Credit Default Prediction

AMOD: 5610

Aditya Chugh - 0718959

Yashaswini Reddy Terala - 0726670

Content:

Introduction:

The process of credit score prediction involves the use of statistical and machine learning techniques to forecast the credit score of an individual. This score is a numerical indication of their creditworthiness, which is determined based on their credit history and other financial activities. Credit bureaus typically generate credit scores using data from credit reports and other sources. To predict credit scores, lenders analyze a borrower's credit history and other financial information to create a statistical model that anticipates the probability of the borrower defaulting on a loan or making late payments. The model may take into account various factors such as payment history, credit utilization, length of credit history, and other relevant data.

Credit default prediction plays a vital role in assisting financial institutions in evaluating the risk involved in lending money to individuals or organizations. The credit default models enable lenders to estimate the probability of a borrower defaulting on their loan. This valuable insight enables lenders to make informed decisions about loan approvals, determining the loan amount, and setting interest rates.

Credit score predictive analysis is a vital instrument that helps lenders decide on loans and the conditions attached in advance. In order to determine the level of risk involved with lending money to that particular person, the analysis involves looking at the borrower's credit history and other financial information. Lenders use this data to decide whether or not to approve loan applications and how much interest to charge. By identifying borrowers who are more likely to fail and taking the necessary action, such as requiring a larger down payment or charging a higher interest rate, lenders can reduce the risk of loan defaults. Credit score predictive analysis can also enhance financial inclusion by extending credit to underserved populations, such as individuals with limited or no credit history, low income, or poor credit history. By using alternative data sources and machine learning algorithms, lenders can develop credit scores for these individuals and

provide them access to credit. Moreover, credit score predictive analysis can help prevent fraud by identifying potential instances of identity theft or financial fraud. By analyzing the borrower's credit history and other financial data, lenders can detect inconsistencies or suspicious activity that may indicate fraud and take measures to prevent it.

There are a lot of models that can be used to predict credit default and every model has different accuracy. After reviewing some articles and going through them, we decided to stick to four models as of now for the dataset that we have chosen. The Four models that we are planning to stick to are Logistic Regression, Random Forest classifier, XGBoost (Extreme Gradient Boost), and LGBM (light gradient boosting machine). After reviewing articles, it is expected that the accuracy of Linear Regression and Random Forest Classifiers will be low compared to XGBoost and LGBM. The reason why we are still using Linear Regression and Random Forest classifier even though the accuracy is expected to be less is that we would be able to compare different models and tell reasons why companies like AMEX and VISA are not sticking to these logistic or linear regression model and are using models like XGBoost or LGBM. In the literature review below articles have used Linear or logistic regression having an accuracy of around maybe 70% and those with XGBoost will be having it around 80%. A lot of articles use k-means or neural networks too. We are not sure as of now whether to include these algorithms in our project or not. Although the model accuracy achieved by Chen, & Zhang, R et al is about 82% for their credit default prediction by using the k-means algorithm. But a study by Jun Wang et al was able to achieve an accuracy of 97% using the XGBoost model. As a group project, we are planning to execute everything on Python, since we both are somehow familiar with Python. In the document below you can find the literature review, overview of the dataset description, methodology that we used, results, and Final conclusion of this project.

Literature Review

Khandani et al. (2010) developed nonlinear non-parametric forecasting methods to assess the risk of consumer credit using machine learning techniques. When assessing potential borrowers, lenders often utilise credit scoring or credit score predictive analytics. By giving an unbiased assessment of their credit history and risk indicators, it aids lenders in determining the creditworthiness of potential borrowers. Credit scoring aids lenders in determining which borrowers are less likely to be approved for loans and which ones are more likely to default on their debt. This supports improved decision-making and risk management on the part of lenders.[7]

XGBoost was described as a scalable machine learning end-to-end tree boosting method by Chen et al. (2016). The last few decades have seen a number of trials using different classifiers to predict default credit, according to Wirot et al. (2021). One classifier, XGBoost, had outstanding performance in predicting credit score. According to Jonah et al. (2022), a naive classifier can obtain high accuracy by categorising every example into the majority class; nevertheless, incorrectly categorising the minority class is frequently expensive. In order to learn from class-imbalanced data going forward, Jonah et al. (2022) suggested a number of XGBoost extensions.[11]

The LGBM is a decision tree-based gradient boosting architecture that is regarded as quick and high-performing. Experimental findings by Dasril et al. (2022) demonstrate that the LightGBM algorithm has been successfully enhanced ,with accuracy of LightGBM with ACO algorithm, LightGBM with BCO algorithm, and LightGBM without swarm algorithm, which is 95.64%, 94.70%, and 94.38%, respectively. This accomplishment also exhibits exceptional performance in loan default prediction and solid generalisations. [3]

Zhao et al. (2015) investigated the efficiency of calculating credit scores using a multi-layer perceptron (MLP) neural network. To train the model and determine its accuracy, the

authors employed a dataset of German credit. According to their findings, an MLP model with nine hidden units had a classification accuracy of 87%, which was better than other comparable studies[15]. Their research's findings supported the pattern of MLP models' scoring accuracy rising with more hidden units. According to research by Ampountolas et al. (2021) comparing several machine learning models, the random forest algorithmic method yields superior outcomes for easily accessible customer data.[1]

A set of tests were carried out by Zhu et al. (2019) utilising the logistic regression, decision tree, SVM, and random forest classifier. With a 98% accuracy rate, the random forest model surpassed logistic regression in forecasting loan default, while the other models had 95%, 75%, and 73% accuracy rates[16]. According to Tien et al. (2022), the trained data model for the LGBM model had a 98.21% accuracy rate[12]. On the other side, Dong et al. (2010) suggested using random coefficients to boost accuracy without giving up desirable features, with the drawback that the model requires more time to estimate parameters.[4]

Over the years, various statistical and machine learning techniques have been used to develop credit score prediction models. These models use historical data on borrower behavior to train the algorithm, which is then used to predict the creditworthiness of new applicants.

One of the earliest credit score prediction models was developed by Fair Isaac Corporation (FICO) in the late 1980s. The FICO score, which ranges from 300 to 850, is still widely used today by lenders and financial institutions to assess credit risk. The FICO score is calculated using a range of factors, including payment history, credit utilization, length of credit history, and other financial behaviors. Since the development of the FICO score, numerous other credit score prediction models have been developed. One such model is the VantageScore, which was developed by the three major credit bureaus (Equifax, Experian, and TransUnion) in 2006. The VantageScore ranges from 300 to 850,

and it uses a range of factors such as payment history, credit utilization, and credit mix to calculate the score.

More recently, machine learning algorithms have been used to develop credit score prediction models. Machine learning algorithms can analyze large volumes of data and identify patterns that may not be apparent to human analysts. These algorithms can also learn and adapt over time, improving the accuracy of the credit score prediction model.

One example of a machine learning algorithm used in credit score prediction is the decision tree algorithm. A decision tree is a model that uses a tree-like structure to represent a set of decisions and their possible consequences. The algorithm works by dividing the data into subsets based on a set of attributes, and then using a set of rules to predict the outcome. Another example of a machine learning algorithm used in credit score prediction is the neural network algorithm. A neural network is a model that simulates the structure and function of the human brain. The algorithm works by analyzing a large dataset and identifying patterns and relationships between variables. The algorithm then uses this information to predict the creditworthiness of new applicants.

In recent years, alternative data sources have also been used to develop credit score prediction models. Alternative data sources include data such as rental history, utility bill payments, and social media activity. These data sources can be particularly useful for individuals with limited credit history or no credit history. One of the challenges of credit score prediction is the potential for bias in the data. Bias can occur if the data used to train the algorithm is not representative of the population as a whole. For example, if the data used to train the algorithm is biased towards a particular demographic group, the algorithm may produce biased results. To address this challenge, researchers have developed methods to detect and correct bias in credit score prediction models. One

approach is to use a technique called fairness constraints, which adjusts the predictions to ensure that the algorithm produces fair and unbiased results.

Dataset Description:

We are using an AMEX dataset that has been provided by the company AMEX itself. The data that is provided by the AMEX is a competition dataset that they used for organizing Kaggle competition online. The objective of the dataset is to find the probability of a customer who will not pay back their credit card balance in the future based on their monthly customer profile. According to AMEX if the customer doesn't payback in 120 days it would be considered as default event.

The data consist of 190 columns with over 450K rows and the total data sum up to 65GB. The good part about this dataset is that AMEX has already provided training dataset and test dataset. The dataset contains following features for each customer.

D_*: Delinquency variables

S_*: Spend variables

P_*: Payment variables

B_*: Balance variables

R_*: Risk variables

There are a total of 190 variables in the dataset with approximately 450,000 customers in the training set and 925,000 in the test set.

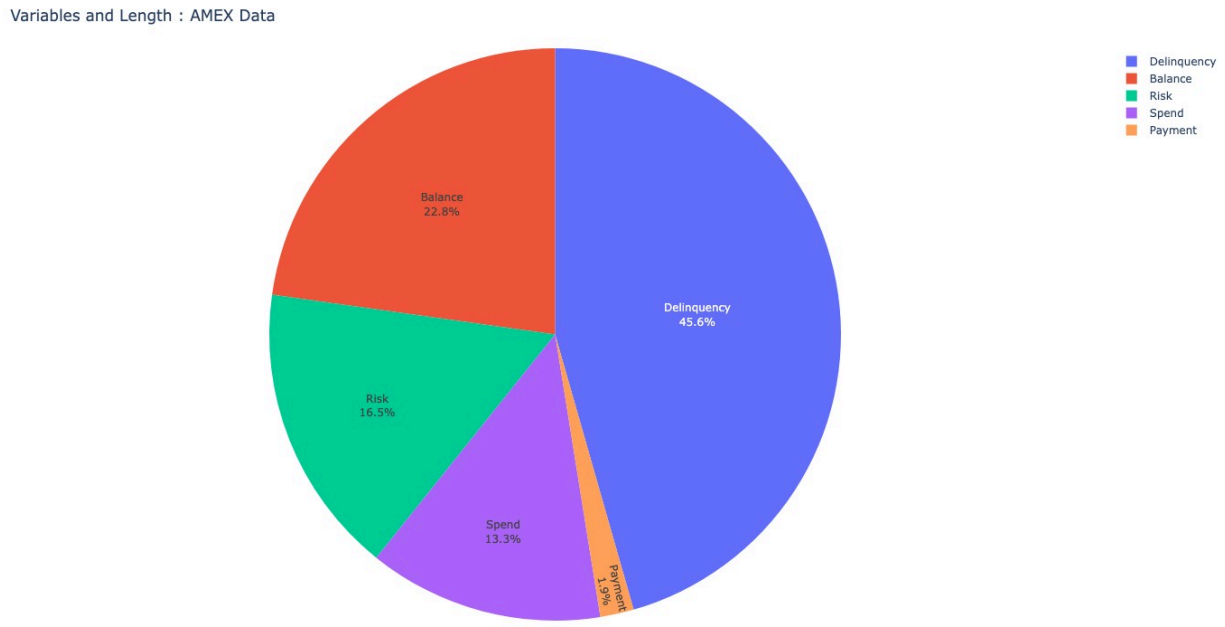

Fig1. Variables and its length

D_*: Delinquency variables

Delinquency variables are a type of credit scoring variable that measures a borrower's history of late or missed payments on their credit accounts. These variables are commonly used by lenders and credit bureaus to assess an individual's creditworthiness and the risk of default on a loan or credit account.

There are typically two types of delinquency variables used in credit scoring:

1. Days Past Due (DPD) is a variable that indicates how many days a borrower's credit account payment has been past due. A DPD of 30 indicates, for instance, that the borrower's payment is 30 days overdue. The probability of default increases with increasing DPD, and the borrower's credit score decreases as a result.
2. Delinquency Frequency: The frequency of a borrower's overdue or missing payments on their credit accounts is shown by the variable known as delinquency frequency. It is frequently stated as a percentage of accounts that have been past due for the previous 12 months. A 10% delinquency frequency, for instance, indicates that within the previous year,

10% of the borrower's accounts were past due. The likelihood of default increases with increasing delinquency frequency, and the borrower's credit score decreases.

S_*: Spend variables

1. Credit Card Spend: This variable measures the amount of money a borrower spends on their credit card accounts over a period of time. Credit card spend is an important factor in credit scoring because it can indicate a borrower's ability to manage credit responsibly. Borrowers who consistently spend within their means and pay off their credit card balances in full each month are generally considered lower risk than those who consistently carry high credit card balances.
2. Total Spend: This variable measures the total amount of money a borrower spends on all of their credit accounts, including credit cards, loans, and other types of credit. Total spend is an important factor in credit scoring because it can indicate a borrower's overall financial health and ability to manage debt. Borrowers who consistently spend more than they earn and carry high levels of debt are generally considered higher risk than those who live within their means and have low levels of debt.
3. Category Spend: This variable measures the amount of money a borrower spends in different spending categories, such as travel, dining, or entertainment. Category spend can provide additional insight into a borrower's spending behavior and help lenders and credit bureaus assess risk more accurately.

P_*: Payment variables

There are different types of payment variables that lenders and credit bureaus may consider when evaluating creditworthiness, including:

1. Payment History: This factor evaluates a borrower's track record of timely or irregular payments. One of the most crucial aspects of credit scoring is payment history because it gives information about a borrower's capacity for responsible credit management. In general, lenders view borrowers who consistently pay their credit accounts on time as being less risky than those who have a history of missed or late payments.
2. Payment Amount: This variable measures the amount of money a borrower pays on their credit accounts each month. Payment amount can provide insight into a borrower's ability to manage debt and make payments within their means.
3. Payment Frequency: This variable measures the frequency of payments made by a borrower on their credit accounts. Borrowers who make frequent payments on their credit accounts are generally considered lower risk than those who make infrequent payments.

B_*: Balance variables

Different types of balance variables are:

1. Credit Card Balance: This variable measures the amount of debt a borrower has on their credit card accounts. High credit card balances can indicate that a borrower is using credit excessively and may have difficulty making payments on time.
2. Total Balance: This variable measures the total amount of debt a borrower has across all of their credit accounts, including credit cards, loans, and other types of credit. High total balances can indicate that a borrower is carrying a significant amount of debt and may have difficulty managing their payments.
3. Balance-to-Limit Ratio: This variable measures the ratio of a borrower's credit card balance to their credit limit. This ratio is important because it provides insight into a borrower's credit utilization, or the amount of available credit that they are using. Borrowers with high balance-to-limit ratios may be considered higher risk because they are using a significant portion of their available credit.

R_*: Risk variables

1. Credit Score: A credit score represents creditworthiness. Credit score depends on variables like utilization, payment history, number of credit cards, history of credit, and type of credit.

If a person has a higher credit score than that person will have a lower risk compared to someone who has a low credit score.

2. Delinquency: Delinquency variables, as we discussed earlier, measure the number of payments a borrower has missed or made late. Borrowers who have a history of delinquent payments are considered higher risk because they may be more likely to default on their loans or credit accounts.
3. Bankruptcy: Bankruptcy variables measure whether a borrower has filed for bankruptcy in the past. Borrowers who have a history of bankruptcy may be considered higher risk because they have a history of financial hardship that could impact their ability to make payments.
4. Income: Income variables measure a borrower's income level. Borrowers with higher incomes may be considered lower risk because they have more resources available to make payments on their loans and credit accounts.
5. Employment: Employment variables measure a borrower's employment status and history. Borrowers with stable employment may be considered lower risk because they have a reliable source of income to make payments.

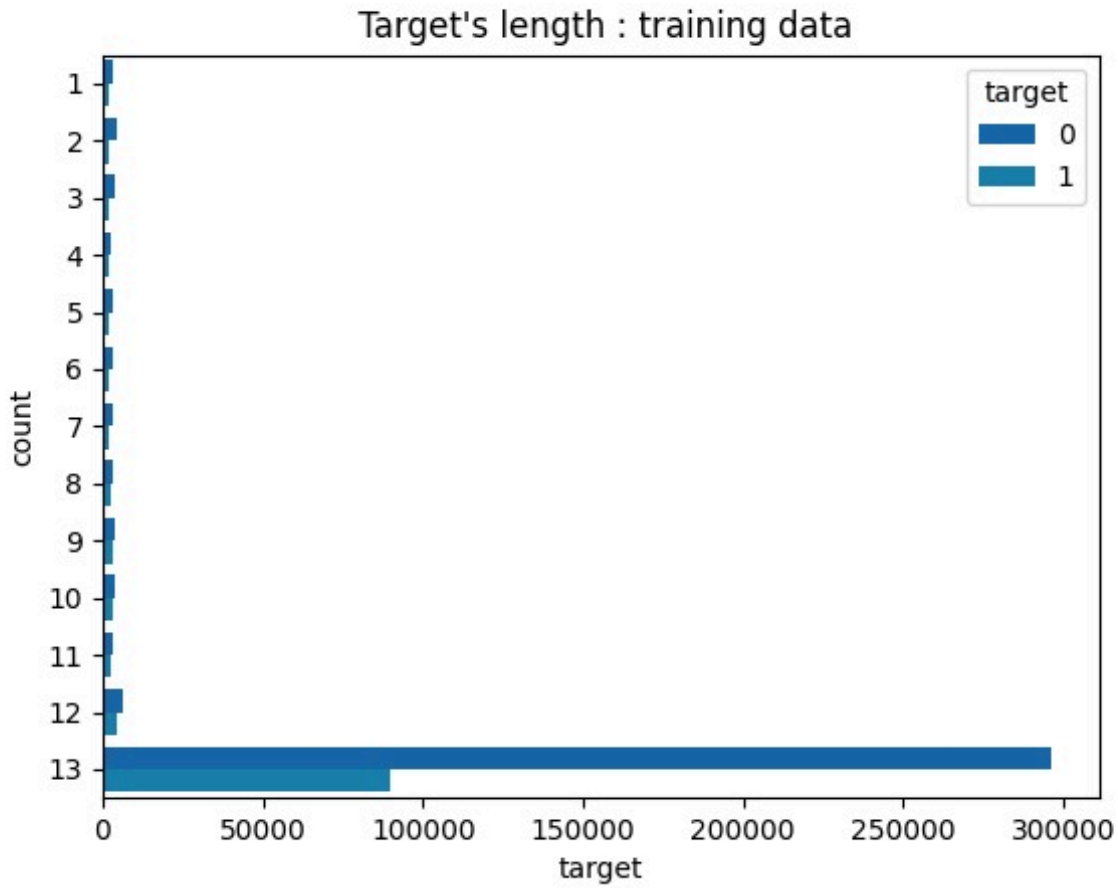

Fig2. Number of Defaulters in 13 cycles

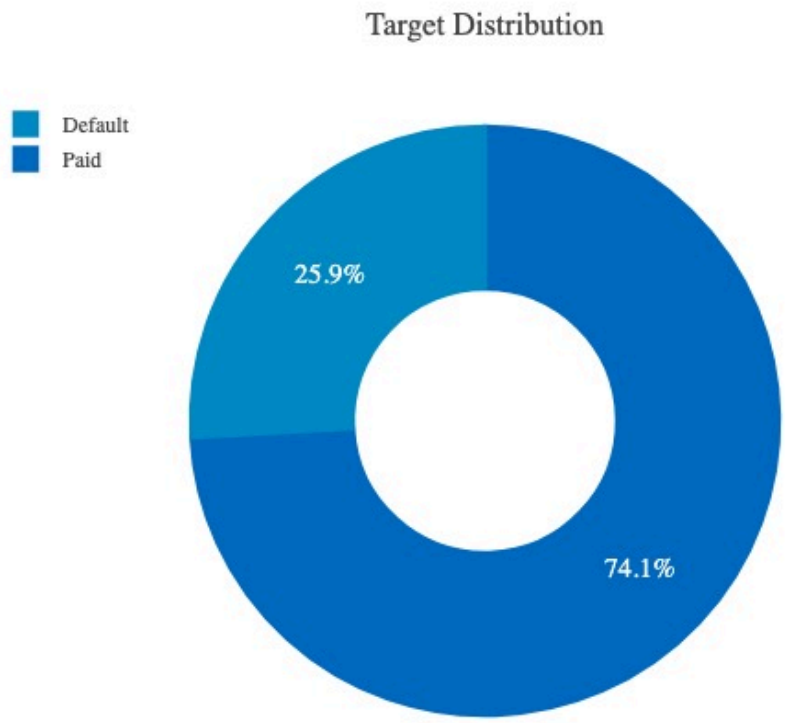

Fig.3 Average Default Target Distribution

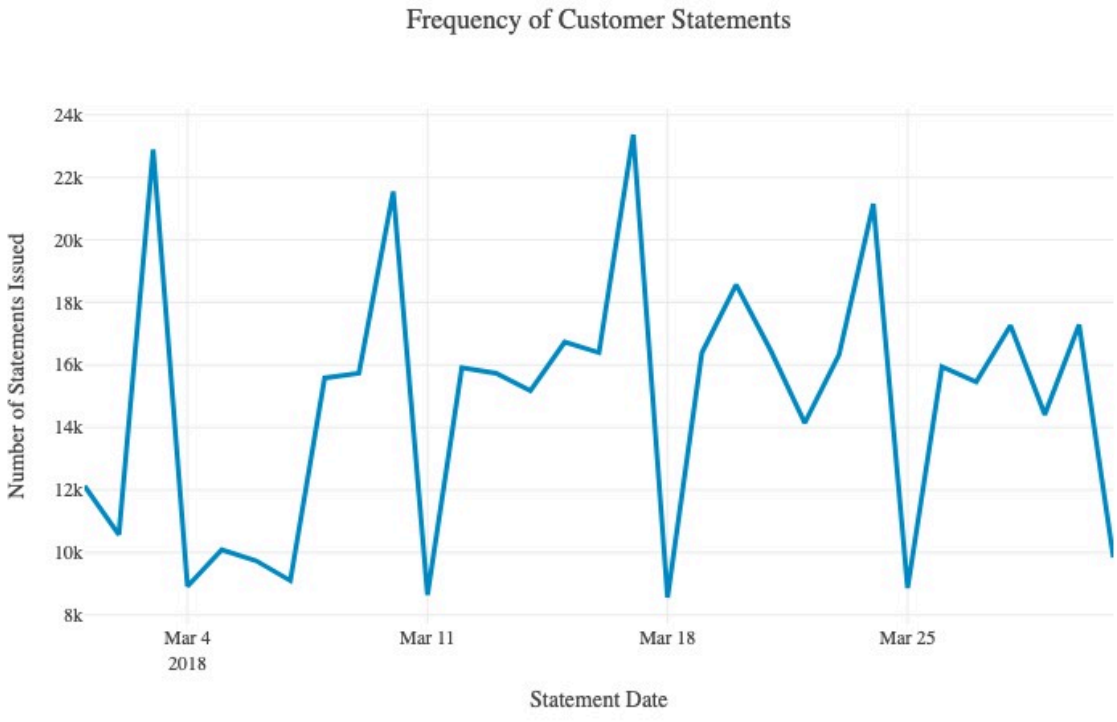

Fig.4 Statement issue date

Methodology:

We are using XGBoost, LGBM, Logistic Regression and Random Forest classifier for our credit default prediction. Other options than these would be neural network models, k-means algorithm. Why we are not using k-means or any other highly efficient model is because we want to do a comparison of low performing model against high performance model. XGBoost and LGBM are some of the most used industry models and are currently in use. When we compare it with logistic regression or even with k-means we could be able to tell why companies like AMEX or VISA are sticking to these high accurate models rather than other models which doesn't have a high accuracy. Currently we are sticking to four models but in future we might add k-means algorithm too if we are able to finish above four models in the limited time.

XGBoost

1. The fact that XGBoost can handle a variety of data types, including categorical and numerical data, is one of its main advantages. Since that credit datasets frequently include a mixture of both sorts of data, this is significant for credit scoring. Examples of numerical variables found in credit history data include credit limit or amount, as well as categorical variables like credit card type or payment frequency. XGBoost is a flexible method for credit rating because it can handle both types of data.
2. Another advantage of XGBoost is its ability to handle missing data. In credit scoring, missing data can be a common issue due to incomplete credit reports or borrower information. XGBoost has built-in functionality to handle missing data, allowing credit card companies to use more of the available data for prediction.

3. XGBoost is also effective in handling imbalanced datasets, which are common in credit scoring. Imbalanced datasets occur when the number of positive outcomes (e.g.,

borrowers who pay on time) is much smaller than the number of negative outcomes (e.g., borrowers who default). XGBoost can handle imbalanced datasets by using techniques such as weighted loss functions or subsampling.

LGBM:

1. Speed and Efficiency: LightGBM is known for its speed and efficiency, making it a good choice for large-scale credit scoring applications. LightGBM uses a novel algorithm to split the data and construct the decision trees, which allows it to handle large datasets efficiently.
2. High Accuracy: LightGBM is designed to produce accurate and reliable predictions, which is important in credit scoring applications. It uses gradient boosting to optimize the loss function and create a strong ensemble of decision trees.
3. Handling Categorical Features: LightGBM has built-in functionality to handle categorical features, which are common in credit scoring datasets. This allows the algorithm to effectively utilize all of the available data, leading to better predictive performance.
4. Handling Imbalanced Data: As with XGBoost, LightGBM is effective at handling imbalanced data, which is common in credit scoring. It can use various techniques such as weighted sampling or boosting to handle imbalanced datasets and improve the accuracy of predictions.

Logistic Regression:

For binary classification issues, where the objective is to predict one of two possible outcomes (for example, default or non-default) based on a collection of input factors, logistic regression is a frequently used statistical model. Although logistic regression can be used for credit scoring, it is less suited for this application than other machine learning methods due to a number of constraints.

First, logistic regression assumes a linear relationship between the input variables and the predicted outcome. However, credit scoring datasets often contain complex and nonlinear relationships between variables, which can lead to poor performance with logistic regression.

Second, logistic regression does not handle interactions between variables very well. In credit scoring, interactions between variables can be important predictors of default risk. For example, the combination of high debt-to-income ratio and low credit score may increase the likelihood of default. Logistic regression may not capture these interactions effectively, leading to less accurate predictions.

Third, logistic regression assumes that the data is linearly separable, which means that there is a clear boundary between the two classes being predicted. However, credit scoring datasets may contain overlapping or ambiguous data points, making it difficult to separate the two classes accurately.

Fourth, logistic regression can be sensitive to outliers and missing data. Credit scoring datasets may contain missing data or outliers, which can impact the accuracy of logistic regression predictions.

Overall, while logistic regression can be used for credit scoring, its limitations make it less suitable for this application than other machine learning algorithms such as decision trees, random forests, XGBoost, or LightGBM, which are better able to handle complex relationships between variables, interactions, nonlinearities, and large datasets.

Random Forrest Classifier:

Despite its strengths, there are some potential drawbacks to using random forest for credit default prediction. One limitation is that it can be computationally intensive, especially when dealing with very large datasets or a large number of input variables. Another potential issue is that random forest can be difficult to interpret, especially when

there are many trees in the model. This can make it challenging to explain the results of the model to stakeholders, such as lenders or regulators.

Overall, random forest is a powerful and widely used machine learning algorithm for credit default prediction. However, as with any modeling technique, it is important to carefully evaluate its performance on specific datasets and consider the tradeoffs between accuracy, interpretability, and computational complexity.

Results:

The evaluation findings on the various metrics employed by this arrangement of machine learning models are summarized in the above images. There are 183,566 support data points in total for this test run, with 136,110 for class 0 and 47,456 for class 1 for all the models constructed. According to the findings, the LGBM model had the highest accuracy of 97.11%, while the XGBoost tests produced accuracy levels above 90%. The XGBoost model comes in second place with 92.50. The random forest comes in second with 88.45% accuracy, followed by logistic regression with 89.6. We achieve 99% accuracy on the random forest classifier when n estimators is 300. However, because there are missing or null values and because logistic regression may not be able to detect non-linear correlations between the characteristics and the target variable, it may not meet industry standards. One metric for assessing classification model performance is accuracy. Accuracy is the percentage of forecasts that our models accurately predicted, to put it simply. A prediction accuracy of 97% demonstrates the model's outstanding performance. Precision is often the real percentage of accurate identifications. With a precision of 0.80, our XGBoost model predicts events with absolute certainty 80% of the time. Random forest models have the lowest precision (0.79), but logistic regression models have the best precision (0.81). The fact that both models have a low recall value when tested, however, indicates that models do not fare well with prediction. Overall, the accuracy, precision, and recall ratings of the LGBM model are respectable.

```
[1]XGB Classifier Accuracy: 0.9250654628523282
[2]LGBM Accuracy: 0.9711309729178091
[3]Logistic regression Accuracy: 0.8964470286583838
[4]Random forest Accuracy: 0.9964953313455386
```

References:

[1] Apostolos Ampountolas, Titus Nyarko Nde, Paresh Date, & Corina Constantinescu. (2021). A Machine Learning Approach for Micro-Credit Scoring. Risks (Basel), 9(3), 50–. https://doi.org/10.3390/risks9030050

[2] Ying Chen, & Ruirui Zhang. (2021). Research on Credit Card Default Prediction Based on k-Means SMOTE and BP Neural Network. Complexity (New York, N.Y.), 2021. https://doi.org/10.1155/2021/6618841

[3] Dasril, Yosza & Sam'an, Muhammad & Ifriza, Yahya & Muslim, Much. (2022). An improved light gradient boosting machine algorithm based on swarm algorithms for predicting loan default of peer-to-peer lending. Indonesian Journal of Electrical Engineering and Computer Science. 28. 1002-1011. 10.11591/ijeecs.v28.i2.pp1002-1011.

[4] Dong, Lai, K. K., & Yen, J. (2010). Credit scorecard based on logistic regression with random coefficients. Procedia Computer Science, 1(1), 2463–2468. https://doi.org/10.1016/j.procs.2010.04.278

[5] Gao, Sun, W., & Sui, X. (2021). Research on Default Prediction for Credit Card Users Based on XGBoost-LSTM Model. Discrete Dynamics in Nature and Society, 2021, 1–13. https://doi.org/10.1155/2021/5080472

[6] Mushava, & Murray, M. (2022). A novel XGBoost extension for credit scoring class-imbalanced data combining a generalized extreme value link and a modified focal loss function. Expert Systems with Applications, 202, 117233–. https://doi.org/10.1016/j.eswa.2022.117233

[7] Khandani, Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. Journal of Banking & Finance, 34(11), 2767–2787. https://doi.org/10.1016/j.jbankfin.2010.06.001

[8] Khemais, Nesrine, D., & Mohamed, M. (2016). Credit Scoring and Default Risk Prediction: A Comparative Study between Discriminant Analysis & Logistic Regression. International Journal of Economics and Finance, 8(4), 39–. https://doi.org/10.5539/ijef.v8n4p39

[9] Liu, Zhang, S., & Fan, H. (2022). A two-stage hybrid credit risk prediction model based on XGBoost and graph-based deep neural network. Expert Systems with Applications, 195, 116624–. https://doi.org/10.1016/j.eswa.2022.116624

[10] Sohn, & Kim, H. S. (2007). Random effects logistic regression model for default prediction of technology credit guarantee fund. European Journal of Operational Research, 183(1), 472–478. https://doi.org/10.1016/j.ejor.2006.10.006

[11] Chen, & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

[12] H. L. Quang Tien, L. Quang Tran and T. Hop Do, An Empirical Study on Bankruptcy Prediction using Ensemble Learning,2022 RIVF International Conference on Computing and Communication Technologies (RIVF), Ho Chi Minh City, Vietnam, 2022, pp. 173-178, doi: 10.1109/RIVF55975.2022.10013848.

[13] Jun Wang, Wei Rong, Zhuo Zhang, & Dong Mei. (2022). Credit Debt Default Risk Assessment Based on the XGBoost Algorithm: An Empirical Study from China. Wireless Communications and Mobile Computing, 2022. https://doi.org/10.1155/2022/8005493

[14] Wirot Y., Pakaket W., & Anongnart Srivihok (2021). Improved credit scoring model using XGBoost with Bayesian hyper-parameter optimization. Int J Elec & Comp Eng, Vol. 11, No. 6, December 2021: 5477 - 5487. https://doi.org/10.11591/ijece.v11i6.pp5477-5487

[15] Zhao, Xu, S., Kang, B. H., Kabir, M. M. J., Liu, Y., & Wasinger, R. (2015). Investigation and improvement of multi-layer perceptron neural networks for credit scoring. Expert Systems with Applications, 42(7), 3508–3516. https://doi.org/10.1016/j.eswa.2014.12.006

[16] Zhu, Qiu, D., Ergu, D., Ying, C., & Liu, K. (2019). A study on predicting loan default based on the random forest algorithm. Procedia Computer Science, 162, 503–513. https://doi.org/10.1016/j.procs.2019.12.017

Appendix:

```
########################## CREDIT DEFAULT PREDICTION ##########################
################################## TEAM 08 ####################################


########## ADITYA CHUGH
########## YASHASWINI REDDY TERALA

########## IMPORTING NECESSARY LIBRARIES

######## BASIC LIBRARIES

import numpy as np
import pandas as pd
import seaborn as sns
import gc

######## VISUALIZATION TOOLS

import matplotlib.pyplot as plt
import matplotlib.colors
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

######## DATA PROCESSING LIBRARIES

from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer

######## MODELING LIBRARIES

#### DISPLAYING SAMPLE DATA

print(training_data.head(5))

def data_report(data):
    data_col = []
    data_type = []
    data_uniques = []
    data_n_uniques = []

    for i in data.columns:
        data_col.append(i)
        data_type.append(data[i].dtypes)
        data_uniques.append(data[i].unique()[:5])
        data_n_uniques.append(data[i].nunique())

    return pd.DataFrame({'Column': data_col,
                         'd_type': data_type,
                         'unique_sample': data_uniques,
                         'n_uniques': data_n_uniques})

# show unique samples of the data

print(data_report(training_data))

def information(a,b):
    a = a
    b = b

    print("\nUNDERSTANDING THE CUSTOMER DATA\n")

    print("\nTRAIN DATA SET INFORMATION\n")
    print(a.info())
    print("TEST DATA SET INFORMATION\n")
    print(b.info())

    print("\nTRAIN DATA SET SHAPE\n")
    print(a.shape)
    print("TEST DATA SET SHAPE\n")
    print(b.shape)

    del_var = [c for c in a.columns if c.startswith('D_')]
    spend_var = [c for c in a.columns if c.startswith('S_')]
    pay_var = [c for c in a.columns if c.startswith('P_')]
    bal_var = [c for c in a.columns if c.startswith('B_')]
    risk_var = [c for c in a.columns if c.startswith('R_')]
    if len(del_var) != 0:
        print(f'\nNumber of Delinquency variables: {len(del_var)}')
        print(f'Number of Spend variables: {len(spend_var)}')
        print(f'Number of Payment variables: {len(pay_var)}')
        print(f'Number of Balance variables: {len(bal_var)}')
        print(f'Number of Risk variables: {len(risk_var)}')

    values= [len(del_var), len(spend_var),len(pay_var), len(bal_var),len(risk_var)]

    return values


values = information(training_data, testing_data)
```

```
,
########## DATA CLEANING

columns = training_data.columns[(training_data.isna().sum()/len(training_data))*100>30]

train_data = training_data.drop(columns, axis=1)
test_data = testing_data.drop(columns, axis=1)

values =information(train_data, test_data)

########## EXPLORATORY DATA ANALYSIS

#colors = ['#0b1d78', '#0045a5', '#0069c0', '#008ac5', '#00a9b5', '#00c698', '#1fe074']

labels=['Delinquency', 'Spend','Payment','Balance','Risk']
values = values
fig = px.pie(train_data, values=values, names=labels)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title= " Variables and Length : AMEX Data")
fig.show()

unique_cx_train = len(train_data.groupby("customer_ID")['customer_ID'].count())
print("\nCount of unique customers in training data: ", unique_cx_train)
unique_cx_test = len(test_data.groupby("customer_ID")['customer_ID'].count())
print("\nCount of unique customers in testing data: ", unique_cx_test)

train_data.groupby("customer_ID").size()

cx_prof = train_data.groupby("customer_ID")['customer_ID'].count().values
cx_test = test_data.groupby("customer_ID")['customer_ID'].count().values

fig = go.Figure()
fig.add_trace(go.Histogram(
    y = cx_prof,
    ybins = dict(size = 0.6),
    marker_color= '#0069c0'))
fig.update_layout(
    template = "plotly_dark",
    title = " Customer profile count : training data",
    yaxis_title = "Number of months",
    bargap = 0.2
)
fig.show()

fig = go.Figure()
fig.add_trace(go.Histogram(
    y = cx_test,
    ybins = dict(size = 0.6),
    marker_color= '#0069c0'))
fig.update_layout(
    template = "plotly_dark",
    title = " Customer profile count : test data",
    yaxis_title = "Number of months"
)
fig.show()

#From here we can see the dsitribution of profile length is common between train and test data.

del cx_prof
del cx_test
gc.collect()

count = train_data.groupby("customer_ID")['customer_ID'].count()
target_data = pd.DataFrame({"customer_ID":count.index, "count": count.values})
target_data = target_data.merge(train_labels, on='customer_ID', how='left')

sns.countplot(data = target_data,
              y='count',hue='target',
              orient='h',
              palette = ['#0069c0','#008ac5']).set(title=" Target's length : training data", xlabel='target')
plt.show()

#nearly 30 - 50 % of all profile length has target 1 (default)
#Can't get a great correlation between profile length and target -- but thinking like keeping this information may help.


del target_data
gc.collect()

temp=dict(layout=go.Layout(font=dict(family="Franklin Gothic", size=12),
                           height=500, width=1000))

train = train_data.groupby('customer_ID').tail(1).set_index('customer_ID')
print("\nThe training data begins on {} and ends on {}.".format(train['S_2'].min().strftime('%m-%d-%Y'),train['S_2'].max().strftime('%m-%d-%Y')))
print("\nThere are {:,.0f} customers in the training set and {} features.".format(train.shape[0],train.shape[1]))

test = test_data.groupby('customer_ID').tail(1).set_index('customer_ID')
print("\nThe test data begins on {} and ends on {}.".format(test['S_2'].min().strftime('%m-%d-%Y'),test['S_2'].max().strftime('%m-%d-%Y')))
print("\nThere are {:,.0f} customers in the test set and {} features.".format(test.shape[0],test.shape[1]))


#There are 5.5 million rows for training and 11 million rows of test data.


del test['S_2']
gc.collect()

titles=['Delinquency '+str(i).split('_')[1] if i.startswith('D') else 'Spend '+str(i).split('_')[1]
        if i.startswith('S') else 'Payment '+str(i).split('_')[1]  if i.startswith('P')
        else 'Balance '+str(i).split('_')[1] if i.startswith('B') else
         'Risk '+str(i).split('_')[1] for i in train.columns[:-1]]
cat_cols=['Balance 30', 'Balance 38', 'Delinquency 63', 'Delinquency 64', 'Delinquency 66', 'Delinquency 68',
          'Delinquency 114', 'Delinquency 116', 'Delinquency 117', 'Delinquency 120', 'Delinquency 126', 'Target']
test.columns=titles[1:]
titles.append('Target')
train.columns=titles

target=train.Target.value_counts(normalize=True)
target.rename(index={1:'Default',0:'Paid'},inplace=True)
pal, color=['#0069c0', '#008ac5'], ['#0069c0', '#008ac5']



fig=go.Figure()
fig.add_trace(go.Pie(labels=target.index, values=target*100, hole=.45,
                     showlegend=True,sort=False,
                     marker=dict(colors=color,line=dict(color=pal,width=2.5)),
                     hovertemplate = "%{label} Accounts: %{value:.2f}%<extra></extra>"))
fig.update_layout(template=temp, title=' Target Distribution',
                  legend=dict(traceorder='reversed',y=1.05,x=0),
                  uniformtext_minsize=15, uniformtext_mode='hide',width=700)
fig.show()


target=pd.DataFrame(data={'Default':train.groupby('Spend 2')['Target'].mean()*100})
target['Paid']=np.abs(train.groupby('Spend 2')['Target'].mean()-1)*100
rgb=['rgba'+str(matplotlib.colors.to_rgba(i,0.7)) for i in pal]
fig=go.Figure()
fig.add_trace(go.Bar(x=target.index, y=target.Paid, name='Paid',
                     text=target.Paid, texttemplate='%{text:.0f}%',
                     textposition='inside',insidetextanchor="middle",
                     marker=dict(color=color[0],line=dict(color=pal[0],width=1.5)),
                     hovertemplate = "<b>%{x}</b><br>Paid accounts: %{y:.2f}%"))
fig.add_trace(go.Bar(x=target.index, y=target.Default, name='Default',
                     text=target.Default, texttemplate='%{text:.0f}%',
                     textposition='inside',insidetextanchor="middle",
                     marker=dict(color=color[1],line=dict(color=pal[1],width=1.5)),
                     hovertemplate = "<b>%{x}</b><br>Default accounts: %{y:.2f}%"))
fig.update_layout(template=temp,title=' Distribution of Default by Day',
                  barmode='relative', yaxis_ticksuffix='%', width=1400,
                  legend=dict(orientation="h", traceorder="reversed", yanchor="bottom",y=1.1,xanchor="left", x=0))
fig.show()


plot_df=train.reset_index().groupby('Spend 2')['customer_ID'].nunique().reset_index()
fig=go.Figure()
fig.add_trace(go.Scatter(x=plot_df['Spend 2'],
                         y=plot_df['customer_ID'], mode='lines',
                         line=dict(color=pal[1], width=3),
                         hovertemplate = ''))
fig.update_layout(template=temp, title=" Frequency of Customer Statements",
                  hovermode="x unified", width=800,height=500,
                  xaxis_title='Statement Date', yaxis_title='Number of Statements Issued')
fig.show()
del train['Spend 2']

########## DATA PREPROCESSING

train_df =train.groupby('customer_ID').tail(1)
test_df = test.groupby('customer_ID').tail(1)

train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

information(train_df, test_df)

#getting numerical and categorical column names
num_cols = train_df.select_dtypes([np.int64,np.float16]).columns
cat_cols = ['Delinquency 63', 'Delinquency 64', 'Delinquency 68', 'Balance 30',
            'Balance 38', 'Delinquency 114', 'Delinquency 116', 'Delinquency 117',
            'Delinquency 120', 'Delinquency 126']
```

```
#Encoding categorical columns

label_enc = LabelEncoder()
for categorical in cat_cols:
    train_df[categorical] = label_enc.fit_transform(train_df[categorical])
    test_df[categorical] = label_enc.transform(test_df[categorical])

train_df = train_df.replace(np.nan, 0)

########## DATA VISUALIZATION & MODEL BUILDING

#defining X and Y data
#X,Y=train_df.drop(['customer_ID','Target','Spend 2'],axis=1),train_df['Target']
X,Y=train_df.drop(['Target'],axis=1),train_df['Target']

#making sure we have same columns in test data as in train
col=[c for c in X.columns]
test_x=test_df[col]

del test_df,train,test,train_df
gc.collect()

# getting data ready to build model
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.40,
                                                    random_state=30,
                                                    shuffle=True)


categorical_features_indices=[]
for c in cat_cols:
    a=X_train.columns.get_loc(c)
    categorical_features_indices.append(a)

del X,Y
gc.collect()

print(f"""
X_train shape: {X_train.shape}
X_test shape: {X_test.shape}
y_train shape: {Y_train.shape}
y_test shape: {Y_test.shape}
""")

# Create our imputer to replace missing values with the mean e.g.
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
imp = imp.fit(X_train)

# Impute our data, then train
X_train = pd.DataFrame(imp.transform(X_train))

def models(X_train,Y_train):

  XGB = XGBClassifier(n_estimators=300, max_depth=6, learning_rate=0.1)
  XGB.fit(X_train, Y_train)

  LGBM = LGBMClassifier(n_estimators=1000,boosting_type ='gbdt',max_depth=6,
                        learning_rate=0.1,num_leaves= 50,max_bins= 500)
  LGBM.fit(X_train, Y_train)

  logreg = LogisticRegression(solver = 'liblinear',random_state=0)
  logreg.fit(X_train, Y_train)

  random= RandomForestClassifier(n_estimators=200,max_features='sqrt',
                                 bootstrap=True,max_depth=6,min_samples_leaf=1,
                                 min_samples_split=5,n_jobs=-1)
  random.fit(X_train, Y_train)


  #print model accuracy on the training data.
  print('[1]XGB Classifier Accuracy:', XGB.score(X_train, Y_train))
  print('[2]LGBM Accuracy:', LGBM.score(X_train, Y_train))
  print('[3]Logistic regression Accuracy:', logreg.score(X_train, Y_train))
  print('[4]Random forest Accuracy:', random.score(X_train, Y_train))

  return XGB, LGBM, logreg, random

model = models(X_train,Y_train)

for i in range(len(model)):
  print()
  print(model)
  #Check precision, recall, f1-score
  print("\nclassification report:" )
  print( classification_report(Y_test, model[i].predict(X_test)) )
  print("\nconfusion matrix:")
  print(confusion_matrix(Y_test, model[i].predict(X_test)))
  #Another way to get the models accuracy on the test data
  print("\nAccuracy score:")
  print( accuracy_score(Y_test, model[i].predict(X_test)))
  pre_score = precision_score(Y_test, model[i].predict(X_test))
  print("\nprecision: ",pre_score)
  rec_score = recall_score(Y_test, model[i].predict(X_test))
  print("\nrecall: ",rec_score)
  f_score = f1_score(Y_test, model[i].predict(X_test), average = 'weighted')
  print("\nf1_score: ",f_score)
  print()#Print a new line

  false, true, thresholds = roc_curve(Y_test, model[i].predict(X_test))
  roc_auc = auc(false, true)
  display = RocCurveDisplay(fpr=false, tpr=true, roc_auc=roc_auc)
  display.plot()
  tot_pos = (Y_test == 1).sum()
  tot_neg = (Y_test == 0).sum() * 20
  fourth = int(0.04 * (tot_pos + tot_neg))
  plt.plot(false, true, color = '#0069c0',lw=3)
  plt.fill_between(false, true, color = '#00a9b5')
  plt.plot([0, 1], [0, 1], color="navy", lw=1, linestyle="--")
  plt.plot([fourth / tot_neg, 0], [0, fourth / tot_pos],
           color="green", lw=3, linestyle="-")
  fourth_index = np.argmax((false * tot_neg + true * tot_pos >= fourth))
  plt.scatter([false[fourth_index]],[true[fourth_index]],
              s=100) # intersection of roc curve with four percent line
  plt.gca().set_aspect('equal')
  plt.xlim([0.0, 1.0])
  plt.ylim([0.0, 1.0])
  plt.xlabel("False Positive Rate")
  plt.ylabel("True Positive Rate (Recall)")
  plt.title("Receiver operating characteristic")
  plt.show()
```