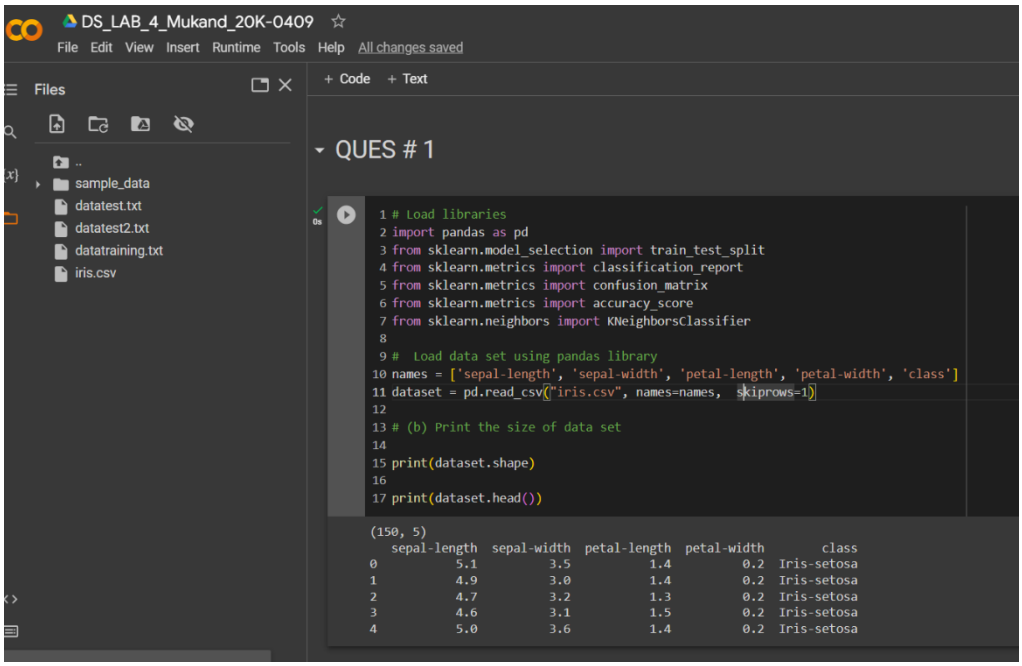
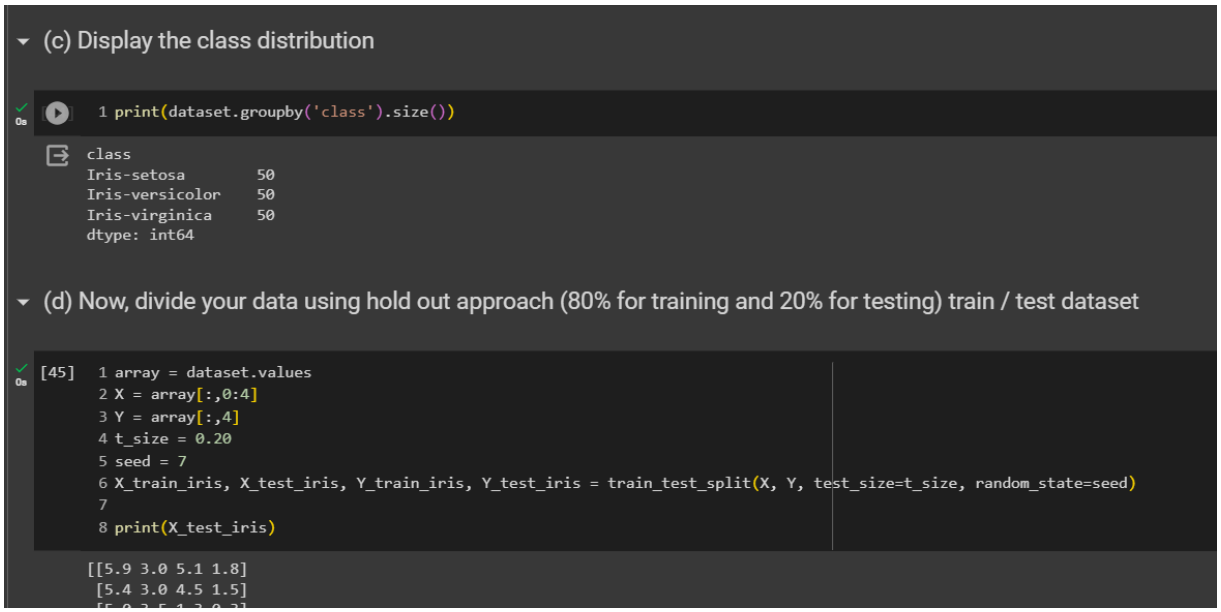
DATA SCIENCE LAB 4

Roll no: 20K-0409

Screen Shots

Ques # 1





make predictions

```
[46]  1 from sklearn.model_selection import train_test_split
      2 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
      3 from sklearn.neighbors import KNeighborsClassifier
      4
      5 knn = KNeighborsClassifier()
      6 knn.fit(X_train_iris, Y_train_iris)
      7
      8 predictions = knn.predict(X_test_iris)
      9
     10 print("Accuracy Score: \n")
     11 print(accuracy_score(Y_test_iris, predictions))
     12
     13 print("\nConfusion Matrix: \n")
     14 print(confusion_matrix(Y_test_iris, predictions))
     15
     16 print("\nClassification : \n")
     17 print(classification_report(Y_test_iris, predictions))
```

```
Accuracy Score:

0.9

Confusion Matrix:

[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]

Classification :

                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11
```

(f) Repeat (e) by changing the value of k (k=1, 2, 3,...., 10). Print only accuracy

```
[47]  1 print("Accuracy for different k values:")
      2 for k in range(1, 11):
      3     knn = KNeighborsClassifier(n_neighbors=k)
      4     knn.fit(X_train_iris, Y_train_iris)
      5     predictions = knn.predict(X_test_iris)
      6     accuracy = accuracy_score(Y_test_iris, predictions)
      7     print(f"k = {k}: Accuracy = {accuracy:.4f}")
      8
```

```
Accuracy for different k values:
k = 1: Accuracy = 0.9000
k = 2: Accuracy = 0.9333
k = 3: Accuracy = 0.9000
k = 4: Accuracy = 0.9333
k = 5: Accuracy = 0.9000
k = 6: Accuracy = 0.8667
k = 7: Accuracy = 0.8667
k = 8: Accuracy = 0.9000
k = 9: Accuracy = 0.9000
k = 10: Accuracy = 0.9000
```

(g) Repeat (e) by changing the value of seed (seed = 1, 2, 3, ... , 10). Print only accuracy

```
[ ]  1 print("\nAccuracy for different seed values:")
     2 for seed_val in range(1, 11):
     3     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=t_size, random_state=seed_val)
     4     knn = KNeighborsClassifier()
     5     knn.fit(X_train, Y_train)
     6     predictions = knn.predict(X_test)
     7     accuracy = accuracy_score(Y_test, predictions)
     8     print(f"Accuracy = {accuracy:.4f}")
     9
```

```
Accuracy for different seed values:
Accuracy = 1.0000
Accuracy = 1.0000
Accuracy = 0.9667
Accuracy = 0.9667
Accuracy = 0.9333
Accuracy = 0.9667
Accuracy = 0.9000
Accuracy = 0.9000
Accuracy = 1.0000
Accuracy = 0.9667
```

Ques # 2

QUES # 2

```
[59] 1 # Load the training dataset
     2 columns = ['date', 'temperature', 'humidity', 'light', 'co2', 'humidity_ratio', 'occupancy']
     3 train_data = pd.read_csv("datatraining.txt", names=columns, header=0)
     4
     5 # Dropping the 'date' attribute
     6 train_data = train_data.drop('date', axis=1)
     7
     8 print(train_data.head())
     9
```

```
   temperature  humidity  light     co2  humidity_ratio  occupancy
1        23.18   27.2720  426.0  721.25        0.004793          1
2        23.15   27.2675  429.5  714.00        0.004783          1
3        23.15   27.2450  426.0  713.50        0.004779          1
4        23.15   27.2000  426.0  708.25        0.004772          1
5        23.10   27.2000  426.0  704.50        0.004757          1
```

```
[60] 1 # (b) size of the dataset
     2 print(train_data.shape)
     3
     4 # (c) class distribution
     5 print("\nClass Dist:")
     6 print(train_data.groupby('occupancy').size())
     7
```

```
(8143, 6)

Class Dist:
occupancy
0    6414
1    1729
dtype: int64
```

(e) Apply kNN classifier

```
[66] 1 knn = KNeighborsClassifier()
     2 knn.fit(X_trainOc, Y_trainOc)
     3
     4 predictions = knn.predict(X_testOc)
     5
     6 print("Accuracy Score: \n")
     7 print(accuracy_score(Y_testOc, predictions))
     8
     9 print("\nConfusion Matrix: \n")
    10 print(confusion_matrix(Y_testOc, predictions))
    11
    12 print("\nClassification : \n")
    13 print(classification_report(Y_testOc, predictions))
```

```
Accuracy Score:

0.9579608601111379

Confusion Matrix:

[[9030  366]
 [ 156 2865]]

Classification :

              precision    recall  f1-score   support

           0       0.98      0.96      0.97      9396
           1       0.89      0.95      0.92      3021

    accuracy                           0.96     12417
   macro avg       0.93      0.95      0.94     12417
```

(f) calculating accuracy

```
     1
     2 print("Accuracy for different k values:")
     3 for k in range(1, 11):
     4     knn = KNeighborsClassifier(n_neighbors=k)
     5     knn.fit(X_trainOc, Y_trainOc)
     6     predictions = knn.predict(X_testOc)
     7     accuracy = accuracy_score(Y_testOc, predictions)
     8     print(f"k = {k}: Accuracy = {accuracy:.4f}")
     9
```

```
Accuracy for different k values:
k = 1: Accuracy = 0.9473
k = 2: Accuracy = 0.9462
k = 3: Accuracy = 0.9530
k = 4: Accuracy = 0.9483
k = 5: Accuracy = 0.9580
k = 6: Accuracy = 0.9564
k = 7: Accuracy = 0.9641
k = 8: Accuracy = 0.9635
k = 9: Accuracy = 0.9648
k = 10: Accuracy = 0.9644
```

Ques#3

QUES # 3

Counter : for counting occurrences of items in a list. In kNN algorithm, Counter is used to identify the most common label among the top k neighbors.

```python
import numpy as np
import pandas as pd
from collections import Counter

# Chi-squared distance function
def chi_squared_distance(A, B):
    return 0.5 * np.sum([((a - b) ** 2) / (a + b)
                         for (a, b) in zip(A, B)])

# kNN classifier
def knn_classify(X_train, y_train, test_point, k=3):
    distances = [(y, chi_squared_distance(X, test_point)) for X, y in zip(X_train, y_train)]
    distances.sort(key=lambda x: x[1])
    top_k_labels = [item[0] for item in distances[:k]]
    most_common = Counter(top_k_labels).most_common(1)
    return most_common[0][0]
```

```python
# using Iris dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pd.read_csv("iris.csv", names=names,  skiprows=1)

array = dataset.values
X = array[:,0:4]
Y = array[:,4]
t_size = 0.20
seed = 7
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=t_size, random_state=seed)

# Testing kNN classifier on Iris
print("Accuracy for different k values:")
for k in range(1, 11):
    y_pred = [knn_classify(X_train, Y_train, x, k=k) for x in X_test]
    accuracy = np.mean(y_pred == Y_test)
    print(f"k = {k}: Accuracy = {accuracy:.4f}")
```

```
Accuracy for different k values:
k = 1: Accuracy = 0.9000
k = 2: Accuracy = 0.9000
k = 3: Accuracy = 0.9000
k = 4: Accuracy = 0.8667
k = 5: Accuracy = 0.8667
k = 6: Accuracy = 0.8667
k = 7: Accuracy = 0.8667
k = 8: Accuracy = 0.8667
k = 9: Accuracy = 0.9000
k = 10: Accuracy = 0.9000
```

for occupancy

```python
# Using a smaller subset for quick verification
X_test_sample = X_testOc[:100]
Y_test_sample = Y_testOc[:100]

print("Accuracy for different k values for kNN classifier:")

for k in range(1, 11):
    y_pred = [knn_classify(X_trainOc, Y_trainOc, x, k=k) for x in X_test_sample]
    accuracy = np.mean(y_pred == Y_test_sample)
    print(f"k = {k}: Accuracy = {accuracy:.4f}")
```

```
Accuracy for different k values for kNN classifier:
k = 1: Accuracy = 0.9200
k = 2: Accuracy = 0.9200
k = 3: Accuracy = 0.9000
k = 4: Accuracy = 0.9100
k = 5: Accuracy = 0.9400
k = 6: Accuracy = 0.9500
k = 7: Accuracy = 0.9500
k = 8: Accuracy = 0.9600
k = 9: Accuracy = 0.9500
k = 10: Accuracy = 0.9600
```