CSCI 380 Project 2
Heart Disease UCI using Logistic Regression

Ryan Fazal, Darnell Regis & Marc Suda May.14th, 2019

Abstract
This paper is based upon using Logistics Regression. It'll be used in order to provide an insight on if a patient has a heart disease. The reason for this, is so that I can predict whether or not a patient is likely to develop a heart disease. This will be on particular attributes that could affect the various data provided. All data provided will be used to predict the target. The project includes data gathered using the sklearn free libraries. Also, 2D visualizations is used to show how the data is set based upon a scatter plot.

1 Introduction

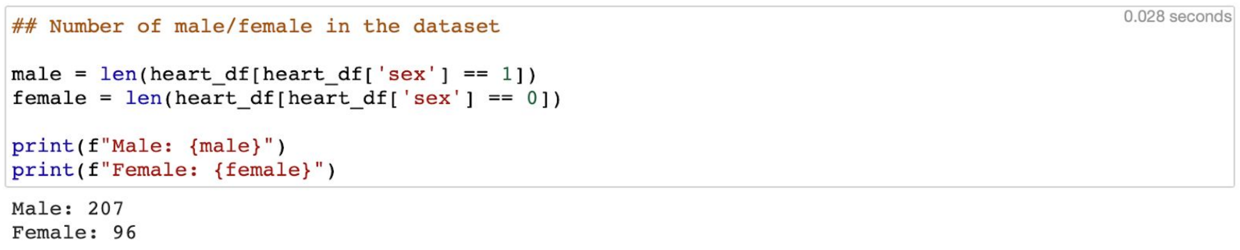
As heart disease is on the rise with individuals at a younger age, facilities are trying to find the signs as to how it'll occur. The various variables that could be indicators are cholesterol, blood pressure, age, etc. The scope of this project is to see how heart disease varies among patients. It is a way of signing what variables vary amongst individuals who have a heart disease or not. More specifically, our objective for this project, is to take all the attributes provided, to build a model in order to identify whether these attributes can identify if a heart disease is present or not. These attributes vary in comparison from age with chest pain type, or age with serum cholesterol etc. This data can be helpful in seeing the causes of a heart disease. If we can pinpoint the attributes that one heart disease patient has, we can determine if a patient without a heart disease can develop one. This is if they have similar or the exact same attributes at play with their health.

2 Dataset

2.1 Explanation of the dataset

The dataset that we have collected for research is the Heart Disease UCI[1]. This dataset that we decided to follow, had experiments which were constructed by the Cleveland database. This database included 303 patients with data from 14 attributes. The ages of the patients ranged from 29 to 78. The 14 attributes consisted of age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, rest electrocardiographic results, maximum heart rate achieved, exercise induced angina, oldpeak, slope, number of major blood vessels, and thal. For this database, we decided to therefore use logistic regression in order to predict whether or not a patient was able to develop a heart disease from one of various attributes.

To start off, we named the file as "heart_df". From there, we used a "heart_df.describe()" command. This command gave us the statistics of the heart.csv file. These statistics included the count, mean, standard deviation, and minimum/maximum of the 14 attributes of the dataset. Next, we found out that there were 207 males and 96 females in the dataset.

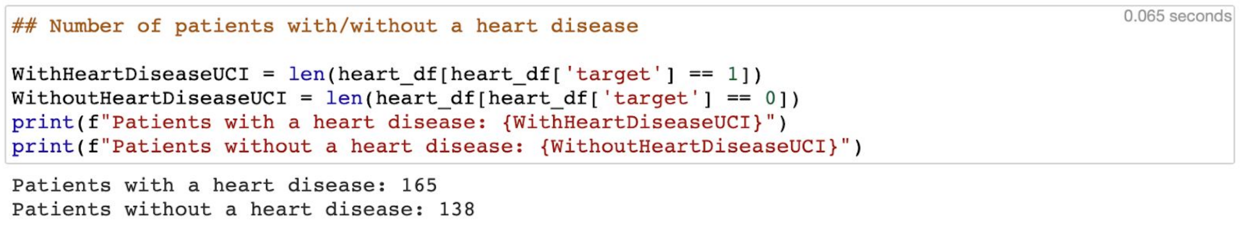


```
## Number of male/female in the dataset

male = len(heart_df[heart_df['sex'] == 1])
female = len(heart_df[heart_df['sex'] == 0])

print(f"Male: {male}")
print(f"Female: {female}")
```

```
Male: 207
Female: 96
```

Figure 1: Number of Male/Female in Dataset

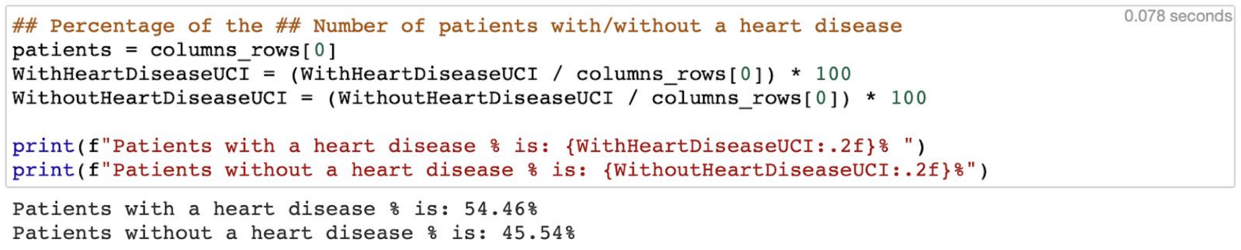
Following that, we determined the amount of patients with or without a heart disease. Patients with a heart disease were 165 or 54.46%. This varied with the patients without a heart disease, which was 138 or 45.54%.



```
## Number of patients with/without a heart disease

WithHeartDiseaseUCI = len(heart_df[heart_df['target'] == 1])
WithoutHeartDiseaseUCI = len(heart_df[heart_df['target'] == 0])
print(f"Patients with a heart disease: {WithHeartDiseaseUCI}")
print(f"Patients without a heart disease: {WithoutHeartDiseaseUCI}")
```

```
Patients with a heart disease: 165
Patients without a heart disease: 138
```

Figure 2: Number of Patient with/without a Heart Disease



```
## Percentage of the ## Number of patients with/without a heart disease
patients = columns_rows[0]
WithHeartDiseaseUCI = (WithHeartDiseaseUCI / columns_rows[0]) * 100
WithoutHeartDiseaseUCI = (WithoutHeartDiseaseUCI / columns_rows[0]) * 100

print(f"Patients with a heart disease % is: {WithHeartDiseaseUCI:.2f}% ")
print(f"Patients without a heart disease % is: {WithoutHeartDiseaseUCI:.2f}%")
```

```
Patients with a heart disease % is: 54.46%
Patients without a heart disease % is: 45.54%
```

Figure 3: Percentage of Patients with/without a Heart Disease

The dataset was now been split into a train and test. To test our data, we used the sklearn libraries. We decided to use x = (x -x.mean())/x.std() as our normalized data. For the training, it was set at 80%. However, the test was set at 20%. This was used in order to test which patients will have or won't have a heart disease. As the data came back, it was equal to 83.61%. To ensure that our data was correct, we used the command "performance = logreg.score(x_test, y_test)." This was used to test the performance of the x_test,y_test of the database. The result of this was 0.8360655737704918. This high accuracy score proves that if a patient has an attribute from the dataset, they can develop a heart disease.

3 Data Visualization

3.1 Scatter Plots

In order for us to present our data in a more distinctive manner, we had to create various visualizations. For this, we first decided to create a scatter plot. This scatter plot included the attributes of age and serum_chlesterol. The purpose of this test, was to see the correlation of the two attributes. It was also used to map out if these individuals that had these attributes also had a heart disease or not.

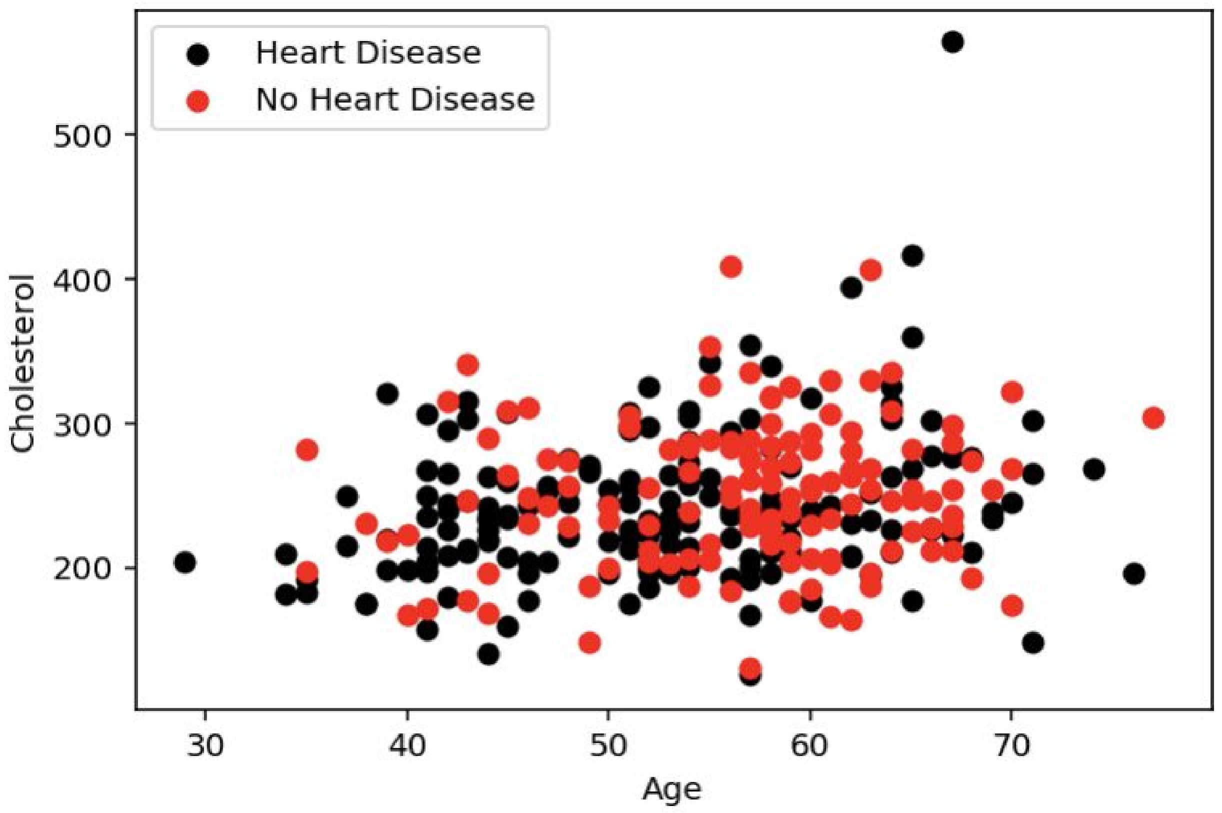


Figure 4: Scatter plot with data of age/cholesterol

Looking at the scatter showed that the higher the age in accordance with cholesterol, there were more patients without a heart disease. However, The younger the patients were, they seemed to develop a heart disease. Specifically, around the ages of 40-60 there were much more patients with a heart disease and a high cholesterol.

3.2 Seaborn library using Pairplots

Next, we decided to import the seaborn library. This was in order to create a pairplot that would help visualize the data of specific attributes. For this test, we chose “chest_pain_type”, “no_of_mjrv_cf”,”slope”, and “thal.” The reason for this, was that they had values which ranged beyond 0 to 1. Chest pain type went from 0 to 3, number of major vessels 0 to 2, slope 0 to 2 and thal 0 to 3. One important note for "chest_pain_type," is that the values mean: Value 1 is typical angina, value 2 is atypical angina, value 3 is non-anginal pain, and Value 4: asymptomatic [1]. These values represent the type of chest pain an individual may be having at that point. This test was to show how their data would be plotted out amongst one another.

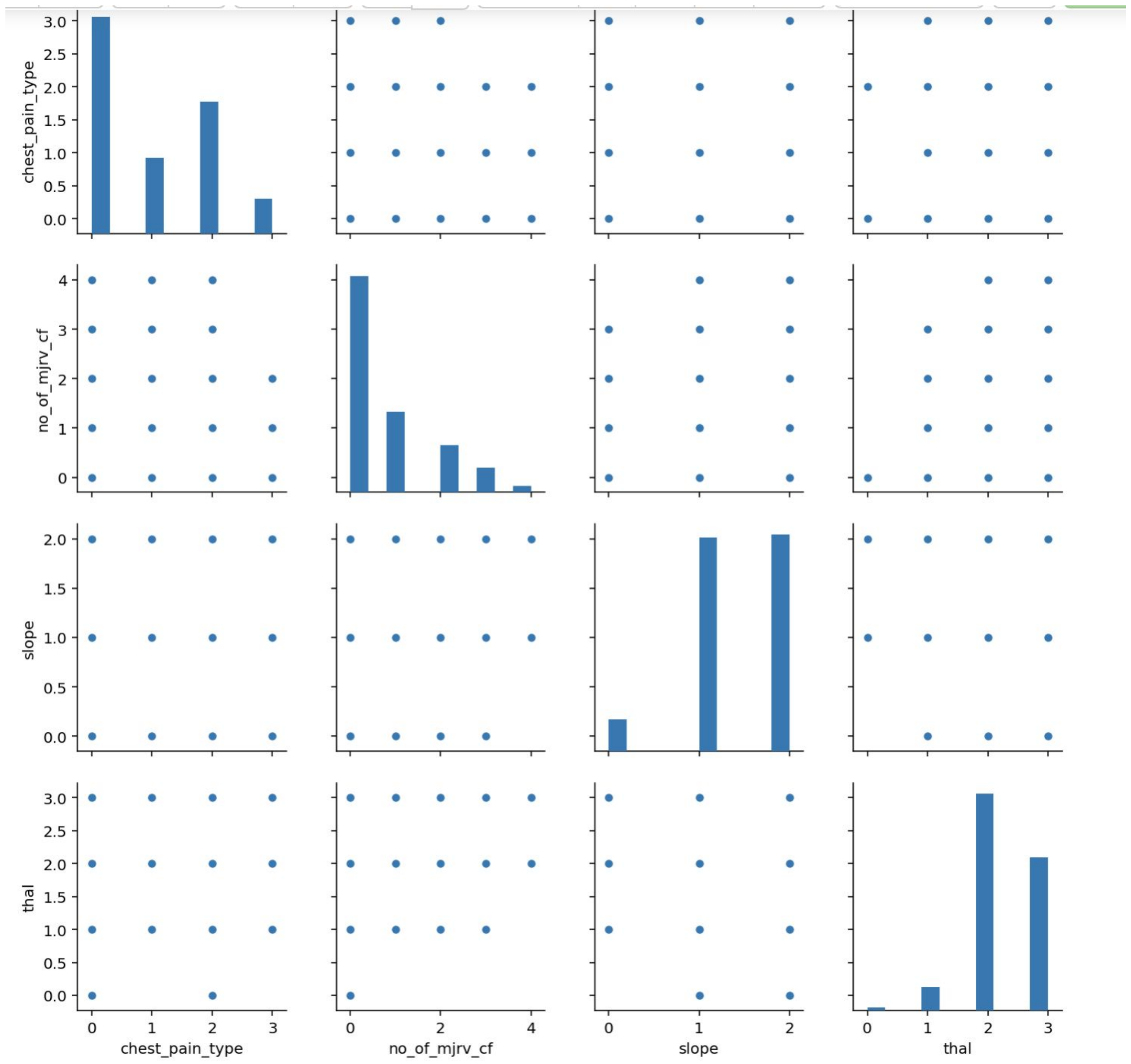


Figure 5: Seaborn Pairplot of chest_pain_type", "no_of_mjrv_cf","slope", and "thal."

Following this, we implemented the "kind='reg' command. This creates a linear regression upon our data . This would be useful in getting the estimation for regression.

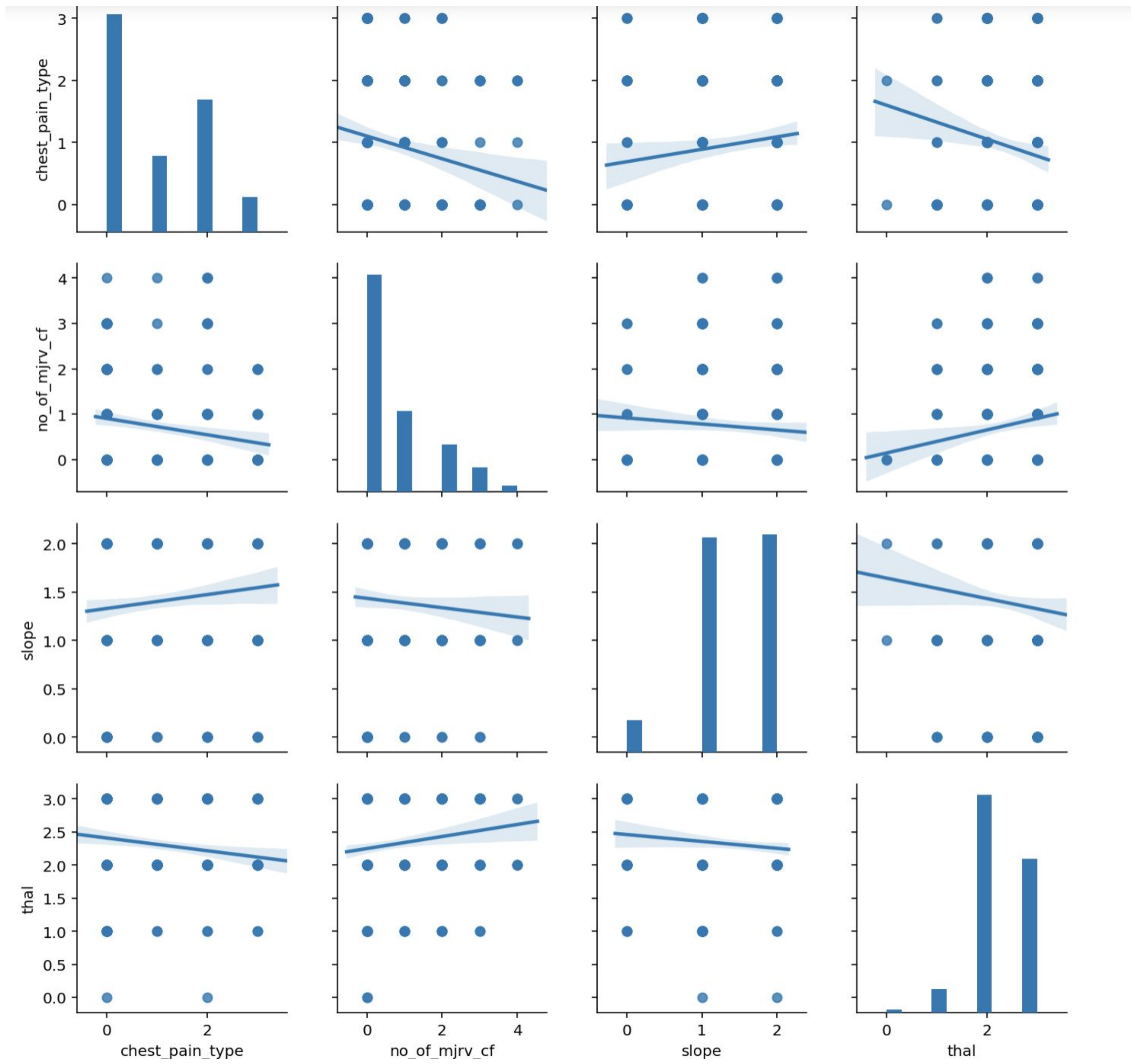


Figure 6: Seaborn Pairplot with linear regression of chest_pain_type", "no_of_mjrv_cf","slope", and "thal."

To add some distinction to our pairplot, we changed the hue to "chest_pain_type." This however created an issue as the pairplot came out peculiar. This was because the dots plotted out ended up overlapping on one another.

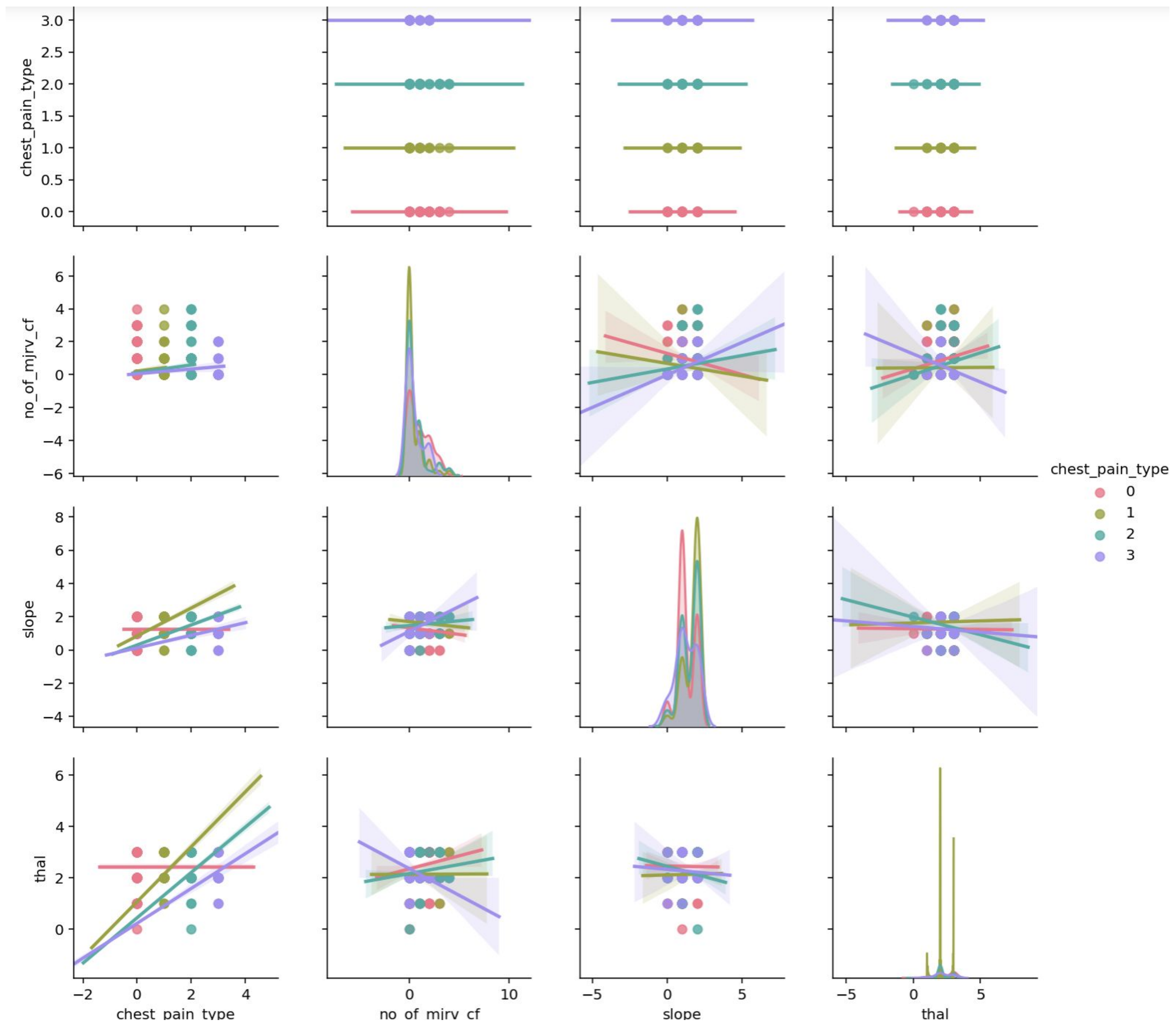
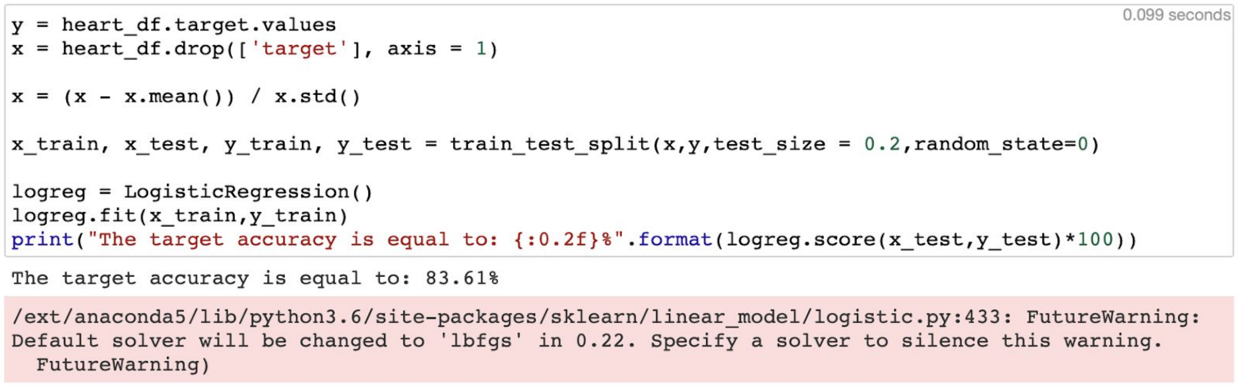


Figure 7: Seaborn Pairplot of chest_pain_type", "no_of_mjrv_cf","slope", and "thal" with palette change.

4 Results

4.1 Finding out who was what in the dataset

Our dataset was merely based off of observations to extract the data that we needed. Our first goal was to see how was the accuracy for a patient having a heart disease within this dataset. Below is a figure showing the commands used, and the percentage score we ended up obtaining.



```
0.099 seconds
y = heart_df.target.values
x = heart_df.drop(['target'], axis = 1)

x = (x - x.mean()) / x.std()

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=0)

logreg = LogisticRegression()
logreg.fit(x_train,y_train)
print("The target accuracy is equal to: {:0.2f}%".format(logreg.score(x_test,y_test)*100))
```

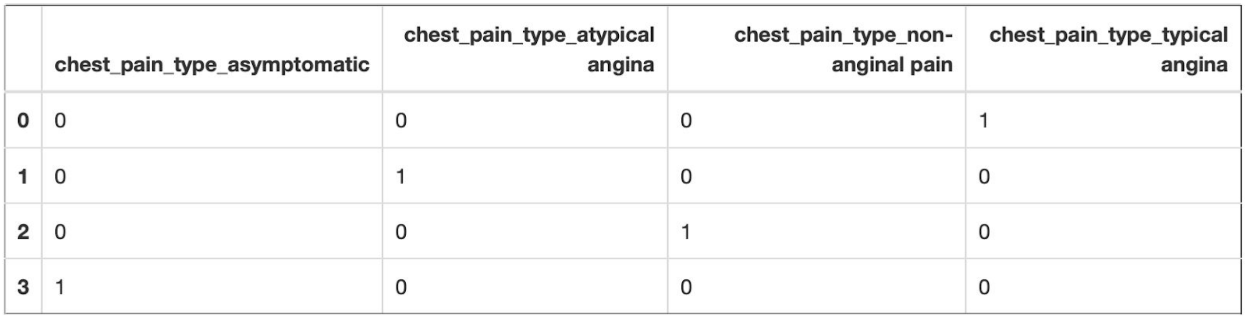
```
The target accuracy is equal to: 83.61%
```

```
/ext/anaconda5/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Figure 8: Target Accuracy test

4.2 Implementing One-Hot Encoding

However some of the data collected at first sure did throw us a curveball. Since the chest_pain_type from the dataset was nominal data, we had to take a different approach. It had values set for the range of chest pain. Thus making them arbitrary Because of this, we needed to use one-hot encoding. The purpose of this, was to be able to achieve a purpose of using the values of the chest_pain_type. The values would be turned into vector operations. Now it was time to implement our data. To do so, we started off with the pandas library. It was used to set the DataFrame for both the chest_pain_type and the values. The command used was: “heart_df = pd.DataFrame({'chest_pain_type': ['typical angina', 'atypical angina', 'non-anginal pain','asymptomatic']}).” Following that, we used the dummies function in order to take the values given and place them as a variable that has an indication. The command for this was : “pd.get_dummies(heart_df,prefix=['chest_pain_type'])”

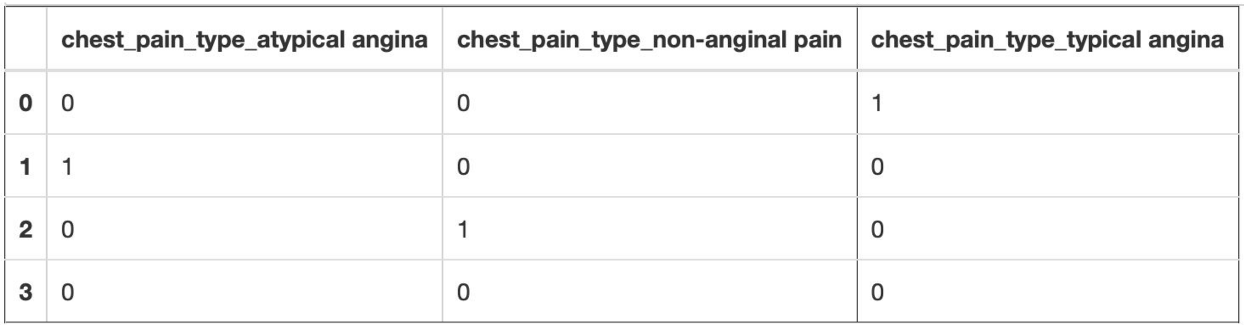


	chest_pain_type_asymptomatic	chest_pain_type_atypical angina	chest_pain_type_non-anginal pain	chest_pain_type_typical angina
0	0	0	0	1
1	0	1	0	0
2	0	0	1	0
3	1	0	0	0

Figure 9: One-Hot Encoding

The figure above shows the values 0-3, which for the dataset of the original chest_pain_type would be values 1-4. The figure has a set of 0-1 place across it. If one of the columns has 0 then there is no pain. If the column as a 1 then there is a pain with that corresponding value.

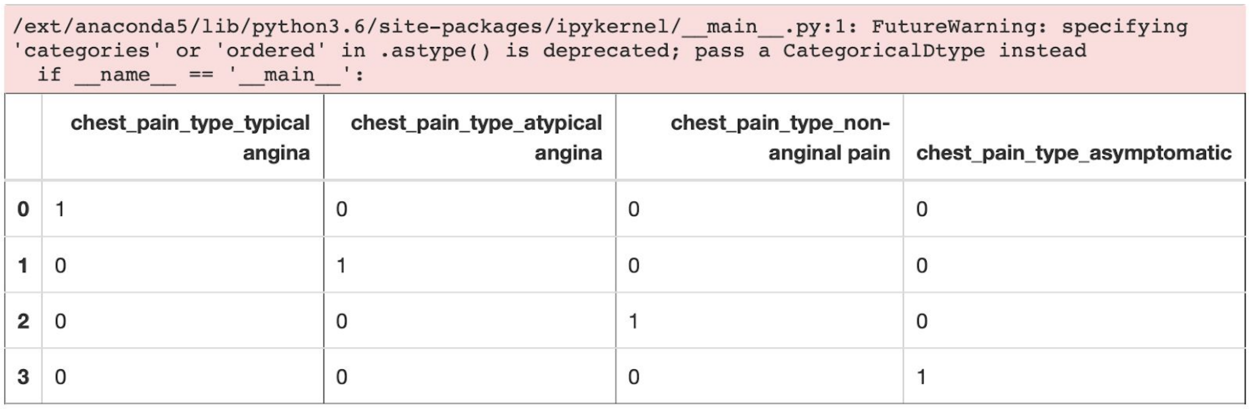
Next, we used drop_first=True command to remove the first level of dummy variables. For this, we used the command “pd.get_dummies(heart_df,prefix=['chest_pain_type'], drop_first=True).”



	chest_pain_type_atypical angina	chest_pain_type_non-anginal pain	chest_pain_type_typical angina
0	0	0	1
1	1	0	0
2	0	1	0
3	0	0	0

Figure 10: One-Hot Encoding without Asymptomatic

The figure above shows that the asymptomatic column has been removed. This was due to the fact that the column would not serve any purpose when looking for a patient with a heart disease. Instead it'll search for a patient that has no sign of chest pain, which could hinder the data when trying to find a heart disease patient. Lastly for one-hot encoding, we had to add columns for the categories. This was in order for the dummy columns to be created for the test set. We even had to include asymptomatic, even though the value would revert back to 0. For this, we used the following commands: "heart_df["chest_pain_type"] = heart_df["chest_pain_type"].astype('category',categories=['typical angina', 'atypical angina', 'non-anginal pain','asymptomatic'])", and then "pd.get_dummies(heart_df,prefix=['chest_pain_type'])."



```
/ext/anaconda5/lib/python3.6/site-packages/ipykernel/__main__.py:1: FutureWarning: specifying
'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instead
  if __name__ == '__main__':
```

	chest_pain_type_typical angina	chest_pain_type_atypical angina	chest_pain_type_non-anginal pain	chest_pain_type_asymptomatic
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

Figure 11: One-Hot Encoding with test variables

5 Conclusion

To conclude, our data achieved a goal that helped to certify of original hypothesis. We believed that heart disease could've been recognized within this dataset. This was if we combined the attributes such as age, and serum_chlestorel. Testing this data had produced a high accuracy level. One that had a less than 20% chance of failure of occuring. To make sure, we even carried out a prediction test of the logistic regression in sklearn, in order for us to get an accurate reading. And once more, we ended up getting the same result. So what does this mean for the

dataset? Well for the dataset we used for our research, it is clear that we can identify the accuracy of if a patient has a heart disease. This is if they have one of the particular attributes.
Our data did end up running into some trouble as we ventured forward. Take in fact the chest_pain_type data. It had values that were nominal. This is the reason as to why the data, when placed as a hue, would not appear in the pairplots. We had to use one-hot encoding in order to see if it'll help to improve our performance of our testing. Also for the pairplots, the data was laying over one another, due to the fact that the values were set in a range of 0 to 3. Thus, causing an assortment of confusion.
However, these instances did not hinder our performance when creating our data. Instead it allowed us to find more data in order to show how our theory began to play out upon the dataset.

6 Reference

[1]Ronit. "Heart Disease UCI." ***Kaggle*****, 25 June 2018, www.kaggle.com/ronitf/heart-disease-uci.**