



CREDIT RISK ASSESSMENT

SAIKRISHNA BALREDDY

Financial Analytics | SDBI

Table of Contents

INTRODUCTION	2
OBJECTIVE	3
STEPS FOR MODEL BUILDING	4
1.IMPORTING DATA & REQUIRED LIBRARIES :	4
2. EXPLORATORY DATA ANALYSIS (EDA):	4
3.WOE & IV:.....	6
4.FEATURE SELECTION USING CORRELATION	8
COEFFICIENT:.....	8
6. PCA FOR DIMENSIONALITY REDUCTION:.....	10
7.MODEL FITTING AND TUNING:.....	11
CONCLUSION:	14

INTRODUCTION

Credit risk refers to the risk of default or non-payment or non-adherence to the contractual obligations by a borrower. The revenue of banks comes primarily from interest on loans and accordingly, loans form a major source credit risk Banks face credit risks from financial instruments such as acceptances, interbank transactions, trade financing, foreign

exchange transactions, futures, swaps, bonds, options, settlement of transactions, and others.

Although credit risk is inherent in lending, various measures can be taken to ensure that the risk is minimized. Poor lending practices result in higher credit risk and related losses.

So the very basic task here in any Financial Institutions is to do Proper Risk assessment based on the various factors and the data available with it . Although credit risk is inherent in lending, it can be kept at a minimum with sound credit practices of which risk assessment is the very first step.

OBJECTIVE

Here we have a dataset from which we have to evaluate and develop a Model that can be used to do the risk assessment of all the applicants /borrower who form the part of dataset. The bank needs to check for the profile of the borrower in various aspects to make the risk assessment, such as credit history of the borrower, capacity to repay, capital, employment status , purpose of loan etc.

The focus of the Financial Institution is to detect the risky applicants and minimize the risk associated with lending and not let the asset turn into liability(loss) for it .So the focus of the Model developed should be to correctly determine various metrics that helps to present both positive and negative outcomes correctly.

STEPS FOR MODEL BUILDING

1.Importing Data & Required libraries :

The very first step that we did is importing the required libraries on the python lab which will be used to build the model. There are numerous libraries that we have imported each have a role to play in the model building , starting with the very basic and important library of pandas which help us to import and read the dataset in various formats.

2. Exploratory Data Analysis (EDA):

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Components of EDA

- Understanding your variables.
- Cleaning your dataset.
- Analysing relationships between variables.

In our Model we have performed EDA in two ways, Firstly using Pandas Profiling Report and secondly by creating various functions and observing interactions between the

variables in the dataset manually and plotting various graphs Which help us in understanding the data better.

EDA with PANDAS PROFILING

The profile report package from pandas profiling is a very using in creating an automated EDA it creates an output html file which has a detailed report which helps us with the variables and their types ,missing and duplicate data , interaction and correlations among variables,etc.

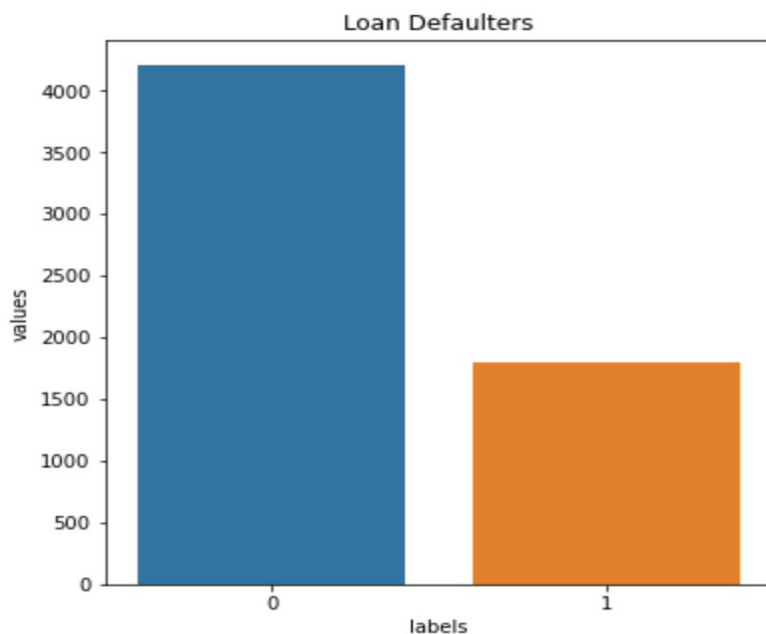
Overview

Overview Alerts 22 Reproduction	
Dataset statistics	
Number of variables	19
Number of observations	6000
Missing cells	1320
Missing cells (%)	1.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	890.8 KiB
Average record size in memory	152.0 B
Variable types	
Numeric	5
Categorical	14

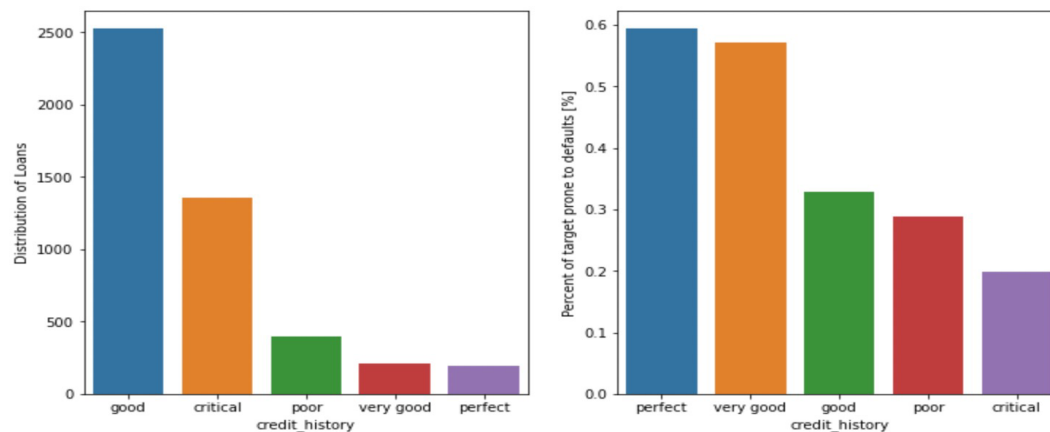
EDA with Functions and graphs:

We have created various functions with the help of which we have come across various observations , the interactions amongst the variables and the graphical representation of the data gives us better insights and is self-explanatory. For example, The bar plot below shows us the split (70-30)of defaulters in the given dataset.

```
temp = df["target"].value_counts()
df1 = pd.DataFrame({'labels': temp.index,
                    'values': temp.values
                    })
plt.figure(figsize = (6,6))
plt.title('Loan Defaulters')
sns.set_color_codes("pastel")
sns.barplot(x = 'labels', y="values", data=df1)
locs, labels = plt.xticks()
plt.show()
```



```
: plot_stats('credit_history')
```



3.WOE & IV:

The WOE(weight of evidence)tells the predictive power of an independent variable in relation to the dependent variable (variable target in our model). Since it is evolved from credit scoring world, it is generally described as a measure of the separation of good and bad customers. **"Bad Customers"** refers to the customers who defaulted on a loan.

and "**Good Customers**" refers to the customers who paid back loan.

$$\ln\left(\frac{\text{percentage of events}}{\text{percentage of non events}}\right)$$

IV (Information value) is one of the most useful technique to select important variables in a predictive model. It helps to rank variables on the basis of their importance.

$$IV = \sum_{i=1}^h (WoE_i * (\text{percentage of events} - \text{percentage of non - events}))$$

where h represents the number of bins of categories in a feature

The table below gives you a fixed rule to help select the best features for your model

Information Value	Predictive power
<0.02	Useless
0.02 to 0.1	Weak predictors
0.1 to 0.3	Medium Predictors
0.3 to 0.5	Strong predictors
>0.5	Suspicious

For example.

WOE & IV

```
In [64]: def get_iv(df, feature, target):
    lst = []
    unq_val = df[feature].unique()
    for i in unq_val:
        lst.append([feature, # Feature name
                    i, # Value of a feature (unique)
                    df[(df[feature] == i) & (df[target] == 0)].count()[feature], # Good (Fraud == 0)
                    df[(df[feature] == i) & (df[target] == 1)].count()[feature] # Bad (Fraud == 1)
                    ])
    df1 = pd.DataFrame(lst, columns = ['variable', 'value', 'good', 'bad'])
    total_good = df[df['target'] == 0].count()[feature]
    total_bad = df.shape[0] - total_good
    df1['distribution_good'] = df1['good'] / total_good
    df1['distribution_bad'] = df1['bad'] / total_bad
    df1['WOE'] = np.log(df1['distribution_good'] / df1['distribution_bad'])
    df1 = df1.replace({'WOE': {np.inf: 0, -np.inf: 0}})
    df1['IV'] = df1['WOE'] * (df1['distribution_good'] - df1['distribution_bad'])
    print(df1)
    print("\n", total_good)
    print("\n", total_bad)
    print("\n", df1['IV'].sum())
```

```
In [65]: get_iv(df, "age", "target")
```

Above we have defined a function for finding WOE & IV and called independent feature against the output feature .i.e. target to check its **Predictive power**.

4.Feature Selection using correlation coefficient:

Correlation Coefficient

Correlation is a measure of the linear relationship of 2 or more variables. Through correlation, we can predict one variable from the other. The logic behind using correlation for feature selection is that the good variables are highly correlated with the target. Furthermore, variables should be correlated with the target but should be uncorrelated among themselves.

If two variables are correlated, we can predict one from the other. Therefore, if two features are correlated, the model only really needs one of them, as the second one does not add additional information. We have used the Correlation here and displayed it with through a heatmap which helps us in finding and selecting the right features for the model .

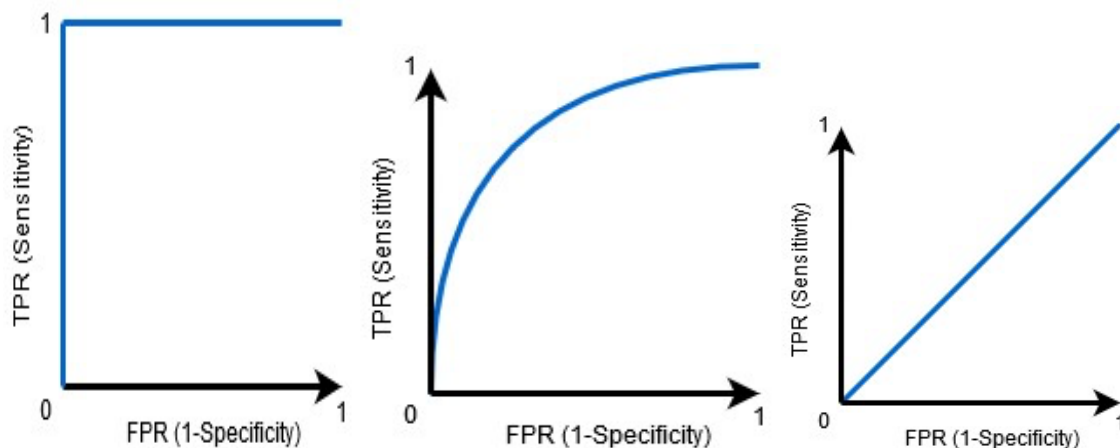
Out[91]:

	cust_id	acc_no	months_loan_duration	amount (USD)	percent_of_income	years_at_residence	age	existing_loans_count	dependants	phone
cust_id	1.000000	0.999951	-0.035236	-0.031351	-0.017903	0.003934	-0.012841	-0.005626	-0.013593	-0.008183
acc_no	0.999951	1.000000	-0.035009	-0.031098	-0.018157	0.003627	-0.012616	-0.006930	-0.013464	-0.007941
duration	-0.035236	-0.035009	1.000000	0.622775	0.072630	0.030536	-0.031360	-0.027386	-0.044311	0.175941
nt (USD)	-0.031351	-0.031098	0.622775	1.000000	-0.275884	0.034042	0.039674	-0.024590	0.028316	0.313197
_income	-0.017903	-0.018157	0.072630	-0.275884	1.000000	0.012557	0.030390	0.024988	-0.100780	-0.010914
isidence	0.003934	0.003627	0.030536	0.034042	0.012557	1.000000	0.261661	0.098284	0.047799	0.107857
age	-0.012841	-0.012616	-0.031360	0.039674	0.030390	0.261661	1.000000	0.156150	0.124728	0.155067
is_count	-0.005626	-0.006930	-0.027386	-0.024590	0.024988	0.098284	0.156150	1.000000	0.108068	0.042641
endants	-0.013593	-0.013464	-0.044311	0.028316	-0.100780	0.047799	0.124728	0.108068	1.000000	-0.010297
phone	-0.008183	-0.007941	0.175941	0.313197	-0.010914	0.107857	0.155067	0.042641	-0.010297	1.000000
target	-0.012407	-0.012834	0.192308	0.126749	0.073725	-0.006429	-0.118894	-0.057800	-0.045548	-0.030583

The feature selection helped us with selecting right features for the model . We Dropped few independent features with high correlation and got dummies for categorical values and converted the dataset into numerical one as we want to apply logistic regression model.

5.ROC & AUC:

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values and essentially **separates the 'signal' from the 'noise'**. The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



When $AUC = 1$, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

When $0.5 < AUC < 1$, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.

6. PCA for Dimensionality reduction:

Reducing the number of input variables for a predictive model is referred to as dimensionality reduction. Fewer input variables can result in a simpler predictive model that may have better performance when making predictions on new data. Perhaps the most popular technique for dimensionality reduction in machine learning is **Principal Component Analysis (PCA)**. This is a technique that comes from the field of linear algebra and can be used as a data preparation technique to create a projection of a dataset prior to fitting a model. The screenshot displayed below is the function we defined for dimensionality reduction **pca_model_results**.

PCA for Dimensionality Reduction

```
In [117]: from sklearn.decomposition import PCA

In [118]: def pca_model_results(x, y, model, comp):
            pca = PCA(n_components=comp)
            x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
            x_train = pca.fit_transform(x_train)
            x_test = pca.fit_transform(x_test)

            print('PCA Explained Variance ratio: ', pca.explained_variance_ratio_)

            # fitting the model
            model.fit(x_train, y_train)

            # obtaining model predictions from the test data
            y_predicted = model.predict(x_test)

            model.fit(x_train, y_train)

            probs = model.predict_proba(x_test)

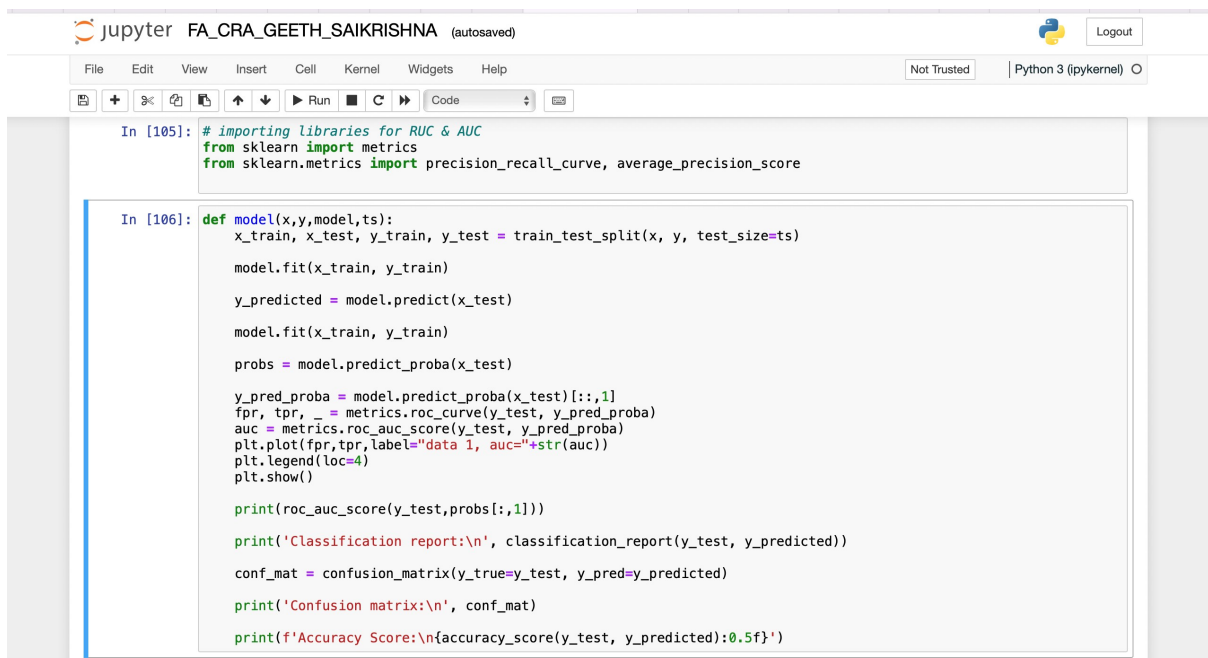
            y_pred_proba = model.predict_proba(x_test)[:,1]
            fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
            auc = metrics.roc_auc_score(y_test, y_pred_proba)
            plt.plot(fpr, tpr, label='data 1, auc='+str(auc))
            plt.legend(loc=4)
            plt.show()

            print(roc_auc_score(y_test, probs[:,1]))

            # printing the classification report and confusion matrix
            print('Classification report:\n', classification_report(y_test, y_predicted))
            conf_mat = confusion_matrix(y_true=y_test, y_pred=y_predicted)
            print('Confusion matrix:\n', conf_mat)
```

7. Model Fitting and Tuning:

We have defined a function named “model” where we are Splitting the dataset into training and test using test size and a later fitting the model passing and parameters. We are also defining and calling the ROC & AUC which displays TPR & FPR(Sensitivity & Specificity) to check probability predicting and for checking the predictive power of the model.



```
In [105]: # importing Libraries for RUC & AUC
from sklearn import metrics
from sklearn.metrics import precision_recall_curve, average_precision_score

In [106]: def model(x,y,model,ts):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=ts)
    model.fit(x_train, y_train)
    y_predicted = model.predict(x_test)
    model.fit(x_train, y_train)
    probs = model.predict_proba(x_test)

    y_pred_proba = model.predict_proba(x_test)[:,1]
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)
    plt.plot(fpr,tpr,label="data 1", auc="+str(auc))
    plt.legend(loc=4)
    plt.show()

    print(roc_auc_score(y_test,probs[:,1]))

    print('Classification report:\n', classification_report(y_test, y_predicted))

    conf_mat = confusion_matrix(y_true=y_test, y_pred=y_predicted)

    print('Confusion matrix:\n', conf_mat)

    print(f'Accuracy Score:\n{accuracy_score(y_test, y_predicted):0.5f}')
```

Classification Report:

Finally we created the Classification report that displays how the model is working on the test dataset and with the help of various parameters such as **Precision ,Recall,Accuracy,F1-Score and support.**

As our the focus of the developed model we choose the parameters

Confusion matrix :

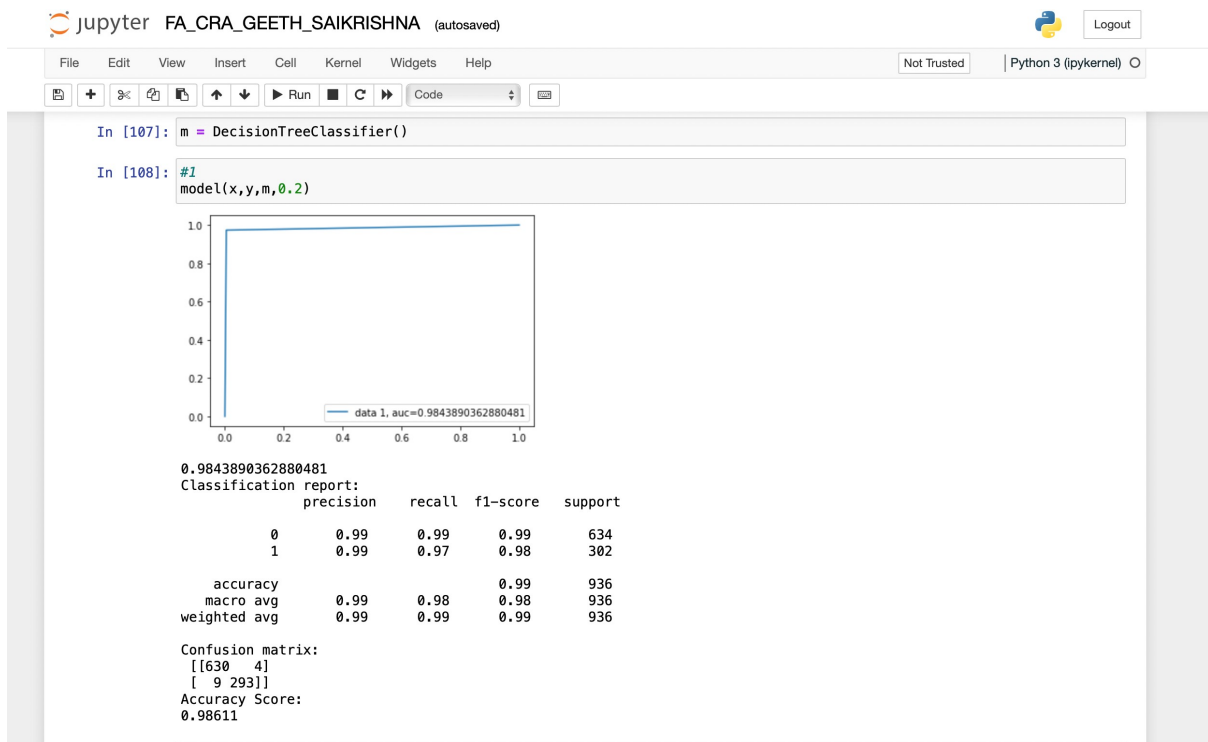
A Confusion matrix is an **N x N matrix** used for evaluating the performance of a classification model, where **N** is the number of **target classes**. The matrix compares the actual target values with those predicted by the machine learning model.

		Predicted	
		Negative (N) -	Positive (P) +
Actual	Negative -	True Negative (TN)	False Positive (FP) Type I Error
	Positive +	False Negative (FN) Type II Error	True Positive (TP)

A **good model** is one which has high **TP** and **TN** rates, while **low FP** and **FN** rates.

Accuracy is used when the **True Positives** and **True Negatives** are more important.

- Accuracy** is a better metric for **Balanced Data**.
- Whenever **False Positive** is much more important use **Precision**.
- Whenever **False Negative** is much more important use **Recall**.



CONCLUSION:

There are various metrics such as Precision , Recall, Accuracy and F1-score for along with which ROC & AUC which has given us several models of which we have to select the one that matches our requirements or Focus.

The dataset provided was a balanced dataset so a good **Accuracy score** is very much essential, along with a balance between **Precision** and **Recall** which is obtained **F-Beta score i.e. (F1-Score)**, here the value of beta is 1 which can be varied considering the focus of the user we have considered F1 in order to have a balance between **Precision** and **Recall** with low **FALSE POSITIVE** and **FALSE NEGATIVE**, the **best model**

we choose will be the one with lowest **FALSE NEGATIVE** along with **high Precision and Accuracy** as the utmost priority of a Financial Institute in **credit risk assessment** is to rightly predict all the possible defaulters and with **minimum TYPE2 error i.e. FALSE NEGATIVES.**