linear regreesion

March 20, 2024

[8]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

[9]:

```
#importing data
data=pd.read_csv("C:\\Users\\user\\Downloads\\Dataset Heart Disease.csv")
data
```

[9]:

```
     age  chest pain type  resting bps  cholesterol  max heart rate
0     40                2          140          289             172
1     49                3          160          180             156
2     37                2          130          283              98
3     48                4          138          214             108
4     54                3          150          195             122
..   …                …            …            …               …
893   42                3          130          180             150
894   45                2          128          308             170
895   57                1          165          289             124
896   64                3          125          309             131
897   41                3          112          250             179

[898 rows x 5 columns]
```

calling arrays

[10]:

```
x=np.array(data["age"]).reshape(-1,1)
```

[11]:

```
y=np.array(data["max heart rate"])
```

checking for missing data

[12]:

```
data.isna().sum()
```

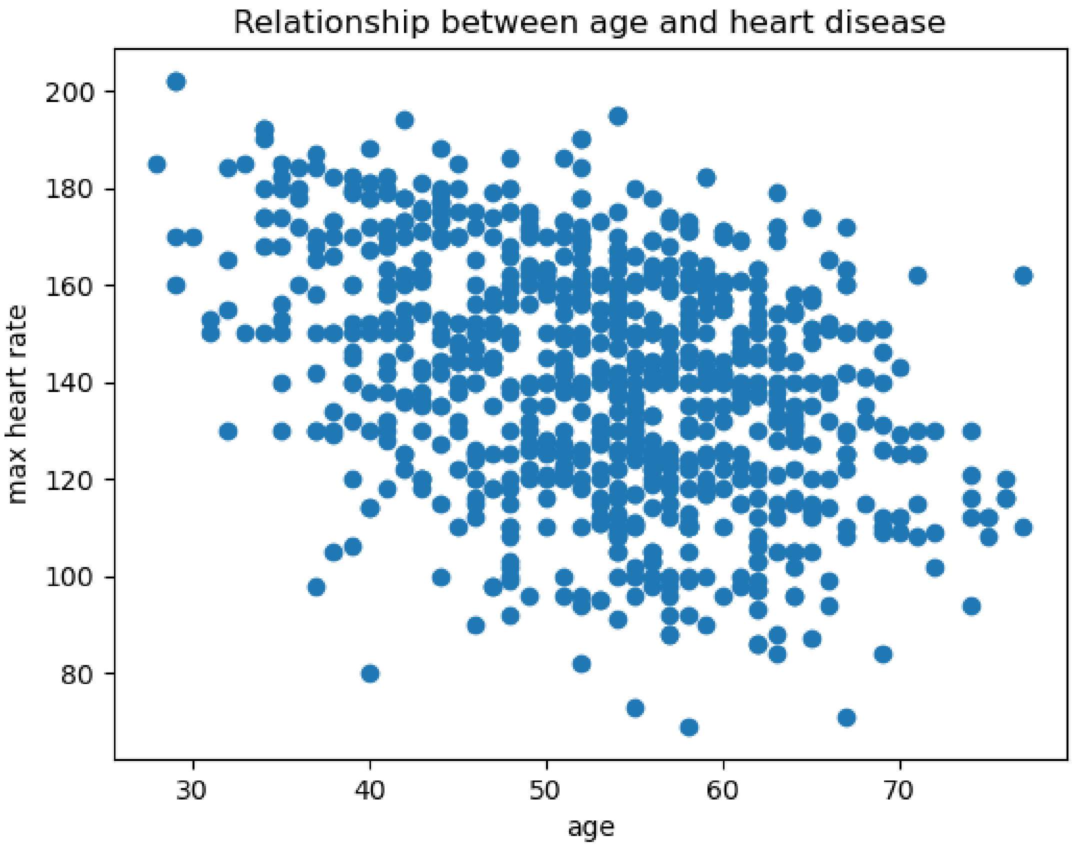
[12]:

```
age                0
chest pain type    0
resting bps        0
cholesterol        0
max heart rate     0
```

```
dtype: int64
```

graph

```
[13]: import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.xlabel("age")
plt.ylabel("max heart rate")
plt.title("Relationship between age and heart disease")
plt.show()
```



splitting data

```
[14]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
  ↪2,random_state=42)
```

#standardizing data from sklearn.preprocessing import StandardScaler scaler = StandardScaler() x_train_scaled = scaler.fit_transform(x_train) x_test_scaled = scaler.transform(x_test)

[15]:

```
#building the model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model
```

[15]:

```
LinearRegression()
```

[16]:

```
#model fitting
model.fit(x_train,y_train)
```

[16]:

```
LinearRegression()
```

[17]:

```
#making predictions
y_pred=model.predict(x_test)
```

[18]:

```
model.coef_
```

[18]:

```
array([-0.96626539])
```

[19]:

```
#finding intercept(regression line intercepts with the y axis)
model.intercept_
```

[19]:

```
193.4557211223224
```

[20]:

```
#model evaluation
#model accuracy on train values
model.score(x_train,y_train)
```

[20]:

```
0.1411214865128494
```

[21]:

```
#model accuracy on the test values
model.score(x_test,y_test)
```

[21]:

```
0.09956280781317806
```

[22]:

```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error
print(mean_squared_error(y_test,y_pred))
```

```
573.7417929335393
```

[23]:

```
print(mean_absolute_error(y_test,y_pred))
```

```
19.89992588788073
```

[24]:

```
print(r2_score(y_test,y_pred))
```

```
0.09956280781317806
```

MODEL OPTIMIZATION

[25]:
```
from sklearn.model_selection import GridSearchCV
model = LinearRegression()
```

[26]:
```
#performing GridSearchCV to find best hyperparameters
param_grid = {'fit_intercept':[True,False],'positive':[True,False],'positive':
  ↪[True,False]}
param_grid
```

[26]:
```
{'fit_intercept': [True, False], 'positive': [True, False]}
```

[27]:
```
#fitting data
data= GridSearchCV(model,param_grid, cv=5)
data.fit(x_train,y_train)
```

[27]:
```
GridSearchCV(cv=5, estimator=LinearRegression(),
             param_grid={'fit_intercept': [True, False],
                         'positive': [True, False]})
```

[28]:
```
#getting the best parameter from the GridSearch
best_params = data.best_params_['fit_intercept']
best_params
```

[28]:
```
True
```

[29]:
```
best_params = data.best_params_['positive']
best_params
```

[29]:
```
False
```

[30]:
```
#training the model using the best parameters
model=LinearRegression(fit_intercept=best_params)
model.fit(x_train,y_train)
```

[30]:
```
LinearRegression(fit_intercept=False)
```

[31]:
```
#making predictions
y_pred = model.predict(x_test)
```

[32]:
```
#evaluating model performance
print(mean_absolute_error(y_test,y_pred))
```

```
32.93329029906092
```

[33]:
```
print(r2_score(y_test,y_pred))
```

```
-1.5683088109567058
```

[34]:
```
#finding accuracy on train values
model.score(x_train,y_train)
```

```
[34]: -1.813959890359337
```

```
[35]: #finding accuracy on test values
      model.score(x_test,y_test)
```

```
[35]: -1.5683088109567058
```