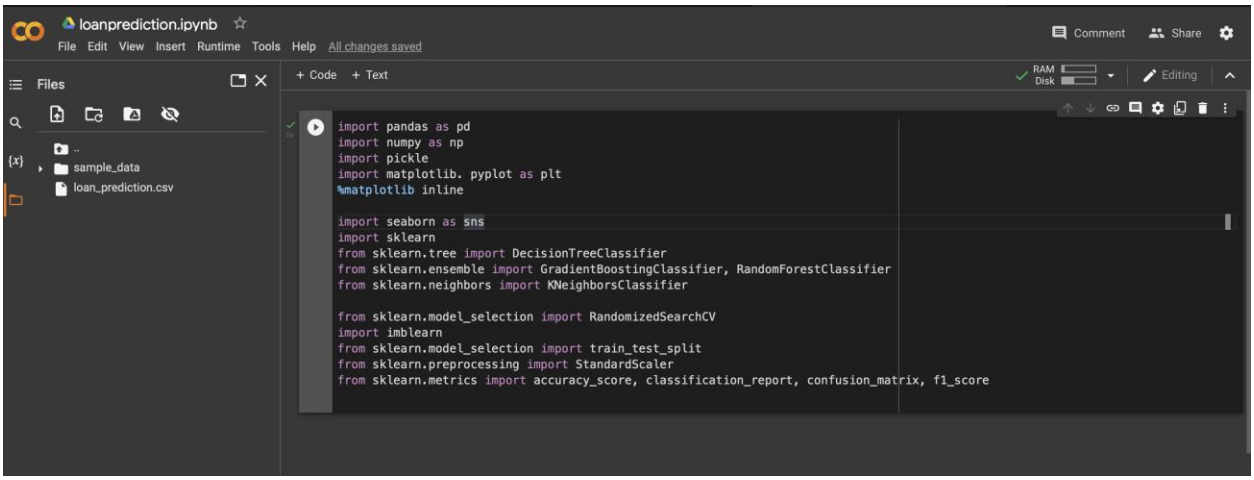
VISUALISING AND ANALYSING THE DATA

1. Import libraries

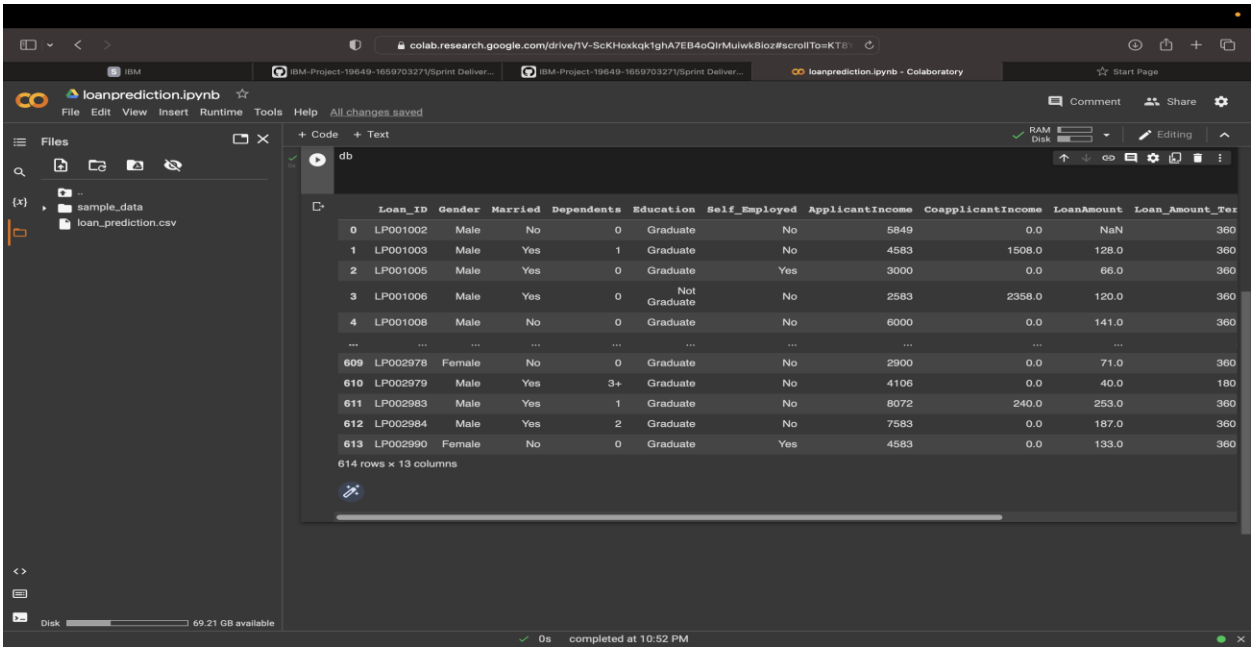
```
import pandas as pd
import numpy as np
import pickle
import matplotlib. pyplot as plt
%matplotlib inline


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
```



2. Reading the dataset

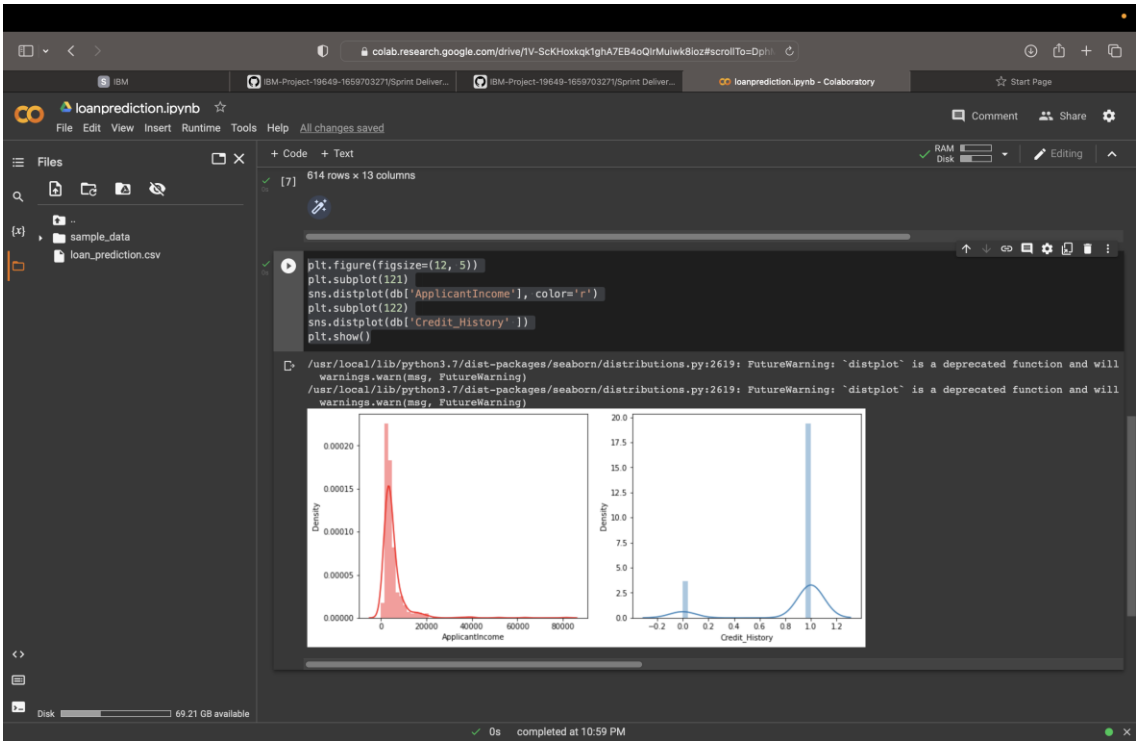
db = pd.read_csv('loan_prediction.csv')

db



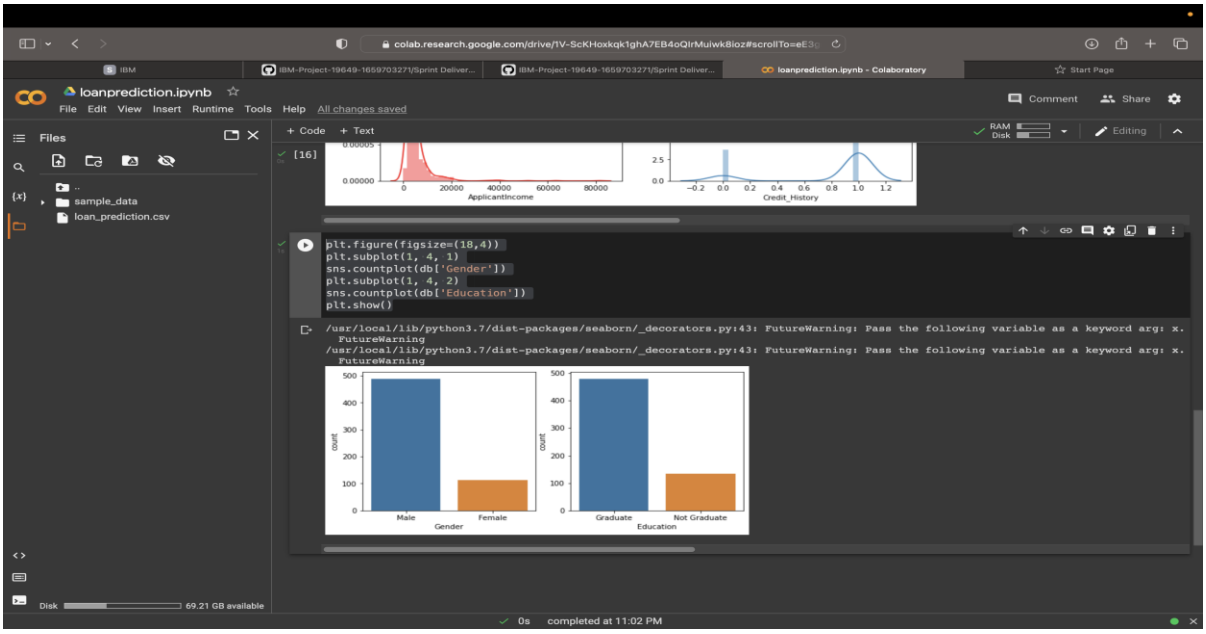
3. Uni-variate Analysis

```
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()
```

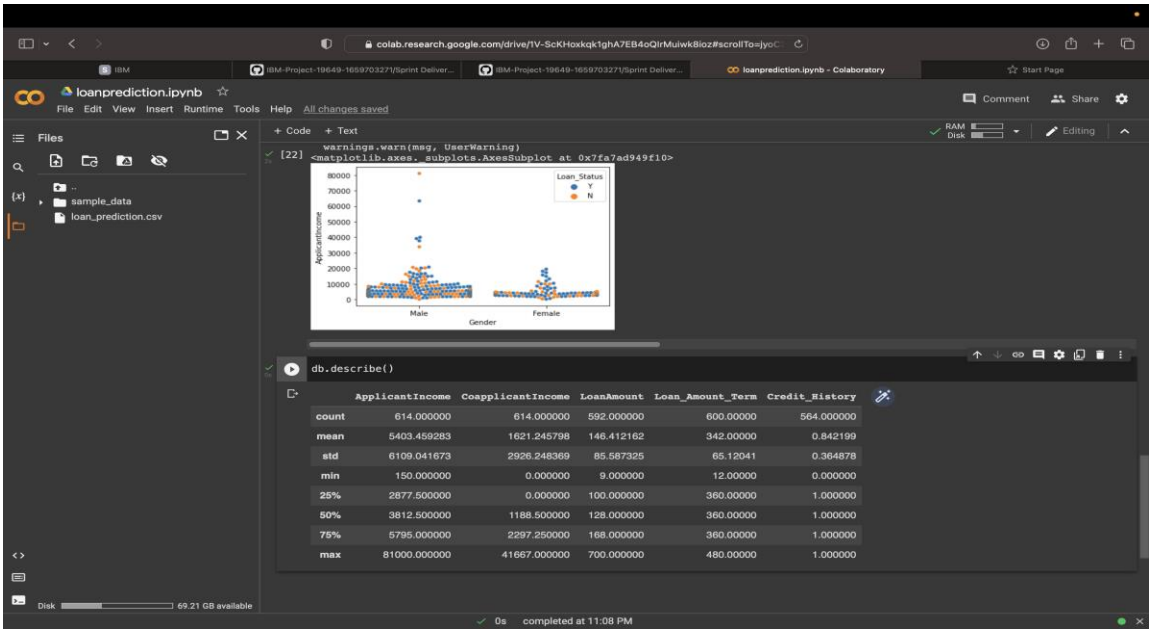


5. Multivariate Analysis

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

6. Descriptive Analysis

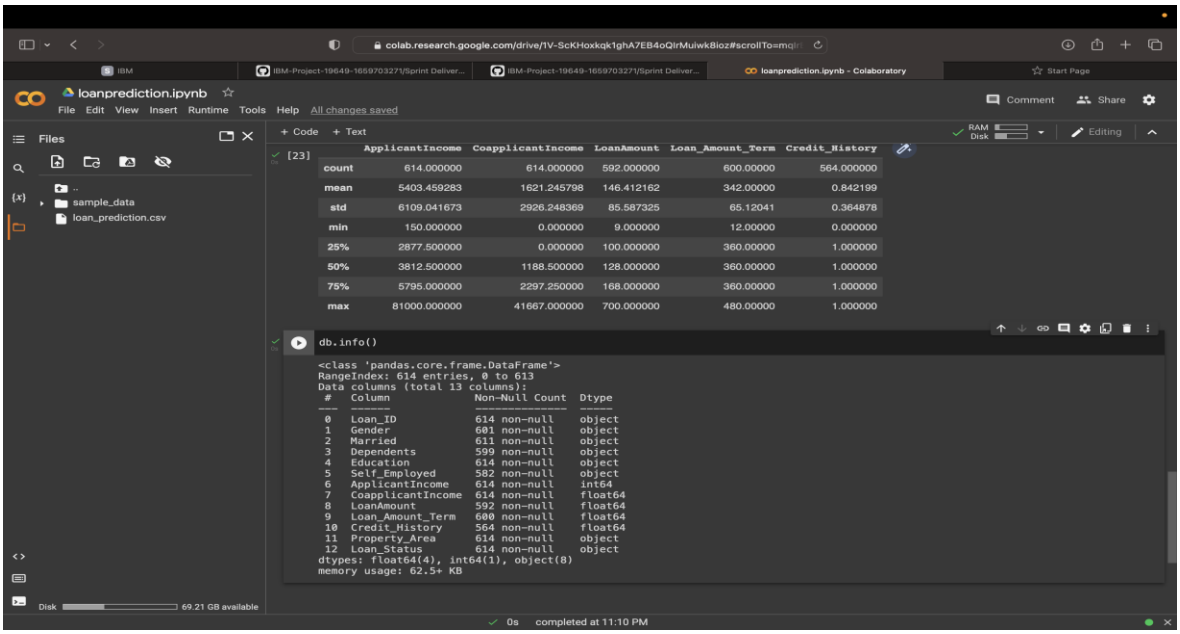
db.describe()



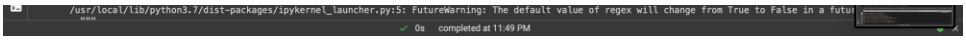
Data Pre-processing

1. Checking for NULL values

db.info()



db.isnull().sum()



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)
```