nalukanga winnie assignment due 25th march

March 19, 2024

[1]:
```
import pandas as pd
data = pd.read_csv("D:\\linear␣
 ↪programming\\Assignment\\Assignment\\student_scores_dataset.csv")
data.head()
```

[1]:

	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

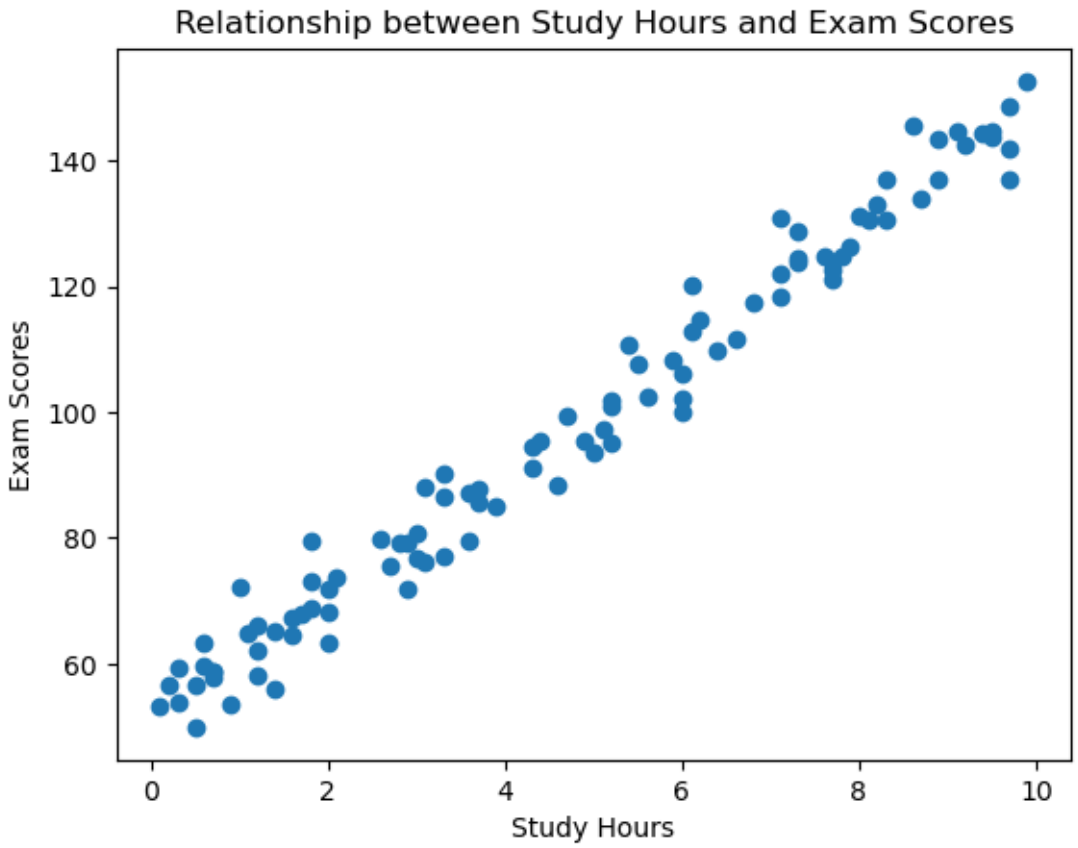
[3]:
```
X = data[['Study Hours']]
y = data['Exam Scores']
```

[4]:
```
# Check for missing values
data.isnull().sum()
```

[4]:
```
Study Hours    0
Exam Scores    0
dtype: int64
```

[4]:
```
import matplotlib.pyplot as plt

# Scatter plot
plt.scatter(data['Study Hours'], data['Exam Scores'])
plt.title('Relationship between Study Hours and Exam Scores')
plt.xlabel('Study Hours')
plt.ylabel('Exam Scores')
plt.show()
```



[7]:
```
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)
X_test.shape
```

[7]:
```
(20, 1)
```

[8]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[32]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Create and train the model
model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

[32]:
```
LinearRegression()
```

[33]:
```
# Make predictions
y_pred = model.predict(X_test_scaled)
y_pred
```

[33]:
```
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 ])
```

[30]:
```
# Evaluate metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mae
```

[30]:
```
2.9365732667749755
```

[11]:
```
mse
```

[11]:
```
16.202109700645348
```

[12]:
```
r2
```

[12]:
```
0.9826924926918468
```

[14]:
```
# Coefficients
model.coef_
```

[14]:
```
array([28.52556103])
```

[29]:
```
# Explore feature engineering techniques like polynomial features or␣
 ↪interaction terms
# Example: Adding a squared term for Study Hours
data['Study Hours Squared'] = data['Study Hours'] ** 2
data
```

[29]:

	Study Hours	Exam Scores	Study Hours Squared
0	3.7	87.9	13.69
1	9.5	143.6	90.25
2	7.3	123.7	53.29
3	6.0	99.9	36.00
4	1.6	64.5	2.56
..	…	…	…
95	4.9	95.3	24.01

```
96          5.2        101.9                27.04
97          4.3         94.5                18.49
98          0.3         53.9                 0.09
99          1.1         64.9                 1.21

[100 rows x 3 columns]
```

[17]:
```
# Update X and y
X = data[['Study Hours', 'Study Hours Squared']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[18]:
```
# Standardize
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[19]:
```
# Retrain the model
model.fit(X_train_scaled, y_train)
```

[19]:
```
LinearRegression()
```

[25]:
```
# Make predictions
y_pred_updated = model.predict(X_test_scaled)
y_pred_updated
```

[25]:
```
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 ])
```

[26]:
```
# Evaluate metrics
mae_updated = mean_absolute_error(y_test, y_pred_updated)
mse_updated = mean_squared_error(y_test, y_pred_updated)
r2_updated = r2_score(y_test, y_pred_updated)
mae_updated
```

[26]:
```
2.8285958469252686
```

[27]:
```
mse_updated
```

[27]:
```
15.64363843629915
```

[28]:
```
r2_updated
```

[28]:
```
0.9832890659571593
```