Comparative analysis of ML algorithms using Credit Scoring Data.

# **Acknowledgements**

\*\*CODE SNIPPETS TAKEN FROM KAGGLE COMMUNITY, HOME CREDIT COMPANY FOR THEIR DATA SET\*\*

# **Abstract**

\*\*ABSTRACT WILL BE ADDED IN THE END WHEN THE REPORT TAKES A FINAL SHAPE\*\*

Contents

[**1.** **Acknowledgements** 1](#_Toc522220485)

[**2.** **Abstract** 1](#_Toc522220486)

[**3.** **Introduction** 3](#_Toc522220487)

[**4.** **Literature review** 6](#_Toc522220488)

[**5.** **Methodology** 7](#_Toc522220489)

[**a.** **Algorithms Analysed (\*\*BRIEF EXPLAINATION OF EACH ALGORITHM\*\*)** 7](#_Toc522220490)

[**i.** **Logistic Regression (LR)** 7](#_Toc522220491)

[**ii.** **Linear Support Vector Classification using Stochastic Gradient Descent** 8](#_Toc522220492)

[**iii.** **LightGBM** 10](#_Toc522220493)

[**b.** **Performance Metric: ROC-AUC (\*\*BRIEF EXPLAINATION OF ROC-AUC\*\*)** 11](#_Toc522220494)

[**c.** **Categorical Features Handling** 12](#_Toc522220495)

[**d.** **About the Data** 15](#_Toc522220496)

[**6.** **EDA and pre-processing** 19](#_Toc522220497)

[**7.** **Data Preparation(\*\*WORK IN PROGRESS\*\*)** 20](#_Toc522220498)

[**8.** **Empirical Results and their analysis** 21](#_Toc522220499)

[**9.** **Conclusion** 22](#_Toc522220500)

[**10.** **Future Work** 22](#_Toc522220501)

[**11.** **References / bibliography (\*\*WILL BE ADDED SOON\*\*)** 22](#_Toc522220502)

[**12.** **Appendices** 22](#_Toc522220503)

# **Introduction**

The advancement in information technology has driven a change in all walks of life. The 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. This has increased almost 37% in just 5 years [1]. With a monthly growth rate going as high as 10.9 percent [2]. 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.

This 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/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 consumer’s life. It estimates a person’s credit worthiness based a person’s credit history and correlation of their financial behaviour to credit risk.

The concept of numeric credit scoring emerged in 1989 and used logistic regression at its core [3]. Logistic regression is still the most widely used algorithm in industry but newer technologies like big data and artificial intelligence are constantly evolving the credit landscape. 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). 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 improved classification algorithms (e.g Hui Sun, et al. (2015), Bing 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 recently 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 data sets 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 data set improve an algorithms performance or what traits of an algorithm make it suitable for a given data set. The lack of conclusive study showing one algorithm to be irrefutably better the others shows that there is a research gap that has not been addressed so far.

With this project I intend to find out which algorithm would suit the kind of data available for modelling. The project is limited to studying only three algorithms, Logistic Regression (LR), Support Vector Machine (SVC) and relatively new LightGBM (LGBM). The data set used in this project is provided by Home Credit company for a competition on Kaggle 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 evaluation results of the mentioned algorithms on different versions of the data set. These versions of data differ by using different subset of features from the original data and varying approaches to handling missing values and the categorical features. To shortlist meaningful features, feature selection is performed initially to reduce dimensionality of the data that will be used as baseline. The variations in data are designed to highlight how the models respond to certain aspects of data set. The aim is to identify the strength and weaknesses of the algorithms under study by comparing their results with respect to the training input. Furthermore, hyper parameters of these algorithms are then fine-tuned to improve upon the base results. The aim of this study is not to obtain a higher absolute accuracy of any single algorithm but to compare their relative accuracy, thus, the results would not be the best ones in absolute terms.

This project is broadly divided into section. Section 1 describes the dataset, describes the steps taken to explore the data, handling of outliers in the data set, feature selection and how the variants of the data sets are extracted. Section 2 briefly explains the algorithms and how they will be evaluated, area under receiver operating characteristic curve. In Section 3 the obtained results are discussed followed by section 4 that covers hyper-parameter tuning of algorithms and their respective results. Section 5 gives a conclusion and what the future direction this project could take. Section 6 is a bonus that tries to use CNN model.

# **Literature review**

The origin of credit scoring dates to about 70 years ago when it was proposed by Durand [4]. 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 (Credit scoring) [5]. Much of research in this area explores the development, application and evaluation of credit scoring model for retail sector [6]. Various sources of data like applicant’s information, their transactional history and customer demographics are used for modelling. This provides a challenge in coming up with an accurate model.

Logistic Regression (LR) and Decision Trees(DT) are some of the widely used algorithms for modelling. One of the main advantages of them being the interpretable by both credit risk managers and regulators [7]. 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. 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 different 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.

# **Methodology**

## **Algorithms Analysed (\*\*BRIEF EXPLAINATION OF EACH ALGORITHM\*\*)**

## **Logistic Regression (LR)**

Logistic regression is one of the basic, yet powerful ML classification algorithm used even today. It is a special case of linear regression with the difference being in it outcome. Linear regression output is continuous while regression is discrete.

Logistic regression 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

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

Linear regression model is simple with only the regularization term γ (gamma) as a hyper-parameter in the model that is tuned to determine optimal model. The performance of this technique adequate enough to be used in industry.

## **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. 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.

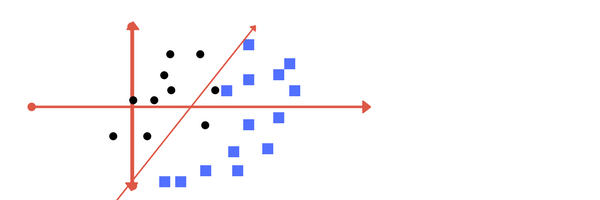


Figure 1: 2D hyper plane (line) SVC

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

The coefficients β and α are estimated by learning algorithm. ‘x’ and ‘xi’ are the input and ith support vector respectively.

LinearSVC is 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 data sets Stochastic Gradient Descent (SGD) technique 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 at the least.

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

## **LightGBM**

Gradient Boosting Decision Tree(GBDT) is a popular technique of machine learning known for its accuracy, efficiency and interpretability [8]. 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 [8]. 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.

A close up of a sign

Description generated with high confidence

A picture containing object

Description generated with high confidence

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

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

## **Performance Metric: ROC-AUC (\*\*BRIEF EXPLAINATION OF 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).

A close up of a logo

Description generated with very high confidenceIt is a combination of two individual concepts. The Reciever Operating Characteristic (ROC) curve and the area under a curve. ROC curve tells us about how well a model can distinguish between classes. It plots the true positive rate versus the false positive rate:

Figure 2: ROC Curve

Each line in the figure above 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, in Figure 1, the blue model is better than the red model, which is better than the black diagonal line which indicates a naive random guessing model.

The Area Under the Curve (AUC) 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 prediction 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 metrices can be used for further insights as well.

## **Categorical Features Handling**

Categorical features are expressed with nominal values in the data set. 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 of the ML 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 among ML community.

1. One-Hot-Encoding (OHE)

In this technique nominal variable is converted in two 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 |

Table 2: OHE variables

|  |
| --- |
| Academic |

|  |  |  |
| --- | --- | --- |
| Student | Professional | Academic |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Table 1: Original Variable

OHE is a simple and straight forward 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 data sparsity i.e many 0’s are added. This is a significant problem because many algorithms, specially 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.

1. Weight of Evidence Encoding (WoE Encoding)

Another encoding technique especially popular in credit scoring paradigm is the WoE encoding. 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:

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 be negative/positive decimal values that measures the strength of that instance in determining the 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 underlaying assumption that this relationship is linear in nature. This can at times be misleading for the ML algorithm while training.

## **About the Data**

The data set 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’. It contains the features from the loan applications filled by customers along with 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. For this project, I will primarily use ‘application\_train.csv’ as it contains labelled data and the ‘application\_test.csv’, which contains unlabelled data, for imputing missing values and outlier detection and handling. The rest of the files contain information that would be helpful in achieving a better absolute performance but that is not relevant to this project as we are more concerned with their relative performance.

The reason for selecting this data set is that it contains the highest number of data points than the ones in previous studies. These numbers are shown in the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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) |
| Application\_test | 48744 | 121 | 78 | 16 | 26 | 1 (ID) |

Table 1: Input files description

The only difference in the features of application\_train.csv and application\_test.csv is the ‘TARGET’ column that represents the label of data points in training file. The test file does not have labels, but it is still useful for the purposes mentioned above.

As can be seen from the table above, the features can be divided into three types, categorical ones depict nominal attributes with two or more types. The second type are the numerical features that have quantitative values. The third is nominal attribute with strictly two types of values. These will be treated separately then categorical types because the data set has them defined by integer that can take 0 or 1 (Boolean) values.

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

A close up of a logo

Description generated with high confidence

Figure 3: Data n 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 meaningful characteristic to identify for the algorithms under study.

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

For the sake of brevity only top and bottom 10 features are shown in the table 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 instances. We can safely ignore such features to simplify our data set. It would impact the performance of our classifiers equally, so we don’t need to worry about them.

From table 4, some features have missing values count small enough that a simple strategy of imputing them would be a better than discarding those features for this project. I have chosen to discard all features that have more than 20% of values missing. And the rest of the features would be imputed with mean values of those features. For imputing values, I will incorporate the application test data points as well.

After excluding the features with missing values greater than 20% we are left with the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Categorical Features | Boolean Features | Numerical Features | Total Count |
| 11 | 26 | 33 | 70 |

# **EDA and pre-processing**

First step for any analysis is to explore the data and perform some sort of pre-processing to prepare the data as required by the analysis. For this project, these pre-processing steps include outlier detection and handling, missing value imputation and feature selection for later use. Before we impute any features, it is important that we identify, and handle, outliers. Otherwise, the values used to impute the data, would be skewed by those outlier values. As an example of outlier values, figure below shows the distribution of ‘DAYS\_EMPLOYED’ column.

A screenshot of a cell phone

Description generated with very high confidence

Figure 4: Days employed feature distribution

The three-different coloured distribution show class wise distributions (green and red), and the distribution of test/unlabelled data (blue). It makes the visualization of outliers better as we have three different distribution to compare. It is evident from figure 2 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. For now, I replace the value with NaN. The distribution after removal of this outlier is shown below.

A screenshot of a social media post

Description generated with very high confidence

Figure 5: Days employed distribution after correction

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

# **Data Preparation(\*\*WORK IN PROGRESS\*\*)**

The variants of data that will be analysed are based upon the following criteria:

1. Data with null values as NaN.

It would highlight how well the algorithm in question can handle NaN values.

1. Data without Null Values.

It would highlight how algorithm performs with lower number of data points but whole otherwise.

1. Data with null values Imputed.

It would highlight how sensitive the algorithm is to data engineering that might or might not be accurate. Imputing strategy we will use is to take mode value for categorical values and mean for numerical values.

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

It would highlight how sensitive the algorithm is to scale of features in data set.

The data would be further varied w.r.t to categorical features are handled.

1. Data without categorical features
2. Data with One-Hot-Encoding of categorical features
3. Weight of Evidence Encoding of categorical features

This would mean we will have 3x4 = 12 variants of the data.

After observing the results for the data variants mentioned above. We will take the best overall data set from above and try feature scaling and feature selection (N-voters strategy) on it. This would highlight how sensitive each algorithm is to the curse of dimensionality or noise.

Another variation of data will counter data imbalance with stratified K-fold training. This would highlight algorithms sensitivity to data imbalance.

# **Empirical Results and their analysis**

The table below shows the performance results for each algorithm run in different version of data set. The versioning of the data set is explained in detail in the ‘Data Preparation’ section above.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | 0.720 | 0.524 | 0.750 |
| b | 0.733 | 0.554 | 0.757 |
| c | 0.734 | 0.529 | 0.757 |

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. A variable ‘X’ is used in the table above to denote these cases. It signifies the strength of LGBM classifier to be more robust over all.

\*\*MORE ON THE TABLE ABOVE TO BE ADDED HERE\*\*

# **Conclusion**

# **Future Work**

# **References / bibliography (\*\*WILL BE ADDED SOON\*\*)**

# **Appendices**