Import libraries Part

In [ ]:

```
# Importing the libraries
import numpy as np #numpy library is to do numerical maths calculations
import matplotlib.pyplot as plt #matplotlib.pyplot library is to do graphical notations for our datapoints supports only 2D and
#3D input, outputs
import pandas as pd # pandas library is to open excel files, tables and comma seperated values in our notebook
```

Data Collection Part

In [2]:

```
#data variablecsv valus is stored
data = pd.read_csv('Salary_Data.csv')#Storing the dataset into a "data" variable using the pandas function(read_csv) bz of csv
#file
```

In [3]:

```
data #executing data
```

Out[3]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0

There is No data preprocessing part here bz our dataset is already processed and has pure data

Input and Output Split up Part

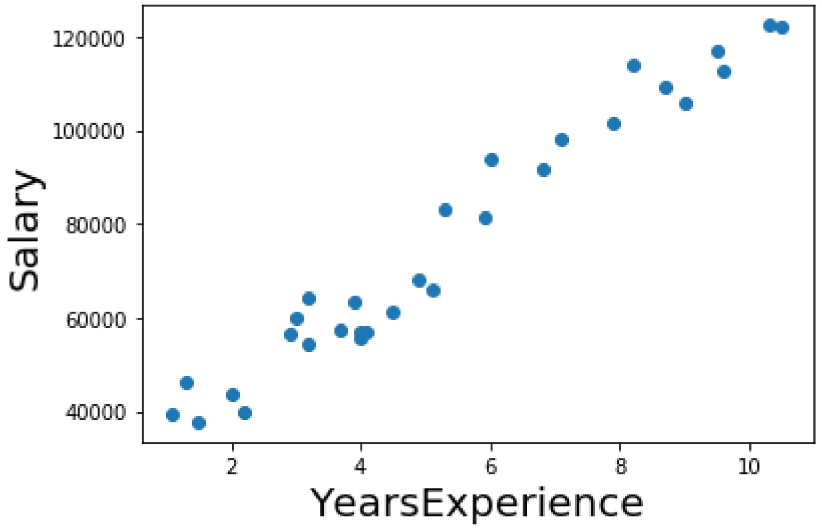
In [4]:

```

independent = data[["YearsExperience"]] #Indepedent Variable,years of Experience,datatype float(copying Years of experience
#data into independent variable)
dependent = data[["Salary"]] #Depedent Variable, Salary, datatype float(copying Salary data into Dependent variable)
```

In [5]:

```
plt.scatter(independent,dependent) #using matplotlib.pyplot library we are using scatter function to plot a graph with indep
# and dep variables
plt.xlabel('YearsExperience',fontsize=20) # Giving name to the X axis as 'YearsExperience' and maximxing its size
plt.ylabel('Salary',fontsize=20) # Giving name to the Y axis as 'Salary' and maximxing its size
plt.show() #show function in matplotlib.pyplot will show your graph with better image quality
```



Splitting Training and Test set part

In [6]:

```
from sklearn.model_selection import train_test_split # From sklearn model we are importing train_test_split function
X_train, X_test, y_train, y_test = train_test_split(independent,dependent, test_size = 1/3, random_state = 0)# this
# t_t_s function will return 4 variables splited 70% training set and 30% test set with random state=0 so that every body can
#get same order integrity
```

In [7]:

```
y_test # executing y_test data
```

Out[7]:

```
       Salary
 2    37731.0
28   122391.0
13    57081.0
10    63218.0
26   116969.0
24   109431.0
27   112635.0
11    55794.0
17    83088.0
22   101302.0
```

Model creation part

In [8]:

```python
from sklearn.linear_model import LinearRegression # we already knew from the graph that we should use Linear formula to solve
# problem statement so from Sklearn we are importing Linear regression function to do the model creation
regressor = LinearRegression() # creating object to the Linear regression function and formula is loaded
regressor.fit(X_train, y_train)#fit function is like substitution in Maths so We are just substituting the training set datas
# into the fit function that is Y=wX+b model created
```

Out[8]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [ ]:

```python
# Viewing the w and b value
weight=regressor.coef_ # creating weight variable and storing the weight value(w)
print("Weight of the model={}".format(weight))
bais=regressor.intercept_ # creating bias variable and storing the bias value(b)
print("Intercept of the model={}".format(bais))
```

Our model is created That is like Y=(9345.94244312)X+26816.19224403

Evaluation phase using Test dataset

In [ ]:

```python
y_pred=regressor.predict(X_test)# we just creating a y_pred variable and sending the training set input datas and
# using predict function we are getting predicted outputs with our created model
```

In [ ]:

```python
y_pred # executing y_pred
```

Error correctness for our model using r2_score function

In [ ]:

```python
from sklearn.metrics import r2_score # from sklearn we are importing r2_score function
r_score=r2_score(y_test,y_pred)# we are just sending our predicted data and test data outputs into it and finding the error value
```

In [ ]:

```python
r_score # executing r_score variable we get 0.9749154407708353 which is nearly 1, so our model is in good working condition
```

In [ ]:

```python

```

In [ ]:

```python
import pickle # pickle function is to store our final model in a seperate file so that noboday nows the backend work and neatly
# we can execute our call to function there
filename = 'finalized_model.sav' # creating file name with .sav (saved backup)
pickle.dump(regressor, open(filename, 'wb')) # using dump function we are storing object data into the file ie regressor
```

In [ ]:

```python
loaded_model = pickle.load(open('finalized_model.sav', 'rb'))# using load function we are loading the model into loaded_model
# variable
result = loaded_model.predict([[15]])# we are just executing the final step prediction on any years of experience
print(result)
```

In [ ]:

```
prediction_input=int(input("Enter the Prediction input value:")) # getting inputs from user and doing our prediction
```

In [ ]:

```
Future_Prediction=regressor.predict([[prediction_input]])# change the paramter,play with it.
print("Future_Prediction={}".format(Future_Prediction))
```

In [ ]:

In [ ]: