Saint Petersburg State University

Faculty of Mathematics and Mechanics

Department of Informatics

Report on practical training on the topic: “A framework for combining Machine Learning with Data-balancing methods for credit scoring”

Student: Mohammadhossei Khalashi

Group: 21.Б14-мм

Scientific Advisor: Ph.D. Dmitry Grigoriev

Introduction

Forecasting the abilities of customers to fulfill their financial obligations is a critical issue in banking activities. To address this problem, financial services companies try to assess the probability of defaulting through credit scoring process, aimed at classifying the applicants into categories corresponding to good and bad credit quality, according to their capability to meet financial obligations.

Credit scoring is defined as statistical model aimed to determine the creditworthiness of customers, I.e. to estimate the probability of defaulting[1] .

Since costumers with bad creditworthiness have high probability of defaulting, the accuracy of credit scoring system is critical to financial institutions profitability such that even a one percent improvement in the accuracy of credit scoring of "bad" customers may significantly decrease the losses of a financial institutions[2] .Also From the economic point of view, inaccurate estimates of creditworthiness in the banking sector were one of the key determinant of the two worst economic crises of modern times (i.e., the Great Depression of 1929 and the Great Recession of 2008)[3] [4] . The latter in particular was triggered by the so-called subprime mortgage crisis, where underestimation of default probabilities and easy credit conditions had catastrophic economic consequences.

In general, there are two approaches to design automatic credit scoring systems; I.e., statistical techniques and AI techniques[5] . Several statistical techniques are applied to design automatic credit scoring systems, However, a common weakness of most statistical approaches is that some assumptions must be made, such as assuming specific data distributions[6] . On the other hand, many studies have shown that AI methods such as artificial neural networks are effective tools for credit scoring and unlike statistical approaches, AI techniques can be used to design credit scoring systems without assuming specific data distributions[7] .

Additional issues when dealing with credit scoring systems are "dimensionality" and "imbalanced dataset". To address dimensionality often feature selection techniques are used, which aimed to extract the most relevant features from the feature space, resulted in overfitting decrease and accuracy improvement as well as reducing training costs. And to address the problem of imbalanced dataset, various sample-modifying techniques are used, which usually help prevent the model from becoming biased towards more representative classes.

Since there is no best overall AI technique, and it depends on the details of problem[8] , this report attempts to answer following question:

> Which combinations of ML models, feature selection and sample-modifying techniques are the most effective for credit scoring system?

Methods

ML models

Random forest

The random forest classifier consists of a combination of tree classifiers where each classifier is generated using a random vector sampled independently from the input vector, and each tree casts a unit vote for the most popular class to classify an input vector[9] . The random forest classifier consists of randomly selected features or a combination of features at each node to grow a tree. Bagging, a method to generate a training data set by randomly drawing with replacement *N* examples, where *N* is the size of the original training set[10] , was used for each feature combination selection. Examples are classified by taking the most popular voted class from all the tree predictors in the forest[11] . There are many approaches to the selection of attributes used for decision tree induction and most approaches assign a quality measure directly to the attribute. The most frequently used attribute selection measures in decision tree induction are Information Gain Ratio criterion and Gini Index[12] . The random forest classifier uses the Gini Index as an attribute selection measure, which measures the impurity of an attribute with respect to the classes. For a given training set T, selecting one case at random and saying that it belongs to some class C_i , the Gini index can be written as:

$$\sum \sum_{j \neq i} \frac{(f(C_i,T)}{|T|} \frac{(f(C_j,T)}{|T|}$$

where $\frac{f\,(C_i,T)}{|T|}$ is the probability that the selected case belongs to class C_i .

Neural Network

Neural networks take as input features $x_1, \ldots, x_d$ and construct a nonlinear function $f(x)$ aimed at predicting the dependent variable y. The peculiarity of the method is the procedure followed to obtain $f(x)$. The most common type of neural network consists of three layers of units: the input, hidden, and output layers. Such a structure is usually called a multilayer perceptron. A layer of “input” units is fed to a layer of “hidden” units, which is finally connected to a layer of “output” units[13] .

Gradient boosting

Gradient boosting is a machine learning technique for regression and classification problems that construct a set of prediction models in the form of an ensemble of weak learners, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function[14] .

This technique leverages the idea of boosting, wherein multiple weak learners (models that are only slightly better than random guessing) are combined to create a strong learner. Gradient boosting takes this concept further by utilizing the gradient of the loss function to guide the construction of the subsequent models.

Generic gradient boosting at the $m-th$ step would fit a decision tree $h\_m(x)$ to pseudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m}, \ldots, R_{J_m m}$ and predicts a constant value in each region. Using the indicator notation, the output of $h_m(x)$ for input x can be written as the sum:

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x)$$

Where b_{jm} is the value predicted in the region R_{jm} .

Then the coefficients b_{jm} are multiplied by some value γ_m , chosen using line search so as to minimize the loss function, and the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \quad \gamma_m = arg\,min_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

A modification is proposed to modify this algorithm so that it chooses a separate optimal value γ_{jm} for each of the tree's regions, instead of a single γ_m for the whole tree. This modified algorithm is called Tree Boost. The coefficients b_{jm} from the tree-fitting procedure can be then simply discarded and the model update rule becomes[14] [15] :

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} b_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = arg\,min_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$$

Logistic regression

the basic setup of logistic regression is as follows: A dataset is given containing N points. Each point i consists of a set of m input variables $x_{1,i},\ldots, x_{m,i}$ (also called independent variables), and a binary outcome variable y_i (also known as a dependent variable), i.e. it can assume only the two possible values 0 (often meaning "no" or "failure") or 1 (often meaning "yes" or "success"). The goal of logistic regression is to use the dataset to create a predictive model of the outcome variable.

As in linear regression, the outcome variables y_i are assumed to depend on the explanatory variables $x_{1,i},\ldots, x_{m,i}$ [16] .

Heterogeneous ensemble

A heterogeneous ensemble is an ensemble learning technique that leverages multiple different base learning algorithms, which are combined to improve the overall predictive performance. The heterogeneity of the base learners increases the diversity and robustness of the ensemble, leading to improved generalization capabilities. Common heterogeneous ensemble techniques include voting ensembles, where the final prediction is based on a majority or weighted vote; stacking ensembles, where a meta-learner is trained to make the final prediction based on the output of the base learners; and cascading ensembles, where models are added sequentially, and each new model makes predictions considering the outputs of the previous models [17] . The heterogeneity in the base learners exposes the ensemble to a broader range of structural patterns and thus, makes it more versatile and better able to model complex data relationships.

Heterogeneous ensemble methods primarily include Voting Ensembles, Stacking Ensembles, and Cascading Ensembles:

- Voting Ensembles: This type of ensemble combines the predictions from multiple models and makes the final prediction based on the majority vote or weighted vote.
- Stacking Ensembles: This involves training a learning algorithm to combine the predictions of several other learning algorithms. It introduces a meta-learner that uses the predictions of the base learners as input and makes the final prediction.
- Cascading Ensembles: Also known as cascaded generalization, it works by adding new models to the ensemble sequentially. Each new model takes the predictions of all the previous models into account when making predictions.

Feature selection techniques

L1 based feature selection with support vector machine

In this study support vector machines are used with linear kernels. The prediction obtained had the general form $pred(x) = sign(b + \sum_{i=1}^{} \alpha i K(x, xi))$. If the kernel was linear (i.e., $K(x, v) = x^T v$), then the prediction became $sign(b + wTx)$ for $w = (w1, \ldots, wd)^T$ $=(\alpha_1 x_1, \ldots, \alpha_d x_d)^T$, where w is a vector of weights that can be computed explicitly. This technique classifies a new observation (y^*, x^*) by testing whether the linear combination $w_1 x^*_1 + \cdots + w_d x^*_d$ of the components of x^* is larger or smaller than a given threshold b[18] . Hence, in this approach, the jth feature is more likely to be important if its weight w_j is above the threshold. This type of feature weighting has some intuitive interpretation, because a predictor with a small $|w_j|$ value has a minor impact on the predictions and can be ignored[19] .

Random forest recursive feature elimination

Recursive feature elimination (RFE) is a greedy algorithm based on feature-ranking techniques. The algorithm measures the classifier performance by eliminating predictors in an iterative manner. In a first step, RFE trains the classifier with all d features, and then it calculates the importance of each feature via the Information Gain method or the mean reduction in the

Gini index[20] [21] . Subsequently, subsets of progressively smaller sizes $m = d, d-1, \ldots, 1$ are created by iterative elimination of the features. The model is retrained within each subset, and its performance is calculated. Hence, RFRFE is a feature selection method that combines RFE and random forests[23] .

Sample-modifying techniques

Random oversampling

One of the common approaches to address the problem of imbalanced datasets is to use resampling techniques to make the dataset balanced. Resampling techniques can be applied either by under-sampling or oversampling the dataset. Under-sampling is the process of decreasing the amount of majority target instances or samples. Oversampling can be performed by increasing the amount of minority class instances[23] [24] .

The basic idea consists of resampling the original dataset, either by oversampling the smallest class or under-sampling the largest class until the sizes of the classes are approximately the same. Since under-sampling may discard some important information and consequently worsen the performance of the classifiers, oversampling tends to be preferred[25] .

Random oversampling is one of the simplest methods, as it increases the minority class through randomly repeated copies of the minority class. A possible disadvantage is that if the dataset is large, it may introduce a significant additional computational burden. Moreover, since it yields exact copies of the minority class, it can increase the risk of overfitting[26] .

Synthetic minority oversampling technique (SMOTE)

The synthetic minority oversampling technique (SMOTE) oversamples the minority class by synthetically creating new instances rather than oversampling with replacement, as random oversampling does. The SMOTE forms new minority examples by interpolating between several minority class observations that are close to each other[27] .

More particularly, the minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen.

Synthetic minority oversampling technique with Tomek links (SMOTE-Tomek)

SMOTE-Tomek performs SMOTE oversampling technique along with Tomek Links under-sampling technique. In Tomek Links technique, the majority class is under-sampled by randomly removing majority class observations until the minority class reaches some specified percentage of the majority class. In more detail, the Tomek Links discard observations from the most represented class that are close to the least represented class in order to obtain a training dataset with a more clear-cut separation between the two classes[28] .

Synthetic minority oversampling technique with one sided selection (SMOTE-OSS)

Similar to SMOTE-Tomek, SMOT-EOSS performs SMOTE oversampling technique along with One-Sided Selection (OSS) under-sampling technique. OSS technique combines Tomek Links and the Condensed Nearest Neighbor (CNN) Rule[29] . Specifically, Tomek Links are ambiguous points on the class boundary and are identified and removed in the majority class. The CNN method is then used to remove redundant examples from the majority class that are far from the decision boundary[30] .

In more details, first CNN procedure occurs in one-step and involves first adding all minority class examples to the store and some number of majority class examples (e.g., 1), then classifying all remaining majority class examples with K nearest neighbors (*k=1*) and adding those that are misclassified to the store.

Synthetic minority oversampling technique with edited nearest neighbor (SMOTE-ENN)

SMOTE-ENN is a hybrid data sampling approach used to address class imbalance in datasets. It consists of two main components: Synthetic Minority Over-sampling Technique (SMOTE) and Edited Nearest Neighbors (ENN).

ENN, is an under-sampling technique used to clean up the over-sampled dataset. It removes any instance of the majority class that has a different class label than two of its three nearest neighbors. This procedure makes the decision boundaries in the feature space less noisy, leading to an improved predictive performance of classifiers.

SMOTE-ENN aimed at mitigating the issues associated with class imbalance in datasets. It combines SMOTE, which generates synthetic instances of the minority class, with ENN, which eliminates instances from the majority class that may introduce noise. This approach creates a more balanced dataset by both adding diversity to the minority class and enhancing the decision boundaries by removing potentially misleading instances from the majority class [31] .

Criteria

The performance of the models was evaluated based on the standard measures in the fields of credit scoring. These measures are as follows:

- Area Under the Curve of the Receiver Operating Characteristics (AUC-ROC) [32]
- Accuracy
- Sensitivity
- Specificity
- F1 score [33]

Where these measures are described as follows:

ROC is a probability curve and AUC represents the degree or measure of separability. It shows how much the model is capable of distinguishing between classes.

Consider the following figure (figure 1) known as confusion matrix, we define Accuracy, Sensitivity (Recall), Specificity and Precision as specified below:

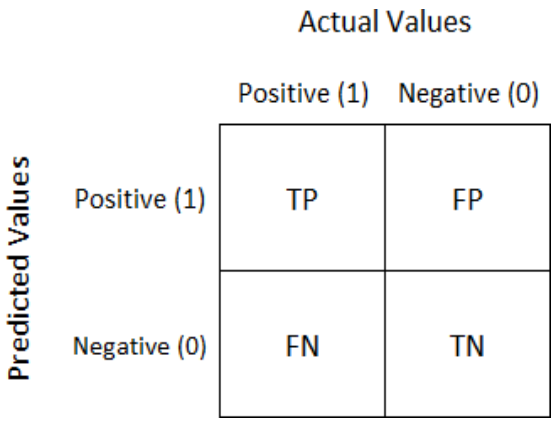


Figure 1, confusion matrix

$$Accuracy = \frac{\sum TP+TN}{\sum TP+FP+FN+TN}$$

$$Sensitivity = Recall = \frac{\sum TP}{\sum TP+FN}$$

$$Specificity = \frac{\sum TN}{\sum TN+FP}$$

$$Precision = \frac{\sum TP}{\sum TP+FP}$$

The F1 score is computed as the harmonic mean of precision and recall:

$$F1 = \frac{2\times Precision\times Recall}{Precision+Recall}$$

Data

For this study two data sets are used[34] [35] . Both of them consists of information about customers of financial institutions. In the following table description of the data sets is shown:

Data set	Number of examples	Numbers of features	Imbalance ration
UCI Credit Card	30000	24	3.50
Default	10000	4	29.02

Where imbalance ratio is the ratio of the number of examples in majority class to the number of examples in minority class.

In the analysis "UCI Credit Card" dataset is considered as the main dataset and after conducting the experiment on it, analysis on "Default" dataset is conducted, in order to show the accordance of results of the main dataset. Notice that "Default" dataset is n an extremely imbalanced dataset, therefore analysis on it can show the effect of balancing techniques on the performance.

As the table above is shown the main data set consists of 30000 examples, with 24 features, which consist of financial and non-financial (age, education, gender, …), with one target and imbalance ratio of 3.5.

The main data set is split into testing set and training set, where testing set consist of 30% of the main data set.

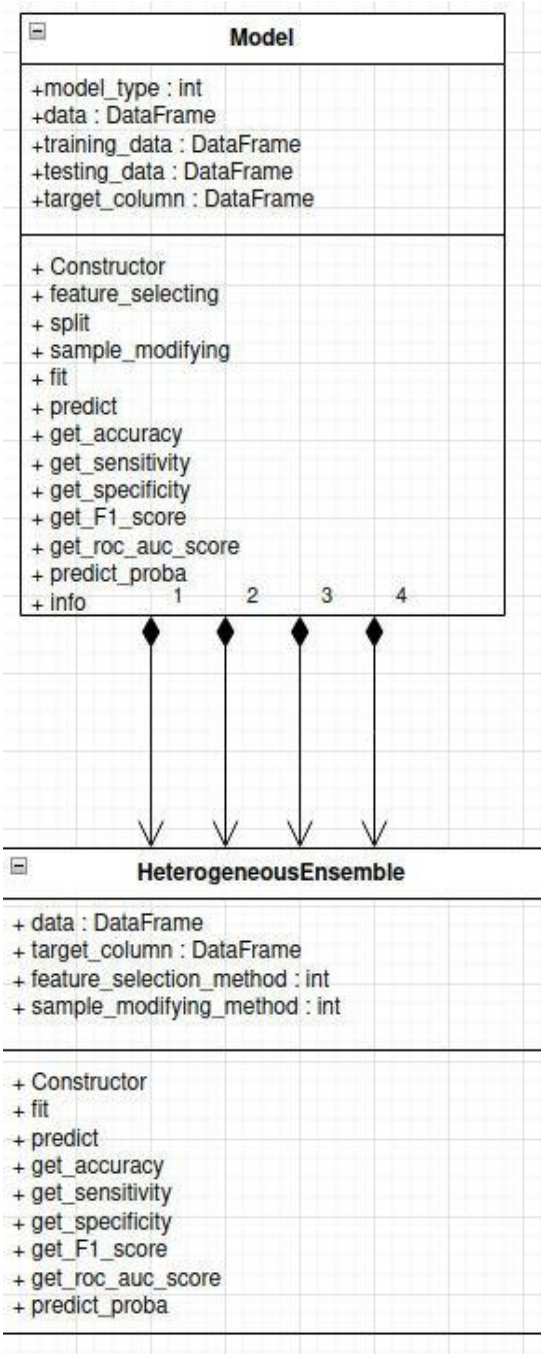
Framework implementation

The provided Python classes[1] constitute a framework designed to address classification tasks in the context of imbalanced datasets. The main goal of the framework is to enhance the prediction performance of credit scoring systems. The framework consists of two main classes, namely Model and HeterogeneousEnsemble. The Model class encapsulates the process of data preprocessing, model selection, training, and evaluation. It accepts a parameter model_type that allows users to choose between different machine learning algorithms: RandomForest, Neural Network, XGBoost, Logistic Regression, and LGBM. Feature selection methods are implemented in the feature_selecting function, with two options, LinearSVC-based feature selection and RandomForest-based feature selection. The split_data function is used for splitting the data into training and testing subsets. The class also includes the sample_modifying method that implements various resampling techniques to mitigate the impact of class imbalance in the data. These techniques include random oversampling, Synthetic Minority Over-sampling Technique (SMOTE), SMOTE combined with Tomek links, SMOTE combined with One-Sided Selection, and SMOTE combined with Edited Nearest Neighbours. The models are then fitted and evaluated using various performance metrics such as accuracy, sensitivity, specificity, F1 score, and ROC AUC score. The HeterogeneousEnsemble class builds upon the Model class to create a heterogeneous ensemble of models. It uses three models – XGBoost, Logistic Regression, and RandomForest – with each of them undergoing the same feature selection and sample modifying process. The StackingClassifier from scikit-learn library is then used to create a stacked ensemble, with Logistic Regression serving as the final, or meta-, classifier. This ensemble approach leverages the complementary strengths of different models, potentially leading to improved overall performance, particularly in handling the imbalanced credit scoring dataset. In conclusion, the presented Python classes provide a flexible, robust, and comprehensive framework for tackling classification problems in imbalanced datasets. The ability to select among multiple models, feature selection methods, and resampling techniques offers considerable versatility, while the adoption of a heterogeneous ensemble strategy can

[1]the implementation is available in [36]

potentially enhance predictive performance. The focus on credit scoring applications, moreover, attests to the framework's utility in a vital area of financial analytics.

Here is the UML diagram of this framework:



Experiments

Methodology

To analyze the data set, first the data set is split into training and testing sets, then considering the testing set, first feature selection method (L1 based feature selection with support vector machine) is used to extract the relevant features, then all data balancing techniques are used respectively (oversampling, SMOTE, SMOTE-Tomek, SMOTE-OSS, SMOTE-ENN) to balance the testing set. Thereafter classifier algorithms (Random Forest, Neural network, Gradient boosting, Logistic regression, Heterogeneous ensemble) are trained on them. The classifiers are also trained on imbalanced training set. After all the performance of classifiers trained on different data sets are measured using the criteria (Accuracy, Sensitivity, Specificity, AUC-ROC, F1 score) and then their performance is compared to find best combination of data balancing techniques and classifier algorithms. Then all the steps are repeated for the second feature selection method (Random Forest feature elimination) and compare the results.

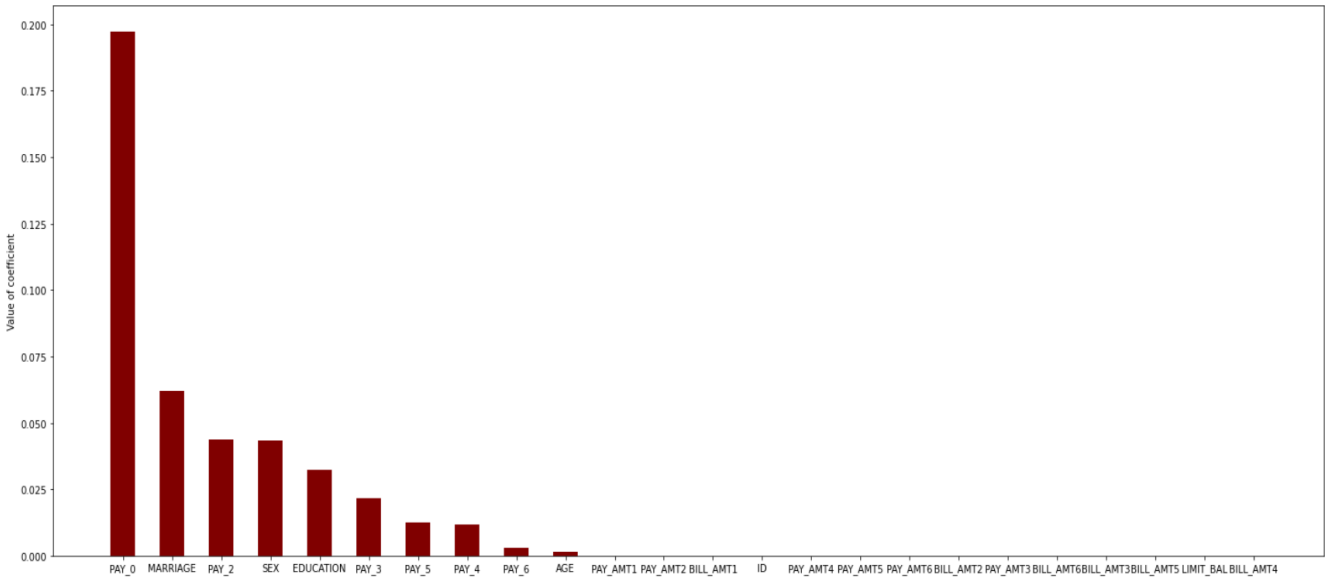
It is worth to mention that a neural network consisting of one hidden layer is used with sigmoid function as activation function for hidden layer and input layer and linear function as activation function for output layer. Also, XGBoost and LightGBM implementations are used for Gradient boosting. Stacking Ensembles is used for Heterogeneous ensemble and in implementation of Heterogeneous ensemble, Random Forest, Logistic regression and XGBoost are used as based models.

Results

Before considering the result of experiments, first let’s overview the features of the main data set:

```
---  ------
 0   ID
 1   LIMIT_BAL
 2   SEX
 3   EDUCATION
 4   MARRIAGE
 5   AGE
 6   PAY_0
 7   PAY_2
 8   PAY_3
 9   PAY_4
 10  PAY_5
 11  PAY_6
 12  BILL_AMT1
 13  BILL_AMT2
 14  BILL_AMT3
 15  BILL_AMT4
 16  BILL_AMT5
 17  BILL_AMT6
 18  PAY_AMT1
 19  PAY_AMT2
 20  PAY_AMT3
 21  PAY_AMT4
 22  PAY_AMT5
 23  PAY_AMT6
```

First the L1 based feature elimination with support vector machine feature selection is run, and 13 numbers of features are eliminated. The following figure shows the features based on their coefficients (or importance):



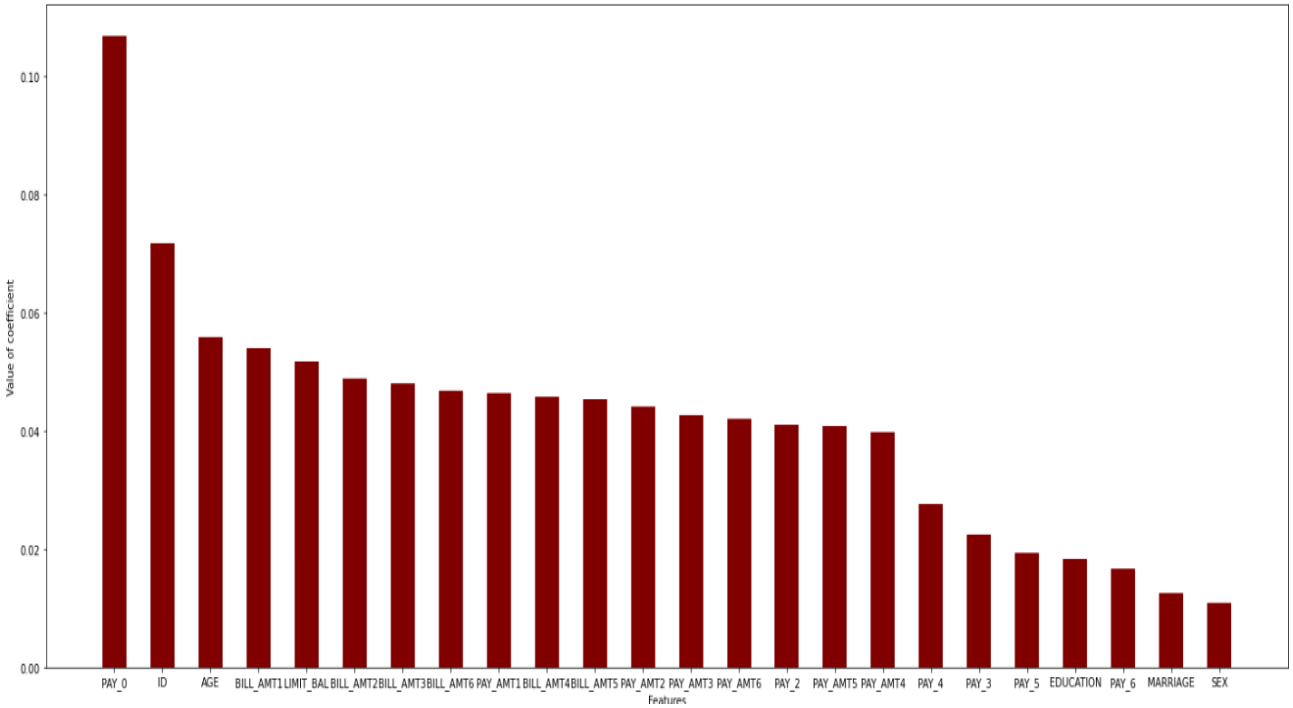
As it is shown above, the feature selection procedure eliminates less important features (features with lower coefficient).

Conducting experiment on the main data set, after performing above-mentioned feature selection technique, the followings results are obtained:

Models	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**
RF-imbalanced	0.784	0.509	0.906	0.627
RF-Random-Oversampling	0.730	0.394	0.812	0.691
RF-SMOTE	0.768	0.462	0.879	0.625
RF-SMOTETomek	0.769	0.464	0.881	0.624
RF-SMOTEOss	0.769	0.463	0.881	0.624
RF-SMOTEENN	0.790	0.400	0.876	0.636
NN-imbalanced	0.816	0.380	0.939	0.747
NN-Random-Oversampling	0.763	0.592	0.811	0.749
NN-SMOTE	0.765	0.585	0.815	0.742
NN-SMOTETomek	0.779	0.563	0.839	0.740
NN-SMOTEOss	0.763	0.580	0.814	0.740
NN-SMOTEENN	0.801	0.537	0.832	0.743
XGB-imbalanced	0.813	**0.632**	0.943	0.647
XGB-Random-Oversampling	0.768	0.473	0.827	0.691
XGB-SMOTE	0.813	0.623	0.937	0.655
XGB-SMOTETomek	0.810	0.608	0.932	0.654
XGB-SMOTEOss	0.811	0.612	0.935	0.654
XGB-SMOTENN	0.809	0.603	0.889	0.652
LGBM-imbalanced	0.799	0.628	0.931	0.632

LGBM -Random-Oversampling	0.709	0.466	0.796	0.678
LGBM -SMOTE	0.806	0.603	0.945	0.657
LGBM -SMOTETomek	0.806	0.608	0.924	0.656
LGBM -SMOTEOss	0.807	0.612	0.924	0.656
LGBM -SMOTENN	0.793	0.598	0.898	0.672
LR-Random-Oversampling	0.736	0.422	0.787	0.671
LR-SMOTE	0.740	0.428	0.791	0.675
LR-SMOTETomek	0.736	0.422	0.786	0.672
LR-SMOTEOss	0.744	0.433	0.797	0.672
LR-SMOTEENN	0.742	0.487	0.751	0.637
HTE-imbalanced	**0.837**	0.401	0.941	0.756
HTE-Random-Oversampling	0.798	0.603	**0.949**	**0.781**
HTE-SMOTE	0.716	0.597	0.819	0.746
HTE-SMOTETomek	0.778	0.578	0.843	0.742
HTE-SMOTEOss	0.778	0.582	0.859	0.746
HTE-SMOTEENN	0.799	0.578	0.867	0.745

Second Random Forest Recursive Feature Elimination selection is run and 11 number of features are eliminated. The following figure shows the features based on their coefficients (or importance):



As it is shown above, the feature selection procedure eliminates less important features (features with lower coefficient).

Conducting experiment on the main data set, after performing above-mentioned feature selection technique, the followings results are obtained:

Models	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**
RF-imbalanced	0.813	0.632	0.943	0.647
RF-Random-Oversampling	0.807	0.581	0.915	0.685
RF-SMOTE	0.806	0.580	0.917	0.663
RF-SMOTETomek	0.809	0.582	0.916	0.667
RF-SMOTEOss	0.781	0.593	0.922	0.667
RF-SMOTEENN	0.760	0.589	0.904	0.659
NN-imbalanced	0.563	0.001	0.987	0.508
NN-Random-Oversampling	0.551	**0.636**	0.543	0.624

NN-SMOTE	0.540	0.629	0.530	0.613
NN-SMOTETomek	0.575	0.632	0.514	0.579
NN-SMOTEOss	0.807	0.596	0.568	0.579
NN-SMOTEENN	0.767	0.628	0.523	0.574
XGB-imbalanced	0.763	0.608	0.939	0.638
XGB-Random-Oversampling	0.805	0.464	0.823	**0.685**
XGB-SMOTE	0.807	0.582	0.924	0.651
XGB-SMOTETomek	0.809	0.595	0.930	0.647
XGB-SMOTEOss	0.781	0.606	0.933	0.647
XGB-SMOTEENN	0.783	0.601	0.34	0.651
LGBM -imbalanced	0.759	0.602	0.931	0.628
LGBM -Random-Oversampling	0.799	0.454	0.813	0.661
LGBM -SMOTE	0.8	0.576	0.914	0.634
LGBM -SMOTETomek	0.811	0.557	0.917	0.651
LGBM -SMOTEOss	0.786	0.601	0.924	0.653
LGBM -SMOTEENN	0.780	0.6	0.343	0.649
LR-imbalanced	0.560	0.000	**0.993**	0.500
LR-Random-Oversampling	0.606	0.290	0.523	0.609
LR-SMOTE	0.606	0.300	0.609	0.604
LR-SMOTETomek	0.560	0.291	0.521	0.611
LR-SMOTEOss	0.593	0.300	0.577	0.611
LR-SMOTEENN	0.594	0.299	0.580	0.608
HTE-imbalanced	0.789	0.616	0.956	0.701
HTE-Random-Oversampling	**0.821**	0.632	0.937	**0.735**

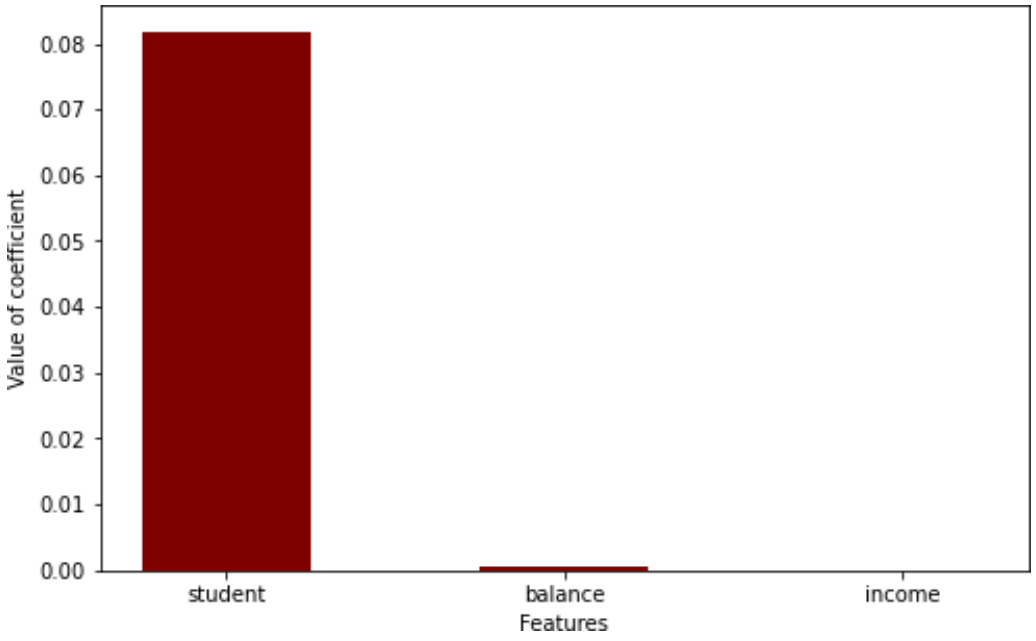
HTE-SMOTE	0.804	0.560	0.939	0.699
HTE-SMOTETomek	0.82	0.578	0.933	0.704
HTE-SMOTEOss	0.799	0.627	0.945	0.708
HTE-SMOTEENN	0.795	0.609	0.898	0.696

Now considering the second data set, lets first overview the features:

```
0   default
1   student
2   balance
3   income
```

Now let's repeat the experiment for the second data set:

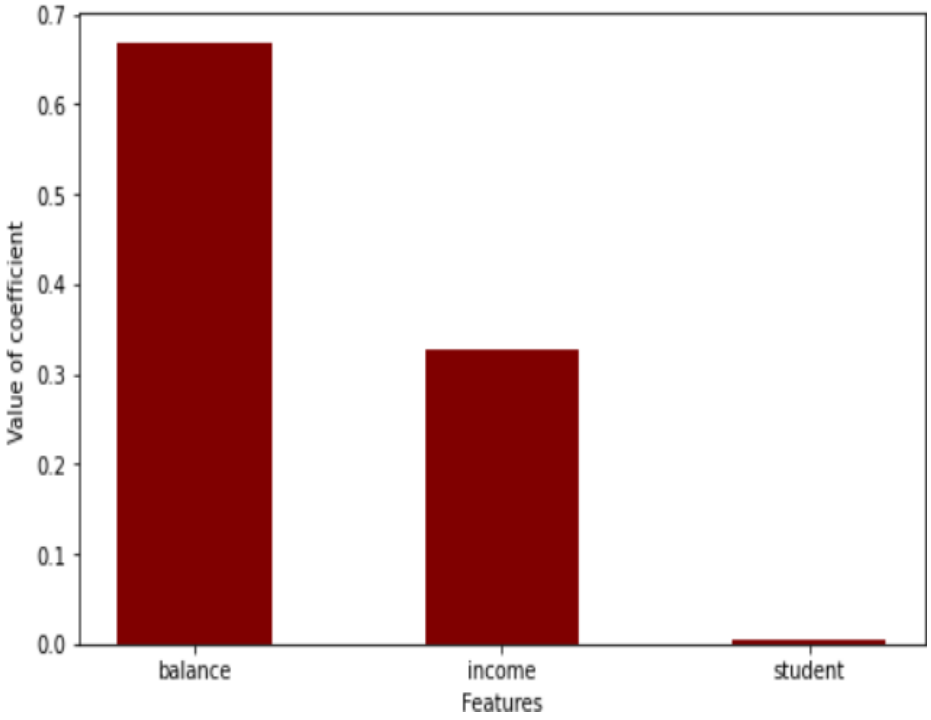
First the L1 based feature elimination with support vector machine feature selection is run, and 2 numbers of features are eliminated.



Models	Accuracy	Sensitivity	Specificity	AUC
RF-imbalanced	0.956	0.302	0.977	0.643
RF-Random-Oversampling	0.956	0.302	0.977	0.643
RF-SMOTE	0.892	0.146	0.905	0.703
RF-SMOTETomek	0.896	0.161	0.907	0.730
RF-SMOTEOss	0.912	0.171	0.926	0.730
RF-SMOTEENN	0.902	0.174	0.909	0.728
NN-imbalanced	0.969	0.000	0.911	0.947
NN-Random-Oversampling	0.692	**0.979**	0.683	0.946
NN-SMOTE	0.756	0.968	0.749	**0.948**
NN-SMOTETomek	0.693	**0.979**	0.683	0.947
NN-SMOTEOss	0.897	0.809	0.900	0.947
NN-SMOTEOENN	0.717	0.876	0.876	0.943
XGB-imbalanced	0.971	0.564	0.992	0.661
XGB-Random-Oversampling	0.936	0.259	0.948	0.756
XGB-SMOTE	0.917	0.202	0.929	0.741
XGB-SMOTETomek	0.914	0.208	0.924	0.770
XGB-SMOTEOss	0.926	0.233	0.937	0.770
XGB-SMOTEENN	0.923	0.243	0.904	0.765
LGBM-imbalanced	0.907	0.578	0.986	0.665
LGBM-Random-Oversampling	0.923	0.261	0.951	0.761
LGBM-SMOTE	0.919	0.218	0.908	0.740
LGBM-SMOTETomek	0.919	0.221	0.929	0.773
LGBM-SMOTEOss	0.933	0.203	0.941	0.774

LGBM-SMOTEENN	0.916	0.254	0.921	0.778
LR-imbalanced	0.974	0.743	**0.997**	0.637
LR-Random-Oversampling	0.867	0.179	0.866	0.885
LR-SMOTE	0.870	0.183	0.869	0.887
LR-SMOTETomek	0.869	0.182	0.868	0.886
LR-SMOTEOss	0.883	0.194	0.884	0.886
LR-SMOTEENN	0.879	0.197	0.865	0.881
HTE-imbalanced	**0.98**	0.007	0.945	0.939
HTE-Random-Oversampling	0.790	0.971	0.698	0.941
HTE-SMOTE	0.756	0.968	0.734	0.947
HTE-SMOTETomek	0.693	0.979	0.690	0.947
HTE-SMOTEOss	0.899	0.811	0.907	0.948
HTE-SMOTEENN	0.719	0.878	0.879	0.944

Second Random Forest Recursive Feature Elimination selection is run and 3 numbers of features are eliminated.



Models	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**
RF-imbalanced	0.955	0.294	0.975	0.647
RF-Random-Oversampling	0.955	0.294	0.975	0.647
RF-SMOTE	0.842	0.132	0.846	0.785
RF-SMOTETomek	0.858	0.142	0.863	0.783
RF-SMOTEOss	0.873	0.150	0.881	0.783
RF-SMOTEENN	0.877	0.159	0.879	0.78
NN-imbalanced	0.969	0.000	**0.991**	0.500
NN-Random-Oversampling	0.761	**0.926**	0.756	**0.944**
NN-SMOTE	0.872	0.862	0.873	**0.944**
NN-SMOTETomek	0.844	0.904	0.842	**0.944**
NN-SMOTEOss	0.872	0.872	0.872	**0.944**

NN-SMOTEENN	0.876	0.873	0.865	0.943
XGB-imbalanced	0.973	0.633	0.994	0.662
XGB-Random-Oversampling	0.936	0.263	0.948	0.761
XGB-SMOTE	0.871	0.164	0.874	0.820
XGB-SMOTETomek	0.870	0.164	0.874	0.820
XGB-SMOTEOss	0.892	0.187	0.897	0.820
XGB-SMOTEENN	0.89	0.179	0.89	0.817
LGBM-imbalanced	0.971	0.63	0.991	0.659
LGBM-Random-Oversampling	0.923	0.256	0.942	0.756
LGBM-SMOTE	0.865	0.161	0.87	0.818
LGBM-SMOTETomek	0.865	0.161	0.872	0.818
LGBM-SMOTEOss	0.888	0.183	0.889	0.817
LGBM-SMOTEENN	0.887	0.18	0.89	0.813
LR-imbalanced	0.975	0.737	0.997	0.647
LR-Random-Oversampling	0.862	0.169	0.862	0.867
LR-SMOTE	0.864	0.172	0.864	0.868
LR-SMOTETomek	0.859	0.168	0.859	0.871
LR-SMOTEOss	0.887	0.194	0.889	0.871
LR-SMOTEENN	0.889	0.19	0.881	0.867
HTE-imbalanced	**0.981**	0.34	0.965	0.621
HTE -Random-Oversampling	0.761	0.921	0.761	**0.944**
HTE -SMOTE	0.879	0.871	0.821	0.941
HTE -SMOTETomek	0.842	0.904	0.839	0.942
HTE -SMOTEOss	0.87	0.852	0.867	0.939
HTE -SMOTEENN	0.869	0.871	0.863	0.942

Discussion and conclusion

Given the experimental findings, it becomes evident that the application of Random Forest or Neural Network classifiers, in conjunction with Random Forest Recursive Feature Elimination and random oversampling, can deliver superior performance compared to other models. Yet, coupling a Neural Network classifier with random oversampling can similarly yield commendable performance, even when an L1-based feature elimination approach with a support vector machine is utilized. As anticipated, the efficiency of the Logistic Regression algorithm diminishes as data set skewness increases. In the experiment conducted on a more imbalanced data set, it was noted that the application of data balancing techniques enhances the model's performance. This is especially pertinent for Logistic Regression, which is significantly impacted by data set imbalance. The experimental findings further illustrate that a heterogeneous ensemble, comprising base models such as logistic regression, XGBoost, and Random Forest along with random oversampling, emerges as the most proficient model among all others. Although the Random Forest and Neural Network models also demonstrate substantial performance, the heterogeneous ensemble prevails as the superior choice.

List of references

[1] West, David. "Neural network credit scoring models." *Computers & operations research* 27.11-12 (2000): 1131-1152.

[2] Hand, David J., and William E. Henley. 1997. Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society: Series A 160: 523–41.

[3] Temin, Peter. "Notes on the Causes of the Great Depression." *The great depression revisited*. Springer, Dordrecht, 1981. 108-124.

[4] Stiglitz, Joseph E. "Interpreting the Causes of the Great Recession of 2008." *Financial system and macroeconomic resilience: revisited* 53.1 (2010): 297.

[5] Huang, Zan, Hsinchun Chen, Chia-Jung Hsu, Wun-Hwa Chen, and Soushan Wu. 2004. Credit rating analysis with support vector machines and neural networks: A market comparative study. Decision Support Systems 37: 543–58.

[6] Huang, Zan, Hsinchun Chen, Chia-Jung Hsu, Wun-Hwa Chen, and Soushan Wu. 2004. Credit rating analysis with support vector machines and neural networks: A market comparative study. Decision Support Systems 37: 543–58.

[7] Van Gestel, Tony, and Bart Baesens. 2009. Credit Risk Management. Basic Concepts: Financial Risk Components, Rating Analysis, Models, Economic and Regulatory Capital. Oxford: Oxford University Press.

[8] Hand, David J., and William E. Henley. 1997. Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society: Series A 160: 523–41.

[9] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

[10] Breiman, L. "Bagging predictors Machine Learning 24 (2), 123-140 (1996) 10.1023." *A: 1018054314350* (1996).

[11] Breiman, Leo, and Ross Ihaka. *Nonlinear discriminant analysis via scaling and ACE*. Davis One Shields Avenue Davis, CA, USA: Department of Statistics, University of California, 1984.

[12] Haykin, Simon S. Neural Networks: A Comprehensive Foundation, 2nd ed. Upper Saddle River: Prentice Hall PTR. He, Haibo, and Edwardo A. Garcia. 2009. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21: 1263–84

[13] Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (1999). "Boosting Algorithms as Gradient Descent" . In S.A. Solla and T.K. Leen and K. Müller (ed.). *Advances in Neural Information Processing Systems 12*. MIT Press. pp. 512–518.

[14] Friedman, J. H. (February 1999). "Greedy Function Approximation: A Gradient Boosting Machine"

[15] Hosmer, David W.; Lemeshow, Stanley (2000). Applied Logistic Regression (2nd ed.)

[16] James, Gareth, Daniela Witten, Trevor Hastie, and Rob Tibshirani. 2021. An Introduction to Statistical Learning, 2nd ed. New York: Springer.

[17] Zhou, Z.-H. (2012). Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC.

[18] Brankl, Janez, M. Grobelnikl, N. Mili´c-Frayling, and D. Mladeni´c. 2002. Feature selectionusing support vector machines. In Data Mining III. Edited by A. Zanasi, C. Brebbia, N. Ebecken and P. Melli. Southampton: WIT Press.

[19] Sindhwani, Vikas, Pushpak Bhattacharya, and Subrata Rakshit. 2001. Information theoretic feature crediting in multiclass support vector machines. In Proceedings of the 2001 SIAM International Conference on Data Mining. Philadelphia: SIAM, pp. 1–18.

[20] Zhou, Qifeng, Hao Zhou, Qingqing Zhou, Fan Yang, and Linkai Luo. 2014. Structure damage detection based on random forest recursive feature elimination. Mechanical Systems and Signal Processing 46: 82–90.

[21] James, Gareth, Daniela Witten, Trevor Hastie, and Rob Tibshirani. 2021. An Introduction to Statistical Learning, 2nd ed. New York: Springer

[22] Ustebay, Serpil, Zeynep Turgut, and Muhammed Ali Aydin. 2018. Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. Paper presented at the 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), Ankara, Turkey, December 3–4. pp. 71–76.

[23] Y. Sun, A. K. Wong, and M. S. Kamel, “Classification of imbalanced data: A review,” International Journal of Pattern Recognition and Artificial Intelligence, vol. 23, no. 04, pp. 687–719, 2009.

[24] H. He and E. A. Garcia, “Learning from imbalanced data,” IEEE Transactions on knowledge and data engineering, vol. 21, no. 9, pp. 1263–1284, 2009

[25] Ganganwar, Vaishali. 2012. An overview of classification algorithms for imbalanced datasets. International Journal of Emerging Technology and Advanced Engineering 2: 42–47.

[26] Moreo, Alejandro, Andrea Esuli, and Fabrizio Sebastiani. "Distributional random oversampling for imbalanced text classification." *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2016.

[27] Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." *Journal of artificial intelligence research* 16 (2002): 321-357.

[28] Tomek, Ivan. 1976. Two modifications of cnn. IEEE Transactions on Systems, Man, and Cybernetics 11: 769–72

[29] Hart, Peter. "The condensed nearest neighbor rule (corresp.)." *IEEE transactions on information theory* 14.3 (1968): 515-516.

[30] Kubat, Miroslav, and Stan Matwin. "Addressing the curse of imbalanced training sets: one-sided selection." *Icml*. Vol. 97. No. 1. 1997.

[31] Batista, Gustavo E. A. P. A., Prati, Ronaldo C., & Monard, Maria Carolina. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. SIGKDD Explorations. 6. 20-29. 10.1145/1007730.1007735.

[32] James, Gareth, Daniela Witten, Trevor Hastie, and Rob Tibshirani. 2021. An Introduction to Statistical Learning, 2nd ed. New York: Springer

[33] Sokolova, Marina, and Guy Lapalme. "A systematic analysis of performance measures for classification tasks." Information Processing & Management 45.4 (2009): 427-437.

[34] https://www.kaggle.com/code/meenavyas/ucicreditcard/notebook

[35] https://www.picostat.com/dataset/r-dataset-package-islr-default

[36] https://colab.research.google.com/drive/1-m2wNMXIVedwIO1Xjab5YJSXbPVE4RCf?usp=sharing