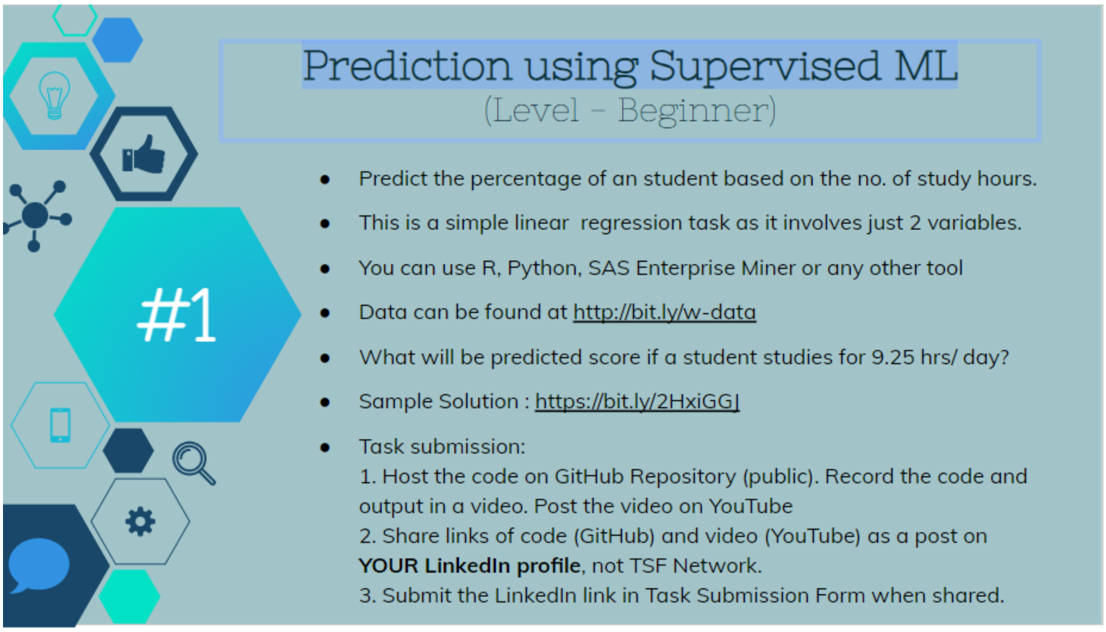
Pridiction using Supervised ML



Prediction using Supervised ML

(Level - Beginner)

- Predict the percentage of an student based on the no. of study hours.
- This is a simple linear regression task as it involves just 2 variables.
- You can use R, Python, SAS Enterprise Miner or any other tool
- Data can be found at http://bit.ly/w-data
- What will be predicted score if a student studies for 9.25 hrs/ day?
- Sample Solution : https://bit.ly/2HxiGGJ
- Task submission:
  1. Host the code on GitHub Repository (public). Record the code and output in a video. Post the video on YouTube
  2. Share links of code (GitHub) and video (YouTube) as a post on **YOUR LinkedIn profile**, not TSF Network.
  3. Submit the LinkedIn link in Task Submission Form when shared.

Importing Libraries

```
In [ ]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
```

Read the Datasets

```
In [ ]: df = pd.read_csv("Datasets.txt")
```

```
In [ ]: df
```

```
Out[ ]:
        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
```

Describe()

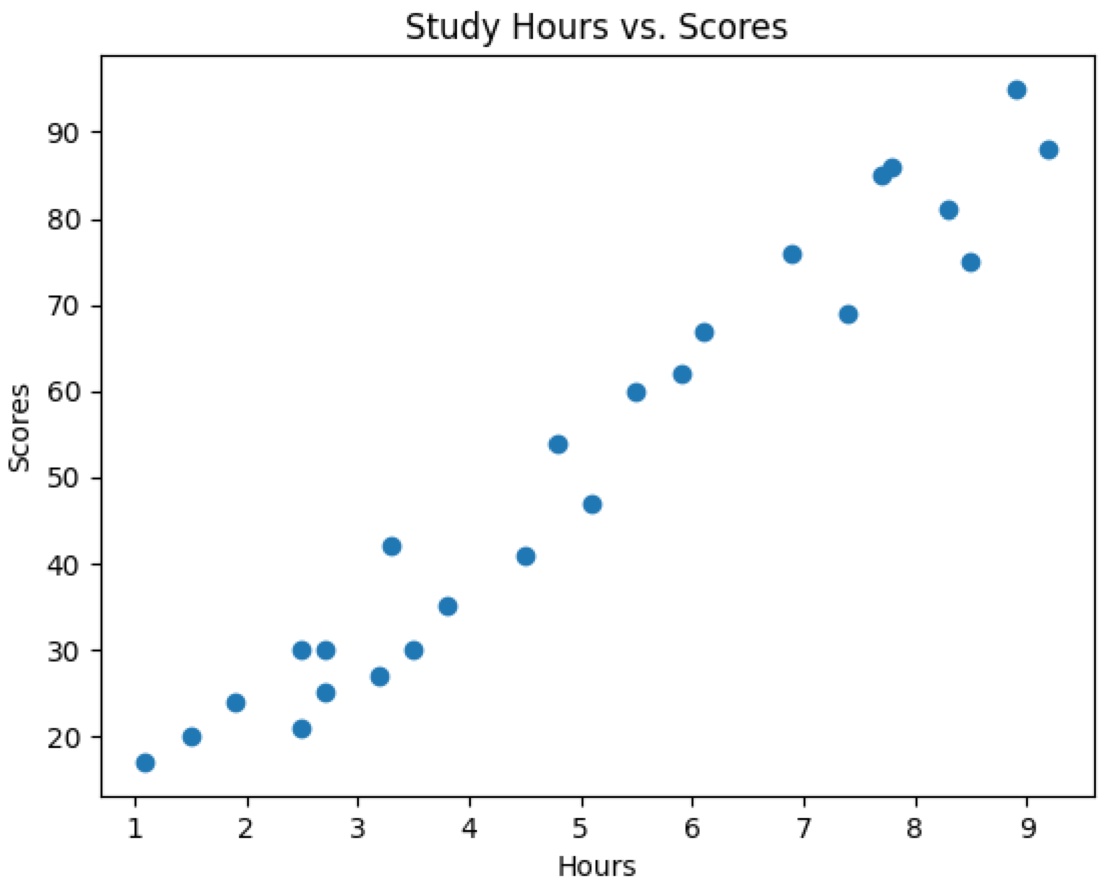
```
In [ ]: df.describe().T
```

```
Out[ ]:
          count    mean        std   min   25%   50%   75%   max
   Hours   25.0   5.012   2.525094   1.1   2.7   4.8   7.4   9.2
  Scores   25.0  51.480  25.286887  17.0  30.0  47.0  75.0  95.0
```

Createing the Scatter plot

In [ ]:

```
# Scatter plot of study hours vs. scores
plt.scatter(df['Hours'], df['Scores'])
plt.title('Study Hours vs. Scores')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.show()
```



Data Preparation

Split the data into training and testing sets:

In [ ]:

```
X = df['Hours'].values.reshape(-1, 1)  # Independent variable (study hours)
y = df['Scores'].values  # Dependent variable (scores)
```

In [ ]:

```
X
```

```
Out[ ]: array([[2.5],
               [5.1],
               [3.2],
               [8.5],
               [3.5],
               [1.5],
               [9.2],
               [5.5],
               [8.3],
               [2.7],
               [7.7],
               [5.9],
               [4.5],
               [3.3],
               [1.1],
               [8.9],
               [2.5],
               [1.9],
               [6.1],
               [7.4],
               [2.7],
               [4.8],
               [3.8],
               [6.9],
               [7.8]])
```

```
In [ ]: y
```

```
Out[ ]: array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 30,
               24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

```
In [ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

```
In [ ]: X_train
```

```
Out[ ]: array([[2.7],
               [3.3],
               [5.1],
               [3.8],
               [1.5],
               [3.2],
               [4.5],
               [8.9],
               [8.5],
               [3.5],
               [2.7],
               [1.9],
               [4.8],
               [6.1],
               [7.8],
               [5.5],
               [7.7],
               [1.1],
               [7.4],
               [9.2]])
```

```
In [ ]: X_test
```

Out[ ]:
```
array([[8.3],
       [2.5],
       [2.5],
       [6.9],
       [5.9]])
```

In [ ]:
```
y_train
```

Out[ ]:
```
array([25, 42, 47, 35, 20, 27, 41, 95, 75, 30, 30, 24, 54, 67, 86, 60, 85,
       17, 69, 88], dtype=int64)
```

In [ ]:
```
y_test
```

Out[ ]:
```
array([81, 30, 21, 76, 62], dtype=int64)
```

Train the Linear Regression Model

Create and train a simple linear regression model:

In [ ]:
```
# Create a linear regression model
model = LinearRegression()
```

In [ ]:
```
# Train the model on the training data
model.fit(X_train, y_train)
```

Out[ ]:
```
▾ LinearRegression
LinearRegression()
```

Make Predictions

Use the trained model to make predictions on the test data and also predict the score for a student who studies for 9.25 hours:

In [ ]:
```
# Predict scores on the test data
y_pred = model.predict(X_test)
y_pred
```

Out[ ]:
```
array([83.18814104, 27.03208774, 27.03208774, 69.63323162, 59.95115347])
```

In [ ]:
```
# Predict the score for a student who studies for 9.25 hours
hours_to_predict = np.array([[9.25]])
predicted_score = model.predict(hours_to_predict)
predicted_score
```

Out[ ]:
```
array([92.38611528])
```

In [ ]:
```
print("Predicted Score for 9.25 hours of study:", predicted_score[0])
```

```
Predicted Score for 9.25 hours of study: 92.38611528261494
```

Evaluate the Model

You should evaluate the model's performance using appropriate metrics (e.g., Mean Absolute Error, Mean Squared Error, R-squared):

In [ ]:

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

# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [ ]:

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

```
Mean Absolute Error: 3.9207511902099244
Mean Squared Error: 18.943211722315272
R-squared: 0.9678055545167994
```