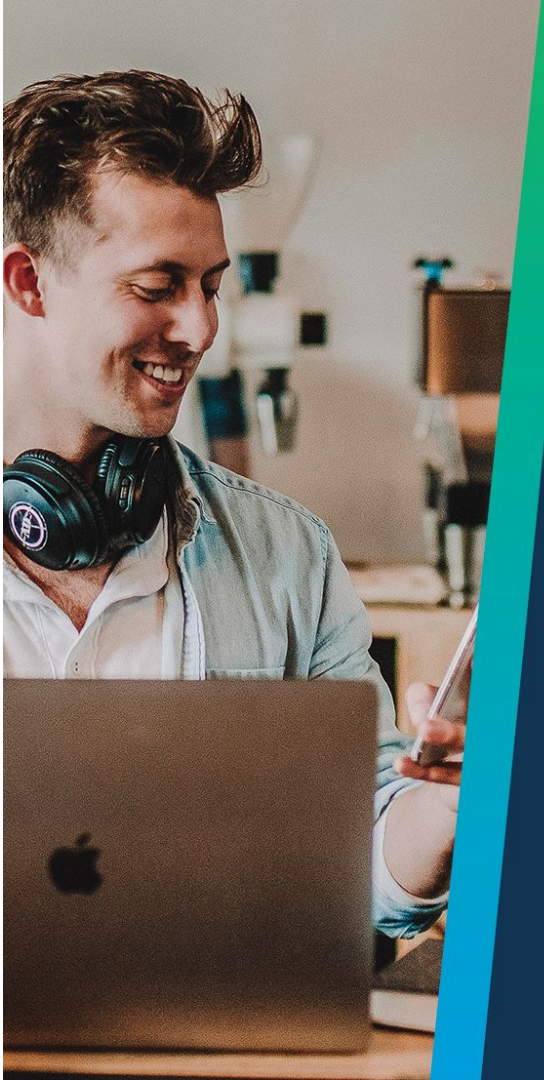


Data Science Bootcamp

Hyperiondev

Multiple Linear Regression

Welcome

Your Lecturer for this session

Sanana Mwanawina

Lecture - Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- You can also submit questions here: hyperiondev.com/sbc4-ds-questions
- For all non-academic questions, please submit a query: hyperiondev.com/support
- Report a safeguarding incident: hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Lecture - Code Repo

Go to: github.com/HyperionDevBootcamps

Then click on the "**C4_DS_lecture_examples**" repository, do view or download the code.

Objectives

- Explore Multiple Linear Regression
- Solidify our understanding of linear regression

Multiple Linear Regression

- In the last task, we saw how simple linear regression can shed light on the relationship between two variables. We looked only at the impact of a single independent variable on a single outcome variable. Our example case, however, had one independent variable: TV. Multiple Linear Regression allows us to model the impact of two or more variables combined on the outcome variable.

- Modelling multiple variables is quite useful. A simple linear regression approach may over- or underestimate the impact of a variable on the outcome because it does not have any information about the impact of other variables. Imagine, for example, that our radio budget and billboards budget were increased simultaneously and sales went up shortly thereafter. Our simple regression models would attribute the entire increase in sales to the single variable they were modelling, which would be incorrect. A multiple linear regression model could make a less biased prediction of how much each type of advertisement contributes to sales.

Multiple Linear Regression

- ★ The extension of simple linear regression is fairly straightforward. Instead of a function for Y with only one coefficient, the function has coefficients for each variable.

- ★ In the case of two variables, the formula takes the form:

$$Y = \beta 0 + \beta 1X1 + \beta 2X2$$

Note that the coefficients (βi) will not just be the same values as independent simple linear regression models would return. This extended model will adjust each value according to the relative contribution of each variable.

Simple vs Multiple Linear Regression

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Multiple Linear Regression

	1	2	3	4	5
1		TV	Radio	Billboard	Sales
2	1	230.1	37.8	69.2	22.1
3	2	44.5	39.3	45.1	10.4
4	3	17.2	45.9	69.3	9.3
5	4	151.5	41.3	58.5	18.5
6	5	180.8	10.8	58.4	12.9
7	6	8.7	48.9	75	7.2
8	7	57.5	32.8	23.5	11.8
9	8	120.2	19.6	11.6	13.2
10	9	8.6	2.1	1	4.8

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \beta 3X3$$

$$Sales = \beta 0 + \beta 1\ TV + \beta 2\ Radio + \beta 3\ Billboard$$

MLR results

```
Coefficients:  [ 0.04576465  0.18853002 -0.00103749]
Intercept:  2.9388893694594085
```

Sales = 2.939 + 0.046 TV + 0.189 Radio + (-0.001) Billboard

If we invest $100 000 towards the TV budget, how will this affect our sales?

Sales = 2.939 + 0.046(100) = 7.539 which is 7539 units

Notice, if we do not advertise at all, we will sell 2939 units.

Sales = 2.939 + 0.046(0) + 0.189(0) + (-0.001)(0) = 2.939
which is 2939 units

Training and Test Data

- When you are teaching a student arithmetic summation, you want to start by teaching them the pattern of summation, and then test whether they can apply that pattern. You will show them that 1+1=2, 4+5=9, 2+6=8, and so forth. If the student memorises all the examples, they could answer the question 'what is 2+6?' without understanding summation. To test whether they have recovered not just the facts but the pattern behind the facts, you need to test them on numbers that you have not exposed them to directly. For example, you might ask them 'what is 4+9?'
- This intuition forms the motivation behind training and testing data in machine learning. We create a model (e.g. regression) based on some data that we have, but we do not use all of our data: we hide some data samples from the model and then test our model on the hidden data samples to confirm that the model is making valid predictions as opposed to reiterating what was given to it in the training dataset.

Training and Testing Data

A **training set** is the actual dataset that we use to train the model. The model sees and learns from this data. A **test set** is a set of held-back examples used only to assess the performance of a model. Although the best ratio depends on how much data is available, a common split is a ratio of 80:20. For example, 8000 training examples and 2000 test cases. Sklearn allows us to do this quite easily:

Example

```
# import from sklearn the train_test_split functionality
from sklearn.model_selection import train_test_split

# pass your x and y variables in the function and get all four at the same
time
# the test size parameter is used to tell how to split, 0.25 means 25% to be
test samples

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25)
```

Training and Testing Data

- One thing to watch out for when dividing the data is that the test set and training set should not be systematically different. If the total data set is ordered in some way, for example by alphabet or by the time of collection, the test set might contain different kinds of instances from the training set. If that happens, the model will perform badly, not because it did not learn from the data well, but because it is tested on a different kind of task from the one it learned to do.

- For this purpose, it is customary to investigate the distribution of labels in your data and make sure they are similar across your training and test data, to check that samples with different labels are distributed suitably. In the next example, notice that there are no 0 samples in the test set.

Example

```
from sklearn.model_selection import train_test_split

X = [1,2,3,4,2,6,7,8,6,7]
y = [0,0,0,0,0,1,1,1,1,1]

x_train,x_test,y_train,y_test= train_test_split(X,y,test_size =
0.2,shuffle=False)
print("y_train {}".format(y_train))

print("y_test {}".format(y_test))

#which prints: (notice there are no 0 labels in the test set)
>> y_train [0, 0, 0, 0, 0, 1, 1, 1]
>>y_test [1, 1]
```

Training and Testing Data

This can largely be solved using shuffle=True as a parameter in train_test_split. This means that the labels would be distributed randomly between the training and testing set, so it would be less likely that the training and testing data would be systematically different.

```
#notice there is now a 0 label in the test set
>>y_train [1, 0, 0, 1, 0, 1, 0, 1]
>>y_test [0, 1]
```

Training and Testing Data

A further way to ensure that data is represented equally in both the training and testing data is to make use of the stratify parameter. This ensures that labels are represented in as close to the same proportions as possible in both the training and testing data

```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size= 0.5,
shuffle=True, stratify=y)
#note that the labels are now distributed in the same proportion across the
train and test sets
>>y_train [0, 0, 1, 1, 1, 0]
>>y_test [1, 0, 0, 1]
```

Training and Test Error

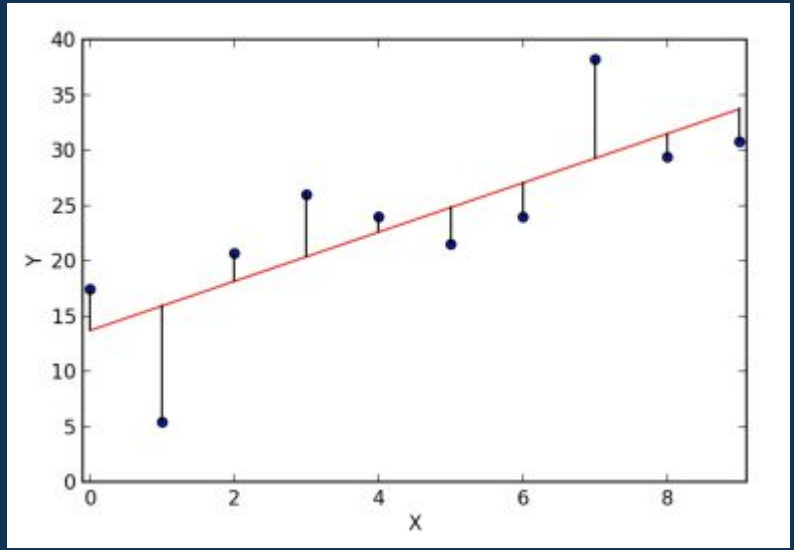
- ★ We've discussed before how a regression model is only an approximation of the data. The model predicts values that are close to the observed training values, in hopes of making good predictions on unseen data.

- ★ The difference between the actual values and the predictions is called the error. The training dataset error and the test dataset error can be obtained by comparing the predictions to the known, true labels (the gold standard). Of the training error and test error, the test error value is more important as it is the more reliable assessment of the value of the model.

- ★ After all, we want to use our model on unknown data points, as opposed to applying it to cases for which we already have the actual outcome. In most cases, the training error value will also be lower than the test error value.

Training and Test Error

There are many different ways to measure the error. As said, the error is the difference between observed and predicted values, as in this plot:



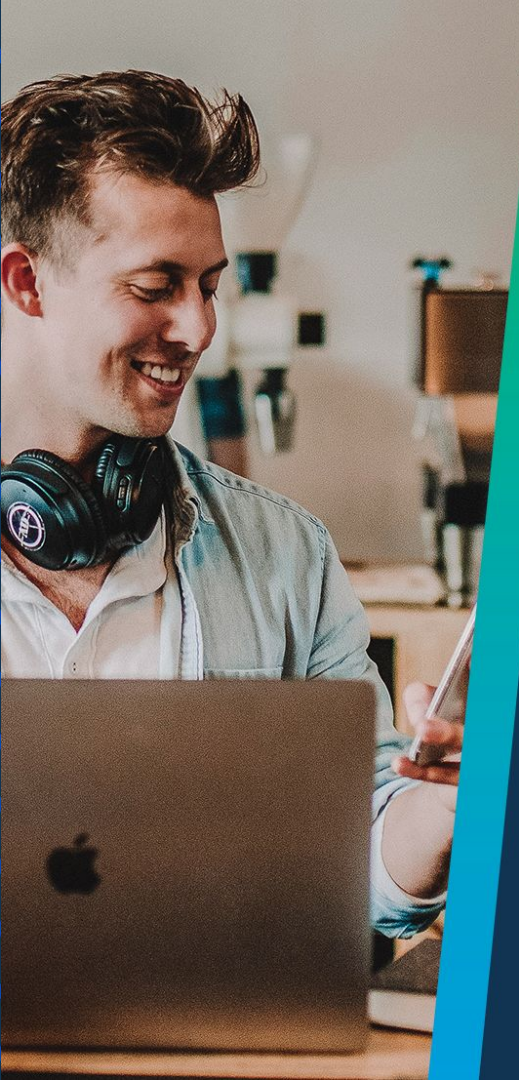
Training and Test Error

We have observed data (**Y**) in dark blue and prediction of a regression model (**y_pred**) along the red line. The error is depicted by vertical black lines. There are a number of different ways one can aggregate the error to get a final score for the model. Two common ones are the Root Mean Squared Error (RMSE) and R squared (R**2).

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us