Task 1: Simple Linear Regression

In this regression task we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables.

Importing all libraries required

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
%matplotlib inline
```

```
# Reading data from remote link
url = "http://bit.ly/w-data"
train = pd.read_csv(url)
print("Data imported successfully")
train.head(10)
```

```
Data imported successfully
```

	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

```
#shape of dataset
train.shape
```

```
(25, 2)
```

```
#Information in Dataset
train.info()
```

```
<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
```

```
train.describe()
```

Data Visualization

```
# Plotting the distribution of scores
train.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

Linear Regression Model

```
X = train.iloc[:, :-1].values
y = train.iloc[:, 1].values
X_train, X_test, y_train, y_test = train_test_split(X, y,train_size=0.80,test_size=0.20
```

Training the Model

```
from sklearn.linear_model import LinearRegression
linearRegressor = LinearRegression()
linearRegressor.fit(X_train, y_train)
y_predict= linearRegressor.predict(X_train)
```

Training the Algorithm

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
print("Training complete.")
```

```
Training complete.
```

```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_
# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line);
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

Checking the accuracy scores for training and test

```
print('Test Score')
print(regressor.score(X_test, y_test))
print('Training Score')
print(regressor.score(X_train, y_train))
```

```
Test Score
0.9678055545167994
Training Score
0.9491209376364416
```

```
y_test
```

```
array([81, 30, 21, 76, 62])
```

```
y_predict
```

```
array([28.96850337, 34.77775026, 52.20549094, 39.61878934, 17.35000959,
       33.80954245, 46.39624405, 88.99738793, 85.12455667, 36.71416589,
       28.96850337, 21.22284085, 49.3008675 , 61.8875691 , 78.34710196,
       56.0783222 , 77.37889414, 13.47717832, 74.4742707 , 91.90201137])
```

```
y_predict[:5]
```

```
array([28.96850337, 34.77775026, 52.20549094, 39.61878934, 17.35000959])
```

```
data= pd.DataFrame({'Actual': y_test,'Predicted': y_predict[:5]})
data
```

```
#Let's predict the score for 9.25 hpurs
```

```
print('Score of student who studied for 9.25 hours a dat', regressor.predict([[9.25]]))
```

```
Score of student who studied for 9.25 hours a dat [92.38611528]
```

Model Evaluation Metrics

```
#Checking the efficiency of model
mean_squ_error = mean_squared_error(y_test, y_predict[:5])
mean_abs_error = mean_absolute_error(y_test, y_predict[:5])
print("Mean Squred Error:",mean_squ_error)
print("Mean absolute Error:",mean_abs_error)
```

```
Mean Squred Error: 1404.2200673968694
Mean absolute Error: 33.80918778157651
```

✓ 0s completed at 7:07 PM