# untitled

October 3, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv("Heart.csv")
```

[3]:
```
print("Number of records in each label are")
print(df['target'].value_counts())
```

```
Number of records in each label are
target
1    165
0    138
Name: count, dtype: int64
```

[4]:
```
print("\nPercentage of records in each label are")
print(df['target'].value_counts() * 100 / df.shape[0], "\n")
```

```


Percentage of records in each label are
target
1    54.455446
0    45.544554
Name: count, dtype: float64

```

[5]:
```
df.head()
```

[5]:
```
   Unnamed: 0  age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  \
0           0   63    1   3       145   233    1        0      150      0
1           1   37    1   2       130   250    0        1      187      0
2           2   41    0   1       130   204    0        0      172      0
3           3   56    1   1       120   236    0        1      178      0
4           4   57    0   0       120   354    0        1      163      1

   oldpeak  slope  ca  thal  target
0      2.3      0   0     1       1
```

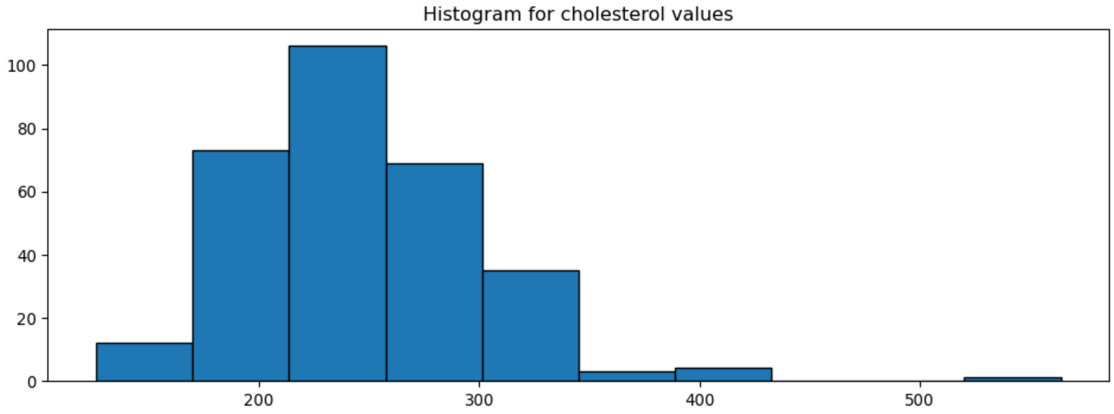
```
1      3.5      0   0     2       1
2      1.4      2   0     2       1
3      0.8      2   0     2       1
4      0.6      2   0     2       1
```

[6]:
```python
def sigmoid(x):
    return pd.Series(1 / ( 1 + np.exp(-x)))
```

[7]:
```python
df['chol'].describe()
```

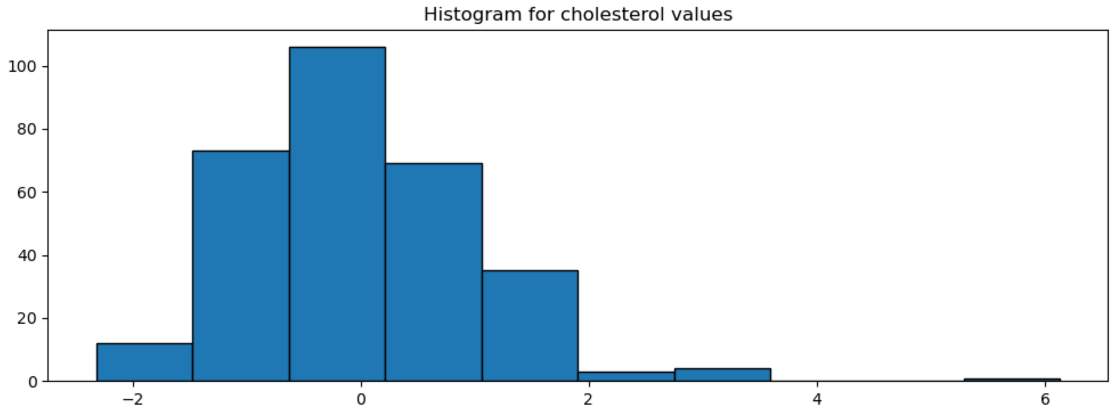
[7]:
```
count    303.000000
mean     246.264026
std       51.830751
min      126.000000
25%      211.000000
50%      240.000000
75%      274.500000
max      564.000000
Name: chol, dtype: float64
```

[8]:
```python
plt.figure(figsize = (12,4), dpi = 96)
plt.title("Histogram for cholesterol values")
plt.hist(df['chol'], bins = 'sturges', edgecolor = 'black')
plt. show()
```



[9]:
```python
def standard_scalar(series):
    new_series = (series - series.mean()) / series.std()
    return new_series
scaled_chol = standard_scalar(df['chol'])
```

[10]:
```python
plt.figure(figsize = (12,4))
plt.title("Histogram for cholesterol values")
plt.hist(scaled_chol, bins = 'sturges', edgecolor = 'black')
plt.show()
```



[11]:

```python
chol_sig_output = sigmoid(df['chol'])
chol_sig_output.describe()
```

[11]:

```
count    303.0
mean       1.0
std        0.0
min        1.0
25%        1.0
50%        1.0
75%        1.0
max        1.0
Name: chol, dtype: float64
```

[12]:

```python
scaled_chol_sig_output = sigmoid(scaled_chol)
scaled_chol_sig_output.describe()
```

[12]:

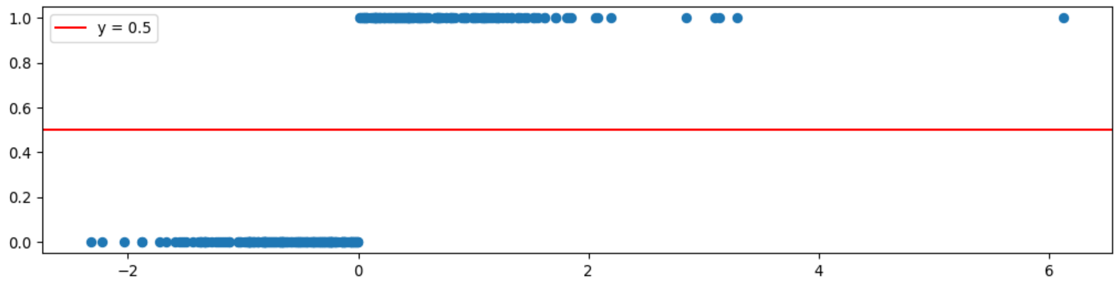
```
count    303.000000
mean       0.492837
std        0.198175
min        0.089454
25%        0.336179
50%        0.469823
75%        0.632919
max        0.997829
Name: chol, dtype: float64
```

[13]:

```python
def predict(sig_output, threshold):
    y_pred = [ 1 if output >= threshold else 0 for output in sig_output]
    return pd.Series(y_pred)
```

[14]:
```
threshold = 0.5
heart_disease_pred = predict(scaled_chol_sig_output, threshold)
```

[15]:
```
plt.figure(figsize=(13,3), dpi = 96)
plt.scatter(scaled_chol, heart_disease_pred)
plt.axhline(y = threshold, label = f'y = { threshold }', color = 'r')
plt. legend()
plt.show()
```



[16]:
```
print(f"Threshold value: {threshold}")
print(f"\nPredicted value counts:\n{heart_disease_pred.value_counts()}")
print(f"\nActual value counts:\n{df['target']. value_counts()}")
```

```
Threshold value: 0.5

Predicted value counts:
0    167
1    136
Name: count, dtype: int64

Actual value counts:
target
1    165
0    138
Name: count, dtype: int64
```

[17]:
```
from sklearn.metrics import confusion_matrix
```

[18]:
```
print(confusion_matrix(df['target'], heart_disease_pred))
```

```
[[ 65  73]
 [102  63]]
```

[19]:
```
from sklearn.metrics import classification_report
```

[20]:
```
print(classification_report(df['target'], heart_disease_pred))
```

```
              precision    recall  f1-score   support

           0       0.39      0.47      0.43       138
           1       0.46      0.38      0.42       165

    accuracy                           0.42       303
   macro avg       0.43      0.43      0.42       303
weighted avg       0.43      0.42      0.42       303
```

[21]:

```python
from sklearn.model_selection import train_test_split
```

[22]:

```python
X = df.drop(columns = 'target')
y = df['target']
```

[23]:

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

[24]:

```python
from sklearn.linear_model import LogisticRegression
```

[25]:

```python
log_clf_1 = LogisticRegression()
log_clf_1.fit(X_train, y_train)
print(log_clf_1.score(X_train, y_train))
```

```
1.0
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[26]:

```python
y_train_pred = log_clf_1.predict(X_train)
```

[27]:

```python
print("\n Confusion Matrix \n")
print(confusion_matrix(y_train, y_train_pred))
```

```

 Confusion Matrix 

[[ 97   0]
 [  0 115]]
```

```
[28]: print("\n Classification Report\n")
      print(classification_report(y_train, y_train_pred))
```

```

 Classification Report

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        97
           1       1.00      1.00      1.00       115

    accuracy                           1.00       212
   macro avg       1.00      1.00      1.00       212
weighted avg       1.00      1.00      1.00       212

```

```
[29]: y_test_pred = log_clf_1.predict(X_test)
```

```
[30]: print(f"{'Test Set'.upper()}\n{'-' * 75}\nConfusion Matrix:")
      print(confusion_matrix(y_test, y_test_pred))
```

```
TEST SET
---------------------------------------------------------------------------
Confusion Matrix:
[[40  1]
 [ 1 49]]
```

```
[31]: print("\nClassification Report")
      print(classification_report(y_test, y_test_pred))
```

```

Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        41
           1       0.98      0.98      0.98        50

    accuracy                           0.98        91
   macro avg       0.98      0.98      0.98        91
weighted avg       0.98      0.98      0.98        91

```

```
[32]: def standard_scaler(series):
        new_series = (series - series.mean()) / series.std()
        return new_series
```

[33]:

```
norm_X_train = X_train.apply(standard_scaler, axis = 0)
norm_X_test = X_test.apply(standard_scaler, axis = 0)
```

[34]:

```
norm_X_train.describe()
norm_X_test.describe()
```

[34]:

```
        Unnamed: 0           age           sex            cp      trestbps  \
count  9.100000e+01  9.100000e+01  9.100000e+01  9.100000e+01  9.100000e+01
mean  -1.249001e-16 -2.488852e-16 -1.146824e-16 -9.760202e-18 -7.051746e-16
std    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min   -1.644808e+00 -2.301763e+00 -1.661622e+00 -8.425578e-01 -1.853721e+00
25%   -8.333169e-01 -8.354271e-01 -1.661622e+00 -8.425578e-01 -6.650121e-01
50%   -6.722410e-02  1.797284e-01  5.952080e-01 -8.425578e-01 -1.662530e-02
75%    9.258592e-01  6.309086e-01  5.952080e-01  1.123410e+00  4.696648e-01
max    1.703301e+00  2.435630e+00  5.952080e-01  2.106394e+00  3.549502e+00

              chol           fbs    restecg       thalach         exang  \
count  9.100000e+01  9.100000e+01  91.000000  9.100000e+01  9.100000e+01
mean  -4.148086e-17 -5.490114e-17   0.000000 -5.343711e-16 -4.880101e-18
std    1.000000e+00  1.000000e+00   1.000000  1.000000e+00  1.000000e+00
min   -2.624853e+00 -4.938276e-01  -0.943037 -3.319275e+00 -7.148350e-01
25%   -7.201088e-01 -4.938276e-01  -0.943037 -6.418709e-01 -7.148350e-01
50%   -1.836075e-02 -4.938276e-01  -0.943037  1.078023e-01 -7.148350e-01
75%    6.165541e-01 -4.938276e-01   0.963994  6.432832e-01  1.383552e+00
max    3.679740e+00  2.002745e+00   2.871025  1.864180e+00  1.383552e+00

           oldpeak         slope            ca          thal
count  9.100000e+01  9.100000e+01  9.100000e+01  9.100000e+01
mean   1.082808e-16 -9.516197e-17 -1.464030e-17  1.903239e-16
std    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min   -8.367971e-01 -2.184053e+00 -8.102615e-01 -3.491486e+00
25%   -8.367971e-01 -5.812398e-01 -8.102615e-01 -4.364358e-01
50%   -3.799059e-01 -5.812398e-01 -8.102615e-01 -4.364358e-01
75%    5.719508e-01  1.021573e+00  9.246514e-01  1.091089e+00
max    3.884412e+00  1.021573e+00  2.659564e+00  1.091089e+00
```

[35]:

```
from sklearn.feature_selection import RFE
from sklearn.metrics import f1_score
from sklearn.linear_model import LogisticRegression
```

[36]:

```
dict_rfe = {}
```

[37]:

```
for i in range(1, len(X_train.columns) + 1):
  lg_clf_2 = LogisticRegression()
  rfe = RFE(lg_clf_2,n_features_to_select=i)
  rfe.fit(norm_X_train, y_train)
  rfe_features = list(norm_X_train.columns[rfe.support_])
  rfe_X_train = norm_X_train[rfe_features]

  lg_clf_3 = LogisticRegression()
  lg_clf_3.fit(rfe_X_train, y_train)

  y_test_pred = lg_clf_3.predict(norm_X_test[rfe_features])

  f1_scores_array = f1_score(y_test, y_test_pred, average = None)
  dict_rfe[i] = {"features": list(rfe_features), "f1_score": f1_scores_array}
dict_rfe
```

```
[37]: {1: {'features': ['Unnamed: 0'], 'f1_score': array([0.98795181, 0.98989899])},
       2: {'features': ['Unnamed: 0', 'oldpeak'],
        'f1_score': array([0.98765432, 0.99009901])},
       3: {'features': ['Unnamed: 0', 'exang', 'oldpeak'],
        'f1_score': array([0.98765432, 0.99009901])},
       4: {'features': ['Unnamed: 0', 'exang', 'oldpeak', 'thal'],
        'f1_score': array([0.97560976, 0.98      ])},
       5: {'features': ['Unnamed: 0', 'restecg', 'exang', 'oldpeak', 'thal'],
        'f1_score': array([0.97560976, 0.98      ])},
       6: {'features': ['Unnamed: 0', 'sex', 'restecg', 'exang', 'oldpeak', 'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       7: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'restecg',
         'exang',
         'oldpeak',
         'thal'],
        'f1_score': array([0.97560976, 0.98      ])},
       8: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'restecg',
         'exang',
         'oldpeak',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       9: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'restecg',
         'exang',
         'oldpeak',
         'slope',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       10: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'chol',
         'restecg',
         'exang',
         'oldpeak',
         'slope',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       11: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'trestbps',
         'chol',
         'restecg',
         'exang',
         'oldpeak',
         'slope',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       12: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'trestbps',
         'chol',
         'fbs',
         'restecg',
         'exang',
         'oldpeak',
         'slope',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       13: {'features': ['Unnamed: 0',
         'sex',
         'cp',
         'trestbps',
         'chol',
         'fbs',
         'restecg',
         'thalach',
         'exang',
         'oldpeak',
         'slope',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])},
       14: {'features': ['Unnamed: 0',
         'age',
         'sex',
         'cp',
         'trestbps',
         'chol',
         'fbs',
         'restecg',
         'thalach',
         'exang',
         'oldpeak',
         'slope',
         'ca',
         'thal'],
        'f1_score': array([0.96385542, 0.96969697])}}
```

[38]:
```
pd.options.display.max_colwidth = 100
f1_df = pd.DataFrame.from_dict(dict_rfe, orient = 'index')
f1_df
```

[38]:
```
              features  \
1
[Unnamed: 0]
2
[Unnamed: 0, oldpeak]
3
[Unnamed: 0, exang, oldpeak]
4                                                                   [Unnamed: 0,
exang, oldpeak, thal]
5                                                          [Unnamed: 0, restecg,
exang, oldpeak, thal]
6                                                     [Unnamed: 0, sex, restecg,
exang, oldpeak, thal]
7                                                 [Unnamed: 0, sex, cp, restecg,
exang, oldpeak, thal]
8                                             [Unnamed: 0, sex, cp, restecg,
exang, oldpeak, ca, thal]
9                                      [Unnamed: 0, sex, cp, restecg, exang,
oldpeak, slope, ca, thal]
10                               [Unnamed: 0, sex, cp, chol, restecg, exang,
oldpeak, slope, ca, thal]
```

```
11                     [Unnamed: 0, sex, cp, trestbps, chol, restecg, exang,
oldpeak, slope, ca, thal]
12                [Unnamed: 0, sex, cp, trestbps, chol, fbs, restecg, exang,
oldpeak, slope, ca, thal]
13       [Unnamed: 0, sex, cp, trestbps, chol, fbs, restecg, thalach, exang,
oldpeak, slope, ca, thal]
14  [Unnamed: 0, age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang,
oldpeak, slope, ca, thal]

                                    f1_score
1     [0.9879518072289156, 0.98989898989899]
2   [0.9876543209876543, 0.9900990099009901]
3   [0.9876543209876543, 0.9900990099009901]
4                  [0.975609756097561, 0.98]
5                  [0.975609756097561, 0.98]
6    [0.963855421686747, 0.9696969696969697]
7                  [0.975609756097561, 0.98]
8    [0.963855421686747, 0.9696969696969697]
9    [0.963855421686747, 0.9696969696969697]
10   [0.963855421686747, 0.9696969696969697]
11   [0.963855421686747, 0.9696969696969697]
12   [0.963855421686747, 0.9696969696969697]
13   [0.963855421686747, 0.9696969696969697]
14   [0.963855421686747, 0.9696969696969697]
```

[39]:

```python
lg_clf_4 = LogisticRegression()
rfe = RFE(lg_clf_4, n_features_to_select = 3)
```

[40]:

```python
rfe.fit(norm_X_train, y_train)
rfe_features = norm_X_train.columns[rfe.support_]
print(rfe_features)
final_X_train = norm_X_train[rfe_features]
```

```
Index(['Unnamed: 0', 'exang', 'oldpeak'], dtype='object')
```

[ ]:

```

```