Day 20/100 of Data Science

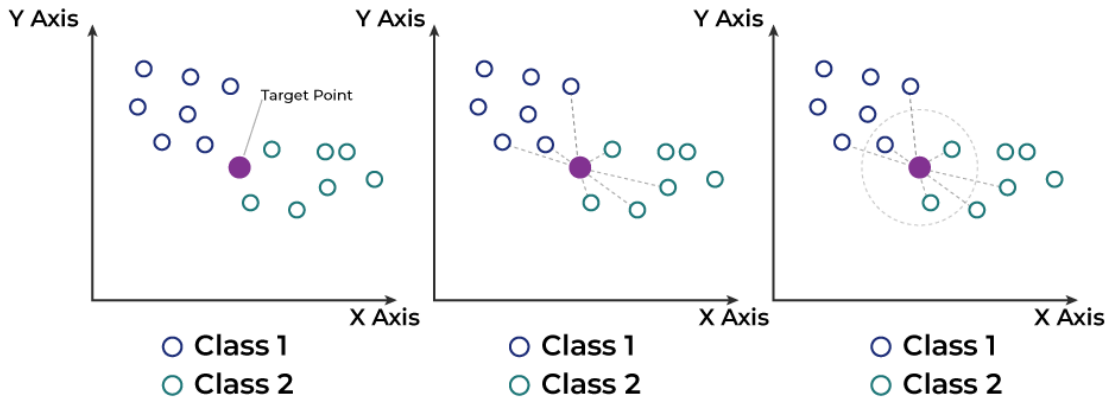
K-Nearest Neighbor(KNN) Algorithm

- K-Nearest Neighbors (KNN) is a simple and widely used **classification and regression** algorithm in machine learning.
- It's a type of instance-based learning, where the function is only approximated locally and all computation is deferred until classification.
- It's known for its simplicity and ease of understanding, making it a great starting point for beginners in machine learning.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- Step-1: Select the number K of the neighbors
- Step-2: Calculate the distance of K number of neighbors
- Step-3: Take the K nearest neighbors as per the calculated distance.
- Step-4: Among these k neighbors, count the number of the data points in each category.
- Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
- Step-6: Our model is ready.

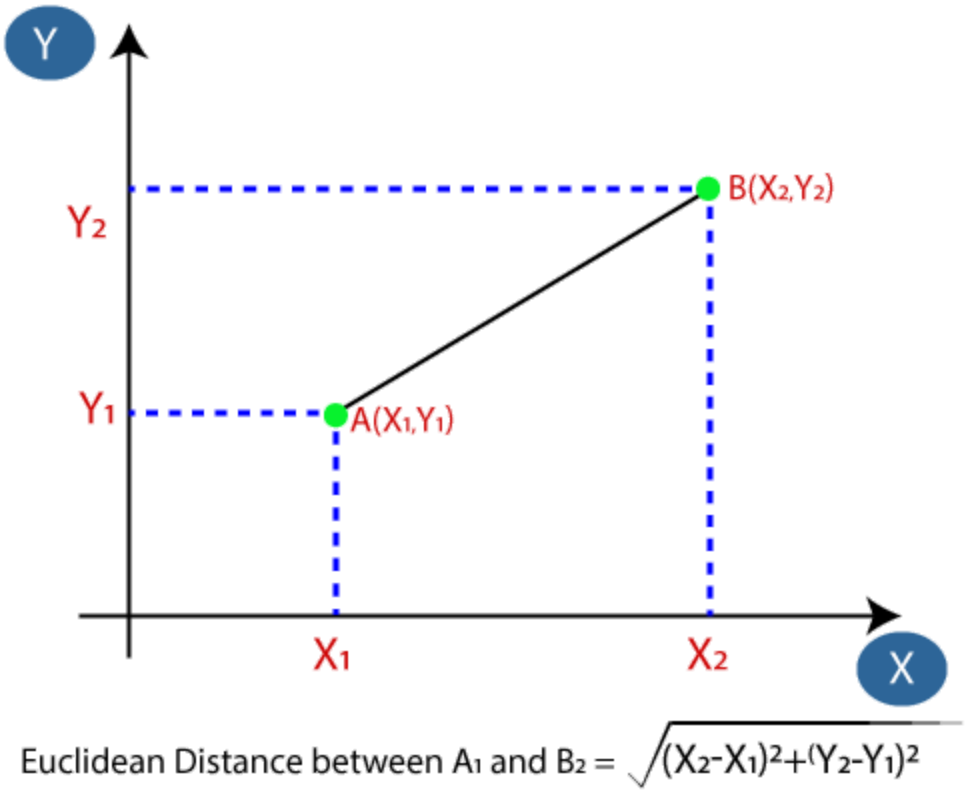


Calculated Distances

- Euclidean distance
- Manhattan Distance

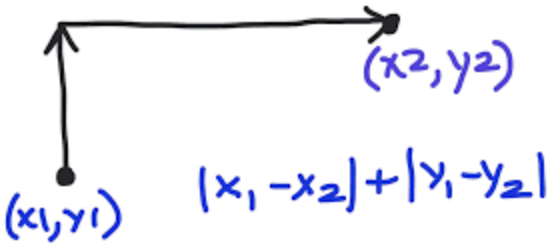
Euclidean Distance

- Euclidean distance, as the name suggests, is a mathematical formula used to calculate the straight-line distance between two points in n-dimensional space.



Manhattan Distance

- Manhattan Distance: This calculates the total distance traveled along each dimension to get from one point to another, like walking along a city grid.



Implementation

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [3]:
```
df = pd.read_csv('diabetes.csv')
df.head(5)
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [5]:
```
df.shape
```

Out[5]:
```
(768, 9)
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:
```
df.Insulin.sum()
```

Out[7]:
```
61286
```

In [8]:
```
df.Insulin
```

Out[8]:
```
0        0
1        0
2        0
3       94
4      168
      ...
763    180
764      0
765    112
766      0
767      0
Name: Insulin, Length: 768, dtype: int64
```

In [9]:
```
X = df.iloc[:, 0:8]
y = df.iloc[:, 8]

xtr, xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=4)
```

In [10]:
```
sc = StandardScaler()

xtr = sc.fit_transform(xtr)
xte = sc.fit_transform(xte)
```

In [11]:
```
clf = KNeighborsClassifier(n_neighbors=11, p=2, metric='euclidean') #p=2 Power paramet
```

In [12]:
```
clf.fit(xtr,ytr)

pred = clf.predict(xte)
```

In [13]:
```
print(confusion_matrix(pred, yte))
```

```
[[88 25]
 [14 27]]
```

In [14]:
```
print(accuracy_score(pred, yte))
```

```
0.7467532467532467
```

In [ ]:

Follow for More

Linkedin: https://www.linkedin.com/company/eternaltek/about/?viewAsMember=true

Medium: https://medium.com/@eternaltek.info

WhatsApp Channel: https://whatsapp.com/channel/0029Va5onCbDjiOTi6D1vU36

Github: https://github.com/Vamsi-2203

In [ ]: