Comparative analysis of ML algorithms using Credit Scoring Data.

MSc Project: Big Data Science

Queen Mary University of London

August 2018

Muhammad Omar Waqar

m.waqar@se17.qmul.ac.uk

Disclaimer

This report, with any accompanying documentation and/or implementation, is submitted as part requirement for the degree of MSc Big Data Science at the University of London.

It is the product of my own labour except where indicated in the text.

1. Acknowledgements

First, I would like to thank my thesis advisor Professor Steve Uhlig of the School of Electronic Engineering and Computer Science at Queen Mary University of London. He has allowed this paper to be my own work but steered me in the right the direction whenever he thought I needed it.

I would also like to thank the Kaggle community in general. The level of expertise, guidance and help that can be found on that forum is unparalleled. I have benefited immensely from the discussions and tutorials which helped me implement my code.

I would also like to acknowledge Home Credit for sharing their data which has made it possible for me to do this project.

I would also like to acknowledge Professor Pasquale Malacaria of the School of Electronic Engineering and Computer Science at Queen Mary University as the second examiner of this thesis, and I am gratefully indebted for his time and efforts.

Finally, I must express my very profound gratitude to my parents and to my spouse, for providing me with unfailing support and continuous encouragement throughout my year of study and through the process of researching and this project. This accomplishment would not have been possible without them. Thank you.

M O Waqar

2. Abstract

The use of Machine Learning (ML) algorithms for consumer credit scoring and classification is a popular topic. However, there is a constant debate on which algorithm is most suitable for the task. Continued advancement in the field of ML and the ever-increasing size of available credit scoring data is constantly evolving the landscape. Studies are either suggesting novel approaches or using more and more data to compare existing algorithms. Amidst all this, little attention is being given to carry an in-depth analysis of credit scoring classification algorithms to understand their capabilities w.r.t data in hand. In this project, I have attempted to shed light on this area. First, I conduct an exploratory analysis of the data to understand its characteristics. Then I prepare different versions from that dataset to highlight some those characteristics. Afterwards, the performance of each algorithm on all versions of the data are observed and analysed. Interesting insights found during the project are related the sensitivity of the three algorithms studied to the size, sparsity, dimensionality, class imbalance and pre-processing steps performed on credit scoring dataset. Logistic Regression (LR) works best with numerical features scaled and categorical features encoded with weight of evidence encoding, it also trains sufficiently well on lesser amount of data compared to other two algorithms, but it is most sensitive to class imbalance. Linear Support Vector Classifier (LinearSVC) performs best when numerical features are scaled, and categorical features One-Hot-Encoded, meaning sparse and increased data dimensions are not an issue, it also is least affected by class imbalance. LightGBM is the most robust of the algorithms but missing values and data imbalance do affect its performance. Emphasis of this project has been on comparing the performance of ML algorithms (Logistic Regression, Linear SVC, LightGBM), on credit scoring data provided by Home Credit company but the scope can be broadened by including more datasets and classification algorithms. This study opens new avenues of analysis that need to be explored as we advance further in the application of machine learning in credit scoring domain.

Keywords: Classification for credit scoring, LightGBM, Logistic Regression, Linear Support Vector Classifier

3. Contents

4. Introduction

Advancement in information technology has driven a change in all walks of life. Its impact can also be seen in consumer credit business. Technological innovations, availability of huge amounts of data, and population growth are continuously changing the consumer credit landscape. In the UK, consumer credit stood at GBP 213.2 billion, as of 30 Jun 18 (Bank of England, 2018). This has increased almost 37% in just 5 years (Bank of England, 2018). Monthly growth rate has gone as high as 10.9 percent. The figures above exclude any student loans, which would make the stats go even higher. Similar statistics can be observed in other countries as well.

It shows that consumer credit is a huge market where the deciding factor for a credit lending company's success is to provide swift loans to its new or existing clients with minimum hassle.

It is imperative for these businesses to analyse loan application thoroughly, and accurately predict the client's behaviour with respect to loan repayment. Statistics, machine learning and domain expertise are the essential ingredients of the evaluation process. They are used to derive a metric known as credit score that plays a massive role in a consumer's life. It estimates a person's creditworthiness based on their credit history and correlation of their financial behaviour to credit risk (Lessmann, et al., 2015).

The concept of numeric credit scoring emerged in 1989 and used Logistic Regression at its core (Desjardins, 2018). Logistic Regression is still the most widely used algorithm in industry but newer technologies in big data science and artificial intelligence are constantly evolving the credit landscape (Hand & Henley, 1997). Models are now developed not only to help make an informed decision on granting loans and its terms (credit scoring), but also to predict whether the client would default or not (probability to default) (Lessmann, et al.,

2015). The later model is the focus of my project. It can be a binary classifier differentiating bad loans from the good ones, or a probability showing the confidence of a decision.

Using machine learning algorithms for credit scoring is a popular topic in research. Existing studies can be classified into two types, ones suggesting new and improved classification algorithms e.g. (Sullivan, et al., 2017), (Zhu, et al., 2018) and the others comparing existing algorithms with each other e.g. (Baesens, et al., 2003).

One of the most detailed study is of (Baesens, et al., 2003). This study was updated by (Lessmann, et al., 2015) to incorporate recent advancements in the field of machine learning. It focuses on using individual algorithms and ensembles on different datasets to compare their performance using a pre-determined evaluation metric. It compares 41 individual algorithms and ensembles on 8 different datasets for benchmarking.

Albeit all this research not much in-depth analysis is done to identify which characteristics of a dataset improve an algorithms performance or what traits of an algorithm make it suitable for a given dataset (Lessmann, et al., 2015). The lack of conclusive study showing one algorithm to be irrefutably better than others, shows that there is a research gap that has not been addressed so far.

With this project, I intend to answer how the algorithms perform when features in the dataset are transformed and why this change occurs. It would help deducing the suitability of an algorithm for the data at hand. The project is limited to studying only three algorithms, Logistic Regression (LR), Support Vector Machine using Linear kernel (LinearSVC) and a relatively new algorithm, LightGBM (LGBM).

I have chosen consumer credit scoring dataset because it poses a few certain challenges that are not present in the dataset from other domains. Consumer credit scoring data is designed to capture a range of properties. It attempts to cover aspects of consumer demographics, their payment patterns and financial history. This results in a dataset that has diverse features that might or might not be applicable to every single case. In other words, the dataset features are generalized to document consumer information coming from a variety of backgrounds. The dataset used in this project is provided by Home Credit[1] company for a competition on Kaggle [2] in which they want to see if Kaggle community can come up with a model better than the one they currently use. The goal of the competition is to classify loans into good and bad ones.

In this project, an analysis is conducted by comparing the results of the mentioned algorithms on different versions of the same dataset. These versions differ by varying approaches to handling missing values and the categorical features and using different subset of features from the original data. The variations in data are designed to highlight how the models respond to certain aspects of dataset. The aim is to identify how algorithm's performance is affected in comparison to the results observed on all the versions of dataset. The aim of this study is not to obtain a higher absolute performance of any single algorithm but to compare the change in their performance, thus, the results would not be the best ones in absolute terms.

In the proceeding sections of this project, section 5 briefly summarizes the related work done on the subject. Section 6, explains the classification algorithms analysed in this project, the performance measuring metric and the strategies used to handle categorical features in the dataset. Section 7, describes the dataset, exploratory data analysis conducted on the dataset

[1] Home Credit Group: http://www.homecredit.net/
[2] Kaggle: https://www.kaggle.com/

and how outliers in the dataset are identified and handled. Section 8, explains the various version of the dataset that have been prepared for the analysis. In section 9, I have stated and analysed the empirical results of this project. Section 10 concludes the project and section 11 suggests possible future work in this area.

5. Literature review

The origin of credit scoring dates to about 70 years ago when it was proposed by (Durand, 1941). Traditional method of credit lending was based on human judgement of the risk of default. However, with increased pressure of a more subjective approach, rise in the demands of credit and greater commercial competition has led to the use of formal statistical methods for classifying applicants for credit into good and bad risk classes i.e. credit scoring (Hand & Henley, 1997). Much of research in this area explores the development, application and evaluation of credit scoring model for retail sector, which is more difficult than the credit score equivalent evaluation in corporate sector. Various sources of data like applicant's information, their transactional history and customer demographics are used for modelling. (Crook, et al., 2007). This provides a challenge in coming up with an accurate model.

LR and Decision Trees (DT) are some of the widely used algorithms for modelling. One of the main advantages of them being interpretable by both credit risk managers and regulators (Sullivan, et al., 2017). Methods using ensembles, artificial neural networks and deep learning also being explored by researchers but have not been widely accepted by the industry so far. This is mainly due to two reasons, first, they are not interpretable and secondly, they do not improve the results so drastically that it outweighs the fact that they are more computationally complex then industry standard. As far as accuracy is concerned, ensembles have been on top of the leader board according to some of the studies (Lessmann,

et al., 2015). The performance measure used in studies also varies with some using as many as six measures, justified by the merits of each measure.

Most of the existing research can be broadly classified into two classes, ones that suggest a novel algorithm and compare it with existing state-of-the-art and others, that compare existing algorithms using various performance measures. Not much literature can be found on the characteristics of individual algorithms and their behaviour with the type of credit scoring data they are modelled on. Hence, this project attempts to conduct in-depth analysis of three individual algorithms with the aim to provide some insight into this area.

6. Methodology

a. Algorithms Analysed

i. Logistic Regression (LR)

Logistic Regression (LR) is one of the basic and powerful ML classification algorithm used even today. It is a special case of linear regression with the difference being in its outcome. Linear regression output is continuous while regression is discrete.

LR models the relationship between one or more dependent variables with an independent variable. The outputs are probability predictions restricted between 0 and 1.

The output is a conditional probability P (Y=C| X=x) modelled by the following function

$$P(x) = \frac{e^{x.\beta + \beta_0}}{1 + e^{x.\beta + \beta_0}}$$

The coefficients β are estimated during training that determines the decision boundary separating two classes. The function is non-linear, meaning constant changes in input variables X do not reciprocate the same in P(x) predictions.

Linear Regression model is simple with only the regularization term, γ (gamma), as a hyper-parameter in the model. The performance of this technique is adequate to be used in industry despite more complex and advanced algorithms being available.

ii. Linear Support Vector Classification using Stochastic Gradient Descent

Linear Support Vector Classifier (LinearSVC) is a discriminative classifier defined by a hyperplane (or a set of hyperplanes) distinguishing between classes (Patel, 2017). In simple terms, given labelled training data the algorithm outputs hyperplanes which classify new examples. In a 2D this hyperplane becomes a line, in 3D it is a plane and so on.

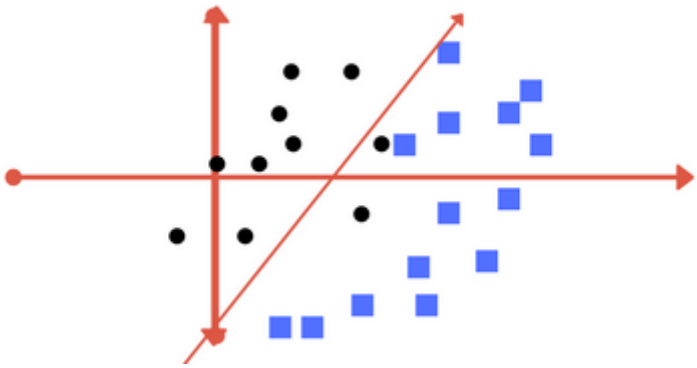

Figure 1: Linear SVC in 2D

Data is transformed using algebra before a hyperplane is learned for the model. Linear SVC uses a linear kernel, meaning linear algebra is used for data transformation. An example of a linear kernel can be

$$f(x) = \beta_0 + sum(\alpha * (x.x_i))$$

The coefficients β and α are estimated by learning algorithm. 'x' and 'x_i' are the input and i^{th} support vector respectively.

LinearSVC is a powerful classifier that works well in many cases. Especially when data is linearly separable. The training of model however, takes time. During learning the

error or loss is reduced by iteratively learning/improving the hyperplanes function using gradient descent to achieve the optimal point where loss is minimum. Thus, using it on large datasets is not recommended. For larger datasets process of Stochastic Gradient Descent (SGD) is used to converge nearest to its optimal solution. This improves upon the training time significantly while the performance of the trained model would be comparable to gradient descent approach.

Hyperparameters for LinearSVC are regularization term, gamma and margin. Therefore, model learning and tuning becomes more complicated compared to LR. Since this project is not concerned with optimization of each algorithm, tuning is not important and will not be done in much detail.

iii. LightGBM

Gradient Boosting Decision Tree (GBDT) is a popular technique of machine learning known for its accuracy, efficiency and interpretability. LightGBM is a variant of GBDT with even better efficiency and performance. This algorithm claims to be 20 times faster while achieving almost the same accuracy (Ke, et al., 2017). The difference between LGBM and other decision tree algorithms is that it grows vertically (leaf-wise) rather than horizontally (level-wise). This enables it to reduce more loss than level-wise propagating trees. (Ke, et al., 2017)

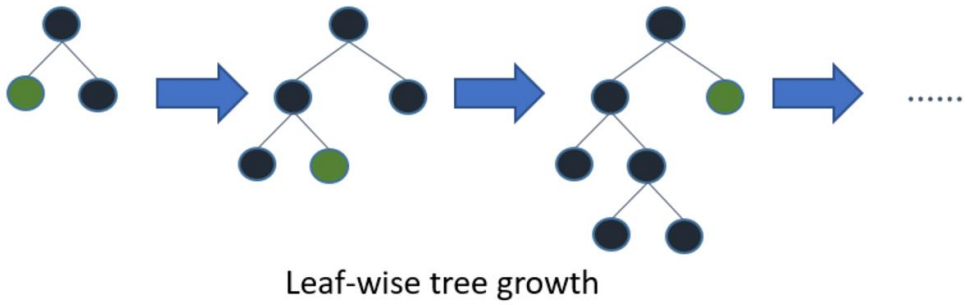

Figure 2: Leaf wise tree progression

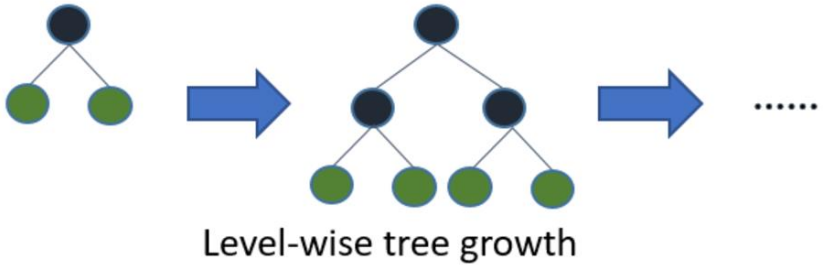

Figure 3: Level wise tree progression

LGBM is designed to handle large datasets by being parallelizable and less memory intensive. A problem with this algorithm is that it is prone to over-fitting (Mandot, 2017). Hence, it is not recommended for small datasets.

Another caveat is that it has even more hyperparameters then LinearSVC. Hence, hyperparameter tuning is not an easy task. Since, fine-tuning any algorithm is not a concern for this project so we would use this algorithms with default parameters.

b. Performance Metric: ROC-AUC

To measure the performance of ML algorithms I use a performance metric commonly used in classification problems. The Receiver Operating Characteristic Area Under the Curve (ROC-AUC, also sometimes called AUROC). It is a combination of two individual concepts. The Receiver Operating Characteristic (ROC) curve and the Area Under a Curve (AUC) (Koehrsen, 2018).

ROC curve tells us about how well a model can distinguish between classes. It plots the true positive rate versus the false positive rate. Each line in the Figure 4, below, shows the curve for a single model, and movement along a line indicates changing the threshold used for classifying a positive instance. The threshold starts at 0 in the upper right and goes to 1 in the lower left. A curve that is to the left and above another curve indicates a better model.

For example, the blue model is better than the red model, which is better than the black diagonal line which indicates a naive random guessing model.

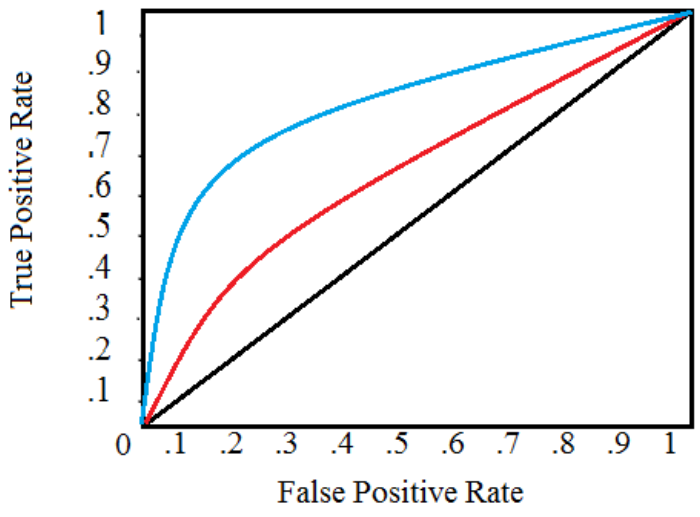

Figure 4: ROC curve

The Area Under the Curve (AUC) (Tape, n.d.) is simply the area under the ROC curve (the integral). This metric is between 0 and 1 with a better model scoring higher. A model that simply guesses at random will have an ROC AUC of 0.5.

To measure ROC AUC of a model, the probability of a predicted outcome is used instead of prediction itself. The advantage of ROC-AUC over traditional accuracy is when we run into problems with unbalanced classes. For example, in our data we have class imbalance of roughly 9:1, if I simply make a model that predicts every instance belongs to class 1, I would have a 90% accuracy score. Clearly, this would not be effective (the recall would be zero). Therefore, more advanced metrics like ROC-AUC, F1, Recall are used in such scenarios. For this project I am just using ROC-AUC, but other metrics can be used for further insights as well.

10-Fold cross-validation method will be used with stratified split of the data for each fold to maintain class imbalance. The mean ROC-AUC score will be calculated and shown in the empirical results section.

c. Categorical Features Handling

Categorical features are expressed with nominal values in the dataset. The issue with such values is that they cannot be quantified by ML algorithms and thus learning from those features, directly is not possible. However, there are some algorithms that are designed to accept nominal features and either convert them to numerical ones internally or the learning is designed so it can consume nominal variable as they are.

Standard practice in most cases is to convert nominal representation into numerical ones before training the model. There are a several techniques to convert variables from nominal to numerical representation.

For this project, I have chosen two techniques that are popular in ML.

i. One-Hot-Encoding (OHE)

In this technique nominal variable is converted into a truth table of all the nominal instances of that variable. For example, we have a categorical variable ‘Occupation’ that can have instances ‘Student’, ‘Professional’ or ‘Academic. The OHE would be as following:

Occupation
Student
Professional
Academic

Student	Professional	Academic
1	0	0
0	1	0
0	0	1

OHE is a simple and straightforward technique to apply and therefore is a popular choice. It has a few disadvantages that need to be considered while training the model. The most obvious being the increase in number of features. In the example above, one feature ‘Occupation’ has been transformed into three features ‘Student’, ‘Professional’ and ‘Academic’. This not only increases the dimensionality of the data but also adds sparsity i.e. many 0’s are added. This is a significant problem because some algorithms, especially classification algorithms are sensitive to both sparsity and dimensionality increase.

Another disadvantage is that new values do not give these features much quantifiably significance. They are just binary coded but the values themselves do not mean anything for the algorithm.

ii. Weight of Evidence Encoding (WoE Encoding)

Another encoding technique popular in credit scoring paradigm is the WoE encoding (Krishnan, 2018). Rather than adding new variables encoding is done in place for each instance depending on the class-based frequency values of that instance.

The formula for WoE is following:

$$WoE_i = \left[\ln \frac{Distr\ Good_i}{Distr\ Bad_i}\right] \times 100$$

Where 'i' denotes the specific instance (e.g. 'Student'), 'Distr Good' and 'Distr Bad' denote the class wise distribution of that instance.

I have used the sum of instances in each class as their respective distribution for this project. Weight of evidence has several advantages over OHE. First it does not increase the number of variable thus training time, data sparsity and dimensionality are not affected. Second, its values can be negative or positive numerical values that measures the strength of that instance in determining respective class. This is more useful in learning stage as it provides further insight into the relationship between a variable's instance and output class. However, it has an underlying assumption that this relationship is linear in nature. This can at times be misleading for the ML algorithm while training.

d. Code Implementation

The code is implemented in Python. I have used Python scikit-learn [3] package for all the exploratory data analysis and machine learning tasks in this project. The data is read and managed using Pandas[4] package.

The code is organized in four python script files. General functions are written in 'UtilityFunctions.py', EDA and pre-processing is done in 'DataExploration.py', data preparation is done in 'DataPrepration.py' and all classification algorithms are run through 'Classifier.py'. Outputs that require saving are saved in the 'CodeOutput' folder and other results are printed in the console. For further details, please refer to Appendix A

[3] Sklearn: http://scikit-learn.org/stable/
[4] Pandas: https://pandas.pydata.org/

7. Exploratory Data Analysis (EDA)

a. About the Data

The dataset provided by Home Credit for the competition is in the form of csv. There are 8 files in total with the main file being 'application_train.csv'. This file contains all the information filled in by the customer in a typical application form. It also contains some numerical features already calculated and used by the company for credit scoring.

Rest of the files are related to applicant's history from previous loans and his loan repayment history. For this project, I will primarily use two files, 'application_train', as it contains labelled data and the 'application_test' that contains the same data without class labels. The only difference in the features of these two files is the 'TARGET' column that represents the label of data points in training file. The second file will be used for imputing missing values and outlier detection. The rest of the files contain information that would be helpful in achieving a better performance for an individual classifier but that is not relevant to this project as we are more concerned with their relative performance.

The reason for selecting this dataset is that it contains higher number of data points than the ones used in previous studies. These numbers are shown in the Table 1, below. A large dataset would mean I have enough data instances for training even if I use a subset of the data to train the models.

File name	Rows (Data point)	Columns (Features)	Numerical Features	Categorical Features	Boolean Features	Miscellaneous Columns
Application_train	307511	122	78	16	26	2 (ID, Label)

Table 1: Input file description

As can be seen from the Table 1 above, the features can be divided into three types, categorical, numerical and Boolean. Categorical features depict nominal attributes with two or more types. Numerical features have quantitative values. The third is nominal attribute with strictly two types of values. These will be treated as categorical types in this project but because the dataset has them defined by integer that can take 0 or 1 (Boolean) values, I have mentioned them separately in Table 1.

b. Data Imbalance

As it is real-world data and the fact that Home Credit is still in business, we expect the data to be imbalanced (i.e. instances of good loans would be more than the bad loans). Otherwise, the company would be out of business for good.

The data imbalance is as below:

Value	Instance Count	%
0	282686	91.927118
1	24825	8.072882

Table 2: Data Imbalance

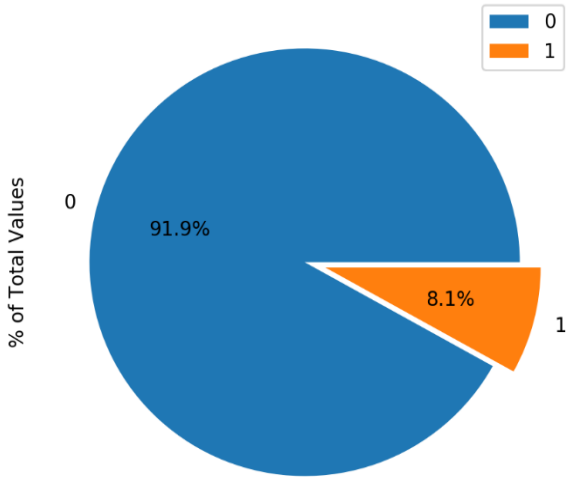

Figure 5: Data imbalance in training file

Imbalance in class instances is a challenge for classification algorithms. As most of the real-world data is not balanced, it would be a good characteristic to identify its effects on algorithms under study.

c. Null Values

All null values in our data are considered as missing values. Total number of features with missing values is 67 out of 121 features.

For the sake of brevity only top and bottom 10 features are shown in the tables below.

Feature	Missing Values	% of Total Values
COMMONAREA_MEDI	214865	69.9
COMMONAREA_AVG	214865	69.9
COMMONAREA_MODE	214865	69.9
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_MODE	213514	69.4
NONLIVINGAPARTMENTS_AVG	213514	69.4
FONDKAPREMONT_MODE	210295	68.4
LIVINGAPARTMENTS_MODE	210199	68.4
LIVINGAPARTMENTS_MEDI	210199	68.4
LIVINGAPARTMENTS_AVG	210199	68.4

Table 3: Top 10 Missing features

Feature	Missing Values	% of Total Values
DAYS_LAST_PHONE_CHANGE	1	0.0
CNT_FAM_MEMBERS	2	0.0
AMT_ANNUITY	12	0.0
AMT_GOODS_PRICE	278	0.1
EXT_SOURCE_2	660	0.2
DEF_60_CNT_SOCIAL_CIRCLE	1021	0.3
OBS_60_CNT_SOCIAL_CIRCLE	1021	0.3
DEF_30_CNT_SOCIAL_CIRCLE	1021	0.3
OBS_30_CNT_SOCIAL_CIRCLE	1021	0.3
NAME_TYPE_SUITE	1292	0.4

Table 4: Bottom 10 Missing Features

From Table 3, some features that represent medians, modes and averages of the same variable have equal amounts of entries missing. This implies the missing value is not due to data entry but the variable itself is not applicable for this data instance.

Table 4, shows the tail end of missing data. Some features have missing values count small enough that a simple strategy of imputing them would not affect the trained model's performance significantly.

d. Outliers

The next step for any analysis is to explore the data for outliers and apply some handling technique. For this project, we will replace outliers with null values. It is important to handle them at an early stage because it would affect not only the proceeding data exploration steps like correlation and feature distribution analysis, but also the strategies employed to handle missing values, imputing. As an example of outlier values in our dataset, the Figure 7, below shows the distribution of 'DAYS_EMPLOYED' feature from the data.

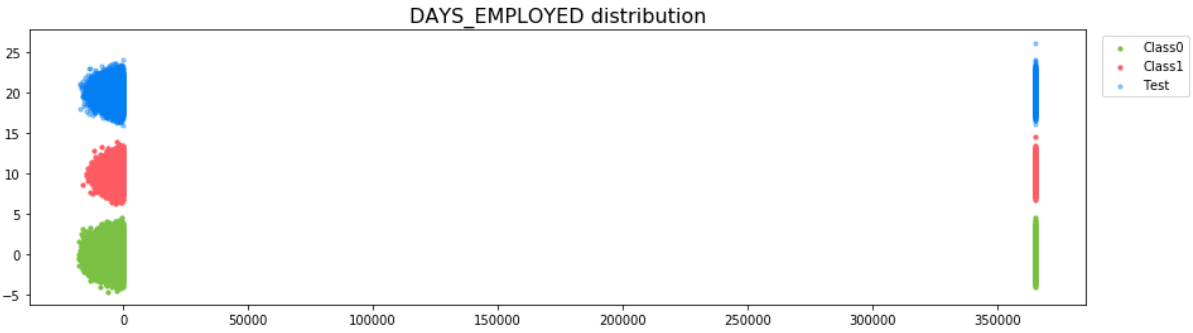

Figure 6: Days employed distribution with outliers

The three-different coloured distribution show class wise distributions (green and red), and the distribution of test/unlabelled data (blue). It makes the visualization and identification of outliers easier as we have three different distribution to compare. It is evident from Figure 6, above that there is a value above 350000, present in both training and test data. According to the description of this column given by the company, the days are counted in negative from the date of application. Thus, a positive value would suggest that

we treat this as a missing value. The distribution after removal of these outlier instances is shown below.

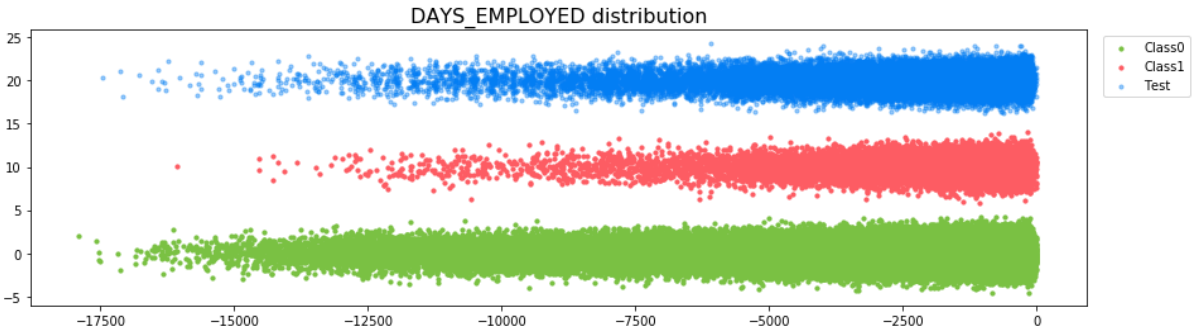

Figure 7: Days employed distribution after correction

Some data instances represent anomalies that might be caused by human error during data entry. As an example, distribution of 'AMT_INCOME_TOTAL' feature is shown below:

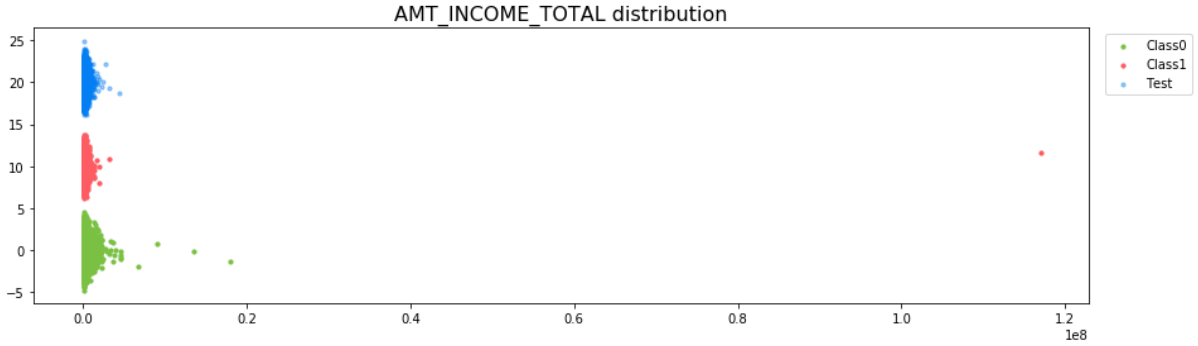

Figure 8: Total income distribution with outliers

After handling the outlier values, corrected distribution looks as follows:

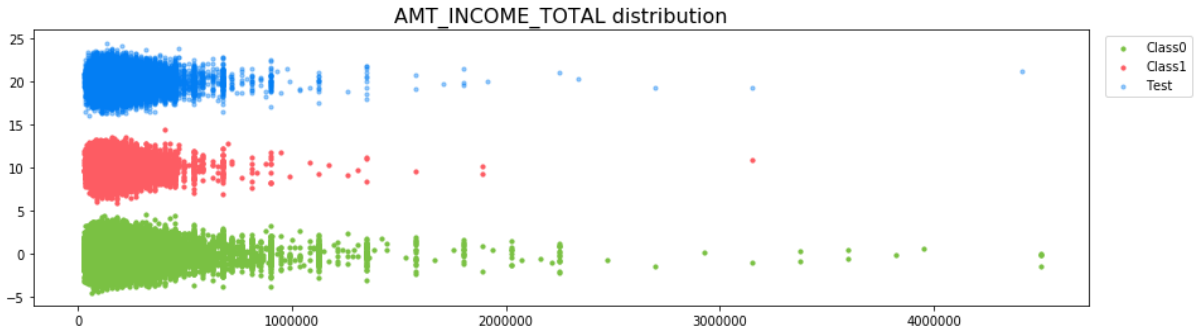

Figure 9: Total income distribution corrected

Now the data distribution seems to be correct. Outliers in the rest of the dataset have been replaced with NaN as well.

e. Correlation coefficient

Correlation coefficient of individual feature gives us a general feel of how our model predictions are going to look like. It is not the greatest method to represent the relevance of a feature, correlation is not equal to causation, but it gives an idea of relation between two variables. I have used Pearson correlation which is measured between the range -1:1.

General interpretations of the absolute value of the correlation coefficient are [9]:

- .00-.19 “very weak”
- .20-.39 “weak”
- .40-.59 “moderate”
- .60-.79 “strong”
- .80-1.0 “very strong”

I have compared correlation of each feature in our dataset with ‘Target’ variable, that is the labelled outcome.

The top 20 features and their correlation values are as follows:

Feature Name	**Correlation**
EXT_SOURCE_3	-0.1789262756758259
EXT_SOURCE_2	-0.16047133438875621
EXT_SOURCE_1	-0.15531709813952868
FLOORSMAX_AVG	-0.04400853854768869
FLOORSMAX_MEDI	-0.04377305441770841
FLOORSMAX_MODE	-0.04323129229345895
AMT_GOODS_PRICE	-0.03964669449606092
REGION_POPULATION_RELATIVE	-0.03722486669369183
ELEVATORS_AVG	-0.03420227894409464
ELEVATORS_MEDI	-0.033866324288584215
……..	…….
……..	…….

FLAG_DOCUMENT_3	0.04434104644754352
REG_CITY_NOT_LIVE_CITY	0.04439430211113305
FLAG_EMP_PHONE	0.04598411839565971
REG_CITY_NOT_WORK_CITY	0.05099248334979859
DAYS_ID_PUBLISH	0.051457257258228584
DAYS_LAST_PHONE_CHANGE	0.05521930350886765
REGION_RATING_CLIENT	0.05890087937253525
REGION_RATING_CLIENT_W_CITY	0.060894746160379966
DAYS_EMPLOYED	0.07495658541678334
DAYS_BIRTH	0.07824216003637195

Table 5: Feature correlations strongest negative to strongest positive

From the Table 5 above, we can see that our feature set has very weak correlation with the target. This makes it a challenge for ML algorithms to train on. A standard practice would be to do further analysis and come up with engineered features that show a stronger relationship with the target variable but since we are not concerned with improving our ML algorithms performance individually, we can skip this step and work with the data as is.

Correlation coefficient can also be used to determine relationships between the features themselves. It identifies features that complement each other. This information is vital because models can end up giving undue weightage to such features and might end up overfitting. I have not considered such detailed analysis of the data for this project as its effect will be seen on all ML algorithms I am analysing and thus, is not important for my analysis.

f. Feature Cass-wise Distribution

Another way of looking at the relation between features and the target variable is to visualize their distribution with respect to assigned classes. This can visually tell us if there is a certain pattern in the feature that can be used to classify the data instance i.e. if specific range of values of a feature occur more in one class than the other then it is easier to predict if a data instance would be classified in that instance or not. On the other hand, if the distribution overlaps for both the classes then it would be difficult.

I will use the Kernel Density Estimate (KDE) plot[5], coloured by the value of the target. It can be thought of as a smoothed histogram.

The KDE plot of two of the features are shown below:

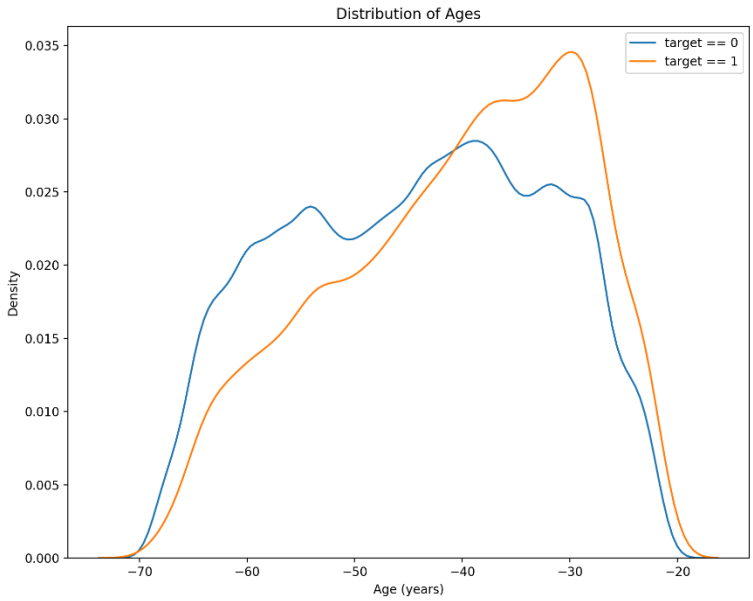

Figure 11: KDE plot of Ages (in years)

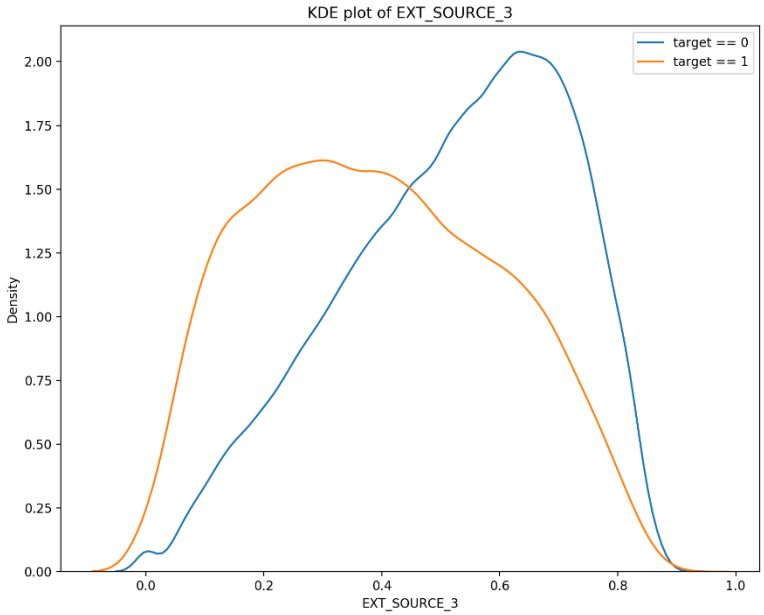

Figure 10: KDE plot of External source 3

[5] KDE Plot: https://en.wikipedia.org/wiki/Kernel_density_estimation

Correlation values of ‘DAYS_BIRTH’ and ‘EXT_SOURCE_3’ are 0.07 and -0.17 respectively. Former is feature with strongest positive correlation and the later, strongest negative correlation value. The KDE plots show that there is a strong overlap among both the classes for the respective features. This enforces the conclusion of our correlation table that distinguishing between classes with the given features is not a simple task for any of the ML algorithm we are going to analyse.

8. Data Preparation

The variants of data have been designed to highlight certain aspects of the algorithm’s performance. Following are the types of variations with their purpose explained:

a. Data with null values as NaN.

It would highlight how well the algorithm can perform with missing data. Real-world data has missing or null values in most cases. Different pre-processing techniques are used to handle them. Some algorithms do not require us to pre-process null values. We can compare and see how these algorithms perform with null values.

b. Data with null values omitted.

Omission of null value instances of data is a basic approach to handle them. If they are not that much in number, it would hardly affect model’s performance. However, if any data point with null values is fed into a trained model at runtime, it would not be able to classify it. In that case some other strategy might be required to handle null values before feeding it to the model. As far as model training is concerned, omitting null values would reduce the number of data points but rest of the data would remain the same. Comparing results of this version with the others would tell us how the algorithm performs with lesser data points but complete without any form of imputation done to it. For our dataset omitting null values reduces it to 1/3 of the original.

c. Data with null values Imputed.

Another approach to handling missing values is to impute them. This approach adulterates the original data but gives an acceptable trained data at the end. Comparing these results with others would highlight how sensitive the algorithm is to data engineering, that might or might not be close to ground truth. Imputing strategy we will use is to take fill the missing values with mode values for categorical features and mean for the numerical features.

d. Data with numerical features imputed and scale between 0-1

Pre-processing of data is an essential step for preparing data for model training. Transformations like imputing, scaling, feature engineering occur at this stage. Deciding what transformations and pre-processing to conduct is dependent on the kind of algorithms that is being used to train the model. In this project I will create a version of dataset that is imputed and then scaled between 0 and 1. This would show us how sensitive the algorithm is to the scale of numerical features in the data. It would also highlight if the scale of numerical features effect the weightage given categorical ones by the algorithm. Comparing it with other version results we can conclude how it affects the performance of our algorithm.

e. Dataset with feature selection.

Feature selection will reduce the number of features available to train our model by choosing only the most relevant features out of the dataset. It will not only reduce the dimensionality of the data but also the noise because only those features would be considered that help in differentiating between the classes. For feature selection I have used a voting system in which 5 different algorithms cast a vote for each feature to be included in top N features or not. These 5 algorithms include univariate feature selection (variance threshold, Chi^2, Pearson correlation coefficient), recursive feature elimination (with LR as estimator)

and model-based ranking (random forest ensemble). Features with 3 or more votes are added in selected feature lists.

f. Data sampled with balanced classes and stratification

A variant of dataset will contain a subset of the data with equal number of samples representing each class. And one with same number of samples but stratified to depict the original imbalance. The purpose of this dataset would be to highlight how data imbalance affects an algorithm's performance. The sub-sampling of the data would equally lower the performance of our algorithm but by comparing it with same sized stratified sub-sample, we can conclude how imbalance is affecting the algorithm.

For the top 5 (a-e) variations mentioned above, the data would be further varied w.r.t to how categorical features are handled. The following techniques of categorical feature handling will be used to create further sub-versions of the data:

- Data without categorical features
- Data with One-Hot-Encoding of categorical features
- Weight of Evidence Encoding of categorical features

All these variants combined make 17 different versions of the dataset. Next section states the results and analysis them.

9. Empirical Results and their analysis

The Table 7, below shows the performance results for each algorithm run in different version of the dataset. The versioning of the dataset is explained in detail in the 'Data Preparation' section above.

The data version and sub-versions are as follows:

	File Version	Sub - version		
		Without Categorical Features	With Categorical Features OHE	With Categorical Features WoE Enc.
Data with null values	1	a	b	c
Data without null values	2	a	b	c
Data with null values imputed	3	a	b	c
With null values imputed and scaled	4	a	b	c
With feature selection	5	a	b	c
Subset with balanced samples	6	X	a	X
Subset with stratified samples	6	X	b	X

Table 6: File versioning

The Table 7 shows the scores for each version against classification algorithm. Refer to Table 6 to see what data version represents.

Data Version	Sub-Version	Logistic Regression ROC-AUC	LinearSVC (SGD Classifier)	Light GBM
1	a	X	X	0.742
	b	X	X	0.752
	c	X	X	0.752
2	a	0.621	0.536	0.746
	b	0.621	0.524	**0.756**
	c	0.621	0.528	**0.756**
3	a	0.621	0.533	0.746
	b	0.622	0.529	0.751
	c	0.622	0.525	0.752
4	a	0.710	0.519	0.740
	b	0.733	**0.554**	0.751
	c	**0.734**	0.529	0.751
5	a	0.712	0.517	0.741
	b	0.729	0.541	0.751
	c	0.730	0.519	0.752
6	a	0.724	0.536	0.755
	b	0.678	0.532	0.733

Table 7: ROC-AUC Scores

Some algorithms have limitations with respect the null values in the data. Standard practise is to implement null/missing values handling strategy before model fitting. They have not been treated in this project and a variable ‘X’ is used in the to denote these cases.

The results in the Table 7 are analysed algorithm wise below.

a. Logistic Regression:

First, it is not able to handle null values out of the box. Some sort of null value handling will be required. Second, null value omission or their imputation has no change in the scores. This means LR's performance is not strongly dependent on the size of the data. Provided it is significantly large enough to get a baseline score, adding more data will have less an effect.

Third, when data is imputed and scaled, there is a stark difference in performance. Where LR has its overall highest score when data is imputed, scaled and categorical features are WoE encoded. This shows that it is most sensitive to the scale and the numeric nature of features. If data can be scaled and represented in this form, then LR can perform well. Fourth, feature selection on top of imputation and scaling has not improved the scores much. This further supports our second point that LR is not affected as significantly by adding more data, in terms of data instances or features, as it is by the numeric nature and scale of the features. Lastly, the difference in the scores of sub-sampled balanced and stratified data is largest amongst the three classifiers. Thus, the balance of training data is most critical for LR among these classifiers.

b. LinearSVC:

Despite using SGD for model training, LinearSVC took roughly 3x longer time to complete. This means it will be a very bad choice for training on large datasets. Overall, the scores are closest to 5 amongst the 3 classifiers, that shows it is not suitable for data that does not show strong linear relationship. SVC is a strong classifier in general, but we only used a linear kernel, changing this and other hyper-parameters will improve the scores. This shows that choosing right hyperparameters is critical for LinearSVC. Like LR it cannot handle null values internally. Looking at the relative scores in the Table 7, we can draw some

conclusions. First, it is not sensitive to amount of data. Data with null values omitted and the one with imputed null values both have score around the same range. Second, from version 3 and 4 of the datasets, categorical values handling lowers the score, whether we do OHE or WoE encoding. Second, imputing and scaling data changes the algorithms behaviour, OHE has the highest overall score when data is imputed and scaled while that for WoE encoded and no encoding, drops significantly. This would mean LinearSVC is sensitive to scaling. The scaling of numerical features makes LinearSVC more responsive to the categorical ones and the sub-set with OHE, that has most features gives the highest score. Third, feature selection reduces the score. This can be because the reduced feature set has features that have even lower linear relationship and thus, the model performs lower. Fourth, data imbalance has least effect on LinearSVC and so it can be ideal for data with imbalance.

c. LightGBM

LGBM is robust to missing data. The algorithm is designed to handle null values internally. The overall score suggests the algorithm performs comparatively well out-of-the-box but there are several hyperparameters that can be tuned to improve results further. The encoding of categorical features improves the result, but type of encoding does not make much of a difference. The best overall results are achieved for data with null values imputed. This means, although the algorithm can handle missing values, its performance is sensitive to it. Thus, LGBM is sensitive to missing value although it is robust enough to perform despite them being present in the dataset. Scaling numerical values does not improve the model's performance provided there are no unscaled categorical features present. Feature selection has not improved the results either. This suggests LGBM would work better with higher dimensional noisy data. Then a low dimensional cleaned data. Class imbalance also influences the results, but they are not as sensitive as LR.

10. Conclusion

The aim of this project was to conduct an in-depth analysis of ML algorithms on credit scoring dataset. Credit scoring dataset is chosen due to its diversity and the difficulty for classification algorithms to cope with it. Using a single large dataset ensures that core of the data does not change so our results are comparable.

I have used 17 different versions of the dataset for comparison. Each version is obtained by applying a different strategy of pre-processing to it which changes the number of features or their types. Main characteristics highlighted in these variants are data quantity, quality (imputed and original), data sparsity (OHE data), the types of features and their number, and the class distribution in training data.

My analysis is restricted by studying only three algorithms and the credit scoring data provided by Home Credit, due to time constraints. More credit scoring data and algorithms would provide further insights.

Results show that LR, the industry standard in credit scoring (Lessman, et al. (2015), is most responsive to numeric nature of the data and its scale, compared to LinearSVC and LGBM. LR shows least improvement when training data size was increased with imputed missing values. This concludes that LR models are better generalized then others as adding more data points improve its performance only slightly. LinearSVC shows the lowest overall performance out of the three algorithms we have analysed. EDA of the dataset shows that there is very low correlation between features and the target classes signifying low linear relationship. From this I conclude that linear relationship is critical for LinearSVC. Other observations about LinearSVC are that it also gives significant weightage to numerical features of the dataset and their scale as the algorithms behaviour changes when scale of numeric features is changed. Sparsity is not an issue for LinearSVC as it performs well when

those features are OHE. Imbalance in training data affect LinearSVC the least of the three. LGBM is a recent addition to ML classification algorithms compared to those mentioned above and it is more robust then the two. It can perform despite null values in the dataset, but its performance can improve significantly if there are no missing values. LGBM is not sensitive to sparsity or the type of features in the dataset as its performance on OHE dataset and WoE encoded data is almost the same. Data imbalance does affect the algorithm's performance but not as much as LR. Overall it has out-performed the other two algorithms and that is the reason it is a popular algorithm amongst Kaggle community for classification problems especially in credit scoring.

11. Future Work

Availability of more data and the advancements in novel classification algorithms for credit scoring, and the fact that there is not a clear winner amongst the algorithms suggests that it is research gap that would constantly need to be updated from time to time. Further studies can be conducted by analysing more algorithms that are used in credit scoring classification problems and including multiple datasets provided by credit lending companies that can be used to compare properties that are common among them. I have used a single performance metric for comparison (ROC-AUC). It has been suggested in previous studies (Lessman et al. (2015)) that a single metric cannot tell us everything about the classifier and that multiple metrics should be used for the purpose.

This project can be considered as first step towards gaining insight into how the characteristics of credit scoring data effects the classification algorithms. More algorithms can be analysed in this manner and publicly available datasets from various domains can provide a clearer picture.

From application perspective, it is important to know why certain algorithms outperform others on a particular dataset. It would help in deciding which algorithm is suitable for the data in hand, what kind of pre-processing steps could help improve the performance and predict how the algorithm would behave if the dataset transforms in the future. Observing the discussions on Kaggle competition (Home Credit Group, 2018), I have realized that general method of algorithm selection is merely based on trial and error and not an informed decision based on the characteristics of the dataset and how they are pre-processed. What this project proposes, is to develop a knowledge base that can be used to pick ML algorithms for credit scoring problems, according to the nature of the dataset available and then select pre-processing steps that would facilitate the most suitable ML algorithm.

12. Bibliography

Baesens, B. et al., 2003. Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring. *The Journal of the Operational Research Society*, 54(6), pp. 627-635.

Bank of England, 2018. *Amounts Outstanding of total sterling net consumer credit linding to induviduals.* [Online]
Available at:
https://www.bankofengland.co.uk/boeapps/database/fromshowcolumns.asp?Travel=NIxAZxSUx&FromSeries=1&ToSeries=50&DAT=RNG&FD=1&FM=Jan&FY=2010&TD=11&TM=May&TY=2025&FNY=Y&CSVF=TT&html.x=66&html.y=26&SeriesCodes=LPQBI2O&UsingCodes=Y&Filter=N&title=LPQBI2O&VPD=Y
[Accessed July 2018].

Bank of England, 2018. *Monthly growth rate of total sterling net consumer credit lending to induviduals.*. [Online]
Available at:
https://www.bankofengland.co.uk/boeapps/database/fromshowcolumns.asp?Travel=NIxSUx&FromSeries=1&ToSeries=50&DAT=RNG&FD=1&FM=Aug&FY=2008&TD=1&TM=Aug&TY=2018&FNY=&CSVF=TT&html.x=95&html.y=41&C=NZU&C=O1P&Filter=N]
[Accessed July 2018].

Crook, J. N., Edelman, D. B. & Thomas, L. C., 2007. Recent developments in consumer credit risk assessment.. *European Journal of Operational Research,* Issue 183, pp. 1447-1465.

Desjardins, J., 2018. *Traditional Credit scoring.* [Online]
Available at: http://uk.businessinsider.com/heres-how-technology-is-changing-the-consumer-credit-industry-2018-5
[Accessed August 2018].

Durand, D., 1941. *Risk Elements in Consumer Instalment Financing.* s.l.:NBER Books.

Hand, D. J. & Henley., W. E., 1997. Statistical classification methods in consumer credit scoring: a review.. *Journal of the Royal Statistical Society: Series A (Statistics in Society),* Issue 160.3, pp. 532-541.

Ke, G. et al., 2017. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* s.l., s.n.

Lessmann, S., Baesens, B., Seow, H.-V. & Thomas, L., 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research.. *European Journal of Operational Research.*.

Mandot, P., 2017. *What is LightGBM, How to implement it? How to fine tune the parameters?.* [Online]
Available at: https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc
[Accessed August 2018].

Owen, A., n.d. *Pearson Correlation.* [Online]
Available at: http://www.statstutor.ac.uk/resources/uploaded/pearsons.pdf
[Accessed August 2018].

Sullivan, H., Christophe, H. & Tokpavi, S., 2017. *Machine Learning for Credit Scoring: Improving Logistic Regression with Non Linear Decision Tree Effects.* s.l., s.n.

Wikipedia, n.d. *KDE Plot.* [Online]
Available at: https://en.wikipedia.org/wiki/Kernel_density_estimation
[Accessed August 2018].

Zhu, B., Yang, W., Wang, H. & Yuan, Y., 2018. A Hybrid Deep Learning Model for Consumer Credit Scoring. *International Conference on Artificial Intelligence and Big Data.*

13. Appendix A: Code Overview

My code for this project is hosted in the following GitHub Repository:

https://github.com/MOWaqar/MSc-Project

Code snippets and ideas are taken from kernels shared in Kaggle competition (Home Credit Group, 2018) where necessary.

i. Packages used:

Following packages have been used in the code:

- Sikit-learn (sklearn)
- Pandas
- Python 3.6
- LightGBM (lgbm)
- Seaborn
- MatplotLib
- Numpy

Other general packages have not been mentioned here.

ii. Relevant Files:

The code is divided in four separate files (python scripts) and need to be executed individually in the order they have been listed below. Details about each function, its output and working can be seen in the comments of the code.

ProjectCode/UtilityFunctions.py:

This file contains general functions like reading setting up folder paths, reading input file and functions used by subsequent files. It does not have a main function that needs to be executed.

ProjectCode/DataExploration.py:

This file has functions used to conduct exploratory data analysis. Some outputs are printed in console while others are saved in 'CodeOutputs' folder.

ProjectCode/DataPreperation.py:

This file has functions used to prepare the 17 different versions of the dataset. It takes some time to execute so be patient. The outputs are saved in csv's in the 'CodeOutput' folder.

ProjectCode/Classifier.py:

This file runs the three classifiers, Logistic Regression, Stochastic Gradient Classifier with LinearSVC, and LightGBM on all the 17 files created in the previous step. The results are printed in the console. This code could take hours to execute.

iii. Setup:

- Clone repository
- Unzip 'ProjectDataFiles.zip' in the root folder
- Install requirement by executing the following command in root folder:
  'install -r requirements.txt'
- Run 'DataExploration.py'
- Run 'DataPreperation.py'
- Run 'Classifier.py'

NOTE: It is important to maintain the folder structure otherwise the folder setup would fail causing errors.