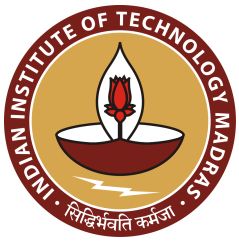
# Credit Default Prediction

April 25, 2023

Project Report submitted for

**Modelling Workshop**




by

**Mohd Umair Ali,**
**Shubham Verma,**
**Shrichand Kushwaha,**
**Nitish Kumar**

**Abstract**

Loan default prediction is a crucial task for financial institutions in managing credit risk and ensuring the stability of their lending portfolio. By using machine learning algorithms, financial institutions can better assess the risk associated with a loan applicant and make informed lending decisions. Furthermore, effective default prediction algorithms can help reduce losses for financial institutions, minimize the impact on their customers, and contribute to a stable financial system overall. In today's complex and rapidly changing financial landscape, accurate default prediction is essential for responsible and sustainable lending practices.

The objective is to compare the accuracy of different algorithms and identify the best model for default prediction. We will use an industrial scale data set to build a machine learning model. The process will have training, validation, and testing datasets. The performance of the model will be evaluated based on accuracy and the count of false negatives. The results of this study will provide insights into the most effective approach for predicting loan default and can be useful for financial institutions in their credit risk management.

# Contents

# 1 Motivation

Loan default prediction is critical for financial institutions offering lending services. When borrowers cannot repay their loans, it can significantly impact the institution's financial stability and reputation. In general, loan default prediction is a challenging problem for several reasons. Firstly many factors can influence a borrower's ability to repay a loan, including their credit score, income, employment history, and more. Secondly, there is often a high degree of variability in the data, as borrowers come from various backgrounds and circumstances. Thirdly, loan defaults are relatively rare, making it difficult to train machine learning (ML) models on them.
Predicting loan defaults can help financial institutions manage their risk more effectively and take proactive steps to prevent defaults. Recently, ML algorithms have emerged as powerful tools for predicting loan defaults. For instance, decision trees, random forests, support vector machines, and neural networks. Each of these algorithms has its strengths and weaknesses, and their performance can vary depending on the dataset's characteristics and the specific problem being addressed. Decision trees are simple and easy to understand but may not be as accurate as other algorithms. Random forests are a more complex version of decision trees that can handle larger datasets and provide more accurate predictions. Support vector machines are good at separating data into different classes but may not be as effective at handling large datasets. Neural networks are highly accurate but can be difficult to understand and require significant computing resources. Selecting the best algorithm for loan default prediction is not a straightforward task. It requires understanding the dataset's characteristics and the problem being addressed. By analyzing large datasets and identifying patterns and trends, these algorithms can accurately predict the likelihood of a borrower defaulting on a loan. The ML algorithms use historical data to train models that help to predict future events. These algorithms can learn from vast amounts of data, making them useful in identifying patterns and trends humans may miss. Nevertheless, with a wide range of algorithms available, it is challenging to determine which is the most effective for this particular task. Different algorithms have different strengths and weaknesses, and their performance can vary depending on the dataset's characteristics and the problem being addressed.

# 2 Objective

We aim to compare the accuracy of different algorithms and identify the best model for default prediction. This work addresses the challenge of finding a suitable ML model by thoroughly evaluating various algorithms for loan default prediction. Specifically, we apply ML algorithms to a single dataset of loan information and then identify the ML algorithm that provides the most accurate predictions of loan defaults. The results of this project will provide financial institutions with a powerful tool for managing their risk and preventing loan defaults.
In short, this study represents an important step toward improving the accuracy and effectiveness of loan default prediction. It has the potential to significantly impact the financial industry as a whole. By testing and comparing different algorithms on a real-world dataset, one can understand the strengths and weaknesses of different approaches. Moreover, the present work also helps to identify areas for future research and development.
We will use an industrial-scale data set from Kaggle to build an ML model. The following steps will be used on the data before applying them to different ML models, namely, random forest and support vector machine: cleaning, training, validation, and testing datasets, including anonymized customer profile information.

# 3 Terminology

- **Confusion Matrix**: A confusion matrix or error matrix is a specific table for summarizing the performance of a classification algorithm.
- Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa.

- A confusion matrix looks as follows:

Table 1: Predicted Class

| actual class | | **Positive** | **Negative** |
|---|---|---|---|
| | **Positive** | TP | FN |
| | **Negative** | FP | TN |

- **TP (True Positive)**: Number of data points correctly predicted to the positive class.
- **FP (False Positive/ Type I Error)**: Number of data points that actually belong to the negative class, but predicted as positive (i.e., falsely predicted as positive).
- **FN (False Negative/ Type II Error)**: Number of data points that actually belong to the positive class, but predicted as negative (i.e., falsely predicted as negative).
- **TN (True negative)**: Number of data points correctly predicted to the negative class.

$$\begin{aligned}
\textbf{Accuracy} &= \frac{\text{number of correctly predicted data points}}{\text{total number of data points}} \\
&= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \\
\textbf{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\
&= \frac{|\{ \text{ relevant data points } \} \cap \{ \text{ selected data points } \}|}{|\{ \text{ retrieved data points } \}|} \\
\textbf{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\
&= \frac{|\{ \text{ relevant data points } \} \cap \{ \text{ selected data points } \}|}{|\{ \text{ relevant data points } \}|}
\end{aligned}$$

# 4 Detailed Literature Survey

The article [1] begin by providing an overview of loan default prediction and its importance in risk management for banks and financial institutions. They note that loan default prediction models have been developed using various statistical and machine learning techniques, including logistic regression, decision trees, neural networks, and support vector machines.

The literature review then examines the existing studies that have compared the performance of different loan default prediction models. The authors note that these studies have found that ensemble methods, such as random forest and gradient boosting, generally outperform single models in terms of predictive accuracy. They suggest that future research should focus on developing more accurate and interpretable models that can be used to inform lending decisions and reduce the risk of loan defaults.

The article [2] compares the performance of decision tree and random forest models for loan default prediction. The study uses a dataset of loan applications and their corresponding outcomes (default or non-default). The authors first pre-process the data by removing missing values, encoding categorical variables, and balancing the classes. They then apply decision tree and random forest algorithms to the dataset and evaluate their performance based on various metrics, such as accuracy, precision, recall, and $F1$-score.

The results of the study in the article [2] shows that the random forest model outperforms the decision tree model in terms of all the evaluation metrics. The authors conclude that the random

forest algorithm is a more effective approach for loan default prediction, and suggest that it can be used by banks and financial institutions to manage their risks more effectively.

The results in [3] talks about the performance Naive Bayes and Decision Tree algorithms and found that Naive Bayes outperformed Decision Tree in terms of accuracy.

## 5 Useful Links

- The data set is called "American Express - Default Prediction" and it is collected from the Kaggle website.
- The source code can be obtained from the google drive link here.

## 6 Flow chart

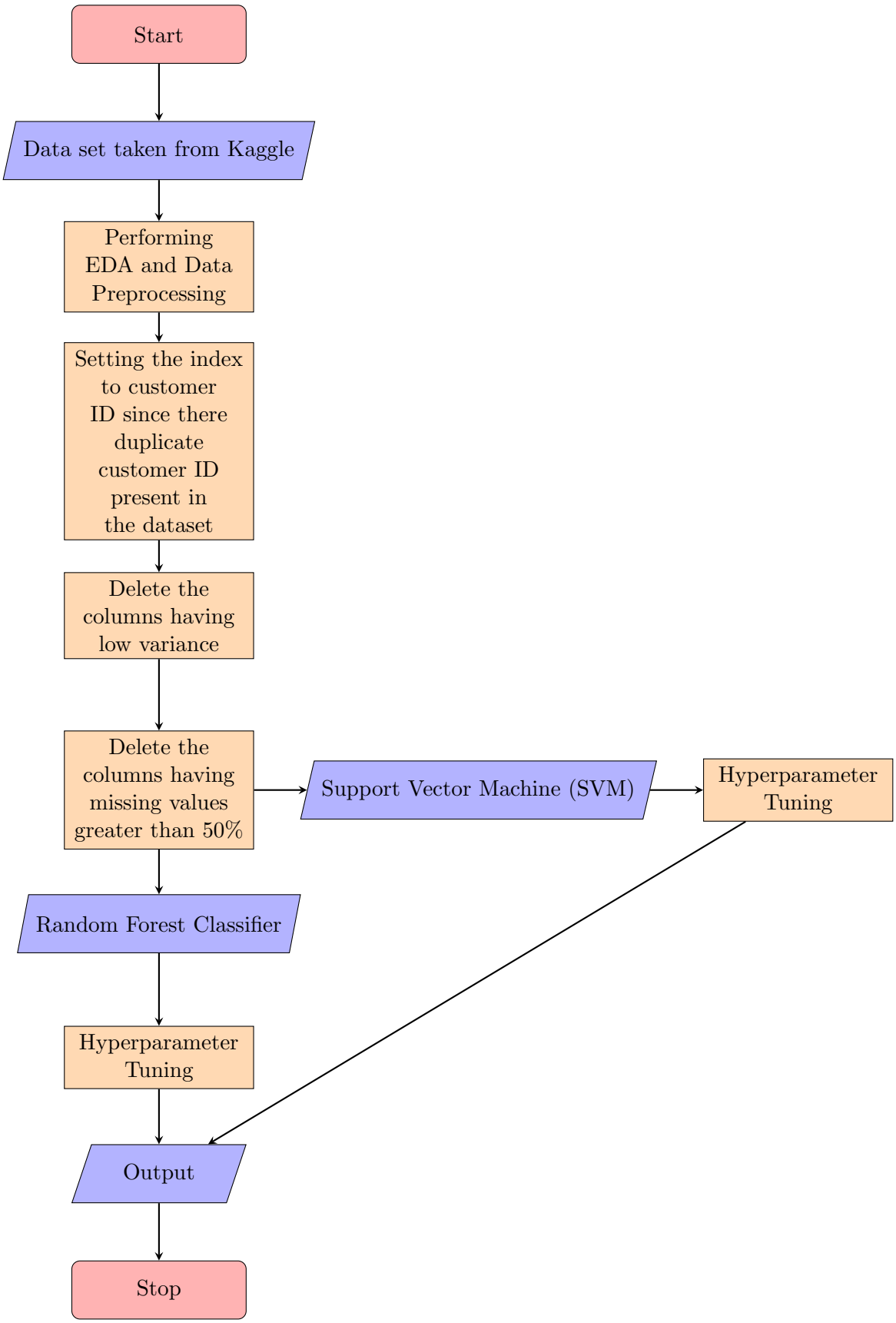
The flow chart outlines the steps taken in this project, beginning with obtaining the data set from Kaggle and performing exploratory data analysis (EDA) and data preprocessing.

In order to prepare the data for modeling, the index is set to customer ID to address duplicate entries, and columns with low variance and missing values greater than 50% are deleted.

The project uses two different algorithms for classification - a random forest classifier and a support vector machine (SVM). Each algorithm undergoes hyperparameter tuning to optimize its performance.

The final output of the project is presented in the output box, completing the project workflow and achieving the project goals.

A more detailed explanation can be found in the methodology section, please see 7.


Start
Data set taken from Kaggle
Performing EDA and Data Preprocessing
Setting the index to customer ID since there duplicate customer ID present in the dataset
Delete the columns having low variance
Delete the columns having missing values greater than 50%
Support Vector Machine (SVM)
Hyperparameter Tuning
Random Forest Classifier
Hyperparameter Tuning
Output
Stop

# 7 Methodology

1. We begin by performing Exploratory Data Analysis (EDA) on the given data. After performing EDA we can make the following observations:

   **Information about Dataset**

   (a) We have six types of features in our data
   **customer ID**:-Unique Customer ID.
   **D_ * Delinquency Variables** :- to measure the past or present payment behaviour of a borrower.
   **S_ * Spend Variables**:- To measure consumer spending behaviour.
   **P_ * Payment variables**:- To measure or predict payment behavior. Example: Payment history, Payment method, Payment terms.
   **B_ * Balance variables**:- To measure or predict the balance of an account or financial transaction. Example: Account balance, Credit limit, Payment due date.
   **R_ * Risk variables**:-To measure or predict the level of risk associated with a particular activity or investment. Example: Credit rating, Historical performance, Market conditions.

   (b) Initially we have **5531451** rows and **191** columns in our Dataset.

   (c) We have to take care of Customer ID's feature since we have duplicated customer ID's present. We found that Customer ID's are duplicated between 1-13 in number and data is present from 2017/03/01 to 2018/03/31.
   In order to avoid duplicates we set our index to customer ID's.

   (d) We checked for the different types statistical data present in the data set, we found that our data comprises of numerical features and categorical features. We'll take care of categorical features later by performing One Hot Encoding (OHE).

   (e) **Distribution of various features**:-
   Total number of delinquency variables: **96**
   Total number of spend variables: **22**
   Total number of payment variables: **3**
   Total number of balance variables: **40**
   Total number of risk variables: **28**

   (f) **Distribution of numerical and categorical features**:-
   Total number of features: **190**
   Number of categorical features: **11**
   Number of continuous features: **177**

   (g) Our target comprises of 0 and 1, where 0 indicates **Payment non-Default** and 1 indicates **Payment Default**. The distribution of target variable is :
   **Target Distribution**:-

| | **Count** | **Percentage** |
|---|---|---|
| **0** | 4153582 | 75.090279 |
| **1** | 1377869 | 24.909721 |

   **Removing non-relevant features**

   (h) We find the number of missing values in each feature in data set and then we remove the columns which have more than 50% missing values.

   **Treatment of Categorical Variables**

   (i) We perform One Hot Encoding technique to represent categorical variables as numerical data in data frame. In this technique, each category in the categorical variable is represented as a binary vector, where the length of the vector is equal to the number of categories in the variable.
   Initially we had 159 features and after doing OHE the features are increased to 167.

```
1
2
3  # since we require numeric values.
4  # we will perform one-hot encoding for D_63 and D_64
5  # and drop column D_63 and D_64 subsequently
6
7  train_D63 = pd.get_dummies(train[['D_63']])
8  train = pd.concat([train,train_D63],axis = 1)
9  train = train.drop(['D_63'],axis = 1)
10
11 train_D64 = pd.get_dummies(train[['D_64']])
12 train = pd.concat([train,train_D64],axis = 1)
13 train = train.drop(['D_64'],axis = 1)
```

Listing 1: Performing One Hot Encoding

(j) We segregated all the columns into 5 categories: Delinquency, Spend, Payment, Balance and Risk. Then we find the correlation among each categories using correlation matrix and we drop the features which are highly correlated i.e. with absolute correlation of more than 90%.
Now after dropping highly correlated features our features drop from 167 to 153 in count.

**Treatment of Columns having Low variance**

(k) We also remove columns with low variance since they do not provide much information or variability in the data, and thus, they are less useful for predictive modeling.

We use *VarianceThreshold* from scikit-learn library to perform this operation.

After removing the columns which have variance less than 0.1, we are left with 58 features only.

```

# remove columns with low variance=0.1. Keep only columns with high variance
from sklearn.feature_selection import VarianceThreshold
from itertools import compress

def fs_variance(df, threshold:float=0.1):
    """
    Return a list of selected variables based on the threshold.
    """
    # The list of columns in the data frame
    features = list(df.columns)

    # Initialize and fit the method
    vt = VarianceThreshold(threshold = threshold)
    _ = vt.fit(df)

    # Get which column names which pass the threshold
    feat_select = list(compress(features, vt.get_support()))

    return feat_select
columns_to_keep=fs_variance(train_drop_highcorr)
# We are left with 58 columns (excluding target), which passed the threshold
    .
train_final=train[columns_to_keep]
len(columns_to_keep)
```

Listing 2: Dropping the features having Low Variance

**In the end, we have** 458913 **rows and** 58 **features or columns.**

(l) Now we proceed to apply the machine learning algorithms to the dataset obtained after doing data pre-processing.

2. We then make use of the following machine learning algorithms :

- We drop the target feature and Customer_ID from our data set and we store the target in a separate object because it is the variable that we want to predict or model in a machine learning problem.

```

    y_train=train_final['target']
    x_train=train_final.drop(['target','customer_ID'],axis=1)

```

- Now, we split the data into training and test using *train_test_split* from scikit-learn module. We do a 75-25 split which means that 75% data is used for training purpose and 25% is used as test data set.

```

    from sklearn.model_selection import train_test_split

    x_train_split, x_test_split, y_train_split, y_test_split =
    train_test_split(x_train, y_train, test_size=0.25, random_state=26)


```

Listing 3: Splitting the data

- **Support Vector Machine (SVM)** is a popular supervised machine learning algorithm used for classification and regression tasks.
  In classification tasks, SVM works by finding the best hyperplane that separates different classes in the feature space. The hyperplane is chosen in a way that maximizes the margin between the closest points from each class, which are called support vectors.

When making predictions, SVM computes the distance between a new data point and the decision boundary. The data point is then classified based on which side of the boundary it falls on.

SVM works well with high-dimensional data and can handle both linear and non-linear classification tasks using techniques such as kernel methods. However, SVM can be sensitive to the choice of hyperparameters and can be computationally expensive to train on large datasets.

- **Random Forest** is a popular ensemble learning algorithm used for classification and regression tasks in machine learning.

  In classification tasks, Random Forest works by building a multitude of decision trees and aggregating their predictions to make a final prediction. Each decision tree is constructed by selecting a random subset of features and a random subset of training data, which helps to reduce overfitting.

  When making predictions, Random Forest aggregates the predictions of each decision tree in the forest and outputs the class that receives the most votes.

  Random Forest can handle high-dimensional data and can capture non-linear relationships between features. It is also less sensitive to overfitting compared to single decision trees. However, Random Forest can be computationally expensive and may not perform well with noisy or irrelevant features.

3. Parameters which define the model architecture are referred to as hyperparameters and thus this process of searching for the ideal model architecture is referred to as *hyperparameter tuning*. In order to make our model learn well from the data we will do "Hyperparameter Tuning".

   We will use the RandomSearchCV in order to select best parameters for our models.

   The following are methods to tune our model in order to select best hyperparameters:

   (a) Grid Search

   (b) Random Search

   We use the grid search CV technique for getting the optimal hyperparameters for random forest classifier and SVM.

```
# Use Random Forest
from sklearn.ensemble import RandomForestClassifier
```

Listing 4: Importing Random Forest Classifier from sklearn library

```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200,
stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
```

```
                'bootstrap': bootstrap}
```

Listing 5: Setting the hyperparameters to consider for hyperparameter tuning in Random Forest Classifier

```
model = RandomForestClassifier(n_estimators=400, max_features='sqrt', bootstrap=
    True, max_depth=30, min_samples_leaf=1, min_samples_split=5, n_jobs=-1)
#rf_random = GridSearchCV(estimator = rf, param_grid = random_grid, cv = 3,
    verbose=1, n_jobs = -1)
# Fit the random search model
model.fit(x_train_split,y_train_split)
```

Listing 6: Fitting the model with the best hyperparameters obtained for Random Forest Classifier

4. After obtaining the optimal parameters for our different models, we go for model fitting and find the time taken for fitting the model.

5. After model fitting we obtain the accuracy, the classification report, the confusion matrix and we do the prediction on our test data set. Please see the results section for the same.

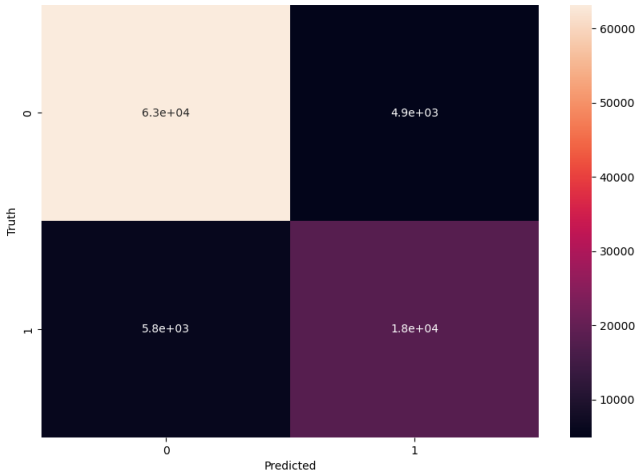
# 8 Results

This section presents the outcomes of the project using two different algorithms - the Random Forest Classifier (RFC) and Support Vector Machine (SVM).

## 8.1 Random Forest Classifier (RFC)

- We achieved an accuracy of 88.39% in 852.85 seconds.
- The time taken for prediction on test data is 4.34 seconds.

Figure 1: Confusion Matrix for Random Forest Classifier.Please refer to 3 for more details about confusion matrix

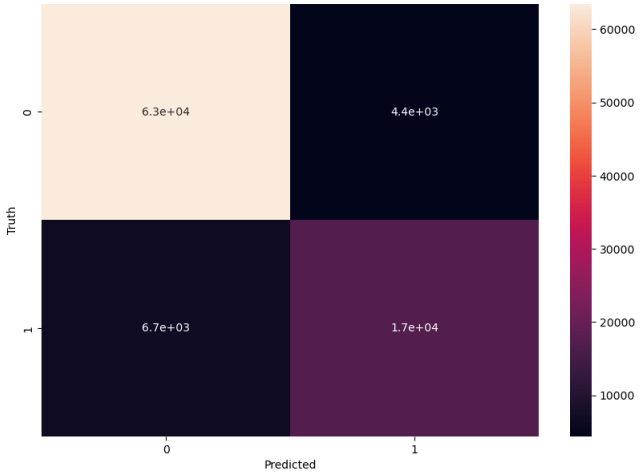



- The false negative count is 5833 out of 91783 values.

## 8.2 Support vector Machine (SVM)

- We achieved an accuracy of 87.94% in 6708.29 seconds.
- The time taken for prediction on test data is 1075.62 seconds.

Figure 2: Confusion Matrix for Support Vector Machine . Please refer to 3 for more details about confusion matrix.




- The false negative count is 6667 out of 91783 values.

## 9 Summary and Conclusion

Loan default prediction is an important task in risk management for banks and financial institutions. Various statistical and machine learning techniques have been used to develop loan default prediction models, including logistic regression, decision trees, neural networks, support vector machines, and ensemble methods such as random forest and gradient boosting.

A literature review of existing studies has found that ensemble methods generally outperform single models in terms of predictive accuracy. However, there is a need to develop more accurate and interpretable models that can be used to inform lending decisions and reduce the risk of loan defaults.

The authors compared the performance of decision tree and random forest models for loan default prediction. They found that the random forest model outperformed the decision tree model in terms of all evaluation metrics. The authors concluded that the random forest algorithm is a more effective approach for loan default prediction and can be used by banks and financial institutions to manage their risks more effectively.

Our own project used two different algorithms, the Random Forest Classifier (RFC) and Support Vector Machine (SVM), to predict loan defaults. We achieved an accuracy of 88.39% with RFC and 87.94% with SVM. The results indicate that RFC is a better approach for loan default prediction, with a lower false negative count.

A false negative in loan default prediction can have serious consequences for lenders as it may lead to lending money to borrowers who are more likely to default. This can result in financial losses for the lenders, and ultimately, harm their business.

To address false negatives in loan default prediction, lenders may adopt a more conservative lending strategy, such as requiring higher collateral or credit score thresholds, or using a more sophisticated model that balances false positives and false negatives based on their business goals and risk tolerance. It's also essential to regularly monitor and evaluate the performance of the model, and adjust it accordingly to reduce false negatives while maintaining an acceptable level of false positives.

Future research should focus on developing more accurate and interpretable models that can aid in lending decisions and reduce the risk of loan defaults.

# References


[1] Aslam, Uzair, Aziz, Hafiz Ilyas Tariq, Sohail, Asim *An Empirical Study on Loan Default Prediction Models..* Journal of Computational and Theoretical Nanoscience, 2019.

[2] Madaan, M et al. *Loan default prediction using decision trees and random forest: A comparative study.* IOP Conference Series: Materials Science and Engineering, 2021.

[3] Vimala S., Sharmili K.C. *Prediction of Loan Risk using Naive Bayes and Support Vector Machine.* International Conference on Advancements in Computing Technologies - ICACT, 2018.


| S.No | YOP | Citation | Remark |
|---|---|---|---|
| 1 | 2014 | Business Analytics using Random Forest Trees for Credit Risk Prediction: A Comparison Study.**Nazeeh Ghatasheh**,International Journal of Advanced Science and Technology Vol.72 (2014), pp.19-30 | The article talks about ML approaches based on SVM and Neural Networks and Logistic Regression.A model based on Random Survival Forests was compared to Logit model for predicting credit risk for a number of German Small and Medium Enterprises. |
| 2 | 2018 | Prediction of Loan Risk using Naive Bayes and Support Vector Machine **Vimala S., Sharmili K.C.**,International Conference on Advancements in Computing Technologies - ICACT 2018 | Talks about the performance Naive Bayes and Decision Tree algorithms and found that Naive Bayes outperformed Decision Tree in terms of accura |
| 3 | 2019 | **Aslam, Uzair, Aziz, Hafiz Ilyas Tariq, Sohail, Asim** An Empirical Study on Loan Default Prediction Models. Journal of Computational and Theoretical Nanoscience,2019 | Ensemble methods, such as random forest and gradient boosting, generally outperform single models in terms of predictive accuracy |
| 4 | 2021 | **Madaan, M et al.** Loan default prediction using decision trees and random forest: A comparative study. IOP Conference Series: Materials Science and Engineering ,2019 | Concluded that the random forest algorithm is a more effective approach for loan default prediction, and suggest that it can be used by banks and financial institutions to manage their risks more effectively |