

Factors Responsible for Heart ♥ Attack Using Machine Learning- Predicting by Logistic Regression & Random Forest Classifier

▼ Data Wrangling

#Necessary Imports

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

#Read the file to using pandas(pd)

```
data =pd.read_excel('/content/data.xlsx')
```

```
data.head(5)
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

Read the columns, Rows and get the shape of the two-dimensional dataframe

```
print("(Rows, columns): " + str(data.shape))
data.columns
```

```
(Rows, columns): (303, 14)
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

data.nunique(axis=0)# returns the number of unique values for each variable.

```
age      41
sex       2
cp        4
trestbps 49
chol     152
fbs       2
restecg   3
thalach   91
exang     2
oldpeak   40
slope     3
ca        5
thal      4
target    2
dtype: int64
```

```
#summarizes the count, mean, standard deviation, min, and max for numeric variables.
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg
--	-----	-----	----	----------	------	-----	---------

```
# Display the Missing Values

print(data.isna().sum())

age      0
sex      0
cp       0
trestbps 0
chol     0
fbs      0
restecg  0
thalach  0
exang    0
oldpeak  0
slope    0
ca       0
thal     0
target   0
dtype: int64
```

No null values identified, pretty clean data!

Lets see if theirs a good proportion between our positive & negative binary predictor.

```
data['target'].value_counts()

1    165
0    138
Name: target, dtype: int64
```

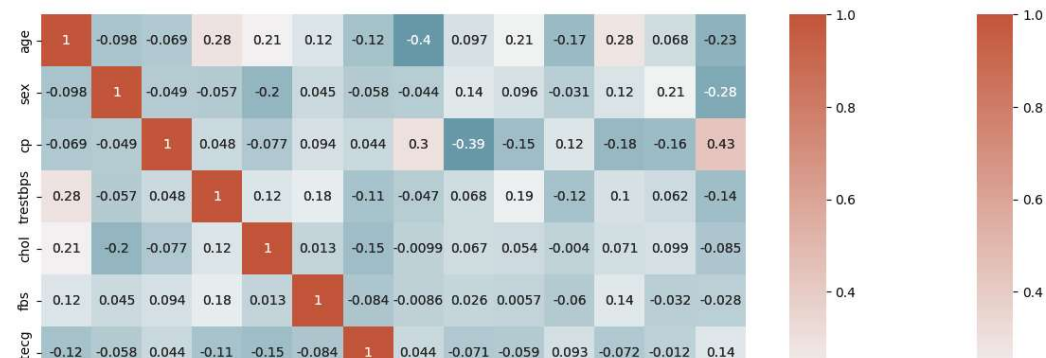
Correlation Matrix- let's you see correlations between all variables.

Within seconds, you can see whether something is positively or negatively correlated with our predictor (target).

```
# calculate correlation matrix

corr = data.corr()
plt.subplots(figsize=(15,10))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging_palette(220, 20, as_cmap=True))
sns.heatmap(corr, xticklabels=corr.columns,
            yticklabels=corr.columns,
            annot=True,
            cmap=sns.diverging_palette(220, 20, as_cmap=True))
```

<Axes: >



We can see there is a positive correlation between chest pain (cp) & target (our predictor). This makes sense since, the greater amount of chest pain results in a greater chance of having heart disease. Cp (chest pain), is a ordinal feature with 4 values: Value 1: typical angina ,Value 2: atypical angina, Value 3: non-anginal pain , Value 4: asymptomatic.

In addition, we see a negative correlation between exercise induced angina (exang) & our predictor. This makes sense because when you exercise, your heart requires more blood, but narrowed arteries slow down blood flow.



Using pairplots to see the continuous columns variable correlation

```
subData = data[['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]
sns.pairplot(subData)
```

```
<seaborn.axisgrid.PairGrid at 0x7e5e117bb850>
```



Making a smaller pairplot with only the continuous variables, to dive deeper into the relationships. Also a great way to see if there's a positive or negative correlation!

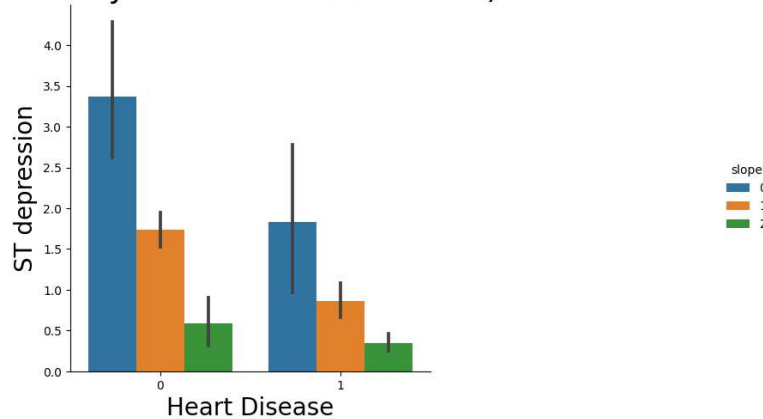


```
sns.catplot(x="target", y="oldpeak", hue="slope", kind="bar", data=data);
```

```
plt.title('ST depression (induced by exercise relative to rest) vs. Heart Disease',size=25)
plt.xlabel('Heart Disease',size=20)
plt.ylabel('ST depression',size=20)
```

```
Text(36.804208333333335, 0.5, 'ST depression')
```

ST depression (induced by exercise relative to rest) vs. Heart Disease



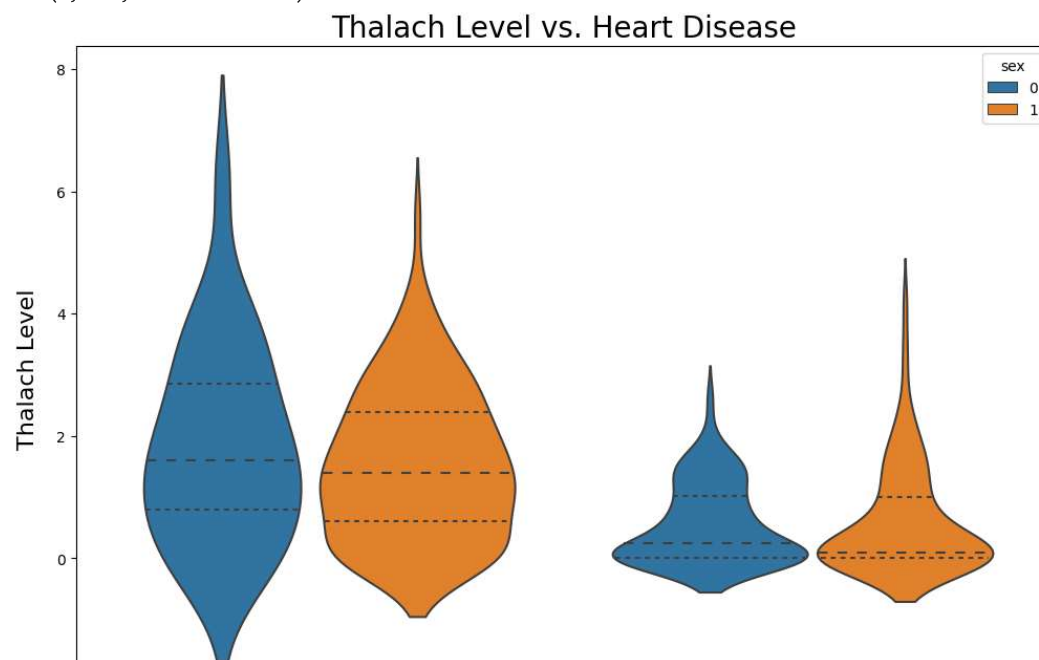
ST segment depression occurs because when the ventricle is at rest and therefore repolarized. If the trace in the ST segment is abnormally low below the baseline, this can lead to this Heart Disease. This supports the plot above because low ST Depression yields people at greater risk for heart disease. While a high ST depression is considered normal & healthy. The "slope" hue, refers to the peak exercise ST segment, with values: 0: upsloping, 1: flat, 2: downsloping). Both positive & negative heart disease patients exhibit equal distributions of the 3 slope categories.*

Outlier detection Using box plot and violin plot

```
# Violin plot
```

```
plt.figure(figsize=(12,8))
sns.violinplot(x= 'target', y= 'oldpeak',hue="sex", inner='quartile',data= data )
plt.title("Thalach Level vs. Heart Disease",fontsize=20)
plt.xlabel("Heart Disease Target", fontsize=16)
plt.ylabel("Thalach Level", fontsize=16)
```

```
Text(0, 0.5, 'Thalach Level')
```




We can see that the overall shape & distribution for negative & positive patients differ vastly. Positive patients exhibit a lower median for ST depression level & thus a great distribution of their data is between 0 & 2, while negative patients are between 1 & 3. In addition, we don't see many differences between male & female target outcomes.

```
# Box Plot
```

```
plt.figure(figsize=(12,8))
sns.boxplot(x= 'target', y= 'thalach',hue="sex", data=data )
plt.title("ST depression Level vs. Heart Disease", fontsize=20)
plt.xlabel("Heart Disease Target",fontsize=16)
plt.ylabel("ST depression induced by exercise relative to rest", fontsize=16)
```

Text(0, 0.5, 'ST depression induced by exercise relative to rest')

ST depression Level vs. Heart Disease



Positive patients exhibit a heightened median for ST depression level, while negative patients have lower levels. In addition, we don't see many differences between male & female target outcomes, expect for the fact that males have slightly larger ranges of ST Depression.

Filtering data by positive & negative Heart Disease patient

```
# Filtering data by POSITIVE Heart Disease patient
pos_data = data[data['target']==1]
pos_data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
count	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.
mean	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.
std	9.550651	0.497444	0.952222	16.169613	53.552872	0.347412	0.504818	19.174276	0.
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	96.000000	0.
25%	44.000000	0.000000	1.000000	120.000000	208.000000	0.000000	0.000000	149.000000	0.
50%	52.000000	1.000000	2.000000	130.000000	234.000000	0.000000	1.000000	161.000000	0.
75%	59.000000	1.000000	2.000000	140.000000	267.000000	0.000000	1.000000	172.000000	0.
max	76.000000	1.000000	3.000000	180.000000	564.000000	1.000000	2.000000	202.000000	1.

```
# Filtering data by NEGATIVE Heart Disease patient
pos_data = data[data['target']==0]
pos_data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
count	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.
mean	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.
std	7.962082	0.380416	0.905920	18.729944	49.454614	0.367401	0.541321	22.598782	0.
min	35.000000	0.000000	0.000000	100.000000	131.000000	0.000000	0.000000	71.000000	0.
25%	52.000000	1.000000	0.000000	120.000000	217.250000	0.000000	0.000000	125.000000	0.
50%	58.000000	1.000000	0.000000	130.000000	249.000000	0.000000	0.000000	142.000000	1.
75%	62.000000	1.000000	0.000000	144.750000	283.000000	0.000000	1.000000	156.000000	1.
max	77.000000	1.000000	3.000000	200.000000	409.000000	1.000000	2.000000	195.000000	1.

```
# Filtering data by NEGATIVE Heart Disease patient
neg_data = data[data['target']==0]
neg_data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
count	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.
mean	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.
std	7.962082	0.380416	0.905920	18.729944	49.454614	0.367401	0.541321	22.598782	0.
min	35.000000	0.000000	0.000000	100.000000	131.000000	0.000000	0.000000	71.000000	0.
25%	52.000000	1.000000	0.000000	120.000000	217.250000	0.000000	0.000000	125.000000	0.
50%	58.000000	1.000000	0.000000	130.000000	249.000000	0.000000	0.000000	142.000000	1.
75%	62.000000	1.000000	0.000000	144.750000	283.000000	0.000000	1.000000	156.000000	1.
max	77.000000	1.000000	3.000000	200.000000	409.000000	1.000000	2.000000	195.000000	1.

```

print("(Positive Patients ST depression): " + str(pos_data['oldpeak'].mean()))
print("(Negative Patients ST depression): " + str(neg_data['oldpeak'].mean()))

(Positive Patients ST depression): 1.5855072463768116
(Negative Patients ST depression): 1.5855072463768116

print("(Positive Patients thalach): " + str(pos_data['thalach'].mean()))
print("(Negative Patients thalach): " + str(neg_data['thalach'].mean()))

(Positive Patients thalach): 139.1014492753623
(Negative Patients thalach): 139.1014492753623

```

From comparing positive and negative patients we can see there are vast differences in means for many of our 13 Features. From examining the details, we can observe that positive patients experience heightened maximum heart rate achieved (thalach) average. In addition, positive patients exhibit about 1/3rd the amount of ST depression induced by exercise relative to rest (oldpeak).

▼ Machine Learning + Predictive Analytics

Prepare Data for Modeling

To prepare data for modeling, just remember ASN (Assign, Split, Normalize).

Assign the 13 features to X, & the last column to our classification predictor, y

```

X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split: the data set into the Training set and Test set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 1)

#Normalize: Standardizing the data will transform the data so that its distribution will have a
#mean of 0 and a standard deviation of 1

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

# Prediction using the Classification model Logistic Regression

from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression

model1 = LogisticRegression(random_state=1) # get instance of model
model1.fit(x_train, y_train) # Train/Fit model

y_pred1 = model1.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred1)) # output accuracy

```

	precision	recall	f1-score	support
0	0.77	0.67	0.71	30
1	0.71	0.81	0.76	31
accuracy			0.74	61
macro avg	0.74	0.74	0.74	61
weighted avg	0.74	0.74	0.74	61

Achieved 74% accuracy

```

# Prediction using Random Forest Classifier

from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model6 = RandomForestClassifier(random_state=1)# get instance of model
model6.fit(x_train, y_train) # Train/Fit model

```

```
y_pred6 = model6.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred6)) # output accuracy
```

	precision	recall	f1-score	support
0	0.88	0.70	0.78	30
1	0.76	0.90	0.82	31
accuracy			0.80	61
macro avg	0.82	0.80	0.80	61
weighted avg	0.81	0.80	0.80	61

Achieved 80% accuracy

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred6)
print(cm)
accuracy_score(y_test, y_pred6)
```

```
[[21  9]
 [ 3 28]]
0.8032786885245902
```

Thus, from confusion matrix we conclude a good outcome as 80% is the ideal accuracy!