heard-disease-linear-reg

April 3, 2025

Importing all necessary libraries

[24]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
```

Reading csv file

[2]:
```
df = pd.read_csv("/content/heart.csv")
```

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

[3]:

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

Data cleaning

[4]:
```
df = df.drop_duplicates()
```

display basic statics

[5]:
```
df.describe()
```

[5]:
```
              age         sex          cp      trtbps        chol         fbs  \
count  302.00000  302.000000  302.000000  302.000000  302.000000  302.000000
mean    54.42053    0.682119    0.963576  131.602649  246.500000    0.149007
std      9.04797    0.466426    1.032044   17.563394   51.753489    0.356686
min     29.00000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     48.00000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.50000    1.000000    1.000000  130.000000  240.500000    0.000000
75%     61.00000    1.000000    2.000000  140.000000  274.750000    0.000000
max     77.00000    1.000000    3.000000  200.000000  564.000000    1.000000

           restecg    thalachh        exng     oldpeak         slp         caa  \
count  302.000000  302.000000  302.000000  302.000000  302.000000  302.000000
mean     0.526490  149.569536    0.327815    1.043046    1.397351    0.718543
std      0.526027   22.903527    0.470196    1.161452    0.616274    1.006748
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.250000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  152.500000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

            thall      output
count  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

display column information

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 302 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       302 non-null    int64
 1   sex       302 non-null    int64
 2   cp        302 non-null    int64
 3   trtbps    302 non-null    int64
 4   chol      302 non-null    int64
 5   fbs       302 non-null    int64
 6   restecg   302 non-null    int64
 7   thalachh  302 non-null    int64
 8   exng      302 non-null    int64
```

```
 9   oldpeak   302 non-null    float64
 10  slp       302 non-null    int64
 11  caa       302 non-null    int64
 12  thall     302 non-null    int64
 13  output    302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

check for null values

[7]:
```
print("\nNull values in each column:")
print(df.isna().sum())
```

```
Null values in each column:
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

Remove outliers function

[8]:
```
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5*IQR
    outlier_mask = (column < Q1 - threshold) | (column > Q3 + threshold)
    return column[~outlier_mask]
```

Remove outliers for selected column

[12]:
```
col_names =['cp', 'thalachh', 'exng', 'oldpeak', 'slp', 'caa']
for col in col_names:
    df[col] = remove_outliers(df[col])
```

Drop any remaining NA values

[13]:
```
df = df.dropna()
```

Drop ‘fbs’ cloumn

[14]:
```
df = df.drop('fbs',axis=1)
```

Compute correlation between features and target

[15]:
```
correlations = df.corr()['output'].drop('output')
print("\nCorrelation between features and Target:")
print(correlations)
```

```
Correlation between features and Target:
age         -0.193798
sex         -0.303271
cp           0.410807
trtbps      -0.135238
chol        -0.052796
restecg      0.122071
thalachh     0.384609
exng        -0.444401
oldpeak     -0.437895
slp          0.329432
caa         -0.460816
thall       -0.366390
Name: output, dtype: float64
```

Data split Using the same feature

[16]:
```
X = df[['cp', 'thalachh', 'exng', 'oldpeak', 'slp', 'caa']]
y = df['output']
```

[17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)
```

[18]:
```
print("\nShapes after split:")
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
Shapes after split:
X_train: (220, 6)
X_test: (55, 6)
y_train: (220,)
y_test: (55,)
```

Data transformation - Standard Scaling

[19]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Data model building - Linear Regression

[20]:
```
model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

[20]:
```
LinearRegression()
```

Make predictions on the test set

[21]:
```
y_pred = model.predict(X_test_scaled)
```

Since output is binary (0 or 1), we'll round the predictions for classification

[22]:
```
y_pred_class = np.round(y_pred).astype(int)
```

Evaluate the model's performance

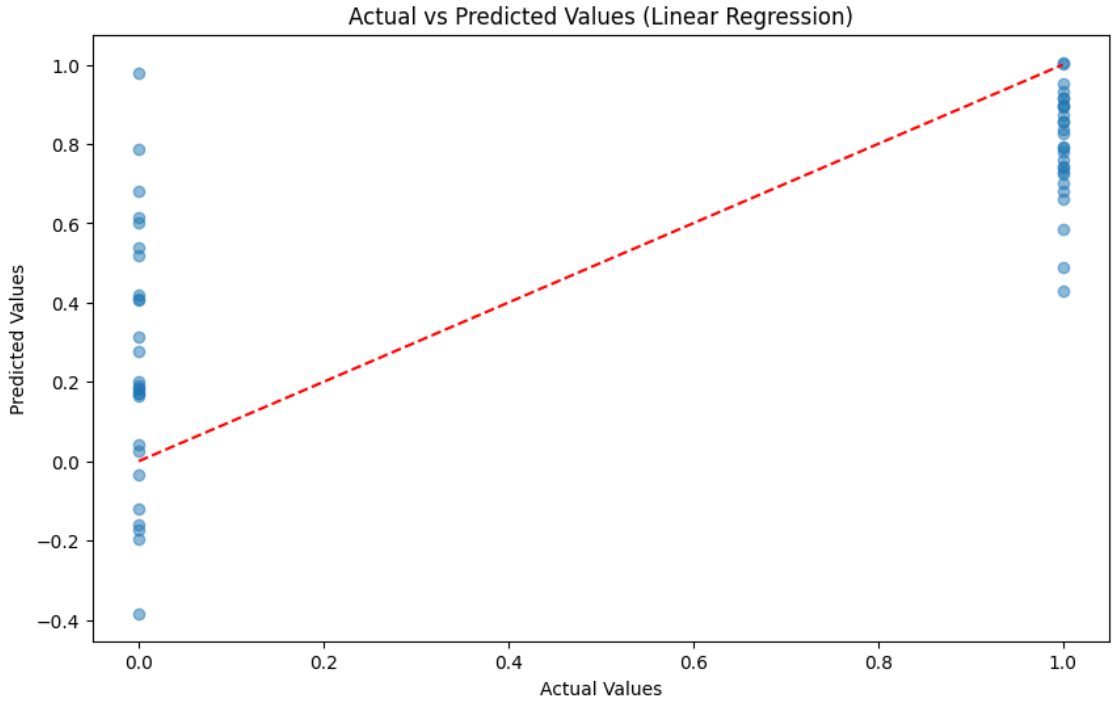
[25]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred_class)
```

[26]:
```
print("\nModel Evaluation:")
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)
print("Accuracy (rounded predictions):", accuracy)
```

```
Model Evaluation:
Mean Squared Error: 0.11305445035073602
R-squared Score: 0.5476326556733115
Accuracy (rounded predictions): 0.8363636363636363
```

Plotting actual vs predicted values

[27]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--r')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values (Linear Regression)')
plt.show()
```



Coefficient analysis

[28]:
```
coefficients = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_
}).sort_values(by='Coefficient', ascending=False)
```

[29]:
```
print("\nFeature Coefficients:")
print(coefficients)
```

```
Feature Coefficients:
    Feature  Coefficient
0        cp     0.108753
4       slp     0.050169
1  thalachh     0.039999
2      exng    -0.072110
3   oldpeak    -0.092916
5       caa    -0.155755
```