GLA UNIVERSITY
ESTD. 2010
ऋते ज्ञानान्न मुक्तिः

GLA
UNIVERSITY
MATHURA
Established vide U.P. Act 21 of 2010.

# DEPARTMENT OF COMPUTER ENGINEERING &APPLICATIONS

# ML LAB FILE

**Name: Jagrati Dixit**

**University Rollno. : 2115000486**

**Subject Name: Machine Learning Lab**

**Subject Code: BCSE 0133**

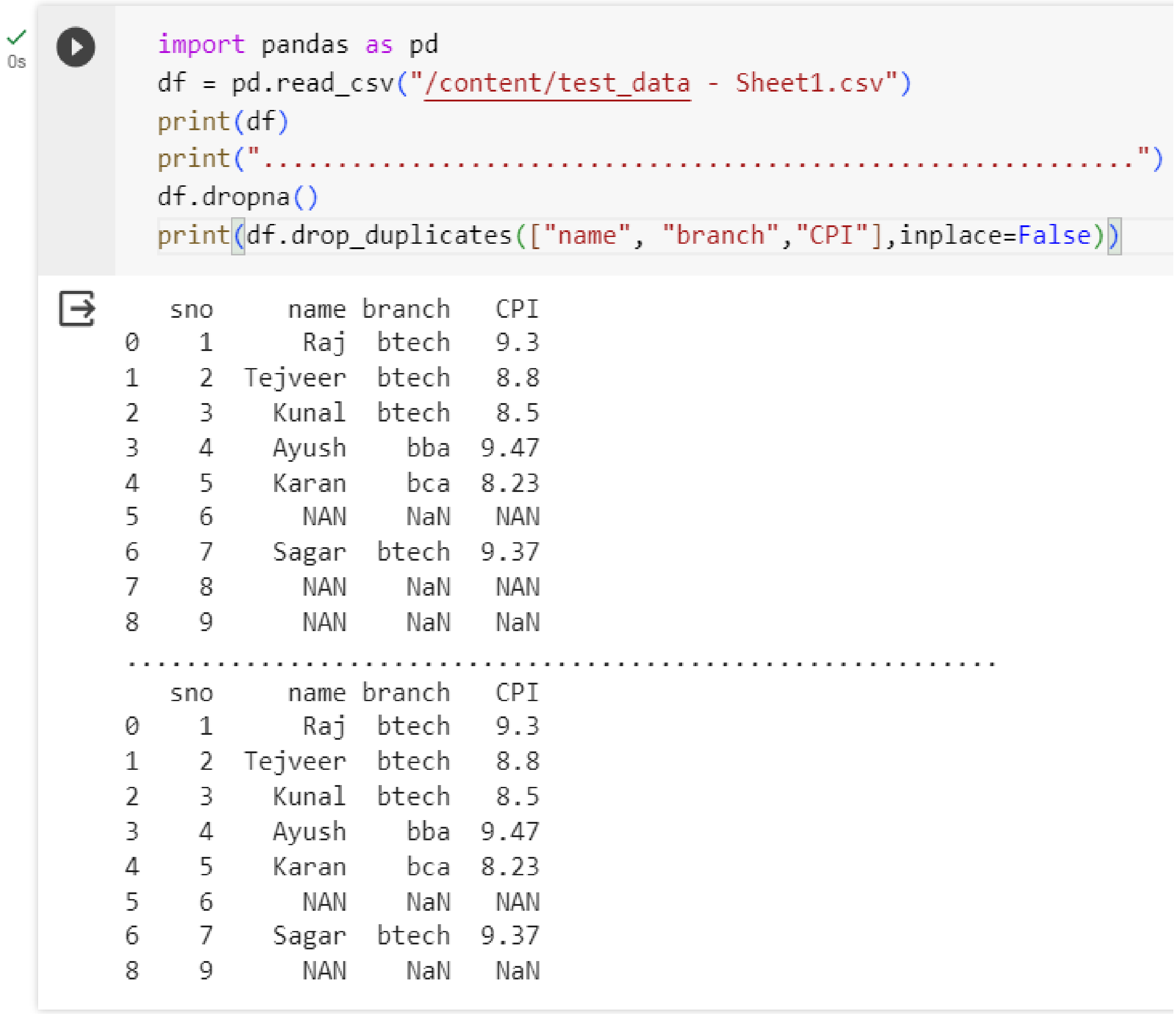
**Course:B.Tech.**

**Year: III Semester: V**

# Experiment-1

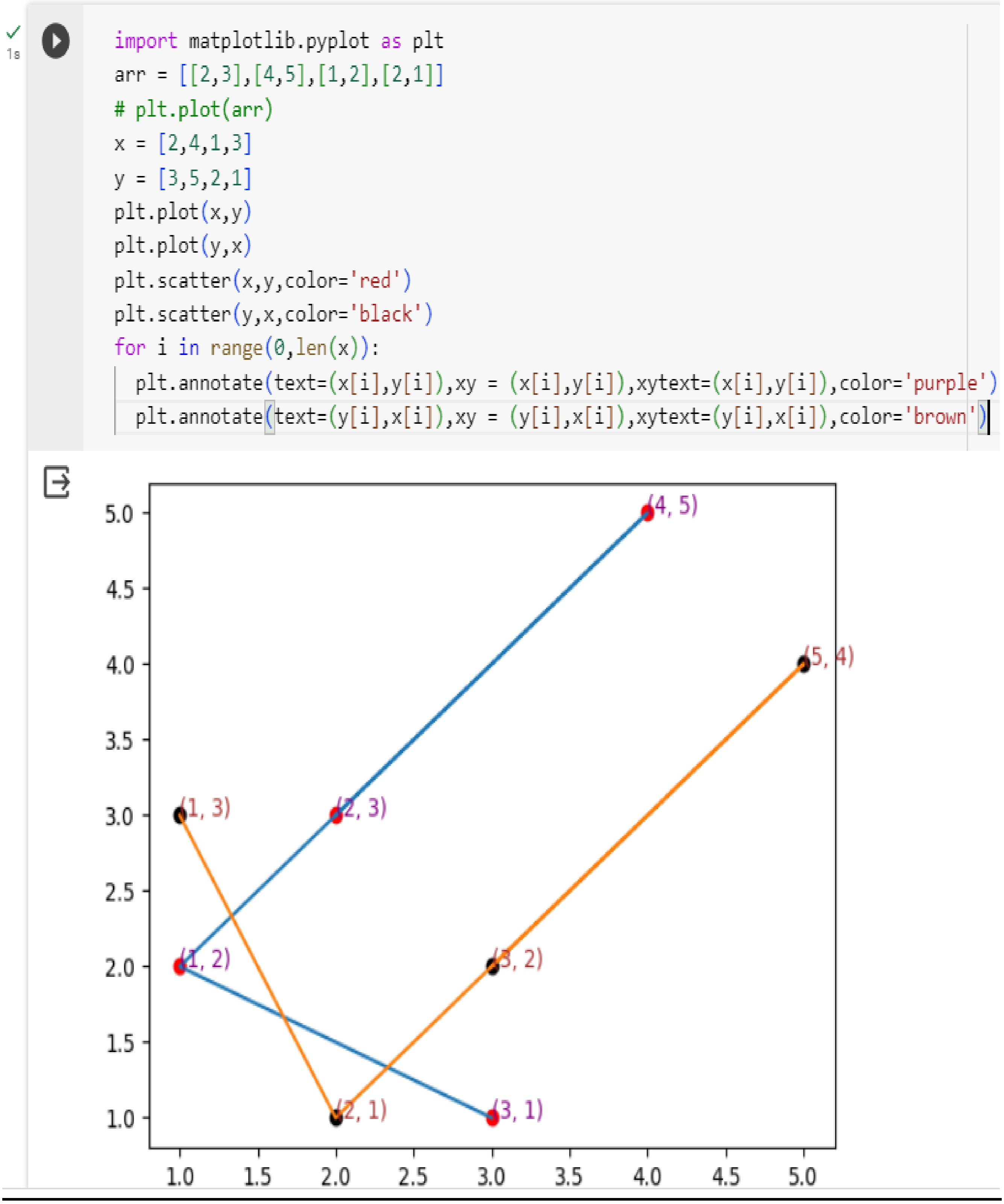
**Objective:-**Introduction to Pandas, Upload, data preprocessing, NumpyandMatplotlib library in Python.

**Implementaion :-**



```
import pandas as pd
df = pd.read_csv("/content/test_data - Sheet1.csv")
print(df)
print("...........................................................")
df.dropna()
print(df.drop_duplicates(["name", "branch","CPI"],inplace=False))
```

```
   sno     name branch   CPI
0    1      Raj  btech   9.3
1    2  Tejveer  btech   8.8
2    3    Kunal  btech   8.5
3    4    Ayush    bba  9.47
4    5    Karan    bca  8.23
5    6      NAN    NaN   NAN
6    7    Sagar  btech  9.37
7    8      NAN    NaN   NAN
8    9      NAN    NaN   NaN
...........................................................
   sno     name branch   CPI
0    1      Raj  btech   9.3
1    2  Tejveer  btech   8.8
2    3    Kunal  btech   8.5
3    4    Ayush    bba  9.47
4    5    Karan    bca  8.23
5    6      NAN    NaN   NAN
6    7    Sagar  btech  9.37
8    9      NAN    NaN   NaN
```



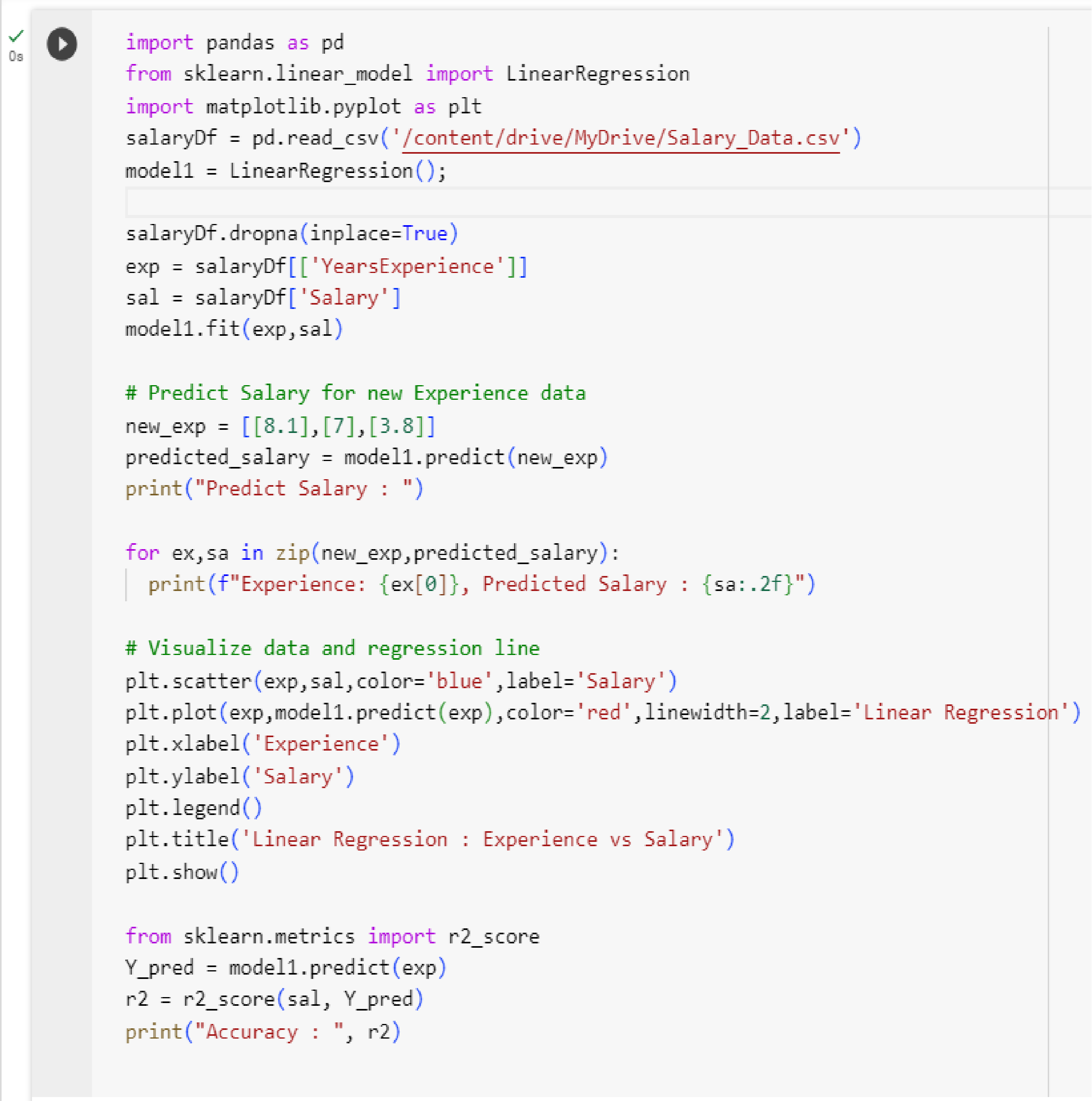
```
import matplotlib.pyplot as plt
arr = [[2,3],[4,5],[1,2],[2,1]]
# plt.plot(arr)
x = [2,4,1,3]
y = [3,5,2,1]
plt.plot(x,y)
plt.plot(y,x)
plt.scatter(x,y,color='red')
plt.scatter(y,x,color='black')
for i in range(0,len(x)):
  plt.annotate(text=(x[i],y[i]),xy = (x[i],y[i]),xytext=(x[i],y[i]),color='purple')
  plt.annotate(text=(y[i],x[i]),xy = (y[i],x[i]),xytext=(y[i],x[i]),color='brown')
```

# Experiment-2

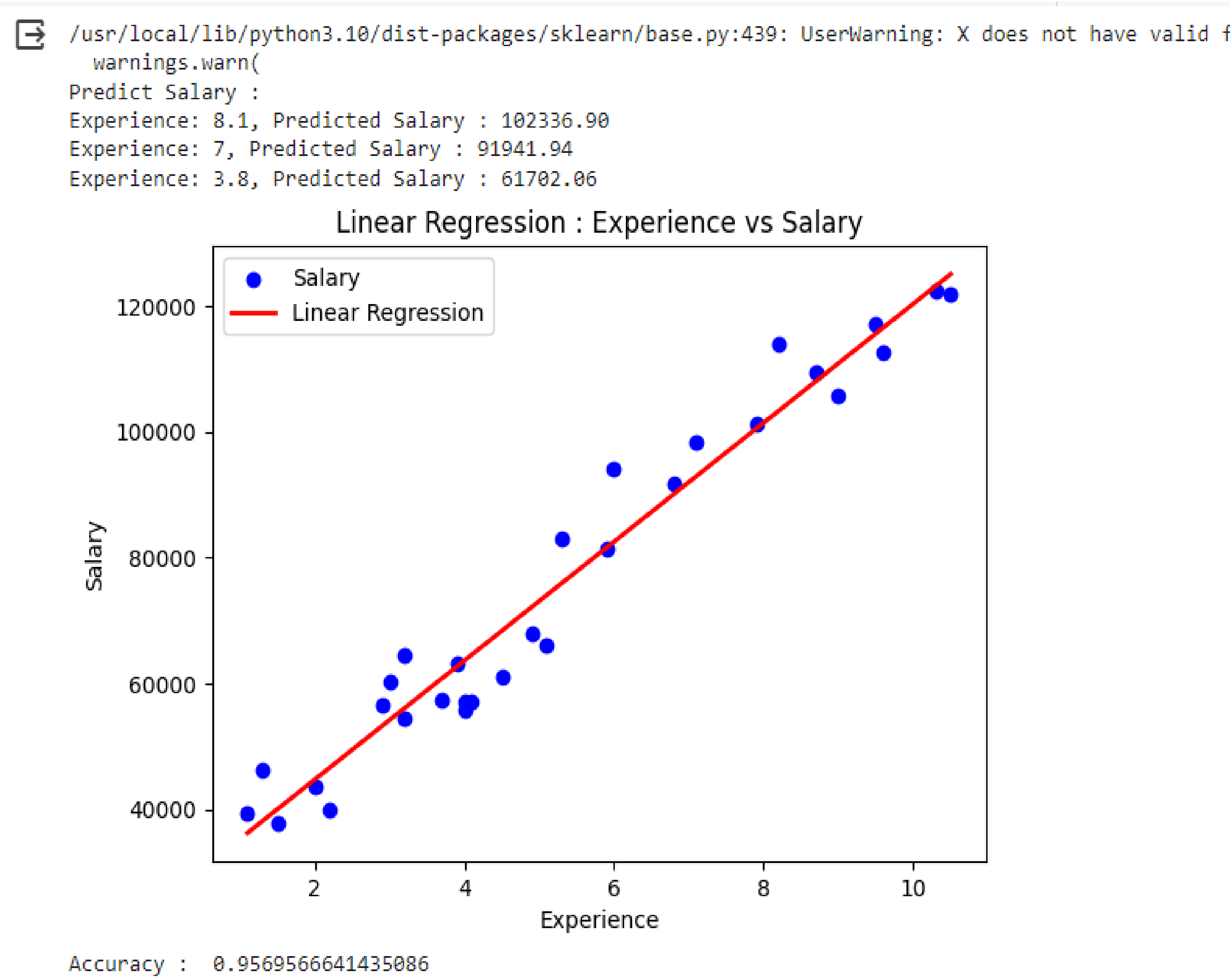
**Objective:-**To Implement Linear Regression with one variable in Python

**Dataset:-**https://www.kaggle.com/datasets/krishnaraj30/salary-prediction-data-simple-linear-regression

## Implementation :-



```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
salaryDf = pd.read_csv('/content/drive/MyDrive/Salary_Data.csv')
model1 = LinearRegression();

salaryDf.dropna(inplace=True)
exp = salaryDf[['YearsExperience']]
sal = salaryDf['Salary']
model1.fit(exp,sal)

# Predict Salary for new Experience data
new_exp = [[8.1],[7],[3.8]]
predicted_salary = model1.predict(new_exp)
print("Predict Salary : ")

for ex,sa in zip(new_exp,predicted_salary):
  print(f"Experience: {ex[0]}, Predicted Salary : {sa:.2f}")

# Visualize data and regression line
plt.scatter(exp,sal,color='blue',label='Salary')
plt.plot(exp,model1.predict(exp),color='red',linewidth=2,label='Linear Regression')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.legend()
plt.title('Linear Regression : Experience vs Salary')
plt.show()

from sklearn.metrics import r2_score
Y_pred = model1.predict(exp)
r2 = r2_score(sal, Y_pred)
print("Accuracy : ", r2)
```



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid f
  warnings.warn(
Predict Salary :
Experience: 8.1, Predicted Salary : 102336.90
Experience: 7, Predicted Salary : 91941.94
Experience: 3.8, Predicted Salary : 61702.06
```

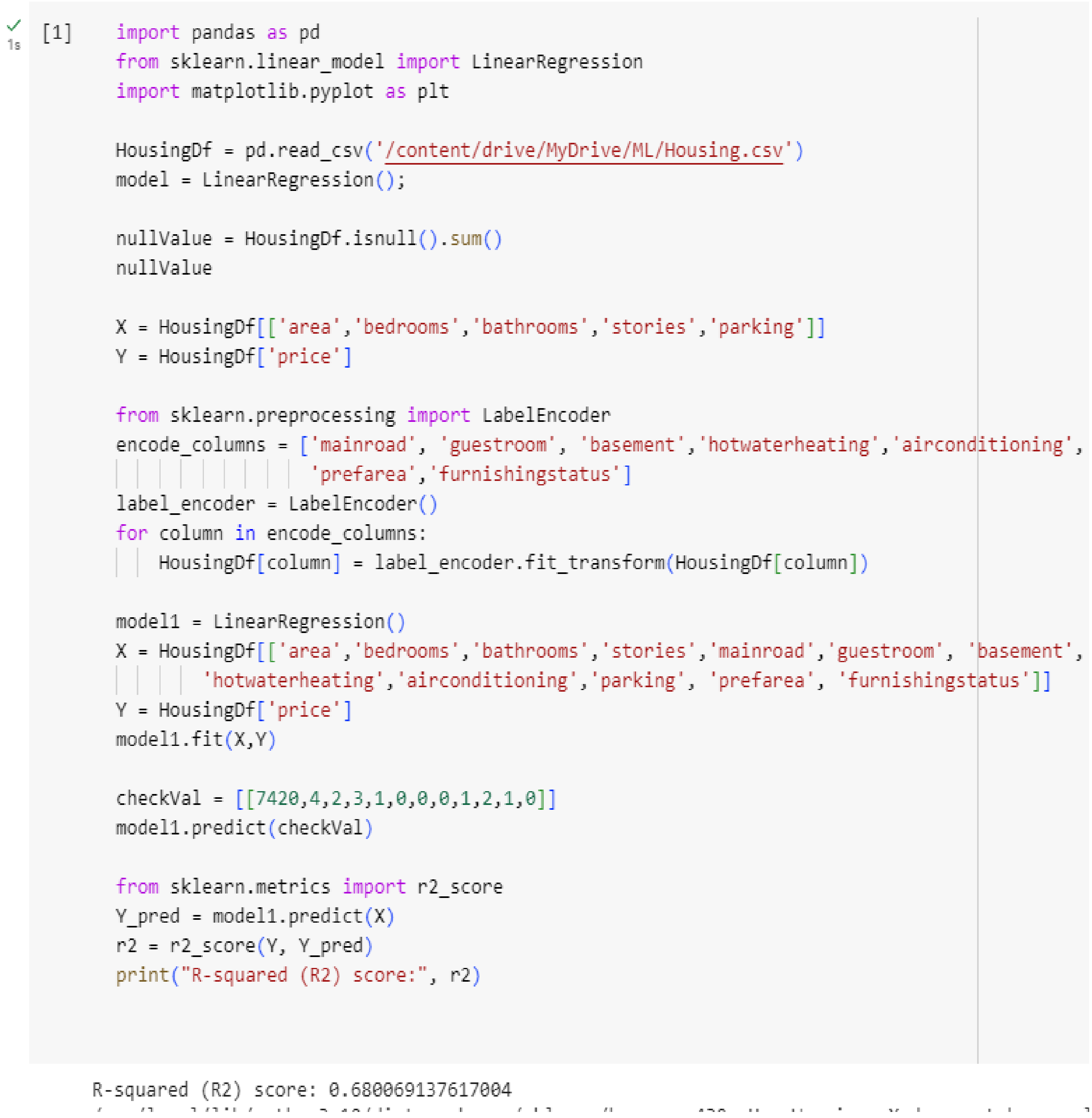
```
Accuracy :  0.9569566641435086
```

# Experiment-3

**Objective:-**To Implement Linear Regression with Multiple variable in Python

**Dataset:-**https://www.kaggle.com/datasets/yasserh/housing-prices-dataset

**Implementation :-**



```python
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

HousingDf = pd.read_csv('/content/drive/MyDrive/ML/Housing.csv')
model = LinearRegression();

nullValue = HousingDf.isnull().sum()
nullValue

X = HousingDf[['area','bedrooms','bathrooms','stories','parking']]
Y = HousingDf['price']

from sklearn.preprocessing import LabelEncoder
encode_columns = ['mainroad', 'guestroom', 'basement','hotwaterheating','airconditioning',
                  'prefarea','furnishingstatus']
label_encoder = LabelEncoder()
for column in encode_columns:
    HousingDf[column] = label_encoder.fit_transform(HousingDf[column])

model1 = LinearRegression()
X = HousingDf[['area','bedrooms','bathrooms','stories','mainroad','guestroom', 'basement',
        'hotwaterheating','airconditioning','parking', 'prefarea', 'furnishingstatus']]
Y = HousingDf['price']
model1.fit(X,Y)

checkVal = [[7420,4,2,3,1,0,0,0,1,2,1,0]]
model1.predict(checkVal)

from sklearn.metrics import r2_score
Y_pred = model1.predict(X)
r2 = r2_score(Y, Y_pred)
print("R-squared (R2) score:", r2)
```

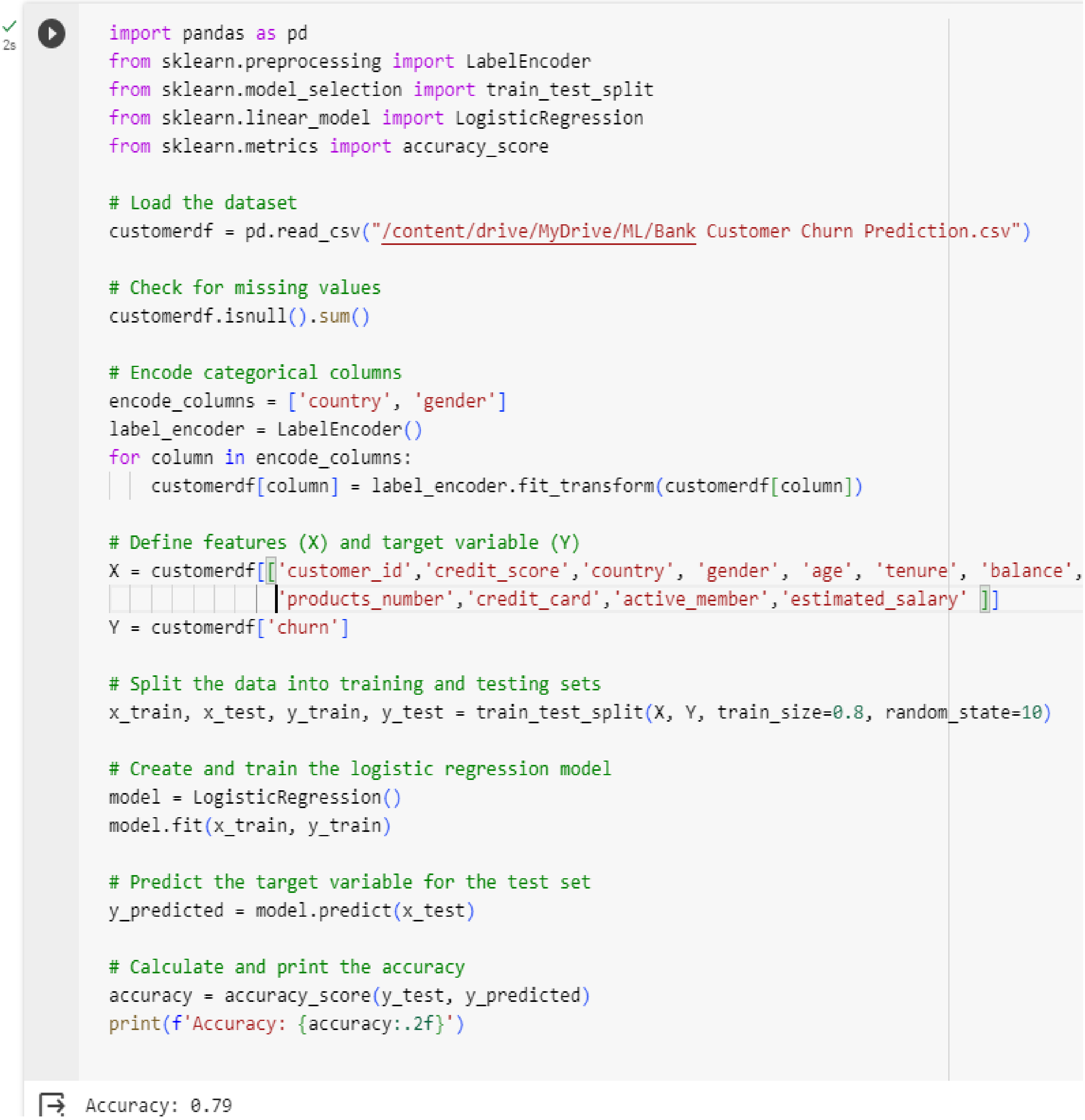
```
R-squared (R2) score: 0.680069137617004
```

# Experiment-4

**Objective:-**To Implement Binary Classification using Logistic Regression in Python

**Dataset:-**https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset

## Implementation :-



```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset
customerdf = pd.read_csv("/content/drive/MyDrive/ML/Bank Customer Churn Prediction.csv")

# Check for missing values
customerdf.isnull().sum()

# Encode categorical columns
encode_columns = ['country', 'gender']
label_encoder = LabelEncoder()
for column in encode_columns:
    customerdf[column] = label_encoder.fit_transform(customerdf[column])

# Define features (X) and target variable (Y)
X = customerdf[['customer_id','credit_score','country', 'gender', 'age', 'tenure', 'balance',
               'products_number','credit_card','active_member','estimated_salary' ]]
Y = customerdf['churn']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, Y, train_size=0.8, random_state=10)

# Create and train the logistic regression model
model = LogisticRegression()
model.fit(x_train, y_train)

# Predict the target variable for the test set
y_predicted = model.predict(x_test)

# Calculate and print the accuracy
accuracy = accuracy_score(y_test, y_predicted)
print(f'Accuracy: {accuracy:.2f}')
```

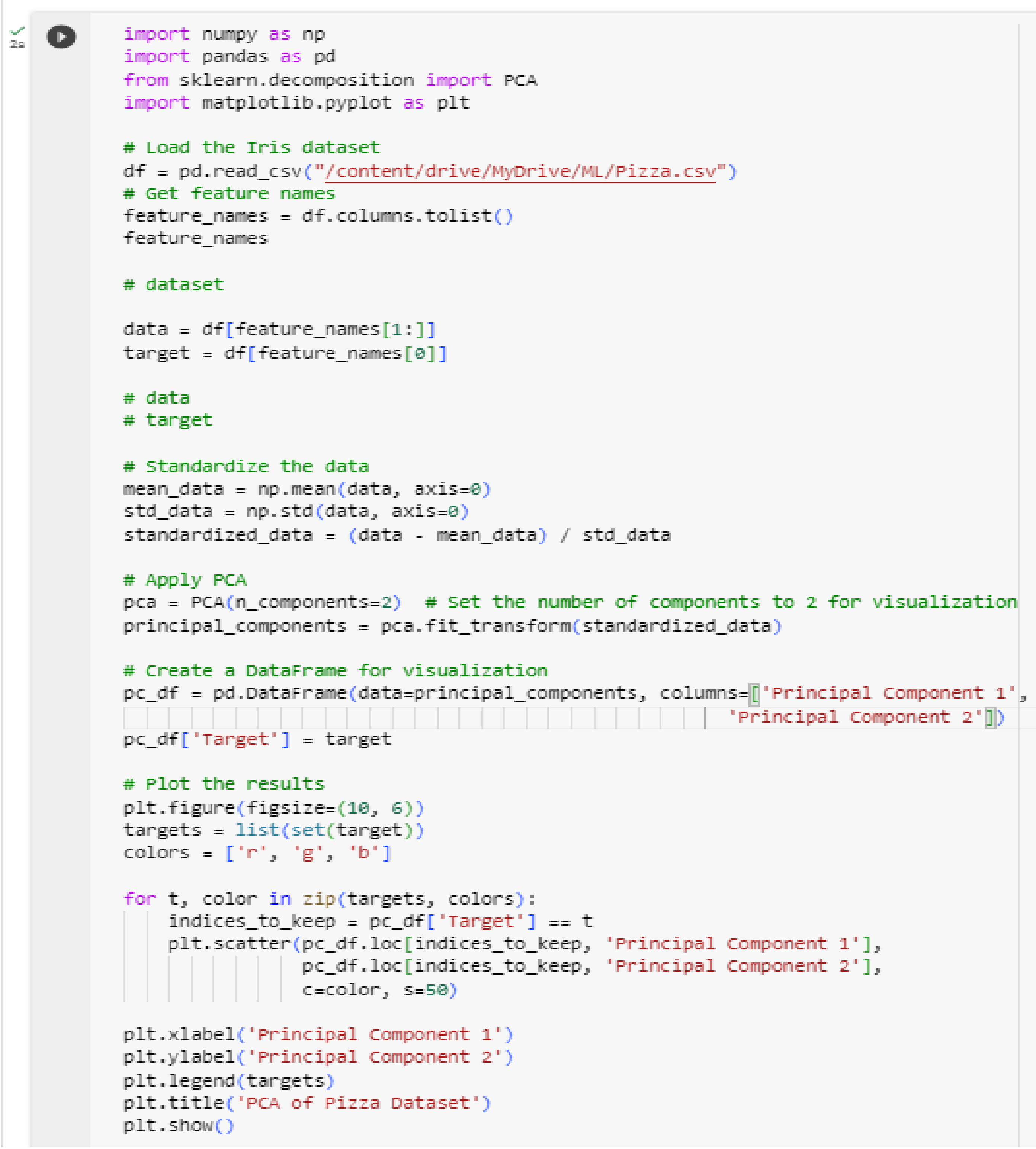
```
Accuracy: 0.79
```

# Experiment-5

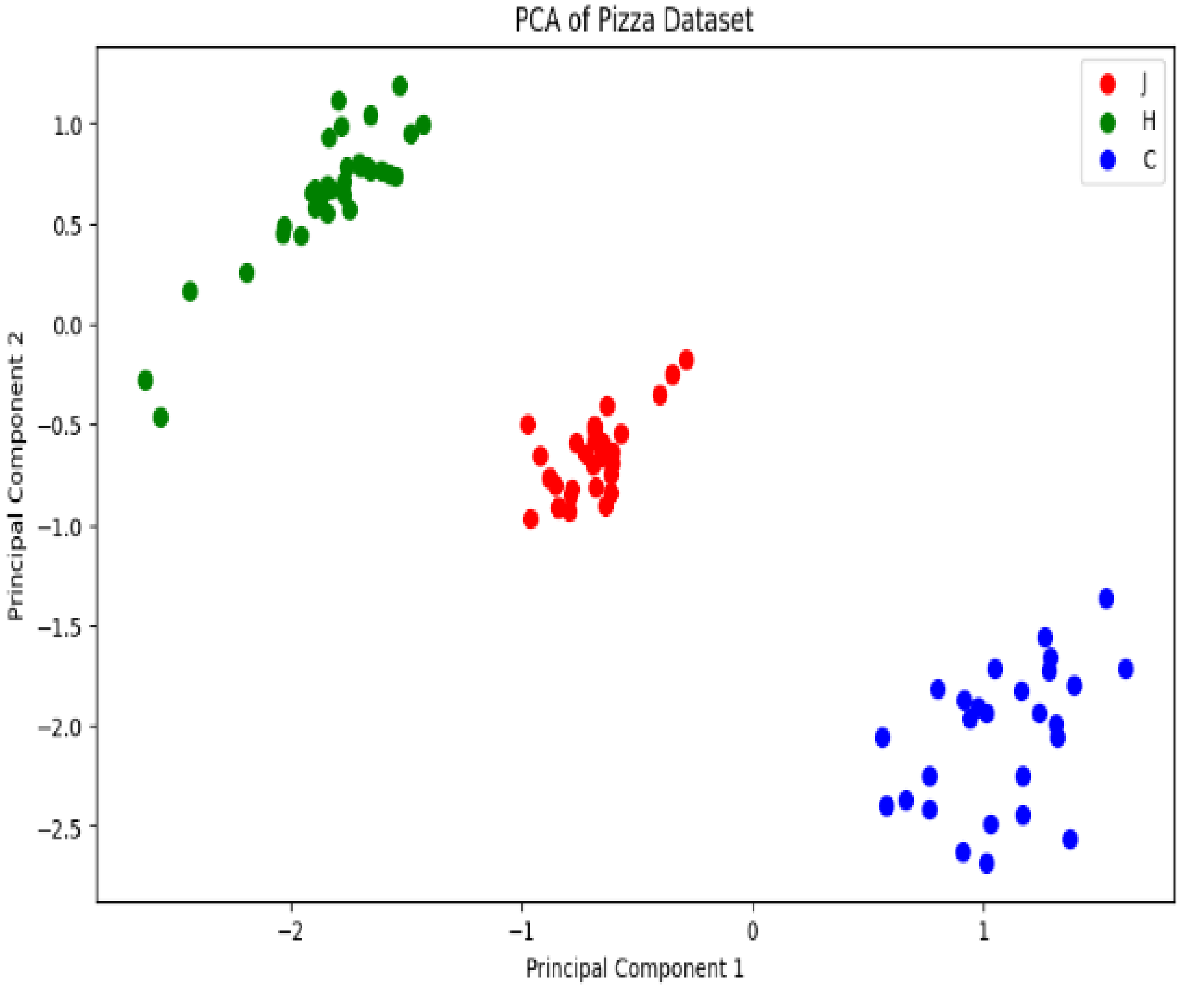
**Objective:-**To Implement Principal Component Analysis in Python

**Dataset:-**https://data.world/sdhilip/pizza-datasets

**Implementation :-**



```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load the Iris dataset
df = pd.read_csv("/content/drive/MyDrive/ML/Pizza.csv")
# Get feature names
feature_names = df.columns.tolist()
feature_names

# dataset

data = df[feature_names[1:]]
target = df[feature_names[0]]

# data
# target

# Standardize the data
mean_data = np.mean(data, axis=0)
std_data = np.std(data, axis=0)
standardized_data = (data - mean_data) / std_data

# Apply PCA
pca = PCA(n_components=2)  # Set the number of components to 2 for visualization
principal_components = pca.fit_transform(standardized_data)

# Create a DataFrame for visualization
pc_df = pd.DataFrame(data=principal_components, columns=['Principal Component 1',
                                                      'Principal Component 2'])
pc_df['Target'] = target

# Plot the results
plt.figure(figsize=(10, 6))
targets = list(set(target))
colors = ['r', 'g', 'b']

for t, color in zip(targets, colors):
    indices_to_keep = pc_df['Target'] == t
    plt.scatter(pc_df.loc[indices_to_keep, 'Principal Component 1'],
                pc_df.loc[indices_to_keep, 'Principal Component 2'],
                c=color, s=50)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(targets)
plt.title('PCA of Pizza Dataset')
plt.show()
```

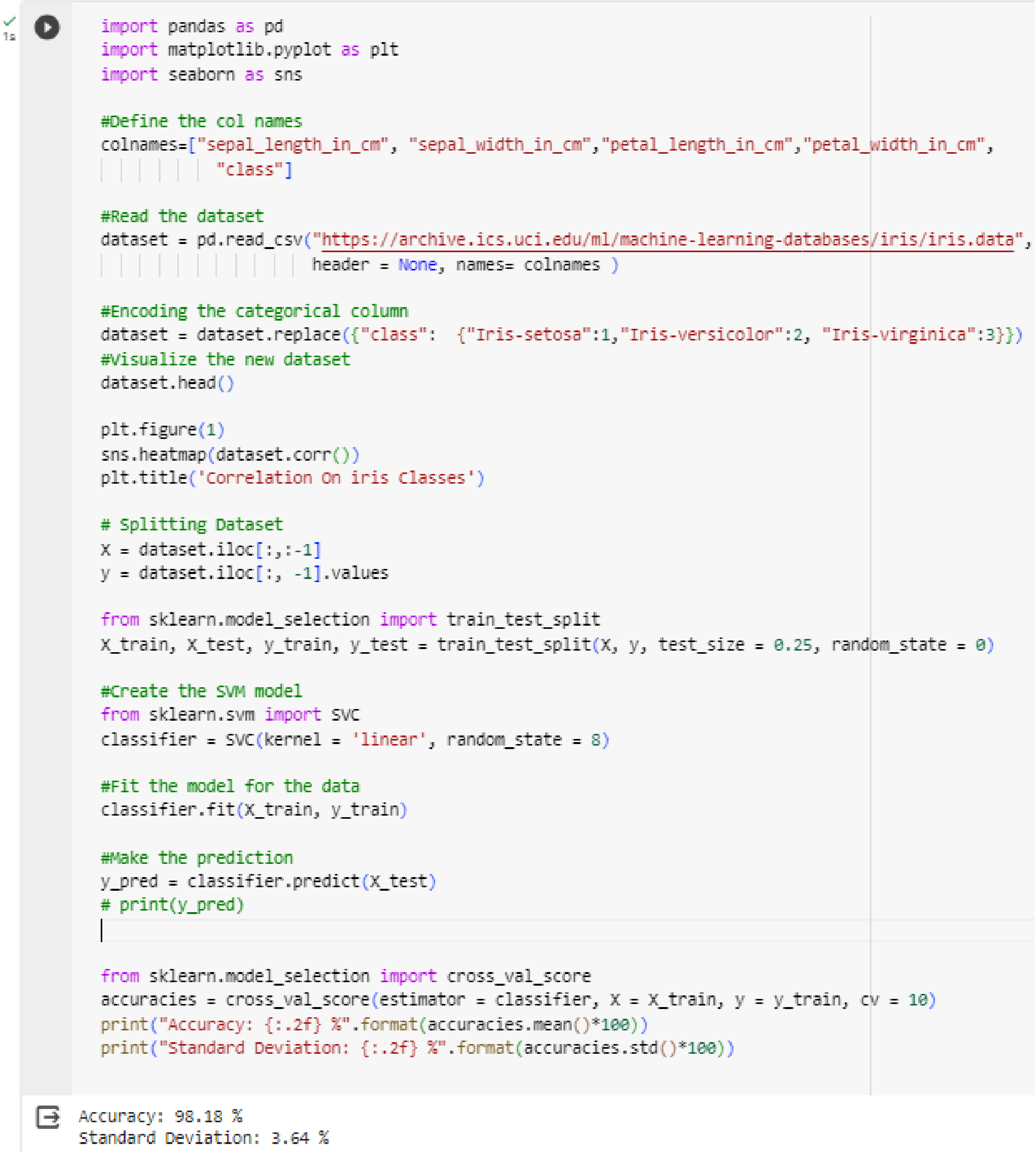

PCA of Pizza Dataset
J
H
C
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
Principal Component 2
−2
−1
0
1
Principal Component 1

# Experiment-6

**Objective:-**To Implement Support Vector Machine Classifier in Python

**Dataset:-**https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

**Implementation :-**



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Define the col names
colnames=["sepal_length_in_cm", "sepal_width_in_cm","petal_length_in_cm","petal_width_in_cm",
            "class"]

#Read the dataset
dataset = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data",
                      header = None, names= colnames )

#Encoding the categorical column
dataset = dataset.replace({"class":  {"Iris-setosa":1,"Iris-versicolor":2, "Iris-virginica":3}})
#Visualize the new dataset
dataset.head()

plt.figure(1)
sns.heatmap(dataset.corr())
plt.title('Correlation On iris Classes')

# Splitting Dataset
X = dataset.iloc[:,:-1]
y = dataset.iloc[:, -1].values

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 = 0)

#Create the SVM model
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 8)

#Fit the model for the data
classifier.fit(X_train, y_train)

#Make the prediction
y_pred = classifier.predict(X_test)
# print(y_pred)


from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

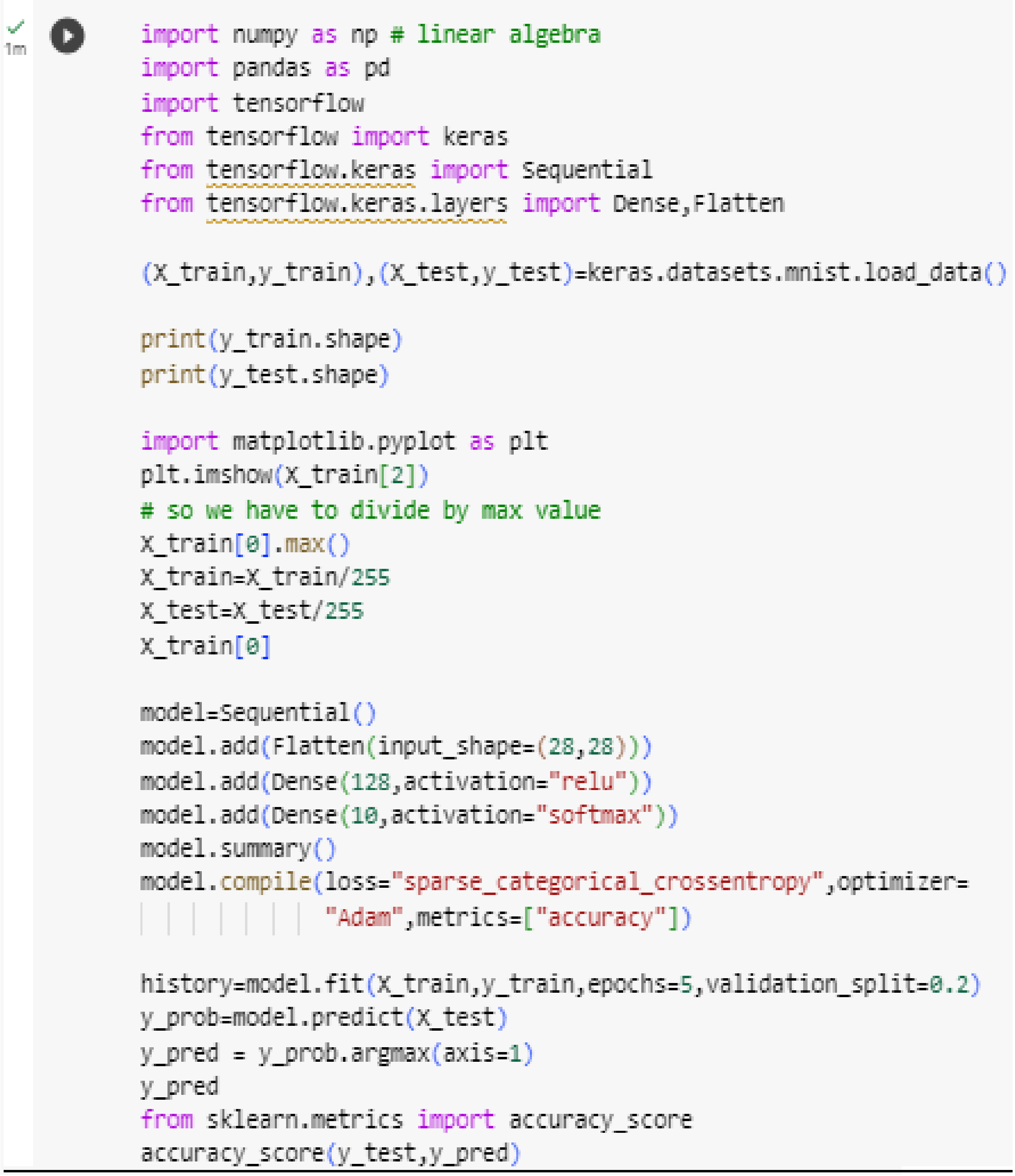
```
Accuracy: 98.18 %
Standard Deviation: 3.64 %
```

# Experiment-7

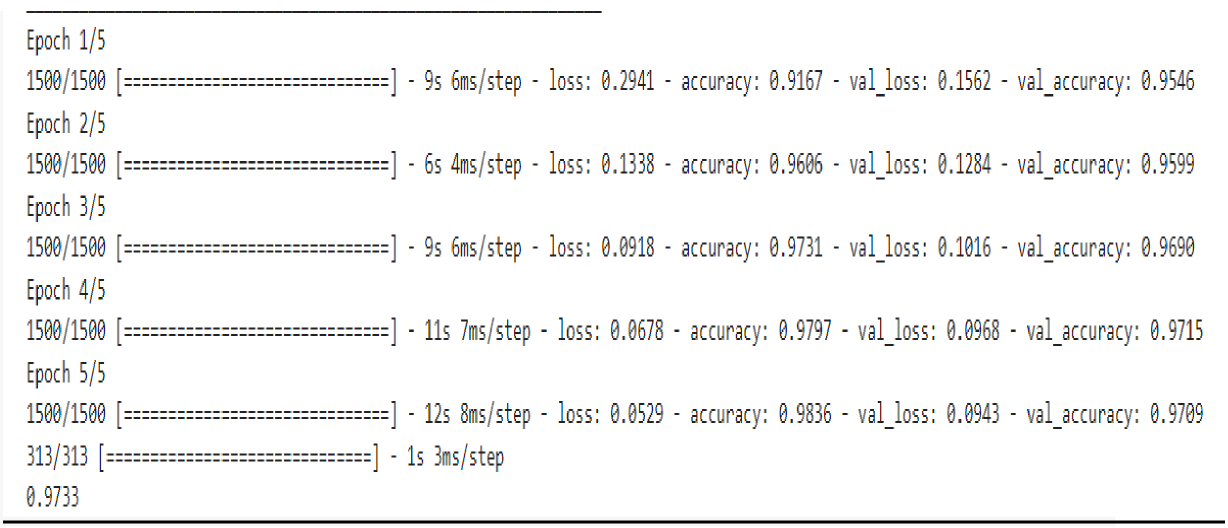
**Objective:-**To Implement Multi-Classification using Artificial Neural Network in Python

**Dataset:-**https://www.kaggle.com/datasets/hojjatk/mnist-dataset

**Implementation :-**



```
import numpy as np # linear algebra
import pandas as pd
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten

(X_train,y_train),(X_test,y_test)=keras.datasets.mnist.load_data()

print(y_train.shape)
print(y_test.shape)

import matplotlib.pyplot as plt
plt.imshow(X_train[2])
# so we have to divide by max value
X_train[0].max()
X_train=X_train/255
X_test=X_test/255
X_train[0]

model=Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128,activation="relu"))
model.add(Dense(10,activation="softmax"))
model.summary()
model.compile(loss="sparse_categorical_crossentropy",optimizer=
              "Adam",metrics=["accuracy"])

history=model.fit(X_train,y_train,epochs=5,validation_split=0.2)
y_prob=model.predict(X_test)
y_pred = y_prob.argmax(axis=1)
y_pred
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```



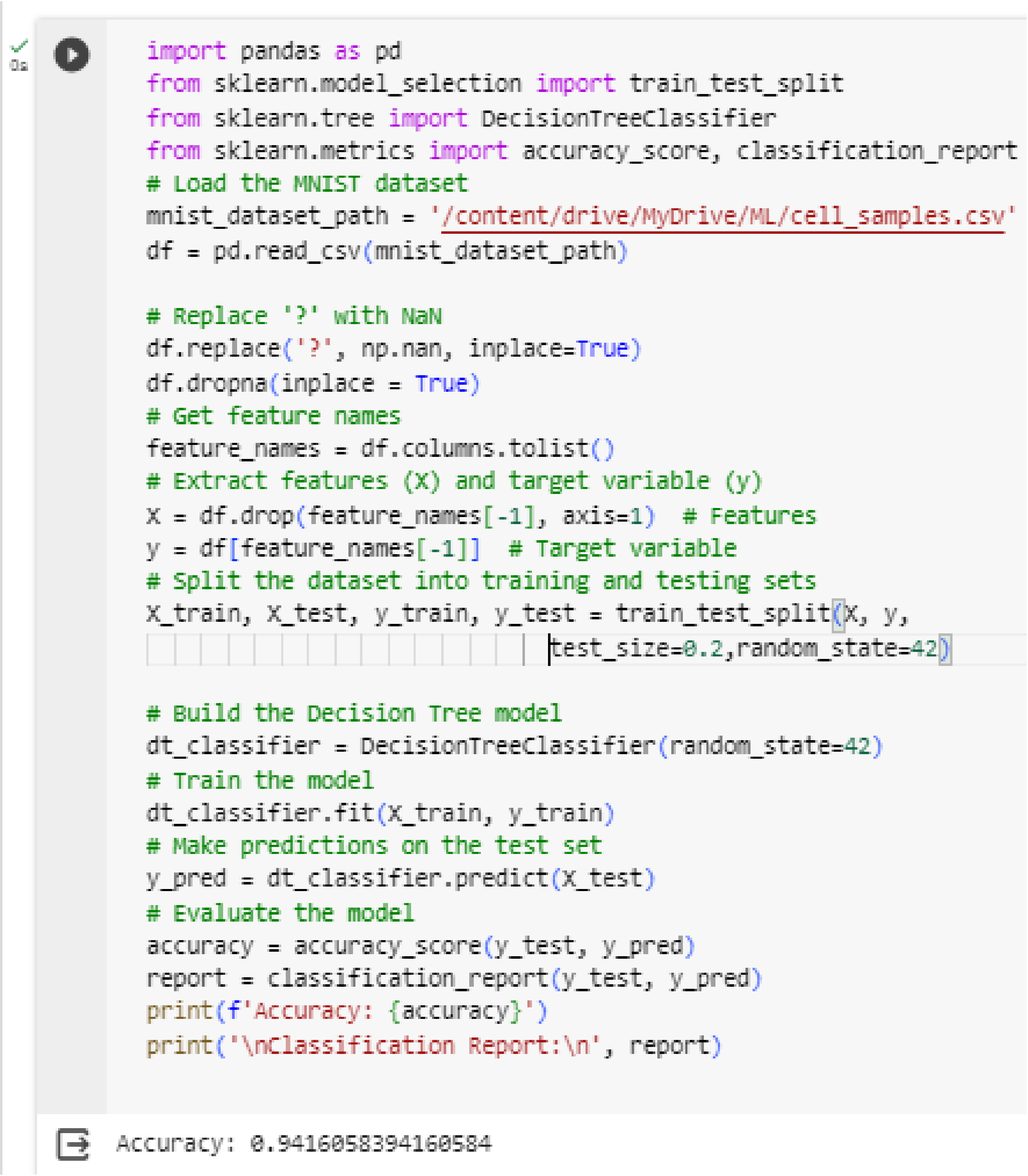
```
_________________________________________________________________
Epoch 1/5
1500/1500 [==============================] - 9s 6ms/step - loss: 0.2941 - accuracy: 0.9167 - val_loss: 0.1562 - val_accuracy: 0.9546
Epoch 2/5
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1338 - accuracy: 0.9606 - val_loss: 0.1284 - val_accuracy: 0.9599
Epoch 3/5
1500/1500 [==============================] - 9s 6ms/step - loss: 0.0918 - accuracy: 0.9731 - val_loss: 0.1016 - val_accuracy: 0.9690
Epoch 4/5
1500/1500 [==============================] - 11s 7ms/step - loss: 0.0678 - accuracy: 0.9797 - val_loss: 0.0968 - val_accuracy: 0.9715
Epoch 5/5
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0529 - accuracy: 0.9836 - val_loss: 0.0943 - val_accuracy: 0.9709
313/313 [==============================] - 1s 3ms/step
0.9733
```

# <u>Experiment-8</u>

**<u>Objective:-</u>** To Implement Decision Tree (DT) classification in Python

**<u>Dataset:-</u>**https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/cell_samples.csv

**<u>Implementation :-</u>**



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load the MNIST dataset
mnist_dataset_path = '/content/drive/MyDrive/ML/cell_samples.csv'
df = pd.read_csv(mnist_dataset_path)

# Replace '?' with NaN
df.replace('?', np.nan, inplace=True)
df.dropna(inplace = True)
# Get feature names
feature_names = df.columns.tolist()
# Extract features (X) and target variable (y)
X = df.drop(feature_names[-1], axis=1)  # Features
y = df[feature_names[-1]]  # Target variable
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                   test_size=0.2,random_state=42)

# Build the Decision Tree model
dt_classifier = DecisionTreeClassifier(random_state=42)
# Train the model
dt_classifier.fit(X_train, y_train)
# Make predictions on the test set
y_pred = dt_classifier.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', report)
```

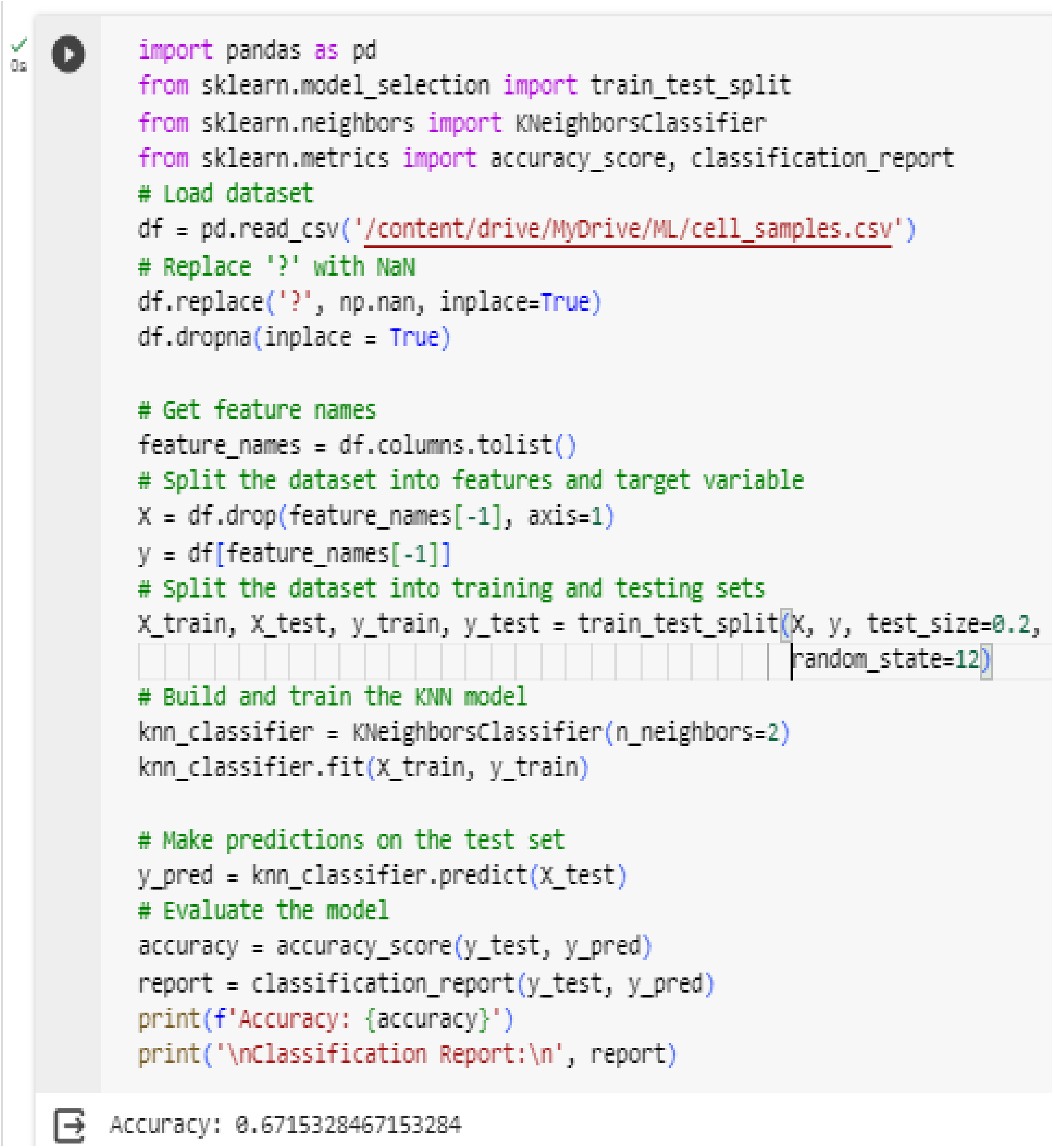
```
Accuracy: 0.9416058394160584
```

# Experiment-9

**Objective:-** To Implement K-Nearest Neighbor (KNN) in Python

**Dataset:-**https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/cell_samples.csv

**Implementation :-**



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load dataset
df = pd.read_csv('/content/drive/MyDrive/ML/cell_samples.csv')
# Replace '?' with NaN
df.replace('?', np.nan, inplace=True)
df.dropna(inplace = True)

# Get feature names
feature_names = df.columns.tolist()
# Split the dataset into features and target variable
X = df.drop(feature_names[-1], axis=1)
y = df[feature_names[-1]]
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=12)
# Build and train the KNN model
knn_classifier = KNeighborsClassifier(n_neighbors=2)
knn_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn_classifier.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', report)
```

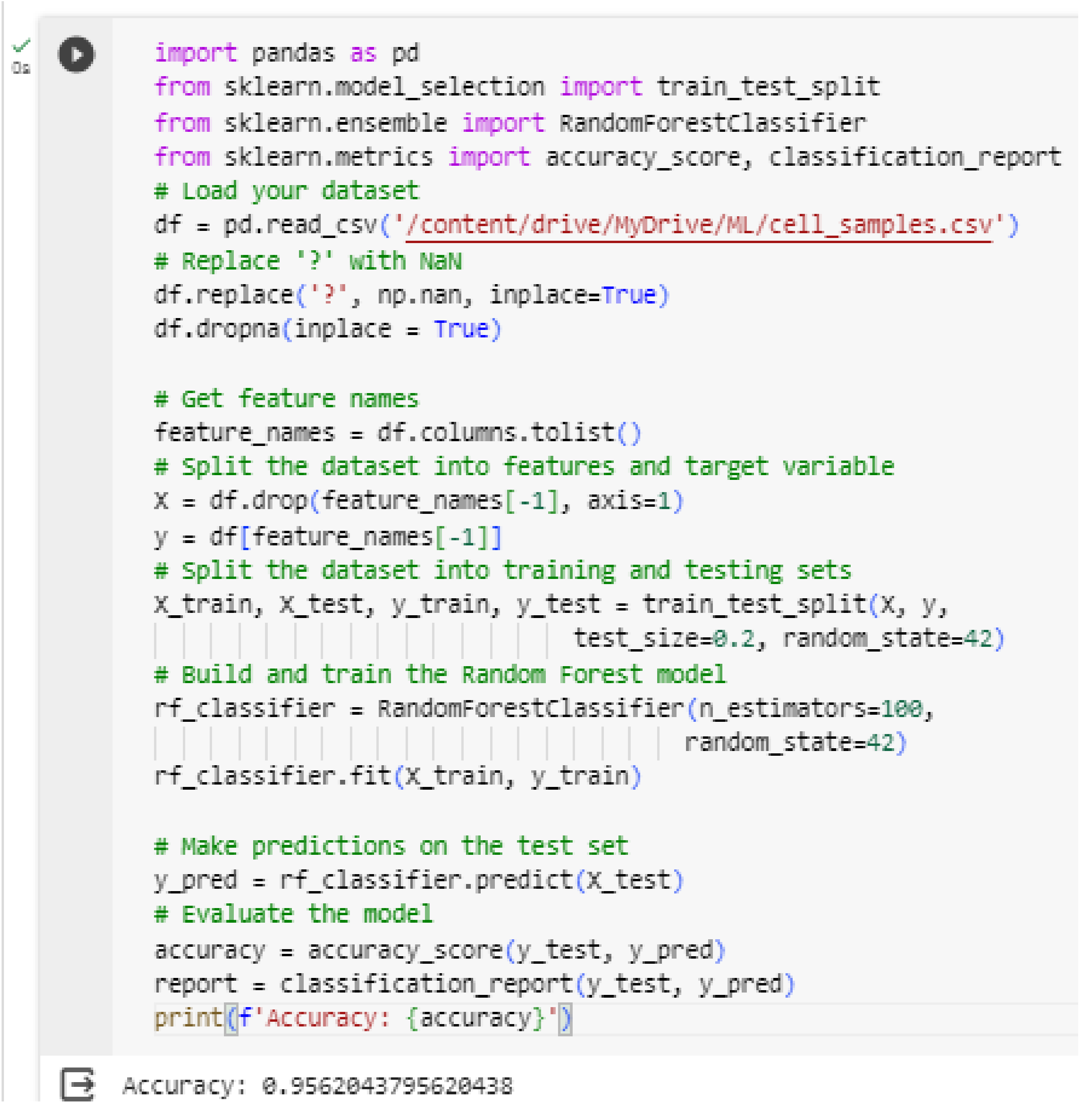
```
Accuracy: 0.6715328467153284
```

# Experiment-10

**Objective:-** To Implement Random Forest in Python

**Dataset:-**https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/cell_samples.csv

**Implementation :-**



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load your dataset
df = pd.read_csv('/content/drive/MyDrive/ML/cell_samples.csv')
# Replace '?' with NaN
df.replace('?', np.nan, inplace=True)
df.dropna(inplace = True)

# Get feature names
feature_names = df.columns.tolist()
# Split the dataset into features and target variable
X = df.drop(feature_names[-1], axis=1)
y = df[feature_names[-1]]
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=42)
# Build and train the Random Forest model
rf_classifier = RandomForestClassifier(n_estimators=100,
                                       random_state=42)
rf_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

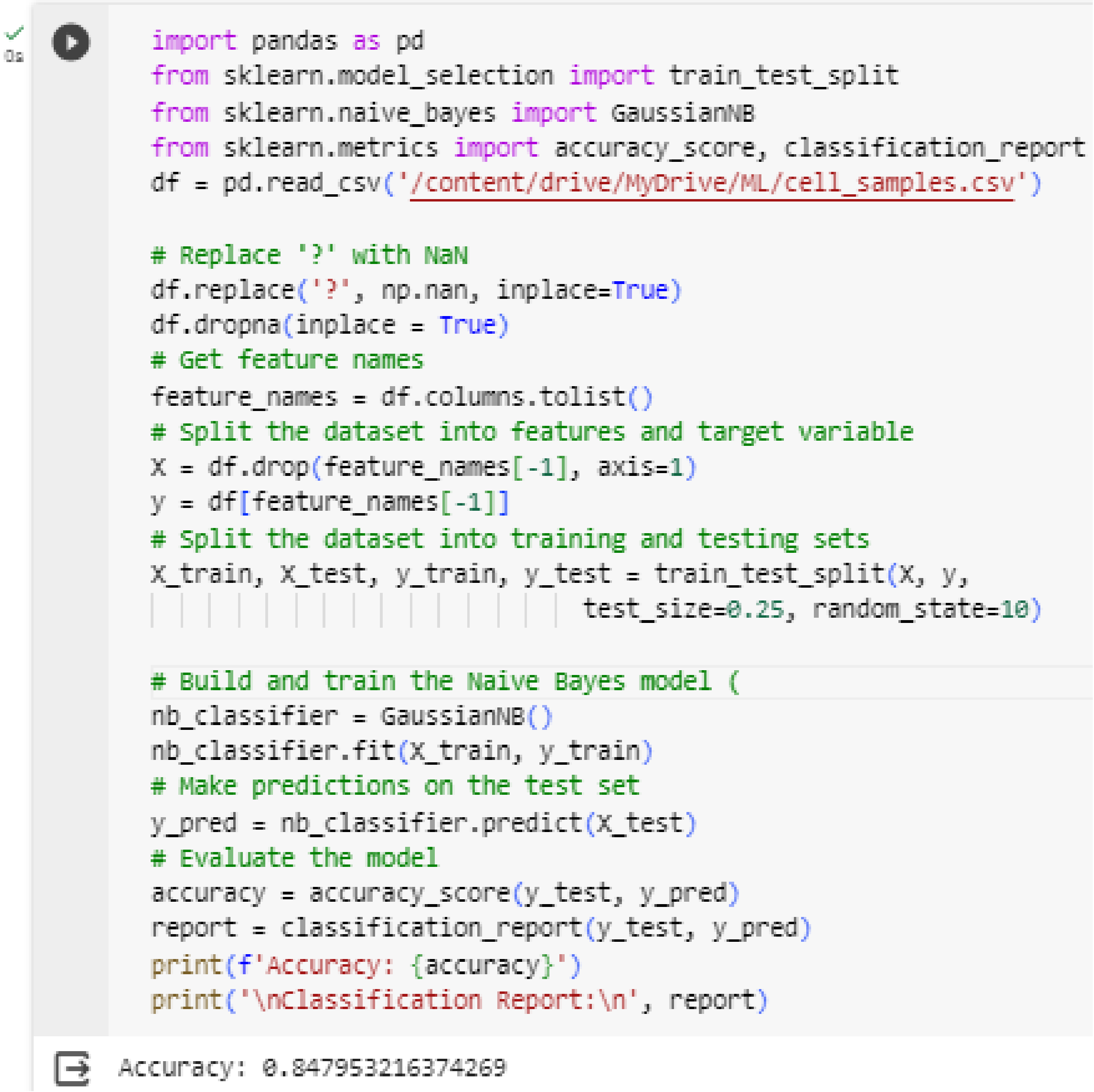
```
Accuracy: 0.9562043795620438
```

# Experiment-11

**Objective:-** To Implement Naïve Bayes Classifier (NB) in Python

**Dataset:-**https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/cell_samples.csv

**Implementation :-**



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
df = pd.read_csv('/content/drive/MyDrive/ML/cell_samples.csv')

# Replace '?' with NaN
df.replace('?', np.nan, inplace=True)
df.dropna(inplace = True)
# Get feature names
feature_names = df.columns.tolist()
# Split the dataset into features and target variable
X = df.drop(feature_names[-1], axis=1)
y = df[feature_names[-1]]
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25, random_state=10)

# Build and train the Naive Bayes model (
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
# Make predictions on the test set
y_pred = nb_classifier.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', report)
```

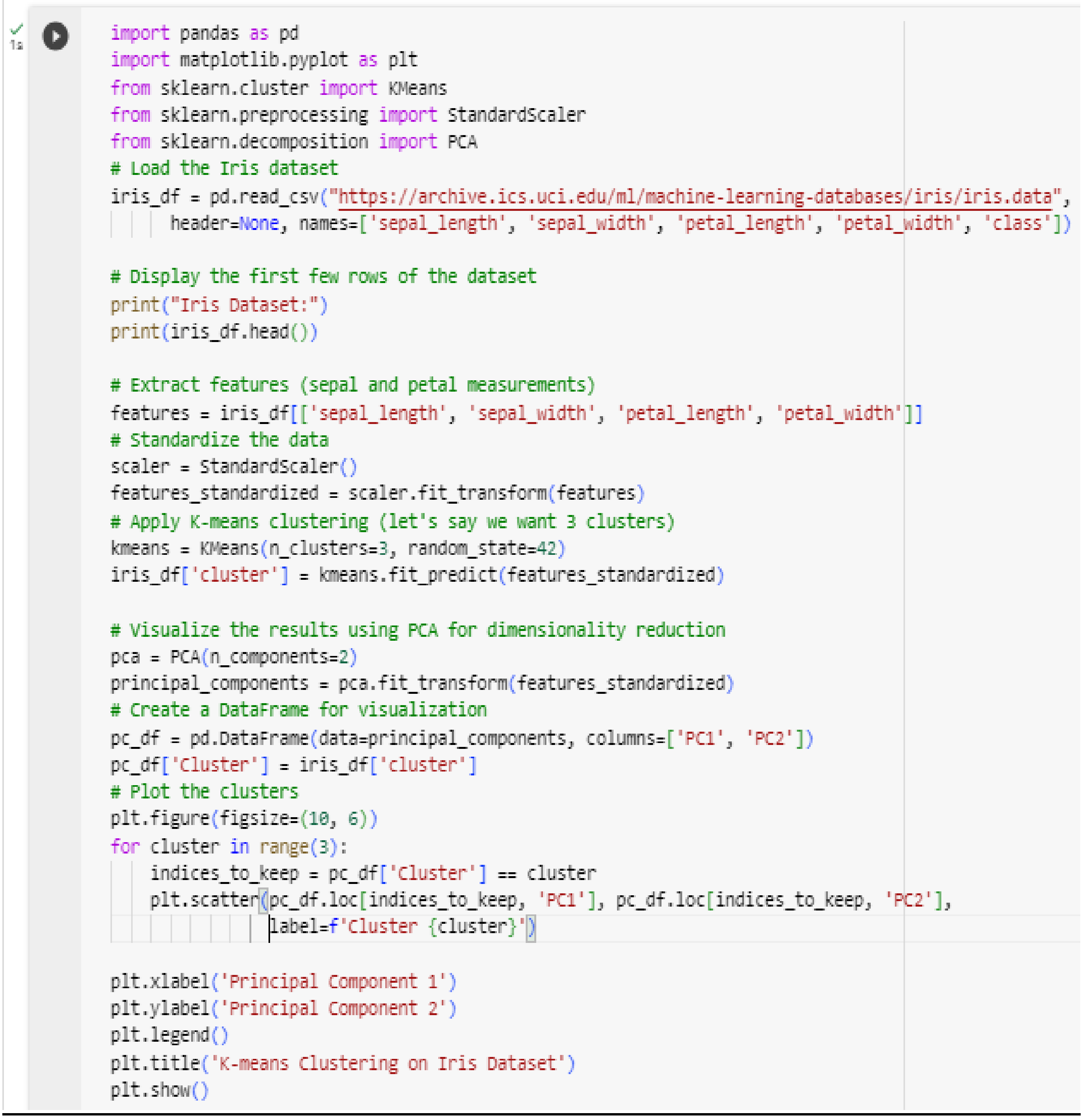
```
Accuracy: 0.847953216374269
```

# Experiment-12

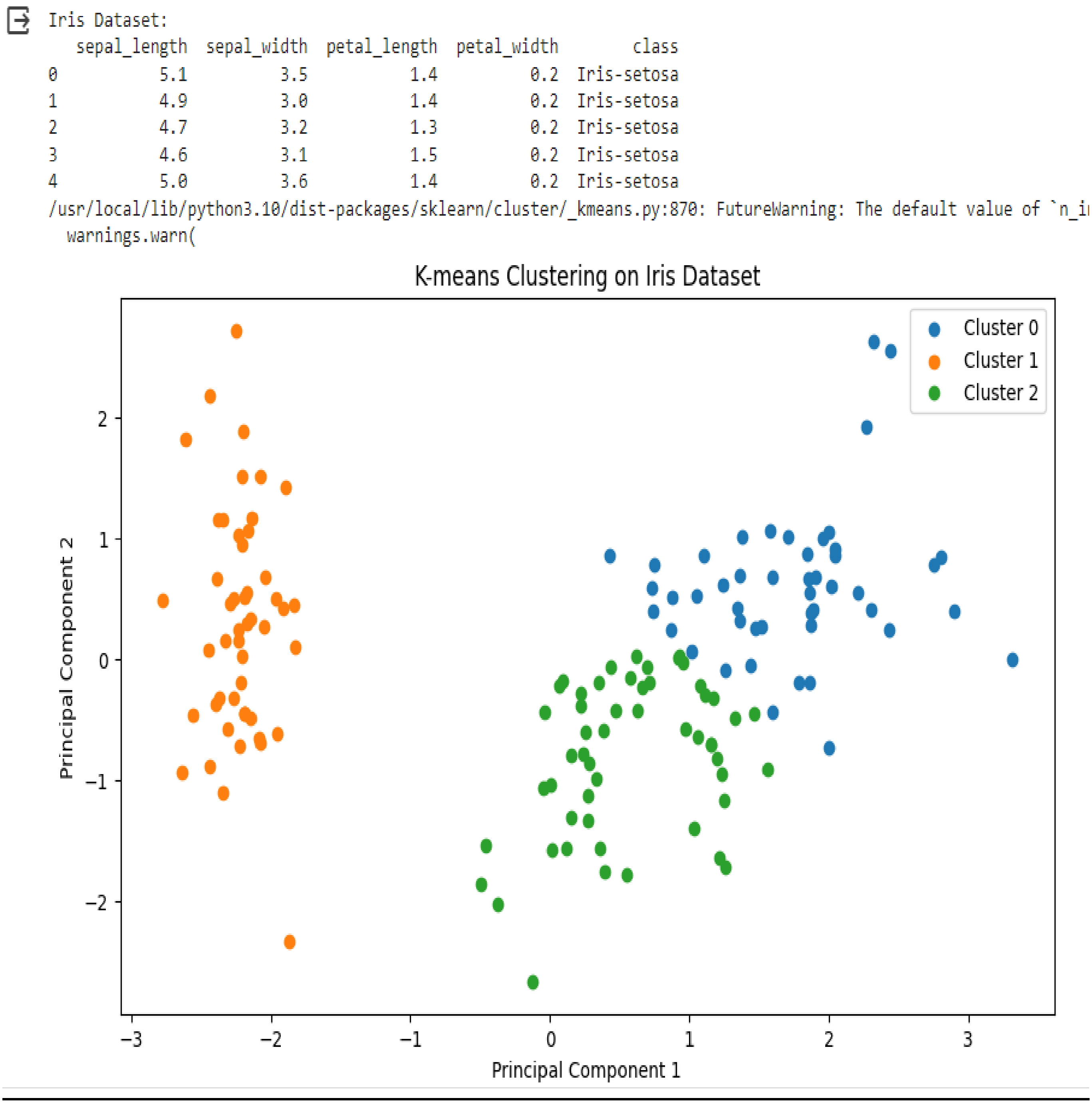
**Objective:-** To Implement K-means Clustering in Python

**Dataset:-**https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

**Implementation :-**



```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# Load the Iris dataset
iris_df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data",
      header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])

# Display the first few rows of the dataset
print("Iris Dataset:")
print(iris_df.head())

# Extract features (sepal and petal measurements)
features = iris_df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
# Standardize the data
scaler = StandardScaler()
features_standardized = scaler.fit_transform(features)
# Apply K-means clustering (let's say we want 3 clusters)
kmeans = KMeans(n_clusters=3, random_state=42)
iris_df['cluster'] = kmeans.fit_predict(features_standardized)

# Visualize the results using PCA for dimensionality reduction
pca = PCA(n_components=2)
principal_components = pca.fit_transform(features_standardized)
# Create a DataFrame for visualization
pc_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
pc_df['Cluster'] = iris_df['cluster']
# Plot the clusters
plt.figure(figsize=(10, 6))
for cluster in range(3):
    indices_to_keep = pc_df['Cluster'] == cluster
    plt.scatter(pc_df.loc[indices_to_keep, 'PC1'], pc_df.loc[indices_to_keep, 'PC2'],
                label=f'Cluster {cluster}')

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.title('K-means Clustering on Iris Dataset')
plt.show()
```



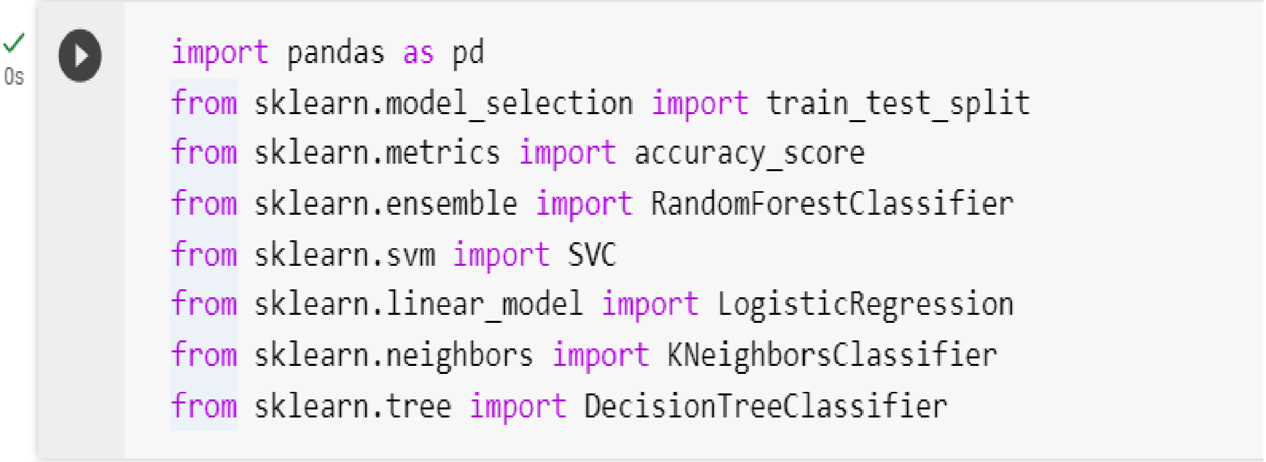
```
Iris Dataset:
   sepal_length  sepal_width  petal_length  petal_width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_i
  warnings.warn(
```

# Project

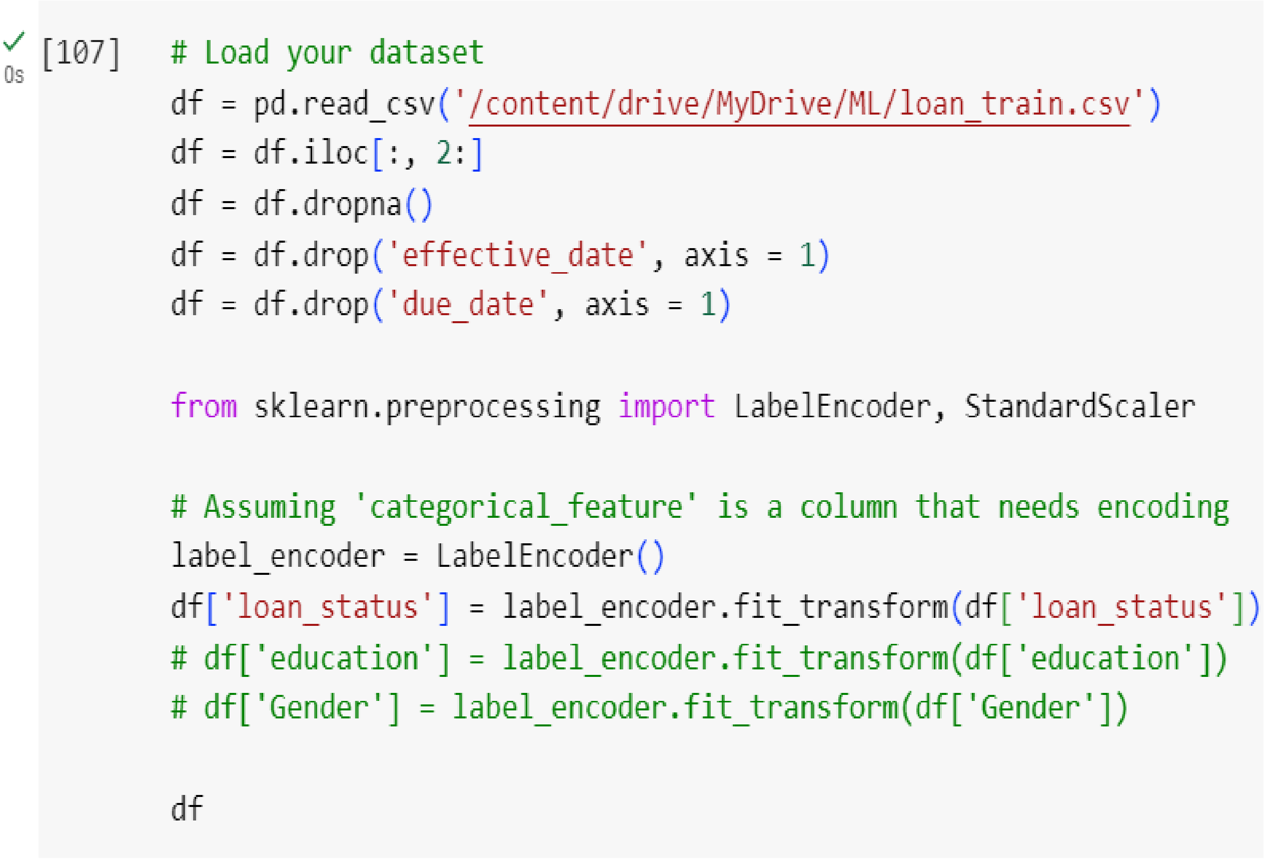
**Objective:-**Classify the loan status using various classification algorithms and their comparison.

**Dataset:-**https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/FinalModule_Coursera/data/loan_train.csv

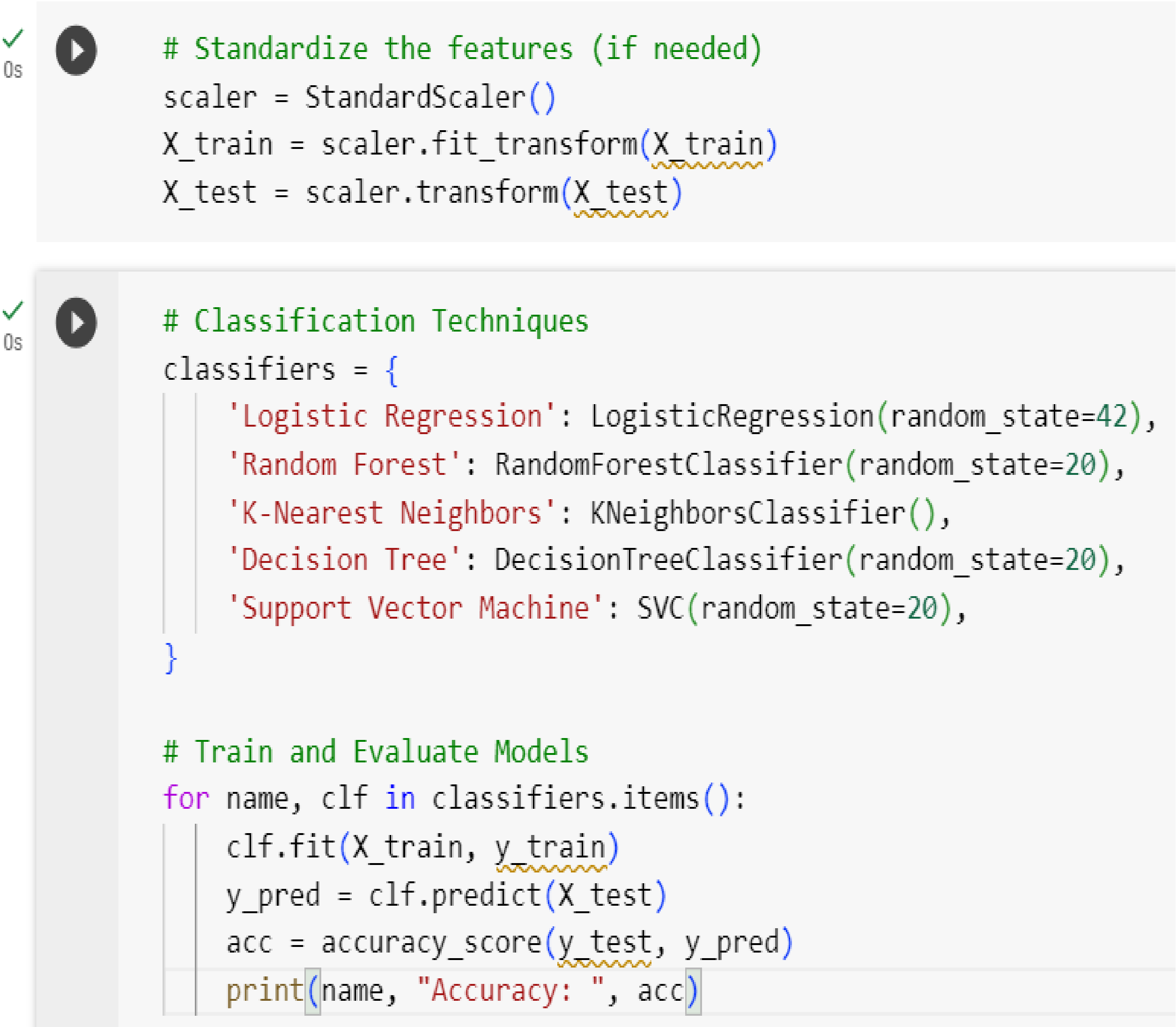
**Implementation :-**



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

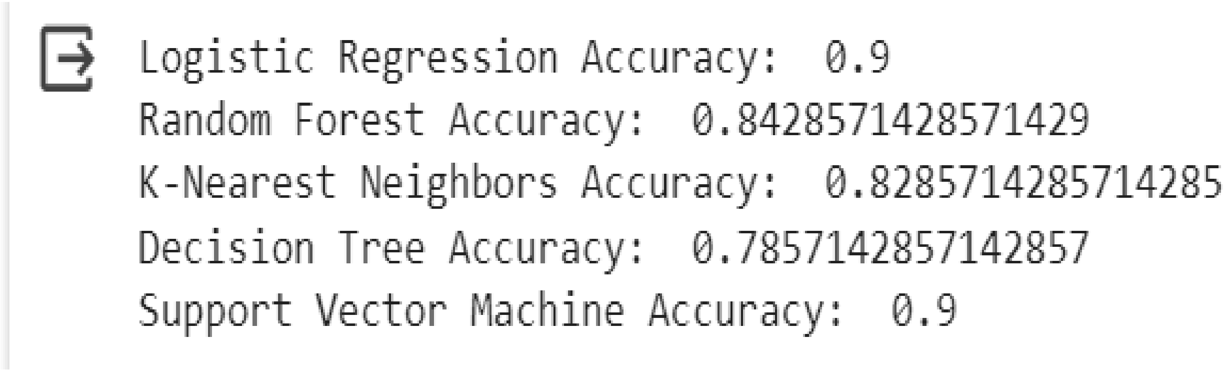


```
# Load your dataset
df = pd.read_csv('/content/drive/MyDrive/ML/loan_train.csv')
df = df.iloc[:, 2:]
df = df.dropna()
df = df.drop('effective_date', axis = 1)
df = df.drop('due_date', axis = 1)

from sklearn.preprocessing import LabelEncoder, StandardScaler

# Assuming 'categorical_feature' is a column that needs encoding
label_encoder = LabelEncoder()
df['loan_status'] = label_encoder.fit_transform(df['loan_status'])
# df['education'] = label_encoder.fit_transform(df['education'])
# df['Gender'] = label_encoder.fit_transform(df['Gender'])

df
```



```python
# Standardize the features (if needed)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```python
# Classification Techniques
classifiers = {
    'Logistic Regression': LogisticRegression(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=20),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(random_state=20),
    'Support Vector Machine': SVC(random_state=20),
}

# Train and Evaluate Models
for name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(name, "Accuracy: ", acc)
```



```
Logistic Regression Accuracy:  0.9
Random Forest Accuracy:  0.8428571428571429
K-Nearest Neighbors Accuracy:  0.8285714285714285
Decision Tree Accuracy:  0.7857142857142857
Support Vector Machine Accuracy:  0.9
```