TABLE OF CONTENTS

CHAPTER 1

TASK PERFORMED

1.1 Bank Loan Sanction Amount (Regression)

Bank Loan Sanction Amount has been of high interest area, it requires noticeable effort and knowledge of the field expert.Considerable number of distinct attributes are examined for the reliable and accurate prediction.Respective performances of different algorithms are compared to find the algorithm that best suits the available dataset.The final prediction models are evaluated using test data and the r2_score for each ML algorithm is obtained.

1.2 Problem Statement and Objective

Problem Statement:

Given a dataset consists of various parameters used to predict loan amount sanctioned to the customers who have applied for loan from the bank by providing their information/data to the bank. Here we have to do following tasks,

- Predict the Loan Sanctioned Amount to the customers from the bank when the information/data are specified accordingly.
- Use Exploratory Data Analysis and plot at least 7 meaningful graphs and illustrate the scenario and relationship between various parameters involved in that graph.
- Measure the r2_score of the Machine Learning Algorithms used to predict the loan sanction amount for the customers.
- Compare the r2_score of all the Machine Learning Algorithms used and deduce which model has the highest r2_score.

Objective:

- Identify the type of model to use in the problem statement. Regression in this case because we have to predict loan amount sanctioned.
- Clean the given dataset for any unwanted values and pass the dataset through a specific Machine Learning Algorithm.
- Fit the data accordingly and plot the graph between the tested data and predicted output and identify the r2_score of the particular Machine Learning Algorithm.
- Repeat the test with a few more Machine Learning Algorithms and note down the corresponding r2_score of each model.
- Plot the graphs to demonstrate Exploratory Data Analysis and explain the relationship between various parameters depicted in the specific graphs. A minimum of 7 graphs are to be plotted.

1.3 Project Description

Bank Loan Amount Sanction is an interesting problem. The number of people applying for the loan from the bank is increasing with time and it is likely that this will continue, and the number of people applying for loan will increase in feature. This adds additional significance to the problem of the loan sanction amount.

Accurate loan sanction amount involves expert knowledge, because loan sanction amount usually depends on many distinctive information and data. Typically, most significant ones are Age, Income, Income Stability, Loan Amount Request, Credit Score, Property Price. The Location of the customer and the Location of his Property also influence in the loan sanction amount. Different information like Profession, Type of Employment, Dependents, Co-Applicant, etc, will also influence in the loan sanction amount. In this project, we applied different methods and techniques of Regression in particular, in order to achieve higher r2_score of the loan sanction amount.

In this project we use various machine learning algorithms and python codes to examine the information/data of respective customers, to get loan sanction amount and also determine which information/data are significant to loan sanction amount and how well those variables describe the loan sanction amount. Finally, conclusions are drawn from the many graphs and plots obtained from the dataset, and ML models are trained to make the possible predictions.

1.4 Output and Results

1.4.1 Exploratory Data Analysis (EDA)

- NOTE:
  - Here df is the data frame.
- Data Cleaning 1:
  - df.drop(['Customer ID', 'Name', 'Property ID', 'Expense Type 1','Expense Type 2','Property Age'], axis=1, inplace = True)
    - Here I have dropped these columns because these columns where not useful to Predict Loan Sanction Amount.
  - df['Current Loan Expenses (USD)']=df['Current Loan Expenses (USD)'].replace(to_replace=-999, value=np.nan)
  - df['Co-Applicant'] = df['Co-Applicant'].replace(to_replace=-999, value=0)
    - Here in df two column had -999 value at some rows so I decided to replace it with some proper value.

➢ Plot 1:

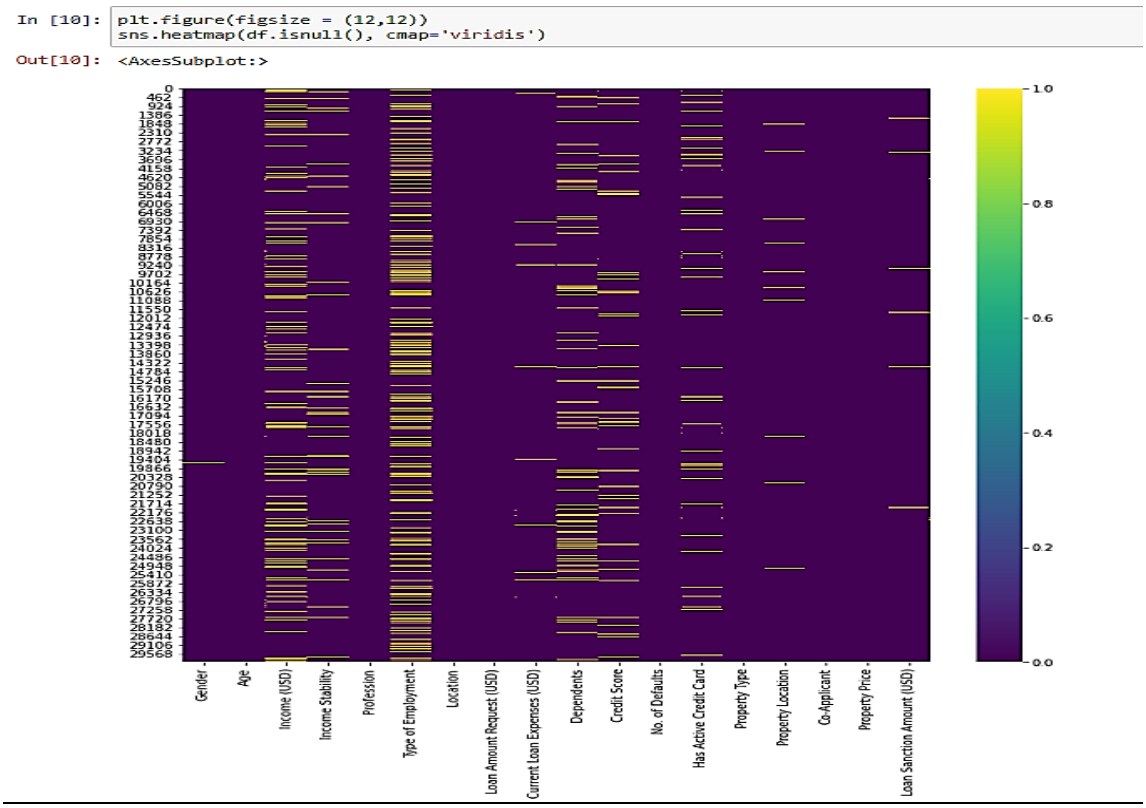


Figure 1: Heatmap to find null Value

Code:

- plt.figure(figsize = (12,12))
- sns.heatmap(df.isnull(), cmap='viridis')

Conclusions:

- In plot 1 by using heatmap form seaborn library with cmap as viridis we are able to find all the null or NaN values in the given dataset.
- Here by studying and analysing this heat map we further proceed to perform different cleaning techniques to add relevant data at the position of these missing values in the dataset.

➢ Plot 2:

```
In [11]: sns.boxplot(x='Income Stability',y='Age',data=df)
Out[11]: <AxesSubplot:xlabel='Income Stability', ylabel='Age'>
```

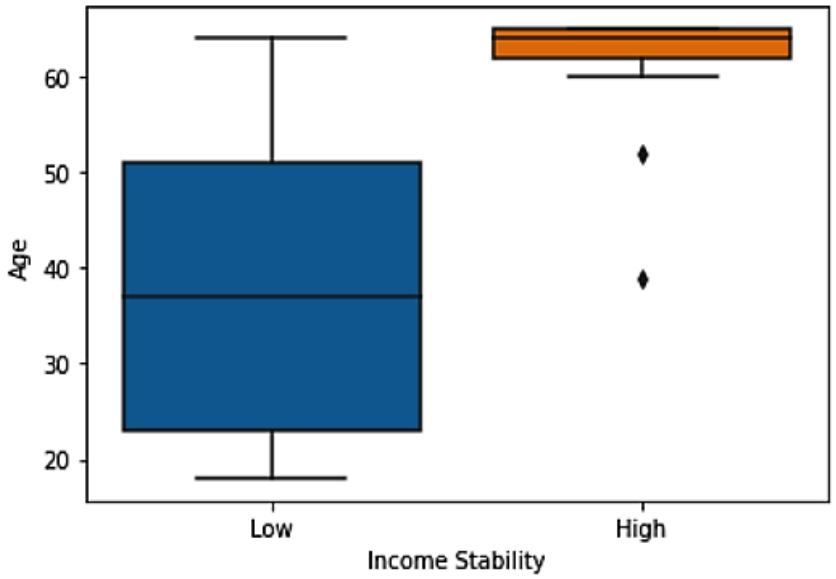


Figure 2: Boxplot of Age and Income Stability

Code:

- sns.boxplot(x='Income Stability', y='Age', data=df)

Conclusions:

- Here plot 2 is a boxplot from seaborn library and I have given Income Stability and Age as parameter to this plot.
- By analysing the plot obtained by boxplot I can say that Income Stability is High for customers having Age greater than 60 and Low for all other Age.
- This data can be used to fill the missing values of the Income Stability column.

- Data Cleaning 2:
  - def impute_Stability(cols):
    IncomeStability= cols[0]
    Age= cols[1]
    if pd.isnull(IncomeStability):
      if Age>60:
        return 'High'
      else:
        return 'Low'
    else:
      return IncomeStability

  - df['Income Stability']= df[['Income Stability', 'Age']].apply(impute_Stability, axis=1)
    - Here in the above function I have filled null row of Income Stability with High if the Age corresponding to that row is greater than 60 and Low for all other Age.

➢ Plot 3:

```
In [14]: sns.boxplot(x='Property Location',y='Property Type',data=df)
Out[14]: <AxesSubplot:xlabel='Property Location', ylabel='Property Type'>
```

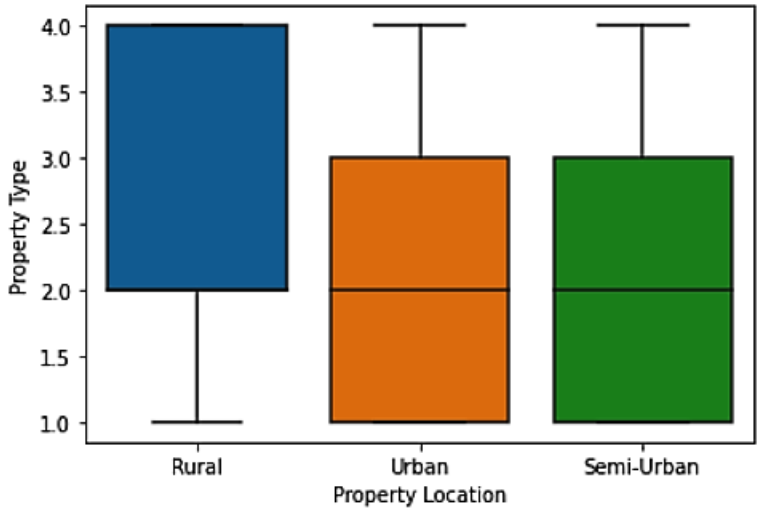


Figure 3: Boxplot of Property Type and Property Location

Code:

- sns.boxplot(x='Property Location', y='Property Type', data=df)

Conclusions:

- Here plot 3 is a boxplot from seaborn library and I have given Property Type and Property Location as parameter to this plot.
- By analysing the plot obtained by boxplot I can say that Property Type of the customers are 1 for Urban Property Location and Property Type of the customers are 2 for Semi-Urban Property Location and for all other Property Type of the customers the Property Location will be Rural.
- This data can be used to fill the missing values of the Property Location column.

- Data Cleaning 3:

  - def impute_loc(cols):

```
PropertyLocation= cols[0]
PropertyType= cols[1]
if pd.isnull(PropertyLocation):
    if PropertyType==1:
        return 'Urban'
    elif PropertyType==2:
        return 'Semi-Urban'
    else:
        return 'Rural'
else:
    return PropertyLocation
```

  - df['Property Location']=df[['Property Location', 'Property Type']].apply (impute_loc, axis=1)

    - Here in the above function I have filled null row of Property Location as Urban if Property Type corresponding to that row is 1 else if Property Type is 2 then Property Location is filled as Semi-Urban else for all other Property Type the Property location is filled as Rural.

➢ Plot 4:

```
In [17]: sns.boxplot(x='Dependents',y='Income Stability',data=df)
Out[17]: <AxesSubplot:xlabel='Dependents', ylabel='Income Stability'>
```

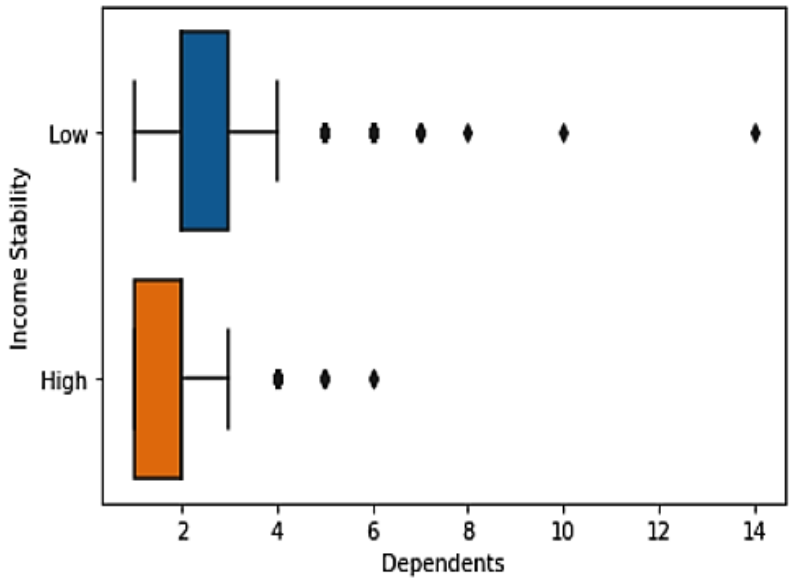


Figure 4: Boxplot of Income Stability and Dependents

Code:

- sns.boxplot(x='Dependents', y='Income Stability', data=df)

Conclusions:

- Here plot 4 is a boxplot from seaborn library and I have given Income Stability and Dependents as parameter to this plot.
- By analysing the plot obtained by boxplot I can say that Income Stability is high customers have 3 Dependents and Income Stability is Low when customers have 4 Dependents.

- Data Cleaning 4:
  - def impute_dep(cols):
    dep= cols[0]
    stab= cols[1]
    if pd.isnull(dep):
      if stab=='High':
        return 3
      else:
        return 4
    else:
      return dep

  - df['Dependents']=df[['Dependents', 'Income Stability']].apply(impute_dep, axis=1)
    - Here in the above function, I have filled null row of Dependents as 3 if Income Stability is High else, I have filled Dependents as 4.

  - df['Income (USD)']=df['Income (USD)'].replace(to_replace=np.nan, value= df ['Income (USD)'].mean())

  - df['Current Loan Expenses (USD)']=df['Current Loan Expenses (USD)'].replace(to_replace=np.nan, value=df['Current Loan Expenses (USD)'].mean())

  - df['Credit Score']=df['Credit Score'].replace(to_replace=np.nan, value= df['Credit Score'].mean())
  - df['Loan Sanction Amount (USD)']=df['Loan Sanction Amount (USD)'].replace(to_replace=np.nan, value=df['Loan Sanction Amount (USD)'].mean())

- Here in the above four codes, I have taken four columns and replaced all the rows with missing values in the columns with the mean of that particular column.

- df= df.fillna(method= 'ffill')
  - Here in the above code, I have used fillna from pandas library with method as ffill. It is used to forward fill all the missing values in the df i.e., all the remaining columns which had some missing values in their row will be filled with their previous row value.

➢ Plot 5:

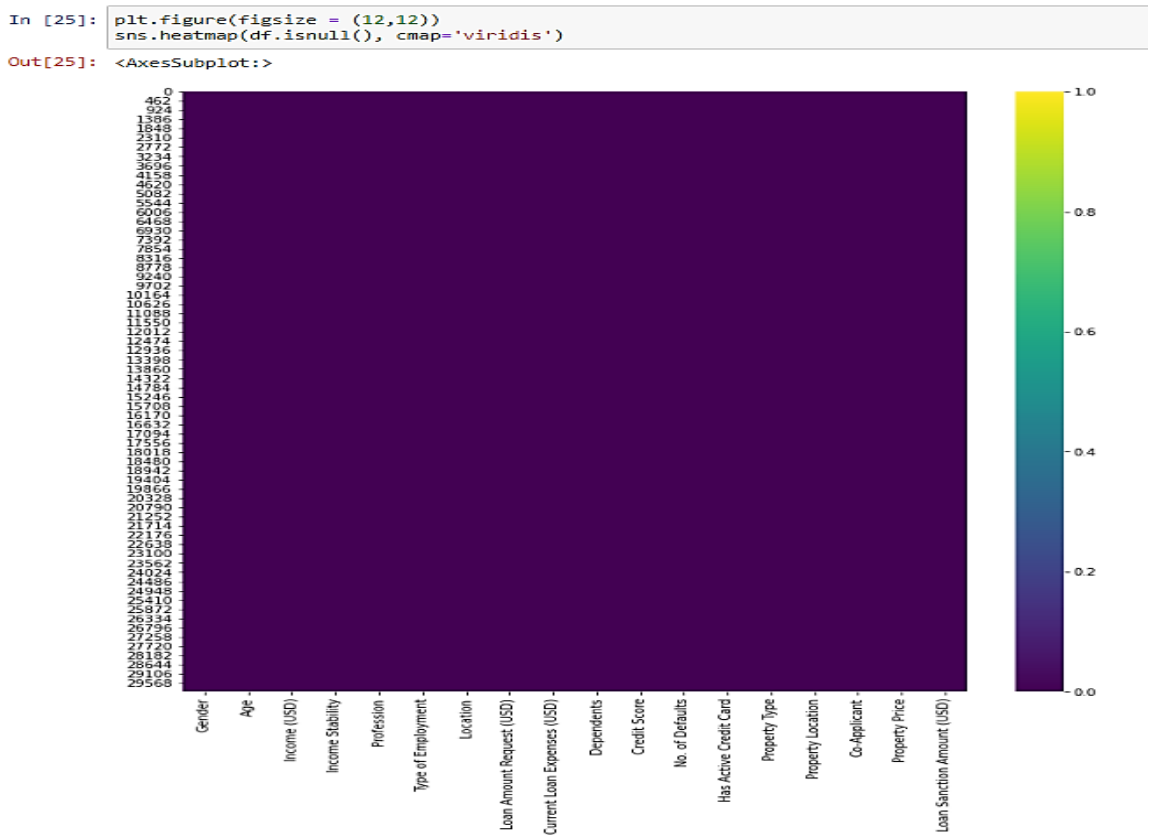


Figure 5: Heatmap to confirm no null values

Code:

- plt.figure(figsize = (12,12))
- sns.heatmap(df.isnull(), cmap='viridis')

Conclusions:

- In plot 5 by using heatmap form seaborn library with cmap as viridis we are able to find all the null or NaN values in the given dataset.
- By analysing the plot, I can say that I have successfully cleared all the missing values in the df.

➢ Plot 6:

```
In [26]: sns.barplot(x='Location',y= 'Loan Amount Request (USD)',hue='Gender',data=df)
Out[26]: <AxesSubplot:xlabel='Location', ylabel='Loan Amount Request (USD)'>
```

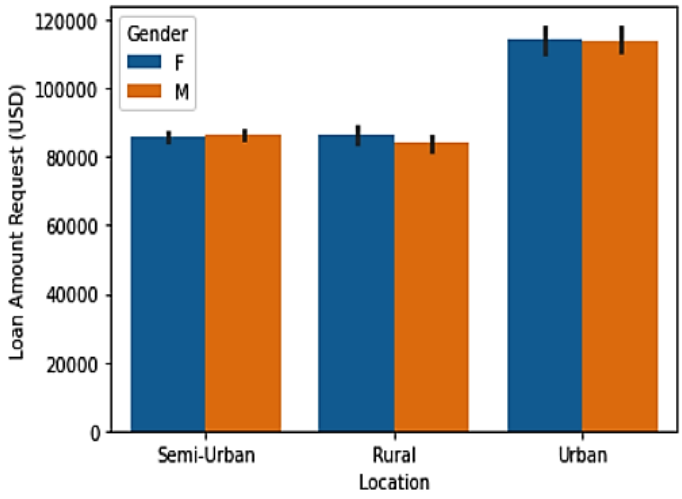


Figure 6: Barplot of Loan Amount Request(USD) and Location

Code:

- sns.barplot(x='Location', y= 'Loan Amount Request (USD)', hue='Gender', data=df)

Conclusions:

- Here plot 6 is a barplot from seaborn library and I have given Loan Amount Request (USD) and Location as parameter and Gender as hue to this plot.
- By analysing I can say that customers who are leave in Semi-Urban Location have requested for loan amount from 20000 to 80000 and the count of Male is slightly more from the Female.
- Customers who are leave in Rural Location have requested for loan amount from 20000 to 80000 USD and the count of Male is slightly less from the Female.

- Customers who are leave in Urban Location have requested for loan amount from 20000 to 120000 USD and the count of Male is slightly less from the Female.
- Count of customer requesting for loan amount is more from Urban Location and then from Rural Location after that from Semi-Urban Location.

➢ Plot 7:

```
In [27]: plt.figure(figsize = (16,8))
         sns.barplot(x='Profession',y='Loan Amount Request (USD)',hue='Gender',data=df)
Out[27]: <AxesSubplot:xlabel='Profession', ylabel='Loan Amount Request (USD)'>
```

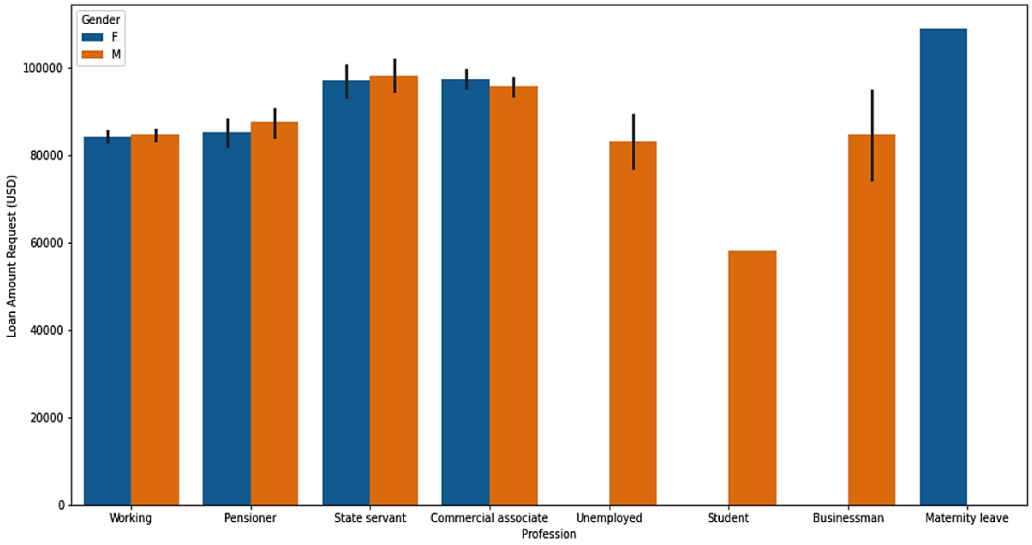


Figure 7: Barplot of 'Loan Amount Request (USD) and Profession

Code:

- plt.figure(figsize = (16,8))
- sns.barplot(x='Profession', y='Loan Amount Request (USD)', hue='Gender', data=df)

Conclusions:

- Here plot 7 is a barplot from seaborn library and I have given Loan Amount Request (USD) and Profession as parameter and Gender as hue to this plot.
- By analysing I can say that customers having Professions like Working, Pensioner, Unemployed, Businessman have requested loan amount from 20000 to 80000 USD and count of Male is slightly more than Female in Profession like Working, Pensioner and there are only male in professions like Unemployed and Businessman.

- customers having Professions like State servant, commercial associate have requested loan amount from 20000 to 90000 USD and count of Male and Female is almost equal in this Professions.
- customers having Profession like Maternity leave have requested loan amount from 20000 to 110000 USD and there are only Female in this Profession.
- customers having Profession like Student have requested loan amount from 20000 to 50000 USD and there are only Male in this Profession.
- The count of customers requesting for loan amount is highest for Profession like Businessman and lowest for Professions like Maternity leave and Student.

➢ Plot 8:

```
In [28]: sns.jointplot(x='Current Loan Expenses (USD)',y='Loan Amount Request (USD)',data=df)
Out[28]: <seaborn.axisgrid.JointGrid at 0x1450af71e80>
```

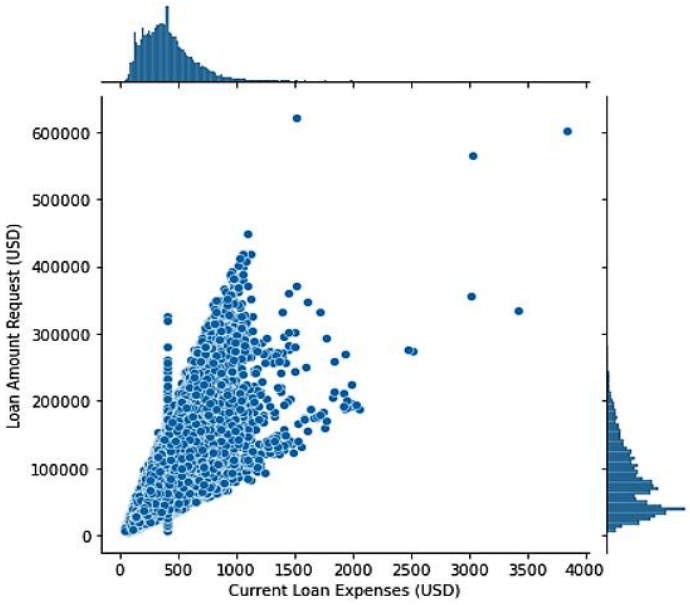


Figure 8: Jointplot of Loan Amount Request (USD) and Current Loan Expenses (USD)

Code:

- sns.jointplot(x='Current Loan Expenses (USD)', y='Loan Amount Request (USD)', data=df)

Conclusions:

- Here plot 8 is a jointplot from seaborn library and I have given Loan Amount Request (USD) and Current Loan Expenses (USD) as parameter to this plot.
- By analysing I can say that Loan Amount Request (USD) and Current Loan Expenses (USD) are directly proportional to each other.
- Customers with Loan Amount Request from 0 to 450000 USD are having Current Loan Expenses from 0 to 2000 USD respectively.

➢ Plot 9:

```
In [29]: sns.displot(df['Age'])
Out[29]: <seaborn.axisgrid.FacetGrid at 0x23ee433e130>
```

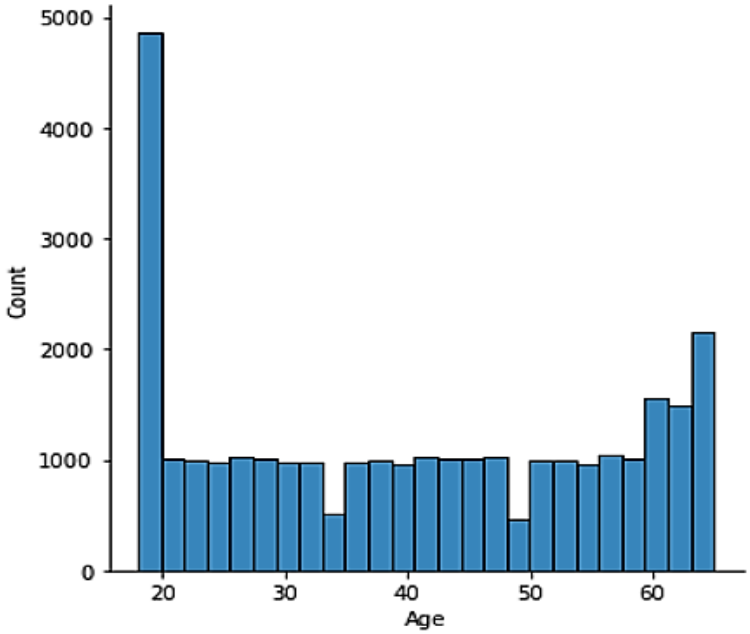


Figure 9: Displot of Age

Code:

- sns.displot(df['Age'])

Conclusions:

- Here plot 9 is a distplot from seaborn library and I have given Age as parameter to this plot.
- distplot is used here to obtain count of customers based to their Age.
- By analysing I can say that Age of customers varies from 18 to 65.
- Customers having Age of 18 to 20 are have count of 4900 each and then customers having Age of 60 to 65 are having count of 1000 to 2000 each and then customers having Age of 21 to 59 are having count of 500 to 1000 each.

➢ Plot 10:

```
In [30]: plt.figure(figsize = (14.5,4.5))
         sns.boxplot(x='Profession',y='Age',data=df)
Out[30]: <AxesSubplot:xlabel='Profession', ylabel='Age'>
```

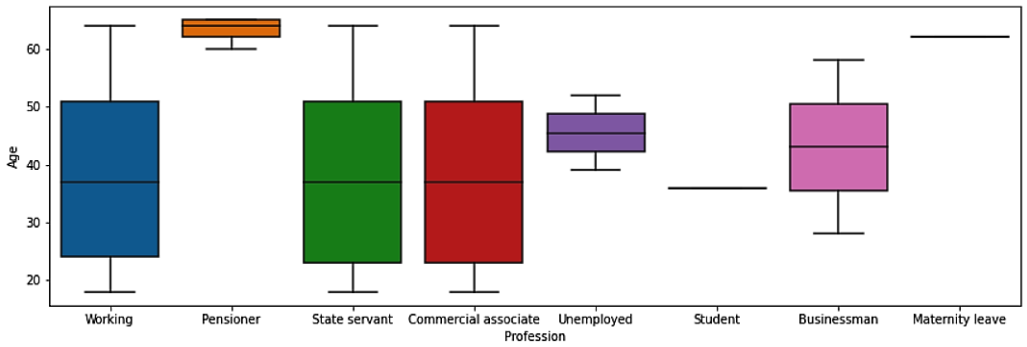


Figure 10: Boxplot of Profession and Age

Code:

- plt.figure(figsize = (14.5,4.5))
- sns.boxplot(x='Profession', y='Age', data=df)

Conclusions:

- Here plot 10 is a boxplot from seaborn library and I have given Profession and Age as parameter to this plot.
- By analysing I can say that customers having Professions like Working, State servant, Commercial associate are having Age of 18 to 65.
- Customers having Professions like Student and Maternity leave are having Age of 38 and 65 respectively.
- Customers having Profession like pensioner are having age of 60 to 65.
- Customers having Profession like unemployed are having age of 40 to 50.
- Customers having Profession like businessman are having age of 28 to 58.
- This plot helps us to understand customers profession with respect to customers Age.

- Data Cleaning 5:
  - One-Hot Encoding:
    - from sklearn.preprocessing import LabelEncoder
    - l= LabelEncoder()
    - sex=pd.get_dummies(df['Gender'], drop_first=True)
    - lowstability=pd.get_dummies(df['Income Stability'], drop_first =True)
    - df=pd.concat([df, sex, lowstability], axis=1)
  - Label Encoding:
    - df['Professions']=l.fit_transform(df['Profession'])
    - df['creditcard']=l.fit_transform(df['Has Active Credit Card'])
    - df['emptype']=l.fit_transform(df['Type of Employment'])
    - df.drop(['Gender', 'Income Stability', 'Profession', 'Location', 'Has Active Credit Card', 'Property Location', 'Type of Employment'], axis=1, inplace=True)

1.4.2 Prediction

- It will predict the Loan Sanction Amount to a person from the dataset give by the Bank.
- Here Exploratory Data Analysis is performed which includes:
- Data Cleaning, and use of different plots to understand the dataset.
- The dataset are been splitted into two data:
- Training (80% of the dataset)
- Testing (20% of the dataset)
- We are using five Machine Learning algorithms with the given dataset.

1.4.3 Preparing Machine Learning Model

Given dataset consists of various parameters used to predict loan amount sanctioned to the customers who have applied for loan from the bank by providing their information/data to the bank.

Since we need to predict loan sanction amount, the models used are called Regression Models. The Machine Learning Algorithms I used here are:

- Linear Regression Model
- Decision Tree Regression Model
- Support Vector Regression Model
- Random Forest Regression Model
- Bayesian Regression Model

1.4.3.1 Linear Regression Model

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models target a prediction value based on independent variables. It is mostly used for finding out the relationship between

Linear Regression Model

```
In [40]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(df.drop('Loan Sanction Amount (USD)',axis=1),
                                                             df['Loan Sanction Amount (USD)'], test_size=0.2, random_state=62)
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import r2_score
         lm = LinearRegression()
         lm.fit(X_train,y_train)
         pred1 = lm.predict( X_test)
         score = r2_score(y_test,pred1)
         print("R2_Score: ",np.round(score,decimals=4))

         R2_Score:  0.6442
```

variables and forecasting.

Figure 11: Linear Regression Model

1.4.3.2 Decision Tree Regression Model

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

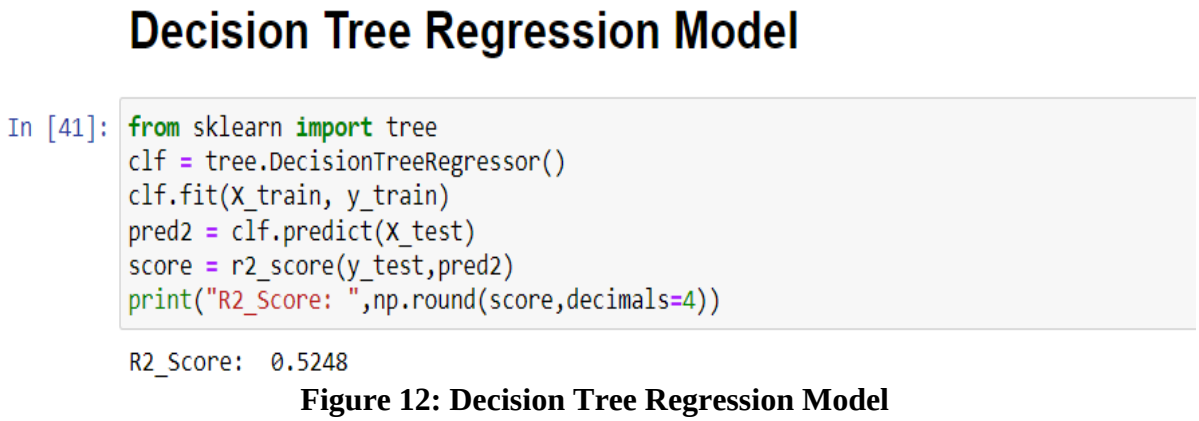
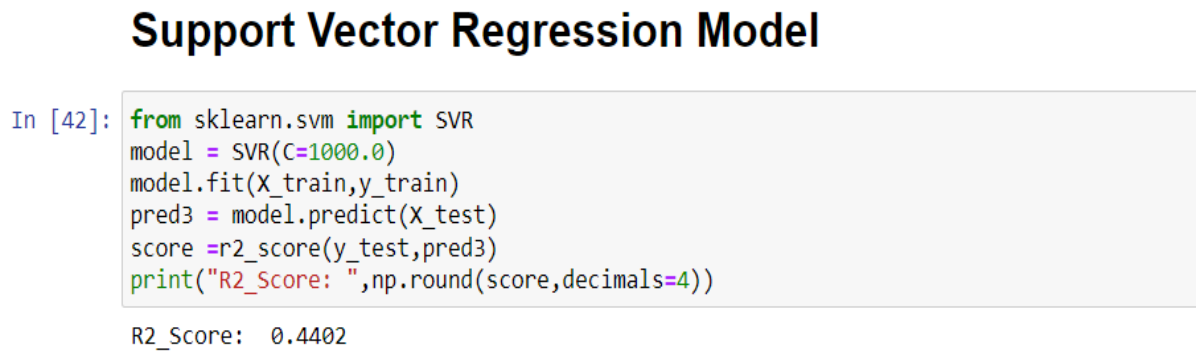


Figure 12: Decision Tree Regression Model

1.4.3.3 SVM(Support Vector Machine)

A support vector machine (SVM) is a type of deep learning algorithm that performs supervised learning for classification or regression of data groups. In AI and machine learning, supervised learning systems provide both input and desired output data,



which are labeled for classification.

Figure 13: SVM(Support Vector Machine)

1.4.3.4 Random Forest Regression Model

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

Random Forest Regression Model

In [43]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)
pred_rf = rfr.predict(X_test)
score = r2_score(y_test,pred_rf)
print("R2_Score: ",np.round(score,decimals=4))
```

```
R2_Score:  0.7432
```

Figure 14: Random Forest Regression Model

1.4.3.5 Bayesian Regression Model

Linear Regression is a very simple machine learning method in which each datapoints is a pair of vectors: the input vector and the output vector.

Bayesian Regression Model

In [44]:
```
from sklearn import linear_model
reg = linear_model.BayesianRidge()
reg.fit(X_train, y_train)
predic = reg.predict(X_test)
score = r2_score(y_test,predic)
print("R2_Score: ",np.round(score,decimals=4))
```

```
R2_Score:  0.6442
```

Figure 15: Bayesian regression model

Machine Learning Model Chart:

Serial Number	ML Algorithm Used	Accuracy Score (approx. 4 decimal places)
01	Random Forest Regression Model	0.7432
02	Linear Regression Model	0.6442
03	Bayesian Regression Model	0.6442
04	Decision Tree Regression Model	0.5248
05	Support Vector Regression Model	0.4402

Figure 16: Machine Learning Model Chart

REFERENCES

- https://www.geeksforgeeks.org/
- https://stackoverflow.com/
- https://seaborn.pydata.org/
- https://scikit-learn.org/stable/