# Predictive Modeling FAQs

Q1. **Hi, I want to understand how do we calculate the 'error term' in the regression model (Y=α+βX+ϵ)?Is it the standard error of the estimate or the RMSE or MAPE or MAE?**

Ans. Error Term in the regression model is the distance between each point and the linear graph which is a non-constant term. Therefore, statisticians have developed summary measurements that take the collection of residuals and condense them into a single value that represents the predictive ability or performance of the model. And those summary measurements are represented by MAE, MSE, MAPE, MPE. To conclude, the error term in the regression equation is not MAE, RMSE or MAPE.

Q2. **I have 9 independent variables in a dataset and two columns has numbers represented in percentages, if i would like to build a Linear Regression with the variables in % what should i do? Let's say value is 0.20%, should i convert it as 0.2 or is there any other way. And what is the effect of scaling if i convert as 0.2 because two of my columns has a values in Millions. Attaching screenshot for reference. Leads is my Target Variable in this case.**

Ans. There are text analytics (NLTK) and string manipulations (re) libraries (mentioned in parenthesis) in Python. However, an easy way to do this would be to remove the '%' in your excel workbook itself before moving to Python to build a Linear Regression model. For parametric algorithms (like Linear Regression), scaling the independent variables has no effect on the model. The values of the coefficients (estimated parameters) will change but no other statistic of the model should be changing

Q3. **Please let us know, how to display the coefficient values and intercept values of Linear Regression model in Python code.**

Ans. In Python, you perform LR like below:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
predicted_test= lr.predict(X_test)

#R Square using .score from LinearRegression()
score_test= lr.score(X_test,y_test)
score_train= lr.score(X_train,y_train)

#Root Mean Squared Error
from sklearn.metrics import mean_squared_error
rmse_test= np.sqrt(mean_squared_error(y_test,predicted_test)

#Intercept &amp; Coefficients of LR model
lr.intercept_
lr.coef_
```

This has been shown in the video "1.15 Hands-on exercise Multivariate Linear regression model Part 3" at 4:47 minute. You can refer to the code attached.

Q4. **In our exercises, we convert all the categorical variables to continuous variables to perform linear regression? But for this question, the answer is "Predictors must be continuous is not an assumption in linear regression". If so then why didn't we use the categorical variables directly?**

Ans. Linearity is an assumption of Linear Regression. Please refer to this documentation:
https://online.stat.psu.edu/stat500/lesson/9/9.2/9.2.3
Assumptions for Simple Linear Regression Linearity: The relationship between X and Y must be linear. Check this assumption by examining a scatterplot of x and y.
Independence of errors: There is not a relationship between the residuals and the Y variable; in other words, Y is independent of errors. Check this assumption by examining a scatterplot of "residuals versus fits"; the correlation should be approximately 0. In other words, there should not look like there is a relationship.
Normality of errors: The residuals must be approximately normally distributed. Check this assumption by examining a normal probability plot; the observations should be near the line. You can also examine a histogram of the residuals; it should be approximately normally distributed.
Equal variances: The variance of the residuals is the same for all values of X. Check this assumption by examining the scatterplot of "residuals versus fits"; the variance of the residuals should be the same across all values of the x-axis. If the plot shows a pattern (e.g., bowtie or megaphone shape), then variances are not consistent, and this assumption has not been met.
Yes, we do use Categorical variables by encoding them. We cannot use them directly as predictors in Python. We need to encode them. The option ' All Predictors must be continuous' is not an assumption of SLR. Hence, the correct option. Moreover, encoding the variables does not mean they are continuous. They may be binary or label encoded

Q5. **If ln(odds) of an event is 0, what is the probability of that event? Note: 'ln' refers to the natural logarithm (i.e. logarithm to the base)Need more clarity for this question.**

Ans. The following is the explanation of the question.
In this case, we know that ln(P/(1-P) = 0. So, for the equation to be 0 we need to have P = 1-P (as ln(1) = 0). Thus, we can say that P=1-P=0.5.regards,

Q6.
**There are 19998 duplicates are coming after creating the dummies. How to treat such large duplicates as we are loosing a large no. of observations of model building**

Ans The duplicate values might have been generated as after the One Hot Encoding process, the categorical columns have been changed into numerical ones. Earlier, when there was a text in categorical variables it would have been the case that there were no duplicate values. For variables that have certain entries as zero can be treated in multiple ways. It could be thought that those are missing values or junk values and can be imputed with an appropriate method. Else, we can discuss and check with a domain expert that whether it is possible to have a value that is zero for a particular variable. We can also choose to drop such a record from the analysis as it might not make sense to include these. But, the interpretation of the coefficients of the linear regression model should only be done once it is certain there is no obvious correlation between the predictor variables as otherwise, the coefficients become unstable. There will be no problem with the prediction whatsoever. Half bathrooms can be interpreted as a bathroom that might not have a bathing facility but only a sink and toilet. Hope this helps!

Q7. **After performing one hot encoding(using Dummies function) I get up to 24 columns. Now if I plot a heat map for all, I find that for 4 variables out of 24 are high in multi collinearity. Now do I need to perform PCA for the whole dataset and proceed with the problem? How many components can I reduce to?**

Ans.
pandas.get_dummies()
Convert categorical variables into dummy/indicator variables. If you don't use "drop_first" you will get a redundant feature, let's see an example. If you have a feature "Is_male", you use "get_dummies" you will get two features "Is_male_0" and "Is_male_1", but if you look carefully they are redundant/highly correlated, actually you just need one of them, the other one will the exact opposite of the other. In pandas.get_dummies there is a parameter i.e. drop_first, which allows you whether to keep or remove the reference (whether to keep k or k-1 dummies out of k categorical levels). Please note drop_first = False meaning that the reference is not dropped and k dummies created out of k categorical levels! You set drop_first = True, then it will drop the reference column after encoding. You should select enough PCA components to explain enough of the variance that you are comfortable with. Multicollinearity is the presence of a strong correlation between the independent variables. Using a correlation heatmap is a way to visually understand the degree of correlation present within the various variables. Pair plots are also a great way to understand how the various variables (taken two at a time) behave with each other. Lastly, you can check the Variance Inflation Factor (VIF) values of the various independent variables and draw a conclusion about the multicollinearity problem. If you feel that there is still some extent of multicollinearity present in the model (which will be present even after all of these treatments), you can always perform Principal Component Analysis. But do remember that the interpretability of the Linear Regression model becomes tedious when we fit the new independent variables (after performing PCA) into the Linear Regression model.

Q8.
**While performing logistic regression, do I need to perform one hot encoding, if a column has categorical values<< I also used the dummies function here , and ran the model, (number of columns increased to 20 as there are 20 categorical values)>>. In other scenario I also dropped that column and ran the model, (I had less columns now) Both the scenarios give me same numbers in terms of the accuracy and classification report. Can I perform in the above way, or how should It be carried away, Kindly let me know**

Ans.
It is required to convert any column having string values to numerical values before applying logistic regression. Dropping any information (whether it is categorical or continuous) without any logical reason is not advisable and can cause information loss

**Q9.**
**Outliers - All the 3 algorithms (Linear Regression, Logistic Regression & LDA) are susceptible to outliers. Hence, treating them is mandatory. Is my understanding right?**

Outliers can be very informative about the subject area and data collection process. It's essential to understand how outliers occur and whether they might happen again as a normal part of the process or study area. Unfortunately, resisting the temptation to remove outliers inappropriately can be difficult. Outliers increase the variability in your data, which decreases statistical power. Consequently, excluding outliers can cause your results to become statistically significant. So treating/Handling Outlier is necessary.

**Q10.**
**It is not mandatory for linear/logistic & LDA but to compare co-efficient of variable importance scaling is necessary. Is this right?**

Feature Scaling or Standardization is a step of Data Pre Processing that is applied to independent variables or features of data. It helps to normalize the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.
Why and Where to Apply Feature Scaling?
The real-world dataset contains features that highly vary in magnitudes, units, and range.
Normalization should be performed when the scale of a feature is irrelevant or misleading and not should Normalize when the scale is meaningful.
The algorithms, which use Euclidean Distance measure, are sensitive to Magnitudes. Here feature scaling helps to weigh all the features equally.
Formally, If a feature in the dataset is big in scale compared to others then in algorithms where Euclidean distance is measured this big scaled feature becomes dominating and needs to be normalized.

**Q11.**
**LDA is a supervised technique and it clusters observations is also right. Is this correct?**

A clustering technique is always an unsupervised technique where we do not know what the target variable could be. However, LDA involves the presence of target variable and is never used to form clusters.

**Q12.**
**If the features are having different scales, is it good to scale/standardize the data because co-efficient will be comparable and we can rank them as to which variable having high co-efficient is of great importance but interpretability will be difficult in case of linear regression as i cannot say that given the change in X value, y value increases by Beta zero.**
**How do we approach scaling for linear, logistic and LDA pls.**

Scaling (Normalization) & standardization are two different concepts and are not to be confused to be the same.

Scaling will not affect the interpretation of the coefficients. If the independent variables are scaled down to a fraction, the coefficients will also get scaled down by that factor. You can check out the following link for a better grasp on how scaling affects the coefficients: http://www2.kobe-u.ac.jp/~kawabat/ch06.pdf

Standardization is used when an interaction is created from two variables that are not centered on 0, some amount of collinearity will be induced. In simple terms, having non-standardized variables interact simply means that when X1 is big, then X1X2 is also going to be bigger on an absolute scale irrespective of X2, and so X1 and X1X2 will end up correlated. Standardization just removes the collinearity.

Z-score is a standardization technique, not a scaling/normalization technique. Scaling will not affect the interpretation, but standardization will affect it. As mentioned earlier, standardization just removes the collinearity while interaction between variables. Therefore, it is for you to decide what scenario will be best suited for your problem at hand.

There is a great example of scaling that has been covered in the following link. Request you to please check it out: https://scikit-learn.org/stable/auto_examples/inspection/plot_linear_model_coefficient_interpretation.html#the-dataset-wages

**Q13.**
**I require the codes for Complete End to End Linear Regression Problem, which includes the Checking the feature Importance of each & every model, & Plotting the bar Graphs for Performance Matrix to check which model is performing good basis each Key performance Indicators used. Kindly Help me out with this.**

Please check out the following link where the model comparison has been done using a Boxplot for classification: https://www.kaggle.com/pratikasarkar/complete-classification-guide
Boxplot will help you compare the bias as well as the variance error of the performance metric. The above link covers classification but you would also be able to do the same for Regression problems too.

**Q14.**
**How do we decide among LDA vs. Logistic vs Random Forest models which one suits well for binary classification problem, can you pls. give some tips/clues**

Generally Logistic model performs very well on binary data, but it is not always about deciding the best model that might suit the data at first hand, but rather a stride to check multiple models on the data and find the one that gives the best performance metric (accuracy/precision/recall/roc_auc_score - depends on the business requirement which metric needs to be chosen). So, you can't decide which model will perform better on binary/multiclass data before actually building multiple models

**Q15.**
**There is a question on the interpretation of linear regression.**

**Suppose, I have a categorical variable Shelve Location which has three categories Good, Medium, and Bad. I am regressing sales of car seats on some variables among which Shelve location is one. I did one-hot encoding and suppose the equation turned out to be**

**Sales = b0 + b1*X1 + b2*ShelveLoc_Good + b3*ShelveLoc_Medium**

**Is my interpretation below correct?**

**Compared to Shelve location bad the sales will change by b2 if ShelveLoc is good keeping X1 constant**

**Please let me know**

The interpretation will depend on the values of the ShelveLoc_Good & ShelveLoc_Medium. So if you have both of these variables set to 0, then obviously the ShelveLoc_Bad is set to 1, and thus there will be no effect of ShelveLoc_Good & ShelveLoc_Medium. If ShelveLoc_Good is set to 1, then you can interpret that 1 unit change in ShelveLoc_Good will increase(if b2 value is +ve) the sales by b2 units or decrease(if b2 value is -ve) the sales by b2 units keeping all the other variables constant. Similarly, you can interpret for ShelveLoc_Medium.

**Q16.**

**I'm not able to understand the gradient descent concept. I would like to get a better understand on this concept. Kindly request your assistance in this regard?**

A gradient simply measures the change in all weights with regard to the change in error. It can also be thought of as the slope of a function. The higher the gradient, the steeper the slope, and the faster a model can learn. But if the slope is zero, the model stops learning. In mathematical terms, a gradient is a partial derivative with respect to its inputs.
Think of gradient descent as hiking down to the bottom of a valley. This is a better analogy because it is a minimization algorithm that minimizes a given cost function. Gradient descent starts at a point (somewhere around the top of our illustration), and it takes one step after another in the steepest downside direction (i.e., from the top to the bottom of the illustration) until it reaches the point where the cost function is as small as possible.

**Q17.**
**So far in the modules covered, I have seen recall as an important metric in the classification problems e.g. cancer/claims etc. Can you pls. share some examples where Precision is highly important metric.**

Please refer to this article that beautifully explains the precision and recall concepts with an example. Hope you will find this useful.
Precision is a ratio of true positives to total predicted positives.

It is important when you want to be more confident of your predicted positives.

It is used when the occurrence of false positives is unacceptable/intolerable. For example, consider a spam email classifier. Here your incoming email will be classified as spam(class 1) or non-spam(class 0). You'd rather have some spam emails in your inbox than missing out on some regular emails that were incorrectly sent to your spam box. This means you would want your precision to be higher because your False Positives (model predicts it spam but it actually isn't) will be lower. In case, if you keep track of Recall(Ratio of true positives to total actual positives), you would be tracking for those mails that the model predicts not to be spam but are actually spam(FN).
Precision = TP/(TP+FP)
Sensitivity or Recall = TP/(TP+FN)

**Q18.**
**As far as I understand distance based/equation based algorithms require scaling e.g. ANN where we use gradient descent. Similarly, in linear regression to choose best fit line gradient descent is used, then why scaling is not needed as it is also equation based algorithm. Can you pls. help me as I m getting confused here.**

You don't "need" to but it's a very good idea.

There are 2 aspects to find the optimal solution for linear regression. You can solve it via the analytical solution(interpreting coefficients and p-values) or via gradient descent.

If you're using the analytical solution, feature scaling won't be of much use. In fact, you may want to refrain from feature scaling so that the model is more comprehensive. However, if you are using the gradient descent algorithm, feature scaling will help the solution converge in a shorter period of time.

Scaling can help in the faster convergence of the algorithm in case you are using Gradient Descent. But, if your features differ in scale then scaling may impact the resultant coefficients of the model and it can be hard to interpret the coefficients.

Generally, scaling helps in case your number of features are too large as it helps is running model quickly else the starting point would be very far from minima if the scaling is not done in preprocessing.

**Q19.**
**While building linear regression, if multicollinearity exist between independent continuous variables and we if decide to do PCA, should we also include the other encoded variables (originally categorical variables with multiple classes) for PCA. If no, then will there be any problem with the linear model if we combine the Principal Components (PC) identified and the encoded variables. I am asking because, while doing PCA, we would scale the variables and the resulting PCs will be scaled value as well. But the encoded variables which we did not include in the PCA will not be scaled. Will there be any issues in combining these two before performing Linear Regression.**

While you can use PCA on binary data (e.g. one-hot encoded data) that does not mean it will necessarily work very well. PCA tries to minimize variance. So yes, you can use PCA.  It works, but it is just much less meaningful. Moreover, you should not be scaling a label encoded data. It will just give higher weightage to records with higher labelled value(eg. value labelled as 1 will have lower weightage as compared to value labelled as 3 after scaling)

**Q20.**
**If PCA is done before Linear Regression, How do we interpret the coefficients of selected PCs in the model so as to identify the significant variables?**

Once you have done PCA, you would not be able to map any values to specific original variables. Please make sure you handle the significant & insignificant variables before performing PCA.

**Q21.**
**If a transformation is performed on the original variable (not scaled) to address skewness in data, how do we interpret the coefficients? and how to reverse the same after predicting the model.**

Example you are doing a log transformation on a feature X1. Now, if you get a coefficient of 2 for that particular transformed variable, you can say that a 1 unit change in log(X1) will increase the target variable by 2 times.

**Q22.**
**In the Predictive Modelling End to End case study, there are 3 null value codes,**

**1) cars.isnull().sum()**
**2) cars.isnull().sum()/cars.index.size*100**
**3) cars.isnull().sum()/cars.isnull().sum().sum()*100**

**Please explain significance of each and how will we use/interpret outputs of each for a given problem.**

1) cars.isnull().sum() - This gives the total number of null values present in each column.
2) cars.isnull().sum()/cars.index.size*100 - There is no such code in the solution.

3) cars.isnull().sum()/cars.isnull().sum().sum()*100 - This gives the percentage of null values present in each column. This helps take decision to drop the columns that have more that 70 to 80 percent of null values as they would not be very helpful in predicting.