Dev Varma Rajasagi

Major Project

July Batch

Email - rsdvarma@gmail.com

Google Classroom - ML-MAJOR-JULY-ML-07-SPB4

Loan Sanction Using Python

PROJECT OVERVIEW

Through data extraction and data analytics and extracting the proper data we can see, highlight and understand the data better, while using said data for predictions and decision making. Data analytics techniques and algorithms are used by almost all businesses now to assist in their decision making process, it is especially used in commercial industries to understand the manufacturing and sales numbers. It is also used by the analysts and the experts to authenticate or negate experimental layouts, assumptions and conclusions.

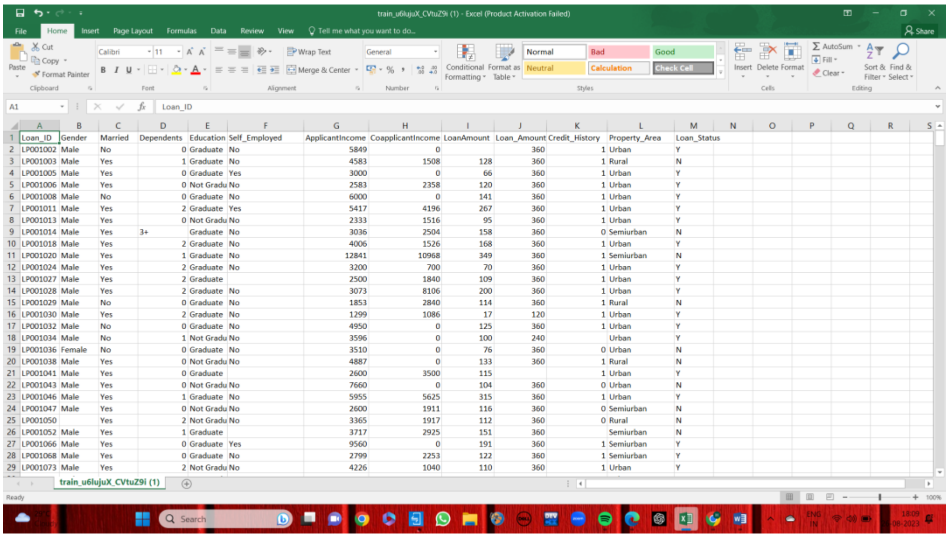
The "Loan Sanction Prediction" project aims to develop a machine learning model that predicts whether a loan application should be approved or rejected based on various features provided in the application. The project will utilize Python programming language along with popular machine learning libraries to pre-process the data, build and train the model, and evaluate its performance.

Project Steps

1. Import all the necessary libraries.
2. Import the dataset provided.
3. Understand the data.
4. Deal with the missing values if any.
5. Do some visualization if necessary.
6. Divide the dataset into training and test datasets.
7. Build the machine learning model whichever is suitable for the dataset.
8. Fit the model on the training dataset.
9. Test the model and find the accuracy of the model on the test and the training datasets.
10. Create a confusion matrix.
11. Evaluate and test the matrix and its effectiveness
12. Make the necessary changes.

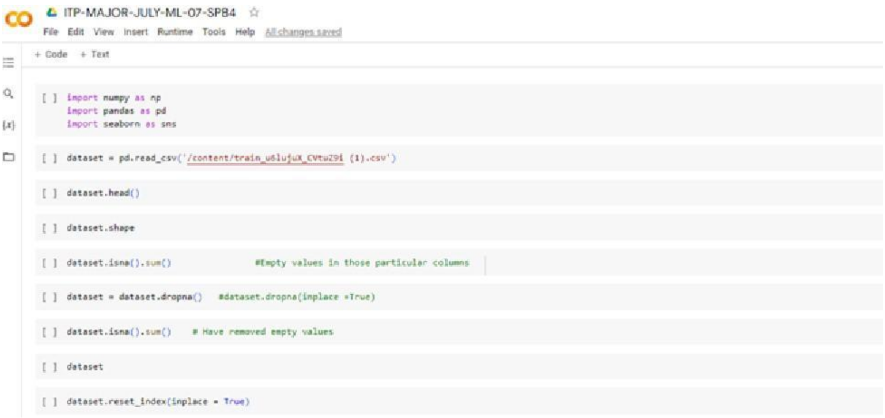
Code Implementation

Dataset

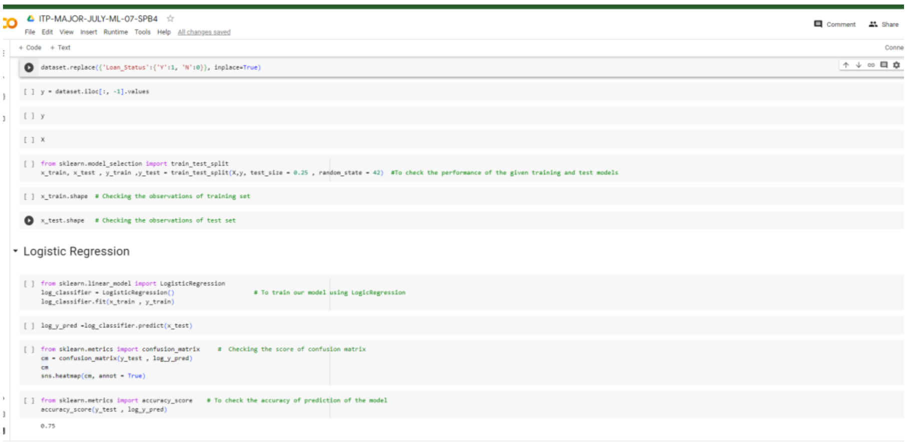


	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount	Credit_History	Property_Area	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
18	LP001034	Male	No	1	Not Gradu	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
20	LP001038	Male	Yes	0	Not Gradu	No	4887	0	133	360	1	Rural	N
21	LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y
22	LP001043	Male	Yes	0	Not Gradu	No	7660	0	104	360	0	Urban	N
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y
24	LP001047	Male	Yes	0	Not Gradu	No	2600	1911	116	360	0	Semiurban	N
25	LP001050		Yes	2	Not Gradu	No	3365	1917	112	360	0	Rural	N
26	LP001052	Male	Yes	1	Graduate		3717	2925	151	360		Semiurban	N
27	LP001066	Male	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurban	Y
28	LP001068	Male	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurban	Y
29	LP001073	Male	Yes	2	Not Gradu	No	4226	1040	110	360	1	Urban	Y

Importing the Necessary Libraries



```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sns

[ ] dataset = pd.read_csv('/content/train_u6lujuX_CVtuZ9i (1).csv')

[ ] dataset.head()

[ ] dataset.shape

[ ] dataset.isna().sum()              #Empty values in those particular columns

[ ] dataset = dataset.dropna()   #dataset.dropna(inplace =True)

[ ] dataset.isna().sum()    # Have removed empty values

[ ] dataset

[ ] dataset.reset_index(inplace = True)
```

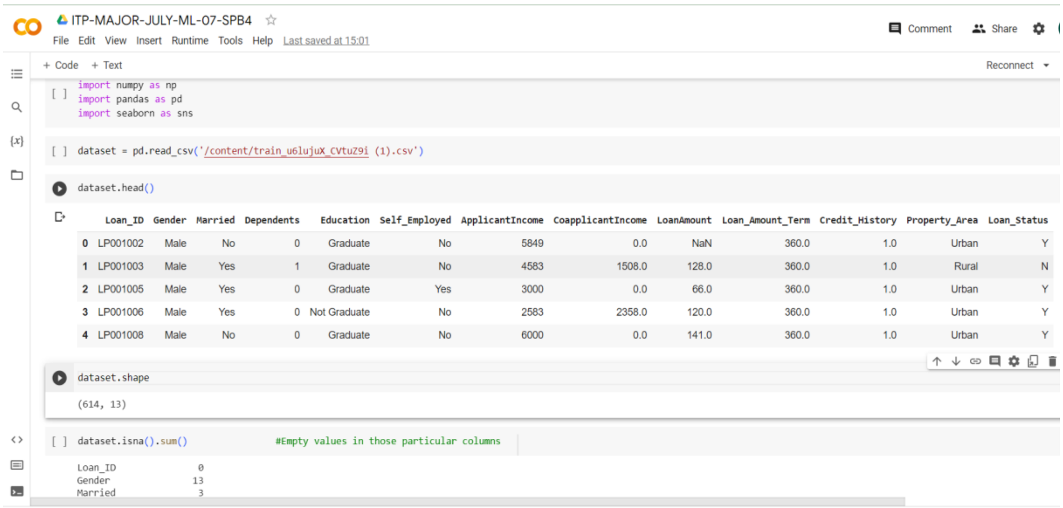


```
dataset.replace({'Loan_Status':{'Y':1, 'N':0}}, inplace=True)

[ ] y = dataset.iloc[:, -1].values

[ ] y

[ ] X

[ ] from sklearn.model_selection import train_test_split
    x_train, x_test , y_train ,y_test = train_test_split(X,y, test_size = 0.25 , random_state = 42)  #To check the performance of the given training and test models

[ ] x_train.shape  # Checking the observations of training set

    x_test.shape   # Checking the observations of test set
```

Logistic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
    log_classifier = LogisticRegression()                    # To train our model using LogicRegression
    log_classifier.fit(x_train , y_train)

[ ] log_y_pred =log_classifier.predict(x_test)

[ ] from sklearn.metrics import confusion_matrix     #  Checking the score of confusion matrix
    cm = confusion_matrix(y_test , log_y_pred)
    cm
    sns.heatmap(cm, annot = True)

[ ] from sklearn.metrics import accuracy_score    # To check the accuracy of prediction of the model
    accuracy_score(y_test , log_y_pred)

    0.75
```

Import the Dataset Provided


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help Last saved at 15:01
Comment
Share
+ Code + Text
Reconnect

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
dataset = pd.read_csv('/content/train_u6lujuX_CVtuZ9i (1).csv')
```

```
dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

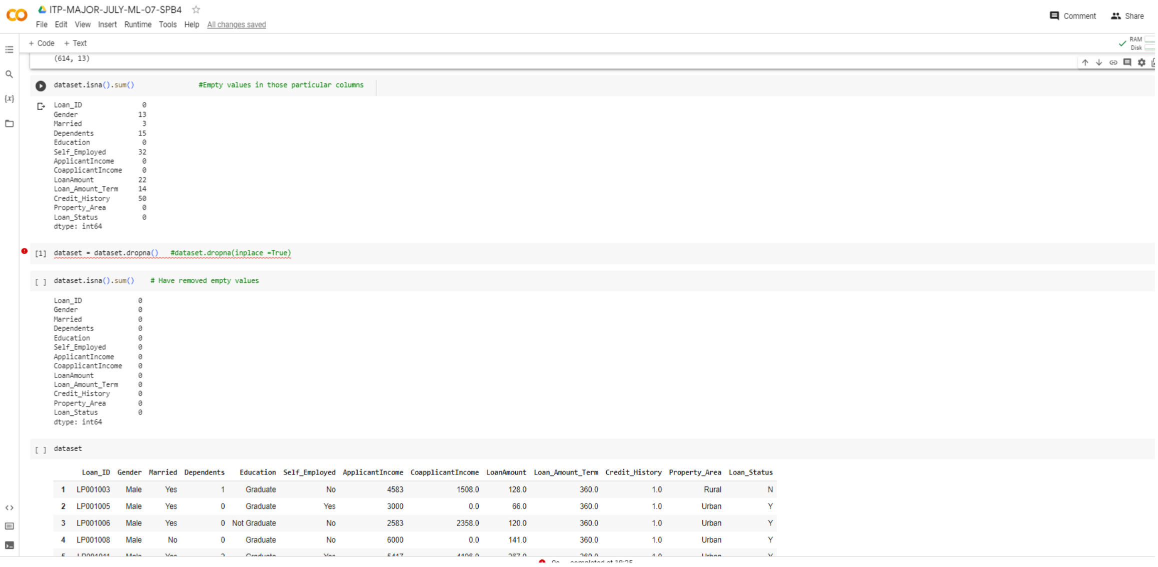
```
dataset.shape
```

```
(614, 13)
```

```
dataset.isna().sum()                    #Empty values in those particular columns
```

```
Loan_ID              0
Gender              13
Married              3
```

Understanding the Data


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text

```
(614, 13)
```

```
dataset.isna().sum()                    #Empty values in those particular columns
```

```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

```
[1] dataset = dataset.dropna()   #dataset.dropna(inplace =True)
```

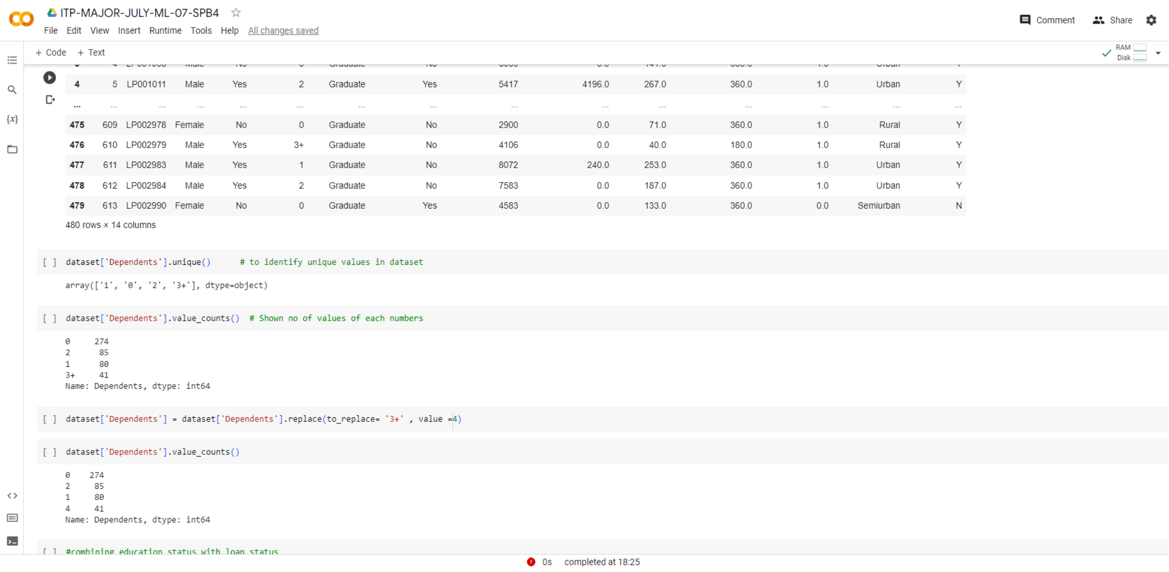
```
dataset.isna().sum()    # Have removed empty values
```

```
Loan_ID             0
Gender              0
Married             0
Dependents          0
Education           0
Self_Employed       0
ApplicantIncome     0
CoapplicantIncome   0
LoanAmount          0
Loan_Amount_Term    0
Credit_History      0
Property_Area       0
Loan_Status         0
dtype: int64
```

```
dataset
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

Dealing With Missing Values


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM
Disk

4	5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
476	610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
477	611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
478	612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
479	613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

480 rows × 14 columns

```
[ ] dataset['Dependents'].unique()      # to identify unique values in dataset

    array(['1', '0', '2', '3+'], dtype=object)
```

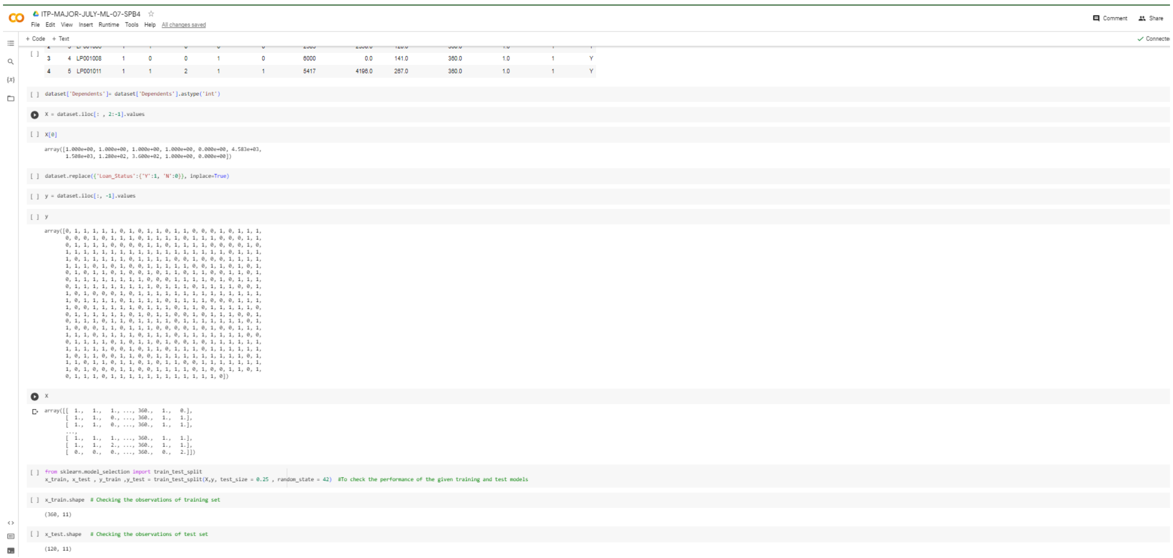
```
[ ] dataset['Dependents'].value_counts()  # Shown no of values of each numbers

    0      274
    2       85
    1       80
    3+      41
    Name: Dependents, dtype: int64
```

```
[ ] dataset['Dependents'] = dataset['Dependents'].replace(to_replace= '3+' , value =4)
```

```
[ ] dataset['Dependents'].value_counts()

    0     274
    2      85
    1      80
    4      41
    Name: Dependents, dtype: int64
```

```
[ ] #combining education status with loan status
```

0s completed at 18:25


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connecte

3	4	LP001008	1	0	0	1	0	6000	0.0	141.0	360.0	1.0	1	Y
4	5	LP001011	1	1	2	1	1	5417	4196.0	267.0	360.0	1.0	1	Y

```
[ ] dataset['Dependents']= dataset['Dependents'].astype('int')
```

```
X = dataset.iloc[: , 2:-1].values
```

```
[ ] X[0]

    array([1.000e+00, 1.000e+00, 1.000e+00, 1.000e+00, 0.000e+00, 4.583e+03,
           1.508e+03, 1.280e+02, 3.600e+02, 1.000e+00, 0.000e+00])
```

```
[ ] dataset.replace({'Loan_Status':{'Y':1, 'N':0}}, inplace=True)
```

```
[ ] y = dataset.iloc[:, -1].values
```

```
[ ] y

    array([0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
           0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1,
           0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
           1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
           1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1,
           1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
           0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
           0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
           0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1,
           1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
           1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1,
           1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
           0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
           0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
           1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
           1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
           0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
           1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
           1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
           1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
           1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
           0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0])
```

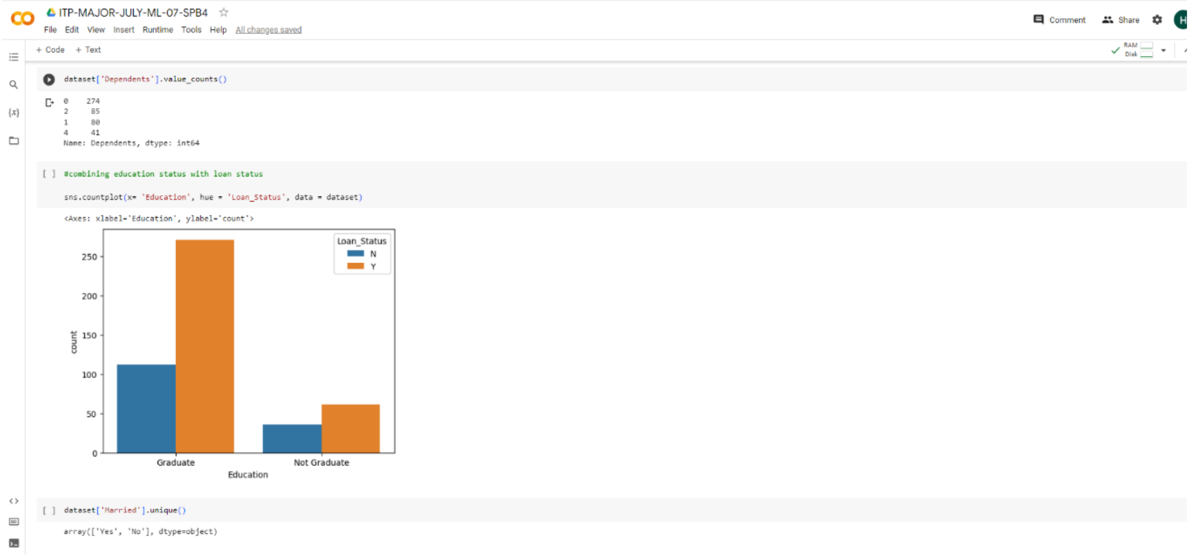
```
X

    array([[  1.,   1.,   1., ..., 360.,   1.,   0.],
           [  1.,   1.,   0., ..., 360.,   1.,   1.],
           [  1.,   1.,   0., ..., 360.,   1.,   1.],
           ...,
           [  1.,   1.,   1., ..., 360.,   1.,   1.],
           [  1.,   1.,   2., ..., 360.,   1.,   1.],
           [  0.,   0.,   0., ..., 360.,   0.,   2.]])
```

```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test , y_train ,y_test = train_test_split(X,y, test_size = 0.25 , random_state = 42)  #To check the performance of the given training and test models
```

```
[ ] x_train.shape  # Checking the observations of training set

    (360, 11)
```

```
[ ] x_test.shape   # Checking the observations of test set

    (120, 11)
```

Data Visualization

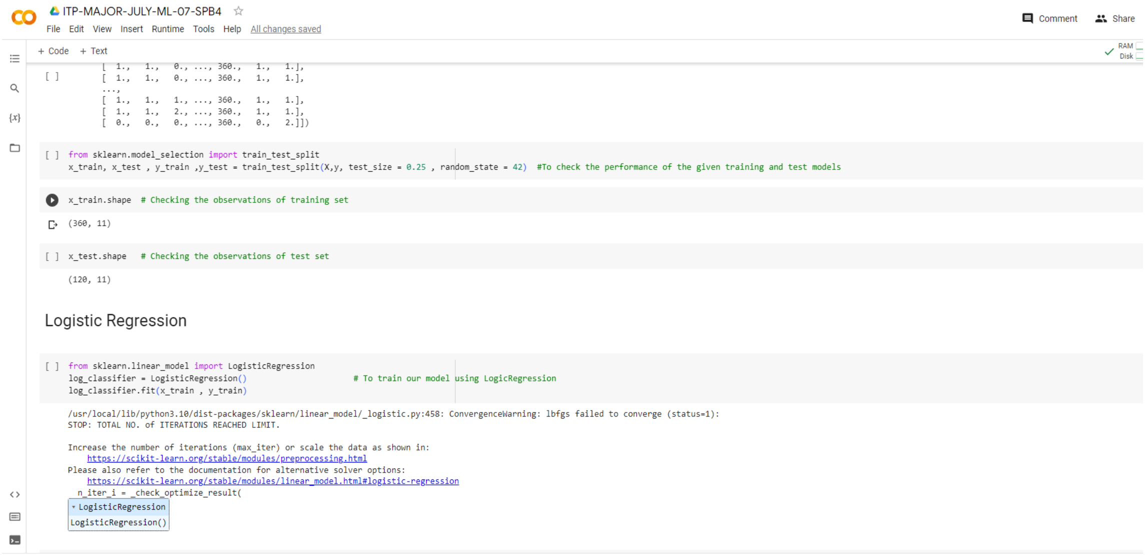


```
dataset['Dependents'].value_counts()

0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

```
#combining education status with loan status

sns.countplot(x= 'Education', hue = 'Loan_Status', data = dataset)

<Axes: xlabel='Education', ylabel='count'>
```

```
dataset['Married'].unique()

array(['Yes', 'No'], dtype=object)
```

Divide the Dataset Into Training and Test Datasets and Building the Model



ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text

```
       [  1.,   1.,   0., ..., 360.,   1.,   1.],
       [  1.,   1.,   0., ..., 360.,   1.,   1.],
       ...,
       [  1.,   1.,   1., ..., 360.,   1.,   1.],
       [  1.,   1.,   2., ..., 360.,   1.,   1.],
       [  0.,   0.,   0., ..., 360.,   0.,   2.]])
```

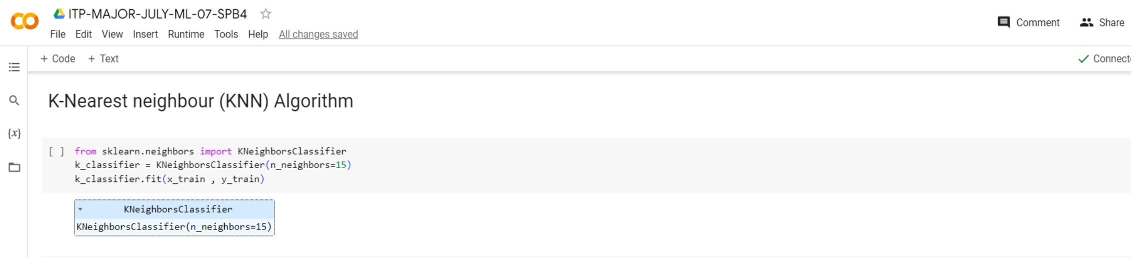
```
from sklearn.model_selection import train_test_split
x_train, x_test , y_train ,y_test = train_test_split(X,y, test_size = 0.25 , random_state = 42)  #To check the performance of the given training and test models
```

```
x_train.shape  # Checking the observations of training set

(360, 11)
```

```
x_test.shape   # Checking the observations of test set

(120, 11)
```

Logistic Regression

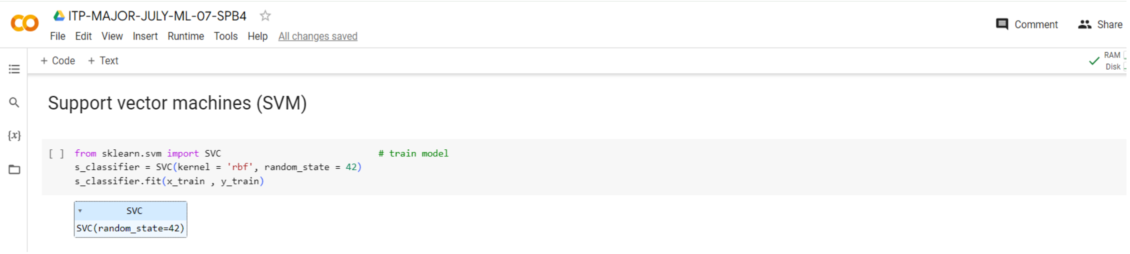
```
from sklearn.linear_model import LogisticRegression
log_classifier = LogisticRegression()                      # To train our model using LogicRegression
log_classifier.fit(x_train , y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect

K-Nearest neighbour (KNN) Algorithm

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    k_classifier = KNeighborsClassifier(n_neighbors=15)
    k_classifier.fit(x_train , y_train)
```

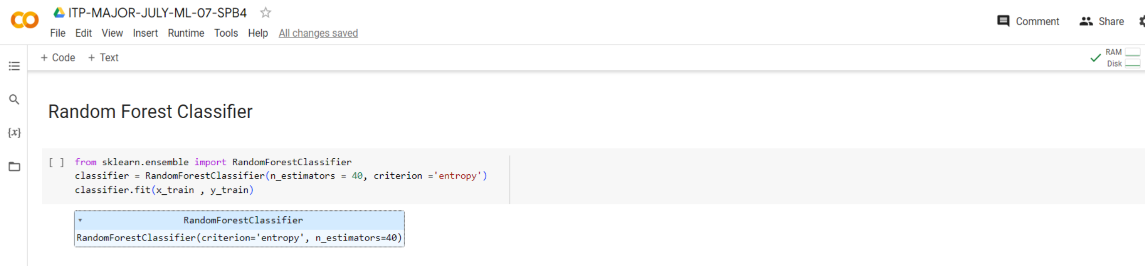
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=15)
```


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

Support vector machines (SVM)

```
[ ] from sklearn.svm import SVC                          # train model
    s_classifier = SVC(kernel = 'rbf', random_state = 42)
    s_classifier.fit(x_train , y_train)
```

```
SVC
SVC(random_state=42)
```

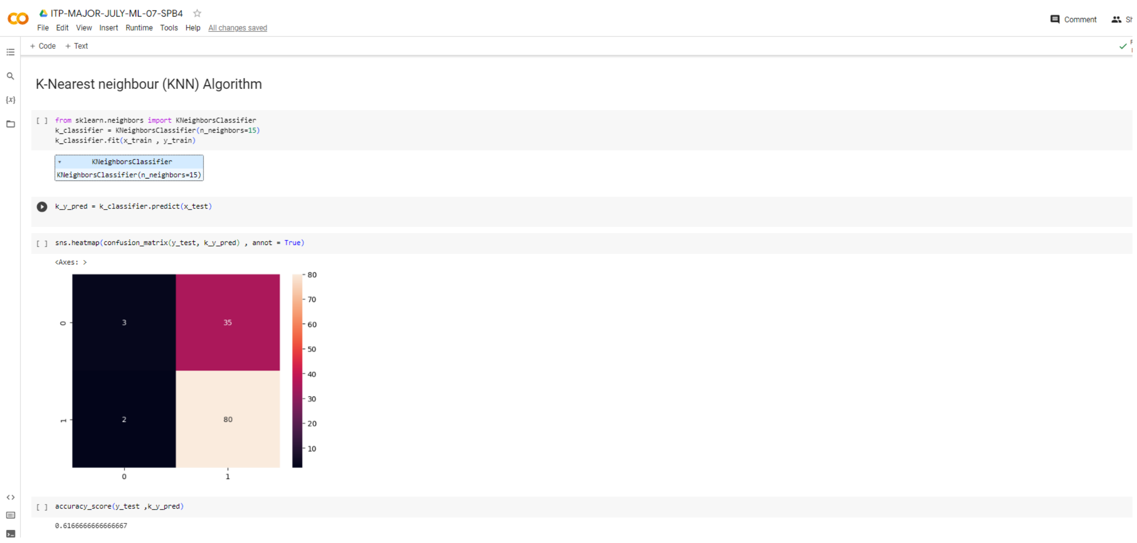

ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

Random Forest Classifier

```
[ ] from sklearn.ensemble import RandomForestClassifier
    classifier = RandomForestClassifier(n_estimators = 40, criterion ='entropy')
    classifier.fit(x_train , y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=40)
```

Test the Model and Find the Accuracy of the Model


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
0
1
0
1
3
35
2
80
80
70
60
50
40
30
20
10

K-Nearest neighbour (KNN) Algorithm

```
from sklearn.neighbors import KNeighborsClassifier
k_classifier = KNeighborsClassifier(n_neighbors=15)
k_classifier.fit(x_train , y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=15)
```

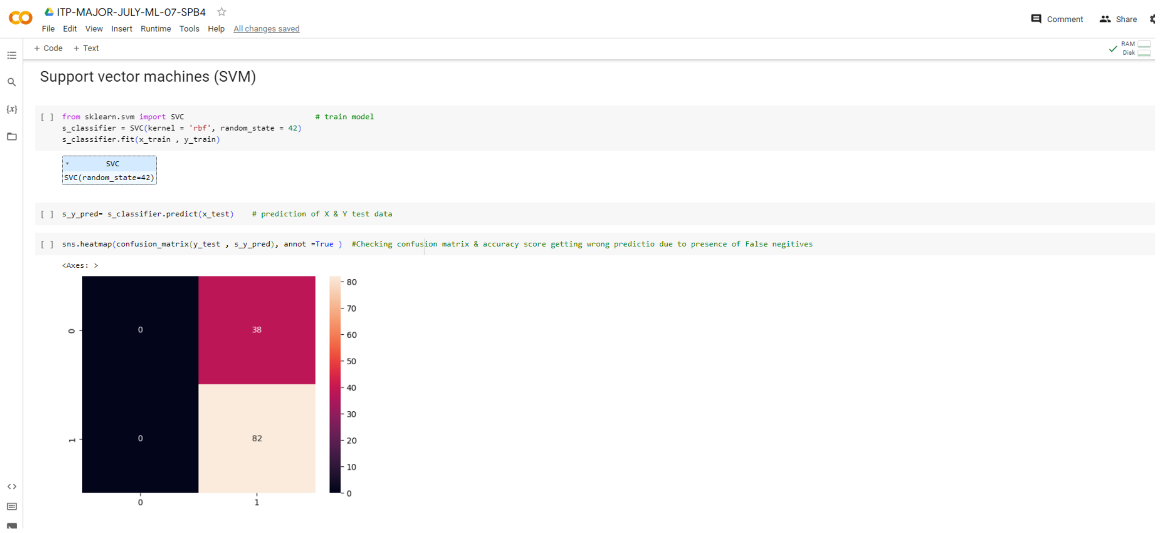
```
k_y_pred = k_classifier.predict(x_test)
```

```
sns.heatmap(confusion_matrix(y_test, k_y_pred) , annot = True)
```

```
<Axes: >
```

```
accuracy_score(y_test ,k_y_pred)
```

```
0.6166666666666667
```


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
RAM
Disk
+ Code + Text
0
1
0
1
0
38
0
82
80
70
60
50
40
30
20
10
0

Support vector machines (SVM)

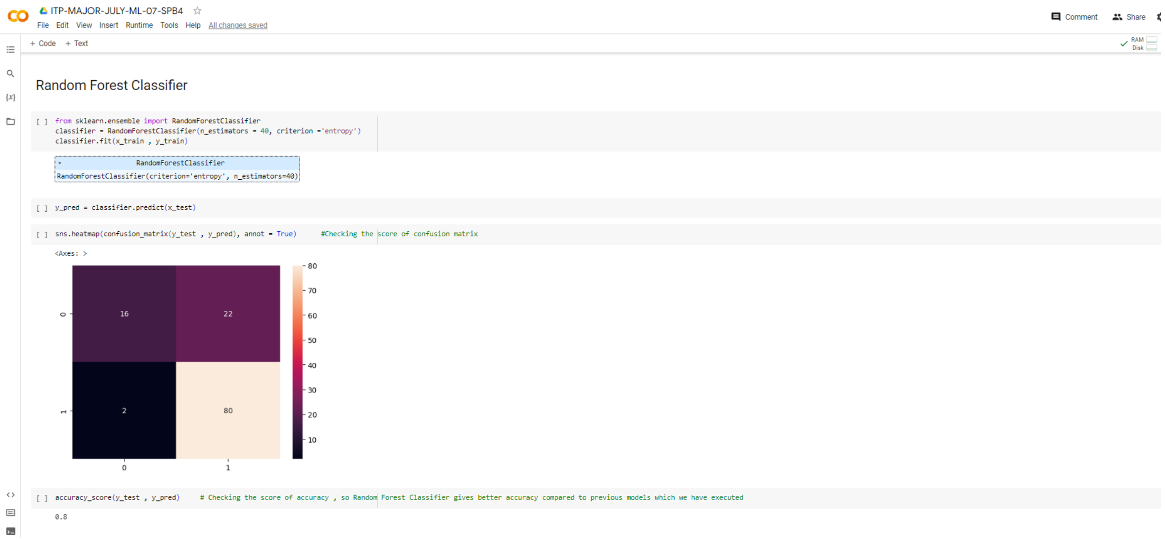
```
from sklearn.svm import SVC                              # train model
s_classifier = SVC(kernel = 'rbf', random_state = 42)
s_classifier.fit(x_train , y_train)
```

```
SVC
SVC(random_state=42)
```

```
s_y_pred= s_classifier.predict(x_test)    # prediction of X & Y test data
```

```
sns.heatmap(confusion_matrix(y_test , s_y_pred), annot =True )  #Checking confusion matrix & accuracy score getting wrong predictio due to presence of False negitives
```

```
<Axes: >
```


ITP-MAJOR-JULY-ML-07-SPB4
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
RAM
Disk
+ Code + Text
0
1
0
1
16
22
2
80
80
70
60
50
40
30
20
10

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 40, criterion ='entropy')
classifier.fit(x_train , y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=40)
```

```
y_pred = classifier.predict(x_test)
```

```
sns.heatmap(confusion_matrix(y_test , y_pred), annot = True)      #Checking the score of confusion matrix
```

```
<Axes: >
```

```
accuracy_score(y_test , y_pred)     # Checking the score of accuracy , so Random Forest Classifier gives better accuracy compared to previous models which we have executed
```

```
0.8
```

Conclusion

The "Loan Sanction Prediction" project involves several key steps, from data collection to model deployment. By following these steps and utilizing Python's machine learning ecosystem, you can create a predictive model that assists in making informed decisions about loan approvals, contributing to efficient and accurate decision-making processes for financial institutions and by implementing several models LR, KNN, SVM , RFC (Random Forest Classifier) model has better accuracy and confusion matrix score as compared to other models . From the above dataset we get to know that the loan is sanctioned to the individual person.