

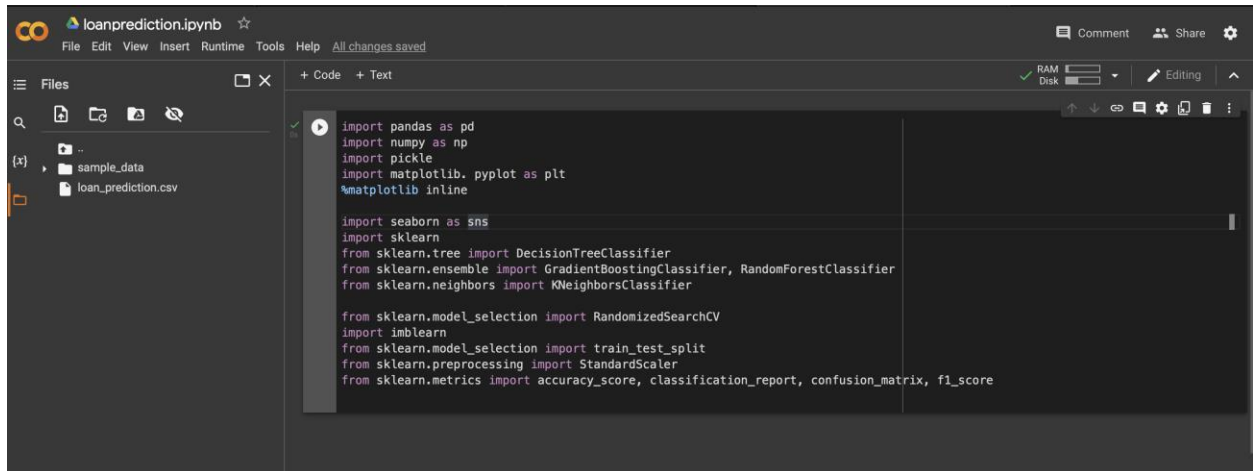
# VISUALISING AND ANALYSING THE DATA

## 1. Import libraries

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```



The image shows a Jupyter Notebook interface with the title 'loanprediction.ipynb'. The left sidebar displays a file explorer with a folder named 'sample\_data' containing a file 'loan\_prediction.csv'. The main code area contains the following Python code:

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline

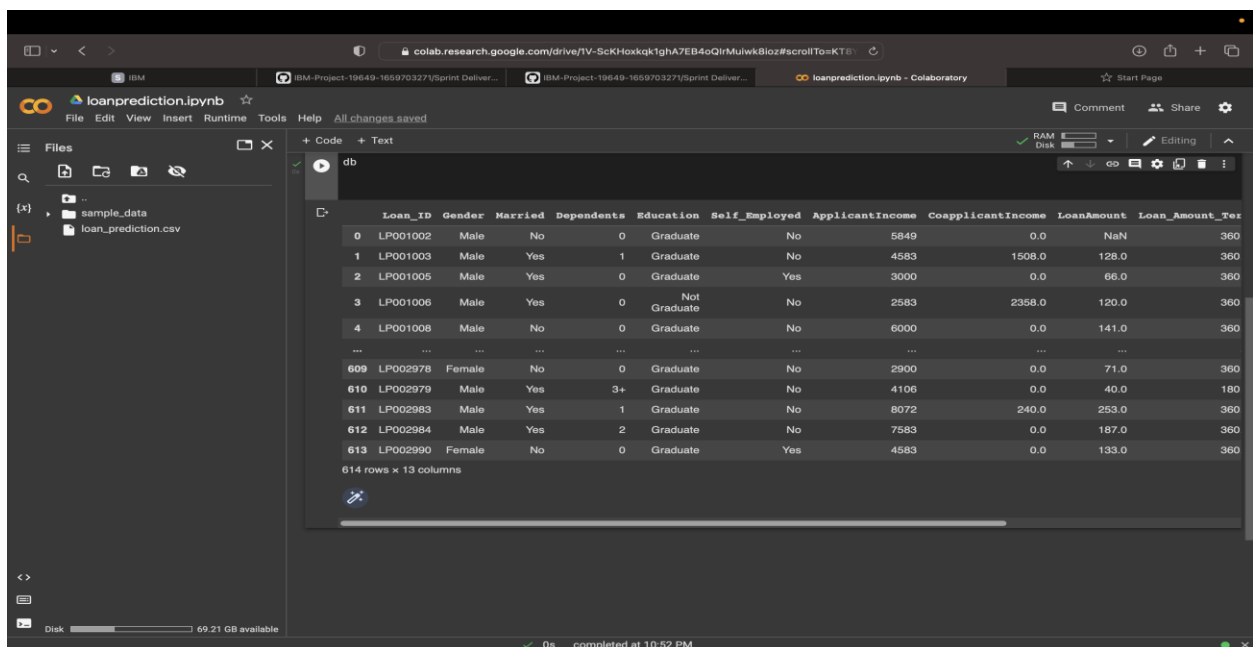
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

## 2. Reading the dataset

```
db = pd.read_csv('loan_prediction.csv')
```

```
db
```



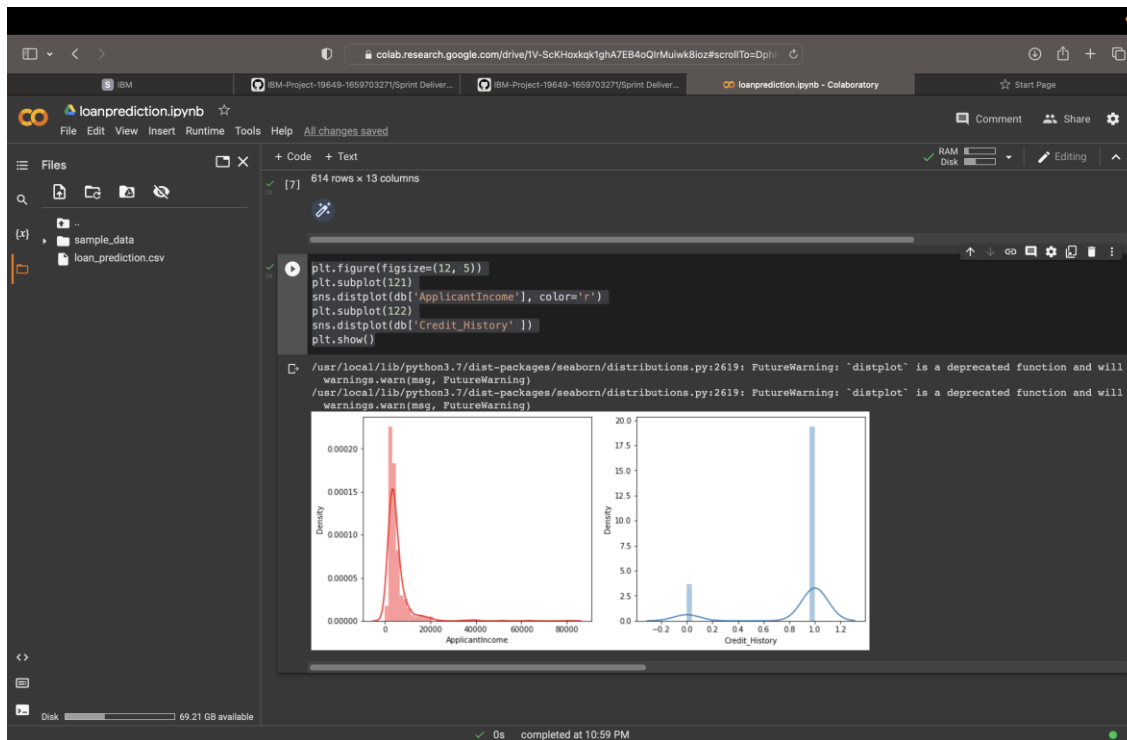
The image shows the same Jupyter Notebook interface, but now the variable 'db' is displayed as a DataFrame. The DataFrame has 614 rows and 13 columns. The columns are: Loan\_ID, Gender, Married, Dependents, Education, Self\_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount, and Loan\_Amount\_Ter. The first few rows of the DataFrame are shown, along with the total number of rows and columns.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Ter
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360

614 rows x 13 columns

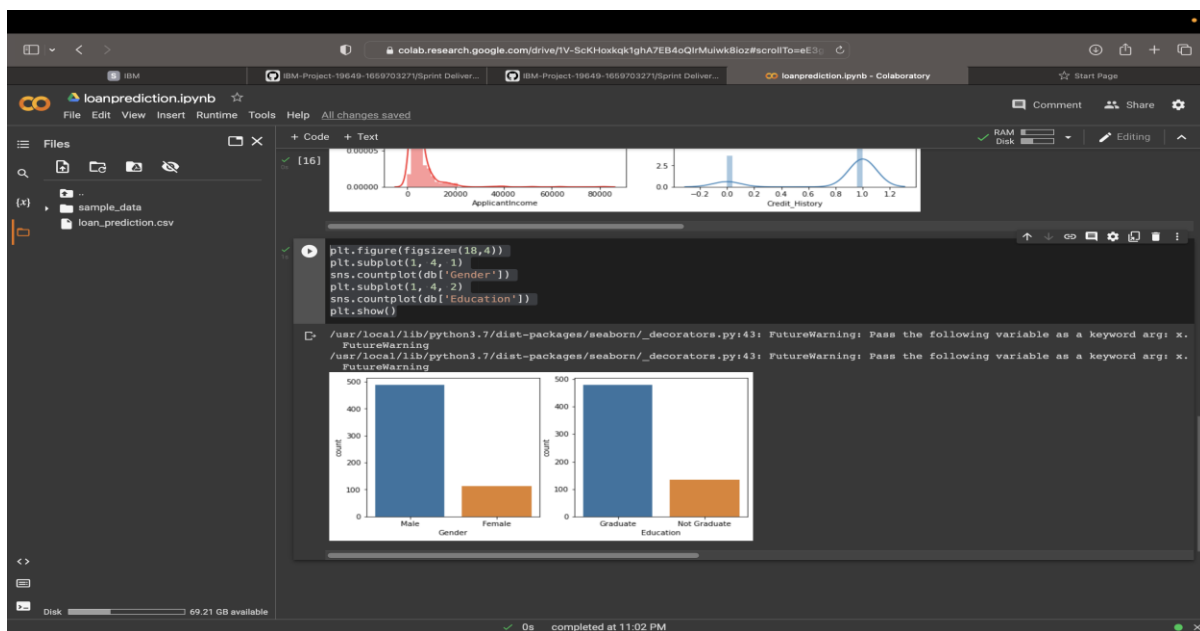
### 3. Uni-variate Analysis

```
plt.figure(figsize=(12, 5))  
plt.subplot(121)  
sns.distplot(db['ApplicantIncome'], color='r')  
plt.subplot(122)  
sns.distplot(db['Credit_History' ])  
plt.show()
```



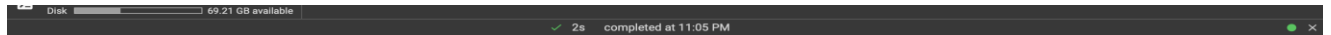
## 4. Bivariate Analysis

```
plt.figure(figsize=(18,4))  
plt.subplot(1, 4, 1)  
sns.countplot(db['Gender'])  
plt.subplot(1, 4, 2)  
sns.countplot(db['Education'])  
plt.show()
```



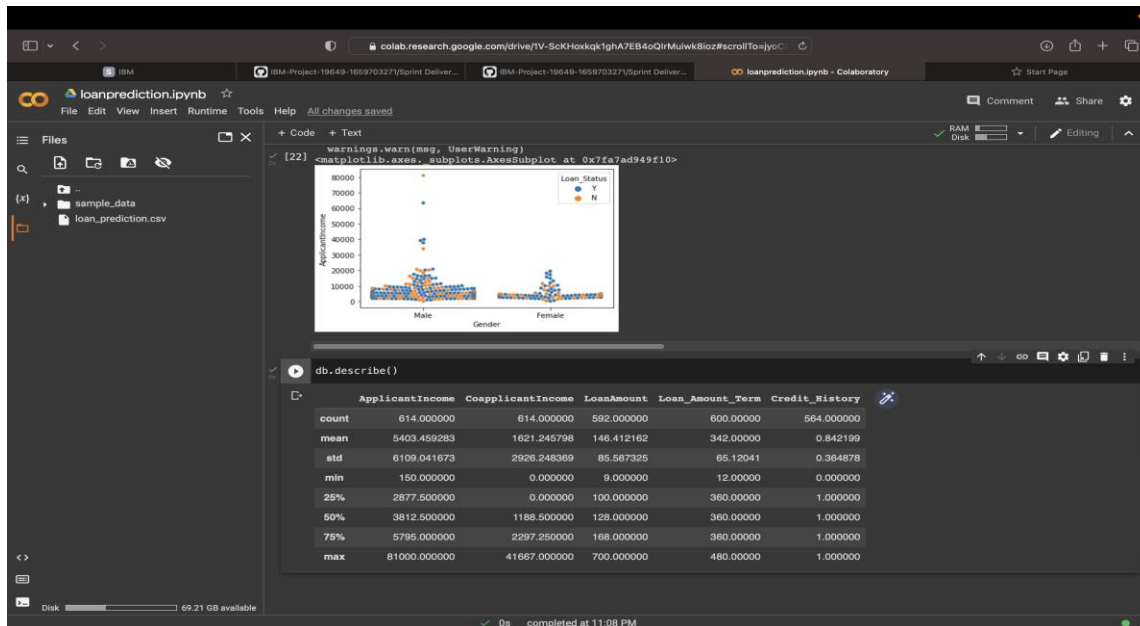
## 5. Multivariate Analysis

```
sns.swarmplot(db['Gender'], db['ApplicantIncome'], hue = db['Loan_Status'])
```



## 6. Descriptive Analysis

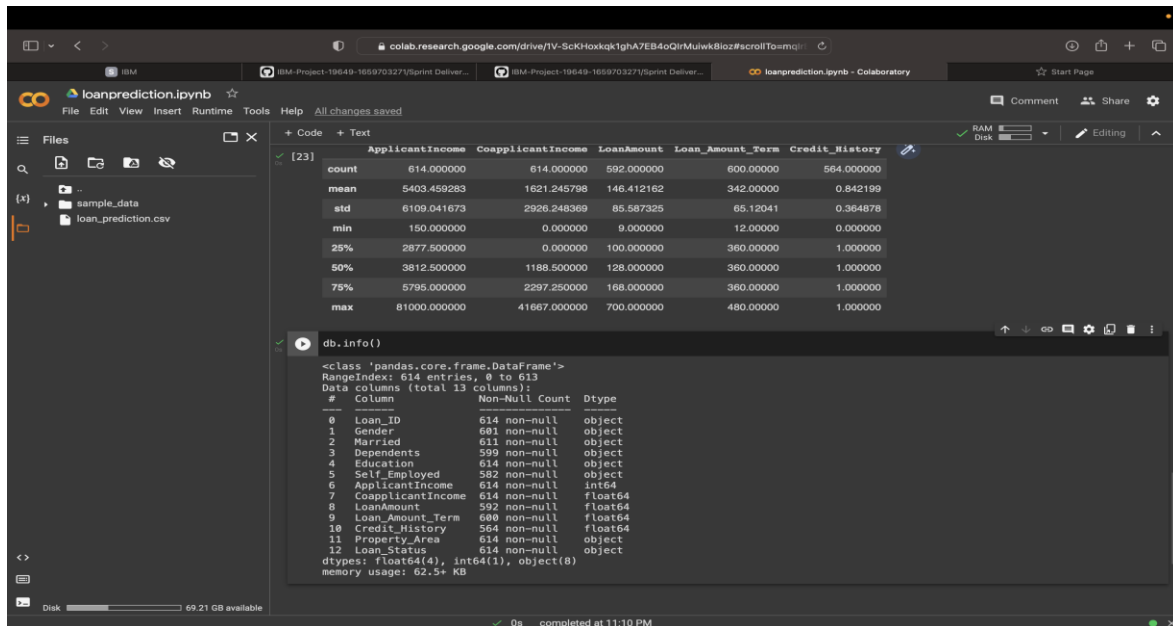
```
db.describe()
```



## Data Pre-processing

### 1. Checking for NULL values

db.info()



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

```
db.info()

<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
```

db.isnull().sum()

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: FutureWarning: The default value of regex will change from True to False in a future
0s completed at 11:49 PM
```

db['Gender'] = db['Gender'].fillna (db['Gender'].mode()[0])

db['Married'] = db['Married'].fillna(db['Married'].mode()[0])

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

db['Dependents'] = db['Dependents'].fillna(db['Dependents'].mode()[0])

db['Self\_Employed'] = db['Self\_Employed'].fillna(db['Self\_Employed'].mode()[0])

```
db['LoanAmount'] = db['LoanAmount'].fillna(db['LoanAmount'].mode()[0])
```

```
db['Loan_Amount_Term'] =  
db['Loan_Amount_Term'].fillna(db['Loan_Amount_Term'].mode()[0])
```

```
db['Credit_History'] = db['Credit_History'].fillna(db['Credit_History'].mode()[0])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: FutureWarning: The default value of  
regex will change from True to False in a future version. In addition, single character regular  
expressions will *not* be treated as literal strings when regex=True. """
```

## 2. Handling categorical Value

```
db['Gender']=db['Gender'].astype('int64')  
db['Married']=db['Married'].astype('int64')  
db['Dependents']=db['Dependents'].astype('int64')  
db['Self_Employed']=db['Self_Employed'].astype('int64')  
db['CoapplicantIncome']=db['CoapplicantIncome'].astype('int64')  
db['LoanAmount']=db['LoanAmount'].astype('int64')  
db['Loan_Amount_Term']=db['Loan_Amount_Term'].astype('int64')  
db['Credit_History']=db['Credit_History'].astype('int64')
```

## 3. BALANCING THE DATASET

```
from imblearn.combine import SMOTETomek  
smote = SMOTETomek(0.90)  
y = db['Loan_Status']  
x = db.drop(columns=['Loan_Status'],axis=1)
```

```
x_bal,y_bal = smote.fit_resample(x,y)
print(y.value_counts())
print(y_bal.value_counts())
```

**Y 422**

**N 192**

**Name: Loan\_Status, dtype: int64**

#### **4. SCALING THE DATA**

```
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
x_bal=pd.DataFrame(x_bal,columns=names)
```

#### **5. SPLITTING DATA INTO TRAIN AND TEST**

```
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
x_bal=pd.DataFrame(x_bal,columns=names)
X_train, X_test, y_train, y_test = train_test_split(
x_bal, y_bal, test_size=0.33, random_state=42)
```



