Εξόρυξη Δεδομένων

Εργασία 2017-2018

Αλεξανδρίδης Γιώργος - 21404

Αλεξίου Μάριος - 21405

Μάρκου Βαγγέλης - 21461

Ευρετήριο

Περιεχόμενα Παραδοτέου

Εκτός από το παρόν κείμενο το παραδοτέο περιέχει:

- **preparation.py** - Ο Κώδικας που χρησιμοποιήσαμε για το Preprocessing.
- **classification.py** - Ο Κώδικας που περιλαμβάνει τις δοκιμές για τους classifiers στα γνωστά δεδομένα καθώς και ο κώδικας για τις τελικές προβλέψεις στα άγνωστα δεδομένα.
- **requirements.txt** - Οι βιβλιοθήκες που είναι απαραίτητες για την Python.
- **prediction_final.csv** - Οι προβλέψεις μας για τα άγνωστα δεδομένα (η μορφή είναι του csv είναι [rowid,class] με την αρίθμηση του rowid να ξεκινάει από το 1).
- **prediction_probabilities_final.csv** - Οι ταξινομημένες προβλέψεις για τις εταιρίες που είναι πιο πιθανόν να πτωχεύσουν (η μορφή είναι του csv είναι [rowid,πιθανότητα πτώχευσης] με την αρίθμηση του rowid να ξεκινάει από το 1).

Πρώτα Βήματα

Διαβάζοντας αναλυτικά την εκφώνηση και την αντίστοιχη βιβλιογραφία προσπαθήσαμε να κατανοήσουμε τι είδους δεδομένα περιλαμβάνει το dataset που μας δόθηκε. Το dataset περιλαμβάνει δεδομένα σχετικά με την οικονομική κατάσταση Πολωνικών εταιριών του Βιομηχανικού τομέα, καθώς αρκετές από αυτές πτώχευσαν από το 2004 και μετά. Επομένως συγκεντρώθηκαν 64 διαφορετικοί χρηματοοικονομικοί δείκτες που υπήρχαν στη βάση δεδομένων EMIS (Emerging Markets Information Service). Οι δείκτες αυτοί ήταν οι κατάλληλοι για την εκτέλεση κατηγοριοποίησης με τη χρήση μιας δικής τους υλοποίησης βασισμένη σε Extreme Gradient Boosting αλγόριθμο. Συνεπώς έπρεπε και εμείς να κάνουμε τις κατάλληλες μετατροπές στο συγκεκριμένο dataset και να το προσαρμόσουμε στις ανάγκες μας.

Προεπεξεργασία (Preprocessing)

Για το κομμάτι του Preprocessing προσπαθήσαμε να κατανοήσουμε ποιες τεχνικές του θα χρειαστούμε ώστε να καταλήξουμε στο όσο ποιο δυνατόν ποιοτικότερο dataset για εκπαίδευση. Φορτώσαμε λοιπόν το αρχείο csv (dataset) που μας δόθηκε στη python, με τη χρήση της βιβλιοθήκης pandas, και ακολουθήσαμε τις παρακάτω διαδικασίες :

Καθαρισμός Δεδομένων (Data Cleaning)

Ελλιπής Τιμές (Missing Values)

Από το dataset παρατηρήσαμε ότι κάποιες τιμές έλλειπαν και στη θέση τους υπήρχε το “?”. Αποφασίσαμε να αντικαταστήσουμε τις ελλιπής τιμές, χρησιμοποιώντας τον Imputer που υπάρχει στο Scikit-Learn. Ο Imputer παίρνει σαν όρισμα τη στρατηγική που θα εφαρμόσει για την αντικατάσταση τους. Εμείς υλοποιήσαμε την αντικατάσταση των ελλιπών τιμών με την χρήση της πιο συνηθισμένης τιμής (most_frequent), του μέσου όρου (mean) και της μέσης τιμής (median).

Παρατηρήσαμε στις δοκιμές μας ότι η χρήση της στρατηγικής αντικατάστασης με τον μέσο όρο (mean) των τιμών κάθε γνωρίσματος είναι πιο αποτελεσματική από τις άλλες δύο, οπότε κατά κύριο λόγο χρησιμοποιούμε αυτήν για το preprocessing. Αρκετά αποδοτική βέβαια ήταν και η χρήση της πιο συνηθισμένης τιμής.

Ακραίες Τιμές (Outliers)

Υπάρχουν ορισμένες τιμές στο dataset οι οποίες είναι ακραίες. Αυτές οι τιμές μπορεί να προκύπτουν από λανθασμένες καταχωρήσεις εγγραφών ή εξαιρέσεις για ορισμένες παραμέτρους. Οι τιμές αυτές καλό θα ήταν να μην υπάρχουν καθώς ορισμένες φορές μπορούν να επηρεάσουν την σωστή κατηγοριοποίηση. Για να εξαλείψουμε αυτές τις τιμές χρησιμοποιήσαμε ορισμένες τεχνικές που υποστηρίζονται από το Scikit-Learn, όπως EllipticEnvelope, IsolationForest και LocalOutlierFactor.

Εν τέλει καταλήξαμε στο IsolationForest καθώς είναι αρκετά αποτελεσματικό σε dataset με αρκετές διαστάσεις όπως το δικό μας. Αρκετά αποτελεσματικός σε πολλά γνωρίσματα είναι επίσης και ο αλγόριθμος του LocalOutlierFactor. Η χρήση EllipticEnvelope αποδίδει καλά σε δεδομένα που ακολουθούν την Gaussian κατανομή.

Μετασχηματισμός Δεδομένων (Data Transformation)

Κανονικοποιήση (Normalization)

Θέλοντας οι αριθμητικές μας τιμές να έχουν μια συνοχή και να περιορίζονται σε ένα μικρότερο διάστημα, όπως το [0,1], προβήκαμε σε μετασχηματισμό των αριθμητικών μας δεδομένων με τεχνικές κανονικοποίησης. Η κανονικοποίηση των αριθμών βοηθάει πολλούς αλγορίθμους κατηγοριοποίησης που χρησιμοποιούν αποστάσεις, όπως ο αλγόριθμος των Κ-Πλησιέστερων γειτόνων.

Για την εκτέλεση της κανονικοποίησης χρησιμοποιήσαμε Scalers του Scikit-Learn. Δοκιμάσαμε τον StandardScaler , MinMaxScaler, MaxAbsScaler και RobustScaler. Η διαδικασία κανονικοποίησης που ακολουθούν ο RobustScaler και StandardScaler είναι συνήθως για δεδομένα που έχουν και ακραίες τιμές. Ωστόσο οι MinMaxScaler και MaxAbsScaler υποφέρουν αν τα δεδομένα έχουν τιμές που απέχουν πολύ από άλλες. Για αυτό το λόγο, αφού ήδη έχουμε αφαιρέσει τις ακραίες τιμές

δοκιμάσαμε αρχικά τον MinMaxScaler. Ο MaxAbsScaler θα ήταν εξίσου αποδοτικός, ωστόσο το εύρος τιμών στο όποιο καταλήγει περιλαμβάνει και αρνητικές τιμές κάτι που δεν υποστηρίζεται από ορισμένους αλγορίθμους μείωσης διαστάσεων όπως θα δούμε παρακάτω.

Εν τέλει παρατηρήσαμε ότι ο StandardScaler απέδιδε ακόμα καλύτερα από τους άλλους δύο και σε περιπτώσεις που δεν μειώναμε τις διαστάσεις τον προτιμούσαμε.

Μείωση Δεδομένων (Data Reduction)

Μείωση Διαστάσεων (Dimensionality Reduction)

Το dataset που μας δόθηκε περιλαμβάνει 64 μεταβλητές όπου η κάθε μια περιγράφει έναν χρηματοοικονομικό δείκτη. Θεωρήσαμε ότι περισσότερες από αυτές δεν είναι σημαντικές για την κατηγοριοποίηση που θα ακολουθήσει. Επομένως θέλοντας να κάνουμε πιο απλό το dataset αναζητήσαμε τρόπους ώστε να μειώσουμε τις διαστάσεις του. Δοκιμάσαμε να χρησιμοποιήσουμε τη μέθοδο επιλογής των K-Καλύτερων features (SelectKBest) χρησιμοποιώντας ως όρισμα κάθε φορά μια διαφορετική τιμή αξιολόγησης. Πιο συγκεκριμένα χρησιμοποιήσαμε τον SelectKBest με chi2 , ANOVA Value και Mutual Information.

Σε πολλούς αλγόριθμους κατηγοριοποίησης η μείωση διαστάσεων επηρέαζε αρνητικά τα Scores για αυτό και σε πολλές δοκιμές μας αποφύγαμε να τη χρησιμοποιήσουμε.

Διακριτοποίηση (Discretization)

Τέλος, θέλοντας να ολοκληρώσουμε το Preprocessing, σκεφτήκαμε να κάνουμε διακριτές τις τιμές μας ώστε να αποφύγουμε τη χρήση αριθμητικών μεταβλητών κατά το Classification. Έπειτα από αναζήτηση καταλήξαμε στο συμπέρασμα ότι οι περισσότεροι Classifiers δέχονται συνεχής τιμές και αναλαμβάνουν αυτοί να μετατρέψουν τα δεδομένα όπως επιθυμούν.

Κατηγοριοποίηση (Classification)

Έχοντας προετοιμάσει τον κώδικα μας έτσι ώστε να μπορεί να προσαρμόζεται εύκολα σε αλλαγές όσον αφορά τα βήματα του Preprocessing, ξεκινήσαμε τις δοκιμές ώστε να εκπαιδεύσουμε το μοντέλο μας. Για την αξιολόγηση του μοντέλου χρησιμοποιήσαμε μετρικές όπως η Πιστότητα (Accuracy), η Ακρίβεια (Precision), η Ανάκληση (Recall) καθώς και το F1 Score (F measure) που προκύπτει από τα δυο τελευταία.

Cross-Validation

Θέλοντας να αποφύγουμε το Overfitting και να καταλήξουμε σε ένα ασφαλές συμπέρασμα όλες οι δοκιμές μας εκτελέστηκαν με Cross-Validation και πιο συγκεκριμένα με τη τεχνική k-fold. Το k-fold ουσιαστικά σπάει το dataset σε k τμήματα και επιλέγει ένα από αυτά για testing και τα υπόλοιπα για training. Αυτό επαναλαμβάνεται k φορές. Επομένως οι μετρικές που θα ακολουθήσουν είναι ουσιαστικά ο μέσος όρος από τις k εκτελέσεις του k-fold Cross Validation.

Κατηγοριοποιητές (Classifiers)

Ακολουθούν οι κατηγοροποιητές που δοκιμάσαμε για το μοντέλο μας. Για κάθε κατηγοριοποιητή αναγράφονται οι δοκιμές μας (τι αλλάξαμε στο Preprocessing καθώς και οι τιμές αξιολόγησης που προκύπτουν από το Cross-Validation) και σχόλια για τον κάθε αλγόριθμο. Σε όλες τις δοκιμές μας χρησιμοποιήσαμε 10-fold Cross Validation. Τα αποτελέσματα είναι στρογγυλοποιημένα σε δύο δεκαδικά ψηφία.

RandomForestClassifier

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score
Impute = most_frequent Outliers = IsolationForest Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.96	0.53	0.05	0.07
Impute = most_frequent Outliers = IsolationForest Dimensionality Reduction = mutual_info	0.96	0.71	0.21	0.34
Impute = most_frequent Outliers = IsolationForest Normalize= MinMaxScaler	0.96	0.53	0.05	0.09
Impute = most_frequent Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.96	0.42	0.06	0.10
No	0.96	0.67	0.17	0.23

Σχόλια

Ο RandomForestClassifier χρησιμοποιεί πολλά decision trees σε υποδείγματα των δεδομένων. Χρησιμοποιήσαμε τη default επιλογή με τους 10 estimators, δηλαδή 10 decision trees. Το precision και

το accuracy ήταν σε καλά επίπεδα με τη χρήση Imputer, Outlier Removal και dimensionality reduction , ωστόσο απογοητευτικό ήταν το recall και ως αποτέλεσμα και το F measure. Δοκιμάσαμε να αλλάξουμε και τις παραμέτρους του classifier όπως τις επιλογές max_depth και min_samples_leaf ώστε να κάνουμε Prunning στα δεντρά χωρίς όμως σημαντικές διαφορές με τα Scores των προηγούμενων εκτελέσεων.

ExtraTreesClassifier

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score
Impute = most_frequent Outliers = IsolationForest Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.95	0.06	0.004	0.007
Impute = most_frequent Outliers = IsolationForest Dimensionality Reduction = mutual_info	0.95	0.09	0.004	0.03
Impute = most_frequent Outliers = IsolationForest Normalize= MinMaxScaler	0.95	0.03	0.012	0
Impute = most_frequent Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.95	0.22	0.01	0.03
No	0.95	0.12	0.01	0.01

Σχόλια

Ο ExtraTreesClassifier ακολουθεί την ίδια σχεδόν λογική με τον RandomForest ωστόσο τα δέντρα που χρησιμοποιεί είναι τυχαία (ExtraTrees). Παρόλο που οι δύο αλγόριθμοι μοιάζουν τα αποτελέσματα ήταν πολύ χειρότερα από τον RandomForest, επομένος ο συγκεκριμένος αλγόριθμος απορρίφθηκε για το δικό μας dataset.

NearestCentroid

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score
Impute = most_frequent	0.53	0.03	0.59	0.06

Outliers = IsolationForest Normalize= MinMaxScaler Dimensionality Reduction = mutual_info				
Impute = most_frequent Outliers = IsolationForest Dimensionality Reduction = mutual_info	0.86	0.07	0.26	0.11
Impute = most_frequent Outliers = IsolationForest Normalize= MinMaxScaler	0.59	0.06	0.70	0.11
Impute = most_frequent Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.78	0.09	0.32	0.12
No	0.24	0.04	0.78	0.08

Σχόλια

Στον NearestCentroid κάθε κλάση (στη περίπτωση μας οι εταιρίες που είναι bankrupt ή non-bankrupt) αντιπροσωπεύονται από ένα Centroid. Επομένως τα δεδομένα δοκιμής (test data) κατηγοριοποιούνται με βάση την απόσταση τους από το Centroid. Για metric χρησιμοποιήσαμε την ευκλείδεια απόσταση που είναι και η προεπιλογή. Ο αλγόριθμος δεν είναι καθόλου αποδοτικός για το δικό μας dataset καθώς και οι δυο κλάσεις μας δεν μπορούν να διακριθούν από αποστάσεις. Αυτό φαίνεται και στα αποτελέσματα που έβγαλε στις δοκιμές μας. Όλα τα Scores ήταν αρκετά χαμηλά για αυτό και απορρίψαμε και αυτόν τον αλγόριθμο κατηγοριοποίησης.

SVMs

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score	Kernel
Impute = most_frequent Outliers = IsolationForest Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.57	0.04	0.72	0.09	rbf
Impute = most_frequent Outliers = IsolationForest Dimensionality Reduction = mutual_info	0.95	0	0	0	rbf
Impute = most_frequent	0.56	0.06	0.72	0.11	rbf

Outliers = IsolationForest Normalize= MinMaxScaler					
Impute = most_frequent Normalize= MinMaxScaler Dimensionality Reduction = mutual_info	0.96	0	0	0	rbf
No	0.95	0	0	0	rbf

Σχόλια

Δοκιμάσαμε τριών ειδών Support Vector Machines Classifiers: SVC, nuSVC και linearSVC. Τα αποτελέσματα ήταν πολύ απογοητευτικά και αποφασίσαμε να μην ασχοληθούμε παραπάνω με την συγκεκριμένη κατηγορία αλγορίθμων. Τα παραπάνω αποτελέσματα προκύπτουν από την εκτέλεση του SVC αλγορίθμου.

Decision Tree Classifier

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score	AUROC
Impute = mean	0.97	0.66	0.51	0.57	0.78
Impute = mean Outliers = IsolationForest	0.98	0.76	0.51	0.60	0.79
Impute = mean Outliers = IsolationForest Normalize= Standard	0.98	0.75	0.55	0.63	0.82

Σχόλια

Ο DecisionTreeClassifier χτίζει ένα μόνο δέντρο απόφασης, που έχει ως αποτέλεσμα ο χρόνος εκπαίδευσης στα δεδομένα να είναι μικρότερος από τους περισσότερους classifiers. Χρησιμοποιήσαμε την **εντροπία** ως κριτήριο για την ποιότητα διαχωρισμού (αντί του προεπιλεγμένου μέτρου gini) αφού αποδίδει καλύτερα στο συγκεκριμένο dataset. Θέσαμε την παράμετρο **min_samples_split** σε 40 με σκοπό να αποφύγουμε την υπερεκπαίδευση του μοντέλου. Από τα αποτελέσματα που δίνονται παραπάνω παρατηρούμε ότι ο αλγόριθμος αποδίδει καλύτερα με την αντικατάσταση των ελλιπών τιμών, την αφαίρεση outliers και την κανονικοποίηση των γνωρισμάτων.

Gradient Boosting Classifier

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score	AUROC
Impute = mean	0.98	0.87	0.58	0.69	0.95
Impute = mean Outliers = IsolationForest	0.98	0.89	0.59	0.70	0.94
Impute = mean Outliers = IsolationForest Normalize= StandardScaler	0.98	0.91	0.58	0.71	0.95

Σχόλια

Ο αλγόριθμος GradientBoosting ανήκει στην ίδια κατηγορία με τους RandomForest και ExtraTrees (Ensemble). Αποτελείται από πολλά decision trees, τα οποία όμως δεν είναι ανεξάρτητα μεταξύ τους (όπως στην περίπτωση του RandomForest). Δημιουργείται ένα προσθετικό μοντέλο στο οποίο σε κάθε βήμα το τρέχον δέντρο προσπαθεί να μειώσει την loss function του προηγούμενου. Από δοκιμές παρατηρήσαμε ότι ο αλγόριθμος αποδίδει καλύτερα με την **deviance loss function**, με **μεγάλο αριθμό (1000) δέντρων** και **μικρό (0.1) learning rate,** είναι αρκετά ανθεκτικός σε υπερεκπαίδευση από πολλούς estimators. Αποδίδει καλύτερα και με την χρήση αδύναμων δέντρων (**max_depth=3**) λόγω του μικρού level of interaction μεταξύ των μεταβλητών. Όπως και στις προηγούμενες περιπτώσεις, η απόδοση αυξάνεται με την αντικατάσταση ελλιπών τιμών, την αφαίρεση outliers και την κανονικοποίηση.

Adaptive Boosting Classifier

Δόκιμες

Preprocessing	Accuracy	Precision	Recall	F1 Score
Impute = mean Estimators = 500	0.97	0.72	0.47	0.56
Impute = mean Outliers = IsolationForest	0.98	0.75	0.49	0.59

Estimators = 500				
Impute = most_frequent Outliers = IsolationForest Normalize= StandardScaler	0.97	0.73	0.49	0.58

Σχόλια

Ο αλγόριθμος που χρησιμοποιήθηκε είναι ο AdaBoostClassifier. Μετά από δοκιμές, καταλήξαμε ότι ο καλύτερος συνδυασμός ήταν με mean imputation, removing the outliers with isolation forest algorithm και χρησιμοποιωντας 500 estimators. Ο συγκεκριμένος αλγόριθμος είναι, όπως αναφέρεται, ένας ensemble meta-estimator, ο όποιος κάνει fit έναν classifier (στη συγκεκριμένη περίπτωση τον DecisionTreeClassifier) σε όλο το dataset. Έπειτα, αφού βλέπει τα αποτελέσματα, ξανά κάνει fit στο dataset με τον ίδιο classifier, μόνο που αυτή τη φορά μοιράζει βάρη στα αντίστοιχα φύλλα του κάθε δέντρου για να πετύχει καλύτερα αποτελέσματα. Στο δικό μας dataset, όπως φαίνεται και από το παραπάνω πίνακα τα αποτελέσματα είναι ελαφρώς καλά και σε καμία περίπτωση δεν είναι ο ιδανικός αλγόριθμος για να χειριστεί το imbalanced dataset που διαθέτουμε.

Bagging Classifier

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score
Impute = mean Estimators = 10	0.98	0.87	0.50	0.61
Impute = mean Outliers = IsolationForest Estimators = 10	0.98	0.88	0.51	0.63
Impute = mean Estimators=50	0.98	0.90	0.50	0.63
Impute=mean Estimators=50 Outliers=IsolationForest	0.98	0.87	0.53	0.66

Impute=mean Estimators=50 Outliers=IsolationForest Normalize=StandardScaler	0.98	0.88	0.51	0.64

Σχόλια

Ο αλγόριθμος που χρησιμοποιήθηκε σε αυτή τη δοκιμή είναι ο BaggingClassifier. Μέτα από το παραπάνω πίνακα δοκιμών, καταλήξαμε πως η καλύτερη δοκιμή είναι mean imputation, removing outliers with Isolation Forest algorithm και 50 estimators. Ο BaggingClassifier είναι επίσης ένας ensemble meta-estimator, ο οποίος χρησιμοποιεί έναν base estimator (DecisionTreeClassifier) και τρέχει αυτόν σε τυχαία υποσύνολα του αρχικού συνόλου του dataset. Έπειτα, μέσω voting or averaging επιλέγει τα καλύτερα υποσύνολα για να εκπροσωπήσουν ολόκληρο το dataset. Βέβαια αυτή η λογική αν δεν χειριστεί σωστά μπορεί να οδηγήσει σε overfitting. Τα αποτελέσματα ήταν ικανοποιητικά, αλλά υπάρχουν άλλοι αλγόριθμοι με ακόμα καλύτερα results.

Extreme Gradient Boosting Classifier

Δοκιμές

Preprocessing	Accuracy	Precision	Recall	F1 Score
Impute = mean Estimators = 100	0.98	0.96	0.52	0.66
Impute = mean Outliers = IsolationForest Estimators = 100	0.98	0.98	0.54	0.69
Impute = mean Estimators=200	0.98	0.95	0.54	0.68
Impute=mean Estimators=200 Outliers=IsolationForest	0.98	0.98	0.55	0.70
Impute=mean Estimators=500 Outliers=IsolationForest	0.98	0.94	0.59	0.72
Impute=mean Estimators=500	0.98	0.93	0.58	0.71

Outliers=IsolationForest Normalize=StandardScaler				
Impute=mean Estimators=1000	0.98	0.86	0.64	0.72
Impute=mean Estimators=1000 Outliers=IsolationForest Evaluation Metric=auc	0.98	0.89	0.64	0.74
Impute=mean Estimators=1000 Outliers=IsolationForest Evaluation Metric=auc Normalize=StandardScaler	0.98	0.89	0.65	0.75

Σχόλια

Ο αλγόριθμος που χρησιμοποιήσαμε εδώ είναι ο Extreme Gradient Boosting Classifier. Έπειτα από αρκετές δοκιμές, που αναφέρονται στο παραπάνω πίνακα, καταλήξαμε στη καλύτερη δυνατή λύση αλγορίθμου για το συγκεκριμένο dataset. Ο XGBClassifier, όπως ονομάζεται το function, παίρνει αρκετές και σημαντικές παραμέτρους. Λόγω του imbalanced dataset που μας δόθηκε, με ratio ~= 1:25, έχουμε λίγα δεδομένα για τις bankrupt εταιρίες. Για να γίνει καλύτερο scaling των δεδομένων χρειάστηκε να δοθεί μεγαλύτερο βάρος στα bankrupt δεδομένα, υπολογισμένη με μαθηματική φόρμουλα (sum(negative cases) / sum(positive cases)). Κάνοντας πειράματα, καταλήξαμε ότι ο XGBClassifier, είναι ο πιο ιδανικός αλγόριθμος όσον αφορά το classification. Με χαρακτηριστικά, όπως mean imputation, removing outliers with Isolation Forest algorithm and normalize data with StandardScaler, είχαμε τα καλύτερα F1-scores & AUC scores. Ένας, επίσης, σημαντικός παράγοντας βελτίωσης του classification στο συγκεκριμένο αλγόριθμο, είναι η αλλαγή του evaluation metric από logloss σε auc. Αυτό το κάναμε, λόγω του imbalanced dataset, διότι μετά από έρευνα συμπεράναμε ότι χρησιμοποιώντας τέτοιου είδους metric έχουμε καλύτερα αποτελέσματα.

Επιλογή του καλύτερου Classifier

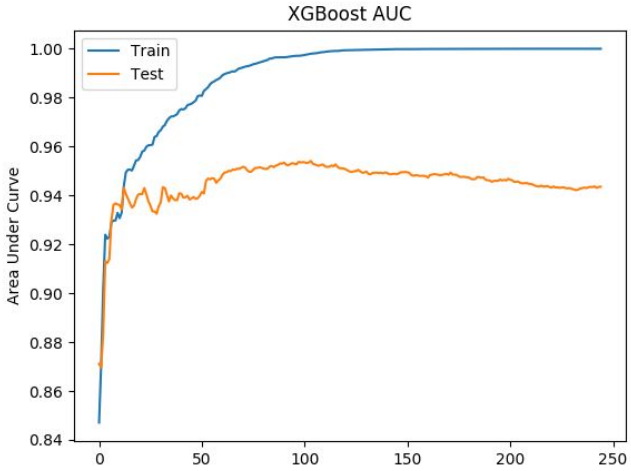
Extreme Gradient Boosting Classifier

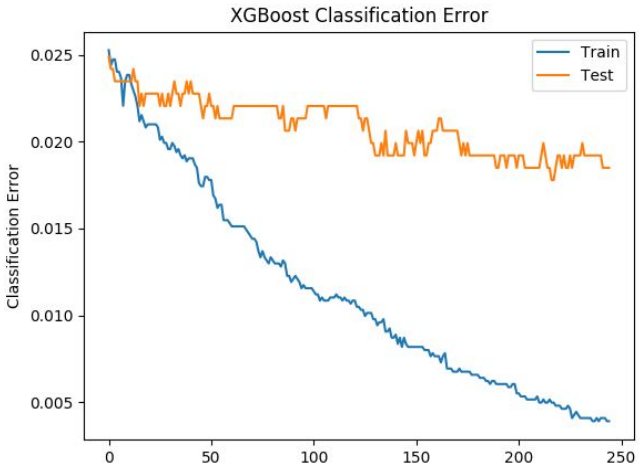
Έπειτα από τις δοκιμές στα άγνωστα δεδομένα, παρατηρήθηκε ότι το αρχικά επιλεγμένο μοντέλο του EXGB με τις συγκεκριμένες, παραπάνω παραμέτρους δεν ανταποκρίθηκε όπως αναμέναμε. Έτσι, χρειάστηκε να τροποποιηθούν οι παράμετροι που χρησιμοποιήθηκαν

προηγουμένως. Για τη καλύτερη επιλογή παραμέτρων με βάση το roc auc score, χρησιμοποιήθηκε η μέθοδος GridSearch. Η μέθοδος αυτή δοκιμάζει εξαντλητικά όλο το εύρος τιμών των παραμέτρων που δίνονται, για όλους τους συνδυασμούς παραμέτρων (χρησιμοποιώντας και cross validation). Επιστρέφει ως αποτέλεσμα την βέλτιστη τιμή για κάθε παράμετρο. Μέσω αυτού, καταλήξαμε στις ακόλουθες τιμές :

Parameters	**Values**
learning_rate	0.1
n_estimators	1000
max_depth	4
min_child_weight	6
gamma	0
subsample	0.8
colsample_bytree	0.8
reg_alpha	0.005
objective	'binary:logistic'
scale_pos_weight	1
seed	26

Με τις παραπάνω νέες τιμές το f1 score μειώθηκε σε σχέση με το 0,75 όπου ήταν το καλύτερο που είχε βρεθεί. Βέβαια, στα άγνωστα δεδομένα, θεωρητικά πηγαίνει πολύ καλύτερα λόγω της γενίκευσης του μοντέλου κυρίως στο scale_pos_weight όπου αποδείχτηκε λανθασμένη η προσπάθεια για δημιουργία balanced dataset μέσω των weights. Επιπλέον, χρησιμοποιήθηκαν plots για να αναλύσουν το error rate & roc auc score, ανάμεσα στα train & test data με σκοπό να βρεθεί ο καλύτερος αριθμός estimators ή να χρησιμοποιηθεί το early_stopping_rounds όπως και έγινε. Μετά τα δυο παρακάτω plots, το early_stopping_rounds πήρε τη τιμή 142 ως βέλτιστη για τον EXGB.





Αλλάζοντας, βέβαια, και τις παραμέτρους φαίνεται ότι και ο Gradient Boosting αλγόριθμος πήγε αρκετά πιο κάλα σε σχέση με τον EXGB στα άγνωστα δεδομένα (στα train και οι δυο αλγόριθμοι είχαν παρόμοια f1-scores χωρίς ιδιαίτερες διαφορές). Επαληθεύσαμε ότι ο GB είναι καλύτερος, μέσω του predict probability και διαπιστώσαμε ότι ο GB βρίσκει τις bankrupt εταιρείες με μεγαλύτερο ποσοστό σιγουριάς σε σχέση με τον EXGB.

Gradient Boosting Classifier

Για την βελτιστοποίηση των τιμών των παραμέτρων του Gradient Boosting Classifier χρησιμοποιήσαμε και εδώ την μέθοδο GridSearch. Στην περίπτωση του δικού μας dataset οι βέλτιστες τιμές είναι:

Parameters	**Values**
learning_rate	0.1
min_samples_split	50
min_samples_leaf	25
max_depth	4
max_features	15
subsample	0.9
n_estimators	140

Με αυτές τις παραμέτρους φαίνεται πως τα scores του μοντέλου μειώνονται στο training dataset (f1~=0.66), άλλα το μοντέλο γίνεται πιο γενικευμένο για την χρήση του στο άγνωστο dataset. Στην αναζήτηση των πιθανότερων προς χρεωκοπία εταιριών διαπιστώσαμε, όπως αναφέρεται και προηγουμένως, ότι ο συγκεκριμένος estimator έχει μεγαλύτερα ποσοστά σιγουριάς για τις bankrupt εταιρίες (~0.98 στην 50η σε σειρά εταιρία) σε σχέση με τον EXGB (~0.95 στην 50η σε σειρά εταιρία).

Πρόβλεψη χρεωκοπίας και μείωση διαστάσεων

Πρόβλεψη

Για την κατάταξη των επιχειρήσεων σε σχέση με τον κίνδυνο να χρεοκοπήσουν χρησιμοποιήσαμε την μέθοδο πρόβλεψης της πιθανότητας του estimator να ανήκει η κάθε επιχείρηση στην κλάση bankrupt (predict_proba(x)), ταξινομώντας τα αποτελέσματα σε φθίνουσα σειρά. Για την πρόβλεψη χρησιμοποιήσαμε τον καλύτερο μας αλγόριθμο για τα άγνωστα δεδομένα, τον Gradient Boosting.

Μείωση Διαστάσεων

Μας ζητήθηκε να βρούμε υποσύνολο με τα καλύτερα γνωρίσματα και να χρησιμοποιήσουμε αυτές με τον πιο αποδοτικό μας Classifier. Τέτοιου είδους διαδικασίες είχαμε δοκιμάσει ήδη στο preprocessing που κάναμε στα γνωστά μας δεδομένα. Ο τρόπος με τον οποίο κάναμε την μείωση των διαστάσεων έχει προαναφερθεί στο κεφάλαιο της προεπεξεργασίας. Φάνηκε ξεκάθαρα και από τα scores στους Classifiers που χρησιμοποιήσαμε ότι η μείωση των γνωρισμάτων δεν βοήθησε καθόλου και δημιουργούσε μεγάλη πτώση στα scores. Επομένως αποφασίσαμε να μην τη χρησιμοποιήσουμε. Πολύ πιθανόν αυτή η πτώση να οφείλεται στο γεγονός ότι τα γνωρίσματα, επειδή αποτελούν χρηματοοικονομικούς δείκτες, έχουν μεγάλες συσχετίσεις μεταξύ τους (με τη μορφή κλασμάτων ή αριθμητικών παραστάσεων).

Στην περίπτωση του GB τα 10 πιο σημαντικά features σύμφωνα με τον αλγόριθμο SelectKBest με στρατηγική mutual_info_classif (η οποία αποδείχθηκε η αποδοτικότερη) είναι:
['X11' 'X12' 'X16' 'X23' 'X24' 'X26' 'X27' 'X41' 'X42' 'X55']

Τα αποτελέσματα που πήραμε με την επιλογή των 10 πιο σημαντικών features είναι τα εξής:

Accuracy	Precision	Recall	F1 Score	ROC AUC
0.97	0.87	0.51	0.64	0.90

Η απόδοση του μοντέλου όμως αυξάνεται με την χρήση όλων των γνωρισμάτων. Τα αποτελέσματα φαίνονται παρακάτω:

Accuracy	Precision	Recall	F1 Score	ROC AUC
0.98	0.93	0.52	0.66	0.94

Τέλος Αναφοράς