Linear Regression with Python Scikit Learn

In this section we will see how the Python Scikit-Learn library for machine learning can be used to implement regression functions. We will start with simple linear regression involving two variables.

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.

Author : Ameya Kalbande

In [55]:
```
# Importing all libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

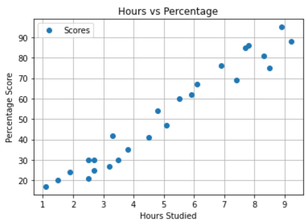
In [3]:
```
# Reading data from remote link
url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv"
data = pd.read_csv(url)
print("Data imported successfully")
data.head(10)
```

```
Data imported successfully
```

Out[3]:

	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 [50]:
```
# Plotting the distribution of scores
data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid(True)
plt.show()
```



In [5]:
```
#Preparing the data
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
```

In [22]:
```
#Splitting the data into training and test sets
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 [26]:
```
#Training the Algorithm
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

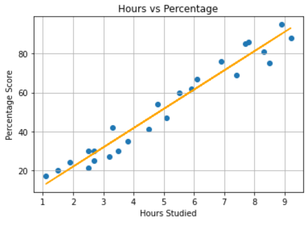
In [27]:
```
#Fitting Training Data
lr.fit(X_train, Y_train)
print("Training complete.")
```

```
Training complete.
```

In [28]:
```
print("X train.shape =", X_train.shape)
print("Y train.shape =", Y_train.shape)
print("X test.shape =", X_test.shape)
print("Y test.shape =", Y_test.shape)
```

```
X train.shape = (20, 1)
Y train.shape = (20,)
X test.shape = (5, 1)
Y test.shape = (5,)
```

In [52]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_
# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line,color='orange');
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid(True)
plt.show()
```



In [33]:
```
#Making Prediction
# Testing data
print(X_test)
# Predicting the scores
Y_pred = lr.predict(X_test)
print(y_pred)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
```

In [34]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': Y_test, 'Predicted': y_pred})
df
```

Out[34]:

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

In [41]:
```
#Accuracy of Model
from sklearn import metrics
metrics.r2_score(Y_test,Y_pred)
```

Out[41]:
```
0.9454906892105355
```

In [47]:
```
#Predicting the Error
from sklearn.metrics import mean_squared_error,mean_absolute_error
MSE = metrics.mean_squared_error(Y_test,Y_pred)
RMSE = np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
MAE = (metrics.mean_absolute_error(Y_test,Y_pred))
print("Mean Squared Error",MSE)
print("Root Mean Squared Error",RMSE)
print("Mean Absolute Error",MAE)
```

```
Mean Squared Error 21.598769307217413
Root Mean Squared Error 4.647447612100368
Mean Absolute Error 4.18385989900298
```

In [56]:
```
#Predicting the score for 9.25hrs
Prediction_score = lr.predict([[9.25]])
print("Predicted Score for a Student Studying 9.25 hours is:",Prediction_score)
```

```
Predicted Score for a Student Studying 9.25 hours is: [93.69173249]
```