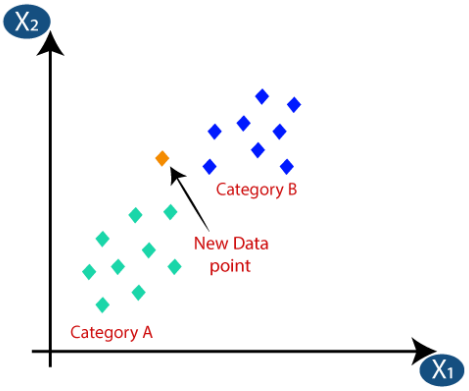
T. Mohamed Nadhim
3122 21 3002 058
ECE A 2025

Ex 3: K – Nearest Neighbors Algorithm



1. Simple KNN Using Euclidian Distance Measurement

Dataset Description:

```
2. apple_sizes = [7, 6.5, 7.2, 6, 6.8, 7.5, 6.3, 6.7, 6.1, 7.1]
3. apple_weights = [150, 140, 155, 130, 145, 160, 135, 142, 132, 148]

4. orange_sizes = [5, 6, 4, 5.5, 6.5, 4.5, 5.2, 5.8, 6.2, 4.8]
5. orange_weights = [100, 120, 90, 110, 130, 95, 105, 115, 125, 85]
```

Let us take a small dataset for Fruit sizes and Fruit Weights for 'Apples' and 'Oranges' respectively.

Sizes: it specifies the width of the given fruit in cm's

Weights: it specifies the weight of the given fruit in grams

The given dataset is very small and the scope for errors is large. So, we are generating more dataset with a slight deviation based on the originally provided dataset.

This gives us with more data and hence improving our results in classification.

ex-2-knn-euclidian

February 27, 2024

1 KNN Classification Algorithm Using Euclidian Distances for a small dataset

1.1 Importing necessary libraries

[46]:
```
import matplotlib.pyplot as plt
import random
import math
```

1.2 Generating the dataset for apple and orange (Size and Weight) and Plotting

[47]:
```
#Seed dataset for Apple
apple_sizes = [7, 6.5, 7.2, 6, 6.8, 7.5, 6.3, 6.7, 6.1, 7.1]
apple_weights = [150, 140, 155, 130, 145, 160, 135, 142, 132, 148]

num_datasets = 50

# Generating dataset for Apple
datasets = []
for _ in range(num_datasets):
    dataset = []
    for _ in range(10):
        size_deviation = random.uniform(-0.5, 0.5)  # Adjust this range as␣
 ↪needed
        weight_deviation = random.uniform(-10, 10)   # Adjust this range as␣
 ↪needed
        fruit_size = apple_sizes[_] + size_deviation
        fruit_weight = apple_weights[_] + weight_deviation
        apple_sizes.append(round(fruit_size,2))
        apple_weights.append(round(fruit_weight,2))


# Seed dataset for Orange
orange_sizes = [5, 6, 4, 5.5, 6.5, 4.5, 5.2, 5.8, 6.2, 4.8]
orange_weights = [100, 120, 90, 110, 130, 95, 105, 115, 125, 85]
```

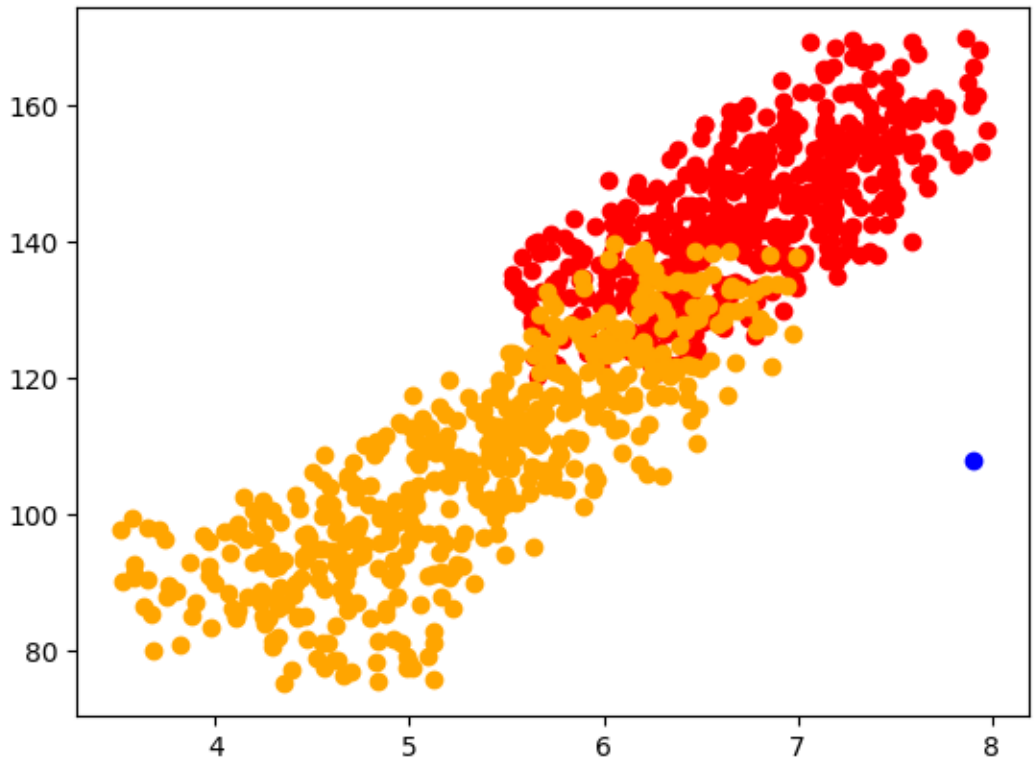
```
# Generating dataset for Orange
for _ in range(num_datasets):
    for _ in range(10):
        size_deviation = random.uniform(-0.5, 0.5)
        weight_deviation = random.uniform(-10, 10)
        fruit_size = orange_sizes[_] + size_deviation
        fruit_weight = orange_weights[_] + weight_deviation
        orange_sizes.append(round(fruit_size,2))
        orange_weights.append(round(fruit_weight,2))


# generating a relevant test data
size_test = round( random.uniform(min(orange_sizes), max(apple_sizes)),2)
weight_test = round(random.uniform(min(orange_weights), max(apple_weights)),2)


plt.scatter(apple_sizes,apple_weights, color='red')
plt.scatter(orange_sizes,orange_weights, color='orange')
plt.scatter(size_test,weight_test, color='blue')

len(apple_sizes)
```

[47]: 510



1.3 Finding the Distance between the testing data and the existing data

[48]:
```
#Finding euclidean distances between the test data and all the points in the␣
 ↪dataset
distances =[]

for i in range(0,len(apple_sizes)):
    dist = math.sqrt(((apple_sizes[i] - size_test)**2) + ((apple_weights[i] -␣
 ↪weight_test)**2))
    distances.append((dist,'apple'))
    dist = math.sqrt(((orange_sizes[i] - size_test)**2) + ((orange_weights[i] -␣
 ↪weight_test)**2))
    distances.append((dist,'orange'))

distances.sort()
```

1.4 Using KNN Algorithm to Classify the Test Data

[49]:
```
k = 5

neighbors =[]

for i in range(k):
    neighbors.append(distances[i][1])

orange_probability = neighbors.count('orange')*100/k
apple_probability = neighbors.count('apple')*100/k

print(f'''Based on the given data, the test fruit is:
            {round(orange_probability,2)}% orange;
            {round(apple_probability,2)}% apple;
      ''')
```

```
Based on the given data, the test fruit is:
            100.0% orange;
            0.0% apple;
```

T. Mohamed Nadhim
3122 21 3002 058
ECE A 2025

Inference:

Hence, KNN algorithm was applied by calculating the Euclidean distance between the test data point and the existing dataset. The Classification was simplified and visualized.

2. KNN using Inbuilt Functions

Let us perform KNN classification using inbuilt python functions

Dataset Description:

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0

This dataset contains various health-related attributes of individuals, including the number of pregnancies, glucose concentration, blood pressure, skin thickness, insulin levels, BMI, a diabetes pedigree function, age, and an outcome variable indicating the presence or absence of diabetes (0 or 1).

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

ex2-knn-using-inbuilt-functions

February 27, 2024

#Importing necessary libraries

[126]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
```

#Importing the dataset

[127]:
```
data = pd.read_csv('/content/diabetes.csv')
```

#Eliminating the zeros in the dataset and replacing with mean value

[128]:
```
zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']

for column in zero_not_accepted:
  data[column] = data[column].replace(0,np.NaN)
  mean = int(data[column].mean(skipna=True))
  data[column] = data[column].replace(np.NaN,mean)
```

#Splitting the dataset into training and testing dataset

[129]:
```
x = data.iloc[:,0:-1]
y = data.iloc[:, -1]

x_train, x_test, y_train, y_test = train_test_split(x,y,random_state = 0,␣
 ↪test_size = 0.2)
```

#Applying feature scaling to the training and testing dataset

[130]:
```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

#Training the KNN Model based on 'Euclidean' metric measure

[131]:
```
classifier = KNeighborsClassifier(n_neighbors =11, p=2, metric='euclidean')
classifier.fit(x_train,y_train)
```

[131]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

#Making Prediction using the trained model

[132]:
```
y_pred = classifier.predict(x_test)
```

#Evaluating the prediction results of the model

[133]:
```
cm = confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[94 13]
 [15 32]]
```

[136]:
```
print(accuracy_score(y_test,y_pred)*100)
```

```
81.81818181818183
```

T. Mohamed Nadhim
3122 21 3002 058
ECE A 2025

Inference:

Hence, KNN Classification was done for the diabetes dataset, which is helpful in predicting the possibility of the patient having diabetes with an accuracy score of 81.81%. Inbuilt python functions for KNN classifiers and Feature Scaling was explored.

Project Link:

https://github.com/Nadhim/ML-Lab/tree/main/Experiment_2%20-%20KNN%20Classifier