Linear Regression Technique:

In [32]:

```
#Importing the libraries "pandas, numpy, matplotlib.pyplot and seaborn" to my python scri
pt
#with the standard short name as "pd, np, plt and sns".

#Uploading the file on google colab and choosing the selected dataset by clicking "choose
files".

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

from google.colab import files
uploaded = files.upload()
```

Choose File **No file selected**

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving New_Data.csv to New_Data (2).csv
```

In [33]:

```
#In the first step, importing the dataset in the project by the "read_csv" function
#and that reads the data into a pandas dataframe object.

df = pd.read_csv('New_Data.csv')
```

In [34]:

```
#df.columns attribute where return the column labels of the given dataframe

df.columns
```

Out[34]:

```
Index(['YearsExperience', 'Salary'], dtype='object')
```

In [35]:

```
#head() method returns a particular rows from the top and where did not mention
#the number below hence returns first 5 rows

df.head()
```

Out[35]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

In [36]:

```
#The shape is a tuple of the array dimensions which gives the number of rows and
# columns of a given dataframe
```

```
df.shape
```

Out[36]:

```
(30, 2)
```

Cleaning the entire dataset

In [37]:

```
#The info() method gives the information about the dataframe where the information contai
ns;
#number of columns
#column labels
#column data types
#memory usage
#range index
#number of cells in each column

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 608.0 bytes
```

In [38]:

```
#The ducplicate() method gives a series with true and false values that describe,
#which rows in the dataframe are duplicated or not

df.duplicated().sum
```

Out[38]:

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of 0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
dtype: bool>
```

Exploring the dataset

In [39]:

```
#The describe() method gives description of the data in the dataframe
#The details are included in the description for each column if the dataframe includes nu
merical data.

df.describe()
```

Out[39]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [40]:

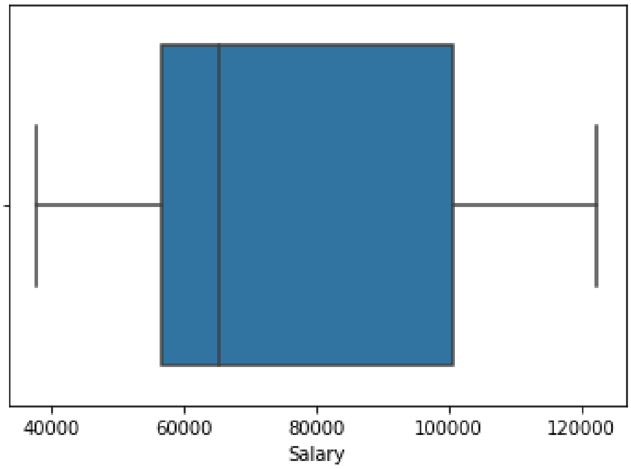
```
#A boxplot is a common visual representation of data acquisition based on a five-digit su
mmary.
#It can inform of the amounts of the outliers.

sns.boxplot(df['Salary'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fae4dddd700>
```



Linear Regression Technique

Importing libraries that we need to fit the model

In [41]:

```
#1. First importing the mean_squared_error from sklearn.metrics where measures the
#average of error squares which is the average squared difference between the
#estimated values and true values.
```

```
#2. Next importing the LinearRegression which is based on supervised learning to
#perform a regression task.

#3. Importing train_test_split function to split arrays or matrices into
#random subsets for train and test data.

#4. Importing mean_absolute_error function is used to find the difference between
#two paired observation sets taken under consideration.

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

In [42]:

```
#Rows and columns can be choosen by position or index using .iloc.
#It displays an index error if the position or index is invalid.

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

In [43]:

```
#Splitting the data using train_test_split.
#The purpose of split is to break the data into training and test or validation set.
#After that can check the validation of the model by fitting the training data and
#predicting using the test or validation by applying the technique.

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state
= 0,shuffle=True)
LRM = LinearRegression().fit(x_train, y_train)

LRM.score(x_train, y_train)
LRM.score(x_test, y_test)
```

Out[43]:

```
0.988169515729126
```

Predicting the Test set results

In [44]:

```
#Predicting the values of test result by using LRM.predict in array format.

y_pred = LRM.predict(x_test)
y_pred
```

Out[44]:

```
array([ 40748.96184072, 122699.62295594,  64961.65717022,  63099.14214487,
       115249.56285456, 107799.50275317])
```

In [45]:

```
#Now comparing the predicted values with real value in the below step.

new_data=pd.DataFrame({'Real value':y_test,'predict value':y_pred})
new_data
```

Out[45]:

	Real value	predict value
2	37731	40748.961841
28	122391	122699.622956
13	57081	64961.657170
10	63218	63099.142145

	Real value	predict value
26	116969	115249.562855
24	109431	107799.502753

In [46]:

```
#The mean_absolute_error function creates a local variable with (round) which is
#used to compute the mean_absolute_error.

mean_abs_error=mean_absolute_error(y_test, y_pred)
round(mean_abs_error,1)
```

Out[46]:

```
2446.2
```

In [47]:

```
#Here, expecting the salary of a person who has 15 years of experince.

new_predict=[[15]]
predict=LRM.predict(new_predict)
print(predict)
```

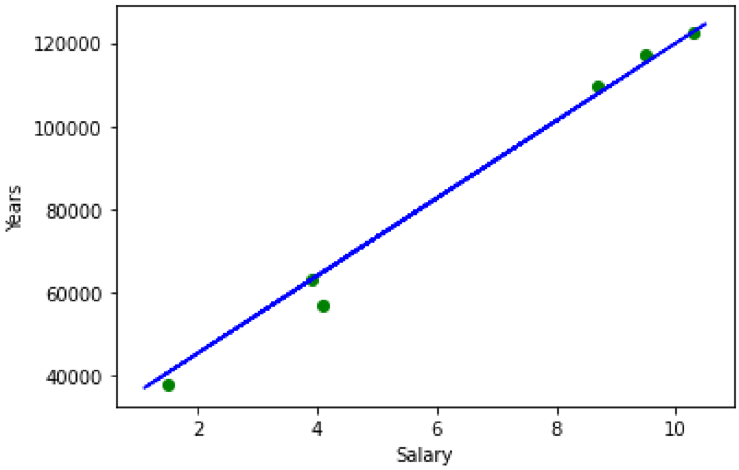
```
[166468.72605157]
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have
valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [48]:

```
# The scatter function plots one dot for each observation.
# It needs two arrays of the same length.
# I have selected 'salary' in the xlabel and 'years' in the y label.

plt.scatter(x_test, y_test, color = 'green')
plt.plot(x_train,LRM.predict(x_train), color = 'blue')
plt.xlabel('Salary')
plt.ylabel('Years')
plt.show()
```


Years
Salary