DL_L_5_Linear_Regression

Linear Regression

In Siraj's video, he predicted the body weight of an animal from the weight of it's brain using linear regression. In this section, you'll use linear regression to predict life expectancy from body mass index (BMI). Before you do that, let's go over the tools required to build this model.
For your linear regression model, you'll be using **scikit-learn's LinearRegression** class. This class provides the function **fit()**to fit the model to your data.
- from sklearn.linear_model import LinearRegression
- model = LinearRegression()
- model.fit(x_values, y_values)
In the example above, the model variable is a linear regression model that has been fitted to the data x_values and y_values. Fitting the model means finding the best line that fits the training data. Let's make two predictions using the model's predict() function.

- print(model.predict([[127], [248]]))

[[438.94308857, 127.14839521]]
The model returned an array of predictions, one prediction for each input array. The first input, [127], got a prediction of 438.94308857. The seconds input, [248], got a prediction of 127.14839521. The reason for predicting on an array like [127] and not just 127, is because you can have a model that makes a prediction using multiple features. We'll go over using multiple variables in linear regression later in this lesson. For now, let's stick to a single value.
Linear Regression Quiz
In this quiz, you'll be working with data on the average life expectancy at birth and the average BMI for males across the world. The data comes from Gapminder.
The data file can be found under the "bmi_and_life_expectancy.csv" tab in the quiz below. It includes three columns, containing the following data:
Country – The country the person was born in. Life expectancy – The average life expectancy at birth for a person in that country. BMI – The mean BMI of males in that country. You'll need to complete each of the following steps:

1. Load the data

The data is in the file called "bmi_and_life_expectancy.csv". Use pandas read_csv to load the data into a dataframe. Assign the dataframe to the variable bmi_life_data.

1. Build a linear regression model

Create a regression model using scikit-learn's LinearRegression and assign it to bmi_life_model. Fit the model to the data.

1. Predict using the model

Predict using a BMI of 21.07931 and assign it to the variable laos_life_exp.

In [ ]:

```
# TODO: Add import statements
import pandas as pd
from sklearn.linear_model import LinearRegression

# Assign the dataframe to this variable.
# TODO: Load the data
bmi_life_data = pd.read_csv("bmi_and_life_expectancy.csv")

# Make and fit the linear regression model
#TODO: Fit the model and Assign it to bmi_life_model
bmi_life_model = LinearRegression()
bmi_life_model.fit(bmi_life_data[['BMI']], bmi_life_data[['Life expectancy']])

# Mak a prediction using the model
# TODO: Predict life expectancy for a BMI value of 21.07931
laos_life_exp = bmi_life_model.predict(21.07931)
```

/usr/local/lib/python3.4/dist-packages/sklearn/utils/validation.py:386: DeprecationWarning: Passing 1d arrays as data is deprecated in 0.17 and willraise ValueError in 0.19. Reshape your data either using X.reshape(-1, 1) if your data has a single feature or X.reshape(1, -1) if it contains a single sample. DeprecationWarning)

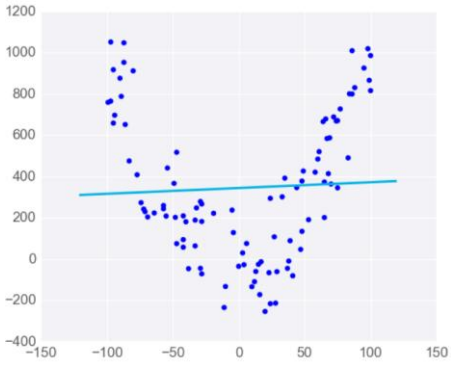
1. The data was loaded correctly!
2. Well done, you fitted the model!
3. Well done, your prediction of a life expectancy 60.315647164 is correct!

Linear Regression Warnings

Linear regression comes with a set of implicit assumptions and is not the best model for every situation. Here are a couple of issues that you should watch out for.

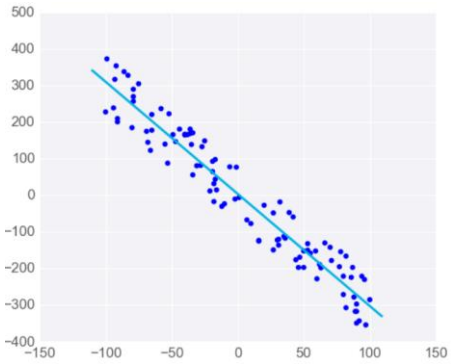
Linear Regression Works Best When the Data is Linear

Linear regression produces a straight line model from the training data. If the relationship in the training data is not really linear, you'll need to either make adjustments (transform your training data), add features (we'll come to this next) or use another kind of model.

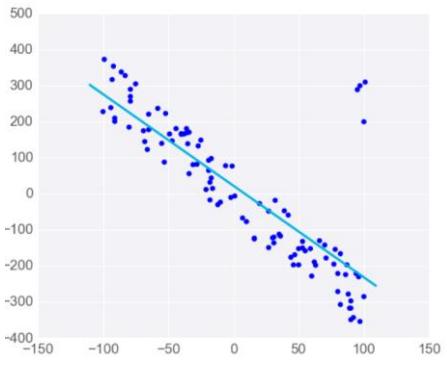


Linear Regression is Sensitive to Outliers

Linear regression tries to find a 'best fit' line among the training data. If your dataset has some outlying extreme values that don't fit a general pattern, they can have a surprisingly large effect.
In this first plot, the model fits the data pretty well.



However, adding a few points that are outliers and don't fit the pattern really changes the way the model predicts.



In most circumstances, you'll want a model that fits most of the data most of the time, so watch out for outliers!

Multiple Linear Regression

In the last section, you saw how we can predict life expectancy using BMI. Here, BMI was the **predictor**, also known as an independent variable. A predictor is a variable you're looking at in order to make predictions about other variables, while the values you are trying to predict are known as dependent variables. In this case, life expectancy was the dependent variable.
Now, let's say we get new data on each person's heart rate as well. Can we create a prediction of life expectancy using both BMI and heart rate?

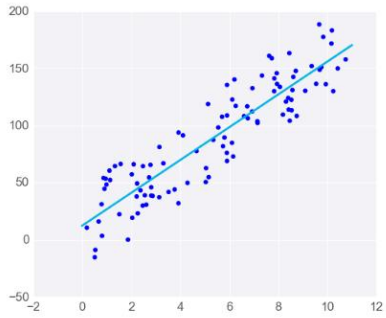
Absolutely! We can do just that using multiple linear regression.

If the outcome you want to predict depends on more than one variable, you can make a more complicated model that takes this into account. As long as they're relevant to the situation, using more independent/predictor variables can help you get a better prediction.

When there's just one predictor, the linear regression model is a line, but as you add more predictor variables, you're adding more dimensions to the picture.
When you have one predictor variable, the equation of the line is

$$y=mx+b$$

and the plot might look something like this:

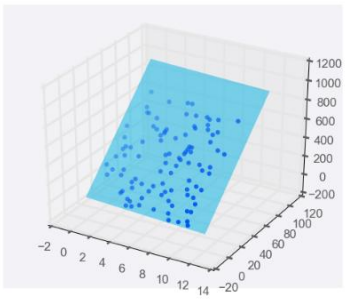


Linear regression with one predictor variable

Adding a predictor variable to go to two predictor variables means that the predicting equation is:

$$y = m_1x_1 + m_2x_2 + b$$

To represent this graphically, we'll need a three-dimensional plot, with the linear regression model represented as a plane:



Linear regression with two predictor variables

You can use more than two predictor variables, in fact you should use as many as is useful! If you use *n*predictor variables, then the model can be represented by the equation

$$y = m_1x_1 + m_2x_2 + m_3x_3 + \ldots + m_nx_n + b$$

As you make a model with more predictor variables, it becomes harder to visualise, but luckily, everything else about linear regression stays the same. We can still fit models and make predictions in exactly the same way - time to try it!

Programming Quiz: Multiple Linear Regression

In this quiz, you'll be using the **Boston house-prices dataset**. The dataset consists of 13 features of 506 houses and their median value in $1000's. You'll fit a model on the 13 features to predict on the value of houses. You'll need to complete each of the following steps:

1. Build a linear regression model

- Create a regression model using scikit-learn's `LinearRegression` and assign it to `model`.
- Fit the model to the data.

2. Predict using the model

- Predict the value of `sample_house`.

In [ ]:

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_boston

# Load the data from the the boston house-prices dataset
boston_data = load_boston()
x = boston_data['data']
y = boston_data['target']

# Make and fit the linear regression model
# TODO: Fit the model and Assign it to the model variable
model = LinearRegression()
model.fit(x, y)

# Make a prediction using the model
sample_house = [[2.29690000e-01, 0.00000000e+00, 1.05900000e+01,
0.00000000e+00, 4.89000000e-01,
                 6.32600000e+00, 5.25000000e+01, 4.35490000e+00,
4.00000000e+00, 2.77000000e+02,
                 1.86000000e+01, 3.94870000e+02, 1.09700000e+01]]
# TODO: Predict housing price for the sample_house
prediction = model.predict(sample_house)
```

```
23.68420569227329 is the correct prediction!
```