Predicting Possible Loan Default Using ML

Name: SHARON SAM S

Roll Number: 215229137

Background:

Loans are the core business of banks. The main profit comes directly from the loan's interest. The loan companies grant a loan after an intensive process of verification and validation. However, they still don't have assurance if the applicant is able to repay the loan with no difficulties.

A loan default occurs when a borrower takes money from a bank and does not repay the loan. People often default on loans due to various reasons. Borrowers who default on loans not only damage their credit but also risk being sued and having their wages garnished.

So there is a need for a predicting loan default. Solution is to build a classification model (Supervised Machine Learning) in which the target variable must need to be classified whether the loan will be approved or not (Yes | No).

Motivation:

In this micro project, I am going to solve by Predicting the possible Loan default by using Machine Learning Algorithms, inspired from a blog acknowledged to Analytics Vidhya which is referenced below.

Problem statement:

Consider a business use case of a Bank or a Finance company which deals in all kinds of loans where the customer who borrows a loan will default or not. The company validates the customer eligibility for the loan default.

The company seeks to automate the possibility of a loan will default or not based on customer detail provided while filling out online application forms.

To automate this process, they have provided a dataset to identify the customer's possibility to default a loan or not so that they can specifically target these customers by the usage of a product (flask web app) for best user interface experiences.

Data set description:

The dataset is real-time based on customer detail provided while filling out online application forms. These contain demographic features of each customer and a target variable showing whether they will default on the loan or not.

The dataset consists of train and test data. The train dataset has 252000 rows with totally 13 features and test dataset has 28000 rows with 12 features in test dataset. Out of 13 features, 12 are input features and 1 is output feature. So train and test dataset would have the same columns except for the target column that is "Risk Flag". Here, the half the features are numeric and half are string data.

The data description is shown below:

- ID: Id of the user(All Unique)
- Income: Salary of the person
- Age: Age of the person
- Experience: Professional experience of the person in years
- Profession: Profession of the person
- Married/Single: Whether married or not
- House_Ownership: Owned or rented or neither
- Car_Ownership: Does the person own a car
- STATE: State of residence
- CITY: City of residence
- CURRENT_JOB_YRS: Years of experience in the current job
- CURRENT_HOUSE_YRS: Number of years in the current residence
- Risk_Flag: Defaulted on a loan(Target variable)

Existing solution and methodology:

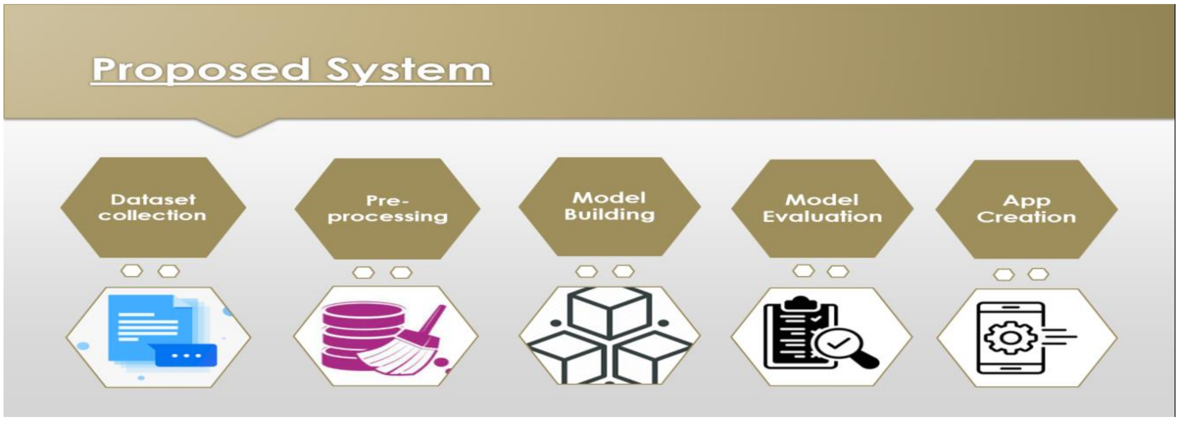
- Data is imported
- Preprocessing categorical variables with LabelEncoder, OneHotEncoder and CountEncoder
- Splitting the data into train and test splits with 20% as test set
- Uses a Random Forest Classifier
- Metrics used are Recall, Precision, F1-Score, Accuracy score, AUC Score

Proposed methodology:

- Data is imported
- Pre-processing includes dropping of unwanted columns, converting categorical variables to numeric with binarise, OneHotEncoding and CountEncoding
- Resampling the Data with Random Oversampler
- Splitting the data into train and test splits with 30% as test set
- Uses 3 different Classification Models and evaluate their metrics to find a best model for prediction
- Metrics used are Recall, Precision, F1-Score, Accuracy score, AUC Score and roc curve
- Creating a product (flask web app) for best user interface experiences

Implementation details:

Flow Chart:



Data collection and importing:

The dataset was imported in csv format using pandas dataframe and using the head function to take a glimpse at the data. The dataset had been explored using some methods like shape, size, info, describe, value_counts, etc.

EDA with visualizations:

Some analysis have been done using count plot, heatmap, box plot, histogram, and pie chart are used for study of distribution features and their relationship. Plotting relation between features and results to recognise patterns.

Data Preprocessing:

1. Dropping the unwanted column:

The following columns are dropped 'Id', 'CURRENT_JOB_YRS', 'CITY', 'STATE' had been dropped

2. Identify missing values:

The dataset had been identified for its missing values using isnull() method and sum() method to count the total no. of missing values in each features. There is no missing value found.

3. Converting categorical variables to numeric:

The following encoding will be done to the categorical features:

Marital_Status, Car_Ownership – binaries

Profession – count encoding

House_Ownership – one-hot

4. Resampling the data with Random Oversampler:

WE resample because the data is highly imbalanced.

Model Building:

X (input variables) and Y (Target Variable) are extracted from the dataet. The dataset is split into train and test with with 30% as test set and 70% as train set to perform training and testing.

1. Building predictive model using Decision Tree Classifier:

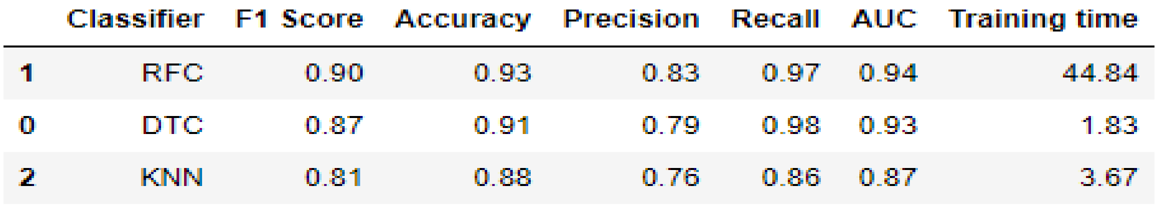
Baseline model created for DTC and yielded an accuracy of 87% with precision as 0.91 and recall as 0.79.

2. Building predictive model using Random Forest Classifier:

Baseline model created for RFC and yielded an accuracy of 93% with precision as 0.83 and recall as 0.97.

3. Building predictive model using K Nearest Neighbors:

Baseline model created for KNN and yielded an accuracy of 81% with precision as 0.76 and recall as 0.86.



	Classifier	F1 Score	Accuracy	Precision	Recall	AUC	Training time
1	RFC	0.90	0.93	0.83	0.97	0.94	44.84
0	DTC	0.87	0.91	0.79	0.98	0.93	1.83
2	KNN	0.81	0.88	0.76	0.86	0.87	3.67

Building baseline models for the above 3 classifiers and evaluating their metrics to choose the best model.

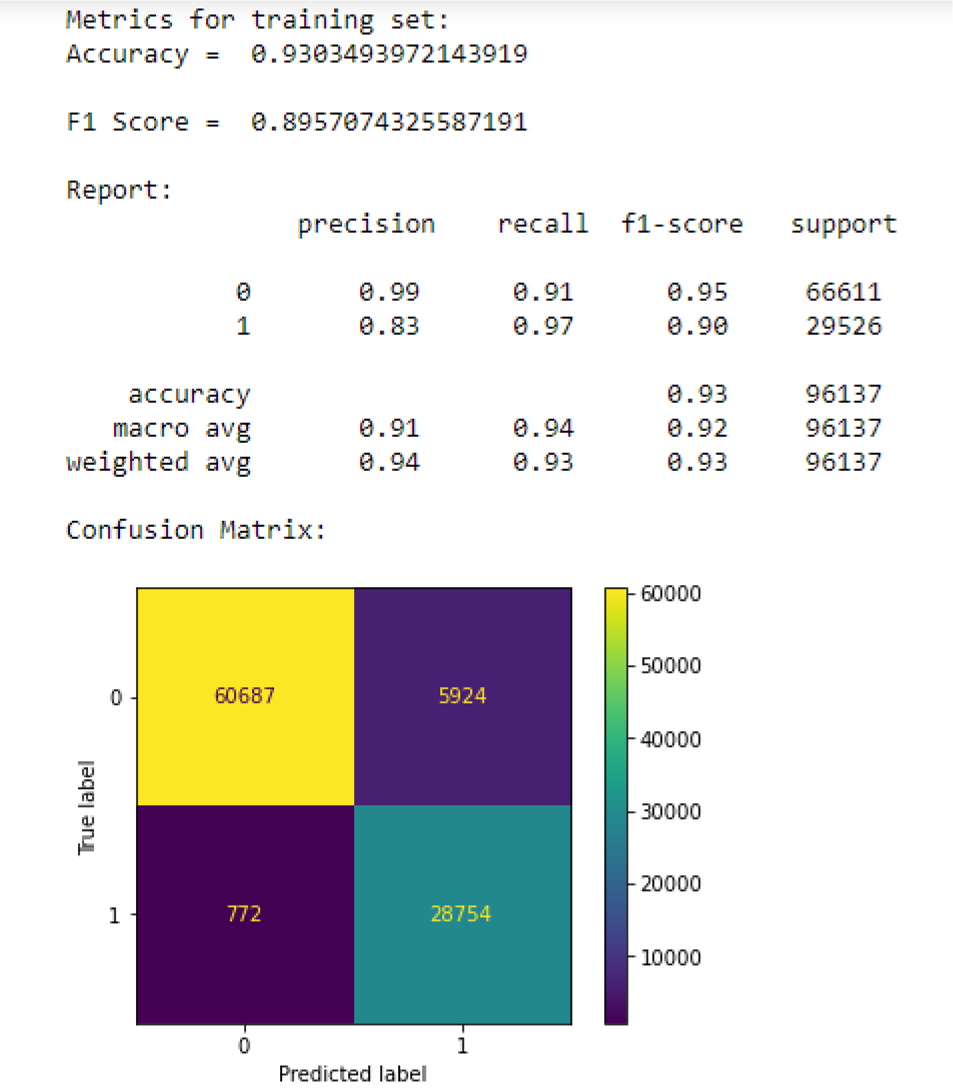
The above image shows the evaluation metrics for the baseline models for the three classifiers. Of which Random Forest out performs other models with accuracy 90% and F1-score as 0.90. Thus we choose the Random Forest Classifier as the best model.

Experimental results and evaluation:

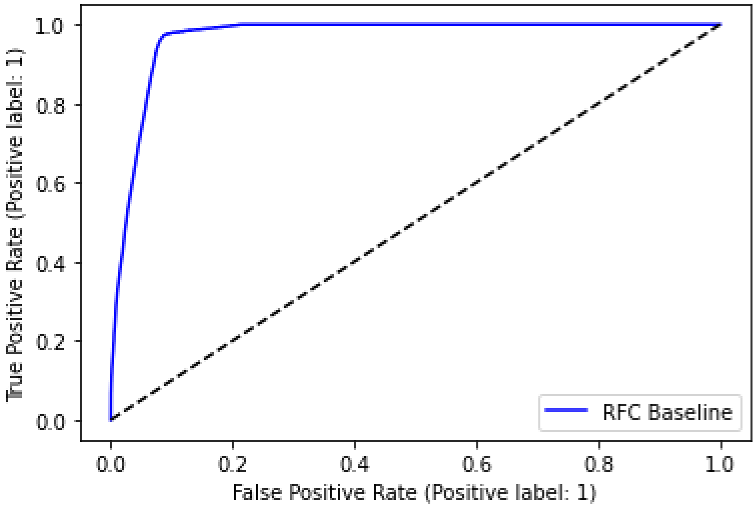
The metrics used for evaluating the models have been considered are Recall, Precision, F1-Score, Accuracy score, AUC Score and roc curve.

The baseline model for Random Forest Classifier is created and the accuracy is 93% with precision as 0.83 and recall as 0.97 which is comparatively good than other models.

Below given is the image about the different evaluation metrics, the classification report, the confusion matrix and ROC curve.



```
Metrics for training set:
Accuracy =  0.9303493972143919

F1 Score =  0.8957074325587191

Report:
               precision    recall  f1-score   support

           0       0.99      0.91      0.95     66611
           1       0.83      0.97      0.90     29526

    accuracy                           0.93     96137
   macro avg       0.91      0.94      0.92     96137
weighted avg       0.94      0.93      0.93     96137

Confusion Matrix:
```



Real-time usage:

To meet the requirements of problem statement, we establish an application to assist the organization by creating an application using flask, a module in python to satisfy the needs.

The app creation is created with some folders :

1. Static which consists of css and image files.

2. Templates which has html files.

3. The python file app.py, which has the codes for running a flask app.

4. The pickle file model.pkl, which has the developed model in it.

5. Finally includes the preprocessed and sampled data.

To run the app locally, we code “python app.py” in the terminal prompt and the link generated is copied and pasted into the local browser to view and run the application. Thus aim of the project had been proposed and satisfied.

Code repository in GitHub:

Github link: https://github.com/sharonsamsimpson/Loan_Default_Pred

Youtube Link:

https://youtu.be/wYkw2P4ix_o

Conclusion:

- The Random Forest approach is appropriate for classification tasks on datasets with many entries and features that are likely to have missing values when we need a highly accurate result while avoiding overfitting.
- Predicting Loan Default is highly dependent on the demographics of the people, people with lower income are more likely to default on loans.

The scope of understanding the case study for Loan Default PREDICTION using ML have been grasped and developed by various steps such as data collection, data preprocessing, model building and evaluation and finally creating a real-time user application.

Successfully performed the classification task using Random Forest Classifier and also improved my accuracy comparing to the existing methodology. (The source accuracy was 88% while mine case study yields an accuracy of 93% which is an improvement).

Thus the problem statement is satisfied by predicting loan default and the flask app is created successfully for best user interface experiences.

Reference URL:

1. https://www.analyticsvidhya.com/blog/2022/04/predicting-possible-loan-default-using-machine-learning/#h2_8