VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DIGITAL ASSESSMENT – 1

TOPIC

RISK OF ANGIO ATTACK DETECTION

NAME: K. HARIHARAN

REG.NO: 20MIC0062

SUBJECT: CSI3026 - MACHINE LEARNING

FACULTY: DR. NALINI. N

DATE: 30 – MARCH - 2023

RISK OF ANGIO ATTACK DETECTION

About the Project :

Early heart attack detection refers to the identification of signs and symptoms of a heart attack in its early stages, before significant damage occurs to the heart muscle. Early detection is important because it can lead to prompt treatment, which can improve outcomes and reduce the risk of complications. Early heart attack detection may involve the recognition of symptoms, assessment of risk factors for heart disease, diagnostic testing such as an electrocardiogram (ECG) or cardiac biomarker testing, and the use of technology such as wearable devices to monitor vital signs. Overall, early heart attack detection aims to identify and treat a heart attack as early as possible to improve outcomes and prevent further damage to the heart.

Functionalities:

Early heart attack detection systems typically use various technologies and methods to identify signs and symptoms of a heart attack in its early stages. Some of the functionalities of early heart attack detection systems include:

- ✓ **Symptom Recognition:** Early heart attack detection systems can recognize symptoms of a heart attack, such as chest pain, shortness of breath, nausea, sweating, and fatigue. They may use machine learning algorithms to analyze these symptoms and predict the likelihood of a heart attack.

- ✓ **Vital Sign Monitoring:** Early heart attack detection systems can monitor vital signs, such as heart rate, blood pressure, and oxygen saturation, to identify any abnormalities that may indicate a heart attack.

- ✓ **Electrocardiogram (ECG) Monitoring:** Early heart attack detection systems may use ECG monitoring to detect changes in the electrical activity of the heart, which can indicate a heart attack.

- ✓ **Wearable Devices:** Some early heart attack detection systems use wearable devices, such as smartwatches, to monitor vital signs and detect changes that may indicate a heart attack. These devices may also use machine learning algorithms to analyze data and predict the likelihood of a heart attack.

- ✓ **Real-Time Alerts:** Early heart attack detection systems can issue real-time alerts when signs of a heart attack are detected, prompting the patient or healthcare provider to take immediate action.

- ✓ **Remote Monitoring:** Early heart attack detection systems can remotely monitor patients who are at risk of a heart attack, such as those with a history of heart disease, and alert healthcare providers if signs of a heart attack are detected.

Overall, early heart attack detection systems aim to identify signs and symptoms of a heart attack as early as possible, allowing healthcare providers to intervene and prevent serious complications. By using a combination of symptom recognition, vital sign monitoring, ECG monitoring, wearable devices, real-time alerts, and remote monitoring, these systems can improve the chances of early detection and treatment of heart attacks.

Early detection of heart attack is critical for timely intervention and prevention of severe damage to the heart muscle. Machine learning can be used as a powerful tool to detect early signs of heart attack by analyzing various patient data.

There are several approaches to using machine learning for early heart attack detection

1) Based on ECG [Electrocardiogram] Analysis
2) Blood biomarker analysis
3) Patient risk factor analysis

But in this Project, I will detect the heart attack based on the Patient risk factor analysis.

Patient Risk Factor Analysis:

Patient risk factor analysis is an important tool for identifying individuals who are at increased risk of developing a heart attack.

Some of the common risk factors for heart attack include:

- **Age:** As people get older, their risk of having a heart attack increases.
- **Family history:** Individuals who have a family history of heart disease are at increased risk of having a heart attack.
- **Smoking:** Smoking damages the blood vessels and increases the risk of heart attack.
- High blood pressure: High blood pressure puts extra strain on the heart and increases the risk of heart attack.
- **High cholesterol:** High levels of cholesterol in the blood can lead to a build-up of plaque in the arteries, which increases the risk of heart attack.
- **Obesity:** Being overweight or obese increases the risk of developing several health conditions, including heart disease.
- **Diabetes:** Diabetes can damage the blood vessels and increase the risk of heart attack.
- **Physical inactivity:** Lack of physical activity can lead to obesity and increase the risk of heart attack.

By considering this data of an individual, Machine learning algorithms can be trained to analyze patient data, such as age, sex, medical history, lifestyle habits, and other risk factors, to predict the likelihood of developing heart disease or having a heart attack. These algorithms can be used to identify individuals who are at increased risk of heart attack and guide clinical decision-making.

Observation on this Project:

1) The dataset using here for the detection purpose is that, “Heart.csv” which included the data’s of an individual related to the health(heart) of an person such as cholesterol, thalach (max heart rate achieved), trestbps (Resting Blood Pressure), etc…,

2) By using describe() function, They found the important values such as mean, count, minmax, std etc...,

3) Then in Feature Engineering, They dropped the constant and duplicate data’s using DropConstantFeatures and DropDuplicateFeatures libraries.

4) By using drop() function, they Removed 3 features which are not which are not that important in detecting the heart attack.

5) StandardScaler removes the mean and scales the data to the unit variance. It Standardizes the data. They make the data to be standardized by using StandardScaler function from sklearn.preprocessing library.

6) Now, They have done Model Building of 4 Different Classifier Algorithms which included Logistic Regression, Decision Tree Classifier, Random Forest Classifier, KNN Classifier.

7) Every Classifier’s Output shows the Test Classification Report (includes precision, recall, f1-score, support and macro average , weighted average) with Confusion Matrix and their Accuracy.

8) **Accuracy of Different Classifier’s are as follows:**

Logistic Regression	0.8524590163934426
Decision Tree Classifier	0.7704918032786885
Random Forest	0.8524590163934426
KNN Classifier	0.639344262295082

Possible Additional Functionalities:

1. To Determine the Total Percentage of person with heart disease attack in the dataset which is differentiated by the low risk and high risk towards the heart attack.

2. Visualizing the Pearson's Correlation

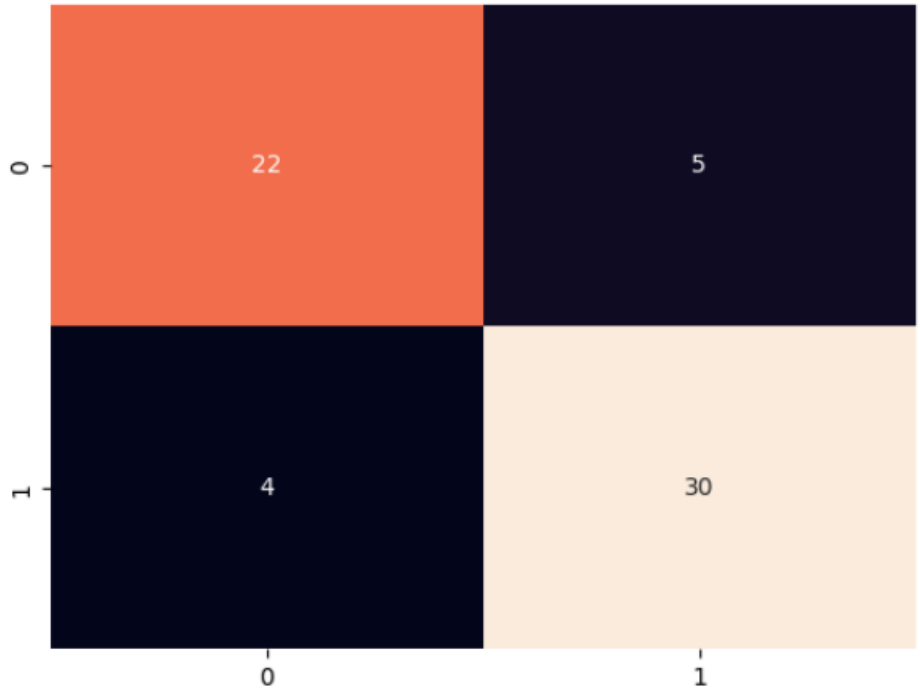
3. Plotting the risk factors of heart attack in Histogram

MAIN FUNCTIONALITY OF THE CODE –

1) Logistic Regression

```
LR_model.fit(xtrain, ytrain)

LR_ypred = LR_model.predict(xtest)

print("Test Accuracy", accuracy_score(ytest, LR_ypred))

print("Test Classification Report\n", classification_report(ytest,LR_ypred))

mat=confusion_matrix(ytest,LR_ypred)

sns.heatmap(mat,annot=True, cbar=False)

plt.show()
```

```
Test Accuracy 0.8524590163934426
Test Classification Report
              precision    recall  f1-score   support

           0       0.85      0.81      0.83        27
           1       0.86      0.88      0.87        34

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61
```

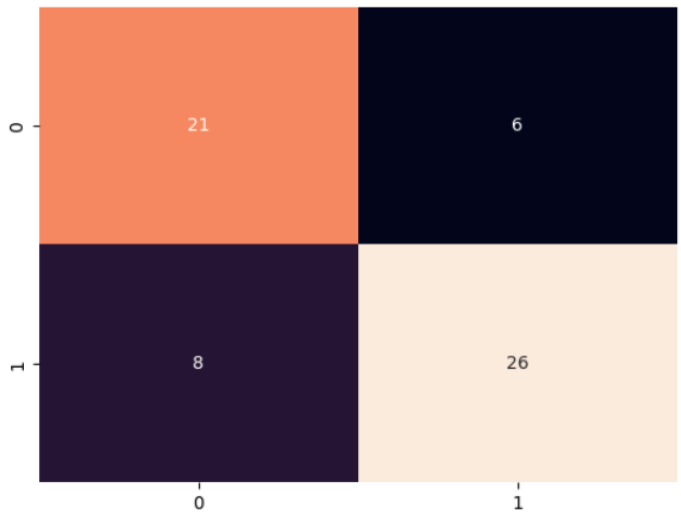


2) Decision Tree

```
from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn import svm

LR_model = LogisticRegression(fit_intercept=True)

DT_model=DecisionTreeClassifier(max_depth=5, random_state=1)
```

```
RFC_model = RandomForestClassifier(n_estimators = 100)

KNN_model = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

SVM_model = svm.SVC()

DT_model.fit(xtrain, ytrain)

DT_ypred = DT_model.predict(xtest)

print("Test Accuracy", accuracy_score(ytest, DT_ypred))

print("Test Classification Report\n", classification_report(ytest,DT_ypred))

mat=confusion_matrix(ytest,DT_ypred)

sns.heatmap(mat,annot=True, cbar=False)

plt.show()
```

```
Test Accuracy 0.7704918032786885
Test Classification Report
              precision    recall  f1-score   support

           0       0.72      0.78      0.75        27
           1       0.81      0.76      0.79        34

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

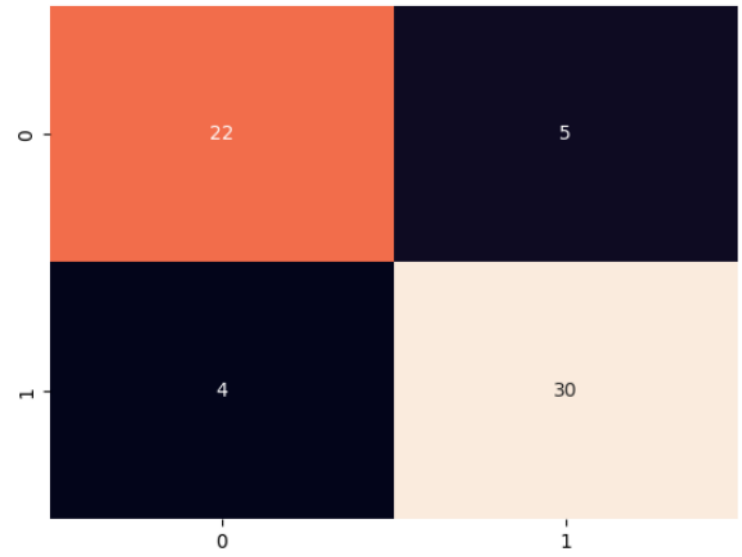


3) Random Forest

```
RFC_model.fit(xtrain, ytrain)
```

```
RFC_ypred = RFC_model.predict(xtest)

print("Test Accuracy", accuracy_score(ytest, RFC_ypred))

print("Test Classification Report\n", classification_report(ytest,RFC_ypred))

mat=confusion_matrix(ytest,RFC_ypred)

sns.heatmap(mat,annot=True, cbar=False)

plt.show()
```

```
Test Accuracy 0.8524590163934426
Test Classification Report
              precision    recall  f1-score   support

           0       0.85      0.81      0.83        27
           1       0.86      0.88      0.87        34

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61
```

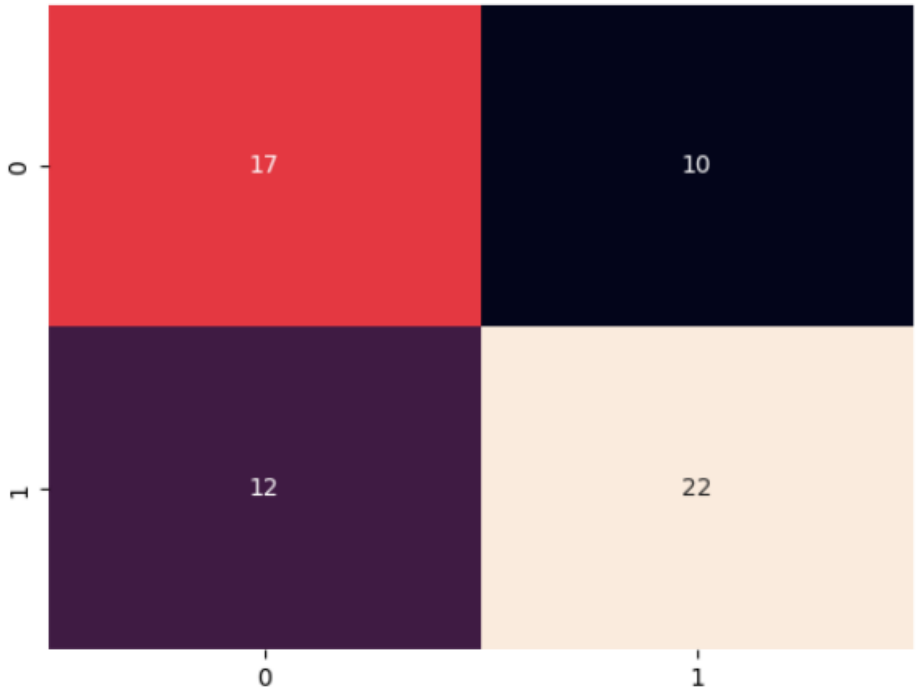


4) KNN

```
KNN_model.fit(xtrain, ytrain)

KNN_ypred = KNN_model.predict(xtest)

print("Test Accuracy", accuracy_score(ytest, KNN_ypred))

print("Test Classification Report\n", classification_report(ytest,KNN_ypred))
```

```
mat=confusion_matrix(ytest,KNN_ypred)

sns.heatmap(mat,annot=True, cbar=False)

plt.show()
```

```
Test Accuracy 0.639344262295082
Test Classification Report
              precision    recall  f1-score   support

           0       0.59      0.63      0.61        27
           1       0.69      0.65      0.67        34

    accuracy                           0.64        61
   macro avg       0.64      0.64      0.64        61
weighted avg       0.64      0.64      0.64        61
```



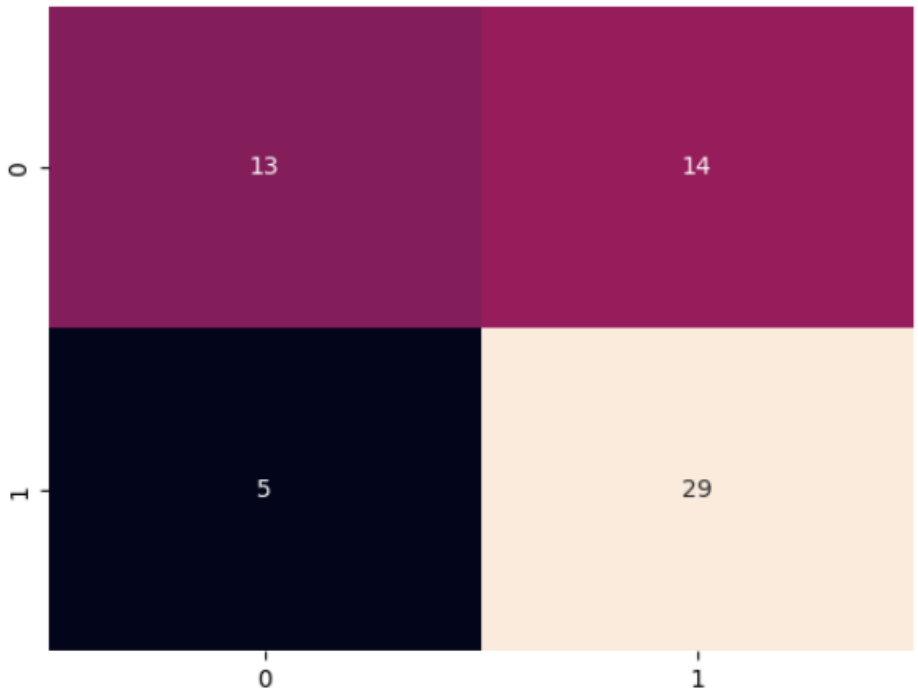
5) SVM

```
SVM_model.fit(xtrain, ytrain)

SVM_ypred = SVM_model.predict(xtest)

print("Test Accuracy", accuracy_score(ytest, SVM_ypred))

print("Test Classification Report\n", classification_report(ytest,SVM_ypred))

mat=confusion_matrix(ytest,SVM_ypred)
```

sns.heatmap(mat,annot=True, cbar=False)

plt.show()

```
Test Accuracy 0.6885245901639344
Test Classification Report
              precision    recall  f1-score   support

           0       0.72      0.48      0.58        27
           1       0.67      0.85      0.75        34

    accuracy                           0.69        61
   macro avg       0.70      0.67      0.67        61
weighted avg       0.70      0.69      0.68        61
```



ADDITIONAL FUNCTIONALITIES I HAVE ADDED

1) Pearson Correlation

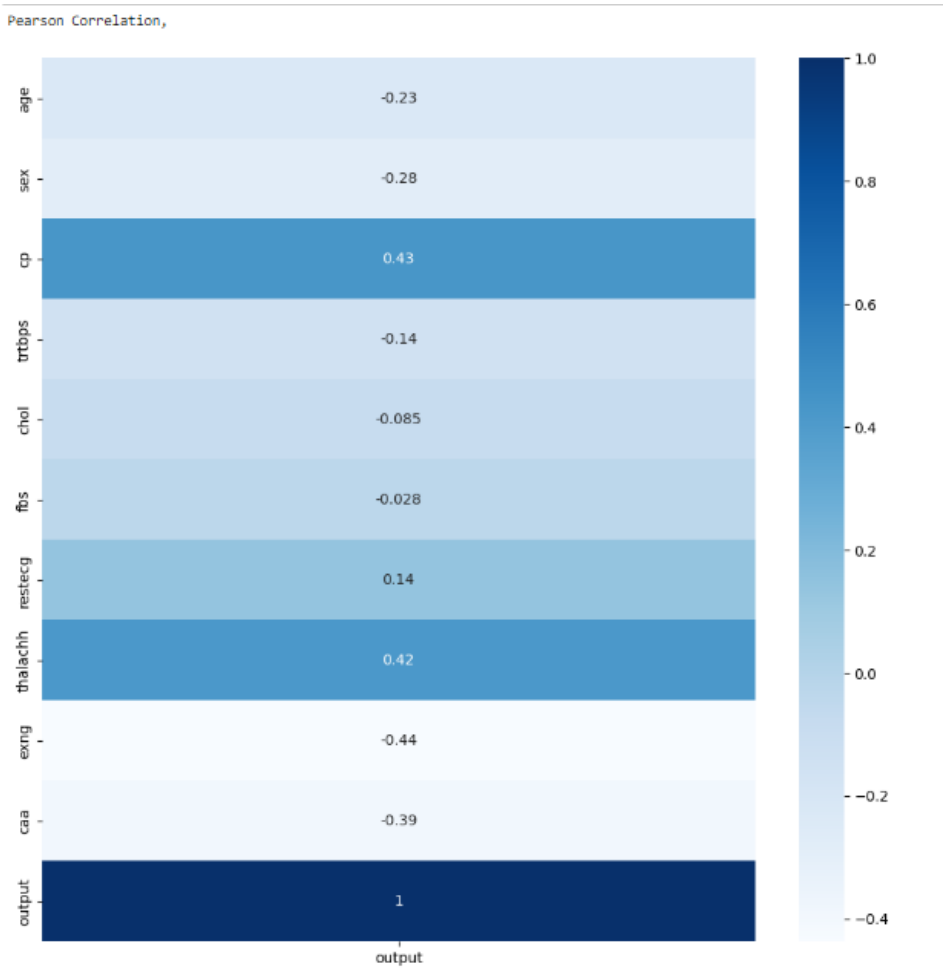
print('Pearson Correlation,')

plt.figure(figsize = (11,11))

cor = data.corr().iloc[:,-1:]

sns.heatmap(cor, annot = True, cmap = plt.cm.Blues)

```
plt.show()

print('abs corr score: ')

print(abs(cor['output'][0:-1]))

cor['output'] = cor['output'][0:-1]

margin = abs(cor['output'][0:-1]).mean()

print('\n')

print('mean {0}'.format(margin))

print('\n')

print('feature selection result: ')

fs = abs(cor['output'][0:-1])[abs(cor['output']) > margin]

print(fs)
```

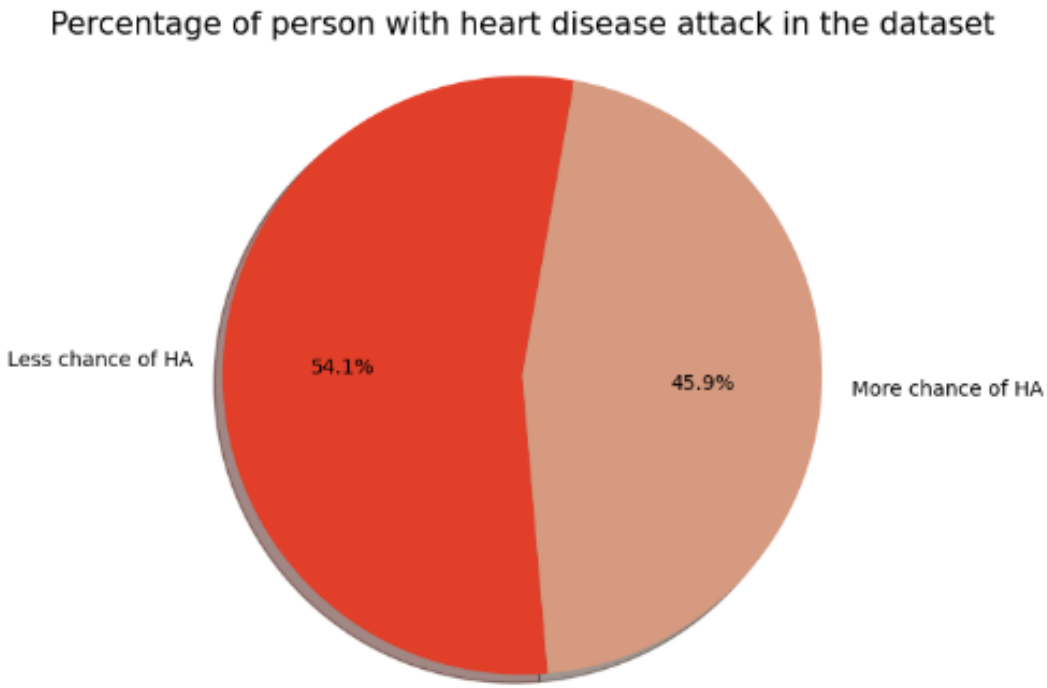


```
abs corr score:
age         0.225439
sex         0.280937
cp          0.433798
trtbps      0.144931
chol        0.085239
fbs         0.028046
restecg     0.137230
thalachh    0.421741
exng        0.436757
caa         0.391724
Name: output, dtype: float64


mean 0.25858410591805364


feature selection result:
sex         0.280937
cp          0.433798
thalachh    0.421741
exng        0.436757
caa         0.391724
Name: output, dtype: float64
```

2) Percentage of person with the Heart/Cardiac Attack in the dataset

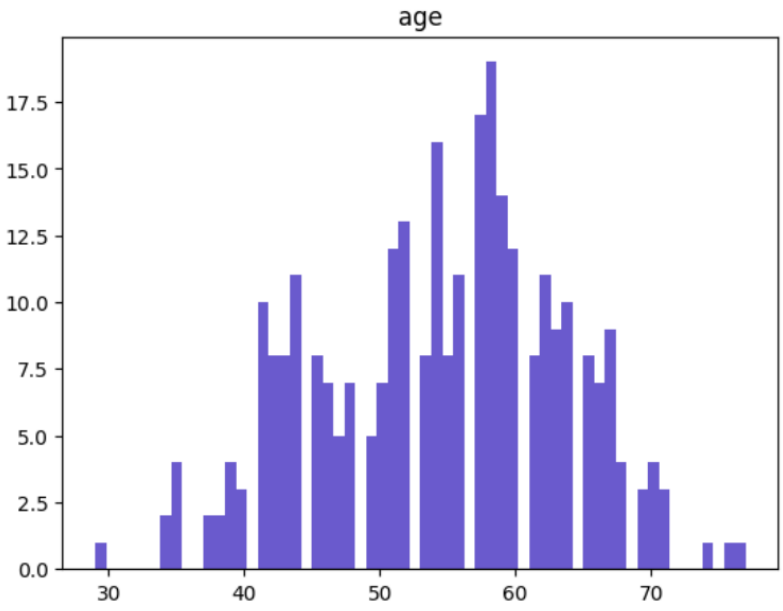
```
data['output'] = data.target.map({0: 0, 1: 1, 2: 1, 3: 1, 4: 1})
# data['sex'] = data.sex.map({0: 'female', 1: 'male'})
data['thalachh'] = data.thal.fillna(data.thal.mean())
data['caa'] = data.ca.fillna(data.ca.mean())
df = data.copy()
ot = {0: "Less chance of HA",1:'More chance of HA'}
df.output = [ot[item] for item in df.output] heart=Counter(df['output'])
classes=[]
count=[]   #list to store no of laels of each class
for i in heart.keys():
    classes.append(i)
    count.append(heart[i])
colors = ["#E13F29", "#D69A80"]

plt.pie(
    count,
    labels = classes,
    shadow = True,
    colors = colors,
    startangle=80,
    autopct='%1.1f%%'
)
plt.axis('equal')
plt.tight_layout()
plt.title("Percentage of person with heart disease attack in the dataset", fontsize=15)
plt.show()
# fig = px.pie(df, names='output', title='Percentage of person with heart disease present in the
dataset')
```

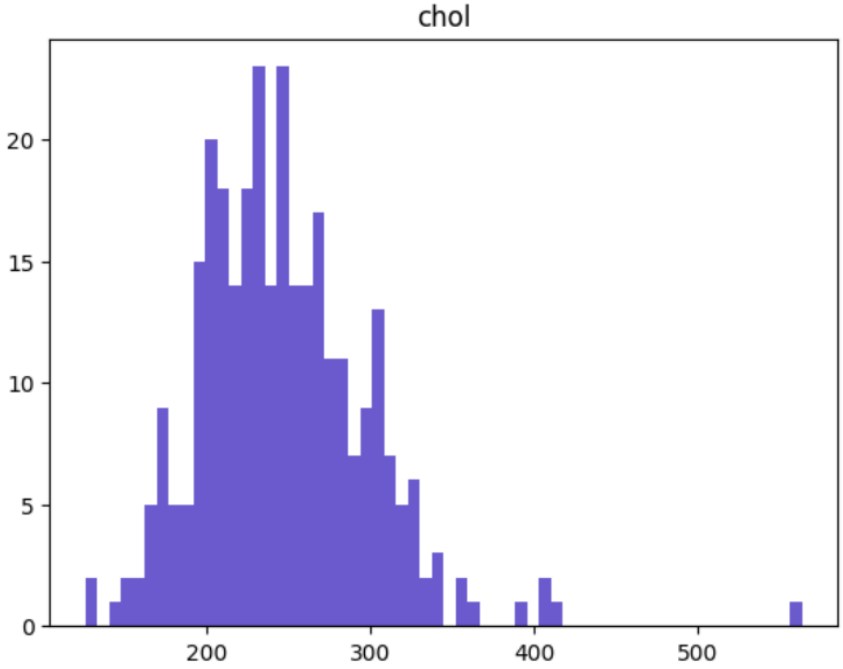
fig.show()

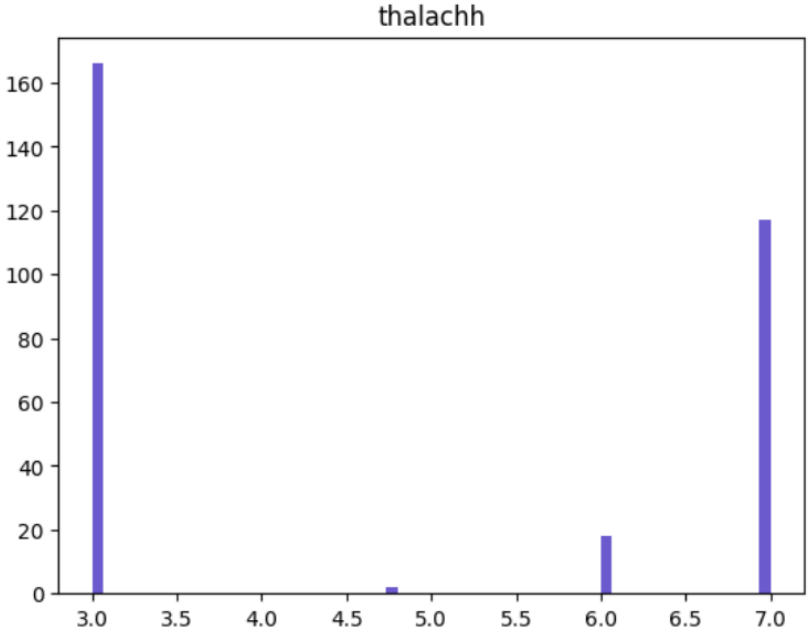


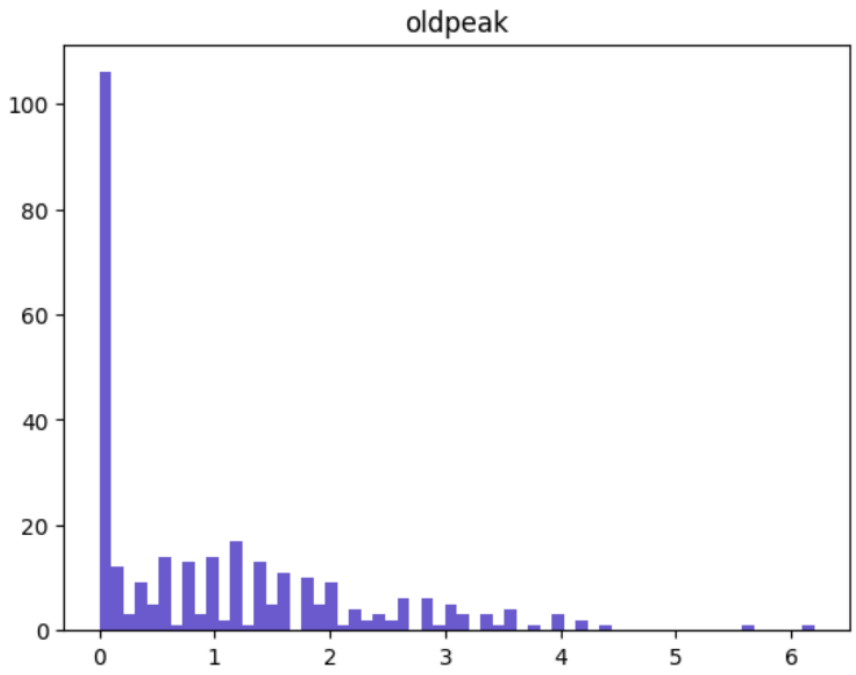
3) Risk factors of heart attack in Histogram

```
i = ['age', 'chol','thalachh','oldpeak']
for j in i:
    plt.hist(df[j],bins = 60, color = 'slateblue')
    plt.title(j)
    plt.show()
```




chol
0
5
10
15
20
200
300
400
500


thalachh
0
20
40
60
80
100
120
140
160
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0



✓ **GITHUB LINK OF THE PROJECT –**

https://github.com/tdishant/Heart-attack-risk-prediction

✓ **GITHUB LINK OF MY PROJECT –**

https://github.com/Hariharan663/RISK-OF-ANGIO-ATTACK-DETECTION

THANK YOU !