ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Μ/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ
ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΥΤΟΜΑΤΗ ΕΥΡΕΣΗ ΑΠΑΤΗΣ ΣΤΟ ΤΡΑΠΕΖΙΚΟ ΤΟΜΕΑ ΣΑΝ ΠΡΟΒΛΗΜΑ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗΣ

Μαρία Αλεξοπούλου

03202904

Επιβλέπων Καθηγητής:
Γεώργιος Ματσόπουλος

Αθήνα, Ιούλιος 2020

Περίληψη

Χάρη στην αύξηση και την ταχύτατη ανάπτυξη του ηλεκτρονικού εμπορίου, η χρήση πιστωτικών καρτών για ηλεκτρονικές αγορές έχει αυξηθεί δραματικά γεγονός το οποίο έχει προκαλέσει έκρηξη στον τομέα της απάτης πιστωτικών καρτών. Καθώς η πιστωτική κάρτα γίνεται ολοένα και πιο δημοφιλής ως τρόπος πληρωμής, τόσο για ηλεκτρονικές όσο και για τακτικές πληρωμές, οι υποθέσεις απάτης που συνδέονται με αυτήν είναι επίσης αυξανόμενες. Στην πραγματικότητα, παράνομες κινήσεις βρίσκονται διεσπαρμένες ανάμεσα σε νόμιμες και έτσι απλές τεχνικές ταιριάσματος δεν είναι συχνά αρκετές ώστε να εντοπίσουν αυτές τις παράνομες κινήσεις με ακρίβεια. Ως εκ τούτου, η εφαρμογή αποτελεσματικών συστημάτων εύρεσης απάτης καθίσταται επιτακτική για όλες τις Τράπεζες οι οποίες έχουν δυνατότητα έκδοσης πιστωτικής κάρτας, ώστε να ελαχιστοποιήσουν τις απώλειές τους.

Πολλές μοντέρνες τεχνικής βασιζόμενες στη Τεχνητή Νοημοσύνη, Εξόρυξη Δεδομένων, Ασαφής Λογική, Μηχανική Εκμάθηση, γενετική Προγραμματιστική έχουν εξελιχθεί στην αυτόματης ανίχνευσης παράνομων κινήσεων. Ωστόσο, στην παρούσα διπλωματική εργασία παρουσιάζεται μία επισκόπηση διάφορων Τεχνικών Ταξινόμησης Μηχανικής Εκμάθησης, οι οποίες χρησιμοποιήθηκαν σε μία προσπάθεια επίτευξης του παραπάνω στόχου.

Τα πειράματα καθώς και κάποιες διαδικασίες προ-επεξεργασίας δεδομένων πραγματοποιήθηκαν με τη χρήση της Python.

Τα αποτελέσματα των πειραμάτων έδειξαν την ακρίβεια πρόβλεψης που είχαν οι διάφοροι ταξινομητές που χρησιμοποιήθηκαν.

Λέξεις Κλειδιά: Ηλεκτρονικό Εμπόριο, Απάτη Πιστωτικής Κάρτας, Τεχνητή Νοημοσύνη, Τεχνητά Νευρωνικά Δίκτυα, Μηχανική Εκμάθηση

Abstract

Due to the rise and rapid growth of E-Commerce, use of credit cards for online purchases has dramatically increased and it caused an explosion in the credit card fraud. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising. In real life, fraudulent transactions are scattered with genuine transactions and simple pattern matching techniques are not often sufficient to detect those frauds accurately. Implementation of efficient fraud detection systems has thus become imperative for all credit card issuing banks to minimize their losses.

Many modern techniques based on Artificial Intelligence, Data mining, Fuzzy logic, Machine learning, Genetic Programming, has evolved in detecting various credit card fraudulent transactions. However, this paper presents a survey of various classification methods used in credit card fraud detection.

All the experiments and some dataset preprocessing routines were conducted with the use of Python language.

The results showed the accuracy level of each classifier which we used.

Keywords: Electronic Commerce, Credit card fraud, Artificial Intelligence, Artificial Neural Networks, Machine Learning

Πίνακας περιεχομένων

Εισαγωγή

Είναι γεγονός πως εν έτη 2020 προωθείται ολοένα και περισσότερο η χρήση πλαστικού χρήματος για οποιαδήποτε συναλλαγή. Η συμμετοχή του κοινού σε Online κυρίως δραστηριότητες έχει οδηγήσει σε αύξηση εγκληματικότητας και ως αποτέλεσμα τεράστιες απώλειες τόσο σε προσωπικό, όσο και σε επιχειρησιακό επίπεδο.

Ενώ τεχνικές επεξεργασίας δεδομένων χρησιμοποιούνται ευρέως ώστε να ελέγξουν χαρακτηριστικά ύποπτων και μη κινήσεων, οι μέθοδοι μηχανικής εκμάθησης στοχεύουν στην πρόβλεψη για το αν μία κίνηση είναι ύποπτη ή όχι, μέσω των λεγόμενων ταξινομητών (classifiers).

Στην παρούσα διπλωματική εργασία περιγράφεται αναλυτικά το πρόβλημα της Απάτης Πιστωτικής Κάρτας καθώς και η τρέχουσα κατάσταση της Βιομηχανίας αναφορικά με το εν λόγω ζήτημα. Στα πρώτα κεφάλαια γίνεται επίσης λόγος για τις τεχνικές απάτης αλλά και το αντίκτυπο που αυτή έχει σε διάφορους τομείς, τονίζοντας έτσι το πόσο σημαντική και αναγκαία καθίσταται η αντιμετώπισης της.

Στη συνέχεια παρουσιάζονται μερικοί από τους τρόπους παρεμπόδισής της απάτης, δίνοντας έμφαση στις Τεχνικές Μηχανικής Εκμάθησης και στους λόγους που αυτές επιλέγονται. Δεδομένου ότι η εύρεση απάτης αποτελεί πρόβλημα δυαδικής ταξινόμησης, περιγράφονται αναλυτικά οι ταξινομητές που χρησιμοποιήθηκαν για την προσπάθεια επίλυσης του προβλήματος.

Στη συνέχεια, στο πειραματικό μέρος της εργασίας αυτής, γίνεται προσπάθεια για ανάπτυξη ενός συστήματος το οποίο ταξινομεί συναλλαγές σε συναλλαγές από τον νόμιμο κάτοχο της κάρτας και συναλλαγές από άλλους (παράνομες). Αρχικά πραγματοποιείται η προ επεξεργασία των δεδομένων με σκοπό τη χρήση τους από τους ταξινομητές. Παραθέτονται αναλυτικά όλα τα βήματα καθώς και ο κώδικας που χρησιμοποιήθηκε σε Python με σκοπό την εκπαίδευση των ταξινομητών Λογιστικής Παλινδρόμησης, Δέντρου Απόφασης, Μηχανής Διανυσμάτων Υποστήριξης και Κ Κοντινότερου Γείτονα. Για τους ταξινομητές αυτούς έγινε σύγκριση των αποτελεσμάτων τους για την ταξινόμηση του test set μέσω ROC Analysis. Έγινε επίσης και μία δοκιμή με χρήση ενός Τεχνητού Νευρωνικού Δικτύου.

Θεωρητικό Μέρος

Απάτη Πιστωτικής Κάρτας

Τη σημερινή εποχή, η Απάτη Πιστωτικής Κάρτας (Credit Card Fraud) αποτελεί μία από τις μεγαλύτερες απειλές των επιχειρηματικών ιδρυμάτων. Ωστόσο, προκειμένου αυτή να αντιμετωπιστεί αποτελεσματικά είναι σημαντικό να κατανοήσουμε πρώτα τους μηχανισμούς εκτέλεσης απάτης.

Οι απατεώνες πιστωτικής κάρτας χρησιμοποιούν ένα μεγάλο αριθμό τρόπων δράσης για να διαπράξουν απάτη. Με απλά λόγια, η Απάτη Πιστωτικής Κάρτας ορίζεται ως: η χρήση της πιστωτικής κάρτας ενός άλλου ατόμου για προσωπικούς λόγους, την ώρα που ο κάτοχος ή εκδότης της κάρτας δεν έχουν επίγνωση του γεγονότος ότι η κάρτα τους χρησιμοποιείται από μη εξουσιοδοτημένο χρήστη. Επιπλέον, το άτομο το οποίο χρησιμοποιεί την κάρτα δεν έχει καμία σχέση με τον κάτοχο ή εκδότη αυτής, και δεν έχει καμία πρόθεση ούτε να έρθει σε επαφή με τον κάτοχο, ούτε και να κάνει αποπληρωμή των αγορών που έχει πραγματοποιήσει.

Οι απάτες πιστωτικής κάρτας διαπράττονται συνήθως με έναν από τους παρακάτω τρόπους:

- Δράση εγκληματικού δόλου (παραπλάνηση εκ προθέσεως) με τη χρήση μη εξουσιοδοτημένου λογαριασμού και/ή προσωπικών πληροφοριών
- Παράνομη ή μη εξουσιοδοτημένη χρήση λογαριασμού για προσωπικό κέρδος
- Παραποίηση πληροφοριών του λογαριασμού για απόκτηση αγαθών και/ή υπηρεσιών

Σε αντίθεση με την κοινή γνώμη, οι έμποροι βρίσκονται σε πολύ μεγαλύτερο κίνδυνο απάτης πιστωτικής κάρτας από ότι οι κάτοχοι των ίδιων των καρτών. Τη στιγμή που οι καταναλωτές μπορεί να αντιμετωπίσουν πρόβλημα προσπαθώντας να αντιστρέψουν μία δόλια χρέωση, οι έμποροι χάνουν το κόστος του προϊόντος που πωλήθηκε, πληρώνουν φόρους επιστροφής χρημάτων και έχουν το φόβο και τον κίνδυνο ο εμπορικός τους λογαριασμός να κλείσει.

Ολοένα και περισσότερο το σενάριο απουσίας κάρτας (card not present scenario), όπως οι αγορές στο διαδίκτυο αποτελούν μεγαλύτερο κίνδυνο καθώς ο έμπορος (το web site) δεν είναι πλέον προστατευμένο με πλεονεκτήματα φυσικής επαλήθευσης στοιχείων όπως η υπογραφή και η ταυτοποίηση μέσω φωτογραφίας. Στην πραγματικότητα, είναι σχεδόν αδύνατο να πραγματοποιηθεί κάποιος από τους απαραίτητους ελέγχους του «φυσικού κόσμου», ώστε να ανιχνευθεί ποιος βρίσκεται πραγματικά πίσω από μία συναλλαγή. Το παραπάνω καθιστά το ίντερνετ άκρως ελκυστικό στους απατεώνες. Σύμφωνα με πρόσφατη έρευνα, η συχνότητα με την οποία εμφανίζεται η ηλεκτρονική απάτη, είναι 12 με 15 φορές μεγαλύτερη από την απάτη στον «φυσικό κόσμο». Ωστόσο, πρόσφατες τεχνολογικές εφαρμογές υπόσχονται έλεγχο της απάτης στα σενάρια απουσίας κάρτας (card not present scenario). [1]

Τρέχουσα Κατάσταση της Βιομηχανίας

Ενώ το ακριβές ποσό της ζημίας λόγω εγκληματικών ενεργειών σε κάρτες είναι άγνωστο, διάφορες αναφορές ερευνητών συμφωνούν πως το ποσό για το έτος 2019 πιθανόν να υπερβαίνει τα 3 δισεκατομμύρια. Επιπλέον, καθώς η χρήση ηλεκτρονικού εμπορίου συνεχίζει να αυξάνεται και οι απατεώνες υιοθετούν όλο και πολυπλοκότερες μεθόδους εξαπάτησης, το προβλεπόμενο ποσό ζημίας για τους εμπόρους που δραστηριοποιούνται στο ηλεκτρονικό εμπόριο αναμένεται στα 5 με 15 δισεκατομμύρια το έτος 2022. Αυτό βεβαίως εξαρτάται από το πόσο γρήγορα η τεχνολογία πρόληψης απάτης θα υιοθετηθεί από την

βιομηχανία. Η συχνότητα απάτης πιστωτικής κάρτας που λαμβάνει χώρα στο διαδίκτυο σύμφωνα με τον Garner, είναι 15 φορές μεγαλύτερη από αυτή που συμβαίνει σε συναλλαγές πρόσωπο με πρόσωπο.

Η αυξανόμενη πιθανότητα της απάτης σε συνδυασμό με την πλήρη οικονομική ευθύνη απέναντι στη ζημιά λόγω απάτης, καθιστά την διαχείριση κρίσεων (risk management) μία από τις πιο σημαντικές προκλήσεις για το ηλεκτρονικό εμπόριο παγκοσμίως. [2]

Τεχνικές Απάτης

Ενώ η χαμένη ή κλεμμένη κάρτα (lost or stolen card) αποτελεί τον πιο σύνηθες τύπο απάτης, άλλοι περιλαμβάνουν κλοπή ταυτότητας (identity theft), κλωνοποίηση κάρτας (skimming), πλαστογραφία κάρτας (counterfeit card), υποκλοπή ταχυδρομείου/email (email intercept fraud) και άλλα. Ο παρακάτω πίνακας περιλαμβάνει τους τρόπους δράσης (modus operandi) για την απάτη πιστωτικής κάρτας και τα ποσοστά με τα οποία αυτοί εμφανίζονται:

Method	Percentage
Lost or stolen card	48%
Identity theft	15%
Skimming (or cloning)	14%
Counterfeit card	12%
Mail intercept fraud	6%
Other	5%

Πίνακας 1: Μέθοδοι Απάτης Πιστωτικής Κάρτας και το ποσοστό εμφάνισής τους

Καθώς η τεχνολογία αλλάζει, το ίδιο συμβαίνει και με τους απατεώνες και τον τρόπο με τον οποίο διαπράττουν τις εγκληματικές τους ενέργειες. Οι απάτες μπορούν σε γενικές γραμμές να κατηγοριοποιηθούν σε παραδοσιακές απάτες πιστωτικής κάρτας (traditional card related frauds), εμπορικές απάτες (merchant related frauds) και ηλεκτρονικές (internet frauds). Οι τρεις εν λόγω τύποι απατών περιγράφονται αναλυτικά στη συνέχεια.

Αντίκτυπο Απάτης Πιστωτικής Κάρτας

Δυστυχώς, περιστατικά απάτης πιστωτικών καρτών έχουν δείξει μόνο ανοδική πορεία προς το παρόν. Τέτοιου είδους δραστηριότητα επηρεάζει όλους τόσο τον κάτοχο της κάρτας, όσο τον έμπορο, τον αγοραστή αλλά και τον εκδότη αυτής. Στο κεφάλαιο αυτό περιγράφονται αναλυτικά οι επιπτώσεις που έχει η απάτη σε όλους τους συμμετέχοντες συναλλαγματικής επιχείρησης (transacting business) μέσω πιστωτικών καρτών.

Αντίκτυπο στον κάτοχο (cardholder)

Στο σημείο αυτό αξίζει να σημειωθεί πως οι κάτοχοι των καρτών είναι οι λιγότερο επηρεασμένοι από την απάτη στις συναλλαγές πιστωτικών καρτών, καθώς η καταναλωτική ευθύνη για τέτοιου είδους συναλλαγές, είναι αρκετά περιορισμένη από την νομοθεσία που επικρατεί στις περισσότερες χώρες. Αυτό ισχύει τόσο σε σενάρια επί παρουσία κάρτας όσο και σε σενάρια απουσίας αυτής (card-present and card not-present scenarios). Πολλές τράπεζες έχουν αρχές οι οποίες περιορίζουν ακόμη περισσότερο την ευθύνη των κατόχων, καθώς επίσης έχουν πολιτικές προστασίας του πελάτη τους οι οποίες καλύπτουν τις περισσότερες απώλειες που μπορεί να προκύψουν σε έναν κάτοχο πιστωτικής κάρτας. Ο κάτοχος δεν έχει παρά μόνο να αναφέρει μία ύποπτη χρέωση στην τράπεζά έκδοσης της κάρτας του, η οποία στη συνέχεια διερευνά το ζήτημα με τον αγοραστή και τον έμπορο και προχωρά σε διαδικασία επιστροφής του ποσού στον πελάτη.

Αντίκτυπο στον έμπορο

Οι έμποροι αποτελούν την περισσότερο επηρεασμένη ομάδα στα πλαίσια την απάτης πιστωτικής κάρτας, ειδικότερα στις συναλλαγές με απουσία κάρτας καθώς θα πρέπει να επωμιστούν όλη την ζημία λόγω απάτης. Κάθε φορά που ένας νόμιμος κάτοχος κάρτας θέτει υπό αμφισβήτηση μία χρέωση στην κάρτα του, η τράπεζα έκδοσης αυτής θα στείλει μία επιστροφή προς τον μέτοχο (μέσω του αγοραστή), αντιστρέφοντας το ποσό την κίνησης. Στην περίπτωση που ο έμπορος δεν έχει διαθέσιμη κάποια φυσική απόδειξη (υπογραφή παραλαβής) ώστε να μπορέσει να αντικρούσει τον κάτοχο της κάρτας, τότε είναι αδύνατο να αντιστρέψει το ποσό κίνησης (chargeback). Έτσι, ο έμπορος θα πρέπει να απορροφήσει το κόστος της συναλλαγής, το οποίο περιλαμβάνει το κόστος των πωλημένων αγαθών, το κόστος μεταφοράς, τους φόρους που σχετίζονται με την κάρτα, τους φόρους της Τράπεζάς του, το διοικητικό κόστος που συνοδεύει μία chargeback συναλλαγή και τέλος την απώλεια της καλής του φήμης.

Αντίκτυπο στις Τράπεζες

Βασιζόμενοι στους κανόνες που διέπουν τόσο την MasterCard όσο και την Visa, είναι πολλές φορές πιθανό ο εκδότης/αγοραστής να επωμιστεί το κόστος της απάτης. Ακόμα και σε περιπτώσεις όπου ο εκδότης/αγοραστής δεν επωμιστεί το άμεσο κόστος της απάτης, υπάρχουν κάποια έμμεσα κόστη τα οποία εν τέλει θα υποστεί εκείνος. Στην περίπτωση chargeback που έχει εκδοθεί από τον έμπορο, υπάρχουν διοικητικά κόστη τα οποία η τράπεζα πρέπει να επιβάλλει. Οι εκδότες και αγοραστές πρέπει επίσης να κάνουν τεράστιες επενδύσεις προκειμένου να αποτρέψουν τις απάτες επιβάλλοντας συστήματα ανίχνευσης απάτης σε κινήσεις καρτών. [3]

Τεχνολογίες Παρεμπόδισης Απάτης

Καθώς οι απατεώνες χρησιμοποιούν σύνθετες και εξελιγμένες μεθόδους για να αποκτήσουν πρόσβαση σε πληροφορίες πιστωτικών καρτών και να διαπράξουν απάτη, νέες τεχνολογίες είναι διαθέσιμες με σκοπό να βοηθήσουν τους εμπόρους να εντοπίσουν και να αποτρέψουν δόλιες κινήσεις. Τεχνολογίες εύρεσης απάτης επιτρέπουν στους εμπόρους και στις τράπεζες να εκτελέσουν αυτόματες και σύνθετες απεικονίσεις εισερχόμενων κινήσεων και να τις σημειώσουν (flagging) σαν ύποπτες.

Παρακάτω αναφέρονται κάποιες από τις τεχνικές πρόληψης απάτης, αν και όπως αναφέρθηκε στην εισαγωγή, στην παρούσα διπλωματική εργασία εξετάζεται η Μηχανική Μάθηση (Machine Learning):

1. **Χειροκίνητος έλεγχος (Manual Review)**: Περιλαμβάνει έλεγχο όλων των κινήσεων για τυχόν σημάδια ύποπτης δραστηριότητας και η ανθρώπινη παρέμβαση καθίσταται απαραίτητη σε πολύ μεγάλο βαθμό. Το γεγονός αυτό καθιστά τη μέθοδο πολύ ακριβή και χρονοβόρα.
2. **Σύστημα επιβεβαίωσης διευθύνσεως (Address Verification System)**: Τεχνική η οποία εφαρμόζεται σε σενάρια απουσίας κάρτας και βασίζεται στο ταίριασμα των πρώτων ψηφίων της διεύθυνσης και του ταχυδρομικού κώδικα τα οποία έχουν δοθεί για την χρέωση (delivery), με αυτά που έχει καταχωρημένα η Τράπεζα για τον κάτοχο της κάρτας. Ο κώδικας που αναπαριστά το επίπεδο ταιριάσματος αποστέλλεται στον έμπορο.
3. **Θετικές και αρνητικές λίστες**: Μία αρνητική λίστα αποτελεί μία βάση δεδομένων που χρησιμοποιείται για να ταυτοποίηση υψηλού κινδύνου κινήσεις βασισμένες σε συγκεκριμένους τομείς δεδομένων. Παράδειγμα αρνητικής λίστας θα μπορούσε να είναι ένα αρχείο που περιέχει όλους τους αριθμούς καρτών που έχουν δημιουργήσει

chargebacks στο παρελθόν, ώστε να αποφευχθούν απάτες από επαναλαμβανόμενους απατεώνες. Αντίθετα, θετικές λίστες χρησιμοποιούνται ώστε να αναγνωρίζονται αξιόπιστοι πελάτες, πιθανόν μέσα από τον αριθμό της κάρτας τους ή το email τους. Τα θετικά αρχεία αποτρέπουν περιττές καθυστερήσεις στην επεξεργασία έγκυρων παραγγελιών.

4. **Επαλήθευση Πληρωτή (Payer Authentication):** Αναπτυσσόμενη τεχνολογία η οποία υπόσχεται ένα νέο επίπεδο ασφάλειας στο "επιχείρηση προς καταναλωτή" (business-to-consumer) ηλεκτρονικό εμπόριο. Η πρώτη εφαρμογή αυτού του τύπου υπηρεσίας είναι τα προγράμματα Verified by Visa (VbV) και Visa Payer Authentication Service (VPAS) τα οποία παρουσιάστηκαν παγκοσμίως το 2002. Τα προγράμματα βασίζονται στο Προσωπικό Αναγνωριστικό Νούμερο (PIN) που συνδέεται με την κάρτα, όμοια με αυτά που χρησιμοποιούνται με τις κάρτες των ATM, και ένα ασφαλές άμεσο κανάλι επαλήθευσης μεταξύ καταναλωτή και τράπεζας έκδοσης. Το PIN εκδίδεται από την τράπεζα όταν ο κάτοχος της κάρτας συνδέει την κάρτα του με το πρόγραμμα, και χρησιμοποιείται αποκλειστικά για την επαλήθευση ηλεκτρονικών κινήσεων. Όταν οι εγγεγραμμένοι κάτοχοι καρτών φτάνουν στο σημείο του check out της αγοράς τους, τότε η τράπεζα τους ζητά να παρέχουν τον κωδικό τους. Εφόσον ο κωδικός επαληθευτεί, τότε μόνο ο έμπορος μπορεί να ολοκληρώσει την συναλλαγή και να αποστείλει την πληροφορία επαλήθευσης στον αγοραστή. [4]

Γιατί Machine Learning

Ένα Σύστημα Ανίχνευσης Απάτης (Fraud Detection System) δεν αρκεί μόνο να μπορεί να εντοπίζει επιτυχώς περιπτώσεις απάτης, αλλά και να είναι αποτελεσματικό αναφορικά με το κόστος του, δεδομένου ότι το ποσό που επενδύεται για τον έλεγχο των συναλλαγών δεν θα πρέπει να είναι μεγαλύτερο από τη ζημία λόγω απάτης.

Προκειμένου να ελαχιστοποιηθεί το κόστος ανίχνευσης απάτης είναι σημαντικό να χρησιμοποιούνται κανόνες και μοντέλα βασιζόμενα στην στατιστική, ώστε να γίνεται ένας πρώτος διαχωρισμός μεταξύ γνήσιας συναλλαγής και πιθανής απάτης και οι ερευνητές να καλούνται να ερευνήσουν μόνο τις περιπτώσεις υψηλού κινδύνου. Τυπικά, οι συναλλαγές σε πρώτη φάση φιλτράρονται ελέγχοντας κάποιες βασικές προϋποθέσεις (για παράδειγμα επαρκές υπόλοιπο) και μετά κατατάσσονται μέσω ενός προγνωστικού μοντέλου όπως φαίνεται στην Σχήμα 1: Η διαδικασία ανίχνευσης απάτης Πιστωτικής Κάρτας

. Το μοντέλο αυτό κατατάσσει την κάθε κίνηση σε υψηλού ή χαμηλού κινδύνου απάτης και στη συνέχεια οι υψηλού κινδύνου εκπέμπουν ειδοποιήσεις για περαιτέρω έλεγχο. Οι ερευνητές με τη σειρά τους ελέγχουν τις εν λόγω ειδοποιήσεις και παρέχουν σχόλια (feedback) για την κάθε μία. Τα σχόλια αυτά αφορούν το αν η ειδοποίηση είναι αληθώς θετική (fraud) ή ψευδώς θετική (genuine). Τέτοιου είδους feedback μπορεί να χρησιμοποιηθεί ώστε να βελτιωθεί το μοντέλο. Ένα Μοντέλο Πρόβλεψης μπορεί να χτιστεί μέσα από κανόνες των ειδικών, ωστόσο απαιτείται χειροκίνητος συντονισμός και ανθρώπινη επίβλεψη. [5]

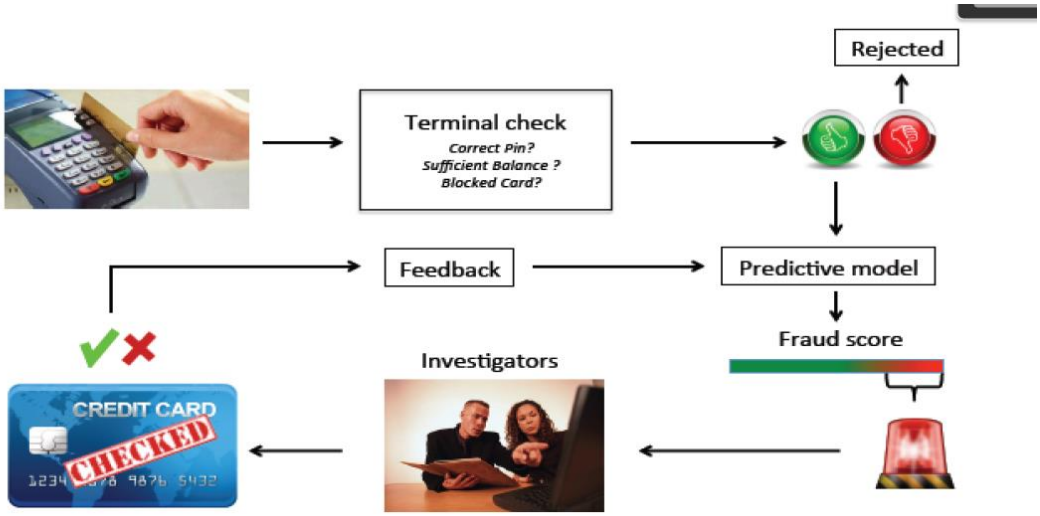
Εναλλακτικά, χρησιμοποιώντας τεχνικές Machine Learning καθίσταται δυνατή η εύρεση τεχνικών και η πρόβλεψη κινήσεων οι οποίες πιθανά να είναι δόλιες. Τέτοιες τεχνικές, απαρτίζονται από μοντέλα τα οποία βασίζονται σε σύνολα παραδειγμάτων. Το μοντέλο είναι στις περισσότερες περιπτώσεις μια παραμετρική λειτουργία, η οποία επιτρέπει την πρόβλεψη της πιθανότητας μία κίνηση να είναι δόλια, βασιζόμενη σε χαρακτηριστικά που περιγράφουν την κάθε κίνηση. [6]

Στον τομέα της Εύρεσης Απάτης, η χρήση Machine Learning είναι ελκυστική για πολλούς λόγους. Πρώτον, επιτρέπουν την ανακάλυψη μοτίβων σε αρκετά υψηλής διάστασης ροή

δεδομένων, κινήσεις δηλαδή οι οποίες καταφθάνουν σαν συνεχόμενη ροή και κάθε κίνηση χαρακτηρίζεται από πολλές μεταβλητές. Δεύτερον, δόλιες κινήσεις συχνά σχετίζονται με τον χώρο και τον χρόνο πραγματοποίησης. Για παράδειγμα, οι απατεώνες τυπικά προσπαθούν να εκτελέσουν απάτες στα ίδια καταστήματα με διαφορετικές κάρτες μέσα σε μικρό χρονικό διάστημα. Τρίτον, τεχνικές Machine Learning μπορούν να χρησιμοποιηθούν για την εύρεση ή μοντελοποίηση ήδη υπαρχόντων στρατηγικών, καθώς επίσης για την αναγνώριση νέων συνδεδεμένων με ασυνήθιστη συμπεριφορά κατόχων πιστωτικής κάρτας. Μοντέλα Πρόβλεψης βασιζόμενα σε τεχνικές Machine Learning είναι επίσης ικανά να αναπτύσσουν αυτόματα feedback ώστε να βελτιώνουν την ακρίβεια της εύρεσης. Κάτι τέτοιο επιτρέπει την μείωση κόστους και χρόνου [1].

Μεθοδολογία

Από πλευράς ανάλυσης δεδομένων, η Εύρεση Απάτης αποτελεί ένα πρόβλημα δυαδικής ταξινόμησης (Binary Classification) κατά το οποίο τα δεδομένα κινήσεων αναλύονται και ταξινομούνται/κατατάσσονται είτε σε νόμιμα ("legitimate"), είτε σε δόλια ("fraudulent"). Η δυαδική ταξινόμηση είναι μία απλή μορφή ταξινόμησης όπου ένα σύνολο δεδομένων κατηγοριοποιείται σε δύο κατηγορίες ανάλογα με κάποια χαρακτηριστικά. Χρησιμοποιείται κυρίως σε περιπτώσεις όπου θέλουμε να προβλέψουμε ένα συγκεκριμένο αποτέλεσμα, το οποίο όμως μπορεί να πάρει μόνο δύο διακριτές τιμές. Μερικά τυπικά παραδείγματα περιλαμβάνουν ιατρικές διαγνώσεις, εύρεση ανεπιθύμητης αλληλογραφίας, ή στην συγκεκριμένη περίπτωση, εύρεση απάτης.



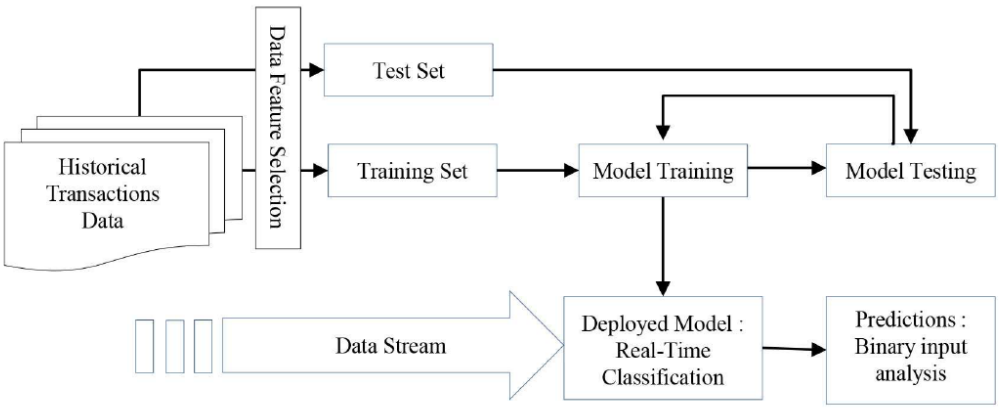
Σχήμα 1: Η διαδικασία ανίχνευσης απάτης Πιστωτικής Κάρτας

Παρόλο που δικαίως η μέθοδος δυαδικής ταξινόμησης είναι ένα πολύ βασικό πρόβλημα, υπάρχουν πάρα πολλά παραδείγματα που χρησιμοποιούνται για την εκμάθηση δυαδικών ταξινομητών (learning binary classifiers) όπως: Δέντρα Αποφάσεων (Decision Trees), Νευρωνικά Συστήματα (Neural Networks), ταξινόμηση Bayes (Bayesian Classification), Λογιστική Παλινδρόμηση (Logistic regression), K-nearest neighborhood και άλλα [7].

Στην παρούσα διπλωματική εργασία δοκιμάστηκαν κάποιες από τις παραπάνω τεχνικές και στο τέλος έγινε σύγκριση μεταξύ των επιδόσεων της κάθε μίας. Το ακόλουθο κεφάλαιο περιγράφει τις τεχνικές αυτές περιληπτικά, καθώς επίσης τα δεδομένα (dataset) και τις μετρήσεις που χρησιμοποιήθηκαν για τον προσδιορισμό της απόδοσης.

Τεχνικές Ταξινόμησης

Η αρχιτεκτονική μίας τεχνικής ταξινόμησης όπως είναι στην πραγματικότητα, με τη χρήση δεδομένων πραγματικού χρόνου, φαίνεται στην παρακάτω απεικόνιση:



Σχήμα 2: Αρχιτεκτονική ενός μοντέλου ταξινόμησης πραγματικού χρόνου.

Η βασική αρχιτεκτονική είναι παρόμοια για κάθε μέθοδο. Γίνεται αρχικά διαχωρισμός των δεδομένων εισόδου (από το παρελθόν) σε 2 ομάδες. Η πρώτη και μεγαλύτερη ομάδα είναι τα δεδομένα εκπαίδευσης που θα χρησιμοποιηθούν για την δημιουργία του μοντέλου. Η δεύτερη και μικρότερη ομάδα είναι τα δεδομένα ελέγχου τα οποία χρησιμοποιούνται για έλεγχο της απόδοσης ταξινόμησης από το μοντέλο. Είναι σημαντικό να αναφερθεί ότι ποτέ δεν πρέπει τα δεδομένα του ελέγχου να χρησιμοποιηθούν με οποιοδήποτε τρόπο για την εκπαίδευση του μοντέλου.

Decision Trees

Υπάρχουν δύο κατηγορίες δέντρων, τα δέντρα οπισθοδρόμησης και τα δέντρα ταξινόμησης. Στην περίπτωσή της εργασίας αυτής, ο αλγόριθμος του δέντρου απόφασης είναι κατασκευασμένος με βάση ένα σύνολο δεδομένων εκπαίδευσης. [8]

Ένα δέντρο απόφασης αποτελείται από κόμβους αποφάσεων οι οποίοι σχηματίζουν μία δομή δέντρου, στο οποίο ο ύψιστος κόμβος καλείται ρίζα. Κάθε κόμβος χωρίς φύλλα δηλώνει τεστ σε ένα χαρακτηριστικό, κάθε κλαδί αναπαριστά το αποτέλεσμα ενός τεστ, και κάθε φύλλο κρατά την ετικέτα μιας τάξης. Οι κόμβοι φύλλων αναπαριστούν τάξεις οι οποίες επιστρέφονται σαν τελική πρόβλεψη του μοντέλου.

Ο αλγόριθμος χωρίζει επαναλαμβανόμενα το σετ δεδομένων σύμφωνα με ένα κριτήριο το οποίο μεγιστοποιεί το διαχωρισμό τους, έχοντας σαν αποτέλεσμα μία δομή δέντρου. Το πιο κοινό κριτήριο που εφαρμόζεται είναι η απόκτηση πληροφορίας (Information gain). Αυτό σημαίνει πως σε κάθε διαχωρισμό, η μείωση εντροπίας λόγω αυτού μεγιστοποιείται. Η πρόβλεψη P(y/x) αποτελεί την αναλογία των y στοιχείων προς όλα τα στοιχεία του κόμβου ο οποίος περιλαμβάνει δεδομένα x. [9]

Logistic Regression

Η λογιστική παλινδρόμηση (Logistic regression) αποτελεί στην ουσία ένα μοντέλο ταξινόμησης των τιμών μιας μεταβλητής απόκρισης Y με βάση τη θεωρία των πιθανοτήτων. Στο μοντέλο αυτό, όπου η μεταβλητή Y συνήθως έχει δυαδικό χαρακτήρα (λαμβάνει δύο τιμές), στόχος είναι η πρόβλεψη της έκβασης αυτής από ένα πλήθος προβλεπτικών μεταβλητών που μπορεί να είναι ονομαστικές, τακτικές ή ποσοτικές. Κατά τη λογιστική παλινδρόμηση η εκτίμηση των παραμέτρων γίνεται με τη μέθοδο του λόγου πιθανοφάνειας

(μέθοδος συνήθως εφαρμοζόμενη στα γενικευμένα γραμμικά υποδείγματα), δηλαδή επιλέγονται οι πιο πιθανοφανείς τιμές των παραμέτρων, προκειμένου να οδηγήσουν στα παρατηρούμενα αποτελέσματα. Ως επακόλουθο, αναπτύσσεται πάντα ετεροσκεδαστικότητα σε κάθε προβλεπόμενη τιμή εξαιτίας του μεταβαλλόμενου ποσοστού διακύμανσης που αναλογεί σε αυτήν. Η λογιστική παλινδρόμηση επινοήθηκε ως εναλλακτική επιλογή της γραμμικής διακριτικής ανάλυσης για την ταξινόμηση των στοιχείων (ονομαστικών ή τακτικών) της εξαρτημένης, με ευρεία απήχηση σε πολλά διαφορετικά επιστημονικά πεδία και κυρίως στην ιατρική και τις κοινωνικές επιστήμες. [10]

Αλγόριθμος Naïve Bayes

Ο αλγόριθμος Naïve Bayes βασίζεται σε δύο υποθέσεις. Πρώτον, όλα τα χαρακτηριστικά τα οποία χρειάζεται να κατηγοριοποιηθούν, συνεισφέρουν εξίσου στην απόφασή (εξίσου σημαντικά). Έπειτα, όλα τα χαρακτηριστικά είναι στατιστικά ανεξάρτητα, το οποίο σημαίνει πως γνωρίζοντας την αξία ενός χαρακτηριστικού δεν παρέχονται περαιτέρω πληροφορίες για τις αξίες των υπόλοιπων χαρακτηριστικών (κάτι που δεν ισχύει απαραίτητα στην πράξη). Η διαδικασία κατηγοριοποίησης μιας παρατήρησης πραγματοποιείται εφαρμόζοντας τον κανόνα του Bayes για κάθε κλάση που έχει δοθεί στην παρατήρηση. Σε ότι αφορά την εύρεση απάτης, ο αλγόριθμος εφαρμόζεται για κάθε μία από τις δύο κλάσεις (δόλια και νόμιμη), και εκείνη η οποία σχετίζεται με την υψηλότερη πιθανότητα είναι και η προβλεπόμενη κλάση της πράξης.

Αλγόριθμος K-Nearest Neighbors

Ο αλγόριθμος K-Nearest Neighbors (KNN) είναι ένας απλός αλγόριθμος ο οποίος συνδυάζει όλες τις εκπαιδευόμενες, χωρίς σήμανση, παρατηρήσεις και τις κατηγοριοποιεί με βάση τους πιο κοντινούς τους γείτονες. Οι ίδιες οι παρατηρήσεις χρησιμοποιούνται για να αναπαραστήσουν το μοντέλο, σε αντίθεση με τους Αλγορίθμους Δέντρων Απόφασης οι οποίοι χρησιμοποιούν τις παρατηρήσεις ώστε να αναπτύξουν ένα δέντρο το οποίο είναι και αυτό που αναπαριστά το μοντέλο. Ωστόσο, θεωρείται πως όλοι οι αλγόριθμοι εκμάθησης βασίζονται σε παρατηρήσεις (instance based) από τη στιγμή που όλες χρησιμοποιούν παρατηρήσεις του εκπαιδευόμενου σετ δεδομένων (training set) για να κατασκευάσουν τα μοντέλα. Στην τεχνική KNN, μια μη χαρακτηρισμένη παρατήρηση του training set κατηγοριοποιείται υπολογίζοντας τις αποστάσεις μεταξύ της εξεταζόμενης παρατήρησης και των υπόλοιπων που το περιβάλλουν. Το παραπάνω βασίζεται σε μία καθορισμένη μέθοδο μέτρησης απόστασης, και η κλάση που επικρατεί αναθέτεται στη μη χαρακτηρισμένη παρατήρηση. [11]

Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines – SVM)

Η μέθοδος SVM παρουσιάστηκε για πρώτη φορά από τον Vapnik το 1992 με σκοπό να επιλύσει προβλήματα δυαδικής κατηγοριοποίησης και στη συνέχεια επεκτάθηκε σε προβλήματα μη γραμμικής παλινδρόμησης. Σε αντίθεσή με την ANN η οποία βασίζεται στην εμπειρική ελαχιστοποίηση του ρίσκου, η SVM βασίζεται στην δομική. Πιο συγκεκριμένα η SVM χαρτογραφεί τα δεδομένα σε έναν προκαθορισμένο, υψηλής διάστασης χώρο μέσω μιας λειτουργίας μετασχηματισμού και βρίσκει το πλάνο που μεγιστοποιεί το κενό μεταξύ των δύο τάξεων. Η λύση βασίζεται μόνο σε αυτά τα σημεία δεδομένων, τα οποία βρίσκονται στο κενό. Τα σημεία αυτά ονομάζονται διανύσματα υποστήριξης. [12]

Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANN)

Η ANN είναι ένα μαθηματικό μοντέλο πρόβλεψης της απόδοσης των συστημάτων (για παράδειγμα αποτέλεσμα ενός συστήματος) το οποίο είναι εμπνευσμένο από την δομή και

λειτουργικότητα των ανθρωπίνων βιολογικών νευρωνικών δικτύων. Η ANN έχει αναπτυχθεί ώστε να έχει μία λειτουργικότητα παρόμοια με αυτή του ανθρώπινου εγκεφάλου, μέσα από την απομνημόνευση και εκμάθηση ποικίλων εργασιών, και ανάλογης συμπεριφοράς. Εκπαιδεύεται έτσι ώστε να προβλέπει συγκεκριμένες συμπεριφορές και να τις «θυμάται» στο μέλλον όπως ακριβώς θα τις θυμόταν και ο ανθρώπινος εγκέφαλος. Η αρχιτεκτονική της μεθόδου είναι επίσης παρόμοια με αυτή των ανθρώπινων νευρωνικών στρωμάτων στον εγκέφαλο, σε ότι αφορά την λειτουργικότητα και την διασύνδεση του δικτύου. Τέλος, αξίζει να σημειωθεί πως η τεχνική Τεχνητών Νευρωνικών Δικτύων έχει χρησιμοποιηθεί επιτυχώς σε διάφορες εφαρμογές. [13]

Πειραματικό Μέρος

Δεδομένα (Dataset)

Καθότι δεν υπάρχουν δημόσια δεδομένα (συμπληρωμένα με δείκτες) σχετικά με απάτη πιστωτικών καρτών, στην παρούσα διπλωματική εργασία χρησιμοποιήθηκε ως πηγή δεδομένων το dataset από το Project DEFEATFRAUD του Πανεπιστήμιου «UNIVERSITE Libre de Bruxelles». Το συγκεκριμένο dataset περιλαμβάνει κινήσεις πιστωτικών καρτών που έλαβαν χώρα το Σεπτέμβριο του 2013 από Ευρωπαίους κατόχους καρτών.

Πρόκειται για ανώνυμα δεδομένα 31 στηλών (χαρακτηριστικών), τα οποία προέκυψαν ύστερα από επεξεργασία με PCA (PRINCIPAL COMPONENT ANALYSIS) . Εξαίρεση στα παραπάνω αποτελούν οι 2 από τις 31 στήλες και συγκεκριμένα οι στήλες χρόνος και ποσό, οι οποίες και δοθήκαν ως έχουν. Τέλος, υπάρχει και η δυαδική στήλη class, που λαμβάνει την τιμή 0 αν η κίνηση είναι νόμιμη αι 1 αν πρόκειται για απάτη (fraud).

Το εν λόγω dataset αναπαριστά κινήσεις δύο ημερών, μεταξύ των οποίων οι 492 από τις συνολικές 284.807, έχουν χαρακτηριστεί ως fraud (στήλη class = 1). Δεδομένου ότι το dataset είναι αρκετά ανισόρροπο (unbalanced), η θετική κλάση (frauds) δηλαδή αποτελεί το 0.172% του συνολικού πλήθους κινήσεων, ακολουθήθηκε επεξεργασία των δεδομένων ώστε αυτά να έχουν την επιθυμητή μορφή, όπως περιγράφεται παρακάτω.

Προ - επεξεργασία Δεδομένων

Πρώτη Φάση

Κανονικοποίηση (Normalization) δεδομένων και συγκεκριμένα των στηλών χρόνος και ποσό, διότι η μορφή στην οποία βρίσκονται είναι τελείως διαφορετική από τα υπόλοιπα χαρακτηριστικά του dataset. Για την επίτευξη του παραπάνω χρησιμοποιήθηκε η RobustScaler της skLearn.

Στη συνέχεια παρατηρήθηκε πως τα δεδομένα είναι ανισόρροπα ως προς το πλήθος δηλαδή των κινήσεων χαρακτηρισμένες ως fraud, το οποίο και είναι πολύ μικρότερο από τις νόμιμες κινήσεις. Το γεγονός αυτό υποδεικνύει την αναγκαιότητα προ - επεξεργασίας των δεδομένων με τη χρήση αλγορίθμου ο οποίος στην ουσία καλείται να διορθώσει την κατανομή αυτή. Προκειμένου τελικά να γίνει αξιολόγηση της απόδοσης του τελικού αλγορίθμου Μηχανικής Εκμάθησης, πραγματοποιείται αρχικά διαχωρισμός των δεδομένων σε δύο μέρη, εκπαίδευσης (train) και τεστ (test) τα οποία θα πρέπει να έχουν και ίση κατανομή μεταξύ fraud και πραγματικών συναλλαγών.

Αναφορικά με τον **διαχωρισμό (splitting the data)** των δεδομένων, επιλέχθηκε να χωριστεί σε δύο μέρη μεγέθους 80% train και 20% test του αρχικού dataset, χρησιμοποιώντας το StratifiedKFold της scikit - learn. Στις ομάδες (set) που δημιουργήθηκαν, έγινε επίσης έλεγχος ότι έχουν πράγματι ίση κατανομή (μεταξύ fraud και legit).

Μία πρώτη προσέγγιση για την επίλυση του προβλήματος ανισορροπίας των δεδομένων μεταξύ των κλάσεων είναι η αφαίρεση των συναλλαγών (δειγμάτων) μέχρι οι δύο κλάσεις να έχουν το ίδιο πλήθος συναλλαγών. Η πρώτη μέθοδος που χρησιμοποιήθηκε για την **εξισορρόπηση** των δεδομένων καλείται **Random Under Sampling** και βασίζεται στην ανάμιξη των συναλλαγών προτού παρθεί δείγμα των αρχικών δεδομένων, έτσι ώστε να εξαλειφθεί η επίδραση που μπορεί να έχει η σειρά των συναλλαγών στο τελικό αποτέλεσμα. Ωστόσο, η μέθοδος αυτή μπορεί να καταστεί προβληματική λόγω της μεγάλης απώλειας πληροφορίας, καθώς απορρίπτονται από τις 284.315 συναλλαγές, οι 283.823.

Συνεχίζοντας με την επίλυση του προβλήματος των imbalanced δεδομένων, χρησιμοποιήθηκε και η **Over – Sampling τεχνική SMOTE** με βάση την οποία δημιουργούνται

εικονικές συναλλαγές, τα χαρακτηριστικά των οποίων είναι παρόμοια με αυτά των ήδη υπάρχοντών, με σκοπό να ενισχυθεί η κλάση με τις λιγότερες συναλλαγές (στην περίπτωσή μας fraud). Μέσω αυτής, οι δύο κλάσεις γίνονται ίσες σε πλήθος παρατηρήσεων ενώ διατηρούνται και όλες οι αρχικές τιμές.

Δεύτερη Φάση

Στο σημείο αυτό μπορεί να γίνει και ένας παραπάνω έλεγχος του βαθμού **Συσχέτισης (Correlation)** μεταξύ των διάφορων χαρακτηριστικών προκειμένου να εντοπιστούν πιθανές συσχετίσεις μεταξύ τους. 4

Ακολουθεί **Εντοπισμός Ανωμαλιών (Anomaly Detection)** και η αφαίρεση ακραίων τιμών που πιθανό να υπάρχουν στο dataset με σκοπό να μην επηρεαστεί το τελικό μοντέλο. Ο τρόπος με τον οποίο επιτυγχάνεται αυτό είναι μέσω επιλογής ενός εύρους τιμών για κάθε χαρακτηριστικό, και οποιαδήποτε παρατήρηση βρίσκεται εκτός των ορίων του εύρους απορρίπτεται. Αξίζει να σημειωθεί πως η επιλογή ορίων επηρεάζει άμεσα το τελικό αποτέλεσμα, καθώς πολύ αυστηρά όρια θα οδηγήσουν σε απόρριψη πολλαπλών συναλλαγών και άρα μείωση της ακρίβειας του τελικού μοντέλου, ενώ πολύ χαλαρά όρια θα επιτρέψουν την παραμονή συναλλαγών οι οποίες απέχουν πολύ από το μέσο όρο και άρα θα επηρεαστεί και πάλι το μοντέλο.

Εφόσον πλέον τα δεδομένα μας είναι έτοιμα να χρησιμοποιηθούν για την εκπαίδευση του μοντέλου ,ξεκινά η διαδικασία εκπαίδευσης και τελικά αξιολόγησης των ταξινομητών (classifiers) η οποία και περιγράφεται αναλυτικά στην επόμενη ενότητα.

Πειράματα και Αποτελέσματα

Σε πρώτη φάση γίνεται η εισαγωγή των βιβλιοθηκών που πρόκειται να χρησιμοποιηθούν:

```
   import numpy as np # linear algebra
   import pandas as pd # data processing, CSV file I/O (e.g. pd.re
ad_csv)
   import tensorflow as tf
   import matplotlib.pyplot as plt
   import seaborn as sns
   from sklearn.manifold import TSNE
   from sklearn.decomposition import PCA, TruncatedSVD
   import matplotlib.patches as mpatches
   import time

   # Classifier Libraries
   from sklearn.linear_model import LogisticRegression
   from sklearn.svm import SVC
   from sklearn.neighbors import KNeighborsClassifier
   from sklearn.tree import DecisionTreeClassifier
   from sklearn.ensemble import RandomForestClassifier
   import collections


   # Other Libraries
   from sklearn.model_selection import train_test_split
   from sklearn.pipeline import make_pipeline
   from imblearn.pipeline import make_pipeline as imbalanced_make_
pipeline
   from imblearn.over_sampling import SMOTE
   from imblearn.under_sampling import NearMiss
   from imblearn.metrics import classification_report_imbalanced
   from sklearn.metrics import precision_score, recall_score, f1_s
core, roc_auc_score, accuracy_score, classification_report
```

```
from collections import Counter
from sklearn.model_selection import KFold, StratifiedKFold
import warnings
warnings.filterwarnings("ignore")
```

```
colors = ["#0101DF", "#DF0101"]
```

Εισαγωγή δεδομένων:

```
df = pd.read_csv('creditcard.csv')
```

Έλεγχος για το εάν λείπουν τιμές από το dataset:

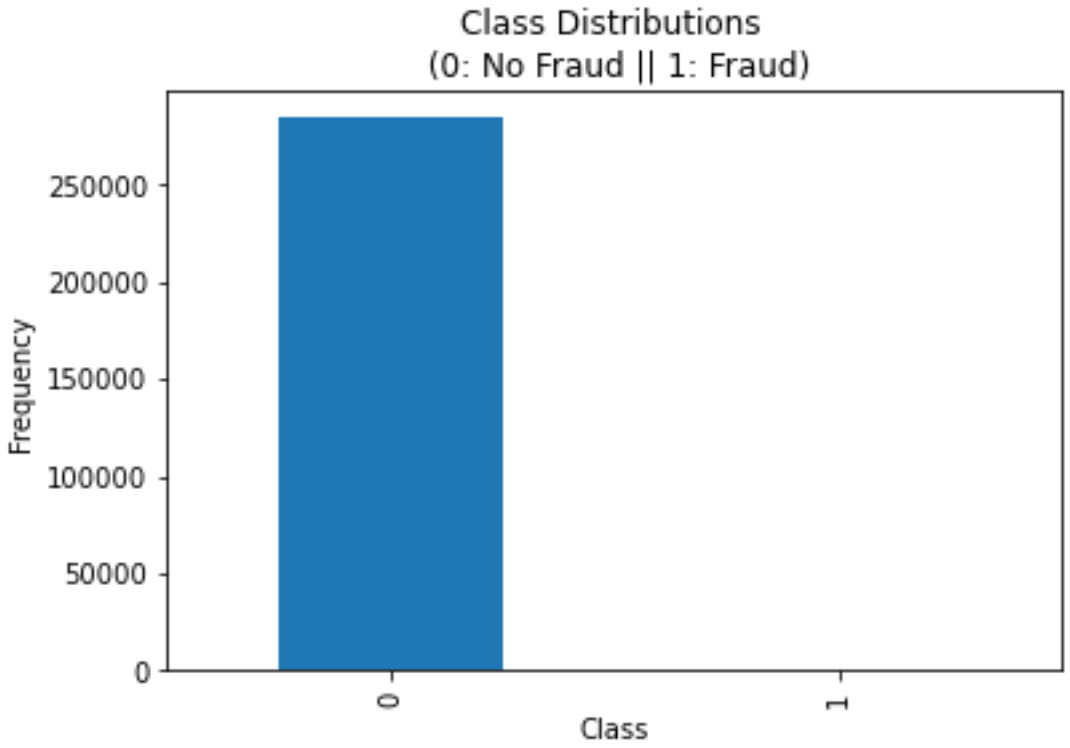
```
df.isnull().values.any()
```

```
False
```

Κατανομή των 2 κλάσεων

```
count_classes = pd.value_counts(df['Class'], sort = True).sort_
index()
count_classes.plot(kind = 'bar')

plt.title('Class Distributions \n (0: No Fraud || 1: Fraud)')
plt.xlabel("Class")
plt.ylabel("Frequency")
```

```
Text(0, 0.5, 'Frequency')
```



Σχήμα 3: Διάγραμμα κατανομής κλάσεων και η συχνότητά τους

Κανονικοποίηση Δεδομένων

Τα δεδομένα έχουν δοθεί κανονικοποιημένα με εξαίρεση τις στήλες "Amount", "Time". Επιλέγεται να χρησιμοποιηθεί το RobustScaler της sklearn.

```
from sklearn.preprocessing import StandardScaler, RobustScaler
```

```
   # RobustScaler

   rob_scaler = RobustScaler()

   df['scaled_amount'] = rob_scaler.fit_transform(df['Amount'].val
ues.reshape(-1,1))
   df['scaled_time'] = rob_scaler.fit_transform(df['Time'].values.
reshape(-1,1))

   df.drop(['Time','Amount'], axis=1, inplace=True)

   scaled_amount = df['scaled_amount']
   scaled_time = df['scaled_time']

   df.drop(['scaled_amount', 'scaled_time'], axis=1, inplace=True)
   df.insert(0, 'scaled_amount', scaled_amount)
   df.insert(1, 'scaled_time', scaled_time)
```

Διαίρεση δεδομένων σε 2 ομάδες με όμοια κατανομή συναλλαγών

Τα αρχικά δεδομένα εισόδου χωρίζονται σε 2 ομάδες με σκοπό η μια ομάδα να χρησιμοποιηθεί για την δημιουργία μοντέλων και η δεύτερη για τον τελικό έλεγχο της απόδοσης τους.

Είναι απαραίτητο να διατηρηθεί η σχετική κατανομή πραγματικών και μη συναλλαγών σε κάθε ομάδα. Χρησιμοποιήθηκε το StratifiedKFold με 5 splits προκειμένου να γίνει ο διαίρεση των αρχικών δεδομένων σε 2 άνισες ομάδες και να διατηρηθεί η σχετική κατανομή συναλλαγών.

```
   from sklearn.model_selection import train_test_split
   from sklearn.model_selection import StratifiedShuffleSplit

   print('Νόμιμες Συναλλαγές:', round(df['Class'].value_counts()[0
]/len(df) * 100,2), '% του dataset')
   print('Παράνομες Συναλλαγές:', round(df['Class'].value_counts()
[1]/len(df) * 100,2), '% του dataset')

   X = df.drop('Class', axis=1)
   y = df['Class']

   sss = StratifiedKFold(n_splits=5, random_state=None, shuffle=Fa
lse)

   for train_index, test_index in sss.split(X, y):
    print("Train:", train_index, "Test:", test_index)
    original_Xtrain, original_Xtest = X.iloc[train_index], X.iloc[
test_index]
    original_ytrain, original_ytest = y.iloc[train_index], y.iloc[
test_index]


   # μετατροπή σε array
   original_Xtrain = original_Xtrain.values
   original_Xtest = original_Xtest.values
   original_ytrain = original_ytrain.values
   original_ytest = original_ytest.values
```

```
   # έλεγχος κατανομής στις 2 ομάδες
   train_unique_label, train_counts_label = np.unique(original_ytr
ain, return_counts=True)
   test_unique_label, test_counts_label = np.unique(original_ytest
, return_counts=True)
   print('-' * 100)

   print('Κατανομή συναλλαγών: \n')
   print(train_counts_label/ len(original_ytrain))
   print(test_counts_label/ len(original_ytest))
```

```
Νόμιμες Συναλλαγές: 99.83 % του dataset
Παράνομες Συναλλαγές: 0.17 % του dataset

------------------------
Κατανομή συναλλαγών:

[0.99827076 0.00172924]
[0.99827952 0.00172048]
```

Random Under-Sampling

Στο σημείο αυτό θα γίνει εφαρμογή του "Random Under Sampling". Σε πρώτη φάση θα μετρηθούν πόσες συναλλαγές ανήκουν σε κάθε κατηγορία. Στη συνέχεια, θα γίνει τυχαία αναδιάταξη των συναλλαγών του πίνακα, και θα κρατηθούν οι πρώτες Χ συναλλαγές από κάθε κατηγορία, ούτως ώστε να είναι ίδιος αριθμός συναλλαγών σε κάθε μία. Επειδή εδώ υπάρχει σημαντική απώλεια πληροφορίας (απομένουν 492 νόμιμες συναλλαγές από 284,315 αρχικά) υπάρχει κίνδυνος να μειωθεί σημαντικά η απόδοση των μοντέλων.

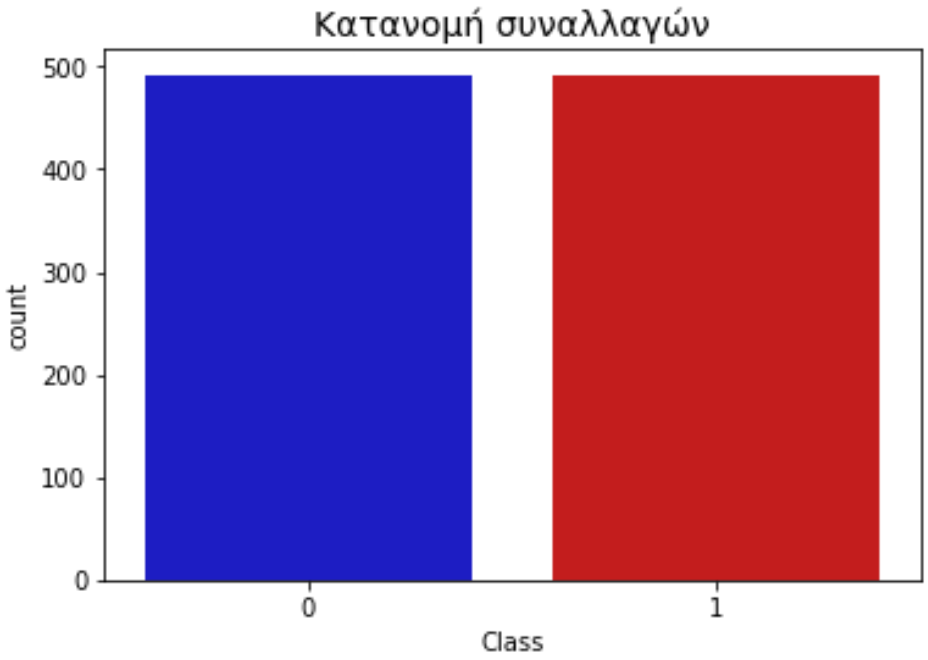
```
   # Ανακατανομή των δεδομένων εισόδου

   df = df.sample(frac=1)

   # διατήρηση ίσων συναλλαγών από κάθε κατηγορία (αρχικά υπήρχαν
492 παράνομες συναλλαγές άρα και εδώ κρατούνται μόνο
   # οι 492 πρώτες συναλλαγές από κάθε κατηγορία)
   fraud_df = df.loc[df['Class'] == 1]
   non_fraud_df = df.loc[df['Class'] == 0][:492]

   normal_distributed_df = pd.concat([fraud_df, non_fraud_df])

   # Ανακατανομή των τελικών δεδομένων
   new_df = normal_distributed_df.sample(frac=1, random_state=42)
```

Έλεγχος

Γίνεται έλεγχος ότι πράγματι έχουν ίση κατανομή τώρα.

```
   print('Κατανομή στο καινούργιο dataset')
   print(new_df['Class'].value_counts()/len(new_df))

   sns.countplot('Class', data=new_df, palette=colors)
   plt.title('Κατανομή συναλλαγών', fontsize=14)
   plt.show()
```

```
Κατανομή στο καινούργιο dataset
1    0.5
0    0.5
Name: Class, dtype: float64
```

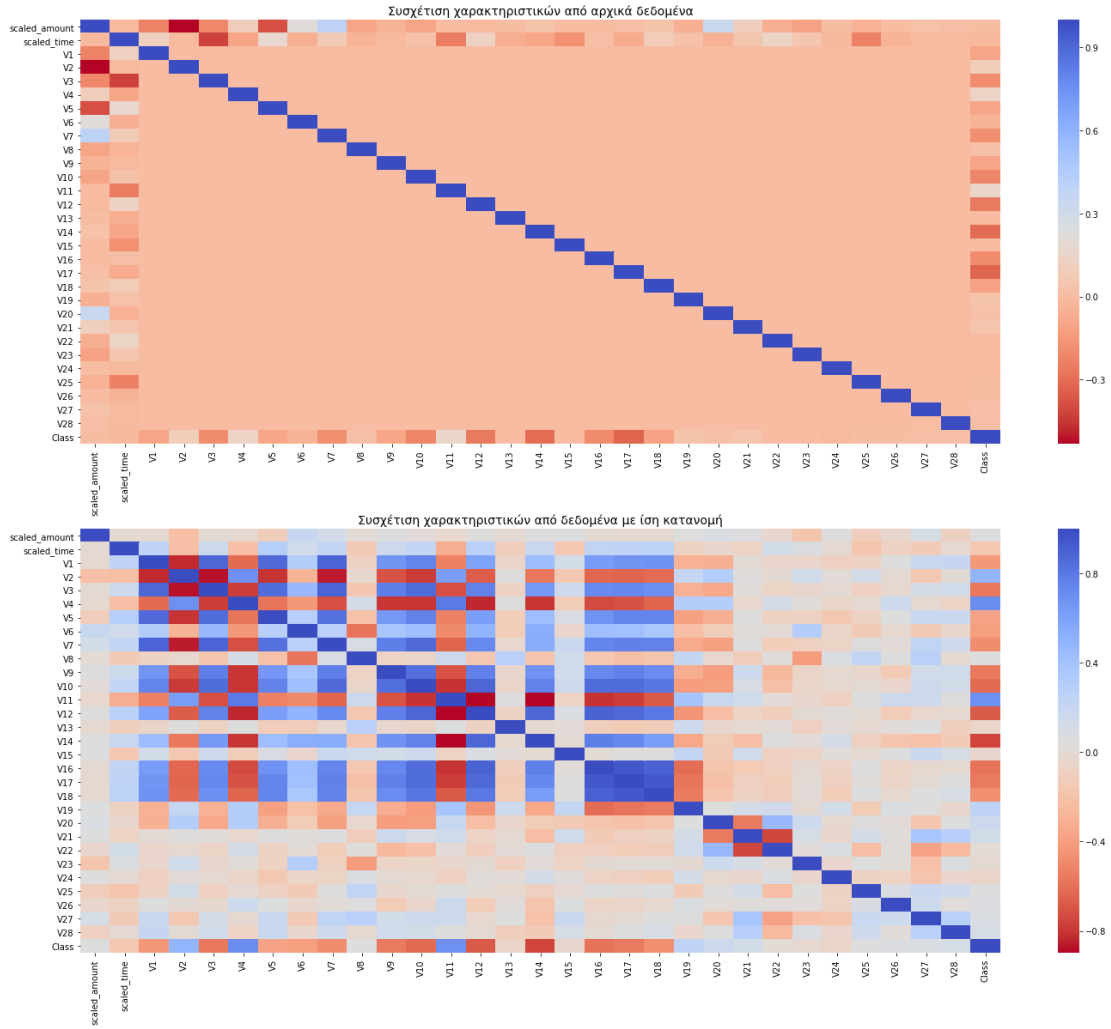


Σχήμα 4: Διάγραμμα κατανομής συναλλαγών έπειτα από την αναδιάταξη

Πίνακες Συσχέτισης

Θα δημιουργηθεί πίνακας συσχέτισης και για τα δύο dataset (το αρχικό και το undersampled). Είναι εμφανές ότι η μεγάλη ανισότητα μεταξύ των 2 κλάσεων επηρεάζει σημαντικά την συσχέτιση.

```
   f, (ax1, ax2) = plt.subplots(2, 1, figsize=(24,20))

   #Όλα τα δεδομένα
   corr = df.corr()
   sns.heatmap(corr, cmap='coolwarm_r', annot_kws={'size':20}, ax=
ax1)
   ax1.set_title("Συσχέτιση χαρακτηριστικών από αρχικά δεδομένα",
fontsize=14)


   sub_sample_corr = new_df.corr()
   sns.heatmap(sub_sample_corr, cmap='coolwarm_r', annot_kws={'siz
e':20}, ax=ax2)
   ax2.set_title('Συσχέτιση χαρακτηριστικών από δεδομένα με ίση κα
τανομή', fontsize=14)
   plt.show()
```



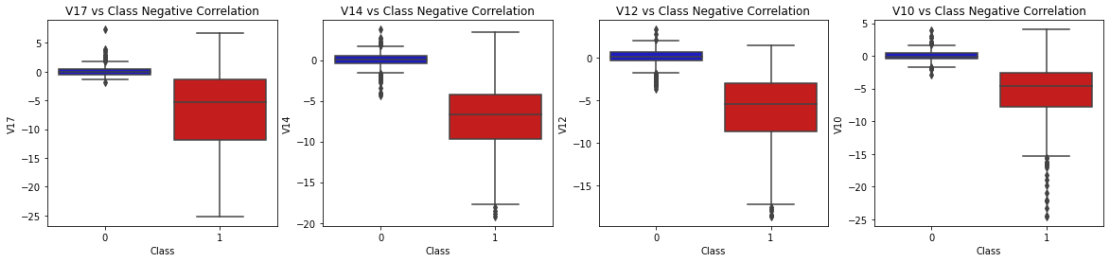
Σχήμα 5: Συσχέτιση χαρακτηριστικών από αρχικά δεδομένα, και από δεδομένα με ίση κατανομή

Στον 2ο πίνακα γίνονται εμφανείς συσχετίσεις δεδομένων. Συγκεκριμένα:

- Αρνητική συσχέτιση: V17, V14, V12 και V10. Όσο χαμηλότερες τόσο πιο πιθανό να είναι παράνομη συναλλαγή.
- Θετική συσχέτιση: V2, V4, V11, και V19. Όσο υψηλότερες τόσο πιο πιθανό να είναι παράνομη συναλλαγή.

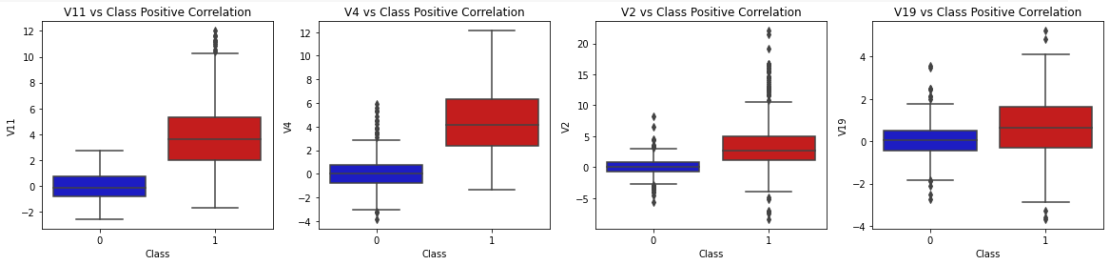
```
   f, axes = plt.subplots(ncols=4, figsize=(20,4))

   # Αρνητική Συσχέτιση
   sns.boxplot(x="Class", y="V17", data=new_df, palette=colors, ax
=axes[0])
   axes[0].set_title('V17 vs Class Negative Correlation')

   sns.boxplot(x="Class", y="V14", data=new_df, palette=colors, ax
=axes[1])
   axes[1].set_title('V14 vs Class Negative Correlation')


   sns.boxplot(x="Class", y="V12", data=new_df, palette=colors, ax
=axes[2])
   axes[2].set_title('V12 vs Class Negative Correlation')
```

```
    sns.boxplot(x="Class", y="V10", data=new_df, palette=colors, ax
=axes[3])
    axes[3].set_title('V10 vs Class Negative Correlation')

    plt.show()
```



Σχήμα 6: Διαγράμματα αρνητικής Συσχέτισης των δύο κλάσεων (fraud και no fraud)

```
    f, axes = plt.subplots(ncols=4, figsize=(20,4))

    # Θετική συσχέτιση
    sns.boxplot(x="Class", y="V11", data=new_df, palette=colors, ax
=axes[0])
    axes[0].set_title('V11 vs Class Positive Correlation')

    sns.boxplot(x="Class", y="V4", data=new_df, palette=colors, ax=
axes[1])
    axes[1].set_title('V4 vs Class Positive Correlation')


    sns.boxplot(x="Class", y="V2", data=new_df, palette=colors, ax=
axes[2])
    axes[2].set_title('V2 vs Class Positive Correlation')


    sns.boxplot(x="Class", y="V19", data=new_df, palette=colors, ax
=axes[3])
    axes[3].set_title('V19 vs Class Positive Correlation')

    plt.show()
```



Σχήμα 7: Διαγράμματα θετικής Συσχέτισης των δύο κλάσεων (fraud και no fraud)

Αφαίρεση ακραίων τιμών

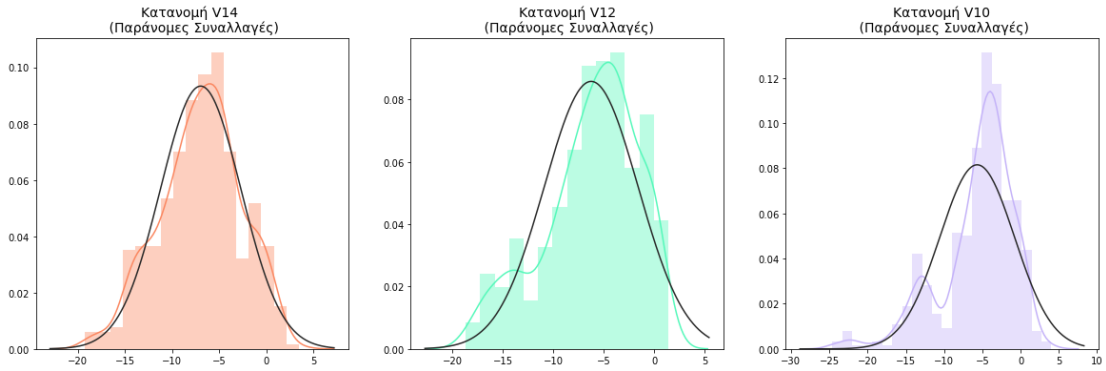
Πριν γίνει η εκπαίδευση, πρέπει να αφαιρεθούν παρατηρήσεις από τα δεδομένα που απέχουν πολύ από τις υπόλοιπες (ακραίες τιμές). Μπορεί να χρησιμοποιηθεί η Interquartile Range Method σύμφωνα με την οποία υπολογίζεται ένα όριο βάσει των τιμών που βρίσκονται στο εύρος μεταξύ 25% και 75 % του μέσου όρου του συνόλου. Παρατηρήσεις που βρίσκονται εκτός θα αφαιρούνται. Απαιτείται προσοχή στην επιλογή του εύρους καθώς

αφαίρεση λιγότερων δεδομένων οδηγεί σε μικρότερη ακρίβεια ταξινόμησης ενώ αφαίρεση περισσότερων οδηγεί σε underfitting.

Τα βήματα για την αφαίρεση είναι τα ακόλουθα:

- **Παρατήρηση Κατανομής**: Μελετάται η κατανομή των χαρακτηριστικών με καλή συσχέτιση και επιλέγεται αυτό με γκαουσιανή κατανομή. Εδώ μόνο το V14 έχει τέτοια κατανομή, ενώ τα V12 και V10 απέχουν αρκετά.
- **Επιλογή ορίου:** Το όριο θα πολλαπλασιαστεί με ένα συντελεστή ο οποίος και καθορίζει πόσο αυστηρό θα είναι.
- **Απόρριψη δεδομένων:** Αφαιρούνται τα δεδομένα που είναι εκτός ορίων
- **Έλεγχος αποτελεσμάτων:** Εμφάνιση με χρήση boxplot των τελικών αποτελεσμάτων.

```
from scipy.stats import norm

f, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(20, 6))

v14_fraud_dist = new_df['V14'].loc[new_df['Class'] == 1].values
sns.distplot(v14_fraud_dist,ax=ax1, fit=norm, color='#FB8861')
ax1.set_title('Κατανομή V14 \n (Παράνομες Συναλλαγές)', fontsiz
e=14)

v12_fraud_dist = new_df['V12'].loc[new_df['Class'] == 1].values
sns.distplot(v12_fraud_dist,ax=ax2, fit=norm, color='#56F9BB')
ax2.set_title('Κατανομή V12\n (Παράνομες Συναλλαγές)', fontsize
=14)


v10_fraud_dist = new_df['V10'].loc[new_df['Class'] == 1].values
sns.distplot(v10_fraud_dist,ax=ax3, fit=norm, color='#C5B3F9')
ax3.set_title('Κατανομή V10\n (Παράνομες Συναλλαγές)', fontsize
=14)

plt.show()
```



Σχήμα 8: Κατανομή Παράνομων Συναλλαγών

```
# -----> Αφαίρεση Ακραίων τιμών (V14)
v14_fraud = new_df['V14'].loc[new_df['Class'] == 1].values
q25, q75 = np.percentile(v14_fraud, 25), np.percentile(v14_frau
d, 75)
print('25%: {} | 75%: {}'.format(q25, q75))
v14_iqr = q75 - q25
print('iqr: {}'.format(v14_iqr))
```

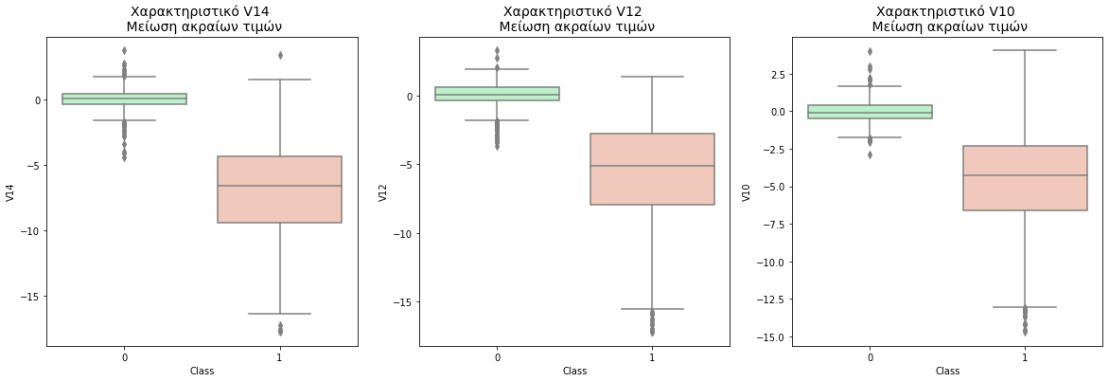
```
    v14_cut_off = v14_iqr * 1.5
    v14_lower, v14_upper = q25 - v14_cut_off, q75 + v14_cut_off
    print('Όριο αποκοπής: {}'.format(v14_cut_off))
    print('V14 κάτω: {}'.format(v14_lower))
    print('V14 άνω: {}'.format(v14_upper))

    outliers = [x for x in v14_fraud if x < v14_lower or x > v14_up
per]
    print('Ακραίες τιμές για το χαρακτηριστικό V14 (παράνομες συναλ
λαγές): {}'.format(len(outliers)))

    new_df = new_df.drop(new_df[(new_df['V14'] > v14_upper) | (new_
df['V14'] < v14_lower)].index)
    print('----' * 44)

    # -----> Αφαίρεση Ακραίων τιμών (V12)
    v12_fraud = new_df['V12'].loc[new_df['Class'] == 1].values
    q25, q75 = np.percentile(v12_fraud, 25), np.percentile(v12_frau
d, 75)
    v12_iqr = q75 - q25

    v12_cut_off = v12_iqr * 1.5
    v12_lower, v12_upper = q25 - v12_cut_off, q75 + v12_cut_off
    print('V12 κάτω: {}'.format(v12_lower))
    print('V12 άνω: {}'.format(v12_upper))
    outliers = [x for x in v12_fraud if x < v12_lower or x > v12_up
per]
    print('Ακραίες τιμές για το χαρακτηριστικό V12: {}'.format(outl
iers))
    print('Ακραίες τιμές για το χαρακτηριστικό V12 (παράνομες συναλ
λαγές): {}'.format(len(outliers)))
    new_df = new_df.drop(new_df[(new_df['V12'] > v12_upper) | (new_
df['V12'] < v12_lower)].index)
    print('Χαρακτηριστικά που απομένουν: {}'.format(len(new_df)))
    print('----' * 44)


    # -----> Αφαίρεση Ακραίων τιμών (V10)
    v10_fraud = new_df['V10'].loc[new_df['Class'] == 1].values
    q25, q75 = np.percentile(v10_fraud, 25), np.percentile(v10_frau
d, 75)
    v10_iqr = q75 - q25

    v10_cut_off = v10_iqr * 1.5
    v10_lower, v10_upper = q25 - v10_cut_off, q75 + v10_cut_off
    print('V10 κάτω: {}'.format(v10_lower))
    print('V10 άνω: {}'.format(v10_upper))
    outliers = [x for x in v10_fraud if x < v10_lower or x > v10_up
per]
    print('Ακραίες τιμές για το χαρακτηριστικό V10: {}'.format(outl
iers))
    print('Ακραίες τιμές για το χαρακτηριστικό V10 (παράνομες συναλ
λαγές): {}'.format(len(outliers)))
    new_df = new_df.drop(new_df[(new_df['V10'] > v10_upper) | (new_
df['V10'] < v10_lower)].index)
    print('Χαρακτηριστικά που απομένουν: {}'.format(len(new_df)))
```

```
25%: -9.692722964972385 | 75%: -4.282820849486866
```

```
   iqr: 5.409902115485519
   Οριο αποκοπής: 8.114853173228278
   V14 κάτω: -17.807576138200663
   V14 άνω: 3.8320323237414122
   Ακραίες τιμές για το χαρακτηριστικό V14 (παράνομες συναλλαγές):
4
   ------------------------------------------------------------------
------------------------------------------------------------------
------------------------------------------------

   V12 κάτω: -17.3430371579634
   V12 άνω: 5.776973384895937
   Ακραίες τιμές για το χαρακτηριστικό V12: [-18.553697009645802,
-18.683714633344298, -18.047596570821604, -18.4311310279993]
   Ακραίες τιμές για το χαρακτηριστικό V12 (παράνομες συναλλαγές):
4
   Χαρακτηριστικά που απομένουν: 976
   ------------------------------------------------------------------
------------------------------------------------------------------
------------------------------------------------
   V10 κάτω: -14.89885463232024
   V10 άνω: 4.920334958342141
   Ακραίες τιμές για το χαρακτηριστικό V10: [-22.1870885620007, -1
6.6011969664137, -15.124162814494698, -24.403184969972802, -16.64
96281595399, -20.949191554361104, -15.2399619587112, -15.56379133
8730098, -23.2282548357516, -15.1237521803455, -16.2556117491401,
-22.1870885620007, -18.2711681738888, -15.346098846877501, -22.18
70885620007, -19.836148851696, -24.5882624372475, -14.92465477354
87, -18.9132433348732, -16.7460441053944, -15.2318333653018, -15.
2399619587112, -15.563791338730098, -16.3035376590131, -17.141513
641289198, -14.9246547735487, -22.1870885620007]
   Ακραίες τιμές για το χαρακτηριστικό V10 (παράνομες συναλλαγές):
27
   Χαρακτηριστικά που απομένουν: 949
```

```
   f,(ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20,6))

   colors = ['#B3F9C5', '#f9c5b3']
   # Αφού αφαιρέθηκαν οι ακραίες τιμές τα δεδομένα επηρεάστηκαν?
   # Χαρακτηριστικό V14
   sns.boxplot(x="Class", y="V14", data=new_df,ax=ax1, palette=col
ors)
   ax1.set_title("Χαρακτηριστικό V14 \n Μείωση ακραίων τιμών", fon
tsize=14)

   # Χαρακτηριστικό 12
   sns.boxplot(x="Class", y="V12", data=new_df, ax=ax2, palette=co
lors)
   ax2.set_title("Χαρακτηριστικό V12 \n Μείωση ακραίων τιμών", fon
tsize=14)


   # Χαρακτηριστικό V10
   sns.boxplot(x="Class", y="V10", data=new_df, ax=ax3, palette=co
lors)
   ax3.set_title("Χαρακτηριστικό V10 \n Μείωση ακραίων τιμών", fon
tsize=14)
```

```
plt.show()
```



Σχήμα 9: Μείωση Ακραίων Τιμών

Ταξινομητές

Εδώ θα γίνει η εκπαίδευση και δοκιμή τεσσάρων διαφορετικών ταξινομητών. Θα μελετηθούν 4 ταξινομητές:

1. Logistic Regression
2. K-Nearest Neighbors Classifier
3. Support Vector Classifier
4. Decision Tree Classifier

```
X = new_df.drop('Class', axis=1)
y = new_df['Class']

from sklearn.model_selection import train_test_split

# Προετοιμασία δεδομένων
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size=0.2, random_state=42)

X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values


# Επιλογή ταξινομητών

classifiers = {
 "LogisiticRegression": LogisticRegression(),
 "KNearest": KNeighborsClassifier(),
 "Support Vector Classifier": SVC(),
 "DecisionTreeClassifier": DecisionTreeClassifier()
}


from sklearn.model_selection import cross_val_score


for key, classifier in classifiers.items():
 classifier.fit(X_train, y_train)
```

```
    training_score = cross_val_score(classifier, X_train, y_train,
cv=5)
    print("Classifiers: ", classifier.__class__.__name__, "Has a t
raining score of", round(training_score.mean(), 2) * 100, "% accu
racy score")
```

```
Classifiers: LogisticRegression Has a training score of 94.0 %
accuracy score

Classifiers: KNeighborsClassifier Has a training score of 93.0
% accuracy score

Classifiers: SVC Has a training score of 93.0 % accuracy score

Classifiers: DecisionTreeClassifier Has a training score of 90.
0 % accuracy score
```

```
# Θα χρησιμοποιηθεί η GridSearchCV για την εύρεση παραμέτρων τα
ξινομητών.
from sklearn.model_selection import GridSearchCV


# Logistic Regression
log_reg_params = {"penalty": ['l1', 'l2'], 'C': [0.001, 0.01, 0
.1, 1, 10, 100, 1000]}


grid_log_reg = GridSearchCV(LogisticRegression(), log_reg_param
s)
grid_log_reg.fit(X_train, y_train)
# Χρησιμοποιούνται αυτόματα οι καλύτερες παράμετροι.
log_reg = grid_log_reg.best_estimator_

knears_params = {"n_neighbors": list(range(2,5,1)), 'algorithm'
: ['auto', 'ball_tree', 'kd_tree', 'brute']}

grid_knears = GridSearchCV(KNeighborsClassifier(), knears_param
s)
grid_knears.fit(X_train, y_train)
# Βέλτιστες παράμετροι για KNN
knears_neighbors = grid_knears.best_estimator_

# Support Vector Classifier
svc_params = {'C': [0.5, 0.7, 0.9, 1], 'kernel': ['rbf', 'poly'
, 'sigmoid', 'linear']}
grid_svc = GridSearchCV(SVC(), svc_params)
grid_svc.fit(X_train, y_train)

# Βέλτιστες παράμετροι για SVC
svc = grid_svc.best_estimator_

# DecisionTree Classifier
tree_params = {"criterion": ["gini", "entropy"], "max_depth": l
ist(range(2,4,1)),
    "min_samples_leaf": list(range(5,7,1))}
grid_tree = GridSearchCV(DecisionTreeClassifier(), tree_params)
grid_tree.fit(X_train, y_train)
```

```
   # Βέλτιστες παράμετροι για DecisionTree
   tree_clf = grid_tree.best_estimator_
```

```
   #εκπαίδευση και πρόβλεψη με τα imbalanced (αρχικά) δεδομένα
   #Οι προβλέψεις υποδεικνύουν overfitting
   log_reg_score = cross_val_score(log_reg, X_train, y_train, cv=5
)
   print('Logistic Regression CV Score: ', round(log_reg_score.mea
n() * 100, 2).astype(str) + '%')


   knears_score = cross_val_score(knears_neighbors, X_train, y_tra
in, cv=5)
   print('KNN CV Score', round(knears_score.mean() * 100, 2).astyp
e(str) + '%')

   svc_score = cross_val_score(svc, X_train, y_train, cv=5)
   print('SVC CV Score', round(svc_score.mean() * 100, 2).astype(s
tr) + '%')

   tree_score = cross_val_score(tree_clf, X_train, y_train, cv=5)
   print('DecisionTree CV Score', round(tree_score.mean() * 100, 2
).astype(str) + '%')
```

```
   Logistic Regression CV Score: 93.92%

   KNN CV Score 93.52%

   SVC CV Score 94.05%

   Decision Tree CV Score 92.34%
```

```
   # Undersample ενώ γίνεται CrossValidation
   undersample_X = df.drop('Class', axis=1)
   undersample_y = df['Class']

   for train_index, test_index in sss.split(undersample_X, undersa
mple_y):
    print("Train:", train_index, "Test:", test_index)
    undersample_Xtrain, undersample_Xtest = undersample_X.iloc[tra
in_index], undersample_X.iloc[test_index]
    undersample_ytrain, undersample_ytest = undersample_y.iloc[tra
in_index], undersample_y.iloc[test_index]

   undersample_Xtrain = undersample_Xtrain.values
   undersample_Xtest = undersample_Xtest.values
   undersample_ytrain = undersample_ytrain.values
   undersample_ytest = undersample_ytest.values

   undersample_accuracy = []
   undersample_precision = []
   undersample_recall = []
   undersample_f1 = []
   undersample_auc = []

   # Χρήση τεχνικής NearMiss και Cross Validation
```

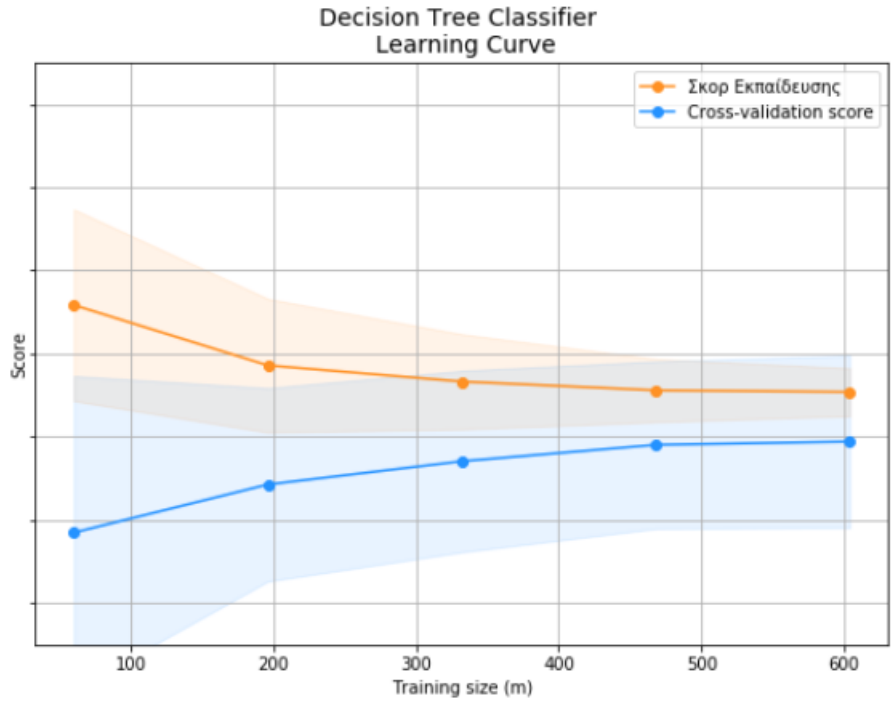
```
    for train, test in sss.split(undersample_Xtrain, undersample_yt
rain):
        undersample_pipeline = imbalanced_make_pipeline(NearMiss(sampl
ing_strategy='majority'), log_reg)
        undersample_model = undersample_pipeline.fit(undersample_Xtrai
n[train], undersample_ytrain[train])
        undersample_prediction = undersample_model.predict(undersample
_Xtrain[test])

        undersample_accuracy.append(undersample_pipeline.score(origina
l_Xtrain[test], original_ytrain[test]))
        undersample_precision.append(precision_score(original_ytrain[t
est], undersample_prediction))
        undersample_recall.append(recall_score(original_ytrain[test],
undersample_prediction))
        undersample_f1.append(f1_score(original_ytrain[test], undersam
ple_prediction))
        undersample_auc.append(roc_auc_score(original_ytrain[test], un
dersample_prediction))
```

```
    #συνάρτηση γενικής χρήσης για plot του learning curve
    from sklearn.model_selection import ShuffleSplit
    from sklearn.model_selection import learning_curve

    def plot_learning_curve(estimator1, estimator2, estimator3, est
imator4, X, y, ylim=None, cv=None,
        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
        f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2, figsize=(20,14
), sharey=True)
        if ylim is not None:
        plt.ylim(*ylim)
        # πρώτος εκτιμητής
        train_sizes, train_scores, test_scores = learning_curve(
        estimator1, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_size
s)
        train_scores_mean = np.mean(train_scores, axis=1)
        train_scores_std = np.std(train_scores, axis=1)
        test_scores_mean = np.mean(test_scores, axis=1)
        test_scores_std = np.std(test_scores, axis=1)
        ax1.fill_between(train_sizes, train_scores_mean - train_scores
_std,
        train_scores_mean + train_scores_std, alpha=0.1,
        color="#ff9124")
        ax1.fill_between(train_sizes, test_scores_mean - test_scores_s
td,
        test_scores_mean + test_scores_std, alpha=0.1, color="#2492ff"
)
        ax1.plot(train_sizes, train_scores_mean, 'o-', color="#ff9124"
,
        label="Σκορ Εκπαίδευσης")
        ax1.plot(train_sizes, test_scores_mean, 'o-', color="#2492ff",
        label="Cross-validation score")
        ax1.set_title("Logistic Regression Learning Curve", fontsize=1
4)
        ax1.set_xlabel('Training size (m)')
        ax1.set_ylabel('Score')
        ax1.grid(True)
        ax1.legend(loc="best")
```

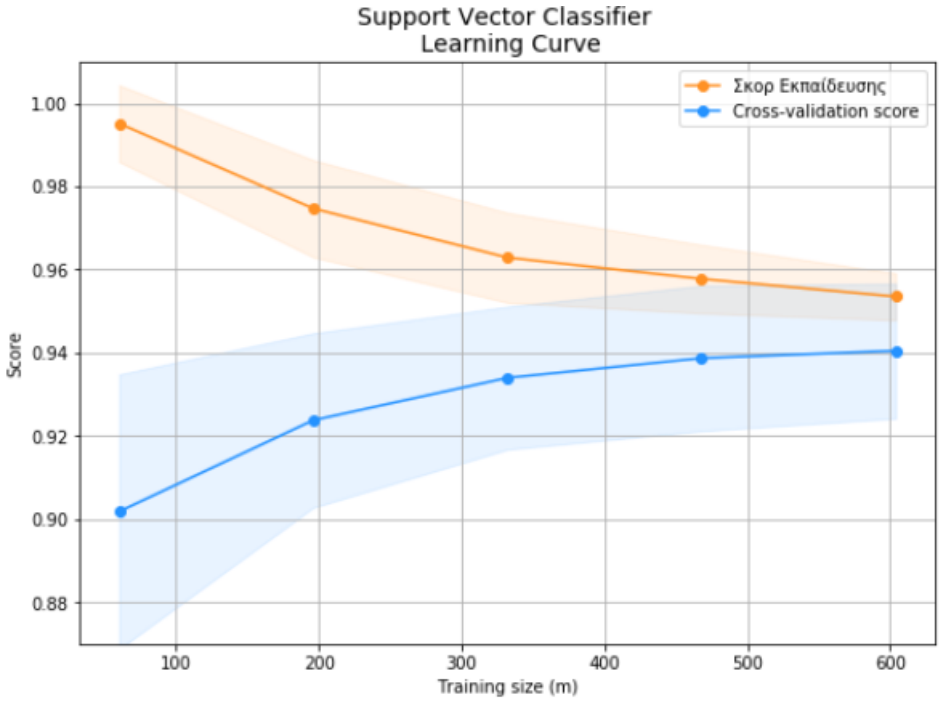
```
    # Δεύτερος εκτιμητής
    train_sizes, train_scores, test_scores = learning_curve(
    estimator2, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_size
s)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    ax2.fill_between(train_sizes, train_scores_mean - train_scores
_std,
    train_scores_mean + train_scores_std, alpha=0.1,
    color="#ff9124")
    ax2.fill_between(train_sizes, test_scores_mean - test_scores_s
td,
    test_scores_mean + test_scores_std, alpha=0.1, color="#2492ff"
)
    ax2.plot(train_sizes, train_scores_mean, 'o-', color="#ff9124"
,
    label="Σκορ Εκπαίδευσης")
    ax2.plot(train_sizes, test_scores_mean, 'o-', color="#2492ff",
    label="Cross-validation score")
    ax2.set_title("Knears Neighbors Learning Curve", fontsize=14)
    ax2.set_xlabel('Training size (m)')
    ax2.set_ylabel('Score')
    ax2.grid(True)
    ax2.legend(loc="best")

    # Τρίτος εκτιμητής
    train_sizes, train_scores, test_scores = learning_curve(
    estimator3, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_size
s)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    ax3.fill_between(train_sizes, train_scores_mean - train_scores
_std,
    train_scores_mean + train_scores_std, alpha=0.1,
    color="#ff9124")
    ax3.fill_between(train_sizes, test_scores_mean - test_scores_s
td,
    test_scores_mean + test_scores_std, alpha=0.1, color="#2492ff"
)
    ax3.plot(train_sizes, train_scores_mean, 'o-', color="#ff9124"
,
    label="Σκορ Εκπαίδευσης")
    ax3.plot(train_sizes, test_scores_mean, 'o-', color="#2492ff",
    label="Cross-validation score")
    ax3.set_title("Support Vector Classifier \n Learning Curve", f
ontsize=14)
    ax3.set_xlabel('Training size (m)')
    ax3.set_ylabel('Score')
    ax3.grid(True)
    ax3.legend(loc="best")

    # τέταρτος εκτιμητής
    train_sizes, train_scores, test_scores = learning_curve(
```

```
    estimator4, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_size
s)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    ax4.fill_between(train_sizes, train_scores_mean - train_scores
_std,
    train_scores_mean + train_scores_std, alpha=0.1,
    color="#ff9124")
    ax4.fill_between(train_sizes, test_scores_mean - test_scores_s
td,
    test_scores_mean + test_scores_std, alpha=0.1, color="#2492ff"
)
    ax4.plot(train_sizes, train_scores_mean, 'o-', color="#ff9124"
,
    label="Σκορ Εκπαίδευσης")
    ax4.plot(train_sizes, test_scores_mean, 'o-', color="#2492ff",
    label="Cross-validation score")
    ax4.set_title("Decision Tree Classifier \n Learning Curve", fo
ntsize=14)
    ax4.set_xlabel('Training size (m)')
    ax4.set_ylabel('Score')
    ax4.grid(True)
    ax4.legend(loc="best")
    return plt
```

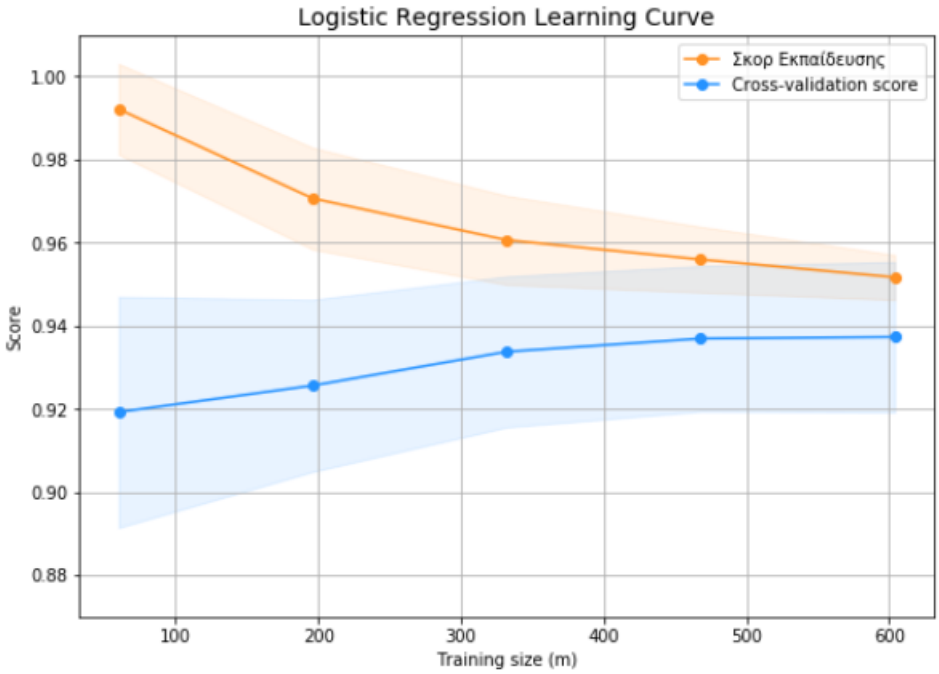
```
   cv = ShuffleSplit(n_splits=100, test_size=0.2, random_state=42)
   plot_learning_curve(log_reg, knears_neighbors, svc, tree_clf, X
_train, y_train, (0.87, 1.01), cv=cv, n_jobs=4)
```



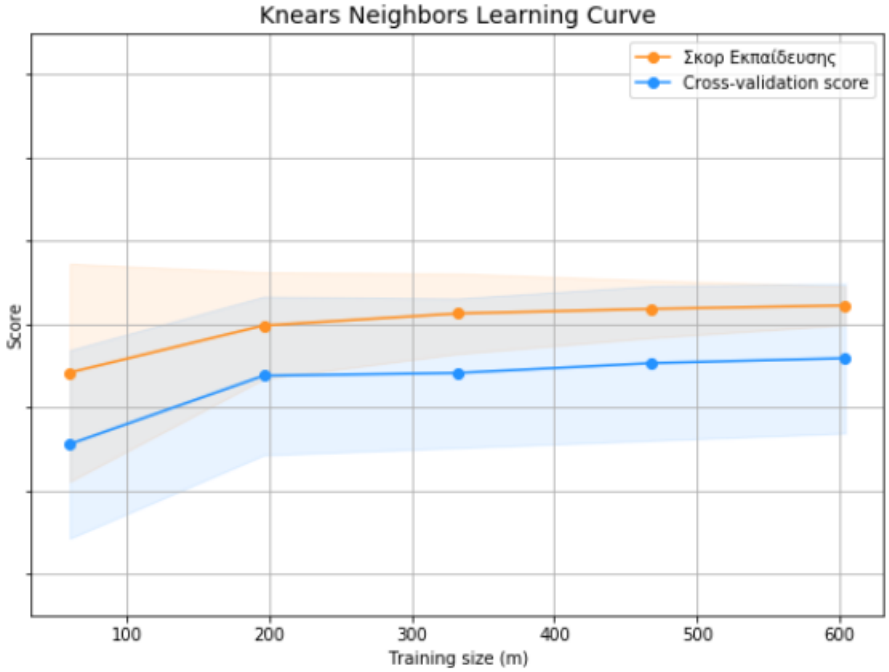
Σχήμα 10: Καμπύλη Εκπαίδευσης Δέντρου Απόφασης



Σχήμα 11: Καμπύλη Εκπαίδευσης Μηχανών Διανυσμάτων Υποστήριξης (SVC)



Σχήμα 12: Καμπύλη Εκπαίδευσης Λογιστικής Παλινδρόμησης



Σχήμα 13: Καμπύλη Εκπαίδευσης K- nearest

Σύγκριση ταξινομητών με χρήση τεχνικής Undersampling

Τα τελικά αποτελέσματα των ταξινομητών που δοκιμάστηκαν είναι:

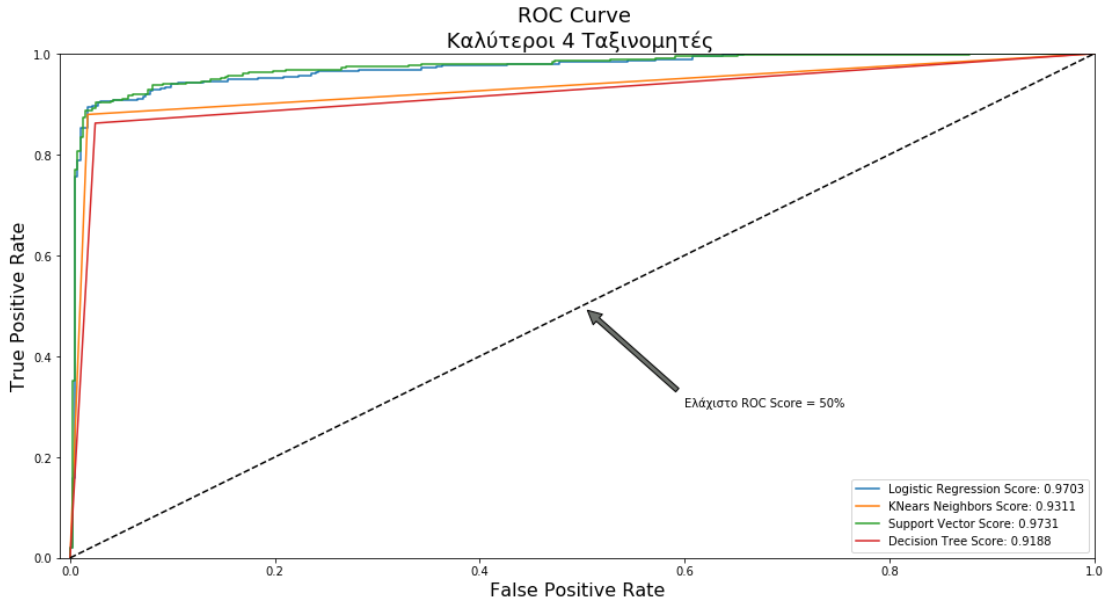
```
from sklearn.metrics import roc_curve
from sklearn.model_selection import cross_val_predict
# Αποθήκευση όλων των αποτελεσμάτων σε ένα dataframe

log_reg_pred = cross_val_predict(log_reg, X_train, y_train, cv=
5,
 method="decision_function")

knears_pred = cross_val_predict(knears_neighbors, X_train, y_tr
ain, cv=5)

svc_pred = cross_val_predict(svc, X_train, y_train, cv=5,
 method="decision_function")

tree_pred = cross_val_predict(tree_clf, X_train, y_train, cv=5)
```

```
from sklearn.metrics import roc_auc_score

print('Τα τελικά ROC σκορ είναι:')
print('Logistic Regression: ', roc_auc_score(y_train, log_reg_p
red))
print('KNears Neighbors: ', roc_auc_score(y_train, knears_pred)
)
print('Support Vector Classifier: ', roc_auc_score(y_train, svc
_pred))
print('Decision Tree Classifier: ', roc_auc_score(y_train, tree
_pred))
```

```
Τα τελικά ROC σκορ είναι:

Logistic Regression: 0.9703205995041694

KNears Neighbors: 0.9310767410412442

Support Vector Classifier: 0.973081473968898

Decision Tree Classifier: 0.9187795807978364
```

```
log_fpr, log_tpr, log_thresold = roc_curve(y_train, log_reg_pre
d)
knear_fpr, knear_tpr, knear_threshold = roc_curve(y_train, knea
rs_pred)
svc_fpr, svc_tpr, svc_threshold = roc_curve(y_train, svc_pred)
tree_fpr, tree_tpr, tree_threshold = roc_curve(y_train, tree_pr
ed)


def graph_roc_curve_multiple(log_fpr, log_tpr, knear_fpr, knear
_tpr, svc_fpr, svc_tpr, tree_fpr, tree_tpr):
 plt.figure(figsize=(16,8))
 plt.title('ROC Curve \n Καλύτεροι 4 Ταξινομητές', fontsize=18)
 plt.plot(log_fpr, log_tpr, label='Logistic Regression Score: {
:.4f}'.format(roc_auc_score(y_train, log_reg_pred)))
```

```
    plt.plot(knear_fpr, knear_tpr, label='KNears Neighbors Score:
{:.4f}'.format(roc_auc_score(y_train, knears_pred)))
    plt.plot(svc_fpr, svc_tpr, label='Support Vector Score: {:.4f}
'.format(roc_auc_score(y_train, svc_pred)))
    plt.plot(tree_fpr, tree_tpr, label='Decision Tree Score: {:.4f
}'.format(roc_auc_score(y_train, tree_pred)))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.01, 1, 0, 1])
    plt.xlabel('False Positive Rate', fontsize=16)
    plt.ylabel('True Positive Rate', fontsize=16)
    plt.annotate('Ελάχιστο ROC Score = 50%', xy=(0.5, 0.5), xytext
=(0.6, 0.3),
    arrowprops=dict(facecolor='#6E726D', shrink=0.05),
    )
    plt.legend()

   graph_roc_curve_multiple(log_fpr, log_tpr, knear_fpr, knear_tpr
, svc_fpr, svc_tpr, tree_fpr, tree_tpr)
   plt.show()
```



Σχήμα 14: Καμπύλη ROC των καλύτερων τεσσάρων ταξινομητών

Τεχνική SMOTE (Over-Sampling)

SMOTE σημαίνει Synthetic Minority Over-sampling Technique. Σε αντίθεση με την Random UnderSampling, η τεχνική SMOTE δημιουργεί νέα τεχνητά σημεία ώστε να υπάρχει ίσος αριθμός παρατηρήσεων σε κάθε κλάση. Η τεχνική SMOTE δημιουργεί σημεία μεταξύ των κοντινότερων γειτόνων της κλάσης με τα λιγότερα μέλη. Συνολικά διατηρεί όλη την πληροφορία.

Είναι σημαντικό εδώ να αναφερθεί ότι όταν χρησιμοποιείται η τεχνική Cross Validation σε συνδυασμό με under ή over sampling απαιτείται αυτά να γίνονται με συγκεκριμένη σειρά. Πρέπει πρώτα να εφαρμόζεται το CrossValidation και για κάθε δείγμα αυτού να γίνεται επαναληπτικά η διαδικασία του under ή over sampling. Σε αντίθετη περίπτωση το μοντέλο που θα παραχθεί θα είναι overfitted.

```
   from imblearn.over_sampling import SMOTE
   from sklearn.model_selection import train_test_split, Randomize
dSearchCV


   print('Length of X (train): {} | Length of y (train): {}'.forma
t(len(original_Xtrain), len(original_ytrain)))
   print('Length of X (test): {} | Length of y (test): {}'.format(
len(original_Xtest), len(original_ytest)))

   # Αρχικοποίηση μεταβλητών
   accuracy_lst = []
   precision_lst = []
   recall_lst = []
   f1_lst = []
   auc_lst = []

   # Ταξινομητής με τις βέλτιστες παραμέτρους
   log_reg_sm = LogisticRegression()

   rand_log_reg = RandomizedSearchCV(LogisticRegression(), log_reg
_params, n_iter=4)


   # Τεχνική SMOTE
   # Παράμετροι
   log_reg_params = {"penalty": ['l1', 'l2'], 'C': [0.001, 0.01, 0
.1, 1, 10, 100, 1000]}
   # Συνδυασμός με cross validation
   for train, test in sss.split(original_Xtrain, original_ytrain):
    pipeline = imbalanced_make_pipeline(SMOTE(sampling_strategy='m
inority'), rand_log_reg) # SMOTE κατά την διάρκεια του Cross Vali
dation
    model = pipeline.fit(original_Xtrain[train], original_ytrain[t
rain])
    best_est = rand_log_reg.best_estimator_
    prediction = best_est.predict(original_Xtrain[test])

    accuracy_lst.append(pipeline.score(original_Xtrain[test], orig
inal_ytrain[test]))
    precision_lst.append(precision_score(original_ytrain[test], pr
ediction))
    recall_lst.append(recall_score(original_ytrain[test], predicti
on))
    f1_lst.append(f1_score(original_ytrain[test], prediction))
    auc_lst.append(roc_auc_score(original_ytrain[test], prediction
))

   print('---' * 45)
   print('')
   print("accuracy: {}".format(np.mean(accuracy_lst)))
   print("precision: {}".format(np.mean(precision_lst)))
   print("recall: {}".format(np.mean(recall_lst)))
   print("f1: {}".format(np.mean(f1_lst)))
   print('---' * 45)
```

```
Length of X (train): 227846 | Length of y (train): 227846
Length of X (test): 56961 | Length of y (test): 56961
```

```
   ------------------------------------------------------------------
------------------------------------------------------------------
-------
```

```
   accuracy: 0.9422335776596601
   precision: 0.06189106389635361
   recall: 0.9137293086660175
   f1: 0.11405943663666888
   ------------------------------------------------------------------
------------------------------------------------------------------
-------
```

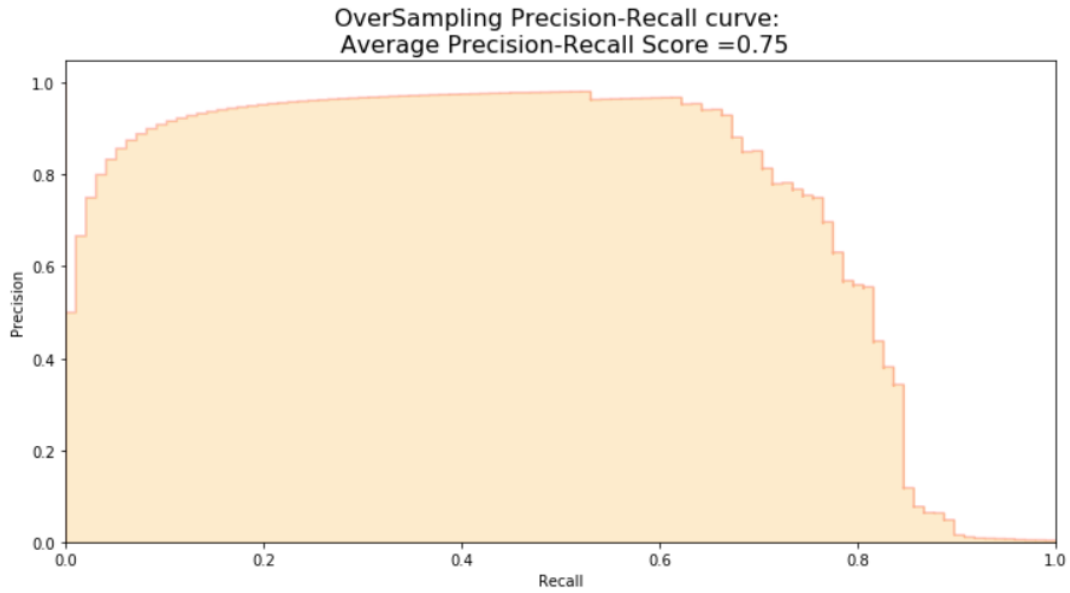
```
   labels = ['No Fraud', 'Fraud']
   smote_prediction = best_est.predict(original_Xtest)
   print(classification_report(original_ytest, smote_prediction, t
arget_names=labels))
```

```
    precision recall f1-score support

    No Fraud 1.00 0.99 0.99 56863
    Fraud 0.11 0.86 0.20 98

    accuracy 0.99 56961
    macro avg 0.56 0.92 0.60 56961
   weighted avg 1.00 0.99 0.99 56961
```

```
   y_score = best_est.decision_function(original_Xtest)
   from sklearn.metrics import average_precision_score
   average_precision = average_precision_score(original_ytest, y_s
core)

   print('Average precision-recall score: {0:0.2f}'.format(
    average_precision))
```

```
   Average precision-recall score: 0.75
```

```
   from sklearn.metrics import precision_recall_curve
   import matplotlib.pyplot as plt
   fig = plt.figure(figsize=(12,6))

   precision, recall, _ = precision_recall_curve(original_ytest, y
_score)

   plt.step(recall, precision, color='r', alpha=0.2,
    where='post')
   plt.fill_between(recall, precision, step='post', alpha=0.2,
    color='#F59B00')

   plt.xlabel('Recall')
   plt.ylabel('Precision')
   plt.ylim([0.0, 1.05])
   plt.xlim([0.0, 1.0])
   plt.title('OverSampling Precision-Recall curve: \n Average Prec
ision-Recall Score ={0:0.2f}'.format(
    average_precision), fontsize=16)
```



Σχήμα 15: Καμπύλη Oversampling

```
# SMOTE Technique (OverSampling) μετά από διαχωρισμό δεδομένων και Cross Validating
sm = SMOTE(sampling_strategy ='minority', random_state=42)
# Xsm_train, ysm_train = sm.fit_sample(X_train, y_train)


# Τελικά δεδομένα που θα χρησιμοποιηθούν
Xsm_train, ysm_train = sm.fit_sample(original_Xtrain, original_ytrain)
# Logistic Regression - με τις βέλτιστες παραμέτρους από πριν (undersampling)
t0 = time.time()
log_reg_sm = grid_log_reg.best_estimator_
log_reg_sm.fit(Xsm_train, ysm_train)
t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
```

```
Fitting oversample data took :4.702227354049683 sec
```

Σύγκριση Undersampling – SMOTE

Όπως αναφέρθηκε και προηγουμένως, από τα δεδομένα του undersampling παρατηρήθηκε ότι καλύτερη απόδοση είχε ο ταξινομητής logistic regression. Με τον ακόλουθο κώδικα γίνεται σύγκριση της απόδοσης του ταξινομητή με τις δύο ομάδες δεδομένων.

```
# Τελικό σκορ για τα δεδομένα test με logistic regression
from sklearn.metrics import accuracy_score

# Logistic Regression με την τεχνική Undersampling
y_pred = log_reg.predict(X_test)
undersample_score = accuracy_score(y_test, y_pred)



# Logistic Regression με την τεχνική SMOTE
```

```
y_pred_sm = best_est.predict(original_Xtest)
oversample_score = accuracy_score(original_ytest, y_pred_sm)


d = {'Technique': ['Random UnderSampling', 'Oversampling (SMOTE
)'], 'Score': [undersample_score, oversample_score]}
final_df = pd.DataFrame(data=d)

# Αναδιάταξη στηλών
score = final_df['Score']
final_df.drop('Score', axis=1, inplace=True)
final_df.insert(1, 'Score', score)

final_df
```

Ο κώδικας αυτός δίνει τα ακόλουθα αποτελέσματα:

	Technique	Score
0	Random UnderSampling	0.947368
1	Oversampling (SMOTE)	0.988080

Νευρωνικά Δίκτυα

Για σκοπούς σύγκρισης έγιναν και δοκιμές με χρήση νευρωνικού δικτύου. Δημιουργήθηκε με τη χρήση των βιβλιοθηκών keras και tensorflow της python, απλό νευρωνικό δίκτυο με τρία επίπεδα. Το πρώτο επίπεδο χρησιμοποιείται για εισαγωγή δεδομένων (οπότε έχει και αριθμό nodes ίσο με τα χαρακτηριστικά) καθώς και ένα bias node. Τροφοδοτεί αυτό ένα κρυφό επίπεδο με 32 nodes το οποίο στη συνέχεια δίνει τα δεδομένα του σε ένα επίπεδο εξόδου το οποίο θα δώσει 2 πιθανά αποτελέσματα 0 ή 1 ανάλογα με το αν είναι απάτη ή όχι.

Για το νευρωνικό επιλέχθηκε ρυθμός εκμάθησης ίσος με 0.001 και σαν αλγόριθμος βελτιστοποίησης ο AdamOptimizer. Τέλος, η συνάρτηση ενεργοποίησης θα είναι η "Relu". Έγιναν αρχικά δοκιμές με τα δεδομένα στα οποία εφαρμόστηκε η τεχνική undersample και στη συνέχεια με τα δεδομένα στα οποία εφαρμόστηκε η τεχνική SMOTE.

Ο κώδικας δημιουργίας του νευρωνικού δικτύου για τα δεδομένα με την τεχνική undersample είναι ο ακόλουθος:

```
import keras
from keras import backend as K
from keras.models import Sequential
from keras.layers import Activation
from keras.layers.core import Dense
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

n_inputs = X_train.shape[1]

undersample_model = Sequential([
 Dense(n_inputs, input_shape=(n_inputs, ), activation='relu'),
 Dense(32, activation='relu'),
```

```
    Dense(2, activation='softmax')
])
```

Ο παραπάνω κώδικας δημιουργεί το ακόλουθο νευρωνικό δίκτυο:

```
undersample_model.summary()
```

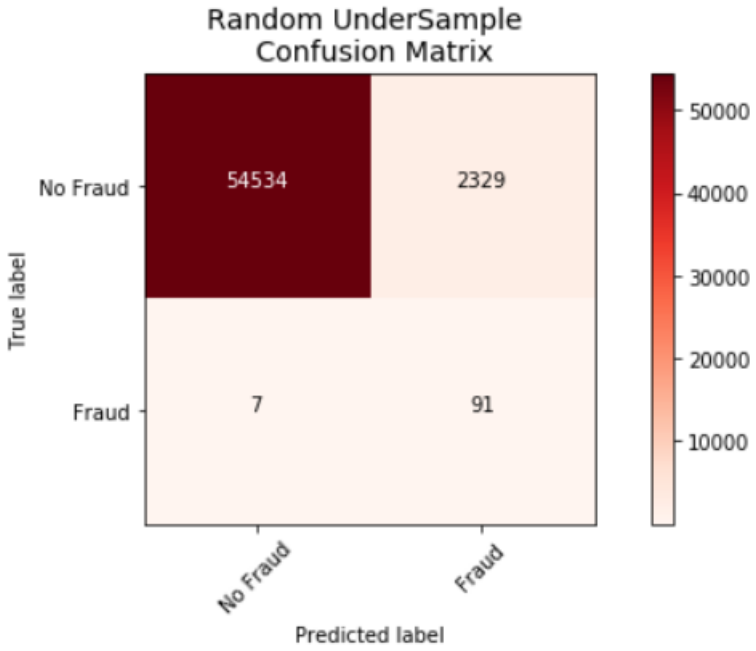
```
Model: "sequential_1"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
dense_1 (Dense) (None, 30) 930
_________________________________________________________________
dense_2 (Dense) (None, 32) 992
_________________________________________________________________
dense_3 (Dense) (None, 2) 66
=================================================================
Total params: 1,988
Trainable params: 1,988
Non-trainable params: 0
_________________________________
```

Με τον ακόλουθο κώδικα γίνεται αρχικά η εκπαίδευση του (fit) και στη συνέχεια η δοκιμή του (predict). Μαζί με τον κώδικα δίνονται και τα διαγράμματα των αποτελεσμάτων.

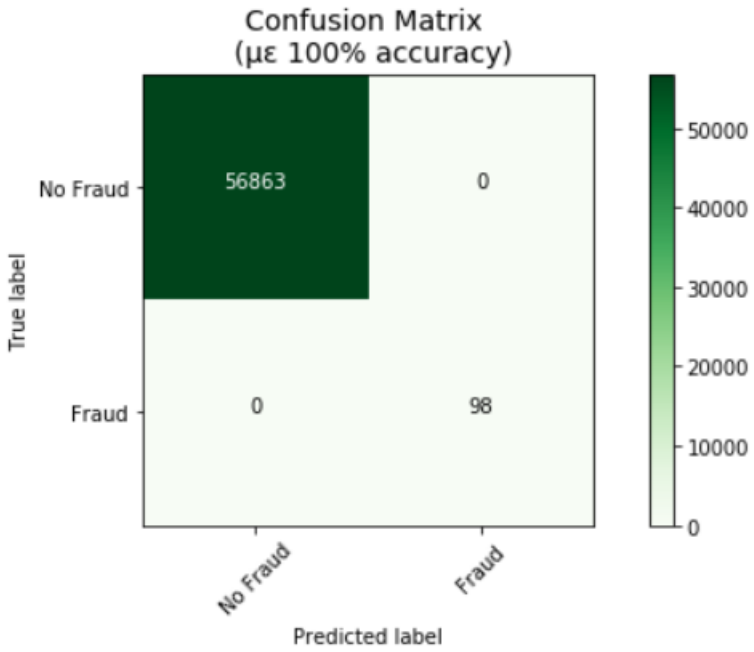
```
undersample_model.compile(Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
undersample_model.fit(X_train, y_train, validation_split=0.2, batch_size=25, epochs=20, shuffle=True, verbose=2)
```

```
undersample_predictions = undersample_model.predict(original_Xtest, batch_size=200, verbose=0)
undersample_fraud_predictions = undersample_model.predict_classes(original_Xtest, batch_size=200, verbose=0)
```

```
undersample_cm = confusion_matrix(original_ytest, undersample_fraud_predictions)
actual_cm = confusion_matrix(original_ytest, original_ytest)
labels = ['No Fraud', 'Fraud']

fig = plt.figure(figsize=(16,8))

fig.add_subplot(221)
conf_matrix_plotter(undersample_cm, labels, title="Random Under Sample \n Confusion Matrix", cmap=plt.cm.Reds)

fig.add_subplot(222)
conf_matrix_plotter(actual_cm, labels, title="Confusion Matrix \n (με 100% accuracy)", cmap=plt.cm.Greens)
```



Σχήμα 16: Random UnderSample Confusion Matrix



Σχήμα 17: Confusion Matrix με 100% ακρίβεια

Η ίδια λογική εφαρμόζεται και για τα δεδομένα από την τεχνική SMOTE. Ο κώδικας και τα αποτελέσματα αυτού εμφανίζονται στην συνέχεια.

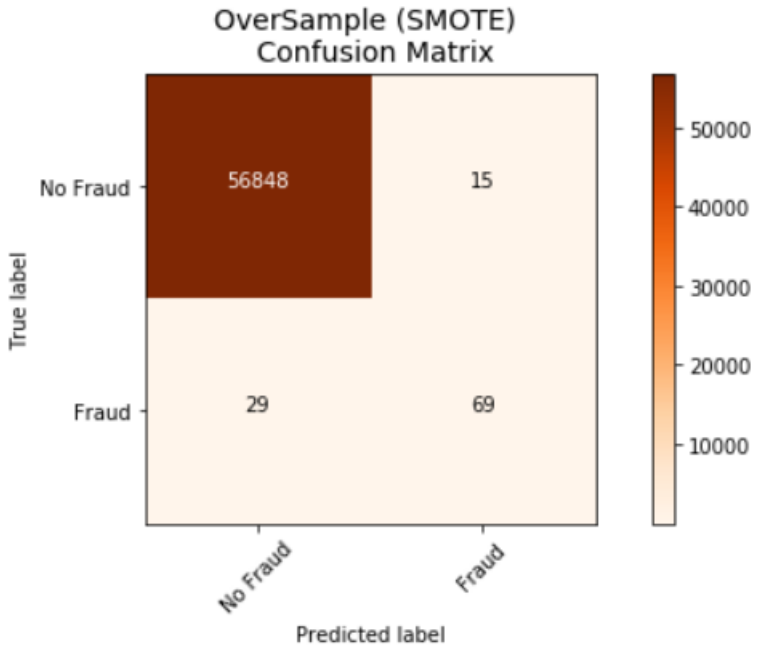
```
   n_inputs = Xsm_train.shape[1]

   oversample_model = Sequential([
    Dense(n_inputs, input_shape=(n_inputs, ), activation='relu'),
    Dense(32, activation='relu'),
    Dense(2, activation='softmax')
   ])

   oversample_model.compile(Adam(lr=0.001), loss='sparse_categoric
al_crossentropy', metrics=['accuracy'])
```

Εκπαίδευση

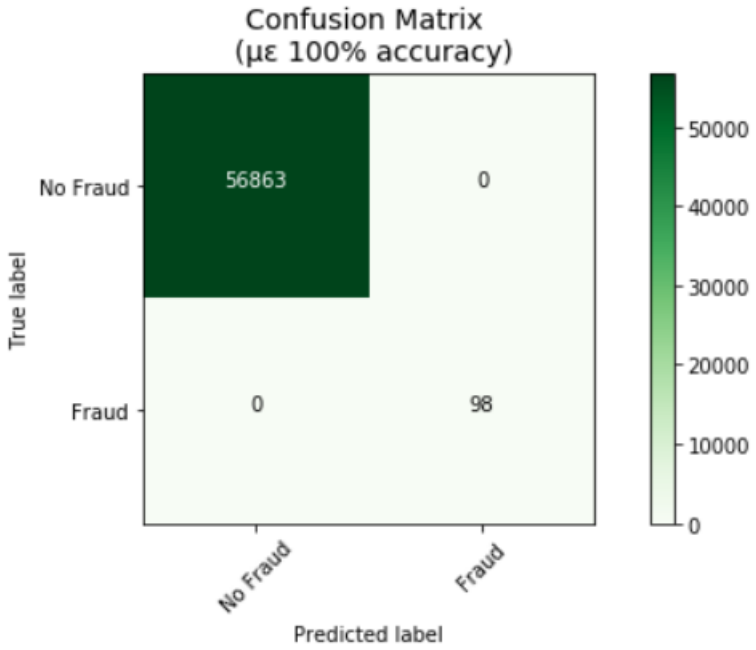
```
   oversample_model.fit(Xsm_train, ysm_train, validation_split=0.2, b
atch_size=300, epochs=20, shuffle=True, verbose=2)
```

Πρόβλεψη

```
   oversample_predictions = oversample_model.predict(original_Xtes
t, batch_size=200, verbose=0)
   oversample_fraud_predictions = oversample_model.predict_classes
(original_Xtest, batch_size=200, verbose=0)
   oversample_smote = confusion_matrix(original_ytest, oversample_
fraud_predictions)
   actual_cm = confusion_matrix(original_ytest, original_ytest)
   labels = ['No Fraud', 'Fraud']

   fig = plt.figure(figsize=(16,8))

   fig.add_subplot(221)
   conf_matrix_plotter(oversample_smote, labels, title="OverSample
(SMOTE) \n Confusion Matrix", cmap=plt.cm.Oranges)

   fig.add_subplot(222)
   conf_matrix_plotter(actual_cm, labels, title="Confusion Matrix
\n (με 100% accuracy)", cmap=plt.cm.Greens)
```



Σχήμα 18: OverSample (SMOTE) Confusion Matrix



Σχήμα 19: Confusion Matrix με 100% ακρίβεια

Τα προηγούμενα διαγράμματα δημιουργήθηκαν με την συνάρτηση ‘conf_matrix_plotter’, ο κώδικας της οποίας δίνεται στην συνέχεια

```
import itertools

# δημιουργία confusion matrix
def conf_matrix_plotter(cm, classes,
 normalize=False,
 title='Confusion matrix',
 cmap=plt.cm.Blues):
 """
 Η συνάρτηση αυτή υπολογίζει και εμφανίζει το confusion matrix.
 Μπορεί να κάνει και κανονικοποίηση με την επιλογή `normalize=T
rue`.
 """
 if normalize:
     cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
     print("Normalized confusion matrix")
 else:
     print('Confusion matrix, without normalization')

 print(cm)

 plt.imshow(cm, interpolation='nearest', cmap=cmap)
 plt.title(title, fontsize=14)
 plt.colorbar()
 tick_marks = np.arange(len(classes))
 plt.xticks(tick_marks, classes, rotation=45)
 plt.yticks(tick_marks, classes)

 fmt = '.2f' if normalize else 'd'
 thresh = cm.max() / 2.
 for i, j in itertools.product(range(cm.shape[0]), range(cm.sha
pe[1])):
 plt.text(j, i, format(cm[i, j], fmt),
 horizontalalignment="center",
 color="white" if cm[i, j] > thresh else "black")

 plt.tight_layout()
 plt.ylabel('True label')
 plt.xlabel('Predicted label')
```

Συμπεράσματα

Συνολικά, το πρόβλημα της Απάτης Πιστωτικών Καρτών είναι αρκετά δύσκολο να επιλυθεί, ωστόσο λόγω των σοβαρών επιπτώσεων που αυτό επιφέρει, η αυτοματοποιημένη ανίχνευση γίνεται όλο και πιο σημαντική. Η επίλυση του δυσχεραίνεται από το γεγονός ότι δεν υπάρχουν διαθέσιμα δεδομένα τα οποία θα μπορούσαν να επιτρέψουν την ανάπτυξη σχετικών αλγορίθμων, καθώς αυτά δεν παρέχονται από τους χρηματοπιστωτικούς οργανισμούς. Τα δεδομένα που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία, προήλθαν από παλιότερο ερευνητικό πρόγραμμα της Ευρωπαϊκής Ένωσης. Δυστυχώς, οι αλγόριθμοι που δημιουργήθηκαν στα πλαίσια της εργασίας δεν μπορούν να χρησιμοποιηθούν ως έχουν σε μελλοντικές εργασίες, καθώς στα αρχικά δεδομένα είχε γίνει ανάλυση PCA για την διατήρηση της ανωνυμίας των συναλλαγών. Παρόλα αυτά, η συνολική λογική πίσω από τον κώδικα που δημιουργήθηκε παραμένει και μπορεί με μικρές προσαρμογές, να χρησιμοποιηθεί και σε άλλα αντίστοιχα σετ δεδομένων.

Τα δεδομένα που χρησιμοποιήθηκαν εδώ, έχουν σημαντικές ανισορροπίες μεταξύ του πλήθους των κλάσεων σε βαθμό που επηρεάζει σημαντικά την διαδικασία προσαρμογής και χρήσης αλγορίθμων μηχανικής εκμάθησης. Πράγματι, η χρήση αλγορίθμων για την «εξισορρόπηση» των δεδομένων, οδήγησε σε σημαντική μεταβολή του πίνακα συσχέτισης χαρακτηριστικών σε σχέση με τα αρχικά δεδομένα (Σχήμα 5). Με βάση το αποτέλεσμα αυτό, έγινε χρήση δύο σχετικών αλγορίθμων, των Random Under-Sampling και του SMOTE με πολύ θετικά αποτελέσματα στην τελική ακρίβεια ταξινόμησης. Σε περίπτωση μη χρησιμοποίησης των παραπάνω, οι αλγόριθμοι ταξινόμησης έδιναν υπερβολικά υψηλά σκορ ταξινόμησης, υποδεικνύοντας ότι γινόταν υπερβολική προσαρμογή τους στα δεδομένα εισόδου (overfitting).

Στα δεδομένα που προέκυψαν από την προηγούμενη διαδικασία δοκιμάστηκαν συνολικά οι ακόλουθοι ταξινομητές:

1. Logistic Regression
2. K-Nearest Neighbors Classifier
3. Support Vector Classifier
4. Decision Tree Classifier

Επιλέχθηκε αρχικά να γίνει δοκιμή στα δεδομένα που προέκυψαν από την τεχνική undersampling λόγω του πολύ μικρότερου πλήθους δεδομένων (και άρα και μικρότερου χρόνου εκτέλεσης). Τα αποτελέσματα αυτών ήταν πολύ ενθαρρυντικά με τα τελικά ROC σκορ να είναι για τους 4 ταξινομητές αντίστοιχα 97.03%, 93.1%,97.3% και 91.87%.

Η επανάληψη της διαδικασίας αυτής θα ήταν ιδιαίτερα χρονοβόρα για τα δεδομένα που προέκυψαν με χρήση του αλγορίθμου SMOTE, καθώς λόγω του τρόπου λειτουργίας αυτού, δημιουργούνται πολλαπλές νέες συνθετικές παρατηρήσεις οι οποίες προστίθενται στις αρχικές με αποτέλεσμα την σημαντική αύξηση μεγέθους του αρχικού dataset. Επιλέχθηκε λοιπόν να γίνει χρήση μόνο του βέλτιστου ταξινομητή από τις προηγούμενες δοκιμές, δηλαδή του Logistic Regression. Συγκρίνοντας τα αποτελέσματα από την χρήση των 2 αλγορίθμων εξισορρόπησης δεδομένων και με χρήση του ίδιου ταξινομητή (Logistic Regression) στα αρχικά δεδομένα test εξάγονται οι ακόλουθες ακρίβειες ταξινόμησης:

Technique	Score
Random UnderSampling	0.947368
Oversampling (SMOTE)	0.988080

Είναι εμφανές ότι η χρήση του αλγορίθμου SMOTE παρέχει καλύτερα αποτελέσματα, με αυξημένο όμως κόστος σαν χρόνο εκτέλεσης.

Τέλος, στα πλαίσια της εργασίας έγιναν και δοκιμές με χρήση νευρωνικού δικτύου. Έγινε μια αρχική προσέγγιση στο πρόβλημα με δημιουργία ενός νευρωνικού δικτύου τα επίπεδα του οποίου ταιριάζουν στα χαρακτηριστικά του dataset και στο αναμενόμενο αποτέλεσμα ήτοι δύο πιθανές κλάσεις ταξινόμησης. Το νευρωνικό δίκτυο που προέκυψε, δοκιμάστηκε στα δεδομένα που δημιουργήθηκαν τόσο από τον αλγόριθμο “Random UnderSampling” όσο και από τον αλγόριθμο “SMOTE”. Από τις δοκιμές αυτές προέκυψαν αποτελέσματα ταξινόμησης όλου του test dataset (σε μορφή επιτυχημένης ή όχι ταξινόμησης) από τα οποία προκύπτουν τα ακόλουθα μετρικά απόδοσης:

	Random Under Sampling	SMOTE
Precision	0.999872	0.99949
Sensitivity	0.959042	0.999736
Specificity	0.928571	0.704082
Accuracy	0.958989	0.999228

Πίνακας 2: Μετρικά Απόδοσης και αποτελέσματα δοκιμών με Random UnderSampling και SMOTE

Μελλοντικά, αυτό που θα μπορούσε να γίνει θα ήταν μια εκτενέστερη μελέτη της συσχέτισης του χρόνου εκτέλεσης της κάθε συναλλαγής με προηγούμενες συναλλαγές, προκειμένου να εξαχθούν πιθανά μοτίβα για τον κάθε κάτοχο. Σύμφωνα με τους Roy et al. [14], κάτι τέτοιο καθίσταται δυνατό με τη χρήση ενός σημαντικά πολυπλοκότερου νευρωνικού δικτύου, καθώς επίσης και με δεδομένα τα οποία θα επέτρεπαν την συσχέτιση κινήσεων με τον κάτοχο της κάρτας.

Βιβλιογραφία

[1] V. P. &. A. D. Tej Paul Bhatla, Understanding Credit, June 2003.

[2] M. Duncan, «The Future Threat of Credit Card Crime,» *RCMP Gazette,* p. 25–26, 1995.

[3] V. Leeuwen, «A Surge in Credit Card Fraud,» *H. Financial Review,* p. 49, 24 September 2002.

[4] Anonymous, «Fraud Prevention Reference Guide,» Certegy, September 2001.

[5] J. T. Q. a. M. Sriganesh, «Real-time credit card fraud detection using computational Intelligence,» *Expert Systems with Applications,* τόμ. 35, αρ. 4, p. 1721–1732, 2008.

[6] C. M. Bishop, Pattern recognition and machine learning, τόμ. 4, New York: Springer, 2006.

[7] N. M. A. D. J. H. C. W. a. D. J. W. Piotr Juszczak, «Off the peg and bespoke classifiers for fraud detection,» τόμ. 52, αρ. 9, p. 4521–4532, 2008.

[8] M. &. M. W. Zaki, Data Mining and Analysis: Funda-mental Concepts and Algorithms, New York City, New York: Cam-bridge University Press, 2014.

[9] L. Breiman, J. Friedman, C. J. Stone και R. Olshen, Classification and Regression Trees (Wadsworth Statistics/Probability), Belmont, 1984.

[10] D. W. Hosmer και S. Lemeshow, Applied Logistic Regression (Wiley Series in Probability and Statistics), N. Jersey: Wiley-Interscience Publication, 2000, p. 373.

[11] S. Ehramikar, «The Enhancement of Credit Card Fraud Detection Systems,» Statistical Learning Theory Wiley, New York, 2000.

[12] V. Vapnik, «Statistical Learning Theory,» New York, Wiley-Interscience, 1998.

[13] S. T. K. V. B. &. M. B. Maes, «Credit Card Fraud Detection Using Bayesian and Neural Networks,» Brussel, Belgium, 2002.

[14] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams και P. Beling, «Deep learning detecting fraud in credit card transactions,» σε *Systems and Information Engineering Design Symposium (SIEDS)*, Charlottesville, VA, USA, 2018.

[15] J. Mayur, Black Cards Forensics: Compilation on Debit and Credit card frauds (Technology and Frauds Book 4), Fraudexpress, 2012.