



Topic: CA Two

- Prediction of Loan Approval using Data Analytics and Machine Learning Models

Module: B9FT114 Data Analytics and Machine Learning

Faculty: Ciara Feely

Participant:

- Siddhi Kelshikar - 10627249

## INTRODUCTION

Being student of Financial Technology, I wanted to incorporate the following into financial data:

Data visualization, Data analytics, Machine learning

This led us to select the data which answers questions related to the prediction of getting a loan approval.

On the basis of patterns found in the data, machine learning can be used to create prediction models that can make predictions on new data. To create prediction models, a variety of machine learning methods can be applied, including Regression, Classification, Clustering, Time series analysis.

Using either supervised learning, unsupervised learning, or a combination of the two, machine learning models can be trained. In supervised learning, the model learns to make predictions based on the input features by being trained on labelled data with a known outcome variable. Unsupervised learning aims to find patterns and organisation in the data by training the model on unlabelled data.

A few essential steps are often taken while developing a prediction model using machine learning, including:

1. Data collection and pre-processing: Gathering pertinent data and preparing it in a way that the machine learning algorithms can use. This could entail formatting the data correctly, managing missing numbers, and cleaning the data.
2. Feature selection and engineering: choosing the characteristics that are most pertinent to the outcome variable and, if necessary, engineering new features.
3. Model selection and training: selecting the best machine learning algorithm for the task at hand and putting the model to use by training it on the data.

4. Model evaluation and tweaking: Optimising the performance of the model by tuning its hyperparameters and evaluating the model's output using the relevant metrics.

5. Deployment and monitoring: Ensuring that the model continues to produce correct predictions by deploying it in a production environment and tracking its performance over time.

Overall, machine learning can be a potent tool for creating prediction models that can correctly forecast new data, and it is becoming more and more significant in a variety of fields, including marketing, finance, and healthcare.

To interpret the Loan Approval Prediction data, we have used various Data Analytics and ML models. However, to analyse the Loan Approval Prediction data, we used different techniques such as data visualization, data analytics, and machine learning. These methods helped us to understand the data better and find the most relevant information for our analysis. More about it is covered in the report below.

### **Research Questions:**

A wide range of questions may be addressed with data analytics and visualisation. However, in order to keep things brief, this study uses the loan approval dataset to provide responses to the following questions:

1. Is any specific gender which is preferred when applying for a loan?
2. Which loan applicants have a high credit score, self-employed or non-self-employed?
3. Is an area preferred when sanctioning of loan?

The main question to be asked with this dataset is whether any machine learning model can help us predict if a loan applicant will get a loan approval based on the parameters provided in the dataset.

## DATA OVERVIEW AND METHODOLOGY

The dataset is taken from kaggle.com. The data is already split into training and testing and has 12 different columns, wherein 9 columns are categorical and 3 are numerical.

The columns are further classified into Gender, Marital Status, Number of Dependents, Education, Self-Employed, Applicant Income, Co-Applicant Income, Loan Amount, Term, Credit History, Area and Status. A few of the line items are missing from some of the columns.

Firstly, we import all the required libraries in Python notebook and import the dataset for further analyses. In data visualization we look at different factors such as

1. Categorical variables
2. Numerical variables
3. Categorical v/s Categorical variables and
4. Numerical v/s Numerical variables which gives us in depth view of each numerical and categorical feature.

Secondly, we look at data pre-processing where we:

1. Deal with null values, however we don't have null values in our dataset
2. Data imputation for categorical and numerical variables
3. One-hot encoding
4. Removing outliers
5. Separating target variables from the train dataset
6. Balancing the dataset and
7. Data Normalization

Additionally, we used different machine learning models to analyse our dataset, including:

1. logistic regression,
2. K-Nearest Neighbours (KNN),
3. Gaussian Naïve Bayes, and
4. decision tree.

By using these models, we can evaluate and analyse the performance of our data. Now let us discuss the findings in detail:

## ANALYSIS AND DISCUSSION

### Logistic Regression Model

Simple binary classification issues are well-suited for logistic regression, which is straightforward to use compared to other classification algorithms. Logistic regression models can benefit from regularisation to avoid overfitting and boost generalisation performance. Both L1 and L2 regularisation are supported by the scikit-learn class called LogisticRegression.

Logistic regression models can shed light on the correlation between the input features and the target variable and are simple to understand. The factors that are most crucial for forecasting the target variable can be identified using the logistic regression model's coefficients.

Logistic regression models can also calculate the likelihood that a given instance belongs to a specific class. When the prediction confidence is crucial, this can be helpful. It can still function well even when the input features are not linearly separable. It is robust to noisy data and outliers.

Even on big datasets with many features, logistic regression models may be trained fairly quickly.

Overall, logistic regression is a flexible and efficient approach for issues involving binary and multiclass classification, and it is a solid option for many real-world applications.

```
✓ [216] print('Logistic Regression Classification Report:\n', classification_report(Y_test, Y_pred))
```

```
0s Logistic Regression Classification Report:
      precision    recall  f1-score   support

     0       0.22      0.33      0.27         6
     1       0.79      0.68      0.73        22

 accuracy          0.61         28
 macro avg          0.51         28
 weighted avg          0.67         28
```

The F1-score in logistic regression is a performance metric that accounts for both precision and recall. The model predicts the target variable with a reasonable degree of accuracy, according to an F1-score of 0.61.

The model's ability in predicting the minority class is relatively subpar, as evidenced by the macro average F1-score of 0.50. Without taking into account the imbalance in the number of samples in each class, the macro average F1-score is produced by averaging the F1-scores for each class. It implies that the model may be under-predicting the minority class and over-predicting the majority class in this instance.

The imbalance in the number of samples in each class is taken into account by the weighted average F1-score of 0.63. This score is determined by dividing the average of the F1-scores for each class by the quantity of samples in that class. An improved overall performance of the model and the ability to balance the accuracy of the majority and minority classes are both indicated by a higher weighted average F1-score.

It is also helpful to have a look at other metrics like precision, recall, and the confusion matrix in order to better comprehend how well the logistic regression model performs. These metrics can show how the model is doing on each class and point out areas that need work.

It is important to note that a logistic regression model's performance can be enhanced using a variety of methods, including feature engineering, regularisation, modifying the classification threshold, and applying more complex algorithms like support vector machines or gradient boosting.



## K-Nearest Neighbours (KNN) Model

KNN is a classification and regression machine learning algorithm.

KNN requires feature normalisation since the distance metric is scale-sensitive. KNN can additionally weight neighbours by distance to prioritise closer neighbours in classification or prediction.

It is a fast and straightforward method that is simple and easy to grasp. It is a non-parametric method, which means it does not rely on any presumptions regarding the distribution of the data at its core.

KNN does not require a training phase, making it possible to train the model rapidly and simply. It is capable of handling complex decision boundaries and can be utilised successfully for nonlinear classification issues.

For large datasets, KNN can be computationally expensive. However, there are effective methods and techniques (such k-d trees) that can be utilised to speed up the computations. It may be easily expanded to solve multi-class classification problems and is applicable to both classification and regression tasks.

Because KNN is a transparent algorithm, it can shed light on the model's prediction process. It is an algorithm that is capable of handling noisy and incomplete data.

KNN is a suitable option for many real-world applications since it is a flexible and efficient method that can be utilised for a variety of classification and regression tasks. For large datasets with numerous characteristics, its computational complexity can be a drawback, hence it's critical to carefully choose the distance metric and value of K to get the best results.

```
✓ [193] print('KNN Classification Report:\n', classification_report(Y_test, Y_pred))
```

```
0s KNN Classification Report:
      precision    recall  f1-score   support

     0       0.33      0.43      0.38         7
     1       0.79      0.71      0.75        21

 accuracy                   0.64         28
 macro avg       0.56      0.57      0.56         28
 weighted avg    0.68      0.64      0.66         28
```



The F1 score, a widely used assessment metric for KNN models, combines precision and recall into a single number that represents the model's performance.

An F1 score of 0.64 indicates that, despite having space for improvement, the KNN model is producing decent predictions. It is crucial to remember that other measures must also be taken into account because the F1 score alone does not give an accurate picture of the model's performance.

The macro average F1 score of 0.56 suggests that for some of the classes, the performance of the KNN model is slightly subpar. By averaging the F1 scores for each class, the macro average score is determined. A low macro average F1 score therefore indicates that the model is having difficulty correctly classifying some of the classes.

The average F1 score for each class is weighted, with a weight equal to the number of samples in each class, yielding a weighted average F1 score of 0.66. The model appears to be operating more effectively for the larger classes when the weighted average F1 score is higher.

It could be required to change hyperparameters like the number of nearest neighbours or the distance metric employed in order to enhance the performance of the KNN model. Investigating various engineering or feature selection methods might also be beneficial. In order to enhance performance for those particular classes, it may also be crucial to look into the classes for which the KNN model struggles.

## Gaussian Naïve Bayes Model

A probabilistic approach that can be applied to classification issues is the Gaussian Naive Bayes model. It is based on the Bayes theorem and makes the assumption that the features are Gaussian distributed and independent.


A classification algorithm is called Gaussian naive Bayes. It is quick and dependable, and little training is needed.

The GNB model implies that the characteristics are continuous and have a Gaussian distribution and is used for classification tasks. The Gaussian Naive Bayes algorithm is implemented by the Scikit-Learn class known as GaussianNB.

The GNB model is trained on a given dataset using the `fit()` technique. It requires the feature matrix `X` and the target vector `Y` as parameters.

The `predict()` method is used to create predictions based on fresh data. It only needs the feature matrix for the new data, `X_new`, as an argument, and outputs a vector of anticipated class labels.

These are some of the salient characteristics of the Python Gaussian NB model. We may train, test, and evaluate the GNB model on our data using these attributes in order to make predictions about brand-new data points.

```
0s  print('GaussianNB Classification Report:\n', classification_report(Y_test, Y_pred))
```

```
↳ GaussianNB Classification Report:
```

	precision	recall	f1-score	support
0	0.33	0.14	0.20	7
1	0.76	0.90	0.83	21
accuracy			0.71	28
macro avg	0.55	0.52	0.51	28
weighted avg	0.65	0.71	0.67	28

The F1 score, a widely used assessment metric for Gaussian Naive Bayes (GNB) models, combines precision and recall into a single number that represents the model's performance.

The GNB model is making predictions reasonably well, according to an F1 score of 0.71, but there may be space for improvement. It is crucial to take into account

other indicators in addition to the F1 score because it does not give a complete picture of the model's performance.

The macro average F1 score of 0.51 suggests that some of the classes have poor performance from the GNB model. By averaging the F1 scores for each class, the macro average score is determined. A low macro average F1 score therefore indicates that the model is having difficulty correctly classifying some of the classes.

The weighted average F1 score for each class, which is 0.67, is calculated using the number of samples in each class as the weight. The model appears to be operating more effectively for the larger classes when the weighted average F1 score is higher.

It may be essential to investigate various feature selection or engineering techniques, modify hyperparameters, or apply a different classification algorithm altogether in order to enhance the performance of the GNB model. Further research into the classes for which the model is having trouble could be helpful in order to enhance performance for those classes.

## Decision Tree Model

Decision Tree is a type of machine learning algorithm used for both classification and regression tasks. We have used the Decision Tree Regressor model to analyse the data as it is easy to interpret and can be visualized as a tree, making it a useful tool for explaining the decision-making process. It can also handle any missing data. With a max depth equals to 10 we have split the data into subsets based on the values of the input features. The splitting criterion can be based on different metrics such as information gain, or entropy.

The decision tree model has been evaluated using several metrics, which provide information about how well the model is performing.

**Mean absolute error (MAE):** This measures the average difference between the actual and predicted values. A lower value of MAE indicates better accuracy of the model.

**Mean absolute percentage error (MAPE):** This measures the percentage difference between the actual and predicted values. A higher value of MAPE indicates higher error in the predictions.

**Mean squared error (MSE):** This measures the average of the squared differences between the actual and predicted values. A lower value of MSE indicates better accuracy of the model.

**R2 score:** This measures the goodness of fit of the model. It is a value between 0 and 1, where higher values indicate better fit. A negative value of R2 score suggests that the model is performing worse than a simple average model.

**Accuracy score:** This measures the proportion of correct predictions made by the model. A higher value of accuracy score indicates better performance.

**Balanced accuracy score:** This measures the average of the recall obtained on each class. It is particularly useful when the classes are imbalanced. A higher value of balanced accuracy score indicates better performance.

Further, we have plot the decision Decision tree model which gives a clear viewpoint of different features in our dataset and finally we have analyse the model fit summary.

```
model.fit(X_train,Y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=10)
```

```
[ ] Y_pred = model.predict(X_test)
```

```
[ ] mean_absolute_error(Y_test,Y_pred)
```

```
0.35714285714285715
```

```
[ ] mean_absolute_percentage_error(Y_test,Y_pred)
```

```
643371375338642.4
```

```
[ ] mean_squared_error(Y_test,Y_pred)**0.5
```

```
0.5976143046671968
```

```
[ ] r2_score(Y_test,Y_pred)
```

```
-1.121212121212121
```

```
[ ] accuracy_score(Y_test,Y_pred)
```

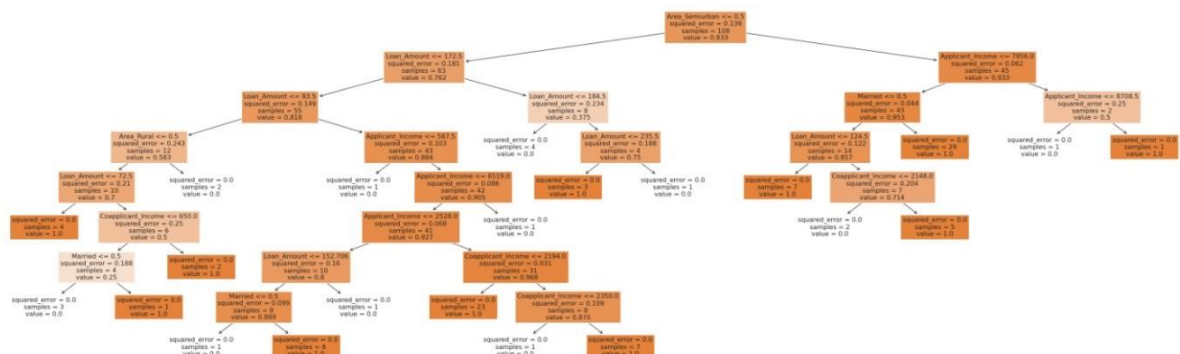
```
0.6428571428571429
```

```
[ ] confusion_matrix(Y_test,Y_pred)
```

```
array([[ 2,  4],
       [ 6, 16]])
```

```
[ ] balanced_accuracy_score(Y_test,Y_pred)
```

```
0.5303030303030303
```



The Decision Tree Classification Report provides information on the performance of a decision tree model on a binary classification problem. It evaluates the model's precision, recall, and f1-score for each class and for the overall model performance.

```
print('Decision Tree Classification Report:\n', classification_report(Y_test, Y_pred))
```

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

     0         0.25        0.33        0.29         6
     1         0.80        0.73        0.76        22

 accuracy          0.64         28
  macro avg         0.53         28
 weighted avg         0.68         28
```

Precision refers to the proportion of true positives (correctly classified instances of a given class) among all instances that the model predicted as positive. Recall, on the other hand, refers to the proportion of true positives that the model correctly identified among all true positives in the data. F1-score is the harmonic mean of precision and recall and provides a single metric that balances both measures.

The report provides the precision, recall, and f1-score for each class (in this case, 0 and 1), as well as the support, which refers to the number of instances of each class in the data.

In this example, the model correctly identified 6 out of 6 instances of class 0 (loan not approved) with a precision of 0.25 and a recall of 0.33. For class 1 (loan approved), the model correctly identified 16 out of 22 instances with a precision of 0.80 and a recall of 0.73.

The overall accuracy of the model was 0.64, meaning that 64% of instances were correctly classified by the model. The macro average of precision, recall, and f1-score was 0.53, indicating that the model performed equally on both classes. The weighted average of precision, recall, and f1-score was 0.68, which takes into account the support for each class and provides an average weighted by the number of instances of each class.

## CONCLUSION AND RECOMMENDATIONS

These are the Answers to our Research Questions:

1. The number of male applicants is higher compared to female applicants.
2. Most non-self-employed applicants have a good credit score compared to self-employed applicants.
3. Most of the loans that got accepted have properties in a semi-urban area compared to urban, or rural.

In conclusion, our analysis demonstrated that the models utilized were moderately accurate in predicting loan eligibility. However, there is still scope for improving the accuracy of the predictions.

One way to enhance the accuracy of the model is by collecting additional relevant data that can improve the features used in the model.

Moreover, handling missing data and outliers appropriately can also play a significant role in improving the model's accuracy. Therefore, further analysis and improvements are necessary to enhance the accuracy of the loan eligibility prediction.

The dataset may be inferred by feature engineering, which is converting the raw data into a format that the machine learning algorithm can better understand. This can be done using methods including scaling, normalisation, and categorical variable encoding.

By continuously refining the models and incorporating additional data sources, we can develop more robust models that can provide more accurate predictions and help to make better decision.

To increase performance, ensemble methods integrate the results of various models. This can involve strategies like stacking, bagging, and boosting.



