Group – 09 Project Report

Title: Data mining on Loan Approval prediction

Dheeraj Kashyap Varanasi	Wt3631
Mohan Srinivas Vendra	Us2205
Sesha Pavan Sri Nehith Lodagala	Ii9220
Darshitha Suravarapu	Kd1215
Sai Charan Attili	Id7353

Problem statement:

Whether a loan would be approved or not with respect to parameters present in the application information that are provided by different financial companies through different means?

Data Source:

We have collected from Kaggle, which is a community platform for data analysis and machine learning.

https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset?select=train_u6lujuX_CVtuZ9i.csv

Introduction and Motive:

Many individuals across the world ask for various loans, whether for themselves or for other reasons. Even with low applicant income, they occasionally seek greater loan amounts, and as a result, their loans may be denied. To correctly educate them and determine the right amount of loan to apply for, our model would be a good choice. When applying for a loan, it is important to consider a person's credit history, including whether or not they complete their payments on time for their finance bills. This is also one of the factors that banks and other financial institutions should stress to their customers.

The importance of credit history in applying for a loan has caught our attention and the main question raised in our minds, is it important for approving a loan. The bank or the financial company runs a profile through a credit score software to just identify whether they can get the loan or not then other factors might come into play.

As this is a classification problem, we are using all classification models which are KNN classification, Decision Tree Classification, Random Forest Classification, and finally Logistic Regression. Finally, we would compare accuracy on validation data for a better data mining method to apply for the dataset to predict for a new customer.

Dataset Information:

We have collected data from Kaggle which is an open community for many machine learning projects.

This data contains a total of 614 records with 13 columns and the column information and description is as given below.

Variable	Description
Loan_ID	A unique ID given to every applicant.
Gender	Male, Female
Married	Yes, No
Dependents	No. of people dependent on the applicant - 0,1,2,3+.
Education	Education level of the applicant (Graduated, Not Graduated).
Self_Employed	Yes or No.
ApplicantIncome	The Income earned by applicant.
CoapplicantIncome	The Income earned by Co – applicant.
LoanAmount	The amount requested for the Loan.
Loan_Amount_Term	No of Days the loan will be repayed.
Credit_History	1- has all debts paid in time, 0- not paid.
Property_Area	The type of location where the applicant's property lies (Rural, Semiurban, Urban).
Loan_Status	Y or N (Yes or No)

Loan_ID is the unique number given to customers to identify their information based on that ID, it will be given after the information is stored in the database of a company. Loan_Status is the output that we need to predict based on the other data present in the dataset information.

Approach to the problem:

1. **Having some research questions:**
   i. Why is credit history important for a person to get loan accepted?
   ii. How does the loan Amount varying with Applicant Income and co applicant Income?

iii. If Applicant has dependents is that effecting the approval status?
iv. What is the average loan amount they are requesting for, if they have dependents?
v. If the loan gets accepted, then what is loan term that they are expected to get to clear the loan?
vi. What are the models that can be applied to predict the output?
vii. Apart from the loan status if we have loan amount to be predicted is it possible or not?

These are some research questions that showed us a way to understand the problem and visualize accordingly to give some strategies to the companies to get more approvals than rejections.

2. **Define goal:**
   Our goal is to find the best model that would predict the loan approval status based on the parameters that are present in our information collected.

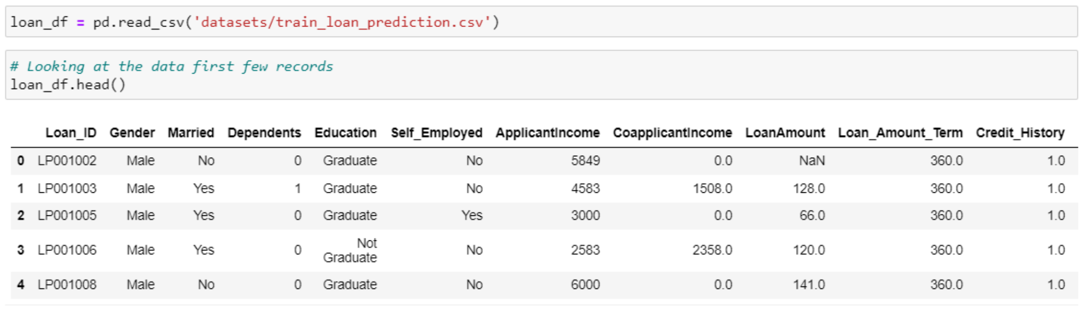
3. **Data Pre – processing and Analysis:**
   We correct the data properly by filling out the null values or any more things to do with the data and explain the relation between variables using the data visualization. Which gives a proper understanding of the variables and helps us choose predictors and then apply the model algorithm.

4. **Applying the models:**
   After knowing the predictors, we would partition the data into training and validation, then start applying the models to the training data to train it properly afterwards we check on the validation data for accuracy measures.

5. **Choosing the best model:**
   After step 4 we gather all the accuracy measures in a table on validation then we decide which method to go ahead with and use it to predict for the new customers or the new applications.

Data Pre – processing:

1. Importing the data and just looking at the first few records to understand the data which can help removing some unwanted columns in the dataset during model application.

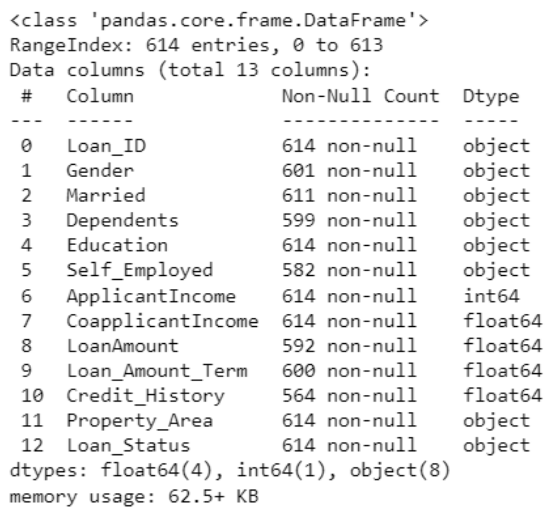


```
loan_df = pd.read_csv('datasets/train_loan_prediction.csv')
```

```
# Looking at the data first few records
loan_df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0

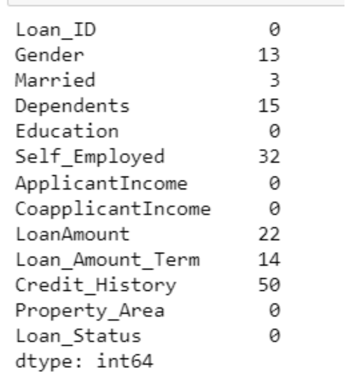
2. To know which are the numerical variables or the categorical variables in the dataset we need use info () function so that we can get the data types of each column.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In this dataset there are a total of 5 integer variables and 8 object variables in which we can find some categorical variables.

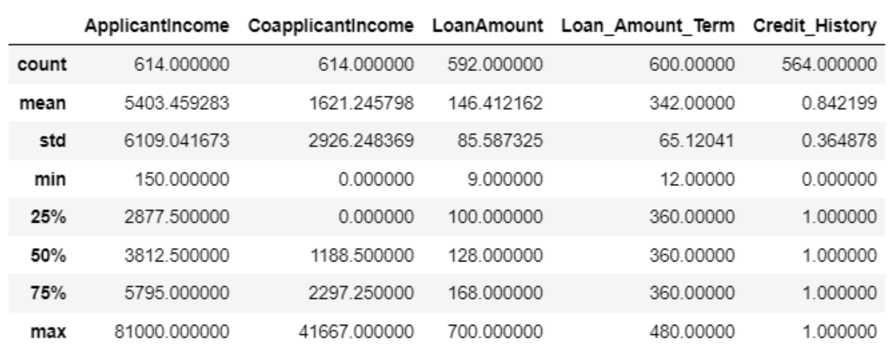
3. Looking at the number of null values in the dataset.



```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Now the null values are present in Gender, Married, Dependents, Self – Employed, Loan Amount, Loan_Amount_Term and Credit_History.

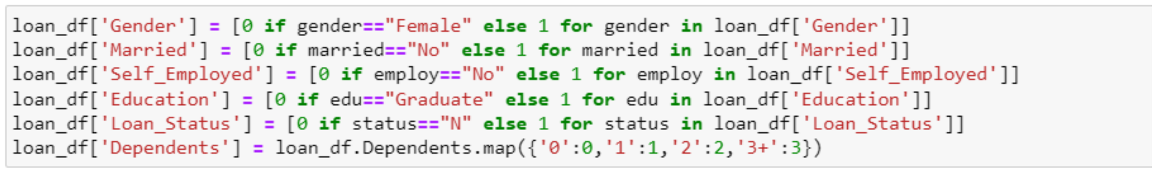
4. Just looking at the description of the data to fill the null values in the numerical varaibales.



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

   a. The average income of an applicant is 5403 dollars, where the average loan amount they are applying for is 146.14 thousand dollars.
   b. The mean of a person having credit_history is 0.84.
   c. The Applicant with average income of 5403 dollars is having a co – applicant average income as 1621 dollars.

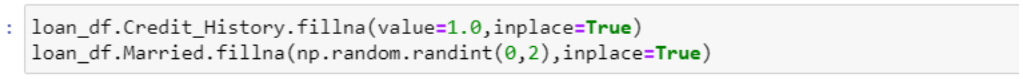
5. Converting all the categorical variables with the string values to Boolean ones using the conditional statement.



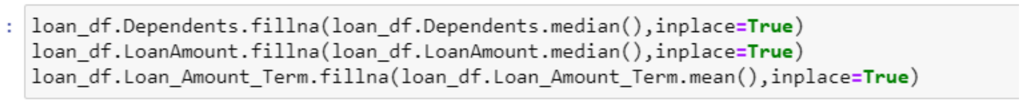
```
loan_df['Gender'] = [0 if gender=="Female" else 1 for gender in loan_df['Gender']]
loan_df['Married'] = [0 if married=="No" else 1 for married in loan_df['Married']]
loan_df['Self_Employed'] = [0 if employ=="No" else 1 for employ in loan_df['Self_Employed']]
loan_df['Education'] = [0 if edu=="Graduate" else 1 for edu in loan_df['Education']]
loan_df['Loan_Status'] = [0 if status=="N" else 1 for status in loan_df['Loan_Status']]
loan_df['Dependents'] = loan_df.Dependents.map({'0':0,'1':1,'2':2,'3+':3})
```

   Dependents are mapped to 0,1,2 and 3 instead of 0,1,2,3+.

6. Filling out the null values as following picture presented in the below picture,

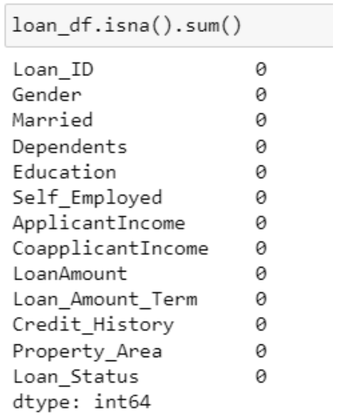


```
loan_df.Credit_History.fillna(value=1.0,inplace=True)
loan_df.Married.fillna(np.random.randint(0,2),inplace=True)
```



```
loan_df.Dependents.fillna(loan_df.Dependents.median(),inplace=True)
loan_df.LoanAmount.fillna(loan_df.LoanAmount.median(),inplace=True)
loan_df.Loan_Amount_Term.fillna(loan_df.Loan_Amount_Term.mean(),inplace=True)
```

   Credit_history is filled with 1.0 as the mean is close to 1 the married is filled, Dependents are filled with the median, Loan amount term is filled with average value.
   Now the null in the data are,



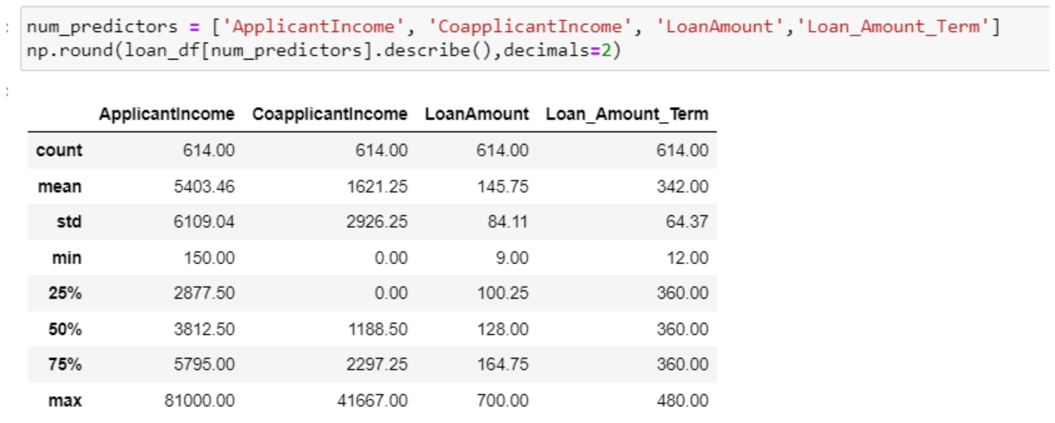
```
loan_df.isna().sum()
```

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

After the pre – processing of the data we need to get a visual analysis on the data.

Data Visualization:

1. Description of the data for numerical variables after filling all the null values

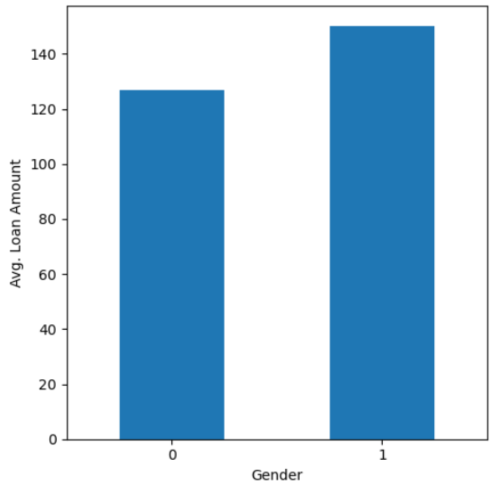


```
num_predictors = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount','Loan_Amount_Term']
np.round(loan_df[num_predictors].describe(),decimals=2)
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
count	614.00	614.00	614.00	614.00
mean	5403.46	1621.25	145.75	342.00
std	6109.04	2926.25	84.11	64.37
min	150.00	0.00	9.00	12.00
25%	2877.50	0.00	100.25	360.00
50%	3812.50	1188.50	128.00	360.00
75%	5795.00	2297.25	164.75	360.00
max	81000.00	41667.00	700.00	480.00

- The data has a mean applicant income of 5403.46 and standard deviation is 6109.04. This shows how data is deviated and spread across the dataset.
- The applicant with an average income of 5403 dollars and co – applicant income of 1621 dollars are applying for a loan amount of around 145.75 thousand dollars.
- The max applicant income is 81000 dollars, and the Loan Amount is at maximum of 700 thousand dollars.

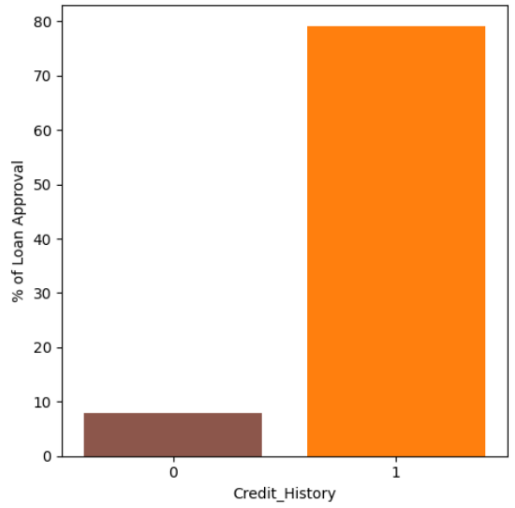
2. A bar graph between Gender and average loan amount



0 – Loan Rejected   1 – Loan Approved

In this graph the males are applying higher loan amount as the average is around 145 thousand dollars and Females are applying a loan of around 123 thousand dollars. The bank can target females to apply for a larger amount of loans with the decrease in interest rate. But this cannot be a big problem to the banks or any financial company.

3. Graph between Credit History and percentage of loan Approval:

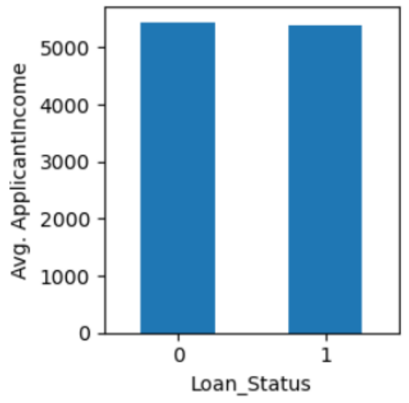


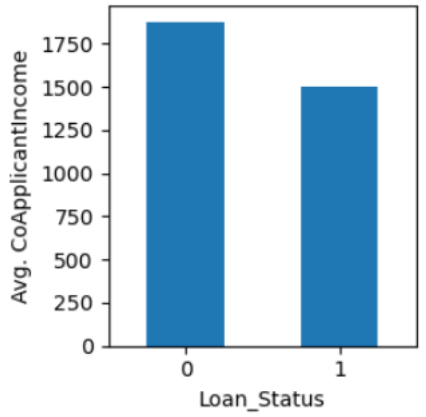
```
Credit_History
0.0     7.865169
1.0    79.047619
Name: Loan_Status, dtype: float64
```

The persons with good credit history are getting approved for loan which is around 79.04%, with no credit history are having less approval percentage 7.8% and this percentage of people might be new to the loan applications and are applying to maintain certain credit history in their financial journey. The bank or company should target the audience to educate the customers about credit history and how it is being used in the loan application process which would be great positive for the customers and the banks as well strategize their loans.

This can be a good factor to predict the loan_status.

4. Graph between loan status and applicant income, co – applicant income

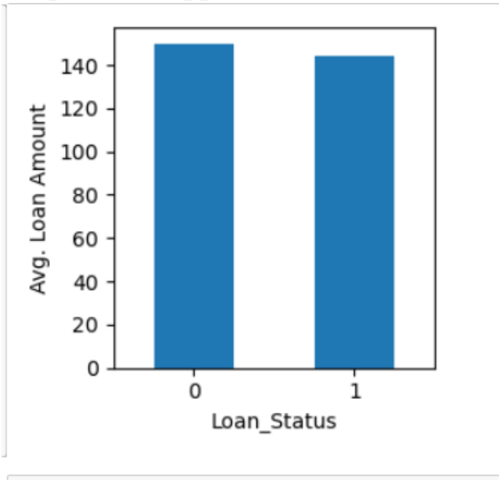




- From the graphs we are unable to draw some conclusions as the loan is getting accepted with the average co – applicant income around 1400 dollars their might be some other reasons like the co applicant with low income are applying a loan with optimum amount may be

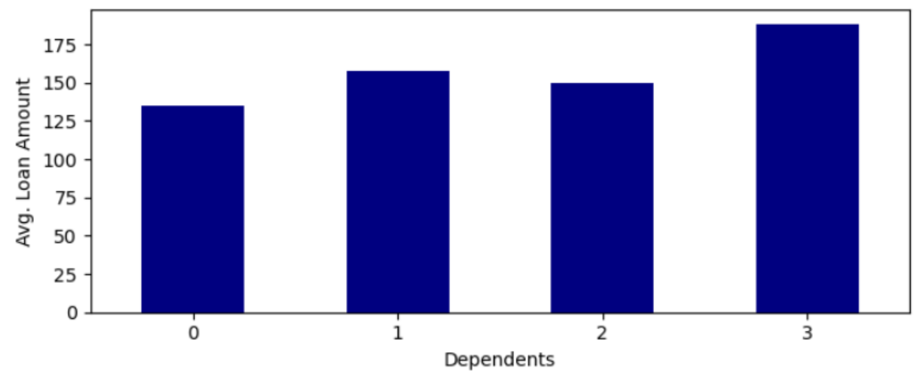
- The Applicant Income is almost similar for both rejected and accepted hence these two might be a factor as they relate to the loan amount to be applied, We can use them as the predictors but not exactly as strong as credit history.

5. Graph Loan Approval status and Average loan amount

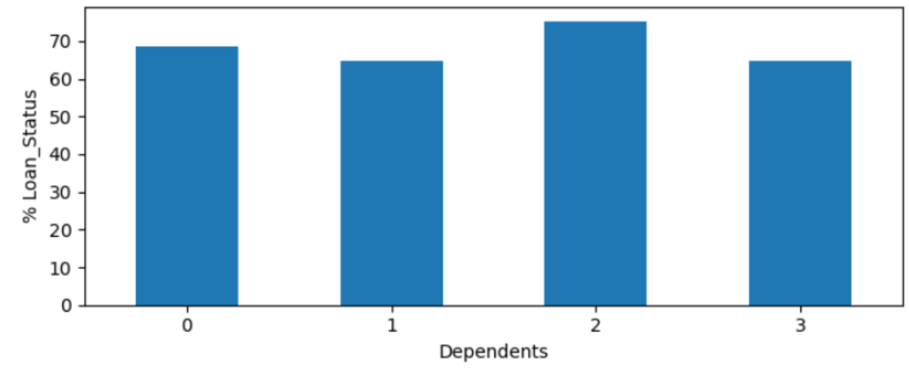


The loan is getting rejected for an average loan amount of 145 thousand dollars but is getting approved for an average loan amount of 140 thousand dollars. Hence, we need to tell the customers that they can apply for a certain amount so that customers can plan accordingly to get their loan approved, which will also be of benefited to the company as well.

6. Graph between Dependents and Loan Approval status with an average loan amount



In the above Image the people with dependents 3 are applying for a large amount of loan while compared to the people with dependents less than 3 are applying for lower amount of loans.

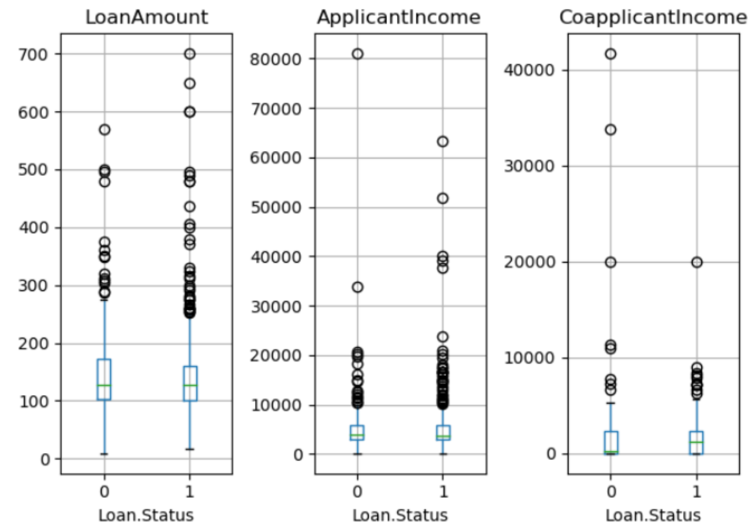


```
Dependents
0.0    68.611111
1.0    64.705882
2.0    75.247525
3.0    64.705882
Name: Loan_Status, dtype: float64
```

The customer with 3 dependents has a low percentage of loan approval compared to others, then the amount of loan they are applying for is not in the optimal range. Hence, they needed to know about the reason for rejection. They need to understand about the amount that is required to apply so that their loan will not be rejected.

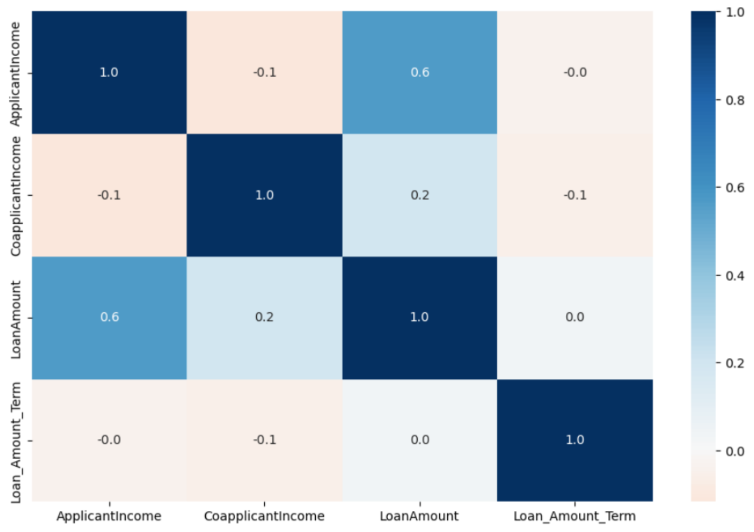
From both the images we could understand that the loan_amount they are applying for is an important factor hence, Loan amount becomes one of our predictors in the machine learning models.

7. Graph of boxplots for all the numerical variables to have a sight on the outliers:



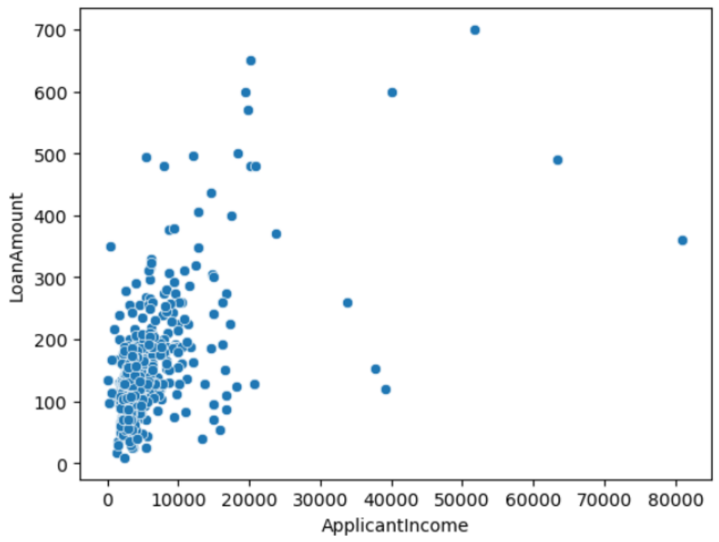
From the above plot we can understand that there are many outliers in all three variables, but the data is being more dispersed in loan amount and the outliers are more in the Applicant Income. We can say that our dataset is somewhat imbalanced. Hence, we are taking out the outliers as the dataset is small with only 614 records.

8. Autocorrelation plot for the numerical variables



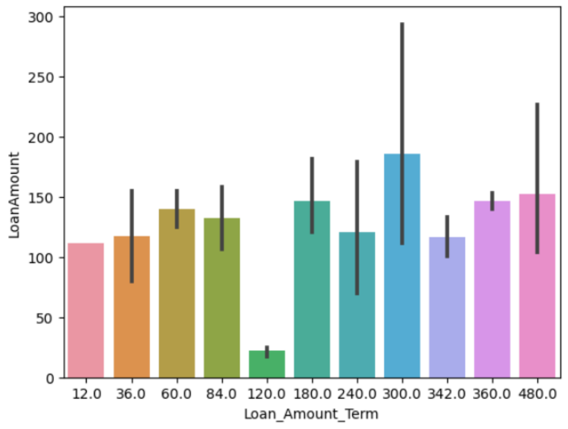
In this plot we can understand that Loan Amount and Applicant Income are related to each other with an autocorrelation coefficient of 0.6. Previously, there is a hesitation in selecting the Applicant Income as our predictor as the visual cannot give a proper illustration. With this plot we can clearly take applicant Income as a predictor.

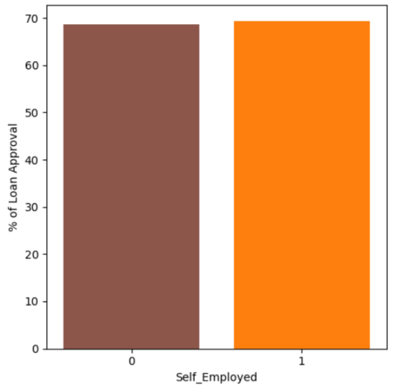
This can also be assumed through the following scatter plot:



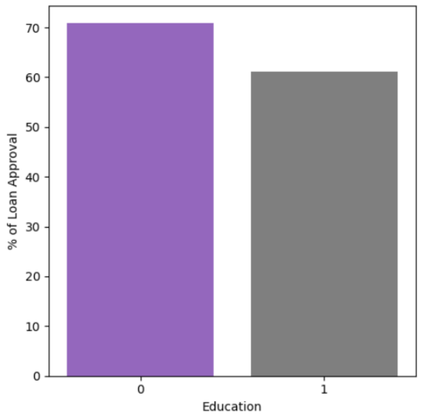
If applicant Income increases, then the loan amount might also increase but it is not completely linear there might be some reverse relation as well. Hence, we can take both as the predictors for the models.

Some other visualizations that we have taken is,





```
[67]: Self_Employed
      0    68.600000
      1    69.298246
      Name: Loan_Status, dtype: float64
```



```
56]: Education
     0    70.833333
     1    61.194030
     Name: Loan_Status, dtype: float64
```

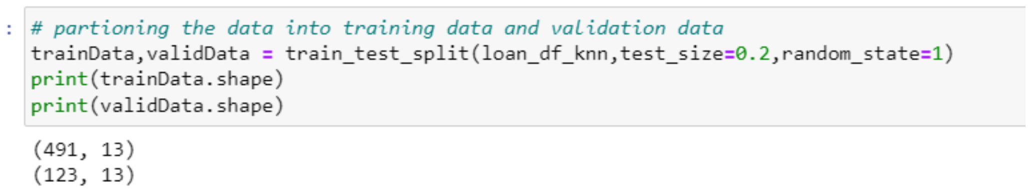
<u>**Model Building:**</u>

In data mining it is important to train the data with some relevant models and finding the accuracy on the validation data.

For this project we have gone with four methods of classification:

1. KNN Classification
2. Decision Tree Classification
3. Random Forest Classifier
4. Logistic Regression

We use these models and find all the accuracies of all models on the validation data then choose the best ones from them to use and predict for the data. In order to do this first we need to partion the data into training and validation datasets with 80% and 20% using train_test_split in the sklearn, dropping some unwanted columns and creating dummies for the categorical variables.



```
# partioning the data into training data and validation data
trainData,validData = train_test_split(loan_df_knn,test_size=0.2,random_state=1)
print(trainData.shape)
print(validData.shape)
```

```
(491, 13)
(123, 13)
```

1. **KNN Classification:**

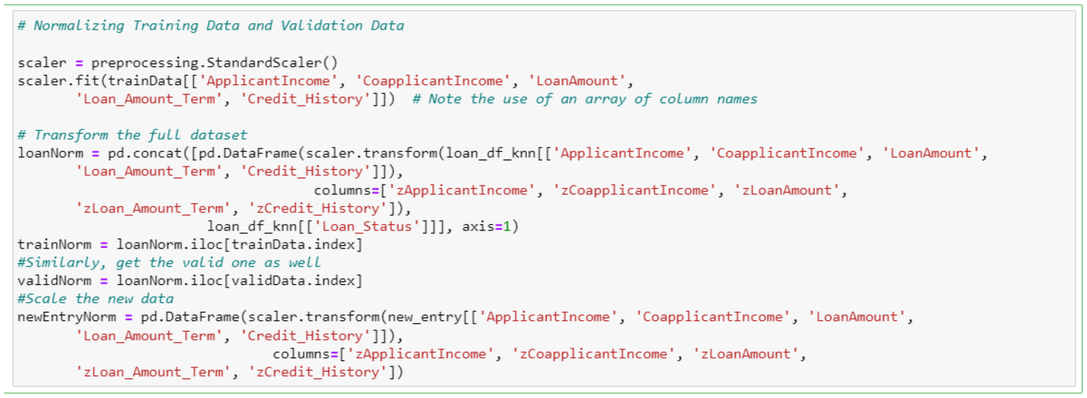
We look at a few records of the data and check whether they are on the same level or not. We will be checking is normalizing required or not. However, in this case we need to normalize the data.

Next step, we look at the predictors which are numerical, and the predictors used here are,

Predictors = 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',

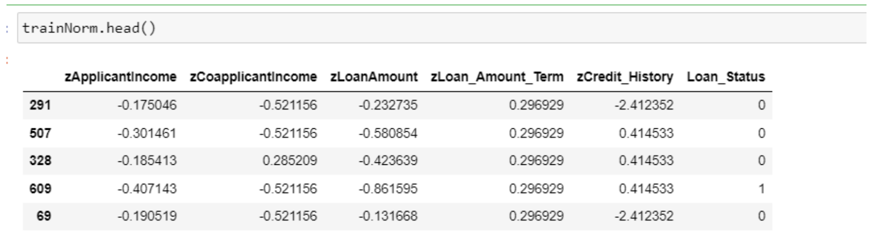
'Loan_Amount_Term', 'Credit_History', only the credit_history is the categorical variable remaining are numerical predictors.

Secondly, we normalize the data to the z – scores.



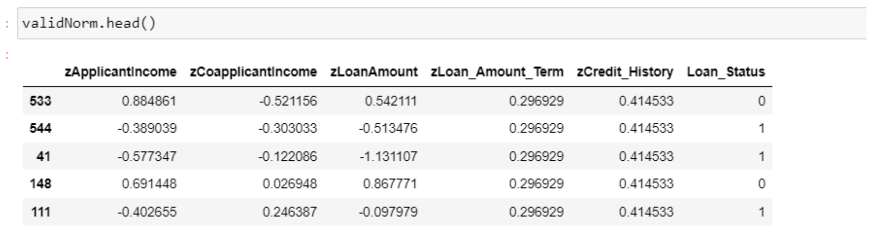
```
# Normalizing Training Data and Validation Data

scaler = preprocessing.StandardScaler()
scaler.fit(trainData[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History']])  # Note the use of an array of column names

# Transform the full dataset
loanNorm = pd.concat([pd.DataFrame(scaler.transform(loan_df_knn[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History']]),
                                    columns=['zApplicantIncome', 'zCoapplicantIncome', 'zLoanAmount',
       'zLoan_Amount_Term', 'zCredit_History']),
                        loan_df_knn[['Loan_Status']]], axis=1)
trainNorm = loanNorm.iloc[trainData.index]
#Similarly, get the valid one as well
validNorm = loanNorm.iloc[validData.index]
#Scale the new data
newEntryNorm = pd.DataFrame(scaler.transform(new_entry[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History']]),
                               columns=['zApplicantIncome', 'zCoapplicantIncome', 'zLoanAmount',
       'zLoan_Amount_Term', 'zCredit_History'])
```

Output:



```
trainNorm.head()
```

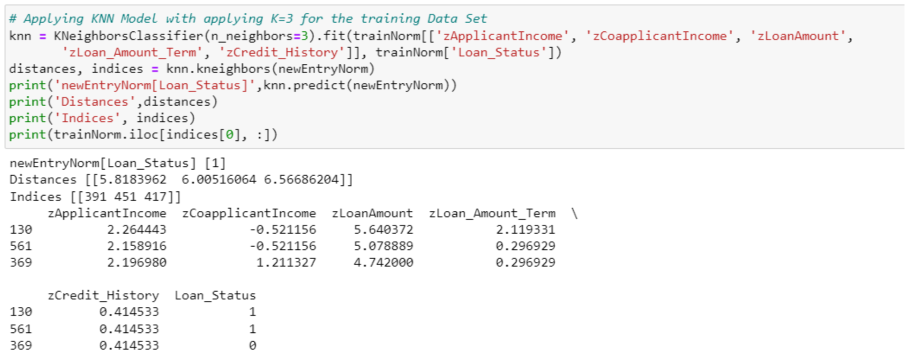
	zApplicantIncome	zCoapplicantIncome	zLoanAmount	zLoan_Amount_Term	zCredit_History	Loan_Status
291	-0.175046	-0.521156	-0.232735	0.296929	-2.412352	0
507	-0.301461	-0.521156	-0.580854	0.296929	0.414533	0
328	-0.185413	0.285209	-0.423639	0.296929	0.414533	0
609	-0.407143	-0.521156	-0.861595	0.296929	0.414533	1
69	-0.190519	-0.521156	-0.131668	0.296929	-2.412352	0



```
validNorm.head()
```

	zApplicantIncome	zCoapplicantIncome	zLoanAmount	zLoan_Amount_Term	zCredit_History	Loan_Status
533	0.884861	-0.521156	0.542111	0.296929	0.414533	0
544	-0.389039	-0.303033	-0.513476	0.296929	0.414533	1
41	-0.577347	-0.122086	-1.131107	0.296929	0.414533	1
148	0.691448	0.026948	0.867771	0.296929	0.414533	0
111	-0.402655	0.246387	-0.097979	0.296929	0.414533	1

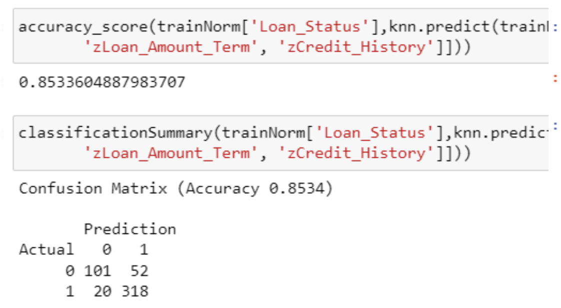
Thirdly, we take the normalized data frames and apply KNN classification. Initially we took K = 3 and found the accuracy values for both training and validation.



```
# Applying KNN Model with applying K=3 for the training Data Set
knn = KNeighborsClassifier(n_neighbors=3).fit(trainNorm[['zApplicantIncome', 'zCoapplicantIncome', 'zLoanAmount',
       'zLoan_Amount_Term', 'zCredit_History']], trainNorm['Loan_Status'])
distances, indices = knn.kneighbors(newEntryNorm)
print('newEntryNorm[Loan_Status]',knn.predict(newEntryNorm))
print('Distances',distances)
print('Indices', indices)
print(trainNorm.iloc[indices[0], :])
```

```
newEntryNorm[Loan_Status] [1]
Distances [[5.8183962  6.00516064 6.56686204]]
Indices [[391 451 417]]
     zApplicantIncome  zCoapplicantIncome  zLoanAmount  zLoan_Amount_Term  \
130          2.264443           -0.521156     5.640372           2.119331
561          2.158916           -0.521156     5.078889           0.296929
369          2.196980            1.211327     4.742000           0.296929

     zCredit_History  Loan_Status
130         0.414533            1
561         0.414533            1
369         0.414533            0
```

For training data:



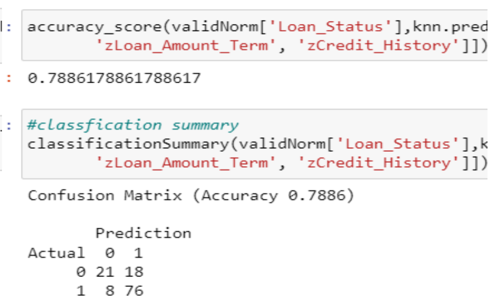
```
accuracy_score(trainNorm['Loan_Status'],knn.predict(trainI
         'zLoan_Amount_Term', 'zCredit_History']]))
```

```
0.8533604887983707
```

```
classificationSummary(trainNorm['Loan_Status'],knn.predic
         'zLoan_Amount_Term', 'zCredit_History']]))
```

```
Confusion Matrix (Accuracy 0.8534)

         Prediction
Actual    0   1
     0 101  52
     1  20 318
```

For Validation data:



```
accuracy_score(validNorm['Loan_Status'],knn.pred
         'zLoan_Amount_Term', 'zCredit_History']])
```

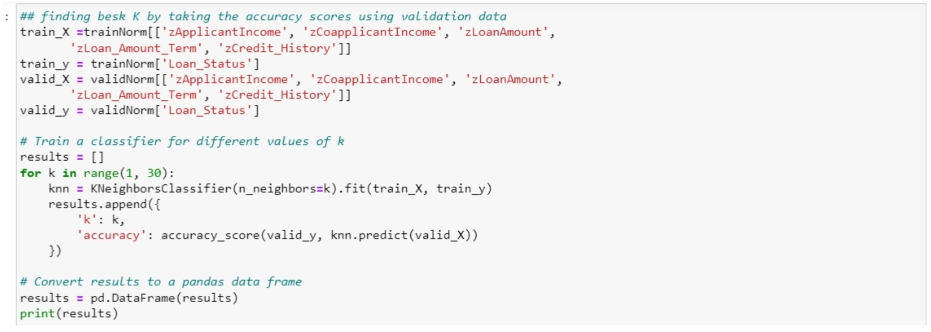
```
0.7886178861788617
```

```
#classfication summary
classificationSummary(validNorm['Loan_Status'],k
         'zLoan_Amount_Term', 'zCredit_History']])
```

```
Confusion Matrix (Accuracy 0.7886)

         Prediction
Actual  0  1
     0 21 18
     1  8 76
```

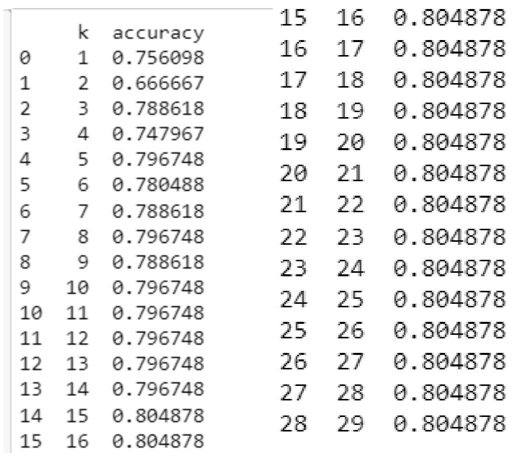
We can have a look over the accuracies of both training data and validation data for K = 3 which are, 0.85 for training and 0.78 which is for validation. We can, clearly look that K = 3 is not the near neighbors as it is overfitting the data as accuracy on validation is more than accuracy on training.

Now we need to search for an optimal K value.



```
## finding besk K by taking the accuracy scores using validation data
train_X =trainNorm[['zApplicantIncome', 'zCoapplicantIncome', 'zLoanAmount',
       'zLoan_Amount_Term', 'zCredit_History']]
train_y = trainNorm['Loan_Status']
valid_X = validNorm[['zApplicantIncome', 'zCoapplicantIncome', 'zLoanAmount',
       'zLoan_Amount_Term', 'zCredit_History']]
valid_y = validNorm['Loan_Status']

# Train a classifier for different values of k
results = []
for k in range(1, 30):
    knn = KNeighborsClassifier(n_neighbors=k).fit(train_X, train_y)
    results.append({
        'k': k,
        'accuracy': accuracy_score(valid_y, knn.predict(valid_X))
    })

# Convert results to a pandas data frame
results = pd.DataFrame(results)
print(results)
```

Hence, We are iterating through different K – values and checking the accuracy on each K value here we got the good accuracy from K =15 as shown in the results below.

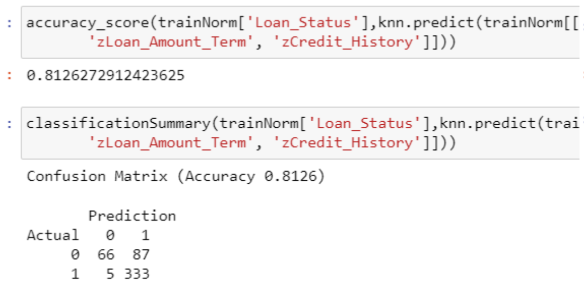


```
     k  accuracy
0    1  0.756098
1    2  0.666667
2    3  0.788618
3    4  0.747967
4    5  0.796748
5    6  0.780488
6    7  0.788618
7    8  0.796748
8    9  0.788618
9   10  0.796748
10  11  0.796748
11  12  0.796748
12  13  0.796748
13  14  0.796748
14  15  0.804878
15  16  0.804878
```

```
15  16  0.804878
16  17  0.804878
17  18  0.804878
18  19  0.804878
19  20  0.804878
20  21  0.804878
21  22  0.804878
22  23  0.804878
23  24  0.804878
24  25  0.804878
25  26  0.804878
26  27  0.804878
27  28  0.804878
28  29  0.804878
```

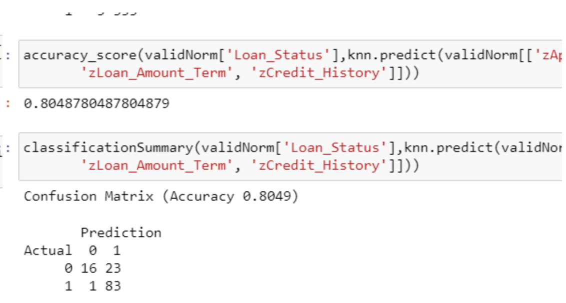
Hence, we can consider the K – value as 15 and take the accuracy values on training and validation.

Accuracy on training :



```
accuracy_score(trainNorm['Loan_Status'],knn.predict(trainNorm[[
        'zLoan_Amount_Term', 'zCredit_History']]))

0.8126272912423625

classificationSummary(trainNorm['Loan_Status'],knn.predict(trai
        'zLoan_Amount_Term', 'zCredit_History']]))

Confusion Matrix (Accuracy 0.8126)

       Prediction
Actual   0   1
     0  66  87
     1   5 333
```

Accuracy on validation:

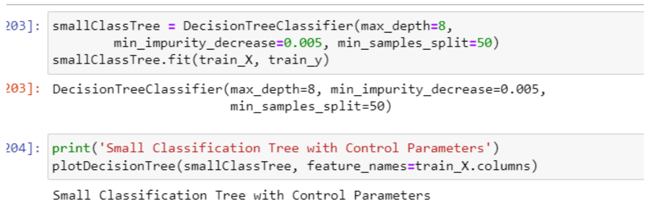


```
accuracy_score(validNorm['Loan_Status'],knn.predict(validNorm[['zAp
        'zLoan_Amount_Term', 'zCredit_History']]))

0.8048780487804879

classificationSummary(validNorm['Loan_Status'],knn.predict(validNor
        'zLoan_Amount_Term', 'zCredit_History']]))

Confusion Matrix (Accuracy 0.8049)

       Prediction
Actual  0  1
     0 16 23
     1  1 83
```

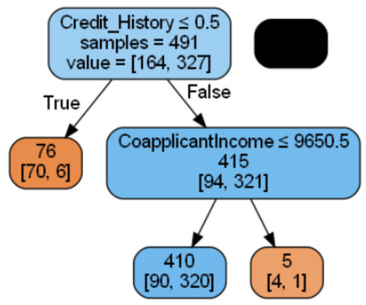
To Conclude, the KNN classification the accuracy for, k =15 good which is around 0.8049 which we can consider and use it for the data let us check on other models as well.

2. Decision Tree Classification:

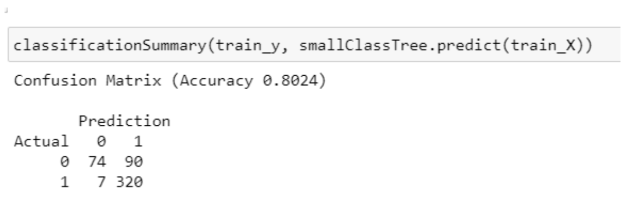
The same data that we have partitioned for the KNN classification is used here as well with 80% training data and 20% validation data.
Firstly, with controlled parameters we have taken a small tree as shown below:



```
203]: smallClassTree = DecisionTreeClassifier(max_depth=8,
              min_impurity_decrease=0.005, min_samples_split=50)
      smallClassTree.fit(train_X, train_y)

203]: DecisionTreeClassifier(max_depth=8, min_impurity_decrease=0.005,
                             min_samples_split=50)

204]: print('Small Classification Tree with Control Parameters')
      plotDecisionTree(smallClassTree, feature_names=train_X.columns)

      Small Classification Tree with Control Parameters

204]:
```

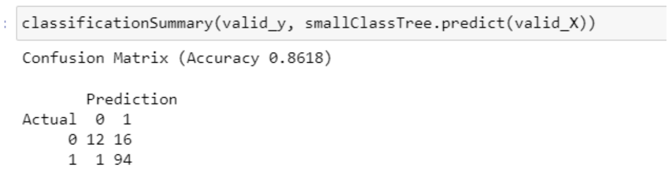


One of the rules that we can derive from this tree is, “**if Credit_History < 0.5 then loan gets rejected.”**
The classification Summary for this tree for training data is:



```
classificationSummary(train_y, smallClassTree.predict(train_X))

Confusion Matrix (Accuracy 0.8024)

       Prediction
Actual   0   1
     0  74  90
     1   7 320
```

The classification Summary for this tree for validation data:



```
classificationSummary(valid_y, smallClassTree.predict(valid_X))

Confusion Matrix (Accuracy 0.8618)

       Prediction
Actual  0  1
     0 12 16
     1  1 94
```

Hence, accuracy on validation is high for this tree with 0.8618 so, we can use this classification for classifying better loan status as output. It is also good to predict the output for this dataset.

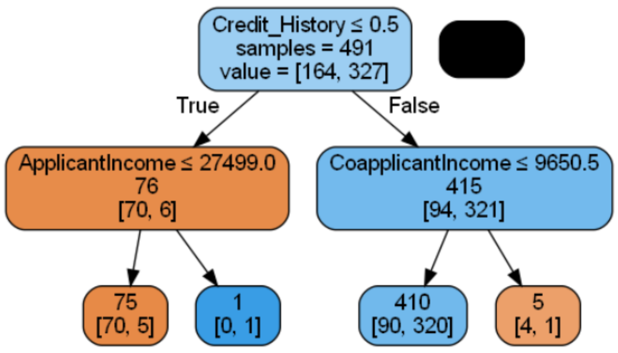
We are also checking on the decision tree that is obtained through grid search:

```
# Start with an initial guess for parameters
param_grid = {
    'max_depth': [10, 20, 30, 40],
    'min_samples_split': [20, 40, 60, 80, 100],
    'min_impurity_decrease': [0, 0.0005, 0.001, 0.005, 0.01],
}
gridSearch = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5, n_jobs=-1) #n_jobs=-1 means
        #that the availalbe computer memory (CPU) will be used to make calculations faster.
gridSearch.fit(train_X, train_y)
print('Initial score: ', gridSearch.best_score_)
print('Initial parameters: ', gridSearch.best_params_)

# Adapt grid based on result from initial grid search
param_grid = {
    'max_depth': list(range(2, 16)),
    'min_samples_split': list(range(10, 22)),
    'min_impurity_decrease': [0.0009, 0.001, 0.0011],
}
gridSearch = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5, n_jobs=-1)
gridSearch.fit(train_X, train_y)
print('Improved score: ', gridSearch.best_score_)
print('Improved parameters: ', gridSearch.best_params_)

bestClassTree = gridSearch.best_estimator_
```

```
Initial score:  0.7942692228406514
Initial parameters:  {'max_depth': 10, 'min_impurity_decrease': 0.005, 'min_samples_split': 20}
Improved score:  0.7922490208204495
Improved parameters:  {'max_depth': 2, 'min_impurity_decrease': 0.0009, 'min_samples_split': 10}
```

For the above tree the tree and classification summary are:

```
plotDecisionTree(bestClassTree, feature_names=train_X.columns)
```



[redacted]

For this tree a rule that can be derived is **“If credit_history is less than 0.5 and Applicant Income is less than 27499 dollars then the loan gets rejected.”**

Classification Summary (training):

```
classificationSummary(train_y, bestClassTre
classificationSummary(valid_y, bestClassTre
```

```
Confusion Matrix (Accuracy 0.8045)

       Prediction
Actual   0   1
     0  74  90
     1   6 321
```

Classification Summary (validation):

```
Confusion Matrix (Accuracy 0.8537)

       Prediction
Actual  0  1
     0 11 17
     1  1 94
```

Hence, here the accuracy on validation data is more than accuracy on training data we can also this tree as it is one of the best trees.

3. **<u>Random Forest Classification:</u>**

After two methods we have applied, next we are applying random forest classifier for the same data that is used for the decision tree classification.

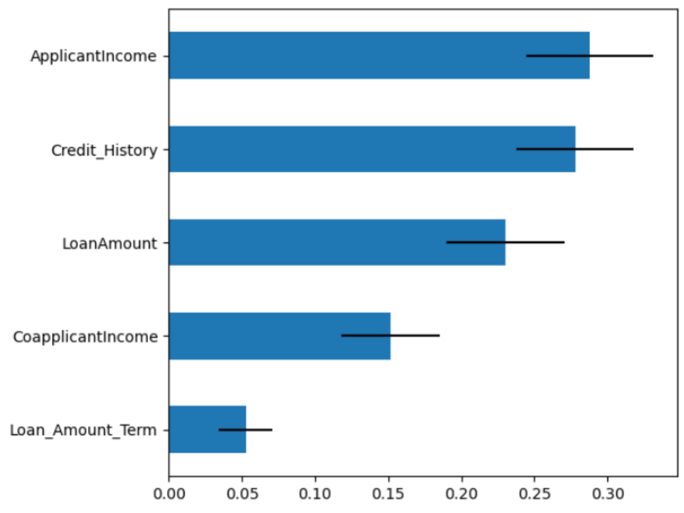
Dividing the data into training partition and validation partition with 80% and 20% respectively.

We are applying the random classifier that is present in the python sklearn packages.

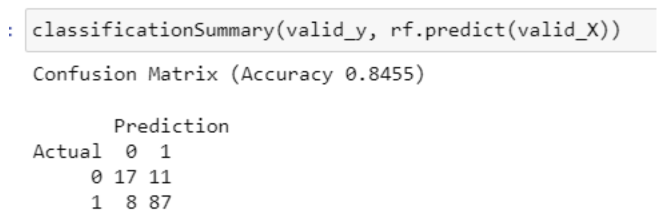
```
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)

df = pd.DataFrame({'feature': train_X.columns, 'importance': importances, 'std': std})
df = df.sort_values('importance')
print(df)

ax = df.plot(kind='barh', xerr='std', x='feature', legend=False)
ax.set_ylabel('')

plt.tight_layout()
plt.show()
```

```
             feature  importance       std
3    Loan_Amount_Term    0.052561  0.018286
1   CoapplicantIncome    0.151512  0.033683
2          LoanAmount    0.230008  0.040419
4      Credit_History    0.277926  0.040219
0     ApplicantIncome    0.287993  0.043176
```

The most importance is given to the Applicant Income and then comes the Credit_History and the least importance is given to the loan_amount_term before this co – applicant was given low importance.



Accuracy of the random forest classifier is :



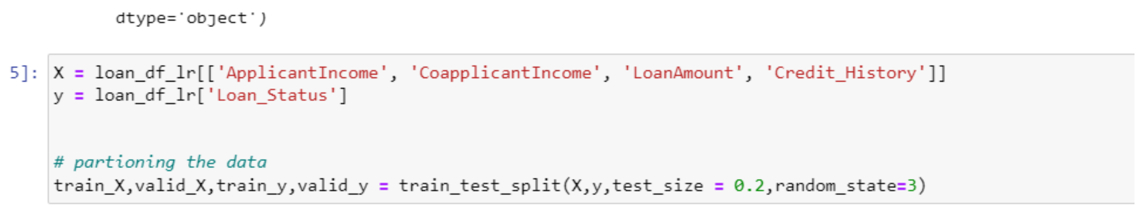
```
: classificationSummary(valid_y, rf.predict(valid_X))

  Confusion Matrix (Accuracy 0.8455)

         Prediction
  Actual  0  1
       0 17 11
       1  8 87
```

The accuracy on validation is 0.8455 hence, this is better than KNN classification but is not as good as the decision tree classification.

4. Logistic Regression:

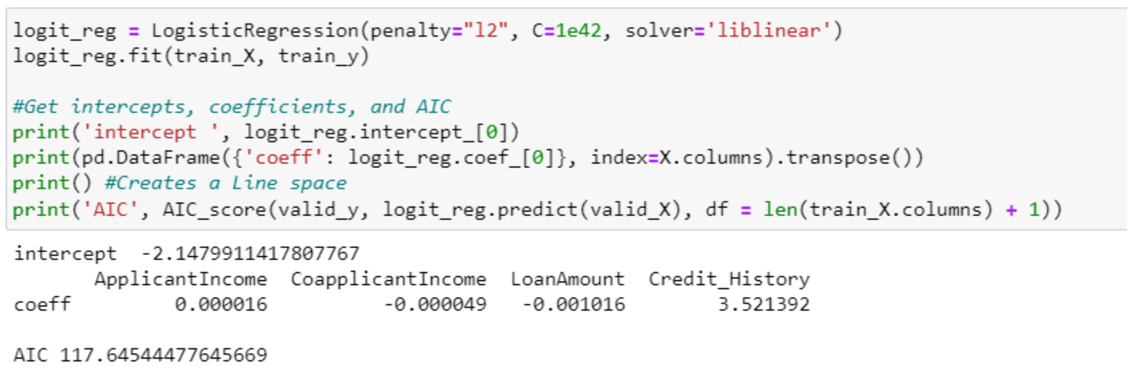
Lastly, we want to execute the logistic regression to choose best among all the methods that we have previously executed and check on their accuracies on validation data. This method is with the logit response function and is available directly in the model_Selection library in python.

1. We use the same data as the KNN, Decision Tree was using them.
2. Partitioning the data into training and validation data as 80% and 20% respectively.



```
        dtype='object')

5]: X = loan_df_lr[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Credit_History']]
    y = loan_df_lr['Loan_Status']


    # partioning the data
    train_X,valid_X,train_y,valid_y = train_test_split(X,y,test_size = 0.2,random_state=3)
```

3. Applying the logistic regression using the package

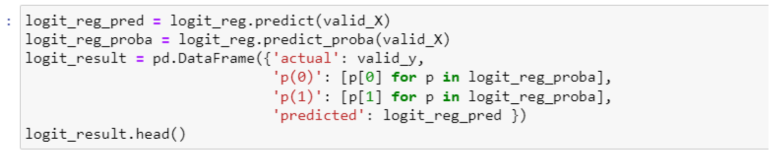


```
logit_reg = LogisticRegression(penalty="l2", C=1e42, solver='liblinear')
logit_reg.fit(train_X, train_y)

#Get intercepts, coefficients, and AIC
print('intercept ', logit_reg.intercept_[0])
print(pd.DataFrame({'coeff': logit_reg.coef_[0]}, index=X.columns).transpose())
print() #Creates a Line space
print('AIC', AIC_score(valid_y, logit_reg.predict(valid_X), df = len(train_X.columns) + 1))
```

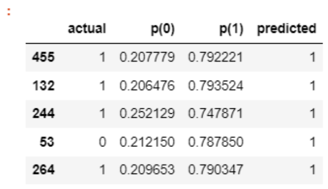
```
intercept  -2.1479911417807767
       ApplicantIncome  CoapplicantIncome  LoanAmount  Credit_History
coeff         0.000016          -0.000049   -0.001016        3.521392

AIC 117.64544477645669
```

The coefficients in the logit model are having two negative ones and two positive ones, Applicant Income and credit history are having the positive one where increase by one unit would increase odds of loan approval. If there is an unit decrease in Co - applicant Income and Loan Amount then the odds might decrease for an loan approval.

4. Just looking at the results of the logistic regression that are:

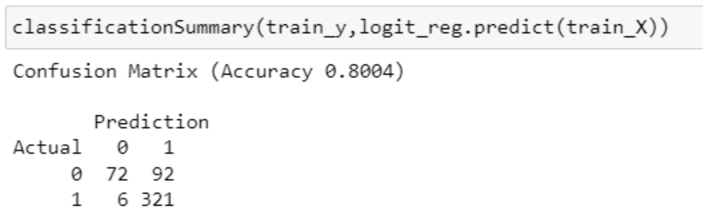


```
logit_reg_pred = logit_reg.predict(valid_X)
logit_reg_proba = logit_reg.predict_proba(valid_X)
logit_result = pd.DataFrame({'actual': valid_y,
                             'p(0)': [p[0] for p in logit_reg_proba],
                             'p(1)': [p[1] for p in logit_reg_proba],
                             'predicted': logit_reg_pred })
logit_result.head()
```



	actual	p(0)	p(1)	predicted
455	1	0.207779	0.792221	1
132	1	0.206476	0.793524	1
244	1	0.252129	0.747871	1
53	0	0.212150	0.787850	1
264	1	0.209653	0.790347	1

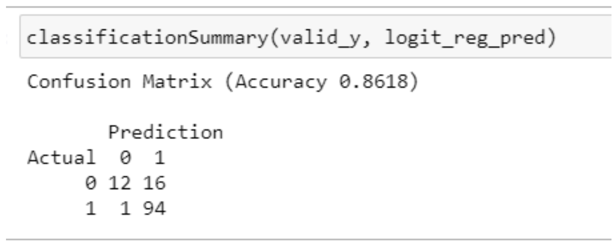
5. Accuracy of training:



```
classificationSummary(train_y,logit_reg.predict(train_X))
```

```
Confusion Matrix (Accuracy 0.8004)

       Prediction
Actual   0   1
     0  72  92
     1   6 321
```

Accuracy of validation:



```
classificationSummary(valid_y, logit_reg_pred)
```

```
Confusion Matrix (Accuracy 0.8618)

       Prediction
Actual  0  1
     0 12 16
     1  1 94
```

Hence, for this model the accuracy on validation data is 0.8618 which represents the model is a good fit hence which is also best as compared to other methods.

<u>Comparison of accuracies:</u>

Model Name	**Accuracy**
KNN classification	**0.8048**
Decision Tree Classification	**0.8618**
Random Forest Classification	**0.8455**
Logistic Regression	**0.8618**

Among all the methods from the comparison table we have Decision Tree and Logistic Regression as the best accuracy value of **0.8618**. from both the models We have selected **Logistic Regression** as the best model to predict the output for the loan approval.

Conclusion:

- In this project, we have taken Loan approval prediction as our problem statement. Writing down some research questions, to understand what predictors to take.
- Secondly, we pre – processed the data to remove all the null values by replacing respective values.
- Thirdly, we visualized some of the data variables to pick some of them as the predictors, we got Loan amount, applicant Income, co – applicant Income, Credit History, Loan amount term as predictors.
- We have used four methods KNN classification, Decision Tree Classification, Random Forest Classifier and Logistic Regression.
- Among all the methods we got Logistic Regression as the best model to predict for the loan approval status.

Recommendations:

- According to the visualizations that we have done Banker, or the loan lenders should educate the customers about the credit history and maintain them correctly to get loans in the future.
- As the applicant income and loan amount are correlated it is important to make the customer understand to apply a loan for an optimal amount with respect to their annual income or monthly income so that the loan rejection probability might decrease.
- If there are more dependents the loan must be applied in accordance with the income, they have not more than the threshold.
- The bankers giving the lower interest rate might increase the number of customers arriving to them.

Limitations:

- There are many outliers in the numerical data with respect to the loan status, hence, this model might not be 100% effective.
- As the data is somewhat imbalanced, the sampling might increase some chances of increase in accuracy.