muhumuza Stephen

March 24, 2024

[1]:
```
# importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,PolynomialFeatures
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score
import matplotlib.pyplot as plt
```

[2]:
```
# importing my dataset
data = pd.read_csv('C:
↪\\Users\\lynda\\Desktop\\Assignment\\student_scores_dataset.csv')
data.head()
```

[2]:

	Study Hours	Exam Scores
0	3.7	87.9
1	9.5	143.6
2	7.3	123.7
3	6.0	99.9
4	1.6	64.5

[3]:
```
# identifying my variables
x=np.array(data['Study Hours']).reshape(-1,1)
x
```

[3]:
```
array([[3.7],
       [9.5],
       [7.3],
       [6. ],
       [1.6],
       [1.6],
       [0.6],
       [8.7],
       [6. ],
       [7.1],
       [0.2],
       [9.7],
       [8.3],
```

```
[2.1],
[1.8],
[1.8],
[3. ],
[5.2],
[4.3],
[2.9],
[6.1],
[1.4],
[2.9],
[3.7],
[4.6],
[7.9],
[2. ],
[5.1],
[5.9],
[0.5],
[6.1],
[1.7],
[0.7],
[9.5],
[9.7],
[8.1],
[3. ],
[1. ],
[6.8],
[4.4],
[1.2],
[5. ],
[0.3],
[9.1],
[2.6],
[6.6],
[3.1],
[5.2],
[5.5],
[1.8],
[9.7],
[7.8],
[9.4],
[8.9],
[6. ],
[9.2],
[0.9],
[2. ],
[0.5],
[3.3],
```

```
       [3.9],
       [2.7],
       [8.3],
       [3.6],
       [2.8],
       [5.4],
       [1.4],
       [8. ],
       [0.7],
       [9.9],
       [7.7],
       [2. ],
       [0.1],
       [8.2],
       [7.1],
       [7.3],
       [7.7],
       [0.7],
       [3.6],
       [1.2],
       [8.6],
       [6.2],
       [3.3],
       [0.6],
       [3.1],
       [3.3],
       [7.3],
       [6.4],
       [8.9],
       [4.7],
       [1.2],
       [7.1],
       [7.6],
       [5.6],
       [7.7],
       [4.9],
       [5.2],
       [4.3],
       [0.3],
       [1.1]])
```

[4]:
```
# my dependent variable
y=data['Exam Scores']
y.head()
```

[4]:
```
0     87.9
1    143.6
```

```
2    123.7
3     99.9
4     64.5
Name: Exam Scores, dtype: float64
```

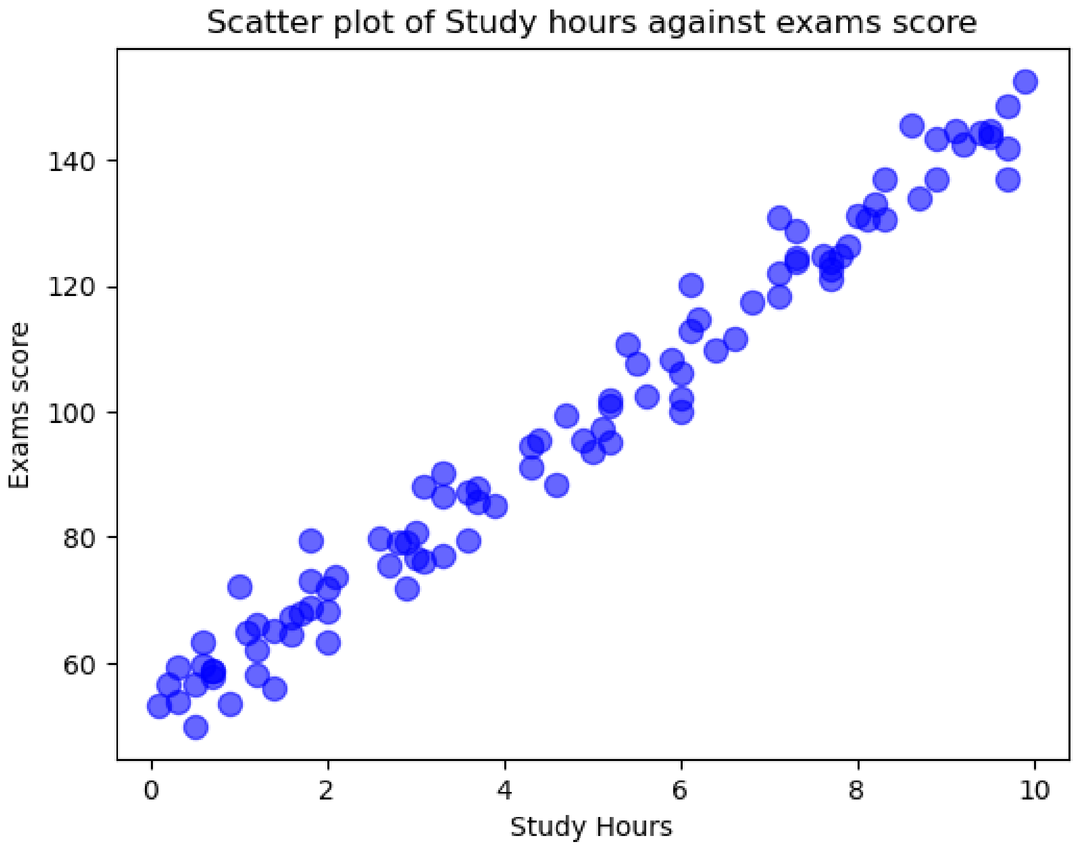
[5]:
```
# checking for missing values
data.isnull().sum
```

[5]:
```
<bound method NDFrame._add_numeric_operations.<locals>.sum of     Study Hours
Exam Scores
0         False        False
1         False        False
2         False        False
3         False        False
4         False        False
..          …            …
95        False        False
96        False        False
97        False        False
98        False        False
99        False        False

[100 rows x 2 columns]>
```

[6]:
```
#visualising the data
plt.scatter(x,y, label='Data_points',alpha=0.6, color='blue', s=75)
plt.title('Scatter plot of Study hours against exams score')
plt.xlabel('Study Hours')
plt.ylabel('Exams score')
```

[6]:
```
Text(0, 0.5, 'Exams score')
```



[7]:
```
# data preproccing
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2,␣
↪random_state=42)
```

[8]:
```
#x-train
x_train
```

[8]:
```
array([[9.2],
       [8.9],
       [2. ],
       [0.3],
       [9.9],
       [1.8],
       [1.2],
       [5.2],
       [7.1],
       [0.1],
       [9.7],
       [5.2],
       [3.3],
```

```
[5.9],
[5.6],
[1.6],
[1.4],
[5.4],
[8.1],
[3. ],
[1.8],
[9.7],
[8.7],
[4.9],
[5.1],
[2.9],
[6.2],
[7.9],
[8.3],
[2.1],
[4.6],
[6. ],
[5.2],
[6.8],
[6. ],
[3.6],
[0.6],
[2.8],
[3. ],
[4.7],
[0.9],
[1.1],
[6. ],
[9.1],
[9.7],
[8. ],
[3.1],
[0.7],
[2.7],
[4.3],
[1.2],
[5. ],
[0.5],
[5.5],
[0.3],
[2. ],
[7.3],
[0.7],
[7.7],
[3.3],
```

```
        [3.6],
        [3.1],
        [1. ],
        [0.5],
        [9.5],
        [9.4],
        [1.4],
        [7.3],
        [3.7],
        [6.4],
        [7.1],
        [7.1],
        [7.3],
        [3.3],
        [6.1],
        [3.9],
        [2. ],
        [1.8],
        [7.6],
        [7.8]])
```

[9]:
```
#x_test
x_test
```

[9]:
```
array([[0.6],
       [8.9],
       [7.7],
       [6.6],
       [2.6],
       [4.4],
       [2.9],
       [8.6],
       [0.2],
       [3.7],
       [4.3],
       [6.1],
       [8.2],
       [9.5],
       [1.2],
       [1.6],
       [7.7],
       [0.7],
       [8.3],
       [1.7]])
```

[10]:
```
# standardazing the independent variable
scaler= StandardScaler()
```

```
x_train_scaled=scaler.fit_transform(x_train)
x_test_scaled=scaler.transform(x_test)
```

[11]:
```
#creating model or building modele
model= LinearRegression()
model.fit(x_train_scaled,y_train)
```

[11]:
```
LinearRegression()
```

[12]:
```
#Evaluting the model
y_pred=model.predict(x_test_scaled)
mae= mean_absolute_error(y_test,y_pred)
mse= mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)

print('Mean Absolute Error:',mae)
print('Mean Squared Error:',mse)
print('R-squared:',r2)
```

```
Mean Absolute Error: 2.9365732667749755
Mean Squared Error: 16.202109700645348
R-squared: 0.9826924926918468
```

[13]:
```
# interppreting my coefficients
model.intercept_
print('model.intercept',model.intercept_)
print('The intercept is 96.5875. It means that if the study hours are zero, the␣
↪predicted exam score would be 96.5875.')
```

```
model.intercept 96.5875
The intercept is 96.5875. It means that if the study hours are zero, the
predicted exam score would be 96.5875.
```

[14]:
```
model.coef_
print('model.coef',model.coef_)
print(' the coefficient for study hours is 28.52556103. It means that for every␣
↪one-unit increase in study hours, the exam scores are expected to increase␣
↪by approximately 28.53, assuming all other factors remain constant.')
```

```
model.coef [28.52556103]
 the coefficient for study hours is 28.52556103. It means that for every one-
unit increase in study hours, the exam scores are expected to increase by
approximately 28.53, assuming all other factors remain constant.
```

[15]:
```
#model improvement
#Applying polynomial Features
poly= PolynomialFeatures(degree=2)
x_train_poly =poly.fit_transform(x_train)
x_test_poly =poly.transform(x_test)
```

[16]:
```
#scaling
scaler = StandardScaler()
x_train_scaled1=scaler.fit_transform(x_train_poly)
x_test_scaled1 = scaler.transform(x_test_poly)
```

[17]:
```
#training the linearRegression model with polynomial features
model_poly = LinearRegression()
model_poly.fit(x_train_scaled1,y_train)
```

[17]:
```
LinearRegression()
```

[18]:
```
#making prediction
y_pred_poly= model_poly.predict(x_test_scaled1)

y_pred_poly
```

[18]:
```
array([ 57.67108889, 138.71725719, 126.05217671, 114.72389663,
        75.79896173,  92.87476622,  78.59488185, 135.52095869,
        54.15228189,  86.1485391 ,  91.90720366, 109.66365139,
       131.2903679 , 145.16991094,  63.01602912,  66.62380909,
       126.05217671,  58.55635145, 132.34467911,  67.5313149 ])
```

[19]:
```
#Evaluattion of my mode
mae_poly= mean_absolute_error(y_test,y_pred_poly)
mse_poly = mean_squared_error(y_test,y_pred_poly)
r2_poly = r2_score(y_test,y_pred_poly)
print('Mean Absolute error:',mae_poly,)
print('Mean Squared Error (Polynomial):',mse_poly)
print('R-squared (polynomial):',r2_poly)
```

```
Mean Absolute error: 2.828595846925279
Mean Squared Error (Polynomial): 15.643638436299161
R-squared (polynomial): 0.9832890659571593
```

[ ]:
```
#comparison in evaluation

Based on these comparisons, it appears that the model with polynomial features
 ↪outperforms the initial model in terms of accuracy, performance, and fit to
 ↪the data.
```

[ ]:
```
```

[ ]:
```
```

[ ]:
```
```