Heart attack analysis and prediction

Variables-

- **Age** - age of patient
- **Sex** - gender of the patient
- **cp** - Chest Pain type
  - Value 1: typical angina
  - Value 2: atypical angina
  - Value 3: non-anginal pain
  - Value 4: asymptomatic
- **trtbps** - resting blood pressure (in mm Hg)
- **chol** - cholestoral in mg/dl
- **fbs** - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- **_restecg** - resting electrocardiographic results
  - Value 0: normal
  - Value 1: having ST-T wave abnormality (possible heart strain)
  - Value 2: showing probable or definite left ventricular hypertrophy (enlarged muscle)
- **thalach** - maximum heart rate achieved
- **exang** - exercise induced (chest pain)angina (1 = yes; 0 = no)
- **oldpeak** - ST depression induced by exercise relative to rest (hearts ability to get oxygen)
- **slp** - the slope of the peak exercise ST segment (2 = upsloping; 1 = flat; 0 = downsloping)
- **ca** - number of major vessels (0-3)
- **thal** - Thallium stress test - 2 = normal; 1 = fixed defect; 3 = reversable defect
- **output** - chance of hear attack
  - 0= less chance of heart attack
  - 1= more chance of heart attak

dataset

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
df=pd.read_csv('datasets/heart.csv')
```

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

In [ ]:
```
df.shape
```

In [ ]:
```
df.info()
```

Output analysis

- the data has 303 rows and 14 columns
- all the variables are in numerical format
- there are no missing values

Preparation for Exploratory Data Analysis (EDA)

missing values

In [ ]:
```
df.isnull().sum()
```

examining unique values

In [ ]:
```
unique=[]
for i in df.columns:
    x=df[i].value_counts().count()
    unique.append(x)
un=pd.DataFrame(unique,index=df.columns,columns=['Total unique values'])
un
```

output analysis

- variables with few unique values are categorical
- variables with high unique values are numeric
- numeric - age, trtbps, chol, thalachh, oldpeak
- categorical - sex, cp, fbs, restecg, exng, slp, caa, thall, output

sperating variables as categoric and numeric

In [ ]:
```
numeric_var = ["age", "trtbps", "chol", "thalachh", "oldpeak"]
categoric_var = ["sex", "cp", "fbs", "restecg", "exng", "slp", "caa", "thall",
```

observing statistics

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

outliers

In [ ]:
```
fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize = (20, 6))

ax1.boxplot(df["age"])
ax1.set_title("age")

ax2.boxplot(df["trtbps"])
ax2.set_title("trtbps")

ax3.boxplot(df["thalachh"])
ax3.set_title("thalachh")

ax4.boxplot(df["oldpeak"])
ax4.set_title("oldpeak")

plt.show()
```

Exploratory Data Analysis

numeric data visualisations

age

In [ ]:
```
sns.set(style= 'whitegrid',rc={"figure.figsize": (6, 4)},palette='tab20c')
sns.histplot(df['age'], kde=True)
```

- it follows a normal distrbution

trtbps

In [ ]:
```
sns.histplot(df['trtbps'],kde=True)
plt.xlabel('resting blood pressure')
plt.title('trtbps')
```

- it is noramly distributed with some right skewness

chol

In [ ]:
```
sns.histplot(df['chol'],kde=True)
plt.xlabel('cholestrol')
plt.title('chol')
```

thalachh

In [ ]:
```
sns.histplot(df['thalachh'],kde=True)
plt.title('thalachh')
plt.xlabel('maximum heart rate acheived')
```

oldpeak

In [ ]:
```
sns.histplot(df['oldpeak'],kde=True)
plt.title('oldpeak')
plt.xlabel('st depression(hearts ability to get o2)')
```

categorical data visulaisations

sex

In [ ]:
```
x=df['sex'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['male (1)','female (0)'])
plt.title(' gender')
plt.show()
```

- number of male patients are more than twice of the female patients

cp

In [ ]:
```
x=df['cp'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['typical angina(0)','atypical angina(1)','n
plt.title(' chest pain type (cp)')
plt.show()
```

fbs

In [ ]:
```
x=df['fbs'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['false(0)','true(1)'])
plt.title(' fasting blood sugar > 120 (fbgs)')
plt.show()
```

restecg

In [ ]:
```
x=df['restecg'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['normal(0)','possible strain(1)','enlarged
plt.title('resting electrocardiographic result (restecg)')
plt.show()
```

exng

In [ ]:
```
x=df['exng'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['no(0)','yes(1)'])
plt.title('excersise induced chest pain (exng)')
plt.show()
```

slp

In [ ]:
```
x=df['slp'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['upsloping(2)','flat(1)','downsloping(0)'])
plt.title('slope of st segment (slp)')
plt.show()
```

caa

In [ ]:
```
x=df['caa'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=x.index)
plt.title('number of major vessels (caa)')
plt.show()
```

thall

In [ ]:
```
x=df['thall'].value_counts()
```

- There are three observation values in the description of this variable. However, the table shows four values.
- 0 is null value. we replace this with 2

In [ ]:
```
df['thall']=df['thall'].replace(0,2)
```

In [ ]:
```
plt.pie(x,autopct='%.0f%%',labels=x.index)
plt.title('thallium stress test result (thall) ')
plt.show()
```

output

In [ ]:
```
x=df['output'].value_counts()
plt.pie(x,autopct='%.0f%%',labels=['more chance(1)','less chance(0)'])
plt.title('chance of heart attack (output)')
plt.show()
```

Target variable with respect to numerical variables

In [ ]:
```
sns.set(palette='deep')
```

age

In [ ]:
```
sns.kdeplot(data=df, x="age", hue="output", fill=True, common_norm=False)
```

- there is a decrease in heart attack risk after the age of 55.

trtbps - resting blood pressure

In [ ]:
```
sns.kdeplot(data=df, x="trtbps", hue="output", fill=True, common_norm=False)
```

chol- cholestrol

In [ ]:
```
sns.kdeplot(data=df, x="chol", hue="output", fill=True, common_norm=False)
```

In [ ]:
```
sns.kdeplot(data=df, x="thalachh", hue="output", fill=True, common_norm=False)
```

- The higher the maximum reached heart rate, the higher the probability of the patient having a heart attack.

In [ ]:
```
sns.kdeplot(data=df, x="oldpeak", hue="output", fill=True, common_norm=False)
```

- if the value of this variable is between 0 and 1.5, there is a significant increase in the probability of having a heart attack.

Target variable with respect to categorical variables

gender

In [ ]:
```
sns.countplot(x='sex',data=df,hue='output')
```

- female patients are at a higher risk of heart attack

cp - chest pain type

In [ ]:
```
sns.countplot(x='cp',data=df,hue='output')
```

- chest pain type 2 - non-anginal pain has highest risk of heart attack

fbs- fasting blood sugar

In [ ]:
```
sns.countplot(x='fbs',data=df,hue='output')
```

restecg

In [ ]:
```
sns.countplot(x='restecg',data=df,hue='output')
```

- people with restecg 1- possible strain on heart have highest risk of attack

exng - pain due to excersise

In [ ]:
```
sns.countplot(x='exng',data=df,hue='output')
```

- patients who do not have pain related to excersie are more likely to have a heart attack

slp - slope

In [ ]:
```
sns.countplot(x='slp',data=df,hue='output')
```

- patients with slope 2 (unsloping) are 3 times more likely to have a heart attack than not having one

caa - no. of major vessels

In [ ]:
```
sns.countplot(x='caa',data=df,hue='output')
```

- the risk of heart attack is almost three times higher in patients with an observation value of 0.

thall - thallium stress test

In [ ]:
```
sns.countplot(x='thall',data=df,hue='output')
```

- Patients with an observation value of 2 are three times more likely to have a heart attack than not having one

correlation

- correlation of all the variables with chance of heart attack

In [ ]:
```
df.corr().iloc[:,-1]
```

correlation of all variables with each other

In [ ]:
```
df.corr()
```

In [ ]:
```
sns.heatmap(df.corr())
```

feature scaling

robust scaler

used when data contains outliers and is not normally distrivuted

In [ ]:
```
from sklearn.preprocessing import RobustScaler
```

In [ ]:
```
rbs=RobustScaler()
```

In [ ]:
```
scaled_df=rbs.fit_transform(df[numeric_var])
sc=pd.DataFrame(scaled_df,columns=numeric_var)
```

In [ ]:
```
df[categoric_var]
```

create df with scaled data

In [ ]:
```
df2=df[categoric_var].join(sc)
df2
```

applying one hot encoding to categorical variables

In [ ]:
```
df[categoric_var] = df[categoric_var].astype(str)
```

In [ ]:
```
df_c = pd.get_dummies(df2, columns=categoric_var[:-1], drop_first = True, dtyp
df_c
```

Model Building

seperating data into test and train

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
X=df_c.drop(['output'],axis=1)
y=df_c[['output']]
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, ran
```

model 1 - Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [ ]:
```
lr= LogisticRegression()
```

In [ ]:
```
lr.fit(X_train,y_train)
```

In [ ]:
```
y_pred=lr.predict(X_test)
y_pred
```

In [ ]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy: ",accuracy)
```

cross validation

In [ ]:
```
from sklearn.model_selection import cross_val_score
```

In [ ]:
```
scores = cross_val_score(lr, X_test, y_test, cv = 10)
print("Cross-Validation Accuracy Scores", scores.mean())
```

model 2 - Decision tree algorithm

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
dt=DecisionTreeClassifier(random_state=10)
dt.fit(X_train,y_train)
```

In [ ]:
```
y_pred=dt.predict(X_test)
```

In [ ]:
```
print("The test accuracy score of Decision Tree is:", accuracy_score(y_test, y
```

cross validation

In [ ]:
```
scores = cross_val_score(dt, X_test, y_test, cv = 10)
print("Cross-Validation Accuracy Scores", scores.mean())
```

model 3 - support vector machine algorithm

In [ ]:
```
from sklearn.svm import SVC
```

In [ ]:
```
sv=SVC(random_state=25)
sv.fit(X_train,y_train)
```

In [ ]:
```
y_pred=sv.predict(X_test)
```

In [ ]:
```
print("The test accuracy score of SVM is:", accuracy_score(y_test, y_pred))
```

cross validation

In [ ]:
```
scores = cross_val_score(sv, X_test, y_test, cv = 10)
print("Cross-Validation Accuracy Scores", scores.mean())
```

model 4 - Random forest algorithm

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:
```
rf = RandomForestClassifier(random_state = 32)
rf.fit(X_train,y_train)
```

In [ ]:
```
y_pred=rf.predict(X_test)
```

In [ ]:
```
print("The test accuracy score of Random Forest is", accuracy_score(y_test, y_
```

cross validation

In [ ]:
```
scores = cross_val_score(rf, X_test, y_test, cv = 10)
print("Cross-Validation Accuracy Scores", scores.mean())
```

conclusion

we should use the Random forest classification model as it has a 83.8% accuracy and 80.8% cross validation accuracy

In [ ]:
```

```