Student Score Prediction Model

Assignment

DATA SERIES 12.0
AI Machine Learning

Muhamad Abdul Qodir Dani

ibimbing

Table of contents

01

Data Understanding

02

Load Data

03

Exploratory Data Analysis & Feature Enginering

04

Data Splitting

05

Modelling Machine Learning

06

Model Evaluation

Dataset: https://drive.google.com/file/d/1tAww2rf2QEHKmq6bLNQA5TKWYnfOqC9x/view?usp=sharing

Data Understanding

student_scores.csv
Spreadsheet

This studen score data contains information about the total study time (in hours) and exam scores achieved by students. Each row represents one student, with two main columns, 'Hours' which shows the total hours studied, and 'Scores' which shows the exam scores achieved. Through this data, we can perform simple analysis and predictions to gain insights”

Load Data

1. import libraries and Dataset

```
# Import libraries and resources
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 7.7s

Import 5 libraries & Dataset from student_scores CSV with Pandas libraries as Dataframe

```
dataset = pd.read_csv('D:/Dibimbing AI&ML/Day 2/student_scores.csv')
dataset
```

✓ 0.1s

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

2. Describe the Data

```
dataset.describe()
```

[11] ✓ 0.0s

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

From the **dataset** there are **24 rows** and **2 columns.**

Explanatory Data Analysis & Feature Engineering

1. Check missing values and missing data all columns

```
dataset.isna().sum()
```

✓ 0.0s

```
Hours     0
Scores    0
dtype: int64
```

from the **data** after we check it, it turns out that there are **no missing values**

2. Check Duplicate

```
duplicate_rows_before = df[df.duplicated()]
print("Number of duplicate rows : ", duplicate_rows_before.shape)
```

✓ 0.0s

```
Number of duplicate rows :  (0, 2)
```

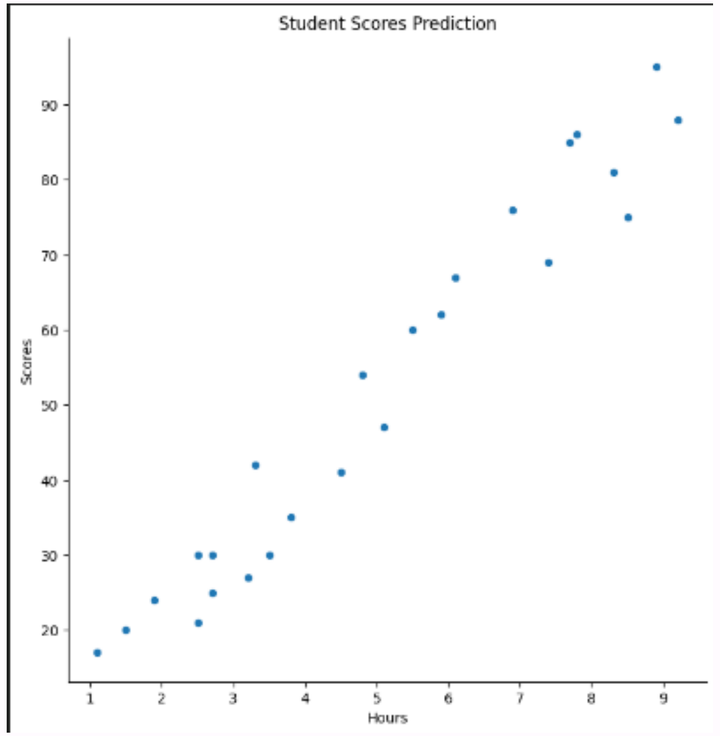
from the **data** after we check it, it turns out that there is **no duplicate data**

Explanatory Data Analysis & Feature Engineering

3. Student Scores Prediction

```
plt.figure(figsize=(12,6))
sns.pairplot(dataset,x_vars=['Hours'],y_vars=['Scores'], size=7, kind="scatter")
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.title('Student Scores Prediction')
plt.show()
```

✓ 0.4s



The graph shows the **relationship between students' number of study hours (Hours)** and **their test scores (Scores).** Based on the pattern of dots on the graph, there is a strong positive relationship between these two variables. **This means that the more time students spend studying, the higher their test scores.**

Explanatory Data Analysis & Feature Engineering

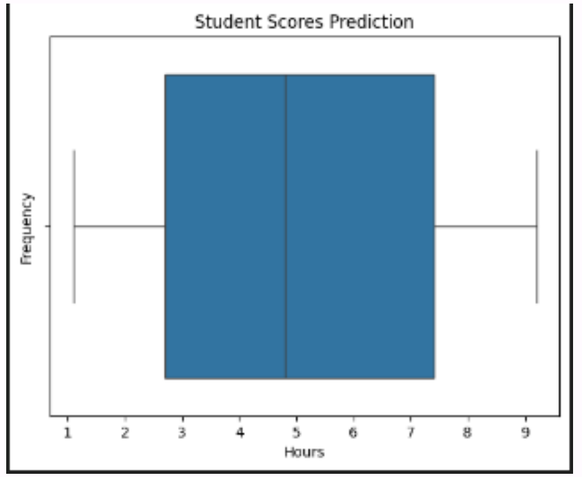
4. Outlier Analysis

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Outlier Analysis
sns.boxplot(x="Hours", data=df)

# Menambahkan label sumbu dan judul
plt.xlabel("Hours")
plt.ylabel("Frequency")  # Anda bisa menggunakan "Years of Experience" atau "Tahun Pengalaman"
plt.title("Student Scores Prediction")

# Menampilkan plot
plt.show()
```



In a **boxplot**, outliers are usually shown as points outside the whiskers. S**ince there are no points outside the whiskers on this graph**, it can be concluded that **there are no significant outliers in the distribution of the hours of study data**. That is, all study hour values are within a reasonable range and there are no extreme or anomalous values.

Data Splitting

1. Import libraries for splitting data

```
# Import machine learning data from scikit learn
from sklearn.model_selection import train_test_split
```

✓ 1.4s

2. Data Splitting

```
# Split the data for train and test
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.75,random_state=42)
```

✓ 0.0s

We **split the dataset** into **two parts**: one part **for training** the model **(75%)** and one part **for testing** the model's performance (**25%).**

Modelling Machine Learning

Transformasi data

```
# Create new axis for x column in array
import numpy as np
# Convert X_train and X_test to NumPy arrays
X_train = X_train.to_numpy()
X_test = X_test.to_numpy()

# Add new axis to the end
X_train = X_train[:, np.newaxis]
X_test = X_test[:, np.newaxis]
```

0.0s

By adding a new dimension to the NumPy array, we can change the shape of the data to fit the requirements of machine learning models or other mathematical operations. This is a common data preparation step before training a model.

Modelling Machine Learning

1. Linear Regression model

Import libraries

```
# Importing Linear Regression model from scikit learn
from sklearn.linear_model import LinearRegression
```

✓ 0.5s

Build model

```
# Fitting the model using Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train,y_train)
```

✓ 0.0s

Modelling Machine Learning

1. Linear Regression model

Predicting the scores for the Test values

```
# Predicting the Scores for the Test values
y_pred = lr_model.predict(X_test)
```

0.0s

```
y_pred
```

0.0s

```
array([83.10733229, 26.76559757, 26.76559757, 69.50760322, 59.79351103,
       28.70841601, 34.53687133])
```

Modelling Machine Learning

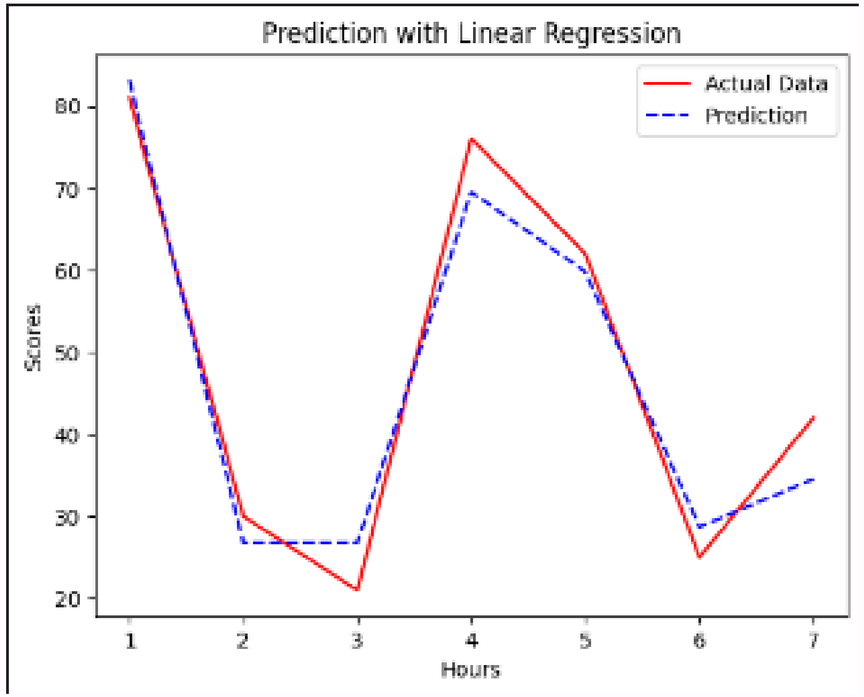
1. Linear Regression model

Plotting the actual and predited values

```
# Plotting the actual and predicted values

c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='r',linestyle='-',label='Actual Data')
plt.plot(c,y_pred,color='b',linestyle='dashed',label='Prediction')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.title('Prediction with Linear Regression')
plt.legend()
plt.show()
```

0.4s



Modelling Machine Learning

2. Decision Tree Regressor

Import libraries

```
from sklearn.tree import DecisionTreeRegressor
```
✓ 0.4s

Build model

```
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
```
✓ 0.0s

DecisionTreeRegressor ⓘ ?

```
DecisionTreeRegressor()
```

Modelling Machine Learning

2. Decision Tree Regressor

Predicting the scores for the Test values

```
# Predicting the Scores for the Test values
y_pred_dt = dt_model.predict(X_test)
```

0.0s

```
y_pred_dt
```

0.0s

```
array([75., 30., 30., 69., 67., 30., 27.])
```

Modelling Machine Learning

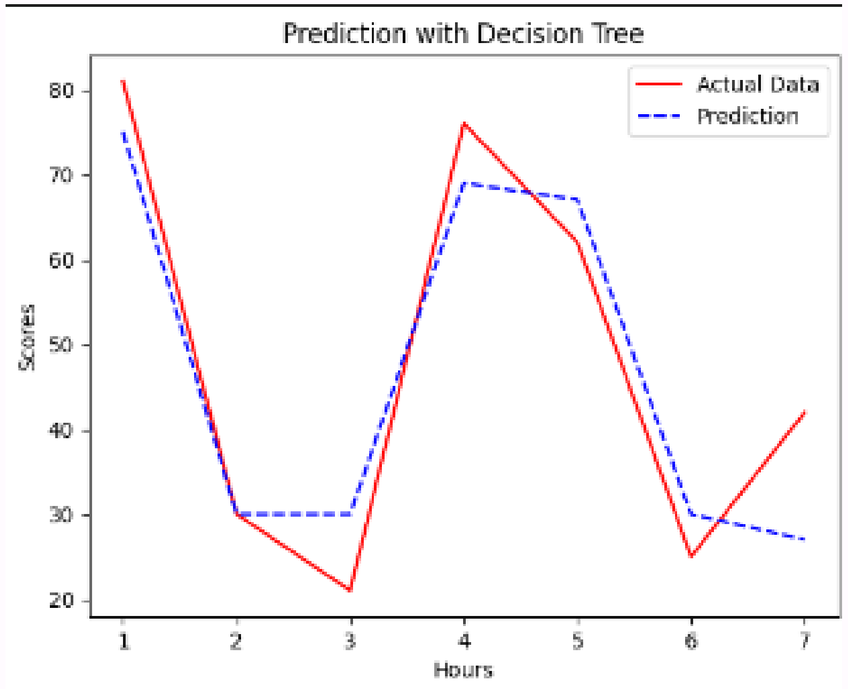
2. Decision Tree Regressor

Plotting the actual and predited values

```
# Plotting the actual and predicted values

c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='r',linestyle='-',label='Actual Data')
plt.plot(c,y_pred_dt,color='b',linestyle='dashed',label='Prediction')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.title('Prediction with Decision Tree')
plt.legend()
plt.show()
```

✓ 0.1s



Modelling Machine Learning

3. Random Forest Regressor

Import libraries

```
from sklearn.ensemble import RandomForestRegressor
model_rf = RandomForestRegressor()
```

✓ 0.4s

Build model

```
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
```

✓ 0.1s

RandomForestRegressor ❶ ❷

RandomForestRegressor()

Modelling Machine Learning

3. Random Forest Regressor

Predicting the scores for the Test values

```
# Predicting the Scores for the Test values
y_pred_rf = rf_model.predict(X_test)
```

✓ 0.0s

```
y_pred_rf
```

✓ 0.0s

```
array([79.79, 28.19, 28.19, 74.58, 63.55, 29.41, 28.56])
```

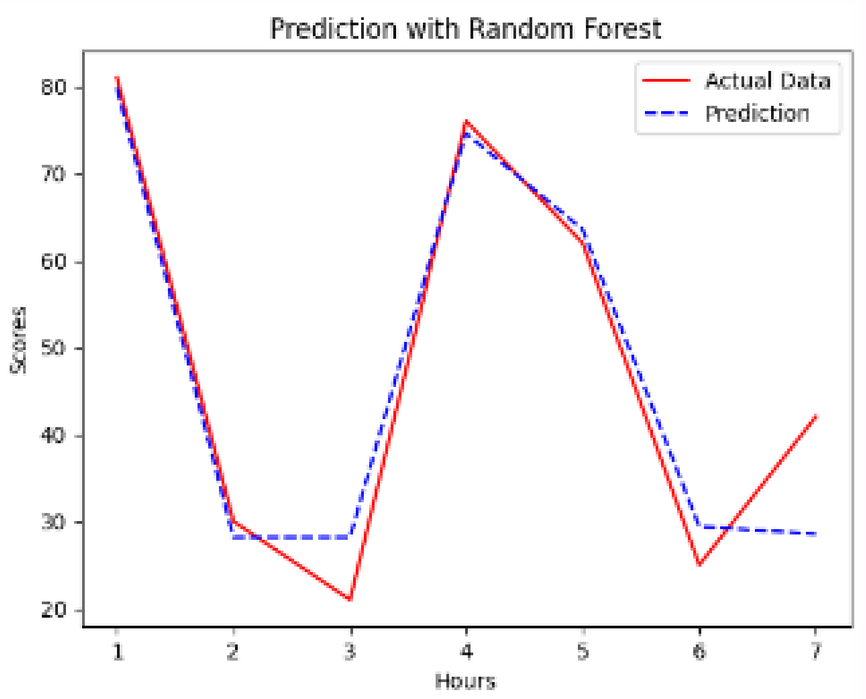
Modelling Machine Learning

2. Decision Tree Regressor

Plotting the actual and predited values

```
# Plotting the actual and predicted values

c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='r',linestyle='-',label='Actual Data')
plt.plot(c,y_pred_rf,color='b',linestyle='dashed',label='Prediction')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.title('Prediction with Random Forest')
plt.legend()
plt.show()
```



Model Evaluation

Coefficient of determination (R-squared) values of three different models

```
print('r square Linear Regression:',rsq)
print('r square Decision Tree Results:',rsq_dt)
print('r square Random Forrest Results:',rsq_rf)
```

✓ 0.0s

```
r square Linear Regression: 0.9553509219739938
r square Decision Tree Results: 0.8803859268443893
r square Random Forrest Results: 0.9292251123682579
```

The **Linear Regression** model is the model that best explains the variability of the data in this case.

The **Random Forest model** also performed very well, and may be an alternative if there are other considerations such as model interpretation or handling non-linear data.

The **Decision Tree model** performed reasonably well, but not as well as the other two models.

Thank You

Muhamad Abdul Qodir Dani

mohamadkodir12@gmail.com

Muhamad Abdul Qodir Dani