#importing pandas library and renaming it as pd , pandas is an inbuilt library used for calculation and everything related to database

import pandas as pd

#the collected and pre-processed data is assigned to the variable "dataset". hence, the data is stored in the memory under dataset

dataset=pd.read\_csv("Salary\_Data.csv")

#execute the dataset to see if the preprocessed data is stored in "dataset" variable

Dataset

#next step is to split input and output variables. Input value will be assigned to "independent" variable and "dataset" contains both input and output

independent=dataset[["YearsExperience"]]

indepenet #upon executing this , input data alone will be displayed

#output data will be assigned to the variable "dependent"

dependent=dataset[["Salary"]]

dependent #upon executing this, output data will be displayed

#to split training and the test set, we make use of the library sklearn and the file name model\_selection and fetch the function " train\_test\_split"

from sklearn.model\_selection import train\_test\_split

#calling and loading the parameterized function with input, output, test size 30% and random state

train\_test\_split(independent, dependent, test\_size=0.3, random\_state=0)

# train and test data are split and stored under the variables, X\_train, X\_test, y\_train and y\_test.

#Upon calling the below function train\_test\_split , the independent and dependent data will be spit and assigned to respected variable

#(input train and test set, output train and test set)

X\_train,X\_test,y\_train,y\_test=train\_test\_split()

#model creation for our problem statement

#the below 2 lines are the procedure to call LinearRegression class and its function and assign it to regressor variable

from sklearn.linear\_model import LinerRegression

regressor=LinearRegression()

#all the function under LinearRegression class will be stored under regressor variable

regressor.fit(X\_train, y\_train)

#the above fit method/function is used to substitute the input and output training set to calculate the bias and intercept value.

#using the above fit method, our model will be created.

#execute the above to check if the model is created and below is the output we'll receive

LinearRegression()

#to check the value of w and b calculated by the model, coef\_ is the method used to calculate weight

weight=regressor.coef\_

#to check the bias value, intercept is the method used

bias=regressor.intercept\_

# evaluation metrics execution

#the test data predicted by the above model will be stored in y\_pred

y\_pred=regressor.predict(X\_test)

#to calculate r2 to evaluate if our model is performing right

from sklearn.metrics import r2\_score

r\_score=r2\_score(y\_test, y\_pred)

#passing the parameter actual and predicted output and assigning to r\_score variable

#to calculate the r2 value , if it is close to 1 or close to 0, to cross check if the model has learnt properly

r\_score

# how to save the model. Import pickle function and save it in a variable named filename with the extension .sav

import pickle

filename="finalized\_linear\_model.sav"

#by executing the below line of code, the created model will be saved in the mentioned path with the same name finalized\_linear\_model.sav

pickle.dump(regressor, open(filename,'wb'))

#to load and check the saved model

loaded\_model=pickle.load(open("finalized\_linear\_model.sav",'rb'))

result=loaded.model.predict[[13]]

DEPLOYMENT PHASE

How to deploy the developed model

Import pickle

#this is where we have stored the model

Copy the loaded\_model line of code and run it through

loaded\_model=pickle.load(open(“finalized\_linear\_model.sav”,’rb’))

result=loaded\_model.predict([[13]])

# train and test data are split and stored under the variables, XTrain, Xtest, ytrain and Ytest.

#Upon calling the below function train\_test\_split , the independent and dependent data will be spit and assigned to respected variables.

X\_train,X\_test,y\_train,y\_test=train\_test\_split()