HEART-ATTACK ANALYSIS

INTRODUCTION

A heart attack occurs when one or more of your coronary arteries becomes blocked. Over time, a buildup of fatty deposits, including cholesterol, form substances called plaques, which can narrow the arteries (atherosclerosis). This condition, called coronary artery disease, causes most heart attacks.During a heart attack, a plaque can rupture and spill cholesterol and other substances into the bloodstream. A blood clot forms at the site of the rupture. If the clot is large, it can block blood flow through the coronary artery, starving the heart of oxygen and nutrients (ischemia).

You might have a complete or partial blockage of the coronary artery.

- A complete blockage means you've had an ST elevation myocardial infarction (STEMI).
- A partial blockage means you've had a non-ST elevation myocardial infarction (NSTEMI).

Diagnosis and treatment might be different depending on which type you've had.Another cause of a heart attack is a spasm of a coronary artery that shuts down blood flow to part of the heart muscle. Using tobacco and illicit drugs, such as cocaine, can cause a life-threatening spasm.Infection with COVID-19 also may damage your heart in ways that result in a heart attack

Cardiovascular diseases (CVDs) or heart disease are the number one cause of death globally with 17.9 million death cases each year. CVDs are concertedly contributed by hypertension, diabetes, overweight and unhealthy lifestyles. You can read more on the heart disease statistics and causes for self-understanding. This project covers manual

exploratory data analysis and using pandas profiling in Jupyter Notebook, on Google Colab. The dataset used in this project is UCI Heart Disease dataset,

Data Set Explanations

Initially, the dataset contains 76 features or attributes from 303 patients; however, published studies chose only 14 features that are relevant in predicting heart disease. Hence, here we will be using the dataset consisting of 303 patients with 14 features set.

Motivation

Exploratory Data Analysis (EDA) is a pre-processing step to understand the data. There are numerous methods and steps in performing EDA, however, most of them are specific, focusing on either visualization or distribution, and are incomplete. Therefore, here, I will walk-through step-by-step to understand, explore, and extract the information from the data to answer the questions or assumptions. There are no structured steps or method to follow, however, this project will provide an insight on EDA for you and my future self.

Variable description:

1. 'age'= Age of the person (in years)
2. 'sex' = Gender of the person (1=Male ; 0=Female)
3. 'cp' = Chest Pain type
    a. Value 0: typical angina
    b. Value 1: atypical angina
    c. Value 2: non-anginal pain
    d. Value 3: asymptomatic
4. 'trtbps' = resting blood pressure (in mm Hg on admission to the hospital)
5. 'chol' = cholestoral in mg/dl (fetched via BMI sensor)
6. 'fbs' = (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. 'restecg' = resting electrocardiographic results
    a. Value 0: normal
    b. Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    c. Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8. 'thalachh' = maximum heart rate achieved
9. 'exng' = exercise induced angina (1 = yes; 0 = no)
10. 'oldpeak' = Previous peak or ST depression induced by exercise relative to rest
11. 'slp' = The slope of the peak exercise ST segment
    a. Value 0: upsloping
    b. Value 1: flat

c. Value 2: downsloping

12. 'caa' = number of major vessels
13. 'thall' = Thal rate
14. 'output' = Target variable OR diagnosis of heart disease (angiographic disease status)
    a. Value 0: < 50% diameter narrowing
    b. Value 1: > 50% diameter narrowing

DATA PREPROCESSING:

- A glimpse of the dataset in use:

```
In [3]: df.head()
```

Out[3]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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

- The datatypes of the columns in the dataset are:

```
In [7]: df.dtypes
Out[7]: age           int64
        sex           int64
        cp            int64
        trtbps        int64
        chol          int64
        fbs           int64
        restecg       int64
        thalachh      int64
        exng          int64
        oldpeak     float64
        slp           int64
        caa           int64
        thall         int64
        output        int64
        dtype: object
```

We see that all the variables in the dataset are integer variables except for the ‘oldpeak’

variable.

- Grouping together the categorical and continuous variables:

```
In [8]: dict = {}
        for x in list(df.columns):
            dict[x] = df[x].value_counts().shape[0]

        pd.DataFrame(dict, index=["Unique Counts"]).transpose()
```

Out[8]:

	Unique Counts
age	41
sex	2
cp	4
trtbps	49
chol	152
fbs	2
restecg	3
thalachh	91
exng	2
oldpeak	40
slp	3
caa	5
thall	4
output	2

  - From the above output, we find that :

- Categorical variables: sex, cp, fbs, restecg, exng, slp, caa, thall, output.
- Continuous variables: age, trtbps, chol, thalachh, oldpeak.
- Inconsistencies:
    - We find that caa and thall have 5 and 4 values respectively.
    - Whereas it should have 4 and 3 values respectively.

- Treatment of the inconsistent data:
    - For column ‘caa’:

■ Checking the values present in the ‘caa’ column:

```
In [10]: df.caa.value_counts()
         #has 5 values '4' which shouldnt be there

Out[10]: 0    175
         1     65
         2     38
         3     20
         4      5
         Name: caa, dtype: int64
```

■ Replacing the values of 4 by NaN:

```
In [13]: #replacing the values of 4 by NaN
         df.loc[df['caa']==4,'caa']=np.NaN
```

- For column ‘thall’:

■ Checking the values present in the ‘thall’ column:

```
In [23]: df.thall.value_counts()

Out[23]: 2    166
         3    117
         1     18
         0      2
         Name: thall, dtype: int64
```

■ Replacing the values of 0 by NaN:

```
In [24]: #replacing the values of 4 by NaN
         df.loc[df['thall']==0,'thall']=np.NaN
```

- **TREATMENT OF THE INCONSISTENT DATA:**

We have treated the inconsistent data in the dataset using 'imputation by median value'.

```
In [34]: #inconsistent value impuatation by median
         df=df.fillna(df.median())
         df.isna().sum()

Out[34]: age         0
         sex         0
         cp          0
         trtbps      0
         chol        0
         fbs         0
         restecg     0
         thalachh    0
         exng        0
         oldpeak     0
         slp         0
         caa         0
         thall       0
         output      0
         dtype: int64
```

- **SUMMARY OF THE DATASET AFTER PREPROCESSING:**

```
In [21]: df.describe()
```

Out[21]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpe
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039(
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161(
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000(
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000(
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200(

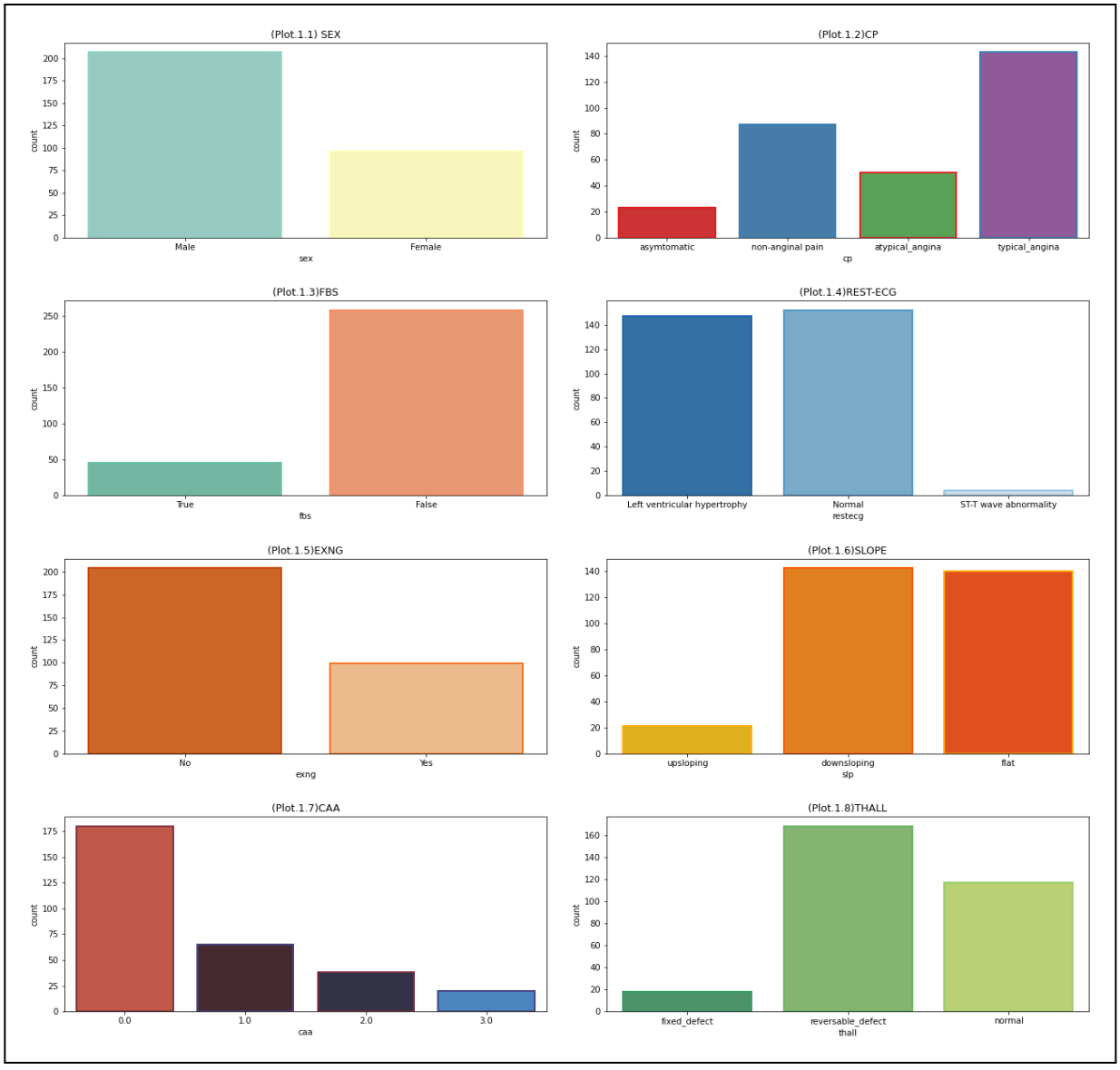
EXPLORATORY DATA ANALYSIS:

- ## UNIVARIATE ANALYSIS:

  - ### Categorical variables:

```
fig, axes = plt.subplots(4,2,figsize=(18,18))
axes[0, 0].set_title('(Plot.1.1) SEX')
sns.countplot(df1.sex,
              palette = 'Set3',
              edgecolor=sns.color_palette("Set3", 4),
              linewidth=2,
             ax=axes[0,0]);

axes[0, 1].set_title('(Plot.1.2)CP')
sns.countplot(df1.cp,
              palette = 'Set1',
              edgecolor=sns.color_palette("Set1", 2),
              linewidth=2,
             ax=axes[0,1]);

axes[1, 0].set_title('(Plot.1.3)FBS')
sns.countplot(df1.fbs,
              palette = 'Set2',
              edgecolor=sns.color_palette("Set2", 2),
              linewidth=2,
             ax=axes[1,0]);

axes[1, 1].set_title('(Plot.1.4)REST-ECG')
sns.countplot(df1.restecg,
              palette = 'Blues_r',
              edgecolor=sns.color_palette('Blues_r', 4),
              linewidth=2,
             ax=axes[1,1]);
axes[2, 0].set_title('(Plot.1.5)EXNG')
sns.countplot(df1.exng,
              palette = 'Oranges_r',
              edgecolor=sns.color_palette('Oranges_r', 4),
              linewidth=2,
             ax=axes[2,0]);
```

```
axes[2, 1].set_title('(Plot.1.6)SLOPE')
sns.countplot(df1.slp,
              palette = 'autumn_r',
              edgecolor=sns.color_palette('autumn_r', 2),
              linewidth=2,
             ax=axes[2,1]);

axes[3, 0].set_title('(Plot.1.7)CAA')
sns.countplot(df1.caa,
              palette = 'icefire_r',
              edgecolor=sns.color_palette('icefire_r', 2),
              linewidth=2,
             ax=axes[3,0]);

axes[3, 1].set_title('(Plot.1.8)THALL')
sns.countplot(df1.thall,
              palette = 'summer',
              edgecolor=sns.color_palette('summer', 4),
              linewidth=2,
             ax=axes[3,1]);

plt.tight_layout(pad=3);
```



Conclusion:- From the above graphs, we can conclude that:

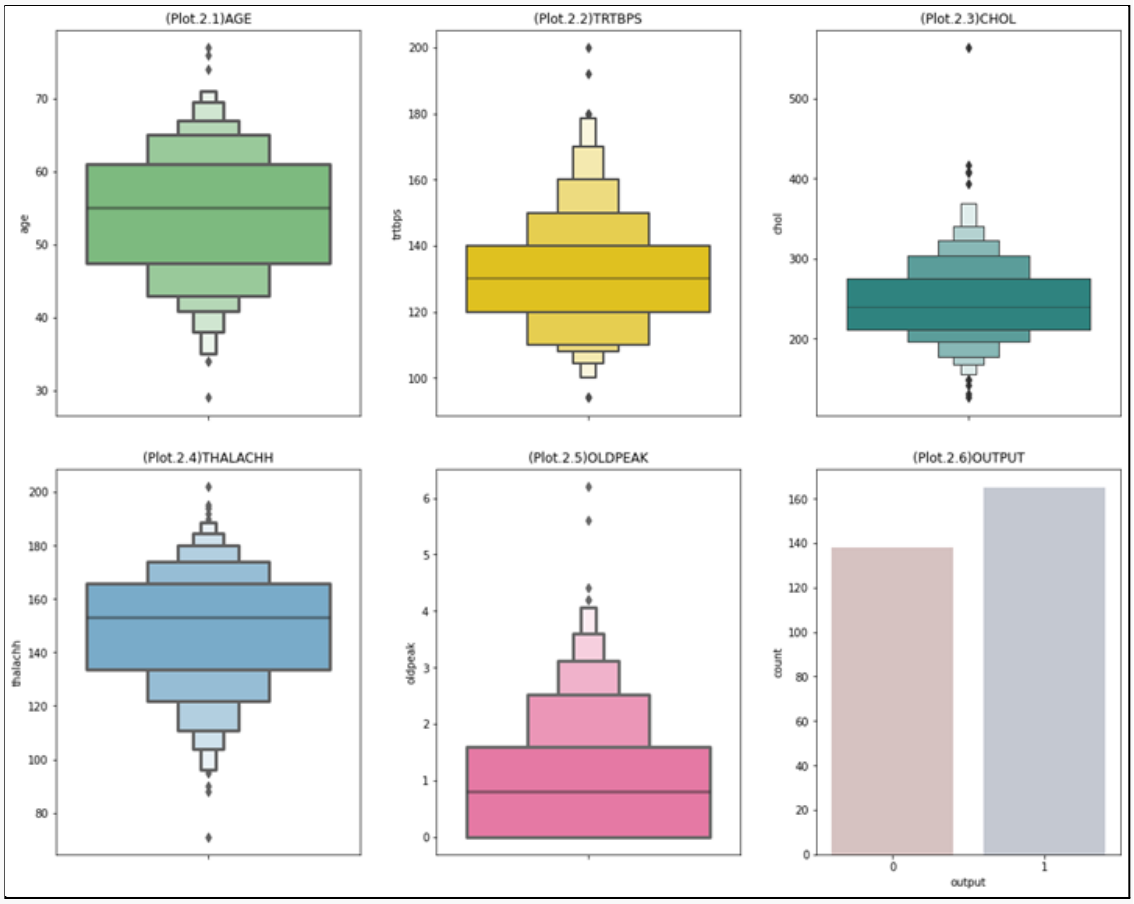
- In the dataset provided, the number of male patients is more than the number of female patients.
- The number of patients in the dataset suffering from typical anginal are much greater in number than other causes.

- Most patients have their fasting blood sugar less than 120mg/dl .
- Most patients have either normal or have left ventricular hypertrophy.
- Most patients' slopes of the peak exercise ST segment are either flat or upsloping.
- Most patients are not suffering with exercise induced angina.
- Majority of the people have 0 major vessels.
- Most patients have a 'reversible defect'.

Continuous and target variable:

```
fig, axes = plt.subplots(2,3, figsize=(15,12))

#use the axis for plotting
axes[0, 0].set_title('(Plot.2.1)AGE')
sns.boxenplot(y=data.age,
              palette='Greens',
              color='red',
             linewidth=3,
             ax=axes[0,0]);


#use the axis for plotting
axes[0,1].set_title('(Plot.2.2)TRTBPS')
sns.boxenplot(y=data.trtbps,
              palette='prism',
              color='red',
             linewidth=2,
             ax=axes[0,1]);


#use the axis for plotting
axes[0, 2].set_title('(Plot.2.3)CHOL')
sns.boxenplot(y=data.chol,
              palette='viridis',
             linewidth=1,
             ax=axes[0,2]);
```

```
#use the axis for plotting
axes[1, 0].set_title('(Plot.2.4)THALACHH')
sns.boxenplot(y=data.thalachh,
              palette='Blues_r',
              color='red',
             linewidth=3,
             ax=axes[1,0]);


#use the axis for plotting
axes[1, 1].set_title('(Plot.2.5)OLDPEAK')
sns.boxenplot(y=data.oldpeak,
              palette='RdPu',
              color='red',
             linewidth=3,
             ax=axes[1,1]);


#use the axis for plotting
axes[1, 2].set_title('(Plot.2.6)OUTPUT')
sns.countplot(data.output,
               palette = 'vlag_r',
               saturation=0.50,
               ax=axes[1,2]);
```



Conclusion:

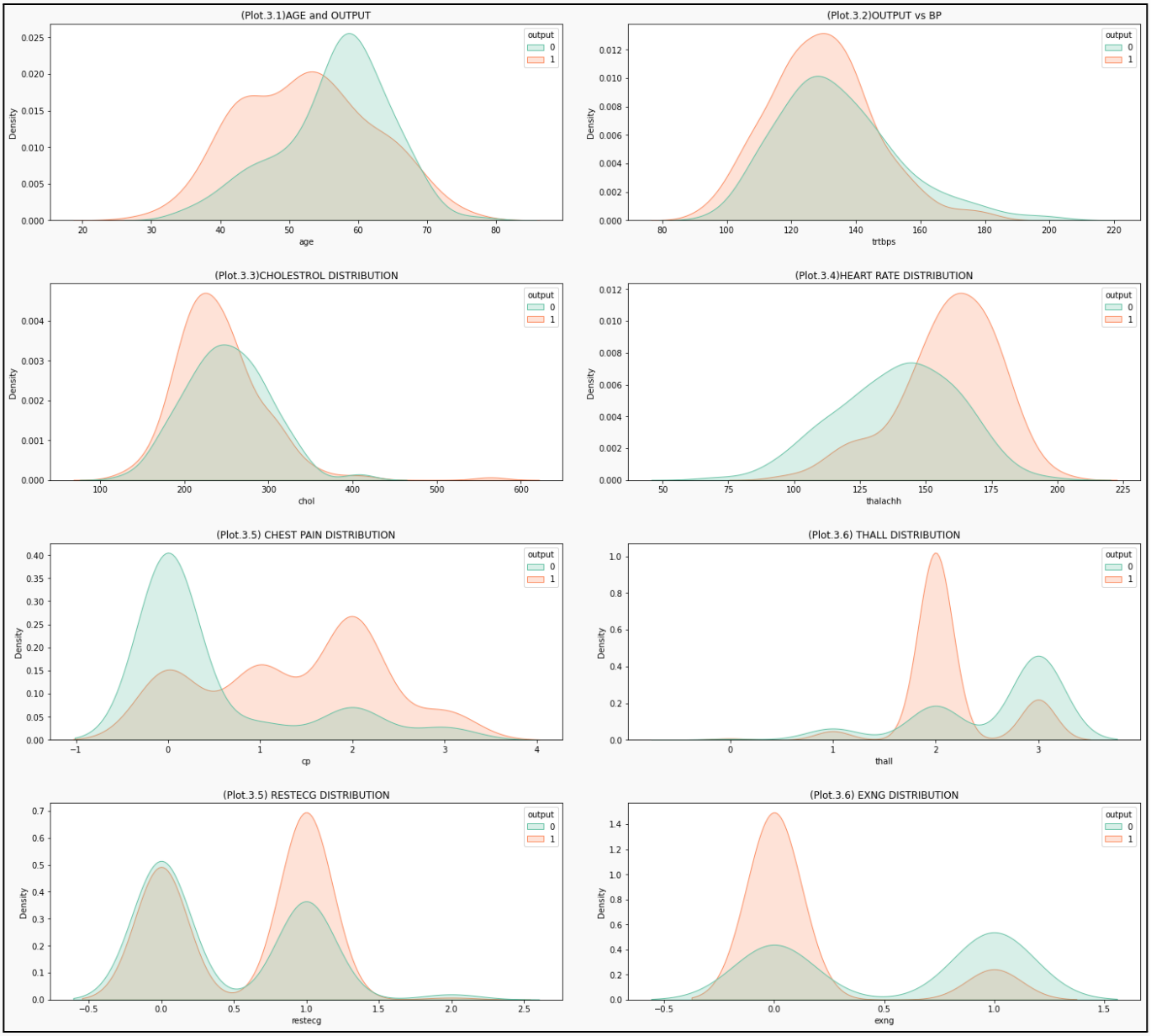
- From the graphical representation of the dataset we can see that most of the patients are between the ages (48-61) .
- A large number of patients have their blood pressure between (120-140).
- Also, most patients have their cholesterol level between (220-260).
- Most patients have their heart rate between (135-165) which is a contributing factor for heart attack.
- From the graphs we can conclude that the number of patients with higher blood pressure , cholesterol and heart rate levels are more prone to have a heart-attack.

- **Bivariate Analysis:**

In [11]:
```
fig, axes = plt.subplots(4,2, figsize=(20,18))

#use the axis for plotting
axes[0, 0].set_title('(Plot.3.1)AGE and OUTPUT')
sns.kdeplot(x=df.age,
            hue=df.output,
            fill=True,
            palette= 'Set2',
            ax=axes[0,0])

#use the axis for plotting
axes[0, 1].set_title('(Plot.3.2)OUTPUT vs BP')
sns.kdeplot(x= df.trtbps,
            hue=df.output,
            fill=True,
            palette='Set2',
            ax=axes[0,1])


#use the axis for plotting
axes[1, 0].set_title('(Plot.3.3)CHOLESTROL DISTRIBUTION')
sns.kdeplot(x=df.chol,
            hue=df.output,
            fill=True,
            palette='Set2',
```

```
            palette='Set2',
            ax=axes[1,0])



#use the axis for plotting
axes[1, 1].set_title('(Plot.3.4)HEART RATE DISTRIBUTION')
sns.kdeplot(x=df.thalachh,
            hue=df.output,
            fill=True,
            palette='Set2',
            ax=axes[1,1])

#use the axis for plotting
axes[2, 0].set_title('(Plot.3.5) CHEST PAIN DISTRIBUTION')
sns.kdeplot(x=df.cp,
           hue=df.output,
           fill= True,
           palette = 'Set2',
           ax= axes[2,0])


#use the axis for plotting
axes[2, 1].set_title('(Plot.3.6) THALL DISTRIBUTION')
sns.kdeplot(x=df.thall,
           hue=df.output,
           fill= True,
           palette = 'Set2',
           ax= axes[2,1])
```

```
#use the axis for plotting
axes[3, 1].set_title('(Plot.3.6) EXNG DISTRIBUTION')
sns.kdeplot(x=df.exng,
           hue=df.output,
           fill= True,
           palette = 'Set2',
           ax= axes[3,1])


plt.tight_layout(pad=3);
```



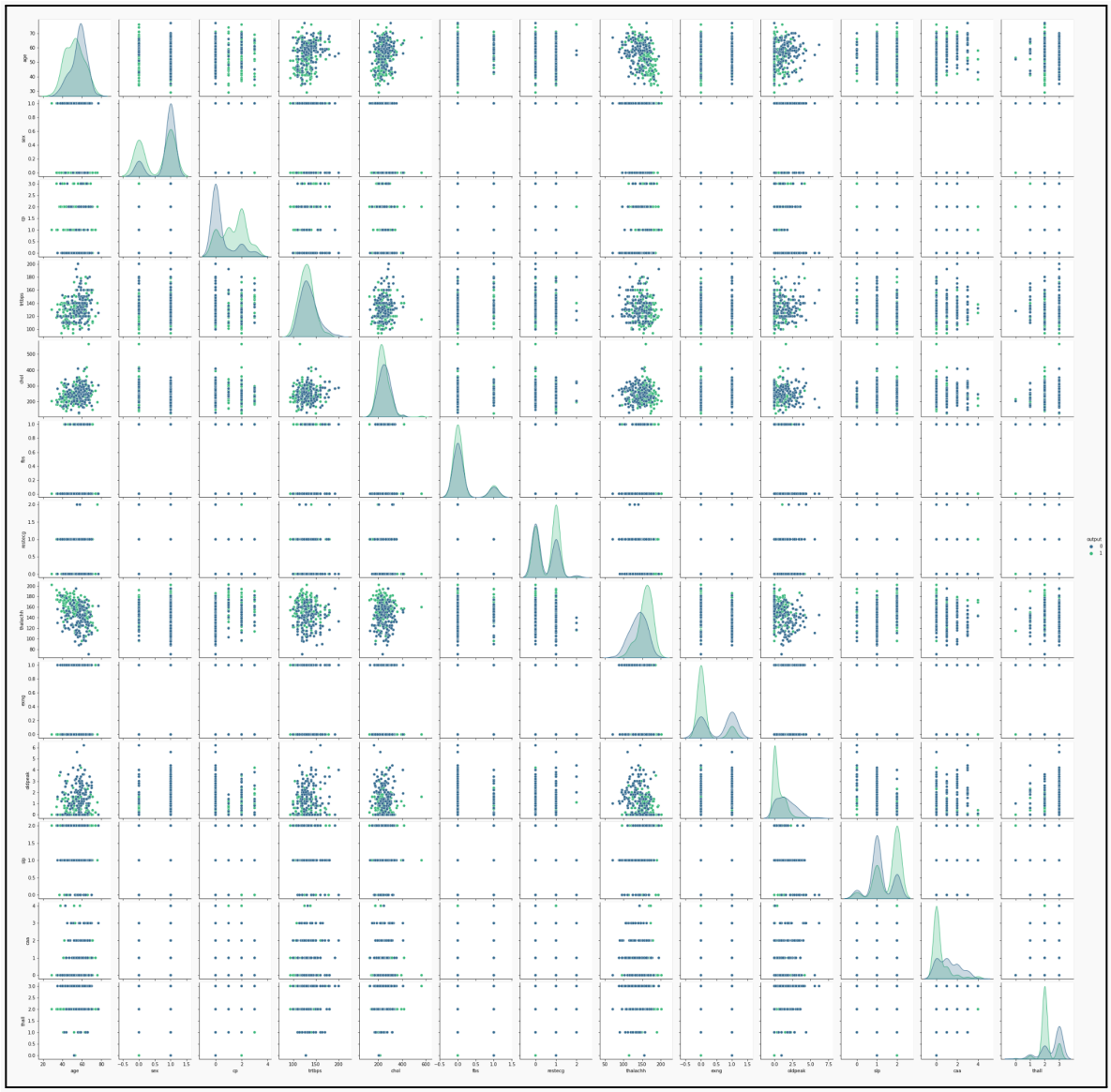
Conclusion:

- There is no relationship between age and heart attack. Therefore, the intuition of older age to heart attack is rejected.
- Blood Pressure and Cholesterol don't contribute to heart attack.

- Patients with higher heart rate,non-anginal chest pain,abnormal ST-T electrocardiographic waves are more prone to heart attack.

• Multivariate Analysis

```
In [12]: sns.pairplot(df,hue='output',palette='viridis')
         plt.savefig("Pairplot.png")
```

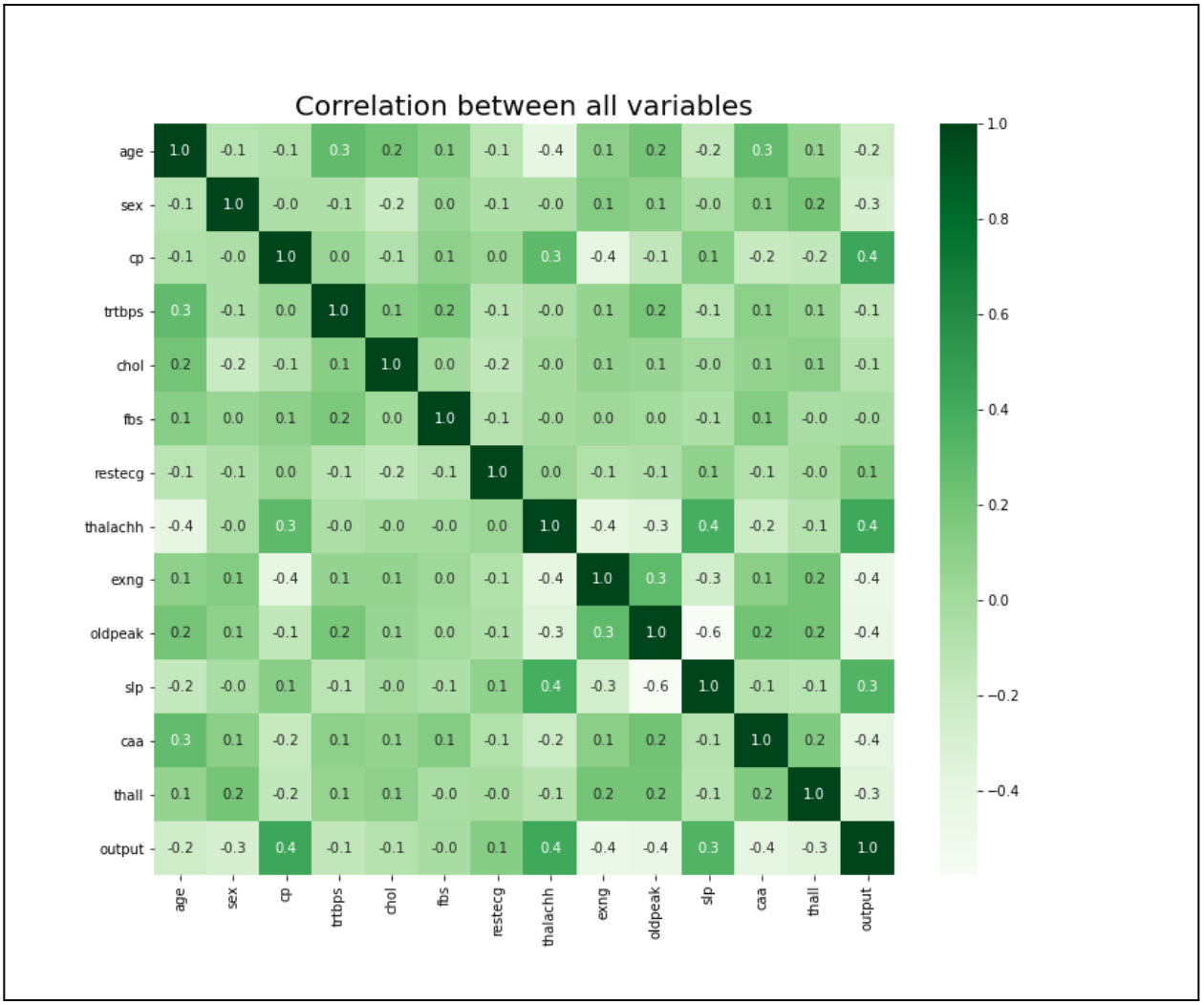


Conclusion:

- oldpeak having a linear separation relation between disease and non-disease.
- thalach having a mild separation relation between disease and non-disease.
- Other features don’t form any clear separation.

Correlation Between Variables

```
In [11]: plt.figure(figsize=(12,10))
         plt.title("Correlation between all variables",fontsize=20)
         sns.heatmap(df.corr(),fmt='.1f',annot=True,cmap='Greens')
```



Conclusions:

From the above diagram, we see that:'cp', 'thalach', 'slope' shows good positive correlation with target 'oldpeak', 'exang', 'ca', 'thal', 'sex', 'age' shows a good negative correlation with target 'fbs' 'chol', 'trestbps', 'restecg' has low very correlation with our target.

- **SCALING DATA:**

Scaling data

```
In [59]: from sklearn.preprocessing import MinMaxScaler
         scaler=MinMaxScaler()
         df[cont_var]=scaler.fit_transform(df[cont_var])
         df.head()
```

We have used MinMaxScaler() to normalize the continuous variables in our dataset.We have used normalization because :Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

We use feature scaling because in Gradient descent based machine learning models such as logistic regression having features on a similar scale can help the gradient descent converge more quickly towards the minima.

FORMATION OF MODELS BASED ON DIFFERENT ALGORITHMS:

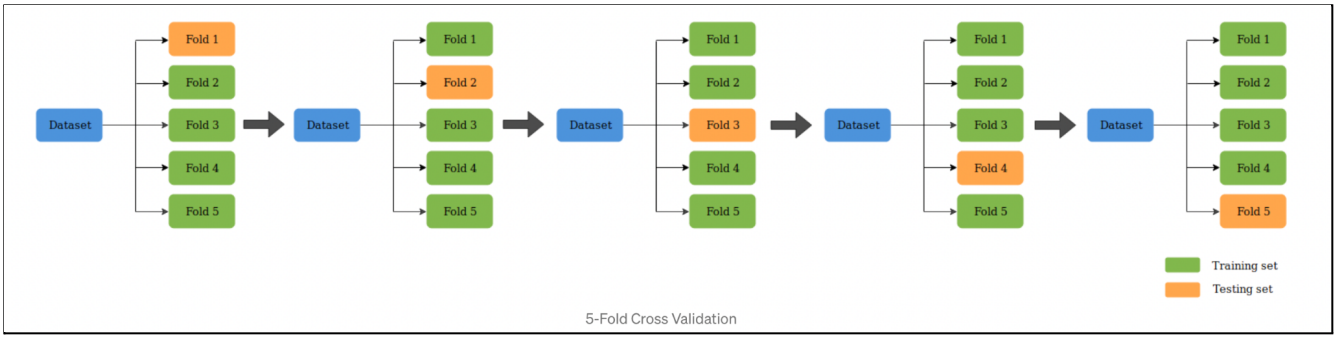
Separating the predictor and outcome variables:

```
In [60]: x=df.drop(columns='output')
         y=df['output']
```

In the above code, we have separated our predictor and outcome variables.

We have used five-fold cross validation in our models, which basically involves:

K-Fold CV is where a given data set is split into a ***K*** number of sections/folds where each fold is used as a testing set at some point. Let's take the scenario of 5-Fold cross validation(K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.



LOGISTIC REGRESSION MODEL :

In regression analysis, logistic regression is estimating the parameters of a logistic model . Logistic regression is a statistical model that in its basic form uses a logistic

function to model a binary dependent variable, although many more complex extensions exist.

```
In [62]: from sklearn.model_selection import StratifiedKFold
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import roc_auc_score
         acc=[]
         auc=[]
         skf = StratifiedKFold(n_splits=5, random_state=None)
         # X is the feature set and y is the target
         for train_index, test_index in skf.split(x,y):
             x_train, x_test = x.iloc[train_index], x.iloc[test_index]
             y_train, y_test = y.iloc[train_index], y.iloc[test_index]
             lr=LogisticRegression()
             lr.fit(x_train,y_train)
             y_pred=lr.predict(x_test)
             acc.append(accuracy_score(y_test,y_pred))
             auc.append(roc_auc_score(y_test,y_pred))
         tab_log=pd.DataFrame({'Accuracy':acc,'AUC':auc})
         tab_log
         print("Mean accuracy:",mean(acc))
         print("Mean AUC:",mean(auc))

         Mean accuracy: 0.8248633879781421
         Mean AUC: 0.8198292448292449
```

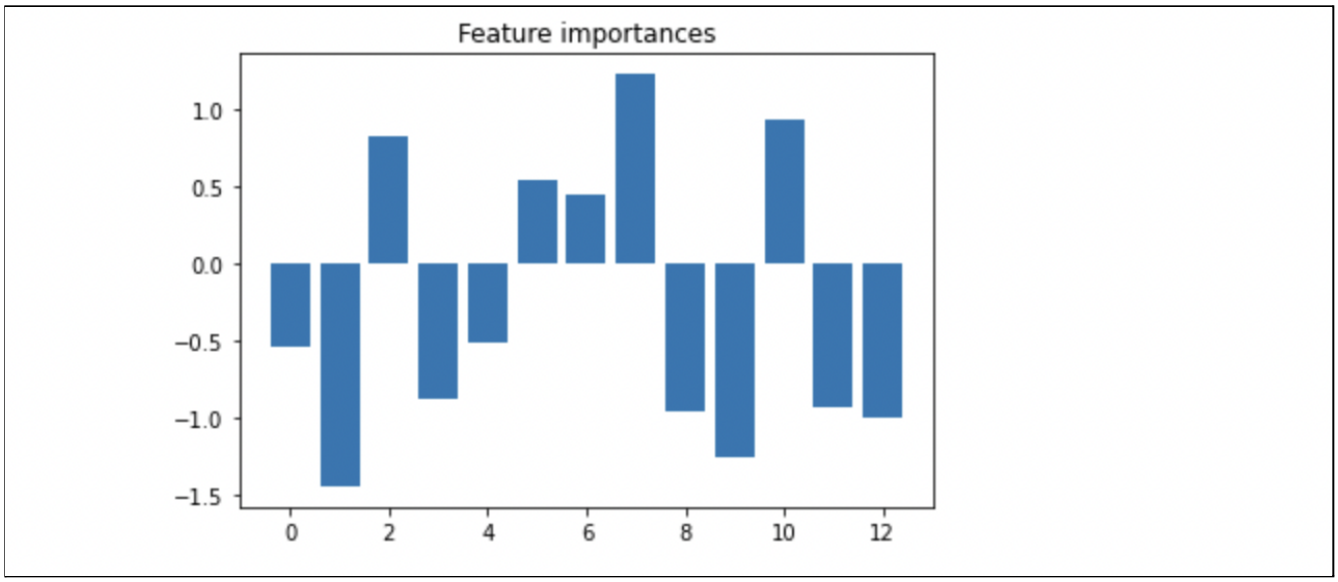
Conclusion

- **82.48%** is the mean accuracy of the logistic regression model using all the features.
- **81.98%** is the mean AUC score of the logistic regression model using all the features.
- AUC - The Area Under the ROC curve (AUC) is **an aggregated metric that evaluates how well a** logistic regression model classifies positive and negative outcomes at all possible cutoffs. It can range from 0.5 to 1, and the larger it is the better.

FEATURE IMPORTANCE IN LOGISTIC REGRESSION MODEL:

In [63]:
```
importance = lr.coef_[0]
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.title('Feature importances')
plt.show()
plt.savefig('F-1.png')
```

```
Feature: 0, Score: -0.54167
Feature: 1, Score: -1.44905
Feature: 2, Score: 0.81624
Feature: 3, Score: -0.88083
Feature: 4, Score: -0.52287
Feature: 5, Score: 0.53684
Feature: 6, Score: 0.44566
Feature: 7, Score: 1.22308
Feature: 8, Score: -0.96155
Feature: 9, Score: -1.25833
Feature: 10, Score: 0.92919
Feature: 11, Score: -0.93867
Feature: 12, Score: -1.00367
```



CONCLUSION: From the above graph, we see that in the Logistic regression model, the columns ‘age’,’restecg’,’exng’,’oldpeak’,’slp’,’caa’ are of utmost importance.

DECISION TREE MODEL:

In [65]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
acc_2=[]
auc_2=[]
skf = StratifiedKFold(n_splits=5, random_state=None)
# X is the feature set and y is the target
for train_index, test_index in skf.split(x,y):
    x_train, x_test = x.iloc[train_index], x.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    decTree = DecisionTreeClassifier(max_depth=6)
    decTree.fit(x_train,y_train)
    y_pred_decTree = decTree.predict(x_test)
    acc_2.append(accuracy_score(y_test,y_pred_decTree))
    auc_2.append(roc_auc_score(y_test,y_pred_decTree))
tab_dectree=pd.DataFrame({'Accuracy':acc_2,'AUC':auc_2})
tab_dectree
print("Mean accuracy:",mean(acc_2))
print("Mean AUC:",mean(auc_2))
```

```
Mean accuracy: 0.7622404371584699
Mean AUC: 0.7578042328042328
```

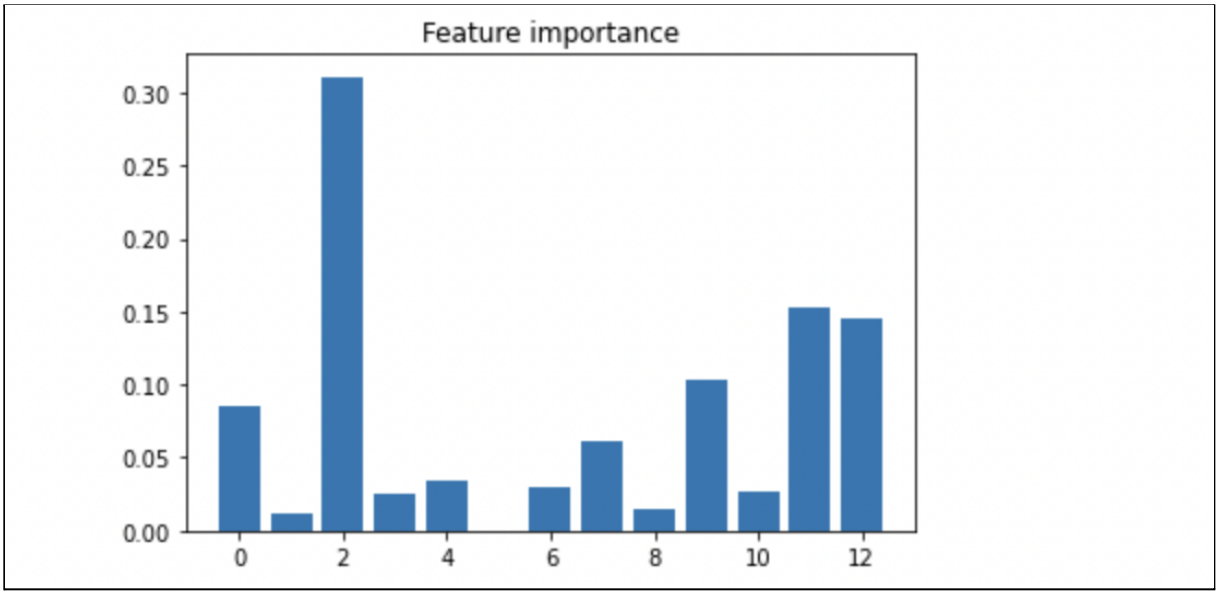
Conclusion

- **76.22%** is the mean accuracy of the decision tree model using all the features.
- **75.78%** is the mean AUC score of the decision tree model using all the features.

FEATURE IMPORTANCE IN DECISION TREE MODEL:

In [56]:
```
importance = decTree.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.02048
Feature: 1, Score: 0.01142
Feature: 2, Score: 0.31370
Feature: 3, Score: 0.02207
Feature: 4, Score: 0.03481
Feature: 5, Score: 0.00000
Feature: 6, Score: 0.03679
Feature: 7, Score: 0.05415
Feature: 8, Score: 0.01429
Feature: 9, Score: 0.10478
Feature: 10, Score: 0.04852
Feature: 11, Score: 0.19269
Feature: 12, Score: 0.14629
```



Conclusion**:** From the above graph, we see that in the decision model, the columns ‘sex’,’exng’,’oldpeak’,’slp’,’caa’ are of utmost importance.

XGBOOST MODEL:

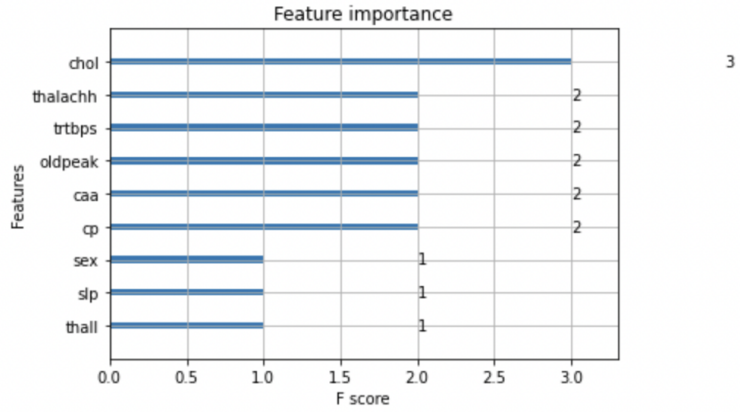
```
In [94]: from sklearn.model_selection import StratifiedKFold
         from xgboost import XGBClassifier
         from sklearn.metrics import accuracy_score
         acc_1=[]
         auc_1=[]
         skf = StratifiedKFold(n_splits=5, random_state=None)
         params = {
                     'objective':'binary:logistic',
                     'max_depth': 3,
                     'alpha': 10,
                     'learning_rate': 0.1,
                     'n_estimators':100,'eval_metric':'auc'
                 }

         # X is the feature set and y is the target
         for train_index, test_index in skf.split(x,y):
             x_train, x_test = x.iloc[train_index], x.iloc[test_index]
             y_train, y_test = y.iloc[train_index], y.iloc[test_index]
             xgb_clf = XGBClassifier(**params)
             xgb_clf.fit(x_train,y_train)
             y_pred_clf=xgb_clf.predict(x_test)
             acc_1.append(accuracy_score(y_test,y_pred_clf))
             auc_1.append(roc_auc_score(y_test,y_pred_clf))
         tab_xgb=pd.DataFrame({'Accuracy':acc_1,'AUC':auc_1})
         tab_xgb
         print("Mean accuracy:",mean(acc_1))
         print("Mean AUC:",mean(auc_1))

         Mean accuracy: 0.8381967213114754
         Mean AUC: 0.8327561327561328
```

Conclusion:

- **83.81%** is the mean accuracy of the xgboost model using all the features.
- **83.27%** is the mean AUC score of the xgboost model using all the features.

FEATURE IMPORTANCE OF XGBOOST MODEL:

```
In [25]: xgb.plot_importance(xgb_clf)
         plt.figure(figsize = (16, 12))
         plt.show()
```



Conclusion:

From the above graph, we see that in the XgBoost model, the column ‘chol’ is of utmost importance.The columns ‘thalachh’, ‘trtbps’, ‘oldpeak’, ‘caa’, ‘cp’ also also very important.

CONCLUSION:

From comparing the above models, we see that the XGBoost Model has the highest mean accuracy as well as mean AUC score. Although there is no definitive age to experience a heart attack, the kind of lifestyle choices you make, your diet plans, your workout routines and how you manage your stress levels can influence your probabilities.

FUTURE WORK:

- Explore the Explainable AI concepts (LIME and SHAP) that can help in making our black-box model more interpretable to the end users.
- Explore deep learning based methods.

References:

https://www.hindawi.com/journals/cin/2021/8387680/

Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning

https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#h2_3

https://www.analyticsvidhya.com/blog/2021/05/classification-algorithms-in-python-heart-attack-prediction-and-analysis/

https://towardsdatascience.com/exploratory-data-analysis-on-heart-disease-uci-data-set-ae129e47b323