HEART DISEASE PREDICTION

Using : Anaconda (jupyter notebook)

Step1 : import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

Step2 : df=pd.read_csv('downloads/heart.csv')

Step3 : df.head()

Step4 : df.isnull().sum()

Step5:print(df.info())

Step6 : plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True, cmap='terrain')

Step7 : sns.pairplot(data=df)

Step8 : df.hist(figsize=(12,22),layout=(5,3));

Step9 : #box and whiskers plot
df.plot(kind='box',subplots=True,layout=(5,3),figsize=(12,12))
plt.show()

Step10 : sns.catplot(data=df,x='sex',y='age',hue='target',palette='husl')

Step11: sns.barplot(data=df,x='sex',y='chol',hue='target',palette='spring')

Step12 : df['sex'].value_counts()

Step13 : df['cp'].value_counts()

Step14 : sns.countplot(x='cp',hue='target',data=df,palette='rocket')

Step15 : gen=pd.crosstab(df['sex'],df['target'])
print(gen)

Step16 : gen.plot(kind='bar',stacked=True,color=['green','yellow'],grid=False)

Step17 : chest_pain = pd.crosstab(df['cp'],df['target'])
chest_pain

Step18 : from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler =StandardScaler()
columns_to_scale=['age','trestbps','chol','thalach','oldpeak']
df[columns_to_scale]=StandardScaler.fit_transform(df[columns_to_scale])

Step19 : df.head()

Step20 : x=df.drop(['target'],axis=1)
y=df['target']

Step21 : x_train, x_test, y_train
,y_test=train_test_split(x,y,test_size=0.3,random_state=40)

Step22 : print('x_train-',x_train.size)
print('x_test-',x_test.size)

```
print('y_train-',y_train.size)
print('y_test-',x_test.size)
```

Step23 :
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier().fit(x_train,y_train)
knn.score(x_test,y_test)
```

Step24 :
```
from sklearn.ensemble import RandomForestClassifier
np.random.seed(41)
rf=RandomForestClassifier().fit(x_train,y_train)
rf.score(x_test,y_test)
```

Step25 :
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()

model1=lr.fit(x_train,y_train)
prediction1=model1.predict(x_test)
```

Step26 :
```
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,prediction1)
cm
```

Step27 :
```
sns.heatmap(cm,annot=True,cmap='BuPu')
```

Step28 :
```
TP=cm[0][0]
```

```
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print('Testing Accuracy: ',(TP+TN)/(TP+TN+FN+FP))
```

Step29 :
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,prediction1)
```

Step30 :
```
from sklearn.metrics import classification_report
print(classification_report(y_test,prediction1))
```

Step31 :
```
print('lr :',accuracy_score(y_test,prediction1))
```

Step32 :
```
from sklearn.metrics import confusion_matrix
prediction=knn.predict(x_test)
prediction
confusion_matrix=confusion_matrix(y_test,prediction)
```

confusion_matrix

Step33: from sklearn.metrics import confusion_matrix
prediction=rf.predict(x_test)
prediction
confusion_matrix1=confusion_matrix(y_test,prediction)
confusion_matrix1

Step34 : sns.heatmap(confusion_matrix,annot=True,cmap='BuPu')

Step35 : TP=confusion_matrix[0][0]
TN=confusion_matrix[1][1]
FN=confusion_matrix[1][0]
FP=confusion_matrix[0][1]
Print('Testing Accuracy: ',(TP+TN)/(TP+TN+FN+FP))

Step36 : sns.heatmap(confusion_matrix1,annot=True,cmap='BuPu')

Step37 : TP=confusion_matrix1[0][0]
TN=confusion_matrix1[1][1]

```
FN=confusion_matrix1[1][0]
FP=confusion_matrix1[0][1]
print('Testing Accuracy: ',(TP+TN)/(TP+TN+FN+FP)
```