Shreeya Sampat

# Lab 3 – Decision Tree and KNN

Step 1: KNN Classifer

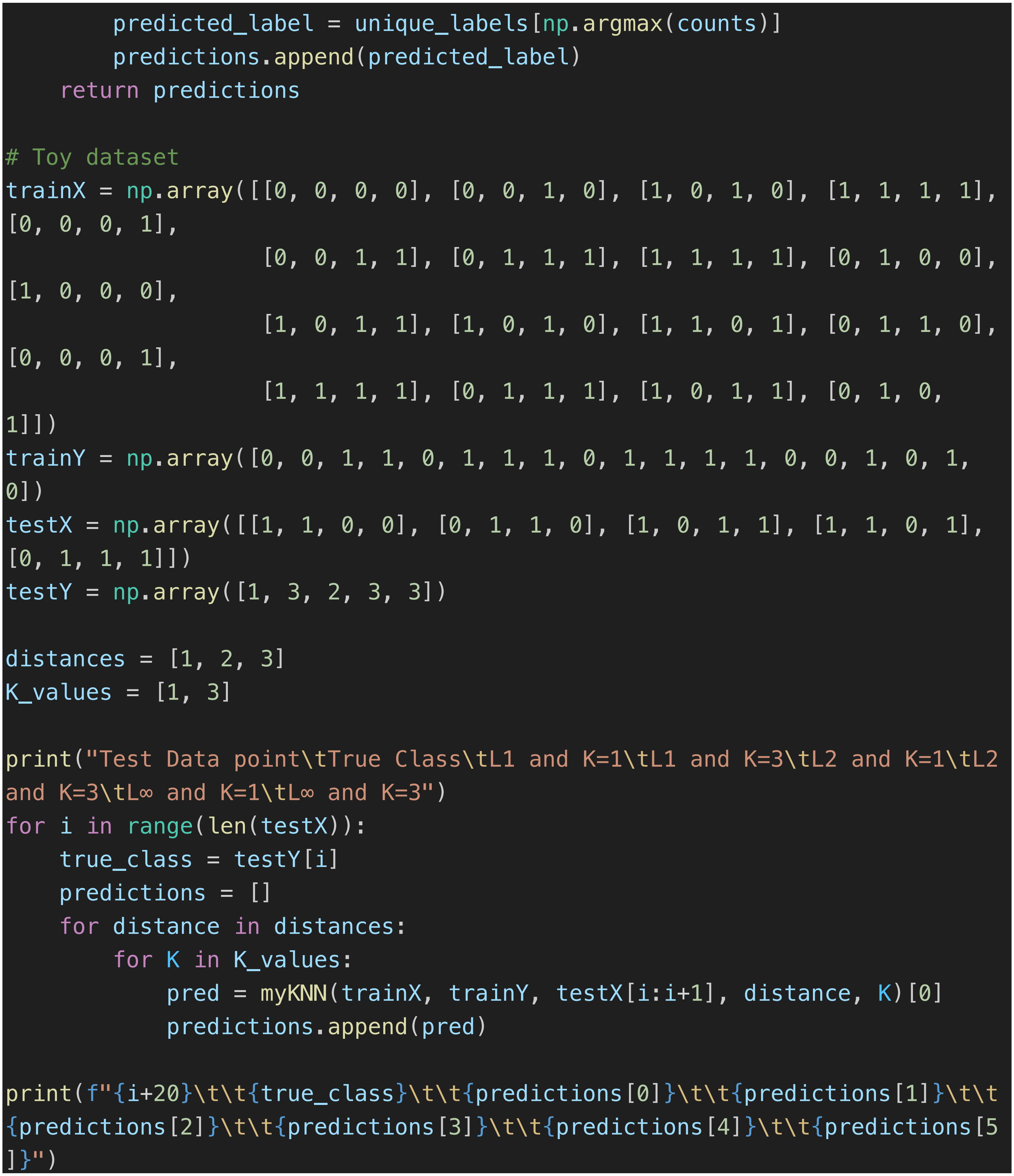
| Test Data Point | True Class | $L_1$ *and* $K = 1$ | $L_1$ and K = 3 | $L_2$ and K = 1 | $L_2$ and K = 3 | $L_\infty$ and K = 1 | $L_\infty$ and K = 3 |
|---|---|---|---|---|---|---|---|
| 20 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 21 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 23 | 3 | 1 | 1 | 1 | 1 | 1 | 0 |
| 24 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |

Python Code:

```
"""
Shreeya Sampat
Lab 3
OMSBA 5067 - Machine Learning
"""
import numpy as np

def calculate_distance(instance1, instance2, distance):
    if distance == 1:
        return np.sum(np.abs(instance1 - instance2))
    elif distance == 2:
        return np.sqrt(np.sum(np.square(instance1 - instance2)))
    elif distance == 3:
        return np.max(np.abs(instance1 - instance2))

def myKNN(trainX, trainY, testX, distance, K):
    predictions = []
    for test_instance in testX:
        distances = []
        for train_instance in trainX:
            dist = calculate_distance(train_instance, test_instance,
distance)
            distances.append(dist)
        sorted_indices = np.argsort(distances)
        k_nearest_neighbors = sorted_indices[:K]
        k_nearest_labels = trainY[k_nearest_neighbors]
        unique_labels, counts = np.unique(k_nearest_labels,
return_counts=True)
```



```
        predicted_label = unique_labels[np.argmax(counts)]
        predictions.append(predicted_label)
    return predictions

# Toy dataset
trainX = np.array([[0, 0, 0, 0], [0, 0, 1, 0], [1, 0, 1, 0], [1, 1, 1, 1],
[0, 0, 0, 1],
                   [0, 0, 1, 1], [0, 1, 1, 1], [1, 1, 1, 1], [0, 1, 0, 0],
[1, 0, 0, 0],
                   [1, 0, 1, 1], [1, 0, 1, 0], [1, 1, 0, 1], [0, 1, 1, 0],
[0, 0, 0, 1],
                   [1, 1, 1, 1], [0, 1, 1, 1], [1, 0, 1, 1], [0, 1, 0,
1]])
trainY = np.array([0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1,
0])
testX = np.array([[1, 1, 0, 0], [0, 1, 1, 0], [1, 0, 1, 1], [1, 1, 0, 1],
[0, 1, 1, 1]])
testY = np.array([1, 3, 2, 3, 3])

distances = [1, 2, 3]
K_values = [1, 3]

print("Test Data point\tTrue Class\tL1 and K=1\tL1 and K=3\tL2 and K=1\tL2
and K=3\tL∞ and K=1\tL∞ and K=3")
for i in range(len(testX)):
    true_class = testY[i]
    predictions = []
    for distance in distances:
        for K in K_values:
            pred = myKNN(trainX, trainY, testX[i:i+1], distance, K)[0]
            predictions.append(pred)

print(f"{i+20}\t\t{true_class}\t\t{predictions[0]}\t\t{predictions[1]}\t\t
{predictions[2]}\t\t{predictions[3]}\t\t{predictions[4]}\t\t{predictions[5
]}")
```

Shreeya Sampat

Lab 3 – Decision Tree and KNN

Step 3: Decision Tree with Larger Dataset

| **Test Data point** | **True Class** | **'gini' and max_depth = None** | **'entropy' and max_depth = None** | **'gini' and max_depth = 1** | **'gini' and max_depth = 2** |
|---|---|---|---|---|---|
| 20 | 1 | 1 | 1 | 1 | 1 |
| 21 | 3 | 0 | 0 | 0 | 0 |
| 22 | 2 | 1 | 1 | 1 | 1 |
| 23 | 3 | 1 | 1 | 1 | 1 |
| 24 | 3 | 1 | 1 | 1 | 1 |

Python Code:

```
from sklearn.tree import DecisionTreeClassifier

# Training dataset from Step 1
X_train = [[0, 0], [1, 1], [0, 1], [2, 2], [0, 0], [0, 0], [0, 1], [1, 1], [0, 1], [1, 0],
        [1, 0], [1, 0], [1, 1], [0, 1], [0, 0], [1, 1], [0, 1], [1, 0], [0, 1], [1, 0]]
Y_train = [0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0]

# Test dataset
X_test = [[1, 1], [0, 1], [1, 0], [1, 1], [1, 2]]
Y_test = [1, 3, 2, 3, 3]

criterions = ['gini', 'entropy']
max_depths = [None, 1, 2]

print("\nTest Data point\tTrue Class\t'gini' and max_depth=None\t'entropy' and max_depth=None\t'gini' and max_depth=1\t'gini' and max_depth=2")
for i in range(len(X_test)):
    true_class = Y_test[i]
    predictions = []
    for criterion in criterions:
        for max_depth in max_depths:
            clf = DecisionTreeClassifier(criterion=criterion, max_depth=max_depth)
            clf = clf.fit(X_train, Y_train)
            pred = clf.predict([X_test[i]])
            predictions.append(pred[0])

print(f"{i+20}\t\t{true_class}\t\t{predictions[0]}\t\t\t{predictions[1]}\t\t\t{predictions[2]}\t\t\t{predictions[3]}")
```