**Regression**

**Linear:** it is one of the most common predictive analytics techniques that attempts to measure the strength of linearly assumed relationship between the dependent and independent variables. The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (all data points) are as small as possible. Error is the distance between the point to the regression line.

In order to implement linear regression, the following four assumptions are to be met:

* **Normality** - When we talk about normality what we mean is that the data should look like a normal distribution. This is important because several statistic tests rely on this (e.g. t-statistics). In this exercise we'll just check univariate normality for 'SalePrice' (which is a limited approach). Remember that univariate normality doesn't ensure multivariate normality (which is what we would like to have), but it helps. Another detail to take into account is that in big samples (>200 observations) normality is not such an issue. However, if we solve normality, we avoid a lot of other problems (e.g. heteroscedacity) so that's the main reason why we are doing this analysis.
* **Homoscedasticity** – It refers to the 'assumption that dependent variable(s) exhibit equal levels of variance across the range of predictor variable(s)'. Homoscedasticity is desirable because we want the error term to be the same across all values of the independent variables.
* **Linearity**- The most common way to assess linearity is to examine scatter plots and search for linear patterns. If patterns are not linear, it would be worthwhile to explore data transformations. However, we'll not get into this because most of the scatter plots we've seen appear to have linear relationships.
* **Absence of correlated errors** - Correlated errors, like the definition suggests, happen when one error is correlated to another. For instance, if one positive error makes a negative error systematically, it means that there's a relationship between these variables. This occurs often in time series, where some patterns are time related. We'll also not get into this. However, if you detect something, try to add a variable that can explain the effect you're getting. That's the most common solution for correlated errors.
* **Assumption of Linear Independence**: we assume that the predictor variables are linearly independent. Point to be noted: Also if the columns of predictor variables are almost linearly independent, i.e., highly correlated then the estimates beta-hat become very unstable, i.e., sensitive to data.
* **Assuming the distributions of the parameters**: we assume the underlying distribution of the parameters.

**sns.heatmap and sns.pairplot**

Can do heatmap (sns.heatmap) to check the correlations amongst the different predictor variables. Also can do the sns.pairplot to get scatterplots between every predictor variable and the dependent variable. This will mostly give a good understanding re which variables have varying variance with the dependent variable, which ones are positively correlated, and which ones are negatively correlated, and which ones exhibit non-linear trend with the target variable, etc.,

A **Normal probability plot using stats.probplot:**

Data distribution should closely follow the diagonal that represents the normal distribution.

In case of the dependent/output variable not following the diagonal line closely a simple data transformation can help to solve the problem. For example: in case of positive skewness, log transformations usually work well. We can’t apply log transformations when we have zero values. In those situations, apply log transformations only on non-zero values and interpret zero values differently. So, the obvious question becomes: should we try to normalize all the variables? In case of regression, normalizing at least the strongly skewed variables (as many of them) will be beneficial to build an accurate model.

**Measuring 'homoscedasticity'**

The best approach to test homoscedasticity for two metric variables is graphically. Departures from an equal dispersion are shown by such shapes as cones (small dispersion at one side of the graph, large dispersion at the opposite side) or diamonds (a large number of points at the center of the distribution).

A regression line with one feature and one target variable can be represented as:

Y(pred) = b0 + b1\*x

The values b0 and b1 must be chosen so that they minimize the error. If sum of squared error is taken as a metric to evaluate the model, then goal to obtain a line that best reduces the error.

/var/folders/_v/g0gkxbk53lvdm02dlx_tstvc0000gn/T/com.microsoft.Word/WebArchiveCopyPasteTempFiles/1*Utp8sgyLk7H39qOQY9pf1A.png

If we don’t square the error, then positive and negative point will cancel out each other.

For model with one predictor,

/var/folders/_v/g0gkxbk53lvdm02dlx_tstvc0000gn/T/com.microsoft.Word/WebArchiveCopyPasteTempFiles/1*1evY0PuCUENCpDP_QRplig.png

Co-efficient formula

b1=sigmani=1 (xi – x^)(yi-y^)/ sigmani=1 (xi – x^)^2

**Exploring ‘b1’**

* If b1 > 0, then x(predictor) and y(target) have a positive relationship. That is increase in x will increase y.
* If b1 < 0, then x(predictor) and y(target) have a negative relationship. That is increase in x will decrease y.

**Exploring ‘b0’**

* If the model does not include x=0, then the prediction will become meaningless with only b0. For example, we have a dataset that relates height(x) and weight(y). Taking x=0(that is height as 0), will make equation have only b0 value which is completely meaningless as in real-time height and weight can never be zero. This resulted due to considering the model values beyond its scope.
* If the model includes value 0, then ‘b0’ will be the average of all predicted values when x=0. But, setting zero for all the predictor variables is often impossible.
* The value of b0 guarantee that residual have mean zero. If there is no ‘b0’ term, then regression will be forced to pass over the origin. Both the regression co-efficient and prediction will be biased.

**Co-efficient from Normal equations**

Apart from above equation co-efficient of the model can also be calculated from normal equation.

/var/folders/_v/g0gkxbk53lvdm02dlx_tstvc0000gn/T/com.microsoft.Word/WebArchiveCopyPasteTempFiles/1*rwgC2rwbjaGqR4YSGXCuow.png

Theta contains co-efficient of all predictors including constant term ‘b0’. Normal equation performs computation by taking inverse of input matrix. Complexity of the computation will increase as the number of features increase. It gets very slow when number of features grow large.

**Optimizing using gradient descent**

Complexity of the normal equation makes it difficult to use, this is where gradient descent method comes into picture. Partial derivative of the cost function with respect to the parameter can give optimal co-efficient value.

**Interpreting Linear Regression outcome**

This is for interpreting the outcome of linear regression done using stats.model. Check for the p-values of the coefficients along with their values in the confidence intervals. This is important as sometimes we get a agreeable p-value at standard significance we set but the values in the confidence interval include negative, zero, and positive values. It’s actually an inconclusive result. So we can drop that feature and try to build the model again. Also need to check for the R^2 value. We can check the Adjusted R^2 value to understand when we drop a variable whether the Adjusted R^2 value goes up or down.

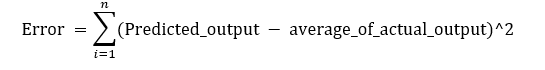
**Metrics for model evaluation**

**R-Squared value**

This value ranges from 0 to 1. Value ‘1’ indicates predictor perfectly accounts for all the variation in Y. Value ‘0’ indicates that predictor ‘x’ accounts for no variation in ‘y’.

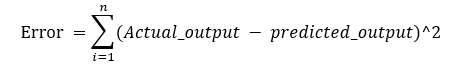
1. Regression sum of squares (SSR)

This gives information about how far estimated regression line is from the horizontal ‘no relationship’ line (average of actual output).



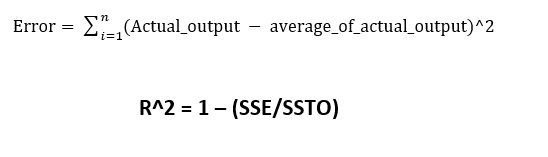
2. Sum of Squared error (SSE)

How much the target value varies around the regression line (predicted value).



3. Total sum of squares (SSTO)

This tells how much the data point move around the mean.



**Is the range of R-Square always between 0 to 1?**

Value of R2 may end up being negative if the regression line is made to pass through a point forcefully. This will lead to forcefully making regression line to pass through the origin (no intercept) giving an error higher than the error produced by the horizontal line. This will happen if the data is far away from the origin.

**Correlation co-efficient (r)**

The correlation coefficient is a measure of linear association between two variables. Values of the correlation coefficient are always between -1 and +1.

**Adjusted R^2 value**

Adjusted R-squared simply penalizes the model for adding more useless variables. It can be calculated as follows:

/var/folders/_v/g0gkxbk53lvdm02dlx_tstvc0000gn/T/com.microsoft.Word/WebArchiveCopyPasteTempFiles/1*hSo8lkdO4TFTpaD_x3noTQ.png

n=no of data points

p=no of independent variables used in the model

**How does Adjusted R-squared work?**

Adjusted R-squared decrease when we add useless variables to our model and increases when we add significant variables. R-squared tells you how well your model fits the data points whereas Adjusted R-squared tells you how important a particular feature to your model is.

**Subset Selection**

**Best-Subset Selection:** Here we fit a separate OLS regression for each possible combination of the *p* predictors and then look at the resulting model fits. The algorithm is broken up into 2 stages:

(1) Fit all models that contain *k* predictors, where *k* is the max length of the models

(2) Select a single model using cross-validated prediction error

**Forward Stepwise Selection** considers a much smaller subset of *p*predictors. It begins with a model containing no predictors, then adds predictors to the model, one at a time until all of the predictors are in the model.

**Backward Stepwise Selection** begins will all *p* predictors in the model, then iteratively removes the least useful predictor one at a time.

**Hybrid Methods** follows the forward stepwise approach, however, after adding each new variable, the method may also remove variables that do not contribute to the model fit.