SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be **University** *u/s 3 of UGC Act, 1956)*

DEPARTMENT : DSBS

Name : Elina Singh, Anu Srishti Bara

Registration No : RA2011042010041, RA2011042010005

Class : CSBS X2

Subject code : 18CSE363J

Title : Machine Learning

Semester : V

Academic year: 2022-2023

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT - DSBS

Register No : RA2011042010041, RA2011042010005

BONAFIDE CERTIFICATE

This is to certify that the Lab record (Machine Learning) of the work done by ______________________________of fifth Semester, Fourth Year, B.Tech Degree course in Computer Science & Engineering Department of SRM University, is a record of a candidate's own work carried out by him/her under my supervision during the academic year 2022-2023.

EXTERNAL EXAMINER   INTERNAL EXAMINER

INDEX

ABSTRACT

Machine Learning is used across many ranges around the world. The healthcare industry is no exception. Machine Learning can play an essential role in predicting presence/absence of locomotors disorders, Heart diseases and more.Such information, if predicted well in advance, can provide important intuitions to doctors who can then adapt their diagnosis and dealing per patient basis. We work on predicting possible Heart Diseases in people using Machine Learning algorithms. In this paper we carried out research on heart disease. Prediction of heart disease is a very recent field as the data is becoming available. Other researchers have approached it with different techniques and methods. We used machine learning to detect and predict disease's patients. Starting with a preprocessing phase, where we selected the most relevant features by the correlation matrix, then we applied logistic regression and random forest classifier on data sets of different sizes, in order to study the accuracy and stability of each of them.

INTRODUCTION

According to the World Health Organization, every year 12 million deaths occur worldwide due to Heart Disease. Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of data analysis. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in an attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduces the complications.

Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry. This project aims to predict future Heart Disease by analyzing data of patients which classifies whether they have heart disease or not using a machine-learning algorithm. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can say that this technique can be very well adapted to do the prediction of heart disease.

INSPIRATION

The main motivation of doing this research is to present a heart disease prediction model for the prediction of occurrence of heart disease. Further, this research work is aimed towards identifying the best classification algorithm for identifying the possibility of heart disease in a patient. This work is justified by performing logistic regression and random forest classifier. Although these are commonly used machine learning algorithms, the heart disease prediction is a vital task involving highest possible accuracy. Hence, the three algorithms are evaluated at numerous levels and types of evaluation strategies. This will provide researchers and medical practitioners to establish a better.

WHAT IT DOES

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either they are expensive or are not efficient to calculate the chance of heart disease in humans. Early detection of cardiac diseases can decrease the mortality rate and overall complications. However, it is not possible to monitor patients everyday in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more patience, time and expertise. Since we have a good amount of data in today's world, we can use various machine learning algorithms to analyze the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

LIMITATIONS

Medical diagnosis is considered as a significant yet intricate task that needs to be carried out precisely and efficiently. The automation of the same would be highly beneficial. Clinical decisions are often made based on the doctor's intuition and experience rather than on the knowledge rich data hidden in the database. This practice leads to unwanted biases, errors and excessive medical costs which affects the quality of service provided to patients. Data mining has the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions.

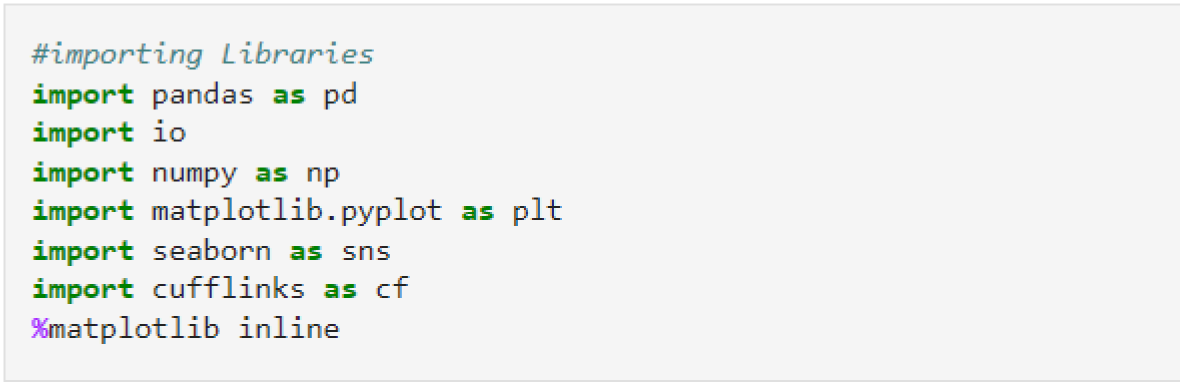
BENEFITS

The project is that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome. This suggestion is promising as data modeling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions

IMPLEMENTATION DETAILS

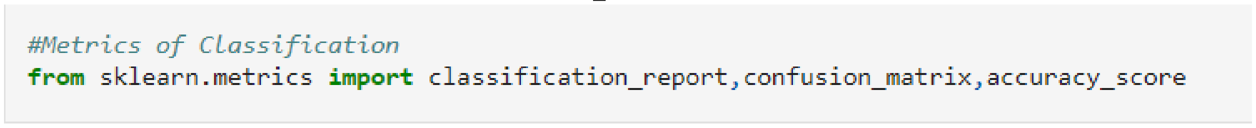
Importing Necessary Libraries

Plotting Libraries



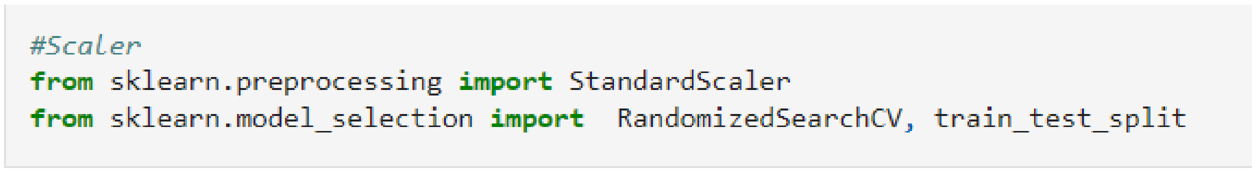
```
#importing Libraries
import pandas as pd
import io
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cufflinks as cf
%matplotlib inline
```

Metrics for Classification technique



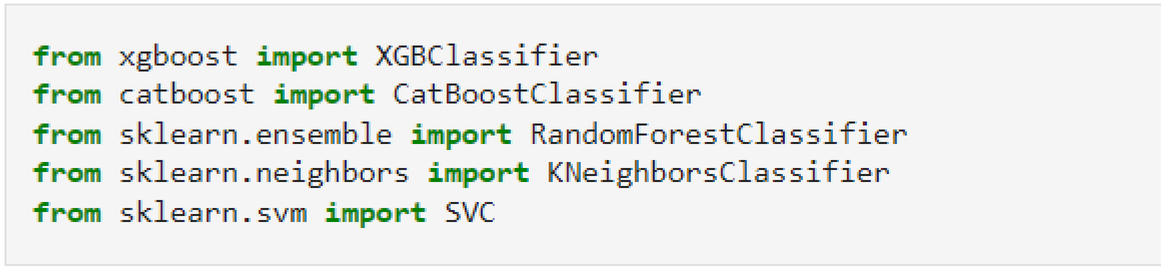
```
#Metrics of Classification
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Scaler



```
#Scaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import  RandomizedSearchCV, train_test_split
```

Model building



```
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

Data Loading

Here we used the pandas read_csv function to read the dataset.

Importing Data

```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Heart Prediction/heart.csv')
data.head(6)
```

Output:

	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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1

Exploratory Data Analysis

Size of the dataset:

```
#shape of data
data.shape
```

```
(303, 14)
```

We have 303 rows with 14 columns

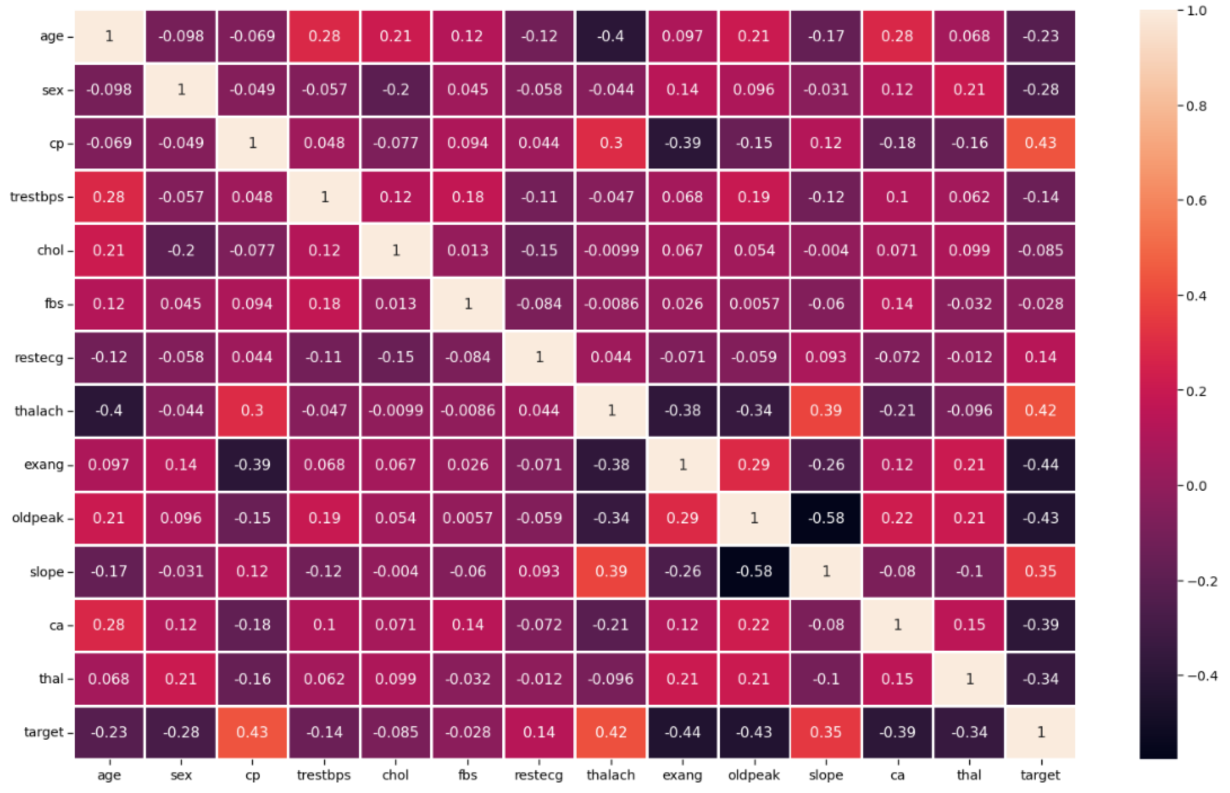
Inference: We have a dataset with 303 rows which indicates a smaller set of data.

```python
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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	303.000000	303.000000
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	2.313531	0.544554
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	0.612277	0.498835
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	0.000000	0.000000
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	2.000000	0.000000
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	2.000000	1.000000
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	3.000000	1.000000
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	3.000000	1.000000

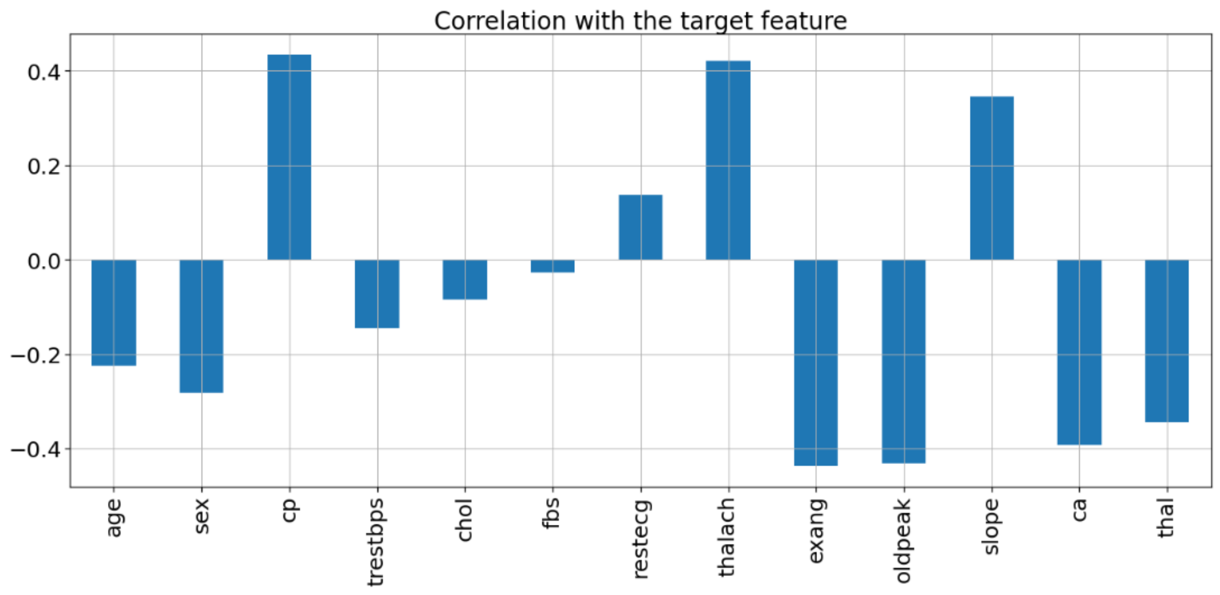
Checking the correlation between various features:

```python
#correlation between various features
plt.figure(figsize=(20,12))
sns.set_context('notebook',font_scale = 1.3)
sns.heatmap(data.corr(),annot=True,linewidth =2)
plt.tight_layout()
```



Checking the correlation of the target variable:

```
sns.set_context('notebook',font_scale = 2.3)
data.drop('target', axis=1).corrwith(data.target).plot(kind='bar', grid=True, figsize=(20, 10),
                                                        title="Correlation with the target feature")
plt.tight_layout()
```



Inference: Insights from the above graph are:

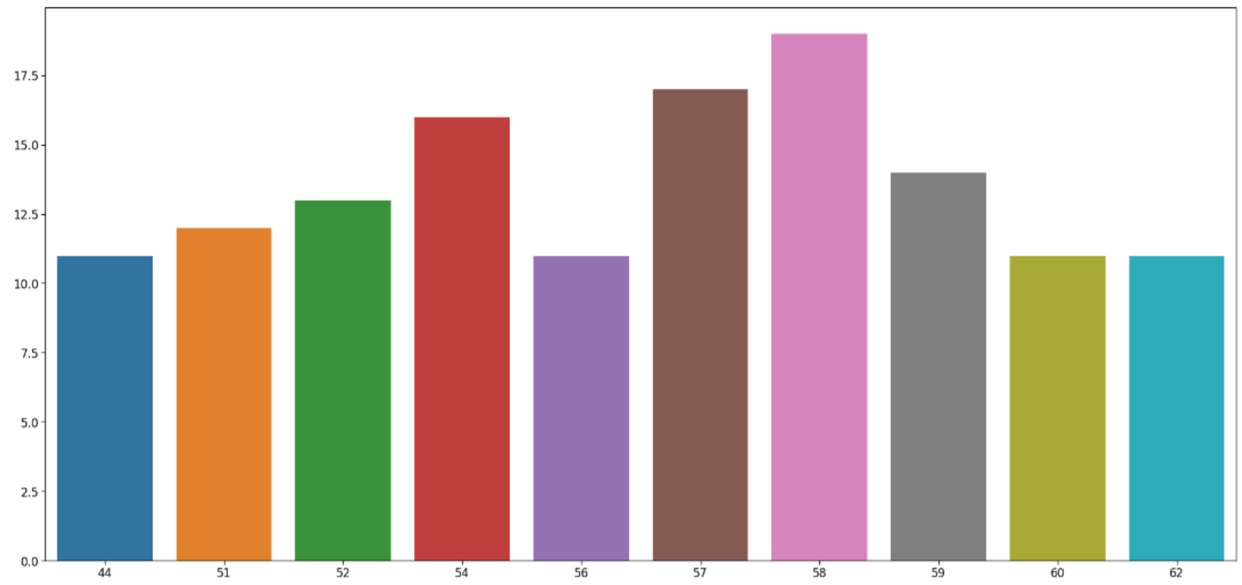
- Four features(“cp”, “restecg”, “thalach”, “slope”) are positively correlated with the target feature.
- Other features are negatively correlated with the target feature.

So, we have done enough collective analysis.

Analysis of the individual features which comprises both univariate and bivariate analysis:

Here we will be checking the 10 ages and their counts.

```
#Age analysis
plt.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=data.age.value_counts()[:10].index,y=data.age.value_counts()[:10].values)
plt.tight_layout()
```



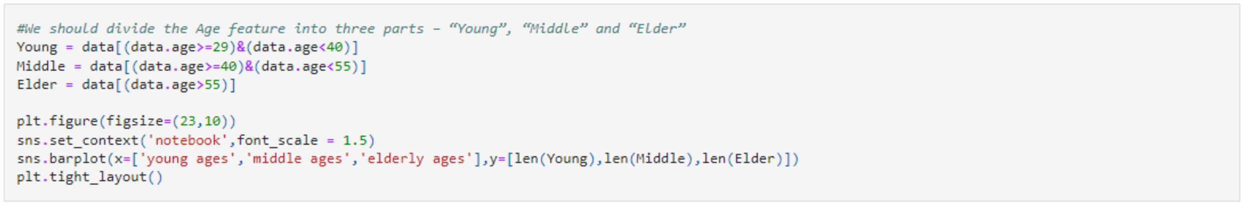
Inference: 58 age column has the highest frequency.

Checking the range of age in the dataset:

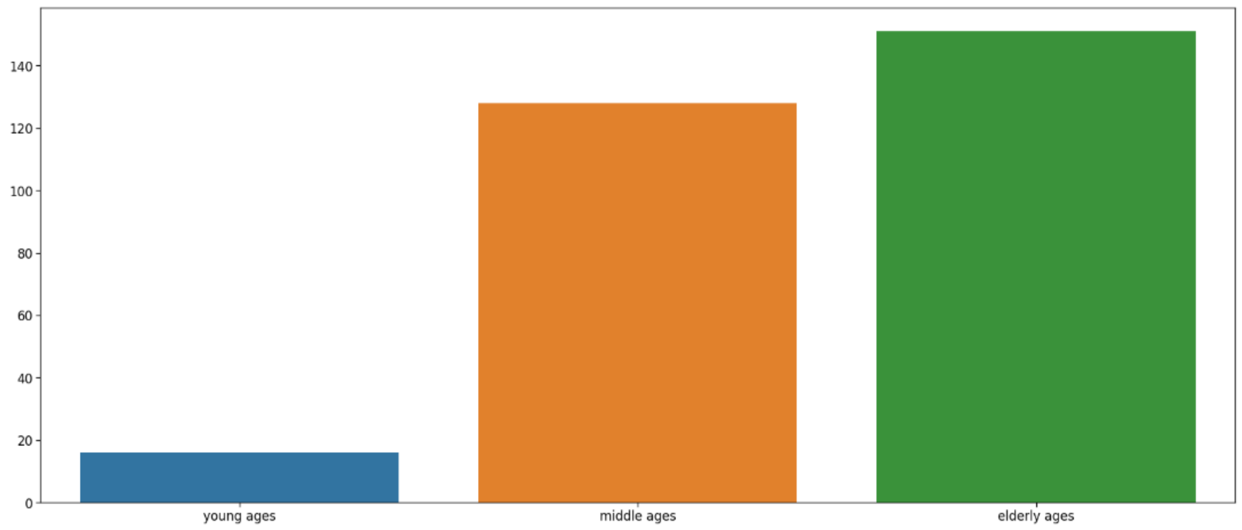
```
minAge=min(data.age)
maxAge=max(data.age)
meanAge=data.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
```

```
Min Age : 29
Max Age : 77
Mean Age : 54.366336633663366
```

We divided the Age feature into three parts – “Young”, “Middle” and “Elder”



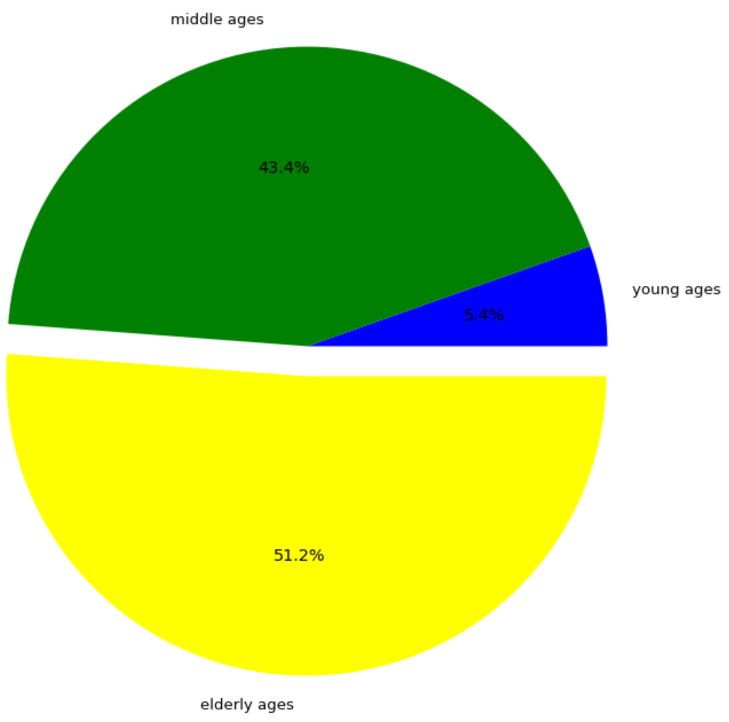
```
#We should divide the Age feature into three parts - "Young", "Middle" and "Elder"
Young = data[(data.age>=29)&(data.age<40)]
Middle = data[(data.age>=40)&(data.age<55)]
Elder = data[(data.age>55)]

plt.figure(figsize=(23,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder)])
plt.tight_layout()
```



Inference: Here we can see that older people are the most affected by heart disease and young ones are the least affected.

To prove the above inference we plot the pie chart:

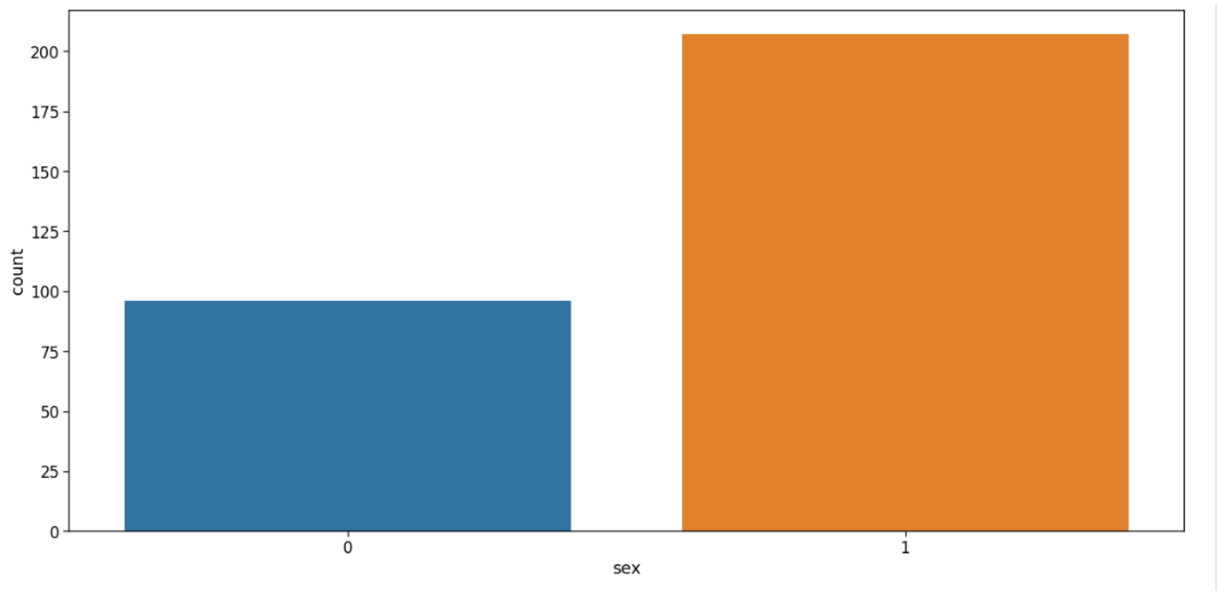
```
colors = ['blue','green','yellow']
explode = [0,0,0.1]
plt.figure(figsize=(10,10))
sns.set_context('notebook',font_scale = 1.2)
plt.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle ages','elderly ages'],explode=explode,colors=colors, autopct='%1.1f%%')
plt.tight_layout()
```



Sex("sex") Feature Analysis

```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['sex'])
plt.tight_layout()
```

Output:

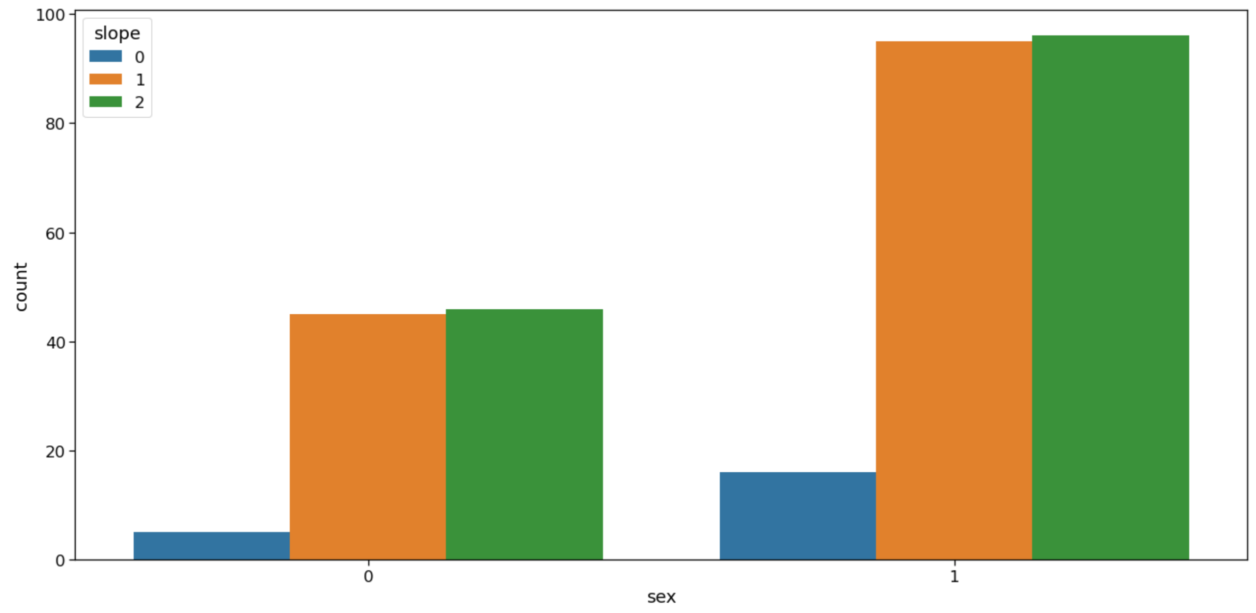


Inference: Here it is clearly visible that, Ratio of Male to Female is approx 2:1.

Plotting the relation between sex and slope.

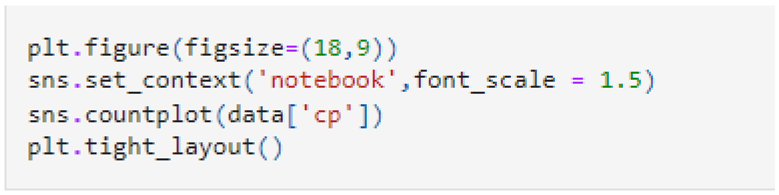
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['sex'],hue=data["slope"])
plt.tight_layout()
```

Output:



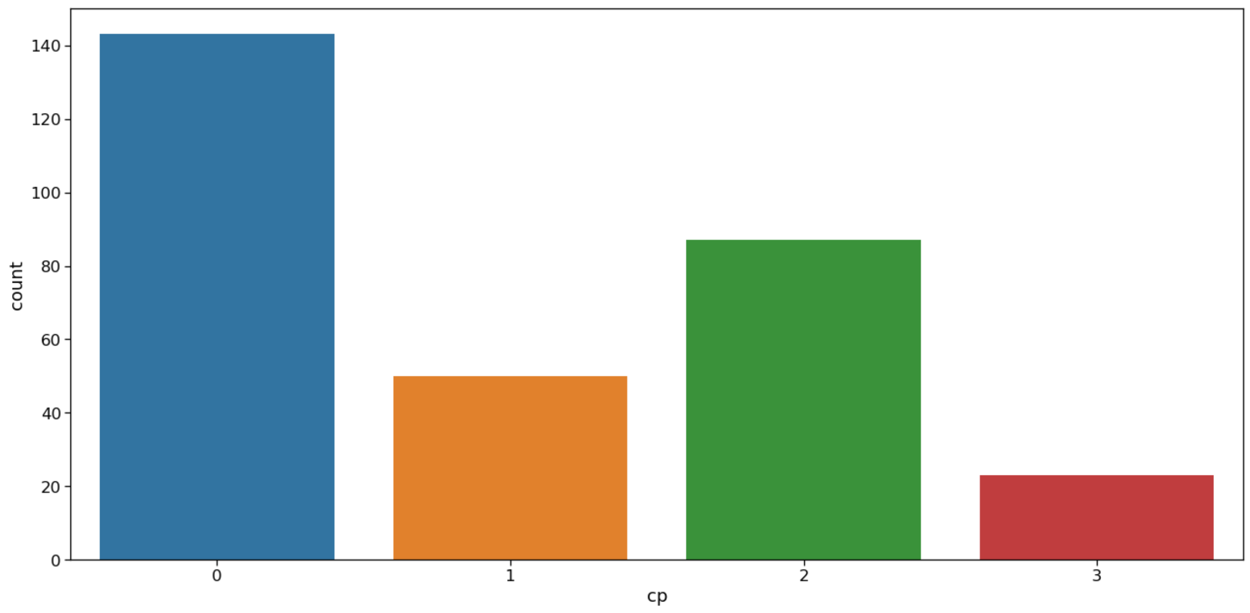
Inference: Here it is clearly visible that the slope value is higher in the case of males(1).

Chest Pain Type("cp") Analysis



```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['cp'])
plt.tight_layout()
```

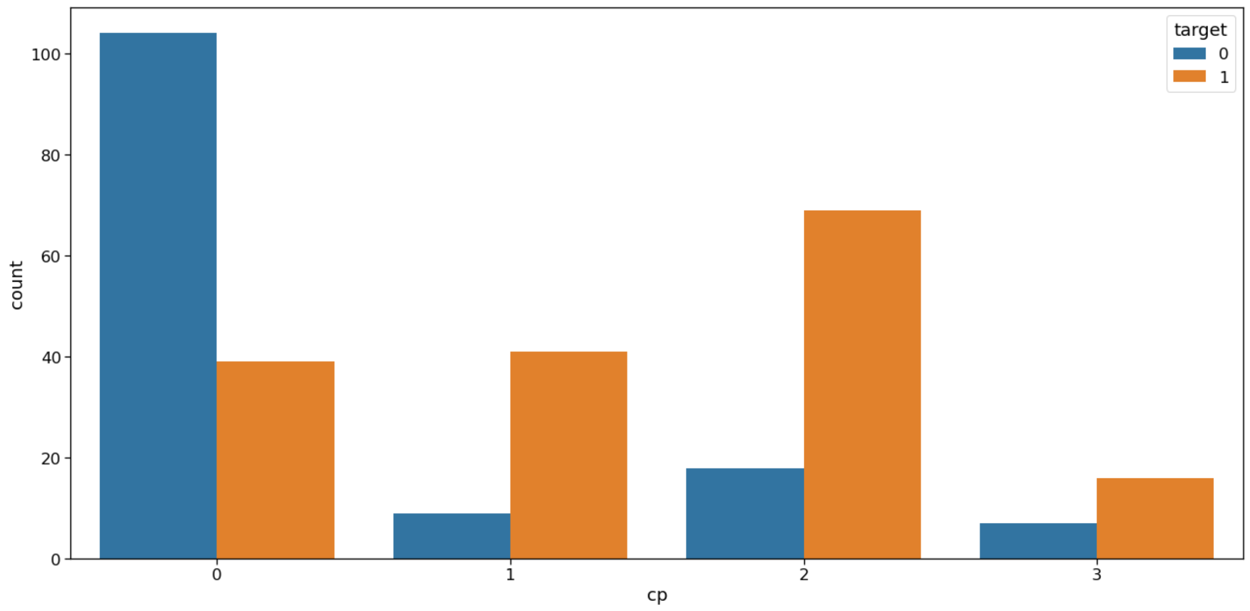
Output:



Inference: As seen, there are 4 types of chest pain

1. status at least
2. condition slightly distressed
3. condition medium problem
4. condition too bad

Analyzing cp vs target column



Inference: From the above graph we can make some inferences,

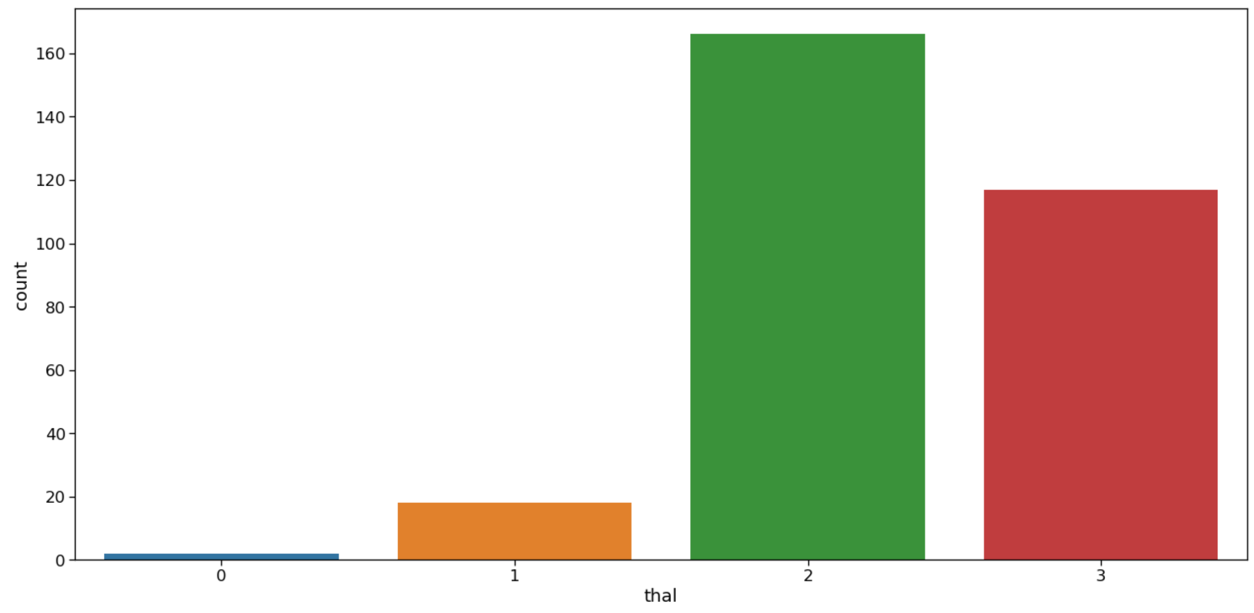
- People having the least chest pain are not likely to have heart disease.
- People having severe chest pain are likely to have heart disease.

Elderly people are more likely to have chest pain.

Thal Analysis

```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['thal'])
plt.tight_layout()
```

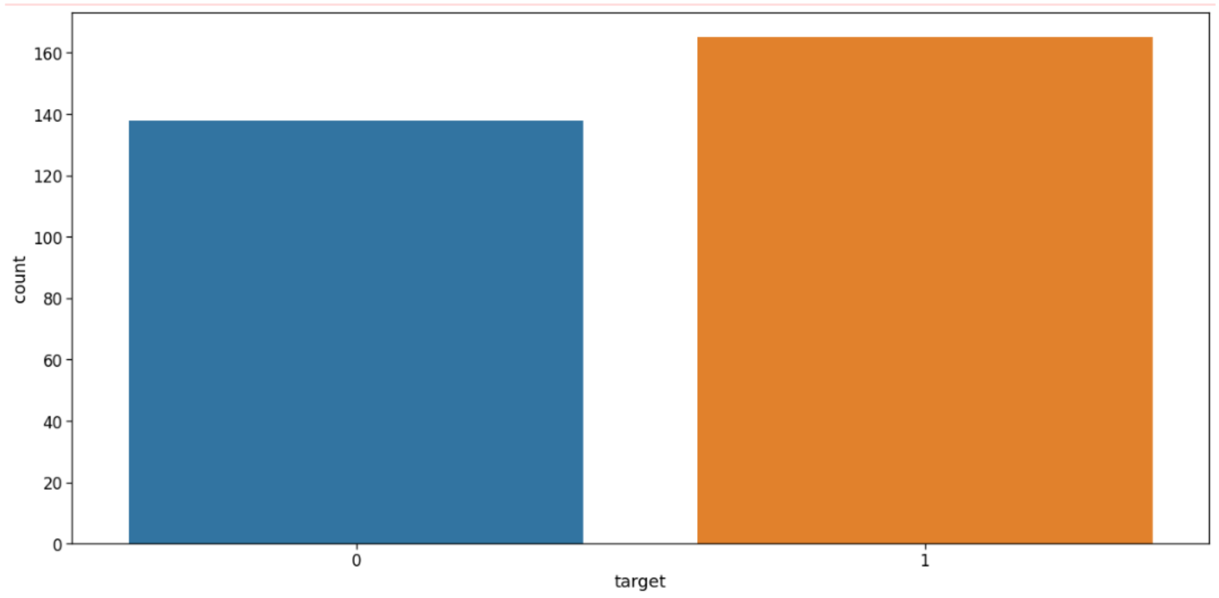
Output:



Target

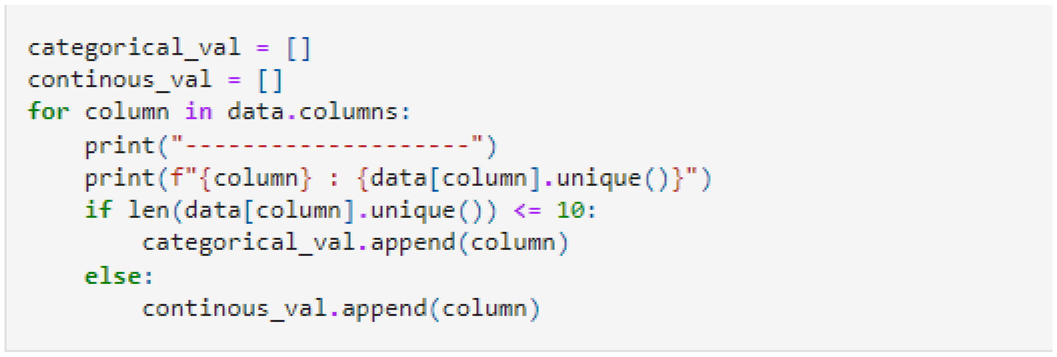
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['target'])
plt.tight_layout()
```

Output:



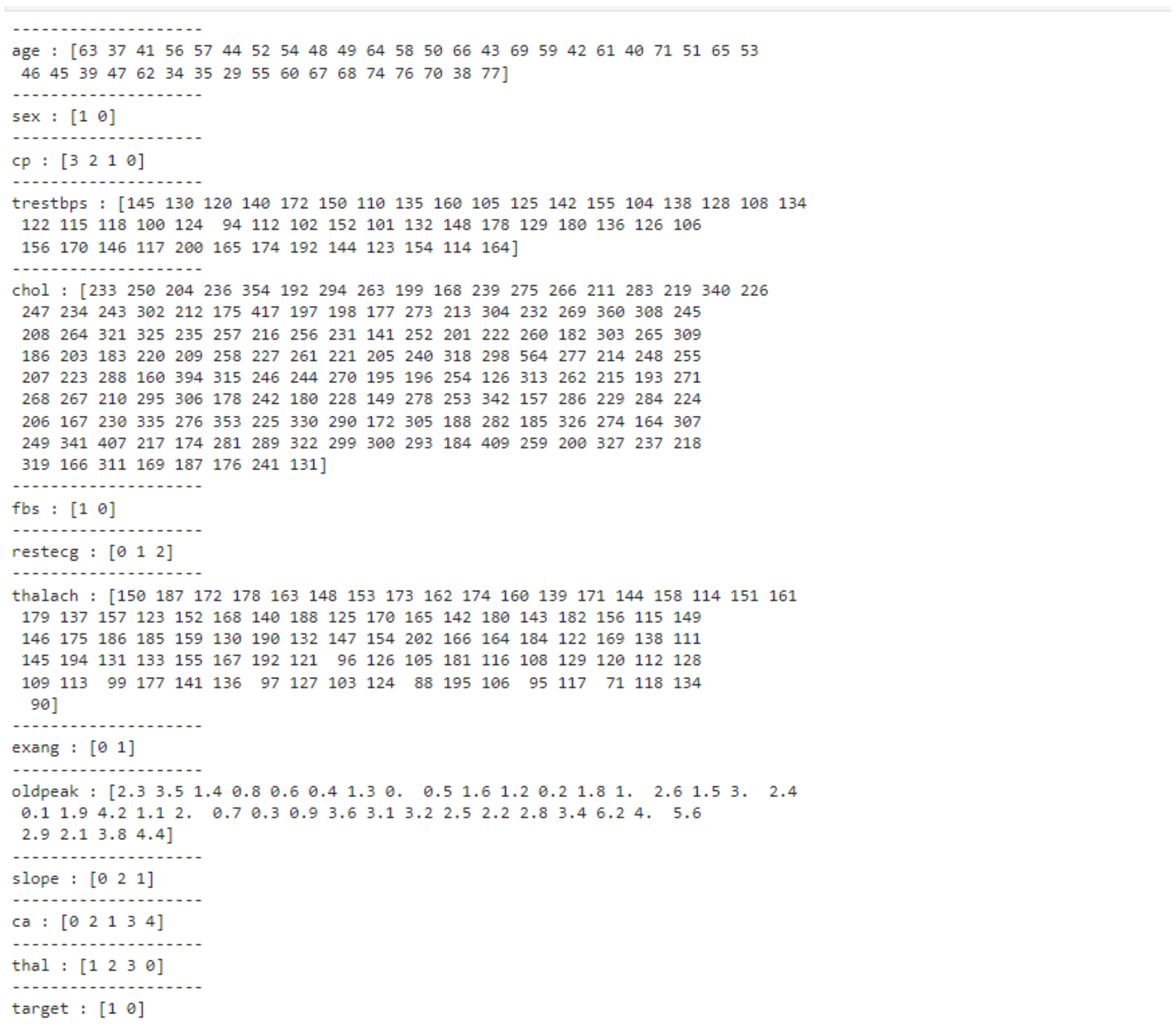
Feature Engineering

Complete description of the continuous data as well as the categorical data



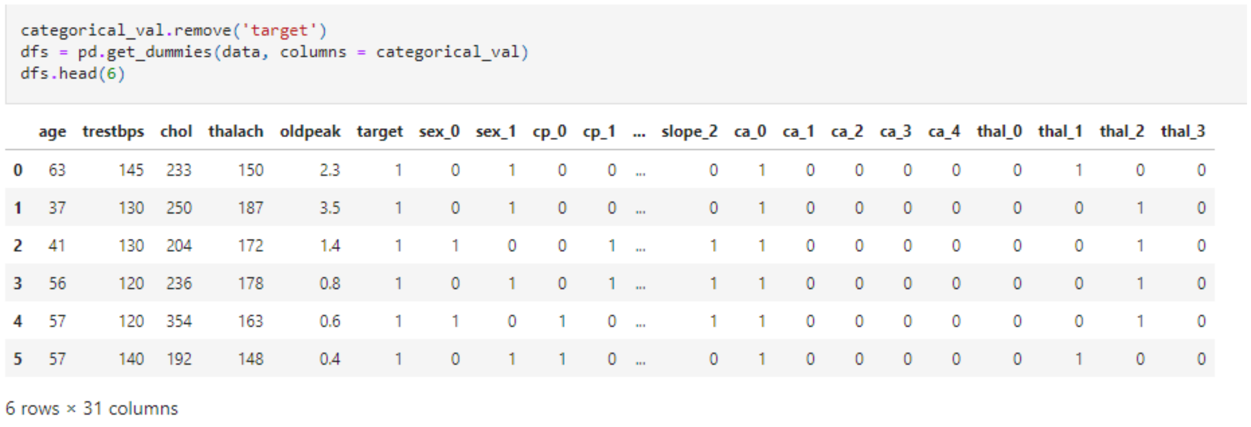
```
categorical_val = []
continous_val = []
for column in data.columns:
    print("--------------------")
    print(f"{column} : {data[column].unique()}")
    if len(data[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

Output:



```
--------------------
age : [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]
--------------------
sex : [1 0]
--------------------
cp : [3 2 1 0]
--------------------
trestbps : [145 130 120 140 172 150 110 135 160 105 125 142 155 104 138 128 108 134
 122 115 118 100 124  94 112 102 152 101 132 148 178 129 180 136 126 106
 156 170 146 117 200 165 174 192 144 123 154 114 164]
--------------------
chol : [233 250 204 236 354 192 294 263 199 168 239 275 266 211 283 219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308 245
 208 264 321 325 235 257 216 256 231 141 252 201 222 260 182 303 265 309
 186 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248 255
 207 223 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193 271
 268 267 210 295 306 178 242 180 228 149 278 253 342 157 286 229 284 224
 206 167 230 335 276 353 225 330 290 172 305 188 282 185 326 274 164 307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237 218
 319 166 311 169 187 176 241 131]
--------------------
fbs : [1 0]
--------------------
restecg : [0 1 2]
--------------------
thalach : [150 187 172 178 163 148 153 173 162 174 160 139 171 144 158 114 151 161
 179 137 157 123 152 168 140 188 125 170 165 142 180 143 182 156 115 149
 146 175 186 185 159 130 190 132 147 154 202 166 164 184 122 169 138 111
 145 194 131 133 155 167 192 121  96 126 105 181 116 108 129 120 112 128
 109 113  99 177 141 136  97 127 103 124  88 195 106  95 117  71 118 134
  90]
--------------------
exang : [0 1]
--------------------
oldpeak : [2.3 3.5 1.4 0.8 0.6 0.4 1.3 0.  0.5 1.6 1.2 0.2 1.8 1.  2.6 1.5 3.  2.4
 0.1 1.9 4.2 1.1 2.  0.7 0.3 0.9 3.6 3.1 3.2 2.5 2.2 2.8 3.4 6.2 4.  5.6
 2.9 2.1 3.8 4.4]
--------------------
slope : [0 2 1]
--------------------
ca : [0 2 1 3 4]
--------------------
thal : [1 2 3 0]
--------------------
target : [1 0]
```

Here first we will be removing the target column from our set of features then we will categorize all the categorical variables using the get_dummies method which will create a separate column for each category suppose X variable contains 2 types of unique values then it will create 2 different columns for the X variable.

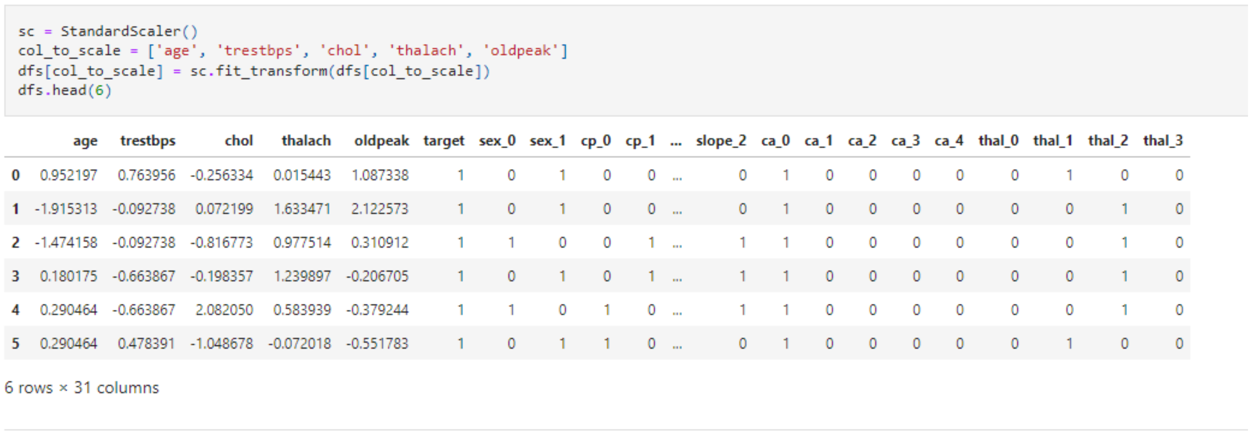


```
categorical_val.remove('target')
dfs = pd.get_dummies(data, columns = categorical_val)
dfs.head(6)
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	63	145	233	150	2.3	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	37	130	250	187	3.5	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	41	130	204	172	1.4	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	56	120	236	178	0.8	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	57	120	354	163	0.6	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0
5	57	140	192	148	0.4	1	0	1	1	0	...	0	1	0	0	0	0	0	1	0	0

6 rows × 31 columns

We used the standard scaler method to scale down the data so that it won’t raise the outliers. Also the dataset which is scaled to general units leads to having better accuracy.



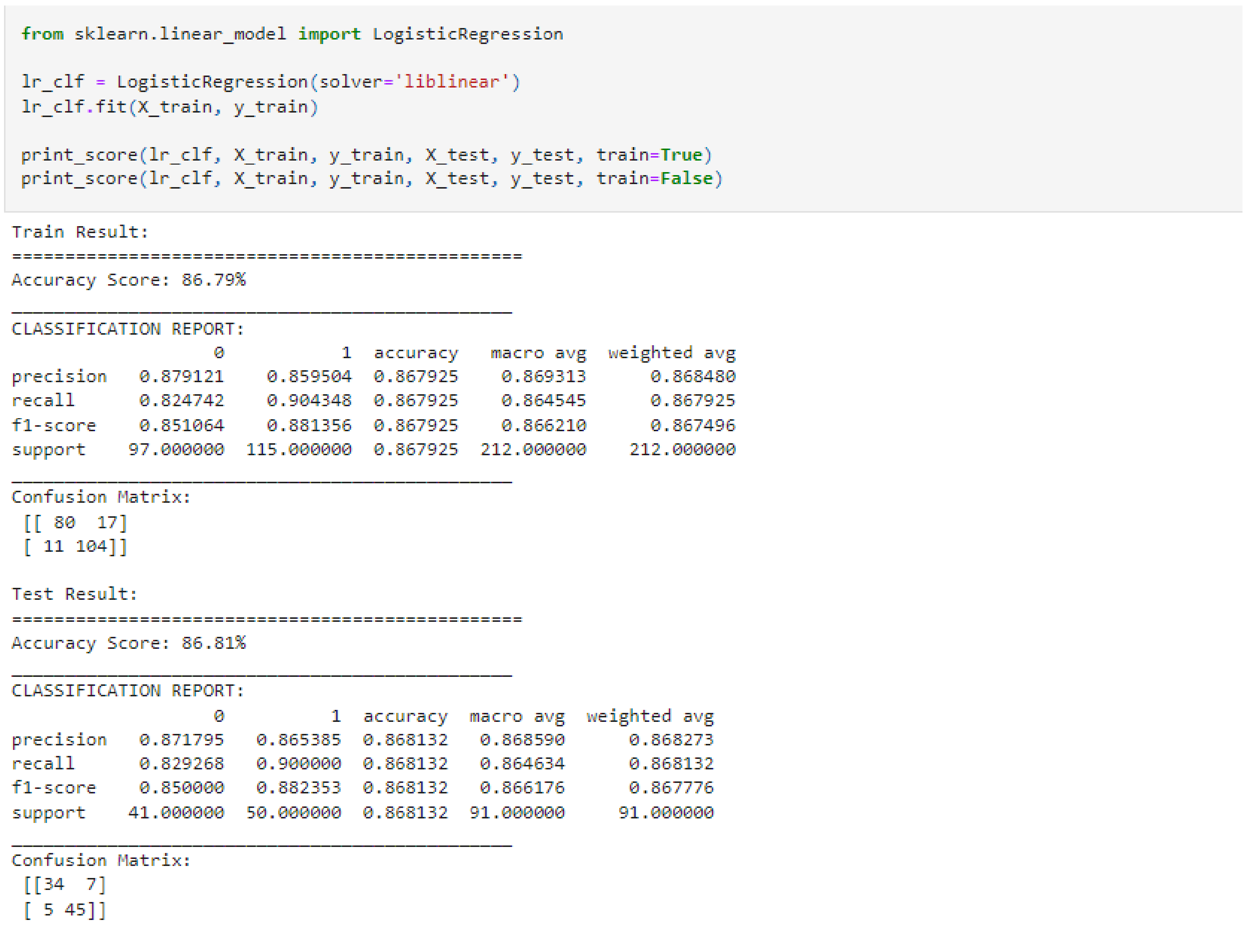
```
sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dfs[col_to_scale] = sc.fit_transform(dfs[col_to_scale])
dfs.head(6)
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	0.952197	0.763956	-0.256334	0.015443	1.087338	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	-1.915313	-0.092738	0.072199	1.633471	2.122573	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	0.290464	-0.663867	2.082050	0.583939	-0.379244	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0
5	0.290464	0.478391	-1.048678	-0.072018	-0.551783	1	0	1	1	0	...	0	1	0	0	0	0	0	1	0	0

6 rows × 31 columns

TESTS REPORTS

Using logistic regression:



```
from sklearn.linear_model import LogisticRegression

lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 86.79%
_______________________________________________
CLASSIFICATION REPORT:
                   0           1  accuracy   macro avg  weighted avg
precision   0.879121    0.859504  0.867925    0.869313      0.868480
recall      0.824742    0.904348  0.867925    0.864545      0.867925
f1-score    0.851064    0.881356  0.867925    0.866210      0.867496
support    97.000000  115.000000  0.867925  212.000000    212.000000
_______________________________________________
Confusion Matrix:
 [[ 80  17]
 [ 11 104]]

Test Result:
================================================
Accuracy Score: 86.81%
_______________________________________________
CLASSIFICATION REPORT:
                   0          1  accuracy  macro avg  weighted avg
precision   0.871795   0.865385  0.868132   0.868590      0.868273
recall      0.829268   0.900000  0.868132   0.864634      0.868132
f1-score    0.850000   0.882353  0.868132   0.866176      0.867776
support    41.000000  50.000000  0.868132  91.000000     91.000000
_______________________________________________
Confusion Matrix:
 [[34  7]
 [ 5 45]]
```

Using random forest classifier:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

rf_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
              0      1  accuracy  macro avg  weighted avg
precision   1.0    1.0       1.0        1.0           1.0
recall      1.0    1.0       1.0        1.0           1.0
f1-score    1.0    1.0       1.0        1.0           1.0
support    97.0  115.0       1.0      212.0         212.0
_______________________________________________
Confusion Matrix:
 [[ 97   0]
 [  0 115]]

Test Result:
================================================
Accuracy Score: 82.42%
_______________________________________________
CLASSIFICATION REPORT:
                   0      1  accuracy  macro avg  weighted avg
precision   0.804878   0.84  0.824176   0.822439      0.824176
recall      0.804878   0.84  0.824176   0.822439      0.824176
f1-score    0.804878   0.84  0.824176   0.822439      0.824176
support    41.000000  50.00  0.824176  91.000000     91.000000
_______________________________________________
Confusion Matrix:
 [[33  8]
 [ 8 42]]
```

REQUIREMENTS

Libraries used

Pandas: Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

Numpy: NumPy is a Python library used for working with arrays.It also has functions for working in the domain of linear algebra, fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.Arrays are very frequently used in data science, where speed and resources are very important.NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.This behavior is called locality of reference in computer science.This is the main reason why NumPy is faster than lists. Also it is optimized to work with the latest CPU architectures.

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

Matplotlib: Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in 2002.
One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Seaborn: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Cufflinks: Cufflink is also a python library that connects plotly with pandas so that we can create charts directly on data frames. It basically acts as a plugin.

Xgboost: XGBoost is short for Extreme Gradient Boost. Unlike Gradient Boost, XGBoost makes use of regularization parameters that helps against overfitting.

Catboost: CatBoost is a high-performance open source library for gradient boosting on decision trees.

Hardware requirements:
Processor : Any Update Processor
Ram : Min 4GB
Hard Disk : Min 100GB

Software requirements:
Operating System : Windows family
Technology : Python3.7
IDE : Jupyter notebook

CHALLENGES FACED

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either they are expensive or are not efficient to calculate the chance of heart disease in humans. Early detection of cardiac diseases can decrease the mortality rate and overall complications. However, it is not possible to monitor patients everyday in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more patience, time and expertise. Since we have a good amount of data in today’s world, we can use various machine learning algorithms to analyze the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

WHAT HAVE YOU LEARNT?

Heart diseases are a major killer in India and throughout the world. Application of promising technology like machine learning to the initial prediction of heart diseases will have a profound impact on society. The early prognosis of heart disease can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine. The number of people facing heart diseases is on a raise each year. This prompts for its early diagnosis and treatment. The utilization of suitable technology support in this regard can prove to be highly beneficial to the medical fraternity and patients.

The Heart Disease prediction have the following key takeaways:

1. Data insight: We worked with the heart disease detection dataset and put out interesting inferences from the data to derive some meaningful results.
2. EDA: Exploratory data analysis is the key step for getting meaningful results.
3. Feature engineering: After getting the insights from the data we altered the features.
4. Model building: In this phase, we built our Machine learning model for heart disease detection.

ACCOMPLISHMENTS

1. We did data visualization and data analysis of the target variable, age features, and whatnot along with its univariate analysis and bivariate analysis.

2. We also did a complete feature engineering part in this article which summons all the valid steps needed for further steps i.e. model building.

3. From the above model accuracy, logistic regression is giving accuracy of 86% and random forest classifier is giving the accuracy of 82%.