Loan Status Prediction Python with Machine Learning

Guided by: Mr Rakesh Kumar Jha

IIHT Gurgoan(Hariyana)

ANKIT KUMAR

CUSB2102312010

SEMESTER: 4TH SEM

SESSION: 2021-23

Table of Content:

1.Introduction of Loan Predication

- ❑Loan is essential part of the lending process.
- ❑Core Business of banks.
- ❑Main Profit comes directly from the loan's interest.
- ❑Faced on financial Institutions such as Bank, credit Unions and other lending organization.
- ❑Accurate Predication helps minimize risks & Make better lending decision.

2. Problem Statement

- Old systems were made using Java, so needed to install a device.
- It didn’t provide feature of online Backup.
- Lot of Human resource required.
- More time consuming on verification Process.

3. Solution

- Our Machine Model calculates all parameters given and if the application eligible loan or not in very less time.
- Uses of Machine Learning Algorithm.

4. Machine Learning

- ❑It is a subset of Artificial Intelligence.
- ❑It performs on Scientific study of algorithm.
- ❑It is used in Statistical models with computer systems.
- ❑Importance of Machine Learning:

→ Fraud Detection

→ Better Decision making

→Medical Diagnosis & Treatment etc.

Machine Learning

Supervised Learning

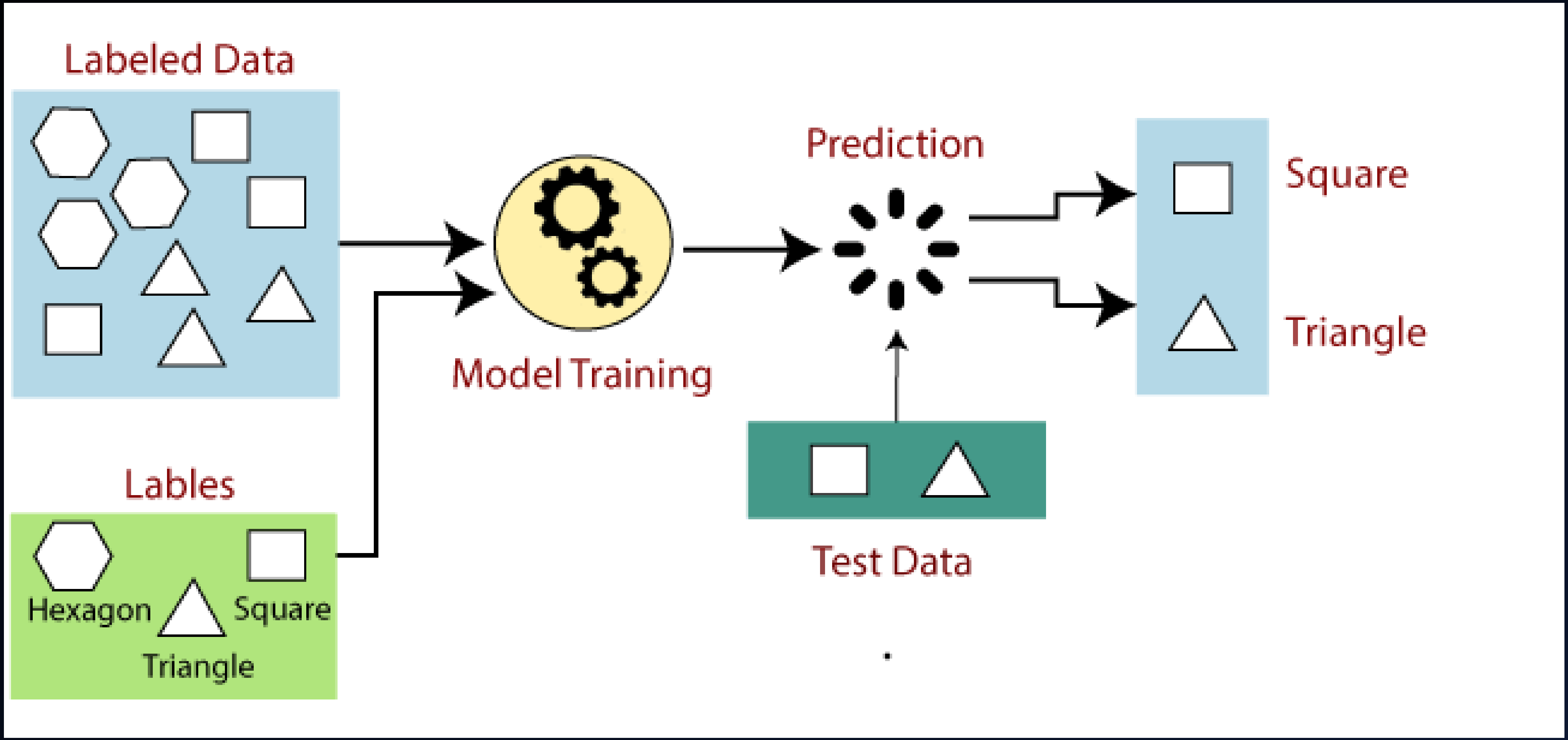
- Learns to make predications based on Labeled training data.
- Ex:- Spam Detection, Risk assessment etc.

Unsupervised Learning

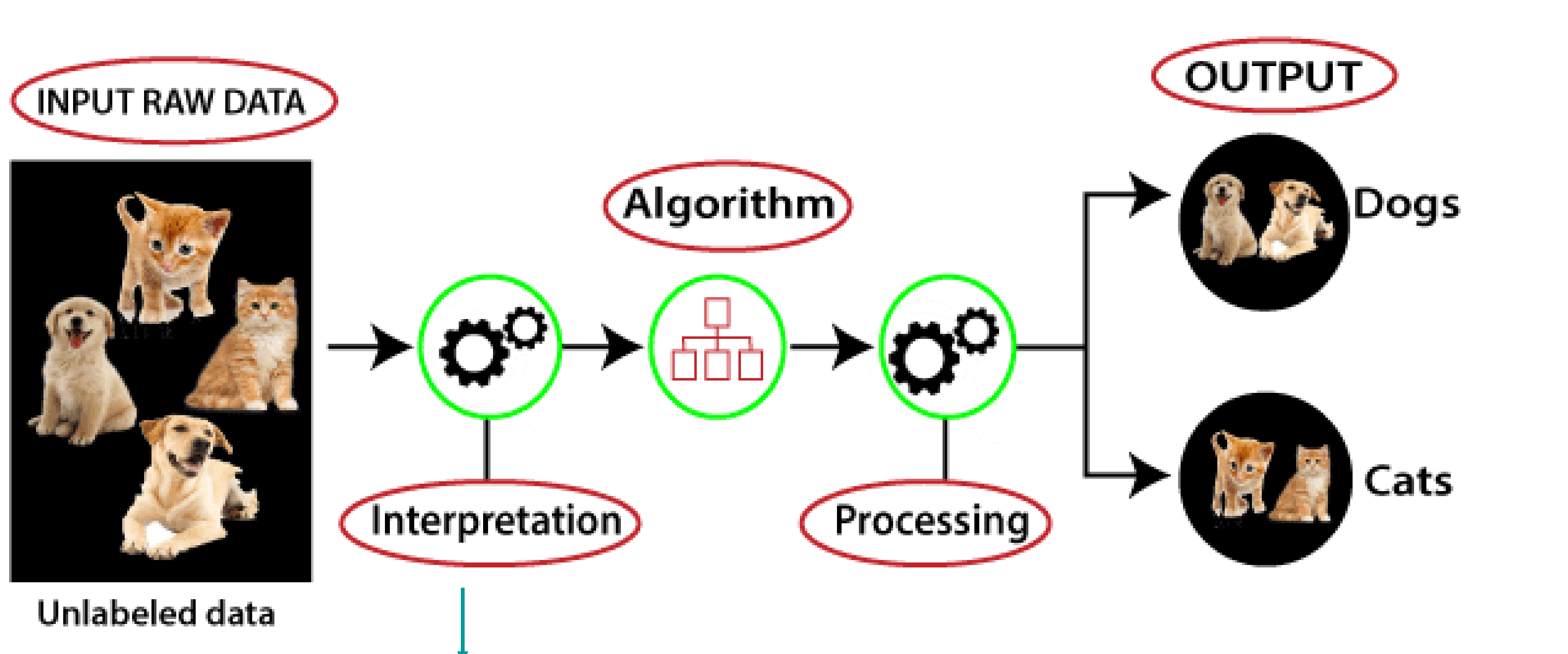
- Input data is not labeled, but goal is discover meaningful data.
- Ex:- Market basket analysis etc.

→ Reinforcement Learning:- Used in Robotics & game playing.

Supervised Learning



Unsupervised Learning



(Identify Hidden Pattern: Relevant features or characteristics from the data, structures of data or relationships of data etc.)

5. HARDWARE & SOFTWARE USED

❑Hardware Used:

→ Windows 10

❑Software/ Code Editor Used

→ JupiterLite, Google colab

❑Libraries used(Python Language)

→ Pandas

→ Matplotlib & Seaborn

→ NumPy

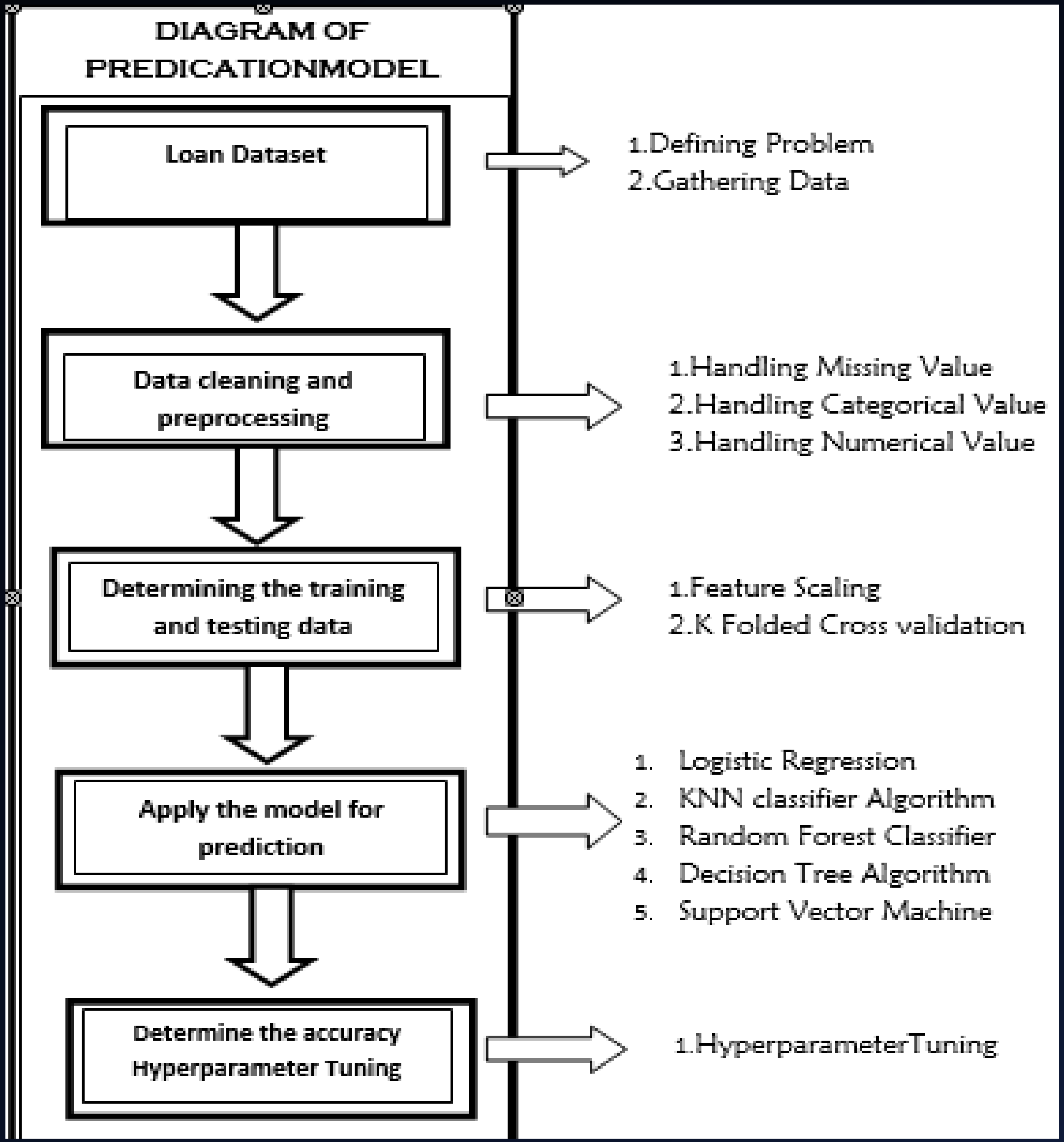
→ SKlearn

Libraries used(Python Language)

- ✓ Collection of pre-written code and functionalities.
- ✓ Used to perform scientific task or solve scientific problems.

- ✓ High Level Language
- ✓ Interpreted Language
- ✓ Easy to learn and simple syntax.
- ✓ Developed by: Guido van Rossum(1990s)
- ✓ 3.11.2 latest release of the python.

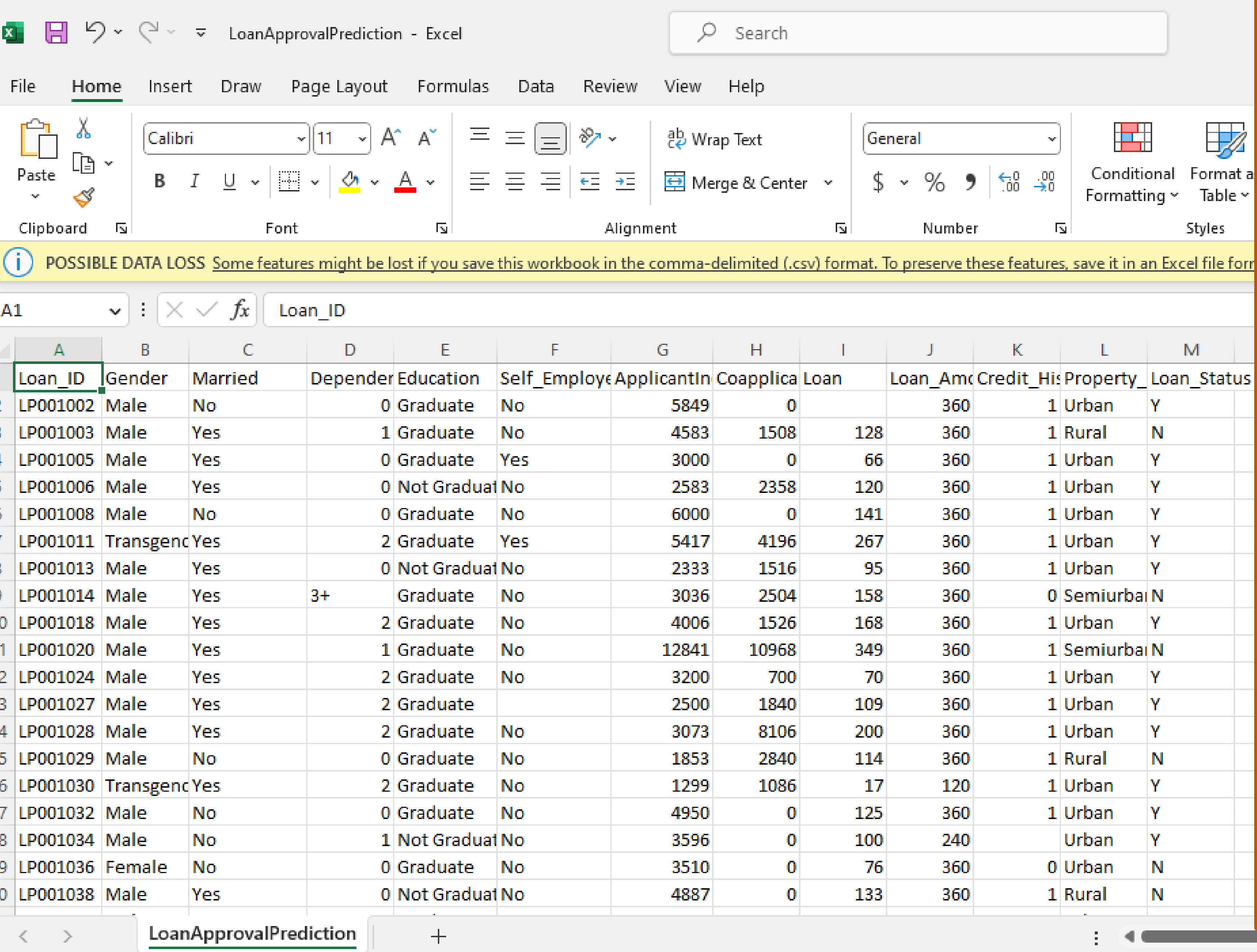
6. Implementation of Model



7. Importing of Dataset

- → Pandas in python provide an interesting method read_csv().
- → The read_csv function reads the entire dataset from a comma separated values(CSV) file.
- → Each and every value can be access using the data frame.
- → Any missing value or NaN value have to be cleaned.

* Dataset & Information*



Loan_ID	Gender	Married	Depender	Education	Self_Employe	ApplicantIn	Coapplica	Loan	Loan_Amc	Credit_His	Property_	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
LP001006	Male	Yes	0	Not Graduat	No	2583	2358	120	360	1	Urban	Y
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
LP001011	Transgenc	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
LP001013	Male	Yes	0	Not Graduat	No	2333	1516	95	360	1	Urban	Y
LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurbai	N
LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurbai	N
LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
LP001030	Transgenc	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
LP001034	Male	No	1	Not Graduat	No	3596	0	100	240		Urban	Y
LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
LP001038	Male	Yes	0	Not Graduat	No	4887	0	133	360	1	Rural	N

```
data.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   Loan         Amount 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: 43.2+ KB
```

Reading the Dataset using Notebook:

```
#Loan Status Prediction python with Machine Learning
```

```
import pandas as pd
import numpy as num
```

```
data = pd.read_csv('LoanStatusPrediction.csv')
```

→data.head(5)
→data.tail(5)
→data.shape()

```
#find the shape of our dataset < No of Rows and No of Columns >
#shape-Pandas Dataframe
data.shape
```

```
(614, 13)
```

```
print("No of Rows = ",data.shape[0])
print("No of Columns= ",data.shape[1])
```

```
No of Rows =  614
No of Columns=  13
```

Handling the Missing Value

- Drop of Loan_Id completely unique and not correlated with any of the other column, So we will drop it using .drop() function.

```
data=data.drop('Loan_ID',axis=1)
```

```
data.head(1)
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loan Amount	Loan_Amount_Term	Cred
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	

- Count and Sum of value in all columns

```
#Count and sum of null value
data.isnull().sum()
```

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

❖ Find percentage of sum of null value of Total value (Columns)

```
#find percentage of sum of null value of Total value
```

```
data.isnull().sum()*100 / len(data)
```

```
Gender                  2.117264
Married                 0.488599
Dependents              2.442997
Education               0.000000
Self_Employed           5.211726
ApplicantIncome         0.000000
CoapplicantIncome       0.000000
Loan          Amount    3.583062
Loan_Amount_Term        2.280130
Credit_History          8.143322
Property_Area           0.000000
Loan_Status             0.000000
dtype: float64
```

❖ Remove null value for all columns (Training Data <75%)

```
#Highest Null value of Percentage is Credit_History->8.14
#if 70% training data:-consider on Self_Employed--> 5.21
#Remove null value is less than 5.21 all columns
```

```
columns=['Gender','Dependents','Loan              Amount','Loan_Amount_Term']
```

```
data= data.dropna(subset=columns)
```

```
data.isnull().sum()*100/ len(data)
```

```
Gender                  0.000000
Married                 0.000000
Dependents              0.000000
Education               0.000000
Self_Employed           5.424955
ApplicantIncome         0.000000
CoapplicantIncome       0.000000
Loan          Amount    0.000000
Loan_Amount_Term        0.000000
Credit_History          8.679928
Property_Area           0.000000
Loan_Status             0.000000
dtype: float64
```

Remove null value: Gender, Dependents, LoanAmount, Loan_Amount_term

```
#check on unique value as Self_Employed and Credit_History
```

```
data['Self_Employed'].unique()
```

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

```
data['Credit_History'].unique()
```

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

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

```
data.isnull().sum()*100/ len(data)
```

```
Gender                     0.0
Married                    0.0
Dependents                 0.0
Education                  0.0
Self_Employed              0.0
ApplicantIncome            0.0
CoapplicantIncome          0.0
Loan              Amount   0.0
Loan_Amount_Term           0.0
Credit_History             0.0
Property_Area              0.0
Loan_Status                0.0
dtype: float64
```

#Successfully Handling the Missing Value (Null Value)

Handling the Categorical Data

```
[ ] data['Dependents'].unique() #Check unique value

    array(['1', '0', '2', '4'], dtype=object)

[ ] data['Gender'].unique()

    array(['Male', 'Transgender', 'Female'], dtype=object)

[ ] data['Married'].unique()

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

[ ] data['Education'].unique()

    array(['Graduate', 'Not Graduate'], dtype=object)

[ ] data['Property_Area'].unique()

    array(['Rural', 'Urban', 'Semiurban'], dtype=object)

    data['Self_Employed'].unique()

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

```
data['Self_Employed'].unique()
```

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

```
[ ] data['Loan_Status'].unique()
```

```
array(['N', 'Y'], dtype=object)
```

```
[ ] obj =(data.dtypes =='object')
    print("Categorical variables:",len(list(obj[obj].index)))
```

```
Categorical variables: 7
```

Successfully Handling the Categorical Data

Handling the Numerical Data

```
[ ]  data['Gender'] = data['Gender'].map({'Male':1,'Transgender':2,'Female':0}).astype('int')

[ ]  data['Gender'].unique()

     array([1, 2, 0])

[ ]  data['Married']=data['Married'].map({'Yes':1, 'No':0}).astype('int')

[ ]  data['Married'].unique()

     array([1, 0])

[ ]  data['Education']=data['Education'].map({'Graduate':1, 'Not Graduate':0}).astype('int')

[ ]  data['Education'].unique()

     array([1, 0])

[ ]  data['Self_Employed']=data['Self_Employed'].map({'No':0, 'Yes':1}).astype('int')

[ ]  data['Self_Employed'].unique()
```

```
[ ] data['Self_Employed'].unique()

array([0, 1])

[ ] data['Property_Area']=data['Property_Area'].map({'Rural':0, 'Urban':1, 'Semiurban':2}).astype('int')

[ ] data['Property_Area'].unique()

array([0, 1, 2])

[ ] data['Loan_Status']=data['Loan_Status'].map({'N':0, 'Y':1}).astype('int')

[ ] data['Loan_Status'].unique()

array([0, 1])
```

```
[ ] data.head(10)
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loan Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1	1	1	1	0	4583	1508.0	128.0	360.0	1.0	0	0
2	1	1	0	1	1	3000	0.0	66.0	360.0	1.0	1	1
3	1	1	0	0	0	2583	2358.0	120.0	360.0	1.0	1	1
4	1	0	0	1	0	6000	0.0	141.0	360.0	1.0	1	1
5	2	1	2	1	1	5417	4196.0	267.0	360.0	1.0	1	1
6	1	1	0	0	0	2333	1516.0	95.0	360.0	1.0	1	1
7	1	1	4	1	0	3036	2504.0	158.0	360.0	0.0	2	0
8	1	1	2	1	0	4006	1526.0	168.0	360.0	1.0	1	1
9	1	1	1	1	0	12841	10968.0	349.0	360.0	1.0	2	0
10	1	1	2	1	0	3200	700.0	70.0	360.0	1.0	1	1

Successfully Handling Numerical Data

```
[ ] #Store feature matrix in X and Response (Target)in vector Y
```

```
[ ] X=data.drop('Loan_Status',axis=1)
```

```
[ ] y=data['Loan_Status']
```

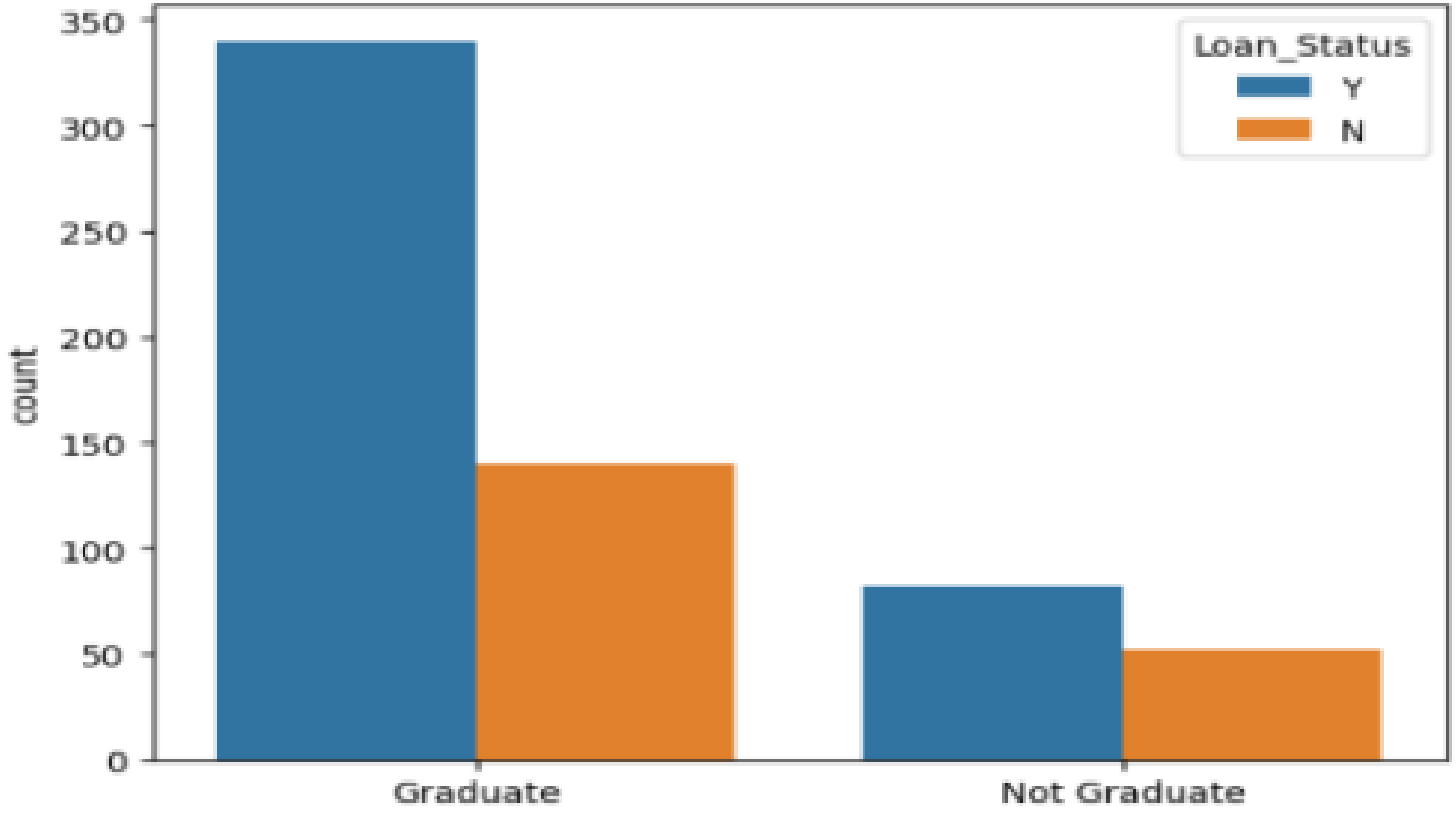
```
[ ] y
```

```
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 553, dtype: int64
```

→ Data Visualization

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

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



8. Training the Model

```
model_df={}
def model_val(model,X,y):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,
                                                   random_state=42)
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    print(f"{model} accuracy is {accuracy_score(y_test,y_pred)}")

    score=cross_val_score(model,X,y,cv=5)
    print(f"{model} Avg cross val score is {np.mean(score)}")
    model_df[model]=round(np.mean(score)*100,2)


model_df

{LogisticRegression(): 80.11,
 SVC(): 79.39,
 RandomForestClassifier(): 78.3,
 KNeighborsClassifier(): 73.23,
 DecisionTreeClassifier(): 70.89}
```

HyperParameter Tuning

1) Process of selecting the optimal values for the hyperparameters of a machine learning model.
2) Tuning:- Achieve good performance of model

```
LogisticRegression score Before Hyperparameter Tuning: 80.11
LogisticRegression score after Hyperparameter Tuning: 80.48

SVC score Before Hyperparameter Tuning: 79.39
SVC score after Hyperparameter Tuning: 80.21

RandomForestClassifier score Before Hyperparameter Tuning: 78.3
RandomForestClassifier score after Hyperparameter Tuning: 80.66

KNeighborsClassifier score Before Hyperparameter Tuning: 73.23
KNeighborsClassifier score after Hyperparameter Tuning: 73.69

DecisionTreeClassifier score Before Hyperparameter Tuning: 70.89
DecisionTreeClassifier score after Hyperparameter Tuning: 75.66
```

9. Save the Model

```
[ ]  RandomForestClassifier()
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
[ ]  import joblib
```

```
[ ]  joblib.dump(rf,'Loan_Status_Predication')
```

```
['Loan_Status_Predication']
```

```
[ ]  #Successfully Save Model
```

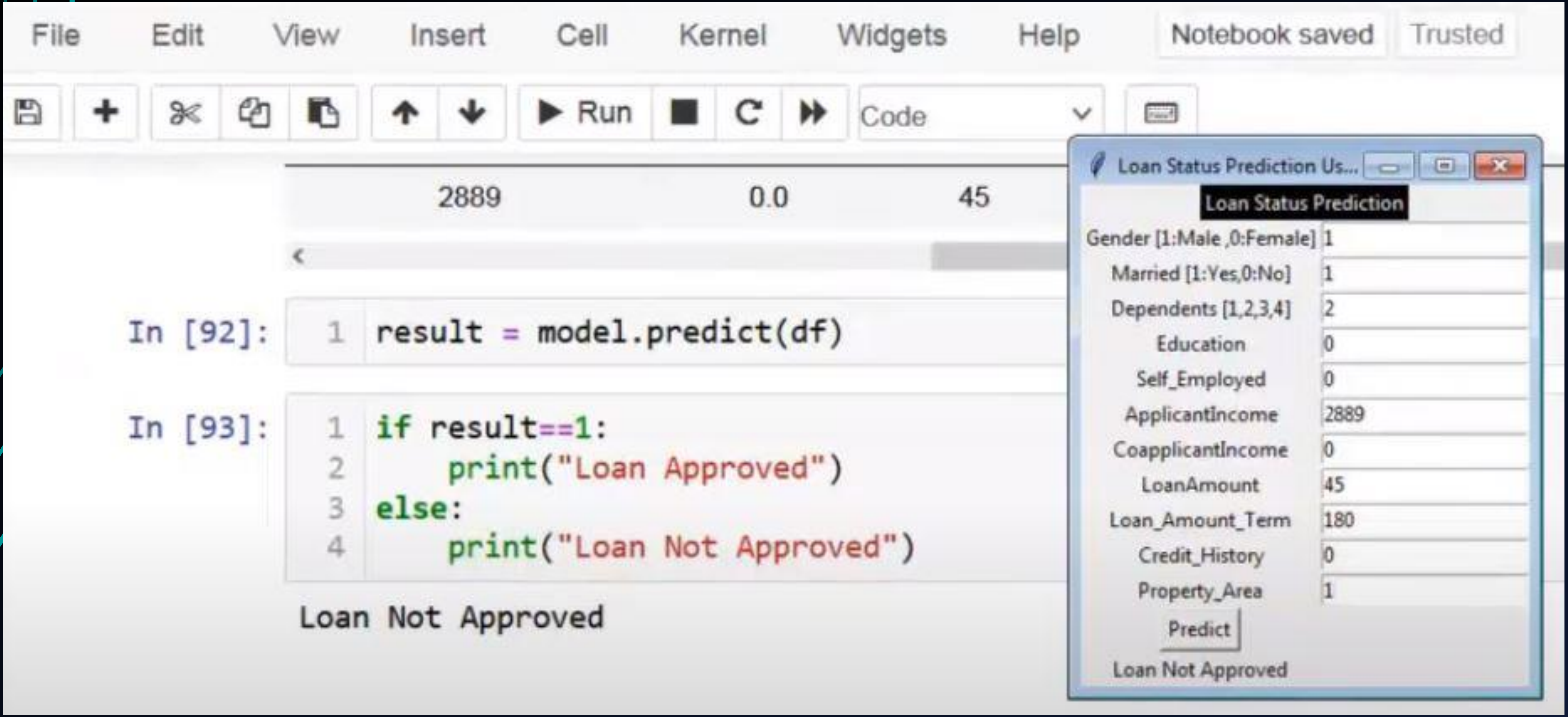
10.Check the Model

```
[ ] import pandas as pd
    df = pd.DataFrame({
        'Gender':1,
        'Married':1,
        'Dependents':2,
        'Education':0,
        'Self_Employed':0,
        'ApplicantIncome':2889,
        'CoapplicantIncome':0.0,
        'Loan                 Amount':45,
        'Loan_Amount_Term':180,
        'Credit_History':0,
        'Property_Area':1
    },index=[0])
```

```
[ ] df
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loan Amount	Loan_Amount_Term	Credit_History	Property_Ar
0	1	1	2	0	0	2889	0.0	45	180	0	

* Check the Model



11.Conclusion

→The loan application system approves or rejects loan applications.

→ Machine learning models are valuable for predicting outcomes with large datasets.

→ In our project, We utilized five machine learning algorithms are used as Logistic regression, support vector machines, random forest classifiers, KNN classifiers, and decision trees to predict loan approval for customers.

→ Based on experimental results, the random forest classifier algorithm exhibited the highest accuracy among the four algorithms.

12.Refrences

https://www.simplilearn.com/tutorials/machine-learning-tutorial/supervised-and-unsupervised-learning

https://machinelearningmastery.com/machine-learning-with-python/

https://www.kaggle.com/datasets/ninzaami/loan-predication?resource=download

https://numpy.org/doc/stable/

https://seaborn.pydata.org/generated/seaborn.boxplot.html

https://colab.research.google.com/

https://jupyter.org/try-jupyter/lab/

THANK YOU