# Predction Using ML

In [1]:
```
# importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Load the dataset
path = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv"

# Read the csv file using read_csv
df = pd.read_csv(path)
```

In [3]:
```
df.head()
```

Out[3]:

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

## Descriptive Statistics

In [4]:
```
df.describe()
```

Out[4]:

| | 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 |

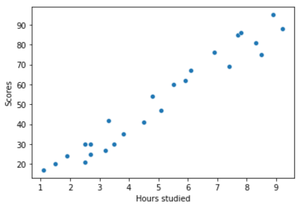
In [5]:
```
df.info()
```

```
<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
```

## Plot the relationship between Hours and scores

In [6]:
```
sns.scatterplot(x='Hours', y='Scores', data=df)
plt.xlabel('Hours studied')
plt.ylabel('Scores')
```

Out[6]:
```
Text(0, 0.5, 'Scores')
```




## Preparing the data

In [7]:
```
# independent variable
X = df.iloc[:, :-1].values

# dependent(target) variable
y = df.iloc[:, 1].values
```

## Training the model

In [8]:
```
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, random_state=0)
```

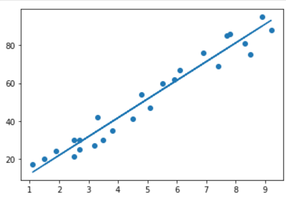
In [9]:
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()

# Fitting the data
lm.fit(X_train, y_train)
```

Out[9]:
```
LinearRegression()
```

## Plotting the regression line

In [12]:
```
Regression_Line = lm.coef_*X+lm.intercept_
plt.scatter(X,y)
plt.plot(X,Regression_Line);
plt.show()
```



## Making Predictions

In [13]:
```
# Predicting the data
predictions = lm.predict(X_test)
```

In [14]:
```
# Comparing Actual vs Predicted
compare_scores = pd.DataFrame({'Actual Marks': y_test, 'Predicted Marks': predictions})
compare_scores
```

Out[14]:

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

## Evaluating the model

In [15]:
```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, predictions)
print('Mean Absolute Error: ', mae)
```

```
Mean Absolute Error:  4.183859899002982
```

## Testing the model by giving my values

In [16]:
```
hours = [9.25]
predicted_score = lm.predict([hours])
print("Predicted Score = {}".format(round(predicted_score[0],3)))
```

```
Predicted Score = 93.692
```

In [ ]:
```

```