**Linear Regression**

Linear regression is a **linear model**, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

**Linear Regression Model Representation**

In a simple regression problem (a single x and a single y), the form of the model would be:

y = B0 + B1\*x

In higher dimensions when we have more than one input (x), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. B0 and B1 in the above example).

It is common to talk about the complexity of a regression model like linear regression. This refers to the number of coefficients used in the model.

When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model (0 \* x = 0). This becomes  relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

Now that we understand the representation used for a linear regression model, let’s review some ways that we can learn this representation from data.

## Linear Regression: Learning the Model

Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available.

In this section we will take a brief look at four techniques to prepare a linear regression model. This is not enough information to implement them from scratch, but enough to get a flavour of the computation and trade-offs involved.

### 1. Simple Linear Regression

With simple linear regression when we have a single input, we can use statistics to estimate the coefficients.

This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics.

### 2. Ordinary Least Squares

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients.

The OLS procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.

This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations.

It is unusual to implement the Ordinary Least Squares procedure yourself unless as an exercise in linear algebra. It is more likely that you will call a procedure in a linear algebra library. This procedure is very fast to calculate.

### 3. Gradient Descent

When there are one or more inputs you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data.

This operation is called Gradient Descent and works by starting with random values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. A learning rate is used as a scale factor and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

When using this method, you must select a learning rate (alpha) parameter that determines the size of the improvement step to take on each iteration of the procedure.

Gradient descent is often taught using a linear regression model because it is relatively straightforward to understand. In practice, it is useful when you have a very large dataset either in the number of rows or the number of columns that may not fit into memory.

## Making Predictions with Linear Regression

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs.

Let’s make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

y = B0 + B1 \* x1

or

weight =B0 +B1 \* height

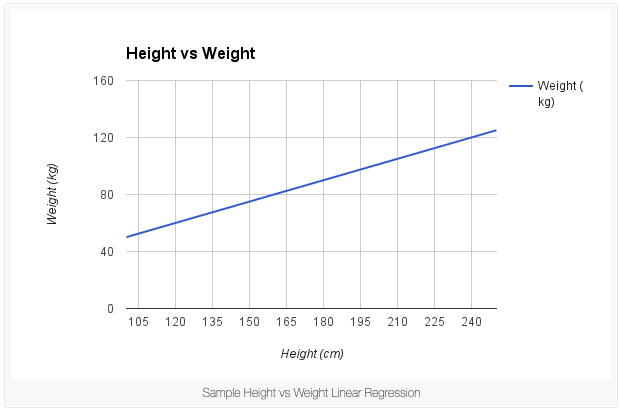
Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

For example, let’s use B0 = 0.1 and B1 = 0.5. Let’s plug them in and calculate the weight (in kilograms) for a person with the height of 182 centimeters.

weight = 0.1 + 0.5 \* 182

weight = 91.1

You can see that the above equation could be plotted as a line in two-dimensions. The B0 is our starting point regardless of what height we have. We can run through a bunch of heights from 100 to 250 centimeters and plug them to the equation and get weight values, creating our line.



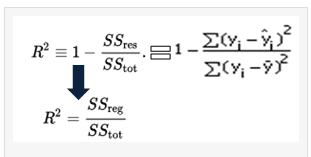
**Difference between R-Squared and Adjusted R Squared?**

**Short Answer:** In case of adjusted R2 we include only the important and useful variables, whereas in R2 all the variables are included.

**Mathematically:**

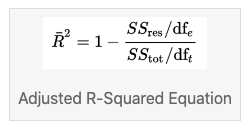
Mathematically, R-squared is calculated by dividing sum of squares of residuals (**SSres**) by total sum of squares (**SStot**) and then subtract it from 1. In this case, **SStot** measures total variation. **SSres**measures explained variation and **SSreg** measures unexplained variation.  
  
As,

**SSres + SSreg = SStot, R² = Explained variation / Total Variation**

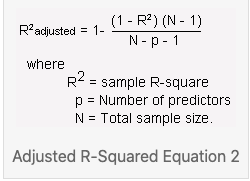
****

R-Squared is also called **coefficient of determination**. It lies between **0%**and **100%.**A r-squared value of 100% means the model explains all the variation of the target variable. And a value of 0% measures zero predictive power of the model.

**Higher R-squared value, better the model.**  
  
**Adjusted R-Squared**  
  
It measures the proportion of variation explained by only those independent variables that really help in explaining the dependent variable. It penalizes you for adding independent variable that do not help in predicting the dependent variable.  
  
Adjusted R-Squared can be calculated mathematically in terms of sum of squares. The only difference between R-square and Adjusted R-square equation is degree of freedom.



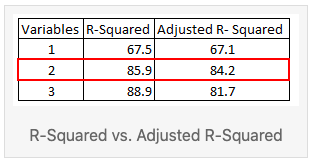
In the above equation, dft is the degrees of freedom n– 1 of the estimate of the population variance of the dependent variable, and dfe is the degrees of freedom n – p – 1 of the estimate of the underlying population error variance.  
  
Adjusted R-squared value can be calculated based on value of r-squared, number of independent variables (predictors), total sample size.



**Difference between R-square and Adjusted R-square**

1. Every time you add an independent variable to a model, the R-squared increases, even if the independent variable is insignificant. It never declines. Whereas Adjusted R-squared increases only when independent variable is significant and affects dependent variable.

In the table below, adjusted r-squared is maximum when we included two variables. It declines when third variable is added. Whereas r-squared increases when we included third variable. It means third variable is insignificant to the model.



2.Adjusted r-squared can be negative when r-squared is close to zero.

3.Adjusted r-squared value always be less than or equal to r-squared value.

**Which is Better?**

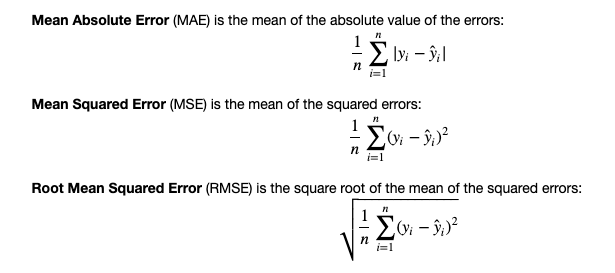
*Adjusted R-square should be used to compare models with different numbers of independent variables. Adjusted R-square should be used while selecting important predictors (independent variables) for the regression model.*

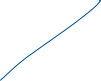
Code:

*import numpy as np  
y = np.array([21, 21, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2])  
yhat = np.array([21.5, 21.14, 26.1, 20.2, 17.5, 19.7, 14.9, 22.5, 25.1, 18])  
R2 = 1 - np.sum((yhat - y)\*\*2) / np.sum((y - np.mean(y))\*\*2)  
R2  
n=y.shape[0]  
p=3  
adj\_rsquared = 1 - (1 - R2) \* ((n - 1)/(n-p-1))  
adj\_rsquared*

**What metrics can we used for regression problems?**







Some Additional Things to keep in mind:

**Assumptions of Linear Regression?**

**Short Trick**: Assumptions can be abbreviated as LINE in order to remember.

**L** : Linearity ( Relationship between x and y is linear)

**I**  : Independence (Observations are independent of each other)

**N** : Normality (for any fix value of x, y is normally distributed)

**E** : Equal Variance (homoscedasticity)

**What is Multicollinearity? How to Avoid?**

Multicollinearity occurs when your model includes multiple factors that are correlated not just to your response variable, but also to each other. In other words, it results when you have factors that are a bit redundant.

You can think about it in terms of a cricket game: If one player catches the ball, it's easy to give credit. But if three players are going for the catch simultaneously, it's much more difficult to determine which of the three makes the biggest contribution to the catch.

**What does it mean**: Multicollinearity increases the standard errors of the coefficients. Increased standard errors in turn means that coefficients for some independent variables may be found not to be significantly different from 0. In other words, by overinflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant. Without multicollinearity (and thus, with lower standard errors), those coefficients might be significant.

If multicollinearity is a problem in your model -- if the VIF for a factor is near or above 5 -- the solution may be relatively simple. Try one of these:

* **Remove highly correlated predictors from the model.** If you have two or more factors with a high VIF, remove one from the model. Because they supply redundant information, removing one of the correlated factors usually doesn't drastically reduce the R-squared.  Consider using stepwise regression, or specialized knowledge of the data set to remove these variables. Select the model that has the highest R-squared value. Also check for Adj. R-Squared.
* **Use Principal Component Analysis**  that cut the number of predictors to a smaller set of uncorrelated components.

**Code:**

*from statsmodels.stats.outliers\_influence import variance\_inflation\_factor*

*def calc\_vif(X):*

*# Calculating VIF*

*vif = pd.DataFrame()*

*vif["variables"] = X.columns*

*vif["VIF"] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]*

*return(vif)*

*…*

*X = df.iloc[:,:-1]*

*calc\_vif(X)*