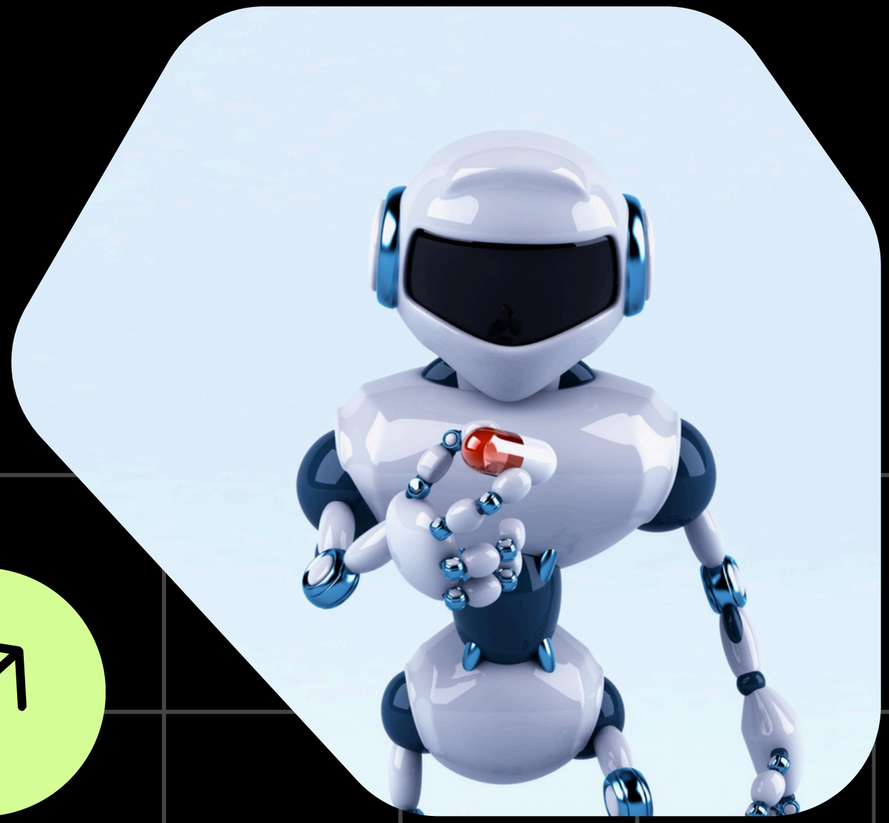
ML Modelling Regression

FINDING THE BEST ML MODEL TO PREDICT STUDENTS' SCORES BASED ON THE NUMBER OF STUDY HOURS



dibimbing.id

About Me

Hello! I’m Raul, a final-year senior high school student who likes to explore and learn many things about Technology, AI, and Robotics.

Current Education :

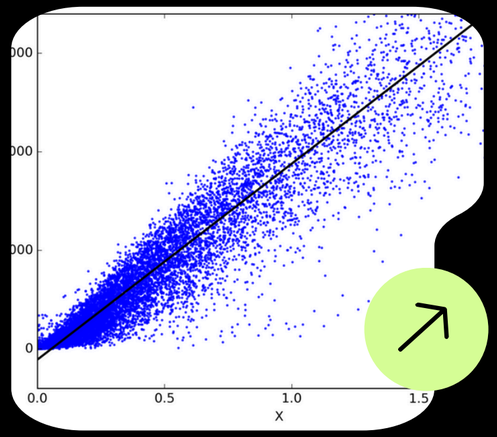
SMA NEGERI 1 BOBOTSARI

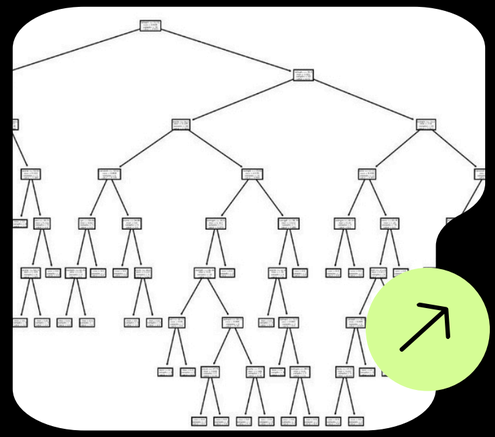
UNDERSTANDING

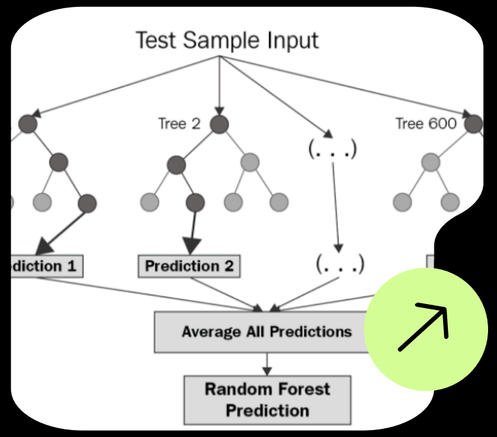
Project Overview

Through this mini project, I trained some ML models from Scikit-learn to predict students' scores based on the number of study hours. Besides this, there are the three ML models that I used.

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor







LINEAR REGRESSION

This model predicts a continuous target variable by fitting a linear relationship between the target and one or more predictor variables. It assumes a straight-line relationship and estimates coefficients to minimize the difference between predicted and actual values.

DECISION TREE REGRESSOR

This model predicts continuous values by splitting the data into subsets based on feature values, forming a tree-like structure. Each node represents a decision rule, and the leaves contain the average target values for the data falling into that subset.

RANDOM FOREST REGRESSOR

This model improves prediction accuracy by constructing multiple decision trees (a "forest") and averaging their predictions. It reduces overfitting and variance compared to individual decision trees by aggregating the results from several trees trained on different data subsets.

UNDERSTANDING

Data Overview

The dataset used in this mini project is a table consisting of two columns, Hours and Scores. Each column consists of 25 rows containing each pair of Hour and Score data of a student.

index	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

TOOL

WorkFlow

01 Preparation

02 Exploratory Data Analysis (EDA)

03 Feature Engineering

WorkFlow

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


data = pd.read_csv('student_scores.csv')
```

PREPARATION

Import the libraries and dependencies needed to start the mini project. The libraries and dependencies needed are Pandas to read and process data, NumPy to convert data into a NumPy array, and other supporting dependencies to visualize the data.

EDA (EXPLORATORY DATA ANALYSIS)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

`data.info()`

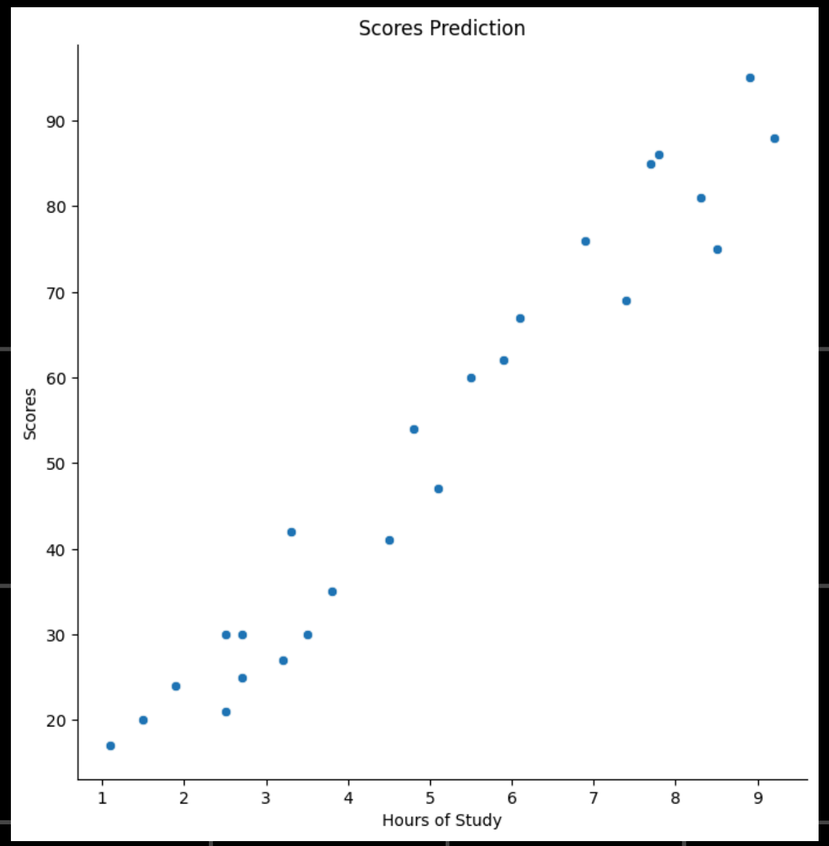
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

`data.describe()`

		count
Hours	Scores	
1.1	17	1
5.1	47	1
8.9	95	1
8.5	75	1
8.3	81	1
7.8	86	1
7.7	85	1
7.4	69	1
6.9	76	1
6.1	67	1
5.9	62	1
5.5	60	1
4.8	54	1
1.5	20	1
4.5	41	1
3.8	35	1
3.5	30	1

`data.value_counts()`

EDA (EXPLORATORY DATA ANALYSIS)




```
plt.figure(figsize=(12,6))
sns.pairplot(data,x_vars=["Hours"].
plt.xlabel("Hours of Study")
plt.ylabel("Scores")
plt.title("Scores Prediction")
plt.show()
```

We can visualize the data by using a library named matplotlib.pyplot (as "plt" variable in this project)

FEATURE ENGINEERING

Check Missing Value Handling

	0
Hours	0
Scores	0

dtype: int64

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

```
data.info()
```

		count
Hours	Scores	
1.1	17	1
5.1	47	1
8.9	95	1
8.5	75	1
8.3	81	1
7.8	86	1
7.7	85	1
7.4	69	1
6.9	76	1
6.1	67	1
5.9	62	1
5.5	60	1
4.8	54	1
1.5	20	1
4.5	41	1
3.8	35	1
3.5	30	1

```
data.value_counts()
```

FEATURE ENGINEERING

Check Duplicated Data

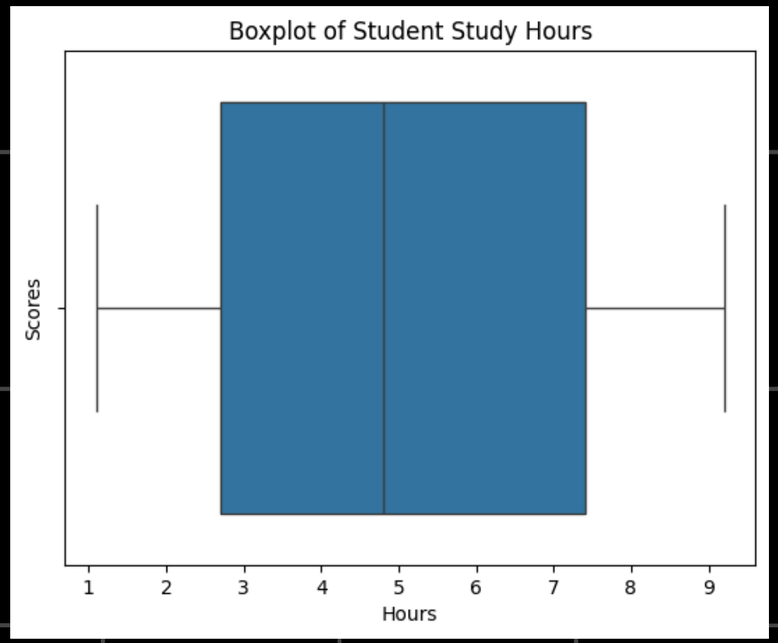
```
df = data.copy()
duplicate_rows_before = df[df.duplicated()]
duplicate_rows_before

# Output :
```

Hours	Scores

This means there is no duplicated data in the Dataframe

FEATURE ENGINEERING

Outlier Analysis




```
sns.boxplot(x="Hours", data=df)

plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Boxplot of Student Study
Hours")

plt.show()
```

There are no unusual dots here outside the blue box. So, there are no any outliers in the Dataframe

ML MODEL TRAINING

PREPARATION - SPLITTING THE DATASET

```
[ ] X = data["Hours"]
    X.head()
```

	Hours
0	2.5
1	5.1
2	3.2
3	8.5
4	3.5

dtype: float64

Declare and assign a variable named “X” with the “Hours” column of the datasset

```
[ ] y = data["Scores"]
    y.head()
```

	Scores
0	21
1	47
2	27
3	75
4	30

dtype: int64

Declare and assign a variable named “y” with the “Scores” column of the datasset

ML MODEL TRAINING

PREPARATION

```
# import the dependencies needed
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=4

X_train = np.array(X_train)[:, np.newaxis]
X_test = np.array(X_test)[:, np.newaxis]
```

- Import the libraries and dependencies needed for splitting the dataframe
- I split the X (Hours column data) into 50% for testing, and 50% for training

I also import the “LinearRegression” to prepare the training for the first model

ML MODEL TRAINING

LINEAR REGRESSION

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

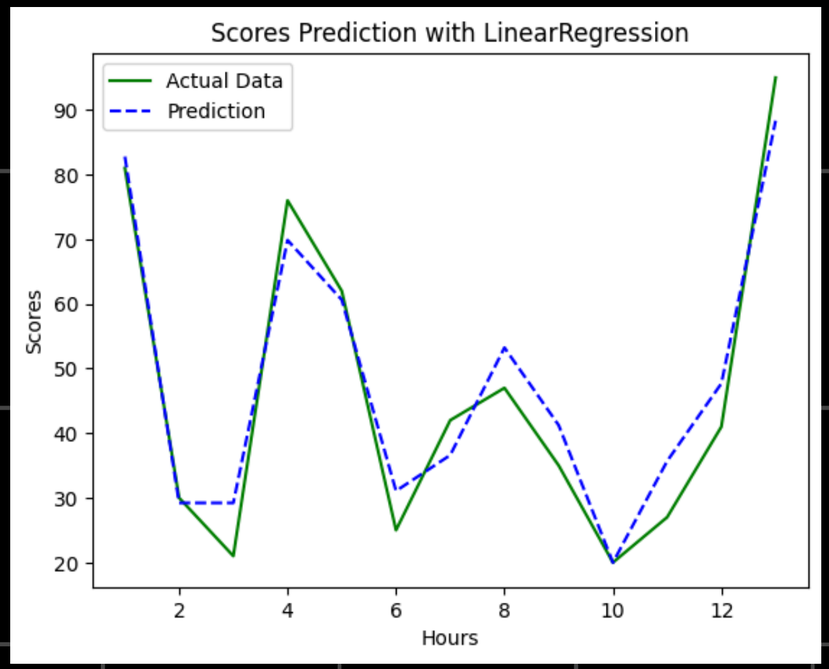
Declare the model in a variable and train it

```
y_pred = lr_model.predict(X_test)
```

Let the model predict the testing data

ML MODEL TRAINING

LINEAR REGRESSION - TRAINING RESULT




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

ML MODEL TRAINING

DECISION TREE REGRESSOR

```
from sklearn.tree import
DecisionTreeRegressor
```

Import the model

```
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
```

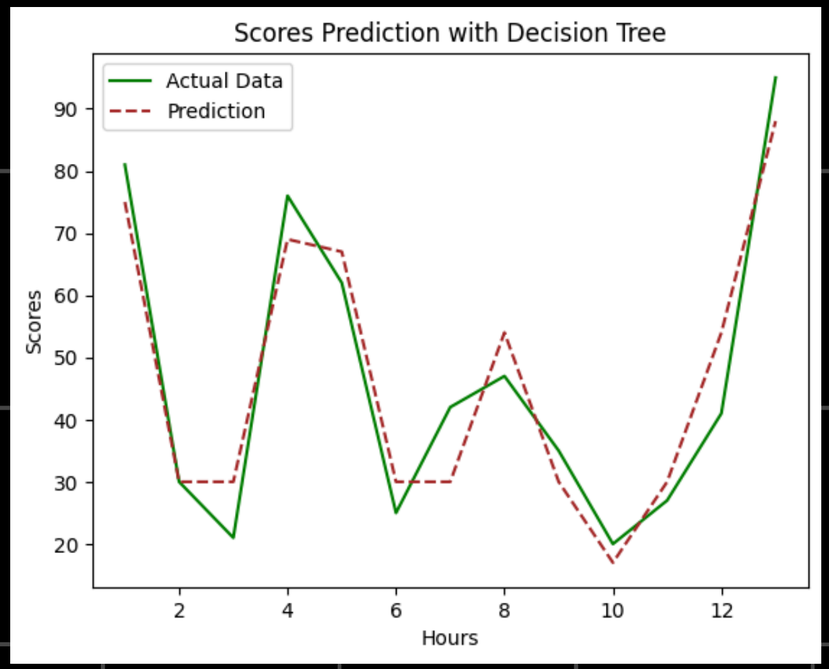
Declare the model in a variable and train it

```
y_pred_dt = dt_model.predict(X_test)
```

Let the model predict the testing data

ML MODEL TRAINING

DECISION TREE REGRESSOR - TRAINING RESULT




```
# Plotting the Actual Data and the Prediction
c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='g',linestyle='-',lab
el='Actual Data')
plt.plot(c,y_pred_dt,color='brown',linestyle=
'--',label='Prediction')
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Scores Prediction with Decision
Tree")
plt.legend()
plt.show()
```

ML MODEL TRAINING

RANDOM FOREST REGRESSOR

```
from sklearn.ensemble import
RandomForestRegressor
```

Import the model

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

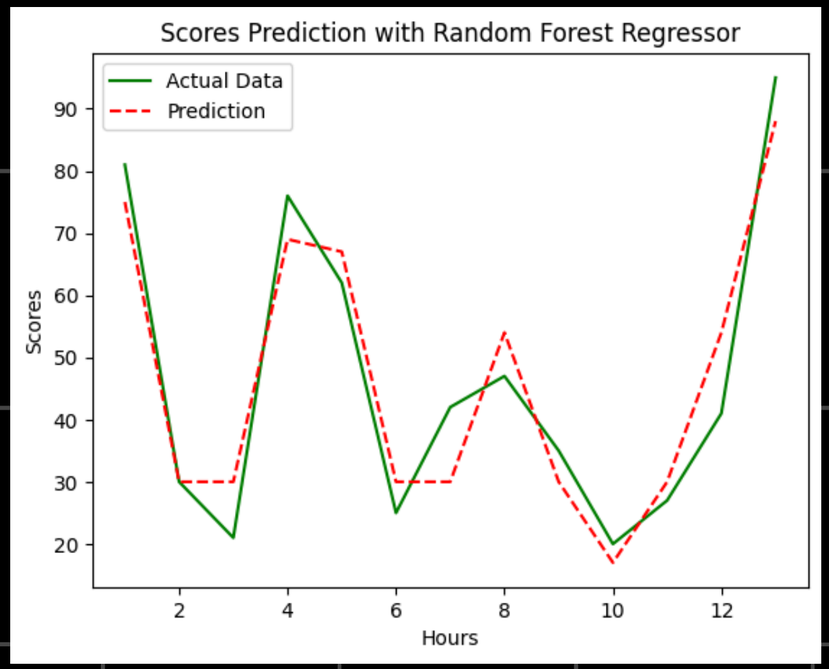
Declare the model in a variable and train it

```
y_pred_rf = rf_model.predict(X_test)
```

Let the model predict the testing data

ML MODEL TRAINING

RANDOM FOREST REGRESSOR - TRAINING RESULT




```
# Plotting the Actual Data and the Prediction
c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='g',linestyle='-',lab
el='Actual Data')
plt.plot(c,y_pred_dt,color='r',linestyle='--'
,label='Prediction')
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Scores Prediction with Random
Forest Regressor")
plt.legend()
plt.show()
```

MODEL EVALUATION

ALL THE THREE ML MODELS

```
from sklearn.metrics import
r2_score, mean_squared_error
```

Import the scoring system libraries. Here, I used R-Squared Scoring

```
rsq = r2_score(y_test, y_pred)
rsq_dt = r2_score(y_test, y_pred_dt)
rsq_rf = r2_score(y_test, y_pred_rf)
```

Declare 3 variables and assign them with the score of each model

```
print(LinearRegression, " : ", rsq)
print(DecisionTreeRegressor, " : ", rsq_dt)
print(RandomForestRegressor, " : ", rsq_rf)
```

Print the results of the models' scores

CONCLUSION

R-SQUARED RESULTS FROM THE THREE ML MODELS

```
<class 'sklearn.linear_model._base.LinearRegression'>   :   0.9426307007429557
<class 'sklearn.tree._classes.DecisionTreeRegressor'>   :   0.9082539816297295
<class 'sklearn.ensemble._forest.RandomForestRegressor'>   :   0.9326361022162298
```

From the results, we already have the winner! The **LinearRegression** model is the best model to predict students' score based on the study hours than the other 2 models.

Contact Me

+62 823 2838 2750

raul.maulidhino@gmail.com

https://raul.maulidhino.pages.dev

Purbalingga, Central Java, Indonesia