Comparative analysis of ML algorithms using Credit Scoring Data.

# 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 an analysis.

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.

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

# 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, in industry and academia, alike. Continued advancement in the field of ML and ever increasing size of available data is constantly evolving the landscape. Studies are either suggesting novel approaches or using more and more data to compare existing algorithms. In the middle of all this, little attention is being given to carry an in-depth analysis of algorithms to understand their capabilities w.r.t data at hand. In this project, I have attempted to shed some light on this area. First, I conduct exploratory analysis of the data to understand its characteristics. Then I prepare different versions from that data set to highlight some those characteristics. Afterwards, the performance of each algorithm on all versions of the data are observed and analysed. This study opens new avenues of analysis that need to be explored before we advance further in the field of ML.

*Keywords:* Supervised Machine Learning, Classification, Credit Scoring.

Contents

[1. Acknowledgements 1](#_Toc522585573)

[2. Abstract 1](#_Toc522585574)

[4. Introduction 2](#_Toc522585575)

[5. Literature review 4](#_Toc522585576)

[6. Methodology 5](#_Toc522585577)

[a. Algorithms Analysed 5](#_Toc522585578)

[i. Logistic Regression (LR) 5](#_Toc522585579)

[ii. Linear Support Vector Classification using Stochastic Gradient Descent 6](#_Toc522585580)

[iii. LightGBM 8](#_Toc522585581)

[b. Performance Metric: ROC-AUC 8](#_Toc522585582)

[c. Categorical Features Handling 10](#_Toc522585583)

[i. One-Hot-Encoding (OHE) 10](#_Toc522585584)

[ii. Weight of Evidence Encoding (WoE Encoding) 11](#_Toc522585585)

[7. Exploratory Data Analysis (EDA) 12](#_Toc522585586)

[a. About the Data 12](#_Toc522585587)

[b. Data Imbalance 13](#_Toc522585588)

[c. Null Values 14](#_Toc522585589)

[d. Outliers 15](#_Toc522585590)

[e. Correlation coefficient 16](#_Toc522585591)

[f. Feature Cass-wise Distribution 18](#_Toc522585592)

[8. Data Preparation 19](#_Toc522585593)

[a. Data with null values as NaN. 19](#_Toc522585594)

[b. Data with null values omitted. 20](#_Toc522585595)

[c. Data with null values Imputed. 20](#_Toc522585596)

[d. Data with numerical features imputed and scale between 0-1 20](#_Toc522585597)

[e. Data set with feature selection. 21](#_Toc522585598)

[f. Data sampled with balanced classes and stratification 21](#_Toc522585599)

[9. Empirical Results and their analysis 22](#_Toc522585600)

[a. Logistic Regression: 23](#_Toc522585601)

[b. LinearSVC: 23](#_Toc522585602)

[c. LightGBM 24](#_Toc522585603)

[10. Conclusion 25](#_Toc522585604)

[11. Future Work 26](#_Toc522585605)

[12. References / bibliography (\*\*WILL BE ADDED SOON\*\*) 27](#_Toc522585606)

[13. Appendices 27](#_Toc522585607)

# 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 UK, consumer credit stood at GBP 213.2 billion, as of 30 Jun 18. This has increased almost 37% in just 5 years (Bank of England, 2018). Monthly growth rate has gone as high as 10.9 percent (Bank of England, 2018) (Desjardins, 2018). 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 a consumer’s life. It estimates a person’s credit worthiness based on their 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 (Desjardins, 2018). 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 (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). 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 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 (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 performs on a certain type of data, and why. 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). 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 results of the mentioned algorithms on different versions of the same data set. These versions of data 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 data set. The aim is to identify how algorithm’s performance is affected by comparing their results with each other. 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 data set. Section 7, describes the dataset, exploratory data analysis conducted on the date and how outliers in the data set are identified and handled. Section 8, explains the various version of the data set 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.

# 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 (Credit scoring) (Hand & Henley., 1997). Much of research in this area explores the development, application and evaluation of credit scoring model for retail sector (Crook, et al., 2007). 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 (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. 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

### Logistic Regression (LR)

Logistic regression 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.

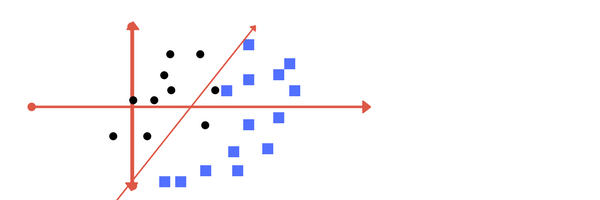
Logistic regression models the relationship between one or more dependent variables with an independent variable. The outputs are probability predictions restricted between 0 and 1.

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

The coefficients β are estimated during training that determines 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 is adequate 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.

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 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 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 optimization of each algorithm, 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. 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)

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. [13]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. Since, fine tuning any algorithm is not a concern for this project so we would use this algorithms with default parameters.

## 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.

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 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.

A close up of a logo

Description generated with very high confidence 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.

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.

## 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.

### 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 some 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.

### 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.

## Code Implementation

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

The code is organized in four python script files. General utility function 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, refer to Appendix at the end of this report.

# Exploratory Data Analysis (EDA)

## 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’. This file contains all the information filed by the company’s 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 to his loan repayment history. For this project, I will primarily use two files, ‘application\_train’, as it contains labelled data and the ‘application\_test’. 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 does not have labels, but it is still useful 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 and how they compare.

The reason for selecting this data set is that it contains the higher number of data points than the ones used in previous studies. These numbers are shown in the ‘Table 1’ below. A large data set 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 files description

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 as categorical types in this project but because the data set has them defined by integer that can take 0 or 1 (Boolean) values, I have mentioned them separately in table above.

## 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

A close up of a logo

Description generated with high confidence

Figure 3: 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.

## 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 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 instance.

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

## 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 data set, A screenshot of a cell phone

Description generated with very high confidencethe figure below shows the distribution of ‘DAYS\_EMPLOYED’ feature from the data.

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 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. The distribution after removal of these outlier instances is shown below.

A screenshot of a social media post

Description generated with very high confidence

Figure 5: 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:

A screenshot of a social media post

Description generated with very high confidence

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

A screenshot of a cell phone

Description generated with very high confidence

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

## 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 data set 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 |

From the table 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 practise 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 it is.

Correlation coefficient can also be used to compare the relation 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 affect will be seen on all ML algorithms I am analysing and thus, is not important for my analysis.

## 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 [10], coloured by the value of the target. It can be thought of as a smoothed histogram.

A close up of a map

Description generated with very high confidenceThe KDE plot of ‘DAYS\_BIRTH’ or age feature is shown below:

A close up of a map

Description generated with high confidenceSimilarly, the KDE plot of ‘EXT\_SOURCE\_3’ feature is shown below:

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 of 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 algorithms we are going to analyse.

# 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:

## Data with null values as NaN.

It would highlight how well the algorithm can perform with missing data. Real-world 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.

## 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 in to a trained model at run time, 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.

## 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 model 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.

## 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 data set that is imputed and then scaled between 0 and 1. This would show us how sensitive the algorithm is to the scale of features in the data. Comparing it with other version results we can conclude how it affects the performance of our algorithm.

## Data set 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 data set. It will not only reduce the dimensionality of the data but also the noise because only most 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, Chi2, Pearson correlation co-efficient), recursive feature elimination (with logistic regression as estimator) and model-based ranking (random forest ensemble). Features with 3 or more votes are added in selected feature lists.

## Data sampled with balanced classes and stratification

A variant of data set 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 data set 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 with comparing it with 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 data set. Next section states the results and analysis them.

# 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.

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 |

The table below shows the scores for each version against classification algorithm. Refer to table above 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 |

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 above will be analysed algorithm wise below.

## 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 logistic regression’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 means that LR 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.

## LinearSVC:

Despite using SGD for model training, LinearSVC took roughly 3x more time to complete. This means it will be a very bad choice for training on large data sets. 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 might 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 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 data sets, 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 lesser effect on LinearSVC than LR and so it can be more ideal for data with imbalance.

## 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. Imputing missing features and scaling numerical values would not improve the model’s performance. Thus, LGBM is sensitive to missing value although it is robust enough to perform despite them being present in the data set. Data imbalance also influences the results, but they are not as sensitive as LR but more than LinearSVC.

# Conclusion

The aim of this project was to conduct an in-depth analysis of ML algorithms with a single large data set to identify what characteristics of those classification algorithms and data set facilitate in improving performance. Using a single large data set ensures that core of the data does not change so our results are comparable.

I have used 17 different versions of the data set for comparison. Each version has a different strategy of pre-processing applied to it which changes the number of features or their types to observe the algorithms response to that change. Main characteristics highlighted in these variants are data quantity, quality (imputed and original), data sparsity (OHE data) and the types of features and their number.

My analysis is restricted to studying only three algorithms as more algorithms would require more time. Results show that LR, the industry standard (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 data set 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 data set and their scale as the algorithms general behaviour changes when numeric features scale is changed. For categorical features, 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 data set, 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 data set as its performance on OHE data set 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.

# Future Work

Availability of more data and the advancements in novel classification algorithms and the fact that there is not a clear winner amongst the algorithms for a particular data set suggests that it is research gap that would constantly need to be updated from time to time. 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 metrices should be used for the purpose.

This project can be considered as first step towards gaining insight in to the internal workings of the algorithms. More algorithms can be analysed in this manner and publicly available data sets from various domains can provide a clearer picture.

From application perspective, it is important to know why certain algorithms outperform others on a particular data set. 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 data set transforms in the future. Observing the discussions on Kaggle, 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 data set and how they are pre-processed. What this project proposes, is to develop a knowledge base that can be used to pick ML algorithms according to the nature of the problem, the data set available and then select pre-processing steps that would facilitate the most suitable ML algorithm.

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

# Appendices