Author : Abhishek Dhawan

Data Science and Business Analytics internship

The Sparks Foundation GRIP

In [ ]:

```
The Sparks Foundation GRIP
The Sparks Foundation GRIP
Batch : April 2024
Task 1: Prediction using Supervised ML
```

Batch : April 2024

Task 1: Prediction using Supervised ML

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.

Question : What will be predicted score if a student studies for 9.25
hrs/ day?
Dataset is consist of only two variables:
Hours : The number of hours students studied.
Scores : The percentage scores obtained by students
```

Importing the required libraries

from sklearn.model_selection import train_test_split from sklearn.linear_model import LinearRegression import matplotlib.pyplot as plt import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns import warnings warnings.filterwarnings("ignore") from sklearn.model_selection import train_test_split from sklearn.linear_model import LinearRegression from sklearn.metrics import mean_absolute_error, mean_squared_error, explained_variance_score sns.set(font_scale= 1) sns.set_style('darkgrid') %matplotlib inline

In [36]:

```
# Reading data from remote link.
url = r"https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/st
data = pd.read_csv(url)
print("Data import successful")
```

```
Data import successful
```

In [37]:
```
#Printing the first 10 rows of the dataset.
data.head(10)
```

Out[37]:

	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

In [8]:
```
#Printing the last few rows of the dataset.
data.tail(10)
```

Out[8]:

	Hours	Scores
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

In [9]:
```
print('The size of Dataframe is: ', data.shape)#Getting the no of rows in a
print('\n')
data.info()#Getting the  infornmation about each datafranme.
```

```
The size of Dataframe is:  (25, 2)


<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 [10]:

```
# To find total_missing_values in different columns of data and their perce
def missing_data(data):
    """
    This will take in a dataframe and
    finds the total_missing_values as well as percentage of the value count
    """
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(a
    return pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
```

In [11]:

```
# For finding missing values I use above created function
missing_data(data)
```

Out[11]:

	Total	Percent
Hours	0	0.0
Scores	0	0.0

As from above dataframe, my observation on missing data are:

There is no missing values in our dataset. Therefore there is no need of data cleaning

In [13]:

```
# To find descriptive statistic summary on the data I used function below:
data.describe()
```

Out[13]:

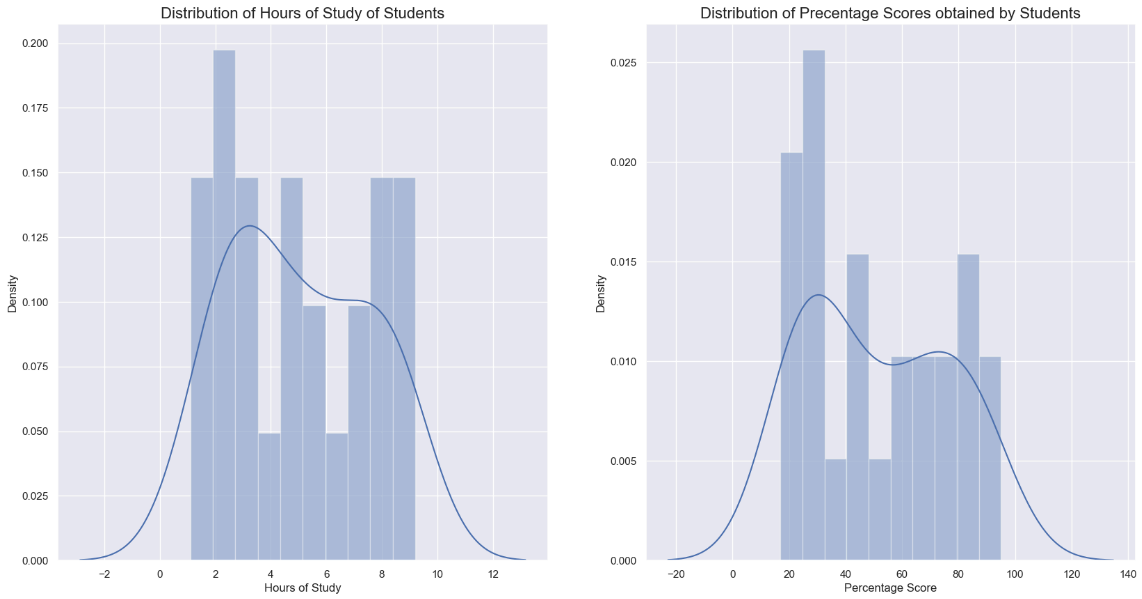
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

DATA VISUALISATION

In [17]:

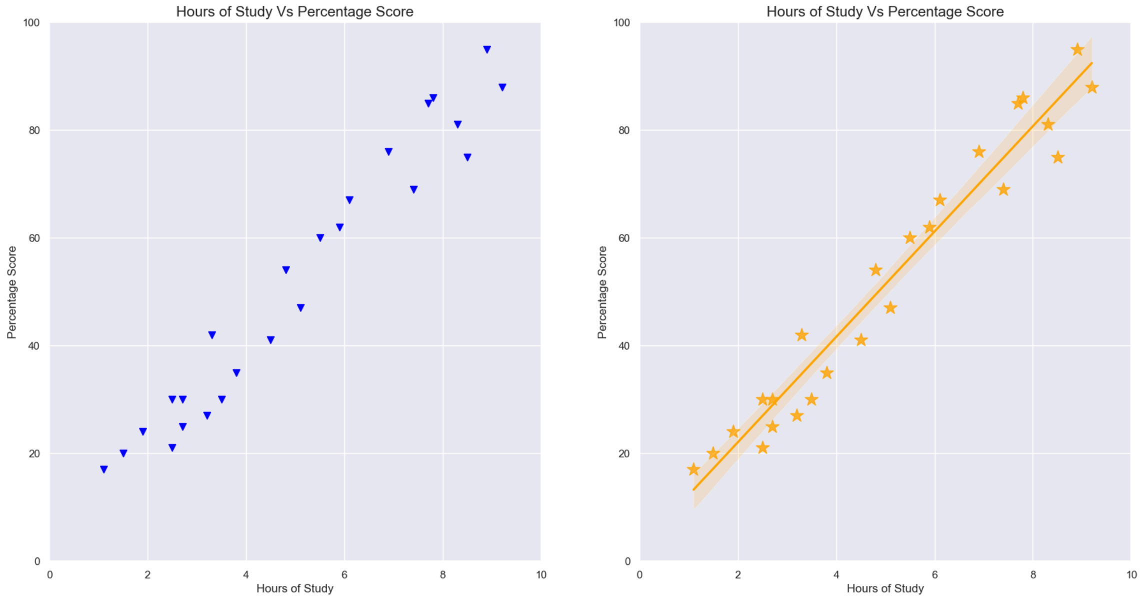
```
# Let's see the distribution of the two variable from our data
fig = plt.figure(figsize=(20,10)) # create figure

ax0 = fig.add_subplot(1, 2, 1) # add subplot 1 (1 row, 2 columns, first plo
ax1 = fig.add_subplot(1, 2, 2) # add subplot 2 (1 row, 2 columns, second pl

# Subplot 1: Distribution plot of 'Hours'
k1 = sns.distplot(data['Hours'], bins=10, ax=ax0) # add to subplot 1
ax0.set_title('Distribution of Hours of Study of Students', fontsize=16)
ax0.set(xlabel= 'Hours of Study', ylabel= 'Density')

# Subplot 2: Distribution plot of 'Score'
k2 = sns.distplot(data['Scores'], bins=10, ax=ax1) # add to subplot 1
ax1.set_title('Distribution of Precentage Scores obtained by Students', fon
ax1.set(xlabel= 'Percentage Score', ylabel= 'Density')

plt.show()
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_18944\1127891130.py:8: UserWar
ning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  k1 = sns.distplot(data['Hours'], bins=10, ax=ax0) # add to subplot 1
C:\Users\abhis\AppData\Local\Temp\ipykernel_18944\1127891130.py:13: UserWa
rning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  k2 = sns.distplot(data['Scores'], bins=10, ax=ax1) # add to subplot 1
```



In [18]:

```
# Let's visualize the relationship between the two variable using 'scatterp
fig = plt.figure(figsize=(20,10)) # create figure
sns.set(font_scale= 1)
ax0 = fig.add_subplot(1, 2, 1, xlim=(0,10), ylim=(0,100)) # add subplot 1 (
ax1 = fig.add_subplot(1, 2, 2, xlim=(0,10), ylim=(0,100)) # add subplot 2 (

# Subplot 1: Distribution plot of 'Hours'
k1 = data.plot(kind= 'scatter', x='Hours', y= 'Scores',color= 'blue', marke
ax0.set_title('Hours of Study Vs Percentage Score', fontsize=15)
ax0.set(xlabel= 'Hours of Study', ylabel= 'Percentage Score')

# Subplot 2: Distribution plot of 'Score'
k2 = sns.regplot(data= data, x= 'Hours', y= 'Scores', color= 'Orange', mark
ax1.set_title('Hours of Study Vs Percentage Score', fontsize=15)
ax1.set(xlabel= 'Hours of Study', ylabel= 'Percentage Score')

plt.show()
```



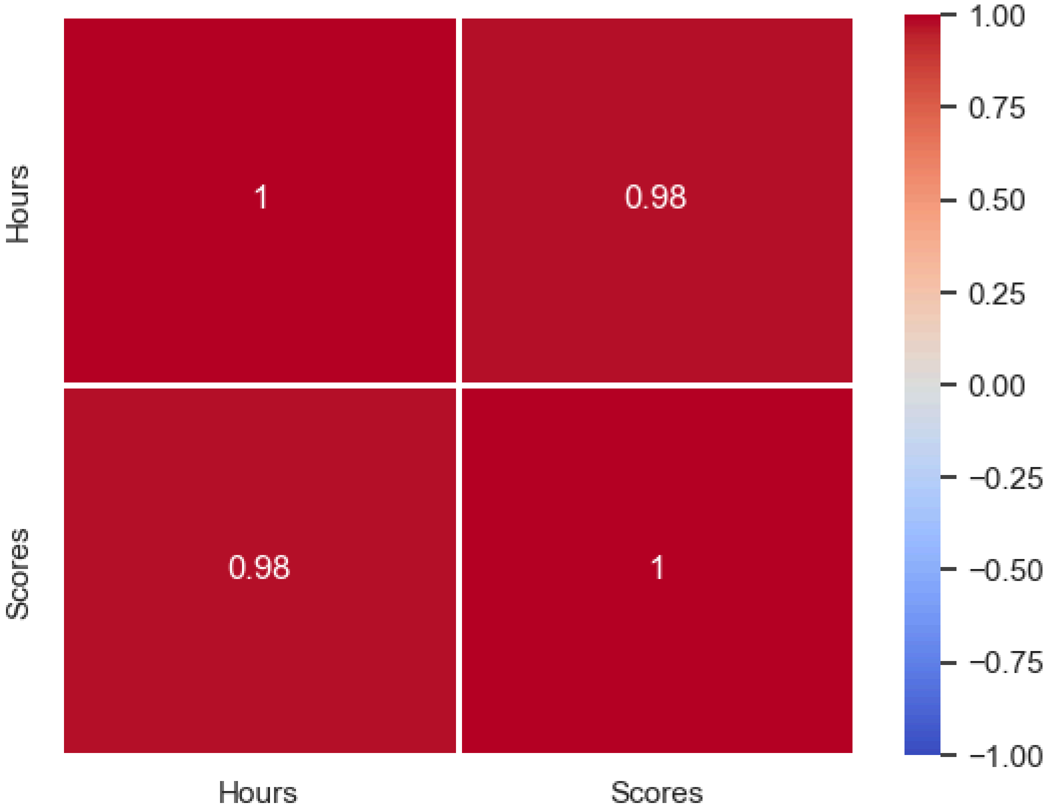
From the graphs above, we can clearly see that there is a positive linear relation between the number of hours of study and percentage of score.

In [19]:

```
# Let's see how much correlation is there between the variables
sns.heatmap(data= data.corr(), annot= True, cmap= 'coolwarm', vmin = -1, vm
```

Out[19]:

```
<Axes: >
```



From the above heatmap, we can clearly see that the number of hours of study and percentage of score have high positive correlation of 0.98 between them.

Preparing the data

This step consists of differentiating between feature variables and target variable.

In [20]:

```
X = data.iloc[: , :-1].values                # Feature variable
y = data.iloc[: , -1].values                # Target variable
```

Now that we have our feature variable and target variable, the next step is to split this data into training and test sets.

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [22]:

```
print('X_train shape: ', X_train.shape)
print('X_tests shape: ', X_test.shape)
print('y_train shape: ', y_train.shape)
print('y_test shape: ', X_test.shape)
```

```
X_train shape:  (20, 1)
X_tests shape:  (5, 1)
y_train shape:  (20,)
y_test shape:  (5, 1)
```

We have split our data into training and testing sets. Now this is finally the time to train our algorithm.

Training the Algorithm

In [23]:

```
# Traning Linear Regression Model
lm = LinearRegression()
lm.fit(X= X_train, y= y_train)

print("Training complete.")
```

```
Training complete.
```

Intercept and coefficients of the model

After training the model, we can see the intercept and coefficients of the model.

In [24]:

```
# Let's see coefficients and Intercept
print('Intercept: ', lm.intercept_)
print('Coefficients: \n', lm.coef_)
```

```
Intercept:  2.826892353899737
Coefficients:
 [9.68207815]
```

Interpreting the coefficients: A 1 unit increase in Hours of study is associated with an increase of 9.6820 in Percentage Score of Student.

Making Prediction

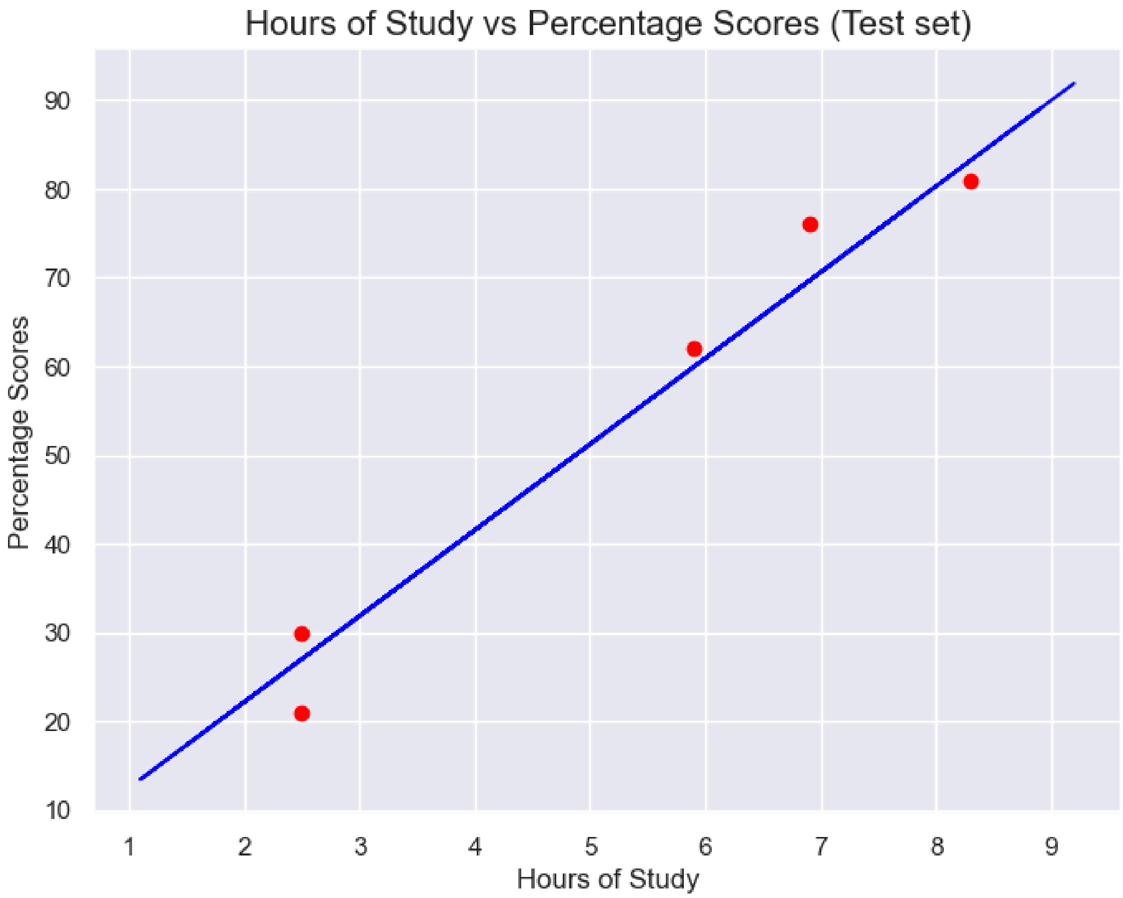
Now that we have trained our algorithm, it's time to make some predictions.

In [25]:

```
# Predicting the scores :
y_pred = lm.predict(X_test)
```

In [26]:

```
plt.figure(figsize=(8,6))
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, lm.predict(X_train), color = 'blue')
plt.title('Hours of Study vs Percentage Scores (Test set)', fontsize=15)
plt.xlabel('Hours of Study')
plt.ylabel('Percentage Scores')
plt.show()
```



In [27]:

```
# Comparing Actual vs Predicted Values :
df1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df1
```

Out[27]:

	Actual	Predicted
0	81	83.188141
1	30	27.032088
2	21	27.032088
3	76	69.633232
4	62	59.951153

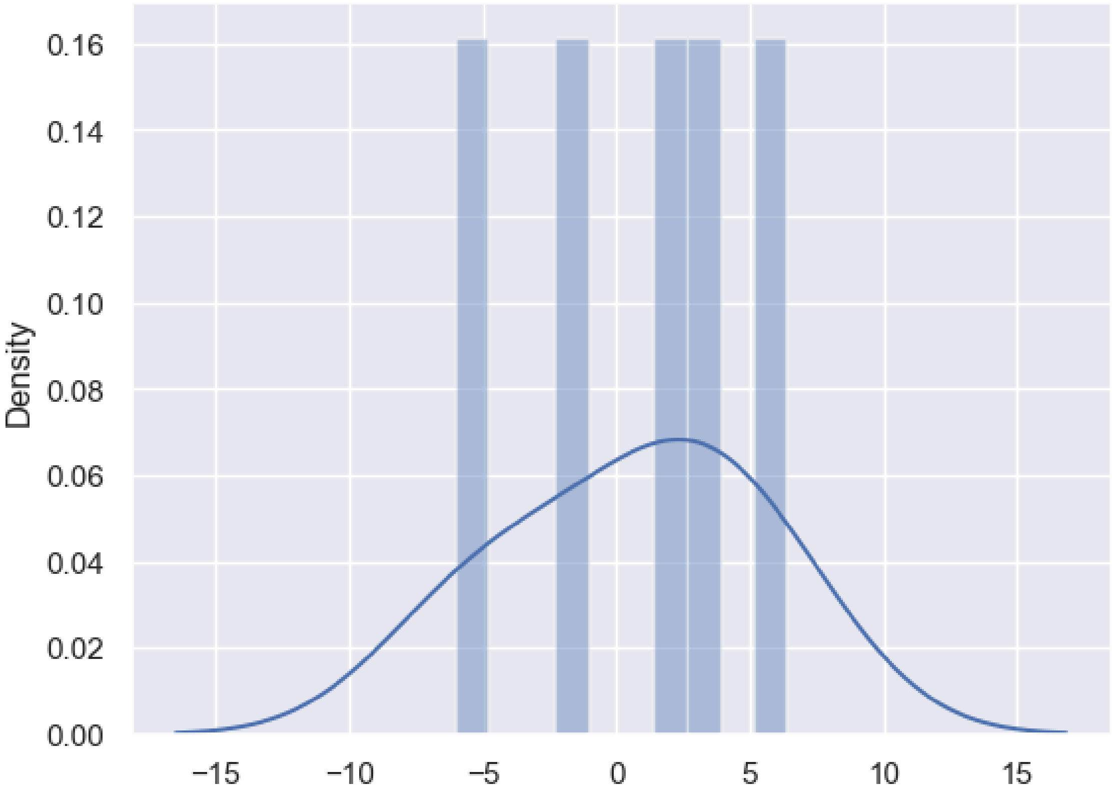
Residuals

Next, I explore the residuals to make sure everything was okay with the data (i.e. it is Normally distributed).

In [28]:
```
sns.distplot(a = (y_test - y_pred), bins= 10)
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_18944\2439728062.py:1: UserWar
ning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(a = (y_test - y_pred), bins= 10)
```

Out[28]:
```
<Axes: ylabel='Density'>
```



Evaluating the Model

The final step is to evaluate the performance of algorithm. This step is particularly important to compare how well different algorithms perform on a particular dataset. For simplicity here, I have chosen the root mean square error. There are many such metrics.

In [31]:

```
print('MAE: ', mean_absolute_error(y_true= y_test, y_pred= y_pred))     # M
print('MSE: ', mean_squared_error(y_true= y_test, y_pred= y_pred))      # M
print('RMSE: ', np.sqrt(mean_squared_error(y_true= y_test, y_pred= y_pred))

# To get R^2 we use the "explained variance score"
print('\nExplaned Variance Score: ', explained_variance_score(y_true= y_tes
```

```
MAE:  3.9207511902099244
MSE:  18.943211722315272
RMSE:  4.352380006653288

Explaned Variance Score:  0.9684858031070392
```

Model Accuracy is 96.84%

Question: What will be predicted score if a student studies for 9.25 hrs/ day?

In [32]:

```
# Testing with some new data :

hours = 9.25
test = np.array([hours])
test = test.reshape(-1, 1)
own_pred = lm.predict(test)
print("Predicted Score if a student studies for 9.25 hrs/ day is {}".format
```

```
Predicted Score if a student studies for 9.25 hrs/ day is 92.3861152826149
4
```

Thank You!

In [ ]:

```

```