A Project Review

on

COMPARATIVE STUDY OF VARIOUS MACHINE LEARNING ALGORITHMS

Submitted in partial fulfillment of the

requirement for the award ofthe degree of

Bachelor of Technology in CSE

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. P. Raja Kumar
Assistant Professor

Submitted By:

Ayushi Sharma
(18SCSE1180030)

Jyotsna Pathak
(18SCSE1010092)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE’S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **“COMPARATIVE STUDY OF VARIOUS MACHINE LEARNING ALGORITHMS”** in partial fulfilment of the requirements for the award of the B.TECH submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during December, 2021 under the supervision of Mr. P. Raja Kumar, Assistant Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ayushi Sharma(18SCSE1180030)
Jyotsna Pathak(18SCSE1010092)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. P. Raja Kumar
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Ayushi Sharma(18SCSE1180030) and Jyotsna Pathak(18SCSE1010092) has been held on December, 2021 and her work is recommended for the award of B.TECH.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021
Place: Greater Noida

Acknowledgement

I wish to express our heartfelt gratitude to the all the people who have played a crucial role in the research for this project, without their active cooperation the preparation of this project could not have been completed within the specified time limit.

We are thankful to our respected Dean to give us this opportunity and for motivating us to complete this project with complete focus and attention. I would also like to thank, Varun Tiwari for supporting and helping us throughout the project.

I am also thankful to my project guide Mr. P. Raja Kumar who supported me throughout this project with utmost cooperation and patience and for helping me in doing this Project.

Abstract

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. We will be focusing on the following models in particular: Simple Linear Regression, Multiple Linear Regression model, Support Vector Regression model, Random Forest model on the datasets to check which model gives the highest accuracy. Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. 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 change as the independent variable(s) change. Simple linear regression is used to estimate the relationship between two quantitative variables. Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc. Spam Filtering, Random Forest, Decision Trees, Logistic Regression, Support vector Machines

Tools and Technologies used: Python, Kaggle, Machine Learning libraries

Table Of Contents

CHAPTER-1
Introduction

1.1 INTRODUCTION

SIMPLE LINEAR REGRESSION

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 change as the independent variable(s) change. Simple linear regression is used to estimate the relationship between two quantitative variables. You can use simple linear regression when you want to know:

1. How strong the relationship is between two variables (e.g. the relationship between rainfall and soil erosion).
2. 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).

CHAPTER-1
Introduction

MULTIPLE LINEAR REGRESSION

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable. MLR is used extensively in econometrics and financial inference.
It is sometimes known simply as multiple regression, and it is an extension of linear regression.
The variable that we want to predict is known as the dependent variable, while the variables we use to predict the value of the dependent variable are known as independent or explanatory variables.

CHAPTER-1
Introduction

RANDOM FOREST REGRESSION

Random Forest Regression is **a supervised learning algorithm that uses ensemble learning method for regression**. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model.

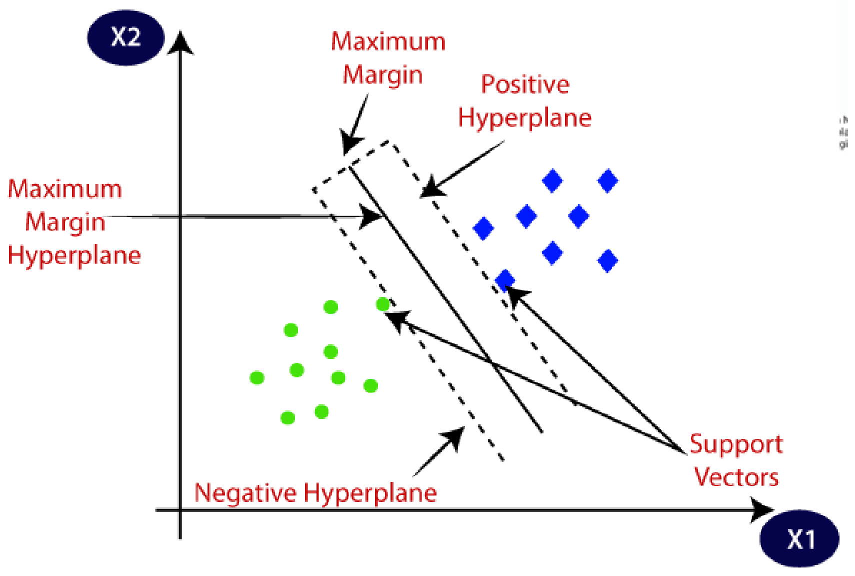
CHAPTER-1
Introduction

SUPPORT VECTOR REGRESSION

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



CHAPTER-1
Introduction

1.2 FORMULATION OF PROBLEM

Machine learning uses programmed algorithms that receive and analyse input data to predict output values within an acceptable range. As new data is fed to these algorithms, they learn and optimise their operations to improve performance, developing ‘intelligence’ over time. There are four types of machine learning algorithms: supervised, semi-supervised, unsupervised and reinforcement. Learning must generally be supervised: Training data must be tagged, Machine learning is stochastic, not deterministic, some of the limitations of ml algorithms.

PROPOSED SOLUTION

We’ll be doing comparisons between ***simple linear regression, multiple linear regression, support vector regression and random forest*** to understand which algorithm works best for various models. Also, checking the accuracy for each system.

CHAPTER-1
Introduction

1.2.1 TOOL AND TECHNOLOGY USED:

Software Requirement:

Kaggle:

It is a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Programming Languages:

- Python:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today

CHAPTER-2
Literature Survey

Simple linear regression

The formula for a simple linear regression is:

$$y = \beta_0 + \beta_1 X + \varepsilon$$

- 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.

Linear regression finds the line of best fit through your data by searching for the regression coefficient (B1) that minimizes the total error (e) of the model. While you can perform a linear regression by hand, this is a tedious process, so most people use statistical programs to help them quickly analyses the data.

CHAPTER-2 Literature Survey

Multiple linear regression

The formula for multiple linear regression is:

$$y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \varepsilon$$

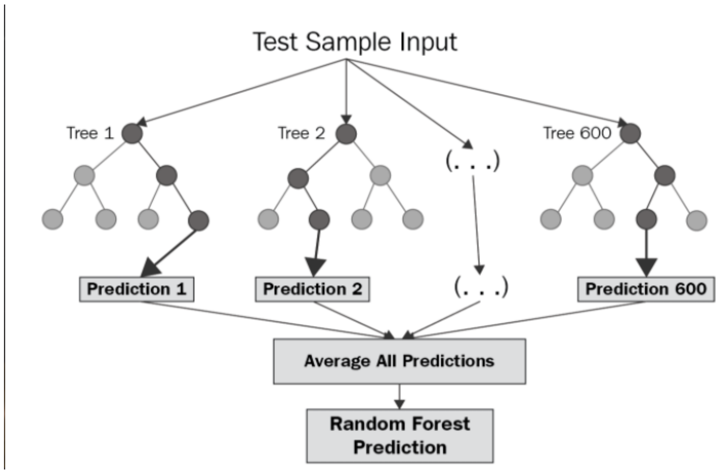
- y = the predicted value of the dependent variable
- B0 = the y-intercept (value of y when all other parameters are set to 0)
- B1X1= the regression coefficient (B1) of the first independent variable (X1) (a.k.a. the effect that increases the value of the independent variable has on the predicted y value)
- … = do the same for however many independent variables you are testing
- BnXn = the regression coefficient of the last independent variable
- e = model error (a.k.a. how much variation there is in our estimate of y)

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

CHAPTER-2
Literature Survey

Random Forest Regression

Random Forest Regression is a supervised learning algorithm that uses the ensemble learning method for regression. The ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



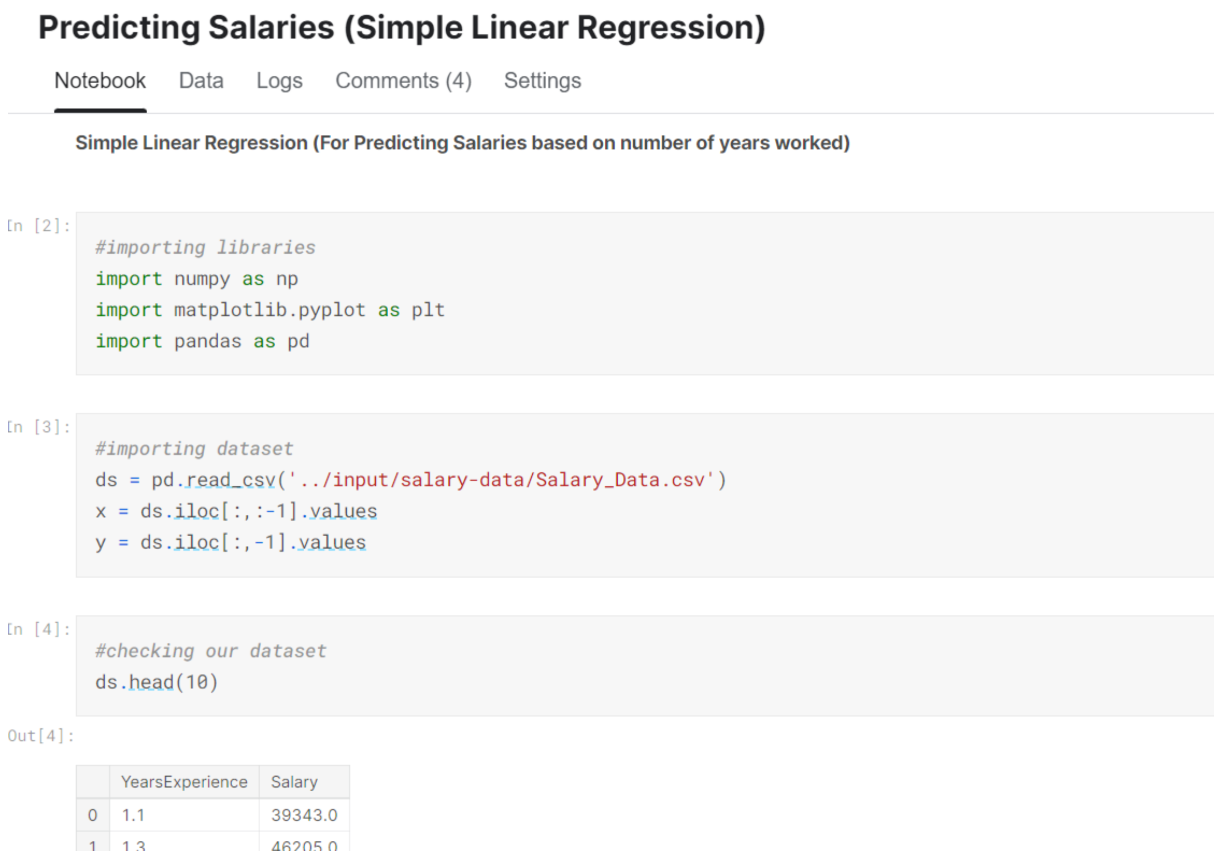
The diagram above shows the structure of a Random Forest. We notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let’s walk through the steps:

a) Pick at random *k* data points from the training set.
b) Build a decision tree associated to these *k* data points.
c) Choose the number *N* of trees you want to build and repeat steps 1 and 2.
d) For a new data point, make each one of your *N*-tree trees predict the value of *y* for the data point in question and assign the new data point to the average across all of the predicted *y* values.

CHAPTER-3
Working and Output

WORKING:

a) Simple Linear Regression ~ For predicting Salary based on no. of years worked



Predicting Salaries (Simple Linear Regression)

Simple Linear Regression (For Predicting Salaries based on number of years worked)

```
In [2]:
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

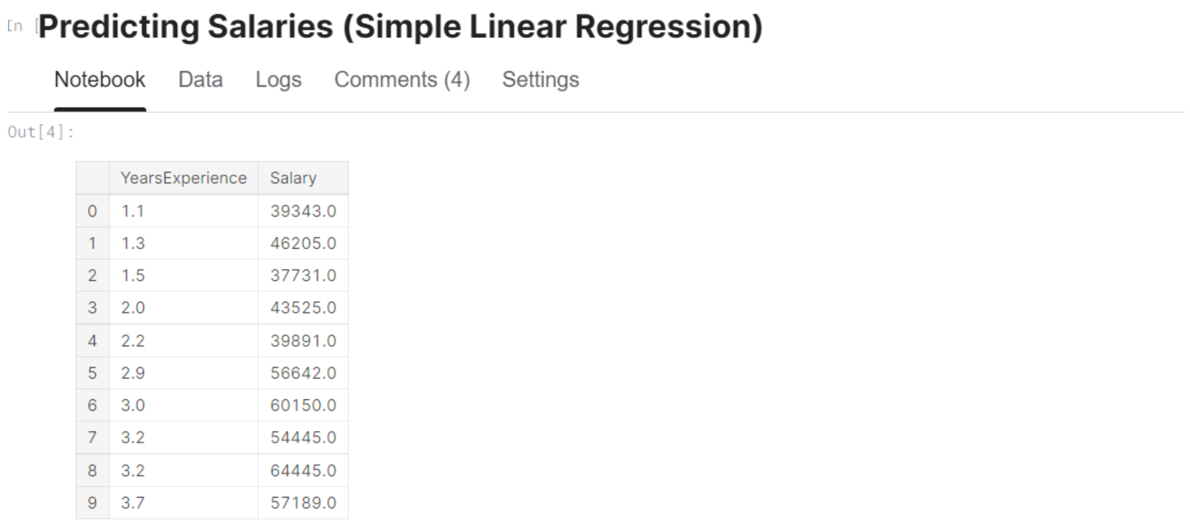
```
In [3]:
#importing dataset
ds = pd.read_csv('../input/salary-data/Salary_Data.csv')
x = ds.iloc[:,:-1].values
y = ds.iloc[:,-1].values
```

```
In [4]:
#checking our dataset
ds.head(10)
```

Out[4]:

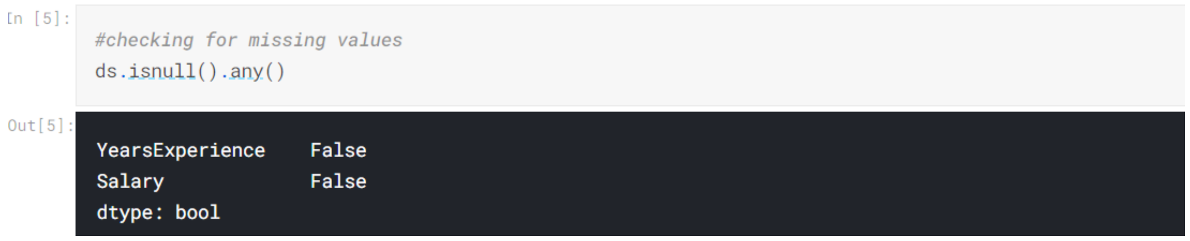
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0

CHAPTER-3
Working and Output



Predicting Salaries (Simple Linear Regression)

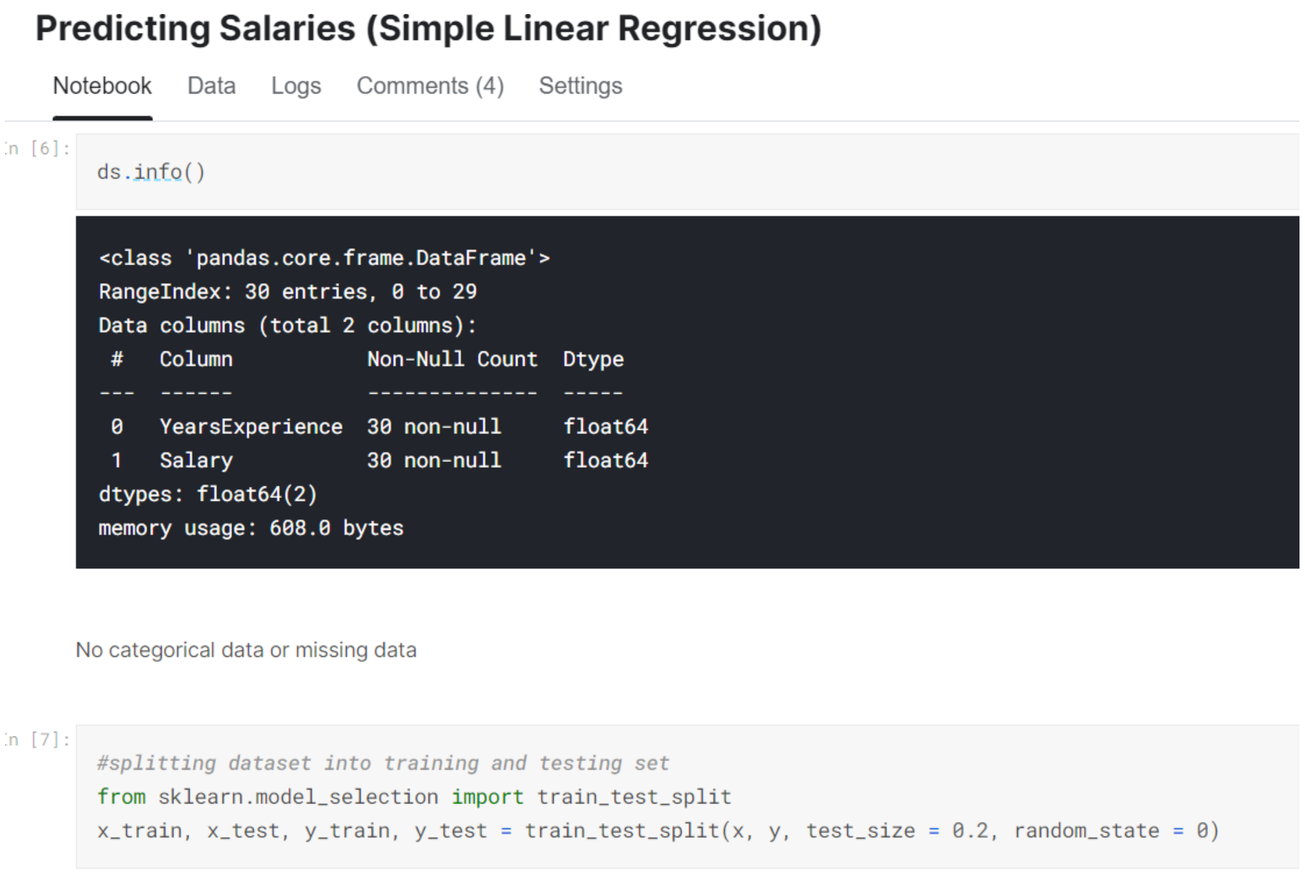
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0



```
#checking for missing values
ds.isnull().any()
```

```
YearsExperience    False
Salary             False
dtype: bool
```

CHAPTER-3
Working and Output



Predicting Salaries (Simple Linear Regression)

In [6]:

```
ds.info()
```

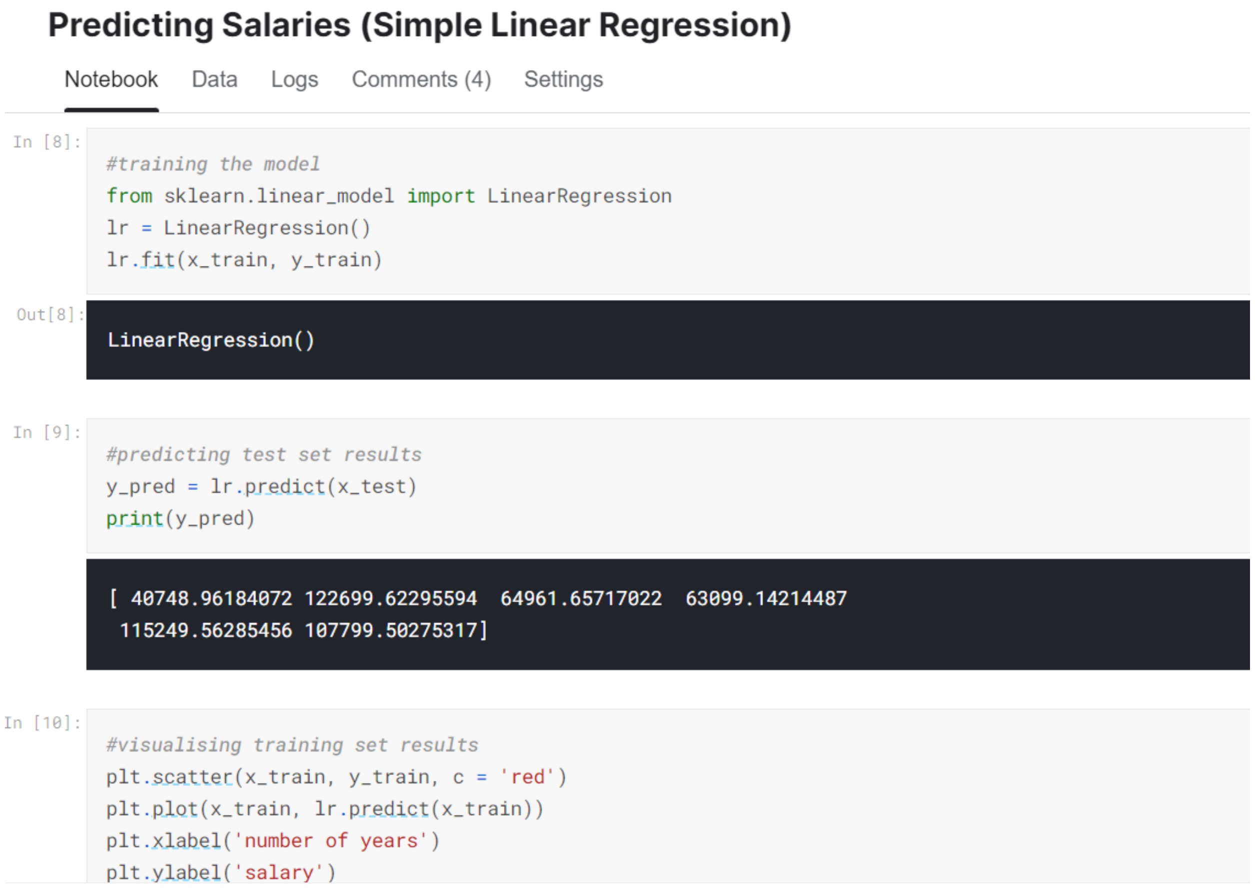
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)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

No categorical data or missing data

In [7]:

```
#splitting dataset into training and testing set
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)
```

CHAPTER-3
Working and Output



Predicting Salaries (Simple Linear Regression)

In [8]:
```
#training the model
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
```

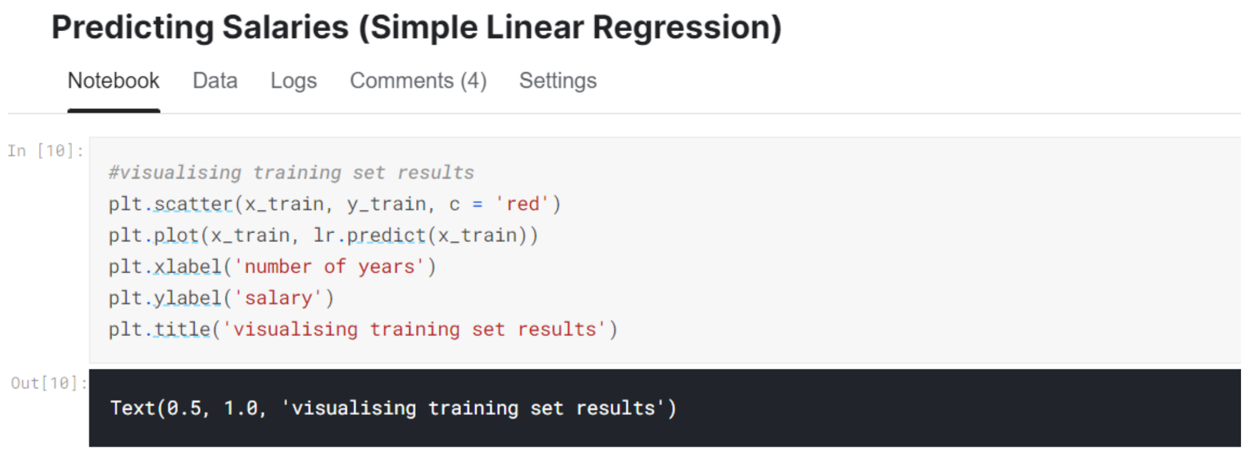
Out[8]:
```
LinearRegression()
```

In [9]:
```
#predicting test set results
y_pred = lr.predict(x_test)
print(y_pred)
```

```
[ 40748.96184072 122699.62295594  64961.65717022  63099.14214487
 115249.56285456 107799.50275317]
```

In [10]:
```
#visualising training set results
plt.scatter(x_train, y_train, c = 'red')
plt.plot(x_train, lr.predict(x_train))
plt.xlabel('number of years')
plt.ylabel('salary')
```

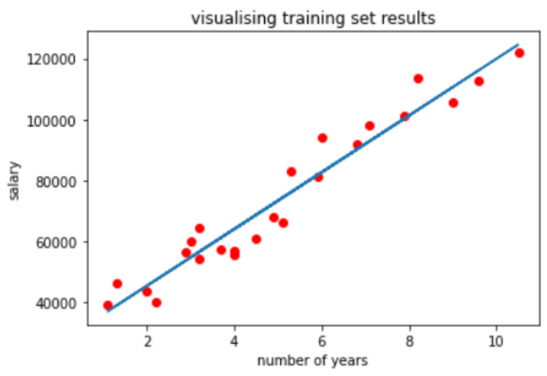
CHAPTER-3
Working and Output



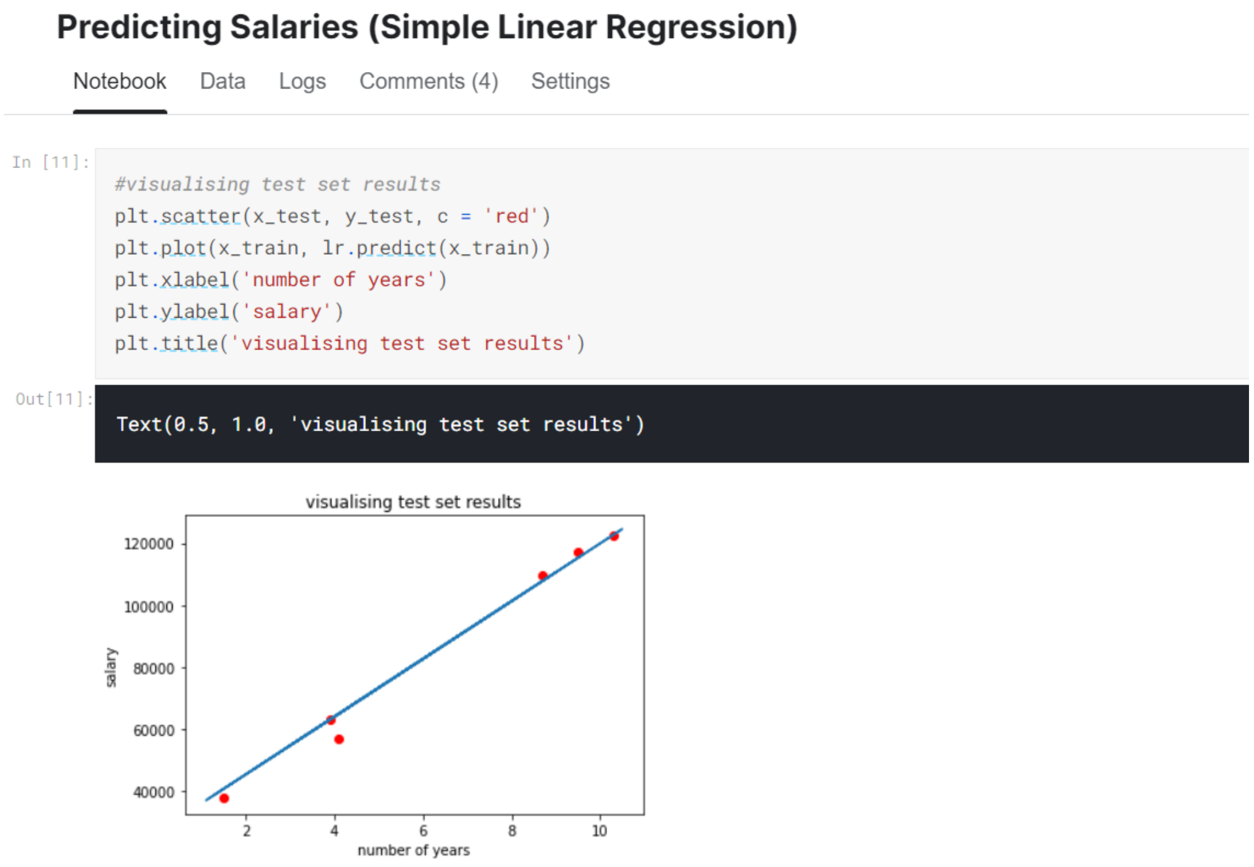
Predicting Salaries (Simple Linear Regression)

In [10]:
```
#visualising training set results
plt.scatter(x_train, y_train, c = 'red')
plt.plot(x_train, lr.predict(x_train))
plt.xlabel('number of years')
plt.ylabel('salary')
plt.title('visualising training set results')
```

Out[10]:
```
Text(0.5, 1.0, 'visualising training set results')
```



CHAPTER-3
Working and Output



Predicting Salaries (Simple Linear Regression)

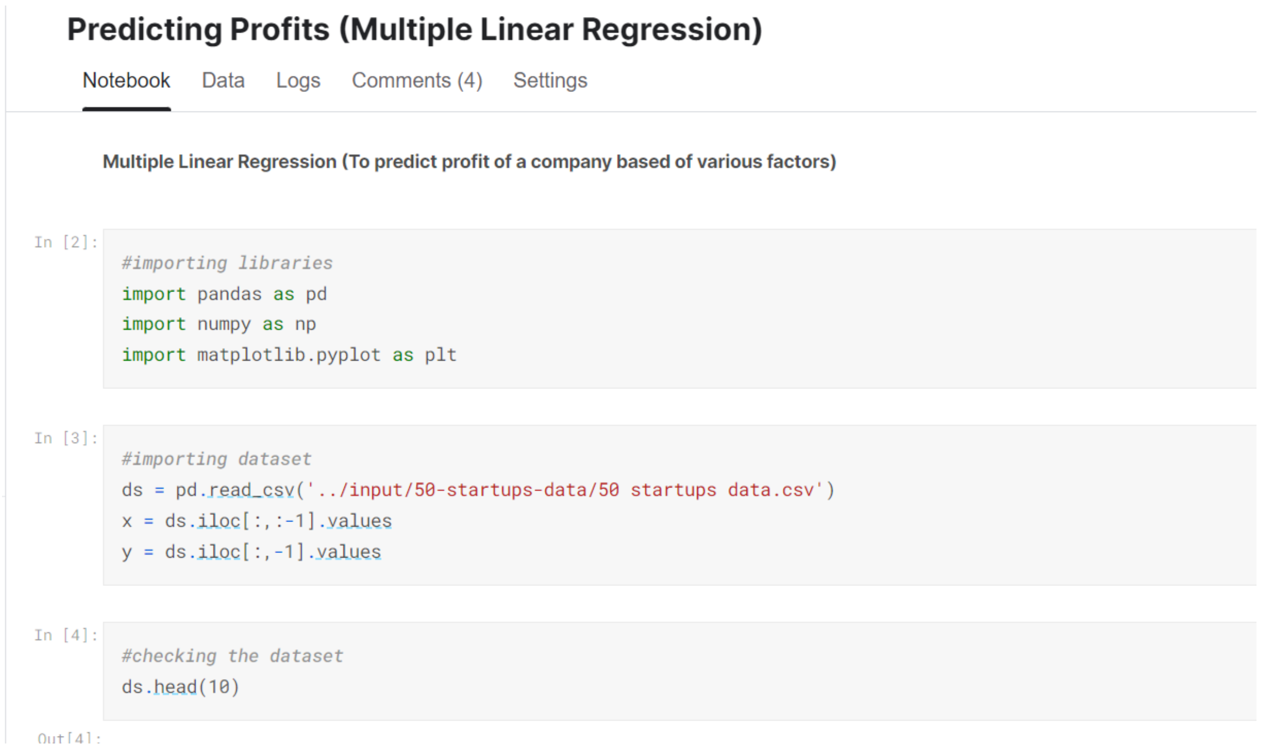
Notebook Data Logs Comments (4) Settings

In [11]:
```
#visualising test set results
plt.scatter(x_test, y_test, c = 'red')
plt.plot(x_train, lr.predict(x_train))
plt.xlabel('number of years')
plt.ylabel('salary')
plt.title('visualising test set results')
```

Out[11]:
```
Text(0.5, 1.0, 'visualising test set results')
```

CHAPTER-3
Working and Output

b) Multiple Linear Regression ~ To Predict profit of a company based of various factors



Predicting Profits (Multiple Linear Regression)

Multiple Linear Regression (To predict profit of a company based of various factors)

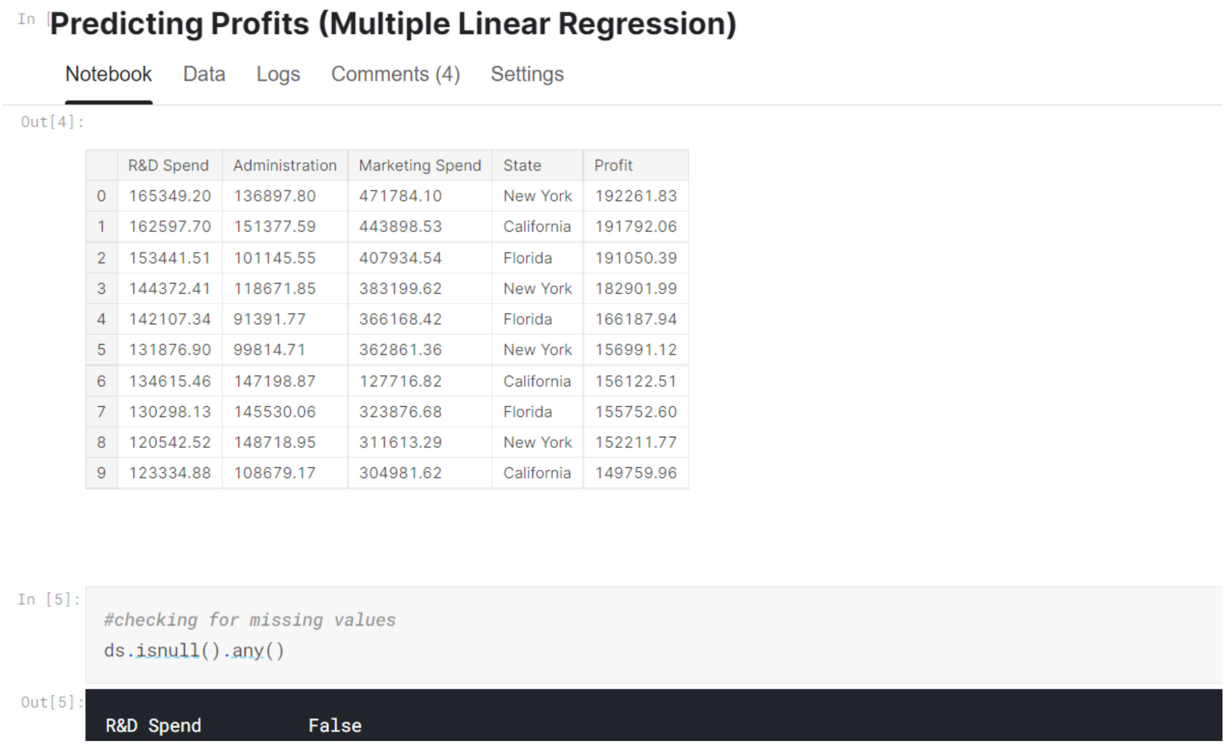
In [2]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
#importing dataset
ds = pd.read_csv('../input/50-startups-data/50 startups data.csv')
x = ds.iloc[:,:-1].values
y = ds.iloc[:,-1].values
```

In [4]:
```
#checking the dataset
ds.head(10)
```

Out[4]:

CHAPTER-3
Working and Output



In [

Predicting Profits (Multiple Linear Regression)

Out[4]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96

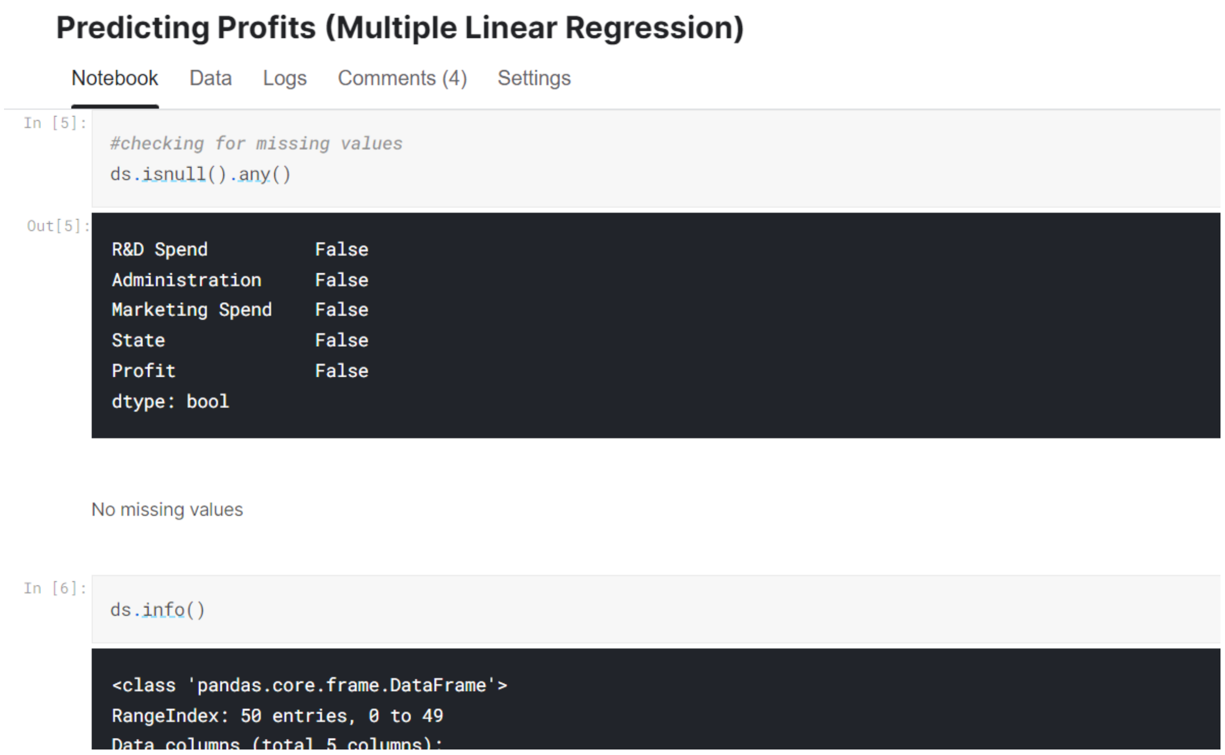
In [5]:

```
#checking for missing values
ds.isnull().any()
```

Out[5]:

```
R&D Spend          False
```

CHAPTER-3
Working and Output



Predicting Profits (Multiple Linear Regression)

In [5]:
```
#checking for missing values
ds.isnull().any()
```

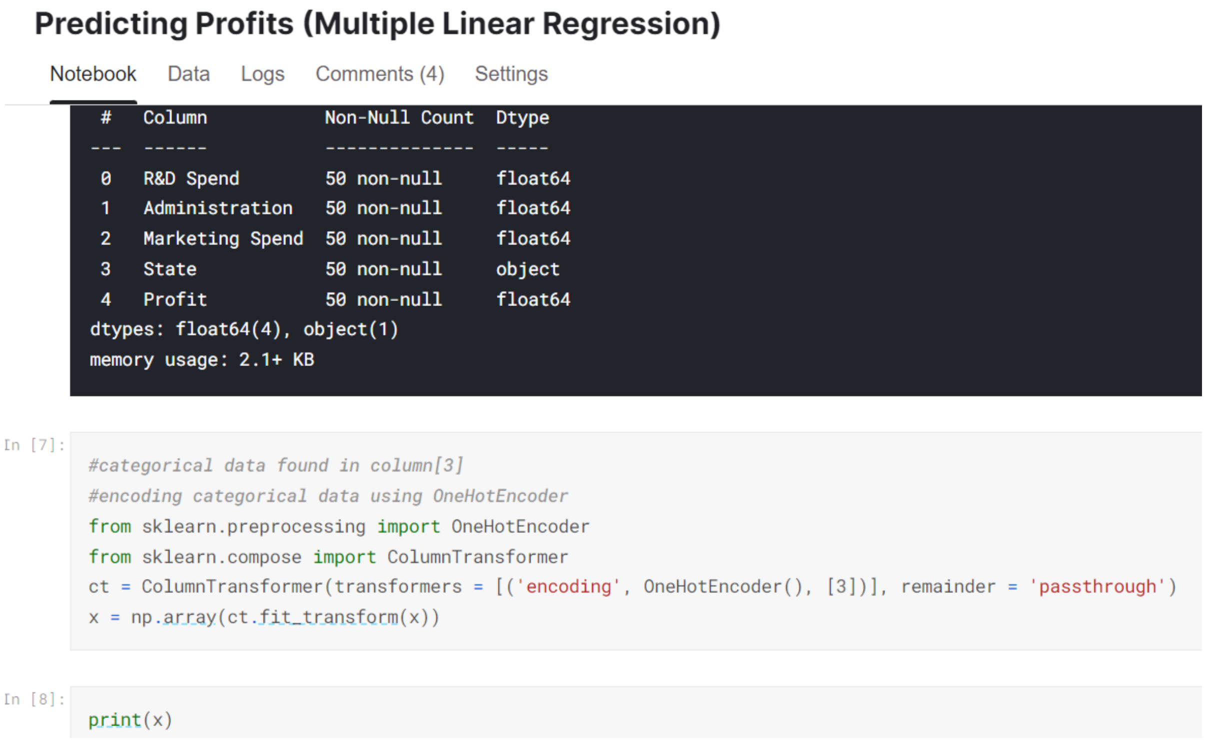
Out[5]:
```
R&D Spend          False
Administration     False
Marketing Spend    False
State              False
Profit             False
dtype: bool
```

No missing values

In [6]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
```

CHAPTER-3
Working and Output



Predicting Profits (Multiple Linear Regression)

Notebook Data Logs Comments (4) Settings

```
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

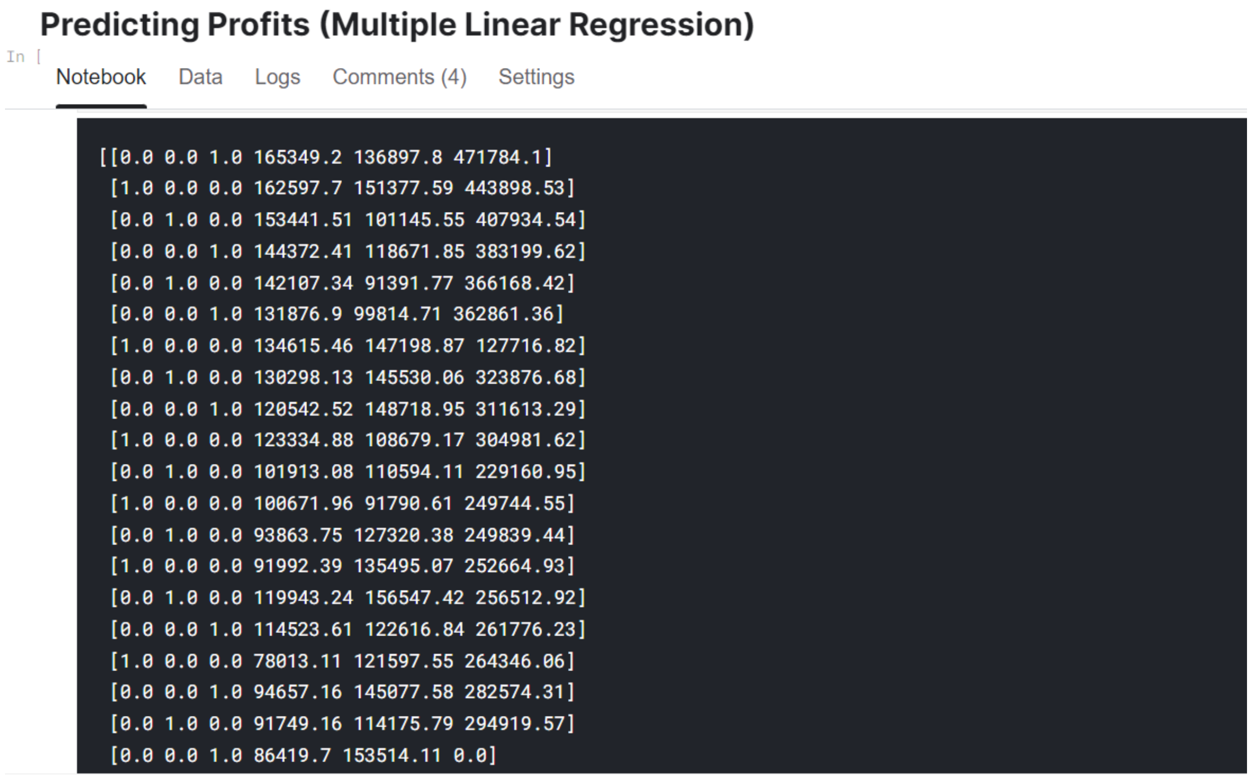
In [7]:

```
#categorical data found in column[3]
#encoding categorical data using OneHotEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer(transformers = [('encoding', OneHotEncoder(), [3])], remainder = 'passthrough')
x = np.array(ct.fit_transform(x))
```

In [8]:

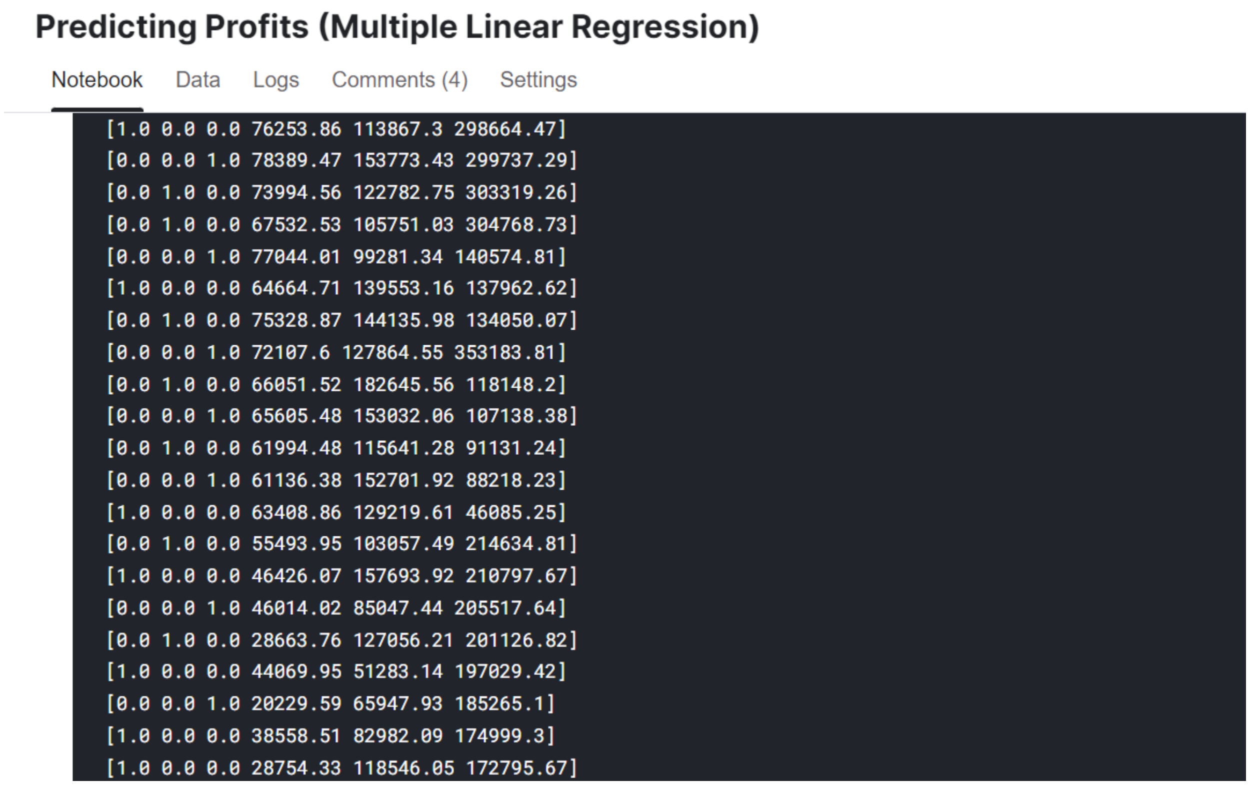
```
print(x)
```

CHAPTER-3
Working and Output



```
[[0.0 0.0 1.0 165349.2 136897.8 471784.1]
 [1.0 0.0 0.0 162597.7 151377.59 443898.53]
 [0.0 1.0 0.0 153441.51 101145.55 407934.54]
 [0.0 0.0 1.0 144372.41 118671.85 383199.62]
 [0.0 1.0 0.0 142107.34 91391.77 366168.42]
 [0.0 0.0 1.0 131876.9 99814.71 362861.36]
 [1.0 0.0 0.0 134615.46 147198.87 127716.82]
 [0.0 1.0 0.0 130298.13 145530.06 323876.68]
 [0.0 0.0 1.0 120542.52 148718.95 311613.29]
 [1.0 0.0 0.0 123334.88 108679.17 304981.62]
 [0.0 1.0 0.0 101913.08 110594.11 229160.95]
 [1.0 0.0 0.0 100671.96 91790.61 249744.55]
 [0.0 1.0 0.0 93863.75 127320.38 249839.44]
 [1.0 0.0 0.0 91992.39 135495.07 252664.93]
 [0.0 1.0 0.0 119943.24 156547.42 256512.92]
 [0.0 0.0 1.0 114523.61 122616.84 261776.23]
 [1.0 0.0 0.0 78013.11 121597.55 264346.06]
 [0.0 0.0 1.0 94657.16 145077.58 282574.31]
 [0.0 1.0 0.0 91749.16 114175.79 294919.57]
 [0.0 0.0 1.0 86419.7 153514.11 0.0]
```

CHAPTER-3
Working and Output

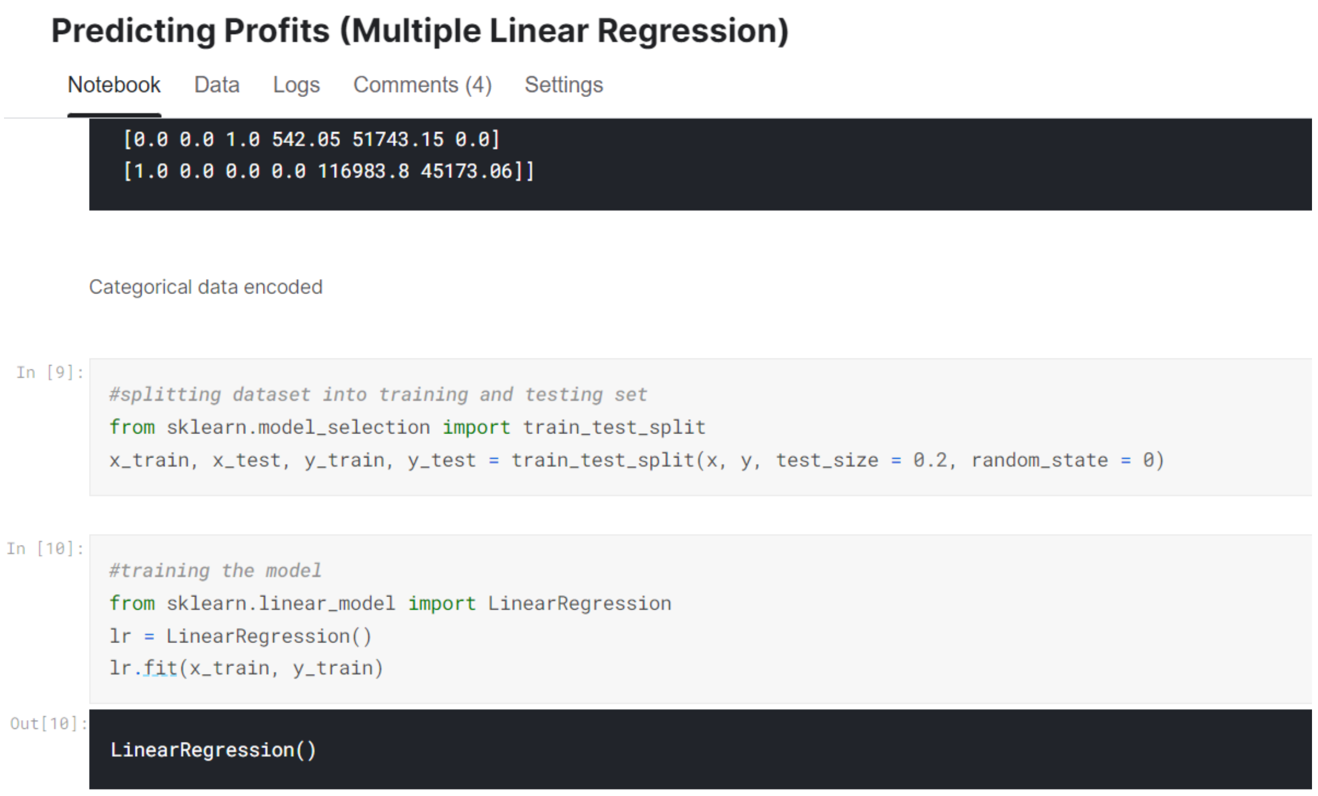


Predicting Profits (Multiple Linear Regression)

Notebook Data Logs Comments (4) Settings

```
[1.0 0.0 0.0 76253.86 113867.3 298664.47]
[0.0 0.0 1.0 78389.47 153773.43 299737.29]
[0.0 1.0 0.0 73994.56 122782.75 303319.26]
[0.0 1.0 0.0 67532.53 105751.03 304768.73]
[0.0 0.0 1.0 77044.01 99281.34 140574.81]
[1.0 0.0 0.0 64664.71 139553.16 137962.62]
[0.0 1.0 0.0 75328.87 144135.98 134050.07]
[0.0 0.0 1.0 72107.6 127864.55 353183.81]
[0.0 1.0 0.0 66051.52 182645.56 118148.2]
[0.0 0.0 1.0 65605.48 153032.06 107138.38]
[0.0 1.0 0.0 61994.48 115641.28 91131.24]
[0.0 0.0 1.0 61136.38 152701.92 88218.23]
[1.0 0.0 0.0 63408.86 129219.61 46085.25]
[0.0 1.0 0.0 55493.95 103057.49 214634.81]
[1.0 0.0 0.0 46426.07 157693.92 210797.67]
[0.0 0.0 1.0 46014.02 85047.44 205517.64]
[0.0 1.0 0.0 28663.76 127056.21 201126.82]
[1.0 0.0 0.0 44069.95 51283.14 197029.42]
[0.0 0.0 1.0 20229.59 65947.93 185265.1]
[1.0 0.0 0.0 38558.51 82982.09 174999.3]
[1.0 0.0 0.0 28754.33 118546.05 172795.67]
```

CHAPTER-3
Working and Output



```
 [0.0 0.0 1.0 542.05 51743.15 0.0]
 [1.0 0.0 0.0 0.0 116983.8 45173.06]]
```

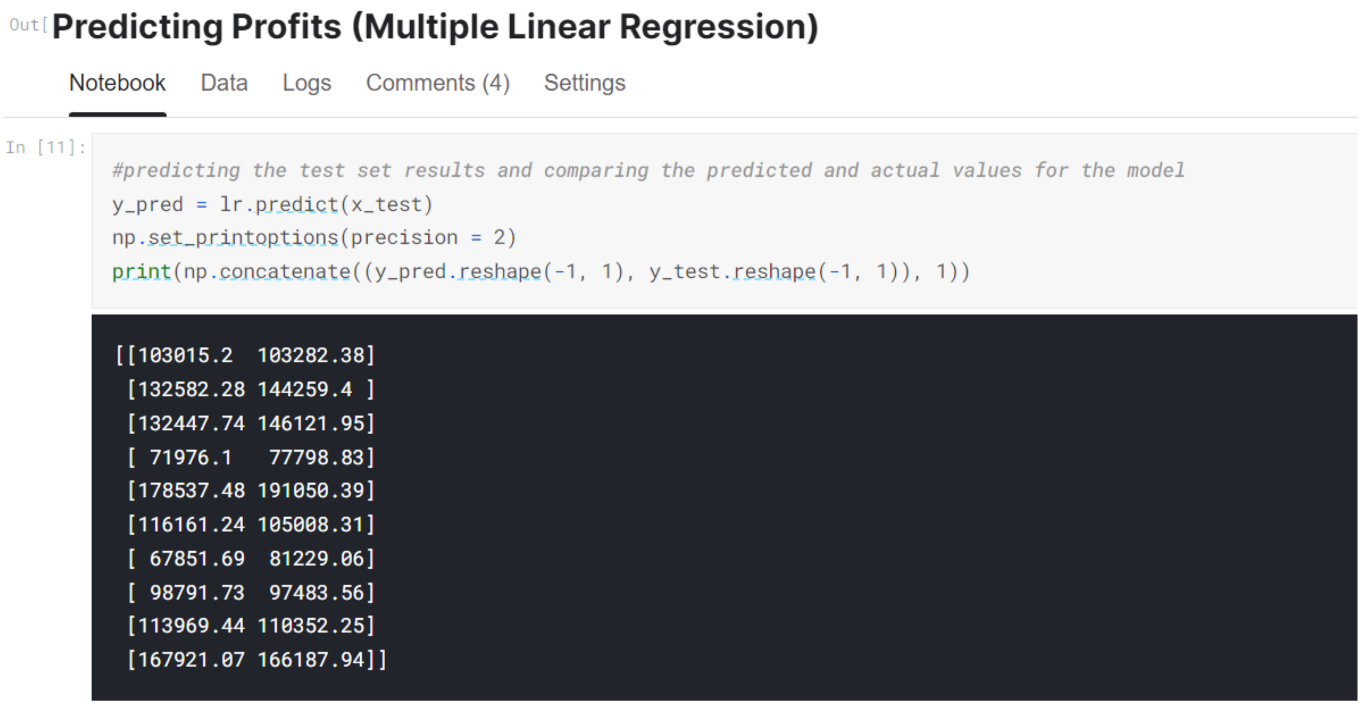
Categorical data encoded

In [9]:
```
#splitting dataset into training and testing set
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 [10]:
```
#training the model
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[10]:
```
LinearRegression()
```

CHAPTER-3
Working and Output



Out[**Predicting Profits (Multiple Linear Regression)**

Notebook Data Logs Comments (4) Settings

In [11]:

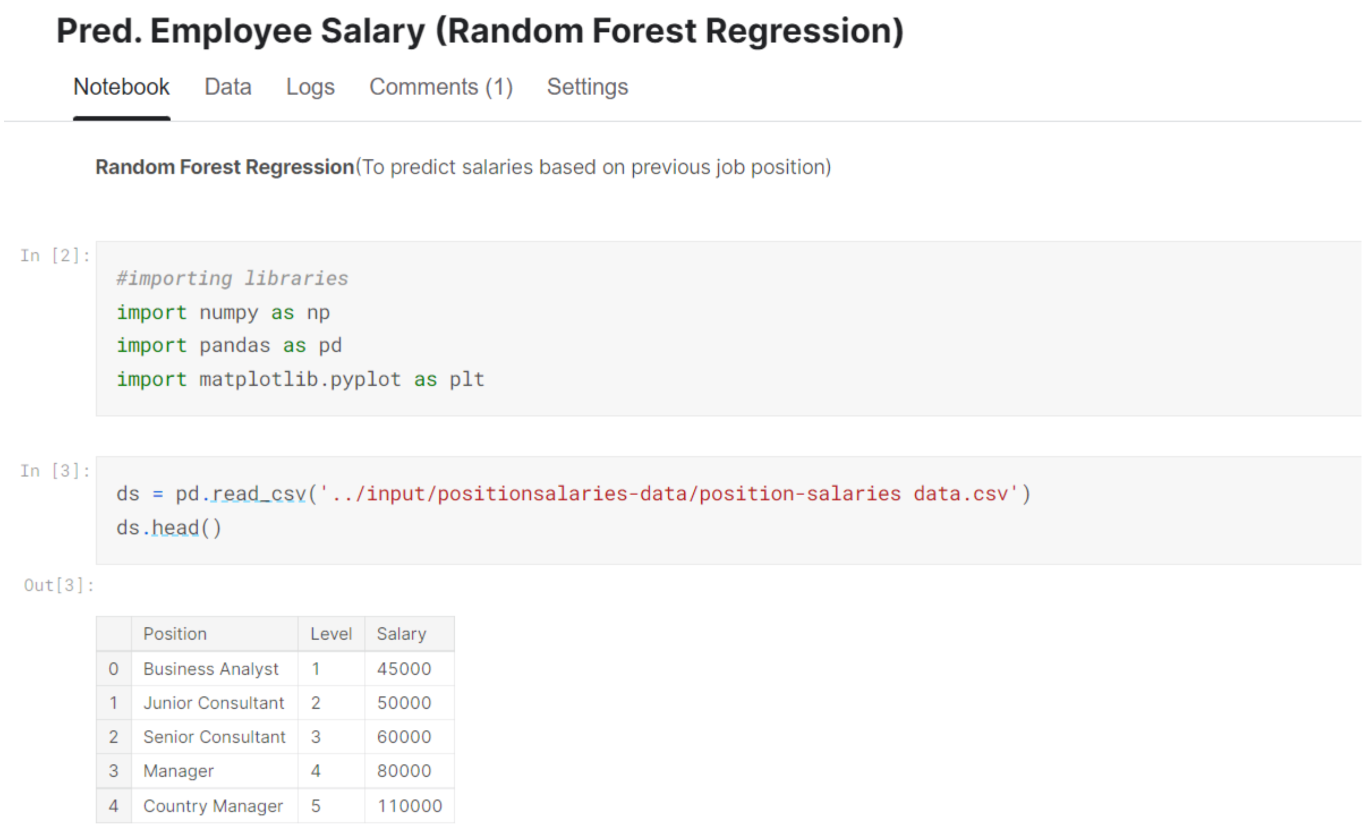
```
#predicting the test set results and comparing the predicted and actual values for the model
y_pred = lr.predict(x_test)
np.set_printoptions(precision = 2)
print(np.concatenate((y_pred.reshape(-1, 1), y_test.reshape(-1, 1)), 1))
```

```
[[103015.2  103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1   77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

Column[0] contains the predicted vlues and column[1] contains the actual values. The model works well.

CHAPTER-3
Working and Output

c) Random Forest Regression ~ To predict Salary based on previous job position



Pred. Employee Salary (Random Forest Regression)

Random Forest Regression(To predict salaries based on previous job position)

In [2]:

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

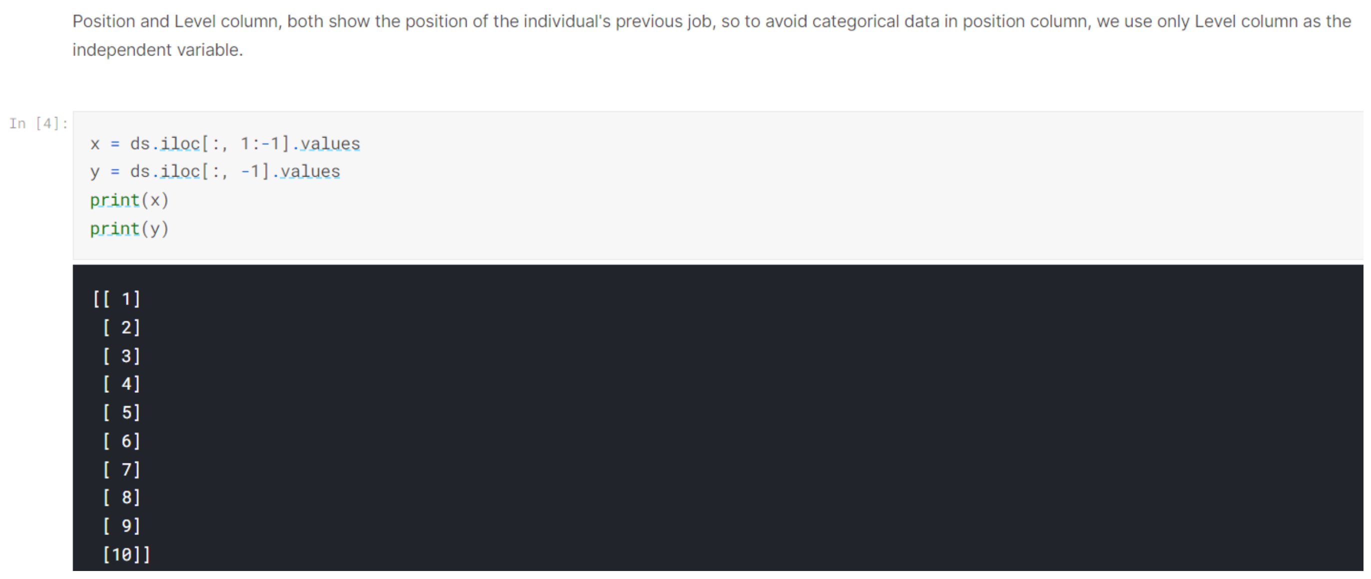
In [3]:

```
ds = pd.read_csv('../input/positionsalaries-data/position-salaries data.csv')
ds.head()
```

Out[3]:

```
   Position            Level  Salary
0  Business Analyst    1      45000
1  Junior Consultant   2      50000
2  Senior Consultant   3      60000
3  Manager             4      80000
4  Country Manager     5      110000
```

CHAPTER-3
Working and Output



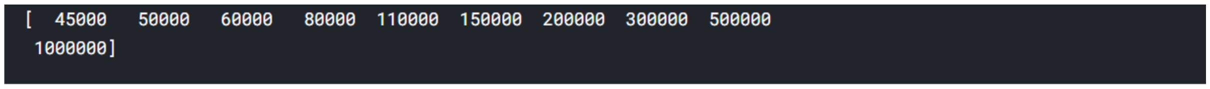
Position and Level column, both show the position of the individual's previous job, so to avoid categorical data in position column, we use only Level column as the independent variable.

In [4]:

```
x = ds.iloc[:, 1:-1].values
y = ds.iloc[:, -1].values
print(x)
print(y)
```

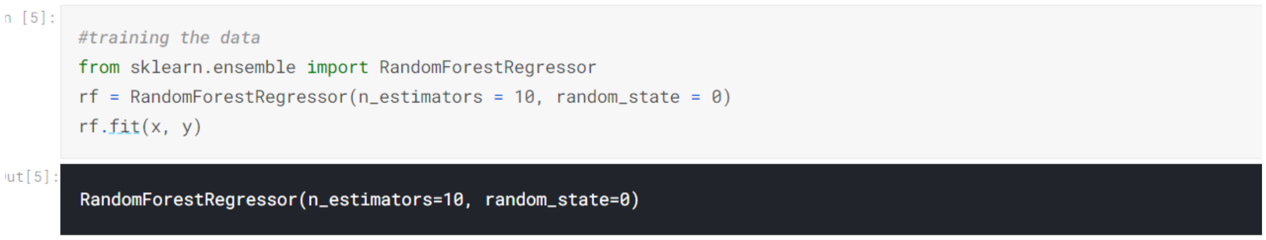
```
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]
```

CHAPTER-3
Working and Output



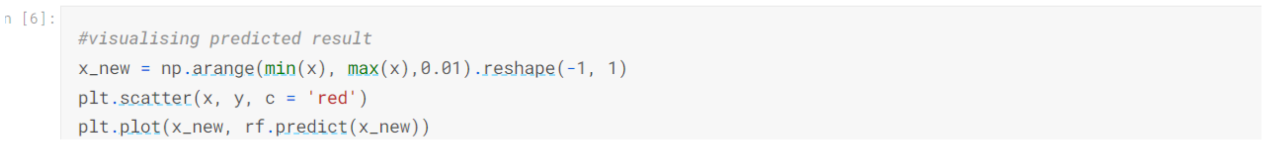
```
[  45000   50000   60000   80000  110000  150000  200000  300000  500000
 1000000]
```

We do not split the data since in the dataset there are only 10 values. So, splitting the data further into test and train would decrease the accuracy.



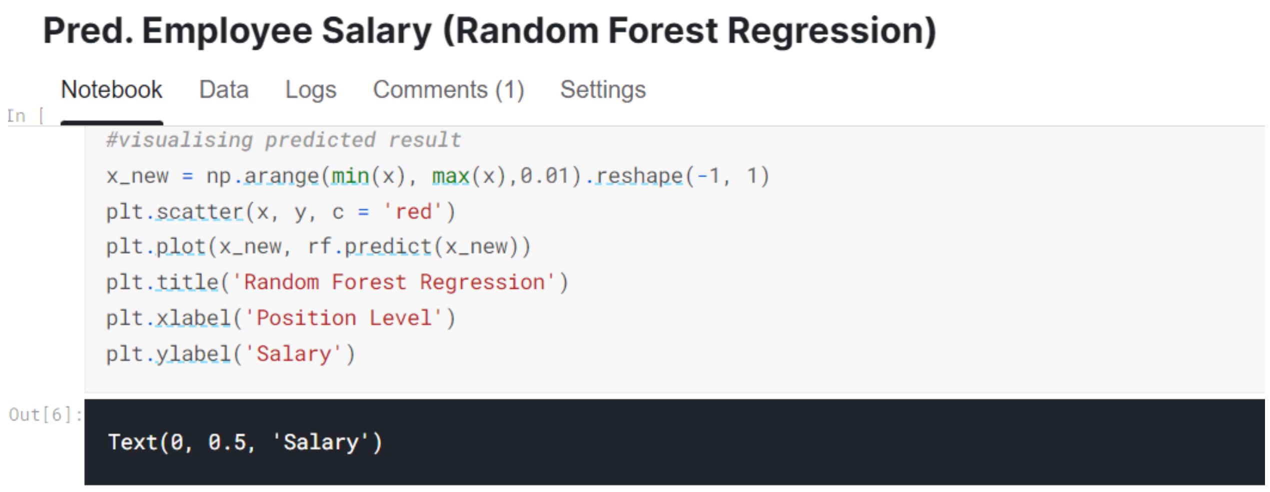
```
n [5]:
#training the data
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 10, random_state = 0)
rf.fit(x, y)
```

```
ut[5]:
RandomForestRegressor(n_estimators=10, random_state=0)
```



```
n [6]:
#visualising predicted result
x_new = np.arange(min(x), max(x),0.01).reshape(-1, 1)
plt.scatter(x, y, c = 'red')
plt.plot(x_new, rf.predict(x_new))
```

CHAPTER-3
Working and Output

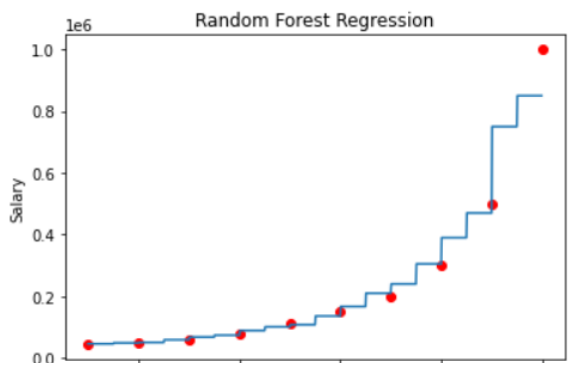


Pred. Employee Salary (Random Forest Regression)

```
#visualising predicted result
x_new = np.arange(min(x), max(x),0.01).reshape(-1, 1)
plt.scatter(x, y, c = 'red')
plt.plot(x_new, rf.predict(x_new))
plt.title('Random Forest Regression')
plt.xlabel('Position Level')
plt.ylabel('Salary')
```

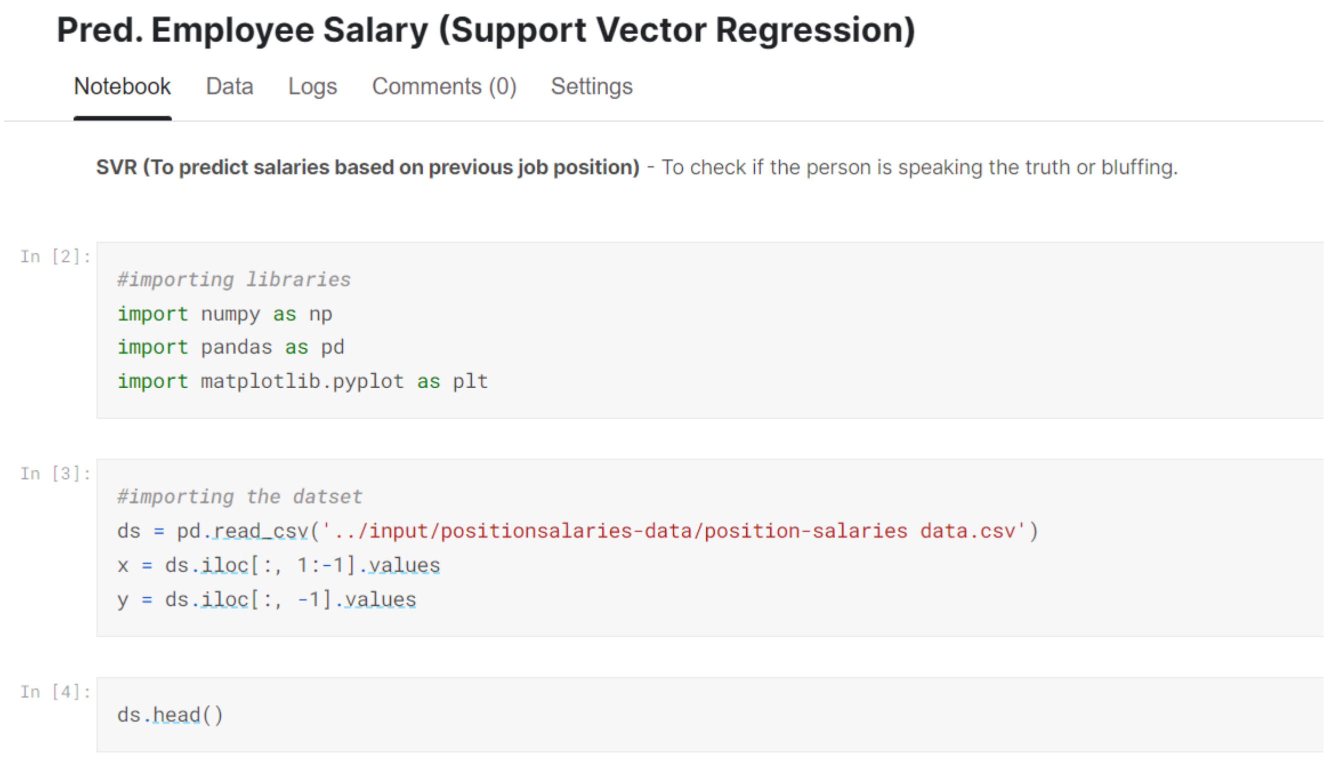
Out[6]:

```
Text(0, 0.5, 'Salary')
```



CHAPTER-3
Working and Output

d) Support Vector Regression ~ To predict Salary based on previous job position



Pred. Employee Salary (Support Vector Regression)

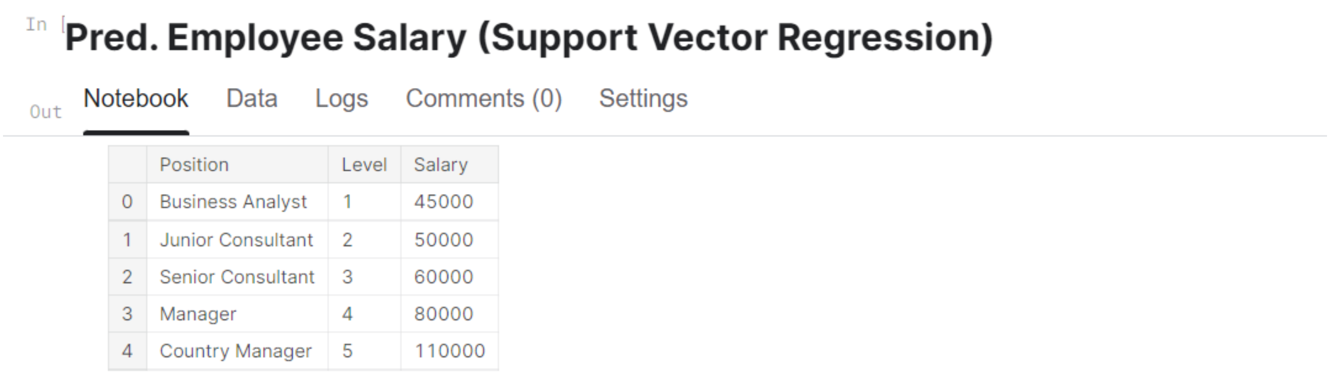
SVR (To predict salaries based on previous job position) - To check if the person is speaking the truth or bluffing.

In [2]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
#importing the datset
ds = pd.read_csv('../input/positionsalaries-data/position-salaries data.csv')
x = ds.iloc[:, 1:-1].values
y = ds.iloc[:, -1].values
```

In [4]:
```
ds.head()
```

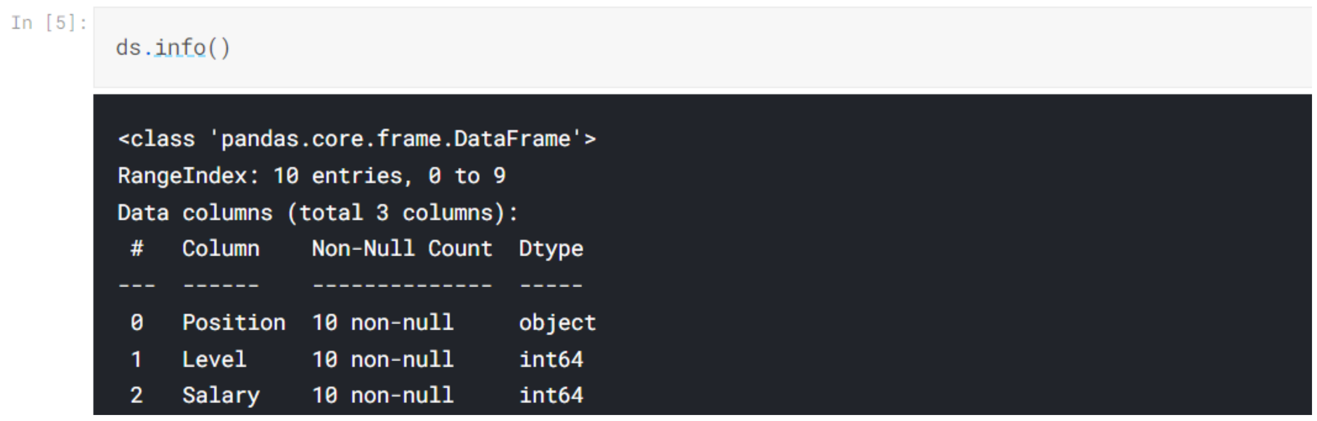
CHAPTER-3
Working and Output


In [

Pred. Employee Salary (Support Vector Regression)

Out

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000


In [5]:

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Position  10 non-null     object
 1   Level     10 non-null     int64
 2   Salary    10 non-null     int64
```

CHAPTER-3
Working and Output

Position and Level column, both show the position of the individual's previous job, so to avoid categorical data in position column, we use only Level column as the independent variable.

In [6]:

```
x = ds.iloc[:, 1:-1].values
y = ds.iloc[:, -1].values
print(x)
print(y)
```

```
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]
```

```
[  45000   50000   60000   80000  110000  150000  200000  300000  500000
 1000000]
```

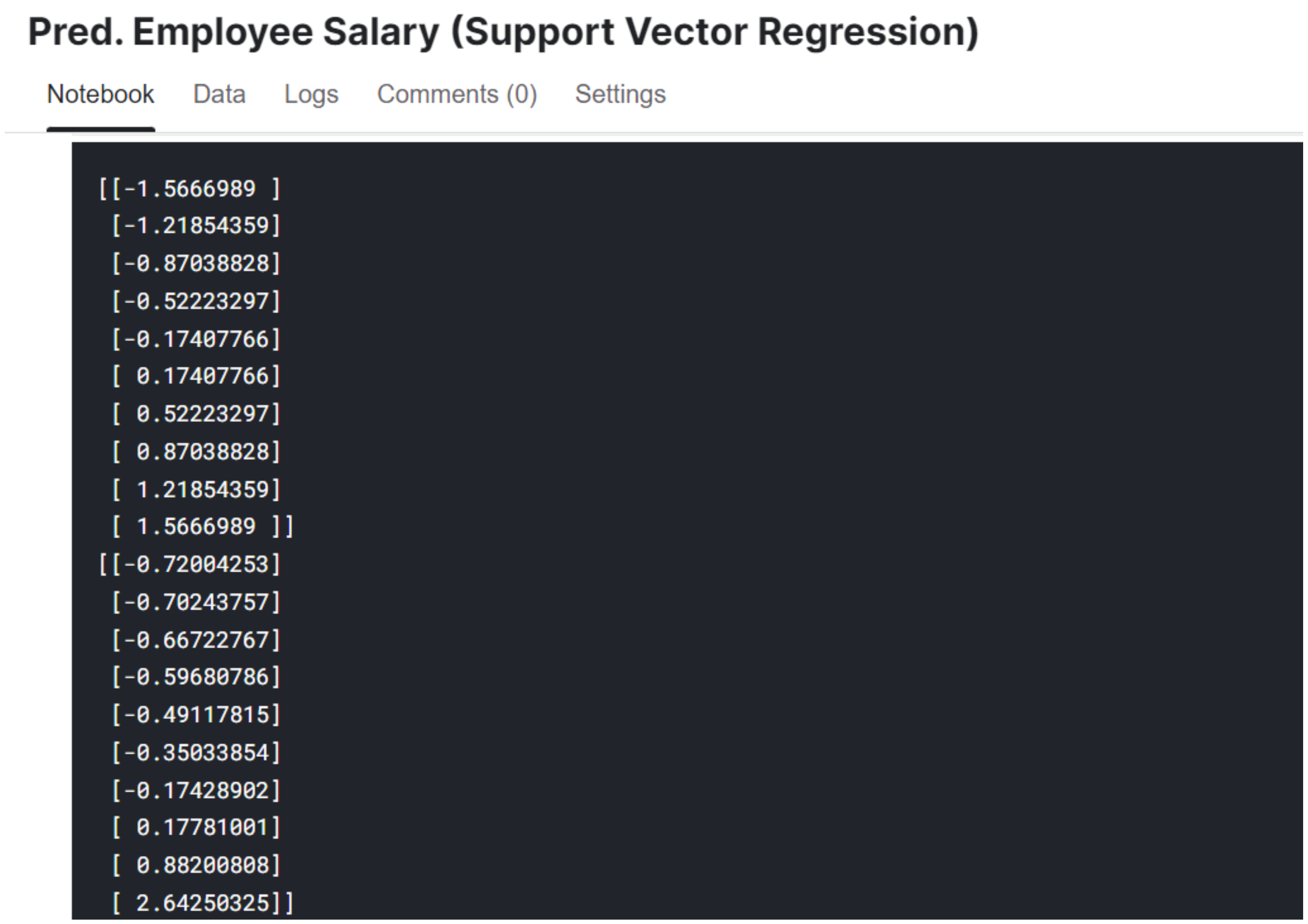
We do not split the data since in the dataset there are only 10 values. So, splitting the data further into test and train would decrease the accuracy.

Feature Scaling It is necessary in SVR since the it has an implicit equation for dependent and independent variables. **Standardization** is done.

n [7]:

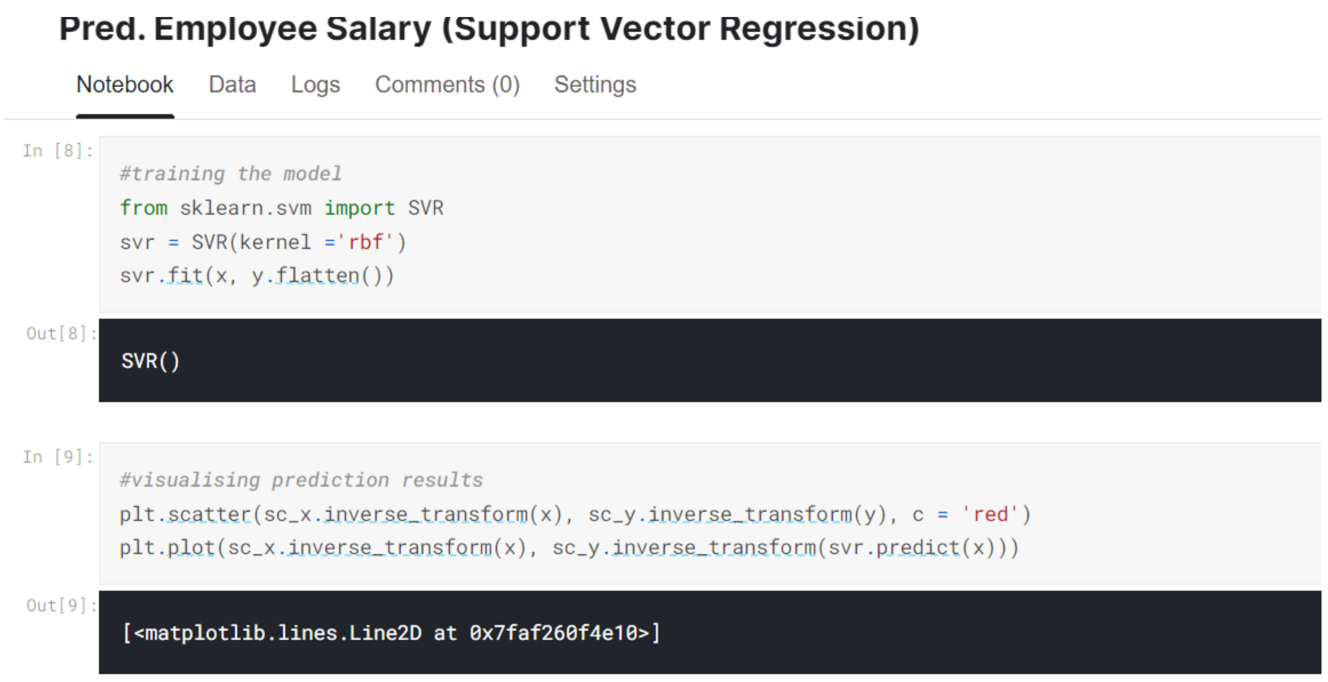
```
#applying feature  scaling
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
x = sc_x.fit_transform(x)
sc_y = StandardScaler()
y = sc_y.fit_transform(y.reshape(-1,1))
print(x)
print(y)
```

CHAPTER-3
Working and Output



```
[[-1.5666989 ]
 [-1.21854359]
 [-0.87038828]
 [-0.52223297]
 [-0.17407766]
 [ 0.17407766]
 [ 0.52223297]
 [ 0.87038828]
 [ 1.21854359]
 [ 1.5666989 ]]
[[-0.72004253]
 [-0.70243757]
 [-0.66722767]
 [-0.59680786]
 [-0.49117815]
 [-0.35033854]
 [-0.17428902]
 [ 0.17781001]
 [ 0.88200808]
 [ 2.64250325]]
```

CHAPTER-3
Working and Output



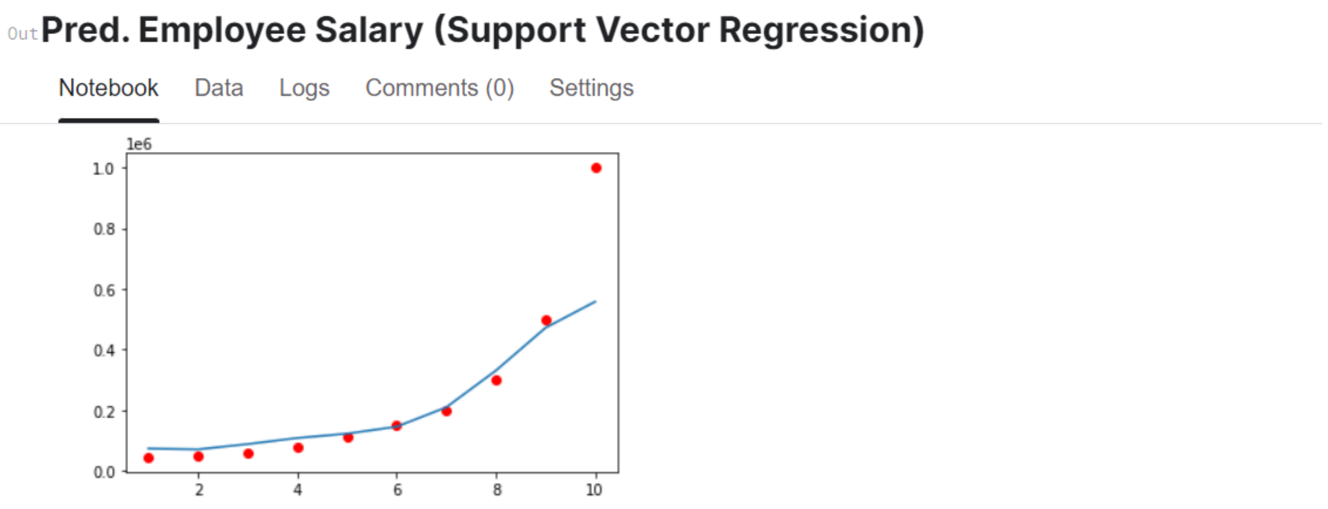
Pred. Employee Salary (Support Vector Regression)

```
In [8]:
#training the model
from sklearn.svm import SVR
svr = SVR(kernel ='rbf')
svr.fit(x, y.flatten())
```

```
Out[8]:
SVR()
```

```
In [9]:
#visualising prediction results
plt.scatter(sc_x.inverse_transform(x), sc_y.inverse_transform(y), c = 'red')
plt.plot(sc_x.inverse_transform(x), sc_y.inverse_transform(svr.predict(x)))
```

```
Out[9]:
[<matplotlib.lines.Line2D at 0x7faf260f4e10>]
```

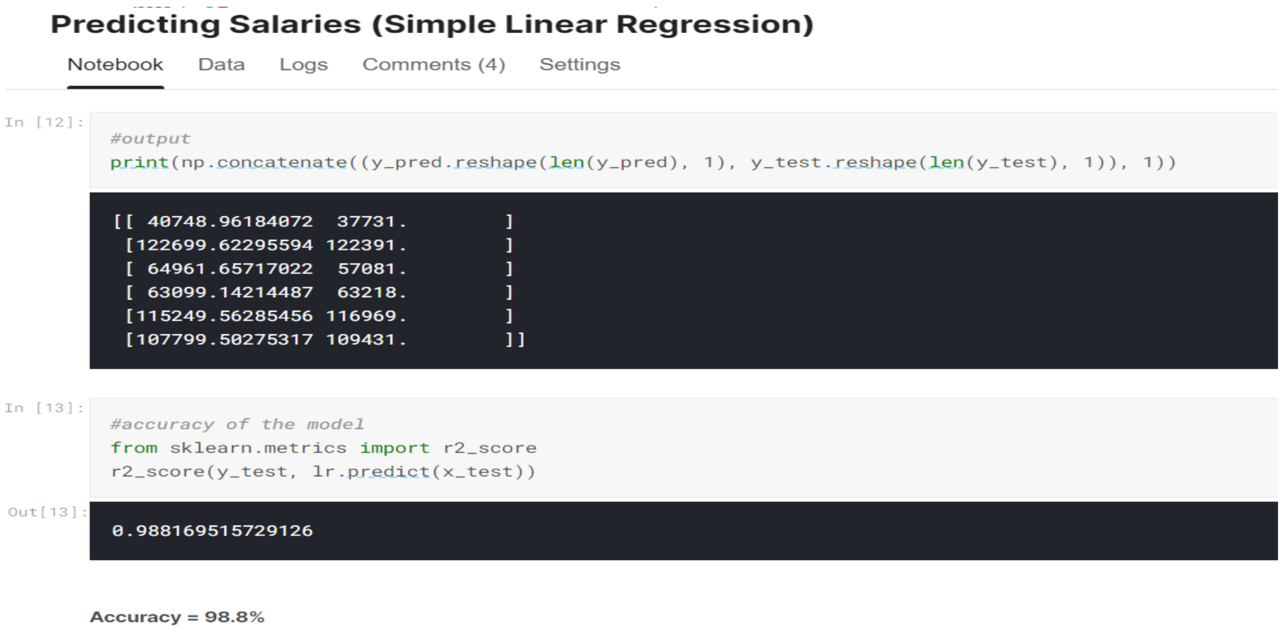


Out Pred. Employee Salary (Support Vector Regression)

CHAPTER-3
Working and Output

RESULTS AND OUTPUT:

- Accuracy of linear regression model = 98.8%



In [12]:
```
#output
print(np.concatenate((y_pred.reshape(len(y_pred), 1), y_test.reshape(len(y_test), 1)), 1))
```

```
[[ 40748.96184072  37731.        ]
 [122699.62295594 122391.        ]
 [ 64961.65717022  57081.        ]
 [ 63099.14214487  63218.        ]
 [115249.56285456 116969.        ]
 [107799.50275317 109431.        ]]
```

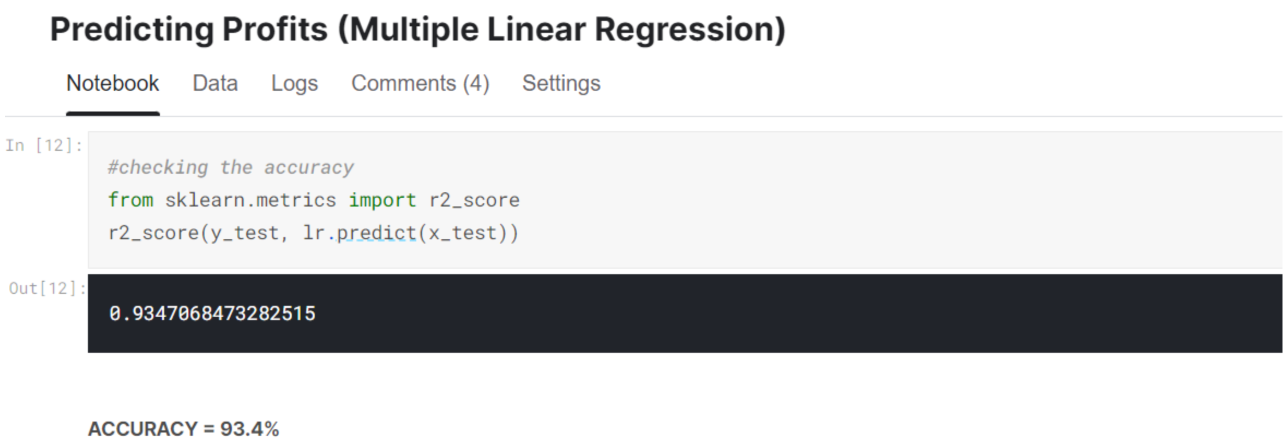
In [13]:
```
#accuracy of the model
from sklearn.metrics import r2_score
r2_score(y_test, lr.predict(x_test))
```

Out[13]:
```
0.988169515729126
```

Accuracy = 98.8%

CHAPTER-3
Working and Output

- Accuracy of multiple regression model = 93.4%



Predicting Profits (Multiple Linear Regression)

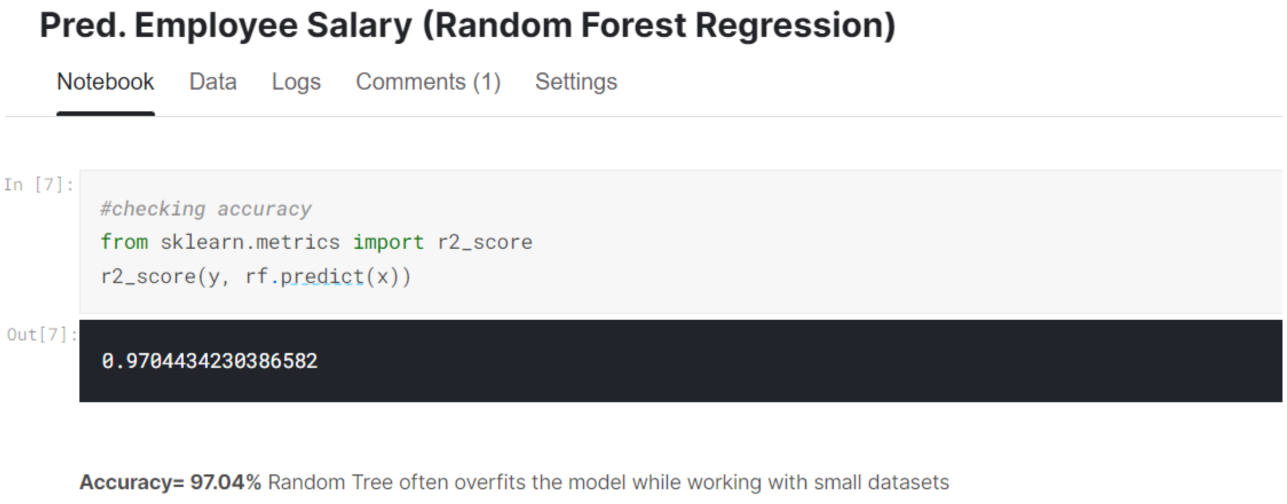
```
In [12]:
#checking the accuracy
from sklearn.metrics import r2_score
r2_score(y_test, lr.predict(x_test))
```

```
Out[12]:
0.9347068473282515
```

ACCURACY = 93.4%

CHAPTER-3
Working and Output

- Accuracy of random forest regression model = 97.8%



Pred. Employee Salary (Random Forest Regression)

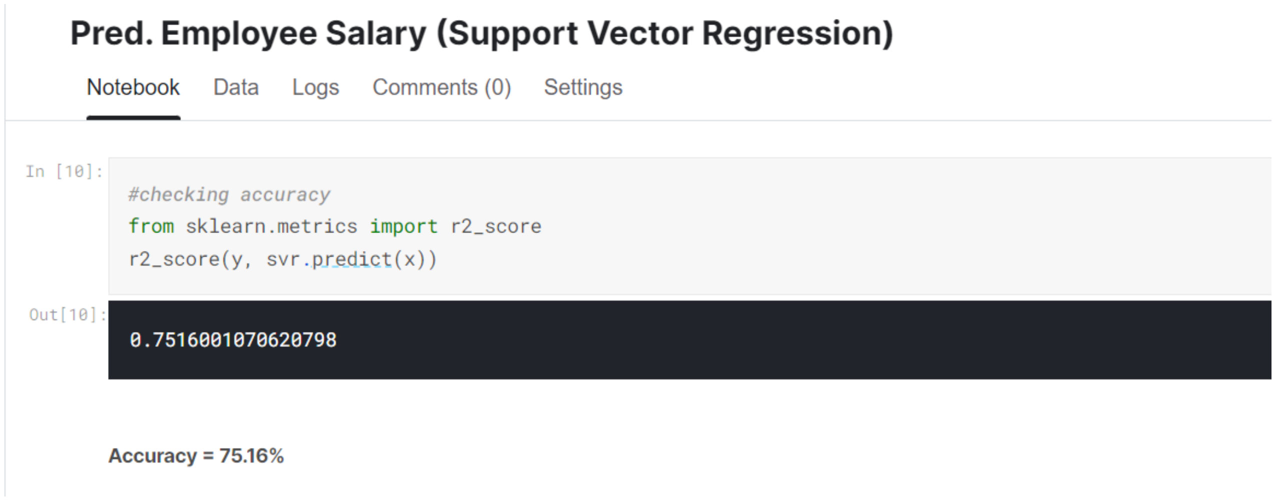
In [7]:
```
#checking accuracy
from sklearn.metrics import r2_score
r2_score(y, rf.predict(x))
```

Out[7]:
```
0.9704434230386582
```

Accuracy= 97.04% Random Tree often overfits the model while working with small datasets

CHAPTER-3
Working and Output

- Accuracy of support vector regression model = 75.16%



```
In [10]:
#checking accuracy
from sklearn.metrics import r2_score
r2_score(y, svr.predict(x))
```

```
Out[10]:
0.7516001070620798
```

Accuracy = 75.16%

CHAPTER-4
Conclusion and Future Scope

5.1 CONCLUSION AND FUTURE SCOPE:

In **Simple linear regression**, one is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other.
The key point in Simple Linear Regression is that the *dependent variable must be a continuous/real value*. However, the independent variable can be measured on continuous or categorical values.

Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y.

A **Random Forest Regression** model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, overfitting may easily occur, we choose the number of trees to include in the model.

The goal of the **Support Vector regression** algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

References

[1] Viola, P & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR, 2001), December 8-14, 2001, Kauai, HI, USA.

[2] Liao, S., Jain, A.K., Li, S. Z. (2016). A fast and accurate unconstrained face detector. IEEE Transaction of Pattern Analysis and Machine Intelligence, Vol 38, No 2.

[3] Luo, D., Wen, G., Li, D., Hu, Y., and Huna, E. (2018). Deep learning-based face detection using iterative bounding-box regression.

[4] Zhang, Y., Wang, X., and Qu, B. (2012). Three-frame difference algorithm research based on mathematical morphology. Proceedings of 2012 International Workshop on Information and Electronics Engineering (IWIEE), pp. 2705 – 2709.

[5] Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: PAMI-8, No: 6, pp. 679-698, November 1986.

[6] Li, J. and Ding, S. (2011). Research on improved Canny edge detection algorithm. Proceedings of the International Conference on Applied Informatics and Communication, pp. 102 – 108, Communications in Computer and Information Science (CCIS), Vol 228, Springer-Verlag.

[7] Lucas, B. D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision.

[8] Ren, Z., Yang, S., Zou, F., Yang, F., Luan, C., and Li, K. (2017). A face tracking framework based on convolutional neural networks and Kalman filter.

[9] Mingxing, J., Junqiang, D., Tao, C., Ning, Y., Yi, J., and Zhen, Z. (2013). An improved detection algorithm of face with combining AdaBoost and SVM. Proceedings of the 25th Chinese Control and Decision Conference, pp. 2459-2463.

[10] Altun, H., Sinekli, R., Tekbas, U., Karakaya, F. and Peker, M. (2011). An efficient color detection in RGB space using hierarchical neural network structure. Proceedings of 2011 International Symposium on Innovations in Intelligent Systems and Applications, pp. 154-158, Istanbul, Turkey