Heart Disease Prediction

Achanta Sai Krishna – CB.EN.U4ELC20004
Kuralanbu S – CB.EN.U4ELC20033
Vimal Dharshan N – CB.EN.U4ELC20081

References

1. Dataset:

   https://www.kaggle.com/datasets/aasheesh200/framingham-heart-study-dataset

2. Reference:

   https://www.dataminingbook.com/book/python-edition

Problem Formulation

Objective: To identify Heart Disease of a patient based on the given features

Dataset details:

- No. of rows: 4239
- No. of columns: 17
- No. of Class: 02
- Method of data collection is unknown

Assumptions:

- From the link mentioned for dataset, "framingham (1).csv" was considered for solving
  - In features, "Male" is changed to "Sex"
  - In features, "TenYearCHD" is changed to "target".
  - In features, "Education" is changed to "Chest Pain Type".

Problem Formulation

Assumptions:

- Missing data were filled with the mean of the rest of the corresponding data.

Link to full code mentioned in slides:

https://colab.research.google.com/drive/1GduUaVWJ3R6NeOGGfrPVotK6tgNeBVMf?usp=sharing

Feature Description

- sex – The person's sex (0 = female; 1 = male)
- age – The person's age in years
- Chest Pain Type – 1: Typical Angina, 2: Atypical Angina, 3:Non-Angina Pain, 4:Asymptomatic
- currentSmoker – The person is currently smoking (0 = false; 1 = true)
- cigsPerDay – Amount of cigarettes smoked per day by a person
- BPMeds – The person is taking medicine for blood pressure (0 = false; 1 = true)
- prevalentStroke – The person has a common stroke (0 = false; 1 = true)
- prevalentHyp – The person has common hypertension (0 = false; 1 = true)
- diabetes – The person has diabetes (0 = false; 1 = true)

Feature Description

- totChol – Total cholesterol of a person (in mg/dl)
- sysBP – Systolic blood pressure of a person (in mm Hg)
- diaBP – Diastolic blood pressure of a person (in mm Hg)
- BP – Total blood pressure of a person (sysBP/diaBP)
- BMI – Body Mass Index of a person (kg/m^2)
- heartRate – Total heart rate achieved by a person (bpm)
- glucose – Fasting blood sugar level of a person (mg/dl)
- target – Heart Disease of a person (0 = false; 1 = true)

Common in all methods/calculations

```
[179] #Authors: Achanta Sai Krishna,Kuralanbu,Vimal Dharshan
      #Objective: To find the optimal k value
      #Input: Dataset
      #Output: Accuracy and Confusion Matrix
      import pandas as pd #data analysis toolkit
      import matplotlib.pyplot as plt #for plotting graphs
      import numpy as np #high level computations
      %matplotlib inline
```

```
[180] from sklearn.preprocessing import StandardScaler #standardization of values
      from sklearn.preprocessing import MinMaxScaler #normalization of values
      from sklearn.model_selection import train_test_split #to split data
      from sklearn.neighbors import KNeighborsClassifier #KNN classifier
      from sklearn.metrics import confusion_matrix,accuracy_score #to get confusion matrix and accuracy
      from sklearn.model_selection import cross_val_score #to perform evaluation and cross-validation
```

```
[181] data_set = pd.read_csv("/content/framingham (1).csv") #dataset_input
```

```
[182] data_set=data_set.fillna(data_set.mean()) #mean for missing data
```

```
[183] data_set = np.round(data_set, decimals=2) #rounding all values in dataset to 2 decimal places
      data_set.head() #first 5 values in dataset
```

Common in all methods/calculations

	Sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	target
0	1	39	4	0	0	0	0	0	0	195	106.0	70.0	26.97	80	77	0
1	0	46	2	0	0	0	0	0	0	250	121.0	81.0	28.73	95	76	0
2	1	48	1	1	20	0	0	0	0	245	127.5	80.0	25.34	75	70	0
3	0	61	3	1	30	0	0	1	0	225	150.0	95.0	28.58	65	103	1
4	0	46	3	1	23	0	0	0	0	285	130.0	84.0	23.10	85	85	0

```
[184] data_set.tail() #It prints the last 5 values in dataset
```

	Sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	target
4235	0	48	2	1	20	0	0	0	0	248	131.0	72.0	22.00	84	86	0
4236	0	44	1	1	15	0	0	0	0	210	126.5	87.0	19.16	86	0	0
4237	0	52	2	0	0	0	0	0	0	269	133.5	83.0	21.47	80	107	0
4238	1	40	3	0	0	0	0	1	0	185	141.0	98.0	25.60	67	72	0
4239	0	39	3	1	30	0	0	0	0	196	133.0	86.0	20.91	85	80	0

```
[185] # no of rows and columns in the data set
      data_set.shape
```

```
(4240, 16)
```

Common in all methods/calculations

```
#checking for missing values
data_set.isnull().sum  #False means no missing data
```

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of         Sex    age  education  currentSmoker  cigsPerDay  BPMeds  \
0     False  False      False          False       False   False
1     False  False      False          False       False   False
2     False  False      False          False       False   False
3     False  False      False          False       False   False
4     False  False      False          False       False   False
...     ...    ...        ...            ...         ...     ...
4235  False  False      False          False       False   False
4236  False  False      False          False       False   False
4237  False  False      False          False       False   False
4238  False  False      False          False       False   False
4239  False  False      False          False       False   False

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0               False         False     False    False  False  False  False
1               False         False     False    False  False  False  False
2               False         False     False    False  False  False  False
3               False         False     False    False  False  False  False
4               False         False     False    False  False  False  False
...               ...           ...       ...      ...    ...    ...    ...
4235            False         False     False    False  False  False  False
4236            False         False     False    False  False  False  False
4237            False         False     False    False  False  False  False
4238            False         False     False    False  False  False  False
4239            False         False     False    False  False  False  False

      heartRate  glucose  target
0         False    False   False
1         False    False   False
2         False    False   False
3         False    False   False
4         False    False   False
...         ...      ...     ...
4235      False    False   False
4236      False    False   False
4237      False    False   False
4238      False    False   False
4239      False    False   False

[4240 rows x 16 columns]>
```

Common in all methods/calculations

```
[187] #Statistical measure about the dataset
      data_set.describe()
```

	Sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	target
count	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000
mean	0.429245	49.580189	1.930425	0.494104	8.944340	0.029245	0.005896	0.310613	0.025708	233.908255	132.354599	82.897759	25.685184	75.861085	74.463208	0.151887
std	0.495027	8.572942	1.053026	0.500024	11.904777	0.168513	0.076569	0.462799	0.158280	51.166237	22.033300	11.910394	4.420501	12.080265	32.862256	0.358953
min	0.000000	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	83.500000	48.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	205.000000	117.000000	75.000000	23.050000	68.000000	68.000000	0.000000
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	233.000000	128.000000	82.000000	25.380000	75.000000	77.000000	0.000000
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	262.000000	144.000000	90.000000	28.032500	83.000000	85.000000	0.000000
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696.000000	295.000000	142.500000	56.800000	143.000000	394.000000	1.000000

```
[188] #counting the  no of people's having Heart Disease ('1') and not having Heart Disease
      data_set['target'].value_counts()
```

```
0    3596
1     644
Name: target, dtype: int64
```

Knn classifier

- Distance metric used for computation is Minkowski distance.
- Splitting the dataset into training and testing: one_train, two_train, one_test, two_test with 70% for training and 30% for testing.
- Cross validation: Re-sampling procedure used to evaluate a model
  - Cross Validation (cv) is set to 5 (value should be equal to or less than the number of features present in our dataset).
- Extra code after running common code is as follows.
- Other information is mentioned in the comments of the code for better understanding.

```
[189] dset_modified = data_set.drop('target',axis=1) #dataset without class feature
```

```
[190] data_set_feat = pd.DataFrame(dset_modified,columns=data_set.columns[:-1]) #dataset without class feature
```

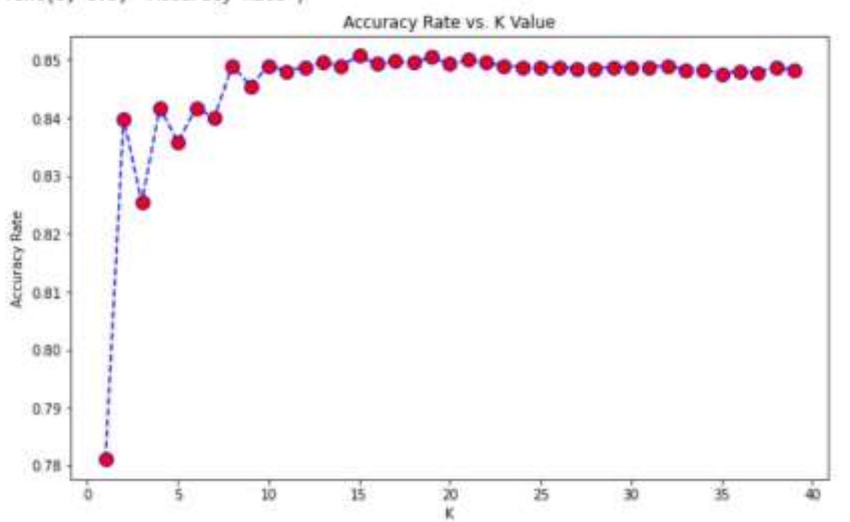
```
[191] data_set_feat = np.round(data_set_feat, decimals=2) #rounding all values to 2 decimal places
```

```
[192] one_train, one_test, two_train, two_test = train_test_split(data_set_feat,data_set['target'],test_size=0.20) #test_train split with test size=30% and train size=70%
```

```
[193] #Computation of accuracy rates for various neighbor values
      Accurate_rates = []

      for i in range(1,40):

        k_nearest_neighbour = KNeighborsClassifier(n_neighbors=i)
        final_score=cross_val_score(k_nearest_neighbour,data_set_feat,data_set['target'], cv=5)
        Accurate_rates.append(final_score.mean())
```

```
[194] #plot
      plt.figure(figsize=(10,6))

      plt.plot(range(1,40),Accurate_rates,color='blue',linestyle='dashed',marker='o',markerfacecolor='red',markersize=10)
      plt.title('Accuracy Rate vs. K Value')
      plt.xlabel('K')
      plt.ylabel('Accuracy Rate')
```

```
Text(0, 0.5, 'Accuracy Rate')
```



```
[195] max_index = Accurate_rates.index(max(Accurate_rates)) #Best case identifier

      k_nearest_neighbour = KNeighborsClassifier(n_neighbors=max_index)

      k_nearest_neighbour.fit(one_train,two_train)
      prediction = k_nearest_neighbour.predict(one_test)

      print('For K=',max_index)
      print('Confusion matrix:')
      print('\n')
      print(confusion_matrix(two_test,prediction)) #Confusion Matrix
      print('\n')
      print('Accuracy rate: ',round(accuracy_score(two_test,prediction),2)*100,'%')
      #Accuracy rate
```

```
For K= 14
Confusion matrix:


[[705   5]
 [134   4]]


Accuracy rate:  84.0 %
```

- Therefore, for the given data the maximum accuracy using K-nearest neighbors method was found as 84% for K = 14 neighbors.
- The corresponding confusion matrix has been printed.

- For a different K value:
- Same accuracy rate has been obtained as in graph

```
t = 100 #Random K value

k_nearest_neighbour = KNeighborsClassifier(n_neighbors=t)

k_nearest_neighbour.fit(one_train,two_train)
prediction = k_nearest_neighbour.predict(one_test)

print('For K=' ,t)
print('Confusion Matrix:')
print('\n')
print(confusion_matrix(two_test,prediction)) #Confusion Matrix
print('\n')
print('Accuracy rate: ',round(accuracy_score(two_test,prediction),2)*100,'%')
#Accuracy rate
```

```
For K= 100
Confusion Matrix:


[[1097    0]
 [ 175    0]]


Accuracy rate:  86.0 %
```

Normalization

```
[197] scaled = MinMaxScaler() #function Minmax scaler for normalising values
```

```
[198] scaled.fit(data_set.drop('target',axis=1)) #dropping class-feature
```

```
MinMaxScaler()
```

```
[199] dset_modified = scaled.transform(data_set.drop('target',axis=1))#dropping class-feature
```

```
[200] data_set_feat = pd.DataFrame(dset_modified,columns=data_set.columns[:-1]) #dropping class-feature
```

```
[201] data_set_feat = np.round(data_set_feat, decimals=2) #rounding all values to 2 decimals
      data_set_feat.head() #dataset after normalization
```

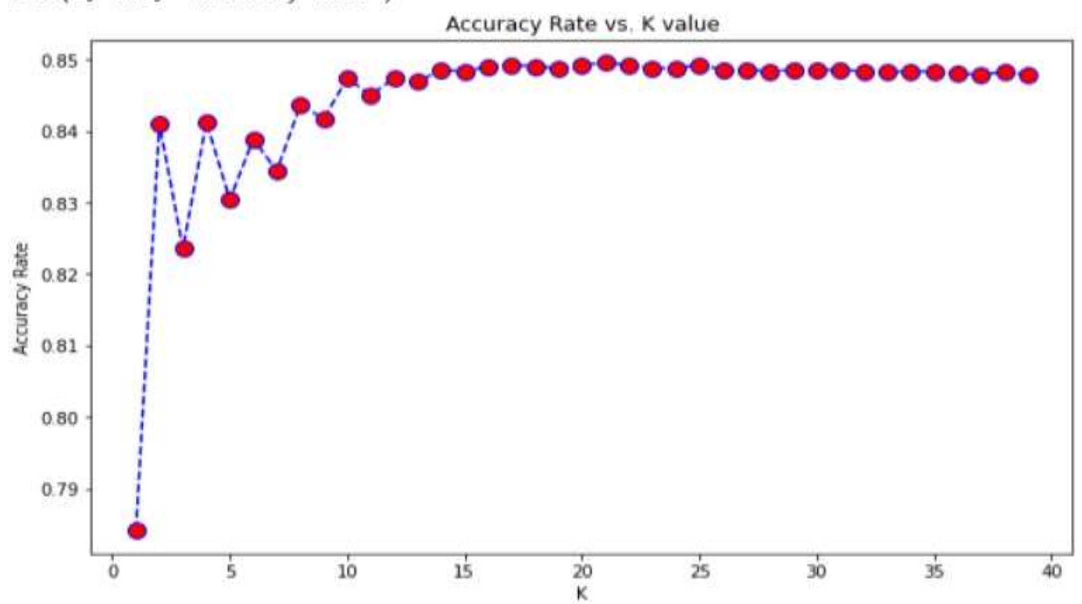
	Sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose
0	1.0	0.18	1.00	0.0	0.00	0.0	0.0	0.0	0.0	0.28	0.11	0.23	0.47	0.56	0.20
1	0.0	0.37	0.50	0.0	0.00	0.0	0.0	0.0	0.0	0.36	0.18	0.35	0.51	0.66	0.19
2	1.0	0.42	0.25	1.0	0.29	0.0	0.0	0.0	0.0	0.35	0.21	0.34	0.45	0.52	0.18
3	0.0	0.76	0.75	1.0	0.43	0.0	0.0	1.0	0.0	0.32	0.31	0.50	0.50	0.45	0.26
4	0.0	0.37	0.75	1.0	0.33	0.0	0.0	0.0	0.0	0.41	0.22	0.38	0.41	0.59	0.22

[206]
```python
#test_train split with test_size 30% and train_size 70%
one_train, one_test, two_train, two_test = train_test_split(data_set_feat,data_set['target'],test_size=0.30)
```

[207]
```python
#Computation of accuracy rates for various neighbour values
Accurate_rates = []

for i in range(1,40):

  k_nearest_neighbour = KNeighborsClassifier(n_neighbors=i)
  final_score=cross_val_score(k_nearest_neighbour,data_set_feat,data_set['target'],cv=5)
  Accurate_rates.append(final_score.mean())
```

```python
plt.figure(figsize=(10,6))

plt.plot(range(1,40),Accurate_rates,color='blue',linestyle='dashed',marker='o',markerfacecolor='red',markersize=10)
plt.title('Accuracy Rate vs. K value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

```
Text(0, 0.5, 'Accuracy Rate')
```



Inference - Knn

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p}$$

- Minowski Distance uses both Manhattan and Euclidean distance in a generalized form for calculation.
- KNN is also called a lazy classifier as it memorizes the training data and not exactly learn and fix the weights. Hence most of the computing work occurs during the classification rather than training time.
- For various values of K, the accuracy rates changes and through plotting all the values, the best case was found.
- In addition, the accuracy rates for other K values can be inferred from graph.
- Confusion matrix which formulates predicted vs actual values

Inference - Normalization

- The best case for k value is k = 100 and the accuracy rate is 86%.
- Normalization is a scaling technique where all the data in the dataset is scaled between a range that is 0 and 1.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- By comparing the accuracy values, the normalized value is decreased than that of the Knn value.

Miscellaneous

- **Libraries used:**
- Pandas – It is an open-source library developed for data analysis, which easily processes the raw data into a data frame.
- NumPy – Consisting of multidimensional array objects, mathematical and logical operations on arrays can be performed using this.
- Matplotlib – It is a multi-platform data visualization library for 2D plots of arrays built on NumPy arrays.
- Scikit-learn – An useful library that contains efficient tools for machine learning and statistical modelling including classification, regression, clustering, and dimension reduction

Miscellaneous

- **Functions used:**
- data_set.shape: the function displays the number of rows and columns from a dataset.
- data_set.isnull().sum: the function checks dataset contains missing data (or) values.
- data_set.describe: the function describes the statistical measures like mean, standard deviation, minimum and maximum of samples from the total dataset.
- data_set['target'].value_counts(): the function displays the total count of healthy persons and heart patients from the dataset.