yzx0bk4lf

October 17, 2024

```
[1]: #Aim : To perform and find the accuracy of Logistic Regression
```

```
[2]: #Name: Ritika Rajesh Junekar
     #RolL no. :30
     #Sub : ET1
     #section:C
```

```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     import warnings
     warnings.filterwarnings('ignore')
```

```
[2]: import os
```

```
[3]: os.getcwd()
```

```
[3]: 'C:\\Users\\USER'
```

```
[7]: os.chdir("C:\\Users\\USER\\Desktop")
```

```
[9]: df=pd.read_csv("framingham.csv")
```

```
[11]: #The "Framingham" heart disease dataset includes over 4,240 records, 15␣
        ↪attributes.
      #The goal of the dataset is to predict whether the patient has 10-year risk of␣
        ↪future (CHD) coronary heart disease
```

```
[13]: df.head()
```

```
[13]:    male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
      0     1   39        4.0              0         0.0     0.0                0
      1     0   46        2.0              0         0.0     0.0                0
      2     1   48        1.0              1        20.0     0.0                0
      3     0   61        3.0              1        30.0     0.0                0
```

```
4     0   46         3.0                1        23.0     0.0                0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
1           0
2           0
3           1
4           0
```

[15]:
```
df.describe()
```

[15]:
```
              male          age    education  currentSmoker   cigsPerDay  \
count  4238.000000  4238.000000  4133.000000    4238.000000  4209.000000
mean      0.429212    49.584946     1.978950       0.494101     9.003089
std       0.495022     8.572160     1.019791       0.500024    11.920094
min       0.000000    32.000000     1.000000       0.000000     0.000000
25%       0.000000    42.000000     1.000000       0.000000     0.000000
50%       0.000000    49.000000     2.000000       0.000000     0.000000
75%       1.000000    56.000000     3.000000       1.000000    20.000000
max       1.000000    70.000000     4.000000       1.000000    70.000000

            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4185.000000      4238.000000   4238.000000  4238.000000  4188.000000
mean      0.029630         0.005899      0.310524     0.025720   236.721585
std       0.169584         0.076587      0.462763     0.158316    44.590334
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   263.000000
max       1.000000         1.000000      1.000000     1.000000   696.000000

             sysBP        diaBP          BMI    heartRate      glucose  \
count  4238.000000  4238.000000  4219.000000  4237.000000  3850.000000
mean    132.352407    82.893464    25.802008    75.878924    81.966753
std      22.038097    11.910850     4.080111    12.026596    23.959998
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.070000    68.000000    71.000000
50%     128.000000    82.000000    25.400000    75.000000    78.000000
75%     144.000000    89.875000    28.040000    83.000000    87.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000
```

```
        TenYearCHD
count  4238.000000
mean      0.151958
std       0.359023
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

[17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4238 non-null   int64
 1   age              4238 non-null   int64
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64
 7   prevalentHyp     4238 non-null   int64
 8   diabetes         4238 non-null   int64
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

[19]:
```
df.isna().sum()
```

[19]:
```
male                 0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
```

```
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

[21]:

```
#Since, only a few rows have null values in them, we are only removing those␣
  ↪rows from the dataset.
#df = df.dropna(subset=['heartRate','BMI','cigsPerDay','totChol','BPMeds'])
```

[23]:

```
df
```

[23]:

```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
0        1   39        4.0              0         0.0     0.0
1        0   46        2.0              0         0.0     0.0
2        1   48        1.0              1        20.0     0.0
3        0   61        3.0              1        30.0     0.0
4        0   46        3.0              1        23.0     0.0
…     …  …         …             …          …      …
4233     1   50        1.0              1         1.0     0.0
4234     1   51        3.0              1        43.0     0.0
4235     0   48        2.0              1        20.0     NaN
4236     0   44        1.0              1        15.0     0.0
4237     0   52        2.0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                …            …         …       …     …     …     …
4233                0             1         0    313.0  179.0   92.0  25.97
4234                0             0         0    207.0  126.5   80.0  19.71
4235                0             0         0    248.0  131.0   72.0  22.00
4236                0             0         0    210.0  126.5   87.0  19.16
4237                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
0          80.0     77.0           0
1          95.0     76.0           0
2          75.0     70.0           0
3          65.0    103.0           1
```

```
4          85.0     85.0           0
…          …      …          …
4233       66.0     86.0           1
4234       65.0     68.0           0
4235       84.0     86.0           0
4236       86.0      NaN           0
4237       80.0    107.0           0

[4238 rows x 16 columns]
```

1 MISSING VALUE TREATMENT

Since, ‘glucose’ and ‘education’ columns had a significant amount of null values, so we replaced them with the mean of values for theirrespective columns

```
[27]: df['glucose'].fillna(value = df['glucose'].mean(),inplace=True)
```

```
[29]: df['education'].fillna(value = df['education'].mean(),inplace=True)
```

```
[31]: df['heartRate'].fillna(value = df['heartRate'].mean(),inplace=True)
```

```
[33]: df['BMI'].fillna(value = df['BMI'].mean(),inplace=True)
```

```
[35]: df['cigsPerDay'].fillna(value = df['cigsPerDay'].mean(),inplace=True)
```

```
[37]: df['totChol'].fillna(value = df['totChol'].mean(),inplace=True)
```

```
[39]: df['BPMeds'].fillna(value = df['BPMeds'].mean(),inplace=True)
```

```
[41]: df.isna().sum()
```

```
[41]: male               0
      age                0
      education          0
      currentSmoker      0
      cigsPerDay         0
      BPMeds             0
      prevalentStroke    0
      prevalentHyp       0
      diabetes           0
      totChol            0
      sysBP              0
      diaBP              0
      BMI                0
      heartRate          0
      glucose            0
```

```
TenYearCHD         0
dtype: int64
```

[43]:

```
#Splitting the dependent and independent variables.
x = df.drop("TenYearCHD",axis=1)
y = df['TenYearCHD']
```

[45]:

```
x #checking the features
```

[45]:

```
      male  age  education  currentSmoker  cigsPerDay   BPMeds  \
0        1   39        4.0              0         0.0  0.00000
1        0   46        2.0              0         0.0  0.00000
2        1   48        1.0              1        20.0  0.00000
3        0   61        3.0              1        30.0  0.00000
4        0   46        3.0              1        23.0  0.00000
…     …   …         …            …          …       …
4233     1   50        1.0              1         1.0  0.00000
4234     1   51        3.0              1        43.0  0.00000
4235     0   48        2.0              1        20.0  0.02963
4236     0   44        1.0              1        15.0  0.00000
4237     0   52        2.0              0         0.0  0.00000

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                 …            …        …       …     …     …     …
4233                0             1         0    313.0  179.0   92.0  25.97
4234                0             0         0    207.0  126.5   80.0  19.71
4235                0             0         0    248.0  131.0   72.0  22.00
4236                0             0         0    210.0  126.5   87.0  19.16
4237                0             0         0    269.0  133.5   83.0  21.47

      heartRate     glucose
0          80.0   77.000000
1          95.0   76.000000
2          75.0   70.000000
3          65.0  103.000000
4          85.0   85.000000
…           …         …
4233       66.0   86.000000
4234       65.0   68.000000
4235       84.0   86.000000
4236       86.0   81.966753
4237       80.0  107.000000
```

```
[4238 rows x 15 columns]
```

2 TRAIN TEST SPLIT

[48]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=42)
```

[50]:

```
y_train
```

[50]:

```
3252    0
3946    0
1261    0
2536    0
4089    0
       ..
3444    0
466     0
3092    0
3772    0
860     0
Name: TenYearCHD, Length: 3390, dtype: int64
```

3 Logistic Regression Algorithm

[53]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(x_train,y_train)
model.score(x_train, y_train)
```

[53]:

```
0.8489675516224189
```

[ ]:

```

```