# Heart Disease Prediction

April 7, 2020

## 1 Introduction

[ ]:
```
#Libraries are imported
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import QuantileTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

[ ]:
```
data = pd.read_csv("../input/heart-disease-uci/heart.csv") #For loading data␣
↪using pandas and data is a data frame.
```

[ ]:
```
data.head() # First five rows of data frame can be seen using this.
```

[ ]:
```
data.isnull().sum() #For checking if any column has null values or not and␣
↪number of null values.
```

[ ]:
```
data.dtypes #Data type of each column.
```

[ ]:
```
data['cp'].unique() #Checking unique values in the column in cp.
```

[ ]:
```
data["fbs"].unique() #Checking unique values in the column in fbs.
```

[ ]:
```
data["restecg"].unique() #Checking unique values in the column in restecq.
```

[ ]:
```
data["exang"].unique()
```

[ ]:
```
data["slope"].unique()
```

[ ]:
```
data['ca'].unique()
```

[ ]:
```
data['thal'].unique()
```

## 2 Data Visualization

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', hue = 'sex', data = data, palette =␣
↪'pink')
plt.title("Gender and Target")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', hue = 'exang', data = data, palette =␣
↪'pastel')
plt.title("Exang and Target")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', hue = 'fbs', data = data, palette =␣
↪'pink')
plt.title("Fbs and Target")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'ca', kind = 'count', data = data, palette = 'pastel')
plt.title("Count of Ca")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'thal', kind = 'count', data = data, palette = 'pink')
plt.title("Count of Thal")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'sex', kind = 'count', data = data, palette = 'pink')
plt.title("Count of Gender")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'cp', kind = 'count', data = data, palette = 'pastel')
plt.title("Count of Cp")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'fbs', kind = 'count', data = data, palette = 'pink')
plt.title("Count of Fbs")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'restecg', kind = 'count', data = data, palette = 'pastel')
plt.title("Count of restecg")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'exang', kind = 'count', data = data, palette = 'pink')
plt.title("Count of Exang")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', data = data, palette = 'pastel')
plt.title("Count of Target")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', hue = 'cp', data = data, palette =␣
↪'pastel')
plt.title("Cp and Target")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', hue = 'thal', data = data, palette =␣
↪'pink')
plt.title("Thal and Target")
plt.show()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.catplot(x = 'target', kind = 'count', hue = 'restecg', data = data, palette␣
↪= 'pink')
plt.title("Restecg and Target")
plt.show()
```

## 3 Data Preprocessing

[ ]:
```
columns = ['age','trestbps','chol','thalach','oldpeak']
for i in columns:
    plt.figure(figsize = (10,10))
    sns.boxplot(data = data[i], palette = 'Set2')
    plt.title("Box plot of " + i)
```

### Removing Outliers

[ ]:
```
data = data[(data["trestbps"]) < 170] #age has no outliers, but trestbps has␣
 ↪outliers, but trestbps has outliers, above 170 outliers lie if we look at␣
 ↪the boxplot.
data = data[(data["chol"]) < 360]
data = data[(data["thalach"]) > 90]
data = data[(data["oldpeak"]) < 4]
```

### Creating new feature

[ ]:
```
feature = data["slope"] * data["oldpeak"] # feature generation.
feature.head()
```

[ ]:
```
data["feature"] = feature #adding feature to dataset.
data.head()
```

[ ]:
```
plt.figure(figsize = (10,10))
sns.boxplot(data["feature"], palette = 'Set2')
plt.title("Box plot of feature column")
plt.show()
```

### Removing Outliers from new feature

[ ]:
```
data = data[(data["feature"]) < 3.8]
```

[ ]:
```
column = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'feature']
for i in column:
    plt.figure(figsize = (10,10))
    sns.distplot(data[i], color = 'b')
    plt.title("Histogram of " + i)
    plt.show()
```

### Normal Distribution

[ ]:
```
data["thalach"] = QuantileTransformer().fit_transform(data["thalach"].values.
 ↪reshape(-1,1))
data["oldpeak"] = QuantileTransformer().fit_transform(data["oldpeak"].values.
 ↪reshape(-1,1))
data["feature"] = QuantileTransformer().fit_transform(data["feature"].values.
 ↪reshape(-1,1))
```

### Min-Max Scaling

```
data["age"] = StandardScaler().fit_transform(data["age"].values.reshape(-1,1))
data["trestbps"] = StandardScaler().fit_transform(data["trestbps"].values.
 ↪reshape(-1,1))
data["chol"] = StandardScaler().fit_transform(data["chol"].values.reshape(-1,1))
data["thalach"] = StandardScaler().fit_transform(data["thalach"].values.
 ↪reshape(-1,1))
data["oldpeak"] = StandardScaler().fit_transform(data["oldpeak"].values.
 ↪reshape(-1,1))
data["feature"] = StandardScaler().fit_transform(data["feature"].values.
 ↪reshape(-1,1))
```

```
data_encoded = data #making a copy of our data.
```

```
column = ["sex","cp","fbs","restecg","exang","slope","ca","thal"]
for i in column:
    dummy = pd.get_dummies(data_encoded[i])
    data_encoded = pd.concat([data_encoded,dummy], axis = 1)
```

```
data.head()
```

```
data_encoded.head()
```

```
data.describe() #pandas function to get statistics.
```

## 4 Train-Test Data

```
y = data["target"] # y is target column
```

```
data.drop(["target"], axis = 1, inplace = True)
```

```
data.head()
```

```
train_x,test_x,train_y,test_y = train_test_split(data,y, test_size = 0.3,␣
 ↪random_state = 50)
```

```
data_encoded.drop(['target','sex','cp', 'fbs', 'restecg', 'exang', 'slope',␣
 ↪'ca', 'thal'], axis = 1, inplace = True)
```

```
train_x_1,test_x_1,train_y_1,test_y_1 = train_test_split(data_encoded,y,␣
 ↪test_size = 0.3, random_state = 50)
```

## 5 Model Fitting

## 6 Logistic Regression

[ ]:
```
logistic = LogisticRegression()
logistic.fit(train_x,train_y)
logistic_y = logistic.predict(test_x)
print(accuracy_score(logistic_y,test_y))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,logistic_y)
confusion_matrix
```

[ ]:
```
logistic_1 = LogisticRegression()
logistic_1.fit(train_x_1, train_y_1)
logistic_y_1 = logistic_1.predict(test_x_1)
print(accuracy_score(logistic_y_1, test_y_1))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,logistic_y_1)
confusion_matrix
```

## 7 RandomForestClassifier

[ ]:
```
random = RandomForestClassifier()
random.fit(train_x,train_y)
random_y = random.predict(test_x)
print(accuracy_score(random_y,test_y))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,random_y)
confusion_matrix
```

[ ]:
```
random_1 = RandomForestClassifier()
random_1.fit(train_x_1,train_y_1)
random_y_1 = random_1.predict(test_x_1)
print(accuracy_score(random_y_1,test_y_1))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,random_y_1)
confusion_matrix
```

## 8 Naive Bayes

[ ]:
```
bayes = GaussianNB()
bayes.fit(train_x,train_y)
bayes_y = bayes.predict(test_x)
print(accuracy_score(bayes_y,test_y))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,bayes_y)
confusion_matrix
```

[ ]:
```
bayes_1 = GaussianNB()
bayes_1.fit(train_x_1,train_y_1)
bayes_y_1 = bayes_1.predict(test_x_1)
print(accuracy_score(bayes_y_1,test_y_1))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,bayes_y_1)
confusion_matrix
```

## 9 SVM

[ ]:
```
svm = SVC()
svm.fit(train_x,train_y)
svm_y = svm.predict(test_x)
print(accuracy_score(svm_y,test_y))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,svm_y)
confusion_matrix
```

[ ]:
```
svm_1 = SVC()
svm_1.fit(train_x_1,train_y_1)
svm_y_1 = svm_1.predict(test_x_1)
print(accuracy_score(svm_y_1,test_y_1))
```

[ ]:
```
confusion_matrix=metrics.confusion_matrix(test_y,svm_y_1)
confusion_matrix
```

**If you liked it please upvote**