**Intro to Feature Selection Methods for Data Science**

[](https://towardsdatascience.com/@madelinemccombe?source=post_page-----4cae2178a00a----------------------)

[Madeline McCombe](https://towardsdatascience.com/@madelinemccombe?source=post_page-----4cae2178a00a----------------------)

[May 24](https://towardsdatascience.com/intro-to-feature-selection-methods-for-data-science-4cae2178a00a?source=post_page-----4cae2178a00a----------------------) · 11 min read

Authored by: Ryan Farmar, Ning Han, Madeline McCombe

***What is feature selection?***

Well, let’s start by defining what a feature is. A feature is an X variable in your dataset, most often defined by a column. Many datasets nowadays can have 100+ features for a data analyst to sort through! That is a ridiculous amount to process normally, which is where feature selection methods come in handy. They allow you to reduce the number of features included in a model without sacrificing the predictive power. Features that are redundant or irrelevant can actually negatively impact your model performance, so it is necessary (and helpful) to remove them. Imagine trying to learn to ride a bike by making a paper airplane. I doubt you’d get very far on your first ride.

***Benefits of feature selection***

The main benefit of feature selection is that it reduces overfitting. By removing extraneous data, it allows the model to focus only on the important features of the data, and not get hung up on features that don’t matter. Another benefit of removing irrelevant information is that it improves the accuracy of the model’s predictions. It also reduces the computation time involved to get the model. Finally, having a smaller number of features makes your model more interpretable and easy to understand. Overall, feature selection is key to being able to predict values with any amount of accuracy.

**Overview**

There are three types of feature selection: Wrapper methods (forward, backward, and stepwise selection), Filter methods (ANOVA, Pearson correlation, variance thresholding), and Embedded methods (Lasso, Ridge, Decision Tree). We will go into an explanation of each with examples in Python below.

***Wrapper methods***

Wrapping methods compute models with a certain subset of features and evaluate the importance of each feature. Then they iterate and try a different subset of features until the optimal subset is reached. Two drawbacks of this method are the large computation time for data with many features, and that it tends to overfit the model when there is not a large amount of data points. The most notable wrapper methods of feature selection are **forward selection, backward selection, and stepwise selection**.

**Forward selection** starts with zero features, then, for each individual feature, runs a model and determines the p-value associated with the t-test or F-test performed. It then selects the feature with the lowest p-value and adds that to the working model. Next, it takes the first feature selected and runs models with a second feature added and selects the second feature with the lowest p-value. Then it takes the two features previously selected and runs models with a third feature and so on, until all features that have significant p-values are added to the model. Any features that never had a significant p-value when tried in the iterations will be excluded from the final model.

**Backward selection** starts with all features contained in the dataset. It then runs a model and calculates a p-value associated with the t-test or F-test of the model for each feature. The feature with the largest insignificant p-value will then be removed from the model, and the process starts again. This continues until all features with insignificant p-values are removed from the model.

**Stepwise selection** is a hybrid of forward and backward selection. It starts with zero features and adds the one feature with the lowest significant p-value as described above. Then, it goes through and finds the second feature with the lowest significant p-value. On the third iteration, it will look for the next feature with the lowest significant p-value, and it will *also* *remove* *any features that were previously added that now have an insignificant p-value*. This allows for the final model to have all of the features included be significant.

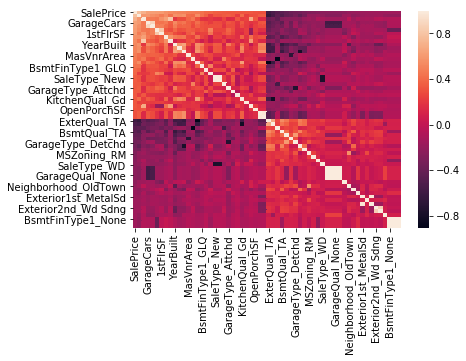
The benefits of the different selection methods above is that they give you a good starting point if you have no intuition about the data and what features may be important. Also, it effectively selects a model with significant features from a large amount of data. However, some drawbacks are that the methods do not run through every single combination of features, so they may not end up with the absolute best model. Also, it can result in a model with high multicollinearity (inflated beta coefficients due to relationships among features), which is not great for predicting accurately.

***Filter methods***

Filter methods use a measure other than error rate to determine whether that feature is useful. Rather than tuning a model (as in wrapper methods), a subset of the features is selected through ranking them by a useful descriptive measure. Benefits of filter methods are that they have a very low computation time and will not overfit the data. However, one drawback is that they are blind to any interactions or correlations between features. This will need to be taken into account separately, which will be explained below. Three different filter methods are **ANOVA, Pearson correlation, and variance thresholding**.

The **ANOVA** (Analysis of variance) test looks a the variation within the treatments of a feature and also between the treatments. These variances are important metrics for this specific filtering method because we can determine whether a feature does a good job of accounting for variation in the dependent variable. If the variance within each specific treatment is larger than the variation between the treatments, then the feature hasn’t done a good job of accounting for the variation in the dependent variable. To carry out an ANOVA test, an F statistic is computed for each individual feature with the variation between treatments in the numerator(SST, often confused with SSTotal) and the variation within treatments in the denominator. This test statistic is then tested against the null hypothesis ( *H0 : Mean value is equal across all treatments*) and the alternative ( *Hα : At least two treatments differ*).

The **Pearson correlation** coefficient is a measure of the similarity of two features that ranges between -1 and 1. A value close to 1 or -1 indicates that the two features have a high correlation and may be related. To create a model with reduced features using this correlation coefficient, you can look at a heatmap (like the one shown below) of all the correlations and pick the features that have the highest correlation with the response variable (Y variable or the predictor variable). The cutoff value of high correlation vs low correlation depends on the range of correlation coefficients within each dataset. A general measure of high correlation is 0.7 < |correlation| < 1.0. This will allow the model that uses the features selected to encompass a majority of the valuable information contained in the dataset.



The response variable for this dataset SalePrice (top row) shows the correlation with the other variables. The light orange and dark purple show high correlations.

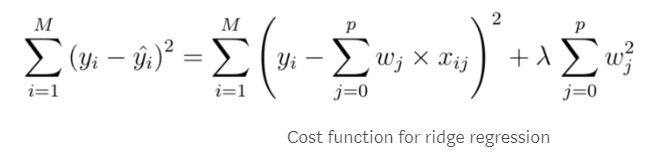
Another filter method of feature reduction is **variance thresholding**. The variance of a feature determines how much predictive power it contains. The lower the variance is, the less information contained in the feature, and the less value it has in predicting the response variable. Given this fact, variance thresholding is done by finding the variance of each feature, and then dropping all of the features below a certain variance threshold. This threshold could be 0 if you only want to remove features that have the same value for each instance of the response variable. However, to remove more features from your dataset, the threshold could be set to 0.5, 0.3, 0.1, or another value that makes sense for the distribution of variances.

As mentioned previously, sometimes **interactions** could be useful to add to your model, especially when you suspect that two features have a relationship that can provide useful information to your model. An interaction can be added to a regression model as an interaction term, shown as a B3X1X2. The beta coefficient (B3) modifies the product of X1 and X2, and measures the effect of the model of the two features (Xs) combined. To see if an interaction term is significant, you can perform a t-test or F-test and look to see if the p-value of the term is significant. One important note is that if the interaction term is significant, both lower order X terms must be kept in the model, *even if they are insignificant*. This is to preserve the X1 and X2 as two independent variables rather than one new variable.

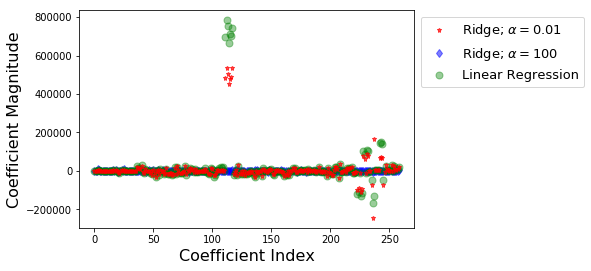
***Embedded Methods***

Embedded methods perform feature selection as a part of the model creation process. This generally leads to a happy medium between the two methods of feature selection previously explained, as the selection is done in conjunction with the model tuning process. **Lasso and Ridge regression** are the two most common feature selection methods of this type, and **Decision tree** also creates a model using different types of feature selection.

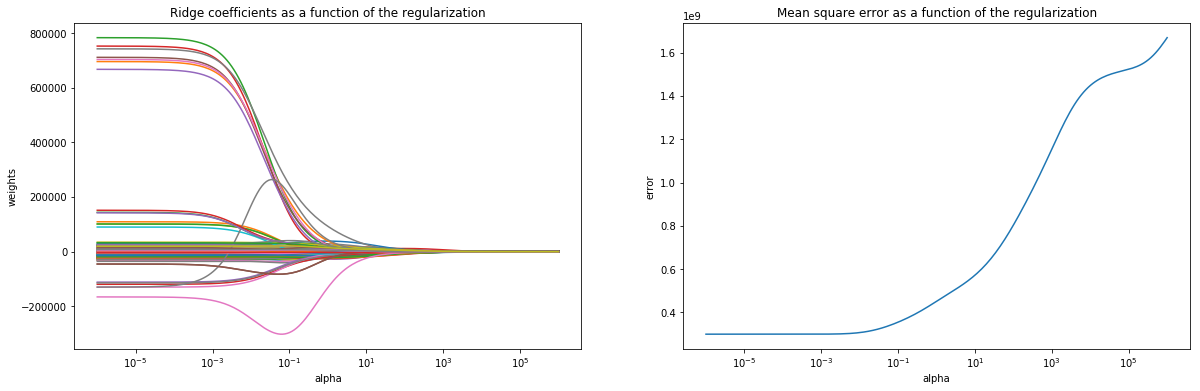
Occasionally you may want to keep all the features in your final model, but you don’t want the model to focus too much on any one coefficient. **Ridge regression** can do this by penalizing the beta coefficients of a model for being too large. Basically, it scales back the strength of correlation with variables that may not be as important as others. This takes care of any multicollinearity (relationships among features that will inflate their betas) that may be present in your data. Ride Regression is done by adding a penalty term (also called ridge estimator or shrinkage estimator) to the cost function of the regression. The penalty term takes all of the betas and scales them by a term lambda (λ) that must be tuned (usually with cross validation: compares the same model but with different values of lambda). Lambda is a value between 0 and infinity, although it is good to start with values between 0 and 1. The higher the value of lambda, the more the coefficients are shrunk. When lambda is equal to 0, the result will be a regular ordinary least squares model with no penalty.



Function from: <https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/>

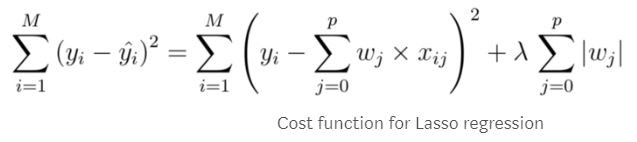


This shows how Ridge regression can adjust some of the large coefficients found in linear regression by making them closer to zero.



As the value of lambda (alpha) increases, the coefficients are pushed toward zero with at the cost of MSE.

**Lasso Regression** is another way to penalize the beta coefficients in a model, and is very similar to Ridge regression. It also adds a penalty term to the cost function of a model, with a lambda value that must be tuned. The most important distinction from Ridge regression is that Lasso Regression can force the Beta coefficient to zero, which will remove that feature from the model. This is why Lasso is preferred at times, especially when you are looking to reduce model complexity. The smaller number of features a model has, the lower the complexity. In order to force the coefficients to zero, the penalty term added to the cost function takes the absolute value of the beta terms instead of squaring it, which when trying to minimize the cost, can negate the rest of the function, leading to a beta equal to zero.



Function from: <https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/>

An important note for Ridge and Lasso regression is that *all of your features must be standardized*. Many functions in Python and R do this automatically, because the lambda must be applied equally to each feature. Having one feature with values in the thousands and another with decimal values will not allow this to happen, hence the standardization requirement.

Another common way to model data with feature selection is called **Decision Tree**, which can either be a regression tree or classification tree depending on whether the response variable is continuous or discrete, respectively. This method creates splits in the tree based on certain features to create an algorithm to find the correct response variable. The way the tree is built uses a wrapper method inside an embedded method. What we mean by that is, when making the tree model, the function has several feature selection methods built into it. At each split, the function used to create the tree tries all possible splits for all the features and chooses the one that splits the data into the most homogenous groups. In plain terms, it chooses the feature that can best predict what the response variable will be at each point in the tree. This is a wrapper method since it tries all possible combinations of features and then picks the best one.

The most important features in predicting the response variable are used to make splits near the root (start) of the tree, and the more irrelevant features aren’t used to make splits until near the nodes of the tree (ends). In this way, decision tree penalizes features that are not helpful in predicting the response variable (embedded method). After a tree has been made, there is an option to go back and ‘prune’ some of the nodes that do not provide any additional information to the model. This prevents overfitting, and is usually done through cross validation with a holdout test set.

***Summary***

So, now that you made it through all that, what is the most important idea to take away? Even though a dataset may have hundreds to thousands of features, that doesn’t mean that all of them are important or useful. Especially now that we live in a world with unimaginable amounts of data, it is important to try to focus on the bits that matter. There are many more (complex) ways to perform feature selection that we haven’t mentioned here, but the methods described are a great place to start! Good luck, and model on!

**Key Vocabulary:**

**Feature:** an x variable, most often a column in a dataset  
**Feature selection:** optimizing a model by selecting a subset of the features to use

**Wrapper method:** trying models with different subsets of features and picking the best combination  
**Forward selection:** adding features one by one to reach the optimal model  
**Backward selection:** removing features one by one to reach the optimal model  
**Stepwise selection:** hybrid of forward and backward selection.adding and removing features one by one to reach the optimal model

**Filter method:** selecting a subset of features by a measure other than error (a measure that is inherent to the feature and not dependent on a model)  
**Pearson Correlation:** a measure of the linear correlation between two variables  
**Variance thresholding:** selecting the features above a variance cutoff to preserve most of the information from the data  
**ANOVA:** (analysis of variance) a group of statistical estimation procedures and models that is used to observe differences in treatment (sample) means; can be used to tell when a feature is statistically significant to a model  
**Interacting term:** quantifies the relationship between two of the features when they depend on the value of the other; alleviates multicollinearity and can provide further insight into the data  
**Multicollinearity:** occurs when two or more independent variables are highly correlated with each other

**Embedded method:** selecting and tuning the subset of features during the model creation process  
**Ridge Regression:** a modified least squares regression that penalizes features for having inflated beta coefficients by applying a lambda term to the cost function  
**Lasso Regression:** similar to ridge regression, but different in that the lambda term added to the cost function can force a beta coefficient to zero  
**Decision Tree:** a non-parametric model that using features as nodes to split samples to correctly classify an observation. In a random forest model, feature importance can be calculated using mean decrease gini score.   
**Cross Validation:** a method to iteratively generate training and test datasets to estimate model performance on future unknown datasets