Final_Project_Q1_JL

December 9, 2020

Jasper Lemberg
Professor Brendan Mort
DSCC 201
9 December 2020
Final Project - Question 1

1. The data set located at /public/bmort/python/heartdisease.csv contains a table of health data from 300 patients and includes age, sex, chest pain, blood pressure, cholesterol, blood sugar, ECG abnormality, heart rate, angina, ST value, ST slope, major vessel number, thal number, and whether or not the patient was diagnosed with heart disease (1 = yes, 0 = no). Use the Python 3 (anaconda3 2020.07) kernel and a Jupyter notebook and perform the following tasks: (50 points)

A. Import the heartdisease.csv data into a Pandas data frame.

In [1]:
```
import pandas as pd

heart_disease = pd.read_csv("/public/bmort/python/heartdisease.csv")
heart_disease
```

Out[1]:

	age	sex	pain	bp	chol	sugar	ecg	rate	angina	stv	sts	mvn	thal	disease
0	63	1	1	145	233	1	2	150.0	0	2.3	3	0	6	0
1	67	1	4	160	286	0	2	108.0	1	1.5	2	3	3	1
2	67	1	4	120	229	0	2	129.0	1	2.6	2	2	7	1
3	37	1	3	130	250	0	0	187.0	0	3.5	3	0	3	0
4	41	0	2	130	204	0	2	172.0	0	1.4	1	0	3	0
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...
295	45	1	1	110	264	0	0	132.0	0	1.2	2	0	7	0
296	68	1	4	144	193	1	0	141.0	0	3.4	2	2	7	1
297	57	1	4	130	131	0	0	115.0	1	1.2	2	1	7	1
298	57	0	2	130	236	0	2	174.0	0	0.0	2	1	3	1
299	35	1	2	122	192	0	0	174.0	0	0.0	1	0	3	0

```
4       3        0
..    ...      ...
295     7        1
296     7        1
297     7        1
298     3        1
299     3        0

[300 rows x 14 columns]
```

B. Is there any missing data in the data frame? What is missing? Perform any necessary data imputation before proceeding. Explain your reasoning for the choice made.

```
In [2]: heart_disease.isna().sum()
```

```
Out[2]: age        0
        sex        0
        pain       0
        bp         0
        chol       0
        sugar      0
        ecg        0
        rate       1
        angina     0
        stv        0
        sts        0
        mvn        0
        thal       0
        disease    0
        dtype: int64
```

There is one missing datum in the rate column. To fix this, I imputed the column's median in that spot, as shown by the code below.

```
In [3]: heart_disease["rate"] = heart_disease["rate"].fillna(heart_disease["rate"].median())
```

C. Check the summary statistics on the data. How do the ranges of the values in the columns compare? Does each column of data have similar magnitudes and ranges? Decide if you will perform any data-preprocessing. If you decide not to do any data preprocessing, explain why.

```
In [4]: heart_disease.describe()
```

```
Out[4]:               age         sex        pain          bp       chol       sugar  \
        count  300.000000  300.000000  300.000000  300.000000  300.00000  300.000000
        mean    54.480000    0.680000    3.153333  131.626667  246.93000    0.146667
        std      9.078049    0.467256    0.965884   17.687759   51.91798    0.354364
        min     29.000000    0.000000    1.000000   94.000000  126.00000    0.000000
```

```
25%     48.000000    0.000000    3.000000  120.000000  211.00000    0.000000
50%     56.000000    1.000000    3.000000  130.000000  241.50000    0.000000
75%     61.000000    1.000000    4.000000  140.000000  275.25000    0.000000
max     77.000000    1.000000    4.000000  200.000000  564.00000    1.000000

                ecg        rate      angina         stv         sts         mvn  \
count   300.000000  300.000000  300.000000  300.000000  300.000000  300.000000
mean      0.986667  149.773333    0.326667    1.049667    1.603333    0.670000
std       0.994881   22.834477    0.469778    1.162471    0.616920    0.936674
min       0.000000   71.000000    0.000000    0.000000    1.000000    0.000000
25%       0.000000  135.500000    0.000000    0.000000    1.000000    0.000000
50%       0.500000  153.000000    0.000000    0.800000    2.000000    0.000000
75%       2.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max       2.000000  202.000000    1.000000    6.200000    3.000000    3.000000

              thal    disease
count   300.000000  300.00000
mean      4.726667    0.46000
std       1.938508    0.49923
min       3.000000    0.00000
25%       3.000000    0.00000
50%       3.000000    0.00000
75%       7.000000    1.00000
max       7.000000    1.00000
```

```
In [5]: heart_disease[["age"]].max() - heart_disease[["age"]].min()
```

```
Out[5]: age    48
        dtype: int64
```

```
In [6]: heart_disease[["sex"]].max() - heart_disease[["sex"]].min()
```

```
Out[6]: sex    1
        dtype: int64
```

```
In [7]: heart_disease[["pain"]].max() - heart_disease[["pain"]].min()
```

```
Out[7]: pain    3
        dtype: int64
```

```
In [8]: heart_disease[["bp"]].max() - heart_disease[["bp"]].min()
```

```
Out[8]: bp    106
        dtype: int64
```

```
In [9]: heart_disease[["chol"]].max() - heart_disease[["chol"]].min()
```

```
Out[9]: chol    438
        dtype: int64
```

```
In [10]: heart_disease[["sugar"]].max() - heart_disease[["sugar"]].min()

Out[10]: sugar    1
         dtype: int64

In [11]: heart_disease[["ecg"]].max() - heart_disease[["ecg"]].min()

Out[11]: ecg    2
         dtype: int64

In [12]: heart_disease[["rate"]].max() - heart_disease[["rate"]].min()

Out[12]: rate    131.0
         dtype: float64

In [13]: heart_disease[["angina"]].max() - heart_disease[["angina"]].min()

Out[13]: angina    1
         dtype: int64

In [14]: heart_disease[["stv"]].max() - heart_disease[["stv"]].min()

Out[14]: stv    6.2
         dtype: float64

In [15]: heart_disease[["sts"]].max() - heart_disease[["sts"]].min()

Out[15]: sts    2
         dtype: int64

In [16]: heart_disease[["mvn"]].max() - heart_disease[["mvn"]].min()

Out[16]: mvn    3
         dtype: int64

In [17]: heart_disease[["thal"]].max() - heart_disease[["thal"]].min()

Out[17]: thal    4
         dtype: int64

In [18]: heart_disease[["disease"]].max() - heart_disease[["disease"]].min()

Out[18]: disease    1
         dtype: int64
```

Age, blood pressure, cholestrol, heart rate, and stv all have ranges over 3 and standard deviations over 1. Therefore, these columns are not all on the same range. In order to remedy this, I have standardized those columns, as shown in the code below.

```
In [19]: from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()
         heart_disease[["age"]] = scaler.fit_transform(heart_disease[["age"]])
         heart_disease[["bp"]] = scaler.fit_transform(heart_disease[["bp"]])
         heart_disease[["chol"]] = scaler.fit_transform(heart_disease[["chol"]])
         heart_disease[["rate"]] = scaler.fit_transform(heart_disease[["rate"]])
         heart_disease[["stv"]] = scaler.fit_transform(heart_disease[["stv"]])

         heart_disease.dtypes

Out[19]: age        float64
         sex          int64
         pain         int64
         bp         float64
         chol       float64
         sugar        int64
         ecg          int64
         rate       float64
         angina       int64
         stv        float64
         sts          int64
         mvn          int64
         thal         int64
         disease      int64
         dtype: object
```

D. Partition your data into a training set (80%) and a testing set (20%) that is randomly selected from the heartdisease.csv data.

```
In [20]: from sklearn.model_selection import train_test_split

         X = heart_disease.drop(['disease'], axis = 1)
         y = heart_disease["disease"]
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

E. Using logistic regression as provided by the Scikit-Learn library, develop a model to predict heart disease diagnosis based on the 13 features provided in the data set for each patient. Using Scikit-Learn's k-fold cross-validation function, perform a 5-fold cross-validation. What is the accuracy of the model according to the k-fold crossvalidation step?

```
In [21]: from sklearn.linear_model import LogisticRegression

         model = LogisticRegression(solver = "liblinear")
         model.fit(X_train, y_train)

Out[21]: LogisticRegression(solver='liblinear')
```

```
In [22]: from sklearn.model_selection import KFold, cross_val_score

         kfold = KFold(n_splits = 5, shuffle = True)
         scores = cross_val_score(model, X_train, y_train, cv = kfold)
         print("The accuracy of the model is %0.2f plus or minus %0.2f."
               % (scores.mean(), scores.std()))
```

```
The accuracy of the model is 0.82 plus or minus 0.04.
```

F. Generate a confusion matrix using the data from your test set to show the accuracy of the model. Comment on the accuracy of the model in your Python notebook. What percent are false positives? What percent are false negatives?

```
In [23]: from sklearn.metrics import confusion_matrix

         y_predicted = model.predict(X_test)
         cm = confusion_matrix(y_test, y_predicted)
         cm
```

```
Out[23]: array([[27,  6],
                [ 5, 22]])
```

```
In [24]: print("%0.2f percent of the results are false positives \nand %0.2f percent of the res
               % (cm[1][0]/len(X_test), cm[0][1]/len(X_test)))
```

```
0.08 percent of the results are false positives
and 0.10 percent of the results are false negatives.
```