# Introduction

Starting a business is a tedious task that only the brave ones survive in the long run. The Small Business Association (SBA) in the US grants loans to small or start-up businesses to help them especially during its foundational years. However, loaning to small businesses poses a higher risk to financial institutions or organizations as the future of the business is highly uncertain. Thus, despite the mission to foster growth in the entrepreneurial space and allow more jobs to be created, there is a need for a strong credit risk assessment.

The creation of a model on how loans are granted benefits the following parties: 1) Banks or Financial Institutions, and 2) Businesses that are applying for a loan. The banks could use this model in order to improve their approval process by looking at the patterns of how they approve loans and easily see the signs (or rather features in this case) of what constitutes a loan approval. From the given set of demographic information and historical transactions, we can identify which among these features contribute largely to the approval or denial of loan grants.

The banks could also improve by narrowing down the features when a loan is defaulted despite being approved. For businesses that are trying to apply for a loan, they could easily learn what the banks look at when they apply for a loan. Furthermore, the data contains industries and states the businesses belong to which can give us an idea on the businesses that are likely to default and get disapproved. This clears up a layer of ambiguity between the banks and the people they are trying to serve. In addition, this also allows the bank to prevent selection bias on which business, state or industry they prefer to serve.

# The Dataset

The dataset retrieved from Kaggle (https://www.kaggle.com/datasets/mirbektoktogaraev/should-this-loan-be-approved-or-denied?select=SBAnational.csv) is from the U.S. Small Business Administration (SBA) which contains a list of small businesses that applied for a grant loan. This dataset contains the following columns:

Table 1. SBA Columns

| Column | Data Type | Description |
|---|---|---|
| LoanNr_ChkDgt | Object | Unique Identifier or Primary Key |
| Name | Object | Borrower Name |
| City | Object | Borrower City |

| | | |
|---|---|---|
| State | Object | Borrower State |
| Zip | Object | Borrower Zip Code |
| Bank | Object | Bank Name |
| BankState | Object | Bank State |
| NAICS | Object | North American Industry Classification Code System |
| ApprovalDate | DateTiime | SBA Commitment Issue Date |
| ApprovalFY | Object | Fiscal Year of Commitment |
| Term | Integer | Loan Term in months |
| NoEmp | Integer | Number of Business Employees |
| NewExist | Integer | 1 = Existing Business, 2 = New Business |
| CreateJob | Integer | Number of jobs created |
| RetainedJob | Integer | Number of jobs retained |
| FranchiseCode | Integer | 0 or 1 = No Franchise, otherwise Franchise |
| UrbanRural | Object | 1 = Urban, 2 = Rural, 0 = Undefined |
| RevLineCr | Object | With Revolving Line of Credit: Y = Yes, N = No |
| LowDoc | Object | LowDoc Loan Program: Y = Yes, N = No |
| ChgOffDate | DateTime | Date when the loan is declared default |
| DisbursementDate | DateTime | Disbursement Date |
| DisbursementGross | Float | Amount disbursed |
| BalanceGross | Float | Gross Amount of Outstanding Balance |
| MIS_Status | Object | CHGOFF = Charge Off, PIF = Paid In Full |
| ChgOffPrinGr | Float | Charged-Off Amount |

| | | |
|---|---|---|
| GrAppv | Float | Loan Gross Amount Approved by the Bank |
| SBA_Appv | Float | SBA's Guaranteed Amount of Loan approved |
| Real Estate | Integer | 1 = Backed By Real Estate, 0 = Not Backed By Real Estate |
| Recession | Integer | 1 = Loan was Active during Great Recession, 0 = No |
| Industry | Object | Derived from NAICS, updated NAICS code to Industry name tagging |
| IndustryDefRate | Float | Loan default rates per Industry |

The column MIS_Status is used as the target variable in which PIF is converted to 1 and CHGOFF is converted to 0. Since the dataset was accompanied by a case study, PIG values represented by 1 are described as "to grant loan" since PIF means Paid In Full Grant and those businesses are less likely to default on their loans. Meanwhile, CHGOFF values represented by 0 are described as "not to grant loan" since CHGOFF means Charged Off or Defaulted and those businesses are highly likely going to default when given a loan. Thus, the target labels for this dataset is "approve loan" (1) or "deny loan" (0).

For numerical and categorical features, the dataset initially tagged a few of the numerical features as categorical due to string characters attached. These were corrected during the data cleaning and transformation process. Numerical features are those with data types Float or Integer, while those with Object are categorical. Those with DateTime are special data types but are generally considered as categorical features. In addition, new columns were added as suggested by the case study supplementary material which were derived from the original columns of the dataset.

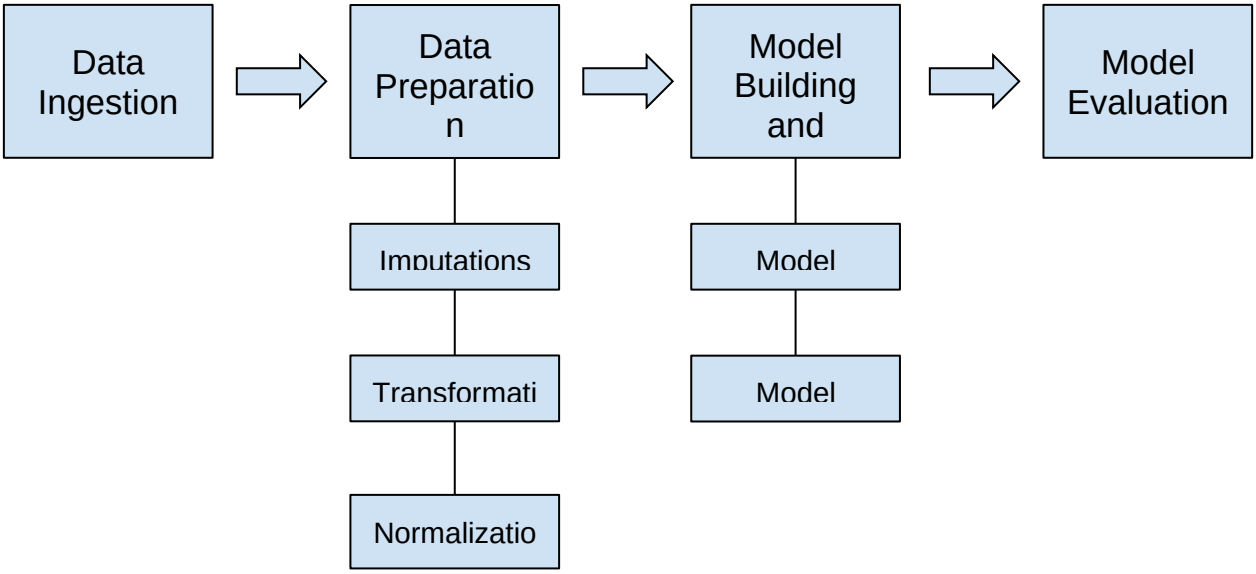
# Methodology

## Machine Learning Pipeline

The machine learning model was developed as simple and straightforward as it can be. First, the dataset was extracted from Kaggle. Next, the dataset was cleaned, transferred and engineered to remove missing values and unwanted variables. Then, the dataset was prepared by determining the target label (y) and predictors (x) and splitting the dataset into train and test. The dataset was further prepared by selecting features and scaling before ingesting into the model train. Then, the final cleaned training dataset was fed to the machine learning algorithms

to be trained. The performance results of each machine learning training model were evaluated, determining the algorithm that produces the best performance metrics.

Figure 1. Machine Learning Pipeline




## Data Processing

The dataset was first examined in terms of number of observations, data types per feature, values found per column, and missing values. After determining which features need to be cleaned, the preparation started with Borrower Name being immediately removed at the beginning since the LoanNR_ChkDgt column was already sufficient to identify the business. Then, all columns with DateTime were converted to Date to prepare the columns for computations later on. Next, those with numerical columns especially those containing amounts that are tagged as Object or categorical were cleaned by removing the strings attached and changing the data type to Float. Next, the City and State have missing values but since Zip column is available, these two columns were imputed using the uszipcode package by searching the City and State by Zip code of the borrower. Those rows that were still with missing values were dropped.

For the Bank State column, the initial assumption was that the bank of the borrower must be residing in the same state. However, almost half of the data did not match, thus both Bank State and Bank Name were dropped as they would not be able to contribute much to the model. For other categorical features with more than two (2) values, those that are still missing after deriving existing conditions were dropped. For columns such as Industry and Industry Default Rates, values were derived from the case study and were mapped to update the NAICS column that was eventually dropped. After computing the Real Estate and Recession columns, the date columns were eventually dropped. Lastly, for the MIS_Status, the PIF and CHGOFF were converted to 1 and 0, respectively. ChgOffPrinGr was also dropped as it created data leakages that resulted in unusual performance of the model upon first initial run.

The dataset was eventually split into train and test using 80% and 20% split. The 80/20 ratio is the common choice in practice. After splitting the dataset, the categorical features for both

train and test were one hot encoded since XGBoost and Naive Bayes algorithms only allow features in numerical form. It is also worth noting that the number of columns of the training data should match with the testing data, thus, this was also checked before performing dropping low variance features and features with high multicollinearity. The low variance features were removed to lessen the noise in the data while the variance inflation factor was used to drop features with high multicollinearity. Then, the final dataset ready for modeling was scaled to normalize the values. A total of 28 features were retained in the final dataset.

## Algorithms

The algorithms used for the project are Extreme Gradient Boosting or XGBoost (decision tree) and Naive Bayes (linear classifier).

### Naive Bayes

How Naive Bayes works for this problem is by analyzing the features that are found in a higher risk loan and a lower risk loan. A simplified example of this is given a higher risk loan had a total amount of X,Y and Z features found in the entire data set of higher risk loans versus a lower risk loan counting the total amount of X Y and Z features found in lower risk loans how likely is a loan to be higher risk or lower risk based on the X Y and Z features found in both categories and if the score was to skew one way or the other (example lower risk) then X , Y and Z would point the loan to be classified as a lower risk loan. The advantages that naive bayes have is, it could scale well and is easy to learn for implementation. The disadvantages of the algorithm is that it could end up processing data that has features that cannot be found in the training dataset and its assumptions may not always hold.

### XGBoost (Decision Tree)

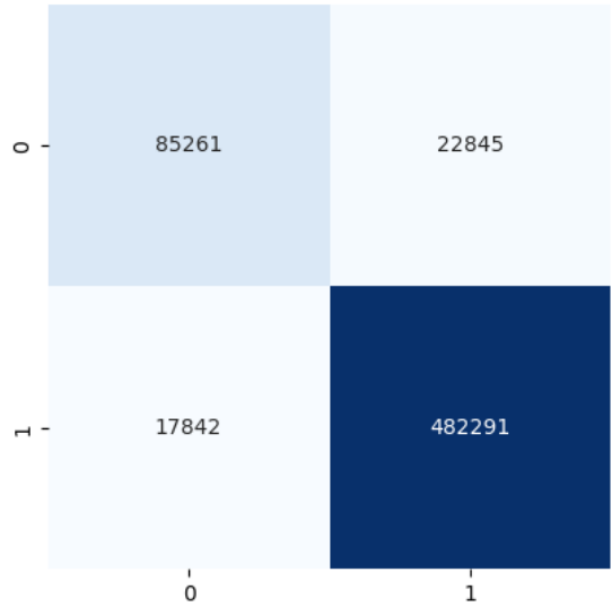
How XGBoost works for this problem is that it uses base learners to classify if a loan is high risk or low risk. By answering yes or no to certain features found in the data set, it could identify if a loan is high risk or low risk. It works well with large datasets since it uses parallel learning, optimizing its speed to run computations and create trees. The advantage is that it is simple to learn where there are a lot of resources online for the library and the algorithm. Also, the hyperparameters are easy to understand and calibrate if unwanted results were produced. The disadvantage is in defining the parameters since the algorithm tends to overfit in the training data which will lead to a performance drop in the model when used in validation sets with stress testing. To address this, it is always better to use few hyperparameters as the more hyperparameters are used, the more the algorithm overfits to produce the developers desired result.

# Results and Analysis

The performance of the algorithms were evaluated based on the following metrics: 1) Accuracy, 2) Precision, 3) Recall, 4) Specificity, and 5) F1 Score. Results for actual versus predicted values were shown using a Confusion Matrix.

For the XGBoost algorithm, the model performed well for both Train and Test even without hyperparameter tuning. The algorithm was already able to accurately predict 93% of the data in the training.

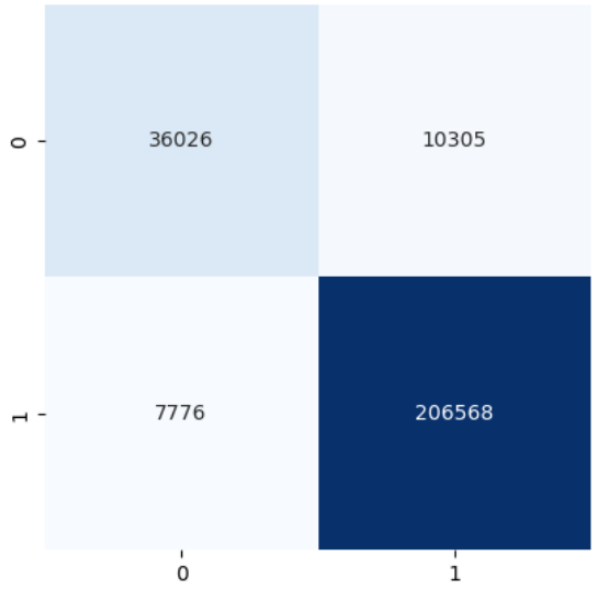
**Figure 1. XGBoost Confusion Matrix (Train)**




```
Accuracy: 0.9331
Precision: 0.9548
Recall: 0.9643
Specificity: 0.7887
F1 Score: 0.9595
True Positive: 482291
False Positive: 22845
False Negative: 17842
True Negative: 85261
```

Also, another worth noting is how the test prediction results were close to the train prediction results which is normally not the case with XGBoost algorithm as its train predictions tend to overfit. Thus, it can be inferred that the data processing techniques implemented and the final features retained were appropriate to use in predicting whether to approve or deny the loan using this algorithm.

**Figure 2. XGBoost Confusion Matrix (Test)**

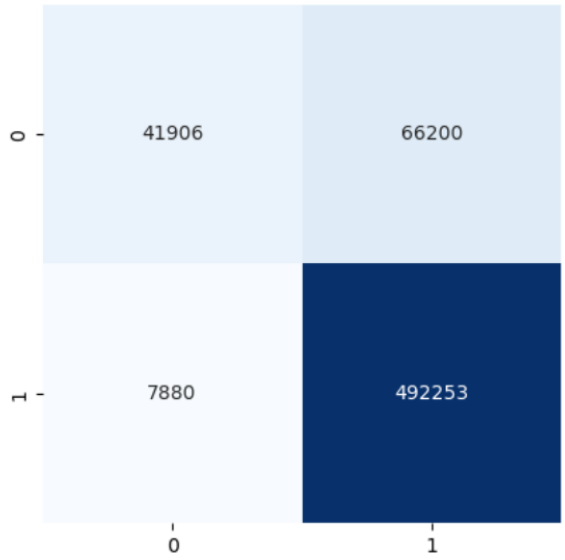



```
Accuracy: 0.9306
Precision: 0.9525
Recall: 0.9637
Specificity: 0.7776
F1 Score: 0.9581
True Positive: 206568
False Positive: 10305
False Negative: 7776
True Negative: 36026
```

Using hyperparameters with the help of RandomizedSearchCV, focusing on improving Recall, the model's Accuracy in the training predictions lowered to 87% but Recall increased to 98%. The number of true positives increased but the false positives also increased since we are focusing on determining who should be granted a loan. It can be seen here that there is a trade off between the metrics when focus is emphasized on one. Thus, there is a need to communicate with the user of the model what is the purpose (e.g. To increase loan approvals? To lessen denied loan grants?) and to what extent can the user handle false positives or false negatives (e.g. Are they willing to have high false positive but low false negative?).

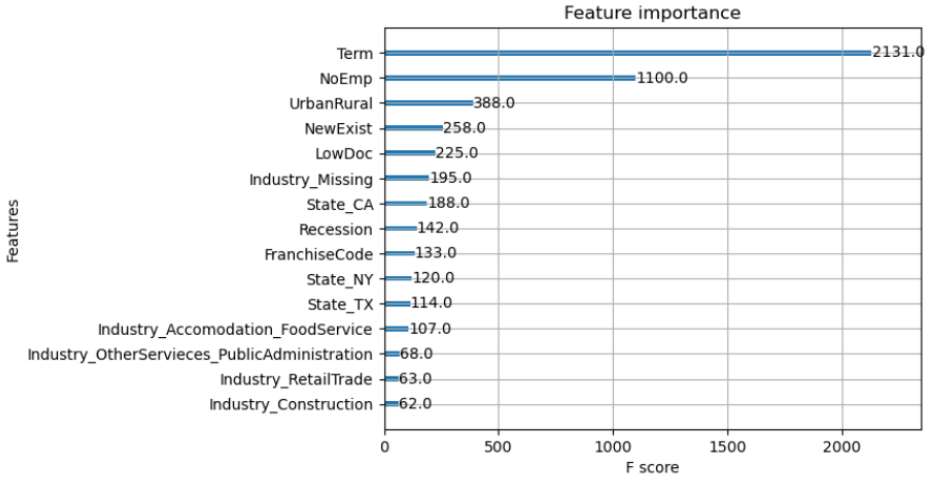
Furthermore, looking at the features, the length of loan term is the most important feature. This may infer that those who have longer loan terms are less likely to default and probably have lower amortization amounts to pay considering that they have longer time to pay off the loan. Interestingly, the number of employees also matters. This may infer that those businesses that probably have scaled enough that they probably have more employees means that the business is earning and sustainable and is able to pay off the loan.

**Figure 3. XGBoost Confusion Matrix with Parameter (Train)**



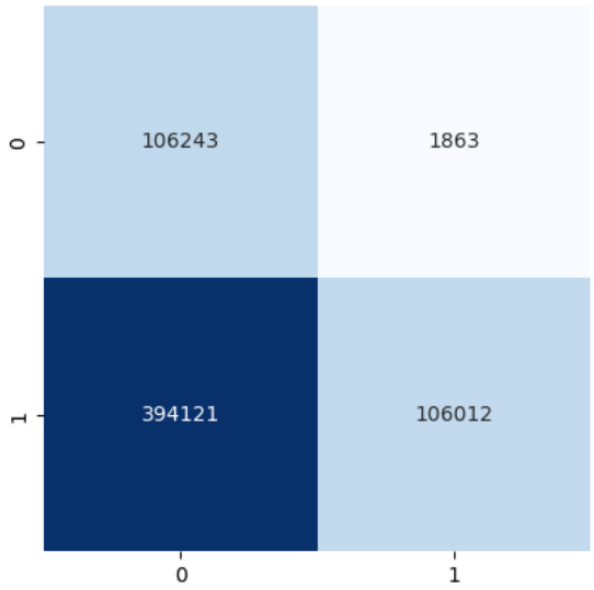

```
Accuracy: 0.8782
Precision: 0.8815
Recall: 0.9842
Specificity: 0.3876
F1 Score: 0.9300
True Positive: 492253
False Positive: 66200
False Negative: 7880
True Negative: 41906
```

**Figure 4. Feature Importance**



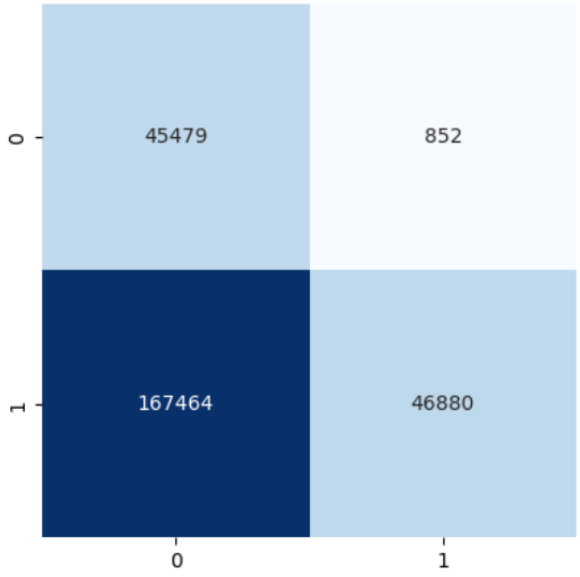

However, for Naive Bayes, the model performed poorly for both training and testing predictions. Despite its simplicity for implementation, the model was not able to perform well on a large, complex dataset. The training model produced a 34% accuracy score and had true positives at only 10% of the total dataset observation. Its recall was only at 21%, rendering the model unable to predict correctly the businesses that are less likely to default most of the time which can be detrimental to a bank since they need customers to avail loans to ensure income from interests coming from these loans.

**Figure 5. Naive Bayes Confusion Matrix (Train)**




Accuracy: 0.3490
Precision: 0.9827
Recall: 0.2120
Specificity: 0.9828
F1 Score: 0.3487
True Positive: 106012
False Positive: 1863
False Negative: 394121
True Negative: 106243

**Figure 6. Naive Bayes**




```
Accuracy: 0.3543
Precision: 0.9822
Recall: 0.2187
Specificity: 0.9816
F1 Score: 0.3578
True Positive: 46880
False Positive: 852
False Negative: 167464
True Negative: 45479
```

Unfortunately, unlike with XGBoost, Naive Bayes cannot use Randomized Grid Search in order to find the best parameter since it barely has any parameters to calibrate. Thus, the model failed to provide a good model to predict which businesses should have their loan applications granted or not.

# Conclusions and Recommendations

Among the algorithms explored, the XGBoost machine learning model performed well in terms of overall performance. Even without hyperparameters, the model exhibited a good performance result. However, the model developers recommend exploring using sampling methods since the target feature is quite imbalanced despite the good performance shown by the current developed model. Also, an additional feature suggested in the case study, SBA Guaranteed Portion which can be derived from SBA Approved Loans/Gross of Approved Loan from the Bank, can be added in future analysis if this will create stronger model performance especially for other algorithms.

# References

*How XGBoost algorithm works*. (n.d.). Retrieved April 21, 2023 from https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-xgboost-works.htm

*What is naïve Bayes*. IBM. (n.d.). Retrieved April 16, 2023, from https://www.ibm.com/topics/naive-bayes#:~:text=The%20Na%C3%AFve%20Bayes%20classifier%20is,a%20given%20class%20or%20category.

T., B. (2023, April 8). *Beginner's Guide to xgboost for classification problems*. Medium. Retrieved April 16, 2023, from https://towardsdatascience.com/beginners-guide-to-xgboost-for-classification-problems-50f75aac5390

# Source Code

https://github.com/jrsmgno/grant-loan-app