Machine Learning & Predictive Analytics

UFCFMJ-15-M

20042768

Coursework Assignment

Discussion Paper : Applied Knowledge Machine Learning & Predictive Analytics

Contents:

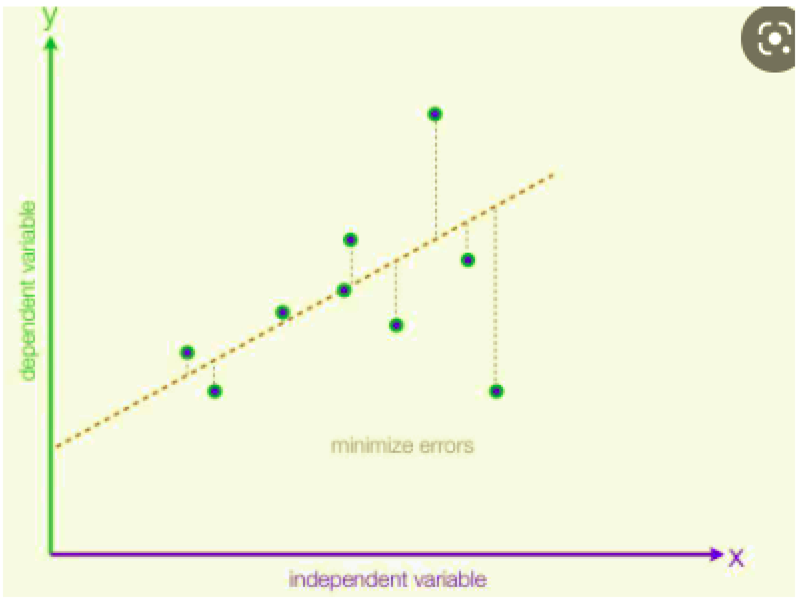
Introduction & Choosing an Algorithm

Within the realm of linear regression there are many underlying relationships between independent variables, and dependent variables. Linear regression is one of the most common machine learning algorithms because there are many relationships between variables that work in correlation. Correlation is defined as a "mutual relationship or connection between two or more things." Naturally we see correlation with many independent variables in relation to the universal dependent price variable. This can be seen in the ever so popular housing datasets, car pricing, salary predicting, etc. This is because the underlying independent features are linearly regressed against the dependent predictor, most commonly known as price. This allows linear regression to be labeled as a multivariate analysis. The formula for linear regression is as follows:

$$Y_i = f(X_i, \beta) + e_i$$

Y_i	= dependent variable
f	= function
X_i	= independent variable
β	= unknown parameters
e_i	= error terms

From the formula we fit a straight line to our actual outcomes. The function of our independent variable, or variables plus our error terms provides a line that attempts to best fit our data. Linear regression or ordinary least squares are synonymous in name. When one is referencing these they are essentially wanting to minimize the sum of squared differences between the observed and the predicted values. This is known as our cost function, and we want to minimize in order to find the optimal fitting line. An example of the positive correlated relationship between an independent variable, and a dependent variable(s) are shown below. For the sake of explanation I will continue with the theme of predicting house prices throughout the paper.



Initial Steps

From a machine learning perspective in order to begin thinking about creating an algorithm we must first have data, and in order to get a respectable scoring outcome the independent feature, and dependent features must have some type of relationship. Given a dataset, the first step is to check for null values. By having null values it will unduly influence our algorithm with outliers and pull our regression line in an unwarranted direction.

Next we must view our variables from a different perspective, do we have categorical variables, binary, or all continuous. It is extremely important to make sure that these variables are represented the right way within our dataframe. For example, say one of our variables is categorical and labels houses on a 1 to 4 scale based on their age. If all of these labels are within one column we are representing a huge problem to our algorithm. Because the linear algorithm understands a specific rating in regards to age we would be providing disproportional weightings. All other variables held constant, if a house age were to land in category one we are saying that the price effect is the same as moving from category one to two and two to three.

Because of these disproportionate weightings we must break these categories up into 4 separate columns, and simply place a one if true and zero if not. This theme also applies to binary variables as well in checking that we only possess ones and zeros.

The next step is to break our data set into a target variable, and independent variables. So in this case we are predicting home price, by utilizing relationships such as rooms, square footage, etc. The next step is to then split up the data into training, and testing sets, and this is most commonly done through Sci Kit Learn’s Train Test Split. When we split up our data, and train our linear regression algorithm we are supplying the model with price outputs. Therefore we are working on a supervised learning problem because the algorithm is trained on data with price outputs.

Scoring Metrics

Once we have a clean data frame broken down correctly, then it is now ready to be fitted as a baseline linear regression model. We pass in our training set for independent, and dependent features, and begin to evaluate the model. Linear Regression's default scoring metric is R Squared, also known as the coefficient of determination. R squared can be said to represent the independent variables explained variation in regard to the dependent. For example, if we received a 78% R squared for independent variables in regards to home price then it can be said that those independent variables represent 78% of the explained variation in home prices. In addition to R squared this gives us a chance to evaluate our coefficients, and p-values. The coefficients relate to the impactful weighting of our independent variables, while the p-value can be said to represent significance in our linear regression equation.

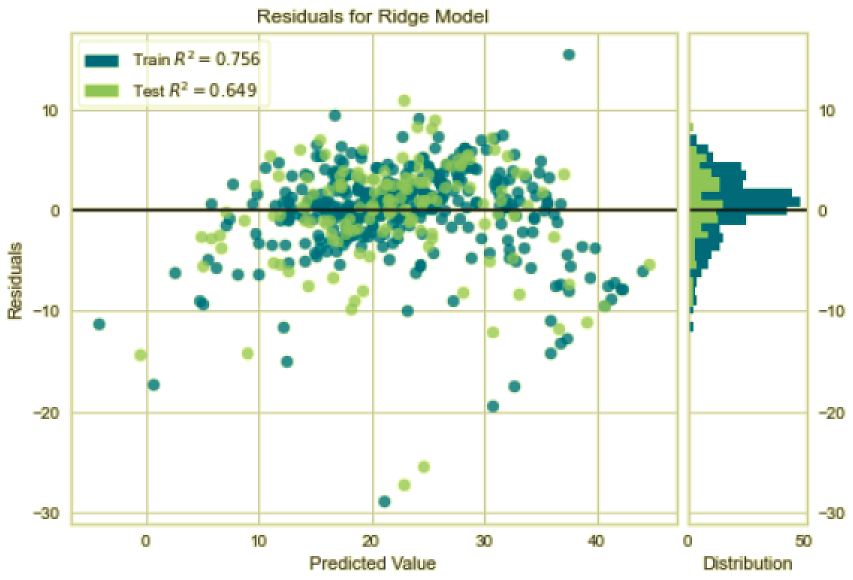
The next thing to assess is the visualization of independent features in respect to our dependent or target variables. These patterns can be quite telling in the functional form of independent variables within our equation. If the graphical visualization of a scatter plot is completely spread out with no pattern, then we do not have a correlated relationship with the target variable. Right off the bat, these features are candidates to be removed from the algorithm. However if the pattern of the scatter plot points represent a curve, or have a general pattern there are methods of improvement such as representing squared relationships, or inverses. For example, say that the age of a home in respect to the price represents a parabola figure, well then we can represent that in our equation by creating a whole new row of data that has age numbers squared. When we make alterations to our algorithm it is important to keep testing our metrics like R2, coefficient, and p-values. From this we can understand if our algorithm is improving or not, and the importance of independent variables.

Another regression assumption that we must test in addition to linearity is known as multicollinearity. Multicollinearity occurs when the independent variables themselves are highly related. For example, suppose we have total square footage, and square footage of the lot, well then there is a high chance that these features will be highly correlated showing the same amount of variance in respect to the target variable. This poses a problem because linear regression equations attempt to isolate the effects of individual independent variables in predicting the dependent target variable. The problem resides in holding other variables constant. We can not keep total square footage constant, if the square footage of the lot is moving. Simply put, our independent variables will have a tough time in making sense of what impacted the price of the house.

An alternative to dropping features based on multicollinearity is known as the variance inflation factor test. To keep the theory behind variance inflation factor simple, it measures the variance by calculating the difference of independent variables in regression equations, versus without them. The higher the variance inflation factor number then the higher the likelihood that the variable information is already accounted for within other variables in the regression equation. For example, if the independent variable known as total square footage received a VIF rating of 8.42, then that lets us know that the variance of the coefficient is inflated by 8.42. Inherently this is calculated through the formula shown below as we get the R squared value of the model when we remove square footage of .88. In essence, this value is a way of letting us know that this independent variable is already accounted for within our ML model.

$$VIF_{Weight} = \frac{Var(b_{Weight})}{Var(b_{Weight})_{min}} = \frac{1}{1 - R^2_{Weight}} = \frac{1}{1 - 0.8812} = 8.42.$$

Further evaluation will lead to checking the residuals. Residuals are the difference between our predicted values, and our actual values. Based on regression assumptions residuals should have a mean of zero, and we can measure the amount of variance explained through our R squared values. Below we see the test set has a lesser value than the training set which makes sense, as the model was trained on that specific training data, and had a greater population. Another evaluation we can make is that the majority of the train and test residuals fall above our regression line. This implicitly states that our predictions are overvalued. More than likely we can attribute this issue in our model to the positive skew in our target variable. In a linear regression model we are always looking for data that is symmetrical and bell shaped, and with house prices most of them fall within a certain range. The solution to this issue is transforming our target variable.



Hyper Parameter Tuning

Ridge and Lasso Regression are regularization parameters of linear regression equations. Normal linear regression gives you coefficients of independent features in order to best predict an outcome. This makes the regression algorithm susceptible to overfitting as it learns the underlying relationships and allocated coefficients to independent variables by linearly regressing against the target variable. When this algorithm is utilized on unseen data it has potential to do poorly because of the data it was too closely trained upon. This leads us to regularization techniques to combat overfitting by numbing, or adjusting the coefficients of our original linear regression.

Specific to ridge regression (L2) , it is the suggested regularization parameter when there is a dataset with many independent features that have a high multicollinearity. This is typically what is seen in housing datasets. Whereas Lasso is best implemented when there are fewer features that have a more significant impact on the model. Lasso is known as the least absolute shrinkage and selection operator because its best impact shows when features have a small variance to begin with.

For ridge regression the number one parameter to consider is alpha. Alpha minimizes the sensitivity towards independent features by causing the regression line to become more horizontal. As alpha grows it causes the slope to decrease. When tuning our alpha parameter it should be stated that the slope will never reach zero meaning our features will never be completely eliminated. The parameter to tune for Lasso or L1 is known as lambda. As lambda grows it will eventually reach a point where we absolutely mute the given coefficient of an independent variable all together.

The optimal way to go about this in Python is through Sci Kit Learn's GridSearchCV. This allows us to pass in a mix of different values for alpha & lambda in order to find the optimal combination for the linear regression model. In the background of the function, our data is broken into a number of different folds. For example, in 5 folds it would be trained upon the first four, and tested on the last, then trained upon the last four, and tested on the first until all folds have been tested upon. It is an ideal function to find optimal parameters.

Less Suitable Algorithm

A less suitable algorithm for a linear dataset would be a clustering algorithm such as K-means clustering. First off this algorithm would not make sense for predicting house price because it is an unsupervised learning algorithm that uses classified predictions. In a housing dataset we are working with a continuous target variable where we want to predict that amount. We are not looking to classify it into some category. In the case of predicting house prices a supervised learning approach will reign supreme over unsupervised.

Clustering algorithms measure distances based on where data falls relative to some starting point. As an unsupervised learning algorithm k-means attempts to learn from its previous iterations. It makes sense of the data based on location, and classifies accordingly. Therefore clustering would not be a suitable algorithm for a dataset where a linear regression would work well.

Word Count

1936

Reference List

Jeff Macaluso. (2018). *Testing Linear Regression Assumptions in Python*. [online] Available at: https://jeffmacaluso.github.io/post/LinearRegressionAssumptions/.

scikit-learn.org. (n.d.). *sklearn.linear_model.Ridge — scikit-learn 0.23.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

online.stat.psu.edu. (n.d.). *10.7 - Detecting Multicollinearity Using Variance Inflation Factors | STAT 462*. [online] Available at: https://online.stat.psu.edu/stat462/node/180/.

Stephanie (2015). *Variance Inflation Factor*. [online] Statistics How To. Available at: https://www.statisticshowto.com/variance-inflation-factor/.

Anon, (n.d.). *Interpreting Residual Plots to Improve Your Regression*. [online] Available at: https://www.qualtrics.com/support/stats-iq/analyses/regression-guides/interpreting-residual-plots-improve-regression// [Accessed 2 Nov. 2021].