Loan Approval Predictor

This project basically helps in determining whether the particular loan of a peoples get approvement or not based on certain features such as:

- education
- employment_status,
- income_annum,
- loan_amount,
- loan_term,
- cibil_score,
- residential_assets_value,
- commercial_assets_value,
- luxury_assets_value.

Importing libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import numpy as np
import statsmodels.stats.api as sms
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
from sklearn.preprocessing import LabelEncoder
import pickle
```

Data Collection

In [2]:
```
df = pd.read_csv("loan_approval_dataset.csv")
df.head()
```

Out[2]:

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loar
0	1	2	Graduate	No	9600000	29900000	
1	2	0	Not Graduate	Yes	4100000	12200000	
2	3	3	Graduate	No	9100000	29700000	
3	4	3	Graduate	No	8200000	30700000	
4	5	5	Not Graduate	Yes	9800000	24200000	

In [3]:
```
1 df.shape
```

Out[3]:
```
(4269, 13)
```

- Here we have a dataset of loan_approval with the shape of (4269,13) i.e 4269 rows and 13 cols.

In [4]:
```
1 df.dtypes
```

Out[4]:
```
loan_id                    int64
 no_of_dependents          int64
 education                object
 self_employed            object
 income_annum              int64
 loan_amount               int64
 loan_term                 int64
 cibil_score               int64
 residential_assets_value  int64
 commercial_assets_value   int64
 luxury_assets_value       int64
 bank_asset_value          int64
 loan_status              object
dtype: object
```

In [5]:
```
1 #correcting the column names by removing any spaces
2 df.columns = df.columns.str.strip()
```

In [6]:
```
1 df["education"].unique()
```

Out[6]:
```
array([' Graduate', ' Not Graduate'], dtype=object)
```

Data Cleaning & Statistical Analysis

Descriptive statistics

In [7]:
```
# Here we will get the 5-Point summary about our data i.e min,quartile
# Also it provides the mean,and std. of numerical_columns

data = df.describe()
data
```

Out[7]:

	loan_id	no_of_dependents	income_annum	loan_amount	loan_term	cibil_s
count	4269.000000	4269.000000	4.269000e+03	4.269000e+03	4269.000000	4269.000
mean	2135.000000	2.498712	5.059124e+06	1.513345e+07	10.900445	599.936
std	1232.498479	1.695910	2.806840e+06	9.043363e+06	5.709187	172.430
min	1.000000	0.000000	2.000000e+05	3.000000e+05	2.000000	300.000
25%	1068.000000	1.000000	2.700000e+06	7.700000e+06	6.000000	453.000
50%	2135.000000	3.000000	5.100000e+06	1.450000e+07	10.000000	600.000
75%	3202.000000	4.000000	7.500000e+06	2.150000e+07	16.000000	748.000
max	4269.000000	5.000000	9.900000e+06	3.950000e+07	20.000000	900.000

In [8]:
```
# Range
range_values = df.max(numeric_only=True) - df.min(numeric_only=True)
print("Range for each column:\n", range_values)

# IQR
Q1 = df.quantile(0.25, numeric_only=True)
Q3 = df.quantile(0.75, numeric_only=True)
IQR = Q3 - Q1
print("\nIQR for each column:\n", IQR)

```

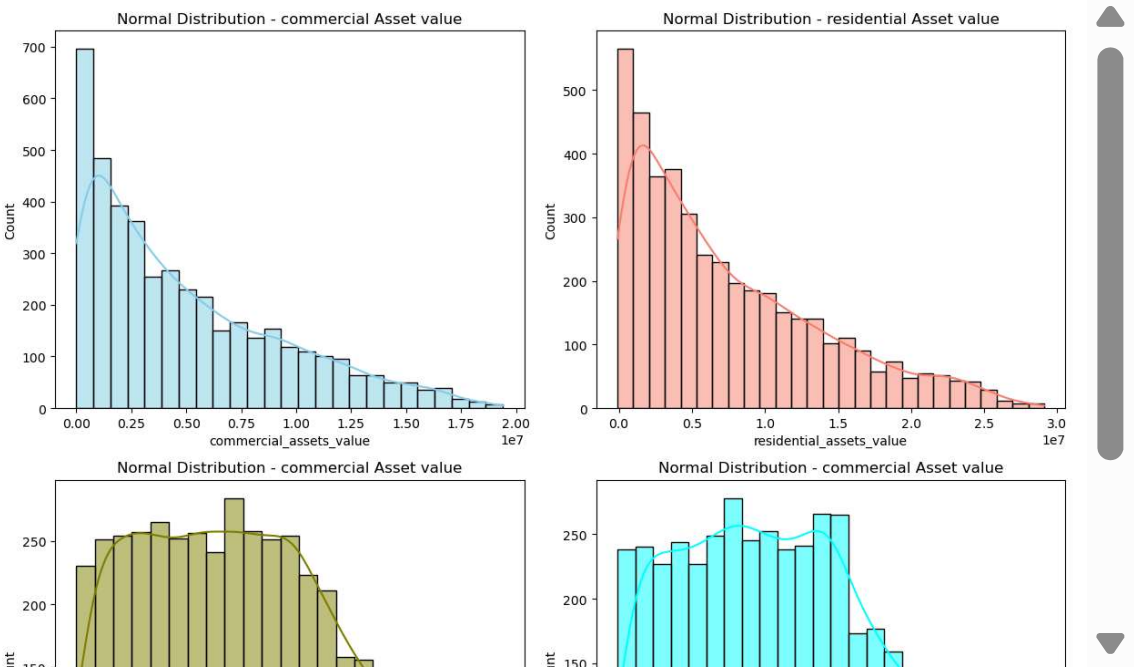
```
Range for each column:
 loan_id                         4268
no_of_dependents                    5
income_annum                  9700000
loan_amount                  39200000
loan_term                          18
cibil_score                       600
residential_assets_value     29200000
commercial_assets_value      19400000
luxury_assets_value          38900000
bank_asset_value             14700000
dtype: int64

IQR for each column:
 loan_id                       2134.0
no_of_dependents                 3.0
income_annum                4800000.0
loan_amount                13800000.0
loan_term                       10.0
cibil_score                    295.0
residential_assets_value    9100000.0
commercial_assets_value     6300000.0
luxury_assets_value        14200000.0
bank_asset_value            4800000.0
dtype: float64
```

In [9]:
```
df.var(numeric_only = True)
```

Out[9]:
```
loan_id                     1.519052e+06
no_of_dependents            2.876111e+00
income_annum                7.878350e+12
loan_amount                 8.178241e+13
loan_term                   3.259482e+01
cibil_score                 2.973224e+04
residential_assets_value    4.229729e+13
commercial_assets_value     1.926302e+13
luxury_assets_value         8.287833e+13
bank_asset_value            1.056370e+13
dtype: float64
```

In [10]:

```
plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1)
sns.histplot(df["commercial_assets_value"], kde=True, color='skyblue')
plt.title("Normal Distribution - commercial Asset value")

# Plot normal distribution of Commercial Asset Value
plt.subplot(2, 2, 2)
sns.histplot(df["residential_assets_value"], kde=True, color='salmon')
plt.title("Normal Distribution - residential Asset value")

plt.subplot(2, 2, 3)
sns.histplot(df["luxury_assets_value"], kde=True, color='olive')
plt.title("Normal Distribution - commercial Asset value")

plt.subplot(2, 2, 4)
sns.histplot(df["loan_amount"], kde=True, color='cyan')
plt.title("Normal Distribution - commercial Asset value")
plt.tight_layout()
plt.show()

```

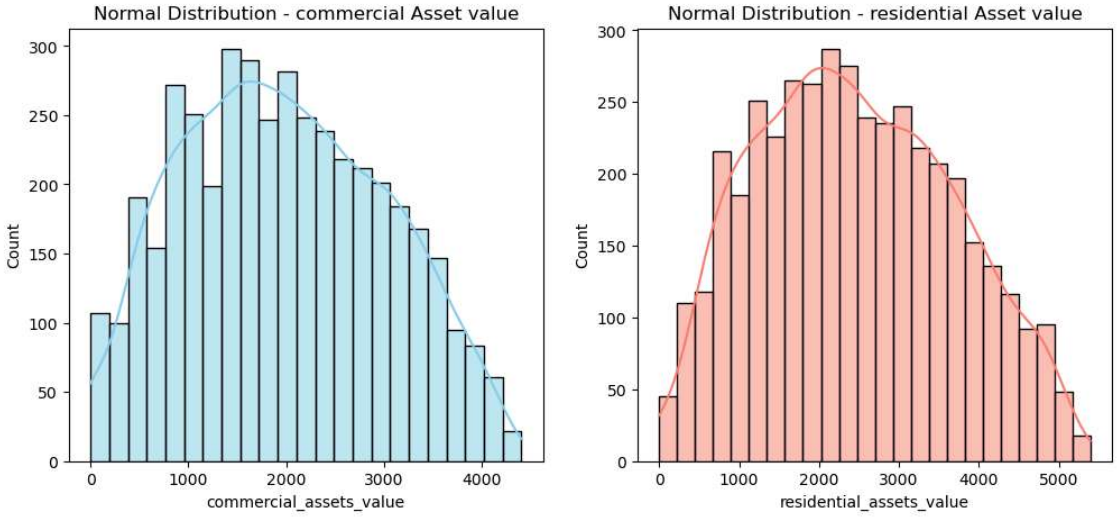


- As we can see that the "commercial assets value" & "residential assets value" plot is positively skewed so we need to transform the data using either log1p() or sqrt()

In [11]:
```
#normally distributed plot

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.histplot(np.sqrt(df["commercial_assets_value"]), kde=True, color=
plt.title("Normal Distribution - commercial Asset value")

# Plot normal distribution of Commercial Asset Value
plt.subplot(1, 2, 2)
sns.histplot(np.sqrt(df["residential_assets_value"]), kde=True, color=
plt.title("Normal Distribution - residential Asset value")


```

```
C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\arraylike.py:396:
RuntimeWarning: invalid value encountered in sqrt
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[11]: Text(0.5, 1.0, 'Normal Distribution - residential Asset value')



In [12]:
```
df.isnull().sum()
```

Out[12]:
```
loan_id                   0
no_of_dependents          0
education                 0
self_employed             0
income_annum              0
loan_amount               0
loan_term                 0
cibil_score               0
residential_assets_value  0
commercial_assets_value   0
luxury_assets_value       0
bank_asset_value          0
loan_status               0
dtype: int64
```

- There is no null values present in the dataset

In [13]:
```
df.duplicated().sum()
```

Out[13]:
```
0
```

- There is no duplicacy available in the dataset

Inferential statistics

In [14]:
```
#chi-square test and the P-value
education_crosstab = pd.crosstab(df['education'], df['loan_status'])
chi2_edu, p_edu, _, _ = stats.chi2_contingency(education_crosstab)
print(f"chi2:{chi2_edu},p-value:{p_edu}")
```

```
chi2:0.08395754138250573,p-value:0.7720042291016309
```

- The values of (chi2 or p-value) > 0.05, hence we come on to the conclusion that there is no significant relationship between education and the loan approval

In [15]:
```
print(education_crosstab)
```

```
loan_status     Approved   Rejected
education
 Graduate           1339        805
 Not Graduate       1317        808
```

In [16]:
```
#confidence Interval
loan_amount_ci = sms.DescrStatsW(df['loan_amount']).tconfint_mean()
print(f"loan_amount_ci",(loan_amount_ci))
```

```
loan_amount_ci (14862095.252453592, 15404805.661109302)
```

- This confidence Innterval suggests that the average loan_amount is in the range of 14.86M and 15.40M.

In [17]:
```
#skewness
df.skew(numeric_only = True)
```

Out[17]:
```
loan_id                    0.000000
no_of_dependents          -0.017971
income_annum              -0.012814
loan_amount                0.308724
loan_term                  0.036359
cibil_score               -0.009039
residential_assets_value   0.978451
commercial_assets_value    0.957791
luxury_assets_value        0.322208
bank_asset_value           0.560725
dtype: float64
```

- 0 Perfectly symmetrical (normal)
- 0.5 to 1 Moderate positive skew

- 1 Highly positively skewed

- -0.5 to -1 Moderate negative skew
- <-1 Highly negatively skewed

In [18]:
```
#Kurtosis
df.kurt(numeric_only = True)
```

Out[18]:
```
loan_id                    -1.200000
no_of_dependents           -1.256992
income_annum               -1.182729
loan_amount                -0.743680
loan_term                  -1.220853
cibil_score                -1.185670
residential_assets_value    0.184738
commercial_assets_value     0.100813
luxury_assets_value        -0.738056
bank_asset_value           -0.397277
dtype: float64
```

- = 0 (excess) Mesokurtic

- 0 Leptokurtic

- < 0 Platykurtic

Data Visualisation

In [19]:
```
numeric_df = df.select_dtypes(include='number')

numeric_df
```

Out[19]:

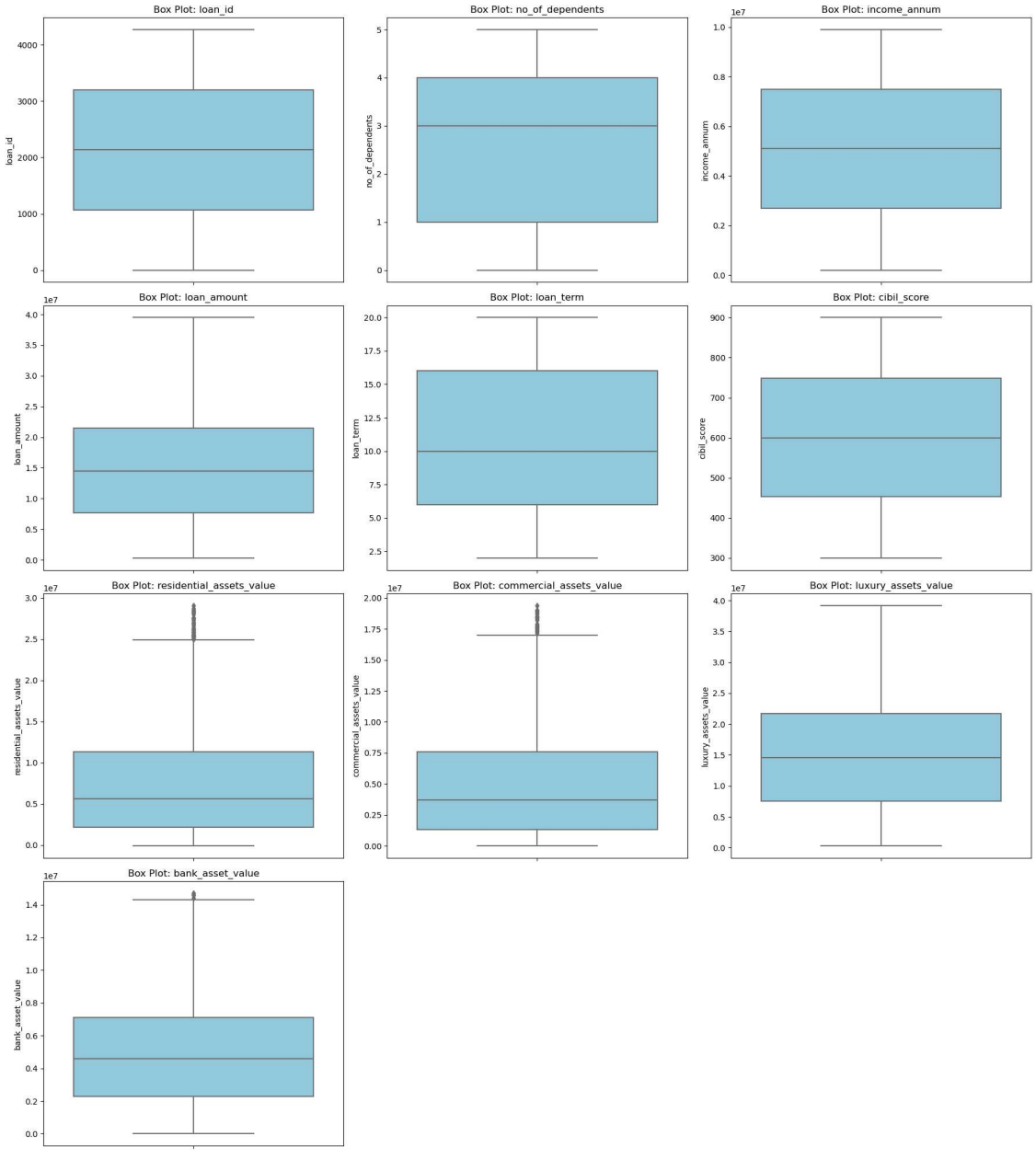
	loan_id	no_of_dependents	income_annum	loan_amount	loan_term	cibil_score	resi
0	1	2	9600000	29900000	12	778	
1	2	0	4100000	12200000	8	417	
2	3	3	9100000	29700000	20	506	
3	4	3	8200000	30700000	8	467	
4	5	5	9800000	24200000	20	382	
...	...	...	...	...	...	...	...
4264	4265	5	1000000	2300000	12	317	
4265	4266	0	3300000	11300000	20	559	
4266	4267	2	6500000	23900000	18	457	
4267	4268	1	4100000	12800000	8	780	
4268	4269	1	9200000	29700000	10	607	

4269 rows × 10 columns

In [20]:
```
# Number of numeric columns
num_cols = numeric_df.shape[1]
num_cols
```

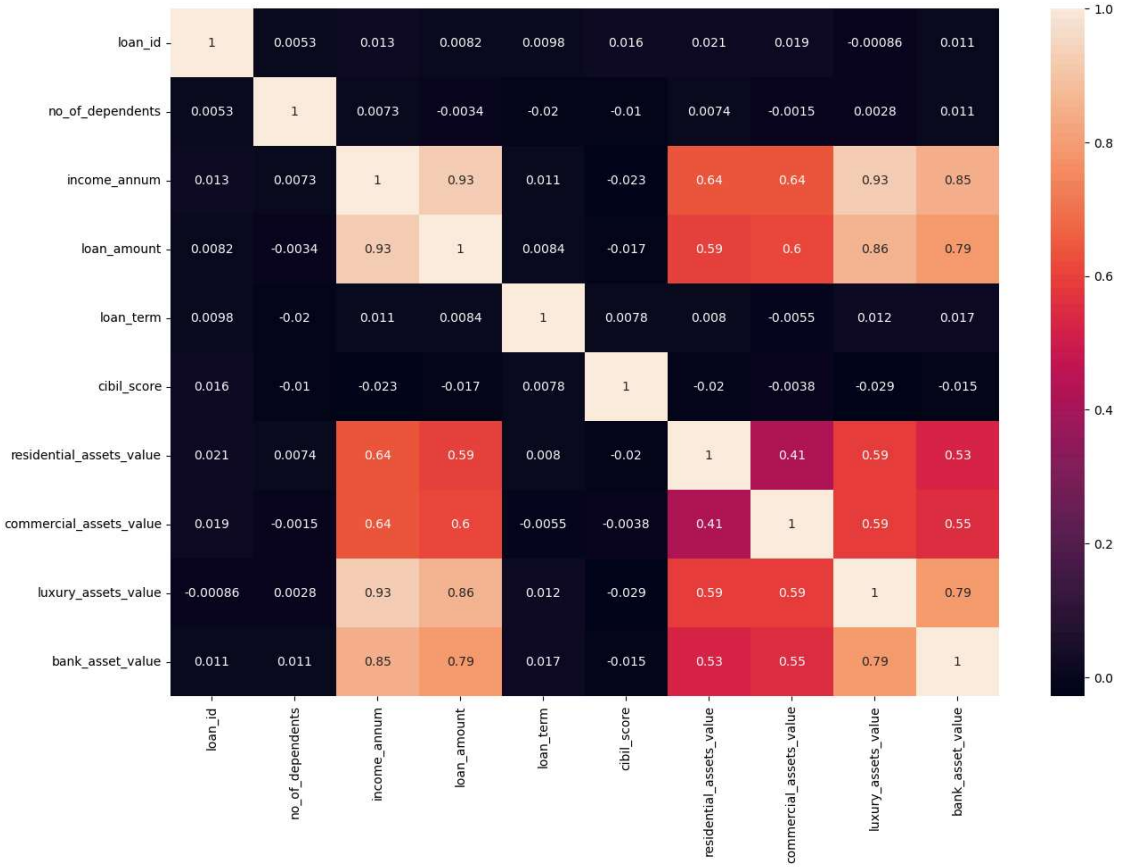
Out[20]: 10

In [21]:
```
import warnings
warnings.filterwarnings("ignore")
cols_per_row = 3
rows = (num_cols + cols_per_row - 1) // cols_per_row

# Create subplots
plt.figure(figsize=(18, 5 * rows))
for idx, column in enumerate(numeric_df.columns, 1):
    plt.subplot(rows, cols_per_row, idx)
    sns.boxplot(y=numeric_df[column], color="skyblue")
    plt.title(f'Box Plot: {column}')
    plt.tight_layout()

plt.show()
```



In [22]:
```
plt.figure(figsize=(15,10))
sns.heatmap(numeric_df.corr(),annot=True)
```

Out[22]: <Axes: >



Key points

- income_annum is highly correlated with the loan_amount(93%) which means any person can only able to issue the loan based on his capacity of earning.

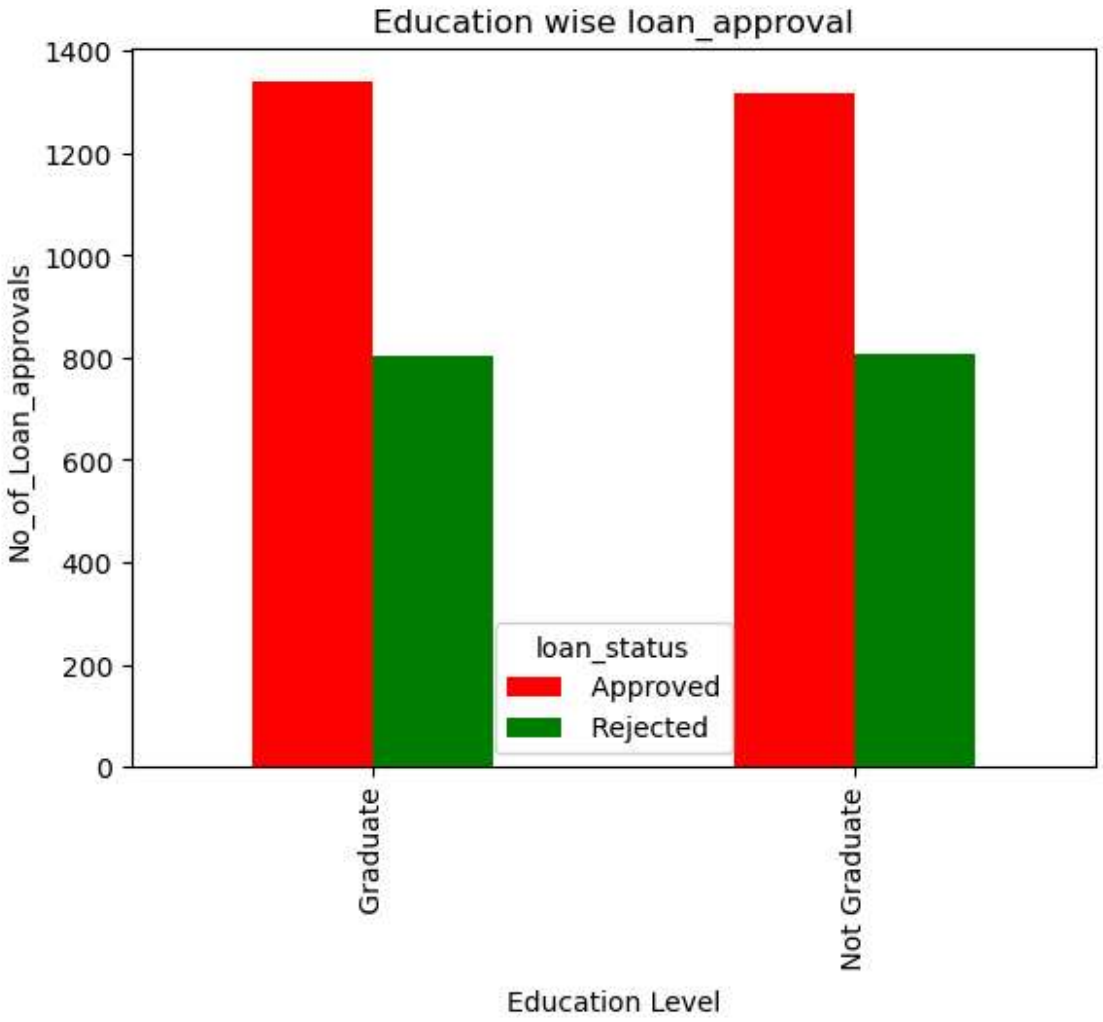
In [23]:
```
edu_wise_approval = df.groupby(["education", "loan_status"]).size().ur
edu_wise_approval.head()
```

Out[23]:

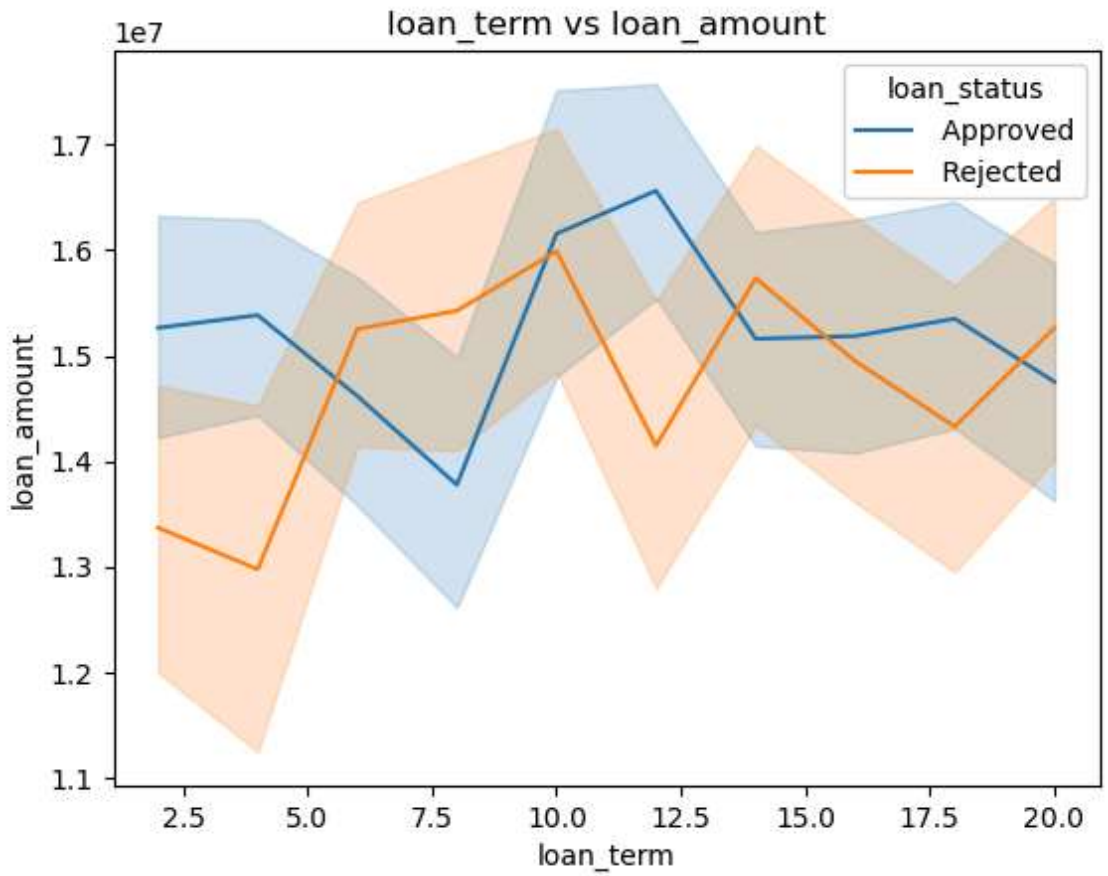
loan_status	Approved	Rejected
education		
Graduate	1339	805
Not Graduate	1317	808

In [24]:

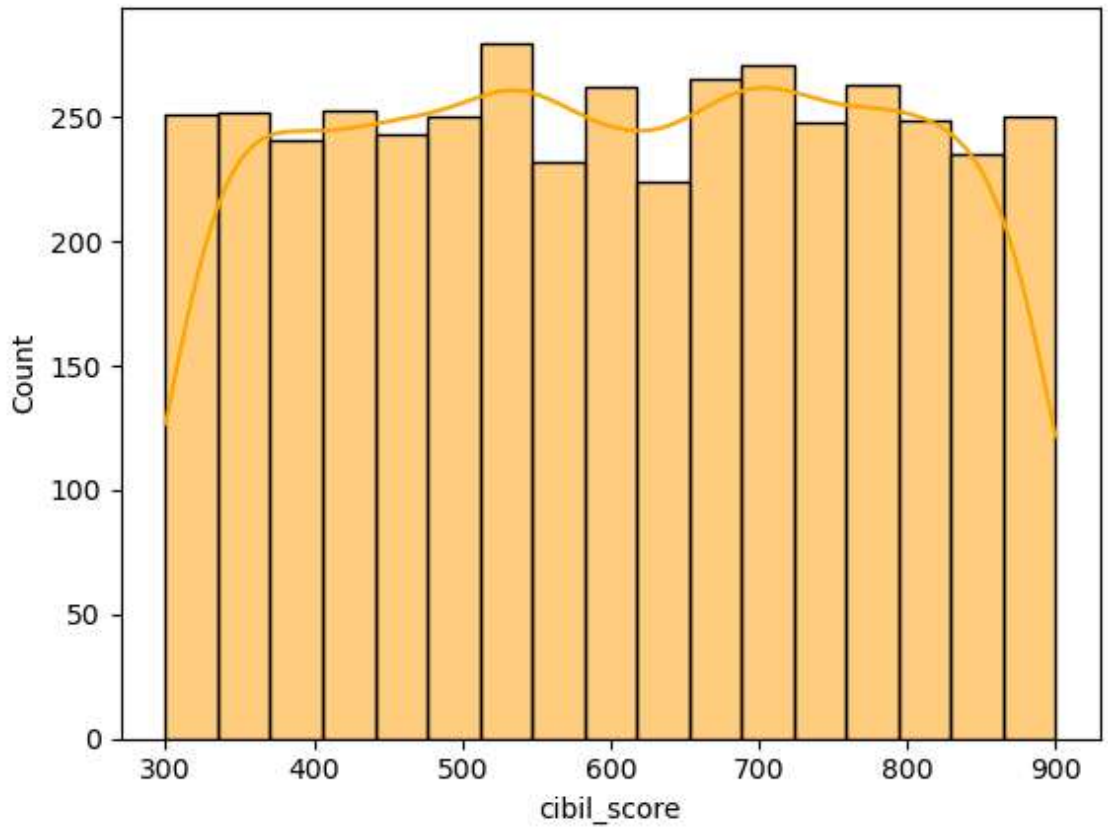
```
edu_wise_approval.plot(kind="bar",color=["red","green"])
plt.xlabel("Education Level")
plt.ylabel("No_of_Loan_approvals")
plt.title("Education wise loan_approval")
plt.show()
```



In [25]:
```
sns.lineplot(x=df['loan_term'],y=df['loan_amount'],hue= df["loan_statu
plt.show()
```



In [26]:
```
sns.histplot(data = df,x= df["cibil_score"],kde = True,color="orange"
plt.show()
```

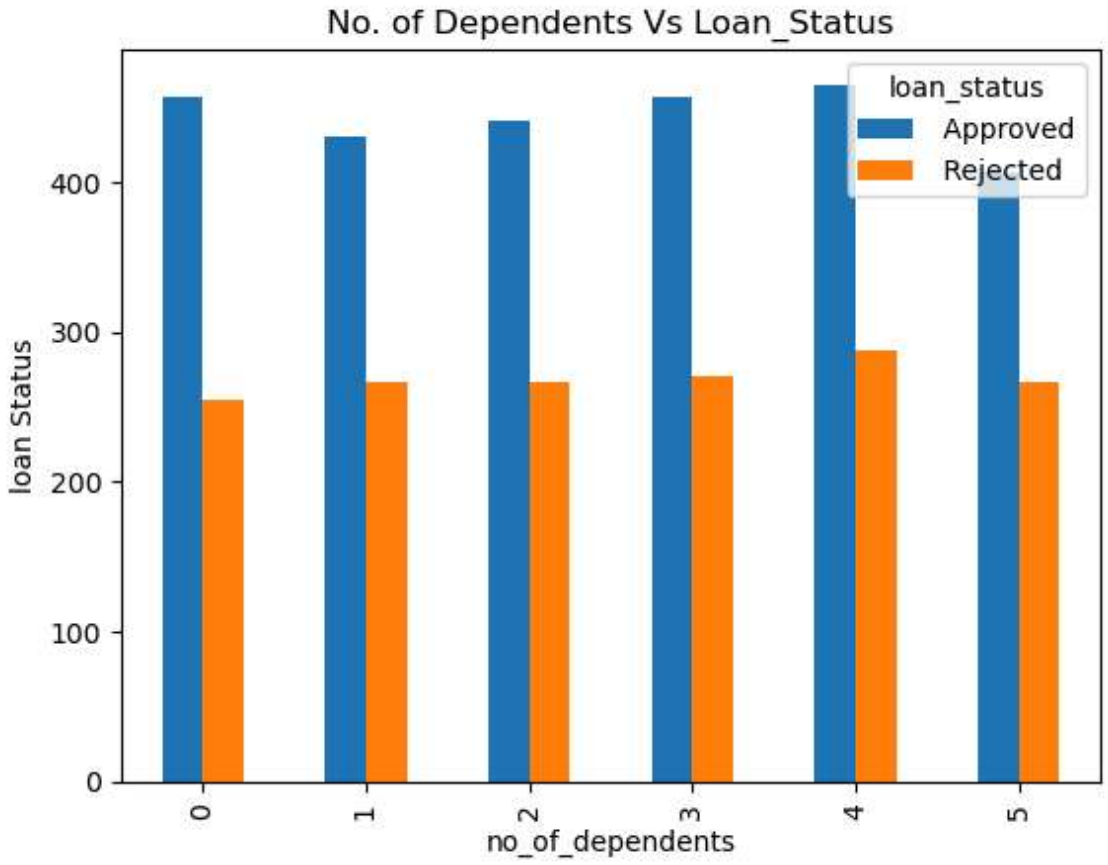


In [27]:
```
dependent_wise_approval = df.groupby(["no_of_dependents", "loan_status
dependent_wise_approval.head()
```

Out[27]:

no_of_dependents	loan_status Approved	Rejected
0	457	255
1	430	267
2	441	267
3	457	270
4	465	287

In [28]:
```
dependent_wise_approval.plot(kind="bar").set_title("No. of Dependents
plt.ylabel("loan Status")
plt.show()
```



In [29]:
```
1 df.head()
```

Out[29]:

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loar
0	1	2	Graduate	No	9600000	29900000	
1	2	0	Not Graduate	Yes	4100000	12200000	
2	3	3	Graduate	No	9100000	29700000	
3	4	3	Graduate	No	8200000	30700000	
4	5	5	Not Graduate	Yes	9800000	24200000	

Data Preprocessing

In [30]:
```
1 new_df = df.drop(["loan_id","bank_asset_value"],axis = 1)
2 new_df.head()
```

Out[30]:

	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	c
0	2	Graduate	No	9600000	29900000	12	
1	0	Not Graduate	Yes	4100000	12200000	8	
2	3	Graduate	No	9100000	29700000	20	
3	3	Graduate	No	8200000	30700000	8	
4	5	Not Graduate	Yes	9800000	24200000	20	

In [31]:
```
1 encoder = LabelEncoder()
2 new_df["education"] = encoder.fit_transform(df["education"])
3 new_df["self_employed"] = encoder.fit_transform(df["self_employed"])
4 new_df["loan_status"] = encoder.fit_transform(df["loan_status"])
5 new_df.head()
```

Out[31]:

	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	c
0	2	0	0	9600000	29900000	12	
1	0	1	1	4100000	12200000	8	
2	3	0	0	9100000	29700000	20	
3	3	0	0	8200000	30700000	8	
4	5	1	1	9800000	24200000	20	

Model Building

In [32]:
```
1 X_train,X_test,y_train,y_test = train_test_split(new_df.drop("loan_sta
```

In [33]:
```
1 dt_clf = DecisionTreeClassifier()
2 dt_clf.fit(X_train,y_train)
```

Out[33]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [34]:
```
1 X_train,X_test,y_train,y_test = train_test_split(new_df.drop("loan_sta
```

In [35]:
```
1 dt_clf.fit(X_train,y_train)
```

Out[35]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [36]:
```
1 y_pred = dt_clf.predict(X_test)
```

In [37]:
```
1 rf_clf = RandomForestClassifier()
2 rf_clf.fit(X_train,y_train)
```

Out[37]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
1 y_pred1 = rf_clf.predict(X_test)
```

Model Evaluation

In [39]:

```
print(f"Accuracy score of Decision Tree Classifier: {accuracy_score(y_
print(f"Accuracy score of Random Forest Classifier: {accuracy_score(y_
print("Classification Report of Decision Tree Classifier\n")
print(classification_report(y_test,y_pred))
print("Classification Report of Random Forest Classifier\n")
print(classification_report(y_test,y_pred1))
```

```
Accuracy score of Decision Tree Classifier: 0.9765807962529274
Accuracy score of Random Forest Classifier: 0.9789227166276346
Classification Report of Decision Tree Classifier

              precision    recall  f1-score   support

           0       0.98      0.98      0.98       536
           1       0.97      0.97      0.97       318

    accuracy                           0.98       854
   macro avg       0.98      0.97      0.97       854
weighted avg       0.98      0.98      0.98       854

Classification Report of Random Forest Classifier

              precision    recall  f1-score   support

           0       0.98      0.99      0.98       536
           1       0.98      0.96      0.97       318

    accuracy                           0.98       854
   macro avg       0.98      0.98      0.98       854
weighted avg       0.98      0.98      0.98       854
```

FINAL LOAN APPROVAL PREDICTOR

In [40]:

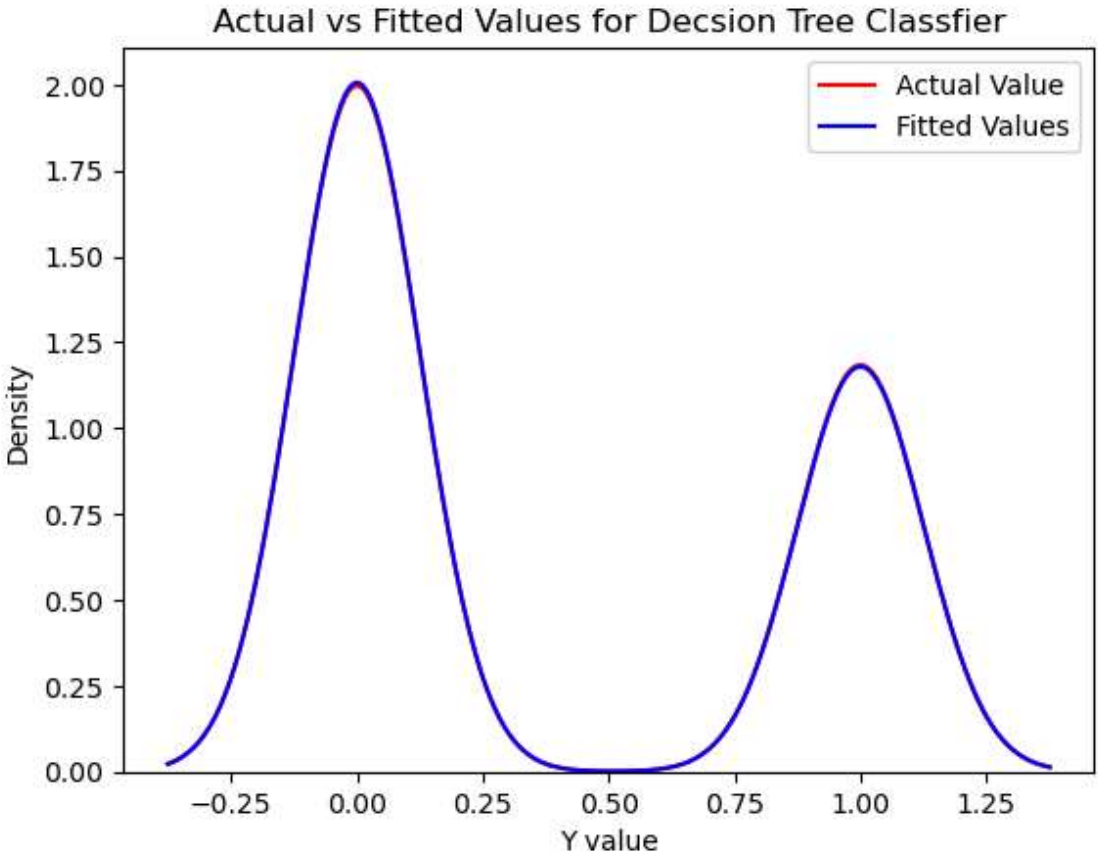
```
no_of_dependents = int(input("Enter the No. of Dependents:"))
education = int(input("Enter the education (Graduate-0,Not Graduate-1)
self_employed = int(input("Enter the Empployment status (self-employed
income_annum = int(input("Enter the Income per Annum (without using ru
loan_amount = int(input("Enter the Loan amount (without using rupees/d
loan_term = int(input("Enter the Term of loan (without mentioning mon,
cibil_score = int(input("Enter the CIBIL SCORE:"))
residential_assets_value = int(input("Enter the Residential Assets Val
commercial_assets_value = int(input("Enter the Commercial Assets Value
luxury_assets_value = int(input("Enter the Luxury_assets_value (withou

pred_val = rf_clf.predict([[no_of_dependents,education,self_employed,i

if pred_val == 0:
    print("\n\033[1m😍Congratulation!! Your Loan is Approved\033[0m")
else:
    print("\n\033[1m😔sorry!! Your Loan is not Approved,Please try la
```

```
Enter the No. of Dependents:2
Enter the education (Graduate-0,Not Graduate-1):0
Enter the Empployment status (self-employed-1,Not self Employed-0):0
Enter the Income per Annum (without using rupees/dollar symbol):340000
Enter the Loan amount (without using rupees/dollar symbol):1000000
Enter the Term of loan (without mentioning mon,yr):24
Enter the CIBIL SCORE:400
Enter the Residential Assets Value (without using rupees/dollar symbol):4
50000
Enter the Commercial Assets Value (without using rupees/dollar symbol):0
Enter the Luxury_assets_value (without using rupees/dollar symbol):0

😔sorry!! Your Loan is not Approved,Please try later!!
```

Actual vs Fitted plot of the Decision Tree Classifier

In [44]:

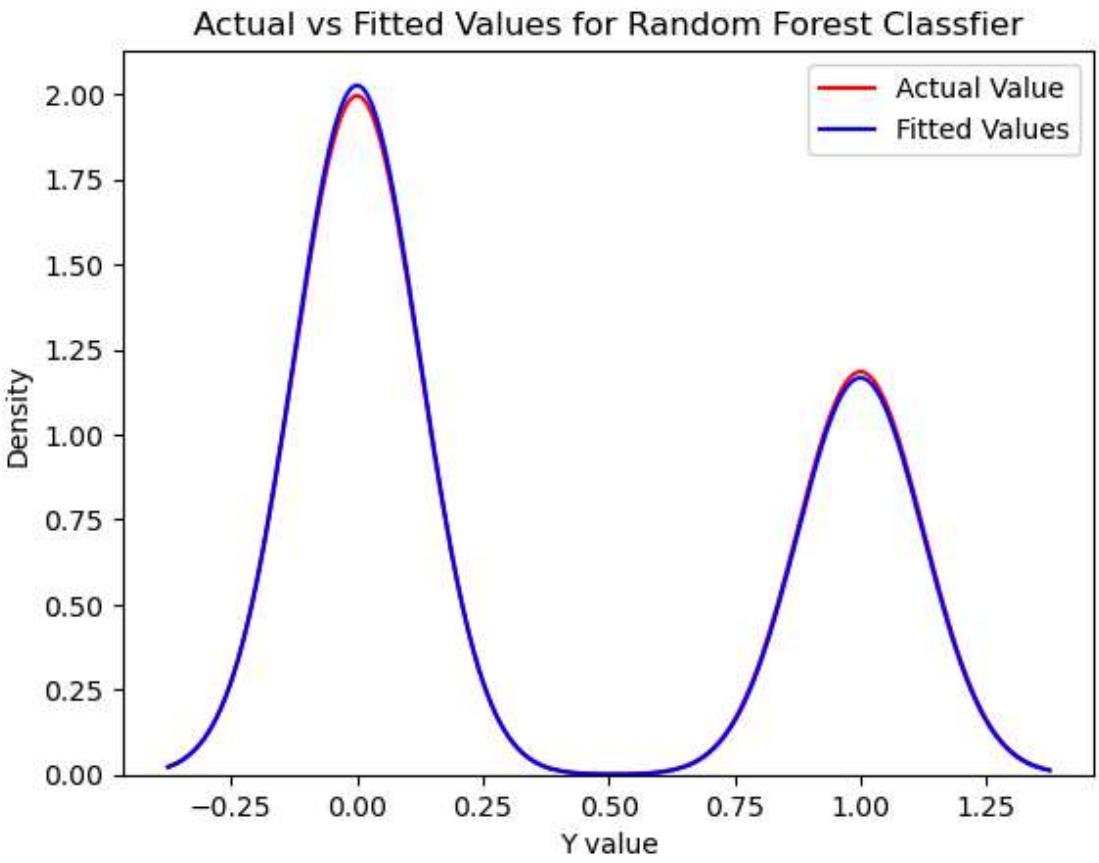
```
ax = sns.distplot( x = y_test, hist = False, color = "r", label = "Act
sns.distplot( x = y_pred, hist = False, color = "b", label = "Fitted V
plt.title('Actual vs Fitted Values for Decsion Tree Classfier')
plt.xlabel("Y value")
plt.legend()
plt.show()
```



Actual vs Fitted plot of the Random Forest Classifier

In [42]:

```
ax = sns.distplot( x = y_test, hist = False, color = "r", label = "Act
sns.distplot( x = y_pred1, hist = False, color = "b", label = "Fitted
plt.title('Actual vs Fitted Values for Random Forest Classfier')
plt.xlabel("Y value")
plt.legend()
plt.show()
```



Final Conclusion

- We have created a **Loan Approval Predictor** which predicts whether the loan will be approved or not based on various features.
- We have applied the **ML Pipelines** for creating this Predictor which consists of:

1)Data Collection

2)Data Cleaning & Statistical Analysis

3)Data Visualisation

4)Data Preprocessing

5)Data Modeling

6)Model Evaluation

7)Model Deployment

- We have used two ML Classification models that are **"Decision Tree Classfier"** and **Random Forest Classifier** which performs very well on our dataset.

Accuracy of Model

- Decision Tree Classifier: **97.42%**
- Random Forest Classifier: **98%**

Model Deployment

In [43]:
```
with open("model.pkl","wb") as file:
    pickle.dump(rf_clf,file)
```

In [ ]:
```

```