EXPERIMENT 9B

ADITHIYA V

241501010

A PYTHON PROGRAM TO IMPLEMENT K MEAN MODEL

AIM:

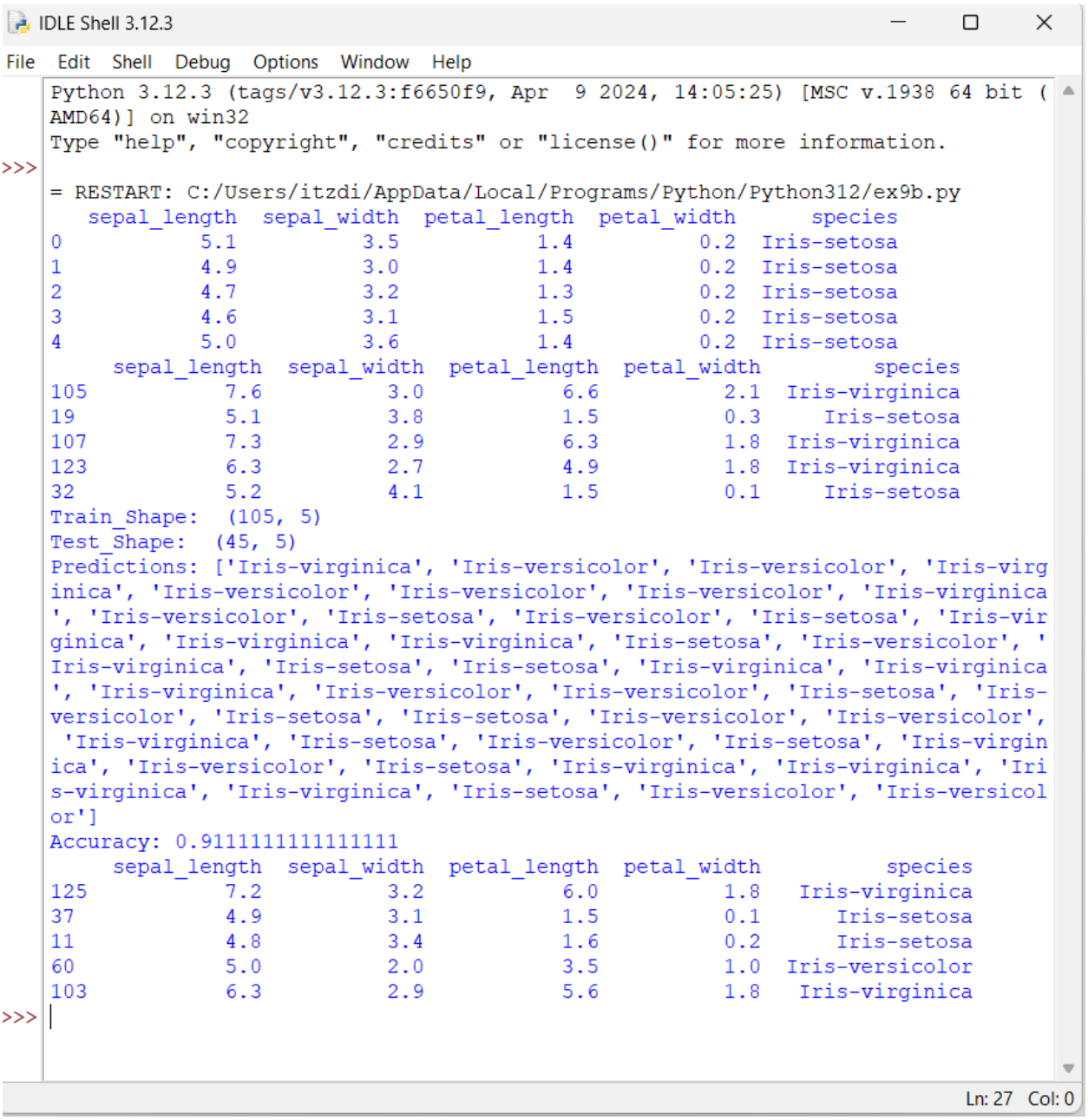
TO IMPLEMENT A PYTHON PROGRAM TO IMPLEMENT KMEAN MODEL

CODE:

```
import pandas as pd
data = pd.read_csv('/content/IRIS.csv')
data.head(5)

import numpy as np
shuffle_index = np.random.permutation(req_data.shape[0])
#shuffling the row index of our dataset
req_data = req_data.iloc[shuffle_index]
req_data.head(5)

train_size = int(req_data.shape[0]*0.7)
train_df = req_data.iloc[:train_size,:]
test_df = req_data.iloc[train_size:,:]
train = train_df.values
test = test_df.values
y_true = test[:,-1]
print('Train_Shape: ',train_df.shape)
print('Test_Shape: ',test_df.shape)

from math import sqrt
def euclidean_distance(x_test, x_train):
    distance = 0
    for i in range(len(x_test)-1):
        distance += (x_test[i]-x_train[i])**2
    return sqrt(distance)
def get_neighbors(x_test, x_train, num_neighbors):
    distances = []
    data = []
    for i in x_train:
        distances.append(euclidean_distance(x_test,i))
```

```
        data.append(i)
    distances = np.array(distances)
    data = np.array(data)
    sort_indexes = distances.argsort() #argsort() function returns indices by sorting
    #distances data in ascending order
    data = data[sort_indexes] #modifying our data based on sorted indices, so that we
    #can get the nearest neighbors
    return data[:num_neighbors]
def prediction(x_test, x_train, num_neighbors):
    classes = []
    neighbors = get_neighbors(x_test, x_train, num_neighbors)
    for i in neighbors:
        classes.append(i[-1])
    predicted = max(classes, key=classes.count) #taking the most repeated class
    return predicted
def predict_classifier(x_test):
    classes = []
    neighbors = get_neighbors(x_test, req_data.values, 5)
    for i in neighbors:
        classes.append(i[-1])
    predicted = max(classes, key=classes.count)
    print(predicted)
    return predicted
def accuracy(y_true, y_pred):
    num_correct = 0
    for i in range(len(y_true)):
        if y_true[i]==y_pred[i]:
            num_correct+=1
    accuracy = num_correct/len(y_true)
    return accuracy
y_pred = []
for i in test:
    y_pred.append(prediction(i, train, 5))
y_pred

accuracy = accuracy(y_true, y_pred)
accuracy

test_df.sample(5)
```

OUTPUT:



RESULT:

TO IMPLEMENT A PYTHON PROGRAM TO IMPLEMENT K MEAN MODEL AS BEEN ANALYSED AND VERIFIED