SALARY PREDICTION USING PYTHON

Project

Abstract

We will use linear regression to model the relationship between the amount of salary with the years of working experience

AYAAN SHAIKH

Contents

Regression Model

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 changes as the independent variable(s) change.

Simple Linear Regression

Simple linear regression is used to estimate the relationship between two quantitative variables. You can use simple linear regression when you want to know:

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

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).

Example: You are a social researcher interested in the relationship between income and happiness. You survey 500 people whose incomes range from 15k to 75k and ask them to rank their happiness on a scale from 1 to 10.

Your independent variable (income) and dependent variable (happiness) are both quantitative, so you can do a regression analysis to see if there is a linear relationship between them.

If you have more than one independent variable, use **multiple linear regression** instead.

Assumptions of simple linear regression

Simple linear regression is a parametric test, meaning that it makes certain assumptions about the data. These assumptions are:

Homogeneity of variance (homoscedasticity): the size of the error in our prediction doesn't change significantly across the values of the independent variable.

Independence of observations: The observations in the dataset were collected using statistically valid sampling methods, and there are no hidden relationships among observations.

Normality: The data follows a normal distribution.

Linear regression makes one additional assumption:

The relationship between the independent and dependent variable is linear: the line of best fit through the data points is a straight line (rather than a curve or some sort of grouping factor).

If your data do not meet the assumptions of homoscedasticity or normality, you may be able to use a nonparametric test instead, such as the Spearman rank test.

Example: Data that doesn't meet the assumptions you think there is a linear relationship between cured meat consumption and the incidence of colorectal cancer in the U.S. However, you find that much more data has been collected at high rates of meat consumption than at low rates of meat consumption, with the result that there is much more variation in the estimate of cancer rates at the low range than at the high range. Because the data violate the assumption of homoscedasticity, it doesn't work for regression, but you perform a Spearman rank test instead.

If your data violate the assumption of independence of observations (e.g. if observations are repeated over time), you may be able to perform a linear mixed-effects model that accounts for the additional structure in the data.

How to perform a simple linear regression

Simple linear regression formula

The formula for a simple linear regression is:

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

y is the predicted value of the dependent variable (**y**) for any given value of the independent variable (**x**).

B_0 is the **intercept**, the predicted value of **y** when the **x** is 0.

B_1 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 line through your data by searching for the regression coefficient (B_1) that minimizes the total error (e) of the model.

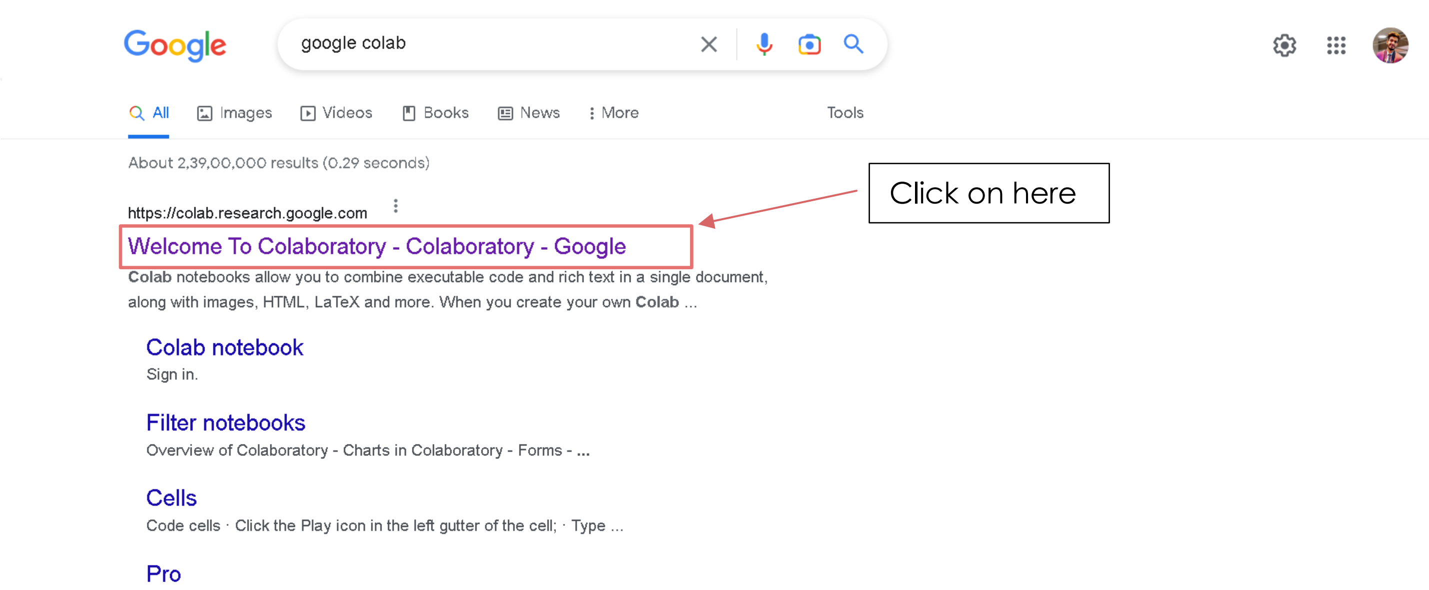
While you can perform a linear regression by hand, this is a tedious process, so most people use statistical programs to help them quickly analyze the data.

Using Google Colab for Simple Linear Regression

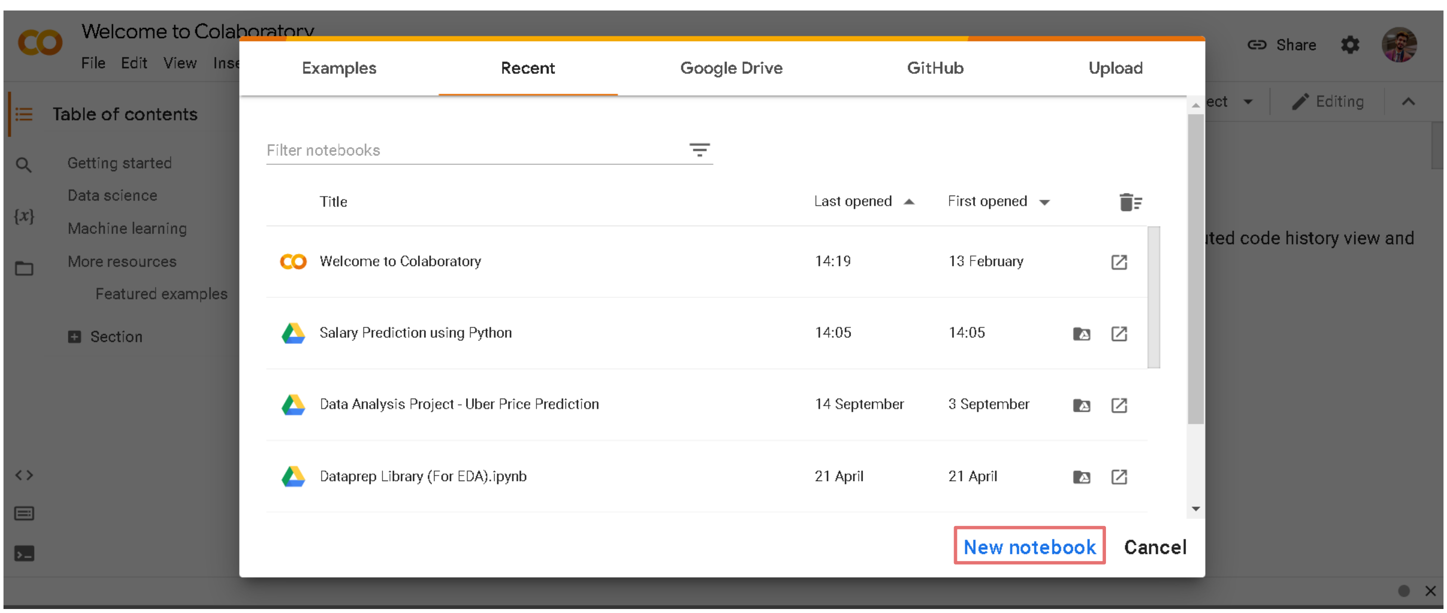
With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses numpy to generate some random data, and uses matplotlib to visualise it. To edit the code, just click the cell and start editing. We will be using Google Colab for running simple linear regression model for predicting Salary by experience.

Steps to perform Linear Regression in Google Colabs.

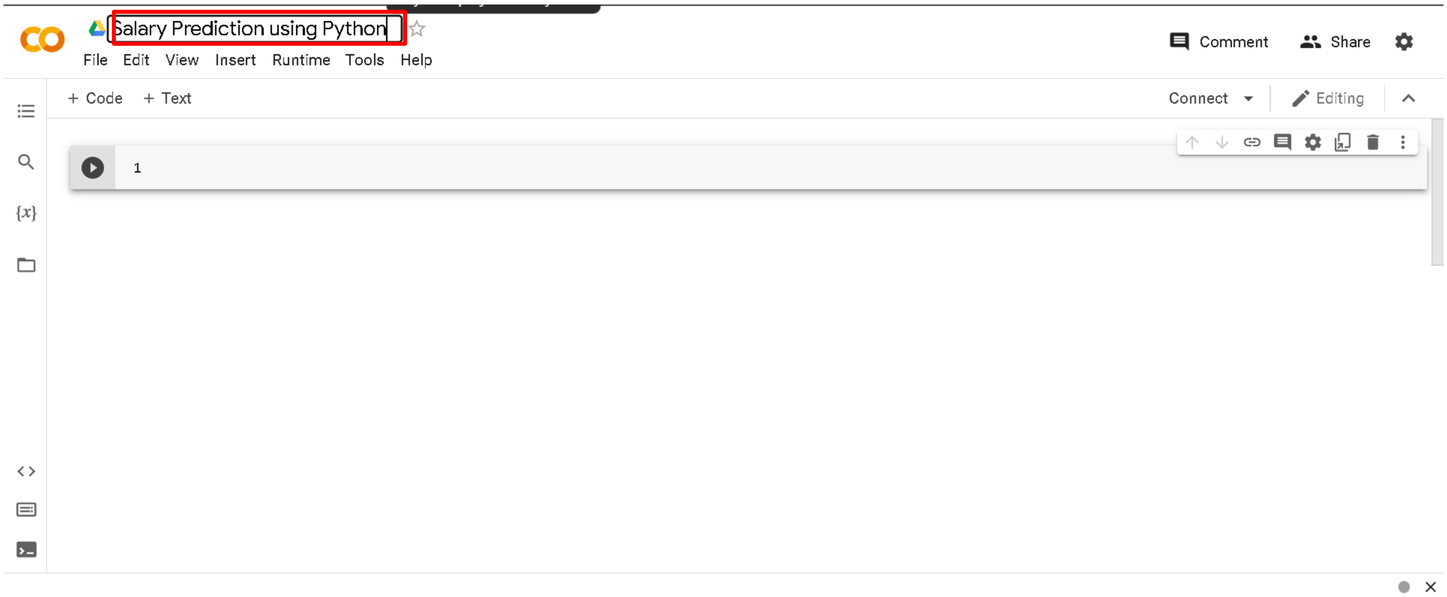
Step 1: Open Google Chrome then search for google colabs



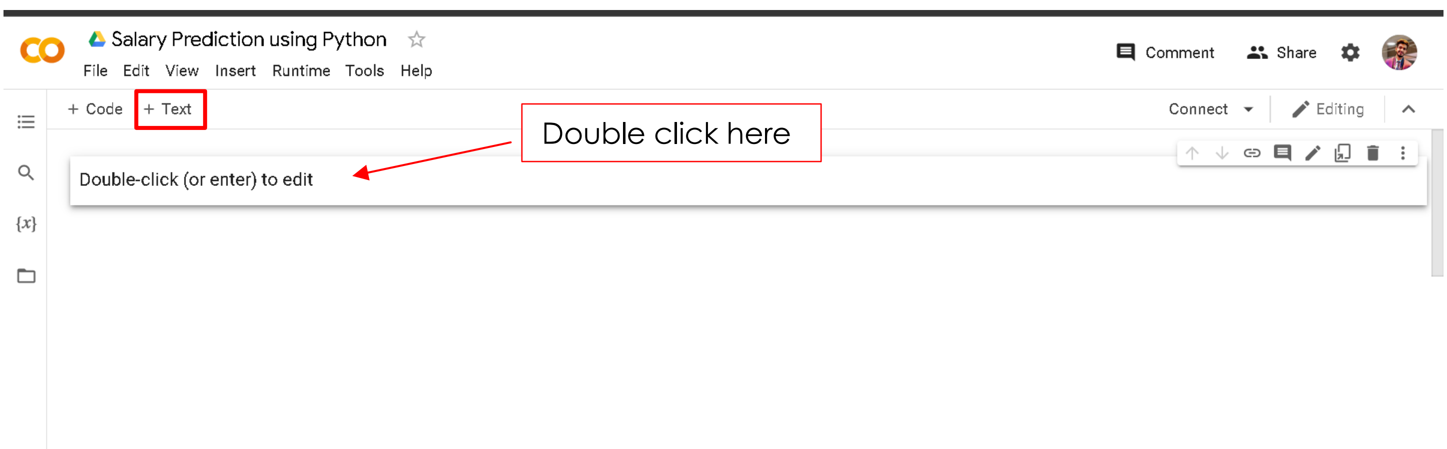
Step 2: Click on **New notebook**



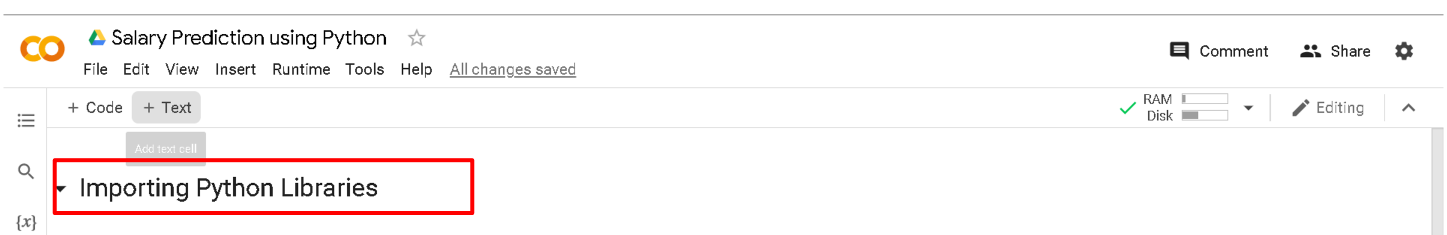
Step 3: Double click to edit the title of the notebook as shown below



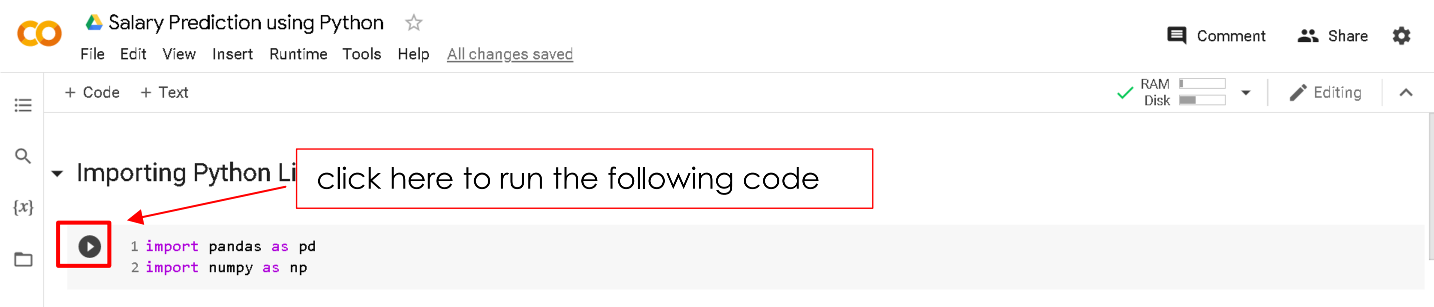
Step 4: Click on **"+ Text"** then double click below to write a title of the cell as shown below.



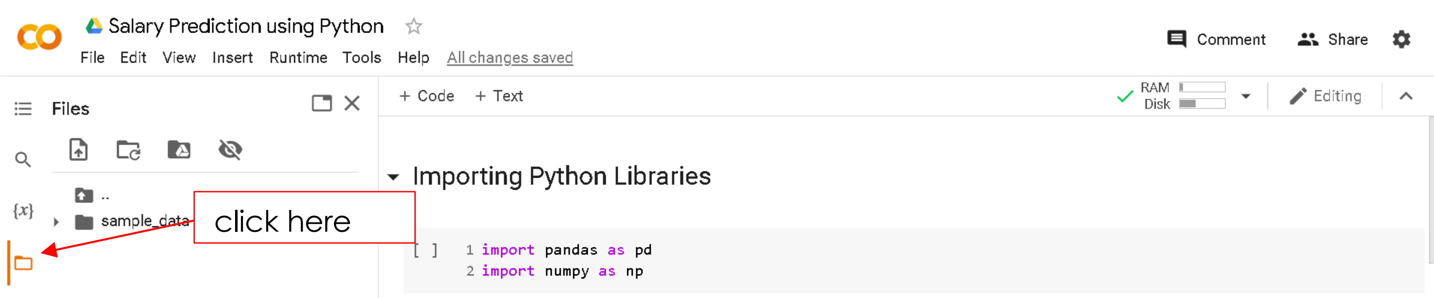
Step 5: We will first import some important python libraries so we will give the title as "**Importing Python Libraries**" as shown below



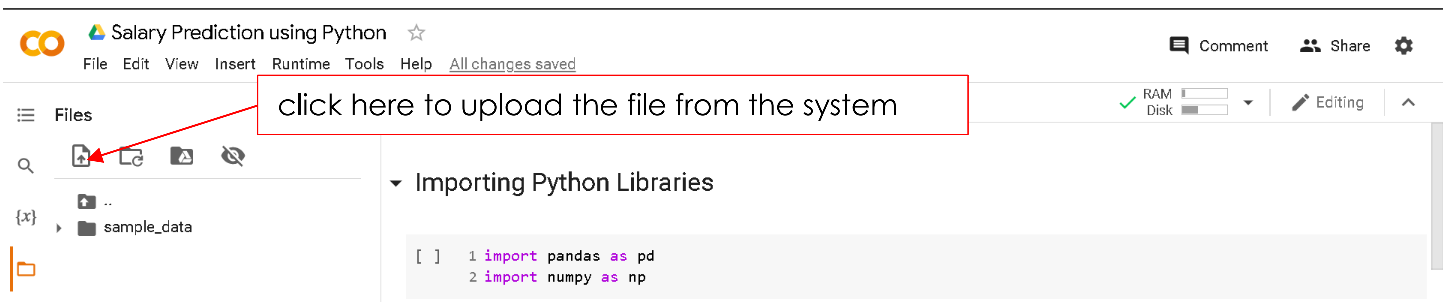
Step 6: Perform the below code to import pandas and numpy and click on Run as shown below



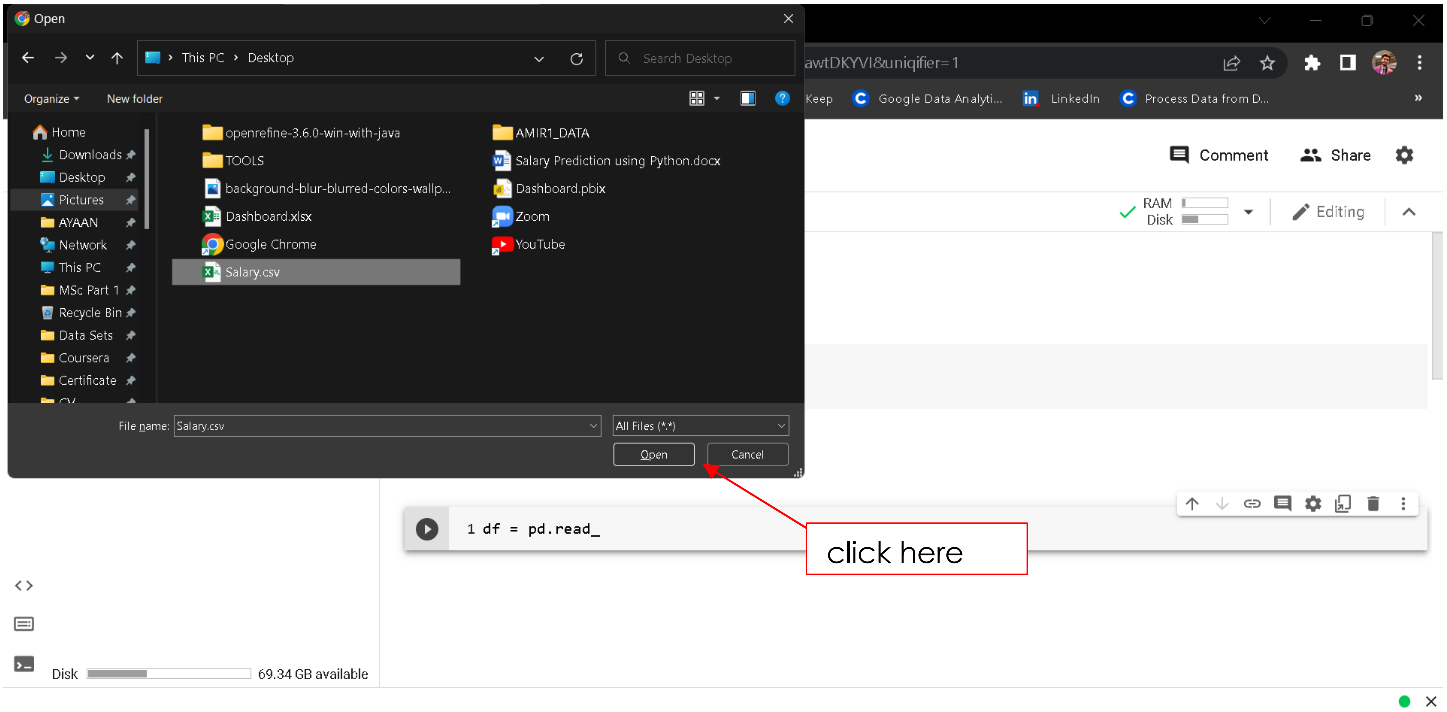
Step 7: Upload the Salary.csv file in to the google colab



Step 8: After step 7 click as shown in below screenshot



Step 9: Find the file and click on open



Note: Your file remain for 12 hours after upload in the google colabs.

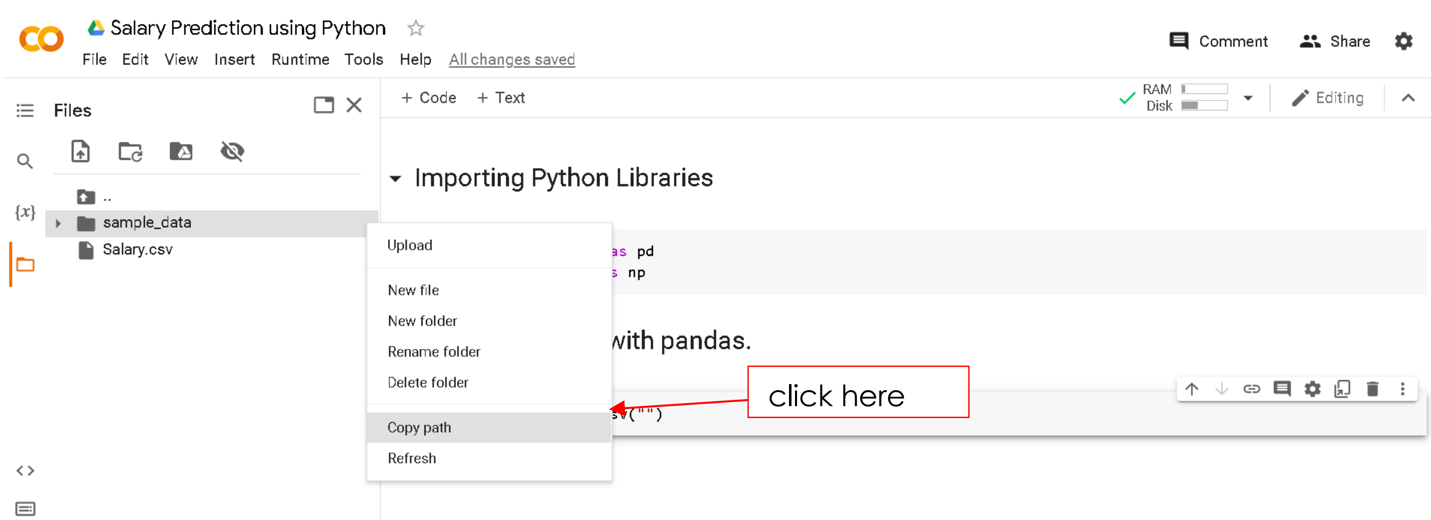
Step 10: Follow step 4 to open title box as now we will be importing **Salary.csv** file and perform the linear regression into it.

What is PD read_csv?

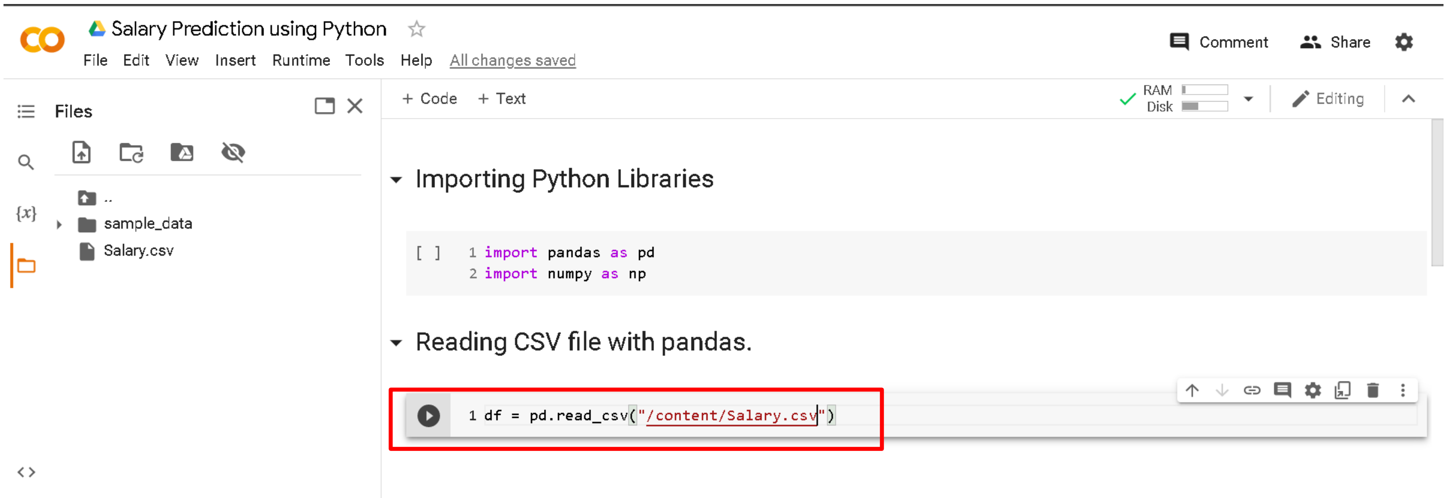
Read a comma-separated values (csv) file into DataFrame. Also supports optionally iterating or breaking of the file into chunks.

Step 11: Copy the file path as shown in below screenshot.

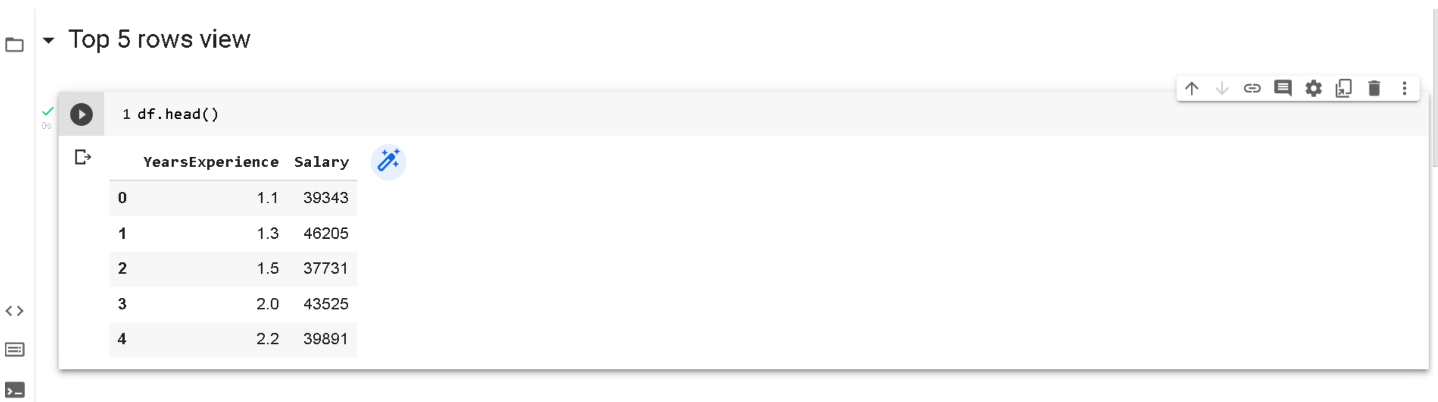
Step 12: Click copy path



Step 13: Paste that path in between the inverted commas



Step 14: To see top five rows of the dataframe we will use **.head()** function



Step 15: To see all columns in the salary.csv dataset use **.columns**

Salary_Prediction_using_Python

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

To view all columns in the dataset

```
1 df.columns
```

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

Step 16: To get a quick insight from the dataset we will use **.describe()** function

Salary_Prediction_using_Python

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

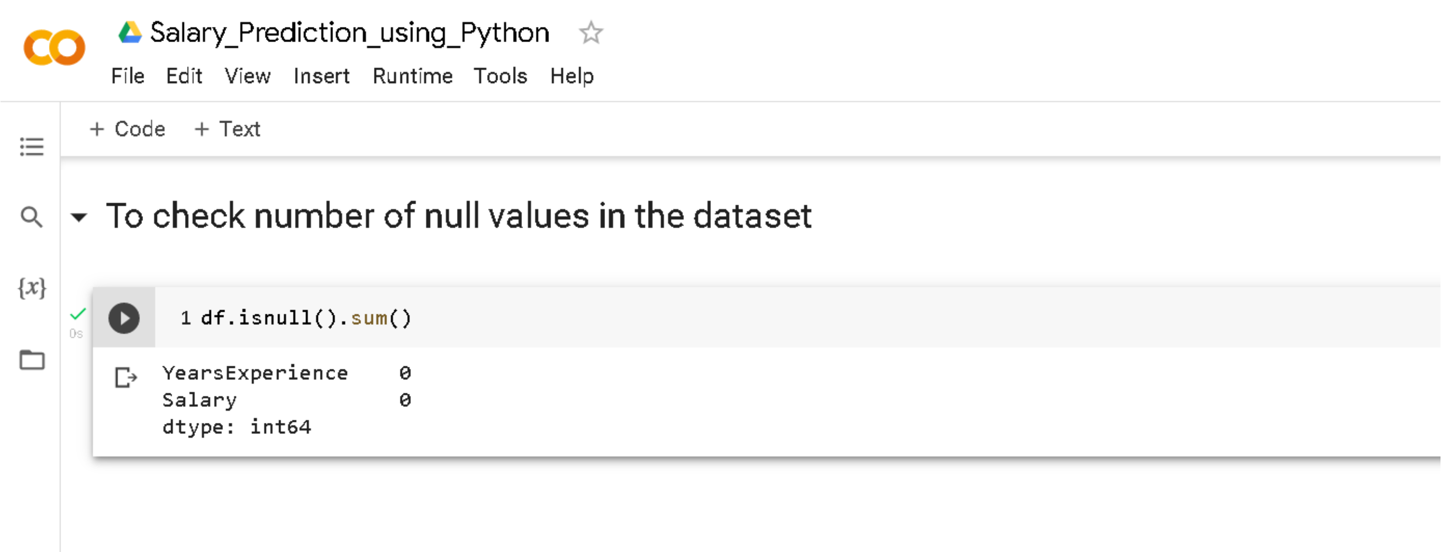
To get the quick insights from the dataset.

```
1 df.describe()
```

	YearsExperience	Salary
count	35.000000	35.000000
mean	6.308571	83945.600000
std	3.618610	32162.673003
min	1.100000	37731.000000
25%	3.450000	57019.000000
50%	5.300000	81363.000000
75%	9.250000	113223.500000
max	13.500000	139465.000000

Step 17: To check null values in the dataset we will use **.isnull()** function if it gives the value false that means there are no null values if it is true then there is a null value.

Step 18: To check number of null values in the dataset we will use .sum() function to sum up the null values.



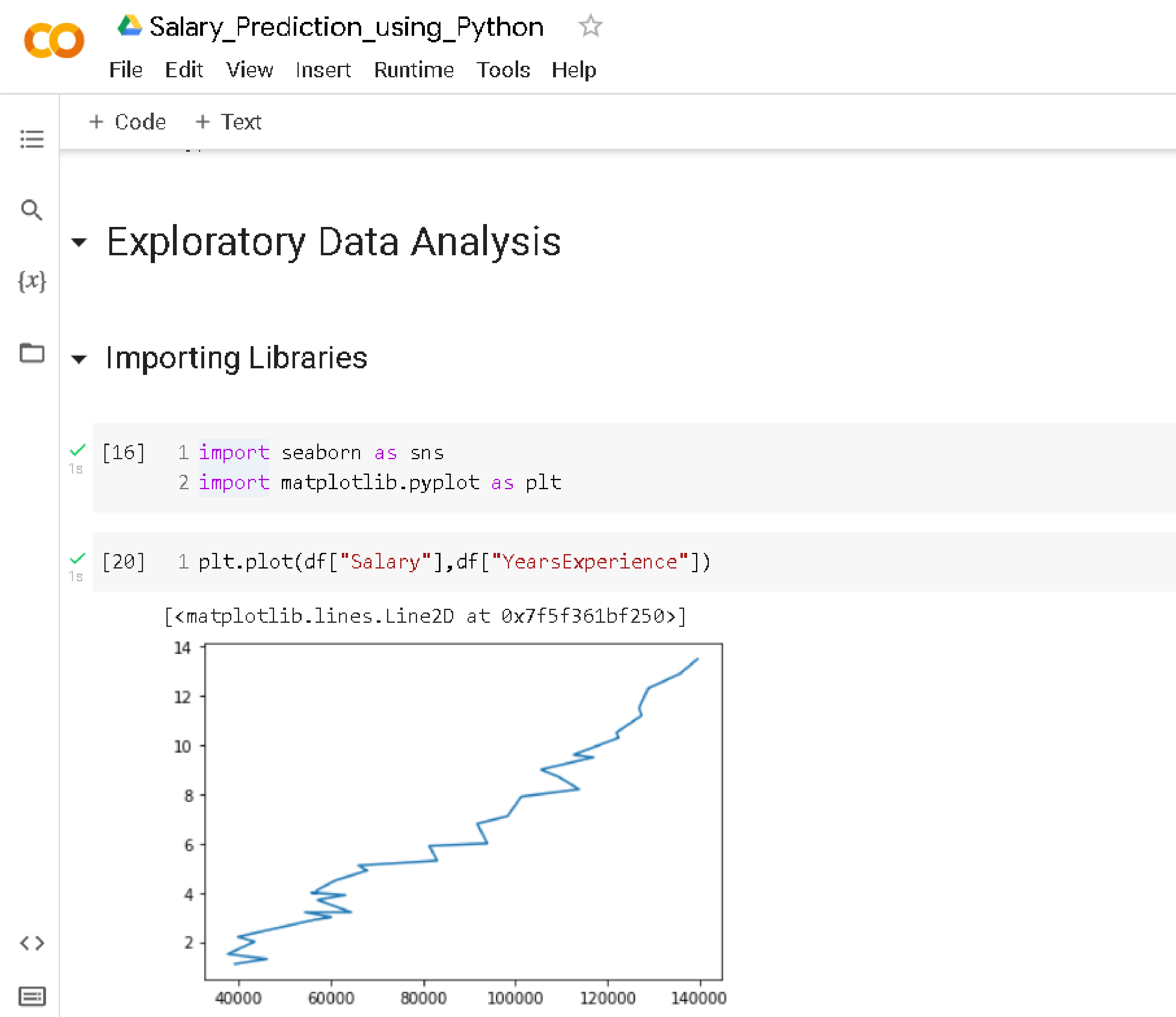
Note: Here there are no null values in the dataset

Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

We will import some libraries called seaborn and matplotlip which will help to perform exploratory data analysis (EDA) on the dataset.

Step 1: Import following libraries as shown bellowed in the screenshot.



Now we will import Scikit learn library which is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines

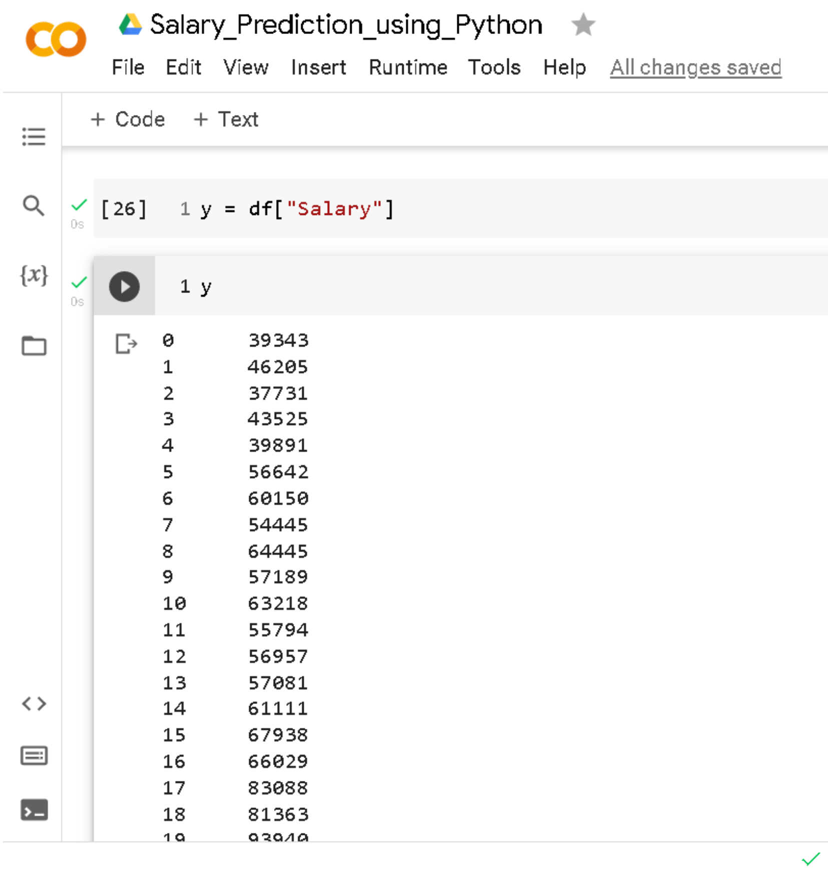
Performing Simple Linear Regression

Overview of dataset

Here **dependent variable is salary(Y) and independent variable is YearsExperience(X)**. If there is an increase in dependent variable when independent variable increases, then there is a positive correlation among them. If decreases, then there is a negative correlation among them.

Step 1: We will put YearsExperience in X and Salary in Y variable to split the data as shown below in the screenshot

Same we will do this for Salary column we will put Salary column in Y variable



Now we have split the dependent variable(Y) which is Salary column and independent variable(X) which is YearsExperience.

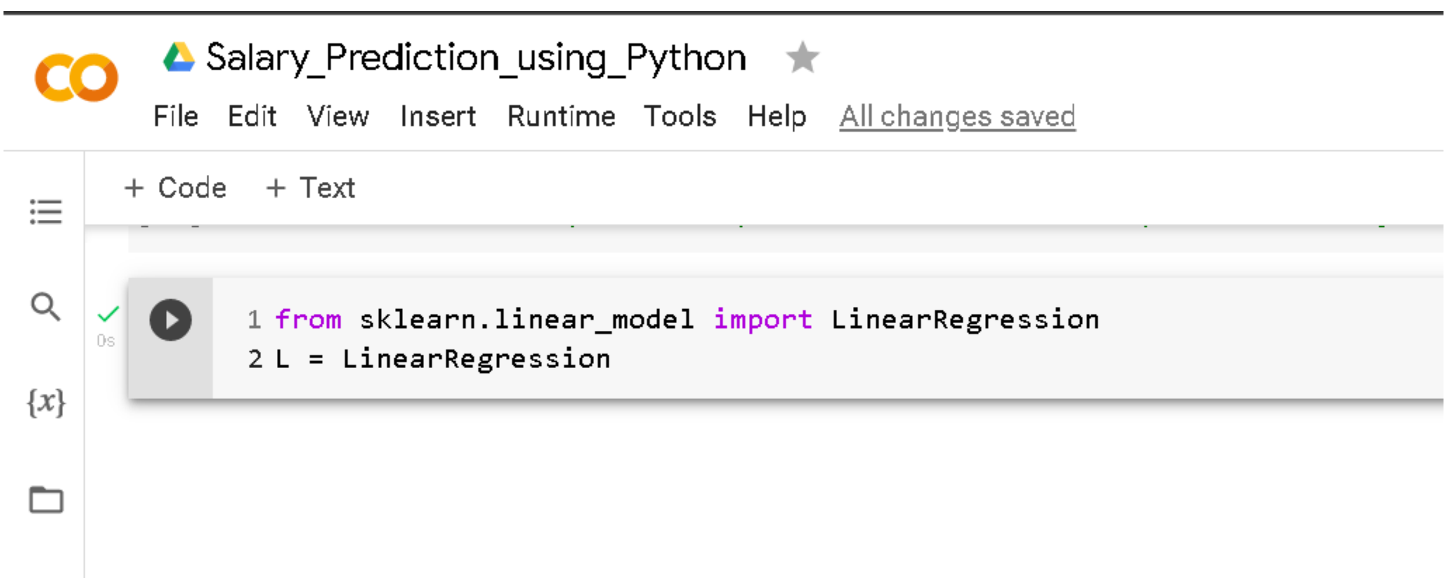
What is Train_test_split?

The train_test_split() method is **used to split our data into train and test sets**. First, we need to divide our data into features (X) and labels (y). The dataframe gets divided into X_train, X_test, y_train, and y_test. X_train and y_train sets are used for training and fitting the model.

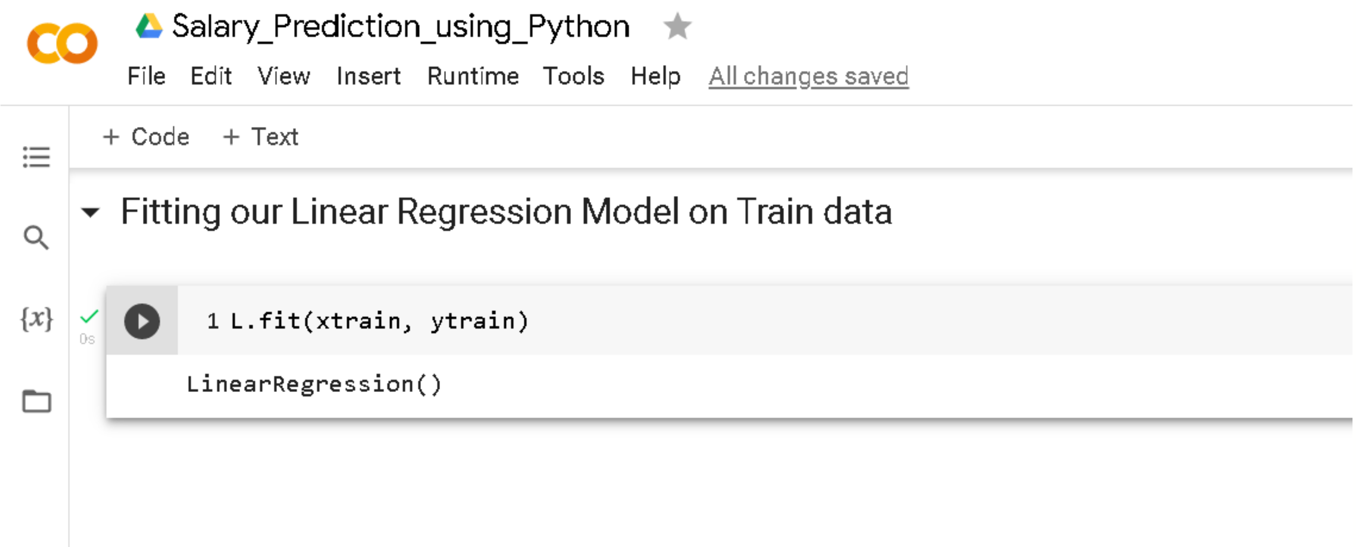
Training the model

Step 1 : Here we are dividing the dataset into xtrain, xtest, ytrain, and ytest we weill give 75% of data for training and rest 25% will be used on testing we have mention 0.75 which means 75% and random_state = 42 means that it will pick up the training selection of the data randomly. The important thing is that everytime you use 42, you will always get the same output the first time you make the split. This is useful if you want reproducible results, for example in the documentation, so that everybody can consistently see the same numbers when they run the examples.

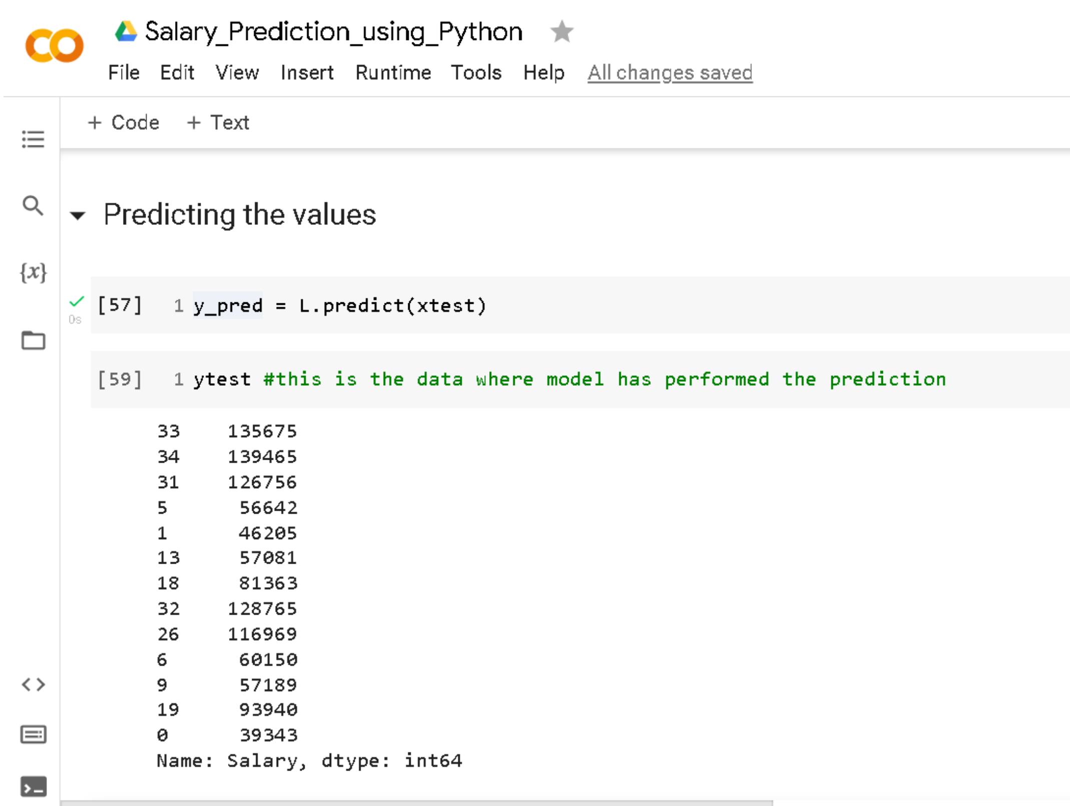
Step 2: We will call LinearRegression module from sklearn library and we will put it in to a variable name “**L**”

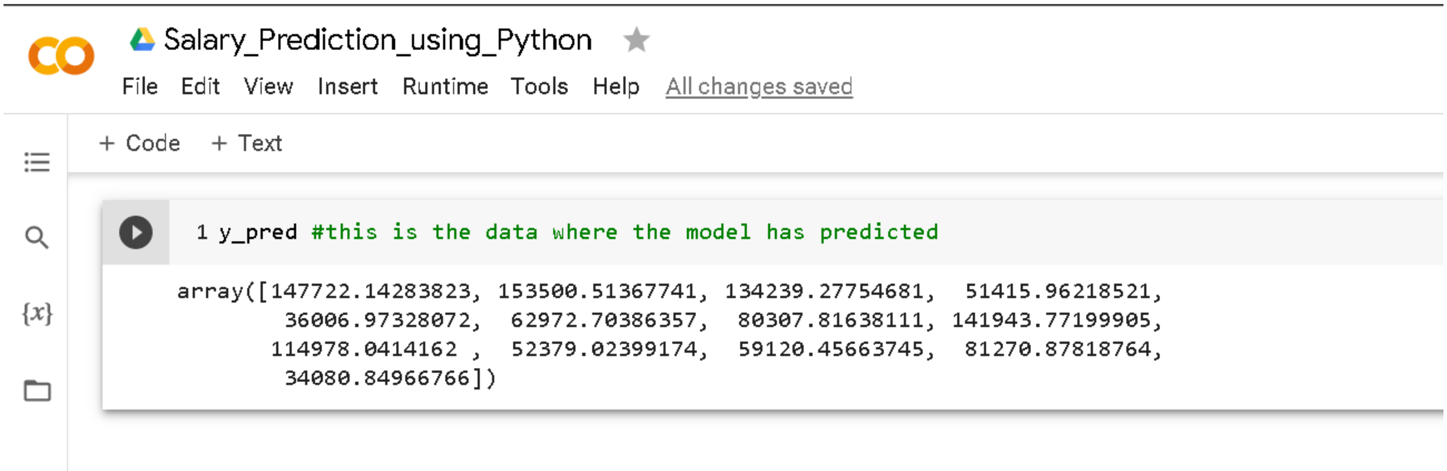


Step 3: Fitting our Linear Regression Model on Train data the model will train itself on 75% of dataset in the data. And the rest will be used for testing the model.



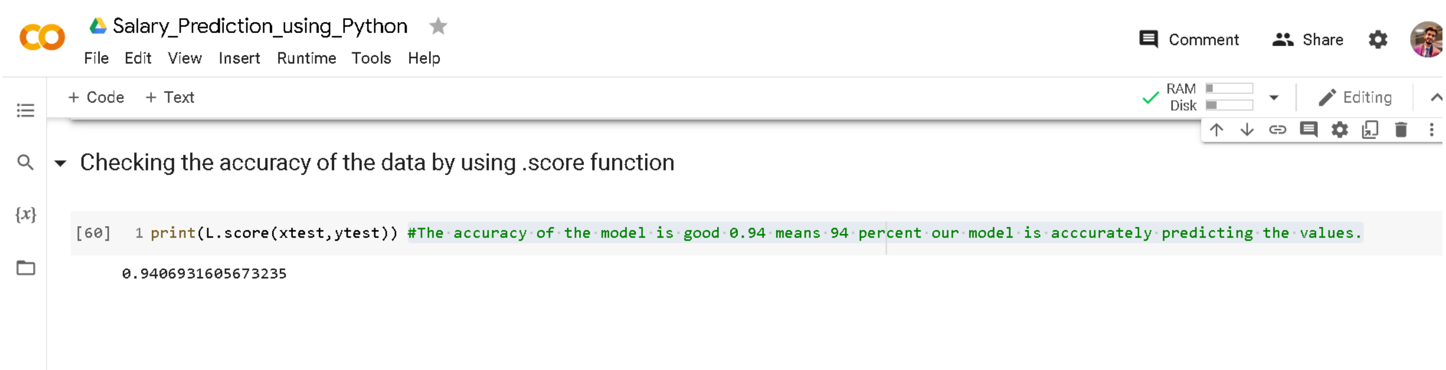
Step 4: We will predict using **.pred** function to see the prediction of the values from the model





Note: Here y_pred values are the values which our model has predicted for ytest.

Step 5: Checking the accuracy of the data by using **.score** function



The accuracy of the model is good 0.94 means 94 percent our model is acccurately predicting the v alues.

Conclusion

The predictions are pretty close to the actual data, which means the variance is pretty less. But we can definitely achieve more accuracy, as I already mentioned. We should also keep in mind that we only have one feature in our dataset, and having more features will definitely improve accuracy.