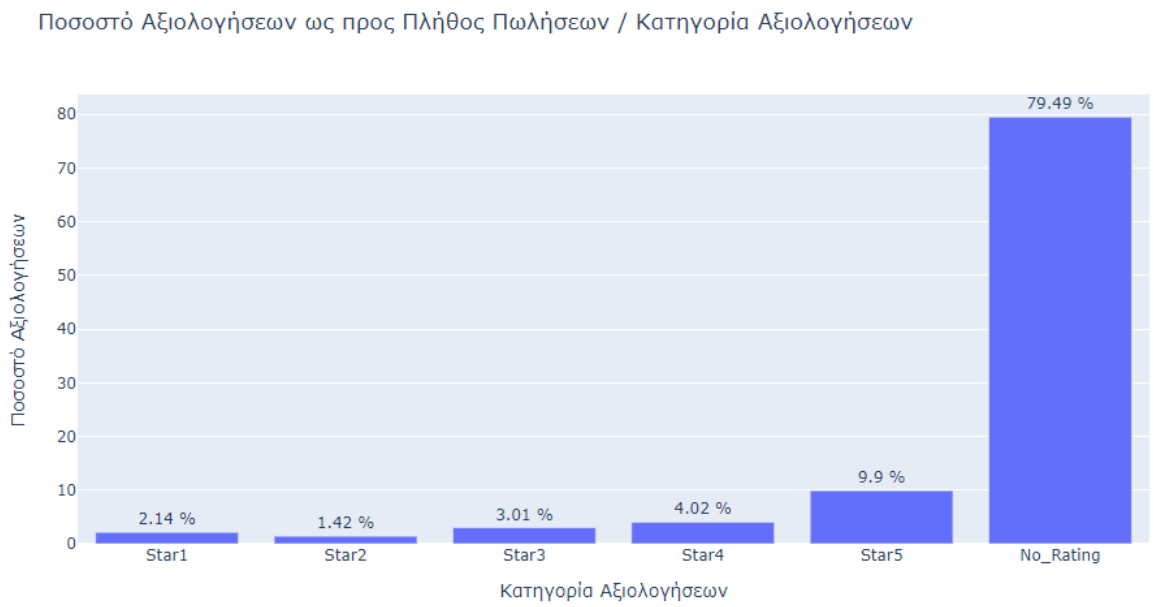
Αρχικά εφαρμόζεται μία διαφορετική οπτική θεωρώντας ότι ο αγοραστής αγοράζει κάθε φορά 1 ποσότητα από κάθε προϊόν. Αυτή η υπόθεση επιλέγεται έτσι ώστε να εξαντληθεί κάθε εκδοχή σχετικά με την επίδραση των αξιολογήσεων στις πωλήσεις αφού από την αρχή, το σύνολο δεδομένων δεν εξακριβώνει αν από τις 100 ποσότητες units_sold ενός συγκεκριμένου προϊόντος ο πελάτης αγοράζει μία κάθε φορά. Επομένως σε αυτή την εκδοχή παρουσιάζεται μία σειρά του πίνακα για καλύτερη κατανόηση, θεωρώντας ότι από τους 100 πελάτες που αγοράζουν αυτό το είδος οι 54 έκαναν αξιολόγηση.

Out[125]:

title	title_orig	price	retail_price	currency_buyer	units_sold	uses_ad_boosts	rating	rating_count	rating_five_count	rating_four_count	rating_three_count
2020 Summer Vintage Flamingo Print Pajamas Se...	2020 Summer Vintage Flamingo Print Pajamas Se...	16.00	14	EUR	100	0	3.76	54	26.0	8.0	10.0

Εικόνα 20 Πίνακας "sales_data" με "units_sold"=100

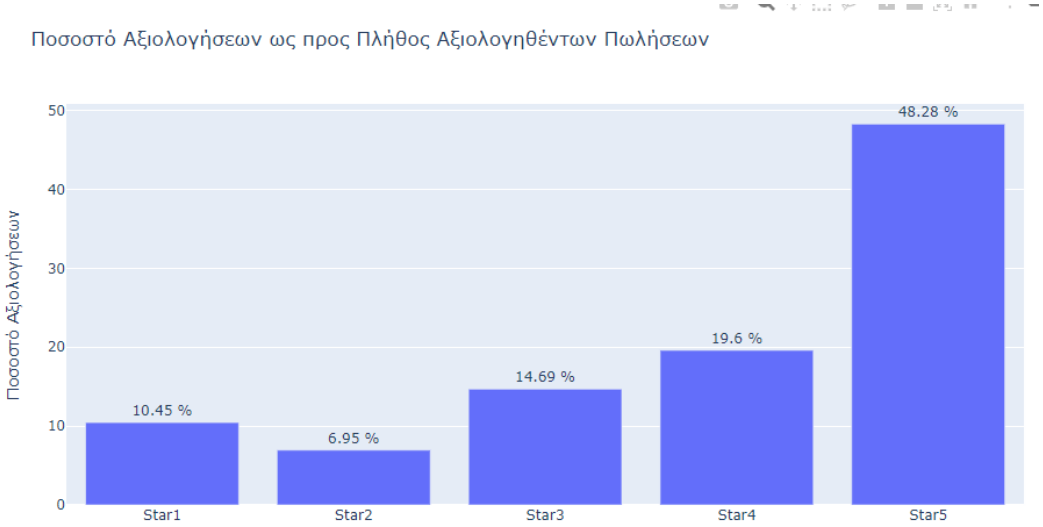
Έτσι δημιουργείται ένα barplot με τον ίδιο τρόπο όπως παραπάνω και υπολογίζεται το ποσοστό αξιολογήσεων ως προς το πλήθος πωλήσεων ανά κατηγορία αξιολογήσεων.



Εικόνα 21 Barplot Ποσοστό αξιολογήσεων ως προς πλήθος πωλήσεων ανά κατηγορία αξιολογήσεων

Από το παραπάνω διάγραμμα φαίνεται ότι το 79,49% των πωλήσεων δεν έχει αξιολογηθεί. Επομένως με την παραπάνω υπόθεση θεωρείται ότι οι αξιολογήσεις δεν λαμβάνουν τόση σημασία στην επιτυχία πωλήσεων.

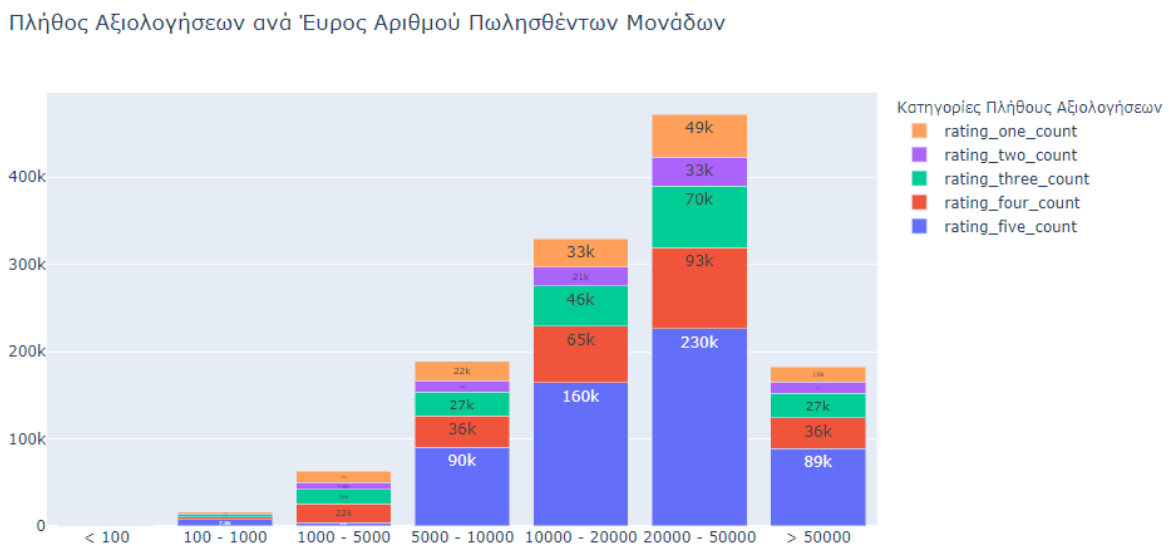
Ωστόσο αν θεωρηθεί ότι ένας πελάτης μπορεί να έχει κάνει περισσότερες από μία αγορές για κάθε προϊόν δηλαδή να έχει προσθέσει στο καλάθι του πάνω από ένα τεμάχιο τα συμπεράσματα για την επίδραση των αξιολογήσεων στην επιτυχία των πωλήσεων αλλάζουν. Αρχικά σημαντικό είναι να φανεί η αναλογία των αξιολογήσεων από 1 έως 5 αστέρια στο σύνολο δεδομένων. Η απεικόνιση γίνεται σε ποσοστά για καλύτερη εικόνα.



Εικόνα 22 Barplot Ποσοστό αξιολογήσεων ως προς πλήθος αξιολογημένων πωλήσεων

Σε πρώτη εικόνα φαίνεται ότι από τα προϊόντα που έχουν αξιολογηθεί, το μεγαλύτερο ποσοστό αξιολογήσεων είναι με 5 αστέρια που σημαίνει ότι στο μεγαλύτερο ποσοστό οι πελάτες είναι ικανοποιημένοι.

Για καλύτερη εικόνα αθροίζεται το πλήθος των αξιολογήσεων από 1 έως 5 αστέρια για κάθε εύρος πωληθέντων μονάδων προϊόντων. Σημαντικό είναι να τονισθεί ότι σε αυτή την ανάλυση δεν διερευνάται ποια προϊόντα έχουν επιτυχία αλλά μια συνολική αποτίμηση για το αν το Wish.com είχε επιτυχία στις πωλήσεις του τη χρονική περίοδο Ιουλίου 2020. Γι’ αυτό το λόγο υπολογίζονται αθροιστικά αποτελέσματα για κάθε εύρος των units_sold.

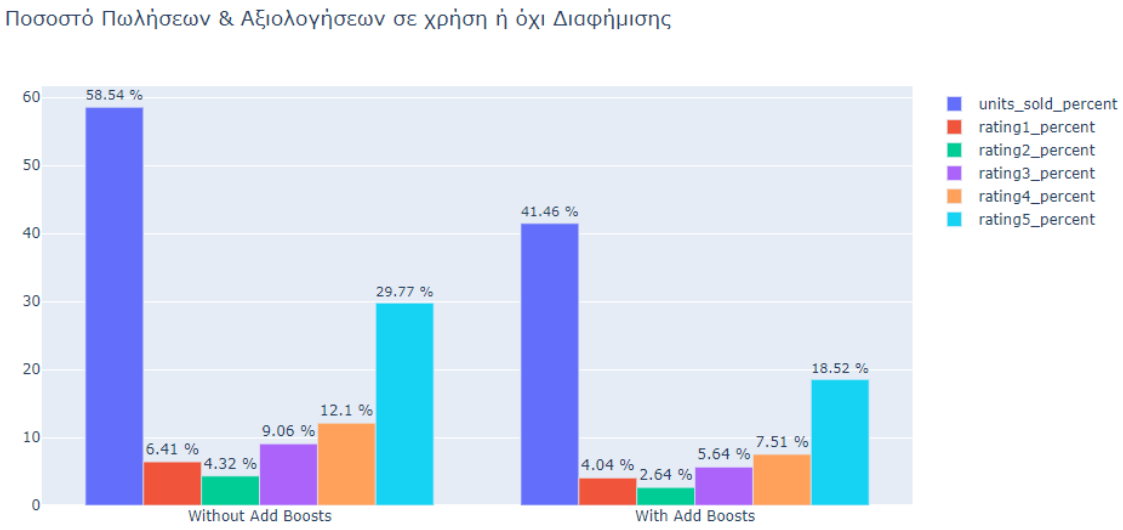


Εικόνα 23 Barplot Πλήθος αξιολογήσεων ανά εύρος αριθμού πωληθέντων μονάδων

Από το παραπάνω barplot, υπάρχει συσχέτιση μεταξύ των πωληθέντων προϊόντων και κάθε επιπέδου βαθμολογίας. Καθώς το εύρος του αριθμού των units_sold αυξάνεται, αυξάνεται επίσης το ποσοστό κάθε επιπέδου αξιολόγησης, πράγμα που σημαίνει ότι ένα προϊόν που έχει καλή βαθμολογία περισσότερο από την βαθμολογία 4 πωλείται καλά. Είναι πιθανόν ότι ο λόγος όπου το εύρος πωλήσεων άνω των 50K έχει πολύ λιγότερες τιμές σε σύγκριση με άλλες κατηγορίες εύρους είναι γιατί ενδέχεται να μην έχει γίνει αξιολόγηση από όλους τους πελάτες. Φιλτράροντας το σύνολο δεδομένων να εμφανίζει μόνο units_sold>50.000 προκύπτουν μόνο έξι προϊόντα. Σε αυτά τα κομμάτια οι τιμές πώλησης είναι πολύ μικρές μεταξύ 5-8 ευρώ σε σχέση με τη μεγαλύτερη τιμή πώλησης που είναι 49 όπως είχε φανεί στο βήμα στατιστικής περιγραφής. Επίσης τα έξοδα αποστολής είναι μεταξύ 1-2 ευρώ. Επομένως ο λόγος που υπάρχουν τόσο μεγάλες πωλήσεις εδώ να ευθύνεται στις χαμηλές τιμές αγοράς και αποστολής γι' αυτά τα έξι κομμάτια και οι πελάτες δεν δίνουν τόσο μεγάλη σημασία στις αξιολογήσεις για την ποιότητα. Ωστόσο, δεδομένου ότι η βαθμολογία 4 και 5 παίρνει περισσότερο από το μισό του γραφήματος, μπορεί κανείς να καταλήξει στο συμπέρασμα ότι όσο υψηλότερη είναι η βαθμολογία, τόσο καλύτερη είναι η πώληση ενός προϊόντος. Στα κομμάτια όπου οι πωλήσεις είναι κάτω των 1000 μονάδων, οι αξιολογήσεις είναι μηδαμινές έως και ανύπαρκτες. Αυτό δείχνει ότι η ποιότητα των προϊόντων ίσως να μην είναι καλή γι' αυτό και δεν αξιολογούνται αρκετά. Άρα οι πωλήσεις θεωρούνται σε αυτά να μην έχουν τόσο μεγάλη επιτυχία.

3.1.5.3 *Επίδραση διαφημίσεων στις πωλήσεις και αξιολογήσεις προϊόντων*

Σε αυτό το ερώτημα χρησιμοποιείται το γράφημα barplot. Η στήλη "uses_ad_boosts" είναι Boolean μεταβλητή. Δηλαδή παίρνει δύο τιμές 0 και 1 που δηλώνει εάν υπάρχει διαφήμιση ή όχι σε κάθε προϊόν. Υπολογίζεται το άθροισμα των units_sold και βαθμολογιών από 1 έως 5 αστέρι για κάθε τιμή της " uses_add_boosts" αντίστοιχα. Στο παρακάτω γράφημα φαίνονται τα αποτελέσματα.

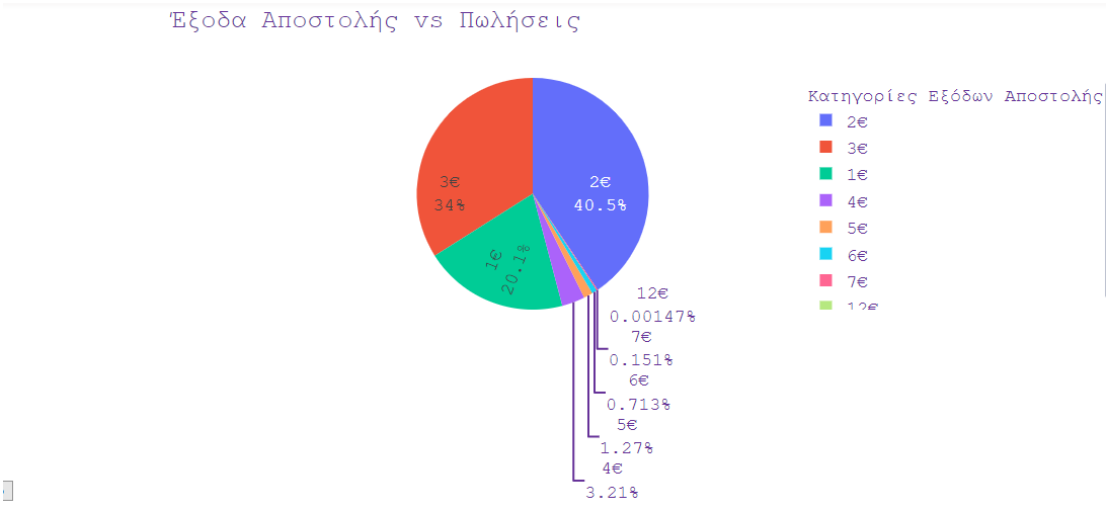


Εικόνα 24 Barplot Ποσοστό πωλήσεων & αξιολογήσεων σε χρήση ή όχι διαφήμισης

Παρατηρείται ότι η χρήση διαφημιστικών ωθήσεων στα προϊόντα δεν προσφέρει αύξηση πωλήσεων και αξιολογήσεων. Το ποσοστό πωλήσεων είναι πιο χαμηλό 41.46 % και αξιολογήσεων επίσης με χρήση διαφημίσεων. Επομένως η χρήση διαφημιστικών ενισχύσεων δεν φαίνεται να επηρεάζει την επιτυχία του Wish.com και ο ιστότοπος ενδέχεται να χάσει έσοδα από αυτές τις διαφημίσεις.

3.1.5.4 Έξοδα Αποστολής

Σε αυτό το ερώτημα διερευνάται τί έξοδα αποστολής προτιμά ο πελάτης να κάνει και σε ποιες τιμές γίνονται οι μεγαλύτερες πωλήσεις. Για την δημιουργία του διαγράμματος γίνεται χρήση του Piechart της λεγόμενης πίτας.

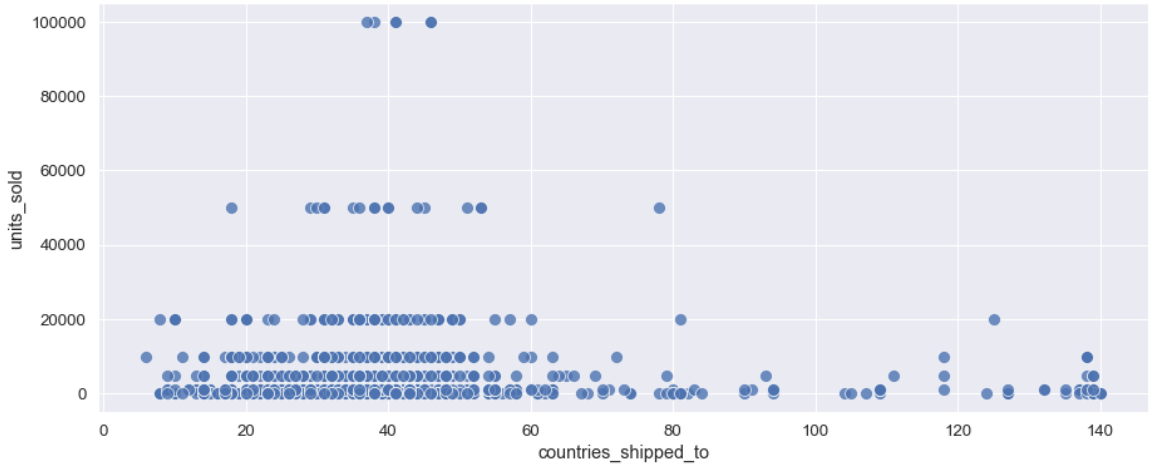


Εικόνα 25 Piechart Έξοδα αποστολής vs Πωλήσεις

Το μεγαλύτερο ποσοστό των πωλήσεων γίνεται στο εύρος τιμών 1-3 ευρώ για έξοδα αποστολής που είναι φυσιολογικό.

3.1.5.5 Χώρες *Αποστολής*

Σε αυτό το ερώτημα χρησιμοποιείται η στήλη " countries_shipped_to" που δηλώνει σε πόσες χώρες γίνεται αποστολή του κάθε προϊόντος. Για τη λήψη πληροφοριών απεικονίζεται ένα scatterplot από τη βιβλιοθήκη Seaborn.

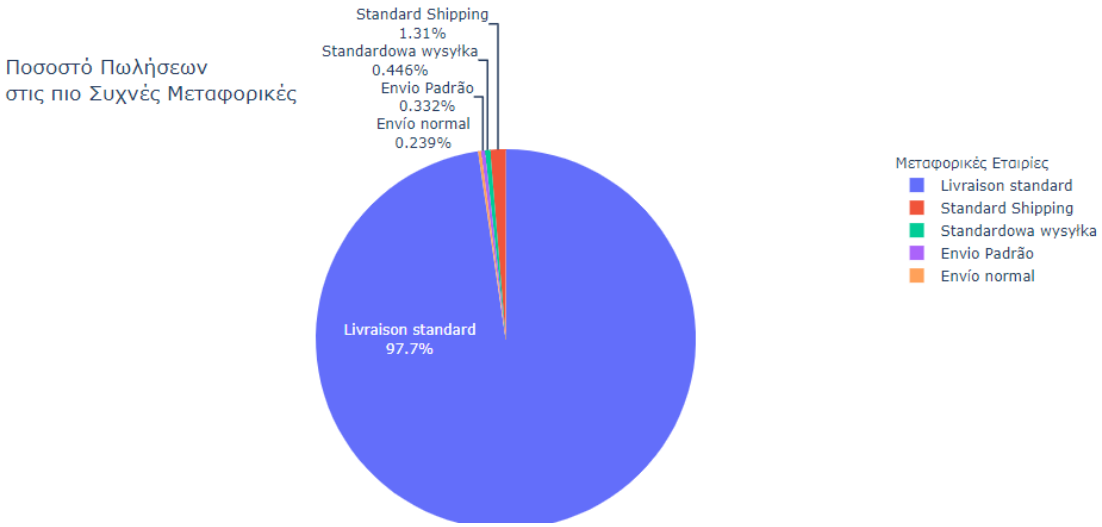


Εικόνα 26 Scatterplot Πλήθος χωρών αποστολής ανά πωλησθέντες μονάδες προϊόντων

Φαίνεται ότι αν ένα προϊόν αποστέλλεται σε μεγάλο αριθμό χωρών δεν σημαίνει ότι θα πωληθεί περισσότερο. Οι μεγαλύτερες πωλήσεις γίνονται όταν τα προϊόντα φτάνουν σε 20-60 χώρες.

3.1.5.6 *Μεταφορικές Εταιρείες*

Σε αυτό το κομμάτι παρουσιάζονται οι μεταφορικές εταιρείες με το αντίστοιχο αθροιστικό ποσοστό πωλήσεων. Για την απεικόνιση της πληροφορίας έγινε η χρήση του piechart με την βιβλιοθήκη pyplot.

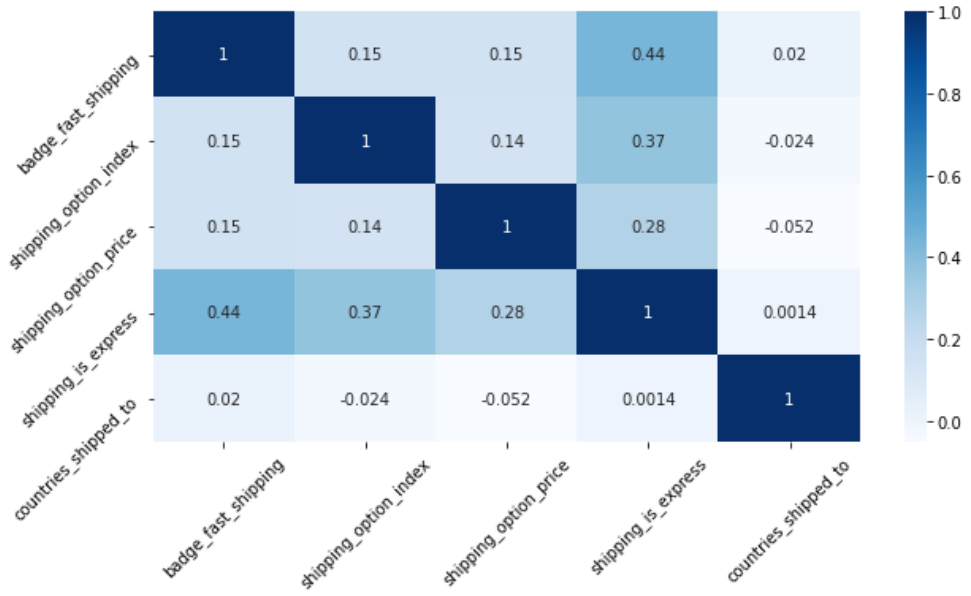


Εικόνα 27 Piechart Ποσοστό Πωλήσεων ανά κατηγορία Μεταφορικών Εταιρειών

Δεξιά αναγράφονται οι μεταφορικές εταιρείες που υπάρχουν στο Wish.com και στο διάγραμμα φαίνεται ότι το μεγαλύτερο μέρος της αποστολής πραγματοποιείται από την εταιρεία Livraison standard.

3.1.5.7 *Παράγοντας που συμβάλει στο σήμα γρήγορης αποστολής*

Το σήμα γρήγορης αποστολής περιγράφεται στο σύνολο δεδομένων από τη στήλη “badge_fast_shipping”. Πρόκειται για μία boolean μεταβλητή με τιμές 1 και 0. Όπου 1 υπάρχει κάποιο ή και όλα απ’ τα πεδία “shipping_option_index, shipping_option_price, shipping_is_express, countries_shipped_to” οι οποίες είναι στήλες του πίνακα όπου η περιγραφή τους έχει δοθεί παραπάνω. Για να φανεί ποιος παράγοντας συντελεί στο σήμα γρήγορης αποστολής γίνεται η χρήση του διαγράμματος heatmap ή αλλιώς πίνακας συσχετίσεων από τη βιβλιοθήκη Seaborn. Όσο πιο κοντά στη τιμή 1 μία στήλη με μία άλλη τόσο μεγαλύτερη συσχέτιση έχουν μεταξύ τους.



Εικόνα 28 Heatmap Παράγοντας που συμβάλει στο σήμα γρήγορης αποστολής

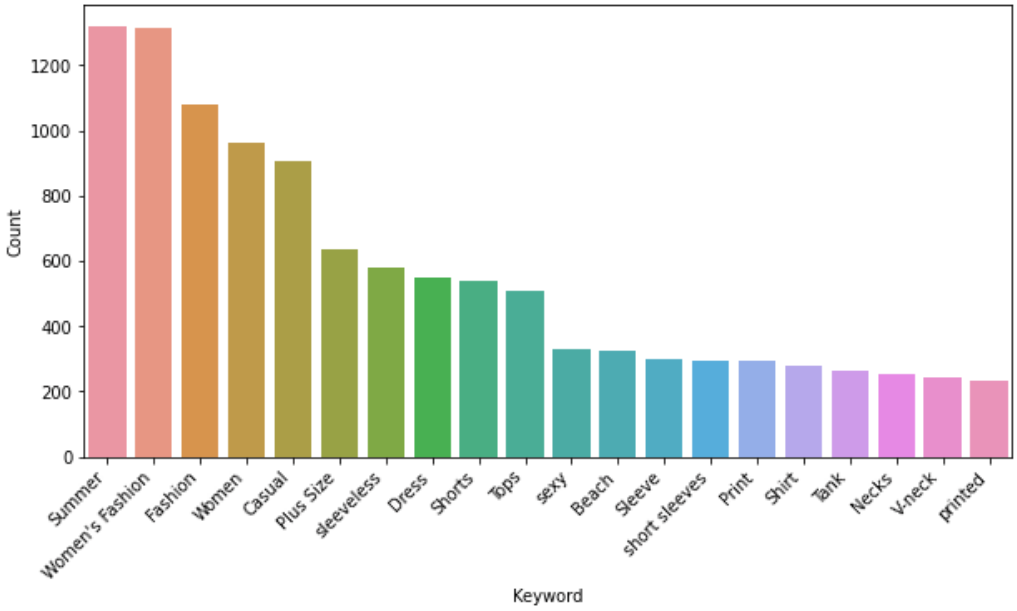
Οι στήλες “ badge_fast_shipping” και “ shipping_is_epress” έχουν την μεγαλύτερη τιμή συσχέτισης που ισούται με 0.44. Επομένως ο παράγοντας που συμβάλλει περισσότερο στο σήμα γρήγορης αποστολής είναι το “ shipping_is_express” . Παρόλα αυτά όμως, η γρήγορη αποστολή γίνεται σε πολύ χαμηλό πλήθος πωλήσεων . Αυτό φαίνεται και πιο καθαρά με το παρακάτω αθροιστικό πίνακα.

	shipping_is_express	units_sold
0	0	6795070
1	1	11200

Εικόνα 29 Αθροιστικός πίνακας "shipping_is_express" με "units_sold"

3.1.5.8 Keywords – Tags_count

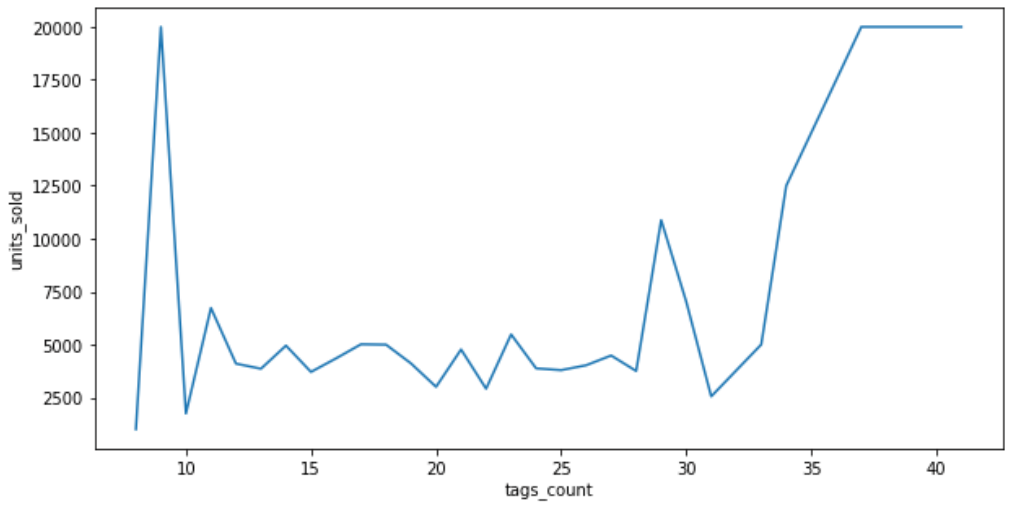
Σε αυτό το ερώτημα απεικονίζονται οι λέξεις-κλειδιά που εμφανίζονται περισσότερο στα προϊόντα. Επιπλέον δημιουργείται μία νέα στήλη στο κύριο πίνακα με το όνομα “tags_count” που εμφανίζει το πλήθος των tags για κάθε εγγραφή και χρησιμοποιείται για το δεύτερο γράφημα. Πρέπει να τονισθεί ότι το feature “keyword” βρίσκεται στο dataset “unique_categories γι’ αυτό και πριν τη δημιουργία του barplot φορτώνεται το dataset με τη χρήση της συνάρτησης .read_csv().



Εικόνα 30 Barplot Συχνότητα εμφάνισης των "keyword"

Από το παραπάνω barplot παρατηρούμε ότι τα περισσότερα προϊόντα είναι καλοκαιρινά και γυναικεία.

Το ακόλουθο διάγραμμα lineplot μέσω της βιβλιοθήκης Seaborn απεικονίζει το πλήθος των tags που γίνονται οι περισσότερες πωλήσεις.



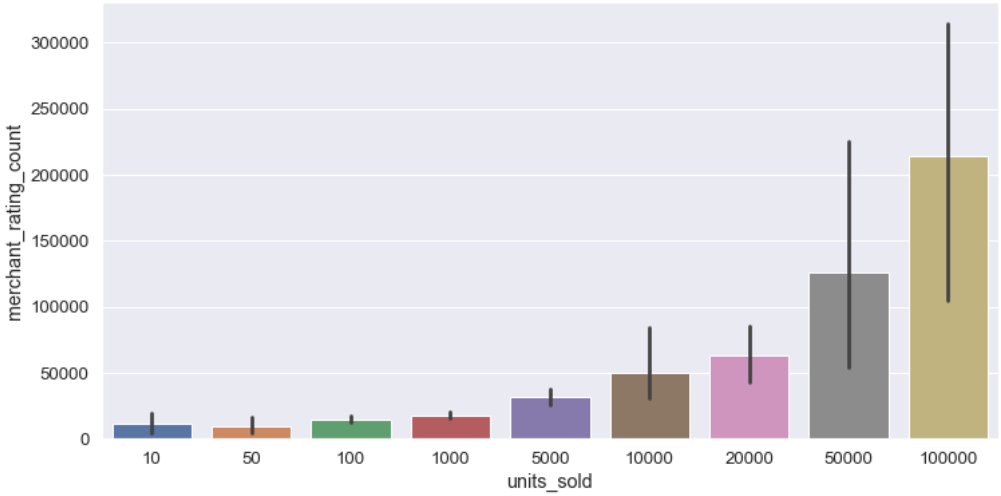
Εικόνα 31 Lineplot "tags_count" ανά "units_sold"

Παρατηρείται ότι τα προϊόντα με ετικέτες άνω των 35 είναι πιο εύχρηστα και έτσι αγοράζονται πιο συχνά. Υπάρχει μία ξαφνική άνοδος ακριβώς κάτω από τις 10 ετικέτες.

Φιλτράροντας το σύνολο δεδομένων να εμφανίζει μόνο πλήθος ετικετών κάτω των 10 και αθροίζοντας πόσα προϊόντα εμφανίζονται σε κάθε εύρος “units_sold” εμφανίζονται 27 προϊόντα συνολικά. Υπάρχουν μόνο 3 με πωλήσεις 20000 και τα υπόλοιπα έχουν πωλήσεις όπως 10,50,100,1000,5000. Έτσι φαίνεται ότι αυτά είναι ακραίες τιμές και συνεπώς δημιουργούν μια ξαφνική άνοδο κάτω από τις 10 ετικέτες.

3.1.5.9 Επίδραση των εμπόρων στην επιτυχία πωλήσεων

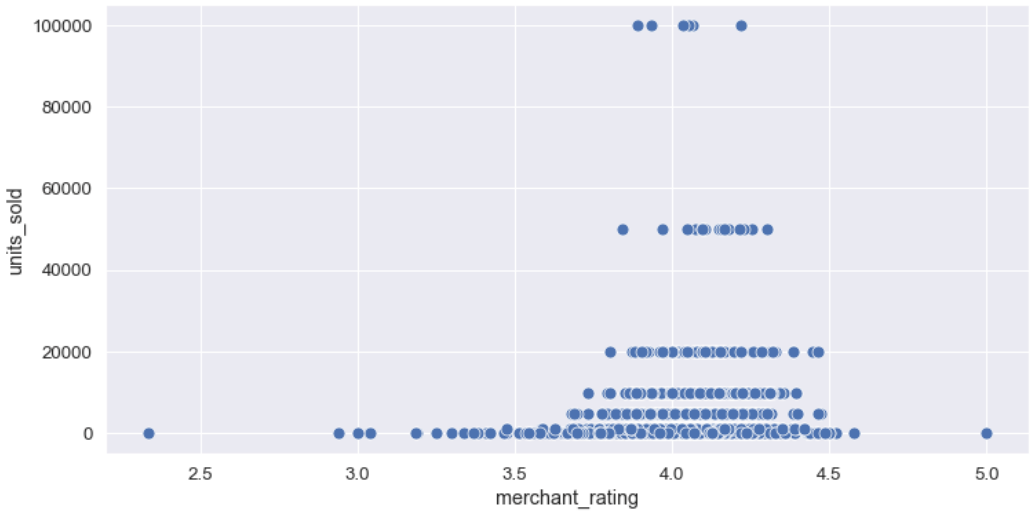
Αρχικά δημιουργείται ένα barplot μεταξύ του πλήθους αξιολογήσεων των εμπόρων από τους αγοραστές “ merchant_rating_count” και των πωληθέντων μονάδων των προϊόντων “units_sold” με τη βιβλιοθήκη Seaborn.



Εικόνα 32 Barplot πλήθος αξιολογήσεων εμπόρων ανά πωλήσεις

Το παραπάνω σχήμα καθιερώνει σαφώς μία θετική συσχέτιση του αριθμού των αξιολογήσεων που λαμβάνονται από ένα έμπορο και των πωληθέντων μονάδων. Με άλλα λόγια, όσο ψηλότεροι είναι οι ψήφοι βαθμολογίας για έναν έμπορο, τόσο υψηλότερη η πώληση.

Στη συνέχεια με το scatterplot μεταξύ των “units_sold” και “merchant_rating” φαίνονται περισσότερες λεπτομέρειες για τις αξιολογήσεις των εμπόρων.



Εικόνα 33 Scatterplot Αξιολογήσεις εμπόρων ανά πωλήσεις

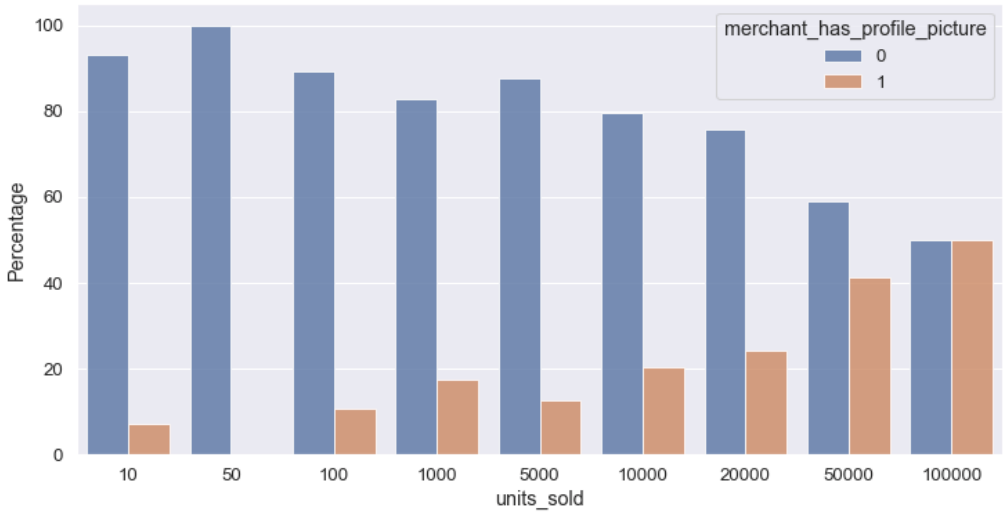
Η κάθε κουκκίδα του παραπάνω γραφήματος αναπαριστά το κάθε προϊόν. Παρατηρείται ότι στο σημείο που έχουμε τις περισσότερες πωλήσεις οι αξιολογήσεις των εμπόρων είναι μεταξύ 3.7 με 4.5 δηλαδή μέτρια προς υψηλή. Αν ο έμπορος έχει βαθμολογία πάνω από 4.5 πουλάει πολύ λίγα προϊόντα.

Σχετικά με το κομμάτι των εμπόρων δημιουργείται ένα νέο feature το οποίο υπολογίζει το σκορ αξιολογήσεων των εμπόρων από τους αγοραστές. Η νέα στήλη ονομάζεται “merchant_rating_score” και γίνεται ένα νέο barplot που υπολογίζει το μέσο σκορ αξιολογήσεων εμπόρων ως προς τις πωλήσεις των προϊόντων.

Εικόνα 34 Barplot Μέσο σκορ αξιολογήσεων εμπόρων ως προς τις πωλήσεις προϊόντων

Από εδώ προκύπτει η ίδια πληροφορία όπως παραπάνω. Όσο αυξάνεται το σκορ των αξιολογήσεων των εμπόρων τόσο αυξάνονται και οι πωλήσεις.

Τέλος, εξετάζεται η σχέση μεταξύ του προφίλ των εμπόρων και των πωλήσεων. Η στήλη “ merchant_has_profile_picture ” είναι μία boolean μεταβλητή με τιμές 0 και 1. Κατασκευάζεται ένα barplot με όλες τις τιμές των “units_sold” και σε κάθε μία υπάρχει η τιμή 0 και 1 αν έχει προφίλ ή όχι ο έμπορος και σε τί ποσοστό.



Εικόνα 35 Barplot Ποσοστό "merchant_has_profile_picture" ανά "units_sold"

Στο εύρος των υψηλότερων πωλήσεων περισσότεροι έμποροι τείνουν να έχουν μία εικόνα προφίλ. Αυτό θα μπορούσε να είναι η βασική ανθρώπινη ψυχολογία. Με τη φωτογραφία του εμπορικού καταστήματος αποκτάται περισσότερη εμπιστοσύνη από τους πελάτες για να αγοράσουν περισσότερο από ένα προϊόν.

3.1.5.10 Συνολικό απόθεμα για όλες τις παραλλαγές προϊόντος

Φιλτράροντας τον σύνολο δεδομένων sales_data με τη συνάρτηση .groupby από το πακέτο Pandas που χρησιμοποιείται στο μεγαλύτερο κομμάτι των οπτικοποιήσεων υπολογίζονται αθροιστικά οι πωλήσεις που γίνονται για κάθε τιμή της στήλης “inventory_total”.

```
inventory=sales_data[["inventory_total","units_sold"]]
inventory=inventory.groupby(["inventory_total"]).sum().reset_index()
inventory
```

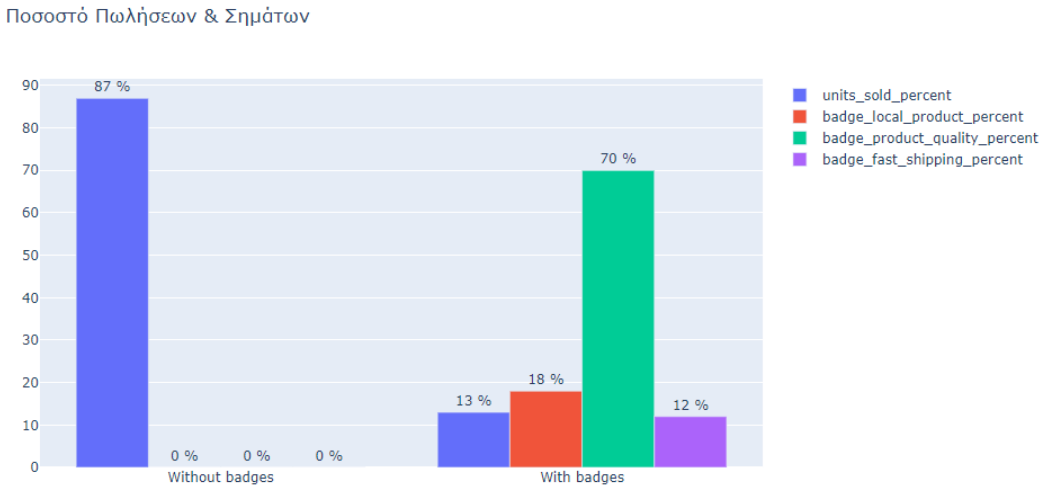
	inventory_total	units_sold
0	1	20000
1	2	150
2	9	100
3	24	50
4	30	100
5	36	100
6	37	50
7	38	100
8	40	1000
9	50	6784620

Εικόνα 36 Αθροιστικός πίνακας "units_sold" ανά "inventory_total"

Οι περισσότερες πωλήσεις γίνονται όταν το σύνολο αποθέματος είναι πολύ χαμηλό είτε είναι πραγματικά υψηλό.

3.1.5.11 Επίδραση πωλήσεων στο σύνολο σημάτων

Στο παρακάτω barplot υπολογίζεται το ποσοστό των πωλήσεων σε προϊόντα που περιέχουν κάποια σήματα ή και όχι . Επίσης υπολογίζεται σε αυτά που υπάρχει σήμα τί ποσοστό αυτών υπάρχει και ποιο υπερτερεί. Χρησιμοποιούνται οι στήλες “units_sold”, “badge_local_product”,”badge_product_quality”,”badge_fast_shipping” .

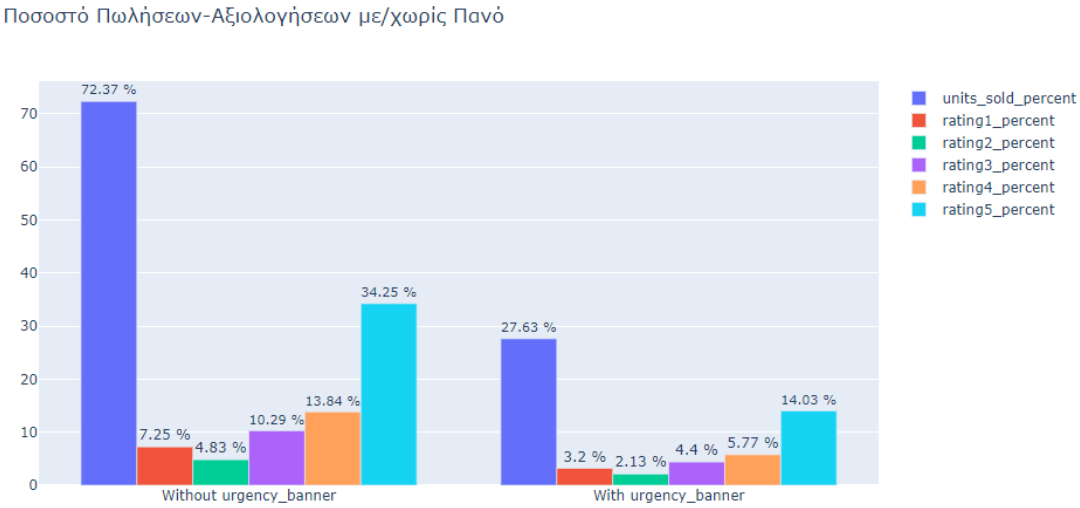


Εικόνα 37 Barplot Ποσοστό Πωλήσεων και σημάτων

Το μεγαλύτερο ποσοστό πωλήσεων γίνεται χωρίς να υπάρχουν κάποιου είδους σήματα στα προϊόντα. Σε αυτά που υπάρχουν, μεγαλύτερο αντίκτυπο στις πωλήσεις έχει το σήμα που δηλώνει την ποιότητα του προϊόντος. Τα σήματα ποιότητας φαίνεται να αυξάνουν την επιτυχία των πωλήσεων.

3.1.5.12 Επίδραση πωλήσεων – αξιολογήσεων στο επείγον πανό

Η στήλη “has_urgency_banner” του συνόλου δεδομένων sales_data είναι μία boolean μεταβλητή με τιμές 0 και 1 και υποδηλώνει εάν ένα προϊόν έχει κάποιο επείγον πανό όπως το κείμενο πάνω σε φωτογραφία “almost gone!”. Με την απεικόνιση του παρακάτω barplot γίνεται ένας έλεγχος αν το επείγον κείμενο επηρεάζει την αύξηση των πωλήσεων άρα και την επιτυχία των προϊόντων και αν τα ποσοστά αξιολογήσεων από πελάτες έχουν μεγάλη διαφορά είτε υπάρχει είτε όχι αυτό.

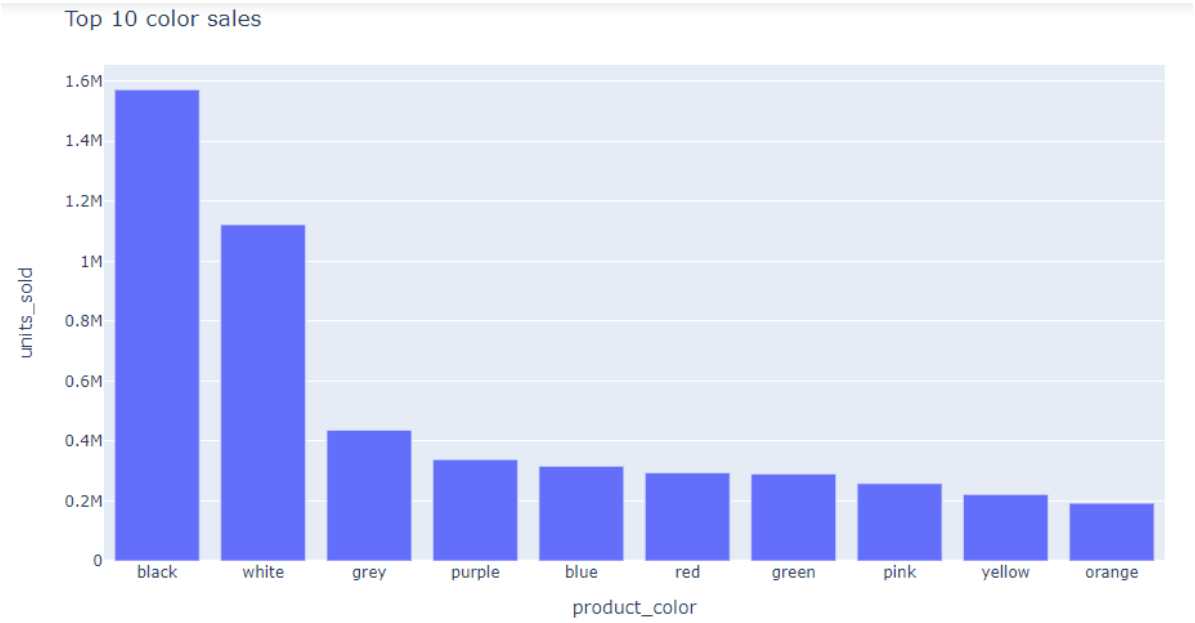


Εικόνα 38 Barplot Ποσοστό Πωλήσεων- Αξιολογήσεων με ή χωρίς πανό

Χωρίς επείγον πανό υπάρχει ένα εμφανής υψηλό ποσοστό πωλήσεων σε σχέση με τα προϊόντα που έχουν. Το ίδιο ισχύει με τις αξιολογήσεις αλλά με πολύ μικρές διαφορές. Φαίνεται οι αγοραστές να μην επηρεάζονται τόσο πολύ με αυτά τα “texts” για να βάλουν κάποια αξιολόγηση. Επομένως το “urgency_banner” δεν παίζει σημαντικό ρόλο στην επιτυχία των πωλήσεων.

3.1.5.13 *Χρώματα προϊόντων με τις μεγαλύτερες πωλήσεις*

Σε αυτό το ερώτημα παρουσιάζεται το barplot με τα κορυφαία δέκα χρώματα που έκαναν τις μεγαλύτερες πωλήσεις.

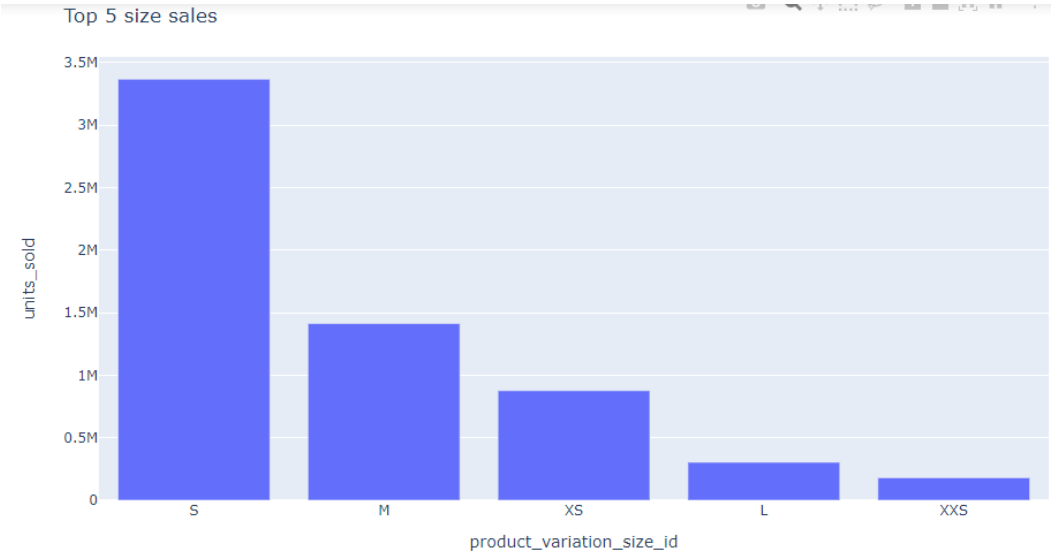


Εικόνα 39 Barplot Πλήθος "units_sold" ανά "product_color"

Τα προϊόντα που προτιμάν οι χρήστες έχουν χρώμα κυρίως μαύρο και άσπρο.

3.1.5.14 *Μεγέθη προϊόντων με τις μεγαλύτερες πωλήσεις*

Σε αυτό το βήμα απεικονίζεται το barplot με τα 5 πιο συχνά μεγέθη που προτιμούν οι χρήστες.

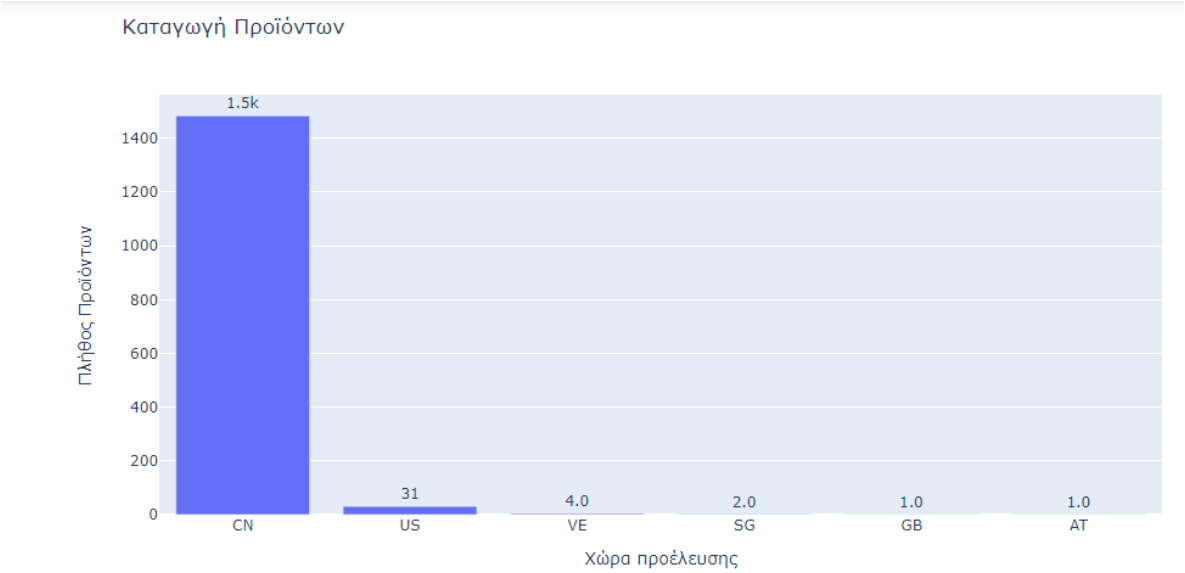


Εικόνα 40 Barplot Πλήθος "units_sold" ανά "product_variation_size_id"

Οι χρήστες φαίνεται να αγοράζουν περισσότερο το small μέγεθος.

3.1.5.15 Χώρα προέλευσης προϊόντων

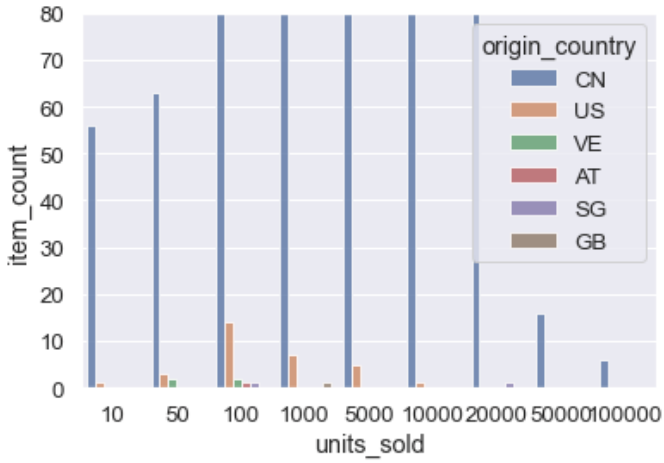
Στο παρακάτω barplot φαίνεται τί καταγωγή έχουν τα προϊόντα που υπάρχουν στο Wish.com και πού γίνονται οι περισσότερες πωλήσεις.



Εικόνα 41 Barplot Πλήθος Πωλήσεων ανά Χώρα προέλευσης

Τα περισσότερα είδη είναι κινέζικης προέλευσης. Αυτό δεν προκαλεί έκπληξη, δεδομένου ότι η Κίνα είναι μία από τις μεγαλύτερες κατασκευαστικές δυνάμεις.

Για να υπάρχει μία καλύτερη εικόνα ανά εύρος πωλήσεων αναφέρεται και ένα επιπλέον barchart που δείχνει σε κάθε τιμή των “units_sold” ποια χώρα επικρατεί.



Εικόνα 42 Barchart Συχνότητα χωρών προέλευσης ανά "units_sold"

Τα κινέζικα είδη κυριαρχούν σε όλα τα εύρη πωλήσεων. Μετά την Κίνα, τα περισσότερα είδη προέρχονται από τις ΗΠΑ. Η ποικιλομορφία στη χώρα προέλευσης υπάρχει στο εύρος πωλήσεων των 100.

3.1.5.16 Προτιμήσεις πελατών στο είδος των προϊόντων

Για να φανεί ποια είδη έχουν περισσότερη ζήτηση από τους χρήστες γίνεται φιλτράρισμα του πίνακα να εμφανίζει μόνο τις παρατηρήσεις όπου το “units_sold” είναι άνω των 1000 εφόσον και πάνω από αυτή τη τιμή θεωρείται ότι το Wish έχει επιτυχία στις πωλήσεις του. Οι πληροφορίες αντλούνται από το γράφημα wordcloud καλώντας τη βιβλιοθήκη Wordcloud. Όσες λέξεις φαίνονται αρκετά μεγάλες σημαίνει ότι έχουν και τη μεγαλύτερη συχνότητα στο σύνολο δεδομένων.

Στο 1ο γράφημα εξετάζονται τα είδη που έχουν τη μεγαλύτερη συχνότητα και στο 2ο τα είδη που ζητούνται περισσότερο σύμφωνα με τις πωλήσεις. Αφαιρούνται οι

λέξεις “Women, Summer, Fashion” για καλύτερη διερεύνηση των ειδών που πωλούνται περισσότερο.

Εικόνα 43 Wordcloud Συχνότητα εμφάνισης προϊόντων

Από το παραπάνω wordcloud τα προϊόντα σε Plus size έχουν υψηλό βαθμό καταχωρίσεων. Η γυναικεία μόδα είναι εμφανής και τα “dress,top,sleeveless” έχουν εμφανή διαφορά.

Εικόνα 44 Wordcould Συχνότητα εμφάνισης πωληθέντων ειδών

Τα είδη που έχουν τη μεγαλύτερη ζήτηση στις πωλήσεις είναι τα “Tank Top, Plus Size, T shirt, V neck”.

3.2 Προ επεξεργασία δεδομένων

Σε αυτό το βήμα γίνεται επεξεργασία στο σύνολο δεδομένων. Το κομμάτι αυτό είναι αρκετά σημαντικό αφού για την εκπαίδευση των μοντέλων που γίνεται στο επόμενο κεφάλαιο, το dataset χρειάζεται να είναι όσο πιο καθαρό γίνεται για να είναι και πιο αξιόπιστα τα αποτελέσματα. Κάποιες ενέργειες έχουν εκτελεστεί στο κεφάλαιο του exploratory καθώς οι οπτικοποιήσεις πρέπει να είναι πιο αντιπροσωπευτικές και χωρίς περεταίρω πληροφορίες που μπερδεύουν τον αναγνώστη. Τέτοιες ενέργειες ήταν η αφαίρεση των διπλότυπων εγγραφών και η εκχώρηση κάποιων τιμών των "units_sold" μέσα σε άλλες εφόσον η συχνότητα τους ήταν αρκετά χαμηλή. Ωστόσο μπορεί να γίνει μία επανάληψη στην επεξεργασία του target "units_sold" ώστε να τελειοποιηθεί.

Δημιουργείται ένα νέο αρχείο με την ονομασία " Preprocessing " όπου καλούνται οι ίδιες βιβλιοθήκες όπως και στα προηγούμενα αρχεία. Για να είναι κατανοητή η επεξεργασία, σε κάθε μεταβλητή αναφέρονται παρατηρήσεις και ενέργειες που εκτελούνται για το καθάρισμα του dataset. Για καλύτερη αντίληψη αναφέρεται και κομμάτι του κώδικα σε αυτό το βήμα.

3.2.1 Χειρισμός ελλιπών – λανθασμένων τιμών

Μη αριθμητικές μεταβλητές.

- "currency_buyer" : Δεν υπάρχουν τιμές που λείπουν και η μοναδική τιμή είναι "EUR".

- "urgency_text : Παρατηρείται ότι όπου υπάρχουν κενές τιμές στη στήλη αυτή υπάρχουν και στην αντίστοιχη " has_urgency_text " η οποία είναι δυαδική μεταβλητή και δείχνει αν κάθε προϊόν έχει κάποιο πανό προειδοποίησης ή όχι. Αυτό σημαίνει ότι συσχετίζονται μεταξύ τους. Συνεπώς όταν υπάρχει ποσότητα 1, υπάρχει και μία αντίστοιχη ποσότητα στο "urgency_text" που δείχνει το είδος προειδοποίησης. Όμοια αν δεν υπάρχει επείγον πανό δεν υπάρχει και "urgency_text". Άρα η ύπαρξη της κενής τιμής εδώ είναι τύπου MNAR (missing not at random). Δηλαδή η

πιθανότητα να λείπει τιμή ποικίλλει για άγνωστους λόγους. (‘sec-MCAR’, no date) Λείπει τυχαία η τιμή αλλά υπάρχει λόγος που αφήνεται κενή ίσως γιατί δεν υπάρχει επείγον πανό στη φωτογραφία του προϊόντος άρα και όχι ποσότητα αυτού. Στη παρακάτω εικόνα φαίνεται ένα κομμάτι του dataset με την συσχέτιση που υπάρχει στις κενές ποσότητες των μεταβλητών.

```
sales_data[["has_urgency_banner", "urgency_text"]]
```

	has_urgency_banner	urgency_text
0	1.0	Quantité limitée !
1	1.0	Quantité limitée !
2	1.0	Quantité limitée !
3	NaN	NaN
4	1.0	Quantité limitée !
...	...	...
1568	NaN	NaN
1569	1.0	Quantité limitée !
1570	NaN	NaN
1571	NaN	NaN
1572	NaN	NaN

Εικόνα 45 Πίνακας Συσχέτισης κενών τιμών

Επίσης οι μοναδικές τιμές της μεταβλητής “urgency_text” είναι 2 και όλες οι εγγραφές πλην μίας καλύπτονται από την μία ποσότητα. Επομένως αφαιρείται η στήλη και μένει η στήλη “has_urgency_banner” όπου οι κενές τιμές αντικαθίστανται με μηδέν.

```
In [51]: # Αφαίρεση της στήλης urgency_text
         sales_data.drop(["urgency_text"], axis=1, inplace=True)

         # Αντικατάσταση ελλειπών τιμών της στήλής has_urgency_banner με 0.
         sales_data["has_urgency_banner"] = sales_data["has_urgency_banner"].fillna(0)
```

Εικόνα 46 Κώδικας αφαίρεσης στηλών και συμπλήρωσης κενών τιμών “urgency_text”,”has_argency_banner”

- “ product_color ” : Υπάρχουν 41 τιμές που λείπουν και πιο συχνή εμφάνιση έχει το χρώμα μαύρο. Υπάρχουν αρκετές διαφορετικές αποχρώσεις κάτι που σίγουρα στην εκπαίδευση των μοντέλων θα δημιουργούσε αρκετή πληροφορία. Γίνεται συνδυασμός των διαφορετικών αποχρώσεων ενός

χρώματος σε ένα. Παρατηρείται ότι κάποια χρώματα είναι διπλά και επαναλαμβανόμενα με διαφορετική γραφή. Επομένως δημιουργείται μία συνάρτηση όπου όποιο στοιχείο περιέχει το σύμβολο “&” τοποθετείται στη κατηγορία “dual”, όποιο έχει άλλη γραφή ή απόχρωση τοποθετείται στο βασικό χρώμα και όλες οι κενές τιμές τοποθετούνται στη κατηγορία “other” . Για να γίνει αυτό πρώτα κατασκευάζεται ένα dictionary το οποίο είναι ένα σύνολο μη ταξινομημένων τιμών που περιέχει κλειδιά και τις αντίστοιχες ποσότητες. (‘Dictionaries in Python – Real Python’, no date) Τα κλειδιά είναι οι τιμές που υπάρχουν στη μεταβλητή και οι ποσότητες οι αντίστοιχες νέες τιμές που δημιουργούνται.

```
shade_to_colour = {
    'navyblue': 'blue', 'lightblue': 'blue', 'skyblue': 'blue', 'lakeblue': 'blue', 'darkblue': 'blue', 'denimblue': 'blue', 'nav
    'navy': 'blue',
    'armygreen': 'green', 'army green': 'green', 'fluorescentgreen': 'green', 'mintgreen': 'green', 'light green': 'green', 'ligh
    'applegreen': 'green', 'darkgreen': 'green', 'army': 'green', 'khaki': 'green', 'lightkhaki': 'green',
    'lightyellow': 'yellow',
    'winered': 'red', 'wine red': 'red', 'lightred': 'red', 'coralred': 'red', 'rose red': 'red', 'watermelonred': 'red', 'orange
    'claret': 'red', 'burgundy': 'red',
    'gray': 'grey', 'silver': 'grey','lightgray': 'grey', 'lightgrey': 'grey', 'greysnakeskinprint': 'grey',
    'coffee': 'brown', 'camel': 'brown', 'tan': 'brown',
    'offwhite': 'white', 'ivory': 'white', 'nude': 'white',
    'lightpink': 'pink', 'dustypink':'pink', 'rosegold': 'pink',
    'lightpurple': 'purple', 'coolblack': 'black', 'apricot': 'orange', 'offblack': 'black'
}

def update_color(col):
    if shade_to_colour.get(col, False):
        return shade_to_colour.get(col)
    elif '&' in col:
        return 'dual'
    elif col in shade_to_colour.values():
        return col
    else:
        return 'other'

sales_data['product_color'].replace(np.nan, 'others', inplace=True)

sales_data['product_color'] = sales_data.product_color.apply(update_color)
```

Εικόνα 47 Κώδικας χειρισμού λανθασμένων και κενών τιμών “product_color”

- “product_variation_size_id” : Αποτελείται από 14 ελλιπείς τιμές. Η πιο συχνή είναι το “S”. Παρατηρείται ότι πολλές τιμές επαναλαμβάνονται με διαφορετική γραφή. Επίσης υπάρχουν κενά χωρίς νόημα και λάθος τιμές. Επομένως με τη συνάρτηση .replace() απτό πακέτο Pandas όλα τα λάθη αντικαθίστανται με τα βασικά νούμερα και όλα τα υπόλοιπα τοποθετούνται σε κατηγορία “ other “ μαζί με τις κενές τιμές.

```
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('2xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('3xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('4xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('5xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('6xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('x   l', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('sizel', 'l')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('size4xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('x   l', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('1 pc - xl', 'xl')

def change_size(cl):
    if cl in 'xl,l,s,xs,m,xxl,xxxs,xxxxxl,xxxxl'.split(','):
        return cl
    else:
        return 'other'

sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace(np.nan, 'OTHER')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].apply(change_size)
```

```
sales_data['product_variation_size_id'].unique()
```

```
array(['m', 'xs', 's', 'other', 'l', 'xxl', 'xxxxxl', 'xl', 'xxxs',
       'xxxxl'], dtype=object)
```

Εικόνα 48 Κώδικας χειρισμού λανθασμένων και κενών τιμών "product_variation_size"

- "origin_country" : Περιέχει 17 ελλιπείς τιμές. Η λογική λέει ότι έτυχε να μην καταγραφούν. Επομένως είναι ελλιπείς τιμές MCAR (missing completely at random). ('sec-MCAR', no date). Όλες οι κενές τιμές αντικαθίστανται με την πιο συχνή που είναι η Κίνα.

```
sales_data['origin_country'].fillna('CN', inplace=True)
```

Εικόνα 49 Κώδικας χειρισμού κενών τιμών "origin_country"

- "merchant_name" : Περιέχει 3 κενές τιμές και δηλώνει το κανονικό όνομα του πωλητή. Εφόσον υπάρχει το όνομα του καταστήματος δεν παίζει σημαντικό ρόλο η στήλη αυτή. Επίσης περιέχει αρκετές διαφορετικές τιμές που είναι δύσκολο να αναλυθούν.

```
sales_data["merchant_name"].unique()
```

```
array(['zgrdejia', 'sarahouse', 'hxt520', 'allenfan', 'happyhorses',
       'zhoulinglinga',
       'uniquelifashionshopbb657bfe91d211e598c7063a14dc88b5', 'soband',
       'chenxiangjunjun', 'luoweiclothe', 'mayuhiao', 'lanniesdesign',
       'easymarket', 'molesfashion', 'pentiumhorse', 'pethasboutique',
       '2312hangm', 'leiston', 'centper3', 'chenqinq',
       'hongkonghaijietradecolimited', 'ouchocoltd', 'huaxianglarou',
       'purefashionltd', 'sjhdstoer', 'terbuer', 'coaluss',
       'happyshoooping', 'memo2', 'meiximeiyo', 'hotdress',
       'newfashionshoppingpark', 'kallyett', 'maxgoods', 'sangboostore',
       'linfashionstore', 'unnistore', 'cuetaes', '广州乔莎服饰有限公司', 'wenj498',
       'baijupingshop', 'wuewi', 'brittany', 'qiaopiaduxiu', 'knights',
       'fengjinying', 'enjoythesunshine', 'chentengying', 'gettrendy',
       'floraboutique', 'cyb654', 'wowbang', 'nalininternational',
       'maryswill', '泷翔', 'zhuangyuping1', 'zanzeaofficialstore',
       'caishuiwang', 'zufanqiudinli', 'happyfeng', 'besagift', 'lianlie',
       'saya2016', 'zaitaowanggou', 'keepahorse', 'taomigogo', 'aidos',
       'smarthomeinternationalcoltd', 'wdhdage', 'charming_family',
       'wslcwm', 'seraih', 'chooop', 'luojiayushop', 'wangjiangxu',
```

```
sales_data.drop(["merchant_name"], axis=1, inplace=True)
```

Εικόνα 50 Τιμές στήλης "merchant_name"

- “merchant_info_subtitle” : Περιέχει 1 κενή τιμή. Η μεταβλητή περιέχει κείμενο που φαίνεται στην ενότητα πληροφοριών πωλητή στο χρήστη. Δίνει μία επισκόπηση των στατιστικών του πωλητή. Συνήθως αποτελείται από “ %<positive feedback>”. Δεν περιέχει κάποια σημαντική πληροφορία και έχει 1058 διαφορετικές εγγραφές. Άρα η στήλη αυτή αφαιρείται.

- “merchant_profile_picture” : Περιέχει 1314 ελλιπείς τιμές. Παρατηρείται ότι σχετίζεται με τη μεταβλητή “merchant_has_profile_picture” η οποία είναι δυαδική και δείχνει αν υπάρχει ή όχι φωτογραφία πωλητή. Διαπιστώνεται ότι όταν η μεταβλητή “merchant_has_profile_picture” είναι μηδέν τότε υπάρχει κενή τιμή στη μεταβλητή “merchant_profile_picture” και αυτό γιατί δεν υπάρχει το url της φωτογραφίας του εμπόρου. Επομένως η κενή τιμή δεν λείπει τυχαία και ανήκει στη κατηγορία MNAR. Οι δύο παραπάνω μεταβλητές δίνουν την ίδια πληροφορία και όλη σχεδόν η στήλη έχει κενές ποσότητες γι’ αυτό και η συγκεκριμένη μεταβλητή αφαιρείται.

```
sales_data[['merchant_has_profile_picture','merchant_profile_picture']]
```

	merchant_has_profile_picture	merchant_profile_picture
0	0	NaN
1	0	NaN
2	0	NaN
3	0	NaN
4	0	NaN
...	...	...
1568	0	NaN
1569	0	NaN
1570	0	NaN
1571	0	NaN
1572	0	NaN

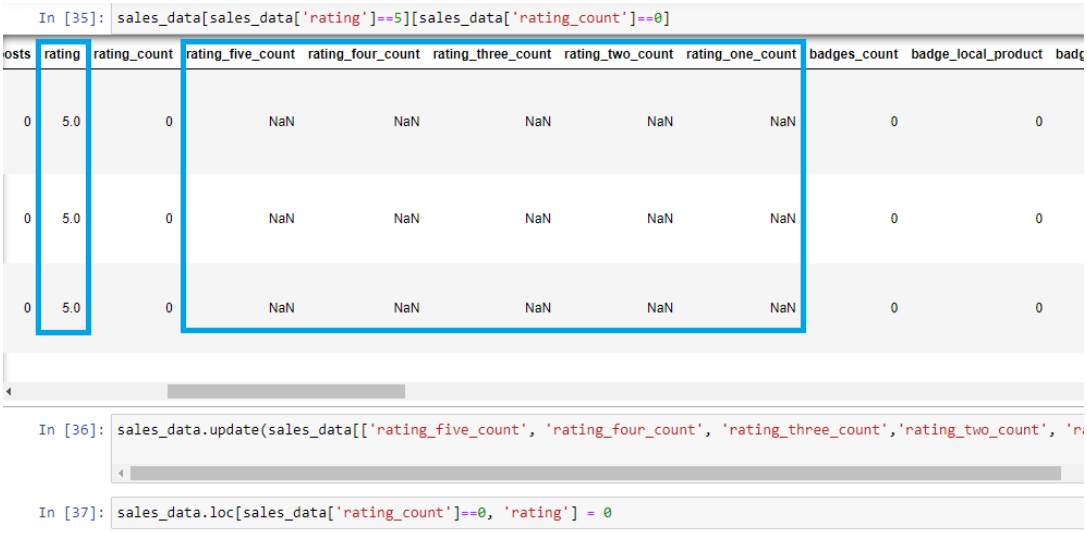
1539 rows × 2 columns

```
sales_data.drop(["merchant_profile_picture"], axis=1, inplace=True)
```

Εικόνα 51 Συσχέτιση κενών τιμών

Αριθμητικές μεταβλητές

- “ rating_one_count, rating_two_count, rating_three_count, rating_four_count, rating_five_count ” : Όλες οι μεταβλητές έχουν πολύ υψηλή συσχέτιση με τη μεταβλητή “rating_count” όπως φάνηκε και στην ενότητα με τις οπτικοποιήσεις. Έχουν 45 κενές τιμές η κάθε μία και αποδεικνύεται ότι τα προϊόντα με ελλιπή πλήθος αξιολογήσεων 1,2,3,4 και 5 αστεριών δεν έχουν καθόλου βαθμολογίες (rating_count=0). Επιπλέον η τιμή της στήλης “rating” για αυτά τα προϊόντα είναι 5 η οποία είναι απίθανο δεδομένου ότι τα προϊόντα έχουν μηδενική βαθμολογία. Επομένως αυτό ίσως να συνέβη από κάποιο λάθος. Η τιμή 5 της μεταβλητής “rating” αντικαθίσταται σε 0 και οι ελλιπείς ποσότητες με 0.



Εικόνα 52 Συσχέτιση κενών τιμών μεταξύ των "rating"

3.2.2 Χειρισμός των ακραίων τιμών

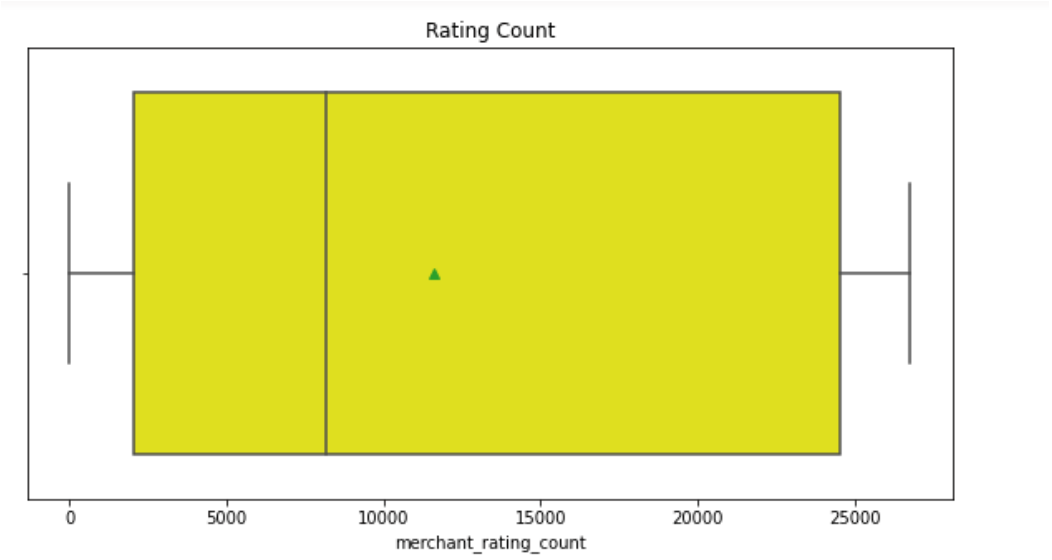
Από την ενότητα 3.1 στη μεταβλητή “ merchant_rating_count” υπήρχαν αρκετές ακραίες τιμές πάνω απτό Q3 για την ακρίβεια 384. Όλες οι ακραίες τιμές που είναι πάνω απ’ το Q3= 24564 αντικαθίσταται με τη μέση τιμή 26495 όπως φαίνεται στη παρακάτω εικόνα.

```
outliers=sales_data[sales_data["merchant_rating_count"]>24564]
outliers['merchant_rating_count'].count()
```

```
384
```

```
mean = sales_data['merchant_rating_count'].mean()
sales_data['merchant_rating_count'] = sales_data['merchant_rating_count'].apply(lambda x: mean if x>24564 else x)
sns.boxplot(sales_data['merchant_rating_count'],color='yellow',showmeans=True)
fig = plt.gcf()
fig.set_size_inches(10,5)
plt.title('Rating Count')
```

Εικόνα 53 Κώδικας χειρισμού ακραίων τιμών "merchant_rating_count"



Εικόνα 54 Boxplot "merchant_rating_count"

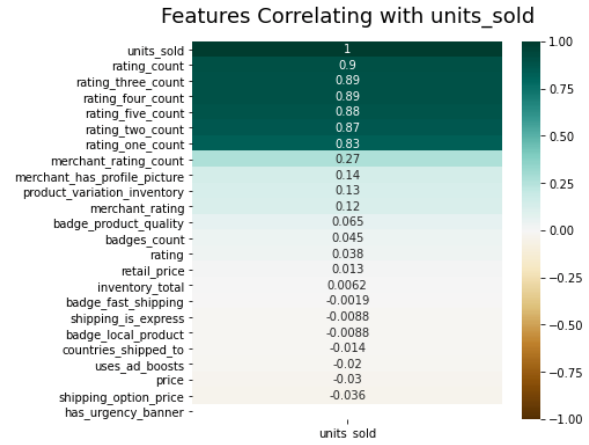
Στο παραπάνω boxplot φαίνεται να έχει διορθωθεί το πρόβλημα ακραίων τιμών της μεταβλητής “merchant_rating_count”.

3.2.3 Δημιουργία νέων στηλών

Από τον παρακάτω πίνακα συσχέτισης ο αριθμός αξιολογήσεων από ένα έως πέντε που συμβάλλει σημαντικά στις πωλήσεις δεν ακούγεται σωστά από την κοινή λογική. Ένα προϊόν με κακές βαθμολογίες όπως το αστέρι 1 δεν είναι πιθανόν να πουλήσει καλά. Πιθανότατα η σύγκριση κάθε είδους βαθμολογίας με το συνολικό αριθμό βαθμολογιών είναι πιο αξιόπιστη.

```
target_col='units_sold'
```

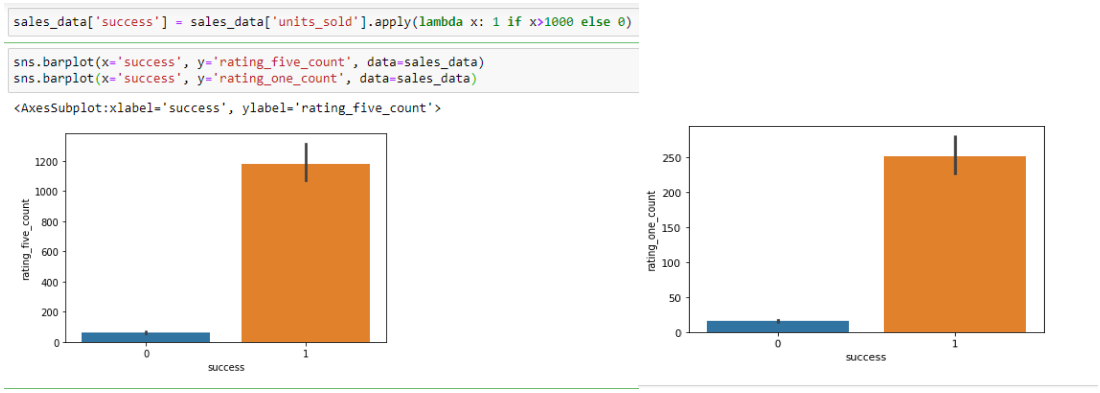
```
plt.figure(figsize=(6, 6))
heatmap = sns.heatmap(sales_data.corr()[[target_col]].sort_values(by=target_col, ascending=False), vmin=-1, vmax=1, annot=True
heatmap.set_title(f'Features Correlating with {target_col}', fontdict={'fontsize':18}, pad=16);
```



Εικόνα 55 Heatmap μεταβλητών με το "units_sold"

Για να συνεχισθεί η ανάλυση σημαντικό βήμα είναι η επεξεργασία του “ units_sold” το οποίο θα παίρνει δυαδική μορφή εφόσον και το πρόβλημα μηχανικής μάθησης που θα αναλυθεί μετέπειτα είναι classification. Για τον όρο classification θα γίνει αναφορά στην ενότητα ML. Επομένως το “units_sold” μετατρέπεται σε binary μεταβλητή (επιτυχής ή μη). Ο διαχωρισμός στα όρια των ποσοτήτων γίνεται με βάση την διάμεσο του “units_sold” που ισούται με 1000. Θεωρείται ότι τα προϊόντα που πωλούνται και είναι κάτω από 1000 δεν είναι τόσο επιτυχημένα ενώ αυτά που είναι πάνω από 1000 είναι περισσότερο επιτυχημένα. Η νέα στήλη ονομάζεται “ success” με τιμές 0 και 1. Η διάμεσος δεν επηρεάζεται από υψηλά outliers.

Σχετικά με το κομμάτι των πιο χαμηλών βαθμολογιών με αστέρι 1 και των αντίστοιχων υψηλών πωλήσεων αναπαρίσταται ένα barplot. Συγκρίνονται η επιτυχία πωλήσεων για τις βαθμολογίες με 1 αστέρι και αυτές με 5.



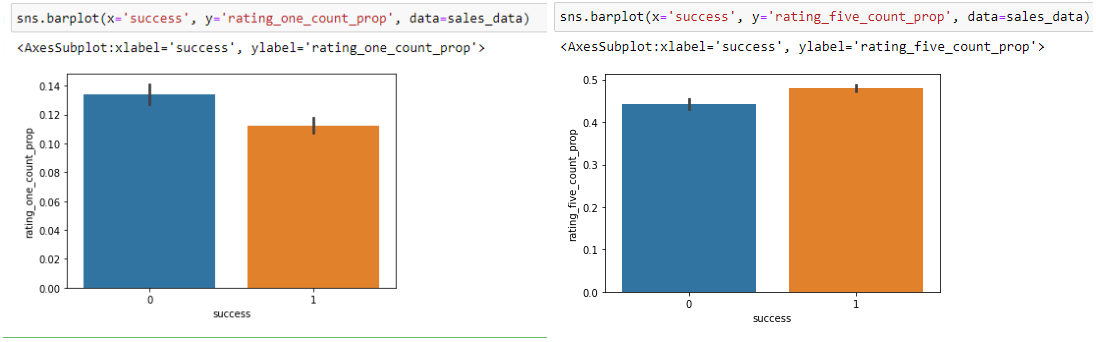
Εικόνα 56 Barplot πλήθους "rating_five_count" & "rating_one_count" ανά "units_sold"

Σύμφωνα με όσα αναφέρθηκαν παραπάνω δημιουργούνται 5 νέες στήλες “rating_one_count_prop, rating_two_count_prop, rating_three_count_prop, rating_four_count_prop, rating_five_count_prop “ που δείχνουν τον αριθμό των βαθμολογιών 1 έως 5 ως προς το συνολικό πλήθος βαθμολογιών. Παρακάτω βρίσκεται και ο κώδικας δημιουργίας των νέων μεταβλητών.

```
sales_data['rating_three_count_prop'] =sales_data['rating_three_count']/sales_data['rating_count']
sales_data['rating_four_count_prop'] = sales_data['rating_four_count']/sales_data['rating_count']
sales_data['rating_five_count_prop'] = sales_data['rating_five_count']/sales_data['rating_count']
sales_data['rating_two_count_prop'] = sales_data['rating_two_count']/sales_data['rating_count']
sales_data['rating_one_count_prop'] = sales_data['rating_one_count']/sales_data['rating_count']
# to remove nan due to zero division
sales_data.update(sales_data[['rating_five_count_prop', 'rating_four_count_prop', 'rating_three_count_prop','rating_two_count_pro
```

Εικόνα 57 Κώδικας δημιουργίας νέων στηλών "rating_count_prop"

Τέλος με τα παρακάτω barplot των νέων στηλών φαίνεται ότι το πρόβλημα αποκαθίσταται.



Εικόνα 58 Barplot πλήθους "rating_five_count_prop" & "rating_one_count_prop" ανά "units_sold"

3.2.4 Κωδικοποίηση μεταβλητών

Σε αυτό το βήμα γίνεται ξεκαθάρισμα του dataset για το ποιες τιμές θα μείνουν. Η κίνηση αυτή γίνεται με βάση το πίνακα συσχέτισης και κατά πόσο οι μεταβλητές συσχετίζονται με τη στήλη πρόβλεψης που είναι το units_sold. Παρακάτω φαίνονται οι στήλες που αφαιρούνται με την συνάρτηση .drop()

```
sales_data = sales_data.drop(['crawl_month','product_id','product_picture', 'product_url','merchant_id',
                     'currency_buyer', 'theme','merchant_title',
                    'title','title_orig','tags', 'shipping_option_name', "inventory_total" , "badge_fast_shipping" ,
                     "badge_local_product" , "shipping_is_express", "units_sold","rating_five_count","rating_four_count",
                           "rating_three_count", "rating_two_count", "rating_one_count"], axis = 1)
```

```
# Drop unnecessary columns
to_drop = ['product_color', 'product_variation_size_id', 'origin_country']
sales_data.drop(to_drop, axis=1, inplace=True)
```

Εικόνα 59 Κώδικας αφαίρεσης μεταβλητών

Στη συνέχεια χρησιμοποιείται η μέθοδος του encoding των μεταβλητών όπου όλες οι τιμές μετατρέπονται σε αριθμητικές. Η μέθοδος encoding είναι υποχρεωτική στο στάδιο αυτό καθώς οι αλγόριθμοι Machine Learning δέχονται μόνο αριθμητικές εισόδους επομένως όσες μεταβλητές είναι κατηγορικές μετατρέπονται σε αριθμητικές με χρήση κωδικοποίησης. Οι τεχνική που χρησιμοποιείται είναι η " One Hot Coding " η οποία είναι η πιο διαδεδομένη μορφή κωδικοποίησης. Μετατρέπει μια κατηγορική μεταβλητή με n παρατηρήσεις και d διακριτές τιμές σε d δυαδικές μεταβλητές με n παρατηρήσεις η κάθε μία και με τιμές 1 και 0 αντίστοιχα. (Potdar, S. and D., 2017) Οι μεταβλητές που μετατρέπονται σε δυαδικές είναι οι "product_color, product_variation_size_id, origin_country ". Με τη συνάρτηση .get_dummies της βιβλιοθήκης Pandas γίνεται η μετατροπή όπως φαίνεται και στην παρακάτω εικόνα. Αφού αφαιρεθούν οι αρχικές κατηγορικές μεταβλητές , προκύπτει το τελικό σύνολο δεδομένων για το κομμάτι του Machine Learning με την ονομασία "sales_data_new".

```
In [52]: dummy_product_color = pd.get_dummies(sales_data['product_color'], prefix='color_', drop_first=True)
         dummy_size = pd.get_dummies(sales_data['product_variation_size_id'], prefix='size_', drop_first=True)
         dummy_country = pd.get_dummies(sales_data['origin_country'], prefix='country_', drop_first=True)
```

Εικόνα 60 Κώδικας One Hot Coding

3.3 Machine Learning – Δημιουργία Μοντέλων

Η μηχανική μάθηση ανήκει στην οικογένεια της τεχνητής νοημοσύνης. Θεωρείται ως μία μηχανή υπολογιστή που μιμείται την ανθρώπινη συμπεριφορά. Μέσα από την επαναλαμβανόμενη εκπαίδευση αλγορίθμων οι οποίοι τροφοδοτούνται με ένα μεγάλο όγκο δεδομένων δημιουργείται ένα μοντέλο που κάνει προβλέψεις και παίρνει αποφάσεις. Τα μοντέλα μπορεί να είναι “descriptive”, “predictive” και “ prescriptive”. Τα μοντέλα εκπαίδευσης στην παρούσα εργασία ανήκουν στη κατηγορία “predictive” με στόχο η χρήση των δεδομένων να προβλέπει τί θα συμβεί. Υπάρχουν τρεις υποκατηγορίες M.M με τα ονόματα επιβλεπόμενη (supervised), μη επιβλεπόμενη(unsupervised) και ενισχυτική (Reinforcement). (“Machine Learning, Explained _ MIT Sloan,” n.d.)

Η εργασία αναφέρεται στη κατηγορία επιβλεπόμενης μάθησης. Η μέθοδος αυτή χρησιμοποιεί δεδομένα με ετικέτες, δεδομένα δηλαδή με προκαθορισμένα χαρακτηριστικά. Δημιουργεί μία συνάρτηση όπου απ’ το σύνολο παραδειγμάτων των δεδομένων εκπαίδευσης κάθε παράδειγμα εισόδου αντιστοιχίζεται σε ένα παράδειγμα εξόδου. Η είσοδος θεωρείται η κάθε τιμή των ανεξάρτητων μεταβλητών και η έξοδος η κάθε τιμή της εξαρτημένης μεταβλητής που είναι και το “ target ” της πρόβλεψης. Η συνάρτηση που παράγεται λειτουργεί και για τη χαρτογράφηση νέων παραδειγμάτων. (‘Supervised learning - Wikipedia’, no date)

Στο κομμάτι της επιβλεπόμενης M.M δημιουργούνται 7 διαφορετικά μοντέλα και ο κώδικας που χρησιμοποιείται για αυτά αναφέρεται στο παράρτημα “ Γ ”. Το πρόβλημα είναι δυαδικό ταξινόμησης (binary classification). Αυτό σημαίνει ότι οι εξαρτημένη μεταβλητή είναι κατηγορική και περιέχει 2 κλάσεις με τιμές “0” και “1” και κάθε ετικέτα κλάσης προβλέπεται για κάθε παράδειγμα των δεδομένων εισόδου. (‘4 Types of Classification Tasks in Machine Learning’, no date)

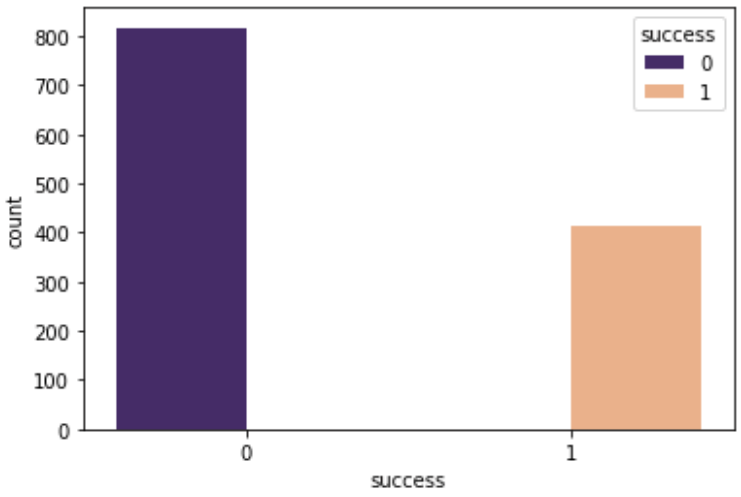
Παρακάτω αναφέρονται τα ονόματα των μοντέλων που επιλέγονται.

- Νευρωνικά Δίκτυα MLP (Multilayer Perceptron)
- KNN (K-nearest neighbors Classifier)

- Decision Tree Classifier
- Logistic Regression
- Gaussian NB
- Random Forest Classifier
- XG Boost Classifier

Πριν την εκπαίδευση των μοντέλων το τελικό dataset χωρίζεται σε X και y όπου X το σύνολο δεδομένων με τις ανεξάρτητες μεταβλητές και y με την εξαρτημένη μεταβλητή “success” που δηλώνει αν τα προϊόντα της σελίδας θα έχουν επιτυχία ή όχι στις πωλήσεις. Στη συνέχεια γίνεται η διαδικασία διαχωρισμού (split) των train- test set σε αναλογία 80% και 20% αντίστοιχα. Το train set περιέχει τα δεδομένα που θα τροφοδοτήσουν το μοντέλο για να μπορέσει να εκπαιδευτεί. Στη συνέχεια το μοντέλο αξιολογείται για την ακρίβεια του χρησιμοποιώντας το test set. Ο λόγος που συμβαίνει αυτό είναι γιατί αν γινόταν χρήση ολόκληρου του dataset στη προσαρμογή του μοντέλου τότε το μοντέλο θα ταίριαζε υπερβολικά στα δεδομένα και θα οδηγούσε σε κακές προβλέψεις λόγω υπερβολικής υπερ. προσαρμογής. (Joseph and Vakayil, 2021)

Επιπλέον στο σύνολο εκπαίδευσης της εξαρτημένης μεταβλητής “ success” παρατηρείται ότι η μία κατηγορία με τη τιμή “0” αντιπροσωπεύεται περισσότερο από την άλλη με τη τιμή “1” επομένως προκύπτει ότι υπάρχει ανισορροπία στα δεδομένα. Αυτό μπορεί να προκαλέσει προβλήματα με την απόδοση των μοντέλων σε ένα πρόβλημα ταξινόμησης. Παρακάτω φαίνεται και ένα γράφημα bar plot με την εξαρτημένη μεταβλητή στο σύνολο εκπαίδευσης με την ανισορροπία αυτή.



Εικόνα 61 Barplot Πλήθους κλάσεων εξαρτημένης μεταβλητής "success" στο σύνολο εκπαίδευσης

Η παραπάνω ανισορροπία δεδομένων θα αντιμετωπιστεί μέσω της υπερβολικής δειγματοληψίας (oversampling) με τη χρήση της βιβλιοθήκης SMOTE η οποία μέθοδος λειτουργεί διαφορετικά από την τυπική υπερ δειγματοληψία. Σε μία κλασική τεχνική oversampling τα δεδομένα της μειοψηφίας αντιγράφονται από τον πληθυσμό των μειονοτικών δεδομένων και δεν δίνει νέες πληροφορίες στο μοντέλο M.M. Η SMOTE λειτουργεί χρησιμοποιώντας έναν αλγόριθμο K-nearest-neighbor για τη δημιουργία συνθετικών δεδομένων. Επιλέγονται τυχαία δεδομένα από τη τάξη μειονότητας και μετά ορίζονται οι κ-πλησιέστεροι γείτονες απ' τα δεδομένα. Συνθετικά δεδομένα θα δημιουργηθούν μεταξύ των τυχαίων δεδομένων και του τυχαίου επιλεγμένου γείτονα. (Wijaya, 2020) Μετά από τη διαδικασία oversampling γίνεται εξισορρόπηση των κλάσεων σε αναλογία κλάσης 0: 817 παρατηρήσεις και κλάσης 1: 817 παρατηρήσεις. Δημιουργούνται 403 ψεύτικες παρατηρήσεις ακύρωσης για το σετ εκπαίδευσης οπότε το train set αυξάνεται κατά 403 παρατηρήσεις και φτάνει απ' τις 1231 στις 1634. Το SMOTE εφαρμόζεται μόνο στο σετ εκπαίδευσης και όχι στο σετ ελέγχου γιατί στόχος είναι το μοντέλο να προβλέπει πραγματικά δεδομένα και όχι συνθετικά. Στη συνέχεια ακολουθεί ο τρόπος υλοποίησης του κάθε μοντέλου. Κάθε μοντέλο εκπαιδεύεται με και χωρίς oversampling έτσι ώστε να φανούν στο τέλος οι διαφορές στην απόδοση.

Σε όλους τους αλγόριθμους γίνεται η χρήση του pipeline που είναι συνάρτηση της sklearn και συγκεντρώνει ένα σύνολο βημάτων στην εκπαίδευση των μοντέλων για τη βέλτιστη απόδοση. Μέσα στο pipeline λαμβάνει χώρα το scaling με τη συνάρτηση StandardScaler και ο κάθε αλγόριθμος με παράμετρο το random_state η οποία τοποθετείται στο SMOTE και στο split επίσης.

Η μέθοδος StandardScaler χρησιμοποιείται γιατί τα χαρακτηριστικά του συνόλου δεδομένων διαφέρουν κατά πολύ μεταξύ τους και οι τιμές μετρούνται σε διαφορετικές μονάδες μέτρησης. Συνεπώς γίνεται μία κλιμάκωση των δεδομένων καθώς η μεγάλη διαφορά στα χαρακτηριστικά μπορεί να προκαλέσει προβλήματα στον υπολογισμό των μοντέλων όπως να υπερπροσαρμοστούν και να βγάλουν λανθασμένα αποτελέσματα. Έτσι οι τιμές επικεντρώνονται γύρω από το μέσο όρο με τυπική απόκλιση μονάδας και ο μέσος όρος γίνεται μηδέν και η προκύπτουσα κατανομή έχει τυπική απόκλιση μονάδας. (‘Bias-Variance in Machine Learning’, no date)

Το random_state είναι μία υπερπαράμετρος που χρησιμοποιείται για τον έλεγχο της τυχαιότητας στα μοντέλα μηχανικής μάθησης. Με την τιμή 42 που έχει τοποθετηθεί στη παράμετρο επιτρέπεται να διασπαστούν τα δεδομένα εκπαίδευσης και ελέγχου σε όχι τυχαία σειρά. Έτσι κάθε φορά που τρέχει ένας αλγόριθμος θα προκύπτει το ίδιο αποτέλεσμα. Αν δεν χρησιμοποιείται το randomstate κάθε φορά που γίνεται διαχωρισμός του συνόλου δεδομένων σε train και test ενδέχεται να ληφθεί διαφορετικό αποτέλεσμα συνεπώς δεν θα βοηθήσει στον εντοπισμό σφαλμάτων σε περίπτωση που αντιμετωπιστούν προβλήματα στον αλγόριθμο. (‘Why do we set a random state in machine learning models_ _ by Rukshan Pramoditha _ Apr, 2022 _ Towards Data Science’, no date)

Στη συνέχεια ορίζεται ένα dictionary με παραμέτρους για κάθε μοντέλο οι οποίες βοηθούν στη καλύτερη απόδοση κατά την εκπαίδευση. Για αυτή τη λειτουργία γίνεται χρήση της συνάρτησης GridSearchCV από το πακέτο model_selection του Scikit-learn η οποία αυτοματοποιεί το συντονισμό υπερπαραμέτρων προκειμένου να καθοριστούν οι βέλτιστες τιμές για την καλύτερη απόδοση του κάθε μοντέλου. Η συνάρτηση GridSearchCV δοκιμάζει όλους τους πιθανούς συνδυασμούς των τιμών που έχουν περάσει στο dictionary και αξιολογεί το μοντέλο για κάθε συνδυασμό χρησιμοποιώντας τη μέθοδο Cross-Validation. (Mujtaba, 2020)

Η μέθοδος Cross-Validation επιλέγεται για την ακρίβεια απόδοσης των μοντέλων Μ.Μ. Είναι απαραίτητη για την αποτροπή της υπερ. προσαρμογής των μοντέλων ειδικά όταν ο όγκος των δεδομένων δεν είναι τόσο μεγάλος. Επομένως γίνεται η χρήση της μεθόδου “ Stratified K Fold Cross Validation “. Η μέθοδος αυτή εγγυάται ότι το σκορ του μοντέλου δεν εξαρτάται από το τρόπο που επιλέχτηκε το train

και test set. Από τις κ=3 πτυχές που έχουν οριστεί, τα σετ K-1=2 χρησιμοποιούνται για εκπαίδευση ενώ το υπόλοιπο σύνολο για δοκιμές. Ο αλγόριθμος εκπαιδεύεται και δοκιμάζεται κ=3 φορές, κάθε φορά που ένα νέο σύνολο χρησιμοποιείται ως σύνολο δοκιμής ενώ τα υπόλοιπα χρησιμοποιούνται για εκπαίδευση. Τέλος τα αποτελέσματα είναι ο μέσος όρος των αποτελεσμάτων που λαμβάνονται σε κάθε σετ. Για να μην υπάρχει ανισορροπία στην επιλογή των πτυχών με το πλήθος των κλάσεων λαμβάνει χώρα το «Stratified K Fold Cross Validation" 'έτσι ώστε σε κάθε πτυχή, κάθε κλάση να περιλαμβάνει περίπου τις μισές παρουσίες. (Great Learning Team, 2020)

Σε όλα τα μοντέλα έχουν τοποθετηθεί τα ίδια ορίσματα στην συνάρτηση Grid Search αλλά φυσικά με υπερπαραμέτρους διαφορετικές για κάθε αλγόριθμο. Παρακάτω αναφέρονται τα ορίσματα αυτά πιο αναλυτικά.

- pipe: είναι η τιμή που έχει οριστεί στο κώδικα για το Pipeline που περιέχει τη μέθοδο StandardScaler και το κάθε μοντέλο.
- params: είναι η τιμή του dictionary που έχει οριστεί στο κώδικα με το σύνολο υπερπαραμέτρων για το κάθε μοντέλο.
- n_jobs=-1: ο αριθμός διεργασιών που εκτελούνται παράλληλα για αυτή την εργασία. Η τιμή -1 δηλώνει ότι χρησιμοποιούνται όλοι οι διαθέσιμοι επεξεργαστές.
- cv=3: αριθμός διασταυρωμένης επικύρωσης που πρέπει να δοκιμαστεί για κάθε επιλεγμένο σύνολο υπερπαραμέτρων. Με την τιμή 3 δηλώνεται ο αριθμός 3 πτυχών (Folds) με τη μέθοδο " Stratified K Fold Cross Validation ".
- verbose=10: δείχνει λεπτομερώς την εκτύπωση ενώ προσαρμόζονται τα δεδομένα στο GridSearchCV
- scoring= " accuracy " : μέτρηση αξιολόγησης

Αφού γίνεται η επιλογή των ορισμάτων γίνεται η εκπαίδευση κάθε μοντέλου πάνω στο σετ εκπαίδευσης που επιλέχτηκε. Έπειτα εκτυπώνονται οι βέλτιστες υπερπαράμετροι του καλύτερου μοντέλου και ο μέσος όρος όλων των πτυχών του cross validation για ένα μόνο συνδυασμό παραμέτρων με τη μετρική "accuracy". Αφού επιλεγεί το καλύτερο μοντέλο με τις βέλτιστες υπερπαραμέτρους του, γίνεται η

πρόβλεψη και η αξιολόγηση απόδοσης. Οι μετρήσεις που λαμβάνουν χώρα στην αξιολόγηση απόδοσης του κάθε μοντέλου είναι :

- Classification Report: Πίνακας που παρέχει καλύτερη κατανόηση της συνολικής απόδοσης του εκπαιδευόμενου μοντέλου και χρησιμοποιείται σε προβλήματα ταξινόμησης. Αποτελείται από τα παρακάτω.
  - Precision : Ποσοστό σωστών θετικών προβλέψεων σε σχέση με τις συνολικές θετικές προβλέψεις.
  - Recall : Ποσοστό σωστών θετικών προβλέψεων σε σχέση με το σύνολο των πραγματικών θετικών.
  - F1 Score: Αρμονικός μέσος του Precision και του Recall. Όσο πιο κοντά στη τιμή 1 τόσο καλύτερη η απόδοση του μοντέλου.
  - Support : Το πλήθος των κλάσεων με τη τιμή 1 και 0 αντίστοιχα στο σύνολο δοκιμής. (‘How to Interpret the Classification Report in sklearn (With Example) - Statology’, no date)
  - Accuracy : Ποσοστό σωστών προβλέψεων ως προς το συνολικό ποσοστό προβλέψεων (Google Developers, 2020)

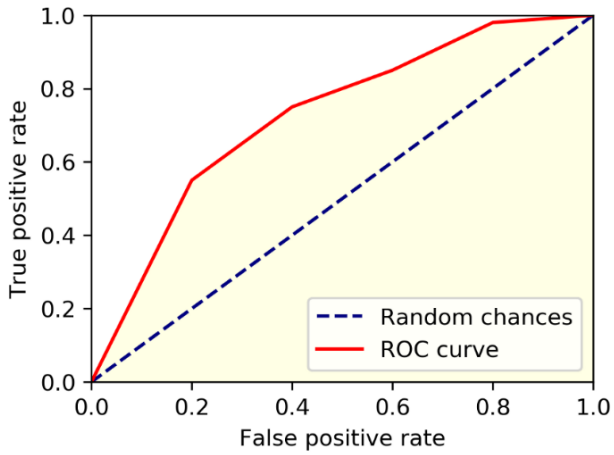
- Γίνεται κλήση και σε κάθε μετρική από τις παραπάνω ξεχωριστά ώστε να γίνει έλεγχος των τιμών και να αποτραπεί οποιοδήποτε σφάλμα.

- Confusion Matrix – Πίνακας Σύγχυσης: Είναι ένας πίνακας με 4 διαφορετικούς συνδυασμούς προβλεπόμενων και πραγματικών τιμών. Αποτελείται από τις τιμές TP, FP, FN, TN οι οποίες εξηγούνται.

  - TP (True Positive): Έγινε πρόβλεψη να είναι θετική κλάση και είναι θετική.
  - TN (True Negative): Έγινε πρόβλεψη να είναι αρνητική κλάση και είναι αρνητική.
  - FP (False Positive): Έγινε πρόβλεψη να είναι θετική η κλάση και είναι αρνητική δηλαδή λάθος.
  - FN (False Negative) : Έγινε πρόβλεψη να είναι αρνητική η κλάση και είναι θετική δηλαδή λάθος. (Narkhede, 2018b)

- Καμπύλη ROC – AUC Score : Η καμπύλη ROC (Receiver Operating Characteristics) είναι μία καμπύλη πιθανοτήτων που δείχνει πόσο καλά γίνεται η ταξινόμηση κλάσεων σε ένα πρόβλημα ταξινόμησης. Η καμπύλη αποτελείται από τις τιμές που παίρνει το TPR (True Positive Rate) και FPR (False Positive Rate) κάθε φορά για κάθε τιμή του threshold στον άξονα y και χ αντίστοιχα. Όσο το TPR μεγαλώνει και φτάνει στη τιμή 1 και όσο το FPR μικραίνει και φτάνει στη τιμή 0 το μοντέλο έχει καλύτερη απόδοση. Το AUC (Area Under Curve) είναι η μετρική που υπολογίζει την ικανότητα ενός μοντέλου να ταξινομεί σωστά. Στο γράφημα υπολογίζεται ως το εμβαδόν του χώρου που βρίσκεται κάτω από τη καμπύλη ROC. Όσο το σκορ AUC πλησιάζει στο 1 τόσο καλύτερη απόδοση υπάρχει στο μοντέλο ταξινόμησης.

    - TPR: εκφράζει ποια αναλογία της θετικής τάξης ταξινομήθηκε σωστά.
    - FPR: εκφράζει ποια αναλογία της αρνητικής τάξης ταξινομήθηκε λανθασμένα.



Εικόνα 62 Καμπύλη ROC (Narkhede 2018a)

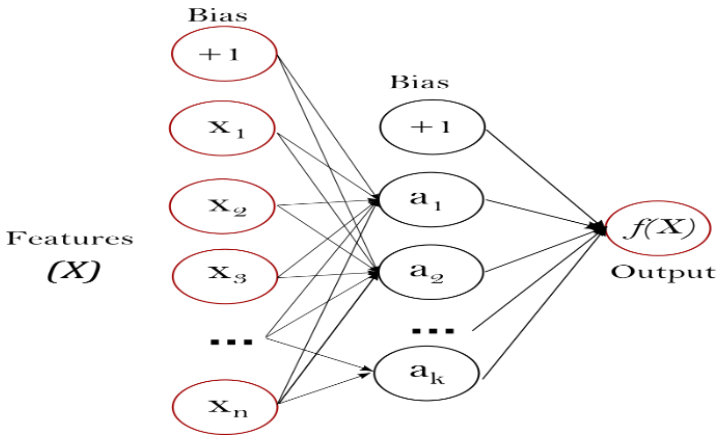
Η διακεκομμένη γραμμή στην εικόνα είναι τυχαίες επιλογές με πιθανότητα 50%. Το εμβαδόν κάτω από τη διακεκομμένη γραμμή είναι 0.5 δηλαδή το AUC=0,5. Όταν συμβαίνει αυτό το μοντέλο δεν έχει καμία ικανότητα διαχωρισμού των κλάσεων. Η τέλεια πρόβλεψη αναπαρίσταται όταν η καμπύλη ROC πάρει ορθή γωνία. (Narkhede, 2018a)

Στις επόμενες υποενότητες γίνεται αναφορά για τη λειτουργία του κάθε μοντέλου και ποιες παράμετροι επιλέχτηκαν για κάθε ένα. Η επιλογή των

παραμέτρων έγινε αυθαίρετα μέσα από την επίσημη σελίδα της Scikit Learn και όπως αναφέρθηκε και παραπάνω με την μέθοδο GridSearchCV η εκπαίδευση και πρόβλεψη των μοντέλων έγινε με το καλύτερο συνδυασμό αυτών των παραμέτρων.

3.3.1 Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα είναι ένα μοντέλο επιβλεπόμενης μάθησης που λειτουργεί σύμφωνα με τη συνάρτηση f(): R^m -> R^O εκπαιδεύοντας ένα σύνολο δεδομένων με "m" πλήθος διαστάσεων για τις μεταβλητές εισόδου και "o" το πλήθος διαστάσεων για τις μεταβλητές εξόδου. Περιέχει επίπεδα και συγκεκριμένα τρία. Το πρώτο είναι το πλήθος των μεταβλητών εισόδου και το τελευταίο είναι το πλήθος των μεταβλητών εξόδου. Ενδιάμεσα υπάρχουν ένα ή περισσότερα μη γραμμικά επίπεδα τα οποία ονομάζονται κρυφά επίπεδα- " hidden layers ".



Εικόνα 63 Νευρωνικά Δίκτυα ('sklearn', no date)

Στη παραπάνω εικόνα το αριστερό στρώμα, γνωστό ως στρώμα εισόδου, αποτελείται από ένα σύνολο νευρώνων $\{x_i|x_1,x_2,..,x_m\}$ που αντιπροσωπεύει τα χαρακτηριστικά εισόδου. Κάθε νευρώνας στο κρυφό στρώμα μετασχηματίζει τις τιμές από το προηγούμενο επίπεδο με ένα σταθμισμένο γραμμικό άθροισμα $w_1x_1+w_2x_2+….+w_mx_m$ ακολουθούμενη από μία μη γραμμική συνάρτηση. Το επίπεδο εξόδου λαμβάνει τις τιμές από το τελευταίο κρυφό στρώμα και τις μετατρέπει σε τιμές εξόδου. ('sklearn', no date)

Για την εκπαίδευση του MLP επιλέγονται οι εξής παράμετροι :

- hidden_layer_sizes : Οι τιμές που πήρε είναι (100,10),(200,10),(50),(100),(200,20),(200,50) . Η κάθε τιμή δηλώνει πόσα hidden layers έχουν οριστεί και πόσοι νευρώνες είναι σε κάθε layer. Δηλαδή για τη τιμή (100,10) ορίζονται 2 hidden layers με 10 και 100 αντίστοιχα νευρώνες.
- activation : Με την τιμή “relu”
- learning_rate_init : Με την τιμή 0.0001 . Ελέγχει το μέγεθος του βήματος στην ενημέρωση των βαρών.
- solver : με την τιμή “adam”

3.3.2 KNN (K-Nearest Neighbors Classifier)

Ο KNN είναι μοντέλο επιβλεπόμενης μάθησης που χρησιμοποιείται σε προβλήματα παλινδρόμησης αλλά κυρίως ταξινόμησης και λειτουργεί με συνάρτηση απόστασης. Κάθε σημείο ανήκει στην κλάση με την οποία μοιράζεται τον περισσότερο αριθμό κοινών σημείων ως προς την απόστασή του. Ένα σημείο προσδιορίζεται σε ποια κλάση ανήκει από την τιμή K. Η τιμή K δηλώνει το πλήθος των γειτόνων που πρέπει να ελεγχθούν ώστε το σημείο να ταξινομηθεί σε κλάση. Έτσι ταξινομείται στη πλειοψηφία των κ πλησιέστερων γειτόνων. Η παρακάτω εικόνα αναπαριστά τον τρόπο λειτουργίας του KNN. (Raschka, 2018)

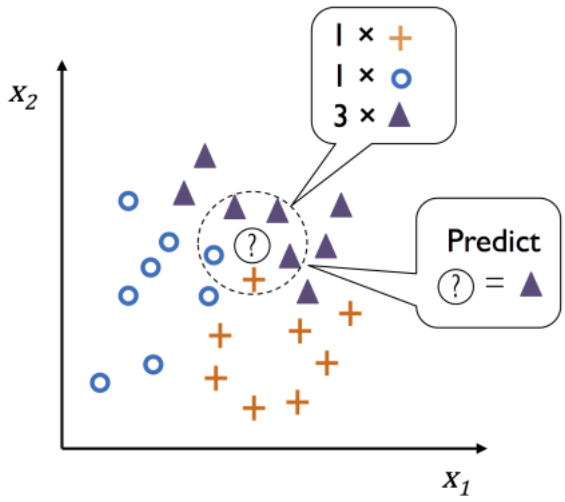


Figure 5: Illustration of *k*NN for a 3-class problem with k=5.

Εικόνα 64 K-Nearest Neighbors Classifier (Raschka, 2018)

Για την εκπαίδευση του KNN επιλέγεται η παράμετρος :

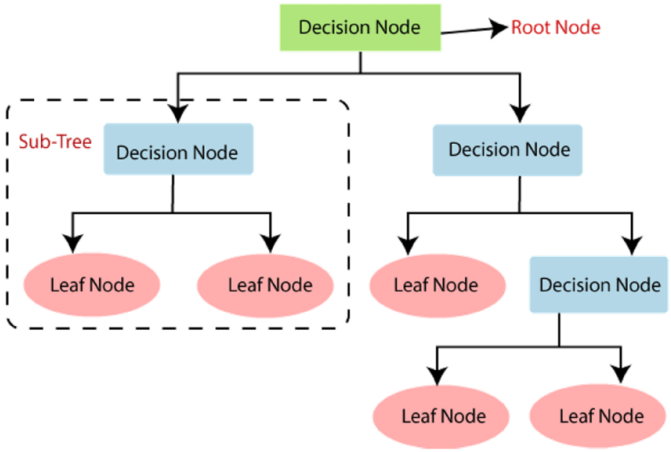
- n_neighbors : Με τιμές [7,9,11,15,20,25,30,35] . Η κάθε τιμή δηλώνει τον αριθμό κ πλησιέστερων γειτόνων κάθε φορά.

3.3.3 Decision Tree Classifier

Το δέντρο απόφασης είναι ένα μοντέλο επιβλεπόμενης μάθησης που βασίζεται σε μοντέλα ταξινόμησης και παλινδρόμησης. Το όνομα του προκύπτει απ' τον τρόπο λειτουργίας του για τα προβλήματα ταξινόμησης καθώς μοιάζει με τη δομή ενός δέντρου. (Charbuty and Abdulazeez, 2021)

Αποτελείται από δύο είδη κόμβων και κλάδους που ενώνουν τους κόμβους αυτούς, το κόμβο απόφασης (Decision Node) και το κόμβο- φύλλου (Leaf Node). Κάθε δέντρο απόφασης ξεκινάει με πρώτο κόμβο το κόμβο-ρίζα (Root node) που αντιπροσωπεύει όλο το σύνολο δεδομένων και χωρίζεται σε δύο ή περισσότερα ομοιογενή σύνολα δέντρων (Sub Tree). Κάθε σύνολο περιέχει κόμβους απόφασης που αντιπροσωπεύουν τα χαρακτηριστικά (features) του συνόλου δεδομένων και λειτουργούν για τη λήψη οποιασδήποτε απόφασης και έχουν πολλαπλούς κλάδους.

Τέλος, οι κόμβοι- φύλλα (Leaf Node) είναι τα τελικά αποτελέσματα και το δέντρο δεν μπορεί να διαχωριστεί περισσότερο μετά τη λήψη ενός Leaf Node. (Hruthik *et al.*, 2022)



Εικόνα 65 Decision Tree Classifier ('Decision-tree-Source-https-pianalytixcom-decision-tree-algorithm-Full-size-DOI', no date)

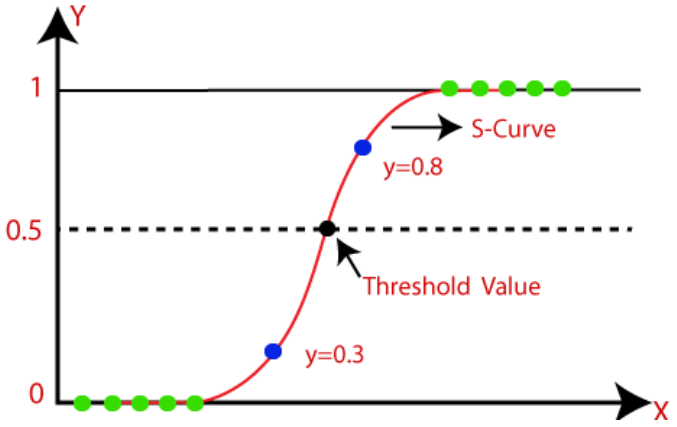
Κατά τη χρήση ενός Δ.Α η βασική λειτουργία είναι να βρεθεί με βάση ποιο παράγοντα θα αποφασιστεί ποιο χαρακτηριστικό είναι ο καλύτερος ταξινομητής σε κάθε επίπεδο κόμβου. Για να βρεθεί η αξία κάθε κόμβου υπάρχουν στατιστικά μέτρα που υπολογίζονται σε κάθε αντίστοιχο κόμβο. Κάποια απ' τα μέτρα αυτά είναι το κέρδος πληροφοριών (information gain), ο δείκτης Gini (Gini index), το Chi-square και η εντροπία (entropy) . (Alzubi, Nayyar and Kumar, 2018)

Για την εκπαίδευση του Δ.Α επιλέγονται οι παράμετροι :

- Criterion : Είναι η λειτουργία για τη ποιότητα του διαχωρισμού του Δ.Α. Οι τιμές – κριτήρια που πήρε είναι οι " gini", "entropy" .
- Splitter : Η στρατηγική για την επιλογή του διαχωρισμού σε κάθε κόμβο. Οι τιμές που πήρε είναι "best", "random".
- Max_depth : Το μέγιστο βάθος του δέντρου. Οι τιμές που πήρε είναι " none", "100","50", "20", "10", "5". ('sklearn', no date)

3.3.4 Logistic Regression

Η μέθοδος λογιστικής παλινδρόμησης είναι μία απ' τις πιο γνωστές μεθόδους προβλημάτων ταξινόμησης με λειτουργία να προβλέπει την έξοδο μίας εξαρτημένης μεταβλητής δυαδικής μορφής. Στο μοντέλο Logistic Regression προσαρμόζεται συνάρτηση που προβλέπει τις μέγιστες τιμές 0 ή 1. Η συνάρτηση αυτή αναπαρίσταται με μία σιγμοειδή καμπύλη που δηλώνει την πιθανότητα η μεταβλητή y να παίρνει την τιμή 0 ή 1. Αντιστοιχίζει οποιαδήποτε πραγματική τιμή σε τιμή εντός του εύρους 0 κ 1. Η παραπάνω λειτουργία καθορίζεται από τη ποσότητα που θα έχει το threshold. Οι τιμές πάνω απ' την ποσότητα κατωφλιού που έχει οριστεί τείνουν στο 1 και αντίστοιχα κάτω απ' την ποσότητα κατωφλιού στο 0. (Java Point, 2018)



Εικόνα 66 Logistic Regression (Java Point, 2018)

Για την εκπαίδευση του L.R επιλέγεται η παράμετρος :

- C: δηλώνει το αντίστροφο της ισχύος κανονικοποίησης. Είναι ένας ρυθμιστής κατά της υπερπροσαρμογής. Μικρότερες τιμές δίνουν λιγότερο βάρος στα δεδομένα και μεγαλύτερο στη ποινή πολυπλοκότητας με αποτέλεσμα την αποφυγή του overfitting. ('sklearn', no date) Οι τιμές που πήρε είναι { 0.001, 0.01, 0.1, 1, 10, 100, 1000 }

3.3.5 Gaussian NB

Το μοντέλο Gaussian Naive Bayes ακολουθεί κανονική κατανομή και στηρίζεται στην θεωρία Bayes για προβλήματα ταξινόμησης. Ο τύπος από το θεώρημα του Bayes δίνεται από την παρακάτω εικόνα.

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Εικόνα 67 Τύπος Θεωρήματος Bayes (‘naive-bayes-classifier-algorithm’, no date)

- P(A|B): πιθανότητα της υπόθεσης A στο παρατηρούμενο γεγονός B.
- P(B|A): πιθανότητα του B δεδομένου ότι η πιθανότητα του A είναι αληθής.
- P(A) : πιθανότητα υπόθεσης A πριν την παρατήρηση των αποδεικτικών στοιχείων
- P(B): πιθανότητα του γεγονότος B. (Javatpoint, 2011)

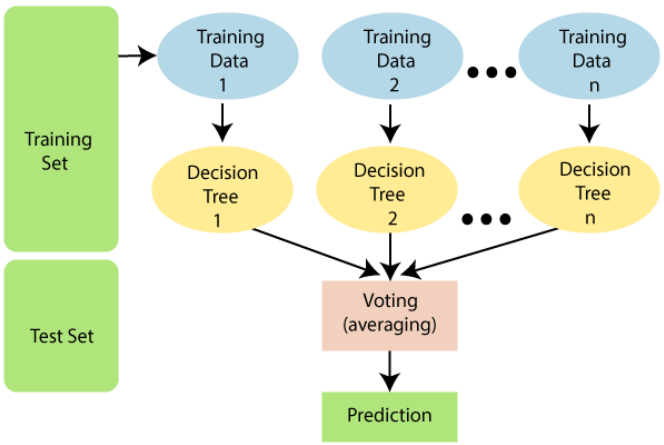
Το όνομα Naive υποδηλώνει την ανεξαρτησία των χαρακτηριστικών που ενσωματώνονται στον αλγόριθμο. Δηλαδή, αλλαγές που γίνονται στη τιμή ενός χαρακτηριστικού δεν επηρεάζουν τα υπόλοιπα. (‘Gaussian Naive Bayes_ Τι πρέπει να γνωρίζετε; _ upGrad blog’, no date)

Για την εκπαίδευση του μοντέλου επιλέγονται οι παράμετροι :

- Priors: Είναι οι πιθανότητες των κλάσεων 0 και 1. Η τιμή που πήρε είναι το “None”.
- var_smoothing : Προστίθεται μία τιμή στη διακύμανση της διανομής με σκοπό να εξομαλύνει την καμπύλη για σταθερότητα υπολογισμού. Οι τιμές που πήρε είναι { 0.00000001, 0.000000001, 0.0000000001} (‘sklearn’, no date)

3.3.6 Random Forest Classifier

Το μοντέλο Random Forest Classifier λειτουργεί σε προβλήματα ταξινόμησης και παλινδρόμησης. Βασίζεται στη μέθοδο εκμάθησης συνόλου δηλαδή στην χρήση πολλαπλών ταξινομητών για την βέλτιστη απόδοση του μοντέλου. Αποτελείται από ένα σύνολο πολλών δέντρων απόφασης. Κάθε δέντρο απόφασης εκπαιδεύει ένα υποσύνολο του συνόλου δεδομένων και ο μέσος όρος αυτών βελτιώνει την ακρίβεια απόδοσης του συνόλου δεδομένων. Κρατάει την πλειοψηφία των προβλέψεων και προβλέπει την τελικό αποτέλεσμα. Όσα περισσότερα δέντρα αποφάσεων τόσο καλύτερη απόδοση του μοντέλου και αποφυγή υπερπροσαρμογής. Η παρακάτω εικόνα αναπαριστά την λειτουργία του αλγόριθμου. (Javapoint, 2021)



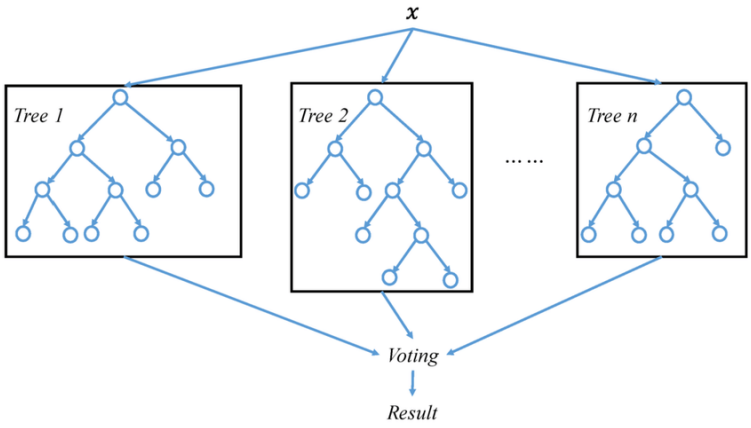
Εικόνα 68 Random Forest Classifier ("Sklearn," n.d.)

Για την εκπαίδευση του μοντέλου επιλέγονται οι παράμετροι :

- n_estimators : Ο αριθμός των δέντρων στο μοντέλο. Οι τιμές που πήρε είναι {200,500}
- max_features : Ο αριθμός των χαρακτηριστικών για το καλύτερο διαχωρισμό. Οι τιμές που πήρε είναι { "auto", "sqrt", "log2" }
- max_depth : Το μέγιστο βάθος του δέντρου. Οι τιμές που πήρε είναι {4,5,6,7,8}
- criterion : λειτουργία ποιότητας διαχωρισμού. Οι τιμές που πήρε είναι {"gini","entropy"}. ('sklearn', no date)

3.3.7 XGBoost Classifier

Ο αλγόριθμος εφαρμόζεται καλώντας τη βιβλιοθήκη XGBoost της Python που παρουσιάστηκε πρώτη φορά στο πανεπιστήμιο της Ουάσιγκτον. Βασίζεται στον αλγόριθμο Gradient boosting που προβλέπει με ακρίβεια μία μεταβλητή στόχο συνδυάζοντας τις εκτιμήσεις ενός συνόλου απλούστερων δενδροειδών μοντέλων. Συνήθως τα μοντέλα αυτά είναι μία σειρά από δέντρα απόφασης και τυχαίων δασών. ('XGBoost ML Model in Python - Javatpoint', no date)



Εικόνα 69 XGBoost Classifier ('A general architecture of Random Forest 5', no date)

Για την εκπαίδευση του μοντέλου επιλέγονται οι παράμετροι :

- Max_depth: Μέγιστο βάθος δέντρου. Οι τιμές που πήρε είναι {2,3,5,7,10}
- N_estimators : πλήθος δέντρων. Όσα πιο πολλά τα δέντρα τόσο μεγαλύτερη πιθανότητα υπερπροσαρμογής. Οι τιμές που πήρε είναι { 10, 100, 500}

4 Αποτελέσματα Μοντέλων

4.1 Νευρωνικά Δίκτυα (MLP)

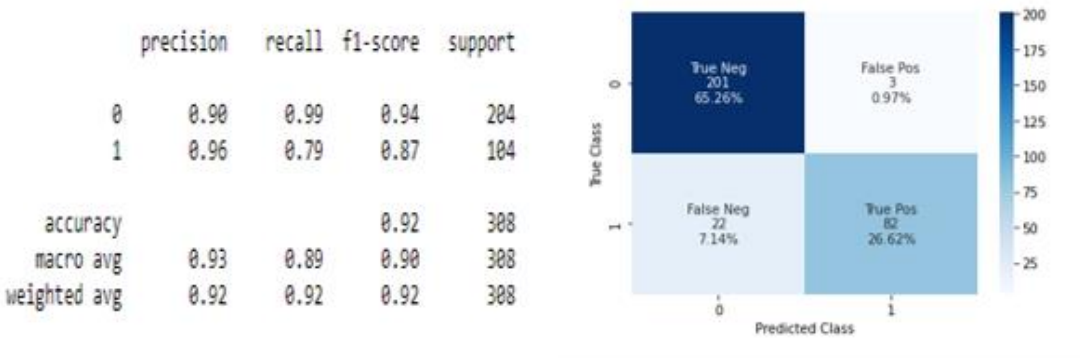
Από τις κ=3 πτυχές που έχουν οριστεί στο Cross Validation έχουν γίνει 18 fits στο μοντέλο εκπαίδευσης.

Χωρίς Oversampling

Με την χρήση GridSearchCv προκύπτει το καλύτερο μοντέλο με τους καλύτερους συνδυασμούς υπερπαραμέτρων με τις ακόλουθες τιμές τους : { "activation": "relu", "hidden_layer_sizes": (200,50), "learning_rate_init": 0.0001, "solver": "adam" }.

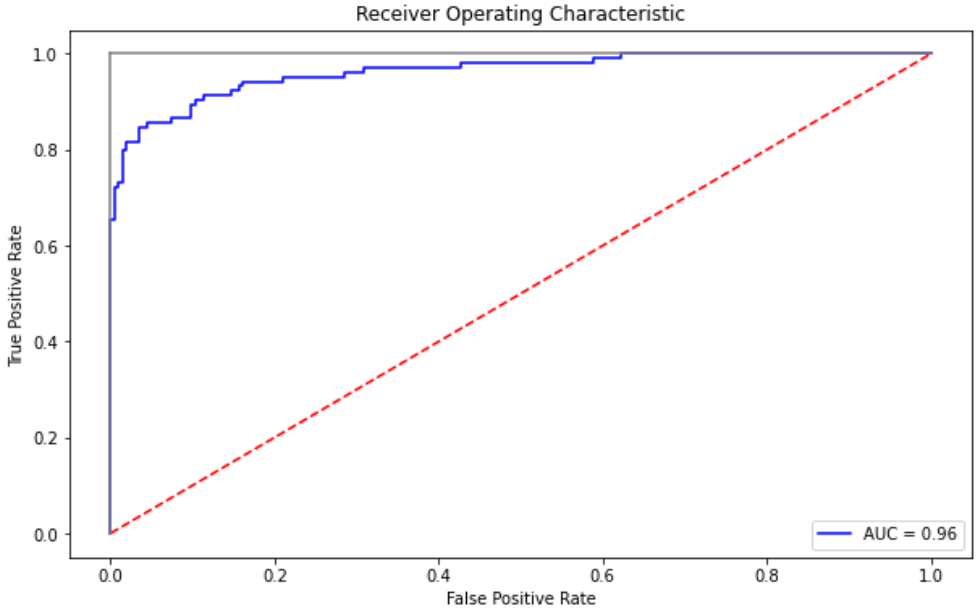
Το μέσο σκορ όλων των πτυχών Cross Validation που έχουν οριστεί για το παραπάνω συνδυασμό παραμέτρων είναι 0.865 με τη μετρική accuracy .

Αφού οριστεί ο καλύτερος εκτιμητής ταξινόμησης, πάνω σε αυτόν γίνεται η πρόβλεψη και εκτίμηση απόδοσης του μοντέλου. Οι μετρικές που χρησιμοποιούνται για την αξιολόγηση απόδοσης του μοντέλου είναι οι ακόλουθες με τα αντίστοιχα αποτελέσματα.



	precision	recall	f1-score	support
0	0.90	0.99	0.94	204
1	0.96	0.79	0.87	104
accuracy			0.92	308
macro avg	0.93	0.89	0.90	308
weighted avg	0.92	0.92	0.92	308

Εικόνα 70 Classification Report - Confusion Matrix MLP



Εικόνα 71 ROC καμπύλη MLP

Με Oversampling

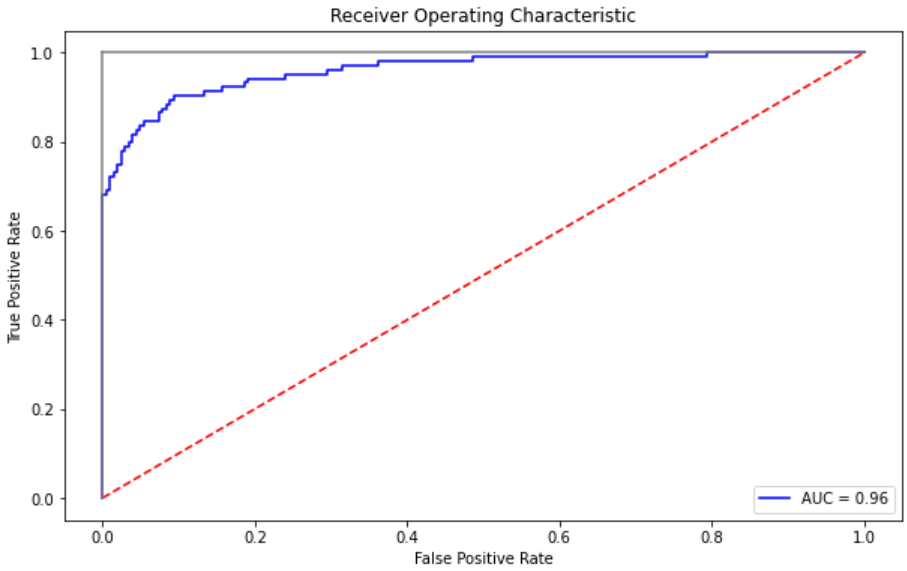
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“activation”: ”relu”, “hidden_layer_sizes”: (200,20), “learning_rate_init”: 0.0001, “solver”: “adam” }.

Το μέσο σκορ όλων των πτυχών Cross Validation που έχουν οριστεί για το παραπάνω συνδυασμό παραμέτρων είναι 0.879 με τη μετρική accuracy .

Οι μετρικές που χρησιμοποιούνται για την αξιολόγηση απόδοσης του καλύτερου εκτιμητή είναι οι ακόλουθες με τα αντίστοιχα αποτελέσματα.

	precision	recall	f1-score	support
0	0.92	0.95	0.93	204
1	0.90	0.83	0.86	104
accuracy			0.91	308
macro avg	0.91	0.89	0.90	308
weighted avg	0.91	0.91	0.91	308

Εικόνα 72 Classification Report - Confusion Matrix MLP (Oversampling)



Εικόνα 73 Καμπύλη ROC MLP (Oversampling)

4.2 KNN (K-nearest neighbors Classifier)

Από τις κ=3 πτυχές που έχουν οριστεί στο Cross Validation έχουν γίνει 24 fits.

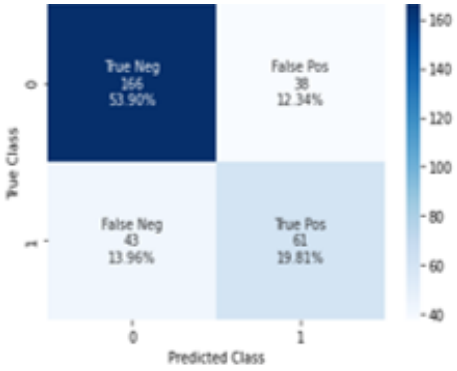
Χωρίς Oversampling

Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθη τιμή : {“n_neighbors”: 35 }

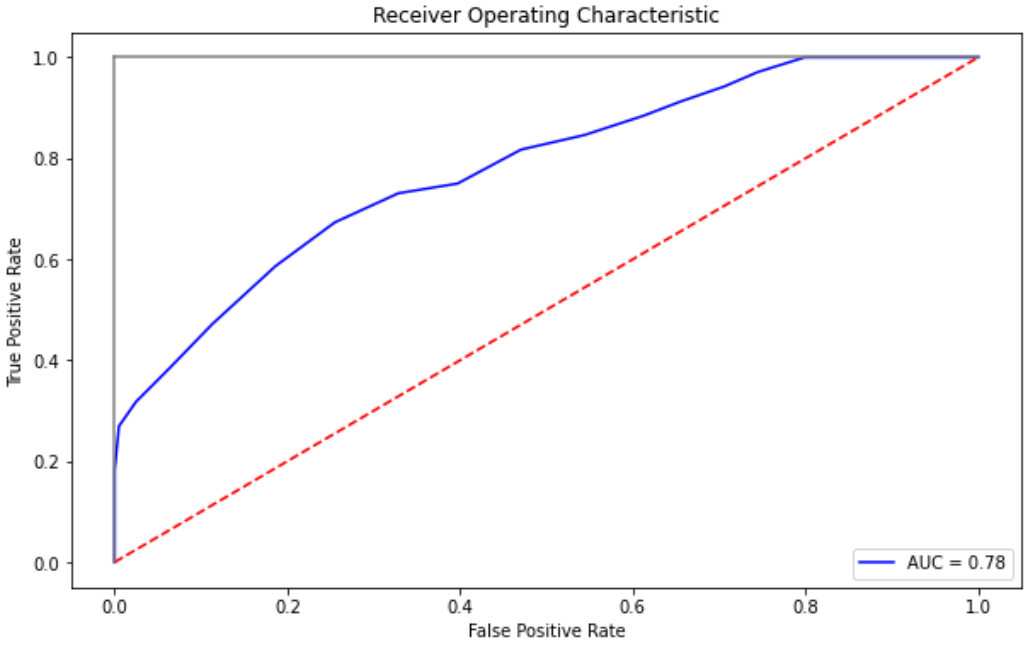
Το μέσο σκορ όλων των πτυχών Cross Validation που έχουν οριστεί για το παραπάνω συνδυασμό παραμέτρων είναι 0.746 με τη μετρική accuracy .

Οι μετρικές που χρησιμοποιούνται για την αξιολόγηση απόδοσης του καλύτερου εκτιμητή είναι οι ακόλουθες με τα αντίστοιχα αποτελέσματα.

	precision	recall	f1-score	support
0	0.79	0.81	0.80	204
1	0.62	0.59	0.60	104
accuracy			0.74	308
macro avg	0.71	0.70	0.70	308
weighted avg	0.73	0.74	0.74	308



Εικόνα 74 Classification Report - Confusion Matrix KNN



Εικόνα 75 ROC Καμπύλη KNN

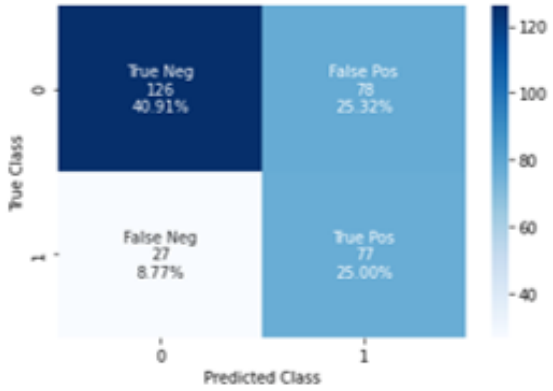
Με Oversampling

Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθη τιμή : {“n_neighbors”: 9}

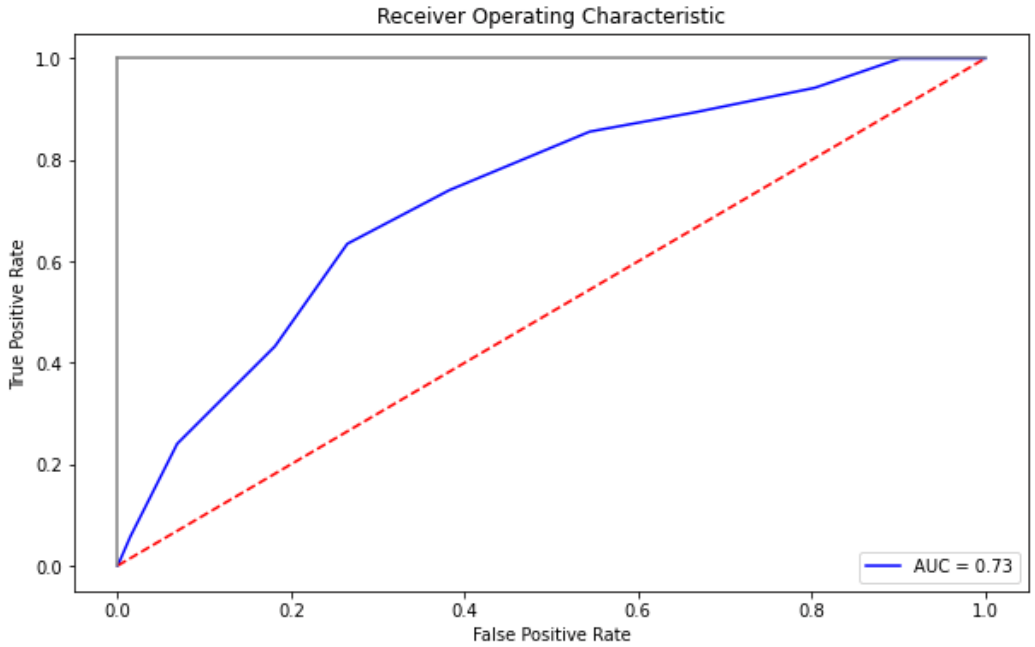
Το μέσο σκορ όλων των πτυχών Cross Validation που έχουν οριστεί για το παραπάνω συνδυασμό παραμέτρων είναι 0.715 με τη μετρική accuracy .

Οι μετρικές που χρησιμοποιούνται για την αξιολόγηση απόδοσης του καλύτερου εκτιμητή είναι οι ακόλουθες με τα αντίστοιχα αποτελέσματα.

	precision	recall	f1-score	support
0	0.82	0.62	0.71	204
1	0.50	0.74	0.59	104
accuracy			0.66	308
macro avg	0.66	0.68	0.65	308
weighted avg	0.71	0.66	0.67	308



Εικόνα 76 Classification Report - Confusion Matrix KNN (Oversampling)



Εικόνα 77 ROC Καμπύλη KNN (Oversampling)

4.3 Δέντρα Απόφασης (Decision Tree)

Από τις κ=3 πτυχές που έχουν οριστεί στο Cross Validation έχουν γίνει 72 fits.

Χωρίς Oversampling

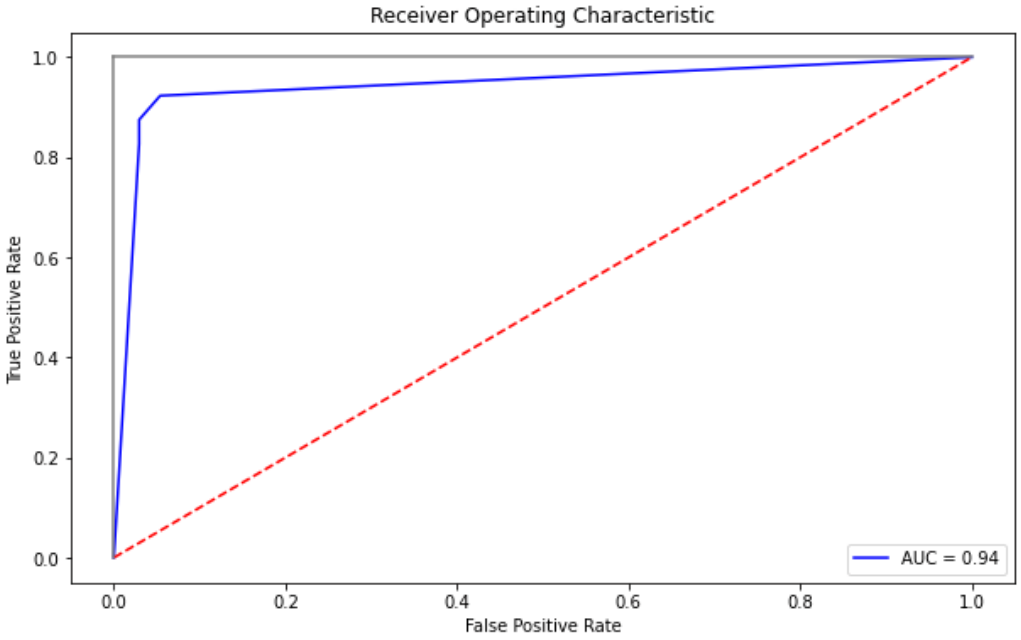
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“criterion”: “entropy”, “max_depth”: 10, “splitter”: “best”}.

Το μέσο σκορ όλων των πτυχών Cross Validation που έχουν οριστεί για το παραπάνω συνδυασμό παραμέτρων είναι 0.936 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι :

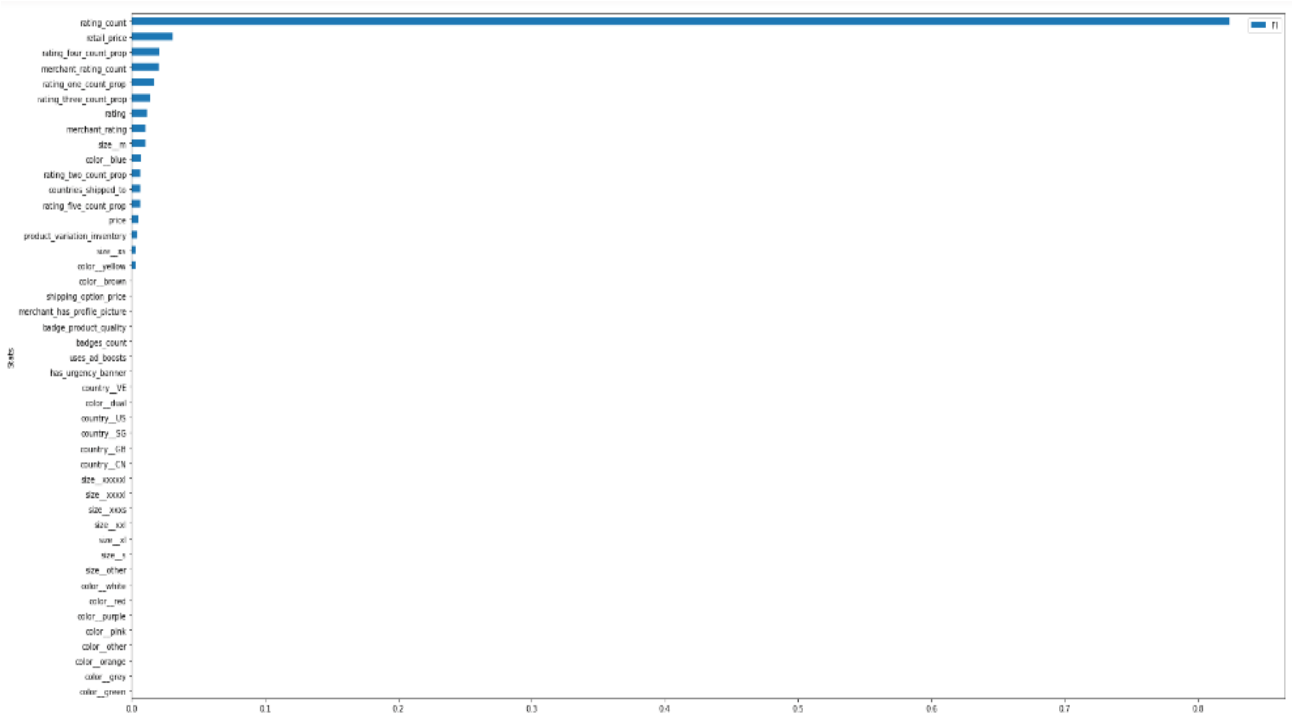
	precision	recall	f1-score	support
0	0.94	0.97	0.95	204
1	0.94	0.88	0.91	104
accuracy			0.94	308
macro avg	0.94	0.92	0.93	308
weighted avg	0.94	0.94	0.94	308

Εικόνα 78 Classification Report - Confusion Matrix Decision Tree



Εικόνα 79 ROC Καμπύλη Decision Tree

Το δέντρο αποφάσεων επιτρέπει την αναπαράσταση σημαντικότητας των χαρακτηριστικών στις ανεξάρτητες μεταβλητές του συνόλου δεδομένων.



Εικόνα 80 Barplot Σημαντικότητας Χαρακτηριστικών Decision Tree

Με Oversampling

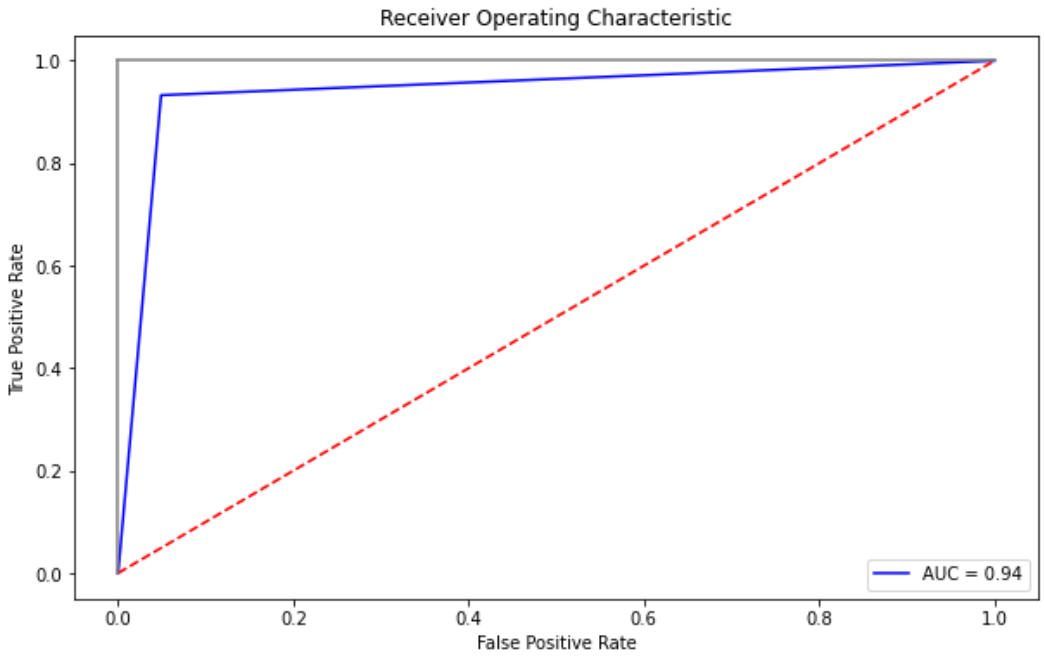
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“criterion”: “entropy”, “max_depth”: “None”, “splitter”: “best”}.

Το μέσο σκορ απ’ το Cross Validation είναι 0.941.

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

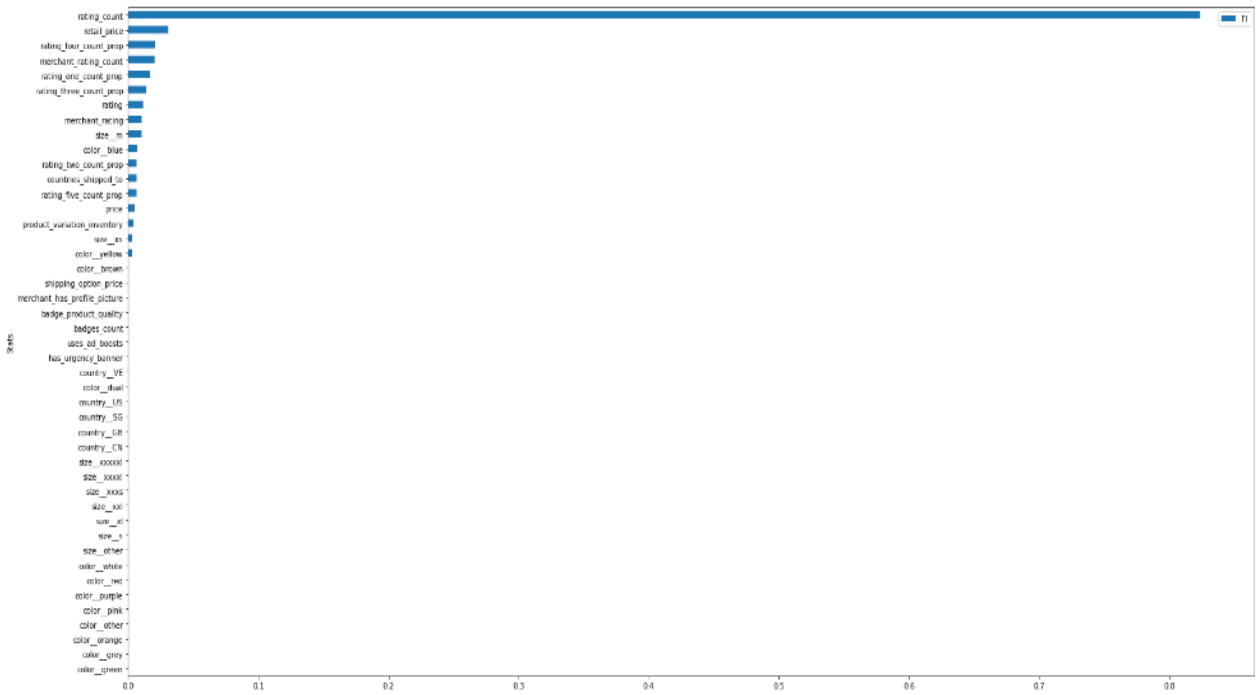
	precision	recall	f1-score	support
0	0.97	0.95	0.96	204
1	0.91	0.93	0.92	104
accuracy			0.94	308
macro avg	0.94	0.94	0.94	308
weighted avg	0.95	0.94	0.94	308

Εικόνα 81 Classification Report - Confusion Matrix Decision Tree (Oversampling)



Εικόνα 82 ROC Καμπύλη Decision Trees (Oversampling)

Σημαντικότητα εξαρτημένων μεταβλητών :



Εικόνα 83 Barplot Σημαντικότητας Χαρακτηριστικών Decision Trees (Oversampling)

4.4 Logistic Regression

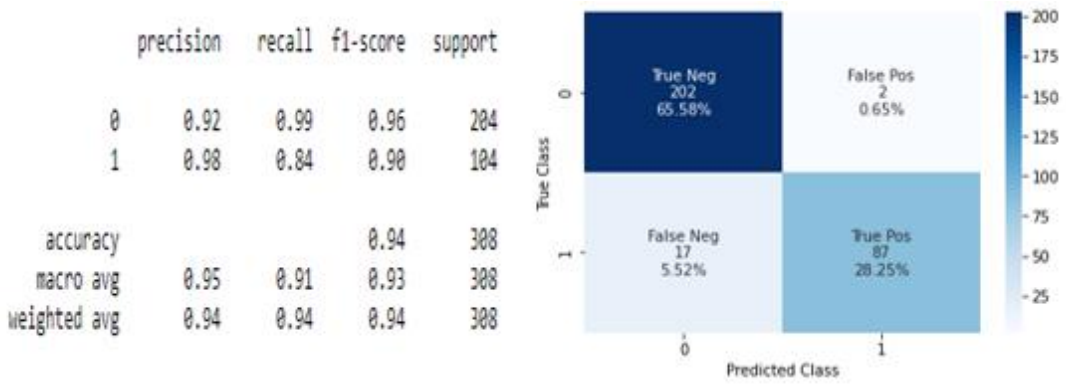
Από τις κ=3 πτυχές που έχουν οριστεί στο Cross Validation έχουν γίνει 21 fits.

Χωρίς Oversampling

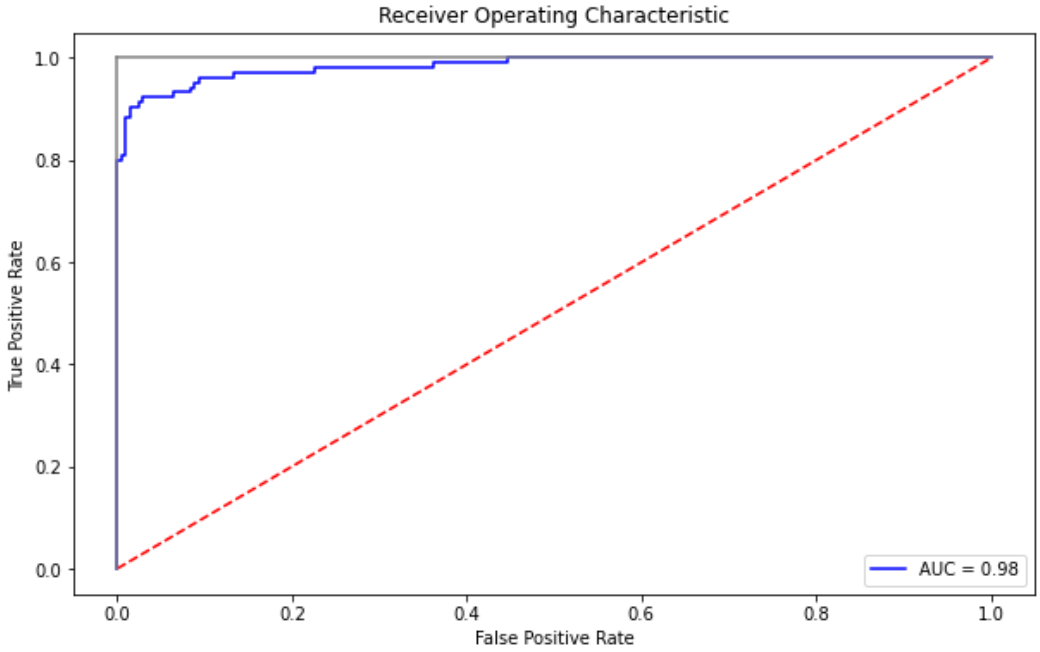
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“C”: 10}.

Το μέσο σκορ απ’ το Cross Validation είναι 0.938 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

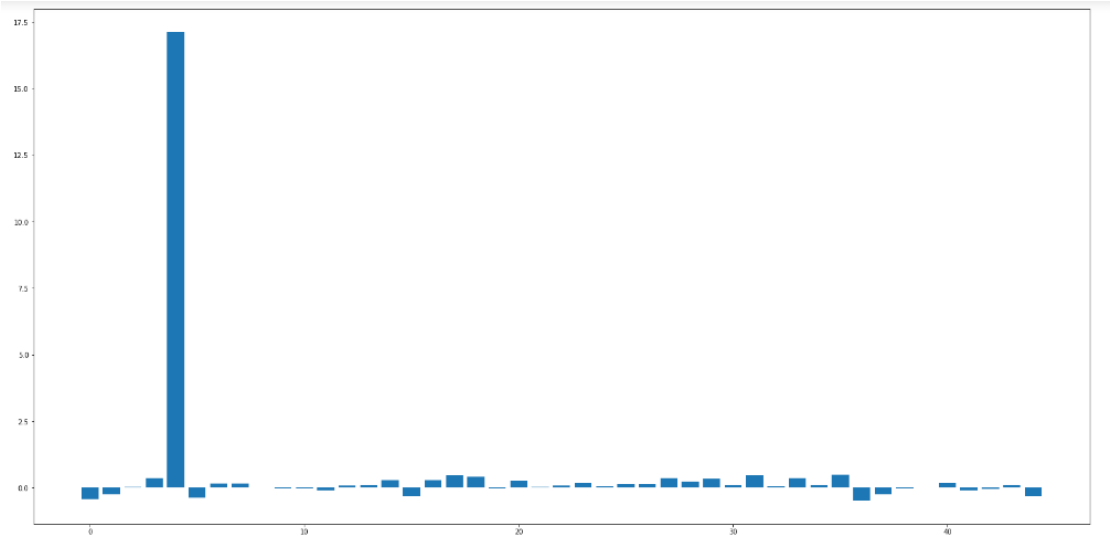


Εικόνα 84 Classification Report - Confusion Matrix Logistic Regression



Εικόνα 85 ROC Καμπύλη Logistic Regression

Σημαντικότητα εξαρτημένων μεταβλητών :



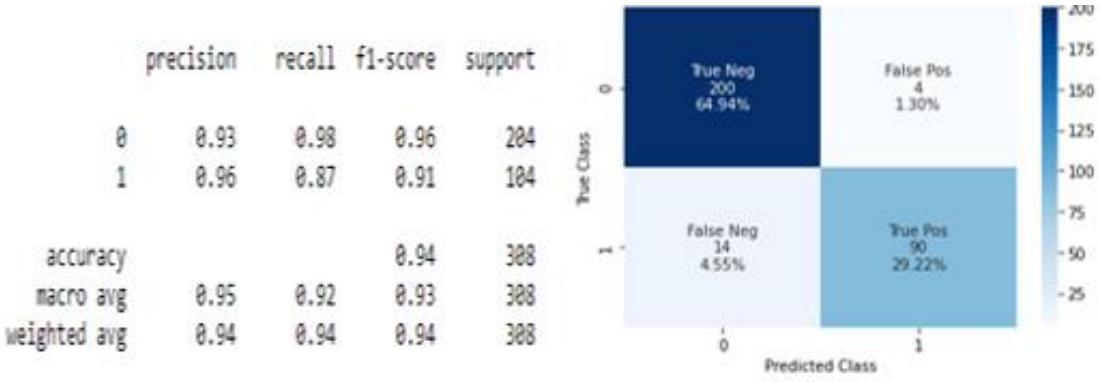
Εικόνα 86 Barplot Σημαντικότητας Εξαρτημένων Μεταβλητών Logistic Regression

Με Oversampling

Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“C”: 100}.

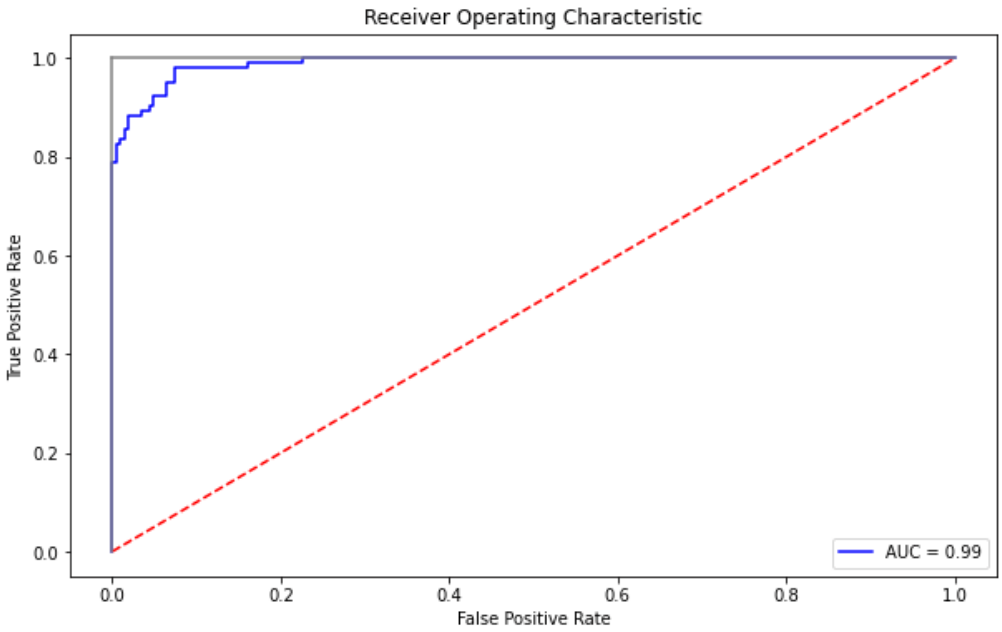
Το μέσο σκορ απ’ το Cross Validation είναι 0.935 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:



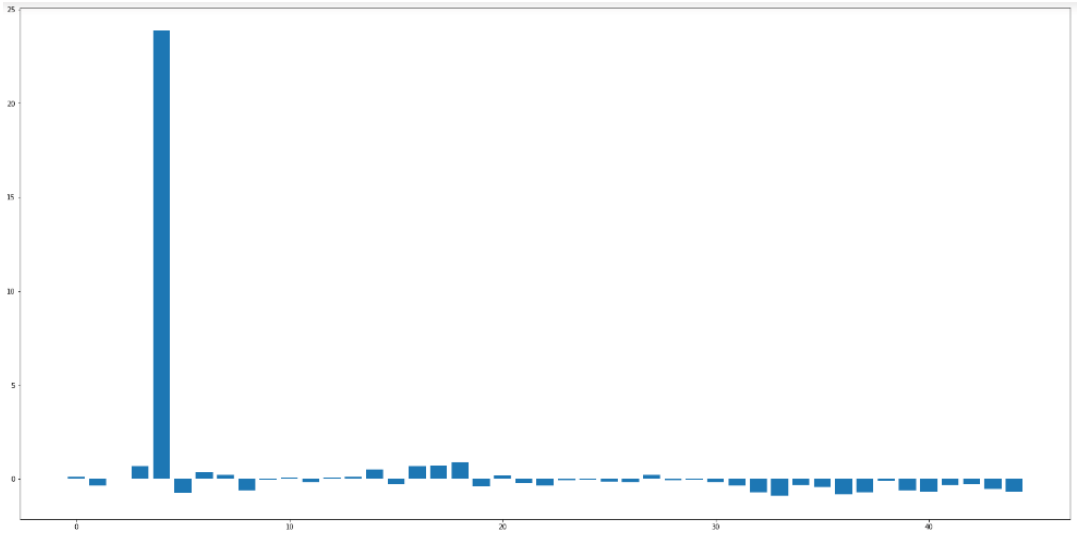
	precision	recall	f1-score	support
0	0.93	0.98	0.96	204
1	0.96	0.87	0.91	104
accuracy			0.94	308
macro avg	0.95	0.92	0.93	308
weighted avg	0.94	0.94	0.94	308

Εικόνα 87 Classification Report - Confusion Matrix Logistic Regression (Oversampling)



Εικόνα 88 ROC Καμπύλη Logistic Regression (Oversampling)

Σημαντικότητα εξαρτημένων μεταβλητών :



Εικόνα 89 Barplot Σημαντικότητας εξαρτημένων μεταβλητών Logistic Regression (Oversampling)

4.5 Gaussian Naïve Bayes

Από τις κ=3 πτυχές που έχουν οριστεί στο Cross Validation έχουν γίνει 9 fits.

Χωρίς Oversampling

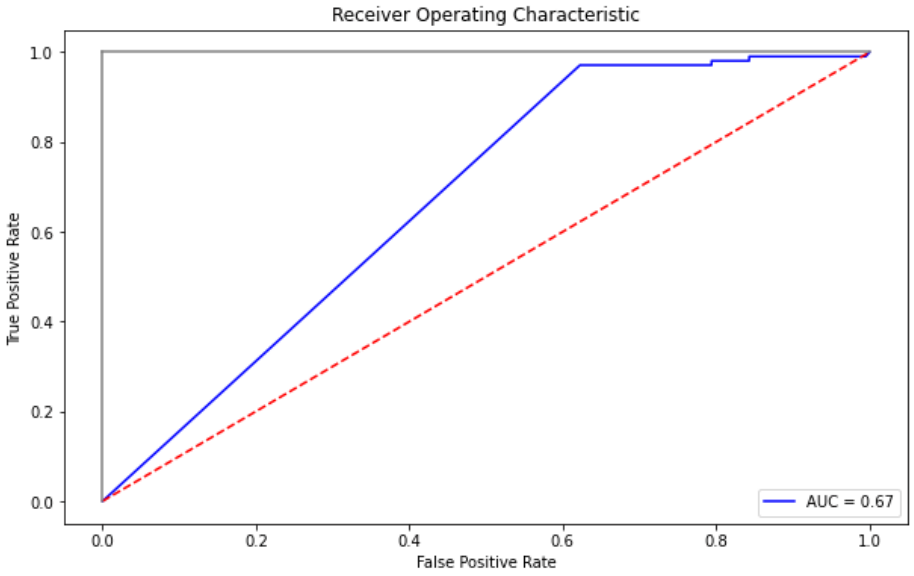
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές: {“priors”: None,”var_smoothing”: 0.000000001}.

Το μέσο σκορ απ’ το Cross Validation είναι 0.480 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

	precision	recall	f1-score	support
0	0.95	0.18	0.30	204
1	0.38	0.98	0.55	104
accuracy			0.45	308
macro avg	0.66	0.58	0.43	308
weighted avg	0.76	0.45	0.39	308

Εικόνα 90 Classification Report - Confusion Matrix GaussianNB



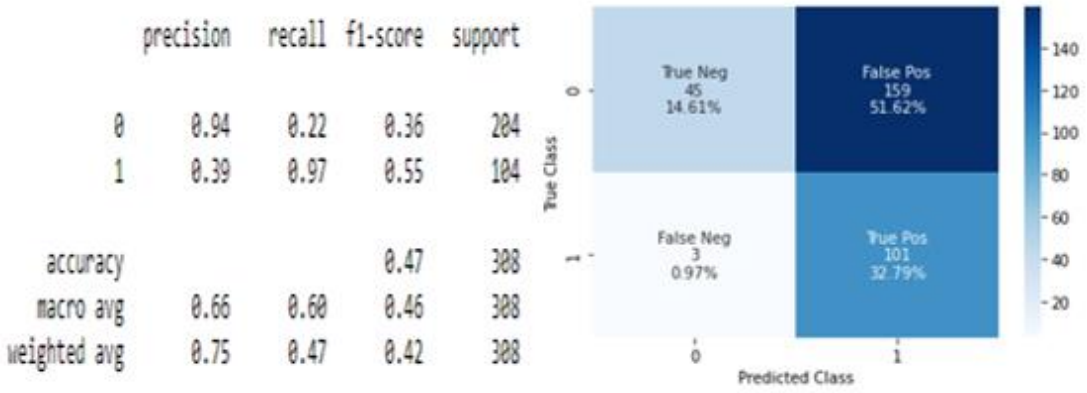
Εικόνα 91 ROC Καμπύλη GaussianNB

Με Oversampling

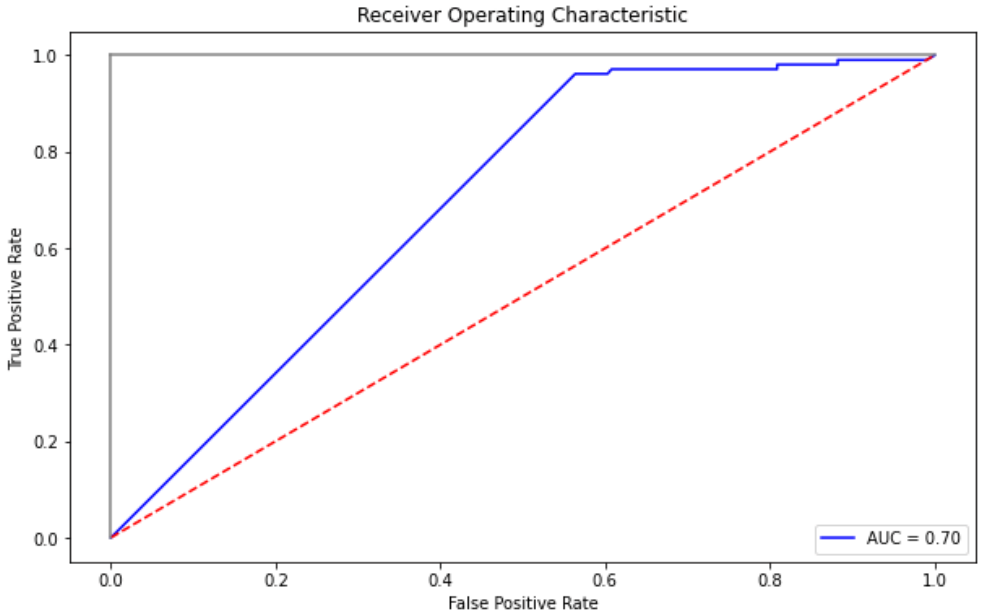
Το μοντέλο παίρνει τις ίδιες τιμές παραμέτρων όπως παραπάνω.

Το μέσο σκορ απ’ το Cross Validation είναι 0.631 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:



Εικόνα 92 Classification Report - Confusion Matrix GaussianNB (Overamping)



Εικόνα 93 ROC Καμπύλη GaussianNB(Oversamping)

4.6 Random Forest

Έχουν γίνει συνολικά 180 fits.

<u>Χωρίς Oversampling</u>

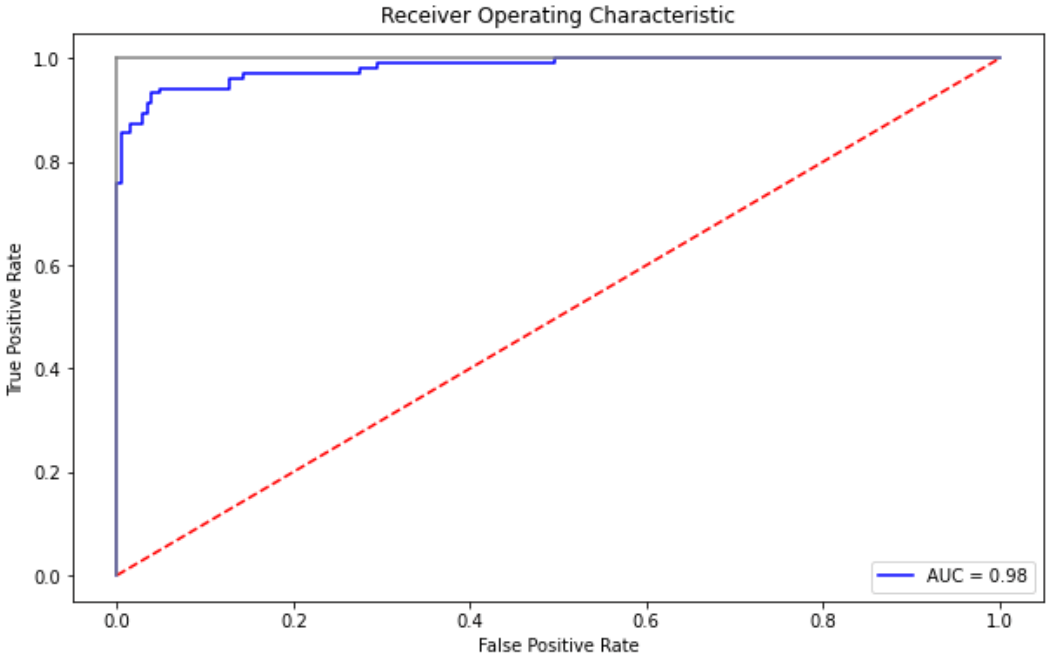
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“criterion”: “gini”,”max_depth”: 8,”max_features”:”auto”, ”n_estimators”: 200}.

Το μέσο σκορ απ’ το Cross Validation είναι 0.939 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

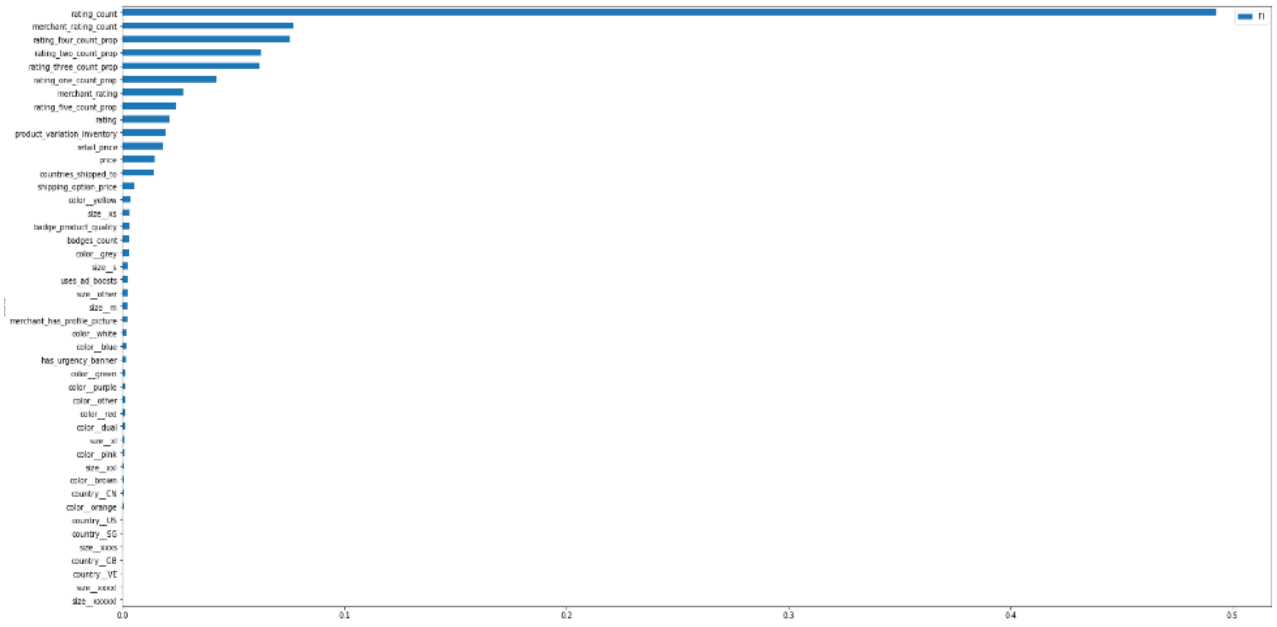
	precision	recall	f1-score	support
0	0.93	0.99	0.96	204
1	0.98	0.86	0.91	104
accuracy			0.94	308
macro avg	0.95	0.92	0.94	308
weighted avg	0.95	0.94	0.94	308

Εικόνα 94 Classification Report - Confusion Matrix Random Forest



Εικόνα 95 ROC Καμπύλη Random Forest

Σημαντικότητα εξαρτημένων μεταβλητών :



Εικόνα 96 Barplot Σημαντικότητας εξαρτημένων μεταβλητών Random Forest

Με Oversampling

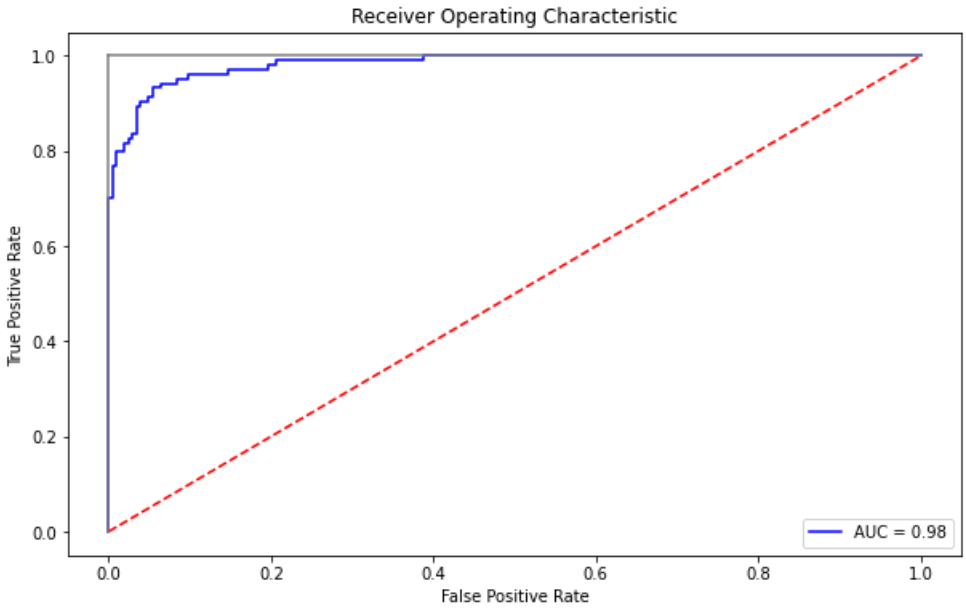
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“criterion”: “entropy”,”max_depth”: 8,”max_features”:”auto”, ”n_estimators”: 500}.

Το μέσο σκορ απ’ το Cross Validation είναι 0.946 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

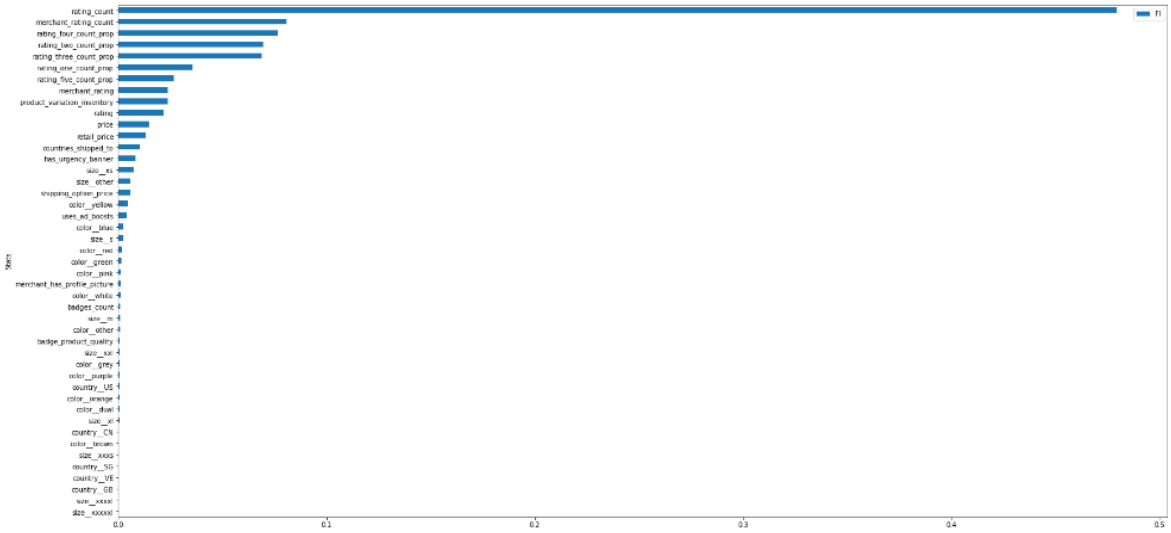
	precision	recall	f1-score	support
0	0.96	0.95	0.95	204
1	0.90	0.91	0.91	104
accuracy			0.94	308
macro avg	0.93	0.93	0.93	308
weighted avg	0.94	0.94	0.94	308

Εικόνα 97 Classification Report - Confusion Matrix Random Forest (Oversampling)



Εικόνα 98 ROC Καμπύλη Random Forest(Oversampling)

Σημαντικότητα εξαρτημένων μεταβλητών :



Εικόνα 99 Barplot Σημαντικότητας εξαρτημένων μεταβλητών Random Forest (Oversampling)

4.7 XGBoost Classifier

Έχουν γίνει συνολικά 45 fits.

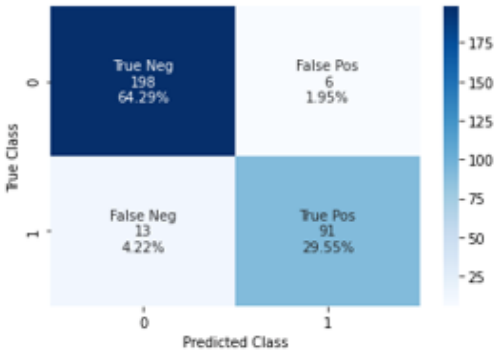
<u>Χωρίς Oversampling</u>

Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“max_depth”: 2 , ”n_estimators”: 100}.

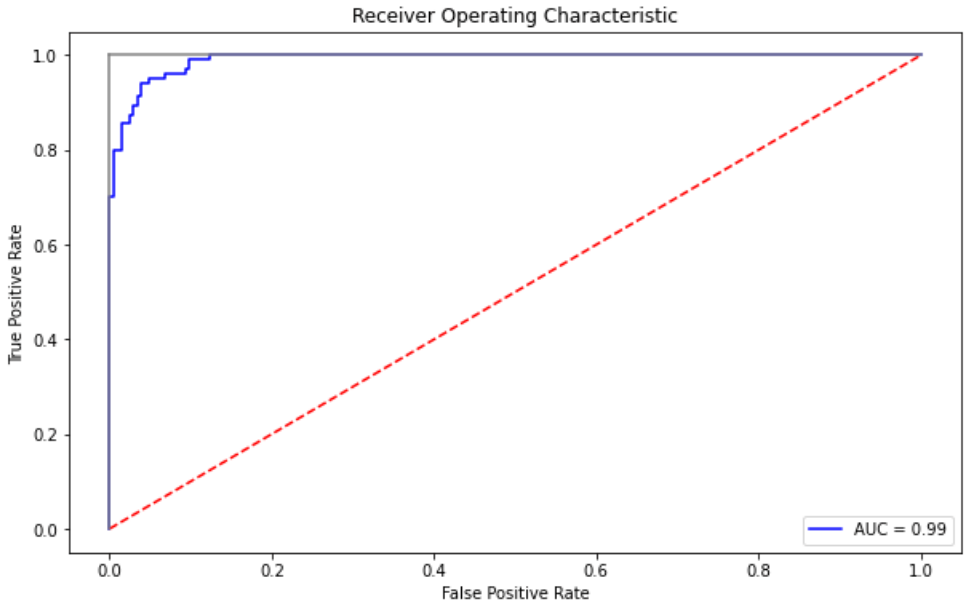
Το μέσο σκορ απ’ το Cross Validation είναι 0.946 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

	precision	recall	f1-score	support
0	0.94	0.97	0.95	204
1	0.94	0.88	0.91	104
accuracy			0.94	308
macro avg	0.94	0.92	0.93	308
weighted avg	0.94	0.94	0.94	308

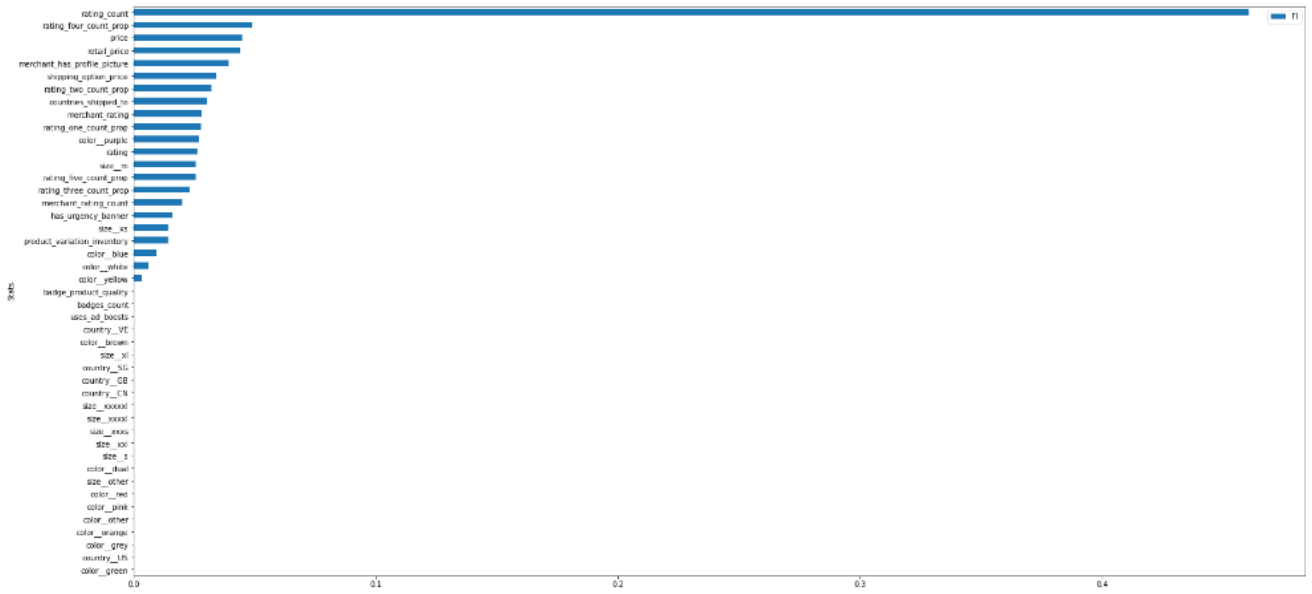


Εικόνα 100 Classification Report - Confusion Matrix XGBoost



Εικόνα 101 ROC Καμπύλη XGBoost

Σημαντικότητα εξαρτημένων μεταβλητών :



Εικόνα 102 Barplot Σημαντικότητας εξαρτημένων μεταβλητών XGBoost

Με Oversampling

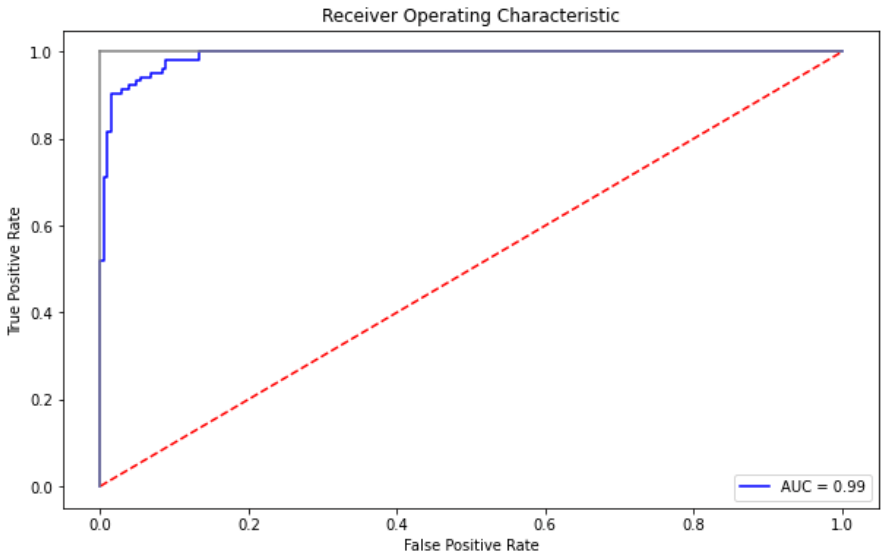
Το μοντέλο με τους καλύτερους συνδυασμούς παραμέτρων παίρνει τις ακόλουθες τιμές : {“max_depth”: 5 , ”n_estimators”: 500}.

Το μέσο σκορ απ’ το Cross Validation είναι 0.963 .

Οι μετρικές απόδοσης του καλύτερου εκτιμητή είναι:

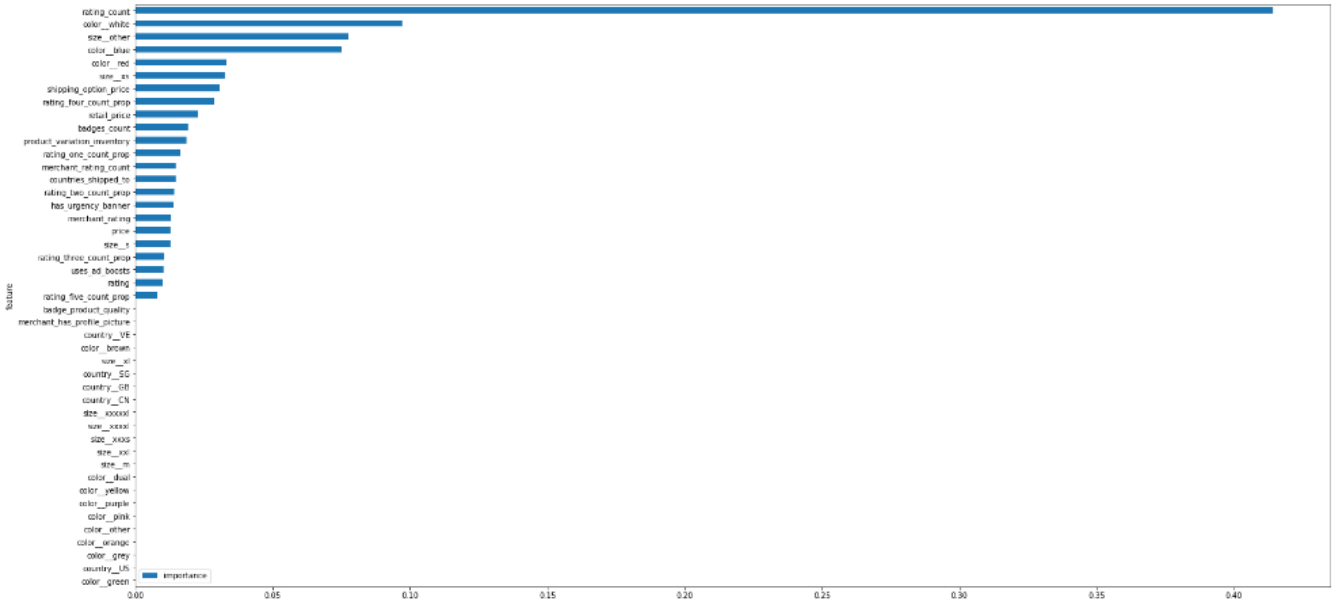
	precision	recall	f1-score	support
0	0.96	0.97	0.96	204
1	0.93	0.91	0.92	104
accuracy			0.95	308
macro avg	0.94	0.94	0.94	308
weighted avg	0.95	0.95	0.95	308

Εικόνα 103 Classification Report - Confusion Matrix XGBoost (Oversampling)



Εικόνα 104 ROC Καμπύλη XGBoost(Oversampling)

Σημαντικότητα εξαρτημένων μεταβλητών :



Εικόνα 105 Barplot Σημαντικότητας εξαρτημένων μεταβλητών XGBoost(Oversampling)

5 Συγκρίσεις Μοντέλων

Παρακάτω αναφέρονται δύο πίνακες με και χωρίς χρήση υπερδειγματοληψίας και γίνονται συγκρίσεις των αποτελεσμάτων για κάθε μετρική.

Χωρίς τη Μέθοδο Υπερδειγματοληψίας (SMOTE)									
ΜΟΝΤΕΛΑ	ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ					Confusion Matrix			
	Precision	Recall	F1-score	Accuracy	AUC	TN	TP	FN	FP
MLP	0.96	0.79	0.87	0.92	0.96	201	82	22	3
KNN	0.62	0.59	0.60	0.74	0.78	166	61	43	38
Decision Trees	0.94	0.88	0.91	0.94	0.94	198	91	13	6
Logistic Regression	0.98	0.84	0.90	0.94	0.98	202	87	17	2
Gaussian Naïve Bayes	0.38	0.98	0.55	0.45	0.67	37	102	2	167
Random Forest	0.98	0.86	0.91	0.94	0.98	202	89	15	2
XG Boost	0.94	0.88	0.91	0.94	0.99	198	91	13	6

Πίνακας 1 Τελικά αποτελέσματα μοντέλων

- Precision: το μεγαλύτερο ποσοστό θετικών προβλέψεων επιτυχίας πωλήσεων των προϊόντων του ιστότοπου Wish.com το έκαναν τα μοντέλα “Logistic Regression” και “Random Forest”με τιμή 98%.
- Recall: Το μοντέλο που έκανε την καλύτερη πρόβλεψη των TP είναι το Gaussian NB με ποσοστό 98% .
- F1-Score : Το καλύτερο σκορ το έκαναν τα μοντέλα “Decision Trees”, “”Random Forest” και “XG Boost” με τιμή 91%.
- Accuracy : Το μεγαλύτερο ποσοστό προβλέψεων που είναι σωστό το έκαναν τα μοντέλα “ Decision Trees”, “Logistic Regression”, “Random Forest” και “ XG Boost” με τιμή 94%.
- AUC : Η καλύτερη επίδοση ταξινόμησης επιτυχίας και μη επιτυχίας πωλήσεων των προϊόντων έγινε από το XG Boost με τιμή 99%.
- Σφάλμα τύπου I : το μοντέλο που έχει το μεγαλύτερο σφάλμα τύπου I είναι το Gaussian NB. Αναμενόμενο αφού το σκορ AUC είναι αρκετά χαμηλό με τιμή 0.67. Προέβλεψε 167 προϊόντα να έχουν επιτυχία στις πωλήσεις από τα συνολικά 308 και δεν είχαν.
- Σφάλμα τύπου II : το μοντέλο με το μεγαλύτερο σφάλμα τύπου II είναι το KNN. Επίσης αναμενόμενο αφού το σκορ AUC είναι επίσης χαμηλό με τιμή

0.78. Προέβλεψε 43 προϊόντα να μην έχουν επιτυχία από τα συνολικά 308 ενώ είχαν .

Με Μέθοδο Υπερδειγματοληψίας (SMOTE)									
ΜΟΝΤΕΛΑ	ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ					Confusion Matrix			
	Precision	Recall	F1-score	Accuracy	AUC	TN	TP	FN	FP
MLP	0.90	0.83	0.86	0.91	0.96	194	86	18	10
KNN	0.50	0.74	0.59	0.66	0.73	126	77	27	78
Decision Trees	0.91	0.93	0.92	0.94	0.94	194	97	7	10
Logistic Regression	0.96	0.87	0.91	0.94	0.99	200	90	14	4
Gaussian Naïve Bayes	0.39	0.97	0.55	0.47	0.70	45	101	3	159
Random Forest	0.90	0.91	0.91	0.94	0.98	194	95	9	10
XG Boost	0.93	0.91	0.92	0.95	0.99	197	95	9	7

Πίνακας 2 Τελικά αποτελέσματα μοντέλων με Oversampling

Με τη μέθοδο SMOTE φαίνεται ότι οι τιμές των μετρικών αλλάζουν. Οι τιμές είναι πιο αντιπροσωπευτικές αφού με την υπερδειγματοληψία στο σετ εκπαίδευσης έγινε αποφυγή υπερπροσαρμογής των μοντέλων.

- Precision : Οι τιμές έχουν πέσει εκτός από το Gaussian NB που είχε σκορ 38 % χωρίς SMOTE και τώρα έχει 39 %. Το καλύτερο σκορ έχει το “Logistic Regression” με τιμή 96%.
- Recall: όλες οι τιμές έχουν ανέβει εκτός από το Gaussian NB που είχε τιμή 98 % και τώρα έχει 97%. Το καλύτερο σκορ το Gaussian NB.
- F1 Score: Πτώση του σκορ υπάρχει στο MLP και KNN. Στα Decision Trees, Logistic Regression και XGBoost υπάρχει αύξηση. Στα Gaussian NB και Random Forest το αποτέλεσμα είναι ίδιο. Το καλύτερο σκορ έχουν τα Decision Trees και XGBoost με τιμή 92%.
- Accuracy : Αυξήσεις έγιναν στο Gaussian NB και XGBoost με τιμές 47% και 95% αντίστοιχα. Αμετάβλητο είναι το σκορ στα μοντέλα Decision Trees, Logistic Regression και Random Forest. Καλύτερο σκορ έχει το XGBoost.

- AUC : αμετάβλητο είναι το σκορ στα MLP, Decision Tree, Random Forest και XGBoost. Έχει ανέβει στα Logistic Regression και Gaussian NB ενώ έχει μειωθεί στο KNN. Το καλύτερο σκορ έχει το μοντέλο XGBoost με τιμή 99%.
- Σφάλμα τύπου I : παρατηρείται αύξηση του FP σε όλα τα μοντέλα εκτός από το Gaussian NB με τιμή 159. Το μεγαλύτερο σφάλμα τύπου I βρίσκεται στο ίδιο μοντέλο.
- Σφάλμα Τύπου II : αντίθετα το FN μειώθηκε σε όλα εκτός από το Gaussian NB που έφτασε στη τιμή 3. Το μεγαλύτερο σφάλμα έχει το KNN με τιμή 27.

6 Συμπεράσματα

6.1 Οπτικοποιήσεις

- Οι καταναλωτές προτιμούν χαμηλές τιμές πώλησης για την αγορά προϊόντων. Οι μεγάλες εκπτώσεις δεν οδηγούν σε επιτυχία πωλήσεων.
- Σε γενικό σύνολο οι πελάτες είναι ικανοποιημένοι στις αγορές λόγω του μεγαλύτερου ποσοστού σε βαθμολογία 5 αστέρων. Επίσης όσο ανεβαίνουν οι βαθμολογίες τόσο μεγαλύτερες και οι πωλήσεις.
- Η χρήση διαφημίσεων δεν έχει επίδραση στις πωλήσεις. Ίσως να χρειαστεί η εταιρεία να διερευνήσει περισσότερο το γεγονός αυτό γιατί ενδέχεται να χάνει έσοδα από διαφημίσεις.
- Η αποστολή προϊόντων γίνεται σχεδόν από μία εταιρεία. Στο μεγαλύτερο ποσοστό πωλήσεων οι καταναλωτές δεν προτιμούν γρήγορη αποστολή. Η προτιμώμενη τιμή αποστολής είναι στο εύρος 1-3 ευρώ. Η αποστολή σε 20 – 60 χώρες φαίνεται ασφαλή για καλές πωλήσεις.
- Τα περισσότερα προϊόντα ανήκουν στη γυναικεία καλοκαιρινή κολεξιόν και αυτά που έχουν πάνω από 35 ετικέτες αγοράζονται πιο συχνά.
- Οι βαθμολογίες εμπόρων δίνουν άνοδο στις πωλήσεις.
- Η εικόνα προφίλ των εμπόρων δε επιδρά στην επιτυχία πωλήσεων.
- Το συνολικό απόθεμα προϊόντων θα πρέπει να είναι άνω των 50 για ασφαλέστερες πωλήσεις. Τα προϊόντα που έχουν πολύ χαμηλά αποθέματα έχουν υψηλές πωλήσεις. Ίσως να πωλούν καλά και να πρέπει να ανανεώνονται πιο συχνά.

- Ο αριθμός σημάτων δεν επηρεάζει τις πωλήσεις. Ωστόσο το σήμα ποιότητας είναι σημαντικό και ίσως να αυξήσει την επιτυχία πωλήσεων.
- Οι περισσότερες πωλήσεις γίνονται χωρίς τη χρήση επείγοντος σήματος. Ίσως γιατί οι περισσότερες πωλήσεις έχουν προϊόντα με εξασφαλισμένο απόθεμα.
- Τα προϊόντα που προτιμούν οι καταναλωτές έχουν χρώμα άσπρο και μαύρο και νούμερο small .
- Τα περισσότερα είδη είναι από Κίνα.
- Στη επισήμανση λέξης- κλειδιού στα προϊόντα υπερτερεί το plus size και οι μεγαλύτερες πωλήσεις γίνονται στα είδη αμάνικο πουλόβερ, plus size, μπλουζάκια, μπλούζες με σχήμα v.

6.2 Μοντέλα Επιβλεπόμενης Μηχανικής Μάθησης

Παρόλο που έγινε χρήση πολλών μετρικών για την αξιολόγηση απόδοσης των παραπάνω μοντέλων, βασικές μετρικές τέθηκαν το F1 Score σε συνδυασμό με το σκορ AUC. Βασικό κλειδί για την παραπάνω επιλογή ήταν το FP και FN καθώς και ο στόχος της πρόβλεψης.

Σχετικά με το Accuracy αν και σε αρκετά μοντέλα ήταν υψηλό, σαν μετρική δηλώνει το ποσοστό προβλέψεων που κάνει ένα μοντέλο και είναι σωστό. Στη περίπτωση του προβλήματος πρόβλεψης της επιτυχίας των πωλήσεων για τα προϊόντα του Wish ο αριθμός του TN που είναι να γίνει πρόβλεψη να μην υπάρχει επιτυχία και δεν υπήρχε, ήταν σε όλα τα μοντέλα υψηλότερος από το TP. Εξαίρεση αποτέλεσε το Gaussian NB που είχε όμως και πολύ χαμηλό accuracy. Από επιχειρηματική πλευρά αυτό που ενδιαφέρει έναν αναλυτή σε ένα ηλεκτρονικό κατάστημα με πωλήσεις προϊόντων είναι να ακολουθήσει το μοντέλο που να προβλέπει το ποσοστό προβλέψεων επιτυχίας πωλήσεων. Το Accuracy εξαρτάται από τα TP και TN και οι υψηλές τιμές TN οδηγεί σε υψηλές τιμές της μετρικής αυτής και σε λάθος συμπεράσματα. Γι’ αυτό και δεν λήφθηκε ως μετρική αξιολόγησης.

Η επιλογή του Precision γίνεται όταν το κόστος του FP είναι υψηλότερο από το FN. Αντίστροφα η επιλογή του Recall λαμβάνεται περισσότερο υπόψιν όταν το κόστος FN είναι υψηλότερο από το FP. Το F1 Score προτιμάται όταν το TN είναι υψηλό και τα

σφάλματα τύπου I και II είναι το ίδιο σημαντικά. Στο FP γίνεται πρόβλεψη να υπάρχει επιτυχία ενώ δεν υπήρχε. Αυτό μπορεί να οδηγήσει σε λάθος συμπεράσματα μία επιχείρηση. Όταν θα χρησιμοποιήσει το μοντέλο που θα προβλέπει λανθασμένα την επιτυχία πώλησης των προϊόντων θα χάνει σημαντικές πληροφορίες για το λόγο που οι καταναλωτές δεν τα προτιμούν. Στο FN γίνεται πρόβλεψη να μην έχουν επιτυχία τα μοντέλα ενώ είχαν. Όμοια οδηγεί μία επιχείρηση σε λάθος αποφάσεις. Τα προϊόντα που έχουν επιτυχία θα προβλέπονται ως μη επιτυχή με αποτέλεσμα ίσως και να αφαιρούνται και αυτό να δυσαρεστεί τους καταναλωτές και να οδηγεί σε μείωση κέρδους. Επομένως επιλέγεται το F1 Score για τη σύγκριση των μοντέλων.

Καλύτερη απόδοση σύμφωνα με το F1 Score έχουν τα μοντέλα :

- Χωρίς Oversampling : Decision Trees, Random Forest, XGBoost με 91%
- Με Oversampling : Decision Trees, XGBoost με 92%

Σε συνδυασμό με το σκορ AUC η τελική επιλογή του καλύτερου ταξινομητή για την επιτυχία πωλήσεων των προϊόντων του Wish.com είναι το XGBoost με F1 Score= 92% και AUC= 99%.

Τα πιο σημαντικά χαρακτηριστικά των προϊόντων του καλύτερου ταξινομητή XGBoost με και χωρίς oversampling παρουσιάζονται στο παρακάτω πίνακα με φθίνουσα σειρά.

ΣΗΜΑΝΤΙΚΟΤΗΤΑ ΕΞΑΡΤΗΜΕΝΩΝ ΜΕΤΑΒΛΗΤΩΝ XGboost	
Χωρίς Oversamping	**Με Oversamping**
rating_count	rating_count
rating_four_count_prop	color_white
price	size_other
retail_price	color_blue
merchant_has_profile_picture	color_red
shipping_option_price	size_xl
rating_two_count_prop	shipping_option_price
countries_shipped_to	rating_four_count_prop
merchant_rating	retail_price
rating_one_count_prop	badges_count
color_purple	product_variation_inventory
rating	rating_one_count_prop
size_m	merchant_rating_count
rating_five_count_prop	countries_shipped_to
rating_three_count_prop	rating_two_count_prop
merchant_rating_count	has_urgency_banner
has_urgency_banner	merchant_rating
rating_five_count_prop	countries_shipped_to
rating_three_count_prop	rating_two_count_prop
merchant_rating_count	has_urgency_banner
has_urgency_banner	merchant_rating
size_xs	price
product_variation_inventory	size_s
color_blue	rating_three_count_prop
color_white	uses_ad_boost
color_yellow	rating
-	rating_five_count_prop

Πίνακας 3 Σημαντικότητα Εξαρτημένων μεταβλητών XGBoost

Η εργασία καλύπτει ένα μεγάλο κομμάτι ανάλυσης δεδομένων με χρήση αρκετών γραφημάτων και αρκετών αλγόριθμων μηχανικής μάθησης. Όσον αναφορά το

κομμάτι των γραφημάτων εκτός από τη δημιουργία τους με χρήση κώδικα python θα μπορούσαν να αξιοποιηθούν και άλλα εργαλεία που εξειδικεύονται στην οπτικοποίηση δεδομένων. Ένα διάσημο εργαλείο που χρησιμοποιείται όλο και περισσότερο στο κόσμο του business marketing είναι το Power Bi. Έχει αναπτυχθεί απ' τη Microsoft και η ευελιξία του στην εξαγωγή και οπτικοποίηση δεδομένων είναι αυτό που το κάνει να ξεχωρίζει.

Επιπλέον σχετικά με τη μηχανική μάθηση σίγουρα θα μπορούσε να γίνει περισσότερη μελέτη μελλοντικά στην επιλογή των μεταβλητών του συνόλου δεδομένων αφού η μεταβλητή "rating_count" στα γραφήματα σημαντικότητας λαμβάνει σχεδόν το μεγαλύτερο μέρος. Θα μπορούσε να γίνει απαλοιφή του συγκεκριμένου χαρακτηριστικού και να γίνει μία σύγκριση με τα αποτελέσματα που ήδη έχουν υλοποιηθεί.

7 Βιβλιογραφία

‘4 Types of Classification Tasks in Machine Learning’ (no date). Available at: https://machinelearningmastery.com/types-of-classification-in-machine-learning/.

‘A general architecture of Random Forest 5’ (no date). Available at: https://www.researchgate.net/figure/A-general-architecture-of-Random-Forest-5_fig2_335483097.

Alzubi, J., Nayyar, A. and Kumar, A. (2018) ‘Machine Learning from Theory to Algorithms: An Overview’, *Journal of Physics: Conference Series*, 1142(1). doi:10.1088/1742-6596/1142/1/012012.

‘Bias-Variance in Machine Learning’ (no date). Available at: https://www.cs.cmu.edu/~wcohen/10-601/bias-variance.pdf.

Charbuty, B. and Abdulazeez, A. (2021) ‘Classification Based on Decision Tree Algorithm for Machine Learning’, *Journal of Applied Science and Technology Trends*, 2(01), pp. 20–28. doi:10.38094/jastt20165.

‘Decision-tree-Source-https-pianalytixcom-decision-tree-algorithm-Full-size-DOI’ (no date).

‘Dictionaries in Python – Real Python’ (no date). Available at: https://realpython.com/python-dicts/.

‘Five ways machine learning can improve your sales - CatalogPlayer’ (no date). Available at: https://catalogplayer.com/en/uncategorized/five-ways-machine-learning-can-improve-your-sales/9872/.

‘Gaussian Naive Bayes_ Τι πρέπει να γνωρίζετε; _ upGrad blog’ (no date). Available at: https://www.upgrad.com/blog/gaussian-naive-bayes/.

Google Developers (2020) ‘Classification: Accuracy | Machine Learning Crash Course | Google Developers’, *Machine Learning Crash Course* [Preprint]. Available at: https://developers.google.com/machine-learning/crash-course/classification/accuracy.

Great Learning Team (2020) ‘What is Cross Validation in Machine learning? Types of Cross Validation’, *Great Learning* [Preprint]. Available at: https://www.mygreatlearning.com/blog/cross-validation.

‘How to Interpret the Classification Report in sklearn (With Example) - Statology’ (no date).

Hruthik, P. *et al.* (2022) ‘MEAL PLAN PREDICTION USING DECISION TREE CLASSIFIER : A REVIEW’, (05), pp. 165–169.

Hunter, J. *et al.* (2020) ‘Matplotlib: Visualization with Python’, *Abgerufen 05.10.2020* [Preprint]. Available at: https://matplotlib.org/.

‘Introduction to Python’s Collections Module’ (no date). Available at: https://stackabuse.com/introduction-to-pythons-collections-module/.

Java Point (2018) ‘Logistic Regression in Machine Learning - Javatpoint’, pp. 1–22. Available at: https://www.javatpoint.com/logistic-regression-in-machine-learning.

Javapoint (2021) ‘Machine Learning Random Forest Algorithm - Javatpoint’. Available at: https://www.javatpoint.com/machine-learning-random-forest-algorithm.

Javatpoint (2011) ‘Naive Bayes Classifier in Machine Learning - Javatpoint’. Available at: https://www.javatpoint.com/machine-learning-naive-bayes-classifier.

Joseph, V.R. and Vakayil, A. (2021) ‘SPlit: An Optimal Method for Data Splitting’, *Technometrics*, 0(0), pp. 1–23. doi:10.1080/00401706.2021.1921037.

‘Kaggle - Wikipedia’ (no date). Available at: https://en.wikipedia.org/wiki/Kaggle.

kunal (2015) ‘SKLearn | Scikit-Learn In Python | SciKit Learn Tutorial’. Available at: https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/.

Lemaitre, G. and Aridas, C. (no date) ‘imbalanced-learn · PyPI’. Available at: https://pypi.org/project/imbalanced-learn/.

McKinney, W. and Team, P.D. (2015) ‘Pandas - Powerful Python Data Analysis Toolkit’, *Pandas - Powerful Python Data Analysis Toolkit*, p. 1625.

Mujtaba, H. (2020) ‘An Introduction to Grid Search CV | What is Grid Search’, *Great Learning* , p. 1. Available at: https://www.mygreatlearning.com/blog/gridsearchcv/. ‘naive-bayes-classifier-algorithm’ (no date).

Narkhede, S. (2018a) ‘Understanding AUC - ROC Curve | by Sarang Narkhede | Towards Data Science’, *Towards Data Science*, p. 1. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

Narkhede, S. (2018b) ‘Understanding Confusion Matrix | by Sarang Narkhede | Towards Data Science’, *Towards Data Science*, p. 1. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

Plotly (2020) ‘Getting Started with Plotly | Python | Plotly’, *Plotly* [Preprint]. Available at: https://plotly.com/r/getting-started/%0Ahttps://plotly.com/python/getting-started/.

Potdar, K., S., T. and D., C. (2017) ‘A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers’, *International Journal of Computer Applications*, 175(4), pp. 7–9. doi:10.5120/ijca2017915495.

Project Jupyter (2019) ‘Project Jupyter | Home’. Available at: https://jupyter.org/.

Raschka, S. (2018) ‘Nearest Neigbor Methods’. Available at: https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf.

Sblendorio, D. (2019) ‘Plotting Data in Python: matplotlib vs plotly | ActiveState’. Available at: https://www.activestate.com/blog/plotting-data-in-python-matplotlib-vs-plotly/.

‘sec-MCAR’ (no date). Available at: https://stefvanbuuren.name/fimd/sec-MCAR.html.

‘sklearn’ (no date). Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

‘Supervised learning - Wikipedia’ (no date). Available at: https://en.wikipedia.org/wiki/Supervised_learning.

‘Violin Plots in Python’ (no date) *Plotly Python API Library Reference* [Preprint]. Available at: https://plot.ly/python/violin-plot/.

Waskom, M. (2021) ‘Seaborn: Statistical Data Visualization’, *Journal of Open Source Software*, p. 3021. doi:10.21105/joss.03021.

‘What Can Data Visualization Do For Your Sales Department _ CustomerThink’ (no date). Available at: https://customerthink.com/what-can-data-visualization-do-for-your-sales-department/.

‘What is NumPy_ — NumPy v1’ (no date).

‘Why do we set a random state in machine learning models_ _ by Rukshan Pramoditha _ Apr, 2022 _ Towards Data Science’ (no date). Available at: https://towardsdatascience.com/why-do-we-set-a-random-state-in-machine-learning-models-bb2dc68d8431.

Wijaya, C.Y. (2020) ‘5 SMOTE Techniques for Oversampling your Imbalance Data | by Cornellius Yudha Wijaya | Towards Data Science’. Available at: https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5.

‘Wish (company) - Wikipedia’ (no date).

XGBoost (2020) ‘XGBoost Documentation’, *XGBoost Documentation*, pp. 2–3. Available at: https://xgboost.readthedocs.io/en/latest/index.html%0Ahttps://xgboost.readthedocs.io/en/latest/#%0Ahttps://xgboost.readthedocs.io/en/latest/.

‘XGBoost ML Model in Python - Javatpoint’ (no date). Available at: https://www.javatpoint.com/xgboost-ml-model-in-python.

‘Εφαρμογή Επιστήμης Δεδομένων _ Παραδείγματα εφαρμογών επιστήμης δεδομένων’ (no date). Available at: https://www.mygreatlearning.com/blog/data-science-applications/?fbclid=IwAR3vExilPOr4n12kPYEHpCjj3MegEg253vSCN9cFbSs-IFCUNXwjkjHImvI.

8 Παραρτήματα

8.1 Παράρτημα Α'- Μελέτη Δεδομένων

```
# Εγκατάσταση βιβλιοθηκών
!pip install seaborn
!pip install plotly
!pip install matplotlib_venn
!pip install  WordCloud
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
import plotly
import plotly.graph_objects as go
import plotly.express as px
import matplotlib.pyplot as plt
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
from plotly.subplots import make_subplots
from plotly.offline import iplot, init_notebook_mode
init_notebook_mode(connected = True)
!pip install cufflinks plotly
from plotly.offline import iplot, init_notebook_mode
import cufflinks
cufflinks.go_offline(connected=True)
init_notebook_mode(connected=True)

sales_data=pd.read_csv("summer_products_with_rating_and_performance_2020_08.csv")
sales_data

pd.set_option('display.max_columns', None)
sales_data.head()
```

Παράρτημα 1 Μελέτη Δεδομένων – Εισαγωγή Βιβλιοθηκών & Dataset

```
target_col = 'units_sold'
print(f"Shape of dataframe {sales_data.shape}")

rows = []
for col in sales_data.columns:
    if sales_data[col].isin([0, 1, np.nan]).all():
        row_dict = {'ColumnName': col, 'DataType': 'binary', 'HasMissing':sales_data.isnull().any().loc[col],
                    'NumberOfMissingCells': sales_data.isnull().sum().loc[col], 'CorrelationWithTarget': sales_data.corr()[target
                    'Mean': np.nan, 'Median': np.nan,'Mode': sales_data.mode()[col].loc[0], 'MinValue': np.nan, 'MaxValue': np.na

    elif sales_data.dtypes.loc[col] == 'int64' or sales_data.dtypes.loc[col] == 'float64':

        row_dict = {'ColumnName': col, 'DataType': sales_data.dtypes.loc[col], 'HasMissing':sales_data.isnull().any().loc[col],
                    'NumberOfMissingCells': sales_data.isnull().sum().loc[col], 'CorrelationWithTarget': sales_data.corr()[target
                    'Mean': sales_data.mean().loc[col], 'Median':sales_data.median().loc[col], 'Mode': sales_data.mode()[col].loc
                    'MinValue': sales_data.min().loc[col], 'MaxValue': sales_data.max().loc[col] }

    else:
        row_dict = {'ColumnName': col, 'DataType': sales_data.dtypes.loc[col], 'HasMissing':sales_data.isnull().any().loc[col],
                    'NumberOfMissingCells': sales_data.isnull().sum().loc[col], 'CorrelationWithTarget': np.nan, 'Mean': np.nan,
                    'Mode':sales_data.mode()[col].loc[0], 'MinValue': np.nan, 'MaxValue': np.nan }

    rows.append(row_dict)

info_df = pd.DataFrame(rows, columns=['ColumnName', 'DataType', 'HasMissing', 'NumberOfMissingCells', 'CorrelationWithTarget', 'N
info_df.set_index('ColumnName', inplace=True)
info_df = info_df.sort_values('CorrelationWithTarget', ascending=False, na_position='last')

print("FOR NUMERICAL COLUMNS")
info_df[info_df['DataType']!='object']
```

Παράρτημα 2 Μελέτη Δεδομένων – Στατιστική Ανάλυση (1/2)

```
print("\nFor categorical/non-numeric columns")
info_df[info_df['DataType']=='object'].drop(['CorrelationWithTarget', 'Mean', 'Median', 'MinValue', 'MaxValue'], axis=1)

sales_data.describe()

print('Dimensions of the df', sales_data.shape)

sales_data = sales_data.drop_duplicates()
print('Dimensions of the df after dropping the duplicates', sales_data.shape)
print("Duplicate product_id :",sales_data['product_id'].duplicated().sum())

def plot_missing_data(sales_data):
    columns_with_null = sales_data.columns[sales_data.isna().sum() > 0]
    null_pct = (sales_data[columns_with_null].isna().sum() / sales_data.shape[0]).sort_values(ascending=False) * 100
    plt.figure(figsize=(8,6));
    sns.barplot(y = null_pct.index, x = null_pct, orient='h')
    plt.title('% Missing values in dataframe by columns');
print(plot_missing_data(sales_data))

sales_data["units_sold"].value_counts()

print('Median of units sold is',sales_data['units_sold'].median())

print('{:<30} {:<15}'.format('column', 'unique values'))
for key in sales_data.keys():
    uniques = len(sales_data[str(key)].unique())
    print('{:<30} {:<15}'.format(str(key) , uniques))
    if uniques <= 20 :
        print('\t:', sales_data[str(key)].unique())
```

Παράρτημα 3 Μελέτη Δεδομένων – Στατιστική Ανάλυση (2/3)

```
sales_data['has_urgency_banner'].value_counts()

plt.figure(figsize=(12,6))
sns.boxplot(sales_data["retail_price"])

plt.figure(figsize=(12,6))
sns.boxplot(sales_data["price"])

df =sales_data
fig = px.box(df, x="retail_price")
fig.show()

fig = px.box(df, x="price")
fig.show()

plt.figure(figsize=(12,6))
sns.boxplot(sales_data['merchant_rating_count'],color='yellow',showmeans=True)
plt.title('Outliers of Merchant ratings')

sales_data['units_sold'] = [10 if x < 10 else x for x in sales_data['units_sold']]

trace1 = go.Violin(y=sales_data["price"],name='Price')
trace2 = go.Violin(y=sales_data["retail_price"],name='Retail price')
fig=go.Figure([trace1, trace2])
fig.update_layout(
title='Σύγκριση μεταξύ "price" και "retail price" ',
yaxis_title='Price(EUR)')
fig.show()
```

Παράρτημα 4 Μελέτη Δεδομένων – Στατιστική Ανάλυση (3/3)

```
def num_units_sold(units_sold):
    units_sold = int(units_sold)

    bracket = ''
    if units_sold in range(0, 1000):
        bracket = '< 1000'
    if units_sold in range(1000, 5000):
        bracket = '1000 - 5000'
    if units_sold in range(5000, 10000):
        bracket = '5000 - 10000'
    if units_sold in range(10000, 20000):
        bracket = '10000 - 20000'
    if units_sold in range(20000, 50000):
        bracket = '20000 - 50000'
    if units_sold in range(50000, 100000):
        bracket = '> 50000'
    return bracket

price=sales_data[["price",'units_sold']]
price["units_sold"] = price["units_sold"].apply(num_units_sold)
units_sold = price['units_sold'].unique().tolist()
units_sold_groupby = price.groupby("units_sold").agg({
    'price': 'sum'})
units_sold_groupby.reset_index()
units_sold_groupby = units_sold_groupby.iloc[1:]
units_sold_groupby=units_sold_groupby.reset_index()
units_sold_groupby=round(units_sold_groupby,0)
units_sold_groupby
```

Παράρτημα 5 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "retail_price", "price" (1/3)

```
values=['< 1000 ', '1000-5000',"5000-10000","10000-20000","20000-50000",">50000"]
y=[5055.0, 3465.0,1827.0,1460.0,899.0,118.0]
fig = go.Figure([go.Bar(x=values, y=y,text=y,texttemplate='%{text:.s} (€)',textposition='outside')])
fig.update_layout(title_text='Τιμή Αγοράς vs Αριθμός Πωλήσεων Προιόντων',xaxis_title=" Αριθμός Πωλήσεων Προιόντων",
    yaxis_title="Συνολική Τιμή Αγοράς")
fig.show()

cols = ['price',"retail_price"]
sales_data[cols] = sales_data[cols].applymap(np.int64)
drop_price= sales_data[["retail_price","price","units_sold"]]
drop_price["price_drop"]=drop_price["retail_price"]-drop_price["price"]

def num_price_drop(price_drop):
    price_drop = int(price_drop)
    bracket = ''
    if price_drop in range(-7, 0):
        bracket = '< 0'
    if price_drop in range(0, 10):
        bracket = '0 - 10'
    if price_drop in range(10, 20):
        bracket = '10 - 20'
    if price_drop in range(20, 30):
        bracket = '20 - 30'
    if price_drop in range(30, 40):
        bracket = '30 - 40'
    if price_drop in range(40, 60):
        bracket = '40 - 60'
    if price_drop in range(60, 245):
        bracket = '>60'
    return bracket
```

Παράρτημα 6 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "retail_price", "price" (2/3)

```
drop_price["range_price_drop"] = drop_price["price_drop"].apply(num_price_drop)

drop_price_sales=drop_price.groupby("range_price_drop")['units_sold'].sum().reset_index()
drop_price_sales

values=['< 0', '0 - 10', '10-20',"20-30","30-40","40-60",">60"]
y=[2401800, 2235180, 493400,411700,146950,410410,706830]
fig = go.Figure([go.Bar(x=values, y=y,text=y,
          texttemplate='%{text:.2s}',textposition='outside')])
fig.update_layout(title_text='Πτώση Τιμής vs Πλήθος Πωλήσεων',xaxis_title="Πτώση Τιμής (€)",
    yaxis_title="Πλήθος Πωλήσεων")
fig.show()

drop_price['difference'] = drop_price['retail_price'] - drop_price['price']
drop_price['discount'] = drop_price['difference']/drop_price['retail_price'] *100
plt.figure(figsize=(12,6))
sns.distplot(drop_price['discount']);
plt.title('Distribution of Discount');
```

Παράρτημα 7 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "retail_price", "price" (3/3)

```
sales_data["units_sold"].sum()

sales_data["rating_count"].sum()

sales_data["rating_one_count"].sum()

sales_data["rating_two_count"].sum()

sales_data["rating_three_count"].sum()

sales_data["rating_four_count"].sum()

sales_data["rating_five_count"].sum()

count1=((sales_data["rating_one_count"].sum())/(sales_data["units_sold"].sum()))*100
count1

count2=((sales_data["rating_two_count"].sum())/(sales_data["units_sold"].sum()))*100
count2

count3=((sales_data["rating_three_count"].sum())/(sales_data["units_sold"].sum()))*100
count3

count4=((sales_data["rating_four_count"].sum())/(sales_data["units_sold"].sum()))*100
count4

count5=((sales_data["rating_five_count"].sum())/(sales_data["units_sold"].sum()))*100
count5
```

Παράρτημα 8 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold","rating_count", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (1/5)

```
rating_0_count=(sales_data["units_sold"].sum())-(sales_data["rating_count"].sum())
rating_0_count

count0=((rating_0_count)/(sales_data["units_sold"].sum()))*100
count0

values=['Star1', 'Star2', 'Star3', 'Star4', 'Star5', 'No_Rating']
y=[2.14,1.42,3.01,4.02,9.9,79.49]
fig = go.Figure([go.Bar(x=values, y=y,text=y,
           texttemplate = "%{value:,s} %",textposition = "outside")])
fig.update_layout(title_text='Ποσοστό Αξιολογήσεων ως προς Πλήθος Πωλήσεων / Κατηγορία Αξιολογήσεων',xaxis_title="Κατηγορία Αξιολ
    yaxis_title="Ποσοστό Αξιολογήσεων")
fig.show()

rating1=((sales_data["rating_one_count"].sum())/(sales_data["rating_count"].sum()))*100
rating1

rating2=((sales_data["rating_two_count"].sum())/(sales_data["rating_count"].sum()))*100
rating2

rating3=((sales_data["rating_three_count"].sum())/(sales_data["rating_count"].sum()))*100
rating3

rating4=((sales_data["rating_four_count"].sum())/(sales_data["rating_count"].sum()))*100
rating4

rating5=((sales_data["rating_five_count"].sum())/(sales_data["rating_count"].sum()))*100
rating5
```

Παράρτημα 9 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold","rating_count", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (2/5)

```
values=['Star1', 'Star2', 'Star3', 'Star4', 'Star5']
y=[10.45,6.95 ,14.69, 19.60, 48.28]
fig = go.Figure([go.Bar(x=values, y=y,text=y,
           texttemplate = "%{value:,s} %",textposition = "outside")])
fig.update_layout(title_text='Ποσοστό Αξιολογήσεων ως προς Πλήθος Αξιολογηθέντων Πωλήσεων',xaxis_title="Κατηγορία Αξιολογήσεων",
fig.show()
```

Παράρτημα 10 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold","rating_count", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (3/5)

```
def num_units_sold(units_sold):
    units_sold = int(units_sold)
    bracket = ''
    if units_sold in range(0, 100):
        bracket = '< 100'
    if units_sold in range(100, 1000):
        bracket = '100 - 1000'
    if units_sold in range(1000, 5000):
        bracket = '1000 - 5000'
    if units_sold in range(5000, 10000):
        bracket = '5000 - 10000'
    if units_sold in range(10000, 20000):
        bracket = '10000 - 20000'
    if units_sold in range(20000, 50000):
        bracket = '20000 - 50000'
    if units_sold in range(50000, 100000):
        bracket = '> 50000'
    return bracket

rating_units_sold=sales_data[["units_sold","rating_five_count","rating_four_count","rating_three_count","rating_two_count","ratir
rating_units_sold["num_units_sold"] = rating_units_sold["units_sold"].apply(num_units_sold)
units_sold = rating_units_sold['num_units_sold'].unique().tolist()
units_sold_groupby = rating_units_sold.groupby("num_units_sold").agg({'rating_five_count': 'sum','rating_four_count': 'sum',
'rating_three_count': 'sum','rating_two_count': 'sum','rating_one_count': 'sum','rating_count':'sum'})
units_sold_groupby.reset_index()
units_sold_groupby = units_sold_groupby.iloc[1:]
units_sold_groupby=units_sold_groupby.reset_index()
units_sold_groupby
```

Παράρτημα 11 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold","rating_count", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (4/5)

```
values=["< 100", "100 - 1000", "1000 - 5000", "5000 - 10000",
              "10000 - 20000", "20000 - 50000", "> 50000"]

fig = go.Figure(data=[
    go.Bar(name='rating_five_count', x=values, y=[225.0, 7529.0, 4169.0, 90094.0, 164130.0, 226350.0, 88685.0],
           texttemplate='%{y:.2s}',textposition = "inside"),
    go.Bar(name='rating_four_count', x=values, y=[89.0, 3151.0, 21729.0, 36015.0, 65087.0, 93245.0, 35783.0],
           texttemplate='%{y:.2s}',textposition = "inside"),
    go.Bar(name='rating_three_count', x=values, y=[58.0, 2328.0, 16041.0, 27271.0, 46207.0, 70335.0, 27439.0],
           texttemplate='%{y:.2s}',textposition = "inside"),
    go.Bar(name='rating_two_count', x=values, y=[40.0, 1187.0, 7769.0, 13327.0, 21399.0, 33191.0, 12960.0],
           texttemplate='%{y:.2s}',textposition = "inside"),
    go.Bar(name='rating_one_count', x=values, y=[47.0, 2264.0, 13459.0, 21971.0, 32677.0, 48738.0, 17576.0],
           texttemplate='%{y:.2s}',textposition = "inside")
])

fig.update_layout(title_text='Πλήθος Αξιολογήσεων ανά Έυρος Αριθμού Πωλησθέντων Μονάδων',
legend_title="Κατηγορίες Πλήθους Αξιολογήσεων")
fig.update_layout(barmode='stack')
fig.show()

m=sales_data[sales_data['units_sold']>50000]
m

ax = sns.barplot(x="units_sold", y="rating_count", data=sales_data,
                 palette="Blues_d")
```

Παράρτημα 12 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "rating_count", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (5/5)

```
plt.figure(figsize=(12,6))
ad_sales = (sales_data.groupby(['units_sold'])['uses_ad_boosts'].value_counts(normalize=True).rename('Percentage').mul(100)
            .reset_index())

sns.barplot(x='units_sold', y='Percentage', hue='uses_ad_boosts', data=ad_sales, palette="Blues", alpha=0.8)
plt.legend(title='uses_ad_boosts', loc=(0.07,0.81))

data_rating=["rating_count","rating_one_count","rating_two_count","rating_three_count","rating_four_count",
             "rating_five_count","units_sold"]
data_ad=sales_data[data_rating+['uses_ad_boosts']]

data_ad=data_ad.groupby('uses_ad_boosts').sum().reset_index()
data_ad

data_ad=data_ad.groupby('uses_ad_boosts').sum().reset_index()
data_ad["rating1_percent"]=(data_ad["rating_one_count"]/(data_ad["rating_count"].sum()))*100
data_ad["rating2_percent"]=(data_ad["rating_two_count"]/(data_ad["rating_count"].sum()))*100
data_ad["rating3_percent"]=(data_ad["rating_three_count"]/(data_ad["rating_count"].sum()))*100
data_ad["rating4_percent"]=(data_ad["rating_four_count"]/(data_ad["rating_count"].sum()))*100
data_ad["rating5_percent"]=(data_ad["rating_five_count"]/(data_ad["rating_count"].sum()))*100
data_ad

data_ad["units_sold_percent"]=((data_ad["units_sold"])/(data_ad["units_sold"].sum()))*100
data_ad[["uses_ad_boosts","units_sold_percent","rating1_percent","rating2_percent","rating3_percent",
         "rating4_percent","rating5_percent"]].round(2)
```

Παράρτημα 13 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "uses_ad_boosts", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (1/2)

```
values=['Without Add Boosts','With Add Boosts']

fig = go.Figure(data=[
    go.Bar(name='units_sold_percent', x=values, y=[58.54,41.46],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating1_percent', x=values, y=[6.41, 4.04],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating2_percent', x=values, y=[4.32, 2.64],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating3_percent', x=values, y=[9.06, 5.64],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating4_percent', x=values, y=[12.10, 7.51],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating5_percent', x=values, y=[29.77, 18.52],texttemplate = "%{value:,s} %",textposition = "outside")
])

fig.update_layout(title_text='Ποσοστό Πωλήσεων & Αξιολογήσεων σε χρήση ή όχι Διαφήμισης')
fig.update_layout(barmode='group')
fig.show()
```

Παράρτημα 14 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "uses_ad_boosts", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count" (2/2)

```
shipping_price=sales_data[["shipping_option_price","units_sold"]]
shipping_price=shipping_price.groupby(["shipping_option_price"]).sum().reset_index()

labels = ['1€','2€','3€','4€','5€','6€','7€','12€']
values = [1369910, 2759600, 2312910, 218370,86580,48500,10300,100]

fig = go.Figure(data=[go.Pie(labels=labels, values=values,textinfo='label+percent',
                             insidetextorientation='radial')])
fig.update_layout(title="Έξοδα Αποστολής vs Πωλήσεις",legend_title="Κατηγορίες Εξόδων Αποστολής",
    font=dict(
        family="Courier New, monospace",
        size=18,
        color="RebeccaPurple"))
fig.show()
```

Παράρτημα 15 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "shipping_option_price"

```
target_countries = sales_data.countries_shipped_to.unique()
fig, ax = plt.subplots(figsize=(15,6))
sns.scatterplot(x=sales_data.countries_shipped_to,y=sales_data.units_sold, s=100, alpha=0.8)

shipping_name=sales_data.groupby('shipping_option_name')['units_sold'].sum().reset_index()
shipping_name.rename(columns={'units_sold': 'Sum of Units Sold'},inplace=True)
shipping_name=shipping_name.sort_values(by="Sum of Units Sold",ascending=False)
shipping_name=shipping_name.head()
shipping_name

labels = ['Livraison standard','Standard Shipping','Standardowa wysyłka','Envio Padrão','Envío normal']
values = [6591720, 88550, 30100, 22400,16100]

fig = go.Figure(data=[go.Pie(labels=labels, values=values,textinfo='label+percent',
                             insidetextorientation='radial')])
fig.update_layout(
    title="Ποσοστό Πωλήσεων <br>στις πιο Συχνές Μεταφορικές",
    legend_title="Μεταφορικές Εταιρίες",
    )
fig.show()

index,name=sales_data['shipping_option_name'].factorize()
sales_data['shipping_option_index']=index
```

Παράρτημα 16 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "shipping_option_name"

```
corr_map=sales_data[['badge_fast_shipping','shipping_option_index','shipping_option_price','shipping_is_express','countries_shipp
corr_map=corr_map.corr()
plt.figure(figsize=(10,5))
sns.heatmap(corr_map,annot=True,cmap='Blues')
plt.xticks(rotation=45,fontsize=10)
plt.yticks(rotation=45,fontsize=10)
plt.show()

express_shipping=sales_data.groupby('shipping_is_express')['units_sold'].sum()
express_shipping=express_shipping.reset_index().sort_values(by='units_sold',ascending=False)
express_shipping

labels = ['Shipping Express','No Shipping Express']
values = [11200, 6795070]

fig = go.Figure(data=[go.Pie(labels=labels, values=values,textinfo='label+percent',
                             insidetextorientation='radial')])
fig.update_layout(
    title="Ποσοστό Πωλήσεων <br> Με Γρήγορη Αποστολή",
    legend_title="Shipping Express",
    )
fig.show()
```

Παράρτημα 17 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "shipping_option_index","shipping_option_price","shipping_is_express","countries_shipping"

```
tags_data = pd.read_csv("unique_categories.sorted_by_count.csv")

fig_dims = (10, 5)
fig, ax = plt.subplots(figsize=fig_dims)
sns.barplot(x = 'keyword',
            y = 'count',
            data = tags_data.iloc[:20],
            ax = ax)
ax.set(xlabel='Keyword', ylabel='Count')
plt.xticks(rotation=45, ha='right')
plt.show()

sales_data['tags_count'] = sales_data['tags'].str.split(',').str.len()

fig = plt.gcf()
fig.set_size_inches( 10, 5)
sns.lineplot(data=sales_data, x="tags_count", y="units_sold", ci=None)

df=sales_data[sales_data['tags_count']<=10]
df=df[["tags_count","units_sold"]]
df.groupby(["units_sold"]).count().reset_index()
```

Παράρτημα 18 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold" , "tags_count"

```
merchant = sales_data[["merchant_rating_count",
       'merchant_rating','units_sold']]
sns.heatmap(merchant.corr(),cmap='CMRmap_r',annot=True)
plt.title('Merchant vs Sales')

sns.set(font_scale=1.2)
plt.figure(figsize=(12,6))
sns.barplot(y='merchant_rating_count', x='units_sold', data=sales_data)
```

Παράρτημα 19 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold" , "merchant_rating_count" (1/2)

```
plt.figure(figsize=(12,6))
sns.scatterplot(x=sales_data.merchant_rating, y=sales_data.units_sold, s=80)

merchant['merchant_rating_score'] =(merchant['merchant_rating']*merchant['merchant_rating_count'])/
                                   ((merchant['merchant_rating']*merchant['merchant_rating_count']).max())
merchant=merchant.groupby(["units_sold"]).mean().reset_index()
merchant=round(merchant,3)
merchant

merchant["merchant_rating_score"]=merchant["merchant_rating_score"]*100
merchant

values=['10', '50', '100', '1000', '5000', '10000','20000','50000','100000']
y=[0.5,0.4,0.6,0.8,1.4,2.2,2.8,5.5,9.2]
fig = go.Figure([go.Bar(x=values, y=y,text=y,
            texttemplate = "%{y:,2s} %",textposition = "outside")])
fig.update_layout(title_text='Μέσο Σκορ Αξιολογήσεων Εμπόρων ως προς Πωλήσεις Προιόντων',xaxis_title="Πωλήσεις Προιόντων",
    yaxis_title="Σκορ Αξιολογήσεων Εμπόρων")
fig.show()
```

Παράρτημα 20 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "merchant_rating_count" (2/2)

```
plt.figure(figsize=(12,6))
merch_pic = (sales_data.groupby(['units_sold'])['merchant_has_profile_picture'].value_counts(normalize=True).rename('Percentage'
            .reset_index())
p = sns.barplot(x='units_sold', y='Percentage', hue='merchant_has_profile_picture', data=merch_pic, alpha=0.8)
```

Παράρτημα 21 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "merchant_has_profile_picture"

```
inventory=sales_data[["inventory_total","units_sold"]]
inventory=inventory.groupby(["inventory_total"]).sum().reset_index()
inventory
```

Παράρτημα 22 Μελέτη Δεδομένων - Αθροιστικός πίνακας του "inventory_total", "units_sold"

```
sales_data[sales_data['badges_count']!=0].head(10)
badges1=sales_data[['badges_count','badge_local_product', 'badge_product_quality', 'badge_fast_shipping']]

badges_cats=[]

for i in badges1.index:
    categories = ['badge_local_product', 'badge_product_quality', 'badge_fast_shipping']
    codes = badges1.loc[[i],['badge_local_product', 'badge_product_quality', 'badge_fast_shipping']].values.reshape(3,).tolist()
    zipped = zip(codes,categories)
    my_cats=[]
    for m,n in list(zipped):
        my_cats.append(m*n)
    badges_cats.append(my_cats)
badges_cats = pd.Series((v[0]+v[1]+v[2] for v in badges_cats))

badges1.drop(columns=['badge_local_product', 'badge_product_quality', 'badge_fast_shipping'],inplace=True)
badges1['badges_cats']=badges_cats.values
badges1['records']=np.ones((1539,))
badges_data=badges1.groupby(['badges_count','badges_cats']).count().reset_index()
badges_data

badge_local_product_percent=((sales_data["badge_local_product"].sum())/((sales_data["badges_count"]).sum()))*100
badge_local_product_percent.round(0)

badge_product_quality_percent=((sales_data["badge_product_quality"].sum())/((sales_data["badges_count"]).sum()))*100
badge_product_quality_percent.round(0)
```

Παράρτημα 23 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold",
"badges_count","badge_local_product","badge_product_quality","badge_fast_shipping" (1/3)

```
badge_fast_shipping_percent=((sales_data["badge_fast_shipping"].sum())/((sales_data["badges_count"]).sum()))*100
badge_fast_shipping_percent.round(0)

no_badges=sales_data[["badges_count","units_sold"]]
no_badges=no_badges[no_badges["badges_count"]==0]
no_badges=no_badges.groupby(["badges_count"]).sum().reset_index()
a=no_badges["units_sold"].sum()
((a/(sales_data["units_sold"].sum()))*100).round(0)

badges=sales_data[["badges_count","units_sold"]]
badges=badges[badges["badges_count"]!=0]
badges=badges.groupby(["badges_count"]).sum().reset_index()
b=badges["units_sold"].sum()
((b/(sales_data["units_sold"].sum()))*100).round(0)

values=['Without badges','With badges']

fig = go.Figure(data=[
    go.Bar(name='units_sold_percent', x=values, y=[87.0,13.0],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='badge_local_product_percent', x=values, y=[0, 18.0],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='badge_product_quality_percent', x=values, y=[0, 70.0],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='badge_fast_shipping_percent', x=values, y=[0, 12.0],texttemplate = "%{value:,s} %",textposition = "outside")
])

fig.update_layout(barmode='group')
fig.update_layout(title_text='Ποσοστό Πωλήσεων & Σημάτων')
fig.show()
```

Παράρτημα 24 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "badges_count","badge_local_product","badge_product_quality","badge_fast_shipping" (2/3)

```
params = ['badges_count', 'badge_local_product', 'badge_product_quality', 'badge_fast_shipping', 'units_sold']
badge_correlation = sales_data[params].corr().round(2)

plt.figure(figsize=(12,6))
badge_sales = (sales_data.groupby(['units_sold'])['badges_count'].value_counts(normalize=True).rename('Percentage').mul(100)
            .reset_index())

sns.barplot(x='units_sold', y='Percentage', hue='badges_count', data=badge_sales, palette="terrain_r")
plt.legend(title='badges_count', loc=(1,0.5))


sales_data['has_urgency_banner'] = sales_data['has_urgency_banner'].fillna(0)
```

Παράρτημα 25 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "badges_count","badge_local_product","badge_product_quality","badge_fast_shipping" (3/3)

```
data_banner=["rating_count","rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count",
             "units_sold"]
data_banner=sales_data[data_banner+['has_urgency_banner']]

data_banner=data_banner.groupby('has_urgency_banner').sum().reset_index()
data_banner["rating1_percent"]=(data_banner["rating_one_count"]/(data_banner["rating_count"].sum()))*100
data_banner["rating2_percent"]=(data_banner["rating_two_count"]/(data_banner["rating_count"].sum()))*100
data_banner["rating3_percent"]=(data_banner["rating_three_count"]/(data_banner["rating_count"].sum()))*100
data_banner["rating4_percent"]=(data_banner["rating_four_count"]/(data_banner["rating_count"].sum()))*100
data_banner["rating5_percent"]=(data_banner["rating_five_count"]/(data_banner["rating_count"].sum()))*100
data_banner["units_sold_percent"]=((data_banner["units_sold"])/(data_banner["units_sold"].sum()))*100
data_banner

data_banner[["has_urgency_banner","units_sold_percent","rating1_percent","rating2_percent",
             "rating3_percent","rating4_percent","rating5_percent"]].round(2)
values=['Without urgency_banner','With urgency_banner']
fig = go.Figure(data=[
    go.Bar(name='units_sold_percent', x=values, y=[72.37,27.63],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating1_percent', x=values, y=[7.25,3.20],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating2_percent', x=values, y=[4.83,2.13],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating3_percent', x=values, y=[10.29,4.40],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating4_percent', x=values, y=[13.84,5.77],texttemplate = "%{value:,s} %",textposition = "outside"),
    go.Bar(name='rating5_percent', x=values, y=[34.25,14.03],texttemplate = "%{value:,s} %",textposition = "outside")
])
fig.update_layout(title_text='Ποσοστό Πωλήσεων-Αξιολογήσεων με/χωρίς Πανό')
fig.update_layout(barmode='group')
fig.show()
```

Παράρτημα 26 Μελέτη Δεδομένων – Γράφημα μεταξύ των "units_sold", "has_urgency_banner", "rating_one_count","rating_two_count","rating_three_count","rating_four_count","rating_five_count"

```
color_sale=sales_data.groupby('product_color')['units_sold'].sum()
color_sale=color_sale.reset_index().sort_values(by='units_sold',ascending=False)

top_10_color_sale=color_sale.head(10)

fig=px.bar(data_frame=top_10_color_sale,
      x='product_color',
      y='units_sold')
fig.update_layout(title='Top 10 color sales')
fig.show()

size_sale=sales_data.groupby('product_variation_size_id')['units_sold'].sum()
size_sale=size_sale.reset_index().sort_values(by='units_sold',ascending=False)

top_10_size_sale=size_sale.head()

fig=px.bar(data_frame=top_10_size_sale,
      x='product_variation_size_id',
      y='units_sold')
fig.update_layout(title='Top 5 size sales')
fig.show()

a=sales_data['origin_country'].value_counts().rename('item_count').reset_index()
values=['CN', 'US', 'VE',"SG","GB","AT"]
y=[1484,31,4,2,1,1]
fig = go.Figure([go.Bar(x=values, y=y,text=y,
           texttemplate='%{text:.2s}',textposition='outside')])
fig.update_layout(title_text='Καταγωγή Προϊόντων',xaxis_title="Χώρα προέλευσης",
    yaxis_title="Πλήθος Προϊόντων")
fig.show()
```

```
geo_sales = (sales_data.groupby(['units_sold'])['origin_country'].value_counts().rename('item_count').reset_index())

sns.barplot(x='units_sold', y='item_count', hue='origin_country', data=geo_sales, alpha=0.8)
plt.legend(title='origin_country', loc='upper right')
plt.ylim(0,80)
```

Παράρτημα 27 Μελέτη Δεδομένων – Γραφήματα μεταξύ των "units_sold", "product_color","product_variation_size_id","origin_country"

```
wish_df=sales_data.copy()

from wordcloud import WordCloud

good_sales = wish_df['units_sold'] >= 1000
# with this we can focus only on those items that are sold considerably
alltitle = ''
alltitle_total = ''
for i, j in enumerate(wish_df[good_sales].title_orig):
    sold_units = wish_df[good_sales]['units_sold'].iloc[i]/1000
    # title_orig only focuses on the keywords
    # it doesn't say how much it is sold
    # so in order to understand the extent of popularity of a keyword,
    # it needs to multiplied by the times it's sold
    # to make things more efficient we divide it by 1000
    alltitle += ' ' + j
    # alltitle makes a huge sentence composed of all the words in the title combined together
    alltitle_total += (' ' + j)*int(sold_units)
    # alltitle_total has each word multiplied by the number of products sold

plt.figure(figsize=(16,16))
cloud = WordCloud(max_words=100, background_color="white").generate(alltitle)
plt.imshow(cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Παράρτημα 28 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "title_orig" (1/2)

```
plt.figure(figsize=(16,16))
cloud = WordCloud(max_words=100, stopwords=['Women', 'Summer', 'Fashion'], background_color="black").generate(alltitle_total)
plt.imshow(cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Παράρτημα 29 Μελέτη Δεδομένων - Γράφημα μεταξύ των "units_sold", "title_orig" 2/2

8.2 Παράρτημα Β'- Επεξεργασία Δεδομένων

```
!pip install seaborn
!pip install plotly
!pip install matplotlib_venn
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
import plotly
import plotly.graph_objects as go
import plotly.express as px
import matplotlib.pyplot as plt
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
from plotly.subplots import make_subplots
from plotly.offline import iplot, init_notebook_mode
init_notebook_mode(connected = True)
!pip install cufflinks plotly
from plotly.offline import iplot, init_notebook_mode
import cufflinks
cufflinks.go_offline(connected=True)
init_notebook_mode(connected=True)

sales_data=pd.read_csv("summer_products_with_rating_and_performance_2020_08.csv")
sales_data

pd.set_option('display.max_columns', None)
sales_data.head()
```

Παράρτημα 30 Εισαγωγή Βιβλιοθηκών & Datasets

```
print('Dimensions of the df', sales_data.shape)
sales_data = sales_data.drop_duplicates()
print('Dimensions of the df after dropping the duplicates', sales_data.shape)

print("Duplicate product_id :",sales_data['product_id'].duplicated().sum())

sales_data['currency_buyer'].unique()

sales_data[["has_urgency_banner", "urgency_text"]]
sales_data['urgency_text'].value_counts()
sales_data[sales_data['urgency_text'].isna()]


sales_data.drop(["urgency_text"], axis=1, inplace=True)

sales_data["has_urgency_banner"] = sales_data["has_urgency_banner"].fillna(0)

count = sales_data['product_color'].value_counts()
count

np.sort(sales_data['product_color'].dropna().unique())
sales_data[sales_data['product_color'].str.contains('&', na=False)]['product_color'].unique()
```

Παράρτημα 31 Επεξεργασία των δεδομένων (1/6)

```
shade_to_colour = {
    'navyblue': 'blue', 'lightblue': 'blue', 'skyblue': 'blue', 'lakeblue': 'blue', 'darkblue': 'blue', 'denimblue': 'blue', 'nav
    'navy': 'blue',
    'armygreen': 'green', 'army green': 'green', 'fluorescentgreen': 'green', 'mintgreen': 'green', 'light green': 'green', 'ligh
    'applegreen': 'green', 'darkgreen': 'green', 'army': 'green', 'khaki': 'green', 'lightkhaki': 'green',
    'lightyellow': 'yellow',
    'winered': 'red', 'wine red': 'red', 'lightred': 'red', 'coralred': 'red', 'rose red': 'red', 'watermelonred': 'red', 'orange
    'claret': 'red', 'burgundy': 'red',
    'gray': 'grey', 'silver': 'grey','lightgray': 'grey', 'lightgrey': 'grey', 'greysnakeskinprint': 'grey',
    'coffee': 'brown', 'camel': 'brown', 'tan': 'brown',
    'offwhite': 'white', 'ivory': 'white', 'nude': 'white',
    'lightpink': 'pink', 'dustypink':'pink', 'rosegold': 'pink',
    'lightpurple': 'purple', 'coolblack': 'black', 'apricot': 'orange', 'offblack': 'black'
}

def update_color(col):
    if shade_to_colour.get(col, False):
        return shade_to_colour.get(col)
    elif '&' in col:
        return 'dual'
    elif col in shade_to_colour.values():
        return col
    else:
        return 'other'

sales_data['product_color'].replace(np.nan, 'others', inplace=True)

sales_data['product_color'] = sales_data.product_color.apply(update_color)
```

Παράρτημα 32 Επεξεργασία των δεδομένων (2/6)

```
count = sales_data['product_color'].value_counts()
count

sales_data['product_variation_size_id'].unique()
sales_data['product_variation_size_id'].value_counts().head(50)
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].str.lower()
.str.replace('.', '').str.replace('size--', '').str.replace('size -', '').str.replace('size/', '')
.str.replace('size ', '').str.replace('size-', '')

sales_data['product_variation_size_id'].unique()
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('2xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('3xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('4xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('5xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('6xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('x   l', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('sizel', 'l')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('size4xl', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('x   l', 'xl')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace('1 pc - xl', 'xl')

def change_size(cl):
    if cl in 'xl,l,s,xs,m,xxl,xxxs,xxxxxl,xxxxl'.split(','):
        return cl
    else:
        return 'other'

sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].replace(np.nan, 'OTHER')
sales_data['product_variation_size_id'] = sales_data['product_variation_size_id'].apply(change_size)
sales_data['product_variation_size_id'].unique()
```

Παράρτημα 33 Επεξεργασία των δεδομένων (3/6)

```
list_of_na_merchants = sales_data[sales_data['origin_country'].isna()]['merchant_id'].values
for m in list_of_na_merchants:
    print("merchant title " + m)
    print(sales_data[sales_data['merchant_id']==m]['origin_country'])

sales_data['origin_country'].fillna('CN', inplace=True)

sales_data[sales_data['merchant_name'].isna()]
sales_data["merchant_name"].unique()
sales_data.drop(["merchant_name"], axis=1, inplace=True)

sales_data['merchant_info_subtitle'].value_counts().reset_index()
sales_data.drop(["merchant_info_subtitle"], axis=1, inplace=True)

a=sales_data[sales_data['merchant_profile_picture'].isna()]
a.shape
b=sales_data[sales_data['merchant_has_profile_picture']==0]
b.shape
sales_data.drop(["merchant_profile_picture"], axis=1, inplace=True)

sales_data[sales_data['rating_count']==0].reset_index()
sales_data[sales_data['rating']==5][sales_data['rating_count']==0]
sales_data.update(sales_data[['rating_five_count', 'rating_four_count', 'rating_three_count',
                              'rating_two_count', 'rating_one_count']].fillna(0))
sales_data.loc[sales_data['rating_count']==0, 'rating'] = 0
```

Παράρτημα 34 Επεξεργασία των δεδομένων (4/6)

```
outliers=sales_data[sales_data["merchant_rating_count"]>24564]
outliers['merchant_rating_count'].count()

mean = sales_data['merchant_rating_count'].mean()
sales_data['merchant_rating_count'] = sales_data['merchant_rating_count'].apply(lambda x: mean if x>24564 else x)
sns.boxplot(sales_data['merchant_rating_count'],color='yellow',showmeans=True)
fig = plt.gcf()
fig.set_size_inches(10,5)
plt.title('Rating Count')

target_col='units_sold'
plt.figure(figsize=(6, 6))
heatmap = sns.heatmap(sales_data.corr()[[target_col]].sort_values(by=target_col, ascending=False),
                      vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title(f'Features Correlating with {target_col}', fontdict={'fontsize':18}, pad=16);

sales_data['success'] = sales_data['units_sold'].apply(lambda x: 1 if x>1000 else 0)
sns.barplot(x='success', y='rating_five_count', data=sales_data)
sns.barplot(x='success', y='rating_one_count', data=sales_data)

sales_data['rating_three_count_prop'] =sales_data['rating_three_count']/sales_data['rating_count']
sales_data['rating_four_count_prop'] = sales_data['rating_four_count']/sales_data['rating_count']
sales_data['rating_five_count_prop'] = sales_data['rating_five_count']/sales_data['rating_count']
sales_data['rating_two_count_prop'] = sales_data['rating_two_count']/sales_data['rating_count']
sales_data['rating_one_count_prop'] = sales_data['rating_one_count']/sales_data['rating_count']
sales_data.update(sales_data[['rating_five_count_prop', 'rating_four_count_prop',
                              'rating_three_count_prop','rating_two_count_prop', 'rating_one_count_prop']].fillna(0))
sns.barplot(x='success', y='rating_one_count_prop', data=sales_data)
sns.barplot(x='success', y='rating_five_count_prop', data=sales_data)
```

Παράρτημα 35 Επεξεργασία των δεδομένων (5/6)

```
ax = sales_data['success'].value_counts().plot(kind='bar', figsize=(10, 6), fontsize=13)
ax.set_title('Success (0 = no success, 1 = success)', size=20, pad=20)
for i in ax.patches:
    ax.text(i.get_x() + 0.19, i.get_height() + 10, str(round(i.get_height(), 2)), fontsize=15)

sales_data = sales_data.drop(['crawl_month','product_id','product_picture', 'product_url','merchant_id',
                      'currency_buyer', 'theme','merchant_title',
                     'title','title_orig','tags', 'shipping_option_name', "inventory_total" , "badge_fast_shipping" ,
                      "badge_local_product" , "shipping_is_express", "units_sold","rating_five_count","rating_four_count",
                            "rating_three_count", "rating_two_count", "rating_one_count"], axis = 1)
sales_data.isna().sum()

dummy_product_color = pd.get_dummies(sales_data['product_color'], prefix='color_', drop_first=True)
dummy_size = pd.get_dummies(sales_data['product_variation_size_id'], prefix='size_', drop_first=True)
dummy_country = pd.get_dummies(sales_data['origin_country'], prefix='country_', drop_first=True)

to_drop = ['product_color', 'product_variation_size_id', 'origin_country']
sales_data.drop(to_drop, axis=1, inplace=True)

sales_data_new = pd.concat([sales_data, dummy_product_color, dummy_size, dummy_country], axis=1)
sales_data_new.head()

sales_data_new.to_csv (r'C:\Users\User\Desktop\jupy\sales_data_new.csv', index = False, header=True)

print (sales_data_new)
```

Παράρτημα 36 Επεξεργασία των δεδομένων (6/6)

8.3 Παράρτημα Γ'- Μηχανική Μάθηση

```
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
import plotly
import plotly.graph_objects as go
import plotly.express as px
import matplotlib.pyplot as plt
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
from plotly.subplots import make_subplots
from plotly.offline import iplot, init_notebook_mode
init_notebook_mode(connected = True)
!pip install cufflinks plotly
from plotly.offline import iplot, init_notebook_mode
import cufflinks
from collections import Counter
cufflinks.go_offline(connected=True)
init_notebook_mode(connected=True)
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.feature_selection import VarianceThreshold, SelectFromModel
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import accuracy_score, classification_report, f1_score, make_scorer
```

Παράρτημα 37 Εισαγωγή βιβλιοθηκών (1/2)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression, LogisticRegression, Lasso
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.metrics import roc_auc_score,roc_curve
import sklearn.metrics as metrics
from sklearn import feature_selection
```

Παράρτημα 38 Εισαγωγή βιβλιοθηκών (2/2)

```
sales_data_new=pd.read_csv("sales_data_new.csv")
sales_data_new
pd.set_option('display.max_columns', None)
sales_data_new.head()
X =sales_data_new.drop(columns=["success"])
y = sales_data_new["success"]

from collections import Counter
print(Counter(y_train))
print(Counter(y_test))

y_t = pd.DataFrame(y_train)
Bar_chart=sns.countplot(data=y_t, x='success', hue="success", palette=['#432371',"#FAAE7B"])
Bar_chart.set_xticklabels(Bar_chart.get_xticklabels())
plt.title("Response Variable Ratio, Raw")
```

```
sm=SMOTE(random_state=42)
X_train_sm,y_train_sm=sm.fit_resample(X_train,y_train)
y_t = pd.DataFrame(y_train_sm)

from collections import Counter
counter= Counter (y_train)
print('Before',counter)
counter= Counter (y_train_sm)
print('After',counter)

Bar_chart=sns.countplot(data=y_t, x='success', hue="success", palette=['#432371',"#FAAE7B"])
Bar_chart.set_xticklabels(Bar_chart.get_xticklabels())
plt.title("Response Variable Ratio, SMOTE")

y_train_sm.shape
```

Παράρτημα 39 Εισαγωγή Τελικού Dataset - Εφαρμογή SMOTE

```
#Without resampling
pipe = Pipeline(steps=[
    ('scaler', StandardScaler()),('mlpclassifier', MLPClassifier(random_state=42))])


params = {
    'mlpclassifier__hidden_layer_sizes': [(100, 10), (200,10), (50),(100), (200,20), (200,50)],
    'mlpclassifier__activation': ['relu'],
    'mlpclassifier__learning_rate_init': [0.0001],
    'mlpclassifier__solver': ['adam']}

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
# Train the MLPClassifier
grid.fit(X_train,y_train)

#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)
# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)
```

Παράρτημα 40 MLP (1/7)

```
# Predict & Evaluate the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
# Classification report
print(classification_report(y_test, best_preds))
print("-"*80)
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)
```

Παράρτημα 41 MLP (2/7)

```
#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 42 MLP (3/7)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

score_auc=roc_auc_score(y_test, preds)
print("AUC for MLP_Classifier:%0.2f" % score_auc)

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 43 MLP (4/7)

```
#With resampling
pipe = Pipeline(steps=[
    ('scaler', StandardScaler()),('mlpclassifier', MLPClassifier(random_state=42))])

params = {
    'mlpclassifier__hidden_layer_sizes': [(100, 10), (200,10), (50),(100), (200,20), (200,50)],
    'mlpclassifier__activation': ['relu'],
    'mlpclassifier__learning_rate_init': [0.0001],
    'mlpclassifier__solver': ['adam']}

grid = GridSearchCV(pipe,params,cv = 3,n_jobs=-1, verbose=10, scoring = 'f1_micro')
# Train the MLPClassifier
grid.fit(X_train_sm,y_train_sm)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 44 MLP (Oversampling) (5/7)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 45 MLP (Oversampling) (6/7)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

score_auc=roc_auc_score(y_test, preds)
print("AUC for MLP_Classifier: %0.2f" % score_auc)

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 46 MLP (Oversampling) (7/7)

```
#without resampling
pipe = Pipeline(steps=[
    ('scaler', StandardScaler()),('kneighborsclassifier', KNeighborsClassifier())])

params= {"kneighborsclassifier__n_neighbors": [7,9,11,15,20,25,30,35]}

grid = GridSearchCV(pipe, params,n_jobs=-1, cv=3, verbose=10, scoring='accuracy')
grid.fit(X_train,y_train)


#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 47 KNN (1/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)
#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 48 KNN (2/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 49 KNN (3/6)

```
#with resampling
pipe = Pipeline(steps=[
    ('scaler', StandardScaler()),('kneighborsclassifier', KNeighborsClassifier())])

params= {
                "kneighborsclassifier__n_neighbors": [7,9,11,15,20,25,30,35]}

grid = GridSearchCV(pipe, params,n_jobs=-1, cv=3, verbose=10, scoring='accuracy')
grid.fit(X_train_sm,y_train_sm)


#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 50 KNN (Oversampling) (4/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 51 KNN (Oversampling) (5/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 52 KNN (Oversampling) (6/6)

```
pipe=make_pipeline(DecisionTreeClassifier(random_state=42))
pipe

#without resampling
pipe = Pipeline(steps=[
    ('scaler', StandardScaler()),('decisiontreeclassifier', DecisionTreeClassifier(random_state=42))])

params = {
    "decisiontreeclassifier__criterion": ["gini","entropy"],
    "decisiontreeclassifier__splitter": ["best","random"],
    "decisiontreeclassifier__max_depth": [None,100,50,20,10,5]
}

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
grid.fit(X_train,y_train)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 53 Decision Tree (1/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 54 Decision Tree (2/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

d = {'Stats':X_train.columns,'FI':grid.best_estimator_.named_steps["decisiontreeclassifier"].feature_importances_}
df = pd.DataFrame(d)
df
df.sort_values(by=['FI'], ascending=True, inplace=True)
df.set_index('Stats', inplace=True)

df.plot(kind='barh', figsize=(28, 15))

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 55 Decision Tree (3/6)

```
pipe = Pipeline(steps=[
    ('scaler', StandardScaler()),('decisiontreeclassifier', DecisionTreeClassifier(random_state=42))])

params = {
    "decisiontreeclassifier__criterion": ["gini","entropy"],
    "decisiontreeclassifier__splitter": ["best","random"],
    "decisiontreeclassifier__max_depth": [None,100,50,20,10,5]
}

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
grid.fit(X_train_sm,y_train_sm)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 56 Decision Tree (Oversampling) (4/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 57 Decision Tree (Oversampling) (5/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

d = {'Stats':X_train_sm.columns,'FI':grid.best_estimator_.named_steps["decisiontreeclassifier"].feature_importances_}
df = pd.DataFrame(d)
df
df.sort_values(by=['FI'], ascending=True, inplace=True)
df.set_index('Stats', inplace=True)

df.plot(kind='barh', figsize=(28, 15))

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 58 Decision Tree (Oversampling) (6/6)

```
pipe=make_pipeline(LogisticRegression(random_state=42))
pipe
#without resampling
pipe = Pipeline(steps=[('scaler', StandardScaler()),('logisticregression', LogisticRegression(random_state=42))])

params = {'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100, 1000] }

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
grid.fit(X_train,y_train)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 59 Logistic Regression (1/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 60 Logistic Regression (2/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

# get importance
importance = grid.best_estimator_.named_steps["logisticregression"].coef_[0]
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
fig, ax = plt.subplots(figsize=(28, 14))
#f, ax = plt.subplots(figsize=(18,5))
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 61 Logistic Regression (3/6)

```
#with resampling
pipe = Pipeline(steps=[('scaler', StandardScaler()),('logisticregression', LogisticRegression(random_state=42))])

params = {'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100, 1000] }

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
grid.fit(X_train_sm,y_train_sm)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 62 Logistic Regression (Oversampling) (4/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 63 Logistic Regression (Oversampling) (5/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

# get importance
importance = grid.best_estimator_.named_steps["logisticregression"].coef_[0]
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
fig, ax = plt.subplots(figsize=(28, 14))
#f, ax = plt.subplots(figsize=(18,5))
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 64 Logistic Regression (Oversampling) (6/6)

```
pipe=make_pipeline(GaussianNB())
pipe
#without resampling
pipe = Pipeline(steps=[('scaler', StandardScaler()),('gaussiannb', GaussianNB())])
params = {
    'gaussiannb__priors': [None],
    'gaussiannb__var_smoothing': [0.00000001, 0.000000001, 0.000000001]
}

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
grid.fit(X_train,y_train)

print("-"*80)
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 65 GaussianNB (1/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 66 GaussianNB (2/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 67 GaussianNB (3/6)

```
#with resampling
pipe = Pipeline(steps=[('scaler', StandardScaler()),('gaussiannb', GaussianNB())])
params = {
    'gaussiannb__priors': [None],
    'gaussiannb__var_smoothing': [0.00000001, 0.000000001, 0.00000001]
}

grid = GridSearchCV(pipe,params,n_jobs=-1,cv = 3, verbose=10, scoring = 'accuracy')
grid.fit(X_train_sm,y_train_sm)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 68 GaussianNB (Oversampling) (4/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 69 GaussianNB (Oversampling) (5/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 70 GaussianNB (Oversampling) (6/6)

```
pipe=make_pipeline(feature_selection.SelectKBest())
pipe
#without resampling
pipe = Pipeline(steps=[('scaler', StandardScaler()),('randomforestclassifier',RandomForestClassifier(random_state=42))])

params = {
    'randomforestclassifier__n_estimators': [200, 500],
    'randomforestclassifier__max_features': ['auto', 'sqrt', 'log2'],
    'randomforestclassifier__max_depth' : [4,5,6,7,8],
    'randomforestclassifier__criterion' :['gini', 'entropy']
}

grid = GridSearchCV(pipe, params,cv= 3,n_jobs=-1, verbose=10,scoring='accuracy')
grid.fit(X_train,y_train)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 71 Random Forest (1/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 72 Random Forest (2/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

d = {'Stats':X_train.columns,'FI':grid.best_estimator_.named_steps["randomforestclassifier"].feature_importances_}
df = pd.DataFrame(d)
df

df.sort_values(by=['FI'], ascending=True, inplace=True)
df.set_index('Stats', inplace=True)

df.plot(kind='barh', figsize=(28, 14))

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 73 Random Forest (3/6)

```
#with resampling
pipe = Pipeline(steps=[('scaler', StandardScaler()),('randomforestclassifier',RandomForestClassifier(random_state=42))])

params = {
    'randomforestclassifier__n_estimators': [200, 500],
    'randomforestclassifier__max_features': ['auto', 'sqrt', 'log2'],
    'randomforestclassifier__max_depth' : [4,5,6,7,8],
    'randomforestclassifier__criterion' :['gini', 'entropy']
}

grid = GridSearchCV(pipe, params,cv= 3,n_jobs=-1, verbose=10,scoring='accuracy')
grid.fit(X_train_sm,y_train_sm)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 74 Random Forest (Oversampling) (4/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 75 Random Forest (Oversampling) (5/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

d = {'Stats':X_train_sm.columns,'FI':grid.best_estimator_.named_steps["randomforestclassifier"].feature_importances_}
df = pd.DataFrame(d)
df
df.sort_values(by=['FI'], ascending=True, inplace=True)
df.set_index('Stats', inplace=True)

df.plot(kind='barh', figsize=(28, 14))

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 76 Random Forest (Oversampling) (6/6)

```
#without resampling
clf = xgb.XGBClassifier(random_state=42)

pipeline = Pipeline([('scaler', StandardScaler()),('clf', clf)])

param_grid = {
    'clf__max_depth': [2, 3, 5, 7, 10],
    'clf__n_estimators': [10, 100, 500],
}

grid = GridSearchCV(pipeline, param_grid, cv=3,n_jobs=-1 ,verbose=10,scoring='accuracy')
grid.fit(X_train,y_train)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 77 XGBoost (1/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 78 XGBoost (2/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

d = {'Stats':X_train.columns,'FI':grid.best_estimator_.named_steps["clf"].feature_importances_}
df = pd.DataFrame(d)
df
df.sort_values(by=['FI'], ascending=True, inplace=True)
df.set_index('Stats', inplace=True)

df.plot(kind='barh', figsize=(28, 14))

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 79 XGBoost (3/6)

```
#with resampling
clf = xgb.XGBClassifier(random_state=42)

pipeline = Pipeline([('scaler', StandardScaler()),('clf', clf)])

param_grid = {
    'clf__max_depth': [2, 3, 5, 7, 10],
    'clf__n_estimators': [10, 100, 500],
}

grid = GridSearchCV(pipeline, param_grid, cv=3,n_jobs=-1 ,verbose=10,scoring='accuracy')
grid.fit(X_train_sm,y_train_sm)

#best parameters & classification report
#print the model parameters of the best model
print(grid.best_params_)
print("-"*80)

# print the average of all cv folds for a single combination of the parameters
print ('Best score: %0.3f' % grid.best_score_)
print("-"*80)

#Predict and Evaluate of the best model
best=grid.best_estimator_
best_preds=best.predict(X_test)
print(classification_report(y_test, best_preds))
print("-"*80)
```

Παράρτημα 80 XGBoost (Oversampling) (4/6)

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test, best_preds)
print('Accuracy: %0.2f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test, best_preds)
print('Precision: %0.2f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, best_preds)
print('Recall: %0.2f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, best_preds)
print('F1 score: %0.2f' % f1)
print("-"*80)

#confusion matrix
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                confusion_matrix(y_test, best_preds).flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     confusion_matrix(y_test, best_preds).flatten()/np.sum(confusion_matrix(y_test, best_preds))]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
print("-"*80)
sns.heatmap(confusion_matrix(y_test, best_preds), annot=labels, fmt='', cmap='Blues')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
print("-"*80)
```

Παράρτημα 81 XGBoost(Oversampling) (5/6)

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
print('Roc_Auc: %0.2f' % roc_auc)

features = pd.DataFrame()
features['feature'] = X_train_sm.columns
features['importance'] = grid.best_estimator_.named_steps["clf"].feature_importances_
#features = features[features['feature']!='rating_count']
features.sort_values(by=['importance'], ascending=True, inplace=True)
features.set_index('feature', inplace=True)

features.plot(kind='barh', figsize=(28, 14))

plt.figure(figsize=(10,6))
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
#plt.xlim([0, 1])
#plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Παράρτημα 82 XGBoost (Oversampling) (6/6)