Loan Prediction

Machine Learning Project

Submitted by
Aswin Raghuprasad - AM.EN.U4CSE16115
Karthika Manoj - AM.EN.U4CSE6131
Priya T.V -AM.EN.U4CSE16150

Abstract

Informal Description

The primary goal of this project is to extract patterns from a common loan approved dataset, and then build a model based on these extracted patterns, in order to predict the likely loan defaulters by using machine learning algorithms. The historical data of the customers like their age, income, loan amount, gender etc. will be used in order to do the analysis.

Formal Description

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Here,

• **Task (T):** Predict if a customer is eligible for loan application. (A customer is found to be eligible for loan if he is not a potential defaulter).
• **Experience (E):** A set of data collected about previous customers and their different details such as Income,Gender,Age etc along with their loan status.
• **Performance (P):** Prediction accuracy, the number of eligible customers predicted correctly who are actually eligible for loan.

Introduction

Motivation

Importance of loans in our day-to-day life has increased to a great extent. People are becoming more and more dependent on acquiring loans, be it education loan, housing loan, car loan, business loans etc. from the financial institutions like banks and credit unions. However, it is no longer surprising to see that some people are not able to properly gauge the amount of loan that they can afford. In some cases, people undergo sudden financial crisis while some try to scam money out of the banks. The consequences of such scenarios are late payments or missing payments, defaulting or in the worst-case scenario not being able to pay back those bulk amount to the banks.

Assessing the risk, which is involved in a loan application, is one of the most important concerns of the banks for survival in the highly competitive market and for profitability. These banks receive number of loan applications from their customers and other people on a daily basis. Not everyone gets approved. Most of the banks use their own credit scoring and risk assessment techniques in order to analyze the loan application and to make decisions on credit approval. In spite of this, there are many cases happening every year, where people do not repay the loan amounts or they default, due to which these financial institutions suffer huge amount of losses.

Benefits

Due to high competition in the business field, customer relationship management has to be considered in the enterprise. Here massive volume of data is analysed and classified based on the customer behaviours and predicted. Customer relationship management is mainly used in banking areas. Data mining provides many technologies to analyze mass volume of data and detect hidden patterns to convert raw data into valuable information. It is a powerful new technology with great potential to help banks focus on most of the information in their data warehouse.

The key idea of data mining techniques is to classify the customer data according to the posterior probability. Here it is used to perform the classification and prediction of loan. With the continuous development and changes in the credit industry, credit products play an important role in the economy. Credit risk evaluation decisions are crucial for financial institutions due to high risks associated with inappropriate credit decisions that may result in major losses. It is an even more important task today as financial institutions have been experiencing serious challenges and competition during the past decade. It concerns those lenders to limit potential default risks, screening the customer's financial history and financial background. Banks should control credit management thoroughly. Sanctioning of loan needs the use of huge data and substantial processing time. Before granting loans, banks have to take various precautions such as performance of the firm by analyzing last year's financial statements and history of the customer. The decisions of sanctioning loans may become wrong and resulted in credit defaults. An intelligent information system that is based on clustering algorithm will provide managers with added information, to reduce the uncertainty of the decision outcome to enhance banking service quality.

Solution Use

In today's world, taking loans from financial institutions has become a very common phenomenon. Everyday a large number of people make application for loans, for a variety of purposes. But all these applicants are not reliable and everyone cannot be approved. Every year, we read about a number of cases where people do not repay a bulk of the loan amount to the banks due to which they suffer huge losses. The risk associated with making a decision on loan approval is immense. So the idea of this project is to gather loan data from multiple data sources and use data mining algorithms on this data to extract important information and predict if a customer would be able to repay his loan or not or in other words predict if the customer would be a defaulter or not. This can be used by different banks to minimise their risks of lending a huge sum to their customers who might not be able to repay it back.

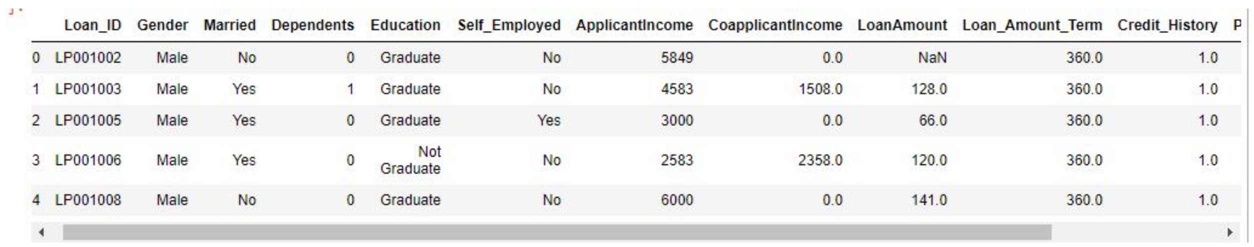
Datasets

- Loan Prediction Problem III Dataset from Analytics Vidhya
- Lending Club Loan Data -Issued Loans (https://www.kaggle.com/wendykan/lending-club-loan-data)
- All Lending Club Loan Data (https://www.kaggle.com/wordsforthewise/lending-club)

Before Preprocessing

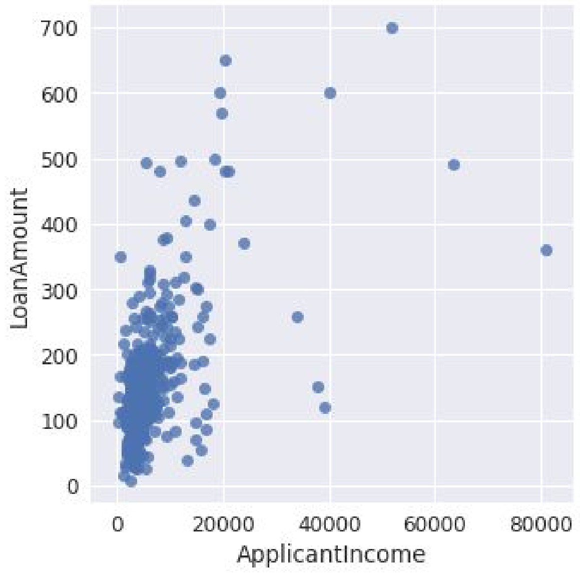
Data Visualisation

Head of dataset

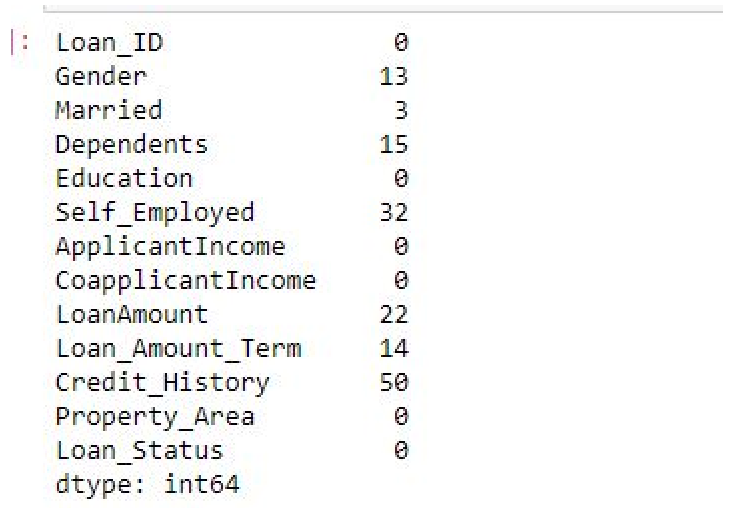


	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	P
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	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	

Scatter plot of Applicant Income vs Loan Amount

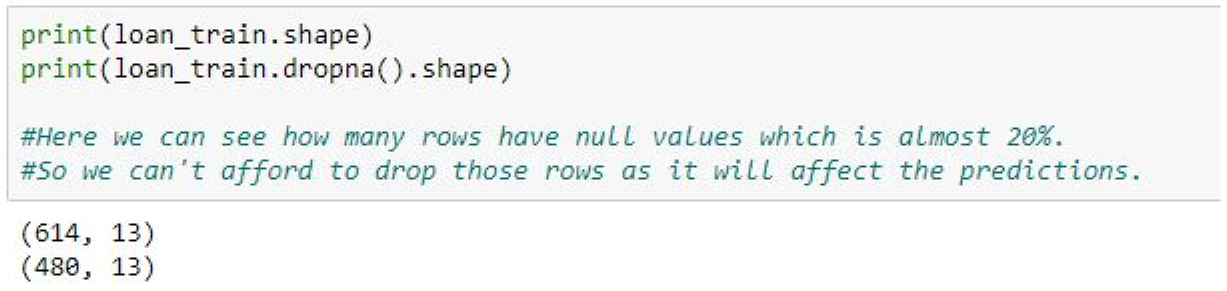


Missing values



```
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
```

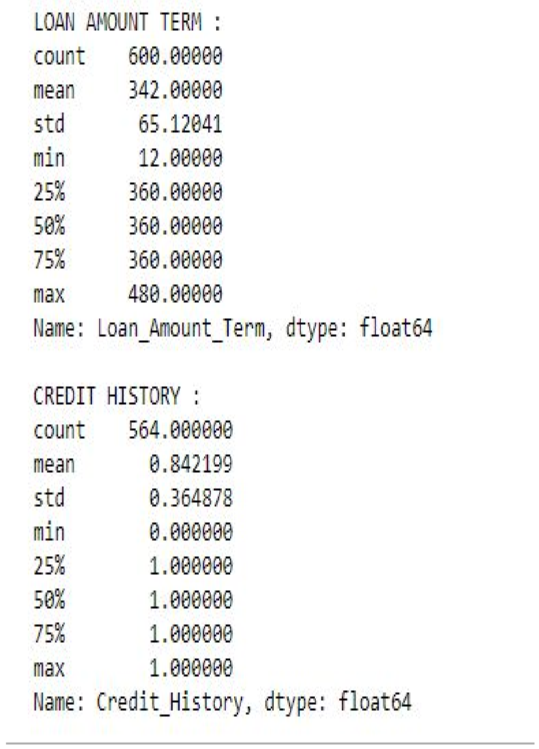
No. of original rows and columns and no of rows having null values



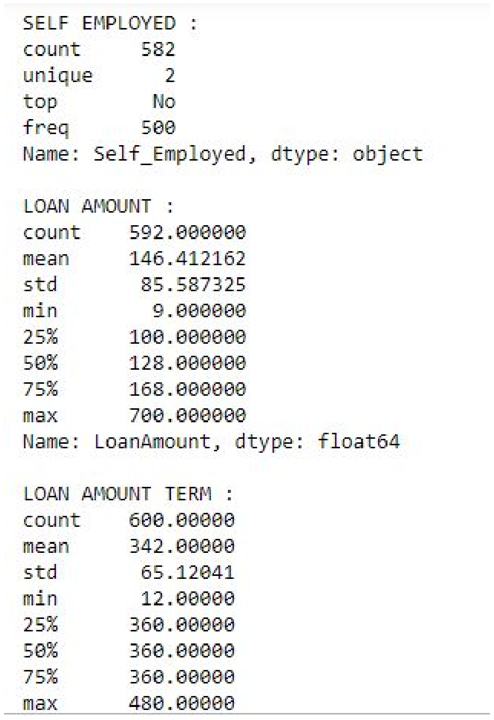
```
print(loan_train.shape)
print(loan_train.dropna().shape)

#Here we can see how many rows have null values which is almost 20%.
#So we can't afford to drop those rows as it will affect the predictions.
```

```
(614, 13)
(480, 13)
```

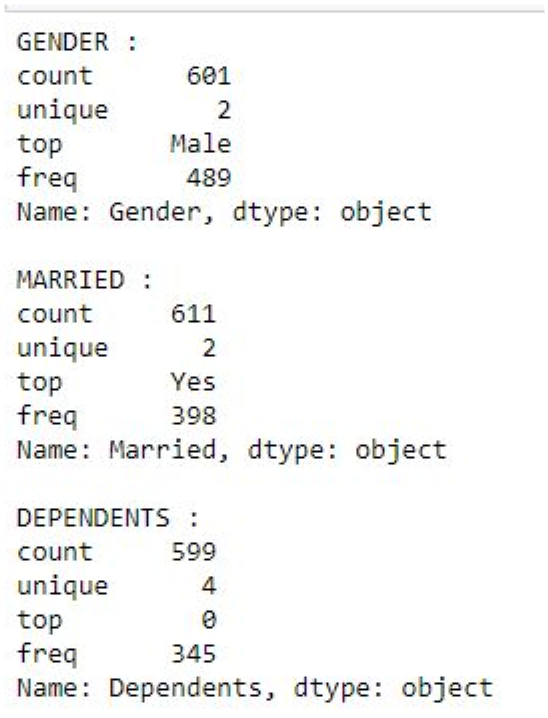
Describing features which have null value to understand each of them better.



```
LOAN AMOUNT TERM :
count    600.00000
mean     342.00000
std       65.12041
min       12.00000
25%      360.00000
50%      360.00000
75%      360.00000
max      480.00000
Name: Loan_Amount_Term, dtype: float64

CREDIT HISTORY :
count    564.000000
mean       0.842199
std        0.364878
min        0.000000
25%        1.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: Credit_History, dtype: float64
```



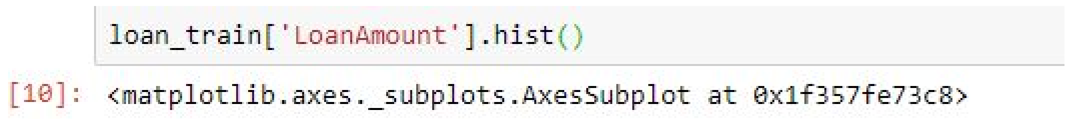
```
SELF EMPLOYED :
count     582
unique      2
top        No
freq      500
Name: Self_Employed, dtype: object

LOAN AMOUNT :
count    592.000000
mean     146.412162
std       85.587325
min        9.000000
25%      100.000000
50%      128.000000
75%      168.000000
max      700.000000
Name: LoanAmount, dtype: float64

LOAN AMOUNT TERM :
count    600.00000
mean     342.00000
std       65.12041
min       12.00000
25%      360.00000
50%      360.00000
75%      360.00000
max      480.00000
```



```
GENDER :
count      601
unique       2
top       Male
freq       489
Name: Gender, dtype: object

MARRIED :
count     611
unique      2
top       Yes
freq      398
Name: Married, dtype: object

DEPENDENTS :
count     599
unique      4
top         0
freq      345
Name: Dependents, dtype: object
```

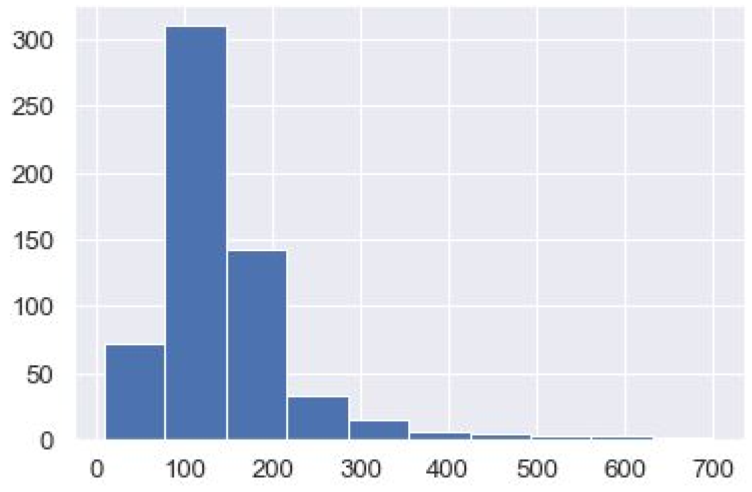
PREPROCESSING

Visualizing LoanAmount to understand it better on how to impute missing values
Histogram plot of loan amount



```
loan_train['LoanAmount'].hist()
```

```
[10]:  <matplotlib.axes._subplots.AxesSubplot at 0x1f357fe73c8>
```

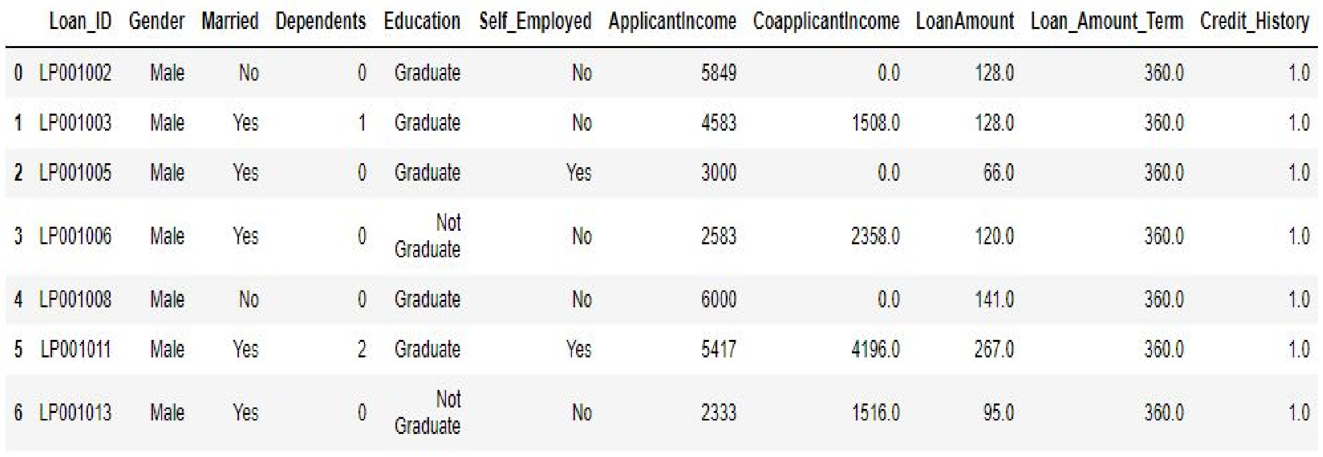


Text Data Preprocessing

All missing values were filled and '+' sign was removed from "Dependent" column.

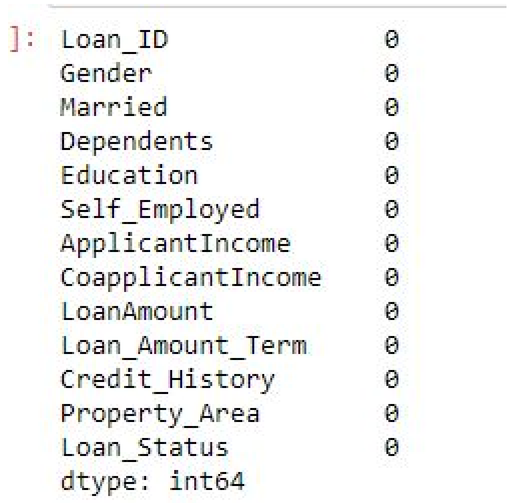
loan_train['Dependents'] = loan_train['Dependents'].str.replace("+","")

Head of data after preprocessing



	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	128.0	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
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0

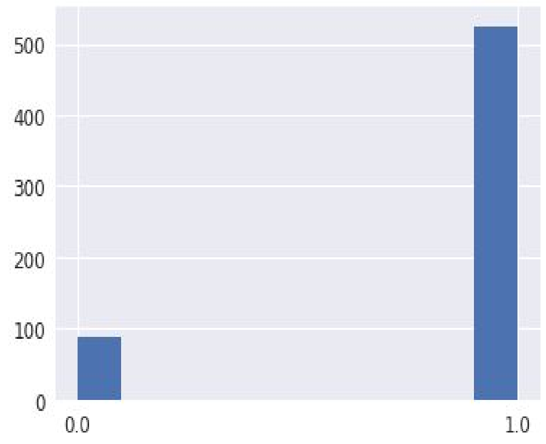
Null values removed



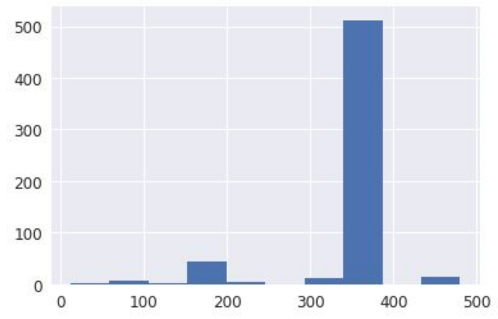
```
]:  Loan_ID              0
    Gender               0
    Married              0
    Dependents           0
    Education            0
    Self_Employed        0
    ApplicantIncome      0
    CoapplicantIncome    0
    LoanAmount           0
    Loan_Amount_Term     0
    Credit_History       0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

<u>Other Plots</u>

Credit history

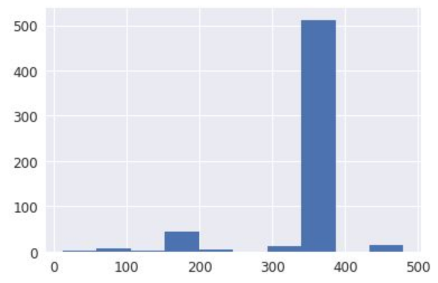


Loan Amount Term



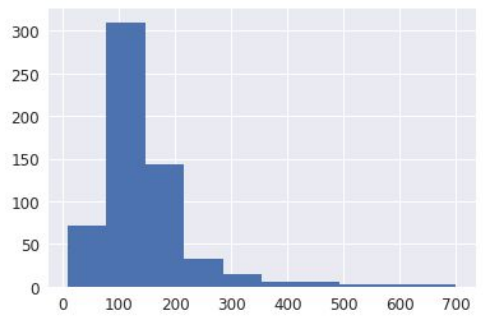
Here we can see that ‘Loan Amount Term’ is right skewed, So median can be used to fill in the missing values.

```
In [13]: loan_train['Loan_Amount_Term'].hist()

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0cca21d550>
```



```
In [14]: #Here we can see that Loan_Amount_Term is left skewed , So imputing with Median would be better.
         loan_train['Loan_Amount_Term'].fillna(loan_train.Loan_Amount_Term.median(),inplace=True)

In [15]: # Now since 'Credit_History' a categorical column hence imputing with mode value
         loan_train['Credit_History'].fillna(loan_train.Credit_History.median(),inplace=True)
```

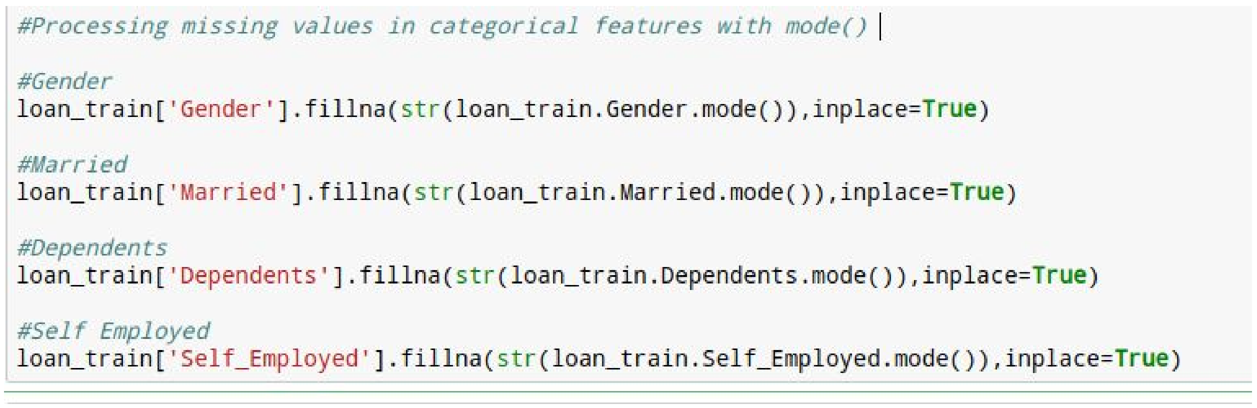
```
#Visualizing LoanAmount to understand it better on how to impute missing values
loan_train['LoanAmount'].hist()
<matplotlib.axes._subplots.AxesSubplot at 0x7f0cd210a048>
```



```
#Here we can see that loan amount is right skewed , So imputing with Median would be better.
loan_train['LoanAmount'].fillna(loan_train.LoanAmount.median(),inplace=True)
```

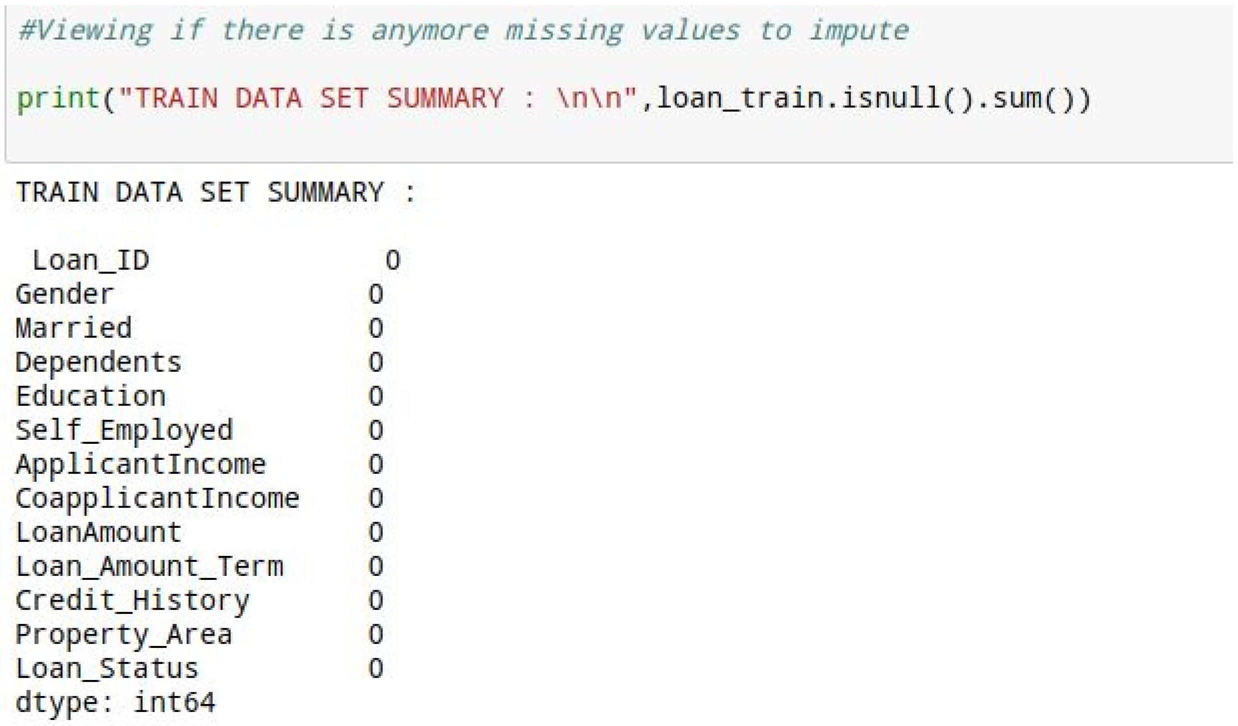
Loan Amount shows a skewed graph too, So we use median to fill in the NA values.

And for the categorical values , we use mode to fill in the NA values.



```
#Processing missing values in categorical features with mode()

#Gender
loan_train['Gender'].fillna(str(loan_train.Gender.mode()),inplace=True)

#Married
loan_train['Married'].fillna(str(loan_train.Married.mode()),inplace=True)

#Dependents
loan_train['Dependents'].fillna(str(loan_train.Dependents.mode()),inplace=True)

#Self Employed
loan_train['Self_Employed'].fillna(str(loan_train.Self_Employed.mode()),inplace=True)
```

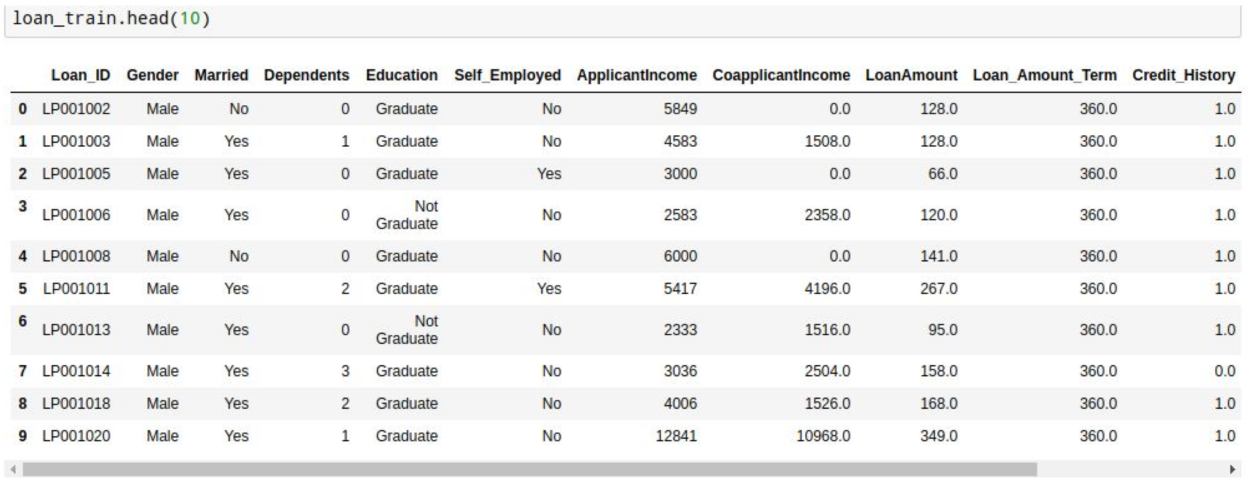
Now the data missing value summary :



```
#Viewing if there is anymore missing values to impute

print("TRAIN DATA SET SUMMARY : \n\n",loan_train.isnull().sum())
```

```
TRAIN DATA SET SUMMARY :

 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
```

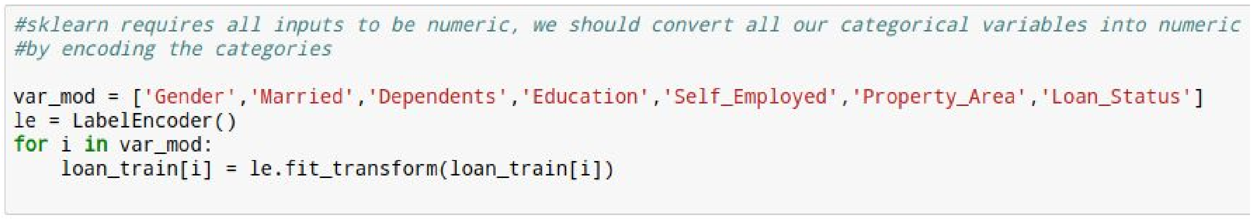
The dataset now is :



```
loan_train.head(10)
```

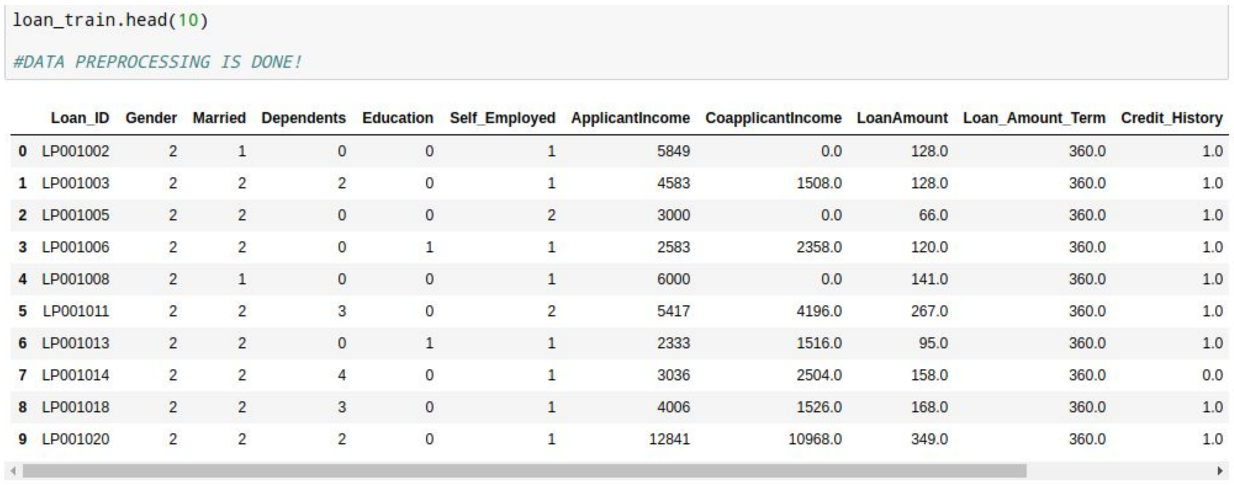
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	128.0	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	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
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0
7	LP001014	Male	Yes	3	Graduate	No	3036	2504.0	158.0	360.0	0.0
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0

Scikit-Learn requires all the data values to be numerical , So we use Label Encoder to convert all the categorical text values to numerical values



```
#sklearn requires all inputs to be numeric, we should convert all our categorical variables into numeric
#by encoding the categories

var_mod = ['Gender','Married','Dependents','Education','Self_Employed','Property_Area','Loan_Status']
le = LabelEncoder()
for i in var_mod:
    loan_train[i] = le.fit_transform(loan_train[i])
```

The data preprocessing is done and the data now looks like :



```
loan_train.head(10)

#DATA PREPROCESSING IS DONE!
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	2	1	0	0	1	5849	0.0	128.0	360.0	1.0
1	LP001003	2	2	2	0	1	4583	1508.0	128.0	360.0	1.0
2	LP001005	2	2	0	0	2	3000	0.0	66.0	360.0	1.0
3	LP001006	2	2	0	1	1	2583	2358.0	120.0	360.0	1.0
4	LP001008	2	1	0	0	1	6000	0.0	141.0	360.0	1.0
5	LP001011	2	2	3	0	2	5417	4196.0	267.0	360.0	1.0
6	LP001013	2	2	0	1	1	2333	1516.0	95.0	360.0	1.0
7	LP001014	2	2	4	0	1	3036	2504.0	158.0	360.0	0.0
8	LP001018	2	2	3	0	1	4006	1526.0	168.0	360.0	1.0
9	LP001020	2	2	2	0	1	12841	10968.0	349.0	360.0	1.0

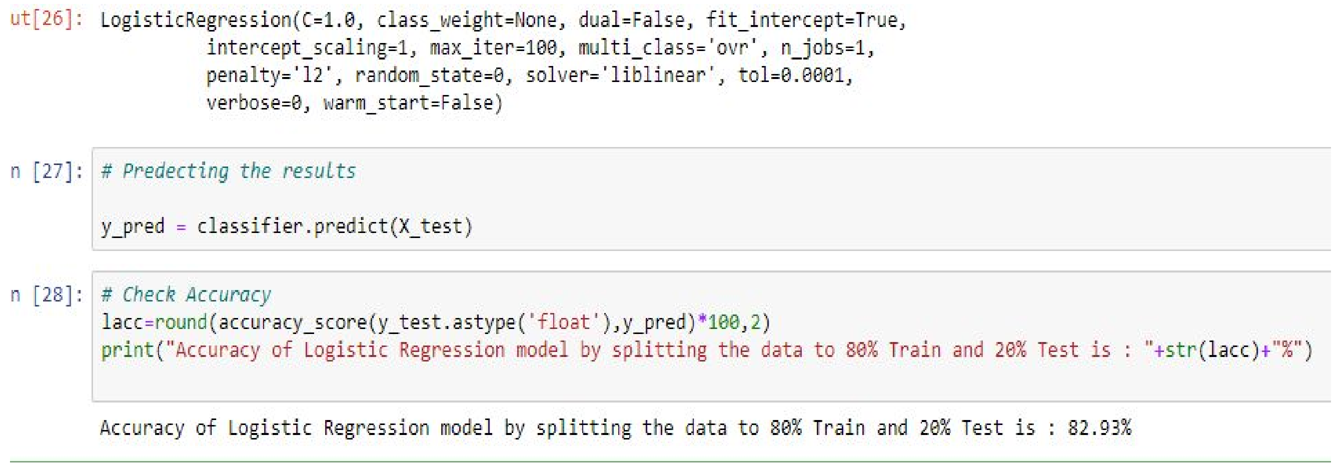
Supervised Learning

We implemented three supervised learning algorithms on our dataset .

- Logistic regression
- Random Forest
- K-nearest neighbours

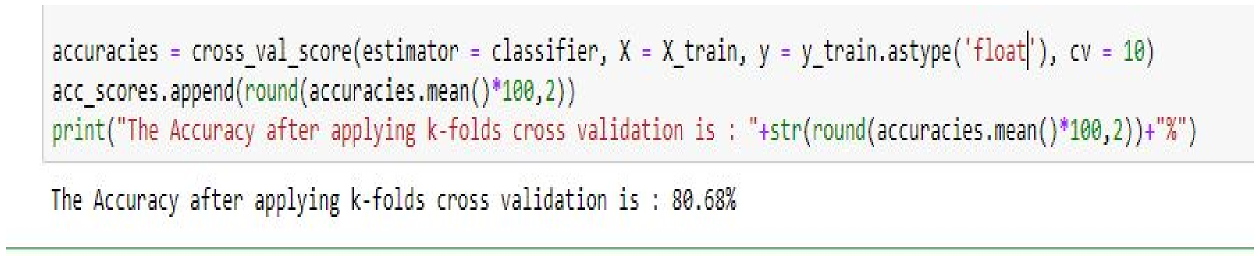
Logistic Regression

An accuracy of 82.93% was achieved using logistic regression on the dataset.



```
ut[26]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                  penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
                  verbose=0, warm_start=False)

n [27]: # Predecting the results

        y_pred = classifier.predict(X_test)

n [28]: # Check Accuracy
        lacc=round(accuracy_score(y_test.astype('float'),y_pred)*100,2)
        print("Accuracy of Logistic Regression model by splitting the data to 80% Train and 20% Test is : "+str(lacc)+"%")

        Accuracy of Logistic Regression model by splitting the data to 80% Train and 20% Test is : 82.93%
```

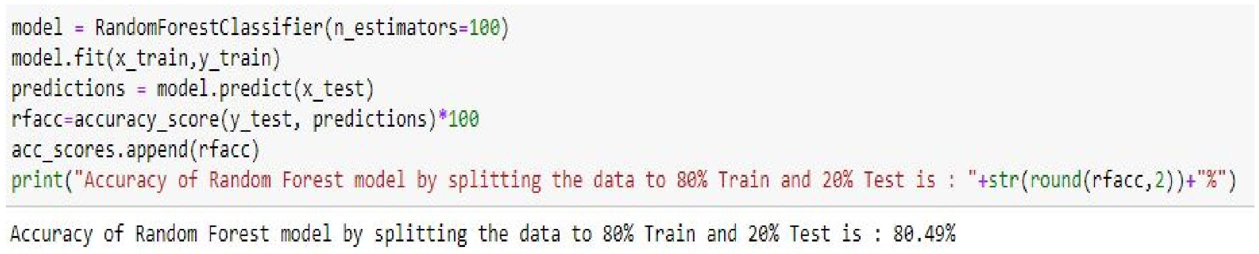
We also did k-fold cross validation (with k=10) to get an accuracy of 80.68%.
Loo



```
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train.astype('float'), cv = 10)
acc_scores.append(round(accuracies.mean()*100,2))
print("The Accuracy after applying k-folds cross validation is : "+str(round(accuracies.mean()*100,2))+"%")

The Accuracy after applying k-folds cross validation is : 80.68%
```

Random Forest

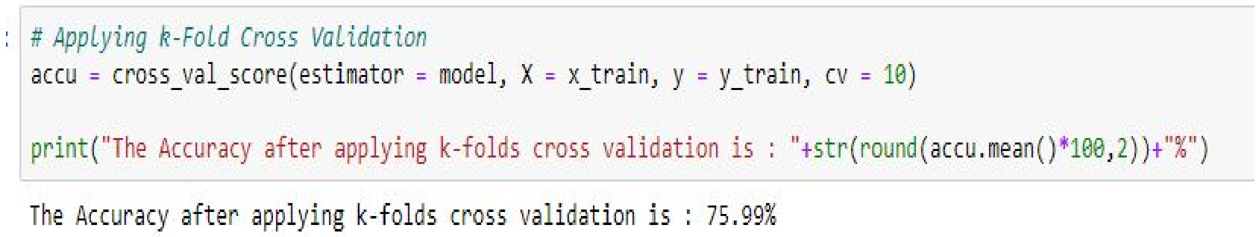
80.49% was the accuracy when Random Forest was implemented.



```
model = RandomForestClassifier(n_estimators=100)
model.fit(x_train,y_train)
predictions = model.predict(x_test)
rfacc=accuracy_score(y_test, predictions)*100
acc_scores.append(rfacc)
print("Accuracy of Random Forest model by splitting the data to 80% Train and 20% Test is : "+str(round(rfacc,2))+"%")
```

```
Accuracy of Random Forest model by splitting the data to 80% Train and 20% Test is : 80.49%
```

After applying k-fold cross validation(k=10) the accuracy became 75.99%

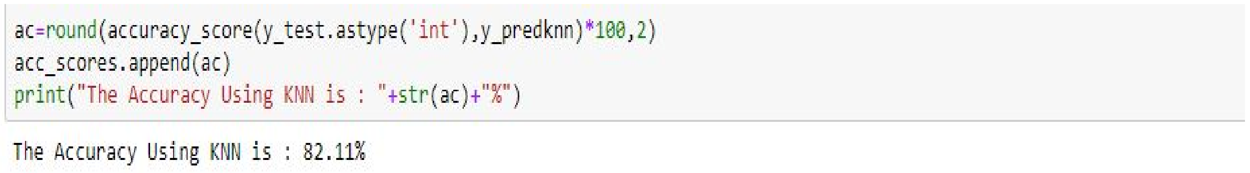


```
# Applying k-Fold Cross Validation
accu = cross_val_score(estimator = model, X = x_train, y = y_train, cv = 10)

print("The Accuracy after applying k-folds cross validation is : "+str(round(accu.mean()*100,2))+"%")
```

```
The Accuracy after applying k-folds cross validation is : 75.99%
```

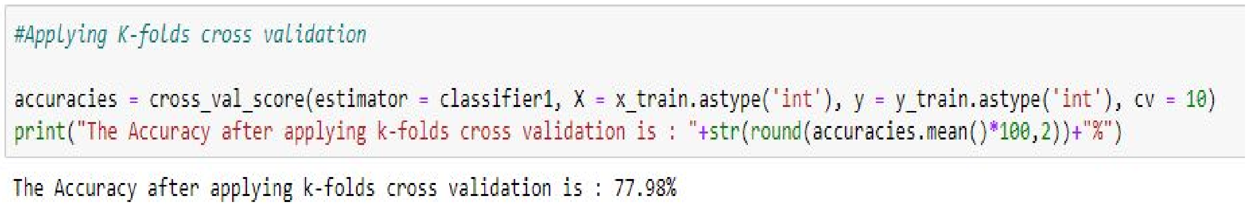
k-Nearest Neighbours

Using k-NN (3 nearest neighbours) accuracy of 82.93% was achieved and 79.62% on k-fold cross validation (with k=10).



```
ac=round(accuracy_score(y_test.astype('int'),y_predknn)*100,2)
acc_scores.append(ac)
print("The Accuracy Using KNN is : "+str(ac)+"%")
```

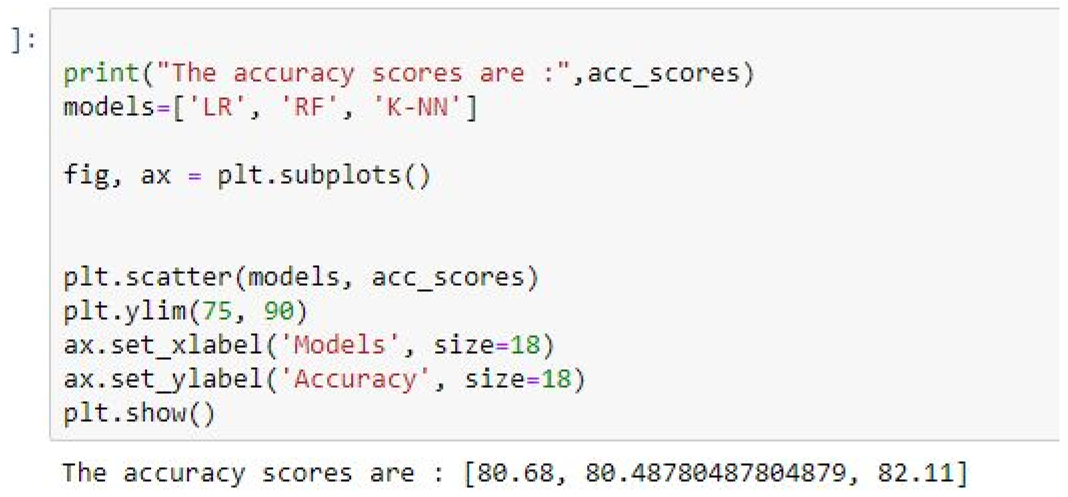
```
The Accuracy Using KNN is : 82.11%
```



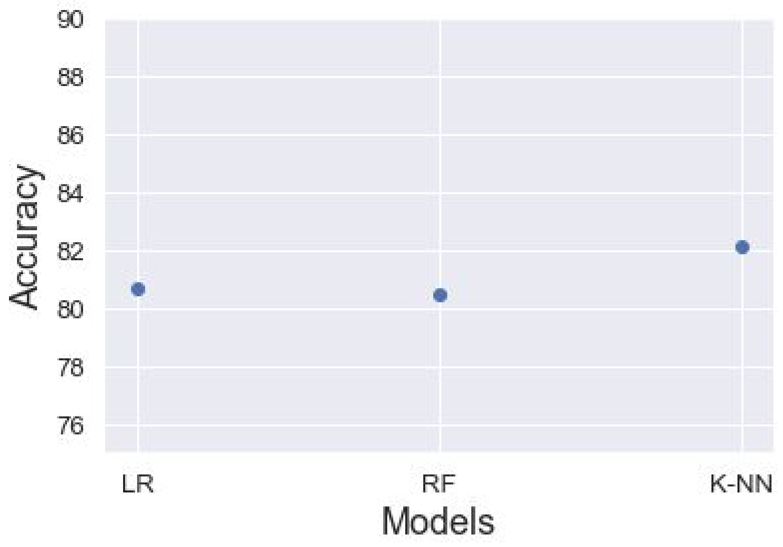
```
#Applying K-folds cross validation

accuracies = cross_val_score(estimator = classifier1, X = x_train.astype('int'), y = y_train.astype('int'), cv = 10)
print("The Accuracy after applying k-folds cross validation is : "+str(round(accuracies.mean()*100,2))+"%")
```

```
The Accuracy after applying k-folds cross validation is : 77.98%
```

Performance Comparison of Supervised Learning Models



```
]:
    print("The accuracy scores are :",acc_scores)
    models=['LR', 'RF', 'K-NN']

    fig, ax = plt.subplots()


    plt.scatter(models, acc_scores)
    plt.ylim(75, 90)
    ax.set_xlabel('Models', size=18)
    ax.set_ylabel('Accuracy', size=18)
    plt.show()

    The accuracy scores are : [80.68, 80.48780487804879, 82.11]
```



K-NN performed better than Random Forest Model and Logistic Regression.

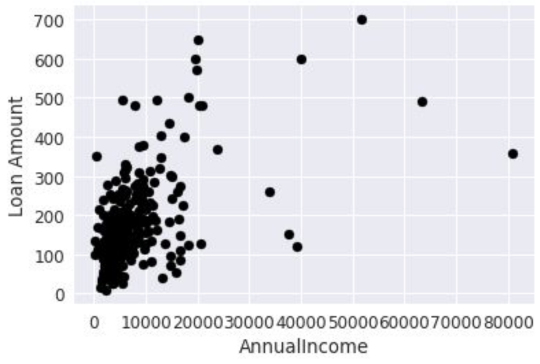
Unsupervised Learning

We implemented the following Unsupervised algorithms on our dataset :

- K-Means Clustering

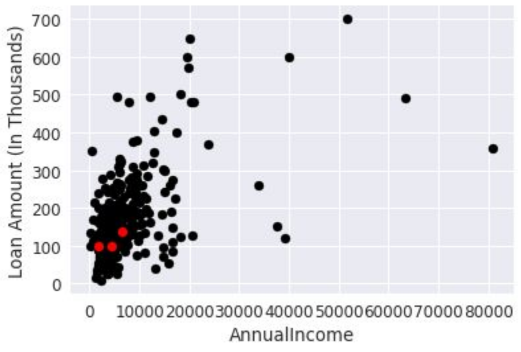
For K-Means algorithm, we took two features of the data set which we thought was the best for clustering : **Applicant Income** and **Loan Amount**

```
X = loan_train[["LoanAmount","ApplicantIncome"]]
#Visualise data points
plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')
plt.xlabel('AnnualIncome')
plt.ylabel('Loan Amount')
plt.show()
```

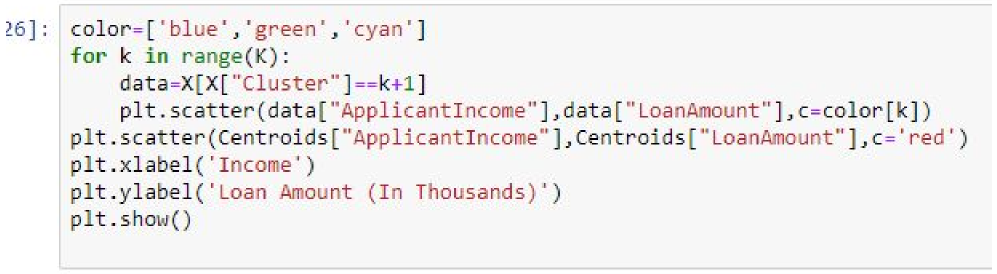


We take random centroids (red data points):

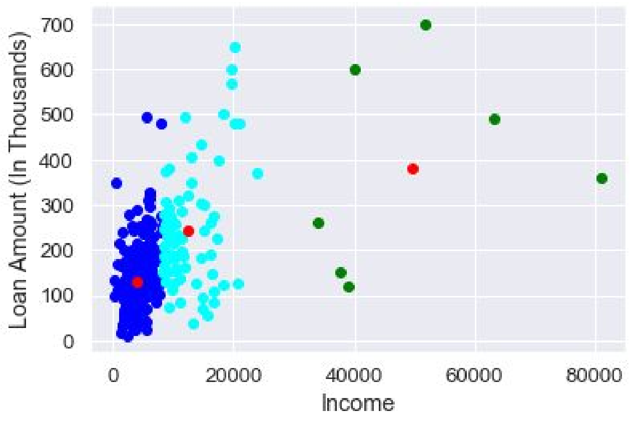
```
#number of clusters
K=3

# Select random observation as centroids
Centroids = (X.sample(n=K))
plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
plt.xlabel('AnnualIncome')
plt.ylabel('Loan Amount (In Thousands)')
plt.show()
```



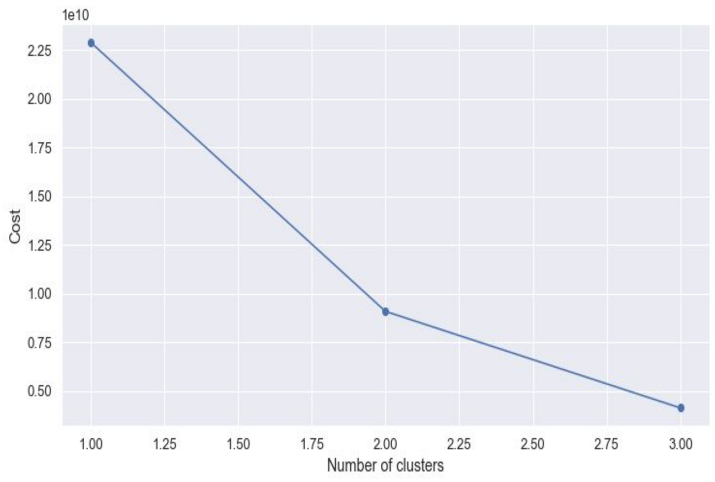
Here after clustering with kmeans clustering algorithm written from scratch we get the following scatter plot with 3 centroids.



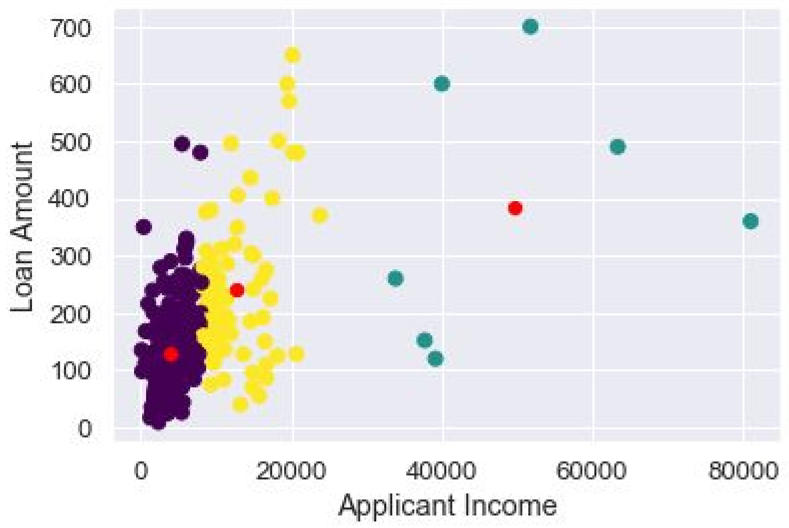
```
26]: color=['blue','green','cyan']
     for k in range(K):
         data=X[X["Cluster"]==k+1]
         plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])
     plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
     plt.xlabel('Income')
     plt.ylabel('Loan Amount (In Thousands)')
     plt.show()
```



We can see how the cost reduces :



With the inbuilt kmeans clustering package in python we get the same result as the algorithm we built from scratch proving the correctness of the devised algorithm.


Loan Amount
Applicant Income