6/14/2022

Technical Report on KNN

From

Raja Fahad Israr Ahmed

TABLE OF CONTENTS

➢ K Nearest Neighbor Introduction:

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. In Credit ratings, financial institutes will predict the credit rating of customers. In loan disbursement, banking institutes will predict whether the loan is safe or risky. In political science, classifying potential voters in two classes will vote or won't vote. KNN algorithm used for both classification and regression problems. KNN algorithm based on feature similarity approach.

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

➢ KNN Working:

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.

Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance.

KNN has the following basic steps:

- Select dataset
- Choose Value of k
- Calculate the distance between test data and training data

Sort the distance values in ascending order

- Now choose K number of values from top
- Assign a class based on the frequent class of these rows.

➢ How to select value of K:

There is no straightforward method to calculate the value of K in KNN. You have to play around with different values to choose the optimal value of K. Choosing a right value of K is a process called Hyper parameter Tuning.The value of optimum K totally depends on the dataset that you are using. The best value of K for KNN is highly data-dependent. In different scenarios, the optimum K may vary. It is more or less hit and trail method.You need to maintain a balance while choosing the value of K in KNN. K should not be too small or too large.

There is no one proper method of estimation of K value in KNN. No method is the rule of thumb but you should try considering following suggestions:

Square Root Method: Take square root of the number of samples in the training dataset.

Cross Validation Method: We should also use cross validation to find out the optimal value of K in KNN. Start with K=1, run cross validation (5 to 10 fold), measure the accuracy and keep repeating till the results become consistent.

➢ Heart Disease Prediction Dataset:

Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of clinical data analysis. The amount of data in the healthcare industry is huge. Data mining turns the large collection of raw healthcare data into information that can help to make informed decisions and predictions.The attributes are:

- Age
- Sex
- Chest Pain

- Chlosteral
- FBS
- EKG
- HR
- Exercise
- ST Depression
- Slope of ST
- Number of vessels fluro
- Thallium
- Heart Disease

➢ What Algorithem actually Do:

There is only two classes who is distinguish the data presence and absence. When a man have symptoms then presence class is display otherwise absence class will be print.

➢ Code Analysis:

- **Import Libraries:**
  import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  import seaborn as sns
  sns.set()
  import warnings
  from sklearn.model_selection import train_test_split
  from sklearn.neighbors import KNeighborsClassifier
- **Load data Set**
  data=pd.read_csv('Heart_Disease_Prediction.csv')
  Divide the data into Training and test set:
  X=data.iloc[:,:8]
  Y=data.iloc[:,-1]
  X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.2, random_state=40,stratify=Y)
  Display Training data size:
  print("X_train shape: {}".format(X_train.shape))
- **Display test data size:**

print("X_test shape: {}".format(X_test.shape))

- **Build Model and set Values of k**
  knn = KNeighborsClassifier(n_neighbors=7)
  knn.fit(X_train, y_train)
- **Model Evalution:**
  print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))

➢ Analyze:

- When we select Test data is 40% , Training data is 60% and Random value is zero then the result is here:

Value of K	Accuracy (%)
1	86
3	73
5	72
7	72
9	68
11	67
13	68
15	66
17	68
19	66
21	66
23	66
25	64
27	65
29	65

Now here is Accuracy values are repeating and decreasing. Here is we receive good accuracy from value one to seven but when the k=1 then there is no other value to compare that's why it gives us good result so we ignore it and we declare good result is on k=3.

- When we select Test data is 40% , Training data is 60% and Random value is 20 then the result is here:

Value of K	Accuracy (%)
1	83
3	74
5	72
7	70
9	69
11	71
13	70
15	71
17	70
19	69
21	67
23	69
25	67

Now here is test data is 40% and Training data is 60% and the random value is 20.here random value change all result. Accuracy values are repeating and decreasing same as when the random value is 0. Here is we receive good accuracy from value one to seven but when the k=1 then there is no other value to compare that's why it gives us good result so we ignore it and we declare good result is on k=3.

- When we select Test data is 40% , Training data is 60% and Random value is 40 then the result is here:

Value of K	Accuracy (%)
1	84
3	74
5	71
7	71
9	69
11	69
13	69
15	70
17	70

19	70
21	68
23	68
25	67

here is test data is 40% and Training data is 60% and the random value is 40.here random value change all result. Accuracy values are repeating and decreasing Here is we receive good accuracy from value one to seven but when the k=1 then there is no other value to compare that's why it gives us good result so we ignore it and we declare good result is on k=3.

- When we select Test data is 20% , Training data is 80% and Random value is 40 then the result is here:

Value of K	Accuracy (%)
1	92
3	76
5	73
7	71
9	71
11	70
13	70
15	69
17	70
19	71
21	72
23	73
25	71

27	67
29	67
31	67
33	68

In this case Accuracy value is decreased when the value of K is increasing.

- When we select Test data is 30% , Training data is 80% and Random value is 40 then the result is here:

Value of K	Accuracy (%)
1	86
3	74
5	72
7	71
9	70
11	70
13	69
15	70
17	72
19	71
21	71
23	71
25	69
27	66
29	67
31	67

- When we select Test data is 50% , Training data is 50% and Random value is 0 then the result is here:

Value of K	Accuracy (%)
1	81
3	72

5	71
7	71
9	70
11	67
13	65
15	67
17	69
19	67
21	66
23	64
25	74
27	67
29	67
31	67
33	68

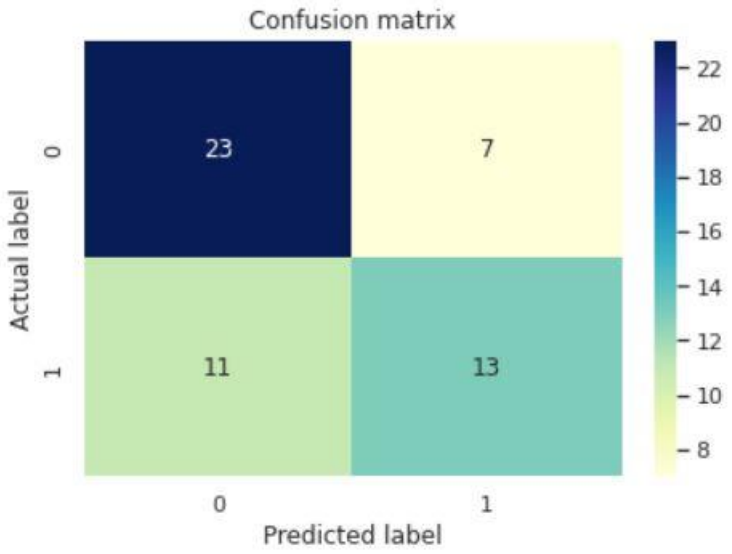
here is test data is 50% and Training data is 50% and the random value is 0.here random value change all result. Accuracy values are repeating and decreasing slowly.

- **Conclussion case Study:**

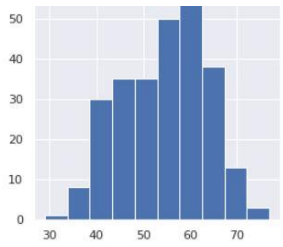
A small value of K means that noise will have a higher influence on the result. Larger the value of K, higher is the accuracy. If K is too large, you are under-fitting your model. In this case, the error will go up again. So, at the same time you also need to prevent your model from under-fitting. Your model should retain generalization capabilities otherwise there are fair chances that your model may perform well in the training data but drastically fail in the real data. Larger K will also increase the computational expense of the algorithm.

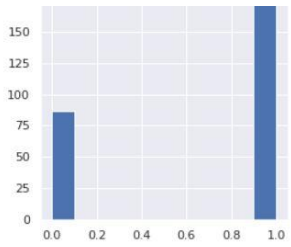
➢ Confusion Matrix:

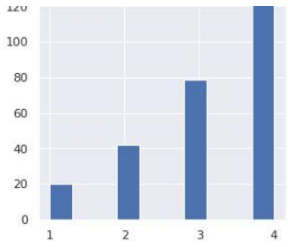
Predicted	Absence	Presence	All
True			
Absence	23	7	30
Presence	11	13	24
All	34	20	54

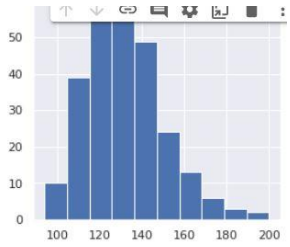


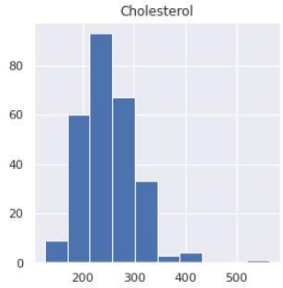
➢ Data Representation:

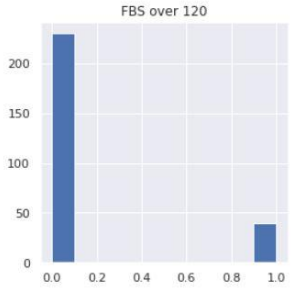


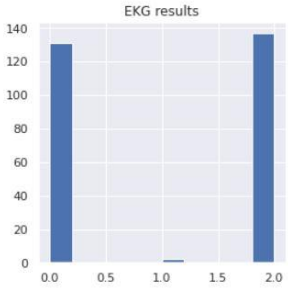


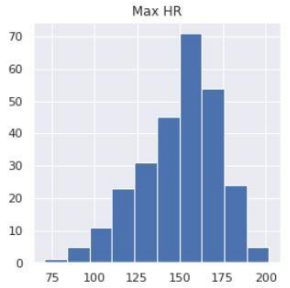


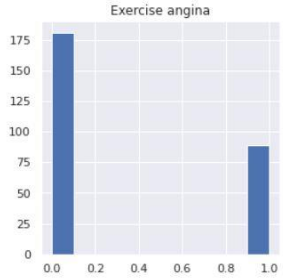


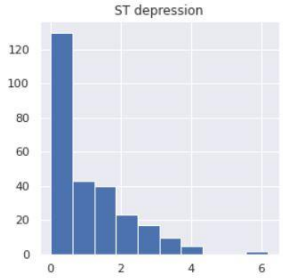


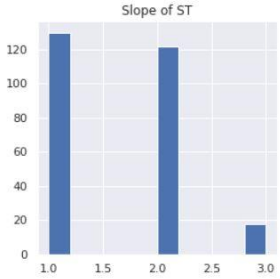


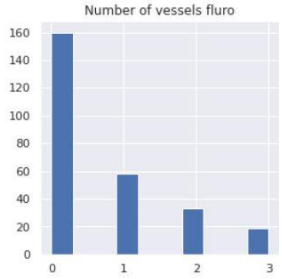


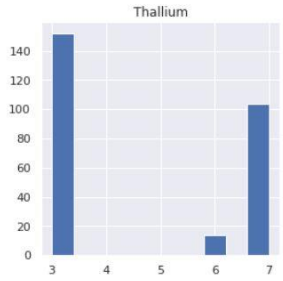












- **Code Representation:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier


warnings.filterwarnings('ignore')
# %matplotlib inline
data=pd.read_csv('Heart_Disease_Prediction.csv')
data.head()
X=data.iloc[:,:8]
Y=data.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.2,
random_state=40,stratify=Y)
start, end = 1, 70
print(X_train.shape)
print(X_test.shape)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X)
# arr.append()
```

```
print("Test set score: {:.2f}".format(knn.score(X, Y)))

##############################

data_copy = data.copy(deep = True)

data_copy[['Chest pain type','BP','Cholesterol','Max HR','Thallium']] =
data_copy[['Chest pain type','BP','Cholesterol','Max
HR','Thallium']].replace(0,np.NaN)

## showing the count of Nans

print(data_copy.isnull().sum())


data_copy['Chest pain type'].fillna(data_copy['Chest pain type'].mean(),
inplace = True)

data_copy['BP'].fillna(data_copy['BP'].mean(), inplace = True)

data_copy['Cholesterol'].fillna(data_copy['Cholesterol'].median(), inplace =
True)

data_copy['Max HR'].fillna(data_copy['Max HR'].median(), inplace = True)

data_copy['Thallium'].fillna(data_copy['Thallium'].median(), inplace = True)

from pandas.plotting import scatter_matrix

p=scatter_matrix(data,figsize=(25, 25))

p=sns.pairplot(data,hue='Heart Disease')


#import confusion_matrix

from sklearn.metrics import confusion_matrix

#let us get the predictions using the classifier we had fit above

y_pred = knn.predict(X_test)

confusion_matrix(y_test,y_pred)

pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'],
margins=True)
```

```
y_pred = knn.predict(X_test)

from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu"
,fmt='g')

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')
```