Task 1: Prediction using Supervised ML

Predict the percentage of an student based on the no. of study hours.

Importing necessay libraries

In [73]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Loading data in DataFrame

In [74]:

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

Out[74]:

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

In [75]:

```
df.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
```

In [76]:

```
df.describe()
```

Out[76]:

	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

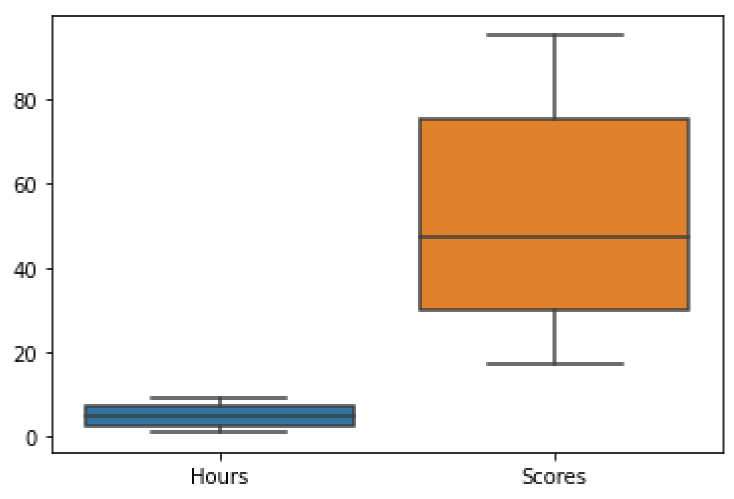
Boxplot of our data

In [77]:

```
sns.boxplot(data=df[["Hours","Scores"]])
```

Out[77]:

```
<AxesSubplot:>
```

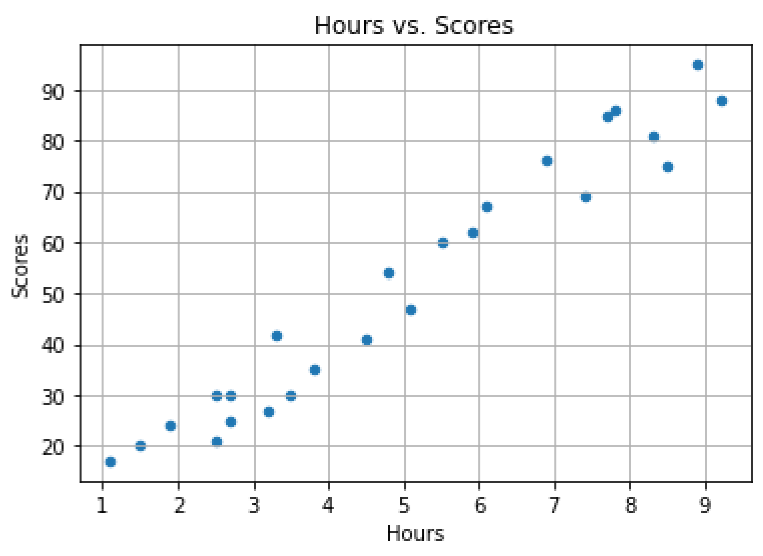


Clearly there are no outliers in our data

Scatterplot of our data

In [89]:

```
df.plot.scatter(x="Hours",y="Scores")
plt.title("Hours vs. Scores")
plt.grid()
plt.show()
```



Clearly we can see that there is positive linear relationship between hours and scores

Preparing the data

In [103…

```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
```

Splitting our Dataset into Train and Test

In [147…

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size = 0.20, random_state = 0)
```

Here we are using 80% of our dataset for training and 20% of the data for testing.

Training the Algorithm

Now we use the training data to train our Algorithm

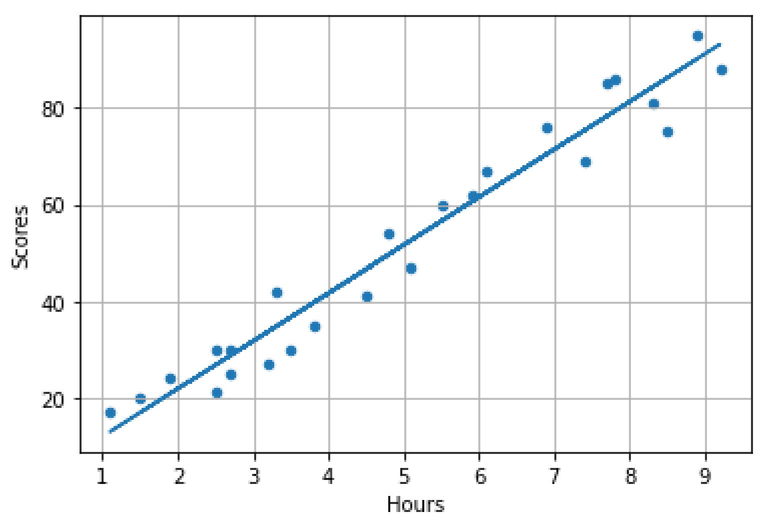
In [148…

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

print("Training complete.")
```

```
Training complete.
```

In [149…

```
line = regressor.coef_*X+regressor.intercept_

df.plot.scatter(x="Hours",y="Scores")
plt.plot(X, line);
plt.grid()
plt.show()
```



Let's make some Predictions

In [150…

```
# Here we predict the scores
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
```

Comparing Actual Score vs. Predicted Score

In [151…

```
df_compare = pd.DataFrame({"Actual Score":y_test,"Predicted Score":y_pred})
df_compare
```

Out[151…

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

Here we are needed to predict the score if a student studies for 9.25 hrs/ day

In [152…

```
my_hours = np.array([[9.25]])
my_pred = regressor.predict(my_hours)
print("No of Hours = {}".format(my_hours[0][0]))
print("Predicted Score = {}".format(my_pred[0]))
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737538
```

Evaluating the Model

In [154…

```
import sklearn.metrics as metrics

explained_variance=metrics.explained_variance_score(y_test, y_pred)
mean_absolute_error=metrics.mean_absolute_error(y_test, y_pred)
mse=metrics.mean_squared_error(y_test, y_pred)
mean_squared_log_error=metrics.mean_squared_log_error(y_test, y_pred)
median_absolute_error=metrics.median_absolute_error(y_test, y_pred)
r2=metrics.r2_score(y_test, y_pred)

print('Explained Variance: ', round(explained_variance,4))
print('mean_squared_log_error: ', round(mean_squared_log_error,4))
print('r2: ', round(r2,4))
print('MAE: ', round(mean_absolute_error,4))
print('MSE: ', round(mse,4))
print('RMSE: ', round(np.sqrt(mse),4))
```

```
Explained Variance:  0.9483
mean_squared_log_error:  0.0185
r2:  0.9455
MAE:  4.1839
MSE:  21.5988
RMSE:  4.6474
```