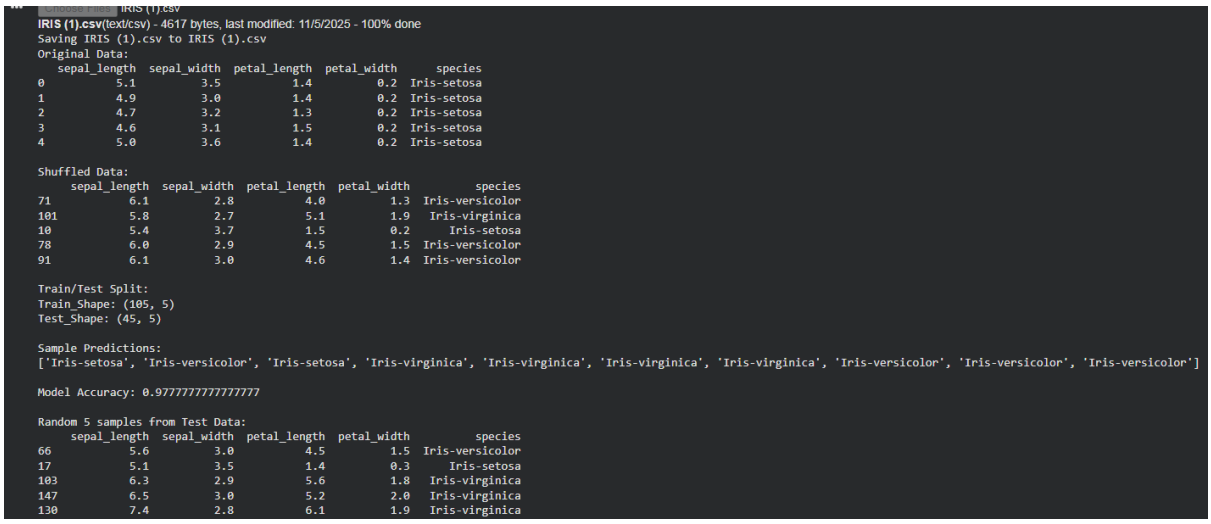
```
# 🧪 KNN Implementation on IRIS Dataset
from google.colab import files
uploaded = files.upload()
import pandas as pd
import numpy as np
from math import sqrt

# Step 1: Load dataset
data = pd.read_csv('/content/IRIS (1).csv')
print("Original Data:")
print(data.head(5))

# Step 2: Shuffle dataset
req_data = data.copy()
shuffle_index = np.random.permutation(req_data.shape[0])
req_data = req_data.iloc[shuffle_index]
print("\nShuffled Data:")
print(req_data.head(5))

# Step 3: Split into train and test sets (70-30)
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("\nTrain/Test Split:")
print('Train_Shape:', train_df.shape)
print('Test_Shape:', test_df.shape)
```

```
# Step 4: Define helper functions for KNN
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 = []
    for i in x_train:
        dist = euclidean_distance(x_test, i)
        distances.append(dist)
    distances = np.array(distances)
    sort_indexes = distances.argsort()
    data = x_train[sort_indexes]
    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)
    return predicted

def accuracy(y_true, y_pred):
    num_correct = sum(y_true[i] == y_pred[i] for i in range(len(y_true)))
    return num_correct / len(y_true)

# Step 5: Predict using KNN (k=5)
```

```
y_pred = []
for i in test:
   y_pred.append(prediction(i, train, 5))

print("\nSample Predictions:")
print(y_pred[:10])

# Step 6: Compute accuracy
acc = accuracy(y_true, y_pred)
print("\nModel Accuracy:", acc)

# Step 7: Display random sample of test data
print("\nRandom 5 samples from Test Data:")
print(test_df.sample(5))
```



```
Choose Files  IRIS (1).csv
IRIS (1).csv(text/csv) - 4617 bytes, last modified: 11/5/2025 - 100% done
Saving IRIS (1).csv to IRIS (1).csv
Original Data:
   sepal_length  sepal_width  petal_length  petal_width      species
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

Shuffled Data:
     sepal_length  sepal_width  petal_length  petal_width          species
71            6.1          2.8           4.0          1.3  Iris-versicolor
101           5.8          2.7           5.1          1.9   Iris-virginica
10            5.4          3.7           1.5          0.2      Iris-setosa
78            6.0          2.9           4.5          1.5  Iris-versicolor
91            6.1          3.0           4.6          1.4  Iris-versicolor

Train/Test Split:
Train_Shape: (105, 5)
Test_Shape: (45, 5)

Sample Predictions:
['Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor']

Model Accuracy: 0.9777777777777777

Random 5 samples from Test Data:
     sepal_length  sepal_width  petal_length  petal_width          species
66            5.6          3.0           4.5          1.5  Iris-versicolor
17            5.1          3.5           1.4          0.3      Iris-setosa
103           6.3          2.9           5.6          1.8   Iris-virginica
147           6.5          3.0           5.2          2.0   Iris-virginica
130           7.4          2.8           6.1          1.9   Iris-virginica
```