TREELEAF TECHNOLOGIES PVT. LTD.

MACHINE LEARNING INTERNSHIP
QUALIFICATION TASK

TASK DESCRIPTION:

#1 Task: Bank loan classification

Submitted By:

Prabin Bohara

June, 2023

Table of Contents

List of Figures and Tables

List of Tables

1 DESCRIPTION

You are given a dataset of the bank loans consisting of 15 columns and a corresponding target column. Your task is to build a machine-learning model that can accurately classify whether the personal loan was accepted or not based on the information provided.

2 Dataset

The dataset is provided in a Xlsx file with the following columns and their details:

- ID: ID of the customer
- Age: Age of the customer
- Gender: M for Male, F for Female and O for Others
- Experience: Amount of work experience in years
- Income: Amount of annual income (in thousands)
- Home Ownership: Home Owner, Rent and Home Mortgage.
- Zipcode: Postal code in which the client lives
- Family: Number of family members
- CCAvg: Average monthly spending with the credit card (in thousands)
- Education: Education level (1: bachelor's degree, 2: master's degree, 3: advanced/professional degree)
- Mortgage: Value of home mortgage, if any (in thousands)
- Securities Account: Does the customer have a securities account with the bank?
- CD Account: Does the customer have a certificate of deposit account (CD) with the bank?
- Online: Does the customer use the internet banking facilities?
- CreditCard: Does the customer use a credit card issued by the bank?
- Personal Loan: Did this customer accept the personal loan offered in the last campaign?
(Target Variable)

3 INTRODUCTION

The objective of this report is to develop a machine learning model that accurately classifies whether a personal loan was accepted or not based on the provided information. The ability to predict loan acceptance is crucial for banks as it allows them to make informed decisions and optimize their lending processes.

Classifying loan acceptance has several benefits for both banks and customers. For banks, accurate loan classification helps in minimizing risks associated with lending by identifying potential defaulters and managing their loan portfolios effectively. On the other hand, customers benefit from a streamlined and efficient loan approval process, ensuring that they receive loans that align with their financial goals.

By leveraging machine learning techniques and the given dataset, I have aimed to build a model that can effectively predict loan acceptance based on various customer attributes such as age, gender, income, education level, and more. This report will outline the steps taken to preprocess the data, perform exploratory data analysis (EDA), engineer relevant features, train the model, and evaluate its performance.

The insights gained from this analysis can provide valuable information to banks and financial institutions, enabling them to make informed decisions regarding loan approvals and optimize their resources efficiently.

4 MY APPROACH

Following steps were taken during my process of development:

1) Exploratory Data Analysis
2) Data Preprocessing
3) Feature Engineering
4) Training of model
5) Evaluation of model performance

5 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns, relationships, and characteristics of the dataset. In this phase, I conducted various analyses and visualizations to gain insights into the bank loan dataset. The key steps involved in EDA are as follows:

5.1 Univariate Analysis

- Categorical Variables: I visualized the distribution of each categorical variable using bar plots. This allowed me to understand the frequency and proportion of different categories within each variable.

- Numeric Variables: For each numeric variable, I created histograms to visualize the distribution and identify any skewness. Additionally, I plotted boxplots to identify outliers and understand the spread and central tendency of the data.

5.2 Bivariate Analysis

- Categorical Variables vs. Target: To analyze the relationship between categorical variables and the target variable 'Personal Loan,' I created stacked bar plots. This helped me observe the distribution of loan acceptance across different categories within each variable.

5.3 Handling Class Imbalance

- I examined the distribution of the target variable 'Personal Loan' to determine whether the dataset suffered from class imbalance. The value counts of loan acceptance (1) and rejection (0) were calculated and visualized. It was observed that the dataset had a case of class imbalance, with significantly fewer loan

acceptances. However, this step will be performed during the model training phase.

5.4 Handling Missing Values

- I calculated the number of missing values in each column using `df.isnull().sum()` and then determined the percentage of missing values. A bar chart was plotted to visualize the percentage of missing values in each column.

- Columns with more than 40% missing values were identified and subsequently dropped from the dataset.

The EDA phase provided valuable insights into the dataset, allowing for a better understanding of the distribution, relationships, and potential challenges. The visualizations helped in identifying any anomalies, patterns, or trends within the data. These findings will guide the subsequent feature engineering and modeling stages

6 DATA PREPROCESSING

In the data preprocessing phase, I performed several essential tasks to prepare the dataset for further analysis and model training. The steps involved are as follows:

6.1 Loading the Data

Firstly, I imported the necessary libraries such as numpy, pandas, matplotlib, seaborn, and math. Then, I used the `read_excel()` function from pandas to load the dataset from the provided Excel file into the `excel_df` dataframe. Additionally, I loaded the "Description" sheet into the `description_df` dataframe.

6.2 Understanding the Data

To gain insights into the dataset, I carried out the following operations:

- Printed the first few rows of the dataset using `excel_df.head()` to examine the column names and sample data.

- Checked the information about the dataset using `excel_df.info()` to identify missing values and data types.
- Calculated descriptive statistics of the dataset using `excel_df.describe()` to understand the distribution of numerical features.

6.3 Handling Missing Values

I addressed missing values in the dataset through the following steps:

- Checked the number of missing values in each column using `df.isnull().sum()`.
- Calculated the percentage of missing values for each column and plotted a bar chart to visualize the missing value percentages.
- Identified columns with more than 40% missing values and dropped them from the dataframe.
- Filled the missing values in numerical columns using the IterativeImputer from the sklearn library.
- Filled the missing values in categorical columns by replacing them with the most frequent value of each respective column.

6.4 Data Integration

After preprocessing the numerical and categorical data separately, I merged them back into a single dataframe named 'main_df' using the `pd.concat()` function. I excluded the unnecessary columns, 'ZIP Code' and 'ID', from the final dataframe.

The data preprocessing phase ensures that the dataset is cleaned, missing values are handled appropriately, and necessary transformations are applied, setting a solid foundation for subsequent exploratory data analysis (EDA) and feature engineering steps.

7 FEATURE ENGINEERING

Feature engineering is a crucial step in machine learning that involves transforming raw data into a format that can be effectively used by the model. In this phase, I performed several feature engineering techniques on the bank loan dataset to enhance its predictive power. The key steps involved in feature engineering are as follows:

7.1 Handling Numeric Features

7.1.1 Log Transformation

As observed during the data preprocessing phase, certain numeric features such as 'Age', 'CCAvg', and 'Mortgage' exhibited skewness. To mitigate the skewness and achieve a more Gaussian distribution, I applied log transformation to these variables. This helps in reducing the impact of extreme values and improving the linearity of the data.

7.2 Handling Categorical Features

7.2.1 Zip Code Encoding

The 'ZIP Code' column provided information related to the region of a customer's residence. To utilize this information effectively, I performed feature encoding on the 'ZIP Code' column. Since the zip codes have a hierarchical structure, I divided them into three categories: 'zip1', 'zip2', and 'zip3'. Binary encoding was applied to each category using the category_encoders library.

7.2.2 Education Encoding

The 'Education' column already had ordinal encoding (1: bachelor's degree, 2: master's degree, 3: advanced/professional degree), so no further encoding was required.

7.2.3 Other Categorical Features

Categorical columns such as 'Securities Account', 'CD Account', 'Online', and 'CreditCard' were already one-hot encoded and required no further processing.

7.3 Data Integration

After preprocessing the numerical and categorical data separately, I merged them back into a single dataframe named 'main_df' using the `pd.concat()` function. I excluded the unnecessary columns, 'ZIP Code' and 'ID', from the final dataframe.

7.4 Final Dataset

- After performing feature engineering, I combined the processed numeric and categorical features into a single dataframe, referred to as 'main_df'. The 'ZIP Code' and 'ID' columns, which were no longer needed, were dropped from the dataset.
- Any remaining missing values in the dataset were dropped using the 'dropna()' function, ensuring a clean and complete dataset for model training.

7.5 Saving the Preprocessed Data

Finally, I saved the preprocessed dataframe 'main_df' to a CSV file named 'result.csv' for further analysis and modeling purposes.

Feature engineering plays a critical role in improving the performance and predictive ability of machine learning models. By transforming and encoding the features appropriately, we provide the model with more meaningful and representative information, enabling it to make more accurate predictions.

8 MODEL BUILDING

8.1 Data Preparation

I split the dataset into the input features (X) and the target variable (y). The 'Personal Loan' column served as the target variable, while the remaining columns were considered as input features. The X DataFrame contained all columns except 'Personal Loan', and y contained only the 'Personal Loan' column.

8.2 Data Sampling

To address class imbalance, I performed random oversampling using the RandomOverSampler from the imbalanced-learn library. This technique replicated instances from the minority class to balance the classes. I resampled the X and y dataframes using the fit_resample method of the RandomOverSampler class, resulting in balanced classes for the target variable (0: No loan, 1: Loan approved).

8.3 Data Concatenation

I combined the oversampled data into a single DataFrame called 'final_df' by concatenating the resampled X and y dataframes along the columns (axis=1).

8.4 Feature Correlation

To examine the relationship between the features, I computed a correlation matrix using 'final_df'. The correlation matrix, 'corr_df', included the columns ‘Age’ through 'CreditCard' to identify significant correlations between the features.

8.5 Model Training

I trained the model using both “result.csv” and “no-drop-result.csv” datasets. I developed and evaluated four different models for predicting loan eligibility: Logistic Regression, Support Vector Machine (SVM), Random Forest, and XGBoost..

8.5.1 Logistic Regression Model

I trained a logistic regression model using the `LogisticRegression` class from scikit-learn. After splitting the data into training and testing sets, I fit the model to the training data. Then, I made predictions on the test set using the trained model. The evaluation metrics for this model are as follows:

- Accuracy: 0.827
- Precision: 0.877
- Recall: 0.756
- F1-Score: 0.812

A confusion matrix and heatmap were generated to visualize the classification results.

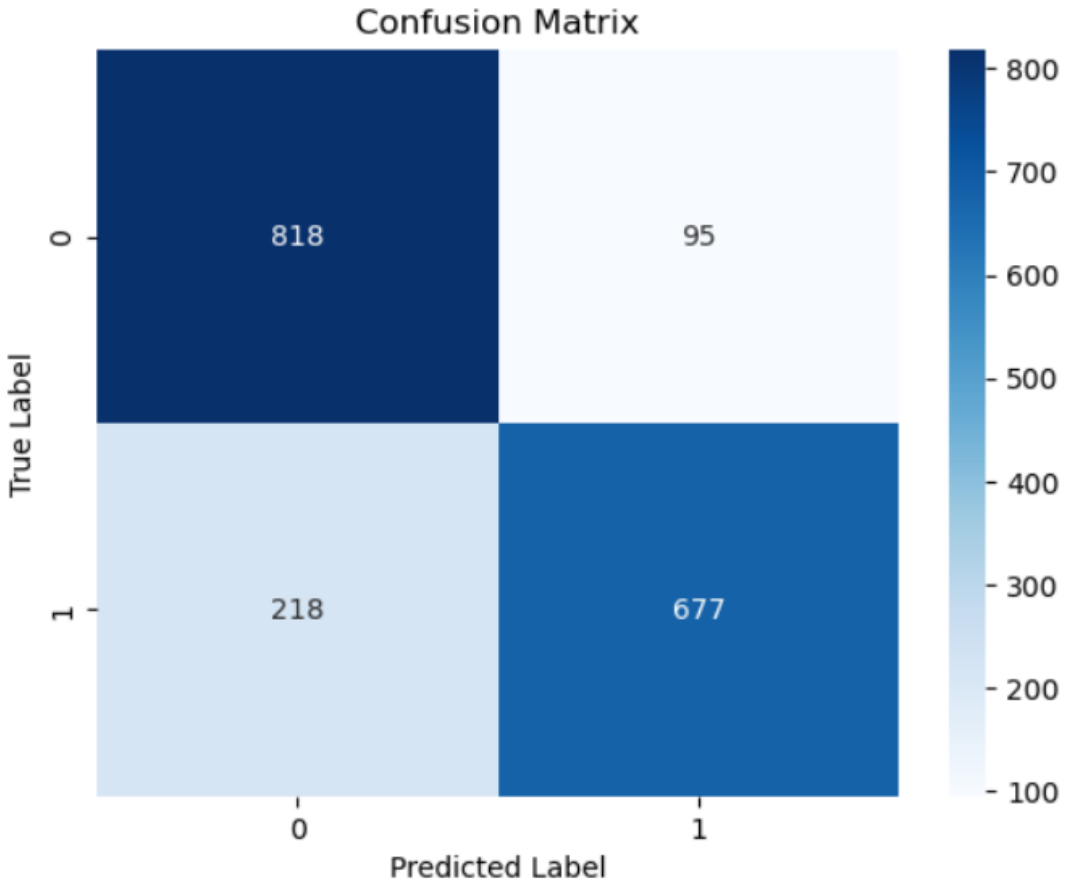

Figure 1 Confusion Matrix of Logistic Regression Model

8.5.2 SVM Model

Next, I built an SVM model using the `SVC` class from scikit-learn. Similar to the logistic regression model, I trained the SVM model on the training set and made predictions on the test set. The evaluation metrics for the SVM model are as follows:

- Accuracy: 0.529
- Precision: 0.520
- Recall: 0.651
- F1-Score: 0.578

The confusion matrix and heatmap were created to provide a visual representation of the model's performance.

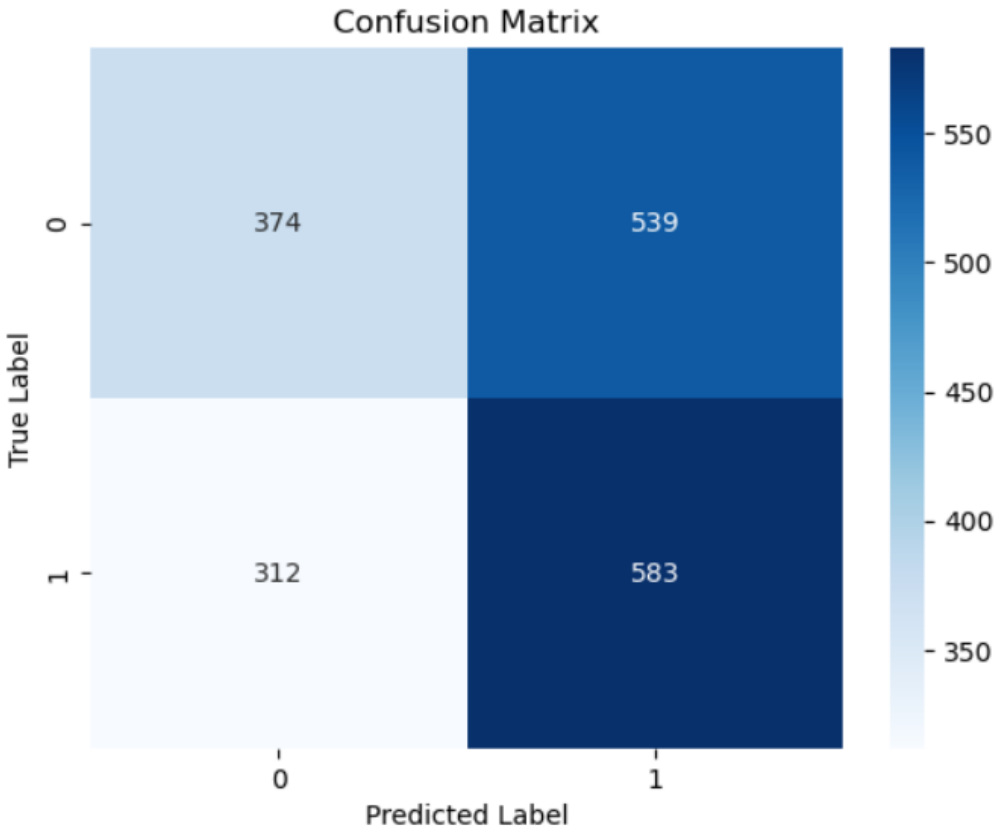

Figure 2 Confusion Matrix of Support Vector Machine

8.5.3 Random Forest Model

For the random forest model, I utilized the `RandomForestClassifier` class from scikit-learn. I split the data into training and testing sets, and then trained the model on the training data. Subsequently, I made predictions on the test set and calculated the following evaluation metrics:

- Accuracy: 0.999
- Precision: 0.998
- Recall: 1.000
- F1-Score: 0.999

The confusion matrix and heatmap were employed to visualize the classification results.

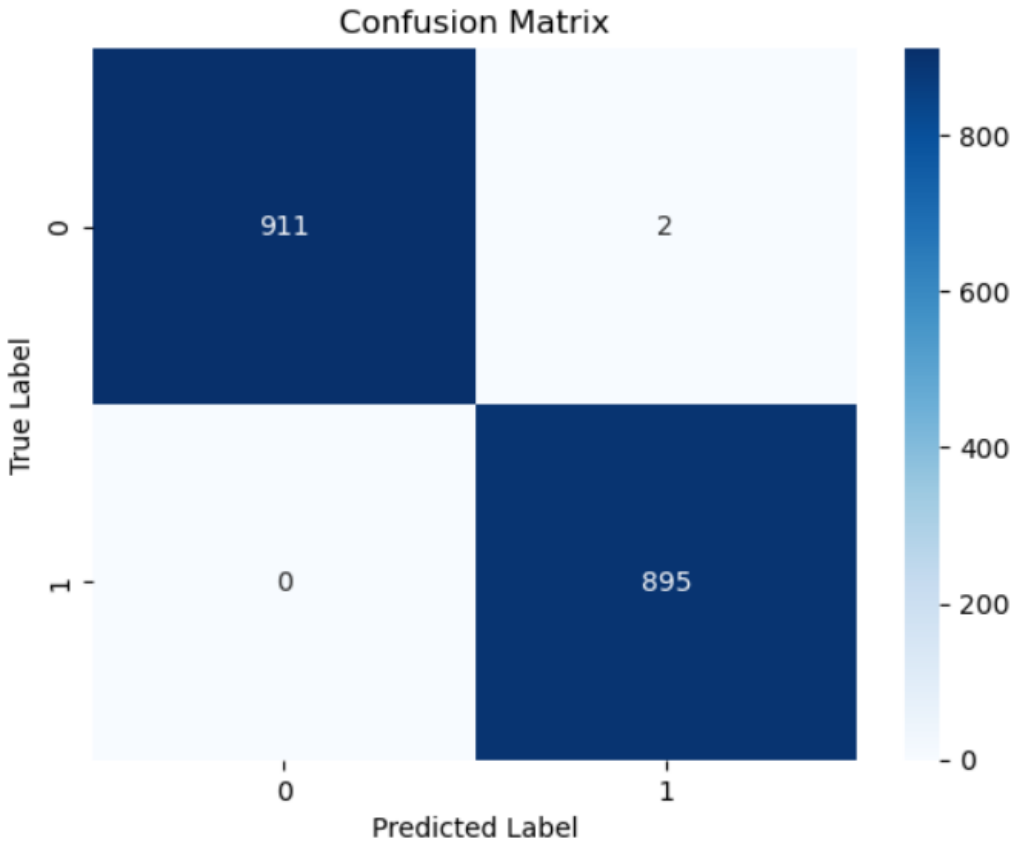

Figure 3 Confusion Matrix of Random Forest

8.5.4 XGBoost Model

Finally, I constructed an XGBoost model using the `XGBClassifier` class. After splitting the data, I trained the XGBoost model on the training set and made predictions on the test set. The evaluation metrics for the XGBoost model are as follows:

- Accuracy: 0.996
- Precision: 0.991
- Recall: 1.000
- F1-Score: 0.996

As with the other models, a confusion matrix and heatmap were generated to visually assess the model's performance.

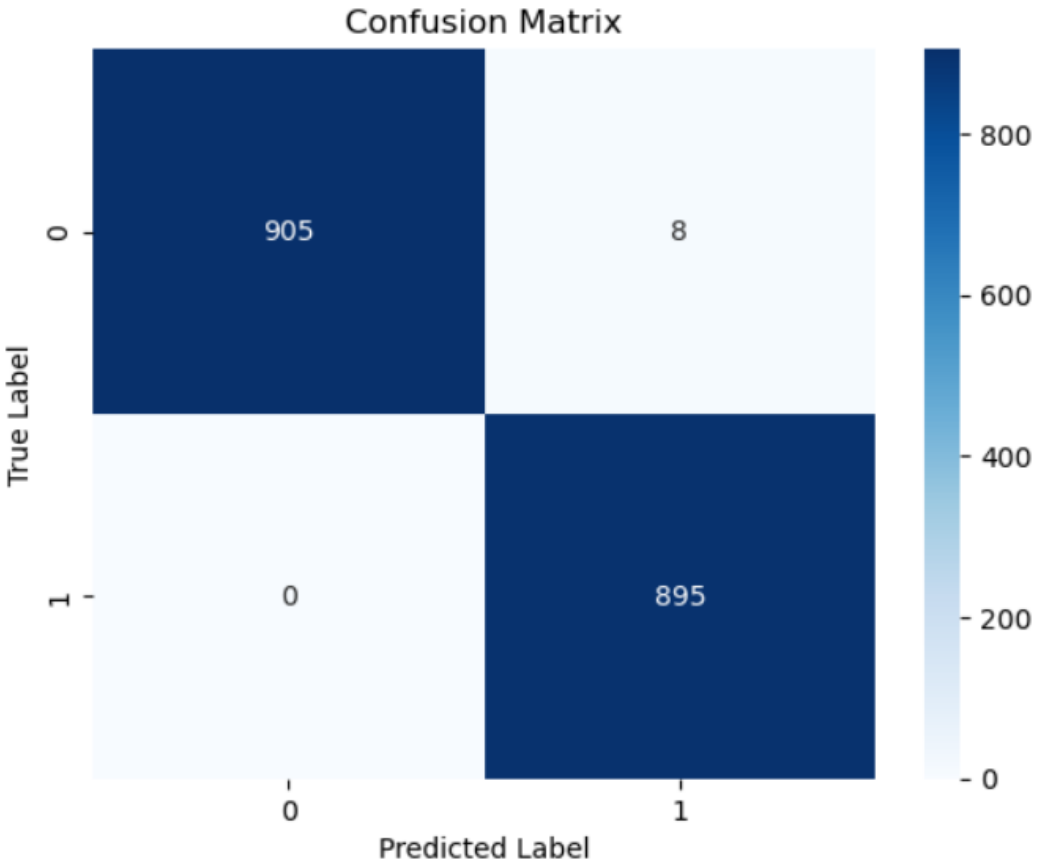

Figure 4 Confusion Matrix of XGBoost Model

Each trained model was saved using the `joblib.dump()` function with the corresponding file names: 'logistic_model.pkl', 'SVM_model.pkl', 'random_forest_model.pkl', and 'XGboost_model.pkl'.

9 MODEL EVALUATION

After completing the model building steps, I proceeded to evaluate the performance of the model and summarized in the following table.

Table 1 Comparison of different Models

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.827	0.877	0.756	0.812
SVM	0.529	0.520	0.651	0.578
Random Forest	0.999	0.998	1.000	0.999
XGBoost	0.996	0.991	1.000	0.996

In conclusion, based on the evaluation metrics, the Random Forest and XGBoost models outperformed the Logistic Regression and SVM models in terms of accuracy, precision, recall, and F1-score. The Random Forest model achieved near-perfect accuracy and high precision, while the XGBoost model also demonstrated excellent performance with a slightly lower accuracy but a comparable F1-score.

10 TRAINING MODEL TAKING ALL COLUMNS

Now, I am building the models without dropping columns like Gender and Home Ownership. The missing values of these columns are imputed using categorical imputation. The models performed with similar results.

Table 2 Comparison of different Models (When all columns are considered)

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.827	0.877	0.756	0.812
Random Forest	0.999	0.998	1.000	0.999

11 DISCUSSION AND CONCLUSION

Finally, after preprocessing the data, feature engineering and training the model, the classification matrices like accuracy, precision, recall and F1-socre were calculated. Machine Learning models like Logistic Regression, SVM, Random Forest and XGBoost were built and compared via different matrices. Among all Random Forest model performed significantly well compared to others. Also, it is worthy to note that XGBoost remains behind by very slight margin which is pretty much considerable.