Heart Disease Prediction using Exploratory Data Analysis

Determine and examine factors that play a significant role in increasing the rate of heart attacks. Also, use the findings to create and predict a model .

Domain: Healthcare

Prediction: Factors playing significant role in Heart Attacks

With the provided data alongwith the below variables in the data, the factors that play a significant role in increasing the rate of heart attacks will be explored using Python & different models will be trained & tested to find best model, as well data visualisation will be done on Tableau.

variable	description
age	age in years {#}
sex	(1 = male; 0 = female) {Binary}
cp	chest pain type {Ordinal – 4 Values}
trestbps	resting blood pressure (in mm Hg on admission to the hospital) {#}
chol	serum cholestoral in mg/dl {#}
fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false) {Binary}
restecg	resting electrocardiographic results {0;1;2}
thalach	maximum heart rate achieved {#}
exang	exercise induced angina (1 = yes; 0 = no) {Binary}
oldpeak	ST depression induced by exercise relative to rest {#}
slope	the slope of the peak exercise ST segment {1;2;3 Ordinal}
ca	number of major vessels (0-3) colored by flourosopy {Ordinal}
thal	3 = normal; 6 = fixed defect; 7 = reversable defect {Ordinal}
target	1 or 0

Predictor 'Y' (Whether Positive or Negative for Heart Attacks) has 13 Features 'X'

3 Types of Data

I. **Continous {#} Quantitative that can be measured**
II. **Binary – Two possible outcomes**
III. **Ordinal – Categorical data with order 0,1,2…**

Importing, Understanding, and Inspecting Data:

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

Converted the .xlsx file to .csv

Loading file name 'Heart.csv' to Python

```
[35]: pd.read_csv('Heart.csv')
```

[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

```
[38]: data.head(5)
```

[38]:

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

No. of rows & columns alongwith the column names:

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

```
(Rows, columns): (303, 14)
```

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

No. of unique values for each variable

```
[40]: data.nunique(axis=0)
```

```
[40]: 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
```

Summary of Count, Min & Max, Mean, Standard Deviation

```
[42]: data.describe()
```

[42]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.3
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.6
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.0
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.0
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.0
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.0

Checking the Missing Values in each Column

As can be seen below, there are NO MISSING VALUES in each column

```
[44]: # 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
```

Checking if there's a good proportion of positive & negative Binary predictor

```
[45]: # Checking whether there's a good proportion of positive & negative binary predictor
      data['target'].value_counts()
```

```
[45]: 1    165
      0    138
      Name: target, dtype: int64
```

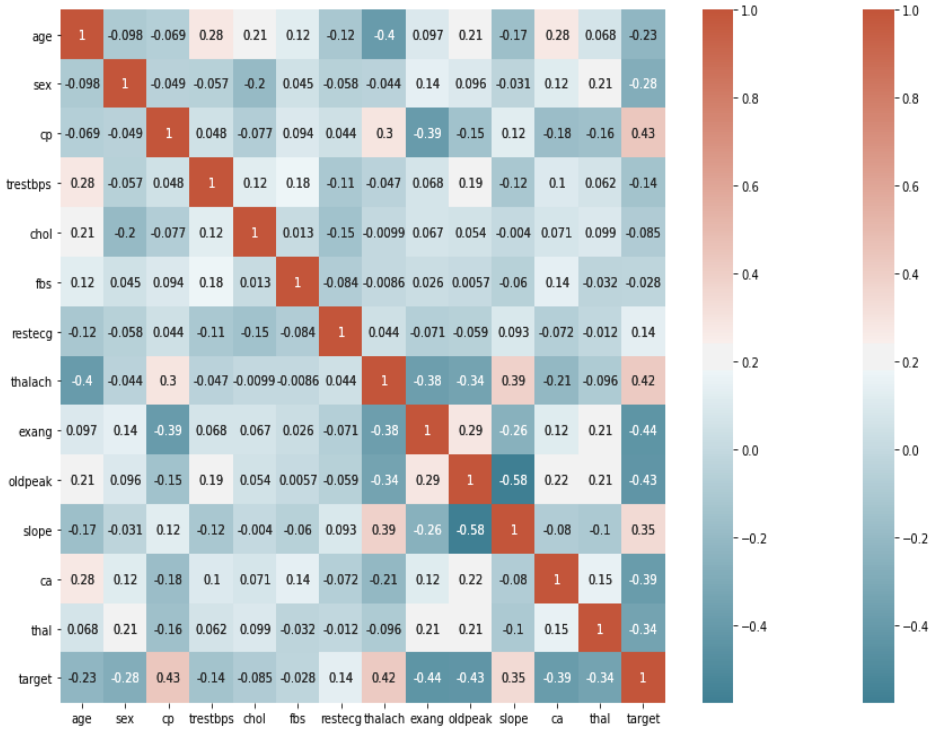
As can be seen from above, there is a good balance between the positive & negative binary predictor.

Correlation Matrix: -

We need to understand whether each of our variables are correlated, therefore Correlation Matrix to check whether a variable is Positively or Negatively Correlated with our Target

```
[46]: # Calculating 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))
```

```
[46]: <AxesSubplot:>
```



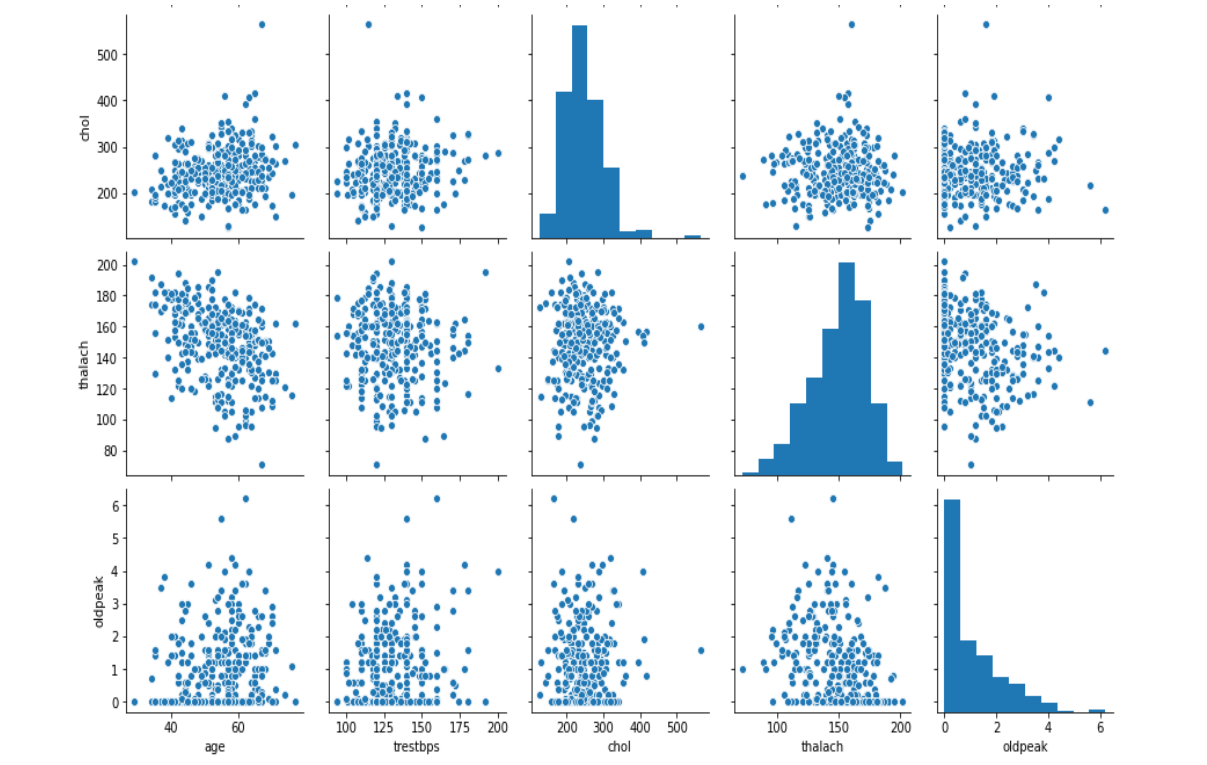
As seen from above Heatmap, a Positive Correlation is seen between Chest Pain (cp) and target {+0.43}. This shows that higher the amount of chest pain, results in higher the chances of heart diseases.

Also, there is Negative Correlation between exercise induced angina (exang) & target {-0.44}. When a person exercise, more blood is required by the heart, but due to narrowed arteries blood flow slows down.

Pairplots with continuous features to check the correlation between each variable

```
[47]: #Pairplots with continuous features to check the correlation between each variable
      subData = data[['age','trestbps','chol','thalach','oldpeak']]
      sns.pairplot(subData)
```

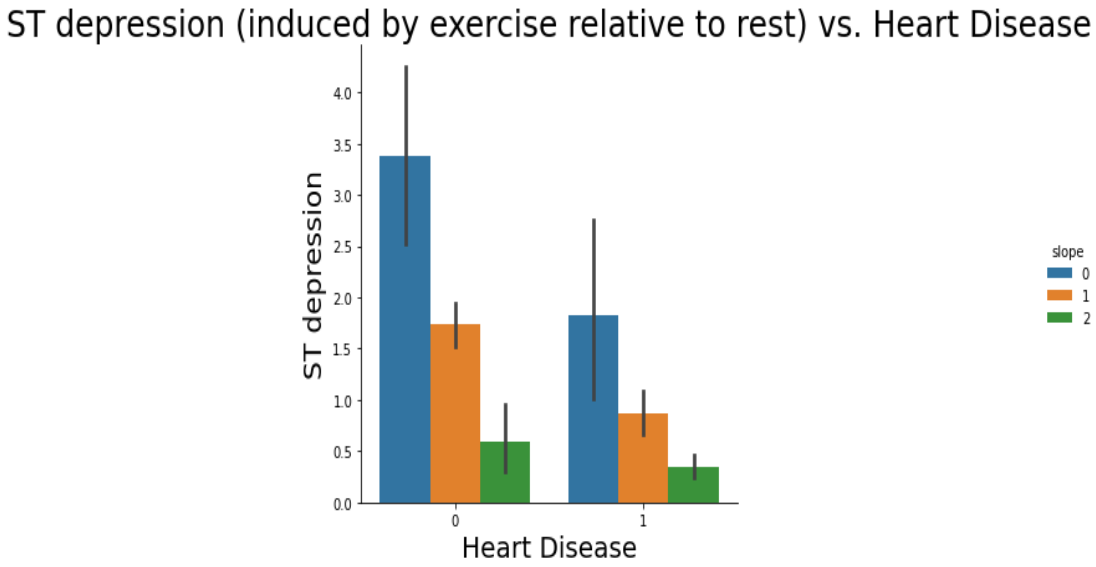
```
[47]: <seaborn.axisgrid.PairGrid at 0x7fde509a9590>
```



```
[48]: 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)
```

```
[48]: Text(26.426458333333343, 0.5, 'ST depression')
```



"In a cardiac stress test, an ST depression of at least 1 mm after adenosine administration indicates a reversible ischaemia, while an exercise stress test requires an ST depression of at least 2 mm to ***significantly indicate reversible ischaemia****."*

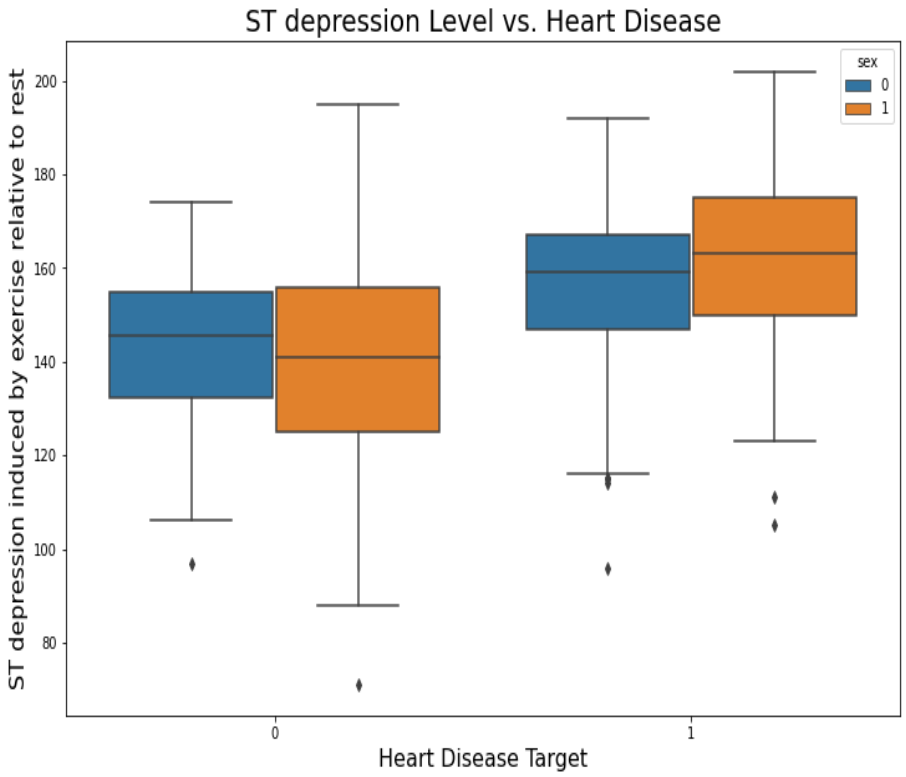
ST segment depression occurs when the ventricle is at rest. A high ST depression is considered normal & healthy. If the ST segment is abnormally low, this could lead to Heart disease. The plot above shows low ST depression puts people at higher risk of Heart disease. The positive & negative heart patients exhibit equal distribution of the 3 slope.

Box Plots :-

To understand the distribution of data, Median, Min, Max, Inter Quartile Range, as well as the Outliers in the Data

```
[49]: 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)
```

```
[49]: Text(0, 0.5, 'ST depression induced by exercise relative to rest')
```



- **Positive patients shows high median for ST depression level**
- **Negative patients shows lower median for ST depression level**
- **Not much difference in the Males & Females target**
- **Males have larger distribution of ST Depression**

Filtering data by POSITIVE Heart Disease patient

[50]:
```
# Filtering data by POSITIVE Heart Disease patient
pos_data = data[data['target']==1]
pos_data.describe()
```

[50]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.0
mean	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212	1.0
std	9.550651	0.497444	0.952222	16.169613	53.552872	0.347412	0.504818	19.174276	0.347412	0.780683	0.593635	0.848894	0.465752	0.0
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	96.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.0
25%	44.000000	0.000000	1.000000	120.000000	208.000000	0.000000	0.000000	149.000000	0.000000	0.000000	1.000000	0.000000	2.000000	1.0
50%	52.000000	1.000000	2.000000	130.000000	234.000000	0.000000	1.000000	161.000000	0.000000	0.200000	2.000000	0.000000	2.000000	1.0
75%	59.000000	1.000000	2.000000	140.000000	267.000000	0.000000	1.000000	172.000000	0.000000	1.000000	2.000000	0.000000	2.000000	1.0
max	76.000000	1.000000	3.000000	180.000000	564.000000	1.000000	2.000000	202.000000	1.000000	4.200000	2.000000	4.000000	3.000000	1.0

Filtering data by NEGATIVE Heart Disease patient

[56]:
```
# Filtering data by NEGATIVE Heart Disease patient
neg_data = data[data['target']==0]
neg_data.describe()
```

[56]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.0
mean	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478	0.0
std	7.962082	0.380416	0.905920	18.729944	49.454614	0.367401	0.541321	22.598782	0.499232	1.300340	0.561324	1.043460	0.684762	0.0
min	35.000000	0.000000	0.000000	100.000000	131.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	52.000000	1.000000	0.000000	120.000000	217.250000	0.000000	0.000000	125.000000	0.000000	0.600000	1.000000	0.000000	2.000000	0.0
50%	58.000000	1.000000	0.000000	130.000000	249.000000	0.000000	0.000000	142.000000	1.000000	1.400000	1.000000	1.000000	3.000000	0.0
75%	62.000000	1.000000	0.000000	144.750000	283.000000	0.000000	1.000000	156.000000	1.000000	2.500000	1.750000	2.000000	3.000000	0.0
max	77.000000	1.000000	3.000000	200.000000	409.000000	1.000000	2.000000	195.000000	1.000000	6.200000	2.000000	4.000000	3.000000	0.0

---

[52]:
```
print("(Positive Patients ST depression): " + str(pos_data['oldpeak'].mean()))
```

```
(Positive Patients ST depression): 1.5855072463768118
```

---

[58]:
```
print("(Negative Patients ST depression): " + str(neg_data['oldpeak'].mean()))
```

```
(Negative Patients ST depression): 1.5855072463768118
```

---

[59]:
```
print("(Positive Patients thalach): " + str(pos_data['thalach'].mean()))
```

```
(Positive Patients thalach): 139.1014492753623
```

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

Negative Patients thalach) : 139.1014492753623
```

Inferences :-

- **There is a vast difference in the means of the 13 variables of positive & negative patients**
- **Positive Patients have heightened maximum heart rate average (thalach)**
- **Also, Positive patients have 1/3rd the amount of ST depression induced by exercise relative to rest (oldpeak)**

Model Building

Logistic regression, predicting the outcome for test data, and validating the results by using the confusion matrix.

Assigning the 13 features to X and last column to classification Predictor Y

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

Spliting the Data into Training & Test

```
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)
```

Normalizing the data will transform the data so its distribution will have mean of 0 & standard dev of 1

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

Training

Training various classification models on Training set and check which yields the highest accuracy

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
```

[19]:
```
### Logistic Regression
```

[20]:
```
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
```

```
# Logistic Regression = Accuracy 74%
```

K-NN (K-Nearest Neighbors)

```
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier

model2 = KNeighborsClassifier() # get instance of model
model2.fit(x_train, y_train) # Train/Fit model

y_pred2 = model2.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred2)) # output accuracy
```

[22]:
```
### K-NN (K-Nearest Neighbors)
```

[23]:
```
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier

model2 = KNeighborsClassifier() # get instance of model
model2.fit(x_train, y_train) # Train/Fit model

y_pred2 = model2.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred2)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.78      0.70      0.74        30
           1       0.74      0.81      0.77        31

    accuracy                           0.75        61
   macro avg       0.76      0.75      0.75        61
weighted avg       0.76      0.75      0.75        61
```

```
# K-NN (K-Nearest Neighbors) = Accuracy 75%
```

Support Vector Machine (SVM)

```
from sklearn.metrics import classification_report
from sklearn.svm import SVC

model3 = SVC(random_state=1) # get instance of model
model3.fit(x_train, y_train) # Train/Fit model

y_pred3 = model3.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred3)) # output accuracy
```

[24]:
```
## Support Vector Machine (SVM)
```

[25]:
```
from sklearn.metrics import classification_report
from sklearn.svm import SVC

model3 = SVC(random_state=1) # get instance of model
model3.fit(x_train, y_train) # Train/Fit model

y_pred3 = model3.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred3)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.80      0.67      0.73        30
           1       0.72      0.84      0.78        31

    accuracy                           0.75        61
   macro avg       0.76      0.75      0.75        61
weighted avg       0.76      0.75      0.75        61
```

Support Vector Machine = Accuracy 75%

Naives Bayes Classifier

```
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB

model4 = GaussianNB() # get instance of model
model4.fit(x_train, y_train) # Train/Fit model

y_pred4 = model4.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred4)) # output accuracy
```

[26]:
```
## Naives Bayes Classifier
```

[27]:
```
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB

model4 = GaussianNB() # get instance of model
model4.fit(x_train, y_train) # Train/Fit model

y_pred4 = model4.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred4)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.79      0.73      0.76        30
           1       0.76      0.81      0.78        31

    accuracy                           0.77        61
   macro avg       0.77      0.77      0.77        61
weighted avg       0.77      0.77      0.77        61
```

Naives Bayes Classifier = Accuracy 77%

Decision Trees

```
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

model5 = DecisionTreeClassifier(random_state=1) # get instance of model
model5.fit(x_train, y_train) # Train/Fit model

y_pred5 = model5.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred5)) # output accuracy
```

[28]:
```
## Decision Trees
```

[29]:
```
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

model5 = DecisionTreeClassifier(random_state=1) # get instance of model
model5.fit(x_train, y_train) # Train/Fit model

y_pred5 = model5.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred5)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.68      0.70      0.69        30
           1       0.70      0.68      0.69        31

    accuracy                           0.69        61
   macro avg       0.69      0.69      0.69        61
weighted avg       0.69      0.69      0.69        61
```

Decision Trees = Accuracy 69%

Random Forest

```
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
```

[30]:
```
## Random Forest
```

[31]:
```
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
```

Random Forest = Accuracy 80%

Random Forest yields the Highest Accuracy of 80 %

Random Forest out of all the Models yields the highest accuracy

Confusion Matrix

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

Confusion Matrix

[32]:
```
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]]
```

[32]:
```
0.8032786885245902
```

[33]:
```
## Any accuracy above 70% is good. Therefore, the 80%  is the ideal accuracy
```

[34]:
```
# Inference of Confusion Matrix
```

True Positives in our data is 21

True Negatives in our data is 28

No. of Errors is 9

There are 9 Type 1 errors (False Positives)- You predicted positive and it's false.

There are 3 Type 2 errors (False Negatives)- You predicted negative and it's false.

Hence, if we calculate the accuracy its # Correct Predicted/ # Total.

In other words, where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives.

Accuracy = (TP + TN)/(TP + TN + FP + FN).

Accuracy =(21+28)/(21+28+9+3) = 0.80 = 80% accuracy

Feature Importance

```
# get importance
importance = model6.feature_importances_

# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))

index= data.columns[:-1]
importance = pd.Series(model6.feature_importances_, index=index)
importance.nlargest(13).plot(kind='barh', colormap='winter')
```

Feature Importance

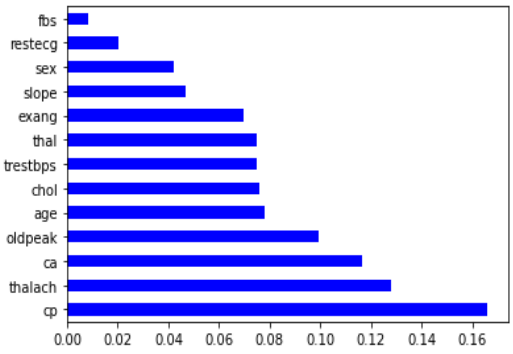
[35]:
```
# get importance
importance = model6.feature_importances_

# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.07814
Feature: 1, Score: 0.04206
Feature: 2, Score: 0.16580
Feature: 3, Score: 0.07477
Feature: 4, Score: 0.07587
Feature: 5, Score: 0.00828
Feature: 6, Score: 0.02014
Feature: 7, Score: 0.12772
Feature: 8, Score: 0.06950
Feature: 9, Score: 0.09957
Feature: 10, Score: 0.04677
Feature: 11, Score: 0.11667
Feature: 12, Score: 0.07473
```

[36]:
```
index= data.columns[:-1]
importance = pd.Series(model6.feature_importances_, index=index)
importance.nlargest(13).plot(kind='barh', colormap='winter')
```

[36]: <AxesSubplot:>



Conclusion from Feature Importance graph

Top 4 significant features were chest pain type (cp), maximum heart rate acheived (thalach), number of major vessels (ca), ST depression induced by exercise relative to rest (oldpeak)

#Predictions

S**cenario**: A patient develops cardiac symptoms & you input his vitals into the Machine Learning Algorithm.

A **20** year old male, with a **chest pain value of 2** (atypical angina), with **resting blood pressure of 110.**

Has a **serum cholestoral of 230 mg/dl.**

His **fasting blood sugar > 120 mg/dl.**

His **resting electrocardiographic result** of **1.**

The **patients maximum heart rate achieved** is **140.**

He was **exercise induced angina.**

His **ST depression** induced by exercise relative to rest value was **2.2.**

The **slope** of the peak exercise ST segment is **flat**.

Has **no major vessels** colored by fluoroscopy, and in addition his **maximum heart rate achieved** is a reversible defect.

Based on this information, **classify** this patient with Heart Disease?

```
print(model6.predict(sc.transform([[20,1,2,110,230,1,1,140,1,2.2,2,0,2]])))
```

[37]:
```
#Predictions
## Scenario: A patient develops cardiac symptoms & you input his vitals into the Machine Learning Algorithm.
## A 20 year old male, with a chest pain value of 2 (atypical angina), with resting blood pressure of 110.
## Has a serum cholestoral of 230 mg/dl.
## His fasting blood sugar > 120 mg/dl.
## His resting electrocardiographic result of 1.
## The patients maximum heart rate achieved is 140.
## He was exercise induced angina.
## His ST depression induced by exercise relative to rest value was 2.2.
## The slope of the peak exercise ST segment is flat.
##  Has no major vessels colored by fluoroscopy, and in addition his maximum heart rate achieved is a reversible defect.
## Based on this information, classify this patient with Heart Disease?
```

[38]:
```
print(model6.predict(sc.transform([[20,1,2,110,230,1,1,140,1,2.2,2,0,2]])))
```

```
[1]
```

[39]:
```
## Outputs Binary 1 meaning Positive Diagnosis of Heart Disease
```

[40]:
```
### Predicting the Test set results:
### First value represents our predicted value, Second value represents our actual value.
### If the values match, then we predicted correctly.
```

[41]:
```
y_pred = model6.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
## Results are accurate at 80%
```

Conclusions

Out of the 13 features examined, the top 4 significant features that helped us classify between a positive & negative Diagnosis were chest pain type (cp), maximum heart rate achieved (thalach), number of major vessels (ca), and ST depression induced by exercise relative to rest (oldpeak).

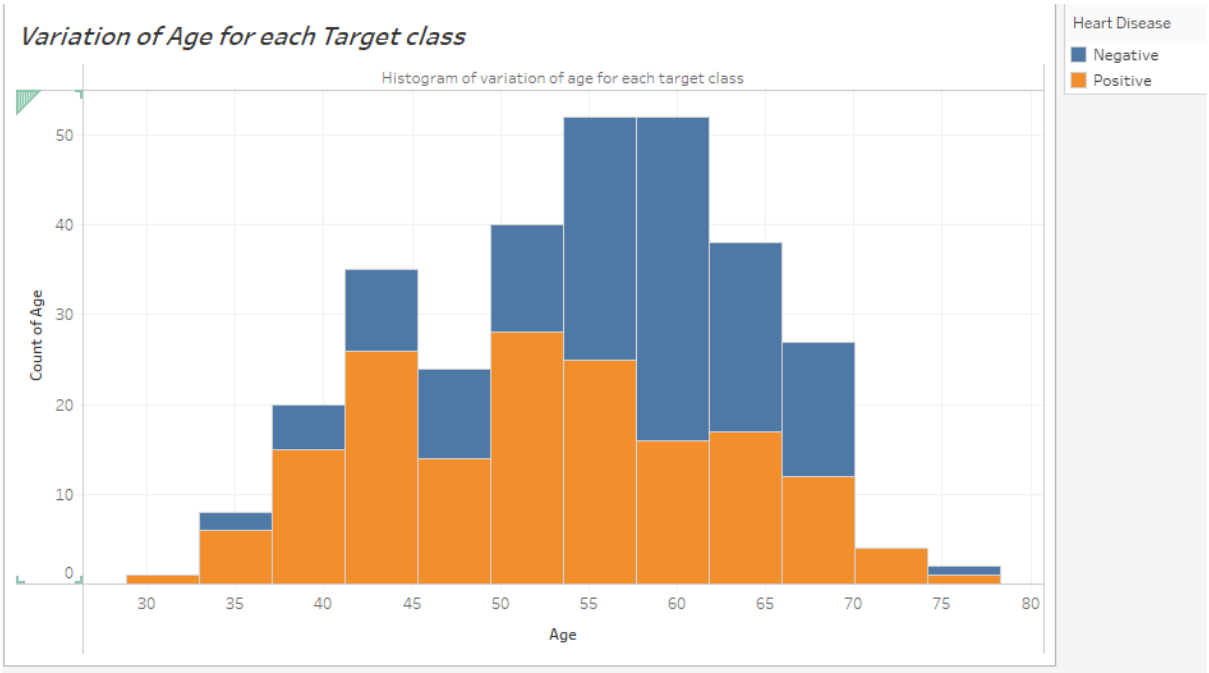
Our machine learning algorithm can now classify patients with Heart Disease. Now we can properly diagnose patients, & get them the help they need to recover. By diagnosing detecting these features early, we may prevent worse symptoms from arising later.

Our Random Forest algorithm yields the highest accuracy, 80%. Any accuracy above 70% is considered good, but be careful because if your accuracy is extremely **high, it may be too good to be true (an example of Over fitting). Thus, 80% is the ideal accuracy!**

TABLEAU:

Tableau is one of the business intelligence software used to analyse data and visualize the insights in the form of graph and charts. It provides visual appealing clusters in order to predict the occurrence of heart disease from the given dataset.

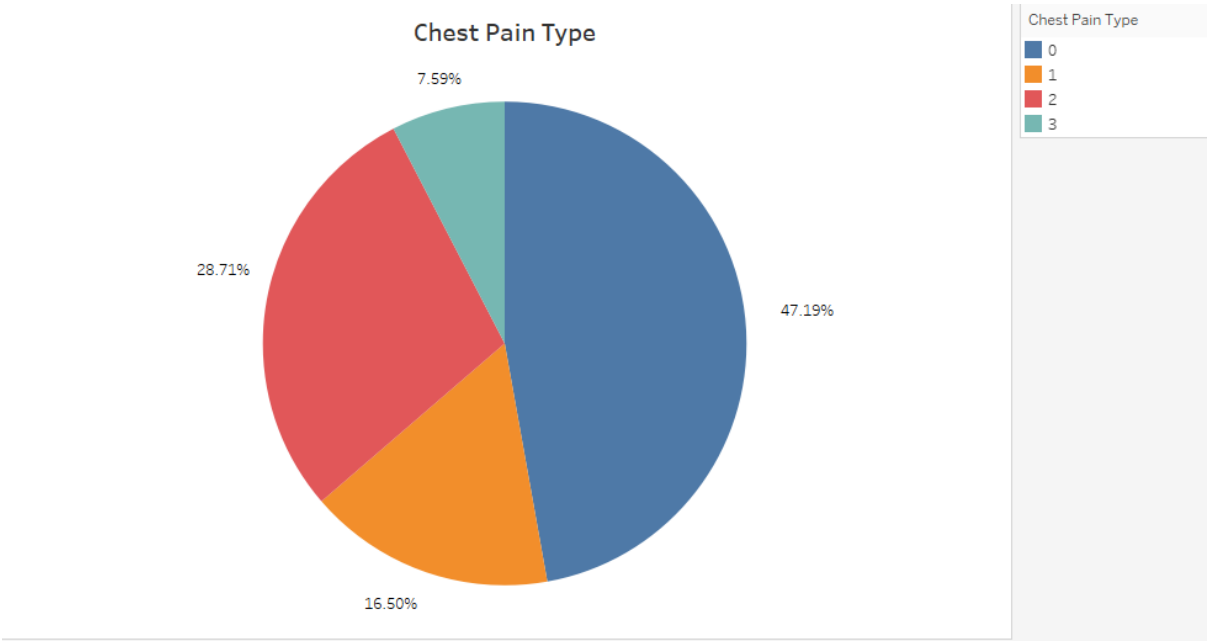
The result of the data analysis to identify the necessary hidden patterns for predicting heart diseases are presented in this section. Here the variables considered to predict the heart disease are age, chest pain type, blood pressure, blood glucose level, ECG in rest, heart rate and four types of chest pain and exercise angina.



Inference:-

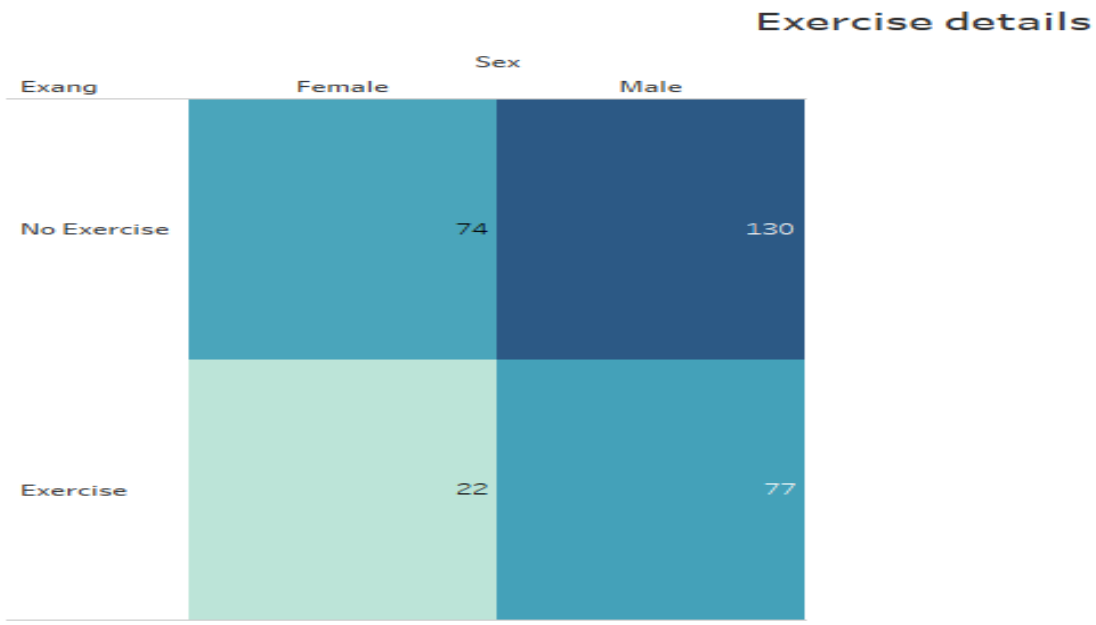
Histogram depicts the distribution of ages and the risk of heart disease for the targeted class. It is observed that target class with the age ranging from 50 to 55 is having high risk of heart disease as the development of coronary fatty streaks starts in this age range.

Chest Pain Type:-



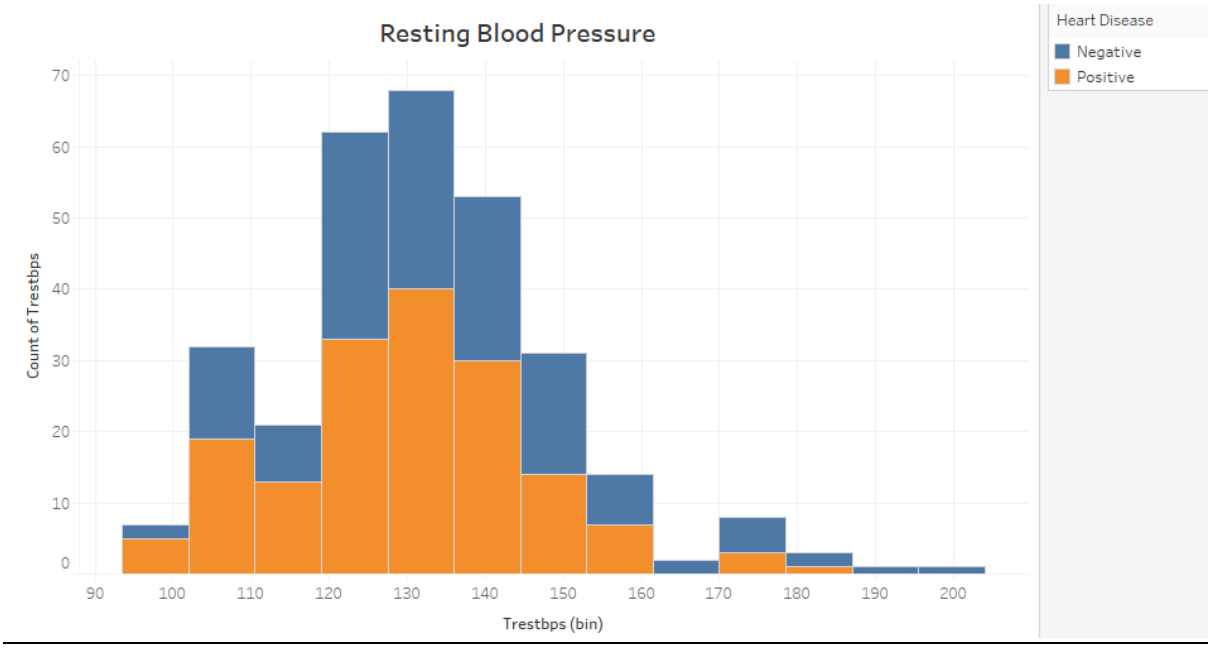
As was depicted earlier by the Heatmap, there was a Positive correlation between the variable chest pain & target variable. Which showed higher the amount of chest pain, higher were the chances of heart disease. Going forward, the above chart denotes the different chest pain types. Chest pain type '0' has the highest percentage.

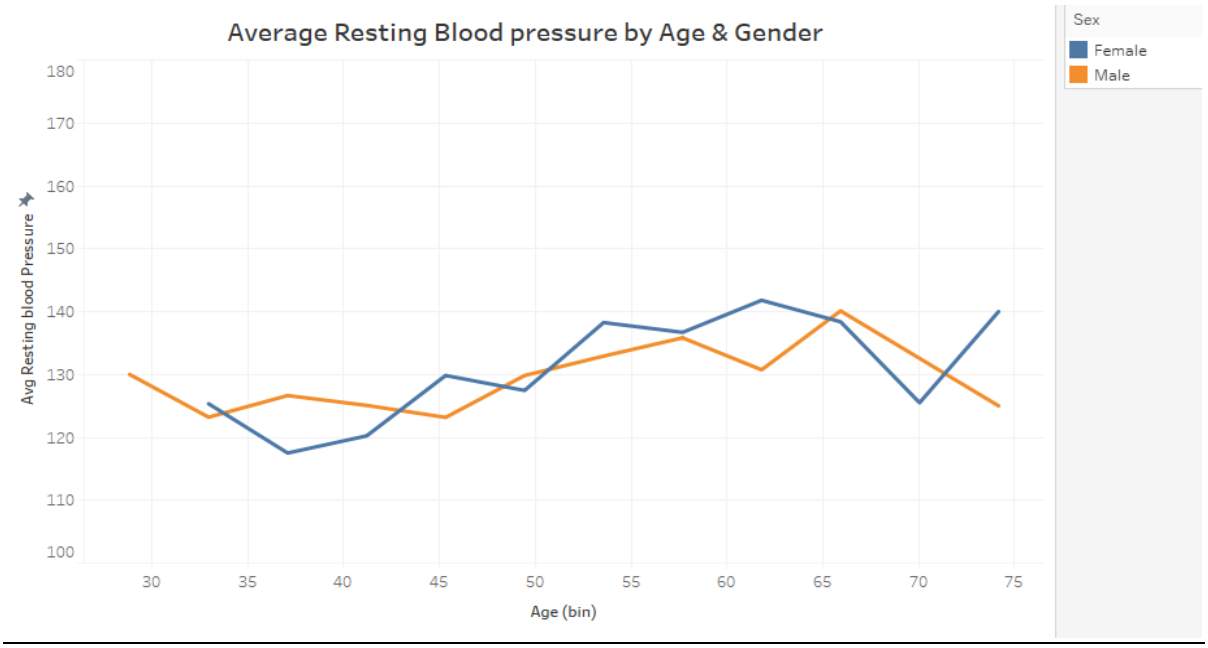
Exercise details:-



As can be seen from above figure the difference between gender and the whether or not exercise plays a role in the disease. Both Male & Female who do not exercise has higher the chances of heart disease, as compared to the one's who do exercise.

Resting Blood Pressure:-

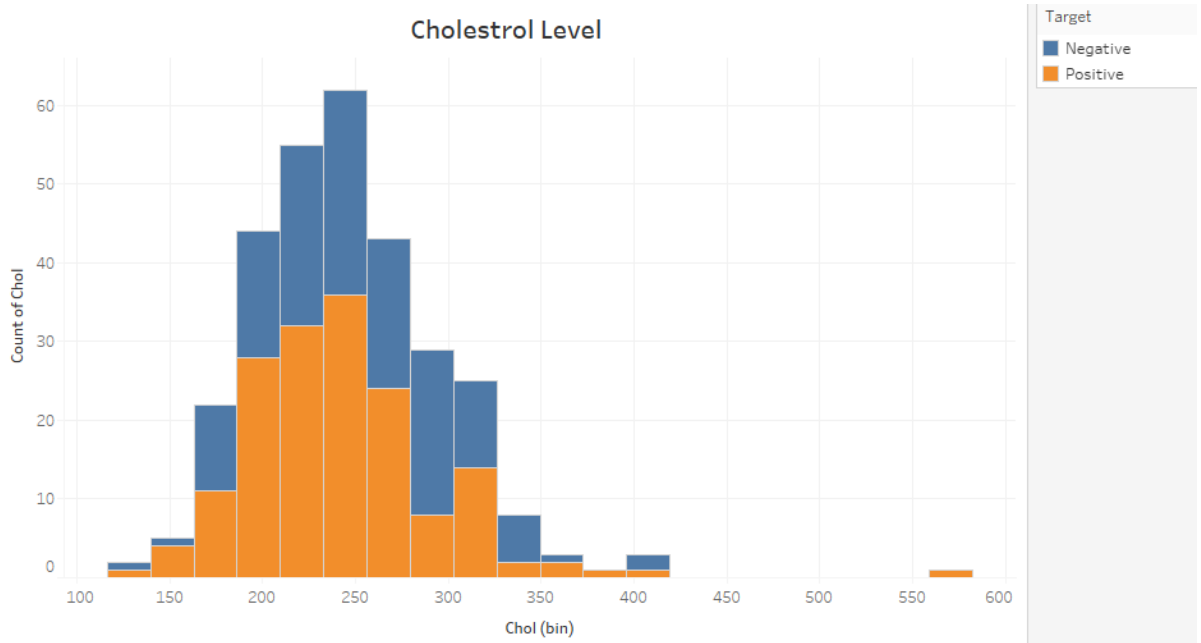




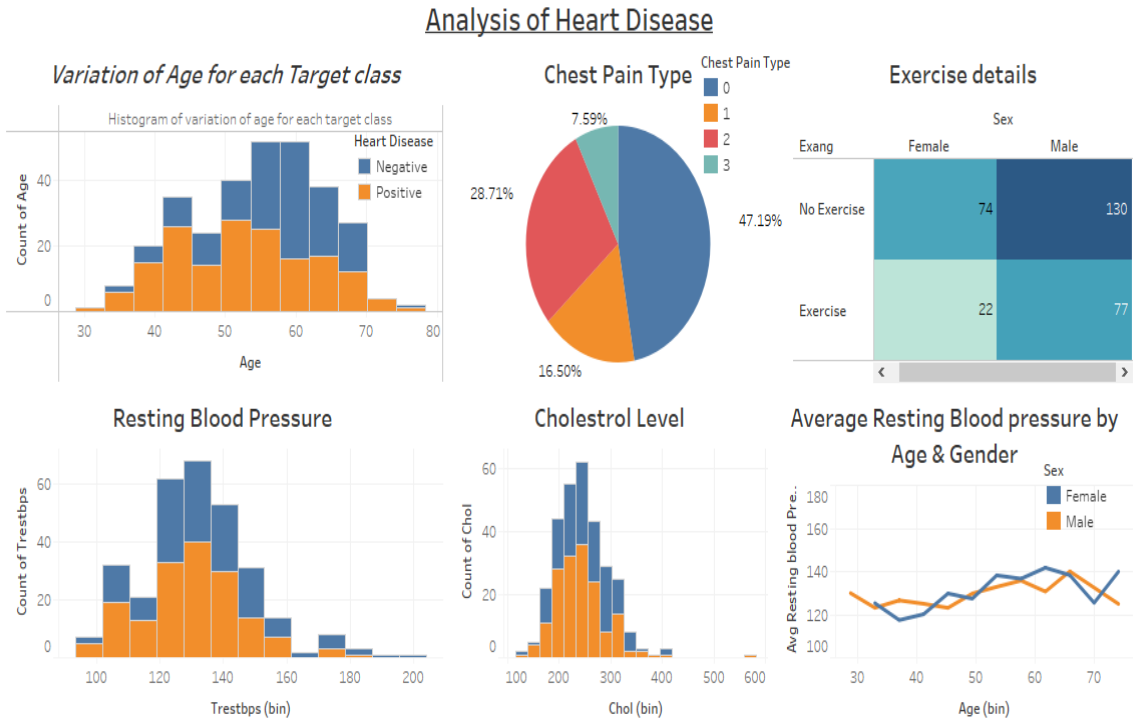
Inference:-

The above chart shows the resting blood pressure of the patient admitted to hospital who whether had a heart disease or not. Patient with Trestbps of 130 to 140 had higher chances of heart disease. Also, the average resting blood pressure by Age & Gender is denoted.

Cholestrol Level:-



In the above graph, the Cholestrol level are measured, wherein it shows the patient having cholesterol level from 200 to 250 are having higher cases of heart diseases.



Heart stroke and vascular disease are the major cause of disability and premature death. Chest pain is the key to recognize the heart disease. The role of exploratory data using tableau provided a visual experience.