# A03

April 29, 2022

[1]:
```
import pandas as pd
import numpy as np
import math
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier as knn

df =pd.read_csv('KNN.csv')
df
```

[1]:

| | x | y | class |
|---|---|---|---|
| 0 | 2 | 4 | negative |
| 1 | 4 | 2 | negative |
| 2 | 4 | 4 | positive |
| 3 | 4 | 6 | negative |
| 4 | 6 | 2 | positive |
| 5 | 6 | 4 | negative |

[2]:
```
def manual_knn(xtrain,xtest,ytrain,ytest,trainclass,testclass=[],weighted=0):
    k=3
    predclass=list()
    distances=list()
    for i in range(len(xtest)):
      distances = []
      for j in range(len(xtrain)):
        xd=(xtest[i]-xtrain[j])**2
        yd=(ytest[i]-ytrain[j])**2
        d=math.sqrt(xd+yd)
        row=(trainclass[j],d)
        distances.append(row)

      l = sorted(distances, key=lambda x: x[1])[:k]
      if weighted ==1:
          pos=0
          neg=0

          for d in l:
            if d[0]=="positive":
              pos=pos+(1/d[1])
```

```
            else:
              neg=neg+(1/d[1])
      else:
          topclasses=[e[0] for e in l]
          pos=topclasses.count("positive")
          neg=topclasses.count("negative")
      predclass.append("positive" if int(pos > neg) else "negative")

      print("Prediction for ("+str(xtest[i])+","+str(ytest[i])+"):
↪",predclass[-1])



    if len(testclass)!=0:
        hit=0
        for i in range(len(testclass)):
          if testclass[i]==predclass[i]:
            hit=hit+1
        n=len(testclass)
        acc=hit/n
        print("Accuracy Score:",acc)
```

[3]:
```
def auto_knn(model,xtrain,xtest,ytrain,ytest=[],acc=0):
    model.fit(xtrain,ytrain)

    ypred = model.predict(xtest)
    print(xtest)
    print(ypred)
    if acc!=0:
        from sklearn.metrics import accuracy_score
        print("Accuracy:",accuracy_score(ypred,ytest))
```

KNN CLASSIFICATION

[4]:
```
x_df = df['x'].values
y_df = df['y'].values
class_df = df['class'].values
# print(x_df,y_df,class_df)
xtrain, xtest, ytrain, ytest, trainclass, testclass  = train_test_split(x_df,␣
↪y_df, class_df, random_state=42)
# print(xtrain,xtest,ytrain,ytest)
manual_knn(xtrain,xtest,ytrain,ytest,trainclass,testclass)
```

```
Prediction for (2,4): negative
Prediction for (4,2): positive
Accuracy Score: 0.5
```

SK-LEARN KNN CLASSIFICATION

[5]:
```
x = df.iloc[:,:-1].values
y = df.iloc[:,2].values
```

[6]:
```
print(x,y)
```

```
[[2 4]
 [4 2]
 [4 4]
 [4 6]
 [6 2]
 [6 4]] ['negative' 'negative' 'positive' 'negative' 'positive' 'negative']
```

[7]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=42)

cf = knn(n_neighbors=3,p=2, metric='euclidean')
auto_knn(cf,x_train, x_test, y_train, y_test,1)
```

```
[[2 4]
 [4 2]]
['negative' 'positive']
Accuracy: 0.5
```

KNN CLASSIFICATION FOR POINT(6,6)

[8]:
```
manual_knn(x_df,[6],y_df,[6],class_df)
```

```
Prediction for (6,6): negative
```

SK-LERAN KNN CLASSIFICATION FOR POINT(6,6)

[9]:
```
# Using Sk-learn Knn for point(6,6)
cf = knn(n_neighbors=3,p=2, metric='euclidean')
auto_knn(cf,x,[[6,6]],y)
```

```
[[6, 6]]
['negative']
```

WEIGHTED KNN CLASSIFICATION

[10]:
```
xtrain, xtest, ytrain, ytest, trainclass, testclass  = train_test_split(x_df,␣
 ↪y_df, class_df, random_state=42)
manual_knn(xtrain, xtest, ytrain, ytest, trainclass, testclass,1)
```

```
Prediction for (2,4): negative
Prediction for (4,2): positive
Accuracy Score: 0.5
```

SK-LEARN WEIGHTED KNN CLASSIFICATION

[11]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y,random_state=42)
cf = knn(n_neighbors=3, weights="distance",p=2, metric='euclidean')
auto_knn(cf,xtrain,xtest,ytrain,ytest,1)
```

```
[[2 4]
 [4 2]]
['negative' 'positive']
Accuracy: 0.5
```

WEIGHTED KNN CLASSIFICATION FOR POINT(6,6)

[12]:
```
manual_knn(x_df, [6], y_df, [6], class_df, [],1)
```

```
Prediction for (6,6): negative
```

SK-LEARN WEIGHTED KNN CLASSIFICATION FOR POINT(6,6)

[13]:
```
cf = knn(n_neighbors=3, weights="distance",p=2, metric='euclidean')
auto_knn(cf,x,[[6,6]],y)
```

```
[[6, 6]]
['negative']
```

[ ]:
```

```