Bank Loan Prediction

Sourav Kumar Dokania

28th January 2021



Table of contents:

[Chapter 1: 3](#_Toc62609673)

[Introduction: 3](#_Toc62609674)

[1.1 Problem Statement 3](#_Toc62609675)

[1.2 Data 3](#_Toc62609676)

[Chapter 2: 4](#_Toc62609677)

[Methodology: 4](#_Toc62609678)

[2.1 Data Pre Processing 4](#_Toc62609679)

[2.1.1 Missing Value Analysis 5](#_Toc62609680)

[2.1.2 Type-casting data types 7](#_Toc62609681)

[2.1.3 Feature Selection 7](#_Toc62609682)

[Correlation 7](#_Toc62609683)

[2.1.4 Feature Scaling 8](#_Toc62609684)

[Normalization: 8](#_Toc62609685)

[Standardization: 9](#_Toc62609686)

[2.1.5 Outlier Analysis 10](#_Toc62609687)

[2.1.6 Creating dummies for categorical variables 11](#_Toc62609688)

[2.2 Modelling 12](#_Toc62609689)

[2.2.1 Model Selection 12](#_Toc62609690)

[Logistic Regression: 12](#_Toc62609691)

[Decision Tree Classification: 13](#_Toc62609692)

[2.3 Evaluation of the Model 13](#_Toc62609693)

[Confusion Matrix 13](#_Toc62609694)

[Logistic Model Evaluation 14](#_Toc62609695)

[Decision Tree Evaluation 16](#_Toc62609696)

[17](#_Toc62609697)

[Chapter 3 19](#_Toc62609698)

[Conclusion 19](#_Toc62609699)

[Appendix A 20](#_Toc62609700)

[Box plot for outlier analysis 20](#_Toc62609701)

[Normality check plots 21](#_Toc62609702)

# Chapter 1:

# Introduction:

* 1. Problem Statement

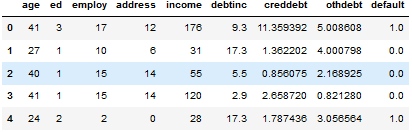
This project requires us to analyse the bank loan historical data and classify new people on the basis of the attributes as default or non-default. For a bank loan to be provided, factors such as age, income, job role, credits as well as debits are required to estimate the eligibility of the customers. Banks analyse this factors and based on this they make decision. It helps the bank to understand the requirements and get their money invested where it is probable that they will get return on time. With the help of predictor variables in the data set, various machine learning models will be applied to have an accurate classification of the new people coming on board for loan.

* 1. Data

The historical data has 8 predictor or independent variables. Based on which we will have to classify people as default on non-default for the loan status.

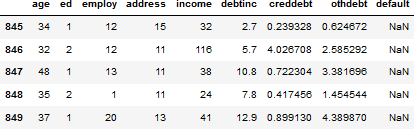
# printing first few 10 lines to see the contents of the file.

bank\_data.head(5)



# printing first few 07 lines to see the contents of the file.

bank\_data.tail(7)



There are some missing values present in the data set as can be observed above.

#checking the categorical feature “ed”

bank\_data['ed'].unique()

array([3, 1, 2, 4, 5], dtype=int64)

It can be understood there are five categories in feature “ed”.

# Chapter 2:

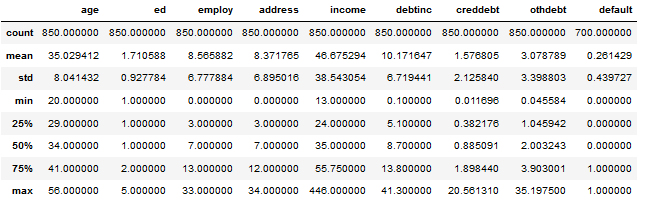
# Methodology:

2.1 Data Pre Processing

For a data to be modelled through machine learning algorithms and predict accurately for the new data, the historical data needs to be in proper shape, the variables should be processed according to the problem statement. The data at a first glance can be misleading and a lot of information cannot be understood by just the look of it. Some of the basic pre-processing techniques that will be used to process bank loan historical data are, missing-value analysis, outlier analysis, typecasting of attributes as per requirement to build a model. Normalization, standardization of data is also important as the features in the data may not be normally distributed and this may result in poor judgement of the data if we use the historical data.

# Checking the whole data and its statistics.

bank\_data.describe()



# Checking the variable “default” in the data

bank\_data["default"].unique()

array([ 1., 0., nan])

2.1.1 Missing Value Analysis

Data sets contain a lot of observations and variables. It is possible that some of data may have values that are missing. This may happen because of typing error, data entry error, or the data may not be present. Before applying machine learning algorithms the data must be evaluated for missing data, so that the prediction made by the model seems relevant. We should remove the missing values as it might lead to information loss and the model may be weak in predicting on new data sets.

missing\_values = pd.DataFrame(bank\_data.isnull().sum()) #calculating missing values of the data frame and saving in a new data frame.

missing\_values = missing\_values.reset\_index() # Reset index of the data frame to make ti more readable.

missing\_values = missing\_values.rename(columns = {"index": "Variable", 0:"Missing Values"}) # rename the column names to understand the data appropriately.

missing\_values



# Checking mode of the feature “default” as it has two levels to impute missing data.

bank\_data['default'].mode()

0 0.0

dtype: float64

# As variable "default" has two levels 0 and 1, and is categorical. The mode of the variable is 0.0 as calculated above. So filling the missing values with 0.

bank\_data['default'] = bank\_data['default'].fillna(0)

2.1.2 Type-casting data types

In due of the problem statement we can see that the feature “ed” is categorical with 5 levels as: 1,2,3,4,5. For modelling purpose we must change this variables data type from ‘int’ to ‘category’.

#changing data type for further analysis.

bank\_data["ed"] = bank\_data["ed"].astype("category")

2.1.3 Feature Selection

In this process we check the correlation of continuous variable as well as the categorical variables. We will use the Correlation process as all the variables are continuous except “ed’.

Correlation:

Correlation tells us about variable on how much they are related to each other. There are two types of correlation positive and negative. It is possible that two variables may be positively related or negatively related. If the correlation coefficient lies -8 or +8, it signifies that the variables are highly correlated. If that is the case then one of the variables must be dropped to avoid Multicollinearity.

# Code for Collinearity

num\_var = ['age', 'employ', 'address', 'income', 'debtinc', 'creddebt', 'othdebt']

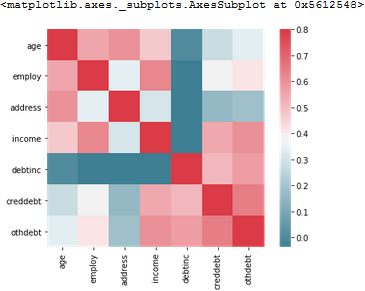
df\_corr = bank\_data.loc[:,num\_var]

f, ax = plt.subplots(figsize =(7,5))

corr = df\_corr.corr()

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True), vmax = 0.8,

square=True, ax=ax)



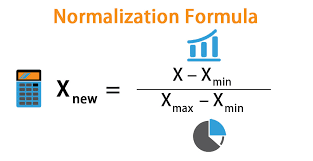
From the plot it is clearly visible that the some of the variables are moderately correlated as the values lies between 0.55 to 0.6. So there will be less effect of Multicollinearity

2.1.4 Feature Scaling

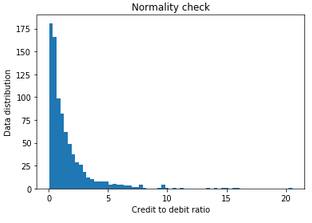
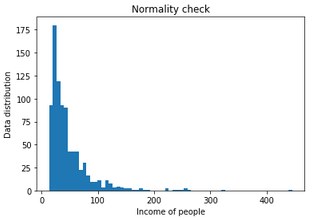
For a classification problem it is recommended that the data should be normalized or standardized. It helps to bring the data in the same range and the model will fit the data better than before.

Normalization:

In this scaling procedure the data is brought down to same range. For instance in this data the “age” and “income” features follow varying ranges. Income can be in thousand but the age cannot be thousands. So normalization is an important criteria and the formula is as follows:



First we need to check the normality of the variables, to see if the data points spread out and make a curved shape bell.



Similarly, we can check for all variables for their normality and find out, they are spread across because of varying ranges.

Imputing the normalized data for all the numerical variables.

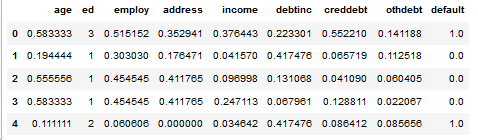
# implementing Normalization technique on the entire continuous variable.

for i in num\_var:

print(i)

bank\_data[i] = (bank\_data[i] - min(bank\_data[i])) / (max(bank\_data[i]) - min(bank\_data[i]))

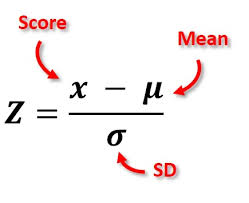
bank\_data.head() # checking the changes in the data.



It can be observed that all the data has been normalized except “ed”, which is categorical.

Standardization:

This process standardizes the data into z scores. It is necessary for classification problem as it converts stand deviation of the continuous variables to 1. Standardization is affected by outliers; because of this we will impute standardized data after imputing the outliers. The formula for the same is given as follows:

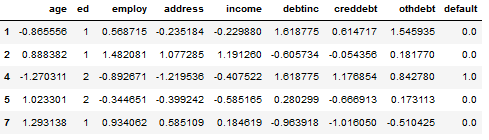


# Standardizing the data after removal of outliers.

bank\_data\_scaled = bank\_data.copy()

for i in num\_var:

bank\_data\_scaled[i] = (bank\_data[i] - bank\_data[i].mean()) / bank\_data[i].std()



2.1.5 Outlier Analysis

Some data points of variables, continuous variables, are spread wide apart. For this reason the mean, median and std of the variables are highly affected. It also creates issues in data modelling and affects the output. Outlier is a point that lies far away from the normal data points. These are mainly due to error in measurement or the data points are skewed. Outliers are very important factor to treat, to make the model approx. to the original data.

In this problem, quartiles will be used for detecting outliers and removing them.

Analysing the outliers with the help of boxplot and remove them by using quartiles.

# Code for boxplot

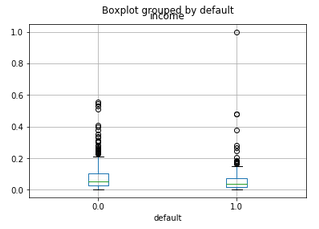
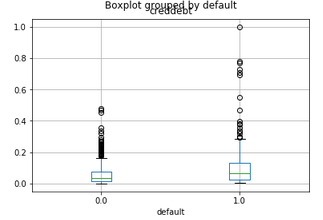
bank\_data.boxplot('income','default')

bank\_data.boxplot('age','default')

bank\_data.boxplot('creddebt','default')

bank\_data.boxplot('othdebt','default')

bank\_data.boxplot('debtinc','default')

Similarly we can check for other variables for the outliers.

# Code for detecting and removing outliers

# Detection of outliers and droping them.

for i in num\_var:

print(i)

q75, q25 = np.percentile(bank\_data.loc[:,i],[75,25]) # Divide data into 75%quantile and 25%quantile.

iqr = q75 - q25 #Inter quantile range

min = q25 - (iqr \* 1.5) #inner fence

max = q75 + (iqr \* 1.5) #outer fence

print(min)

print(max)

bank\_data = bank\_data.drop(bank\_data[bank\_data.loc[:,i] < min].index)

bank\_data = bank\_data.drop(bank\_data[bank\_data.loc[:,i] > max].index)

Here we can observe that before removing the outliers the shape of the data was:

(850, 9)

And after removing the outliers the data shape has become:

(662, 9)

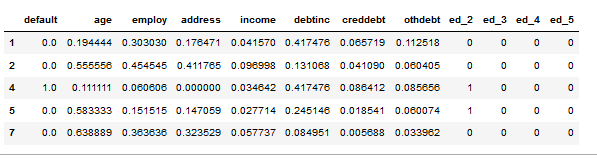
2.1.6 Creating dummies for categorical variables

In a classification problem when modelling with logistic regression, decision tress, it becomes necessary to convert categorical variables into numeric by making dummies of the variable. It is a very common practice in data science as it gives better idea for prediction.

#Creating dummy variables

temp = pd.get\_dummies(bank\_data["ed"], prefix = "ed", drop\_first = True)

bank\_data\_logit = bank\_data\_logit.join(temp)



After getting the dummies, the data has been transformed and can be seen above.

# 2.2 Modelling

2.2.1 Model Selection

At the beginning of the data analysis we understand the problem and based on it, it is decided that the problem regarding classification or regression. This information helps us in choosing machine learning models that would be suited for the given data. But it is not possible to know exactly which model would be the best fit. So in order to figure out we need to try different types of models and evaluate the model based on a new data set.

As this is a classification problem we would use below mentioned algorithms for building models, and based on the evaluation we will decide which model to use for predicting loan.

1. Logistic Regression.
2. Decision Tree.

Logistic Regression:

Logistic Regression is mainly used for classification problems. It can predict binomial, multinomial as well as ordinal variables. It is different than linear regression in a way that it has label and target variables consists of two values. In this data set the target variable we will be using binomial type as we have 0 and 1 in the target variable.

This model works on predicting probabilities of each outcome and by choosing threshold limit we can classify the predictions into 0 or 1. By default the threshold is set to 0.5.

**Train and Test data set splitting:**

First we split the data into train and test, so that we can check our model on some unseen data and see how it predicts.

# Code to split

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x\_features, y\_target, test\_size = 0.3, random\_state = 42)

**Logistic regression code:**

logreg = LogisticRegression() #Creating the classifier

logreg.fit(X\_train, y\_train) #fitting the data into the logistic model

Decision Tree Classification:

This model as the name specifies builds tree with the features data to predict the target variable. It takes decision based on the impurities calculated, and chooses one feature as its top node for classification. Eventually the tree is subdivided into subsets based on the algorithm. The final output of the tree consists of leaf which is used for classifications. This model has parameters such as max depth of the tree, min-leaf nodes of the tree which are used for better performance of the model on the data.

# code for decision tree

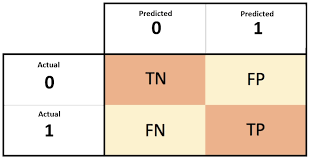
dt = DecisionTreeClassifier(max\_depth = 2, criterion = 'entropy', random\_state = 42)

dt.fit(X\_train, y\_train)

# 2.3 Evaluation of the Model

Confusion Matrix

This evaluation matrix is used for classification model to understand how well the predicted values on the test data set match with the actual values. This gives an idea on how well our model will perform on a real time data.



Let’s understand the terms:

1. TP - Predicted value matches the actual value.
2. FP - Predicted value was wrong.
3. TN – Predicted value matched the actual value.
4. FN – Predicted wrong values.

Below are some metrics that can be calculated to understand better the significance of the confusion matrix.

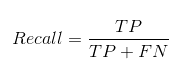
1. Accuracy: *( It tells us how accurate our model is in predicting)*

Equation_Accuracy

1. Precision: *( It tells us how many cases were actually positive)*

Confusion Matrix Precision

1. Recall: *( It tells us how many actual cases were predicted correctly by the model)*



Logistic Model Evaluation

**Confusion Matrix**

[[153 0]

[ 44 2]]

**Classification Report**

precision recall f1-score support

0.0 0.78 1.00 0.87 153

1.0 1.00 0.04 0.08 46

accuracy 0.78 199

macro avg 0.89 0.52 0.48 199

weighted avg 0.83 0.78 0.69 199

**ROC curve**

It tells about the variation of the predictions as the threshold is varied for a binary classifier.

#computing roc curve values such as fpr, tpr, thresholds

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_prob)

plt.plot([0, 1], [0, 1], 'k--')

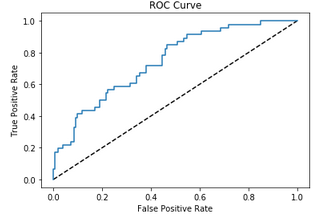
plt.plot(fpr, tpr)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.show()



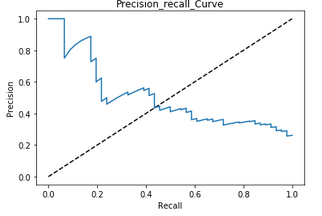
**Model ROC curve area : 75%**

The closer the curve gets to the top left of TPR the better the model is and the close it gets to the diagonal line, the less accurate the model is.

Area under the curve indicates the model ‘s score to perform on the data and the quality of classification.

**Precision-Recall Curve**

It is a graph for precision vs recall i.e total positive cases by total correctly classified positive cases.



**Cross Validation metric**

By this we can first check the error on the trained model itself. Then we can compare the error of prediction on test data set as well as the train data set by finding the MSE of all the methods.

#Cross Validation code.

MSE\_logreg = - cross\_val\_score(logreg, X\_train, y\_train, cv = 10, scoring = 'neg\_mean\_squared\_error', n\_jobs = -1)

print('LogisticRegression MSE: {:.2f}'.format(MSE\_logreg.mean())) # MSE of Logistic Regression

print('Train MSE: {:.2f}'.format(MSE(y\_train, x\_pred))) # MSE of Train

print('Test MSE: {:.2f}'.format(MSE(y\_test, y\_pred))) # MSE of Test

LogisticRegression MSE: 0.19

Train MSE: 0.19

Test MSE: 0.22

Decision Tree Evaluation

Confusion Matrix:

[[145 8]

[ 34 12]]

Classification report:

precision recall f1-score support

0.0 0.81 0.95 0.87 153

1.0 0.60 0.26 0.36 46

accuracy 0.79 199

macro avg 0.71 0.60 0.62 199

weighted avg 0.76 0.79 0.76 199

**Cross Validation report:**

#code

MSE\_cv = - cross\_val\_score(dt, X\_train, y\_train, cv = 10, scoring = 'neg\_mean\_squared\_error', n\_jobs = -1)

# Printing MSE values of cross validation, train data and test data w.r.t to predictions

from sklearn.metrics import mean\_squared\_error as MSE

print('Cross\_validation MSE: {:.2f}'.format(MSE\_cv.mean())) # MSE of CV

print('Train MSE: {:.2f}'.format(MSE(y\_train, y\_pred\_train))) # MSE of Train

print('Test MSE: {:.2f}'.format(MSE(y\_test, y\_pred\_test))) # MSE of Test

Cross\_validation MSE: 0.17

Train MSE: 0.16

Test MSE: 0.21

***It can be seen that Cross\_validation MSE is > MSE of Train. (max\_Depth = 4)***

***Cross\_validation MSE: 0.20***

***Train MSE: 0.14***

***Test MSE: 0.25***

***This implies that that data points are overfitting in the model.***

***Now we will check with changing the complexity such as max\_depth = 3 of the decision tree, to improve the model.***

***Cross\_validation MSE: 0.19***

***Train MSE: 0.16***

***Test MSE: 0.22***

***Now Cross\_validation MSE is = MSE of Train > Test MSE***

***Now we will check with changing the complexity such as max\_depth = 2 of the decision tree, to improve the model.***

***Cross\_validation MSE: 0.17***

***Train MSE: 0.16***

***Test MSE: 0.21***

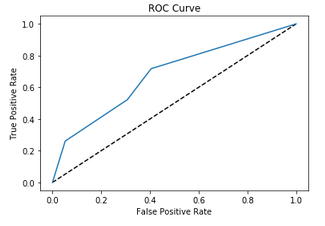
# 

When,

CV score > Train MSE ( Overfitting)

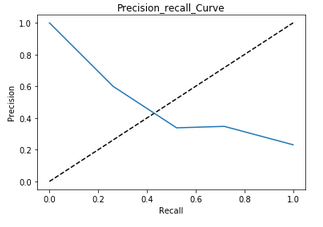
CV score ~ Train MSE >> Desired error (underfitting)

**Roc** **Curve**:



**Model ROC curve score score : 69%**

**Precision Recall Curve:**

****

# Chapter 3

# Conclusion

Logistic Regression Score:

Accuracy = 0.7788944723618091

FNR = 0.9565217391304348

Decision Tree Score:

Accuracy = 0.7889447236180904

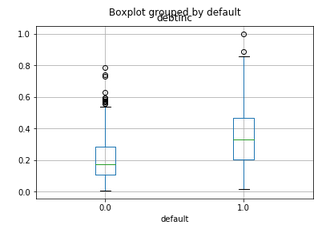
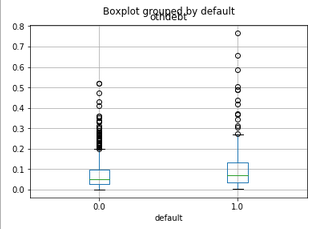
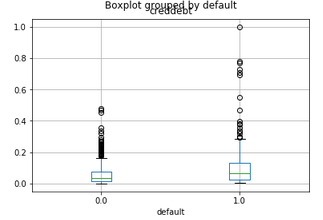
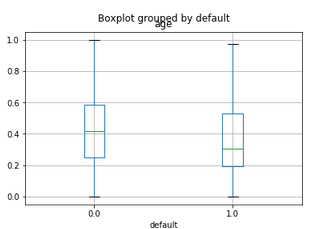
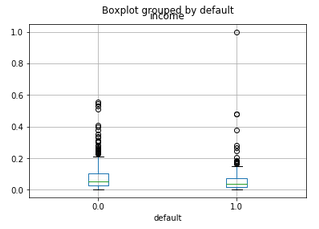
FNR = 0.7391304347826086

Here, it can be observed that both Logistic Regression model and Decision Tree Classification Model has an accuracy of 78 and 79 % respectively. However, FNR rate is less in Decision Tree Model.

So we choose Decision Tree Classification Model as our Model for the Dataset.

# Appendix A

Box plot for outlier analysis



Normality check plots

