Loan amount prediction based on credit history and other variables

Group members:

Kinjal Joshi, Satvika Sai, Shweta Nagumalli, Mallesh Mittapalli, Rohit Nanaware

Introduction:

In this project, we are exploring classification problem using loan amount prediction data. This problem deals with whether the loan would be approved or not. This prediction is based on credit history and other variables.

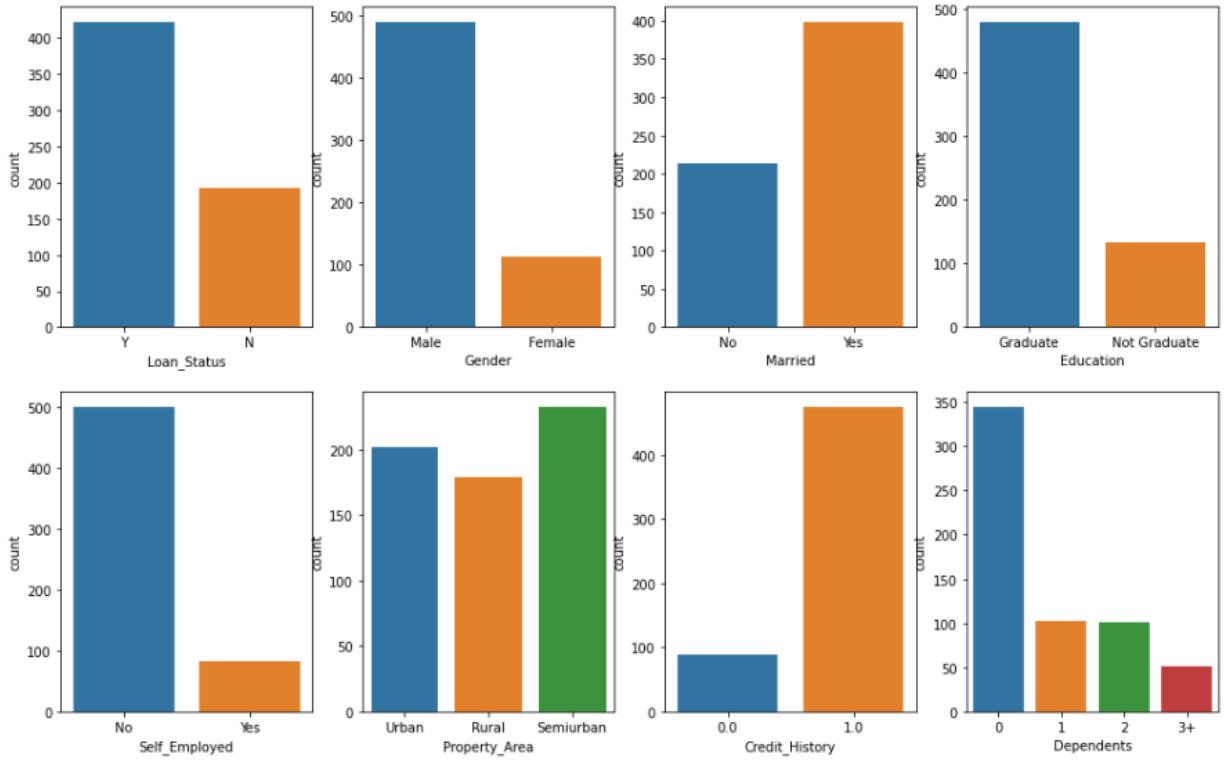
Data Description:

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	(Target) Loan approved (Y/N)

Approach:

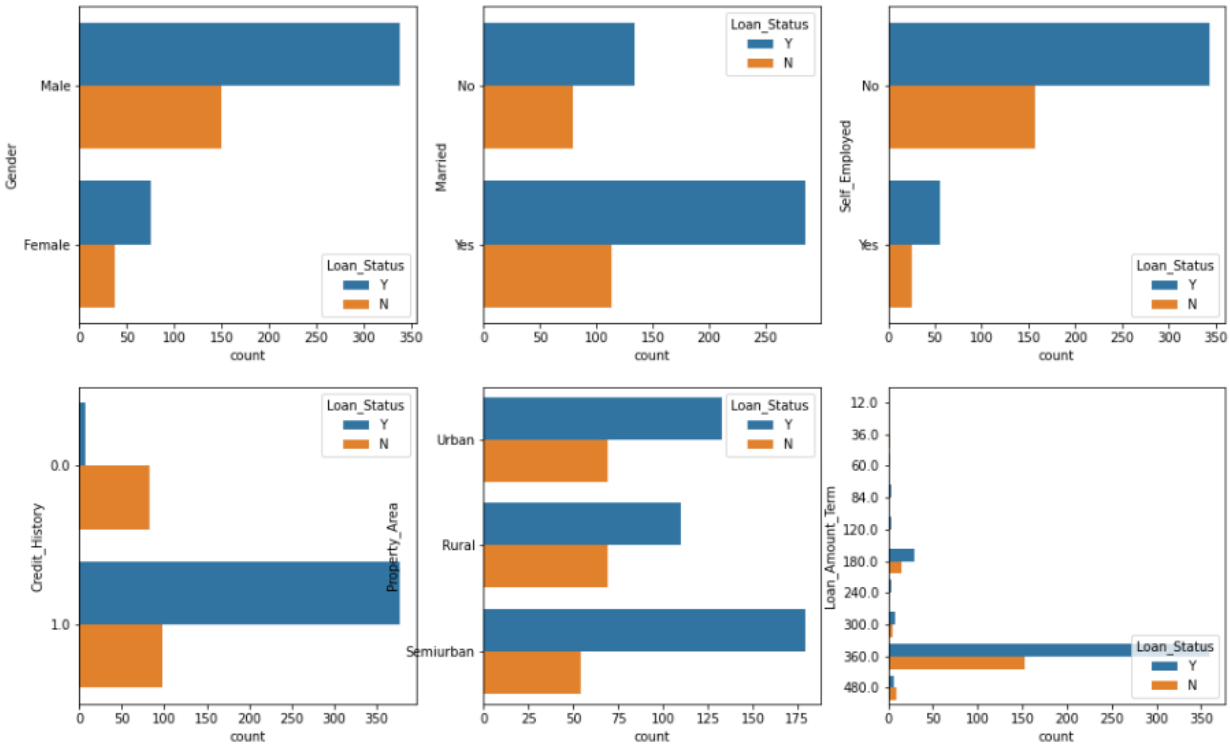
Our approach to solve this project, at first, we had imported all the useful modules and libraries. Then we accessed the data regarding the bank loan amount allocation, followed by the pre-processing and the exploratory data analysis which includes plotting and layouts. Then we went ahead with the visualization and normalization process. And then, finally we predicted the value and compared the error scores of all the models used, to find which model gives the optimal solution.

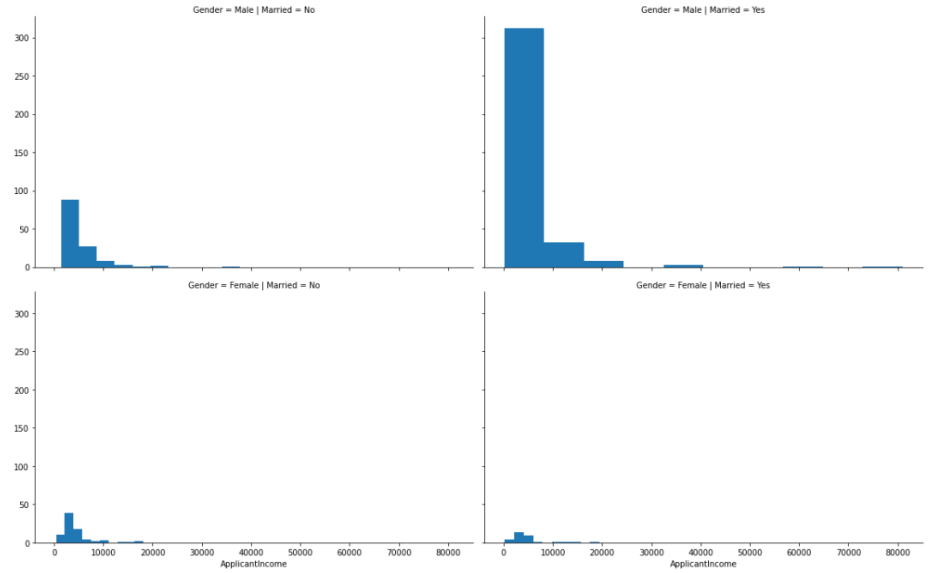
EDA:



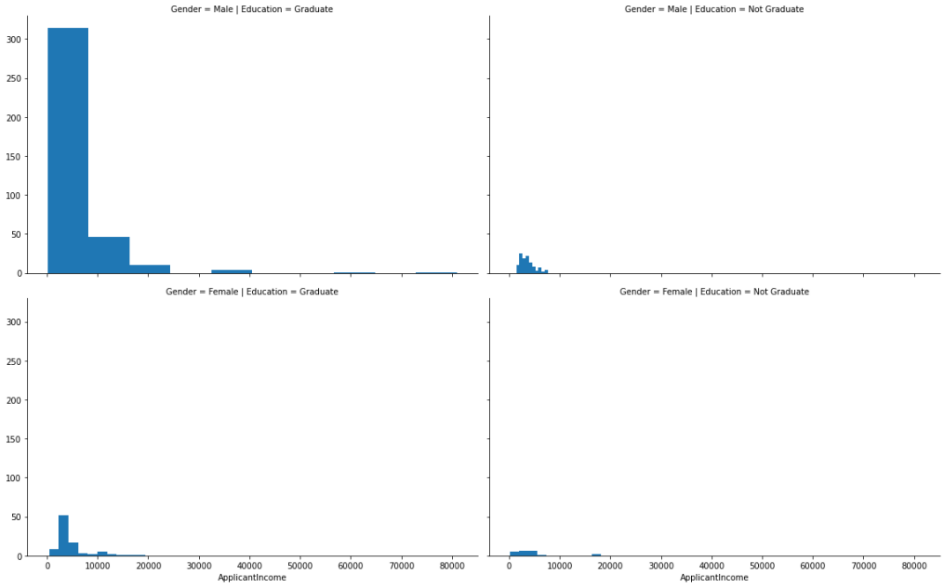
OBSERVATIONS:

1) More Loans are approved Vs Rejected
2) Male applicants are more than Female
3) Married applicant is more than non-married
4) Graduates are more than non-Graduates
5) Self-employed people are less than that of non-Self-employed
6) Maximum properties are located in Semiurban areas
7) Credit History is present for many applicants
8) The count of applicants with several dependents=0 is maximum

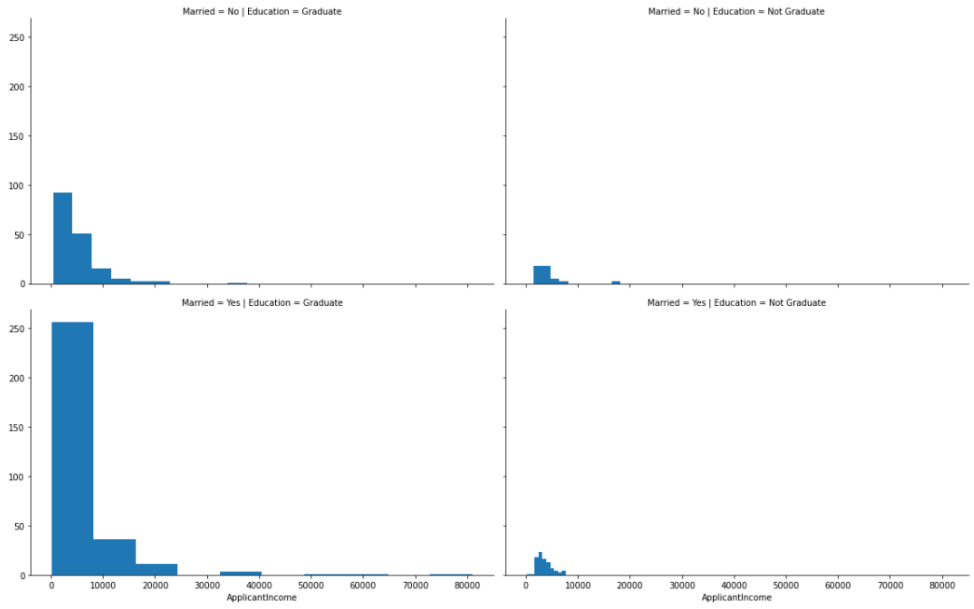




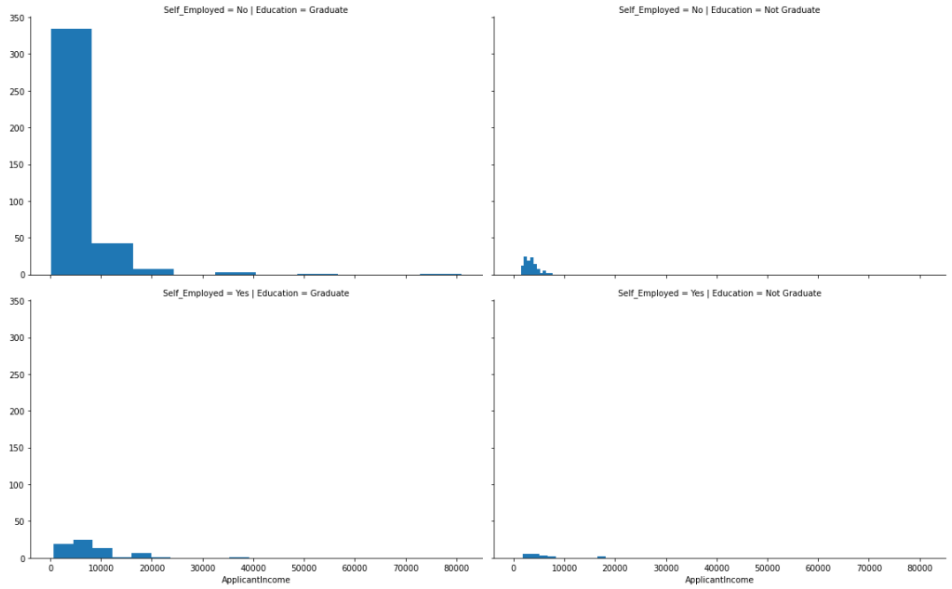
Observation: Males generally have the highest income. Explicitly, Males that are married have greater income that unmarried male.



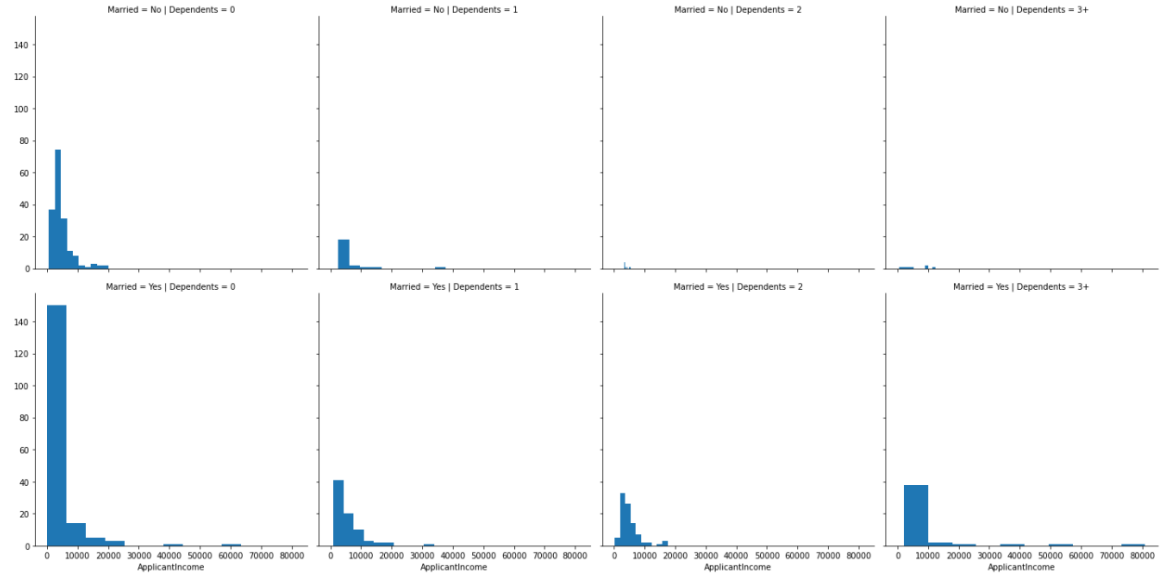
Observation: A graduate who is a male has more income.



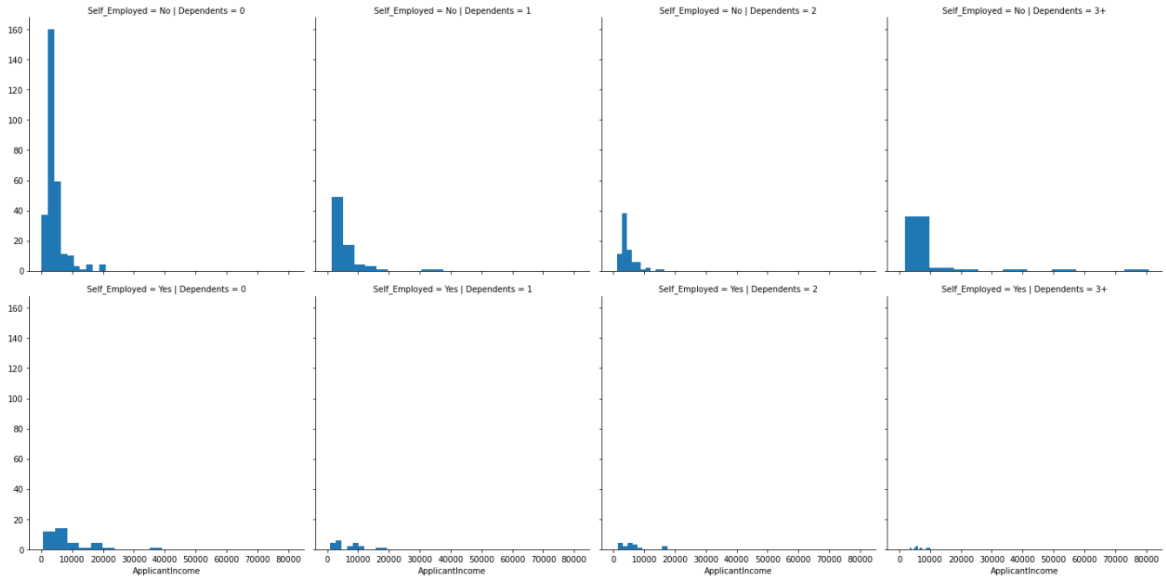
Observation: A graduate and married individual has more income.



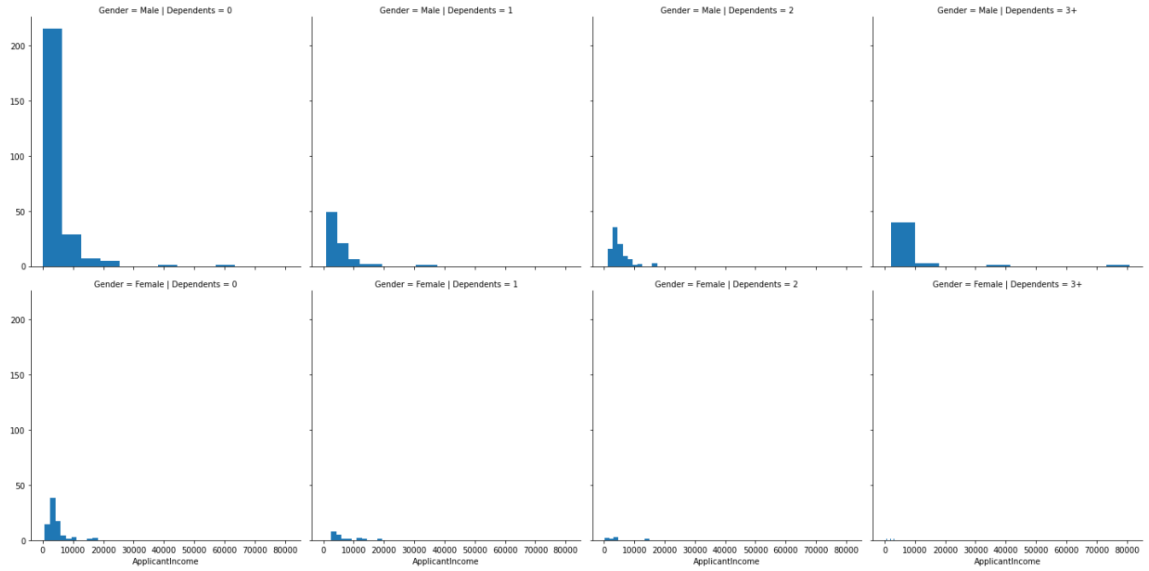
Observation: A graduate but not self-employed has more income.



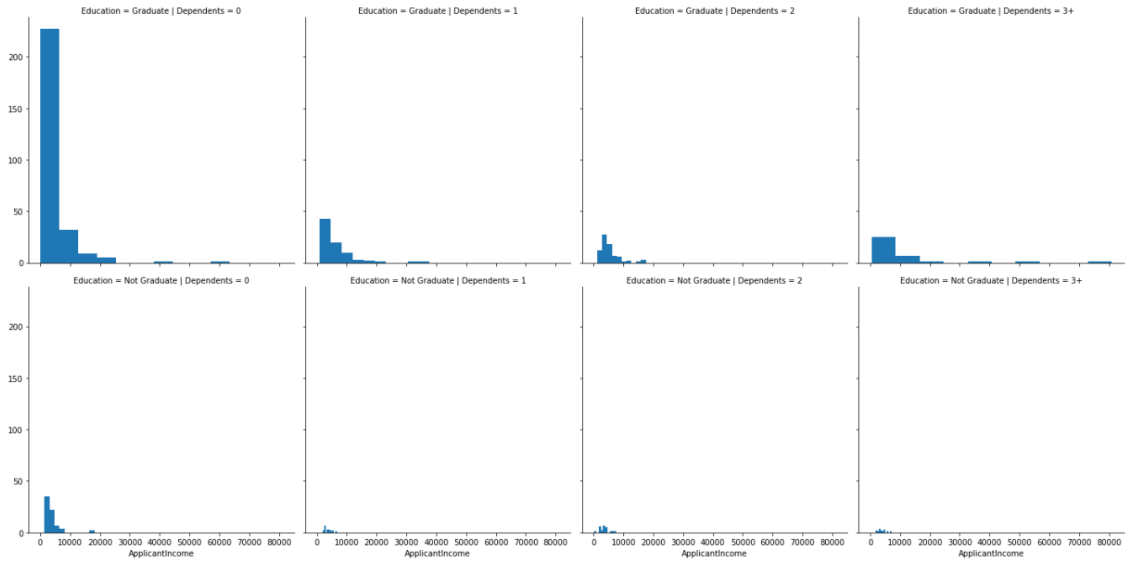
Observation: Unmarried and no one is dependent on such has more income. Also, Married and no one dependent has greater income with a decreasing effect as the dependents increases.



Observation: No one is dependent and self-employed has more income.

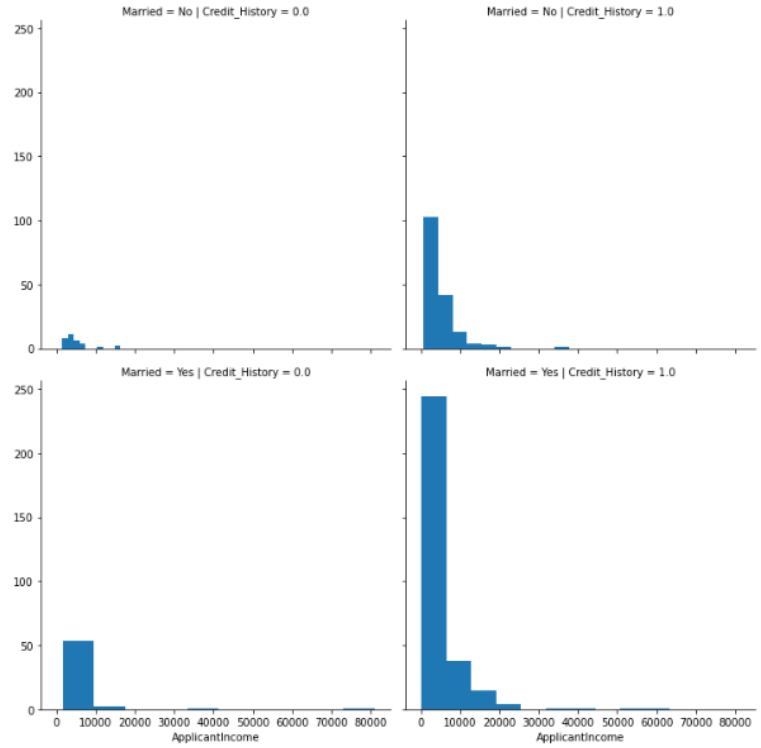


Observation: No one is dependent and a male tremendously has more income.

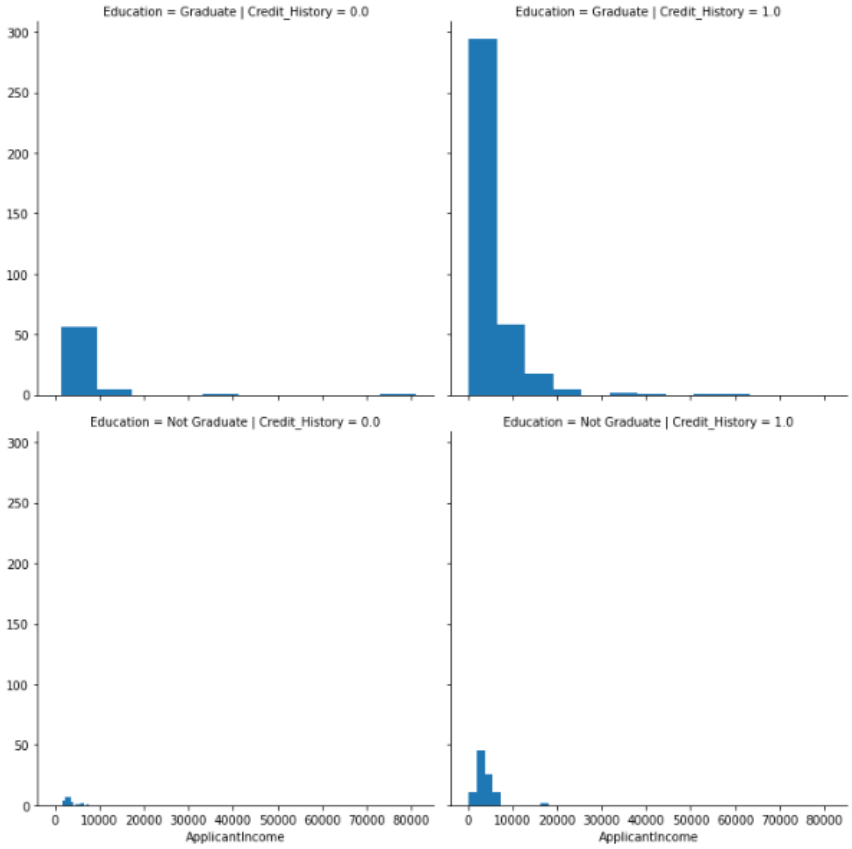


Observation: A graduate with no one dependent has more income.

Observation: No one is dependent and have property in urban, rural and semiurban has more income.

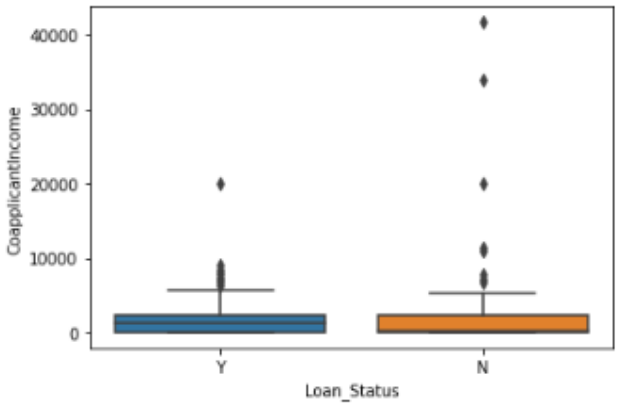


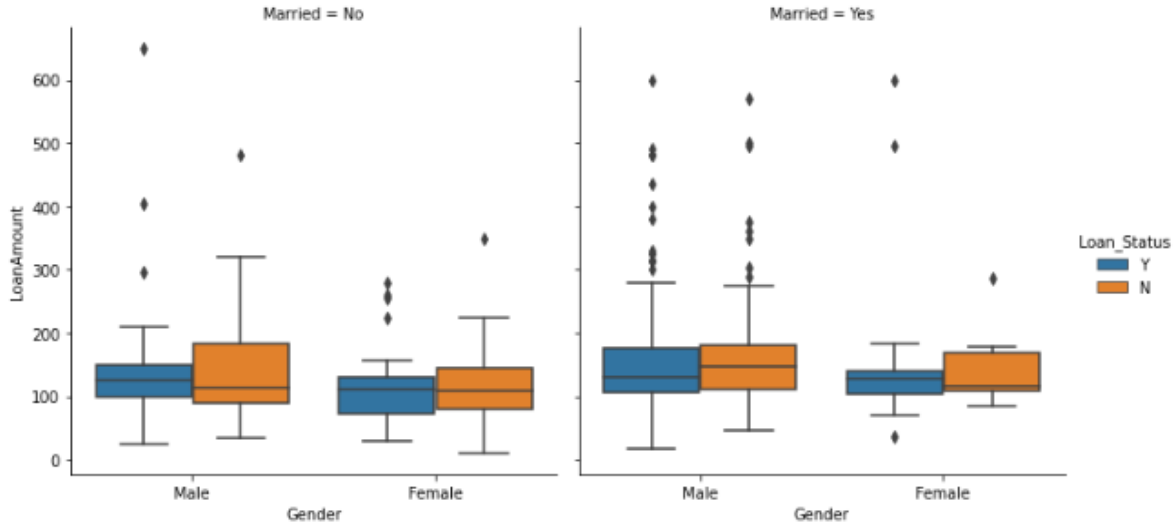
Observation: Married and has a good credit history depicts more income. Also, Not married but has a good credit history follows in the hierarchy.

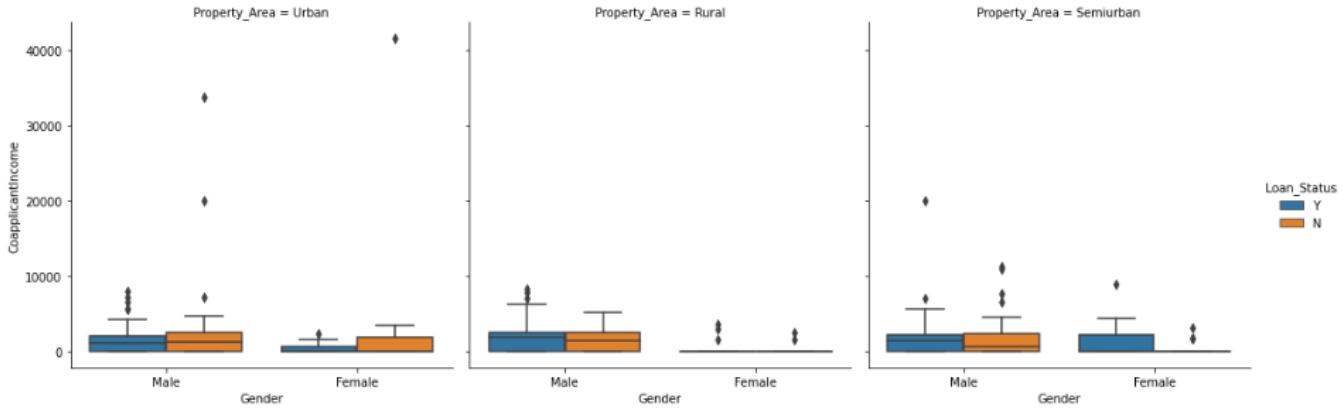


Observation: Educated with good credit history depicts a good income. Also, not a graduate and have a good credit history can be traced to having a better income than a fellow with no degree.

Box Plots:

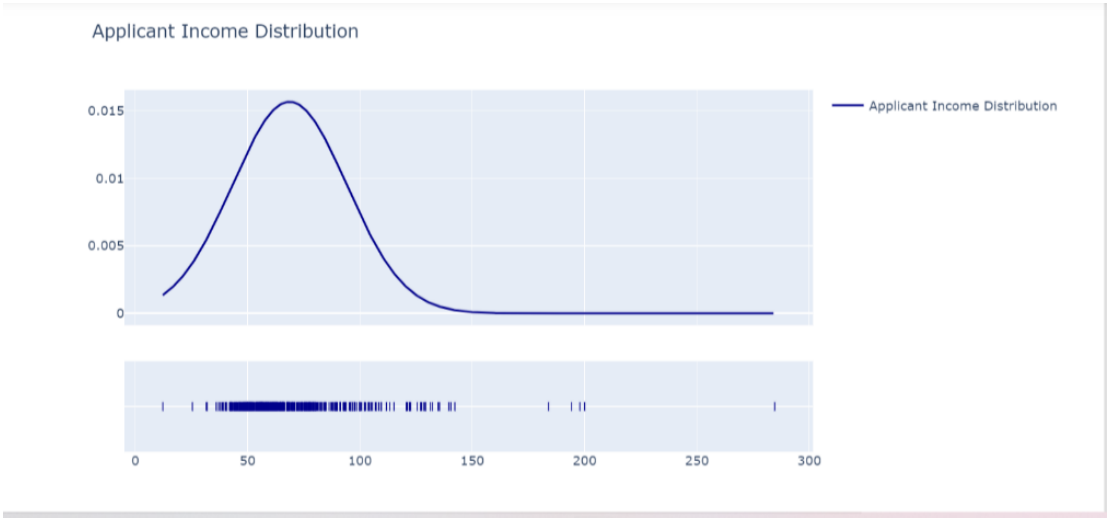

CoapplicantIncome
40000
30000
20000
10000
0
Y
N
Loan_Status

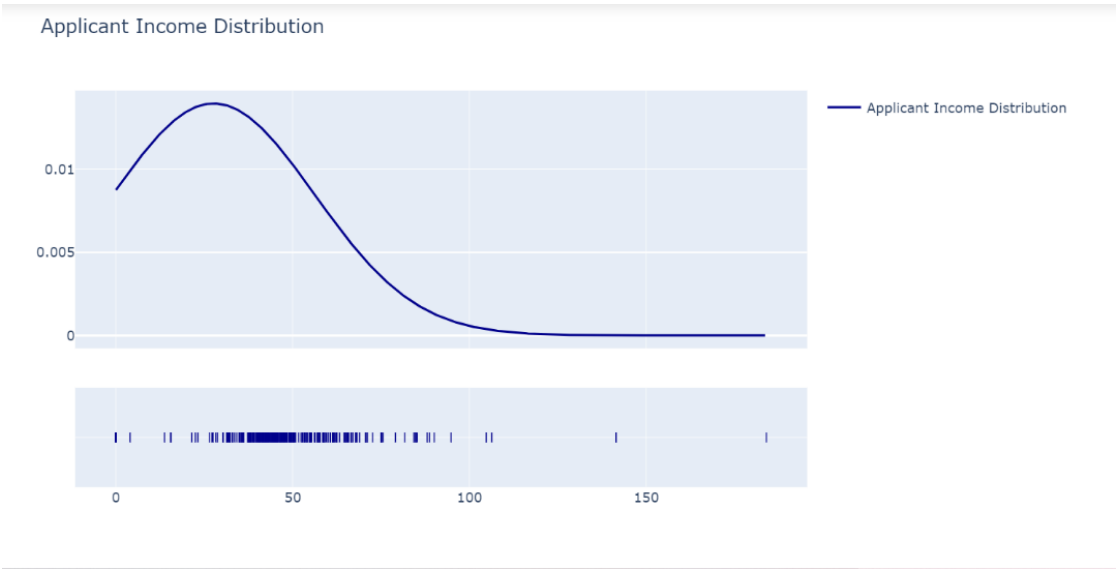

Married = No
Married = Yes
LoanAmount
600
500
400
300
200
100
0
Male
Female
Gender
Loan_Status
Y
N

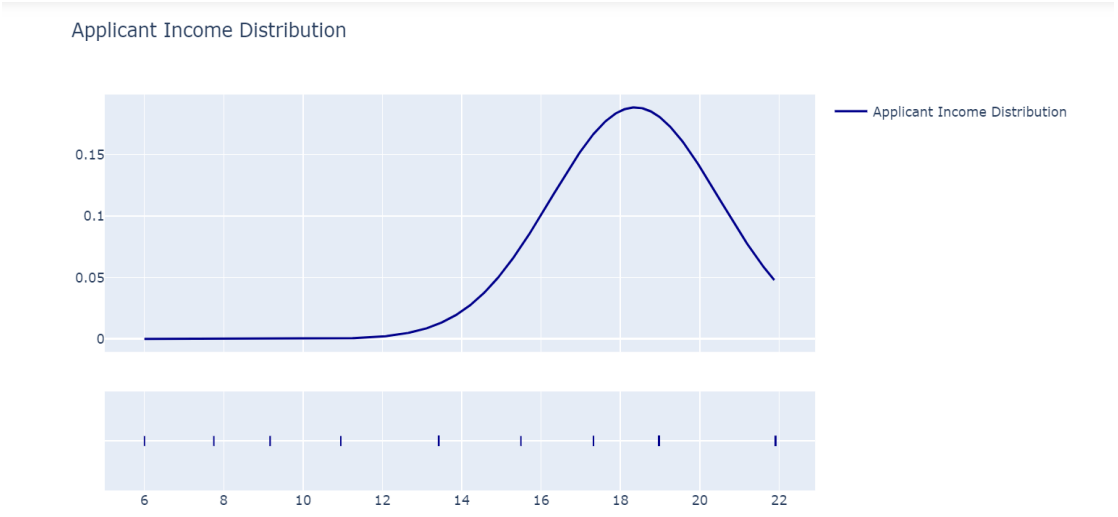

Property_Area = Urban
Property_Area = Rural
Property_Area = Semiurban
Coapplicantincome
40000
30000
20000
10000
0
Male
Female
Gender
Loan_Status
Y
N

Normalization:

Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is not necessary for all datasets in a model. It is required only when features of machine learning models have different ranges.

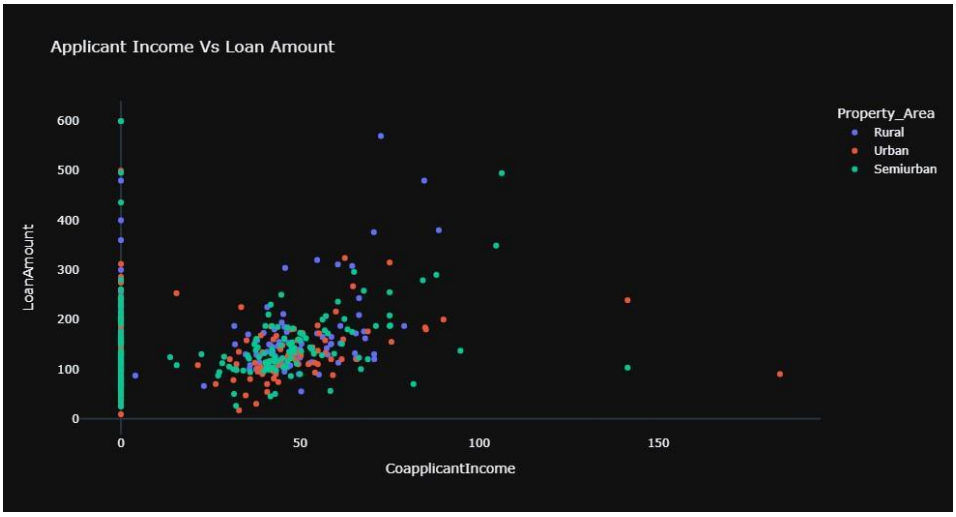


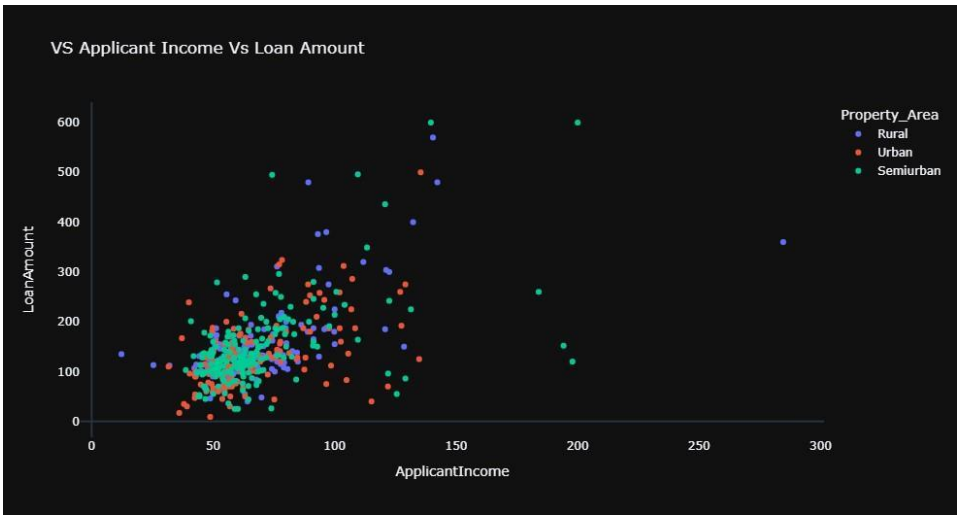


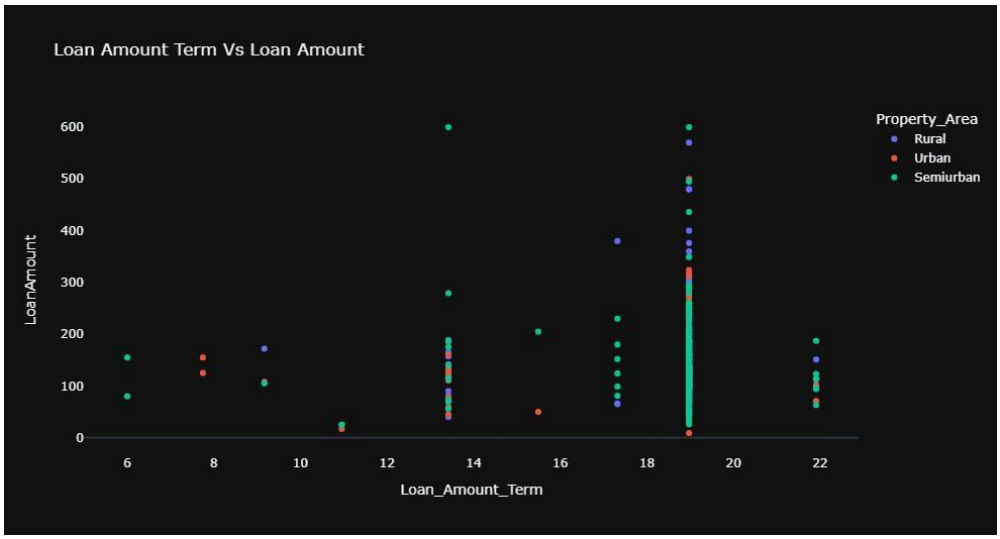


Visualization:

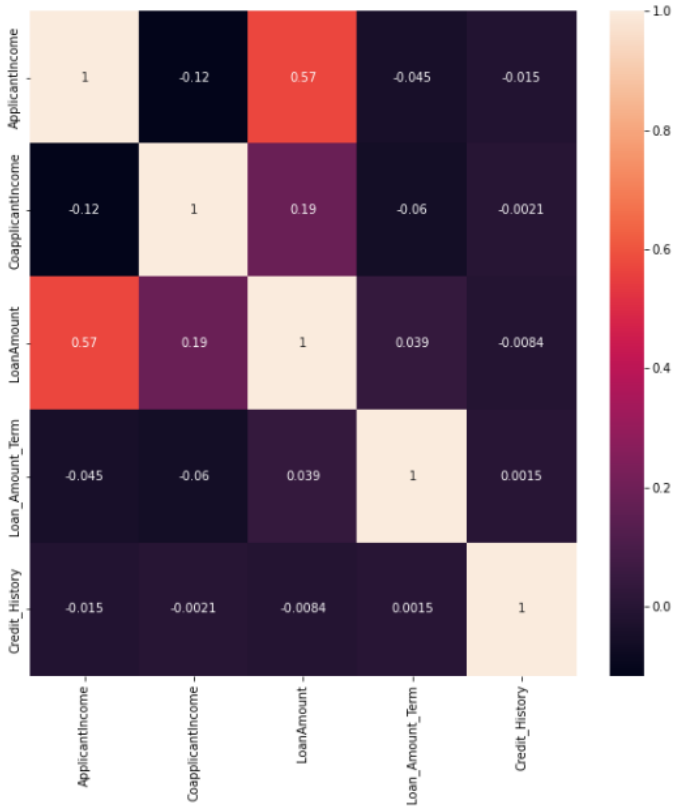
1) Linearity test: We tried to know whether we can segregate the Loan Status based on Applicant Income and loan amount, coapplicant income vs loan amount and finally loan amount term vs loan amount. So, by this we can get idea of the amount to be allocated.







2) Correlation matrix: A correlation matrix is simply a table which displays the correlation. It is best used in variables that demonstrate a linear relationship between each other. coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table.



Algorithm:

As we mentioned, we have used various classification models on this dataset and they have different accuracy and other performance measures. We have used the following machine learning algorithms on our dataset.

1) Linear Regression Model: Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

LINEAR REGRESSION MODEL :-

```
In [74]: linear_model = LinearRegression()

In [75]: linear_model.fit(X_train,Y_train)

Out[75]: LinearRegression()

In [76]: X_test = np.nan_to_num(X_test)
         Y_test = np.nan_to_num(Y_test)

In [77]: Y_train_pred = linear_model.predict(X_train)
         Y_test_pred = linear_model.predict(X_test)

In [78]: print(mean_absolute_error(Y_train,Y_train_pred))
         print(mean_absolute_error(Y_test,Y_test_pred))

         36.13170285533496
         36.58878462186987
```

2) Decision Tree 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.

DECISION TREE MODEL ;-

```
In [79]: dt_model = DecisionTreeRegressor()
         dt_model.fit(X_train,Y_train)

Out[79]: DecisionTreeRegressor()

In [80]: Y_train_pred = dt_model.predict(X_train)
         Y_test_pred = dt_model.predict(X_test)

In [81]: print(mean_absolute_error(Y_train,Y_train_pred))
         print(mean_absolute_error(Y_test,Y_test_pred))

         0.07291666666666667
         43.197916666666664
```

3) Random Forest Model: It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

RANDOM FOREST MODEL ;-

```
In [82]: rf_model =RandomForestRegressor(max_depth=3)
         rf_model.fit(X_train,Y_train)

Out[82]: RandomForestRegressor(max_depth=3)

In [83]: Y_train_pred = rf_model.predict(X_train)
         Y_test_pred = rf_model.predict(X_test)

In [84]: print(mean_absolute_error(Y_train,Y_train_pred))
         print(mean_absolute_error(Y_test,Y_test_pred))

         35.74557889822062
         38.52698284655441
```

4) Support Vector Regression: Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.

SUPPORT VECTOR REGRESSOR MODEL ;-

```
In [85]: svr_model=SVR(kernel = 'linear')
         svr_model.fit(X_train,Y_train)

Out[85]: SVR(kernel='linear')

In [86]: Y_train_pred = svr_model.predict(X_train)
         Y_test_pred = svr_model.predict(X_test)

In [87]: print(mean_absolute_error(Y_train,Y_train_pred))
         print(mean_absolute_error(Y_test,Y_test_pred))

         35.54781520317576
         35.880497790164604
```

Result and Discussion:

Linear Regression:

```
Results for Test data:
RMSE is 51.75286584336139
Absolute error score is 36.58878462186987
R2 score is 0.3411980827732026
Testing accuracy for this model is 34.11980827732026 %
Accuracy for this model is 34.11980827732026 %

Results for Train data:
RMSE is 58.74640441772771
Absolute error score is 36.13170285533496
R2 score is 0.5099279634760394
Testing accuracy for this model is 50.99279634760394 %
Accuracy for this model is 50.99279634760394 %
```

Decision Tree model:

```
MSE for train data:  0.07291666666666667
MSE for test data:  43.604166666666664
```

Random Forest model:

```
MSE for train data:  35.69306079349079
MSE for test data:  38.35123264939914
```

Support vector regression model:

```
MSE for train data:  35.54781520317576
MSE for test data:  35.880497790164604
```

Conclusion:

The support vector regression model performs better than all the other model due to low MSE difference between test and train data.