Surabhi Singh

The Spark Foundation

Linear Regression Task

Predict the percentage of marks of an student based on the number of study hourse

Python Libraries being used:-

A:Data Analysis

In [5]:

```
import pandas as pd
import numpy as np
```

B:Data Visualization

In [6]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

C:Model Prediction

In [7]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
```

Data Extraction from File

- **Importing Dataset**
  - **Score data from our Task file**

In [8]:

```
url = "http://bit.ly/w-data"
score = pd.read_csv(url)
score.head()
```

Out[8]:

	Hours	Scores
0	**2.5**	**21**
1	**5.1**	**47**
2	**3.2**	**27**
3	**8.5**	**75**
4	**3.5**	**30**

In [9]:

```
score.describe()
```

Out[9]:

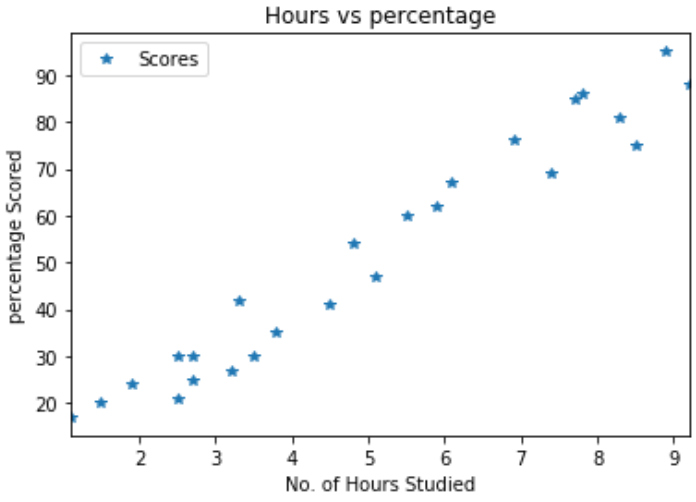
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

Data Visualization

Plotting data Points on 2-D

In [10]:

```
# Distribution of score
score.plot(x='Hours', y='Scores', style='*')
plt.title('Hours vs percentage')
plt.xlabel('No. of Hours Studied')
plt.ylabel ('percentage Scored')
plt.show()
```

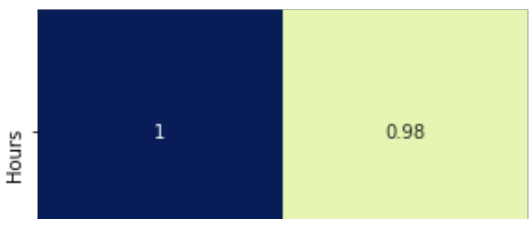


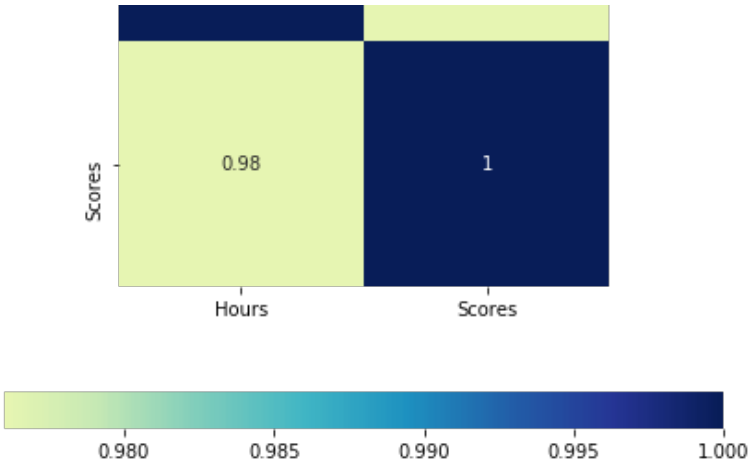
A linear relation can be observed

Correlation Plot-

In [13]:

```
plt.figure(figsize=(7,7))
sns.heatmap(score.corr(), annot=True, square=True, vmin=0.976, vmax=1, center= 0.986, cm
ap= 'YlGnBu', cbar_kws= {'orientation': 'horizontal'})
plt.show()
```





Predictive Modelling

Splitting Data Train and Test sets

In [14]:

```
x= score.iloc[:, :-1].values
y= score.iloc[:, 1].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

Training

In [16]:

```
train = LinearRegression()
train.fit(x_train, y_train)
```

Out[16]:

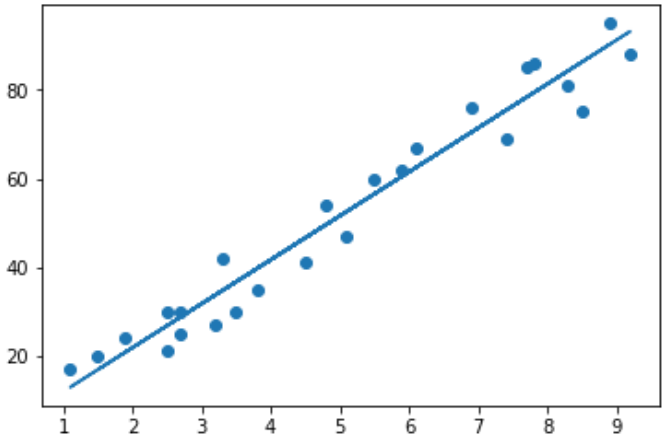
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Testing

In [17]:

```
# Regression Line
line= train.coef_*x+train.intercept_

plt.scatter(x, y)
plt.plot(x, line);
plt.show()
```



Evaluating Accuracy

In [19]:

```
print('Training Score')
print(train.score(x_train, y_train))
print('Test Score')
print(train.score(x_test, y_test))
```

```
Training Score
0.9515510725211553
Test Score
0.9454906892105356
```

In [21]:

```
# To Find mean Absolute Error(mse)
y_pred= train.predict(x_test)
mse= (mean_absolute_error(y_test, y_pred))
print('MAE:',mse)

#To find Root Mean  Squared Error (rmse)
rmse= (np.sqrt(mean_squared_error(y_test, y_pred)))
print('RMSE:',rmse)

#To find coefficient of determination
r2= r2_score(y_test, y_pred)
print('R-Square:', r2)
```

```
MAE: 4.183859899002975
RMSE: 4.6474476121003665
R-Square: 0.9454906892105356
```

Predicting

In [22]:

```
print(x_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [24]:

```
compare = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
compare
```

Out[24]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [27]:

```
User_input = float(input())
print('Hours: {}'.format(User_input))
print('Predicted Score{}'.format(train.predict([[User_input]])))
```

```
2.5
Hours: 2.5
Predicted Score[26.79480124]
```

In [ ]: