The Uplift Project

Polynomial Regression

Ans 1: **Source code**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Reading the dataset

dataset = pd.read\_csv('Position\_Salaries.csv')

dataset

# identifying and separating dependent and independent variables

# Here independent variable is level and dependent variable is salary

X = dataset.iloc[:,1:2].values  #Independent variable

y = dataset.iloc[:,2].values  #Dependent variable

# Linear regression model is used for comparing the result with polynomial regression

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X,y)

# Visualizing the result of linear regression

plt.scatter(X,y, color='red')

plt.plot(X, lin\_reg.predict(X),color='blue')

plt.title("Truth or Bluff(Linear)")

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

# Fitting the polynomial regression model

from sklearn.preprocessing import PolynomialFeatures

pr = PolynomialFeatures(degree=2) # Degree of the polynomial

X\_poly = pr.fit\_transform(X)

X

X\_poly

# The first column is the column of 1s for the constant.

#X containing real values is the middle column ie x1.

#The second column is square of x1.

# The fit must be included in a multiple linear regression model

lin\_reg2 = LinearRegression()

lin\_reg2.fit(X\_poly,y)

# Visualizing the polynomial regression model

from sklearn.preprocessing import PolynomialFeatures

pr = PolynomialFeatures(degree=4)

X\_poly = pr.fit\_transform(X)

lin\_reg2 = LinearRegression()

lin\_reg2.fit(X\_poly,y)

X\_grid = np.arange(min(X),max(X),0.1)

X\_grid = X\_grid.reshape(len(X\_grid),1)

plt.scatter(X,y, color='red')

plt.plot(X\_grid, lin\_reg2.predict(pr.fit\_transform(X\_grid)),color='blue')

plt.title("Truth or Bluff(Polynomial)")

plt.xlabel('Position level')

plt.ylabel('Salary')

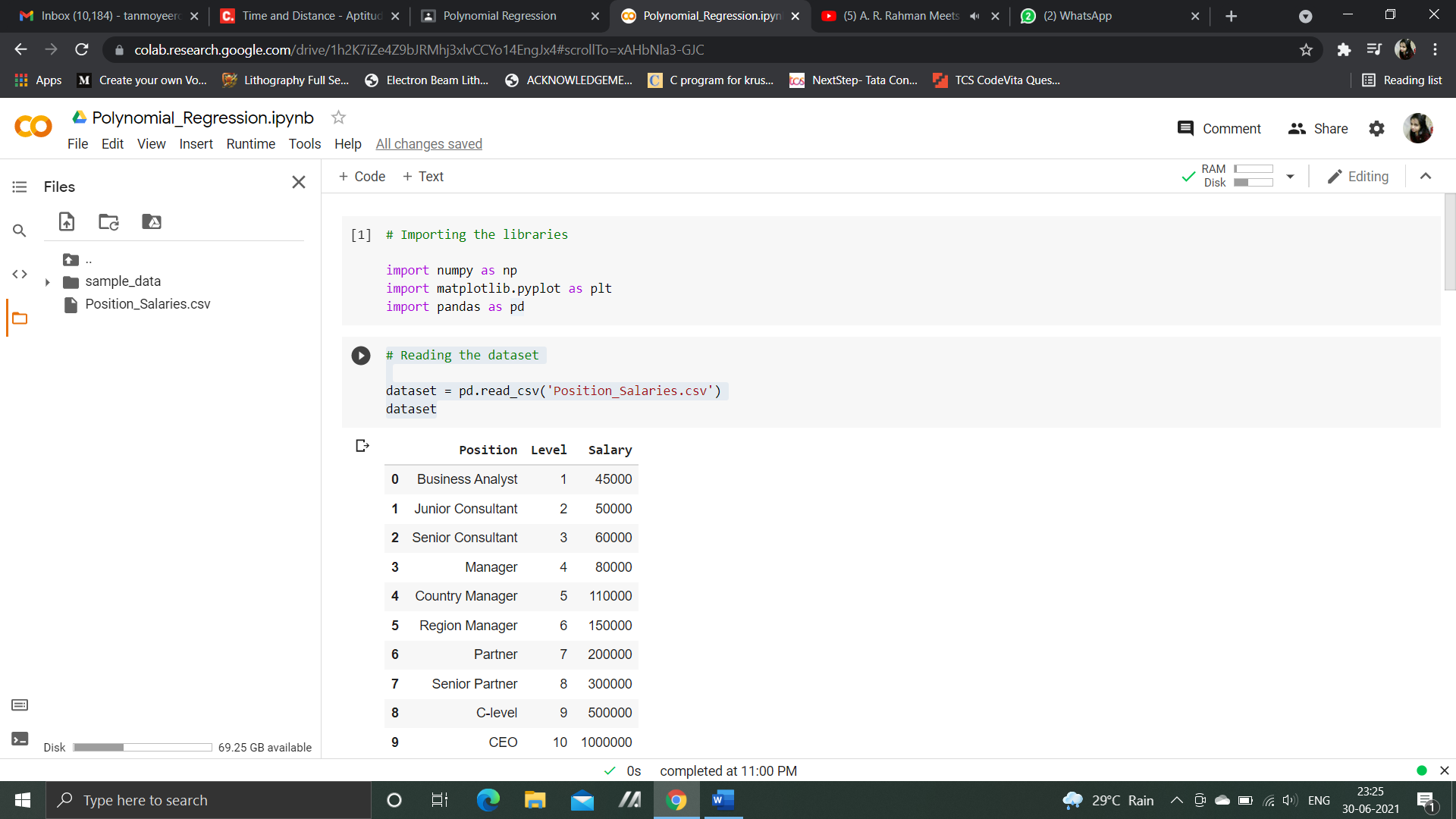
plt.show()

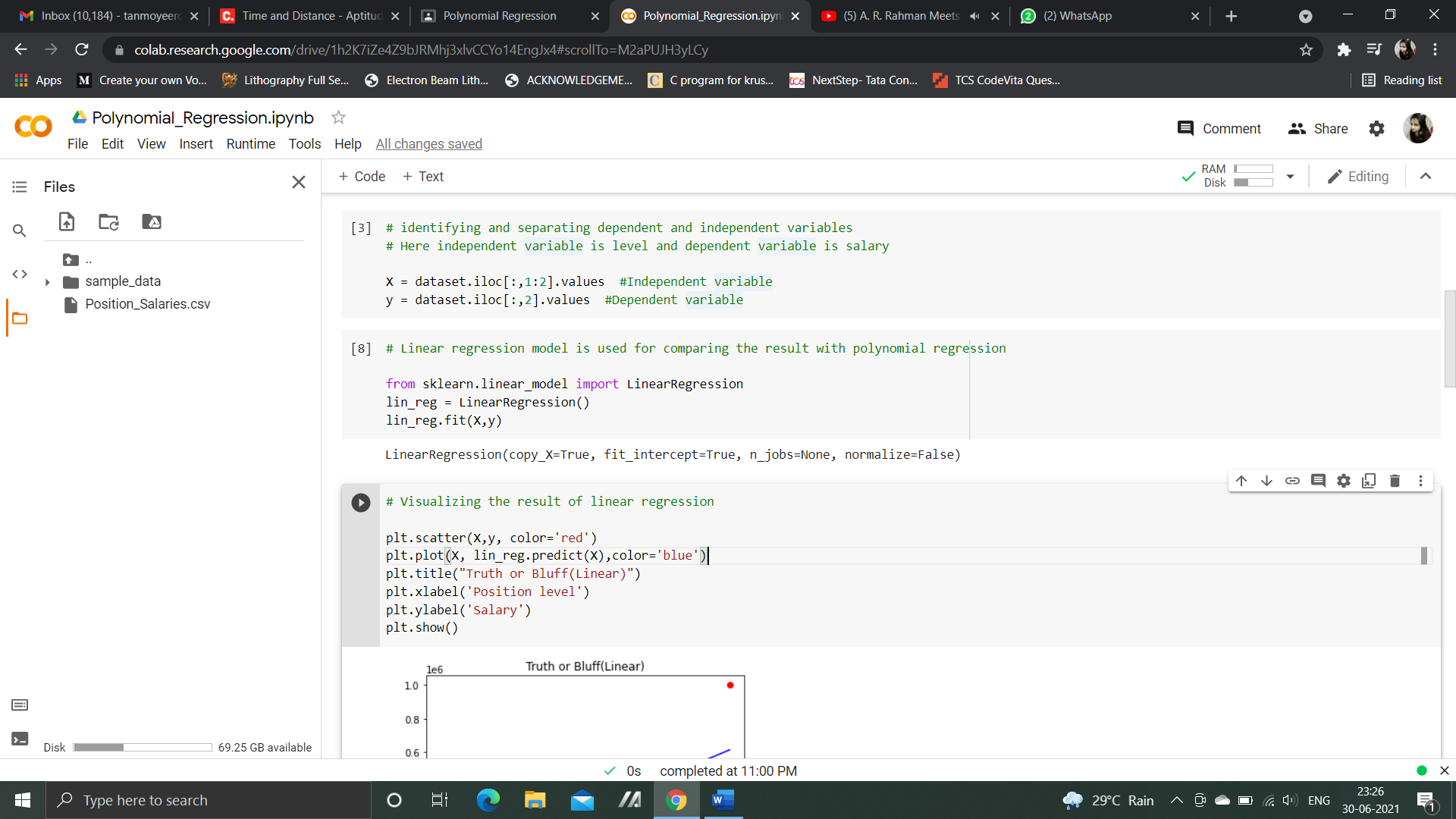
# Predicting the result

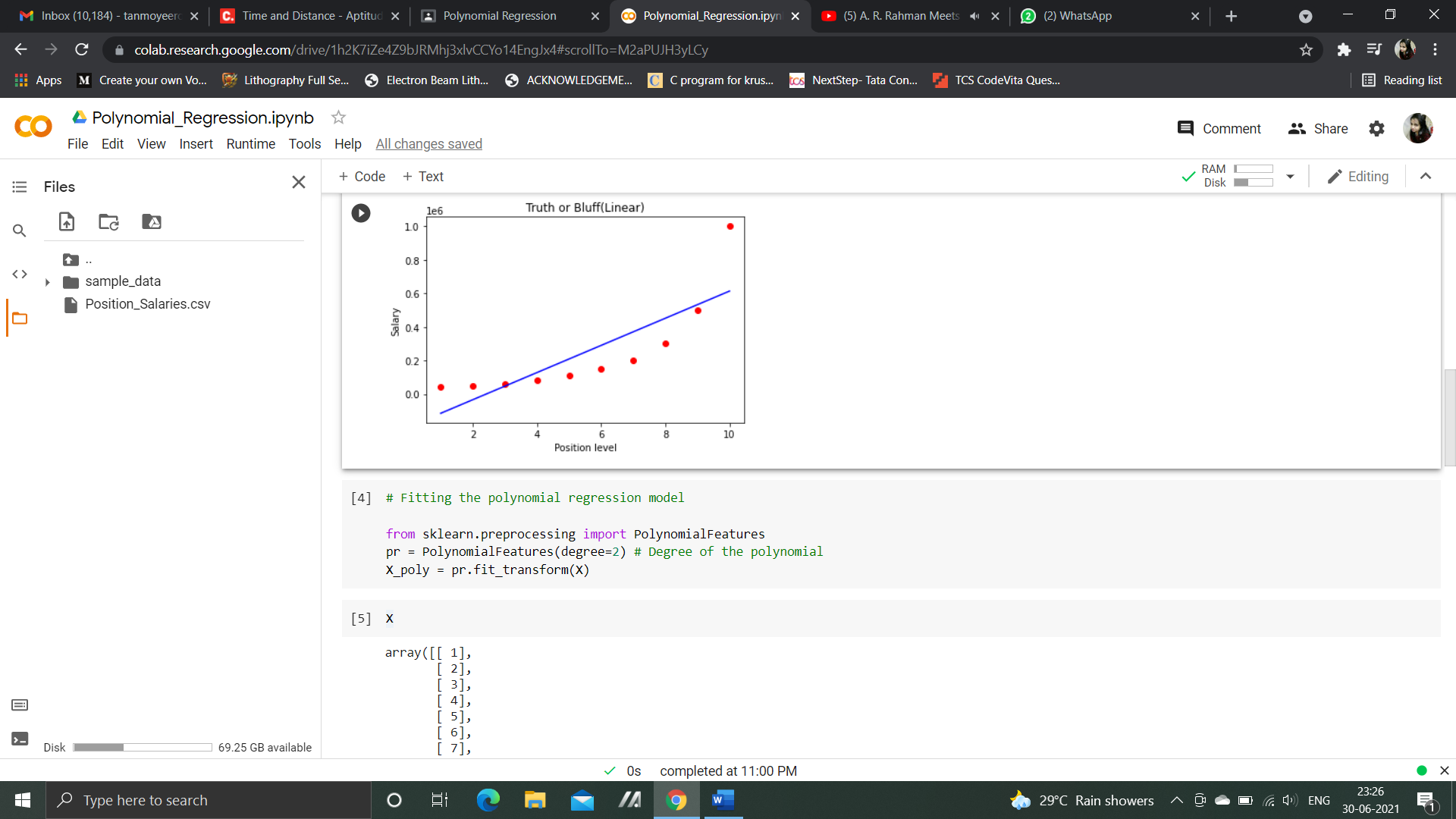
lin\_reg.predict([[6.5]])

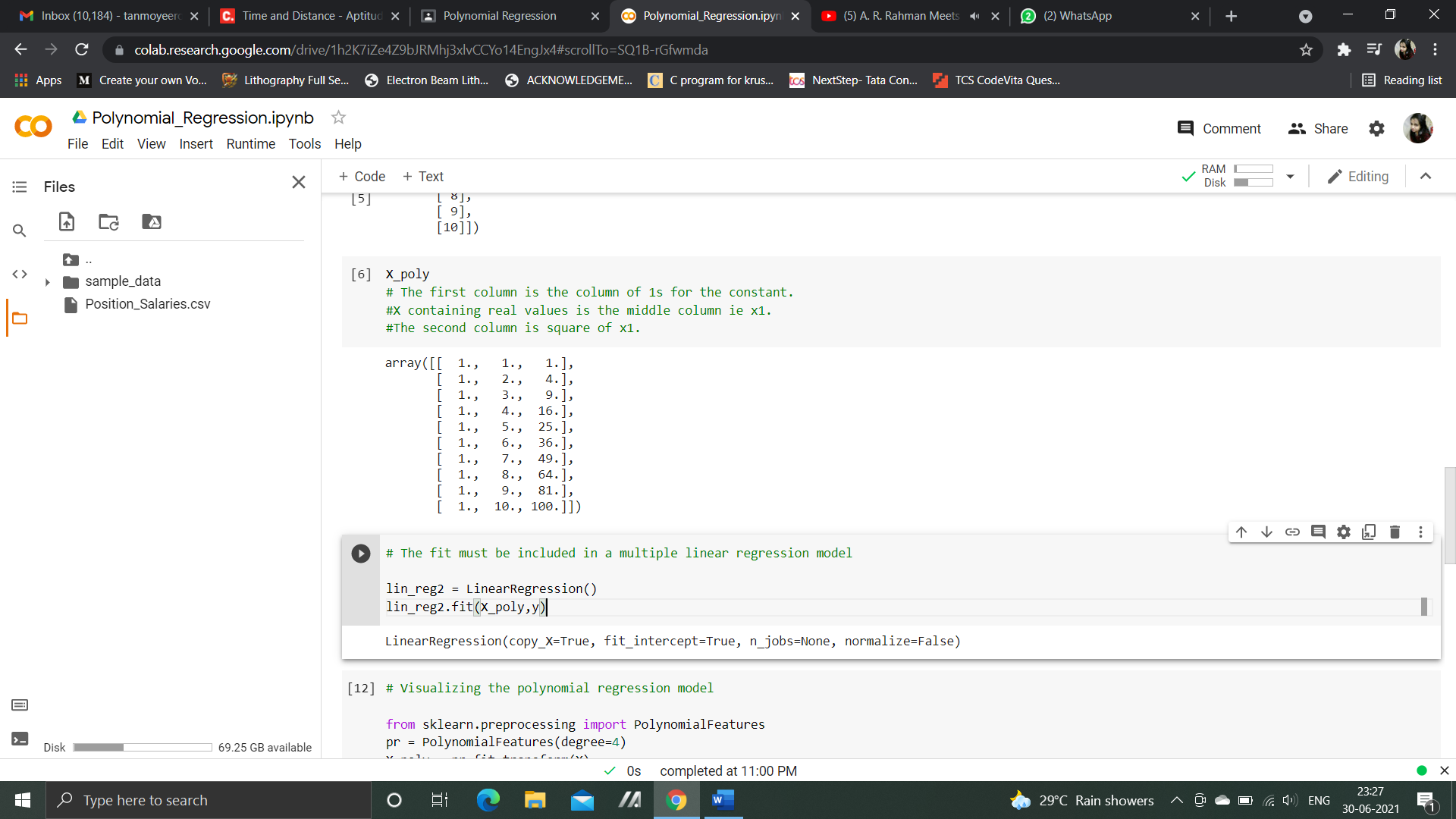
lin\_reg2.predict(pr.fit\_transform([[6.5]]))

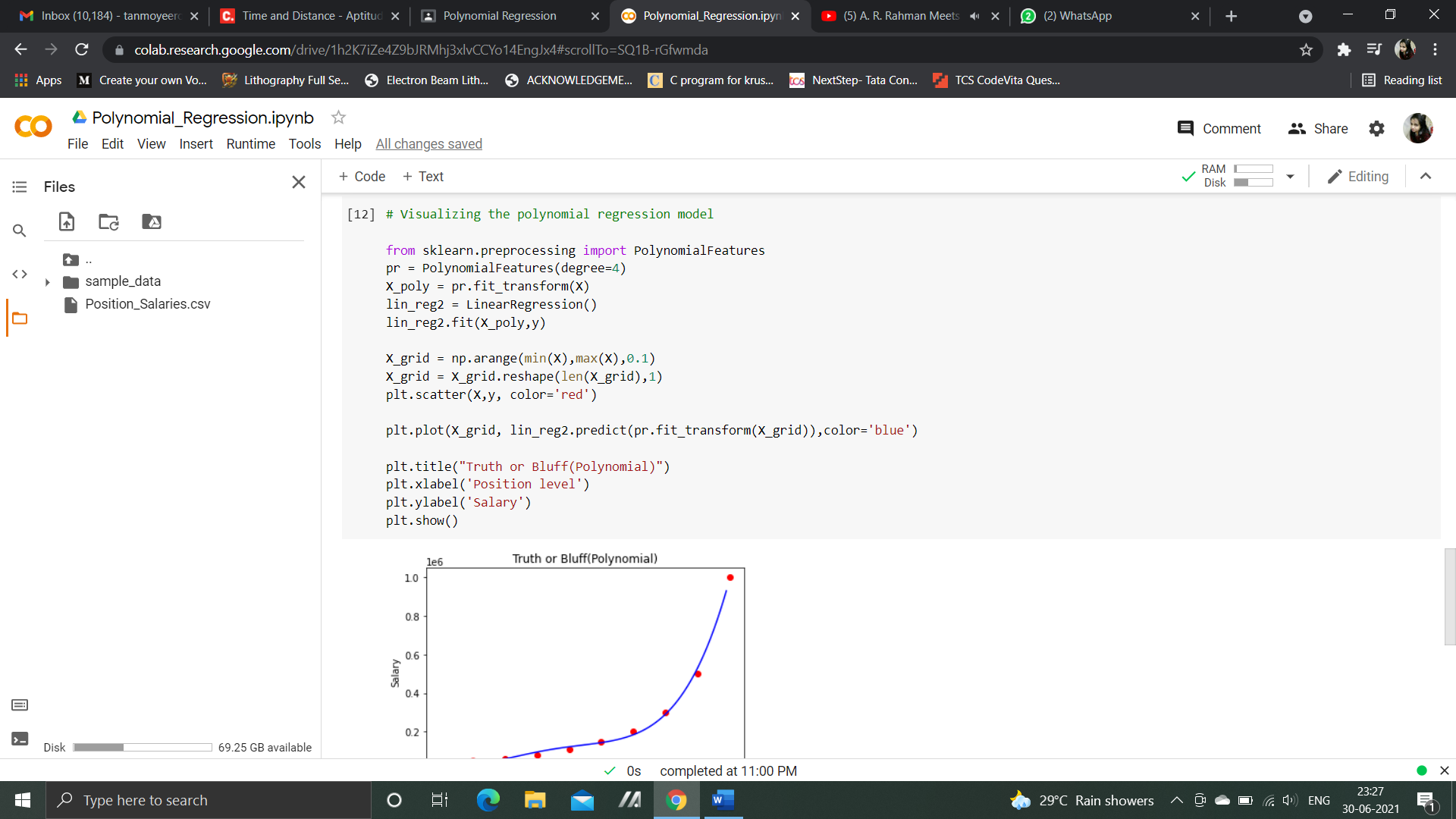
**Output:**

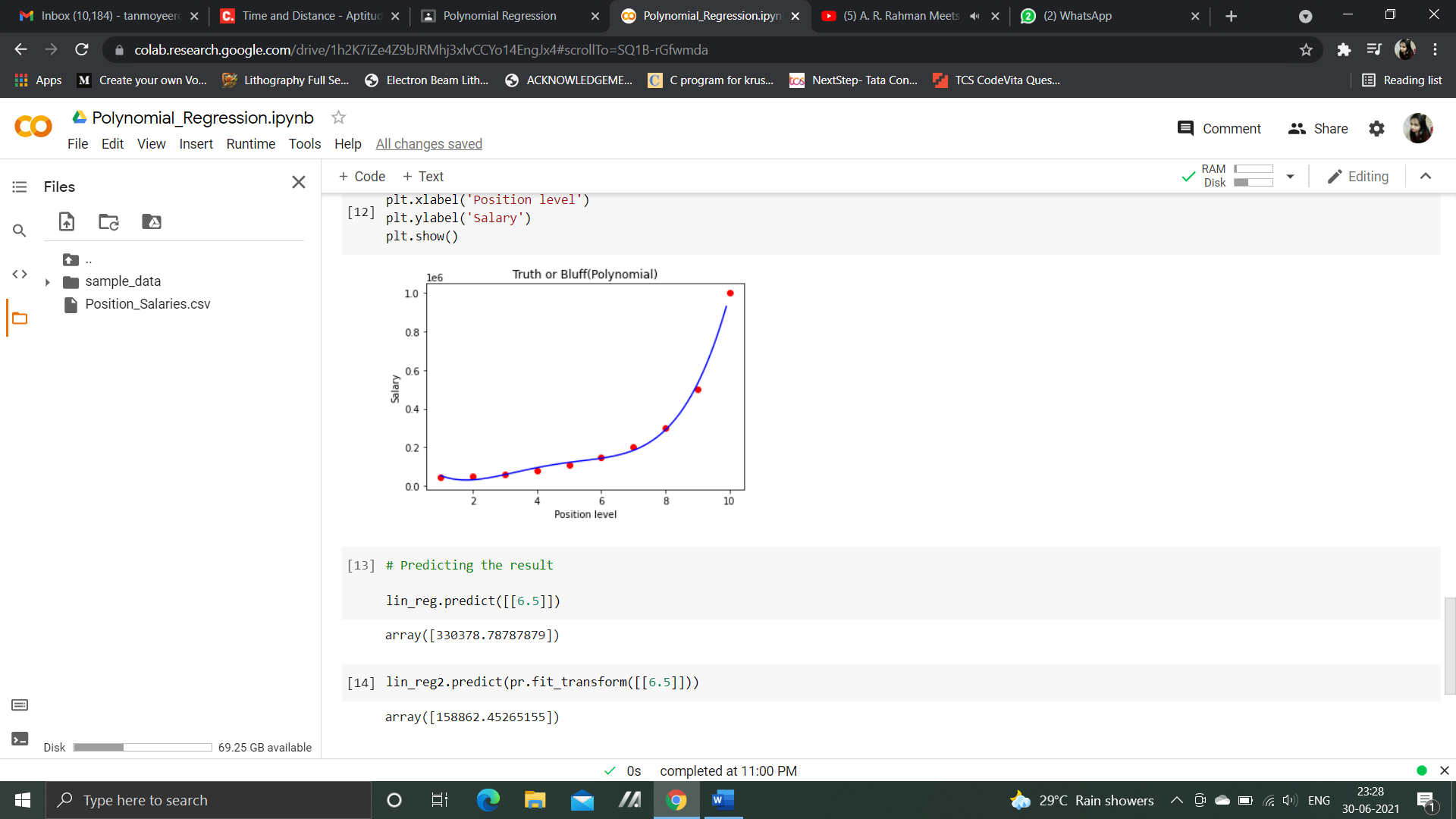












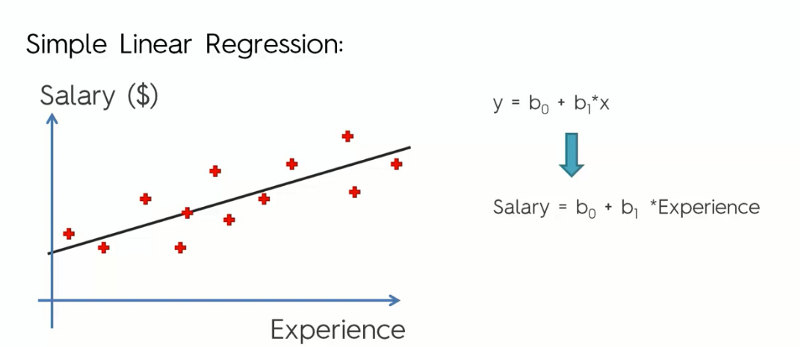
Ans 2:

1. Linear Regression

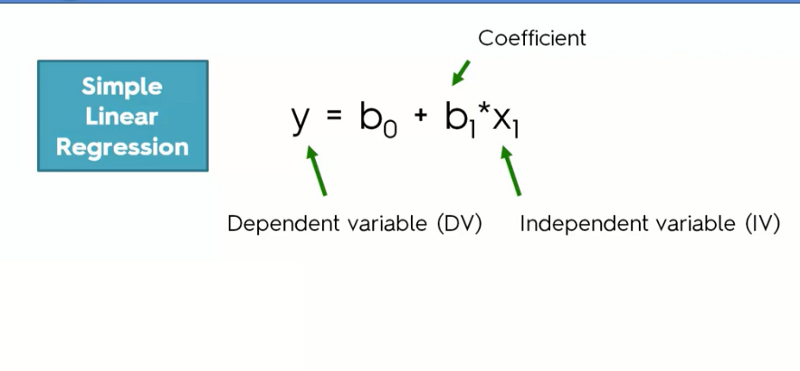
It is the simplest form of regression. It is a technique in which the dependent variable is continuous in nature. The relationship between the dependent variable and independent variables is assumed to be linear in nature. It follows the equation of the straight line i.e.

Y = mx + c

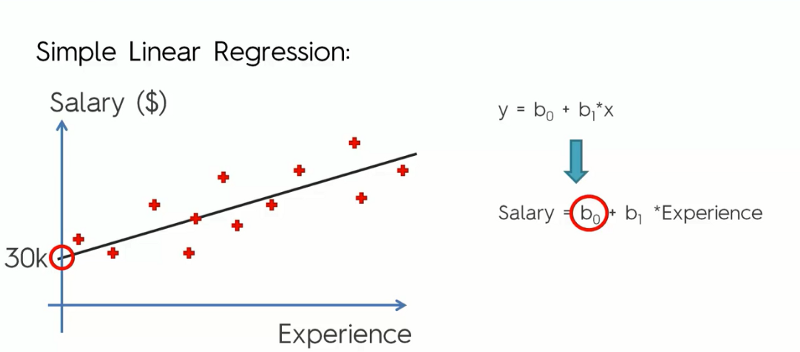
The output varies linearly based on the input. How much value of x has its impact on y is determined by m. In graph, a is the slope of the line and c is constant.

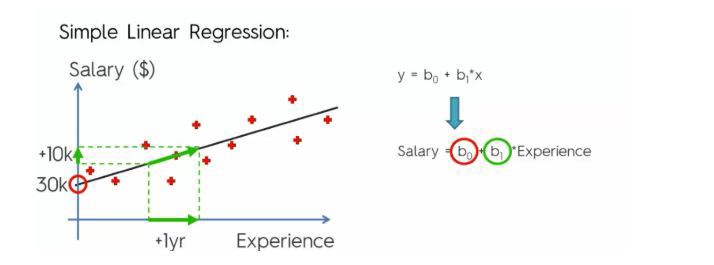


In the above graph it’s a simple case study of the Salary of various employees with their years of experience in particular work field.

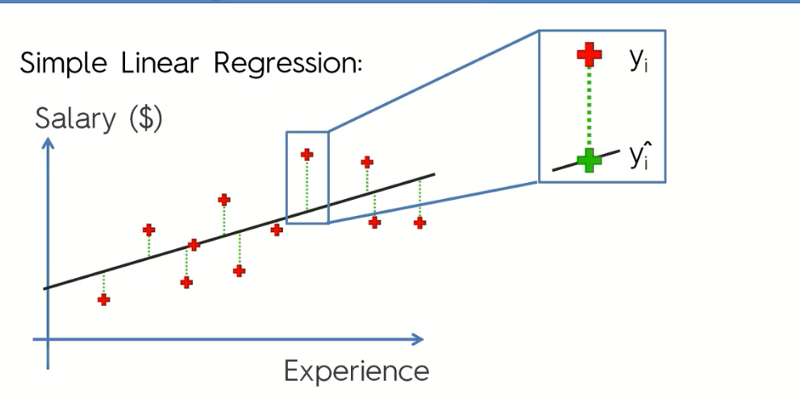


Here as we can see, **b0** is constant, which is fixed, and **b1**is the score to plot the graph or its merely a coefficient.





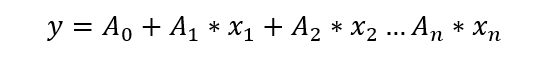
Finally, our objective is to minimize the distance between actual and observed values in the graph as we can see in the graph below if we reduce the cost more the chances of our better prediction.



Here **Yi is the original value, while y^ is the observed value.**So here is the equation used to minimize the distance between Actual and Observed values.

1. Multilinear Regression

**Multiple linear regression** is used to estimate the relationship between **two or more independent variables**and**one dependent variable. We can use multilinear regression when we want to know about** how strong the relationship is between two or more independent variables and one dependent variable (e.g. how rainfall, temperature, and amount of fertilizer added affect crop growth) or The value of the dependent variable at a certain value of the independent variables (e.g. the expected yield of a crop at certain levels of rainfall, temperature, and fertilizer addition). The equation for MLR is:



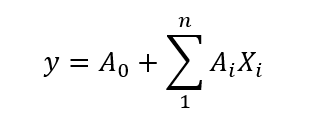
Here,

Y – dependent variable

Xi – independent variable

Ai – coefficient for feature

So, this model predicts what value of y will be depending on features Xi and with coefficient Ai deciding how much each feature is affecting the predicted value.



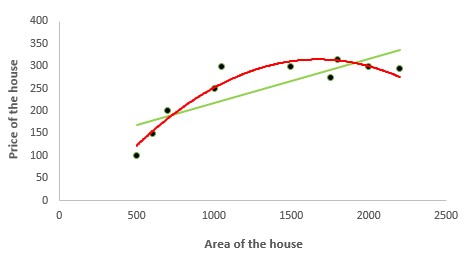
In translation, predicted value Y is sum of all features multiplied with

their coefficients, summed with base coefficient A0.

The graph for multilinear regression cannot be plotted since it has multiple dimensions and is not 2D.

1. Polynomial Regression

It is a technique to fit a nonlinear equation by taking polynomial functions of independent variable.  
In the figure given below, you can see the red curve fits the data better than the green curve. Hence in the situations where the relation between the dependent and independent variable seems to be non-linear, we can deploy Polynomial Regression Models.



Thus, a polynomial of degree k in one variable is written as:

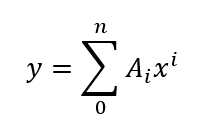


Here we can create new features like



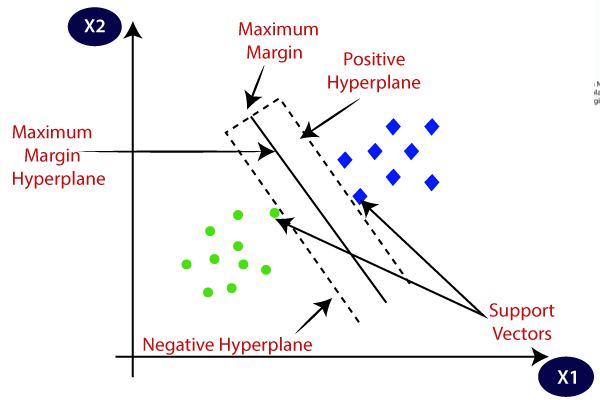
and can fit linear regression in the similar manner.

It can be better shown with:



1. Support Vector Regression

Support vector regression can solve both linear and non-linear models. SVM uses non-linear kernel functions (such as polynomial) to find the optimal solution for non-linear models.

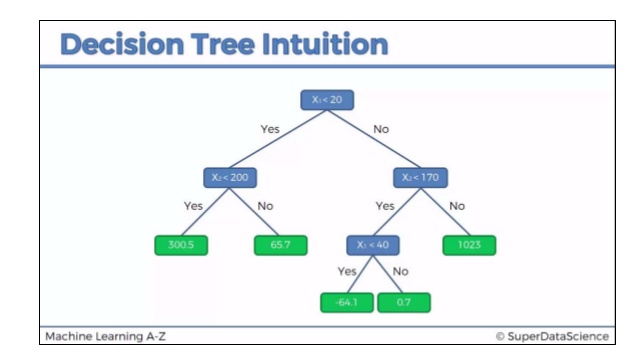


The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration.

However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

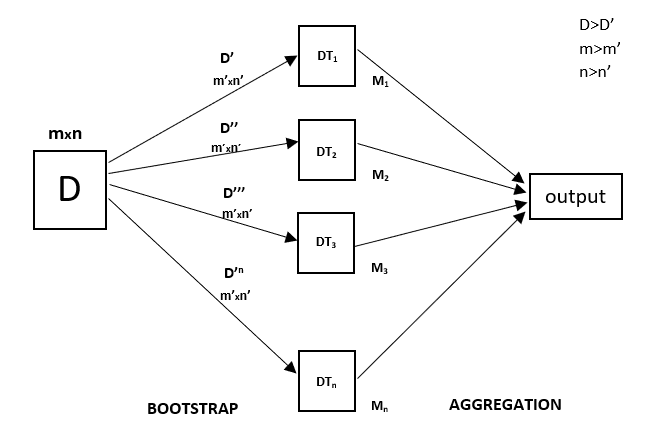
1. Decision Tree Regression

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.



1. Random Forest Regression

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.



Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

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. This part is called Bootstrap.

