

Not True

• [File](#)

- [Python3](#)
- [Open...](#)
- [Make a Copy...](#)
- [Save As...](#)
- [Rename...](#)
- [Save and Checkpoint Ctrl+S](#)
- [Revert to Checkpoint Dronedown](#)
  - [Wednesday, February 17, 2021 12:53 AM](#)
- [Print Preview](#)
- [Download as Dronedown](#)
  - [AsciiDoc \(.asciidoc\)](#)
  - [HTML \(.html\)](#)
  - [LaTeX \(.tex\)](#)
  - [Markdown \(.md\)](#)
  - [Notebook \(.ipynb\)](#)
  - [PDF via LaTeX \(.pdf\)](#)
  - [reST \(.rst\)](#)
  - [Python \(.py\)](#)
  - [Prezi \(.slides / slides.html\)](#)
  - [PDF via pypptree \(.html\)](#)
- [Dronedown](#)
- [Trust Notebook](#)
- [Close and Halt](#)
- [Edit](#)
  - [Cut Cells](#)
  - [Copy Cells](#)
  - [Paste Cells Above/Shift+V](#)
  - [Paste Cells Below+V](#)
  - [Paste Cells & Replace](#)
  - [Delete Cells+D](#)
  - [Undo Delete Cells+Z](#)
  - [Split Cells Ctrl+Shift+Minus](#)
  - [Merge Cell Above](#)
  - [Merge Cell Below](#)
  - [Move Cell Up](#)
  - [Move Cell Down](#)
  - [Edit Notebook Metadata](#)
  - [Find and Replace](#)
  - [Cut Cell Attachments](#)
  - [Copy Cell Attachments](#)
  - [Paste Cell Attachments](#)
  - [Insert Image](#)
- [View](#)
  - [Toggle Header](#)
  - [Toggle Toolbar](#)
  - [Toggle Line Numbers/Shift+L](#)
  - [Cell Toolbar](#)
    - [None](#)
    - [Edit Metadata](#)
    - [Raw Cell Format](#)
    - [Slideshow](#)
    - [Attachments](#)
    - [Tags](#)
- [Insert](#)
  - [Insert Cell Above+O](#)
  - [Insert Cell Below+I](#)
- [Cell](#)
  - [Run Cells Ctrl+Enter](#)
  - [Run Cells and Select Below/Shift+Enter](#)
  - [Run Cells and Insert Below/Alt+Enter](#)
  - [Run All](#)
  - [Run All Above](#)
  - [Run All Below](#)
  - [Cell Type](#)
    - [Code](#)
    - [Markdown](#)
    - [Raw NBConvent](#)
  - [Current Outputs](#)
    - [Toggle](#)
    - [Toggle Scrollingshift+O](#)
    - [Clear](#)
  - [All Output](#)
    - [Toggle](#)
    - [Toggle Scrolling](#)
    - [Clear](#)
- [Kernel](#)
  - [Interrupt+I](#)
  - [Restart+R](#)
  - [Restart & Clear Output](#)
  - [Restart & Run All](#)
  - [Reconnect](#)
  - [Shutdown](#)
  - [Change kernel](#)
    - [Python 3](#)
- [Widgets](#)
  - [Save Notebook Widget State](#)
  - [Clear Notebook Widget State](#)
  - [Download Widget State](#)
  - [Embed Widgets](#)
- [Help](#)
  - [User Interface Tour](#)
  - [Keyboard Shortcuts](#)
  - [Edit Keyboard Shortcuts](#)
  - [Notebook Help](#)
  - [Markdown](#)
  - [Python Reference](#)
  - [IPython Reference](#)
  - [NumPy Reference](#)
  - [SciPy Reference](#)
  - [Matplotlib Reference](#)
  - [SymPy Reference](#)
  - [pandas Reference](#)
  - [About](#)

```
## Task 1 - Prediction using Supervised ML

### **Submitted by - Rishika Rai**

### **Simple Linear Regression**

Task 1 - Prediction using Supervised ML

Submitted by - Rishika Rai👤

Simple Linear Regression📄

xxxxxxxxxxxx

### Importing the Required Libraries

Importing the Required Libraries📄

In [1]:

xxxxxxxxxxxx

import pandas as pd
```

```
import seaborn as sns
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
xxxxxxxxxxxx

## Read the data
Read the Data

In [2]:
xxxxxxxxxxxx

DataFrame = pd.read_csv("http://bit.ly/w-data")
In [3]:
xxxxxxxxxxxx

DataFrame.head()
Out[3]:


| Hours | Score |
|-------|-------|
| 0.25  | 21    |
| 1.5   | 47    |
| 3.2   | 27    |
| 3.5   | 75    |
| 4.5   | 30    |


In [4]:
xxxxxxxxxxxx

DataFrame.tail()
Out[4]:


| Hours | Score |
|-------|-------|
| 20.2  | 7     |
| 21.4  | 54    |
| 22.3  | 35    |
| 23.6  | 76    |
| 24.7  | 86    |


In [5]:
xxxxxxxxxxxx

DataFrame.shape
Out[5]:
(25, 2)

xxxxxxxxxxxx

## Lets check the data type
Lets check the data type

In [6]:
xxxxxxxxxxxx

DataFrame.dtypes
Out[6]:
Hours      float64
Scores     int64
dtype: object

In [7]:
xxxxxxxxxxxx

DataFrame.count()
Out[7]:
Hours      25
Scores     25
dtype: int64

xxxxxxxxxxxx

## Let's check the summary of data
Lets check the summary of data

In [8]:
xxxxxxxxxxxx

DataFrame.info()

# summary of data
<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

In [9]:
xxxxxxxxxxxx

DataFrame.describe().T
Out[9]:


|               | count | mean   | std       | min  | 25%  | 50%  | 75%  |
|---------------|-------|--------|-----------|------|------|------|------|
| <b>Scores</b> | 25.0  | 51.480 | 25.286887 | 17.0 | 30.0 | 47.0 | 75.0 |


xxxxxxxxxxxx
```

```

Total Unique Value
In [10]:
XXXXXXXXXXXX

DataFrame.nunique()
Out[10]:
Hours      23
Scores     23
dtype: int64

XXXXXXXXXXXX

## Missing Value
Missing Value
In [11]:
XXXXXXXXXXXX

DataFrame.isnull().sum()
Out[11]:
Hours      0
Scores     0
dtype: int64

XXXXXXXXXXXX

As we see, there is no missing value in the dataset.
As we see, there is no missing value in the dataset.

XXXXXXXXXXXX

## Data Visualizing
Data Visualizing
In [12]:
XXXXXXXXXXXX

sns.set_style("whitegrid")

sns.countplot(x = 'Hours', data = DataFrame)
plt.xlabel('Hours')
plt.ylabel('Count')
plt.title('Count Plot')
Out[12]:
Text(0.5, 1.0, 'Count Plot')


In [13]:
XXXXXXXXXXXX

sns.set_style("whitegrid")

sns.countplot(x = 'Scores', data = DataFrame)
plt.xlabel('Scores')
plt.ylabel('Count')
plt.title('Count Plot')
Out[13]:
Text(0.5, 1.0, 'Count Plot')


In [14]:
XXXXXXXXXXXX

sns.scatterplot(x = 'Hours', y = 'Scores', data = DataFrame)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()



From the graph above, we can clearly see that there is a positive correlation between hours studied and percentage score.
From the graph above, we can clearly see that there is a positive correlation between hours studied and percentage score.

XXXXXXXXXXXX

## **Preparing the data**
Preparing the data
In [15]:
XXXXXXXXXXXX

X = DataFrame.iloc[:, :-1].values

```

```

# Splitting the data into train & test data
Splitting the data into train & test data

In [16]:
XXXXXXXXXX

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)
XXXXXXXXXX

Perform Linear Regression

In [17]:
XXXXXXXXXX

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)

print("Training complete.")
Training complete.

XXXXXXXXXX

Model Evaluation

Model Evaluation

In [18]:
XXXXXXXXXX

slope = regressor.coef_
slope
Out[18]:
array([9.91065648])
In [19]:
XXXXXXXXXX

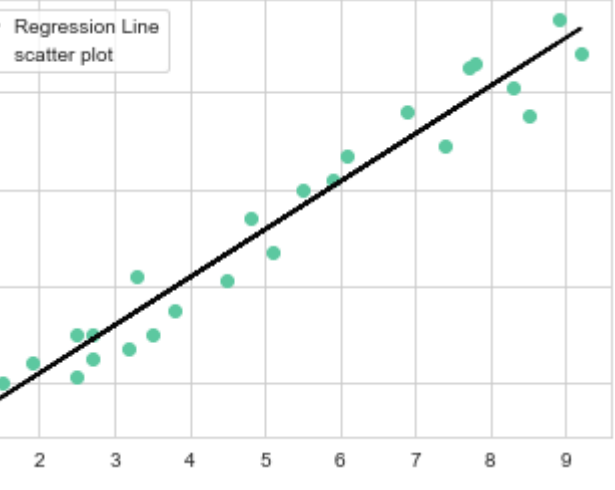
intercept = regressor.intercept_
intercept
Out[19]:
2.01816041434683
XXXXXXXXXX

Plotting the Regression Line

Plotting the Regression Line

In [20]:
XXXXXXXXXX

line = slope*X+intercept

plt.scatter(X, Y,color = "#CC3A66",label = "scatter plot")
plt.plot(X, line,color = "#080808",label = "Regression line");
plt.legend()
plt.show()

XXXXXXXXXX

Making Predictions for Scores

In [21]:
XXXXXXXXXX

y_pred = regressor.predict(x_test)

y_pred
Out[21]:
array([16.8841476, 33.73226078, 75.357018 , 26.7948124, 60.4910328])
In [22]:
XXXXXXXXXX

Comparing Actual vs Predicted

DataFrame = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
DataFrame
Out[22]:


|   | Actual | Predicted |
|---|--------|-----------|
| 0 | 20     | 16.884145 |
| 1 | 27     | 23.732261 |
| 2 | 69     | 75.357018 |
| 3 | 30     | 26.794801 |
| 4 | 62     | 60.491033 |


XXXXXXXXXX

Predicting Scores at 9.25 hours

Predicting Scores at 9.25 hours

In [23]:
XXXXXXXXXX

pred = regressor.predict([[9.25]])
print("No of Hours = 9.25")
print("Predicted Score = {}".format(pred[0]))
No of Hours = 9.25
Predicted Score = 93.6917324877358
XXXXXXXXXX

Evaluating the model

Evaluating the model

In [24]:
XXXXXXXXXX

from sklearn import metrics
print("Mean Absolute Error:",
      metrics.mean_absolute_error(y_test, y_pred))
Mean Absolute Error: 4.185890860275

```