Simple Linear Regression Model using Python: Machine Learning

Learning how to build a simple linear regression model in machine learning using Jupyter notebook in Python

tds

Kaushik Katari

·

Follow

Published in

Towards Data Science

·

8 min read

·

Oct 9, 2020

141

Photo by Kevin Ku on Unsplash

In the previous article, **the Linear Regression Model,** we have seen how the linear regression model works theoretically using Microsoft Excel. This article will see how we can build a linear regression model using Python in the Jupyter notebook.

Simple Linear Regression

To predict the relationship between two variables, we'll use a simple linear regression model.

In a simple linear regression model, we'll predict the outcome of a variable known as the dependent variable using only one independent variable.

We'll directly dive into building the model in this article. More about the linear regression model and the factors we have to consider are explained in detail here.

Building a linear regression model

To build a linear regression model in python, we'll follow five steps:

1. Reading and understanding the data
2. Visualizing the data
3. Performing simple linear regression
4. Residual analysis
5. Predictions on the test set

Reading and understanding the data

In this step, first, we'll import the necessary libraries to import the data. After that, we'll perform some basic commands to understand the structure of the data.

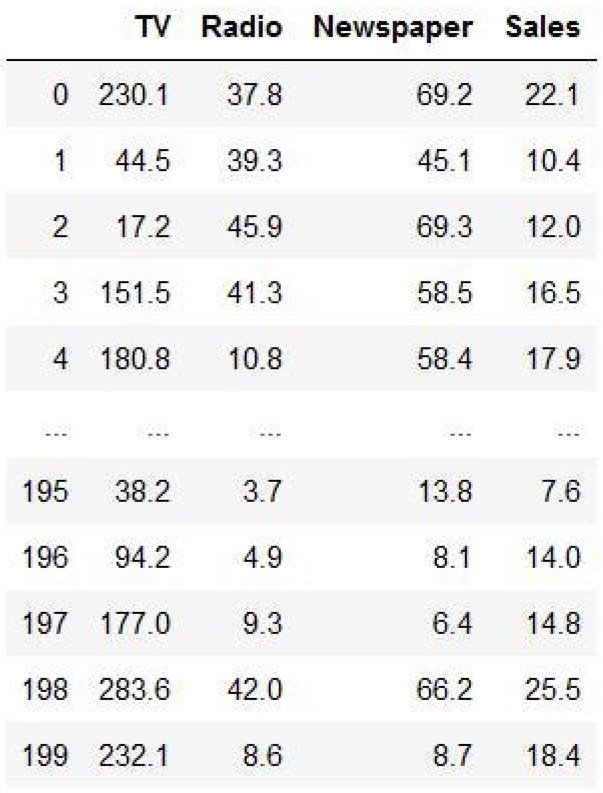
We can download the sample dataset, which we'll be using in this article from here.

Let's assume we have a company's data, where there is the amount spent on different types of advertisements and its subsequent sales.

Import libraries

We'll import the `numpy` and `pandas` library in the Jupyter notebook and read the data using `pandas`.

The dataset looks like this. Here our target variable is the Sales column.



	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

Advertising data of a company

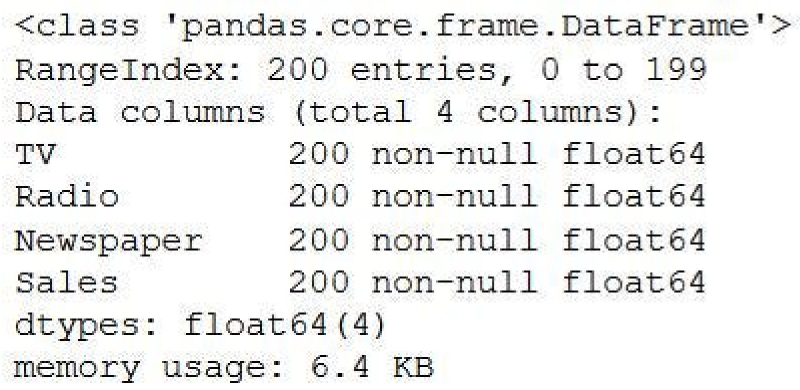
Understand the data

Let’s perform some tasks to understand the data like `shape`, `info`, and `describe`.

The `shape` of our dataset is,

```
(200, 4)
```

Using the `info`, we can see whether there are any null values in the data. If yes, then we have to do some data manipulation.

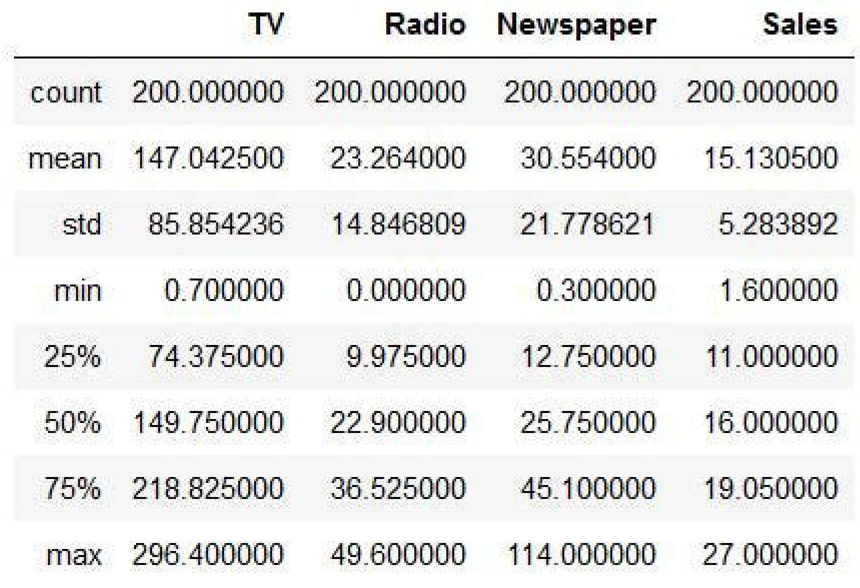


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
TV           200 non-null float64
Radio        200 non-null float64
Newspaper    200 non-null float64
Sales        200 non-null float64
dtypes: float64(4)
memory usage: 6.4 KB
```

Info of the dataset

As we can observe, there are no null values present in the data.

Using `describe`, we'll see whether there is any sudden jump in the data's values.



	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

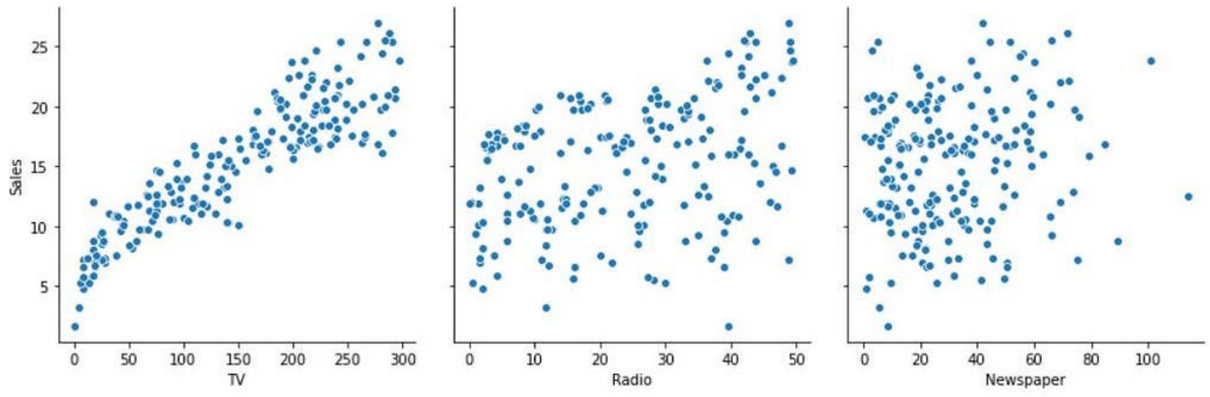
Describing the dataset

The values present in the columns are pretty consistent throughout the data.

Visualizing the data

Let's now visualize the data using the `matplolib` and `seaborn` library. We'll make a pairplot of all the columns and see which columns are the most correlated to `Sales`.

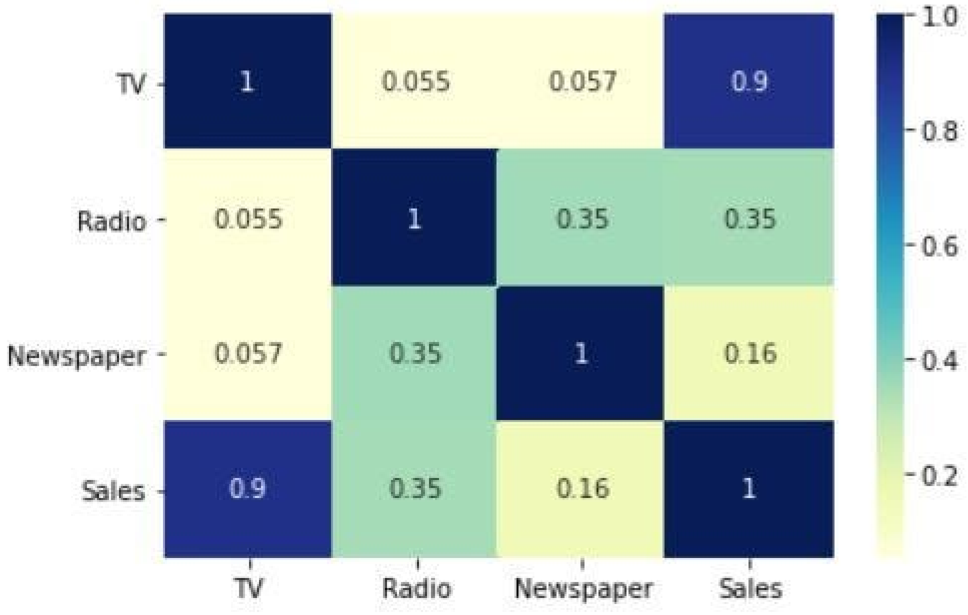
It is always better to use a scatter plot between two numeric variables. The pairplot for the above code looks like,



Pairplot of each Column w.r.t. Sales column

If we cannot determine the correlation using a scatter plot, we can use the seaborn heatmap to visualize the data.

The heatmap looks like this,



Heatmap of all the columns in the data

As we can see from the above graphs, the TV column seems most correlated to Sales.

Let’s perform the simple linear regression model using TV as our feature variable.

Performing Simple Linear Regression

Equation of simple linear regression

`y = c + mX`

In our case:

`y = c + m * TV`

The m values are known as **model coefficients** or **model parameters.**

We’ll perform simple linear regression in four steps.

1. Create X and y
2. Create Train and Test set
3. Train your model
4. Evaluate the model

Create X and y First, we’ll assign our feature variable/column `TV` as `X` and our target variable `Sales` as `y`.

To generalize,

The independent variable represents `X`, and `y` represents the target variable in a simple linear regression model.

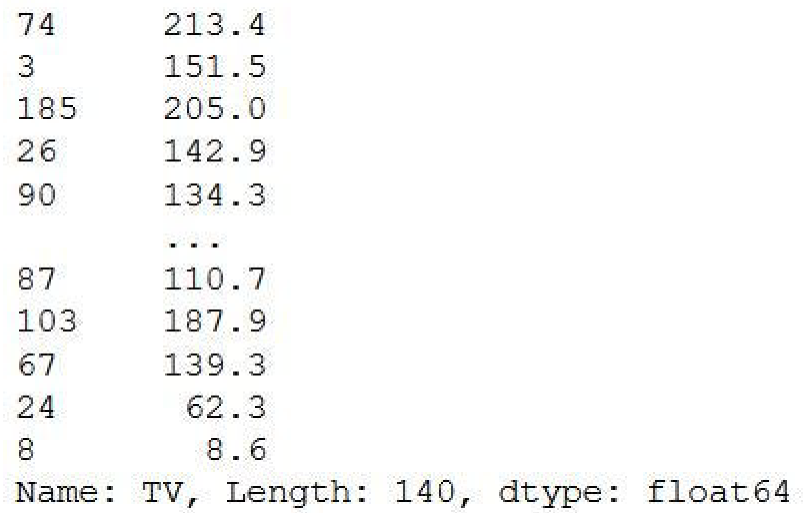
Create Train and Test sets

We need to split our variables into training and testing sets. Using the training set, we'll build the model and perform the model on the testing set. We'll divide the training and testing sets into a 7:3 ratio, respectively.

We'll split the data by importing `train_test_split` from the `sklearn.model_selection` library.

Let's take a look at the training dataset,

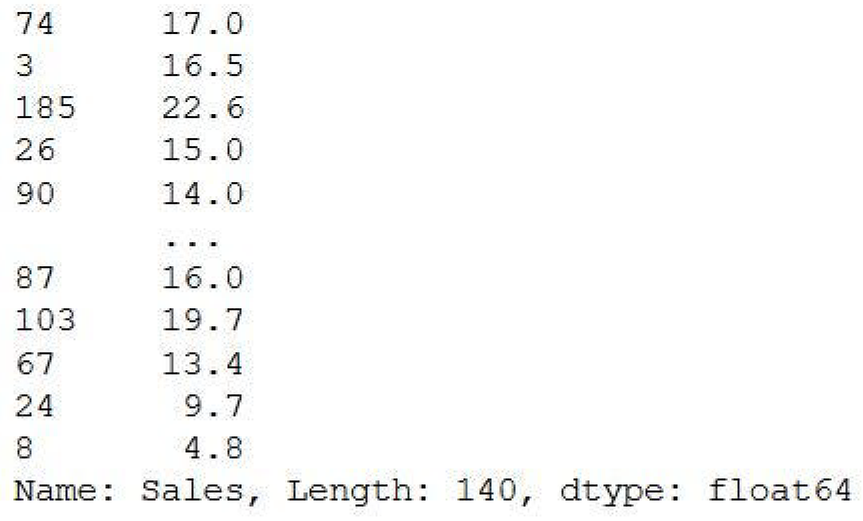
X_train data looks like this after splitting.



```
74     213.4
3      151.5
185    205.0
26     142.9
90     134.3
       ...
87     110.7
103    187.9
67     139.3
24      62.3
8        8.6
Name: TV, Length: 140, dtype: float64
```

X_train data after splitting

y_train data looks like this after splitting.



```
74     17.0
3      16.5
185    22.6
26     15.0
90     14.0
       ...
87     16.0
103    19.7
67     13.4
24      9.7
8       4.8
Name: Sales, Length: 140, dtype: float64
```

y_train data after splitting

Building and training the model

Using the following two packages, we can build a simple linear regression model.

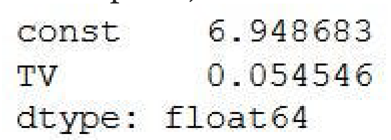
- `statsmodel`
- `sklearn`

First, we'll build the model using the `statsmodel` package. To do that, we need to import the `statsmodel.api` library to perform linear regression.

By default, the `statsmodel` library fits a line that passes through the origin. But if we observe the simple linear regression equation `y = c + mX`, it has an intercept value as `c`. So, to have an intercept, we need to add the `add_constant` attribute manually.

Once we've added constant, we can fit the regression line using `OLS` (Ordinary Least Square) method present in the `statsmodel`. After that, we'll see the parameters, i.e., `c` and `m` of the straight line.

The output is,

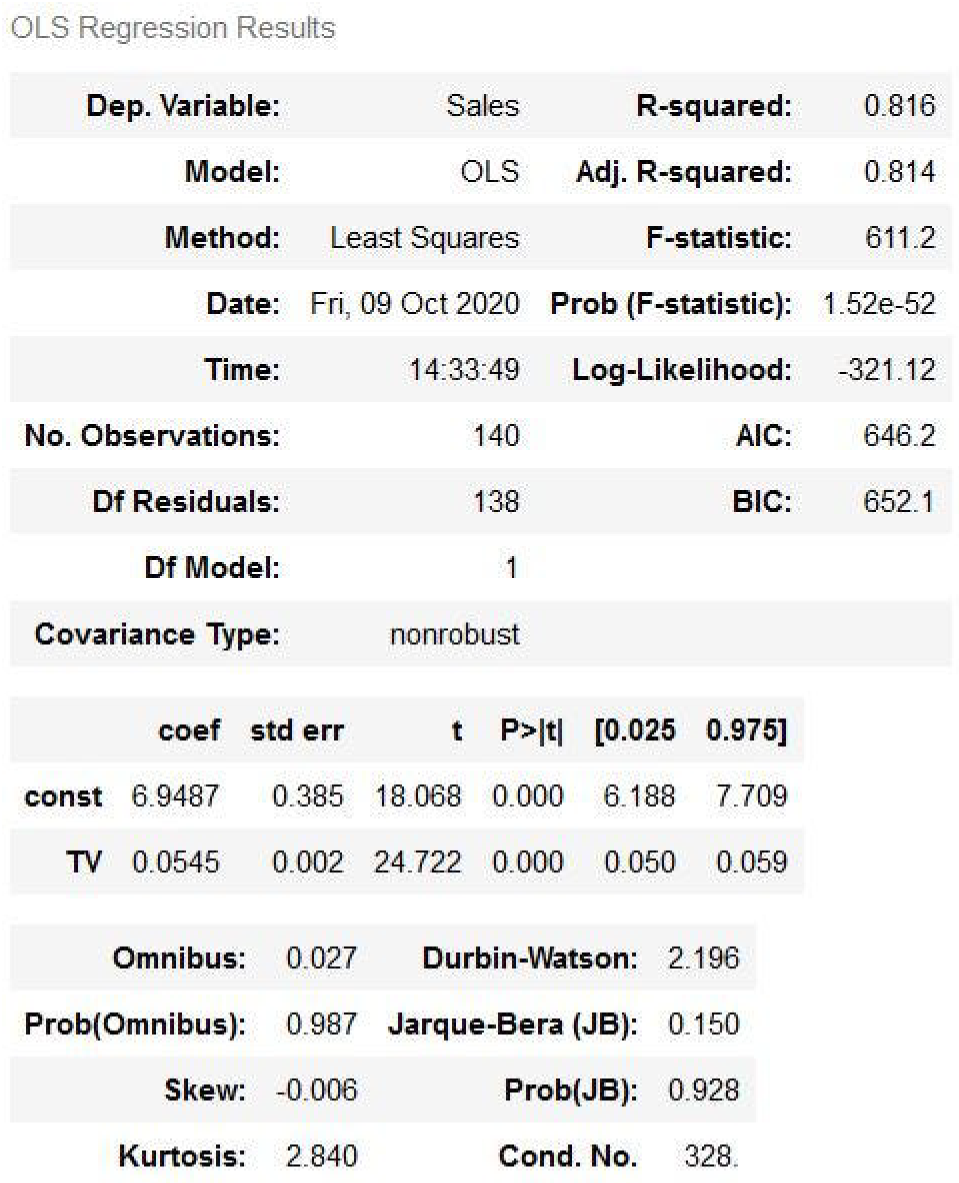


```
const     6.948683
TV        0.054546
dtype: float64
```

Intercept and Slope of the line

Let’s see the summary of all the different parameters of the regression line fitted like R^2, probability of `F-statistic`, and `p-value`.

The statistics for the above regression line is,



OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.816
Model:	OLS	**Adj. R-squared:**	0.814
Method:	Least Squares	**F-statistic:**	611.2
Date:	Fri, 09 Oct 2020	**Prob (F-statistic):**	1.52e-52
Time:	14:33:49	**Log-Likelihood:**	-321.12
No. Observations:	140	**AIC:**	646.2
Df Residuals:	138	**BIC:**	652.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	6.9487	0.385	18.068	0.000	6.188	7.709
TV	0.0545	0.002	24.722	0.000	0.050	0.059

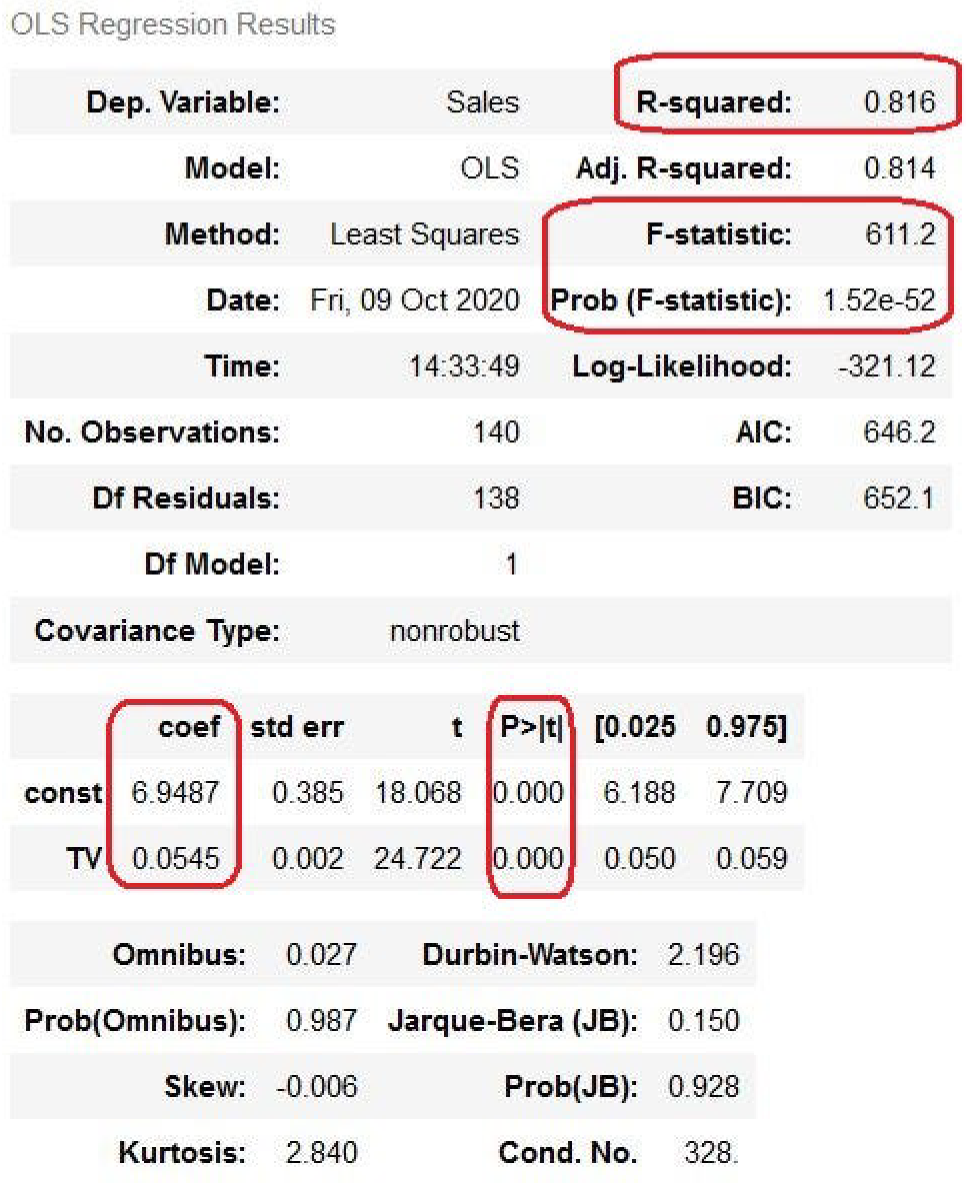
Omnibus:	0.027	**Durbin-Watson:**	2.196
Prob(Omnibus):	0.987	**Jarque-Bera (JB):**	0.150
Skew:	-0.006	**Prob(JB):**	0.928
Kurtosis:	2.840	**Cond. No.**	328.

All the statistics for the above best-fit line

So, the statistics we are mainly concerned with to determine whether the model is viable or not are:

1. The `coefficients` and its `p-value`(significance)
2. `R-squared` value

3. `F-statistic` and its significance



OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.816
Model:	OLS	**Adj. R-squared:**	0.814
Method:	Least Squares	**F-statistic:**	611.2
Date:	Fri, 09 Oct 2020	**Prob (F-statistic):**	1.52e-52
Time:	14:33:49	**Log-Likelihood:**	-321.12
No. Observations:	140	**AIC:**	646.2
Df Residuals:	138	**BIC:**	652.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	6.9487	0.385	18.068	0.000	6.188	7.709
TV	0.0545	0.002	24.722	0.000	0.050	0.059

Omnibus:	0.027	**Durbin-Watson:**	2.196
Prob(Omnibus):	0.987	**Jarque-Bera (JB):**	0.150
Skew:	-0.006	**Prob(JB):**	0.928
Kurtosis:	2.840	**Cond. No.**	328.

Statistics we need to look

1. The `coefficient` for TV is 0.054, and its corresponding `p-value` is very low, almost 0. That means the `coefficient` is statistically significant.

We have to make sure that the p-value should always be less for the coefficient to be significant

2. `R-squared` value is 0.816, which means that 81.6% of the `Sales` variance can be explained by the `TV` column using this line.

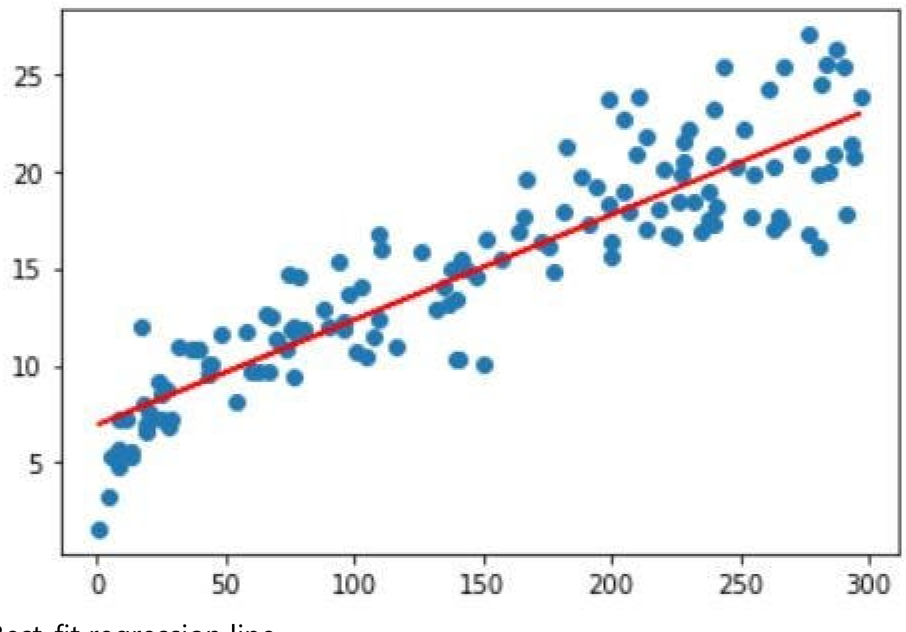
3. Prob `F-statistic` has a very low `p-value`, practically zero, which gives us that the model fit is statistically significant.

Since the fit is significant, let's go ahead and visualize how well the straight-line fits the scatter plot between `TV` and `Sales` columns.

From the parameters, we got the values of the `intercept` and the `slope` for the straight line. The equation of the line is,

```
Sales = 6.948 + 0.054 * TV
```

The graph looks like this,



Best-fit regression line

This is how we build a simple linear regression model using training data. Now before evaluating the model on test data, we have to perform residual analysis.

Residual Analysis

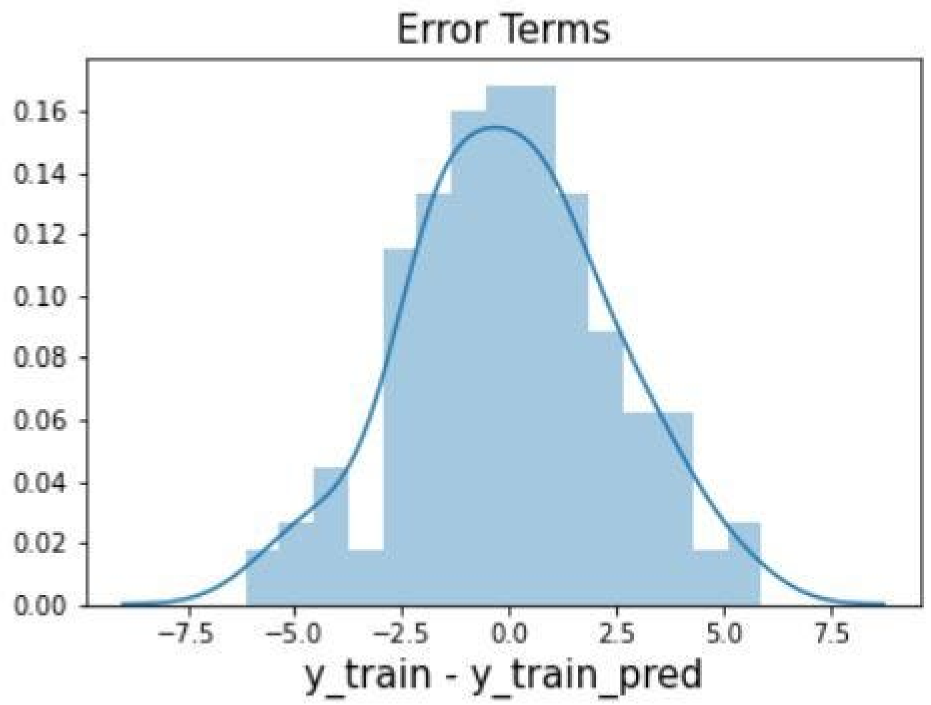
One of the major assumptions of the linear regression model is the error terms are normally distributed.

```
Error = Actual y value - y predicted value
```

Now from the dataset,
We have to predict the y value from the training dataset of X using the `predict` attribute. After that, we'll create the error terms(Residuals) from the predicted data.

Now, let's plot the histogram of the residuals and see whether it looks like normal distribution or not.

The histogram of the residuals looks like,

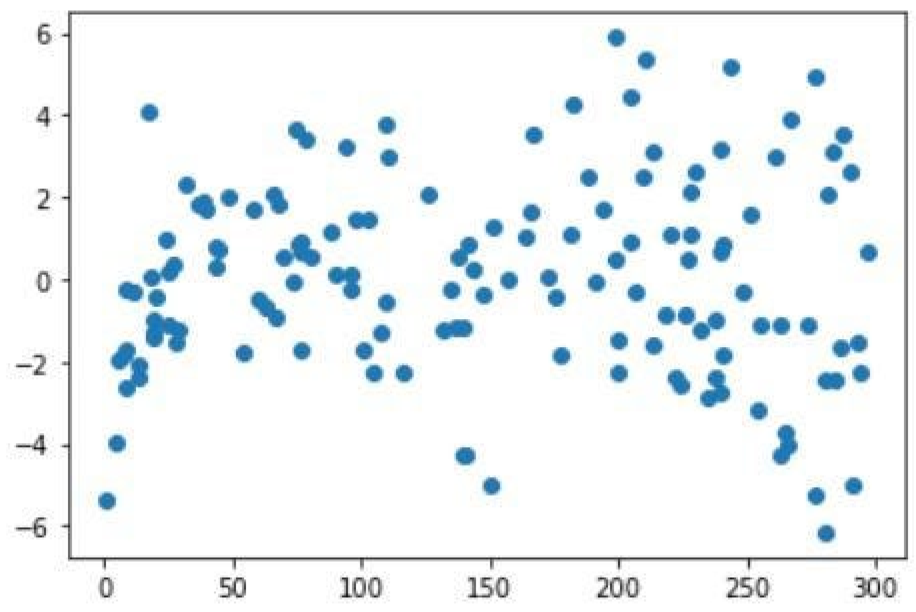


Residuals distribution

As we can see, the residuals are following the normal distribution graph with a mean 0.

Now, make sure that the residuals are not following any specific pattern.

The scatter plot looks like,



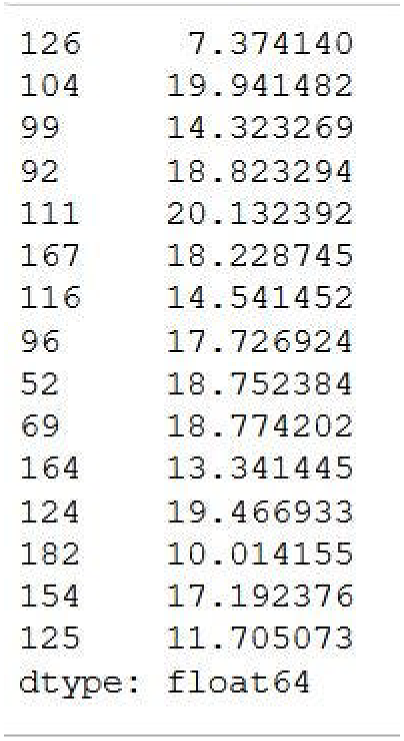
Scatter plot of Residual values

Since the Residuals follow a normal distribution and do not follow any specific pattern, we can use the linear regression model we have built to evaluate test data.

Predictions on the Test data or Evaluating the model

Now that we have fitted the regression line on our train dataset, we can make some predictions to the test data. Similar to the training dataset, we have to `add_constant` to the test data and predict the y values using the `predict` attribute present in the `statsmodel`.

The predicted y-values on test data are,



```
126     7.374140
104    19.941482
99     14.323269
92     18.823294
111    20.132392
167    18.228745
116    14.541452
96     17.726924
52     18.752384
69     18.774202
164    13.341445
124    19.466933
182    10.014155
154    17.192376
125    11.705073
dtype: float64
```

Predicted y-values

Now, let’s calculate the `R²` value for the above-predicted y-values. We can do that by merely importing the `r2_score` library from `sklearn.metrics` package.

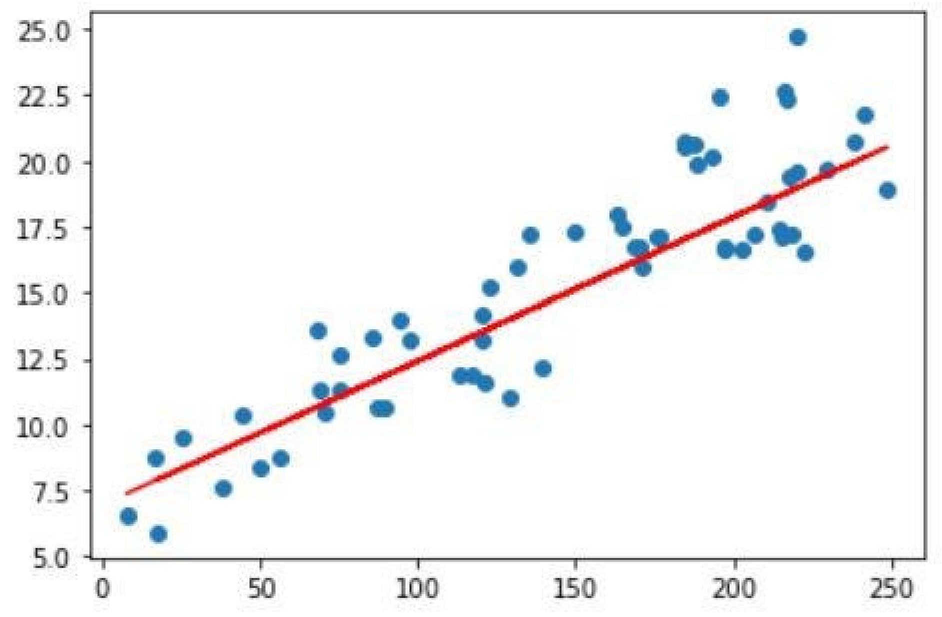
The R^2 value by using the above code = 0.792

If we can remember from the training data, the R^2 value = 0.815

Since the R^2 value on test data is within 5% of the R^2 value on training data, we can conclude that the model is pretty stable. Which means, what the model has learned on the training set can generalize on the unseen test set.

Let’s visualize the line on the test data.

The scatter-plot with best-fit line looks like,



Best-fit line on test data

This is how we build a linear regression model using the `statsmodel` package.

Apart from the `statsmodel`, we can build a linear regression model using `sklearn`. Using the `linear_model` library from `sklearn`, we can make the model.

Similar to `statsmodel`, we'll split the data into `train` and `test`.

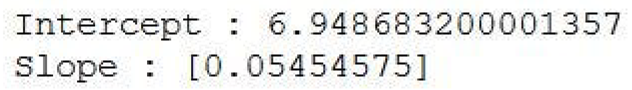
For simple linear regression, we need to add a column to perform the regression fit properly.

The shape of X_train before adding a column is `(140, )`.
The shape of X for train and test data is `(140, 1)`.

Now, let’s fit the line to the plot importing the `LinearRegression` library from the `sklearn.linear_model`.

Now, let’s find the coefficients of the model.

The value of intercept and slope is,



```
Intercept : 6.948683200001357
Slope : [0.05454575]
```

Coefficient Values

The straight-line equation we get for the above values is,

`Sales = 6.948 + 0.054 * TV`

If we observe, the equation we got here is the same as the one we got in the `statsmodel`.

After that, we’ll make the predictions and on the data and evaluate the model by comparing the R^2 values.

The R^2 values of the train and test data are

R^2 train_data = 0.816

R^2 test_data = 0.792

Same as the `statesmodel`, the R^2 value on test data is within 5% of the R^2 value on training data. We can apply the model to the unseen test set in the future.

Conclusion

As we have seen, we can build a linear regression model using either a `statsmodel` or `sklearn`.

We have to make sure to follow these five steps to build the simple linear regression model:

1. Reading and understanding the data
2. Visualizing the data
3. Performing simple linear regression
4. Residual analysis
5. Predictions on the test set

In the next article, we'll see how the multiple linear regression model works.

Thank you for reading and **happy coding!!!**

Check out my previous articles here

- **Linear Regression Model: Machine Learning**
- **Exploratory Data Analysis(EDA): Python**
- **Central Limit Theorem(CLT): Data Science**
- **Inferential Statistics: Data Analysis**
- **Seaborn: Python**
- **Pandas: Python**

- **[Matplotlib: Python]()**
- **[NumPy: Python]()**

References

- **Machine Learning — Linear Regression:** https://www.w3schools.com/python/python_ml_linear_regression.asp
- **Linear Regression in Python:** https://realpython.com/linear-regression-in-python/
- **Linear Regression (Python Implementation):** https://www.geeksforgeeks.org/linear-regression-python-implementation/
- **A Beginner's Guide to Linear Regression in Python with Scikit-Learn:** https://www.kdnuggets.com/2019/03/beginners-guide-linear-regression-python-scikit-learn.html

Machine Learning

Python

Linear Regression

Residual

R Square

141

tds

Written by Kaushik Katari

456 Followers

·Writer for

Towards Data Science

Software Engineer | Python | Machine Learning | Writer

Follow

More from Kaushik Katari and Towards Data Science

Kaushik Katari

in

Towards Data Science

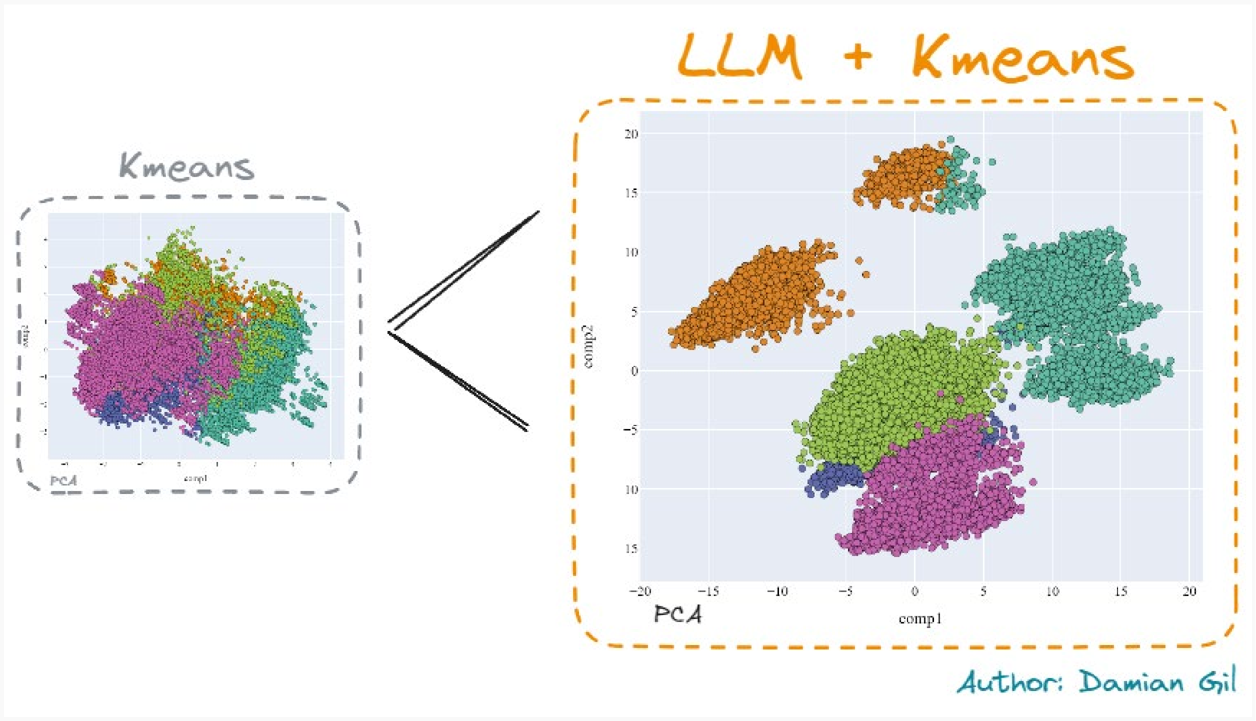
Exploratory Data Analysis(EDA): Python

Learning the basics of Exploratory Data Analysis using Python with Numpy, Matplotlib, and Pandas.

·11 min read·Aug 21, 2020

585

5



Damian Gil

in

Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

23 min read·Sep 26

2.8K

24

Khouloud El Alami

in

Towards Data Science

Don’t Start Your Data Science Journey Without These 5 Must-Do Steps From a Spotify Data Scientist

A complete guide to everything I wish I’d done before starting my Data Science journey, here’s to acing your first year with data

·18 min read·Sep 24

2.1K

21

Kaushik Katari

in

Towards Data Science

Inferential Statistics: Data Analysis

Statistics is one of the essential subject matters in data science, which provides tools and methods to give more in-depth insights into…

·12 min read·Aug 27, 2020

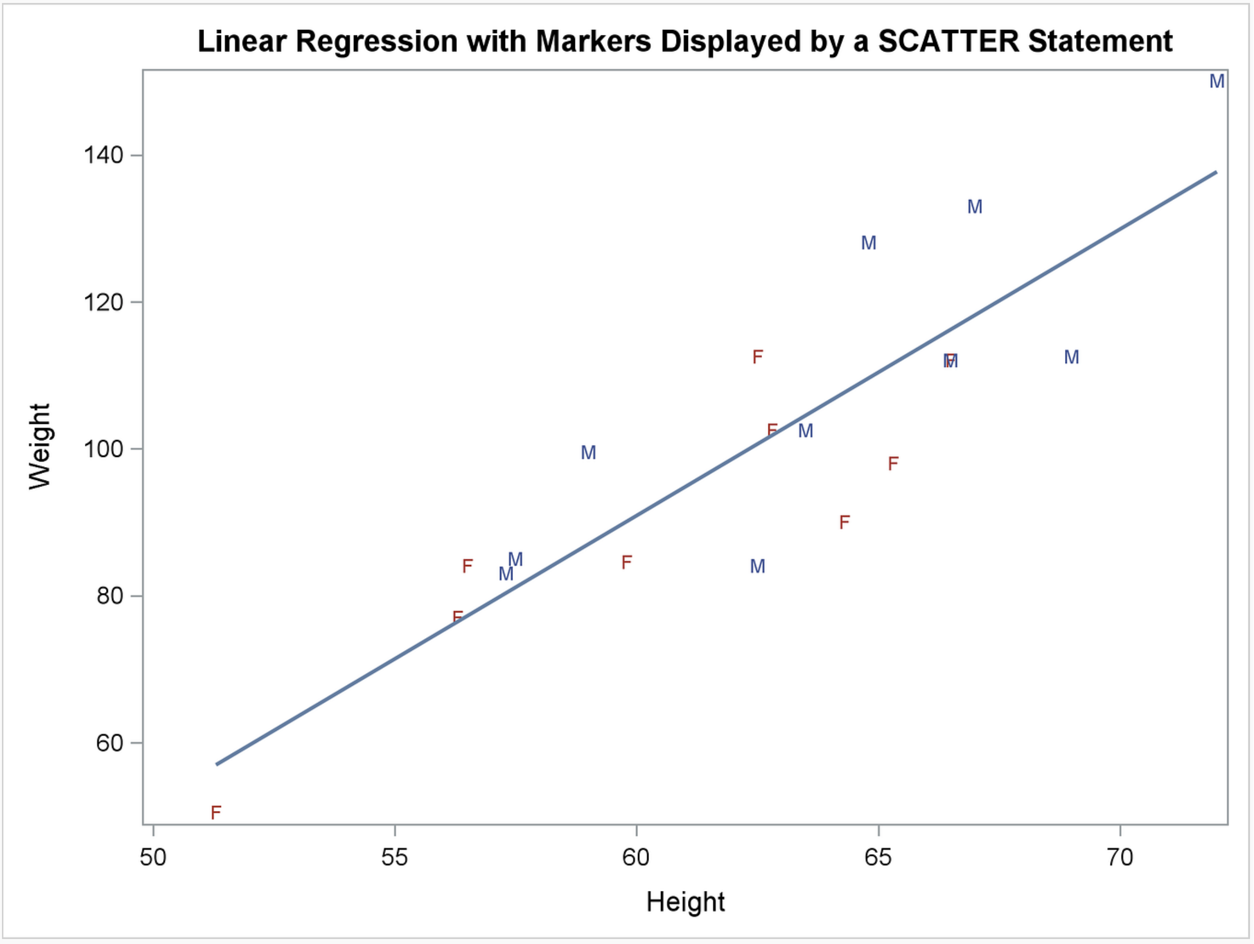
50

1

See all from Kaushik Katari

See all from Towards Data Science

Recommended from Medium



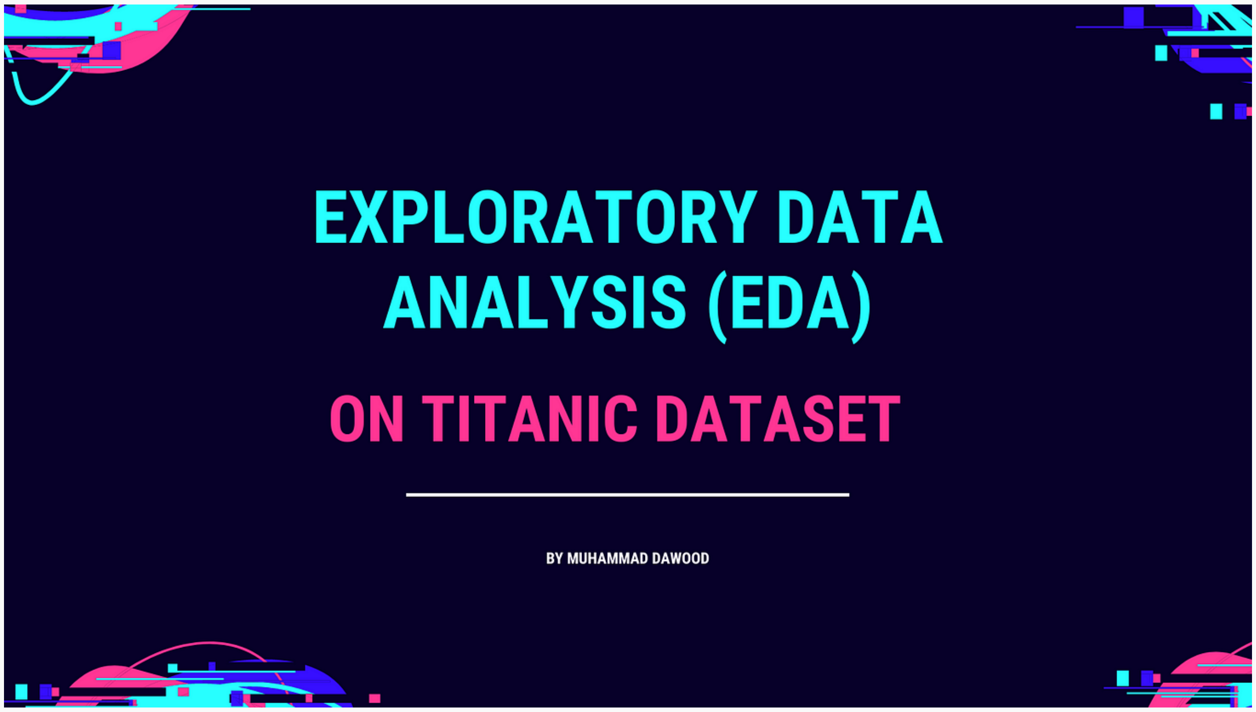
D Sunitha

Linear Regression

What is Linear Regression?

6 min read·Jul 22

4



Muhammad Dawood

Exploratory Data Analysis (EDA) on Titanic Dataset

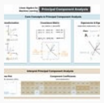
The Titanic dataset is popular for data analysis and machine learning. It contains information about the passengers onboard the Titanic, 3 min read·Jun 15

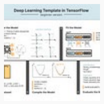
136

Lists

Predictive Modeling w/ Python

20 stories·474 saves





Practical Guides to Machine Learning

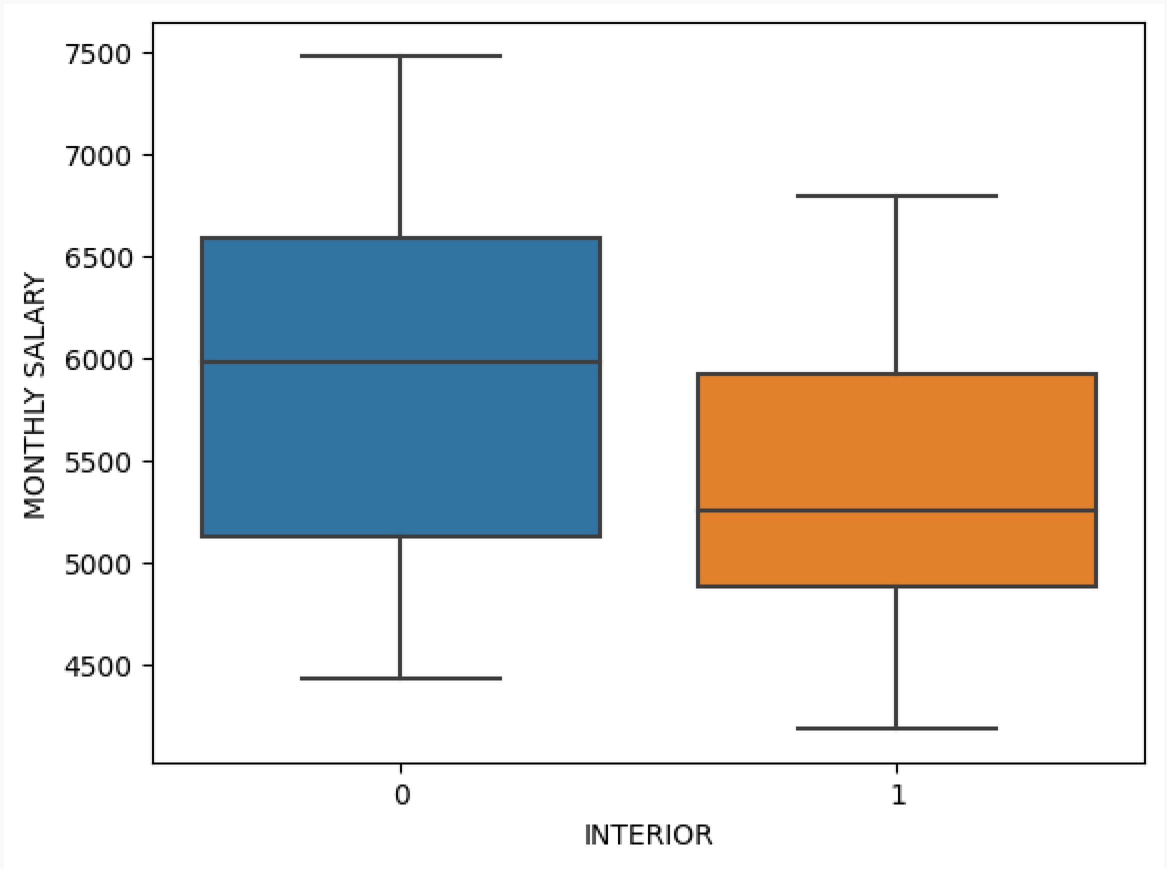
10 stories·545 saves

Coding & Development

11 stories·207 saves

Natural Language Processing

689 stories·305 saves



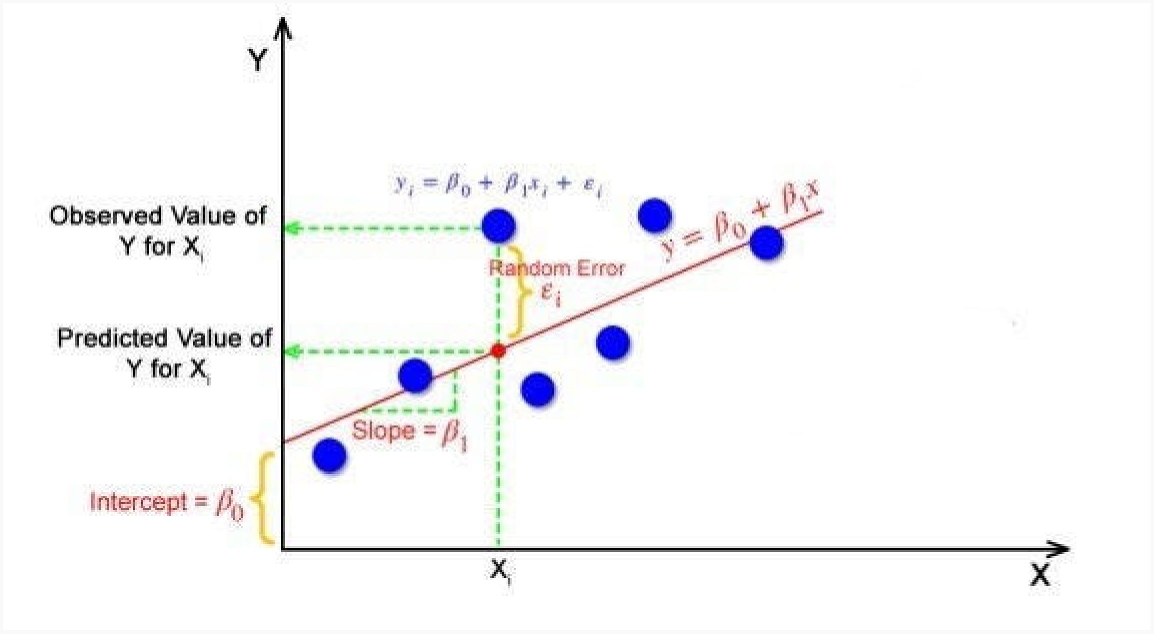
Vitor Sampaio

Understanding Ordinary Least Squares (OLS): The Foundation of Linear Regression

Understanding the Inner Workings of Ordinary Least Squares for Effective Predictive Modeling

11 min read·Jun 2

21


Y
X
$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
$y = \beta_0 + \beta_1 x$
Observed Value of Y for X_i
Predicted Value of Y for X_i
Random Error ε_i
Slope = β_1
Intercept = β_0
X_i

lakshya ruhela

LINEAR REGRESSION

HOW DO WE DEFINE IT?

6 min read·Sep 19

58

Manoj Mangam

Multicollinearity Problems in Linear Regression. Clearly Explained!

A behind-the-scenes look at the infamous multicollinearity

11 min read·Mar 21

237

7

Vijay Reddiar

Multivariate time series forecasting and analysis of the US unemployment rate—Part 2

In the previous post (linked here: Part 1) we discussed the importance of unemployment rate forecasting. We also framed how we approach…

9 min read·May 8

61