# Linear Regression Introduction

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It’s used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog). There are two main types:

The reason is because linear regression has been around for so long (more than 200 years). It has been studied from every possible angle and often each angle has a new and different name.

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).

When there is a single input variable (x), the method is referred to as **simple linear regression**. When there are **multiple input variables**, literature from statistics often refers to the method as multiple linear regression.

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called **Ordinary Least Squares**. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression.

Mathematically, we can write a linear relationship as:

https://miro.medium.com/max/221/1*-N98-9lrYGcvUNuRjobFOQ.png

**Where:**

* y is the response
* β values are called the model coefficients. These values are “learned” during the model fitting/training step.
* β0 is the intercept
* β1 is the coefficient for X1 (the first feature)
* βn is the coefficient for Xn (the nth feature)

When training a linear regression model, its trying to find out coefficients for the linear function that best describe the input variables.

## Cost Function

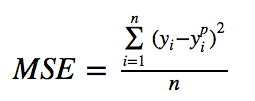
When building a linear model, it’s said that we are trying to minimize the error an algorithm does making predictions, and we got that by choosing a function to help us measure the error also called cost function.

In ML, cost functions are used to estimate how badly models are performing. Put simply**, a cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y.** This is typically expressed as a difference or distance between the predicted value and the actual value. The cost function (you may also see this referred to as loss or error.) can be estimated by iteratively running the model to compare estimated predictions against “ground truth” — the known values of y.

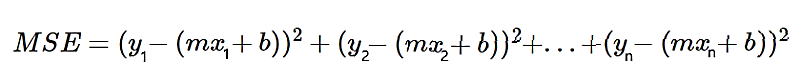
The objective of a ML model, therefore, is to find parameters, weights or a structure that minimizesthe cost function.

### MSE (Mean Square Error, Quadratic loss, L2 Loss)

Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.



From here we get the following equation:



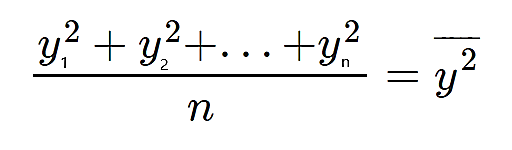
https://cdn-media-1.freecodecamp.org/images/JFi5pzT7YtJ-0Fkx59jP0hCNHzc8tvsrXgPg

https://cdn-media-1.freecodecamp.org/images/vWLTze9HzNDSg4LRM5dbpkYUpkXkhTW6TnRl

We will take all the y, and (-2ymx) and etc, and we will put them all side-by-side.

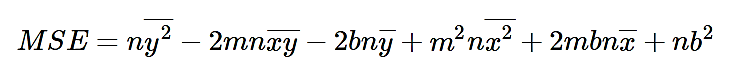
https://cdn-media-1.freecodecamp.org/images/y3gkwSWxwAOcxfxMILLV0teW1273PFtFiqW4

let’s take all the y values, and divide them by n since it’s the mean

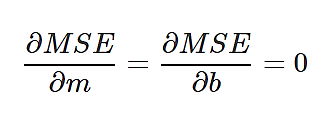


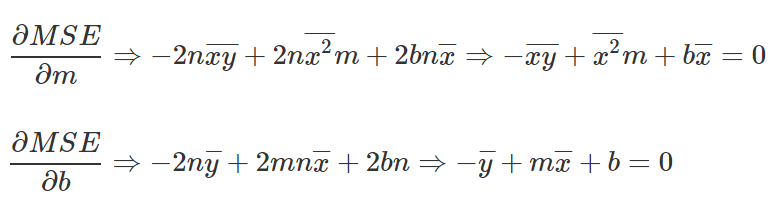
Thus

https://cdn-media-1.freecodecamp.org/images/jyiOt9MVCg460395d6mkHlrmK9ssfr8nQGJC

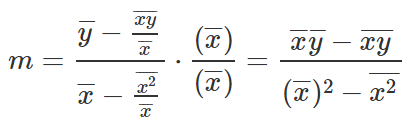


Since we are looking for a minimum point, we will take the partial derivatives and compare to 0.





By solving above 2 equations we will get Final equation to find M.

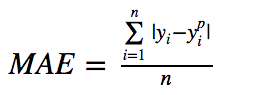


And Final equation to find B.

https://cdn-media-1.freecodecamp.org/images/pjxjeSICBJNckegf3WXCHtfrf7dyIxVfqbBB

### ****Mean Absolute Error, L1 Loss****

[Mean Absolute Error](https://medium.com/@ewuramaminka/mean-absolute-error-mae-sample-calculation-6eed6743838a) (MAE) is another loss function used for regression models. MAE is the sum of absolute differences between our target and predicted variables. So it measures the average magnitude of errors in a set of predictions, without considering their directions. (If we consider directions also, that would be called Mean Bias Error (MBE), which is a sum of residuals/errors). The range is also 0 to ∞.



### RMSE (RMSE)

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.



**In other words,** root mean square error is interpreted as how far on an average, the residuals are from zero. It nullifies squared effect of MSE by square root and provides the result in original units as data.

**In short,** **using the squared error is easier to solve, but using the absolute error is more robust to outliers.** If the outliers represent anomalies that are important for business and should be detected, then we should use MSE. On the other hand, if we believe that the outliers just represent corrupted data, then we should choose MAE as loss.

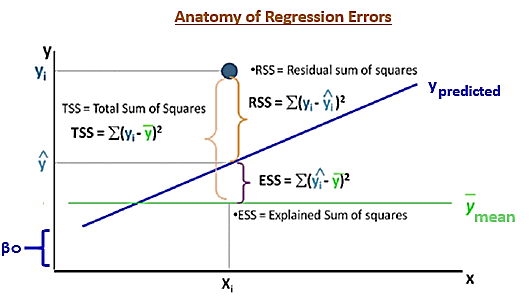
One big problem in using MAE loss (for neural nets especially) is that its gradient is the same throughout, which means the gradient will be large even for small loss values. This isn’t good for learning. To fix this, we can use dynamic learning rate which decreases as we move closer to the minima. MSE behaves nicely in this case and will converge even with a fixed learning rate.

### R-SQUARED

**R-squared** is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

R-square is a comparison of residual sum of squares *(SSres)* with total sum of squares*(SStot)*.

Total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line.



Residual sum of squares in calculated by the summation of squares of perpendicular distance between data points and the best fitted line.

R square is calculated by using the following formula:

https://media.geeksforgeeks.org/wp-content/uploads/20190415232942/CodeCogsEqn-10.gif

Where SSres is the residual sum of squares and SStot is the total sum of squares.

The goodness of fit of regression models can be analyzed on the basis of R-square method. The more the value of r-square near to 1, the better is the model.

**Note:** The value of R-square can also be negative when the models fitted is worse than the average fitted model.

**Limitation of using R-square method –**

The value of r-square always increases or remains same as new variables are added to the model, without detecting the significance of this newly added variable (i.e. value of r-square never decreases on addition of new attributes to the model). As a result, non-significant attributes can also be added to the model with an increase in r-square value.

This is because *SStot* is always constant and regression model tries to decrease the value of *SSres* by finding some correlation with this new attribute and hence the overall value of r-square increases, which can lead to a poor regression model.

### ADJUSTED R-SQUARE

R-square test is used to determine the goodness of fit in regression analysis. Goodness of fit implies how better regression model is fitted to the data points. More is the value of r-square near to 1, better is the model. But the problem lies in the fact that the value of r-square always increases as new variables(attributes) are added to the model, no matter that the newly added attributes have a positive impact on the model or not. also, it can lead to over fitting of the model if there are large no. of variables.

Adjusted r-square is a modified form of r-square whose value increases if new predictors tend to improve model’s performance and decreases if new predictors do not improve performance as expected.

Clearly, SStot is always fixed for some data points if new predictors are added to the model, but value of SSres decreases as model tries to find some correlations from the added predictors. Hence, r-square’s value always increases.

https://media.geeksforgeeks.org/wp-content/uploads/20190623171003/CodeCogsEqn2.png

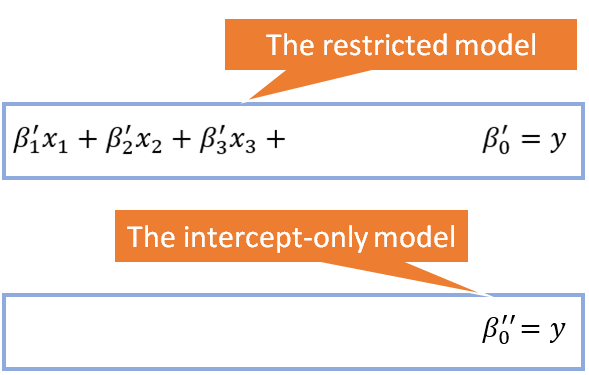
Here, k is the no. of regressors and n is the sample size.  
if the newly added variable is good enough to improve model’s performance, then it will overwhelm the decrease due to k. Otherwise, increase in k will decrease adjusted r-square value.

### F-STATISTICS

F-Statistics test can be used in linear regression analysis for below applications.

* The F-test can be used in regression analysis to determine whether a complex model is better than a simpler version of the same model in explaining the variance in the dependent variable.
* The test statistic of the F-test is a random variable whose **P**robability **D**ensity **F**unction is the F-distribution *under the assumption that the null hypothesis is true.*

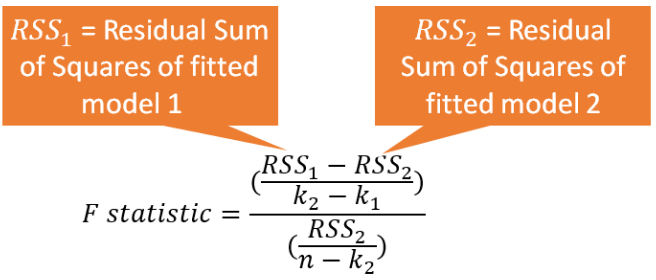
The simpler model is called the restricted model.It is as if we are restricting it to use fewer regression variables. The complex model is called the unrestricted model.It contains all the variables of the restricted model and at least one more variable.



The restricted model is said to be nested within the unrestricted model.

The null hypothesis for F-test is

* **H\_0: The Null hypothesis:**The unrestricted model does not explain the variance in the predicted variable any better than the intercept only model.
* **H\_1: The alternate hypothesis:**The unrestricted model does a better job (in a statistically significant way) of explaining the variance in the predicted variable than the intercept only model.



Formula for the F-statistic when applied to regression analysis

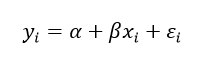
**If the probability of F-test is less than the error threshold such as 0.05, reject the null hypothesis and accept the alternate hypothesis** at a confidence level of (1.0 — error threshold), for e.g. 1–0.05 = 0.95 (i.e. 95% confidence level). Otherwise, accept the null hypothesis with a probability of error equal to the threshold error, for e.g. at 0.05 or 5%.

## Linear Regression Learning the Model

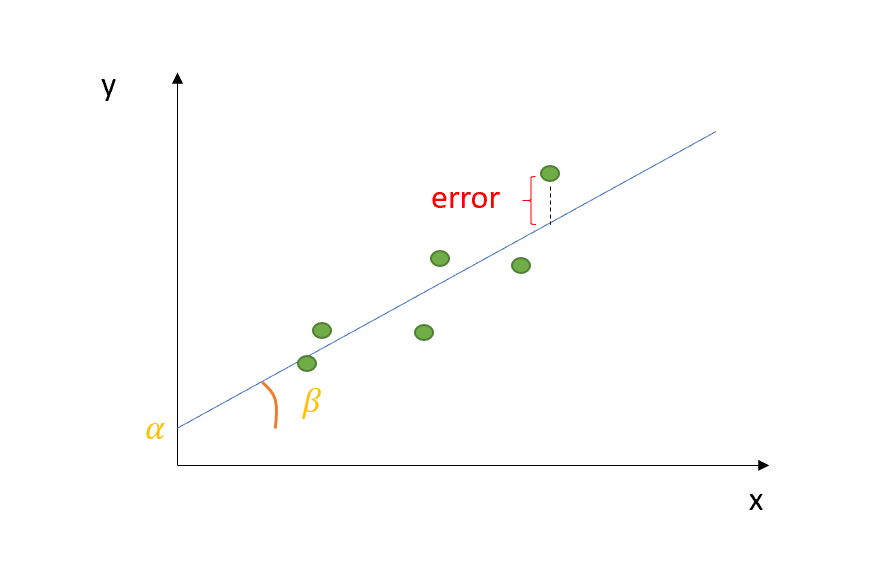
Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available. The objective of a ML model, therefore, is to find parameters, weights or a structure that **minimizes**the cost function.

### Ordinary Least Square

Simple Linear Regression is a statistical model, widely used in ML regression tasks, based on the idea that the relationship between two variables can be explained by the following formula:

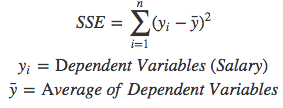


Where εi is the error term, and α, β are the true (but unobserved) parameters of the regression. The parameter β represents the variation of the dependent variable when the independent variable has a unitary variation, the parameter α represents the value of our dependent variable when the independent one is equal to zero.

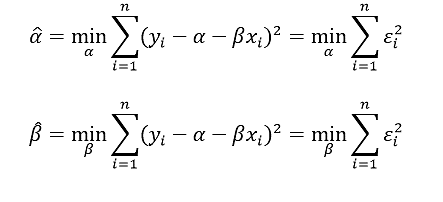


In order to fit the best intercept line between the points in the above scatter plots, we use a metric called “Sum of Squared Errors” (SSE) and compare the lines to find out the best fit by reducing errors. The errors are sum difference between actual value and predicted value.

To find the errors for each dependent value, we need to use the formula below.

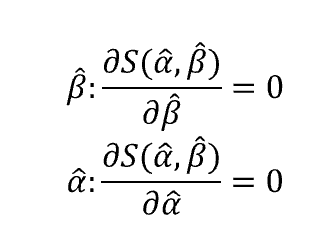


Now, the idea of Simple Linear Regression is finding those parameters α and β for which the error term is minimized. To be more precise, the model will minimize the squared errors: indeed, we do not want our positive errors to be compensated by the negative ones, since they are equally penalizing for our model.

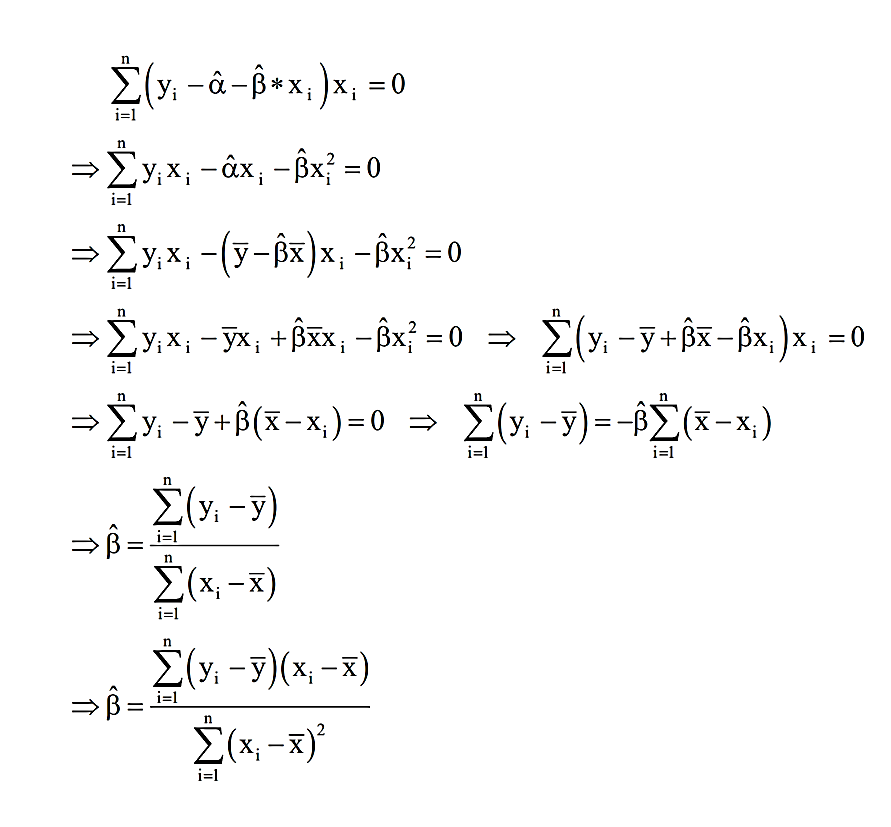


This procedure is called Ordinary Least Squared error — OLS.

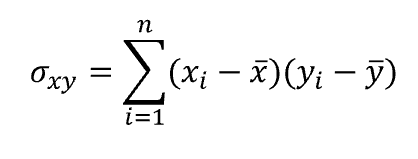
We can set our optimization problem as follows:



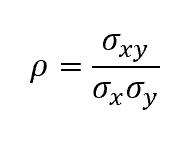
So let’s start with β:



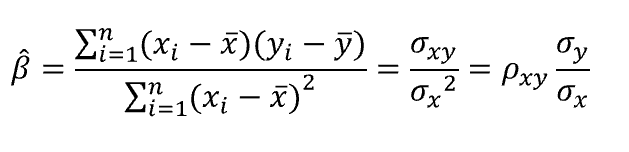
Knowing that the sample covariance between two variables is given by:



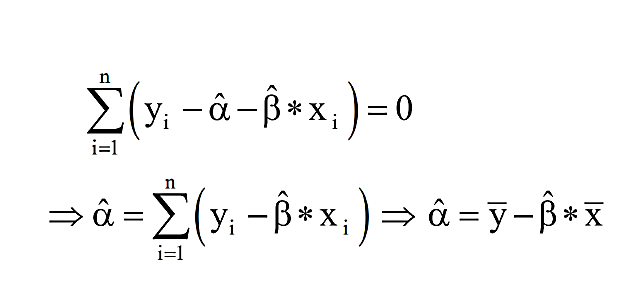
And knowing that the sample correlation coefficient between two variables is equal to:



We can reframe the above expression as follows:



The same reasoning holds for our α:



Once obtained those values of α and β which minimize the squared errors, our model’s equation will look like that:

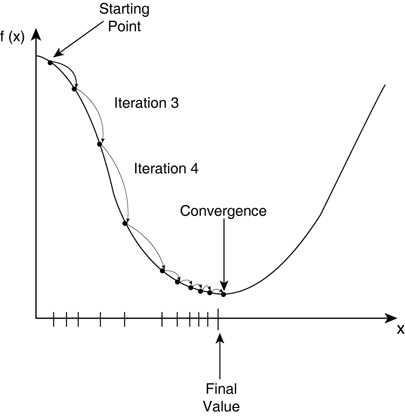
https://miro.medium.com/max/165/1*H56EajsCRnQWfUF0pDn_8Q.png

To sum up, you can consider the OLS as a strategy to obtain, from your model, a ‘straight line’ which is as close as possible to your data points. Even though OLS is not the only optimization strategy, it is the most popular for this kind of tasks, since the outputs of the regression (that are, coefficients) are unbiased estimators of the real values of alpha and beta. Indeed, according to the Gauss-Markov Theorem, under some assumptions of the linear regression model (linearity in parameters, random sampling of observations, conditional mean equal to zero, absence of multicollinearity, homoscedasticity of errors), the OLS estimators α and β are the Best Linear Unbiased Estimators (BLUE) of the real values of α and β​.

### Gradient descent

Gradient descent is an efficient optimization algorithm that attempts to find a local or global minima of a function.

Gradient descent enables a model to learn the gradient or *direction* that the model should take in order to reduce errors (differences between actual y and predicted y). Direction in the simple linear regression example refers to how the model parameters b0 and b1 should be tweaked or corrected to further reduce the cost function. As the model iterates, it gradually converges towards a minimum where further tweaks to the parameters produce little or zero changes in the loss — also referred to as convergence.



At this point the model has optimized the weights such that they minimize the cost function. This process is integral (no calculus pun intended!) to the ML process, because it greatly expedites the learning process — you can think of it as a means of receiving corrective feedback on how to improve upon your previous performance. The alternative to the gradient descent process would be brute forcing a potentially infinite combination of parameters until the set that minimizes the cost are identified. For obvious reasons this isn’t really feasible. Gradient descent, therefore**,**enables the learning process to make corrective updates to the learned estimates that move the model toward an optimal combination of parameters.

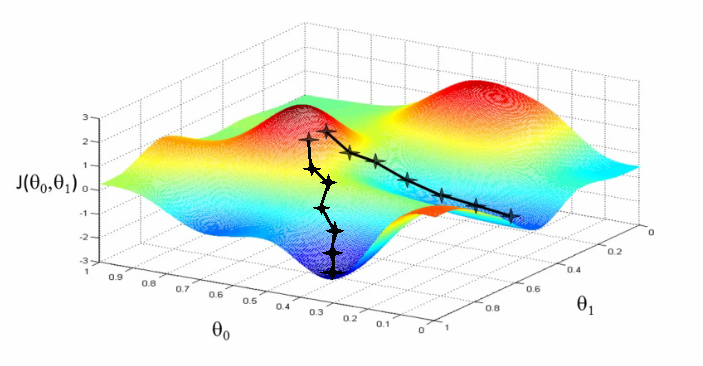
#### **Algorithm**

Let’s represent the hypothesis h, which is function or a learning algorithm.

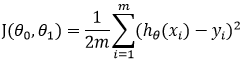
https://miro.medium.com/max/200/1*K1-0bnoMqxSv6tZXpOki1Q.png

The goal is similar like the above operation that we did to find out a best fit of intercept line ‘y’ in the slope ‘m’. Using Gradient descent algorithm also, we will figure out a minimal cost function by applying various parameters for theta 0 and theta 1 and see the slope intercept until it reaches convergence.

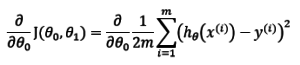
In a real world example, it is similar to find out a best direction to take a step downhill.



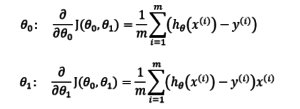
We take a step towards the direction to get down. From each step, you look out the direction again to get down faster and downhill quickly. The similar approach is using in this algorithm to minimize cost function.

We can measure the accuracy of our hypothesis function by using a cost function and the formula is

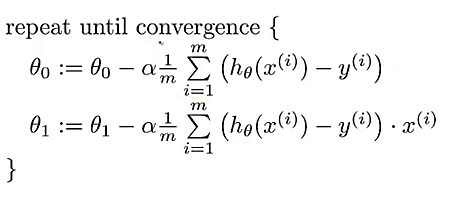
Gradient Descent for Linear Regression

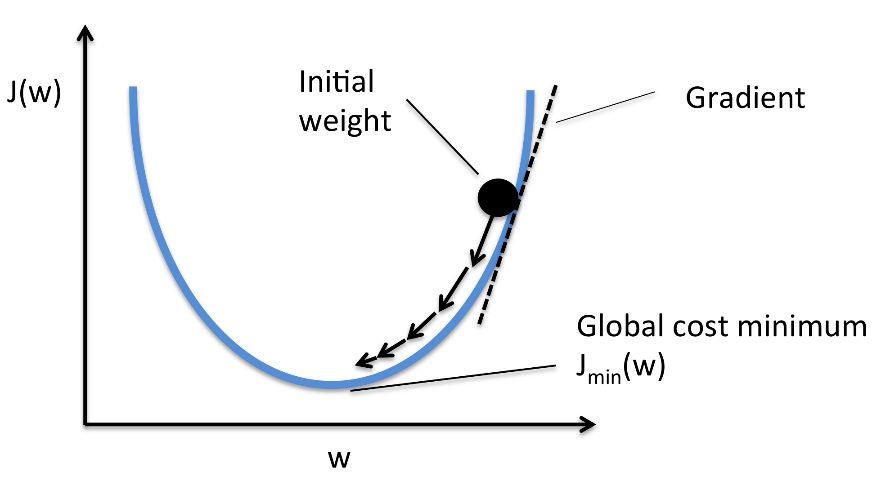


Why do we use partial derivative in the equation? Partial derivatives represent the rate of change of the functions as the variable change. In our case we change values for theta 0 and theta 1 and identifies the rate of change. To apply rate of change values for theta 0 and theta 1, the below are the equations for theta 0 and theta 1 to apply it on each epoch.



To find the best minimum, repeat steps to apply various values for theta 0 and theta 1. In other words, repeat steps until convergence.





* where alpha (a) is a learning rate / how big a step takes to downhill.

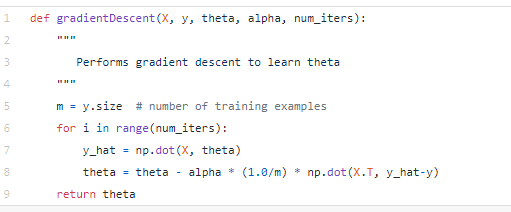
Types of Gradient Descent Algorithms

There are three types of Gradient Descent Algorithms:

1. Batch Gradient Descent  
2. Stochastic Gradient Descent  
3. Mini-Batch Gradient Descent

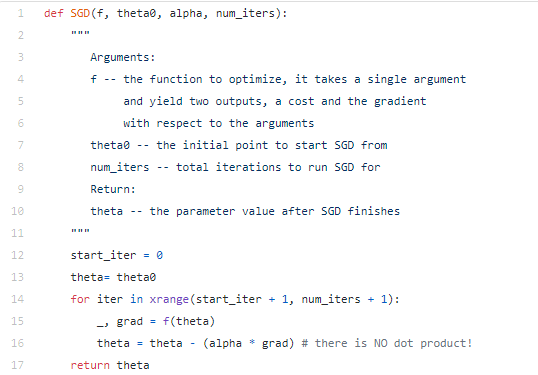
#### **Batch Gradient Descent**

* In the batch gradient descent, to calculate the gradient of the cost function, we need to sum all training examples for each steps
* If we have 3 million samples (m training examples) then the gradient descent algorithm should sum 3 million samples for every epoch. To move a single step, we have to calculate each with 3 million times!
* Batch Gradient Descent is not good fit for large datasets
* Below is python code implementation for Batch Gradient Descent algorithm.



#### **Stochastic Gradient Descent (SGD)**

* In stochastic Gradient Descent, we use one example or one training sample at each iteration instead of using whole dataset to sum all for every steps
* SGD is widely used for larger dataset trainings and computationally faster and can be trained in parallel
* Need to randomly shuffle the training examples before calculating it
* Python code implementation for SGD in below



#### **Mini-Batch Gradient Descent**

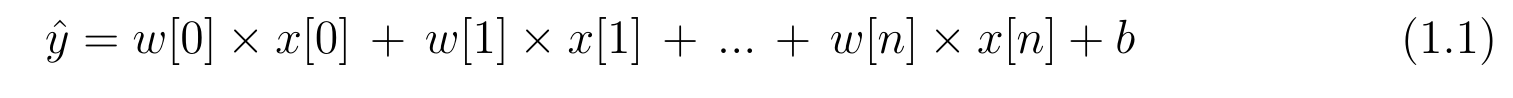
It is similar like SGD; it uses ***n***samples instead of 1 at each iteration.

### Regularization

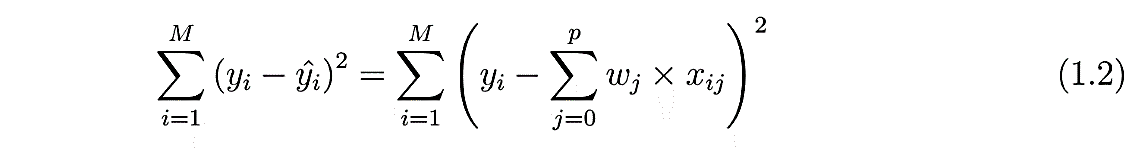
One of the major aspects of training your machine learning model is avoiding overfitting. The model will have a low accuracy if it is overfitting. This happens because your model is trying too hard to capture the noise in your training dataset. By noise we mean the data points that don’t really represent the true properties of your data, but random chance. Learning such data points, makes your model more flexible, at the risk of overfitting.

The concept of balancing bias and variance, is helpful in understanding the phenomenon of overfitting.

first let’s clarify that linear models are one of the simplest way to predict output using a linear function of input features.



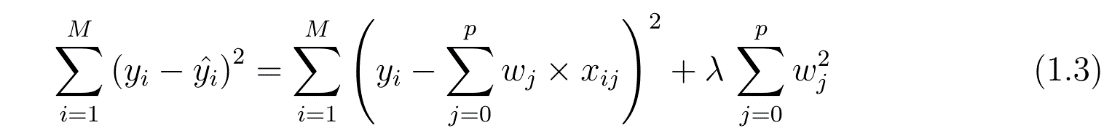
In the equation (1.1) above, we have shown the linear model based on the n number of features. Considering only a single feature as you probably already have understood that *w[0]* will be slope and *b*will represent intercept. Linear regression looks for optimizing *w* and *b* such that it minimizes the cost function. The cost function can be written as



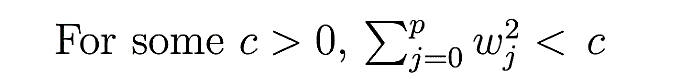
In the equation above the data-set has M instances and p features. Once we use linear regression on a data-set divided in to training and test set, calculating the scores on training and test set can give us a rough idea about whether the model is suffering from over-fitting or under-fitting. If we have very few features on a data-set and the score is poor for both training and test set, then it’s a problem of under-fitting. On the other hand, if we have large number of features and test score is relatively poor than the training score then it’s the problem of over-generalization or over-fitting. Ridge and Lasso regression are some of the simple techniques to reduce model complexity and prevent over-fitting which may result from simple linear regression.

#### **Ridge Regression**

In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients.



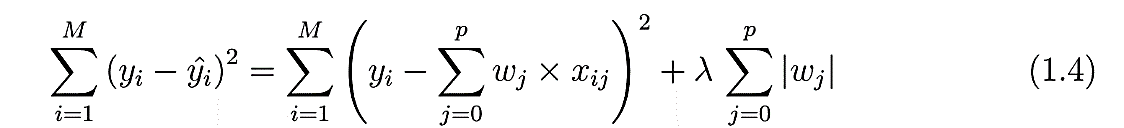
This is equivalent to saying minimizing the cost function in equation 1.2 under the condition as below

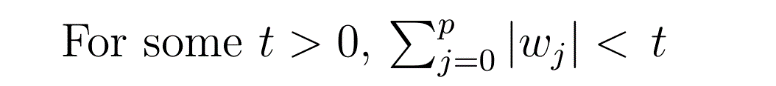


So ridge regression puts constraint on the coefficients (w). The penalty term (lambda) regularizes the coefficients such that if the coefficients take large values the optimization function is penalized. So, **ridge regression shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity**. Going back to eq. 1.3 one can see that when λ → 0, the cost function becomes similar to the linear regression cost function (eq. 1.2). So lower the constraint (low λ) on the features, the model will resemble linear regression model. Let’s see an example using Boston house data and below is the code I used to depict linear regression as a limiting case of Ridge regression.

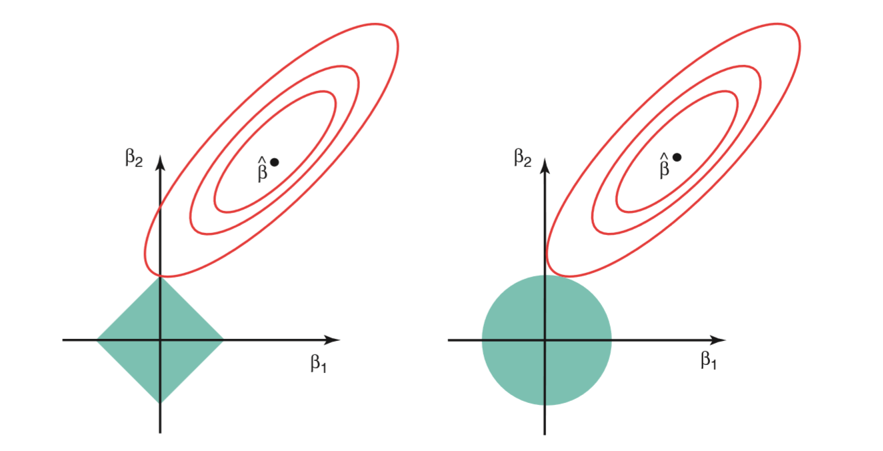
#### **Lasso Regression**

The cost function for Lasso (least absolute shrinkage and selection operator) regression can be written as





Just like Ridge regression cost function, for lambda =0, the equation above reduces to equation 1.2. The only difference is instead of taking the square of the coefficients, magnitudes are taken into account. This type of regularization (L1) can lead to zero coefficients i.e. some of the features are completely neglected for the evaluation of output. So Lasso regression not only helps in reducing over-fitting but it can help us in feature selection. So feature selection using Lasso regression can be depicted well by changing the regularization parameter.



The above image shows the constraint functions (green areas), for lasso(left) and ridge regression(right), along with contours for RSS (red ellipse). Points on the ellipse share the value of RSS. For a very large value of s, the green regions will contain the center of the ellipse, making coefficient estimates of both regression techniques, equal to the least squares estimates. But, this is not the case in the above image. In this case, the lasso and ridge regression coefficient estimates are given by the ﬁrst point at which an ellipse contacts the constraint region. Since ridge regression has a circular constraint with no sharp points, this intersection will not generally occur on an axis, and so the ridge regression coeﬃcient estimates will be exclusively non-zero. However, the lasso constraint has corners at each of the axes, and so the ellipse will often intersect the constraint region at an axis. When this occurs, one of the coeﬃcients will equal zero. In higher dimensions (where parameters are much more than 2), many of the coeﬃcient estimates may equal zero simultaneously.

This sheds light on the obvious disadvantage of ridge regression, which is model interpretability. It will shrink the coefficients for least important predictors, very close to zero. But it will never make them exactly zero. In other words, the final model will include all predictors. However, in the case of the lasso, the L1 penalty has the eﬀect of forcing some of the coeﬃcient estimates to be exactly equal to zero when the tuning parameter λ is suﬃciently large. Therefore, the lasso method also performs variable selection and is said to yield sparse models.

A standard least squares model tends to have some variance in it, i.e. this model won’t generalize well for a data set different than its training data. Regularization, significantly reduces the variance of the model, without substantial increase in its bias. So the tuning parameter λ, used in the regularization techniques described above, controls the impact on bias and variance. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance. Till a point, this increase in λ is beneficial as it is only reducing the variance (hence avoiding overfitting), without losing any important properties in the data. But after certain value, the model starts loosing important properties, giving rise to bias in the model and thus under fitting. Therefore, the value of λ should be carefully selected.

This is all the basic you will need, to get started with Regularization. It is a useful technique that can help in improving the accuracy of your regression models.

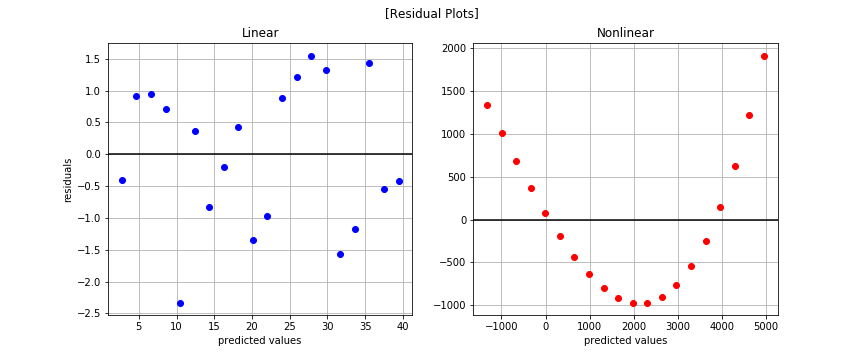
# Assumptions & Evaluation

1. Linear relationship between target and features
2. Homoscedasticity of error terms
3. Uncorrelated error terms / No Auto correlation
4. Independent features / No multi collinearity
5. Multivariate normality / Normality of error terms

## Linear Relationship Between Target & Features

The dependent variable (y) is assumed to be a linear function of the independent variables (X, features) specified in the model. The specification must be linear in its parameters. Fitting a linear model to data with non-linear patterns results in serious prediction errors, especially out-of-sample (data not used for training the model).

To detect nonlinearity one can inspect plots of observed vs. predicted values or residuals vs. predicted values. The desired outcome is that points are symmetrically distributed around a diagonal line in the former plot or around a horizontal line in the latter one. In both cases with a roughly constant variance.

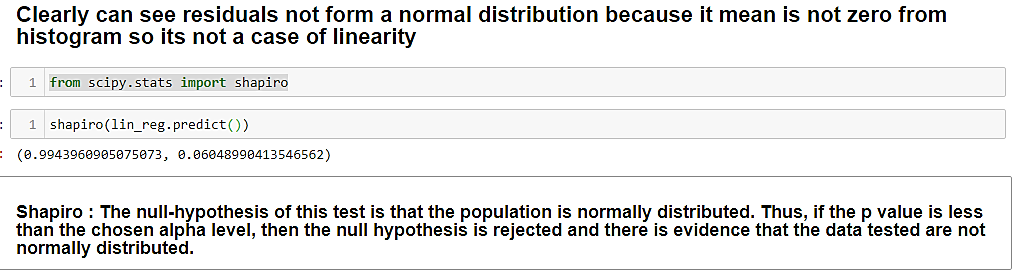
Observing a ‘bowed’ pattern indicates that the model makes systematic errors whenever it is making unusually large or small predictions. When the model contains many features, nonlinearity can also be revealed by systematic patterns in plots of the residuals vs. individual features.

The nonlinear pattern is overwhelmingly obvious in the residual plots. You may be wondering why we bothered plotting at all since we saw the nonlinear trend when plotting the observed data.

|  |  |
| --- | --- |
| image | image |

The histogram of the linear model on linear data looks approximately Normal (aka Gaussian) while the second histogram shows a skew. But is there a more quantitative method to test for Normality? Absolutely. SciPy has a normaltest method. Let’s see it in action.

**Shapiro (from scipy.stats import shapiro):** The **null**-**hypothesis** of this test is that the population is normally distributed. Thus, if the p value is less than the chosen alpha level, then the **null hypothesis** is rejected and there is evidence that the data tested are not normally distributed.



**Potential solutions:**

* non-linear transformations to dependent/independent variables
* adding extra features which are a transformation of the already used ones (for example squared version)
* Consider applying a different algorithm

## Homoscedasticity of Error Terms

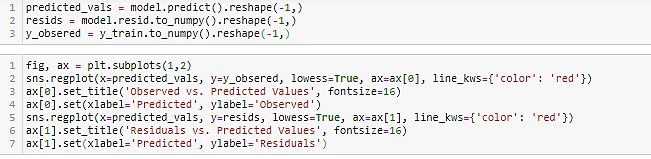
This assumption means that the variance around the regression line is the same for all values of the predictor variable (X). The plot shows a violation of this assumption. For the lower values on the X-axis, the points are all very near the regression line. For the higher values on the X-axis, there is much more variability around the regression line.

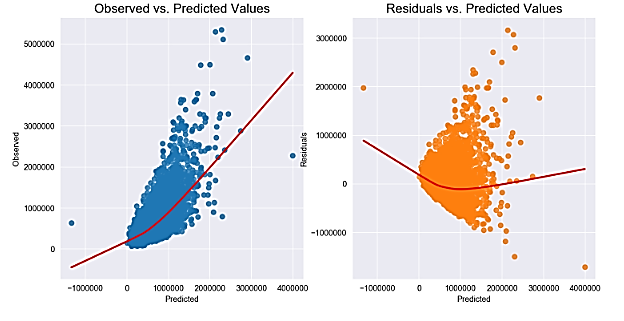
When residuals do not have constant variance (they exhibit heteroscedasticity), it is difficult to determine the true standard deviation of the forecast errors, usually resulting in confidence intervals that are too wide/narrow. For example, if the variance of the residuals is increasing over time, confidence intervals for out-of-sample predictions will be unrealistically narrow. Another effect of heteroscedasticity might also be putting too much weight to a subset of data when estimating coefficients — the subset in which the error variance was largest.

To investigate if the residuals are homoscedastic, we can look at a plot of residuals (or standardized residuals) vs. predicted (fitted) values. What should alarm us is the case when the residuals grow either as a function of predicted value or time (in case of time series). To identify homoscedasticity in the plots, the placement of the points should be random and no pattern (increase/decrease in values of residuals) should be visible.

|  |  |
| --- | --- |
| http://www.statisticssolutions.com/wp-content/uploads/2010/01/linearregression06.jpg | http://www.statisticssolutions.com/wp-content/uploads/2010/01/linearregression07.jpg |

We can also use two statistical tests: **Breusch-Pagan** and **Goldfeld-Quandt.** In both of them, the null hypothesis assumes homoscedasticity and a p-value below a certain level (like 0.05) indicates we should reject the null in favor of heteroscedasticity.



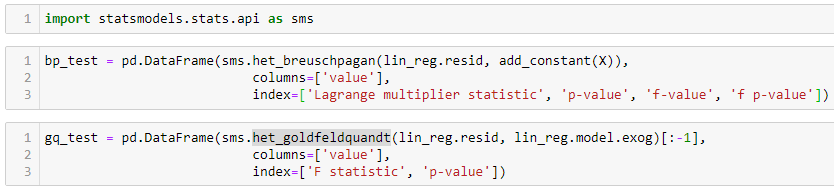


The **Breusch-Pagan-Godfrey Test** (sometimes shorted to the Breusch-Pagan test) is a test for heteroscedasticity of errors in regression. Heteroscedasticity means “differently scattered”; this is opposite to homoscedastic, which means “same scatter.” Homoscedasticity in regression is an important assumption; if the assumption is violated, you won’t be able to use regression analysis.

* The tests the hypothesis that the residual variance does not depend on the variables in x in the form
* If you find p < 0.05, you reject the null hypothesis and infer that heteroscedasticity is present.

**Goldfeld–Quandt** test checks for homoscedasticity in regression analyses. It does this by dividing a dataset into two parts or groups, and hence the test is sometimes called a two-group test.

It assumes that heteroscedastic variance σi2 is positively related to one of the explanatory variables and errors are assumed to be normal. Thus if heteroscedasticity is present then the variance would be high for large values of X.



|  |  |
| --- | --- |
|  |  |

The results indicate that the assumption is not satisfied and we should reject the hypothesis of homoscedasticity.

Potential solutions:

* log transformation of the dependent variable
* in case of time series, deflating a series if it concerns monetary value
* using ARCH (auto-regressive conditional heteroscedasticity) models to model the error variance. An example might be stock market, where data can exhibit periods of increased or decreased volatility over time (volatility clustering, see [this article](https://towardsdatascience.com/introduction-to-quantitative-finance-part-i-stylised-facts-of-asset-returns-5190581e40ea) for more information)

## Uncorrelated error terms / No Auto correlation

Linear regression analysis requires that there is little or no autocorrelation in the data. Autocorrelation occurs when the residuals are not independent from each other. In other words, when the value of y(x+1) is not independent from the value of y(x).

Autocorrelation occurs when the residuals are not independent from each other.  For instance, this typically occurs in stock prices, where the price is not independent from the previous price.

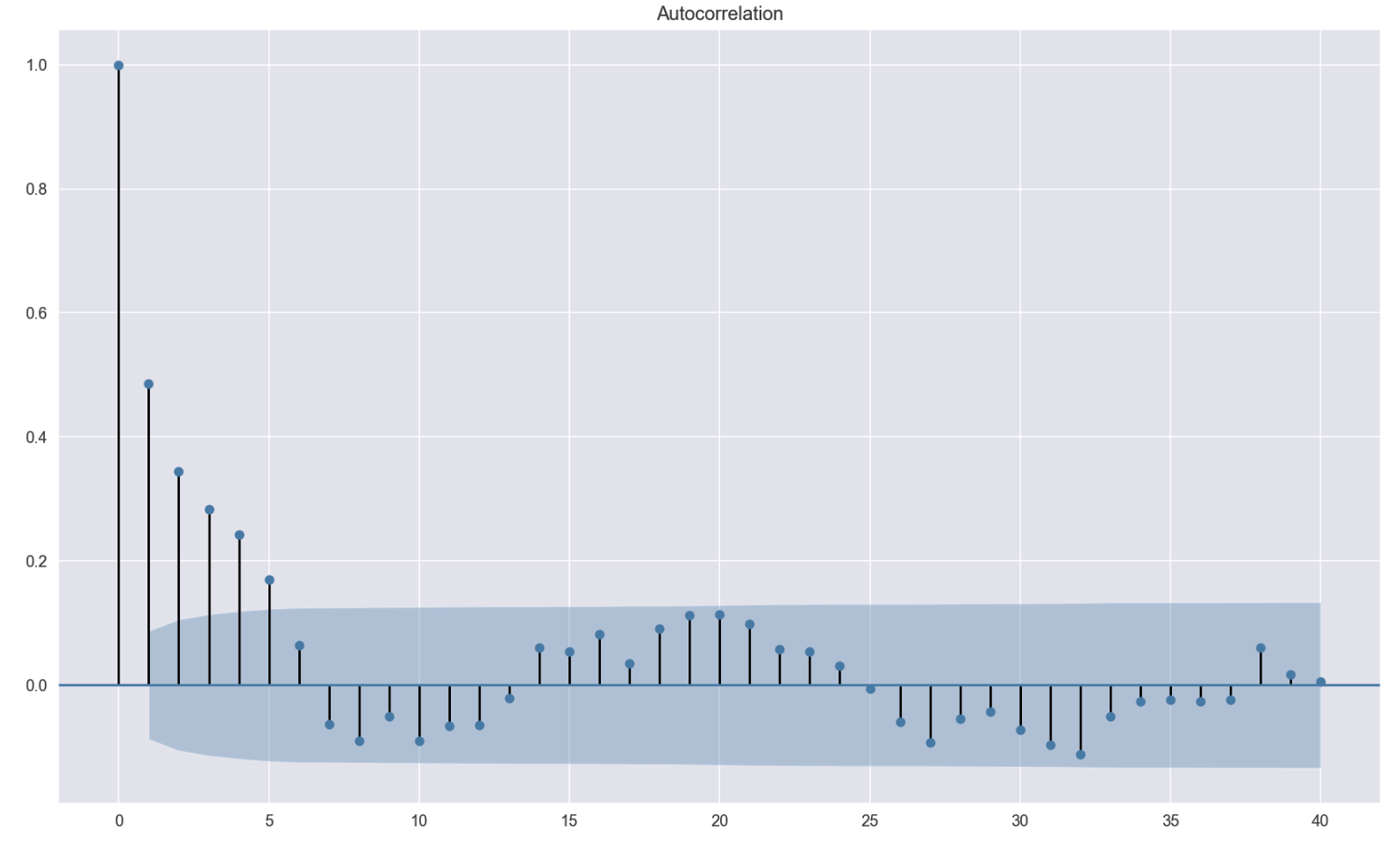
This assumption is especially dangerous in time-series models, where serial correlation in the residuals implies that there is room for improvement in the model. Extreme serial correlation is often a sign of a badly miss specified model. Another reason for serial correlation in the residuals could be a violation of the linearity assumption or due to bias that is explainable by omitted variables (interaction terms or dummy variables for identifiable conditions). An example of the former case might be fitting a (straight) line to data, which exhibits exponential growth over time.

This assumption also has meaning in the case of non-time-series models. If residuals always have the same sign under particular conditions, it means that the model systematically under predicts/over predicts what happens when the predictors have a particular configuration.

To investigate if autocorrelation is present, we can use **ACF (autocorrelation function) plots** and **Durbin-Watson test.**

In the former case, we want to see if the value of ACF is significant for any lag (in case of no time-series data, the row number is used). While calling the function, we indicate the significance level, we are interested in and the critical area is plotted on the graph. Significant correlations lie outside of that area.

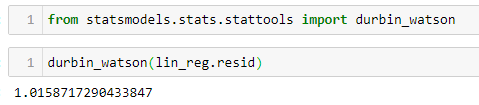
Note: when dealing with data without the time dimension, we can alternatively plot the residuals vs. the row number. In such cases, rows should be sorted in a way that (only) depends on the values of the feature(s).



The second approach is using the Durbin-Watson test. The test statistic provides a test for significant residual autocorrelation at lag 1. The DW statistic is approximately equal to 2(1-a), where a is the lag 1 residual autocorrelation. The DW test statistic is located in the default summary output of statsmodels’s regression.

Some notes on the Durbin-Watson test:

* the test statistic always has a value between 0 and 4
* value of 2 means that there is no autocorrelation in the sample
* values < 2 indicate positive autocorrelation, values > 2 negative one.



Potential solutions:

* in case of minor positive autocorrelation, there might be some room for fine-tuning the model, for example, adding lags of the dependent/independent variables
* some seasonal components might not be captured by the model, account for them using dummy variables or seasonally adjust the variables
* if DW < 1 it might indicate a possible problem in model specification, consider stationarizing time-series variables by differencing, logging, and/or deflating (in case of monetary values)
* in case of significant negative correlation, some of the variables might have been overdifferenced
* use Generalized Least Squares
* include a linear (trend) term in case of a consistent increasing/decreasing pattern in the residuals

## Independent features / No multi collinearity

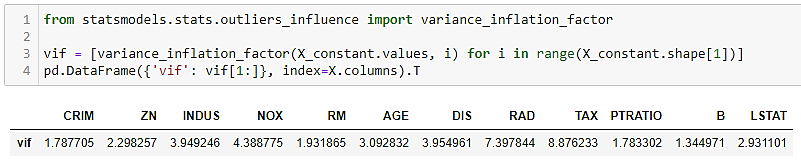
Linear regression assumes that there is little or no multicollinearity in the data.  Multicollinearity occurs when the independent variables are too highly correlated with each other.

One scenario to watch out for is the ‘dummy variable trap’ — when we use dummy variables to encode a categorical feature and do not omit the baseline level from the model. This results in a perfect correlation between the dummy variables and the constant term.

Multicollinearity can be present in the model, as long as it is not ‘perfect’. In the former case, the estimates are less efficient but still unbiased. The estimates will be less precise and highly sensitive to particular sets of data.

Multicollinearity may be tested with three central criteria:

1. Correlation matrix – when computing the matrix of Pearson’s Bivariate Correlation among all independent variables the correlation coefficients need to be smaller than 1.
2. Tolerance – the tolerance measures the influence of one independent variable on all other independent variables; the tolerance is calculated with an initial linear regression analysis.  Tolerance is defined as T = 1 – R² for these first step regression analysis.  With T < 0.1 there might be multicollinearity in the data and with T < 0.01 there certainly is.
3. Variance Inflation Factor (VIF) – the variance inflation factor of the linear regression is defined as VIF = 1/T. With VIF > 5 there is an indication that multicollinearity may be present; with VIF > 10 there is certainly multicollinearity among the variables.

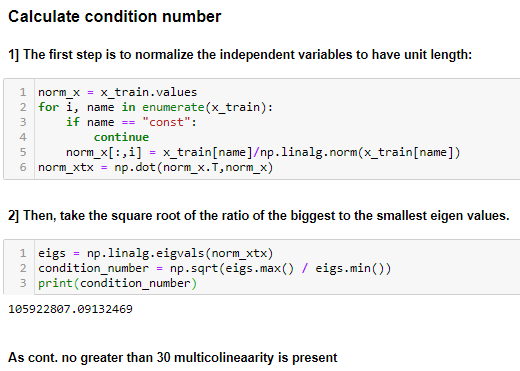


We can detect multicollinearity using the variance inflation factor (VIF). Without going into too many details, the interpretation of VIF is as follows: the square root of a given variable’s VIF shows how much larger the standard error is, compared with what it would be if that predictor was uncorrelated with the other features in the model. If no features are correlated, then all values for VIF will be 1.

To deal with multicollinearity we should iteratively remove features with high values of VIF. A rule of thumb for removal could be VIF larger than 10 (5 is also common). Another possible solution is to use PCA to reduce features to a smaller set of uncorrelated components.

Tip: we can also look at the correlation matrix of features to identify dependencies between them.

Condition number from OLS regression summary can take for multicolinearity, if cont. no > 30 means multicolinearity is present.



## Multivariate normality / Normality of Residuals

When this assumption is violated, it causes problems with calculating confidence intervals and various significance tests for coefficients. When the error distribution significantly departs from Gaussian, confidence intervals may be too wide or too narrow.

Some of the potential reasons causing non-normal residuals:

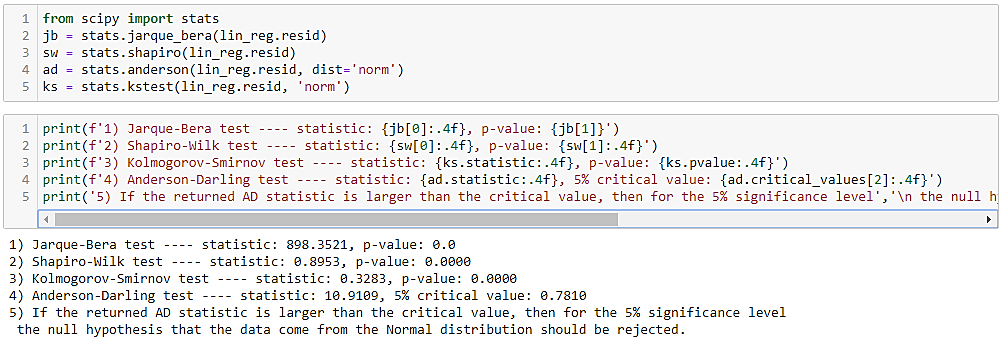
* presence of a few large outliers in data
* there might be some other problems (violations) with the model assumptions
* another, better model specification might be better suited for this problem

Technically, we can omit this assumption if we assume instead that the model equation is correct and our goal is to estimate the coefficients and generate predictions (in the sense of minimizing mean squared error).

However, normally we are interested in making valid inferences from the model or estimating the probability that a given prediction error will exceed some threshold in a particular direction. To do so, the assumption about the normality of residuals must be satisfied.

To investigate this assumption, we can look at:

* QQ plots of the residuals. For example, a bow-shaped pattern of deviations from the diagonal implies that the residuals have excessive skewness (i.e., the distribution is not symmetrical, with too many large residuals in one direction). The s-shaped pattern of deviations implies excessive kurtosis of the residuals — there are either too many or two few large errors in both directions.
* use statistical tests such as the Kolmogorov-Smirnov test, the Shapiro-Wilk test, the Jarque-Bera test, and the Anderson-Darling test



From the results above we can infer that the residuals do not follow Gaussian distribution — from the shape of the QQ plot, as well as rejecting the null hypothesis in all statistical tests. The reason why Kolmogorov-Smirnov from ols\_test\_normality shows different results is that it does not run the `two-sided` version of the test.

Normality depends on sample size. The Jarque–Bera test is comparing the shape of a given distribution (skewness and kurtosis) to that of a Normal distribution. like other Normality tests, as sample size increases you have a higher change of falling below the typical alpha value (i.e., 0.05)

If the test is being used to decide if the data will likely meet the assumptions of parametric tests, then one needs to consider that the central limit theorem usually renders the data "...sufficiently normal for the underlying assumption of normality to be reasonable for the practical purpose of the analysis.

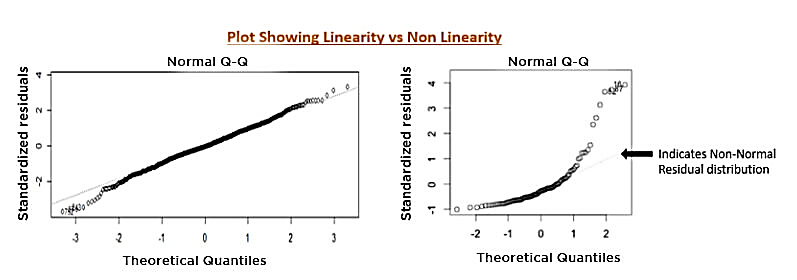
As a reminder, checks for Normality should be checked on the residuals of a model, because those assumptions apply to the *unexplained*variance of a model. Also note that the assumptions of independence, linearity, and heteroscedasticity can have more influence on the reliability of test than distribution assumptions, though highly skewed data can be just as problematic.

Null hypothesis for JB test says data is normally distributed, if p value <0.05 can reject the null hypothesis, but again these tests are highly prone to sample size.

Residual plots are better suited.

Normality Q-Q Plot

As the name suggests, this plot is used to determine the normal distribution of errors. It uses standardized values of residuals. Ideally, this plot should show a straight line. If you find a curved, distorted line, then your residuals have a non-normal distribution (problematic situation).

[](https://blog-c7ff.kxcdn.com/blog/wp-content/uploads/2016/12/boll.jpg)

Potential solutions:

* nonlinear transformation of target variable or features
* remove/treat potential outliers
* it can happen that there are two or more subsets of the data having different statistical properties, in which case separate models might be considered

# Simple Linear Regression

A model is a transformation engine that helps us to express dependent variables as a function of independent variables. independent variable's data that can be controlled directly, Dependent variable's data that cannot be controlled directly.

simple linear regression is a linear regression model with a single explanatory variable. That is, it concerns two-dimensional sample points with one independent variable and one dependent variable (conventionally, the x and y coordinates in a Cartesian coordinate system) and finds a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variables. The adjective simple refers to the fact that the outcome variable is related to a single predictor.

The robustness of the model is evaluated using hypothesis testing.

H0 and Ha need to be defined. They are defined as follows:

* H0 (NULL hypothesis): There is no relationship between x and y.
* Ha (Alternate hypothesis): There is some relationship between x and y.

**β1:** The value of β1 determines the relationship between price and engine size. If β1 = 0 then there is no relationship. In this case, β1 is positive.

**t-stat:** The t-stat value is how many standard deviations the coefficient estimate (β1) is far away from zero. The coefficient is significant for example t-stat is 21.09. It is far enough from zero.

**p-value:** p-value is a probability value. It indicates the chance of seeing the given t-statistics, under the assumption that NULL hypothesis is true. If the p-value is small e.g. < 0.0001, it implies that the probability that this is by chance and there is no relation is very low.

# Multiple Linear Regression

Multiple Linear Regression attempts to model the Relationship between two or more features and a response by fitting a linear equation to observed data. The steps to perform multiple linear Regression are almost similar to that of simple linear Regression. The Difference Lies in the Evolution. We can use it to find out which factor has the highest impact on the predicted output and now different variable relate to each other.

**Y= b0 + b1\*x1 + b2\*x2 + b3\*x3 +…… bn\*xn**  
Y = Dependent variable and x1, x2, x3, …… xn = multiple independent variables

### Steps Involved in any Multiple Linear Regression Model

**Step #1:** Data Pre Processing

a) Importing The Libraries.  
b) Importing the Data Set.  
c) Encoding the Categorical Data.  
d) Avoiding the Dummy Variable Trap.  
e) Splitting the Data set into Training Set and Test Set.

**Step #2:** Fitting Multiple Linear Regression to the Training set  
**Step #3:** Predicting the Test set results.

# Polynomial Regression

**Polynomial Regression**is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as a *nth*degree polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y |x)

### Why Polynomial Regression:

* There are some relationships that a researcher will hypothesize is curvilinear. Clearly, such type of cases will include a polynomial term.
* Inspection of residuals. If we try to fit a linear model to curved data, a scatter plot of residuals (Y axis) on the predictor (X axis) will have patches of many positive residuals in the middle. Hence in such situation it is not appropriate.
* An assumption in usual multiple linear regression analysis is that all the independent variables are independent. In polynomial regression model, this assumption is not satisfied.

**Uses of Polynomial Regression:**  
These are basically used to define or describe non-linear phenomenon such as:

* Growth rate of tissues.
* Progression of disease epidemics
* Distribution of carbon isotopes in lake sediments

The basic goal of regression analysis is to model the expected value of a dependent variable y in terms of the value of an independent variable x. In simple regression, we used following equation –

**y** = a + bx + e

Here y is dependent variable, a is y intercept, b is the slope and e is the error rate.

In many cases, this linear model will not work out for example if we analyzing the production of chemical synthesis in terms of temperature at which the synthesis take place in such cases we use quadratic model

**y** = a + b1x + b2^2 + e

Here y is dependent variable on x, a is y intercept and e is the error rate.

In general, we can model it for nth value.

**y** = a + b1x + b2x^2 +....+ bnx^n

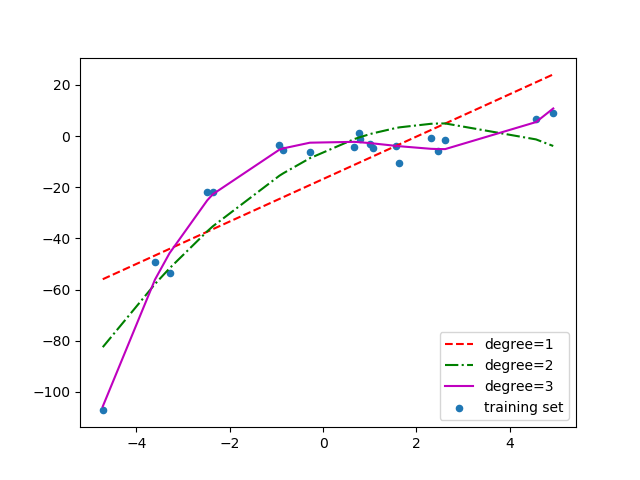
Since regression function is linear in terms of unknown variables, hence these models are linear from the point of estimation.

Hence through Least Square technique, let’s compute the response value that is y.

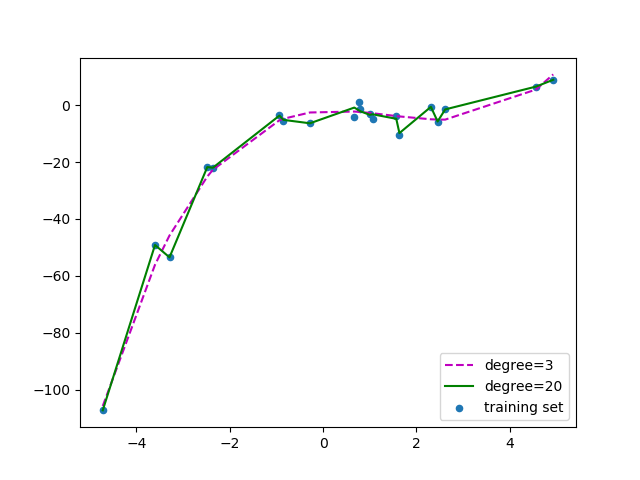
Polynomial regression is implemented using sklearn library is given below.

|  |
| --- |
|  |
|  |

Below is a comparison of fitting linear, quadratic and cubic curves on the dataset.



If we further increase the degree to 20, we can see that the curve passes through more data points. Below is a comparison of curves for degree 3 and 20.



For degree=20, the model is also capturing the noise in the data. This is an example of **over-fitting.** Even though this model passes through most of the data, it will fail to generalize on unseen data.

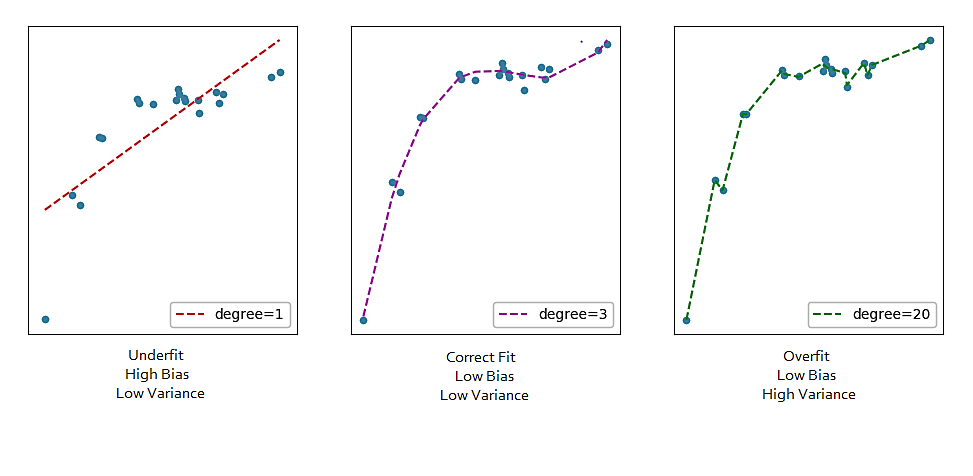
To prevent over-fitting, we can add more training samples so that the algorithm doesn’t learn the noise in the system and can become more generalized***.****(Note: adding more data can be an issue if the data is itself noise).*

### The BiasVs Variance trade-off

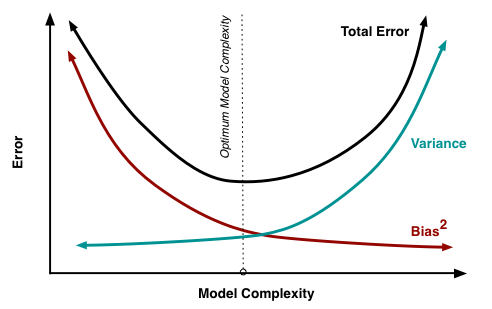
**Bias**refers to the error due to the model’s simplistic assumptions in fitting the data. A high bias means that the model is unable to capture the patterns in the data and this results in **under-fitting**.

**Variance**refers to the error due to the complex model trying to fit the data. High variance means the model passes through most of the data points and it results in **over-fitting** the data.

The below picture summarizes our learning.



From the below picture we can observe that as the model complexity increases, the bias decreases and the variance increases and vice-versa. Ideally, a machine learning model should **have low variance and low bias**. But practically it’s impossible to have both. Therefore, to achieve a good model that performs well both on the train and unseen data, a **trade-off** is made.



#### **How to detect a high bias problem?**

If two curves are “close to each0 other” and both of them but have a low score. The model suffers from an under fitting problem (High Bias). A high bias problem has the following characteristics

1. High training error.
2. Validation error is similar in magnitude to the training error.

#### **How to detect a high variance problem?**

If training curve has a much better score but testing curve has a lower score, i.e., there are large gaps between two curves. Then the model suffers from an over fitting problem (High Variance). A high variance problem on the other hand has the following characteristics

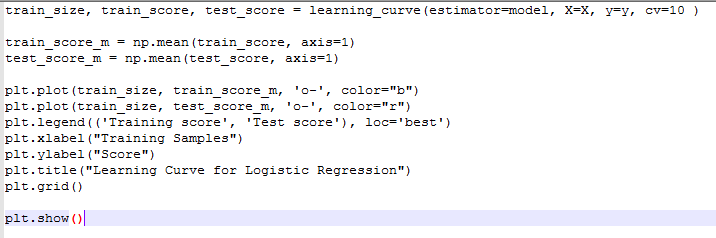
1. Low training error
2. Very high Validation error

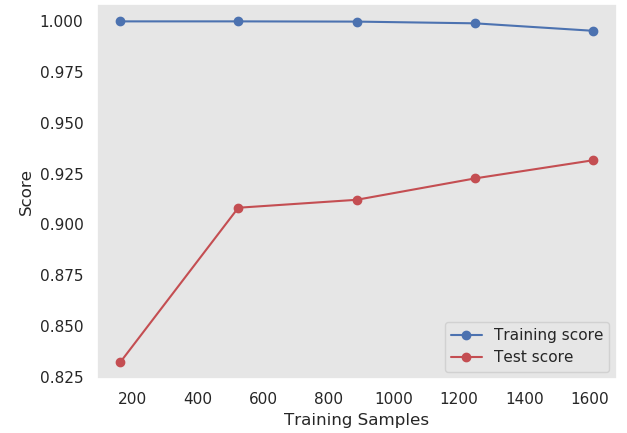
#### **Learning Curve**

A learning curve shows the relationship of the training score vs the cross validated test score for an estimator with a varying number of training samples. This visualization is typically used two show two things:

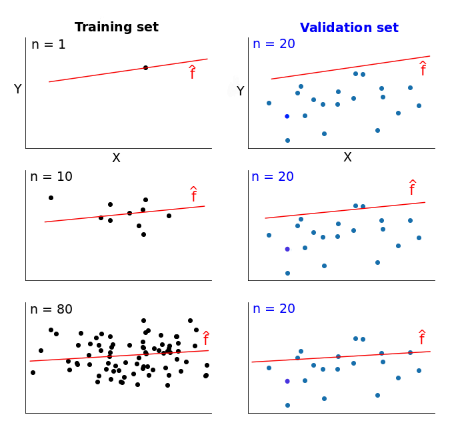
1. How much the estimator benefits from more data (e.g. do we have “enough data” or will the estimator get better if used in an online fashion).
2. If the estimator is more sensitive to error due to variance vs. error due to bias.

Using built-in learning\_curve() function from sklearn module we can easily draw learning curve.

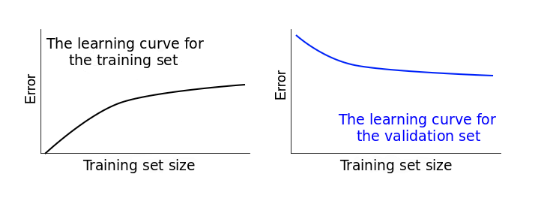




As we increase the training set size, the model cannot fit perfectly anymore the training set. So the training error becomes larger. However, the model is trained on more data, so it manages to fit better the validation set. Thus, the validation error decreases. To remind you, the validation set stays the same across all three cases.



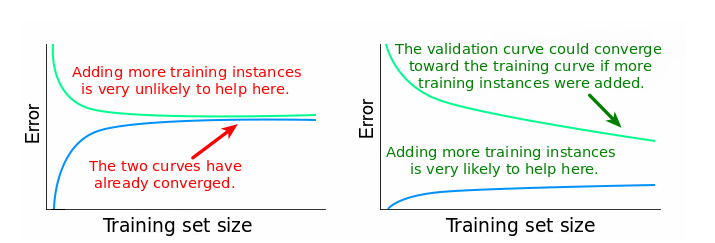
If we plotted the error scores for each training size, we’d get two learning curves looking similarly to these:

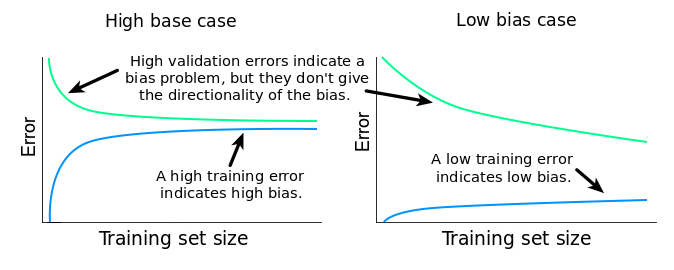


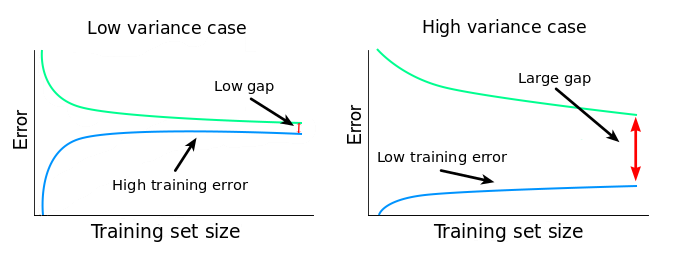
Learning curves give us an opportunity to diagnose bias and variance in supervised learning models.

If the training error is high, it means that the training data is not fitted well enough by the estimated model. If the model fails to fit the training data well, it means it has *high* bias with respect to that set of data.

A narrow gap indicates low variance. Generally, the narrower the gap, the lower the variance. The opposite is also true: the wider the gap, the greater the variance.







A learning curve shows the validation and training score/ error of an estimator for varying numbers of training samples. It is a tool to find out how much we benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error. If both the validation score and the training score converge to a value that is too low with increasing size of the training set, we will not benefit much from more training data.

#### **Fixing High Bias**

* When training and testing errors converge and are high
* No matter how much data we feed the model, the model cannot represent the underlying relationship and has high systematic error
* Poor fit
* Poor generalization

A high bias model has few parameters and may result in under fitting. Essentially we’re trying to fit an overly simplistic hypothesis, for example linear where we should be looking for a higher order polynomial. In a high bias situation, training and cross-validation error are both high and more training data is unlikely to help much.

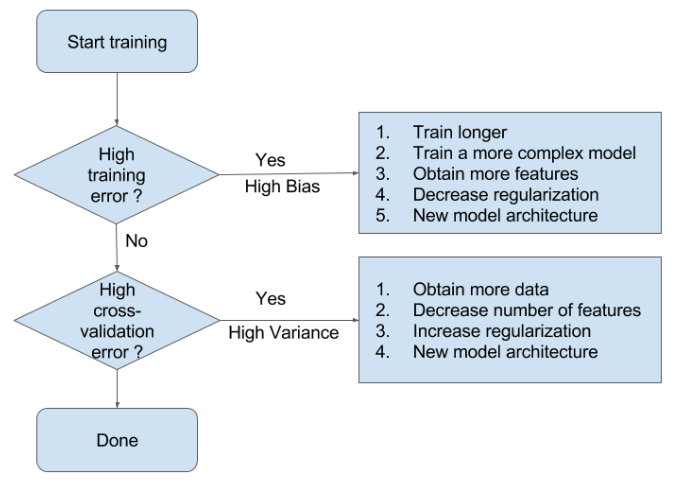
* find more features
* add polynomial features
* increase parameters (more hidden layer neurons, for example)
* decrease regularization

#### **Fixing High variance**

* When there is a large gap between the errors
* Require data to improve
* Can simplify the model with fewer or less complex features

Variance is the opposite problem, having lots of parameters, which carries a risk of overfitting. If we are overfitting, the algorithm fits the training set well, but has high cross-validation and testing error. If we see low training set error, with cross-validation error trending downward, then the gap between them might be narrowed by training on more data.

* more training data
* reduce number of features, manually or using a model selection algorithm
* increase regularization



It is normal that your training accuracy goes down when the dataset size grows. Think of it this way: when you have fewer samples (imagine that you have just one, at the extreme) it is easy to fit a model that has good accuracy for the training data, however that fitted model is not going to generalize well for test data. As you increase the dataset size, in general it is going to be harder to fit the training data, but hopefully your results generalize better for the test data.

### Advantages of using Polynomial Regression:

* Broad range of function can be fit under it.
* Polynomial basically fits wide range of curvature.
* Polynomial provides the best approximation of the relationship between dependent and independent variable.

### Disadvantages of using Polynomial Regression

* These are too sensitive to the outliers.
* The presence of one or two outliers in the data can seriously affect the results of a nonlinear analysis.
* In addition, there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.

# Generalized linear models

Generalized Linear Models (GLM) estimate regression models for outcomes following exponential distributions. In addition to the Gaussian (i.e. normal) distribution, these include Poisson, binomial, and gamma distributions. Each serves a different purpose, and depending on distribution and link function choice, can be used either for prediction or classification.

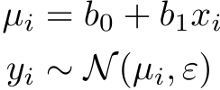
The GLM suite includes:

* Gaussian regression
* Poisson regression
* Binomial regression (classification)
* Quasibinomial regression
* Multinomial classification
* Gamma regression
* Ordinal regression
* Negative Binomial regression
* Tweedie distribution

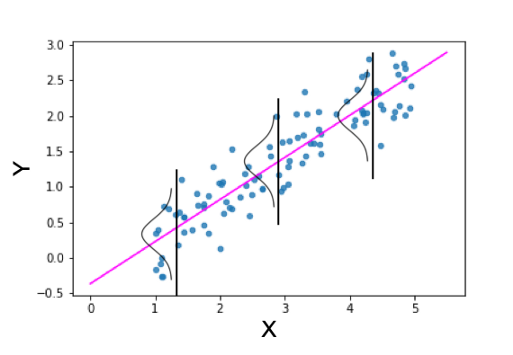
### Linear regression revisited

Linear regression is used to predict the value of continuous variable y by the linear combination of explanatory variables X.

In the univariate case, linear regression can be expressed as follows;



Here, i indicates the index of each sample. Notice this model assumes normal distribution for the noise term. The model can be illustrated as follows;



If you’d like to apply statistical modeling in real problems, you must know more than linear regression.

### Poisson regression

Poisson regression is typically used for datasets where the response represents counts, and the errors are assumed to have a Poisson distribution. In general, it can be applied to any data where the response is non-negative.

It models the dependency between the response and covariates as:



There are several problems if you try to apply linear regression for this kind of data.

1. The relationship between X and Y **does not look linear**. It’s more likely to be exponential.
2. **The variance of Y does not look constant**with regard to X. Here, the variance of Y seems to increase when X increases.
3. As Y represents the number of products, it always has to be a positive integer. In other words, Y is a **discrete variable**. However, the normal distribution used for linear regression assumes continuous variables. This also means the prediction by linear regression can be negative. It’s not appropriate for this kind of count data.

Here, the more proper model you can think of is the **Poisson regression**model. Poisson regression is an example of **generalized linear models (GLM)**.

There are three components in generalized linear models.

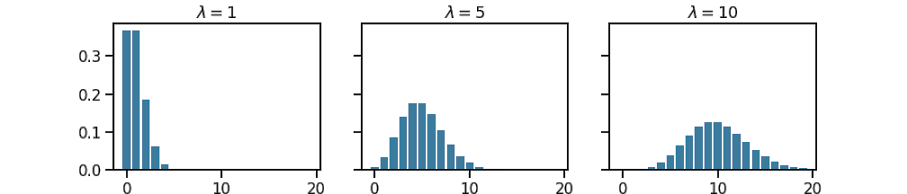
1. **Linear predictor**
2. **Link function**
3. **Probability distribution**

Linear predictor is just a linear combination of parameter (b) and explanatory variable (x).

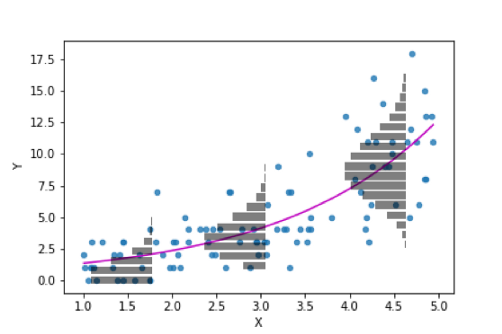
Link function literally “links” the linear predictor and the parameter for probability distribution. In the case of Poisson regression, the typical link function is the log link function. This is because the parameter for Poisson regression must be positive (explained later).

The last component is the probability distribution which generates the observed variable y. As we use Poisson distribution here, the model is called Poisson regression.

Poisson distribution is used to model count data. It has only one parameter which stands for both mean and standard deviation of the distribution. This means the larger the mean, the larger the standard deviation. See below.

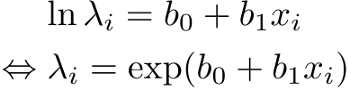


If we apply Poisson regression to respective data, the result should look like this.



The magenta curve is the prediction by Poisson regression. The bar plot of the probability mass function of Poisson distribution to make the difference from linear regression clear.

The prediction curve is exponential as the inverse of the log link function is an exponential function. From this, it is also clear that the parameter for Poisson regression calculated by the linear predictor guaranteed to be positive.



If you use Python, statsmodels library can be used for GLM. The code for Poisson regression is pretty simple.

# Poisson regression code  
import statsmodels.api as sm  
exog, endog = sm.add\_constant(x), y  
mod = sm.GLM(endog, exog,  
 family=sm.families.Poisson(link=sm.families.links.log))  
res = mod.fit()

endog (endogenous) and exog (exogenous) are how you call y and X in statsmodels. Notice you need to add the constant term to X. Without this, your linear predictor will be just b\_1\*x\_i.

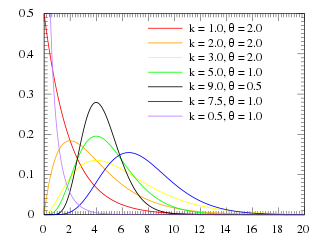
Actually, you don’t need to supply link argument here as log link is the default for the Poisson family.

### Linear Regression (Gaussian Family)

Linear regression corresponds to the Gaussian family model. The link function *g* is the identity, and density *f* corresponds to a normal distribution. It is the simplest example of a GLM but has many uses and several advantages over other families. Specifically, it is faster and requires more stable computations. Gaussian models the dependency between a response *y* and a covariates vector *x* as a linear function:

### Gamma Models

The gamma distribution is useful for modeling a positive continuous response variable, where the conditional variance of the response grows with its mean, but the coefficient of variation of the response *σ*2(*yi*)/*μi* is constant. It is usually used with the log link *g*(*μi*)=log(*μi*) or the inverse link *g*(*μi*)=1/*μi*, which is equivalent to the canonical link.



Suppose some event occurs https://chart.googleapis.com/chart?cht=tx&chl=r times in unit (1) interval. Then the probability density function https://chart.googleapis.com/chart?cht=tx&chl=p for interval https://chart.googleapis.com/chart?cht=tx&chl=y with https://chart.googleapis.com/chart?cht=tx&chl=s times occurrence of the same event is known as follows :

https://chart.googleapis.com/chart?cht=tx&chl=p(y+%7c+s%2c+r)+%3d+%5cfrac%7br%5es%7d%7b%5cGamma(s)%7d+y%5e%7bs-1%7d+exp(-ry)

where https://chart.googleapis.com/chart?cht=tx&chl=%5cGamma() is Gamma function. (https://chart.googleapis.com/chart?cht=tx&chl=%5cGamma(n%2b1)+%3d+n! for any positive integer, and it is an extension for real and complex number.

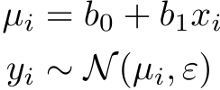
This regression is often used for the analytics, such as, waiting time, time between failure, accident rate and so forth.  
There exist two parameters https://chart.googleapis.com/chart?cht=tx&chl=s%2c%5c%3br (called “shape” and “rate” respectively), but we suppose https://chart.googleapis.com/chart?cht=tx&chl=s is the same (i.e, constant) for any observation in this regression.

For example, let’s consider the time to the failure (breakdown) for some equipment. Suppose, this occurrence depends on the count of initial bugs https://chart.googleapis.com/chart?cht=tx&chl=x_1 and the number of claims by customers https://chart.googleapis.com/chart?cht=tx&chl=x_2. Suppose, 1 failure (breakdown) is caused by 10 times errors. That is, https://chart.googleapis.com/chart?cht=tx&chl=s+%3d+10 .  
When the count of initial bugs increases, the errors in production would also increase. That is, when https://chart.googleapis.com/chart?cht=tx&chl=x_1 (the count of initial bugs) increases, https://chart.googleapis.com/chart?cht=tx&chl=r (the count of errors for unit interval) increases, though https://chart.googleapis.com/chart?cht=tx&chl=s%5c%3b(%3d10) (the count of errors for failure) is always constant. On the contrary, the average https://chart.googleapis.com/chart?cht=tx&chl=%5cfrac%7bs%7d%7br%7d decreases.

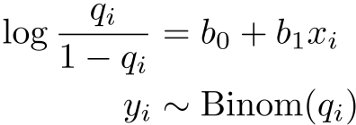
Then the following inverse link function is often used for Gamma regression. (I describe the details about the link function later.) Here https://chart.googleapis.com/chart?cht=tx&chl=%5cfrac%7b1%7d%7br%7d is called “scale”.

### Other typical GLM

Linear regression is also an example of GLM. It just uses identity link function (the linear predictor and the parameter for the probability distribution are identical) and normal distribution as the probability distribution.

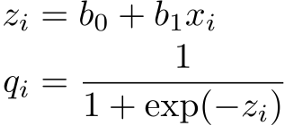


If you use logit function as the link function and binomial / Bernoulli distribution as the probability distribution, the model is called logistic regression.



Logistic regression

If you represent the linear predictor with z, the above equation is equivalent to the following.

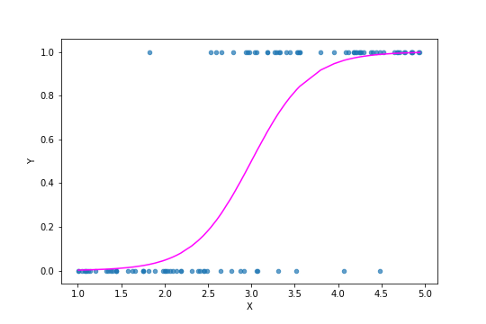


Logistic function

The right-hand side of the second equation is called logistic function. Therefore, this model is called logistic regression.

As the logistic function returns values between 0 and 1 for arbitrary inputs, it is a proper link function for the binomial distribution.

Logistic regression is used mostly for binary classification problems. Below is an example to fit logistic regression to some data.



### Custom GLM

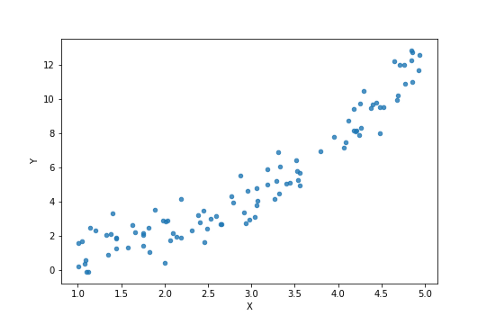
The models I’ve explained so far uses a typical combination of probability distribution and link function. In other words, all the models above use the **canonical link function.**

This is the list of probability distributions and their canonical link functions.

* Normal distribution: identity function
* Poisson distribution: log function
* Binomial distribution: logit function

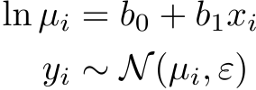
However, you don’t necessarily use the canonical link function. Rather, the advantage of statistical modeling is that you can make any kind of model that fits well with your data.

For example, let’s consider the following data.



However, if you see the data carefully, it seems the variance of y is constant with regard to X. Besides, y is continuous, not discrete.

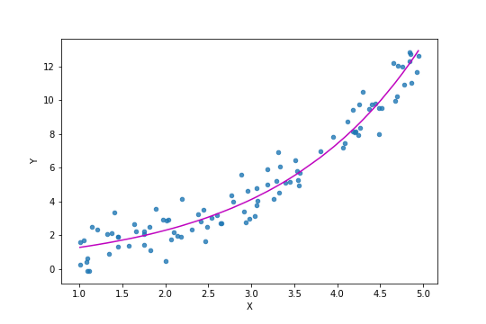
Therefore, it’s appropriate to use normal distribution here. As the relationship between X and y looks exponential, you had better choose the log link function.



With statsmodels you can code like this.

mod = sm.GLM(endog, exog,  
 family=sm.families.Gaussian(sm.families.links.log))  
res = mod.fit()

Notice you need to specify the link function here as the default link for Gaussian distribution is the identity link function. The prediction result of the model looks like this.



Various link functions are implemented in statsmodels. However, if you need to use more complex link functions, you have to write models yourself. For this purpose, probabilistic programming frameworks such as Stan, PyMC3 and TensorFlow Probability would be a good choice.

# Support Vector Regression

Support Vector regression is a type of Support vector machine that supports linear and non-linear regression. SVR for working with continuous Values instead of Classification which is SVM. Below giving some common terms for the working explanation.

**Kernel**: The function used to map a lower dimensional data into a higher dimensional data.

**Hyper Plane**: In SVM this is basically the separation line between the data classes. Although in SVR we are going to define it as the line that will help us predict the continuous value or target value

**Boundary line**: In SVM there are two lines other than Hyper Plane which creates a margin. The support vectors can be on the Boundary lines or outside it. This boundary line separates the two classes. In SVR the concept is same.

**Support vectors**: This are the data points which are closest to the boundary. The distance of the points is minimum or least.

In simple regression we try to minimize the error rate. While in SVR we try to fit the error within a certain threshold.

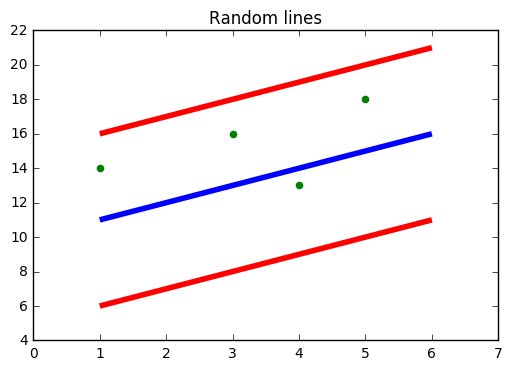
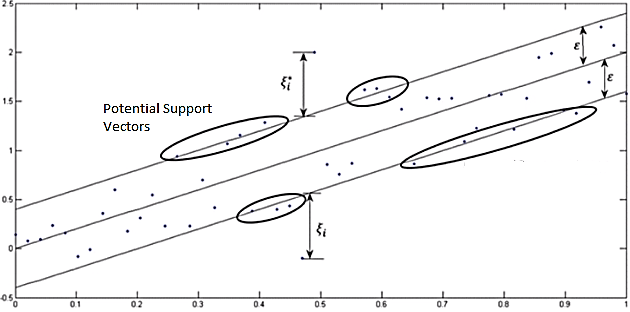


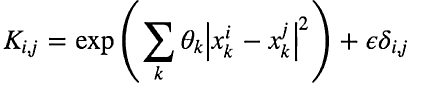
Fig shows Blue line: Hyper Plane; Red Line: Boundary Line. The boundary line is the lines that we draw are at ‘+e’ and ‘-e’ distance from Hyper Plane.



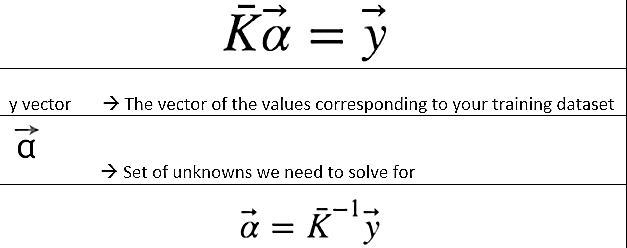
As it seems in the below graph, the mission is to fit as many instances as possible between the lines while limiting the margin violations. The violation concept in this example represents as ε (epsilon). SVR requires the training data: {X, Y} which covers the domain of interest and is accompanied by solutions on that domain. The work of the SVM is to approximate the function we used to generate the training set to reinforce some of the information we’ve already discussed in a classification problem.

Steps to Build a Support Vector Regression Model:

1. Collect a training {X, Y}
2. Choose a kernel and parameter and regularization if needed. (Gaussian Kernel and noise regularization are an instance for both steps)
3. Form the correlation matrix:



1. Train your machine, exactly or approximately, to get contraction coefficient by using the main part of the algorithm.

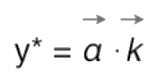
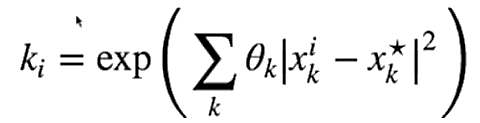


Y vector is the value corresponding to the training set

α Vector is the unknown values need to solve



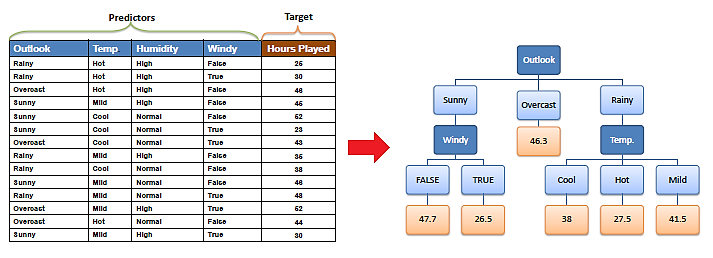
1. Use this coefficient to create an estimator.

 where 

The goal in linear regression is to minimize the error between the prediction and data. In SVR, the goal is to make sure that the errors do not exceed the threshold.

# Decision Tree Regression

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.



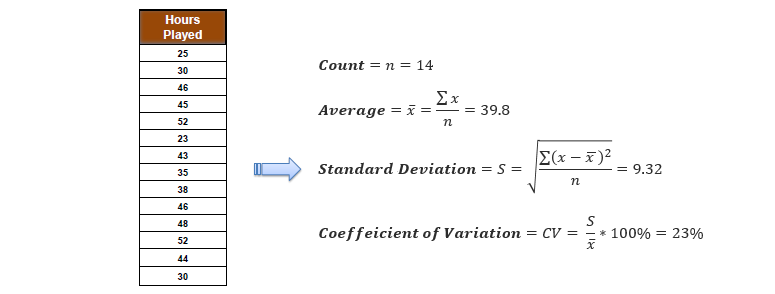
## Decision Tree Algorithm

The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction.

### Standard Deviation

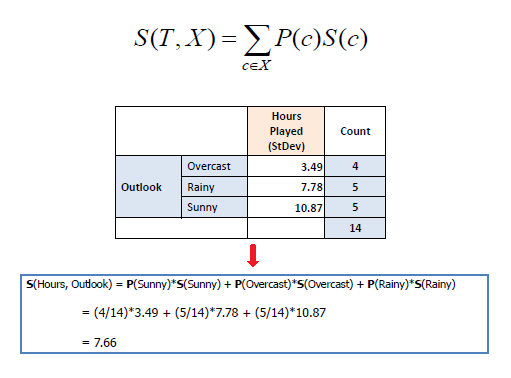
A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

#### a) Standard deviation for one attribute:



* Standard Deviation (S) is for tree building (branching).
* Coefficient of Deviation (CV) is used to decide when to stop branching. We can use Count (n) as well.
* Average (Avg) is the value in the leaf nodes.

#### b) Standard deviation for two attributes (target and predictor):



### Standard Deviation Reduction

The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest standard deviation reduction (i.e., the most homogeneous branches).

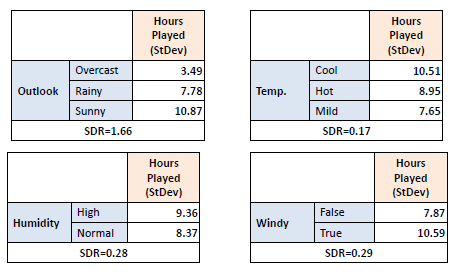
#### Step 1

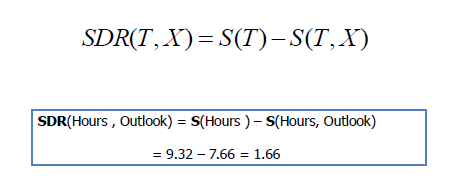
The standard deviation of the target is calculated.

Standard deviation (Hours Played) = 9.32

#### Step 2

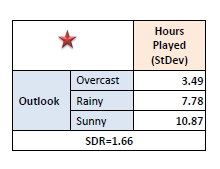
The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.





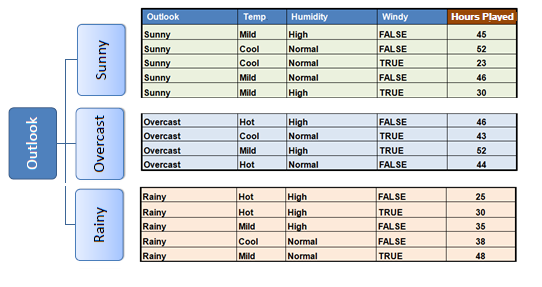
#### Step 3

The attribute with the largest standard deviation reduction is chosen for the decision node.



#### Step 4a

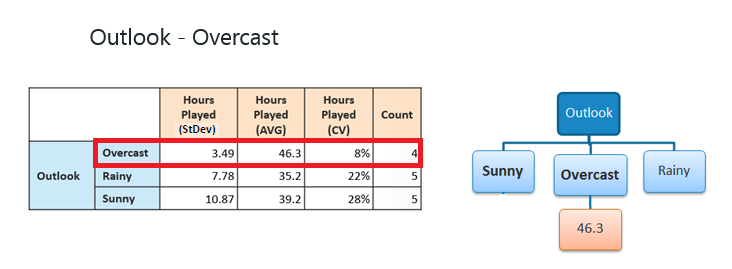
The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.



In practice, we need some termination criteria. For example, when coefficient of deviation (CV) for a branch becomes smaller than a certain threshold (e.g., 10%) and/or when too few instances (n) remain in the branch (e.g., 3).

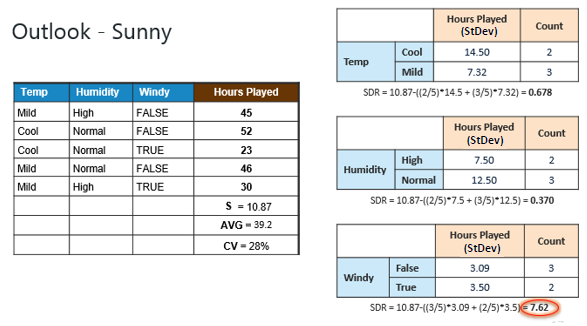
#### Step 4b

"Overcast" subset does not need any further splitting because its CV (8%) is less than the threshold (10%). The related leaf node gets the average of the "Overcast" subset.

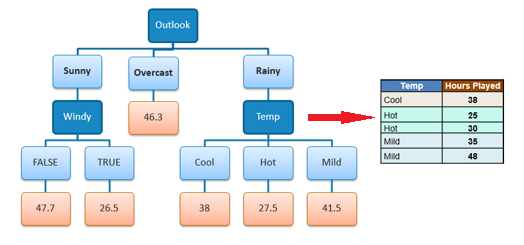


#### Step 4c

However, the "Sunny" branch has an CV (28%) more than the threshold (10%) which needs further splitting. We select "Windy" as the best node after "Outlook" because it has the largest SDR.



Because the number of data points for all three branches (Cool, Hot and Mild) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.

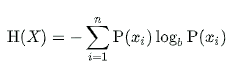


When the number of instances is more than one at a leaf node we calculate the average as the final value for the target.

Some important terms related to decision trees.

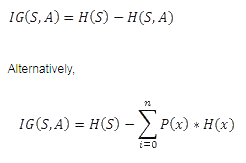
## Entropy

In machine learning, entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.



## Information Gain

Information gain can be defined as the amount of information gained about a random variable or signal from observing another random variable. It can be considered as the difference between the entropy of parent node and weighted average entropy of child nodes.

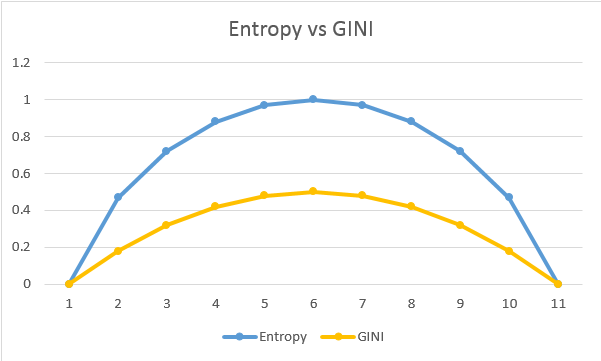


## Gini Impurity

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

https://miro.medium.com/max/182/1*FAijCE_5ypjW5VTFWiAFcw.png

Gini impurity is **lower bounded by 0**, with 0 occurring if the data set contains only one class.



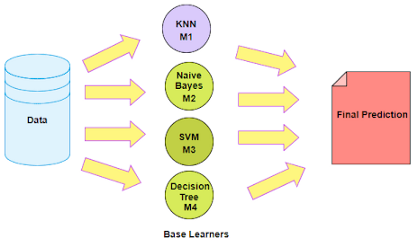
There are many algorithms there to build a decision tree. They are

1. **CART** (Classification and Regression Trees) — This makes use of Gini impurity as metric.
2. **ID3** (Iterative Dichotomiser 3) — This uses entropy and information gain as metric.

# Random Forest Regression

## ****Ensemble Learning****

An Ensemble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model. A model comprised of many models is called an Ensemble model.



## ****Types of Ensemble Learning****

1. Boosting.
2. Bootstrap Aggregation (Bagging).

### ****1. Boosting****

Boosting refers to a group of algorithms that utilize weighted averages to make weak learners into stronger learners. Boosting is all about “teamwork”. Each model that runs, dictates what features the next model will focus on.

In boosting as the name suggests, one is learning from other which in turn boosts the learning.

### ****2. Bootstrap Aggregation (Bagging)****

Bootstrap refers to random sampling with replacement. Bootstrap allows us to better understand the bias and the variance with the dataset. Bootstrap involves random sampling of small subset of data from the dataset.

It is a general procedure that can be used to reduce the variance for those algorithm that have high variance, typically decision trees. Bagging makes each model run independently and then aggregates the outputs at the end without preference to any model.

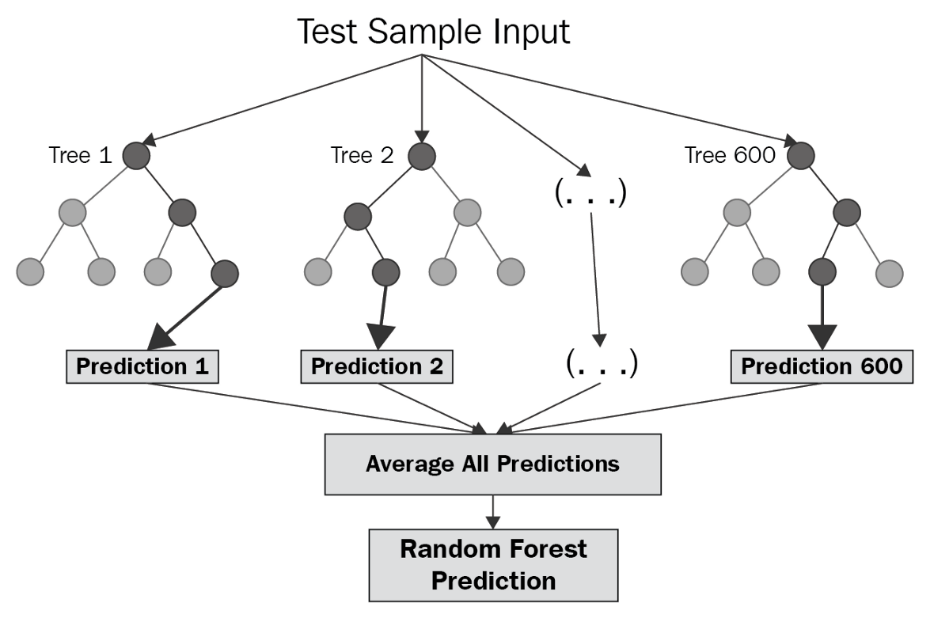
## ****Problems with Decision Trees****

Decision trees are sensitive to the specific data on which they are trained. If the training data is changed the resulting decision tree can be quite different and in turn the predictions can be quite different.

Also Decision trees are computationally expensive to train, carry a big risk of overfitting, and tend to find local optima because they can’t go back after they have made a split.

To address these weaknesses, we turn to Random Forest :) which illustrates the power of combining many decision trees into one model.

## ****Random Forest****



Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression.

Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees.

It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which aggregates many decision trees, with some helpful modifications:

* The number of features that can be split on at each node is limited to some percentage of the total (which is known as the hyper parameter). This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features.
* Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents overfitting.

The above modifications help prevent the trees from being too highly correlated.

## ****Feature and Advantages of Random Forest****

* It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
* It runs efficiently on large databases.
* It can handle thousands of input variables without variable deletion.
* It gives estimates of what variables that are important in the classification.
* It generates an internal unbiased estimate of the generalization error as the forest building progresses.
* It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

## ****Disadvantages of Random Forest****

* Random forests have been observed to over fit for some datasets with noisy classification/regression tasks.
* For data including categorical variables with different number of levels,random forests are biased in favor of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.