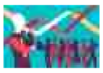
Applied Multivariate Regression



Article 4 of the Machine Learning Series

In this article we will be getting introduced to the concepts of Multivariate regression. We will also be discussing about a common problem associated with the algorithm i.e. The Dummy Variable Trap.

First we will be getting familiar with the concepts of Multivariate regression and then we will build our very own multivariate regression model. However before getting started I would recommend you to take a look at my article on Simple Linear Regression for understanding the concepts even better .

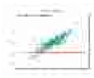
What is Multivariate Regression ?

Multivariate regression is an extension of simple linear regression. It is used when we want to predict the value of a variable based on the value of two or more different variables. The variable we want to predict is called the **Dependent Variable**, while those used to calculate the dependent variable are termed as **Independent Variables**.

The mathematical function/hypothesis of a Multivariate regression is of the form:

$$y = \beta_0 + \beta_1.x_1 + \ldots + \beta_n.x_n$$

where, n represents the number of independent variables, β0~ βn represent the coefficients and x1~xn, are the independent variable



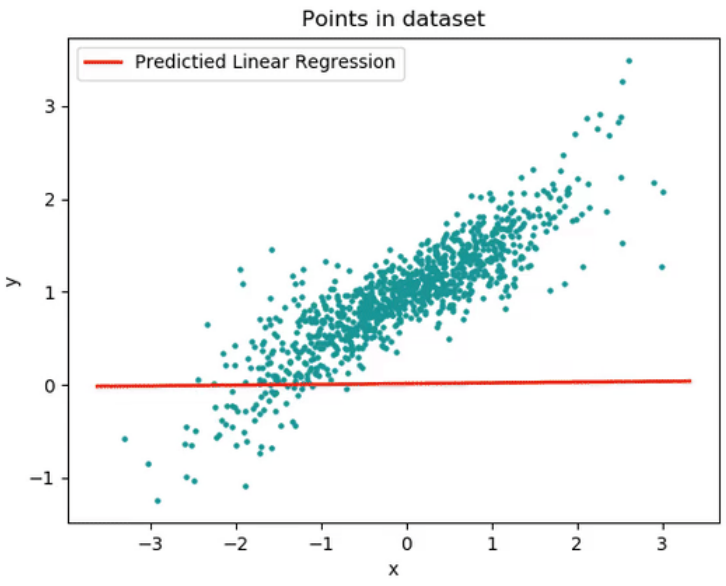


Image by Franky from CDOT Wiki

Using a multivariate model helps us compare coefficients across outcomes. What makes a multivariate or multiple linear regression a better model is a small cost function.

Cost Function

In simple words it is a function that assigns a cost to instances where the model deviates from the observed data. In this case, our cost is the **sum of squared errors**. The cost function for multiple linear regression is given by:

$$\mathrm{MSE} = \frac{1}{2m} \Sigma\left(h_\theta(x)^{(i)} - y^i\right)^2$$

We can understand this equation as the summation of square of difference between our predicted value and the actual value divided by twice of length of data set. A smaller mean squared error implies a better performance. Generally a cost function is used along with the Gradient Descent algorithm to find the best parameters.

Image by Nattanan Kanchanaprat from Pixabay

Now we will be practically applying what we have learned by building a multivariate linear regression.

> *You can access the complete code and other resources for this regression model on my GitHub handle.*

In this example we will be predicting the value of a house, given the features like median income, average house age, number of rooms, households, average area, number of bedrooms and area population.

STEP 1: IMPORTING LIBRARIES AND LOADING THE DATA

Our first step is to import the libraries required to build our model. It is not necessary to import all the libraries at just one place. Python allows us to import any library at any place. To get started we will be importing Pandas, Numpy, Matplotlib and Seaborn libraries.

```
#Importing the libraries and and reading the data into a Pandas DataFrameimport pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as snstest=pd.read_csv("california_housing_test.csv")
train=pd.read_csv("california_housing_train.csv")
```

Once these libraries have been imported our next step will be fetching the dataset and loading the data into our notebook. For this example I have taken the California Housing dataset.

STEP 2: VISUALISING THE DATA

After successfully loading the data, our next step is to visualize this data. **Matplotlib** and **Seashore** are excellent libraries that can be used to visualize our data on various different plots.

```
#Visuaising our data using Seaborn Libraryplt.figure()
sns.heatmap(data.corr(), cmap='coolwarm')
plt.show()sns.lmplot(x='median_income', y='median_house_value', data=train)
sns.lmplot(x='housing_median_age', y='median_house_value', data=train)
```

STEP 3: FEATURE ENGINEERING

Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. I have selected only few columns to work with continuous numerical values only for this model.

```
data = data[[‘total_rooms’, ‘total_bedrooms’, ‘housing_median_age’, ‘median_income’, ‘population’, ‘
data.info()data['total_rooms'] = data['total_rooms'].fillna(data['total_rooms'].mean())
data['total_bedrooms'] = data['total_bedrooms'].fillna(data['total_bedrooms'].mean())
```

Feature Engineering becomes even more important when the number of features are very large. One of the most important use of feature engineering is that it reduces overfitting and improves the accuracy of a model.

STEP 4: FITTING THE MODEL

After selecting the desired parameters the next step is to import train_test_split from sklearn library which is used to split the dataset into training and testing data.

```
#Splitting training and testing datafrom sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train, y, test_size = 0.2, random_state = 0)
y_train = y_train.reshape(-1,1)
y_test = y_test.reshape(-1,1)
```

After this **LinearRegression** is imported from **sklearn.model_selection** and the model is fit over the training dataset. The intercept and coefficient of our model can be calculated as shown below:

```
#Fitting the model on training datafrom sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)#Calculating the Intercept and Coefficientprint(regressor.intercept_)
print(regressor.coef_)
```

The performance of the model can be evaluated by finding the root mean squared error of the model.

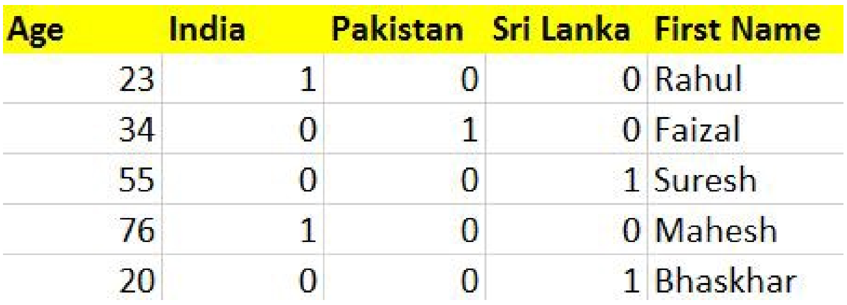
```
predictions = regressor.predict(X_test)
predictions = predictions.reshape(-1,1)#Calculate RMSE of the modelfrom sklearn.metrics import mean_
print(‘MSE:’, mean_squared_error(y_test,predictions))
print(‘RMSE:’, np.sqrt(mean_squared_error(y_test,predictions)))
```

The RMSE for our model is 0.6628934048044981. The plot for the calculated values against predicted values is as shown below:

Dummy Variable Trap

When working with multivariate models, it is easy to handle quantitative or numerical data. However, this is not the same for categorical data. They can’t be used directly and needs to be transformed.

Dummy variables are “proxy” variables used in place for categorical data that are sometimes present in the datasets that we use for building regression models. Using all dummy variables for regression models lead to dummy variable trap.



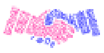
Age	India	Pakistan	Sri Lanka	First Name
23	1	0	0	Rahul
34	0	1	0	Faizal
55	0	0	1	Suresh
76	1	0	0	Mahesh
20	0	0	1	Bhaskhar

Image by Author

The **Dummy Variable trap** is a scenario in which the independent variables are multicollinear i.e. two or more variables are highly correlated. In order to avoid such a scenario it is recommended to design the regression model such that it excludes one of the dummy variables.

Here is an example to understand how this works:

Let's consider the case of gender which can have either of the two values male (0) or female (1). After using label encoding procedures to transform these categorical features to numerical attributes, if we include both the dummy variables then it causes redundancy, leading to Dummy Variable trap.



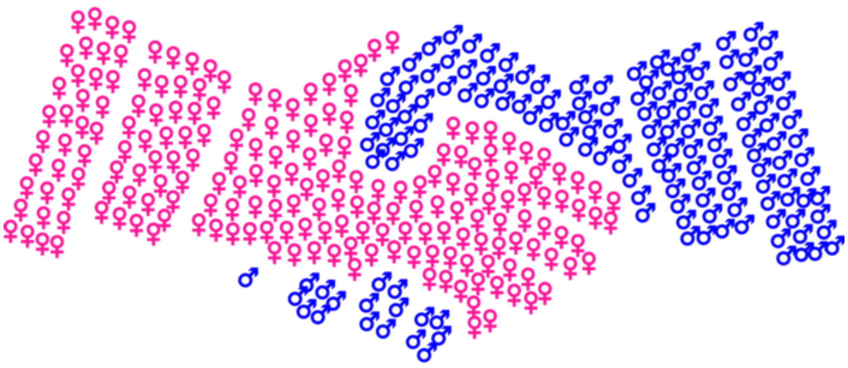


Image by Gordon Johnson from Pixabay

Such a problem can be avoided, if we don't use both the variables in regression models, as if a person is not male then it is expected that the person is a female. This way the dummy variable trap can be avoided

Resources

Here are some resources that you can use to get an even deeper understanding of the concepts of multivariate regression.

Now that we have reached the end of this article, I hope you would have found this article really informative. I hope you found it informative! If you have any question or if i have made any mistake, please contact me! You can get in touch with me via: Email or LinkedIn