Machine Learning with Logistical Regression on Heart Disease Dataset: Predicting If A Patient Will Have Heart Disease

by Joshua Nguyen

Abstract Logistical Regression is a statistical machine learning algorithm that models the classification of a binary response variable. It predicts the probability of an outcome while using a logistic function. Overall, **Logistic regressional analysis** describes and estimates the relationship between one dependent binary variable and one or more independent variables - provides a probability score for observations. This is a python project done in Visual Studio Code.

Import Libraries and Dataset

Before we build our heart disease prediction model, we will import neccessary python libraries and the HeartDisease dataset which can downloaded at the following link: https://www.kaggle.com/fedesoriano/heart-failure-prediction.

```
# Import Libraries
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn import metrics

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

```
# Import Dataset and Obtain First Four Rows
data = pd.read_csv("heart.csv")
data = data.dropna()
data.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Data Cleaning

```
# Creating Dummy Variables For Categorical Variables
mapping = {0: 'Normal', 1: 'High'}
data['FastingBS_code'] = data.loc[:, 'FastingBS']
data = data.replace({'FastingBS_code' : mapping})
data.head()

sex_mapping = {'M':0, 'F':1}
data['Sex_code'] = data.loc[:, 'Sex']
data = data.replace({'Sex_code' : sex_mapping})

CPT_mapping = {'ATA':0,'NAP':1,'ASY':2,'TA':3}
data['CPT_code'] = data.loc[:, 'ChestPainType']
data = data.replace({'CPT_code' : CPT_mapping})

ECG_mapping = {'Normal':0,'ST' : 1,'LVH':2}
data['ECG_code'] = data.loc[:, 'RestingECG']
data = data.replace({'ECG_code' : ECG_mapping})

data.head()
```

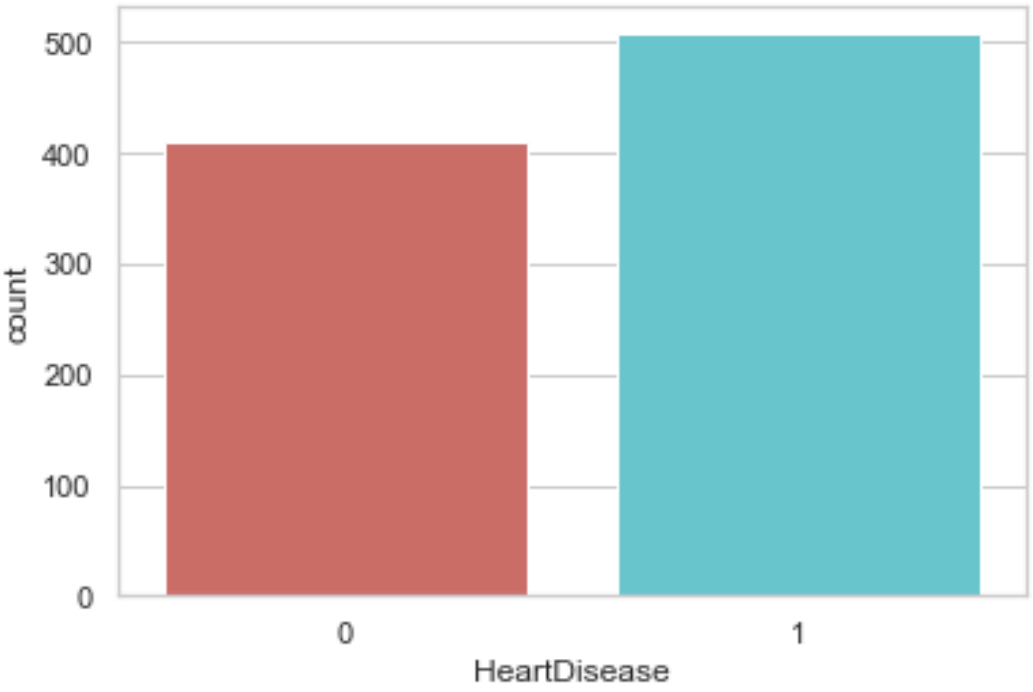
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	FastingBS_code	Sex_code	CPT_code	ECG_code
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0	Normal	0	0	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1	Normal	1	1	0
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0	Normal	0	0	1
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1	Normal	1	2	0
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0	Normal	0	1	0

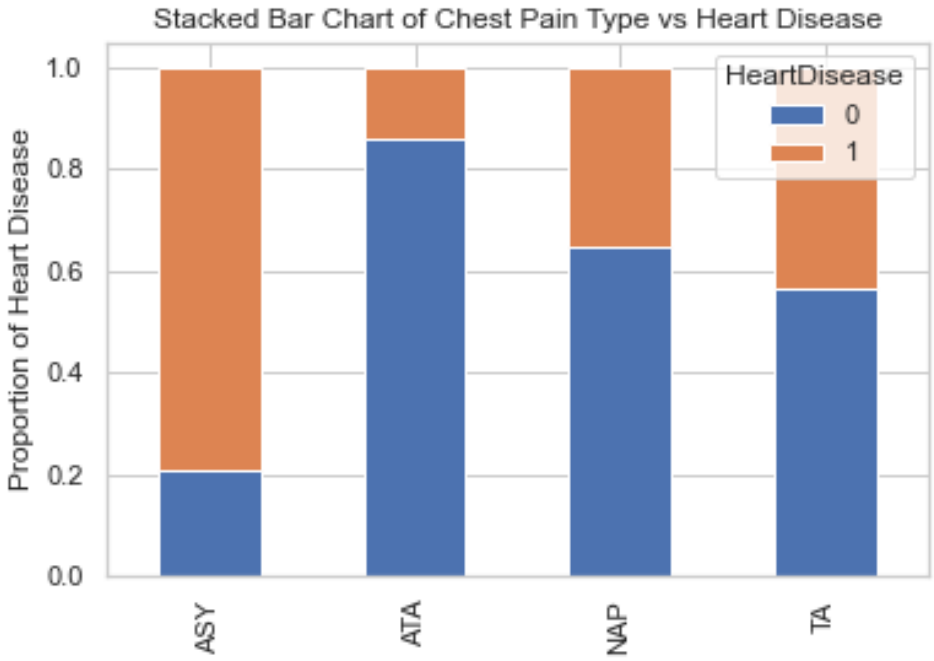
Exploratory Data Analysis

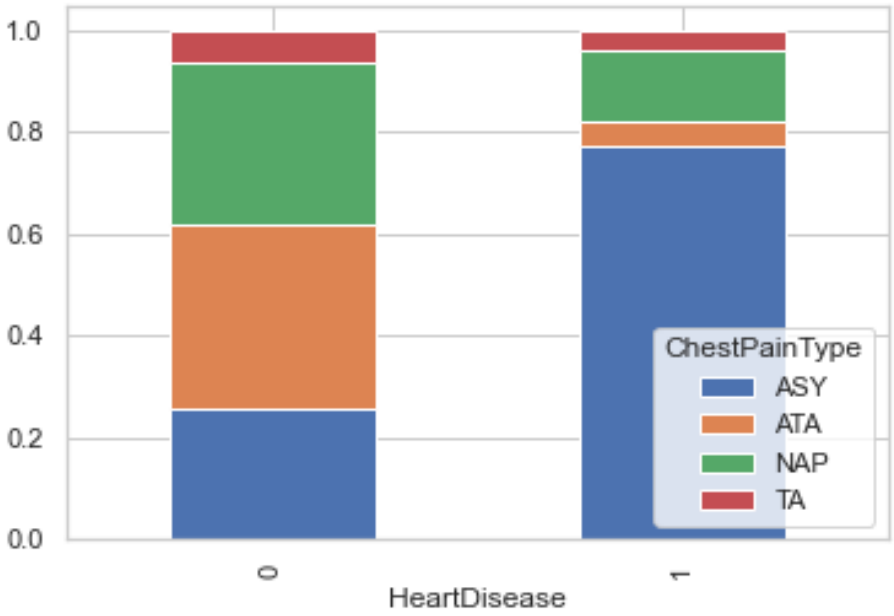
```
# Table of Heart Disease
data['HeartDisease'].value_counts()

1    508
0    410
Name: HeartDisease, dtype: int64
```

```
# Boxplot of Heart Disease
sns.countplot(x = 'HeartDisease', data = data, palette= 'hls')
plt.show()
plt.savefig('count_plot')
```

```
# Heart Disease Frequency Plots
table=pd.crosstab(data.ChestPainType,data.HeartDisease)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Chest Pain Type vs Heart Disease')
plt.xlabel('Chest Pain Type')
plt.ylabel('Proportion of Heart Disease')
plt.savefig('CPT_vs_HD')

table=pd.crosstab(data.HeartDisease,data.ChestPainType)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.savefig('CPT_vs_HD_V2')
```


500
400
300
200
100
0
count
0
1
HeartDisease


Stacked Bar Chart of Chest Pain Type vs Heart Disease
HeartDisease
0
1
1.0
0.8
0.6
0.4
0.2
0.0
Proportion of Heart Disease
ASY
ATA
NAP
TA



Interestingly, we see that for our group including patients with *heart disease*, the majority of the patients exhibit asymptomatic chest pain type, meaning they experience no chest pain whatsoever. Additionally, the majority of patients who exhibit no symptoms of chest pain apparently have heart disease.

Logistical Regressional Analysis

Selecting Feature Variables

Here, we will divide the columns into the two independent or dependent variables type. Our dependent variables (heart disease) is the target variable, and the independent variables (others) are considered the feature variables.

```
# Divide Columns Into Two Types of Variables: independent and dependent
feature_cols = ['Age', 'Sex_code', 'CPT_code', 'RestingBP', 'Cholesterol',
↪  'FastingBS', 'ECG_code', 'MaxHR', 'Oldpeak' ]

X = data[feature_cols]
y = data.HeartDisease
```

Implementing Model

We will use the *logit_model* function to run a summary of our model.

```
# Model Summary
logit_model = sm.Logit(y,X)
result = logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
Current function value: 0.432866
Iterations 6
Results: Logit
=================================================================
Model: Logit Pseudo R-squared: 0.370
Dependent Variable: HeartDisease AIC: 812.7415
```

```
Date: 2022-01-17 14:05 BIC: 856.1413
No. Observations: 918 Log-Likelihood: -397.37
Df Model: 8 LL-Null: -631.07
Df Residuals: 909 LLR p-value: 6.9165e-96
Converged: 1.0000 Scale: 1.0000
No. Iterations: 6.0000
----------------------------------------------------------------------
Coef. Std.Err. z P>|z| [0.025 0.975]
----------------------------------------------------------------------
Age 0.0177 0.0089 1.9920 0.0464 0.0003 0.0351
Sex_code -1.2163 0.2318 -5.2470 0.0000 -1.6706 -0.7620
CPT_code 0.8846 0.1115 7.9351 0.0000 0.6661 1.1031
RestingBP 0.0058 0.0043 1.3650 0.1723 -0.0025 0.0142
Cholesterol -0.0025 0.0009 -2.7780 0.0055 -0.0043 -0.0007
FastingBS 1.0587 0.2317 4.5692 0.0000 0.6046 1.5129
ECG_code -0.0030 0.1128 -0.0263 0.9790 -0.2241 0.2182
MaxHR -0.0212 0.0030 -7.0563 0.0000 -0.0270 -0.0153
Oldpeak 0.7989 0.1004 7.9568 0.0000 0.6021 0.9957
======================================================================
```

Notice that the p-values for RestingBP and ECG_code are not statistically significant. We will keep this in mind when fitting our model.

Splitting the Data

Next, we will need to divide the dataset into a training and testing set. This will help us understand our model performance. I will divide my dataset such that the training to testing ration is 7:3. That is, 70 percent of the data will be used for training the model and the final 30 percent is utilized for testing our model.

```
# Testing and Training Sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,
↪  random_state= 0)
```

Model Development and Prediction

Using the LogisticRegression function, we will next create a logistic regression classifier object. After, we will fit our model on the training set and start performing predictions on the training set. This will help us predict the test set results and calculate the accuracy of our model above.

```
# Initiate and fitting model
logreg = LogisticRegression()
logreg.fit(X_train,y_train)
y_pred = logreg.predict(X_test)
```

Evaluating Model With a Confusion Matrix

In summary, we can use a **confusion matrix** in order to evaluate the performance of our classification model above. Here, we will use *metrics* from the **sklearn** library.

```
# Confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test,y_pred)
cnf_matrix
```

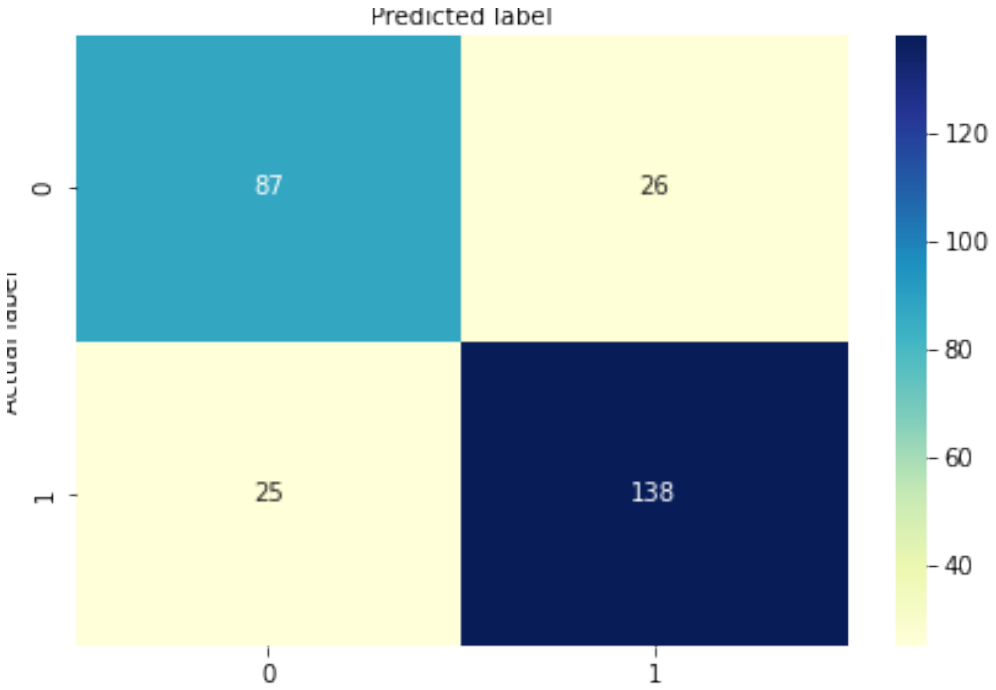
```
array([[ 87, 26],
    [ 25, 138]], dtype=int64)
```

Ergo, we see that our model has $87 + 138 = 225$ actual correct predictions and $25 + 26 = 51$ incorrect predictions - not bad!

Visualizing and Evaluating Confusion Matrix

Next, we will create a visualization of our confusion matrix to help us intrepet its results.

```
# Heatmap
class_names = [0,1]
fig, ax = plt.subplots()
tick_marks = np.arrange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

sns.heatmap(pd.DataFrame(cnf_matrix),annot=True,cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```




```
# Confusion Matrix Evaluation Metrics
print("Accuracy:",metrics.accuracy_score(y_test,y_pred)*100)
print("Precision:",metrics.precision_score(y_test,y_pred)*100)
print("Recall:",metrics.recall_score(y_test,y_pred)*100)
```

```
Accuracy: 81.52173913043478
Precision: 84.14634146341463
Recall: 84.66257668711657
```

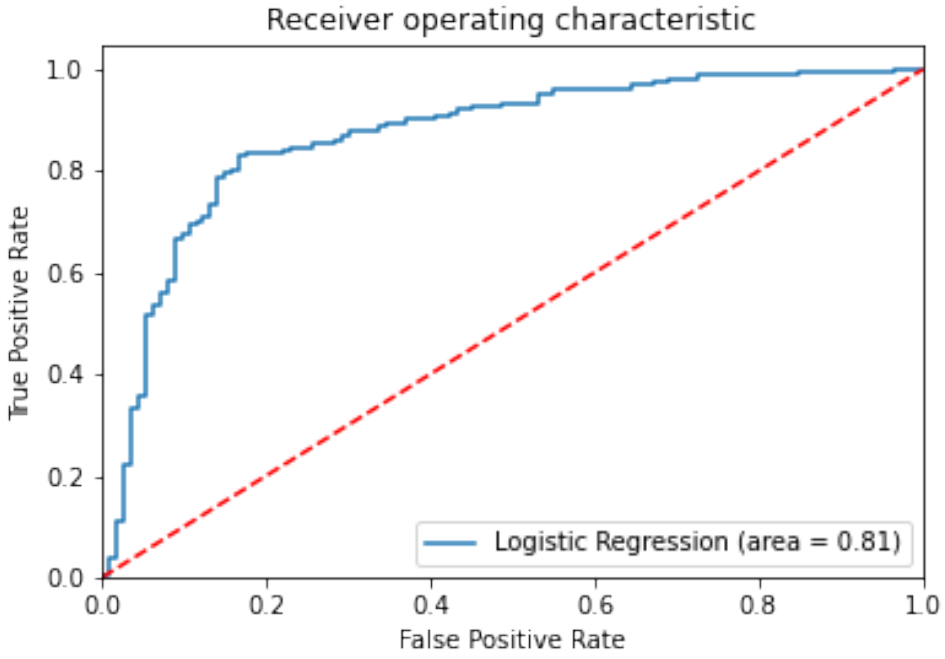
Above, we see that our classification percentage is 81% - which is considered good accuracy! Our prediction and recall percentage are both 84% which is also considered good. In other words, when our model predicted that a patient will have heart disease, those patients did in fact have heart disease about 84% of the time. Additionally, when there existed a patient with heart disease in our testing set, our model was able to correctly identify it 84% of the time.

Receiver Operating Characteristic (ROC) Curve

A ROC curve will plot the true positive rate against the false positive rate. Essentially, it shows the tradeoff between sensitivity and specificity. The dotted line will represent the ROC curve of a

random classifier. Moreover, a good classifier will deviate away from that line as much as possible and towards the top-left corner.

```
# ROC Curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
plt.savefig('ROC_FIG')
```



In our model, we obtained a AUC (area under ROC curve) score of 0.81. A AUC value closer to 1 represents a perfect classifier, while a AUC score of 0.5 indicates poor classifier.

Conclusion

In this project, we have used Logistical Regression, a form of machine learning, to build a classifying model on a Heart Disease dataset and built their corresponding plots to visualize and help interpret results.