Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

ML - Experiment 6

1

60004210155
Jigar Siddhpura
C22

Experiment 6 — ML

Aim: To implement K-Nearest Neighbor

Theory: KNN is a simple ML algo based on supervised learning. It assumes similarity between new case & seen cases & put new case into category that is most likely. This means new data can be easily clasfied into a well suite category by using KNN algorithm. It can be used as regression as well as classification. It is called lazy learner algo as it doesn't learn from training set, immediately it stores dataset and at time of classification, it performs action on dataset. KNN working is as:

1. Select no. of K neighbors.
2. Calculate euclidean distance of K number of neighbors.
3. Take K-NN as per calculated euclidean distance.
4. Among these K numbers, count no. of data pts in each category.
5. Assign new data points to that category for which number of neighbors are max.

For selection of KNN:

- There is no way to determine best value for 'K', most preferred is 5.
- A very low value such as K=1 or K=2, can be noisy & lead to effects of outliers in model.
- Large values of K are good but it may find some difficulties.

Advantages:

1. Simple to implement
2. Robust to noisy training data
3. More effective if training data is large.

Disadvantages:

1. Determining 'K' value is complex.
2. Computation cost is high bcz of calculating dist. between data pts for all training samples.

Problem 1 :- Dataset 1

Let new data pt : $(x_n, y_n) = (5.5, 38)$

ID	Height (x_i)	Age (y_i)	$\sqrt{(x_i - x_n)^2 + (y_i - y_n)^2}$	weight
1	5	45	$((5-5.5)^2 + (45-38)^2)^{1/2} = 7.018$	77
2	5.11	26	$((5.11-5.5)^2 + (26-38)^2)^{1/2} = 12.006$	47
3	5.6	30	$((5.6-5.5)^2 + (30-38)^2)^{1/2} = 8$	55
4	5.9	34	$((5.9-5.5)^2 + (34-38)^2)^{1/2} = 4.011$	59
5	4.8	40	$((4.8-5.5)^2 + (40-38)^2)^{1/2} = 2.118$	72
6	5.8	36	$((5.8-5.5)^2 + (36-38)^2)^{1/2} = 2.022$	60
7	5.3	19	$((5.3-5.5)^2 + (38-19)^2)^{1/2} = 19.001$	40
8	5.8	28	$((5.8-5.5)^2 + (28-38)^2)^{1/2} = 10.004$	60
9	5.5	23	$((5.5-5.5)^2 + (23-38)^2)^{1/2} = 15$	45
10	5.6	32	$((5.6-5.5)^2 + (32-38)^2)^{1/2} = 6$	58

Now, for data point (5.5, 38) if K = 1 then weight will be 60 (as euclidean distance is less than that point as K = 1, so only 1 nearest neighbor is checked)

For K more than 1, average will be taken.

If $K=3$, then weight $= (59+72+60)/3 = 63.6667$

If $K=5$, then weight $= (59+72+60+58+77)/5 = 65.2$

Dataset 2: Let the new point be $(x_n, y_n) = (~~170,~~ 170, 57)$

Height (H_i)	Weight (W_i)	$\sqrt{(x_n - x_i)^2 + (y_n - y_i)^2}$	Class
167	51	$\sqrt{(167-170)^2 + (51-57)^2} = 6.708$	Underweight
182	62	$\sqrt{(182-170)^2 + (62-57)^2} = 13$	Normal
176	69	$\sqrt{(176-170)^2 + (69-57)^2} = 13.416$	Normal
173	64	$\sqrt{(173-170)^2 + (64-57)^2} = 7.615$	Normal
172	65	$\sqrt{(172-170)^2 + (65-57)^2} = 8.246$	Normal
174	56	$\sqrt{(174-170)^2 + (56-57)^2} = 4.123$	Underweight
169	58	$\sqrt{(169-170)^2 + (58-57)^2} = 1.414$	Normal
173	57	$\sqrt{(173-170)^2 + (57-57)^2} = 3$	Normal
170	55	$\sqrt{(170-170)^2 + (55-57)^2} = 2$	Normal

for $K=1$, class for $(170, 57)$ will be Normal

for $K=3$, Nearest neighbors for $(170, 57)$ are

Normal (1.414), Normal (2), Normal (3).

So final class is Normal

for $K=5$, Normal (1.4), Normal (2), Normal (3), Underweight (4.123)

Underweight (6.708).

So final is Normal.

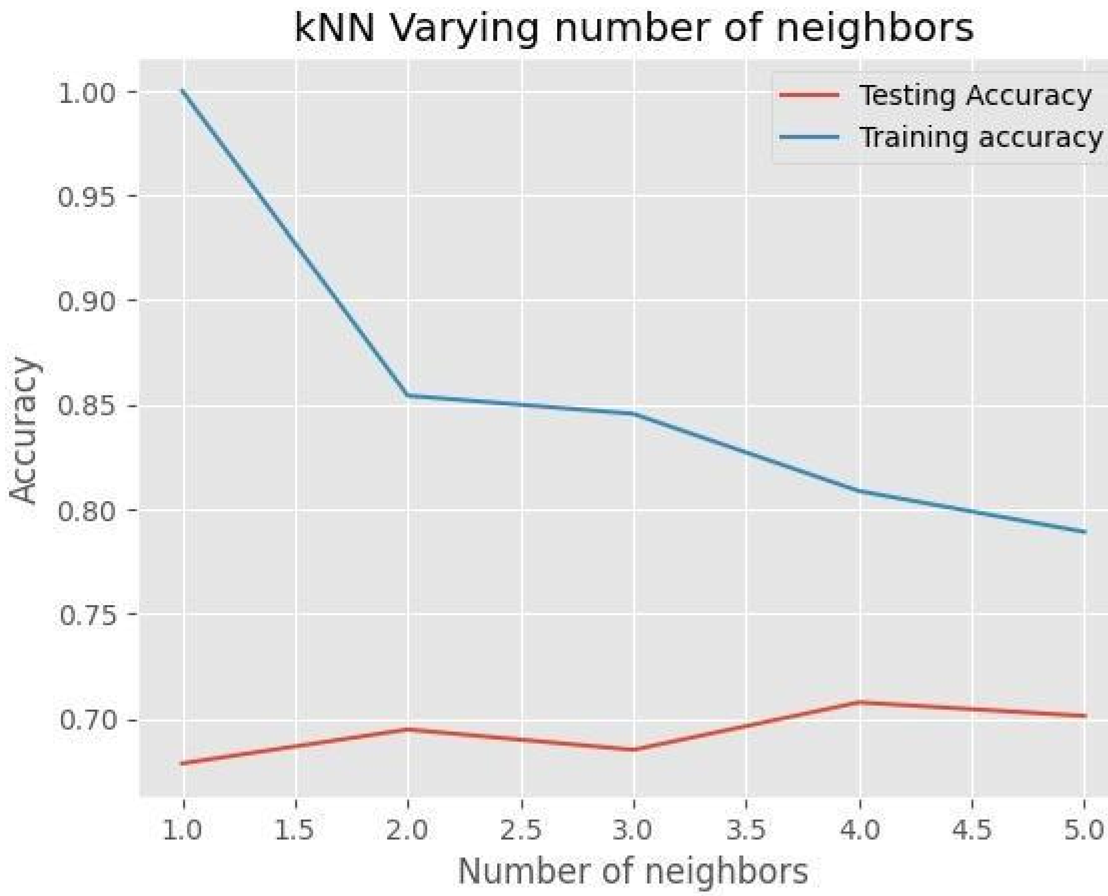
```
### 1st DATASET

import numpy as np

def euclidean_distance(p1, p2):
    return np.sqrt(np.sum((p1 - p2)**2))

dataset1 = np.array([[5, 45], [5.11, 26], [5.6, 30], [5.9, 34], [4.8, 40], [5.8, 36], [5.3, 19], [5.8, 28], [5.5,
23], [5.6, 32]])
target1 = np.array([77, 47, 55, 59, 72, 60, 40, 60, 45, 58])
test1 = np.array([5.5, 38])

for k in [1,3,5]:
  # Calculate distances to all points in the dataset
  distances = np.array([euclidean_distance(test1, d) for d in dataset1])

  # Get indices of K nearest neighbors
  nearest_indices = np.argsort(distances)[:k]

  # Predict the target value based on the average of K nearest neighbors
  predicted_target = np.mean(target1[nearest_indices])

  print(f"Predicted target for height=5.5 and age=38: for k = {k} is {predicted_target}")
```

```
Predicted target for height=5.5 and age=38: for k = 1 is 60.0
Predicted target for height=5.5 and age=38: for k = 3 is 63.666666666666664
Predicted target for height=5.5 and age=38: for k = 5 is 65.2
```

```
"""### 2nd DATASET"""

import math

def euclidean_distance(p1, p2):
    return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

# Define the dataset
data = [
    [167, 51, 'under'],
    [182, 62, 'normal'],
    [176, 69, 'normal'],
    [173, 64, 'normal'],
    [172, 65, 'normal'],
    [174, 56, 'under'],
    [169, 58, 'normal'],
    [173, 57, 'normal'],
    [170, 55, 'normal']
]

point = [170, 57]

k = 3

for k in [1,3,5]:
  distances = [(euclidean_distance(point, d[:2]), d) for d in data]

  nearest_neighbors = sorted(distances)[:k]

  category_count = {}
  for _, neighbor in nearest_neighbors:
      category = neighbor[2]
      category_count[category] = category_count.get(category, 0) + 1

  predicted_category = max(category_count, key=category_count.get)

  print(f"Predicted category for height=170 and weight=57: for k = {k} is {predicted_category}")
```

```
Predicted category for height=170 and weight=57: for k = 1 is normal
Predicted category for height=170 and weight=57: for k = 3 is normal
Predicted category for height=170 and weight=57: for k = 5 is normal
```

```
"""### 3rd DATASET"""

from google.colab import
drive
drive.mount('/content/gdrive
')

import numpy as
np import pandas
as pd
import matplotlib.pyplot as plt
from          sklearn.model_selection          import
train_test_split  from   sklearn.neighbors   import
KNeighborsClassifierplt.style.use('ggplot')

df = pd.read_csv('/content/gdrive/MyDrive/ML/diabetes.csv')

X =
df.drop('Outcome',axis=1).values
y = df['Outcome'].values

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.4,random_state=42,

stratify=y)neighbors = np.arange(1,6)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
  #Setup a knn classifier with k neighbors
  knn =
  KNeighborsClassifier(n_neighbors=k)

  #Fit the model
  knn.fit(X_train,
  y_train)

  #Compute accuracy on the training set
  train_accuracy[i] = knn.score(X_train,
  y_train)

  #Compute accuracy on the test set
  test_accuracy[i] = knn.score(X_test, y_test)


plt.title('kNN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing
Accuracy') plt.plot(neighbors, train_accuracy,
label='Training accuracy')plt.legend()
plt.xlabel('Number of
neighbors')
plt.ylabel('Accuracy')
plt.show()
```



kNN Varying number of neighbors
Testing Accuracy
Training accuracy
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Number of neighbors