In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docke
r-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list al
l files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets p
reserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved out
side of the current session
```

Heart Attack EDA and Prediction

Importing Libraries

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Lasso, Ridge
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")
```

Read The Dataset

In [3]:

```
heart=pd.read_csv("../input/heart-attack-analysis-prediction-dataset/heart.csv")
saturation=pd.read_csv("../input/heart-attack-analysis-prediction-dataset/o2Satu
ration.csv")
```

In [4]:

```
heart.head()
```

Out[4]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

In [5]:

```
saturation.head()
```

Out[5]:

	98.6
0	98.6
1	98.6
2	98.6
3	98.1
4	97.5

Data Description

- **age - Age of the patient**
- **sex - Sex of the patient**
- **cp - Chest pain type ~ 0 = Typical Angina, 1 = Atypical Angina, 2 = Non-anginal Pain, 3 = Asymptomatic**
- **trtbps - Resting blood pressure (in mm Hg)**
- **chol - Cholestoral in mg/dl fetched via BMI sensor**
- **fbs - (fasting blood sugar > 120 mg/dl) ~ 1 = True, 0 = False**
- **restecg - Resting electrocardiographic results ~ 0 = Normal, 1 = ST-T wave normality, 2 = Left ventricular hypertrophy**
- **thalachh - Maximum heart rate achieved**
- **oldpeak - Previous peak**
- **slp - Slope**
- **caa - Number of major vessels**
- **thall - Thalium Stress Test result ~ (0,3)**
- **exng - Exercise induced angina ~ 1 = Yes, 0 = No**
- **output - target : 0= less chance of heart attack 1= more chance of heart attack**

In [6]:

```
heart.shape
```

Out[6]:

In [7]:

```
col=heart.columns
col
```

Out[7]:

In [8]:

```
heart.describe()
```

Out[8]:

	age	sex	cp	trtbps	chol	fbs
count	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
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000

In [9]:

```
heart.nunique()
```

Out[9]:

In [10]:

```
heart
```

Out[10]:

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

303 rows × 14 columns

In [11]:

```
categorical_cols = ['sex','exng','caa','cp','fbs','restecg','slp','thall']
numerical_cols = ["age","trtbps","chol","thalachh","oldpeak"]
target_col = ["output"]
```

In [12]:

```
# check for the missing values
heart.isnull().sum()
```

Out[12]:

No missing values are found

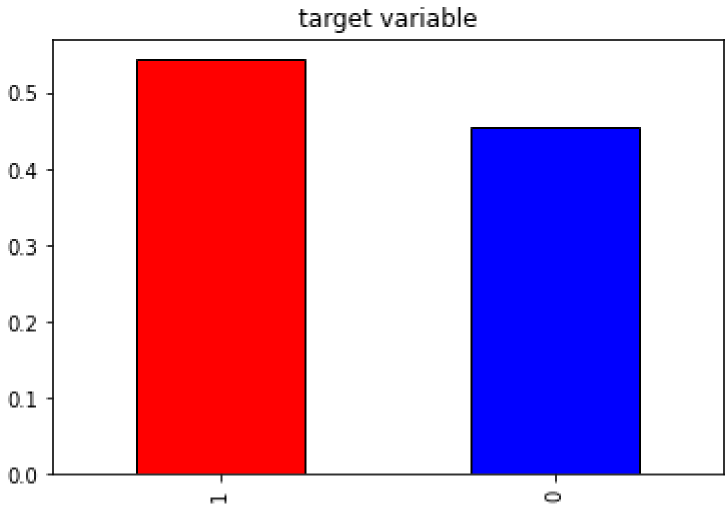
Univariate Analysis

Categorical and Target features

In [13]:

```
# target variable
heart['output'].value_counts(normalize=True).plot.bar(color=['red','blue'],edgecolor='black',title='target variable')
```

Out[13]:



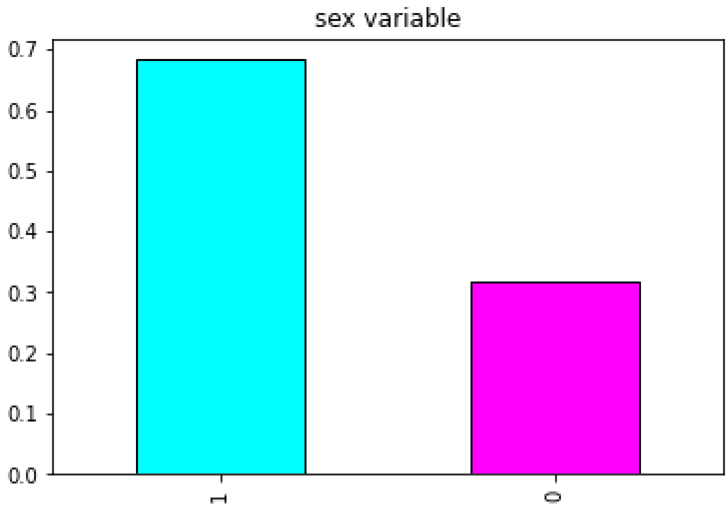
- **Around 55% people have more chances to get heart attack**
- **Around 45% people have less chances to get heart attack**

Sex Feature

In [14]:

```
# sex variable
heart['sex'].value_counts(normalize=True).plot.bar(color=['cyan','magenta'],edge
color='black',title='sex variable')
```

Out[14]:



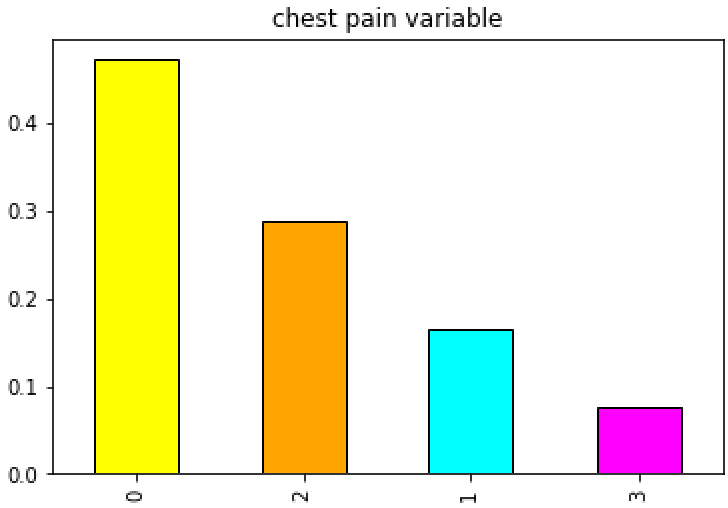
- **Around 68 % people are with sex=1**
- **Around 30 % people are with sex=0**

Chest Pain Feature

In [15]:

```
# cp variable
heart['cp'].value_counts(normalize=True).plot.bar(color=['yellow','orange','cyan','magenta'],edgecolor='black',title='chest pain variable')
```

Out[15]:



- **Around 50 % of the people have chest pain type: Typical Angina**
- **Around 28 % of the people have chest pain type: Non-anginal Pain**
- **Around less than 20 % of the people have chest pain type: Atypical Angina**
- **Around less than 10% of the people have chest pain type: Asymptomatic**

1.exercise induced angina

2.fasting blood sugar > 120 mg/dl

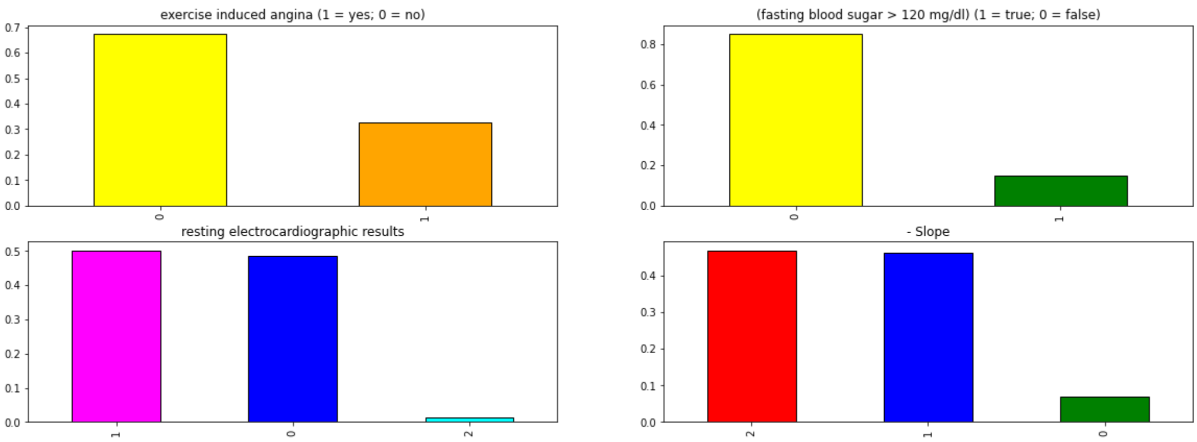
3.resting electrocardiographic results

4. Slope

In [16]:

```
plt.figure(figsize=(20,7))
plt.subplot(221)
heart['exng'].value_counts(normalize=True).plot.bar(color=['yellow','orange'],edgecolor='black',title='exercise induced angina (1 = yes; 0 = no)')
plt.subplot(222)
heart['fbs'].value_counts(normalize=True).plot.bar(color=['yellow','green'],edgecolor='black',title='(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)')
plt.subplot(223)
heart['restecg'].value_counts(normalize=True).plot.bar(color=['magenta','blue','cyan'],edgecolor='black',title='resting electrocardiographic results')
plt.subplot(224)
heart['slp'].value_counts(normalize=True).plot.bar(color=['red','blue','green'],edgecolor='black',title='- Slope')
```

Out[16]:



- More than 65 % of the people Exercise don't induced angina
- More than 35 % of the people Exercise induced angina
- less than 20 % of the people have fasting blood sugar > 120 mg/dl
- More than 80 % of the people have fasting blood sugar <= 120 mg/dl
- less than 50 % of the people have resting electrocardiographic results normal
- 50 % of the people have resting electrocardiographic results: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
- 1% or 2% of the people have resting electrocardiographic results: showing probable or definite left ventricular hypertrophy by Estes' criteria

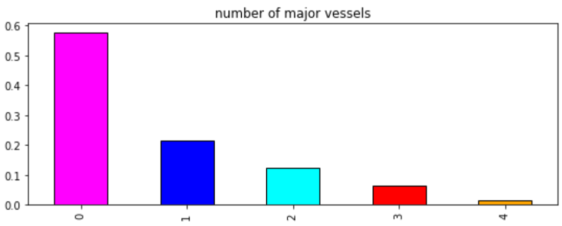
1.number of major vessels

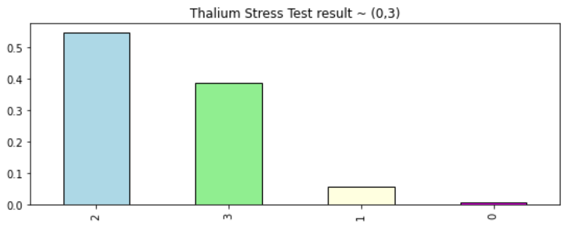
2.Thalium Stress Test result ~ (0,3)

In [17]:

```
plt.figure(figsize=(20,7))
plt.subplot(221)
heart['caa'].value_counts(normalize=True).plot.bar(color=['magenta','blue','cyan','red','orange'],edgecolor='black',title='number of major vessels')
plt.subplot(222)
heart['thall'].value_counts(normalize=True).plot.bar(color=['lightblue','lightgreen','lightyellow','magenta'],edgecolor='black',title='Thalium Stress Test result ~ (0,3)')
```

Out[17]:





Numerical features

In [18]:

```
numerical_cols
```

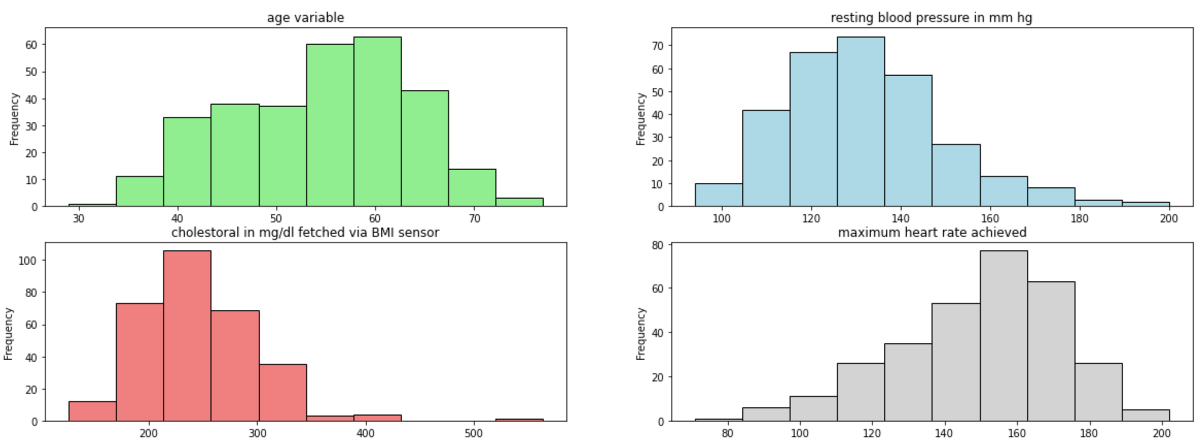
Out[18]:

age , blood pressure , cholestoral , Heart Rate

In [19]:

```
plt.figure(figsize=(20,7))
plt.subplot(221)
heart['age'].plot.hist(edgecolor='black',color='lightgreen',title='age variable')
plt.subplot(222)
heart['trtbps'].plot.hist(edgecolor='black',color='lightblue',title='resting blood pressure in mm hg')
plt.subplot(223)
heart['chol'].plot.hist(edgecolor='black',color='lightcoral',title='cholestoral in mg/dl fetched via BMI sensor')
plt.subplot(224)
heart['thalachh'].plot.hist(edgecolor='black',color='lightgray',title='maximum heart rate achieved')
```

Out[19]:

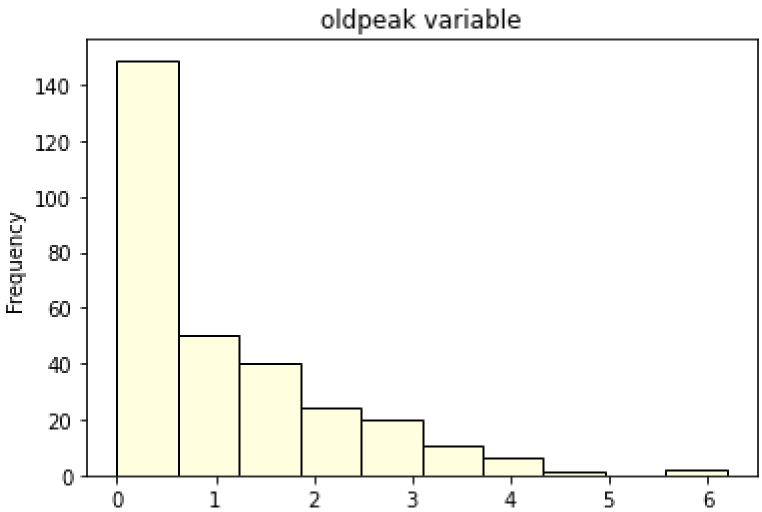


Oldpeak

In [20]:

```
heart['oldpeak'].plot.hist(edgecolor='black',color='lightyellow',title='oldpeak
 variable')
```

Out[20]:



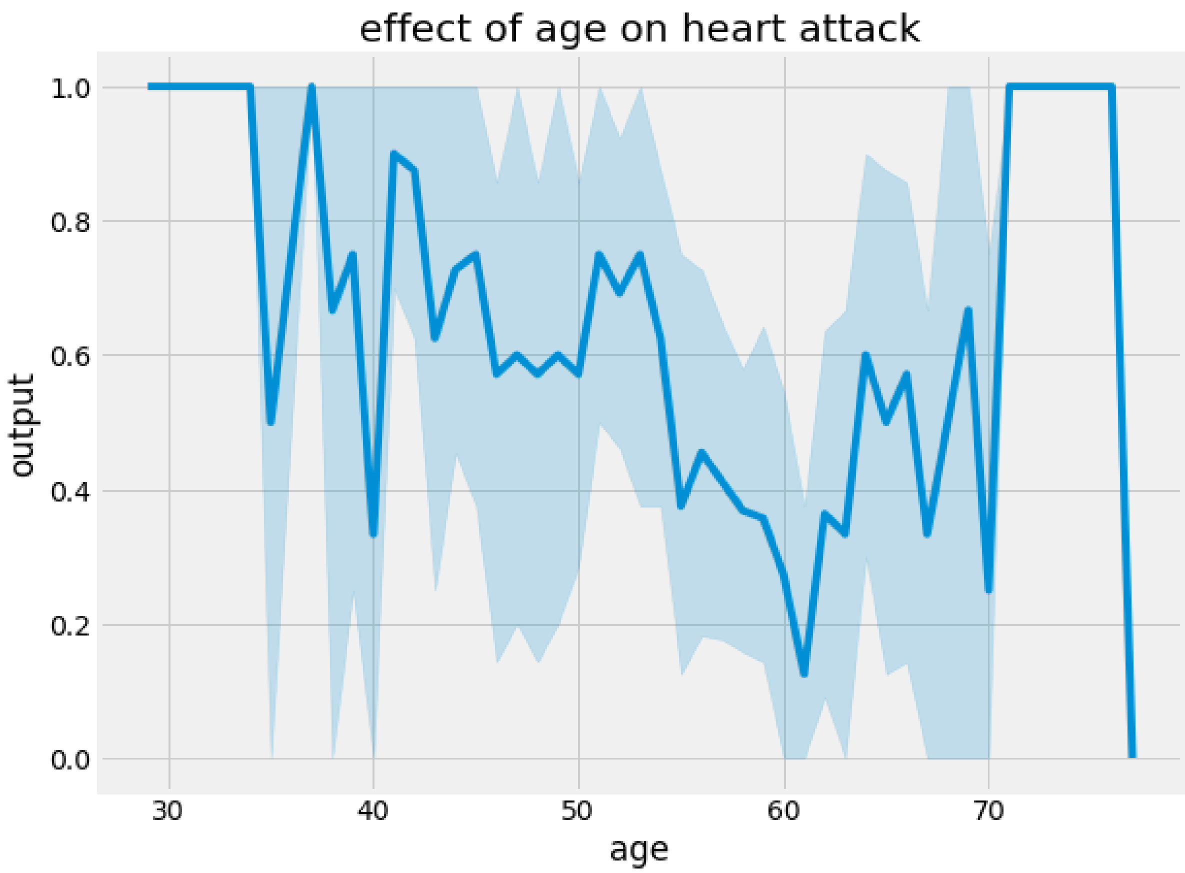
Bivariate Analysis

effect of age on heart attack

In [21]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of age on heart attack")
sns.lineplot(x=heart['age'],y=heart['output'])
```

Out[21]:



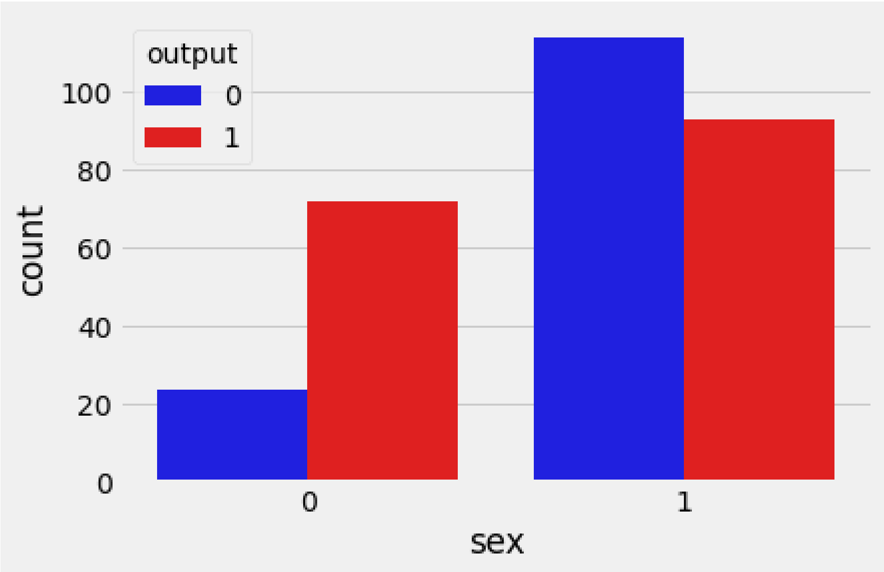
- **The people with the age 30 to 35 have higher chance of heart attacks**
- **The people with the age than 70 and less than 75 have higher chance of heart attacks**
- **apart from it no certain trend i will be able to find**

heart attack related with sex

In [22]:

```
sns.countplot(data=heart,x='sex',palette=["blue","red"], hue='output')
```

Out[22]:



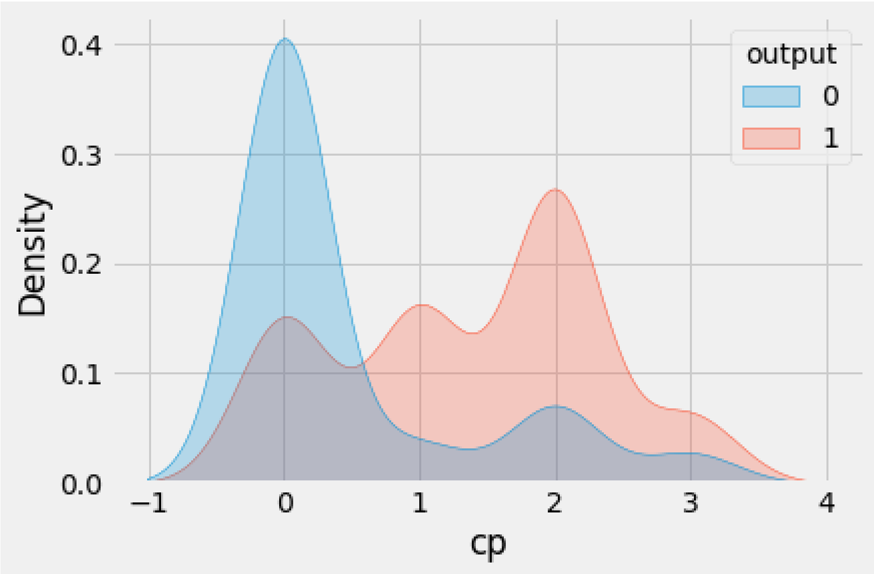
people of sex=1 have higher chances of getting heart attacks

heart attack related with chest pain

In [23]:

```
sns.kdeplot(data=heart, x='cp',hue="output", fill=True)
```

Out[23]:



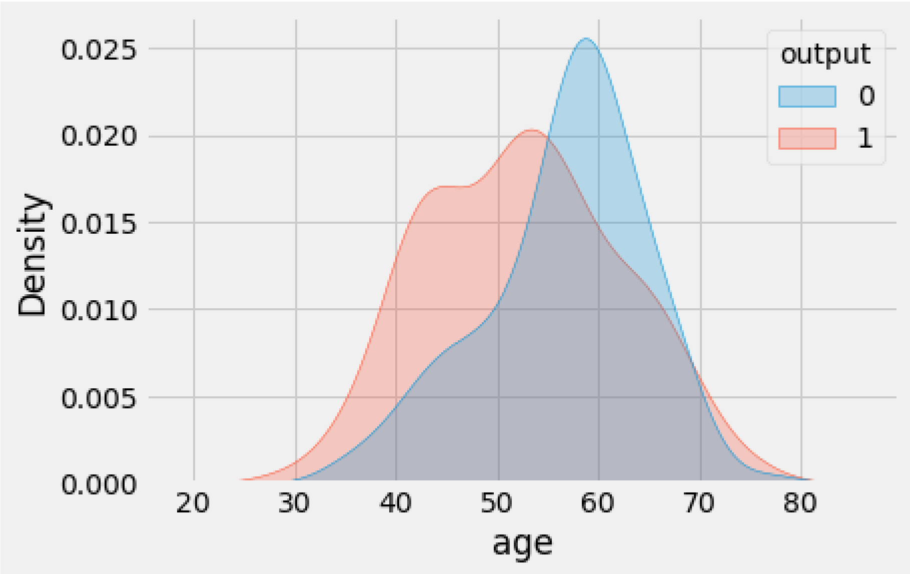
people with chest pain type=2 have higher chance of getting heart attacks

heart attack related with age

In [24]:

```
sns.kdeplot(data=heart, x='age',hue="output", fill=True)
```

Out[24]:



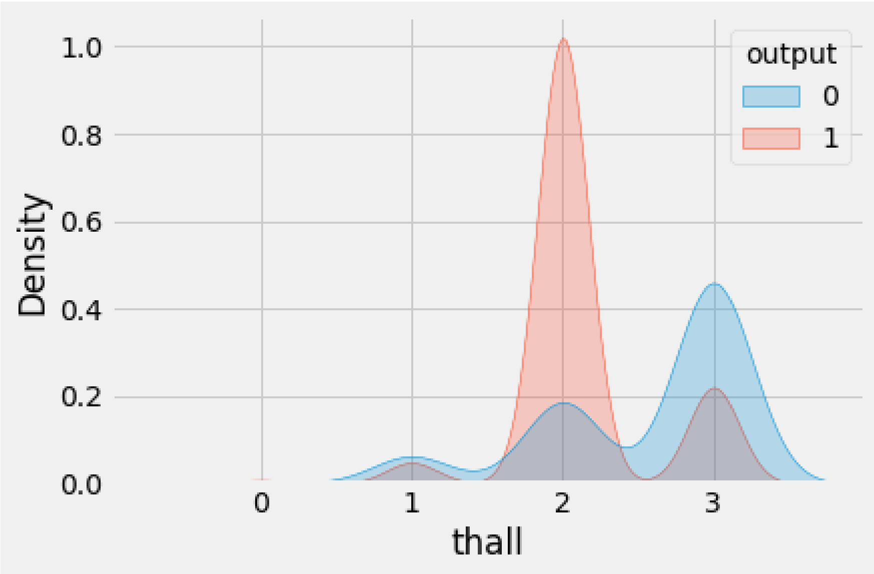
according to the data people with lower age have more chances of getting heart attacks than those of higher ages

heat attack realted with thalium stress test

In [25]:

```
sns.kdeplot(data=heart, x='thall',hue="output", fill=True)
```

Out[25]:



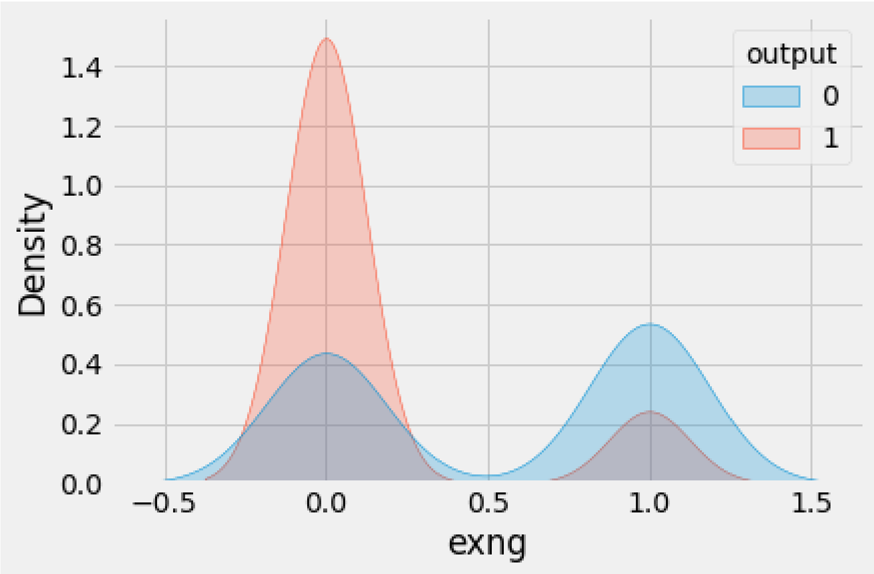
people with thall test=2 have higher chance of getting heart attacks

heat attack realted with Exercise induced angina

In [26]:

```
sns.kdeplot(data=heart, x='exng',hue="output", fill=True)
```

Out[26]:



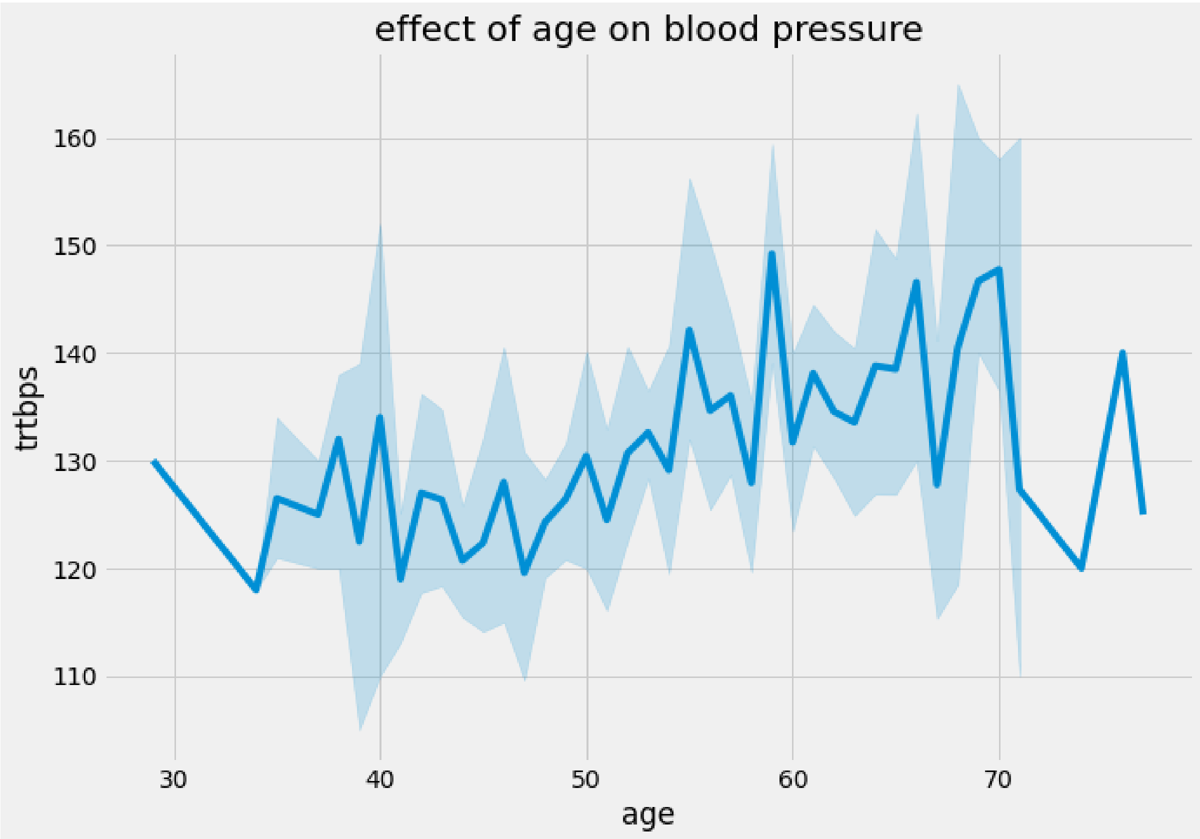
people with exng=0 have higher chances of getting heart attacks

effect of age on blood pressure

In [27]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of age on blood pressure")
sns.lineplot(x=heart['age'],y=heart['trtbps'])
```

Out[27]:



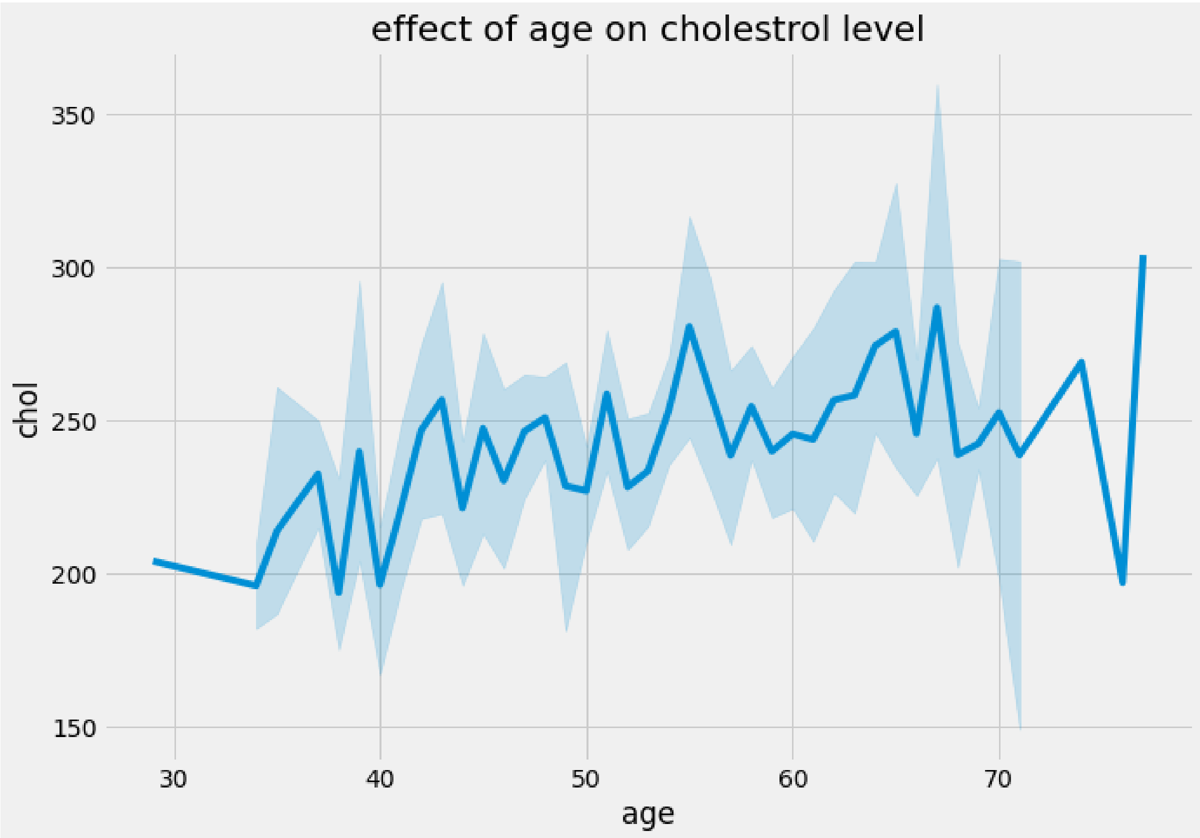
- as age is incresing the increase in the blood pressure has been founded

effect of age on cholestrol level

In [28]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of age on cholestrol level")
sns.lineplot(x=heart['age'],y=heart['chol'])
```

Out[28]:



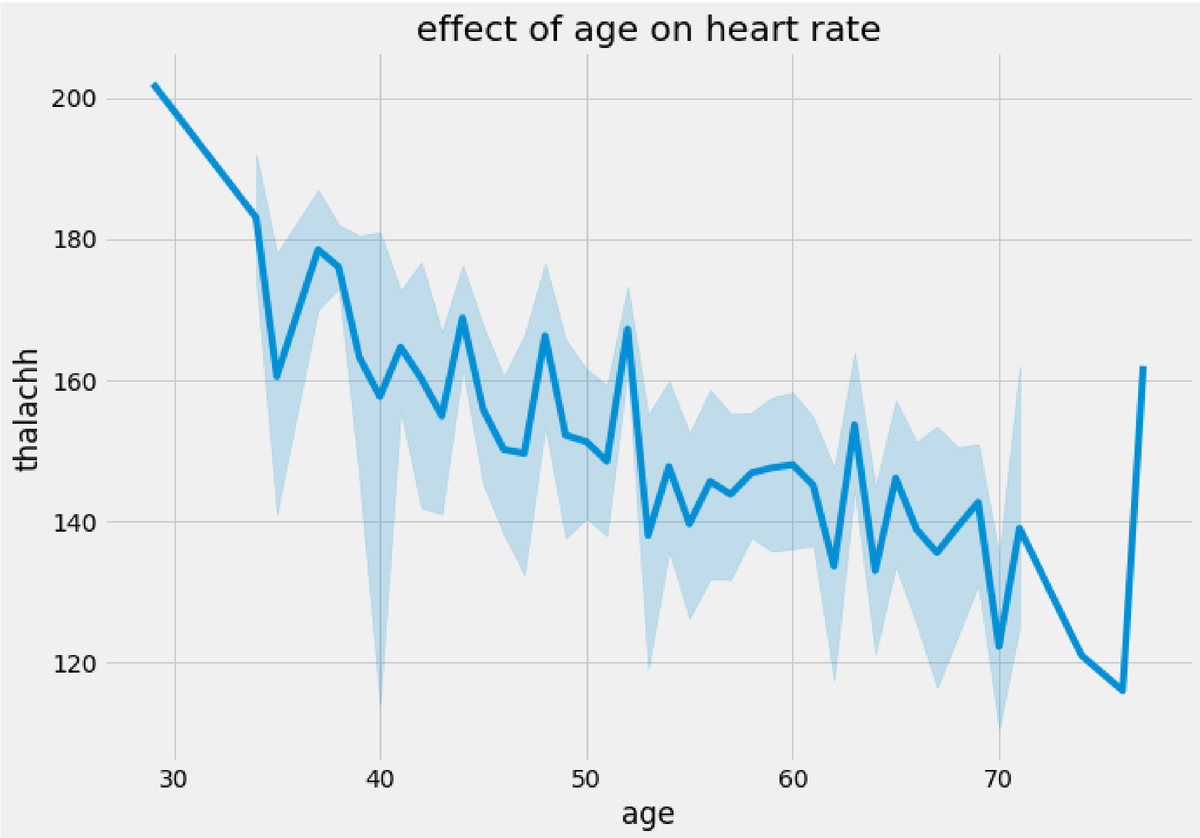
- **as age is incresing the increase in the cholestrol level has been founded**

effect of age on heart rate

In [29]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of age on heart rate")
sns.lineplot(x=heart['age'],y=heart['thalachh'])
```

Out[29]:



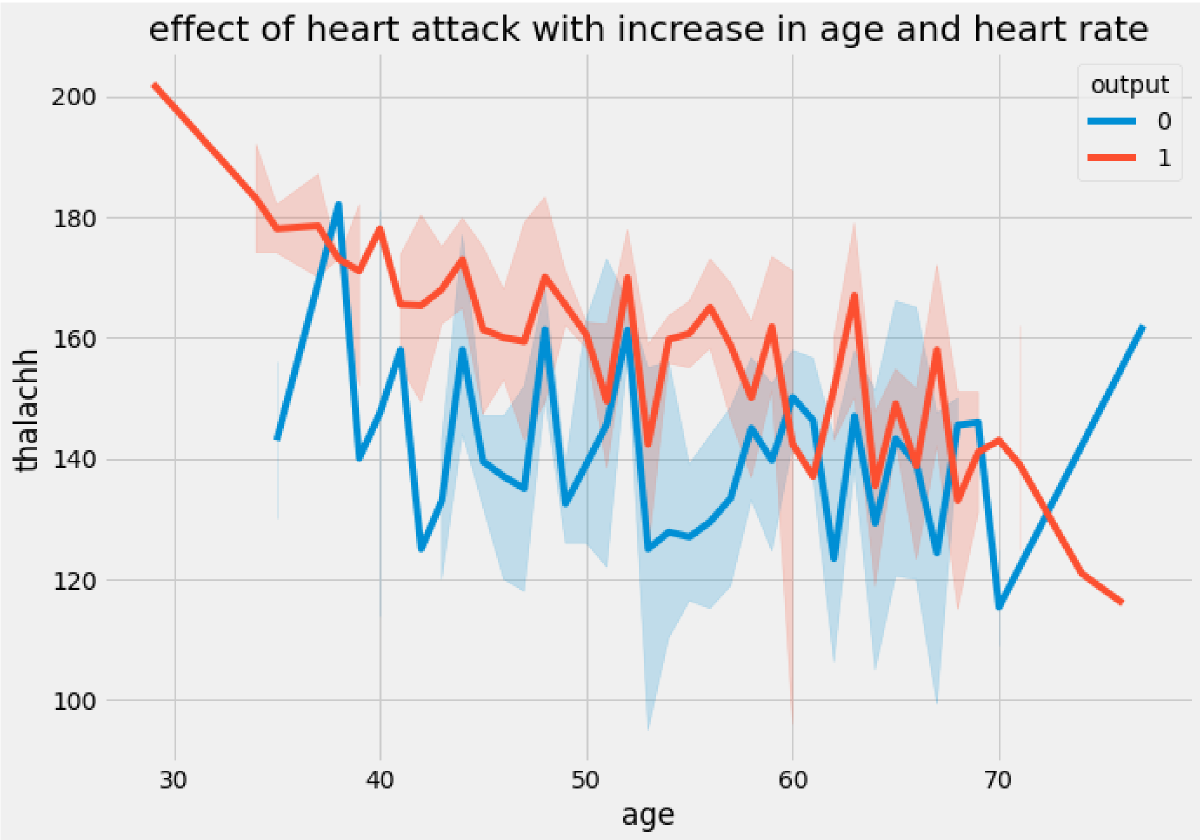
- **as age is incresing the decrease in the heart rate has been founded**

How does incresed heart rate and age affect the heart attack

In [30]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of heart attack with increase in age and heart rate")
sns.lineplot(x=heart['age'],y=heart['thalachh'],hue=heart['output'])
```

Out[30]:

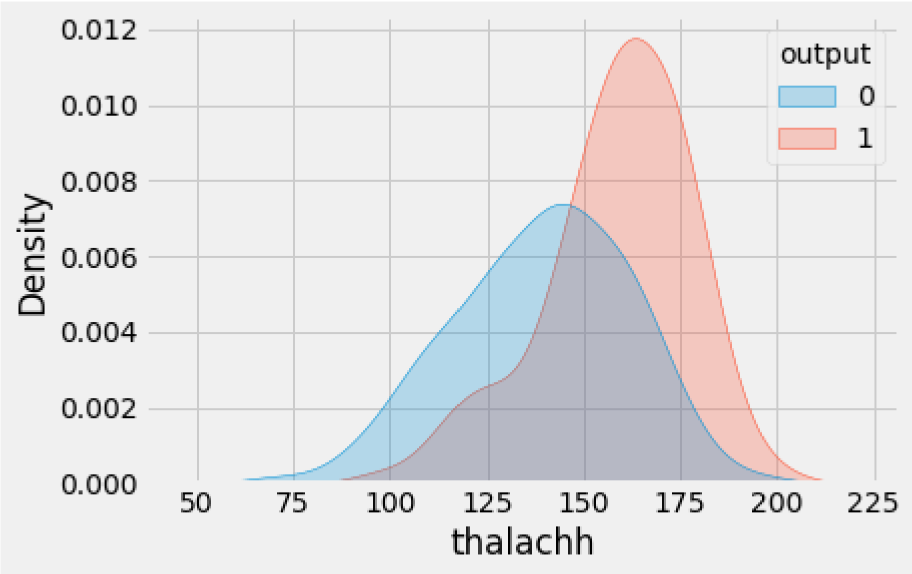


- **as with the increase in the age the heart rate is decresing and also the people with more chances of heart attacks are decreasing hence we can say higher heart rate increases the chance of heart attack**

In [31]:

```
sns.kdeplot( data=heart, x='thalachh',hue="output",fill=True)
```

Out[31]:

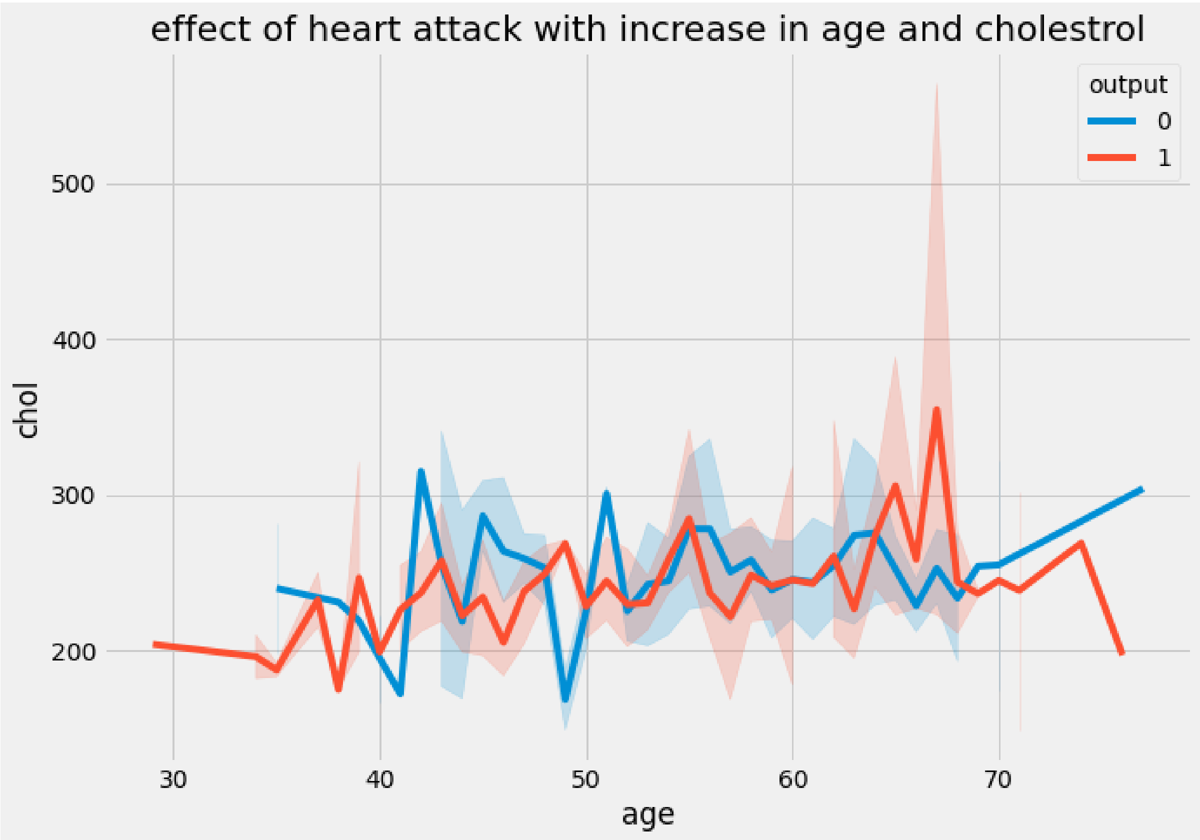


How does incresed cholestrol and age affect the heart attack

In [32]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of heart attack with increase in age and cholestrol")
sns.lineplot(x=heart['age'],y=heart['chol'],hue=heart['output'])
```

Out[32]:

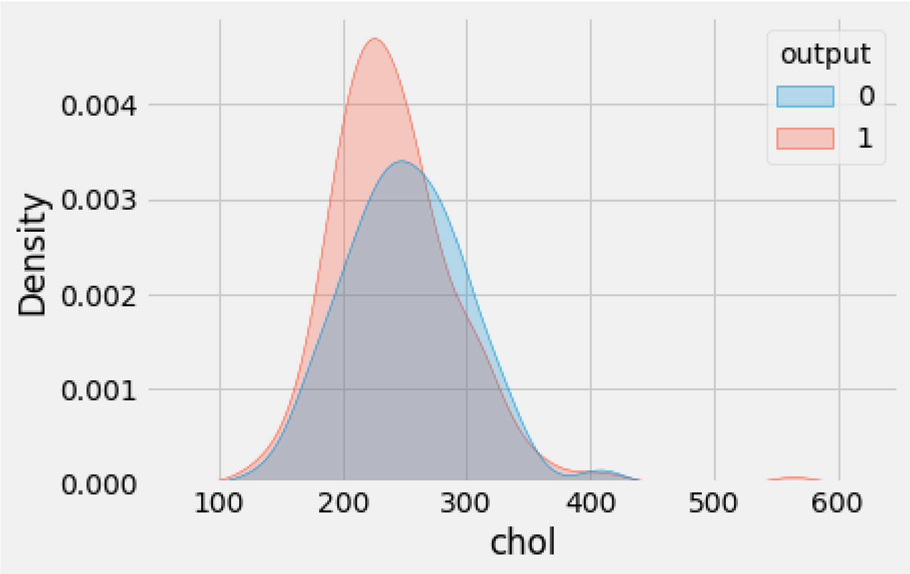


- **as with the increase in the age the cholestrol level is incresing and also the people with more chances of heart attacks are also increasing hence we can say higher cholestrol level increases the chance of heart attack**

In [33]:

```
sns.kdeplot( data=heart, x='chol',hue="output",fill=True)
```

Out[33]:

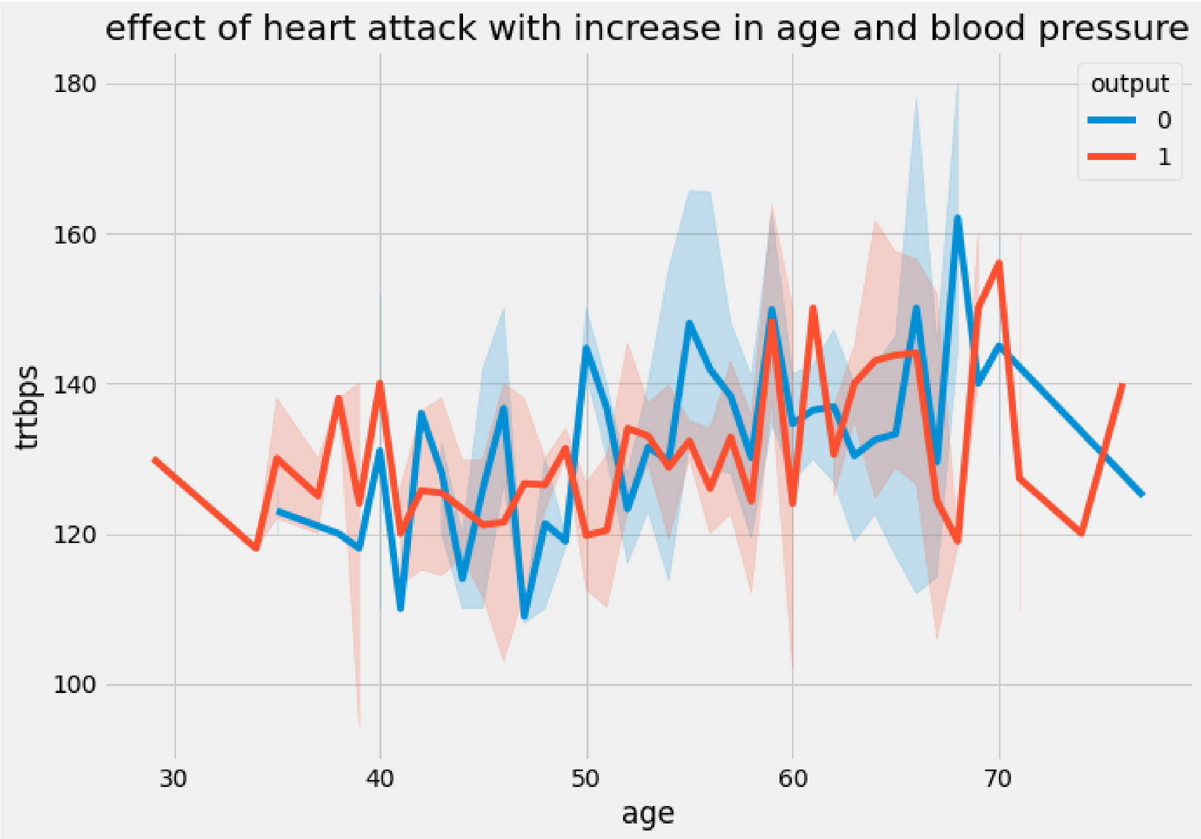


How does incresed blood pressure and age affect the heart attack

In [34]:

```
plt.figure(figsize=(10,7))
plt.style.use("fivethirtyeight")
plt.title("effect of heart attack with increase in age and blood pressure")
sns.lineplot(x=heart['age'],y=heart['trtbps'],hue=heart['output'])
```

Out[34]:

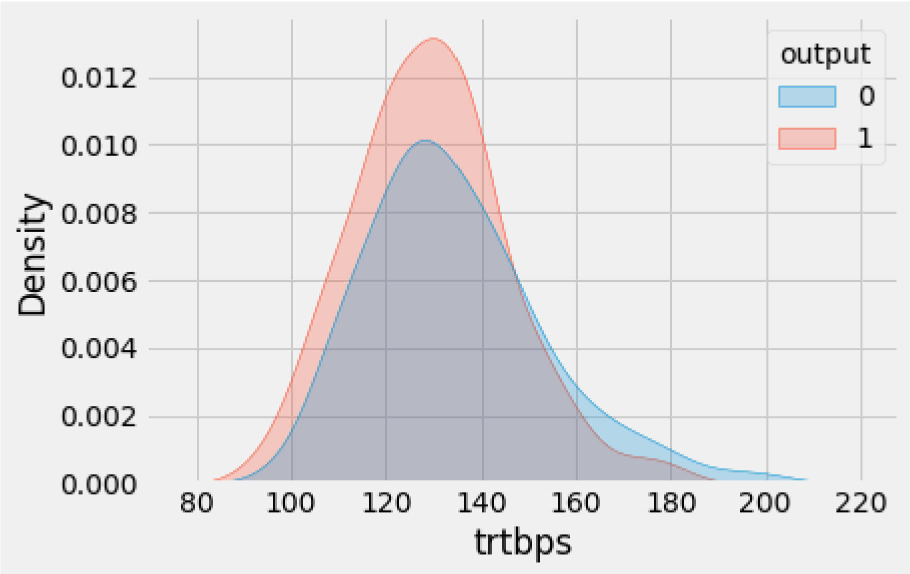


- **as with the increase in the age the blood pressure is incresing and also the people with more chances of heart attacks are also increasing hence we can say blood pressure increases the chance of heart attack**

In [35]:

```
sns.kdeplot( data=heart, x='trtbps',hue="output",fill=True)
```

Out[35]:



Model Building

In [36]:

```
target=heart['output']
target
```

Out[36]:

In [37]:

```
heart.drop(['output'],axis=1,inplace=True)
heart.head()
```

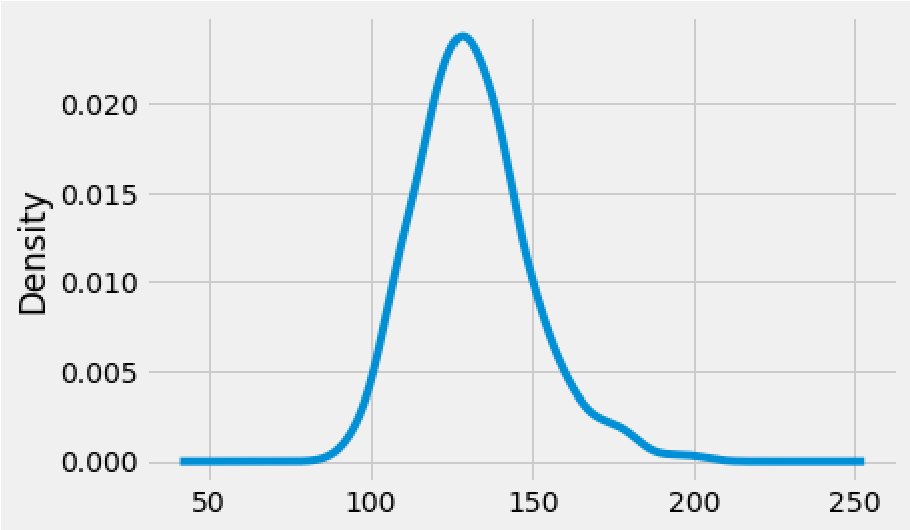
Out[37]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0

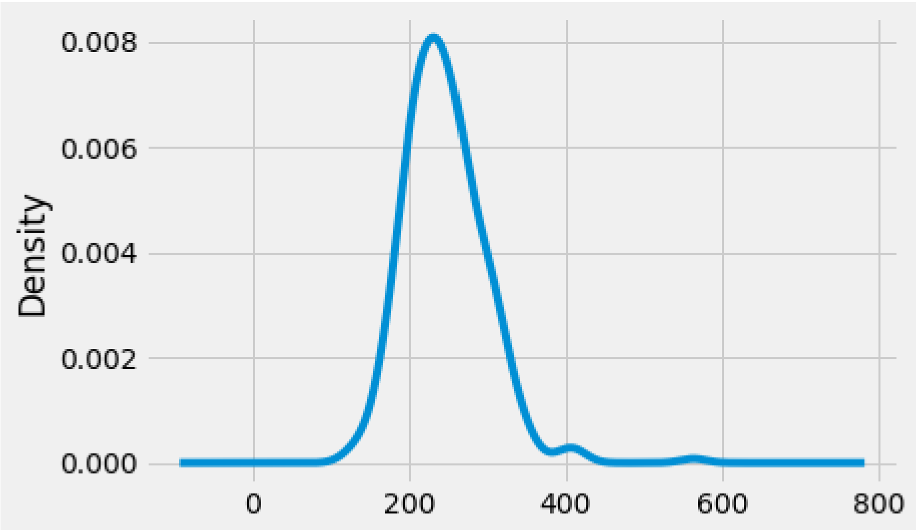
Checking for skewness

In [38]:

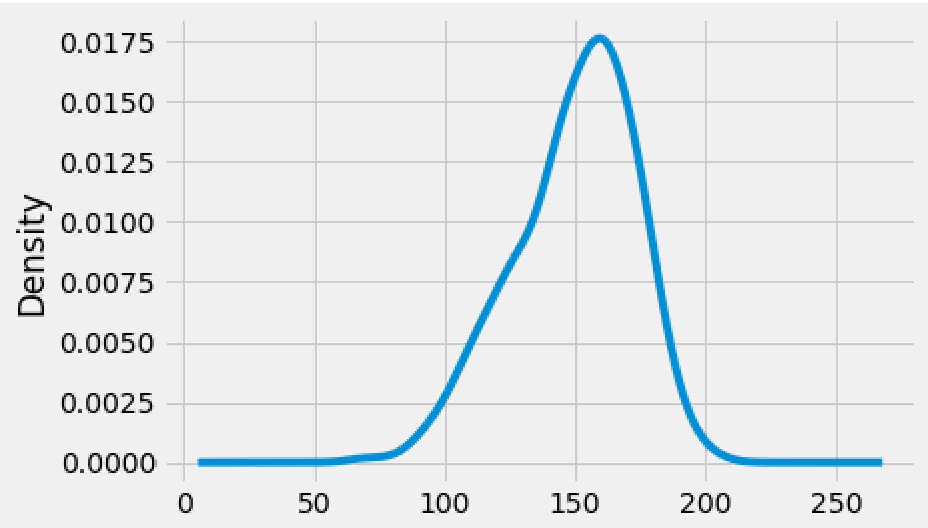
```
heart['trtbps'].plot(kind='density')
plt.show()
heart['chol'].plot(kind='density')
plt.show()
heart['thalachh'].plot(kind='density')
plt.show()
heart['age'].plot(kind='density')
plt.show()
```



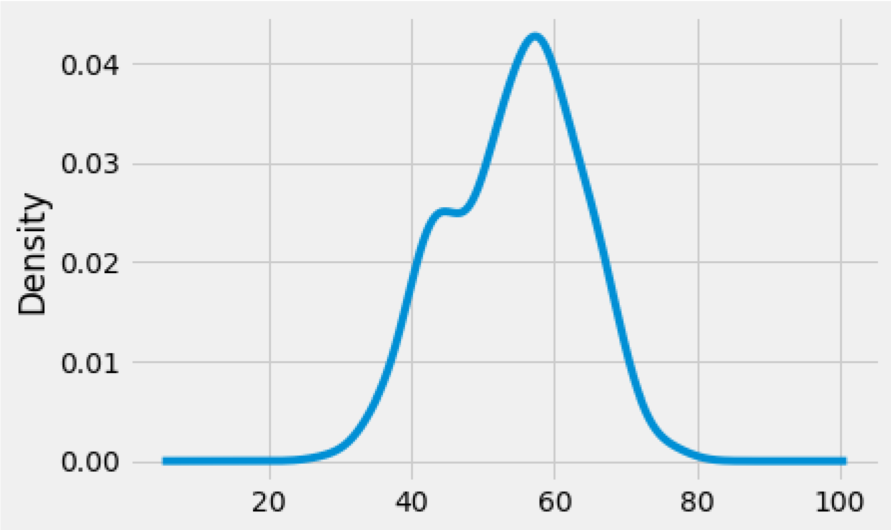
Density
0.020
0.015
0.010
0.005
0.000
50
100
150
200
250



Density
0.008
0.006
0.004
0.002
0.000
0
200
400
600
800



Density
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
50
100
150
200
250



In [39]:

```
heart.head(1)
```

Out[39]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa
0	63	1	3	145	233	1	0	150	0	2.3	0	0

Robust Scaler

In [40]:

```
from sklearn import preprocessing
scaler = preprocessing.RobustScaler()
robust_df = scaler.fit_transform(heart)
robust_df = pd.DataFrame(robust_df, columns =['age','sex','cp','trtbps','chol',
'fbs','restecg','thalachh','exng','oldpeak','slp','caa','thall'])
robust_df
```

Out[40]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpe
0	0.592593	0.0	1.0	0.75	-0.110236	1.0	-1.0	-0.092308	0.0	0.93
1	-1.333333	0.0	0.5	0.00	0.157480	0.0	0.0	1.046154	0.0	1.68
2	-1.037037	-1.0	0.0	0.00	-0.566929	0.0	-1.0	0.584615	0.0	0.37
3	0.074074	0.0	0.0	-0.50	-0.062992	0.0	0.0	0.769231	0.0	0.00
4	0.148148	-1.0	-0.5	-0.50	1.795276	0.0	0.0	0.307692	1.0	-0.1
...	...	...	...	...	...	...	...	...	...	...
298	0.148148	-1.0	-0.5	0.50	0.015748	0.0	0.0	-0.923077	1.0	-0.3
299	-0.740741	0.0	1.0	-1.00	0.377953	0.0	0.0	-0.646154	0.0	0.25
300	0.962963	0.0	-0.5	0.70	-0.740157	1.0	0.0	-0.369231	0.0	1.62
301	0.148148	0.0	-0.5	0.00	-1.716535	0.0	0.0	-1.169231	1.0	0.25
302	0.148148	-1.0	0.0	0.00	-0.062992	0.0	-1.0	0.646154	0.0	-0.5

303 rows × 13 columns

Standard Scaler

In [41]:

```
scaler = preprocessing.StandardScaler()
standard_df = scaler.fit_transform(robust_df)
standard_df = pd.DataFrame(standard_df, columns =['age','sex','cp','trtbps','cho
l','fbs','restecg','thalachh','exng','oldpeak','slp','caa','thall'])
```

In [42]:

```
standard_df.head()
```

Out[42]:

	age	sex	cp	trtbps	chol	fbs	restecg
0	0.952197	0.681005	1.973123	0.763956	-0.256334	2.394438	-1.005832
1	-1.915313	0.681005	1.002577	-0.092738	0.072199	-0.417635	0.898962
2	-1.474158	-1.468418	0.032031	-0.092738	-0.816773	-0.417635	-1.005832
3	0.180175	0.681005	0.032031	-0.663867	-0.198357	-0.417635	0.898962
4	0.290464	-1.468418	-0.938515	-0.663867	2.082050	-0.417635	0.898962

Train Test Split

In [43]:

```
x_train,x_test,y_train,y_test=train_test_split(heart,target,test_size=0.1,random
_state=42)
```

Logistic Regresson

In [44]:

```
logistic=LogisticRegression(max_iter=100,random_state=1,n_jobs=-1)
logistic.fit(x_train,y_train)
pred1=logistic.predict(x_test)
pred1
```

Out[44]:

In [45]:

```
logistic.score(x_train,y_train)*100
```

Out[45]:

In [46]:

```
logistic.score(x_test,y_test)*100
```

Out[46]:

In [47]:

```
from sklearn.metrics import accuracy_score

print('Logistic Regresson model accuracy score: {0:0.4f}'. format(accuracy_score
(y_test, pred1)))
```

In [48]:

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(x_train, y_train)
d_pred = decision_tree.predict(x_test)
acc_decision_tree = round(decision_tree.score(x_train,y_train)*100,2)
print(f'{acc_decision_tree}%')
```

In [49]:

```
from sklearn.metrics import accuracy_score

print('Decision Tree model accuracy score: {0:0.4f}'. format(accuracy_score(y_te
st, d_pred)))
```

LightGBM

Lightgbm

In [50]:

```
import lightgbm as lgb
lgbm= lgb.LGBMClassifier()
lgbm.fit(x_train,y_train)
pred2=lgbm.predict(x_test)
acc_lgbm=round(lgbm.score(x_train,y_train)*100,2)
print(f'{acc_lgbm}%')
```

In [51]:

```
from sklearn.metrics import accuracy_score

print('LightGBM model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, p
red2)))
```

XGBoost

In [52]:

```
import xgboost as xgb
# define data_dmatrix
data_dmatrix = xgb.DMatrix(data=heart,label=target)
```

DMatrix is an internal data structure that is used by XGBoost, which is optimized for both memory efficiency and training speed.

In [53]:

```
params = {
            'objective':'binary:logistic',
            'max_depth': 4,
            'alpha': 10,
            'learning_rate': 0.01,
            'n_estimators':100
        }
```

In [54]:

```
import xgboost as xgb
xgbo= xgb.XGBClassifier(**params)
xgbo.fit(x_train,y_train)
pred3=xgbo.predict(x_test)
acc_xgbo=round(xgbo.score(x_train,y_train)*100,2)
print(f'{acc_xgbo}%')
```

In [55]:

```
from sklearn.metrics import accuracy_score

print('XGBoost model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, pr
ed3)))
```

XGBoost with Cross Validation

In [56]:

```
# cross validation
from xgboost import cv

params = {"objective":"binary:logistic",'colsample_bytree': 0.3,'learning_rate':
0.1,
                'max_depth': 5, 'alpha': 10}

xgb_cv = cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=50, early_stopping_rounds=10, metrics="auc",
as_pandas=True, seed=123)
```

In [57]:

```
xgb_cv.head()
```

Out[57]:

	train-auc-mean	train-auc-std	test-auc-mean	test-auc-std
0	0.736104	0.021106	0.728390	0.038283
1	0.796951	0.023294	0.722476	0.018610
2	0.845681	0.041675	0.792923	0.038228
3	0.886830	0.030383	0.855427	0.025536
4	0.896603	0.008693	0.866903	0.008230

In [58]:

```
xgb_cv.shape
```

Out[58]:

In [59]:

```
accuracy_xgb=xgb_cv["test-auc-mean"][49]
print(accuracy_xgb)
```

The accuracy is increased from 0.8387 to 0.9028

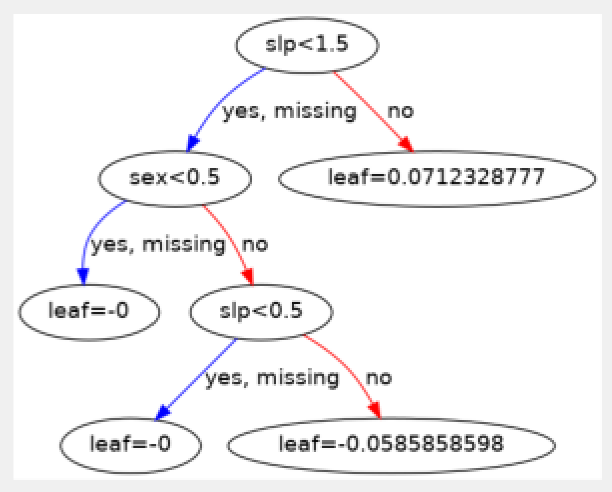
Insights how model has arrived at its final decision

In [60]:

```
xg_reg = xgb.train(params=params,dtrain=data_dmatrix, num_boost_round=10)
```

In [61]:

```
xgb.plot_tree(xg_reg,num_trees=0)
plt.rcParams['figure.figsize'] = [50, 20]
plt.show()
```

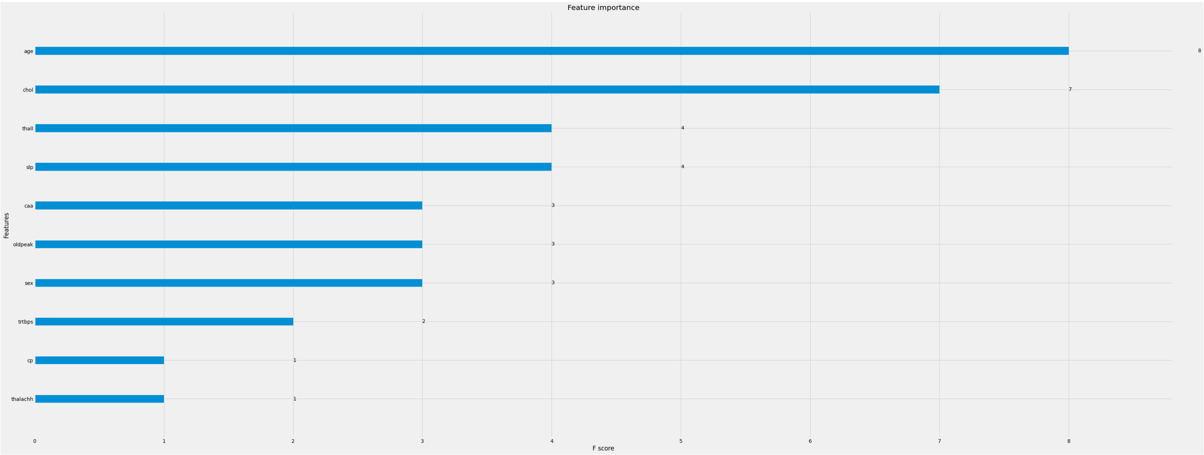


These plots provide insight into how the model arrived at its final decisions and what splits it made to arrive at those decisions.

Feature Importance

In [62]:

```
xgb.plot_importance(xg_reg)
plt.rcParams['figure.figsize'] = [6, 6]
plt.show()
```



The most imporatant feature of the dataset is age and the less important feature id chest pain and thalachh

Score Comparison

In [63]:

```
models = pd.DataFrame({
    'Model' : ['Logistic Regression','Decision Tree', 'Lightgbm', 'XgBoost','XgBoost with cross validation'],
    'Score' : [accuracy_score(y_test, pred1)*100,accuracy_score(y_test, d_pred)*100,accuracy_score(y_test, pred2)*100,accuracy_score(y_test, pred3)*100,accuracy_xgb*100]
})


models.sort_values(by = 'Score', ascending = False)
```

Out[63]:

	Model	Score
4	XgBoost with cross validation	90.275833
3	XgBoost	83.870968
0	Logistic Regression	80.645161
1	Decision Tree	80.645161
2	Lightgbm	80.645161