In [1]:
```
'''
What is Machine Learning?
Well, Machine Learning is a concept which allows the machine to learn from examples and experience,
and that too without being explicitly programmed. So instead of you writing the code, what you do is
you feed data to the generic algorithm, and the algorithm/ machine builds the logic based on the given data.
Have you ever shopped online? So while checking for a product,
did you noticed when it recommends for a product similar to what you are looking for?
or did you noticed "the person bought this product also bought this" combination of products.
Now are they doing this recommendation? This is machine learning.
ML makes computers learn the data and making own decisions. How does Machine Learning Work?
Machine Learning algorithm is trained using a training data set to create a model.
When new input data is introduced to the ML algorithm, it makes a prediction on the basis of the model.
The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed.
If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented training dataset.
What is Supervised Learning? Supervised Learning is the one,
where you can consider the learning is guided by a teacher.
We have a dataset which acts as a teacher and its role is to train the model or the machine.
Once the model gets trained it can start making a prediction or decision when new data is given to it.
What is Unsupervised Learning? The model learns through observation and finds structures in the data.
Once the model is given a dataset, it automatically finds patterns and relationships in the dataset by creating clusters in it.
Suppose we presented images of apples, bananas and mangoes to the model,
so what it does, based on some patterns and relationships it creates clusters and divides the dataset into those clusters.
Now if a new data is fed to the model, it adds it to one of the created clusters.
'''
# Supervised Machine Learning
# Regression
# Regression is a supervised machine learning technique which is used to predict
# continuous values. The ultimate goal of the regression algorithm is to
# plot a best-fit line or a curve between the data.
# Regression analysis is a statistical method to model the relationship between
# a dependent (target) and independent (predictor) variables with one or more
# independent variables. More specifically, Regression analysis helps us
# to understand how the value of the dependent variable is changing corresponding to
# an independent variable. It predicts continuous/real values such as temperature, age, salary, price, etc.
# Example: Suppose there is a marketing company A, who does various advertisement
# every year and get sales on that. The below list shows the advertisement
# made by the company in the last 5 years and the corresponding sales:
      # Advertisement    Sales
        # $90            $1000
        # $120           $1300
        # $150           $1600
        # $100           $1200
        # $170           ??
# Now, the company wants to do the advertisement of $170 in the year 2024 and wants to know the
# prediction about the sales for this year.
# So to solve such type of prediction problems in machine learning, we need regression analysis.
'''
In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot,
the machine learning model can make predictions about the data. In simple words,
"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in
such a way that the vertical distance between the datapoints and the regression line is minimum."
The distance between datapoints and line tells whether a model has captured a strong relationship or not.
'''
# Terms Used # The variable that we are trying to explain or predict is called the
# response variable. It is also sometimes called the dependent variable
# because it depends on another variable.
# The variable that is used to explain or predict the response variable is called
# the explanatory variable. It is also sometimes called the independent
# variable because it is independent of the other variable.
# Simple Linear Regression
'''
Simple linear regression is used to estimate the relationship between two quantitative variables.
You can use simple linear regression when you want to know: a.) How strong the relationship is between two variables
(e.g., the relationship between rainfall and soil erosion).
b.) The value of the dependent variable at a certain value of the independent variable
(e.g., the amount of soil erosion at a certain level of rainfall).
Regression models describe the relationship between variables by fitting a line to the observed data.
Linear regression models use a straight line, while logistic and nonlinear regression models use a curved line.
Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.
Simple linear regression example:You are a social researcher interested in the relationship between income and happiness.
You survey 500 people whose incomes range from 15k to 75k and ask them to rank their happiness on a scale from 1 to 10.
Your independent variable (income) and dependent variable (happiness) are both quantitative,
so you can do a regression analysis to see if there is a linear relationship between them.
'''
'''
Simple linear regression formula : The formula for a simple linear regression is: y = B0 + B1X + E
y is the predicted value of the dependent variable (y) for any given value of the independent variable (x).
B0 is the intercept, the predicted value of y when the x is 0.
B1 is the regression coefficient - how much we expect y to change as x increases.
x is the independent variable ( the variable we expect is influencing y).
e is the error of the estimate, or how much variation there is in our estimate of the regression coefficient.
'''
```

Out[13]:
```
'\nSimple linear regression formula : The formula for a simple linear regression is: y = B0 + B1X + E\ny is the predicted value of the dependent variable (y) for any given value of the independent variable (x).\nB0 is the intercept, the predicted value of y when the x is 0.\nB1 is the regression coefficient - how much we expect y to change as x increases. \nx is the independent variable ( the variable we expect is influencing y).\ne is the error of the estimate, or how much variation there is in our estimate of the regression coefficient. \n'
```

In [2]:
```
# supervised ML => Simple Linear Regression
# import all necessary library
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
# stage1: Data Gathering
data = pd.read_csv('https://raw.githubusercontent.com/yash240990/Python/master/Grade_Set_1.csv')
data
```

Out[4]:

	Hours_Studied	Test_Grade	Status	Result
0	2	57	fail	D
1	3	66	fail	D
2	4	73	pass	C
3	5	76	pass	C
4	6	79	pass	C
5	7	81	pass	B
6	8	90	pass	B
7	9	96	pass	A
8	10	100	pass	A

In [5]:
```
# stage2: EDA
data.shape
```

Out[5]:
```
(9, 4)
```

In [6]:
```
data.columns
```

Out[6]:
```
Index(['Hours_Studied', 'Test_Grade', 'Status', 'Result'], dtype='object')
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Hours_Studied  9 non-null     int64 
 1   Test_Grade     9 non-null     int64 
 2   Status         9 non-null     object
 3   Result         9 non-null     object
dtypes: int64(2), object(2)
memory usage: 420.0+ bytes
```

In [8]:
```
# stage3: data preparation
# a) check for the duplicate values
# b) convert non-numerical feature into numerical feature
# c) Normalization
# d) select dependent and independent variable
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
Hours_Studied    0
Test_Grade       0
Status           0
Result           0
dtype: int64
```

In [10]:
```
data.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
# convert non-numerical feature into numerical feature
import sklearn.preprocessing as pp
lb = pp.LabelBinarizer()
```

In [12]:
```
lb.fit_transform(data.Status)
```

Out[12]:
```
array([[0],
       [0],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1]])
```

In [13]:
```
data.Status = lb.fit_transform(data.Status)
```

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Hours_Studied  9 non-null     int64
 1   Test_Grade     9 non-null     int64
 2   Status         9 non-null     int32
 3   Result         9 non-null     object
dtypes: int32(1), int64(2), object(1)
memory usage: 384.0+ bytes
```

In [15]:
```
data.Test_Grade.values
```

Out[15]:
```
array([ 57,  66,  73,  76,  79,  81,  90,  96, 100], dtype=int64)
```

In [16]:
```
# normalization
# the goal of the normalization is to change the value of numeric column in dataset to a scale[0-1]
# min-max normalization
#Xnorm = (x-xmin)/(xmax-xmin)
```

In [17]:
```
Xnorm = 57-57/100-57
Xnorm
```

Out[16]:
```
-0.5700000000000003
```

In [17]:
```
vals = data.Test_Grade.values
```

In [18]:
```
vals
```

Out[18]:
```
array([ 57,  66,  73,  76,  79,  81,  90,  96, 100], dtype=int64)
```

In [19]:
```
# nrml_vals = pp.normalize([vals])
# print(nrml_vals)
```

In [ ]:
```
# nrmlz = pp.MinMaxScaler()
# nrmlz_vals = nrmlz.fit_transform([vals])
# print(nrmlz_vals)
```

In [25]:
```
# d) select dependent and independent variable
# dependent variable must be one and independent variable can be one or more
data.columns
# independent variable(define the value in which we have to predict)
x= data.Hours_Studied.values # x is the independent variable
x = x.reshape(9,1) # add one dimension to it
x.shape
```

Out[25]:
```
(9, 1)
```

In [26]:
```
# dependable variable
y = data.Test_Grade.values
y
```

Out[26]:
```
array([ 57,  66,  73,  76,  79,  81,  90,  96, 100], dtype=int64)
```

In [27]:
```
# stage 4 building a predictive model
# apply a suitable ml algo dataset
import sklearn.linear_model as lm
lin_reg = lm.LinearRegression()
```

In [28]:
```
# fit apply formula to machine y = B0 + B1X + E
lin_reg.fit(x,y) # pass the dependent and independent variable inside()this circle braket
```

Out[28]:
```
LinearRegression()
```

In [31]:
```
# stage 5:Evaluation: Evaluate the model
data['pred_values'] = lin_reg.predict(x)
```

In [32]:
```
data[['Hours_Studied','Test_Grade','pred_values']]
```

Out[32]:

	Hours_Studied	Test_Grade	pred_values
0	2	57	59.711111
1	3	66	64.727778
2	4	73	69.744444
3	5	76	74.761111
4	6	79	79.777778
5	7	81	84.794444
6	8	90	89.811111
7	9	96	94.827778
8	10	100	99.844444

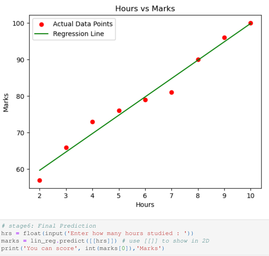
In [34]:
```
pred__value = lin_reg.predict(x)
print(pred__value)
```

```
[59.71111111 64.72777778 69.74444444 74.76111111 79.77777778 84.79444444
 89.81111111 94.82777778 99.84444444]
```

In [38]:
```
# r2_score --> regression score function above 80%
from sklearn.metrics import r2_score
accuracy = r2_score(y,pred__value)
print(accuracy)
print(f'Accuracy of the model: {int(accuracy*100)}%')
```

```
0.9757431207895347
Accuracy of the model: 97%
```

In [41]:
```
# data visualization
# plot actual values
plt.scatter(x,y,color='red',label='Actual Data Points')
# plot predicted value
plt.plot(x,pred__value,color='green',label='Regression Line')
plt.title('Hours vs Marks')
plt.xlabel('Hours')
plt.ylabel('Marks')
plt.legend()
plt.show()
```



In [47]:
```
# stage6: Final Prediction
hrs = float(input('Enter how many hours studied : '))
marks = lin_reg.predict([[hrs]]) # use [[]] to show in 2D
print('You can score', int(marks[0]),'Marks')
```

```
Enter how many hours studied : 4.5
You can score 72 Marks
```

In [ ]:
```

```