Gurucharan M K Follow
Jun 30, 2020 · 5 min read ★ · Listen

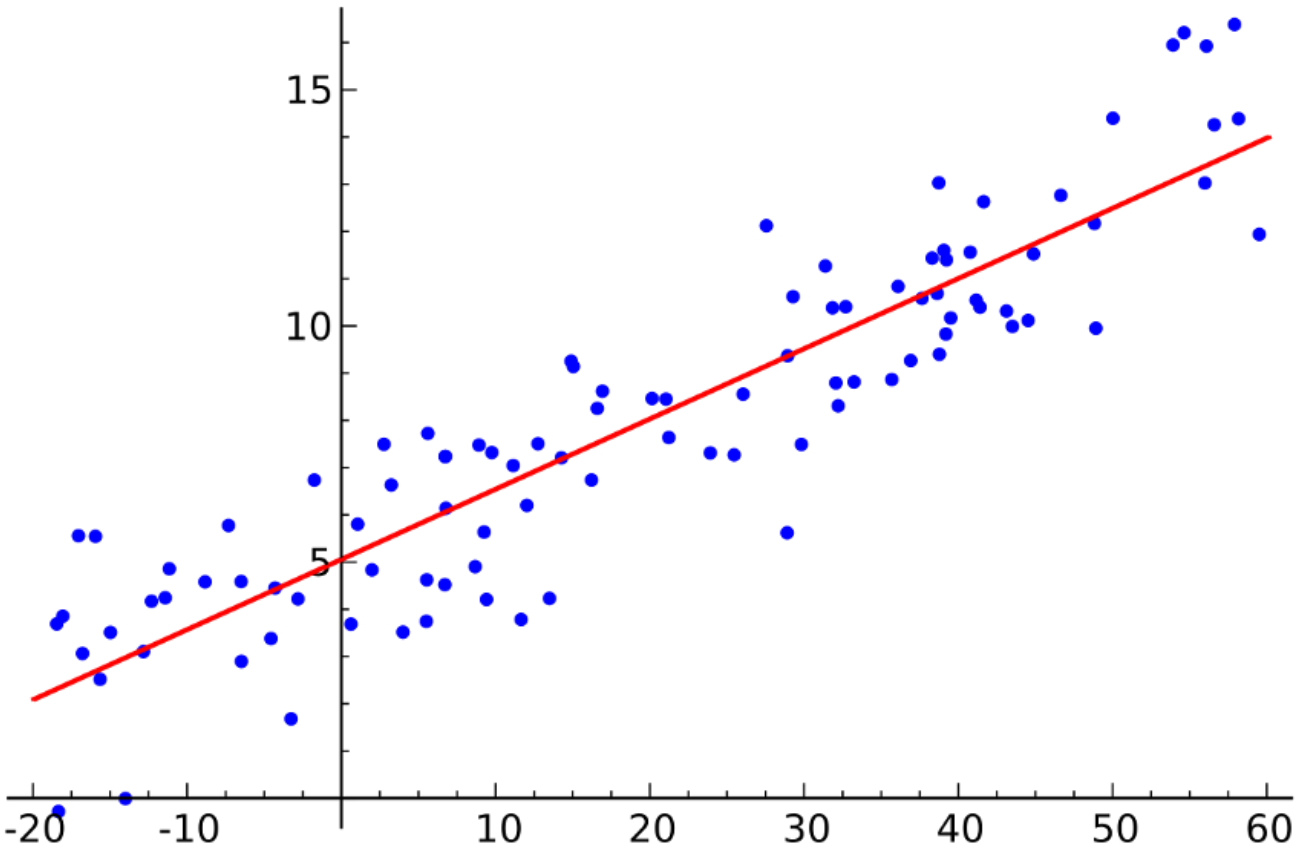
Machine Learning Basics: Multiple Linear Regression

Learn to Implement Multiple Linear Regression with Python programming.

In the previous story, I had given a brief of Linear Regression and showed how to perform Simple Linear Regression. In Simple Linear Regression, we had one dependent variable (y) and one independent variable (x). What if the marks of the student depended on two or more independent variables?

Overview

In this example, we will go through the implementation of ***Multiple Linear Regression***, in which we will predict the profit of startups for a venture capitalist who wants to analyse whether a startup is worth investing to get good returns.



Linear Regression (Source)

Problem Analysis

In this data, we have the four independent variables namely, *R&D Spend, Administration, Marketing Spend* and *State*. There is one independent variable i.e., *Profit*. So, our job is to train the ML model with this data to understand the correlation between each of the four features (or independent variables) and predict a profit for another new company with all these data.

Step 1: Importing the libraries

In this first step, we will be importing the libraries required to build the ML model. The ***NumPy*** library and the ***matplotlib*** are imported. Additionally, we have imported the ***Pandas*** library for data analysis.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In the next step, we shall use pandas to store the data obtained from my github repository and store it as a Pandas DataFrame named as "***dataset***" using the function "*pd.read_csv*".

We go through our dataset and assign the independent variable (x) to the first four columns of our dataset, namely R&D Spend (index=0), Administration (index=1), Marketing Spend (index=2) and State (index=3).

```
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Regression/master/Startups_Data.csv')

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

dataset.head(5)

>>
R&D Spend  Administration  Marketing Spend   State       Profit
165349.20  136897.80       471784.10         New York    192261.83
162597.70  151377.59       443898.53         California  191792.06
153441.51  101145.55       407934.54         Florida     191050.39
144372.41  118671.85       383199.62         New York    182901.99
142107.34  91391.77        366168.42         Florida     166187.94
```

We use the corresponding .iloc function to slice the DataFrame to assign these indexes to X. Here, we use **[:, :-1]** which can be interpreted as **[include all rows, include all columns upto -1 (excluding -1)]**. In this, -1 refers to the first column from the last. Thus, we assign the 0th, 1st, 2nd and 3rd column as X.

We assign the last column (-1) to the dependent variable which is y. We print the DataFrame to see if we have got the correct columns for our training data.

Step 3: Encoding Categorical Data

As long as there are numbers in the dataset, we can easily apply mathematical computations on the dataset and create prediction models. In this dataset, we come across a non-number variable that is "***State***". This is also called as categorical data.

We encode this categorical data using another important library called as **sklearn**. In

transformed separately. In our case, we use the OneHotEncoder to transform our "State" column (index=3) to numerical data.

After encoding the categorical data, We print our DataFrame X and see the changes. We see that there has been an inclusion of three new columns at the beginning. Each column represents one of the "States". For example, in the first row, the third column represents "New York" and hence the value "1" in the third column.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
[3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

Step 4: Splitting the dataset into the Training set and Test set

Once we have our dataset ready, the next important task is to split our dataset into training set and test set. We do this in order to train our model with one portion of the data called the "**training set**" and test the prediction results on another set of data called the "**test set**".

We use the "train_test_split" function to split our data. Here, we give the "test_size =0.2", which indicates that 20% of the data is the test set. In our case, 10 random startup data will be chosen as the test set and 40 remaining startup data will be chosen for the training 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)
```

Step 5: Training the Multiple Linear Regression model on the Training set

In the next step, we import the "**LinearRegression**" class which is going to be applied to our training set. We assign a variable "**regressor**" to the LinearRegression class. We then use the "**regressor.fit**" to fit the training dataset (X_train and y_train) to this

```
regressor.fit(X_train, y_train)
```

Step 6: Predicting the Test Set results

In the next step, we are going to predict the profit of the test set using the trained model namely "regressor". The real values (profits) of the test set data(X_test) is stored in the variable y_test.

We then use the "**regressor.predict**" function to predict the values for our test data X_test. We assign the predicted values as y_pred. We now have two data, y_test (real values) and y_pred (predicted values).

```
y_pred = regressor.predict(X_test)
```

Step 7: Comparing the Test Set with Predicted Values

In this step, we shall print both the values of y_test as ***Real Values*** and y_pred values as ***Predicted Values*** of each X_test in a Pandas DataFrame. In this way, we obtain the values for all the 10 X_test data.

```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df

>>
Real Values Predicted Values
78239.91    74963.602167
182901.99   173144.548525
64926.08    45804.248438
105733.54   108530.843936
141585.52   127674.466487
108552.04   111471.421444
146121.95   133618.038644
105008.31   114655.651664
96778.92    96466.443219
97483.56    96007.236281
```

In the first row, the Real Values has a value of ***78239.91*** and the Predicted Values has a

Congratulations! You have now expanded your knowledge from building a Simple Linear Regression model to a Multiple Linear Regression model. I am attaching a link of my Github repository where you can find the Python notebook and the data files for your reference.

mk-gurucharan/Regression

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build...

github.com

You can also find the explanation of the program for other Regression models below:

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression

We will come across the more complex models of Regression, Classification and Clustering in the upcoming articles. Till then, Happy Machine Learning!