**Feature Selection**

So if we have a high dimensional dataset, we might want to simplify matters by looking for ways to reduce its dimensionality. There are a few ways to proceed. One is Feature Selection, which has to do with dropping features from our dataset. The other methods have to do with Feature Extraction, which more or less keeps all of the features, but looks for a new reference frame with axes along which the data varies the most. Essentially it reorganizes the N features into a set of N new features (which will be linear combinations of the old features). Ultimately, usually, we will drop the least variant of the features of this new reference frame. And so this kind of employs Feature Selection too, in the end. Well, this kind of Feature Selection describes PCA and NNMF, but not t-SNE. One of the advantages of Feature Selection and Feature Extraction is that we eliminate less predictive features (or linear combinations thereof), and so make overfitting our models less likely.

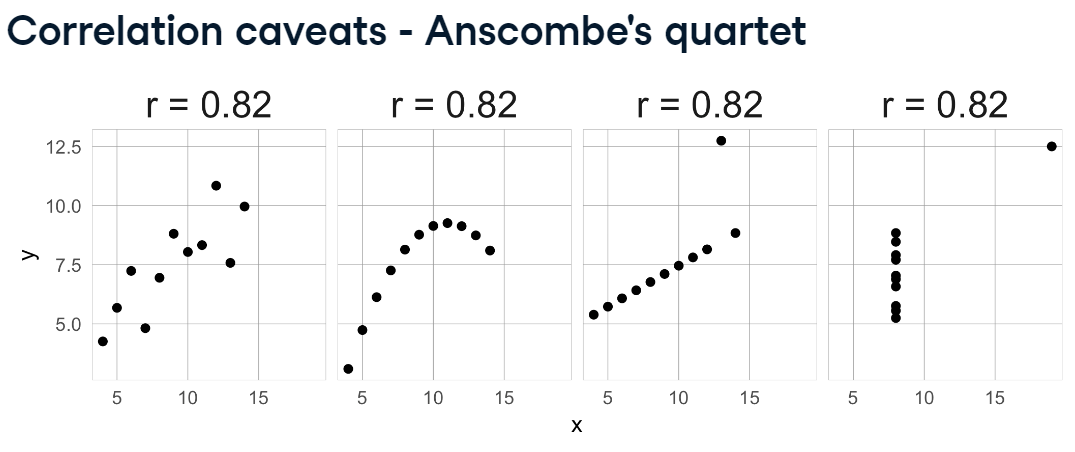
**Feature Selection**

One way to eliminate features is to look for ones that are **redundant**. One example is if we have a bunch of height measurements, taken throughout the year. We may need only one (if the purpose of the model isn’t to predict height, say). Or if we have multiple location data, like city, state, lattitude, longitude. We probably only need the first two or last two. Or if we TFIDF encode a bunch of documents, we may not need to keep all the words, just the ones whose weights surpass some minimum threshold.

Somewhat relatedly, we can also look for features that are **highly correlated** with one another. If two features are highly correlated, then we can just keep one of them w/o sacrificing too much information. For instance, saw this nice graph on Datacamp. The ANSUR dataset has a large number of categorical and numerical features. Guy ran PCA (next file) on the numerical ones and then concatenated with the categorical ones. Here he plotted the first two PCA components in x, y directions, and then colored according to gender. Can see that gender is fairly highly correlated with pc1 and pc2 as we can almost linearly separate the genders with a plane. So in might make sense to drop Gender as a feature, since we can *almost* express it in terms of pc1 and pc2.



Besides this, we can find correlations either using pandas’ df.corr() method. Or we can use seaborn’s pairplot method (guess we’d use with a scatter plot) to visually inspect the correlations. Have to be careful with correlations though. The picture below shows four relationships with the same correlation coefficients. Normally, we wouldn’t remove (I don’t think) a feature only 82% correlated with another. So I don’t think I’d remove the first one. But we definitely could remove the second and third; the parabola is perfectly correlated, and so is the line, sans the single outlier. As for the last one, I’d say we could remove the x-axis variable, as it’s constant, but not the y-axis guy?

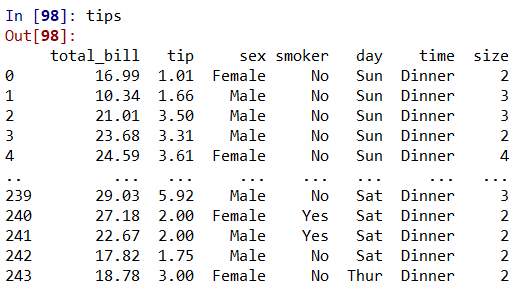


Another option is to eliminate features with less than some **minimum** **low variance** threshold. Probably better to use relative variance = var/mean, rather than just var. Features that don’t change can’t be predictive. Certainly if a feature’s variance were *actually* zero, then it could be removed. But I’d be wary of removing it otherwise, as really, we *should* be scaling all features to unit variance, so wouldn’t this critique be vacuous?

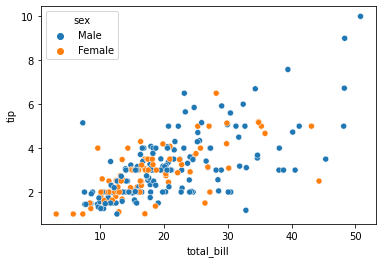
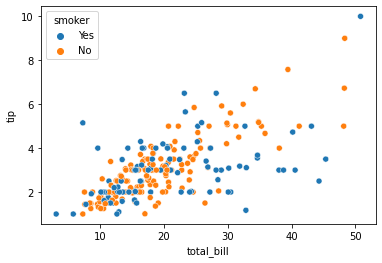
Another tool is to employ decision trees, and better, ensemble methods employing decision trees, like random forest, or gradient boost, etc. Then we can calculate a feature’s importance via the **feature\_importance** property of the tree. This more or less gives us a fraction of the total entropy/gini impurity reduction (or mean square error reduction) that is due to the given feature. Can see those files for more on this. Another option, if doing regression, is to do **Lasso Regression**, and keep increasing the λ penalty. This will make the coefficients of less predictive variables go to zero. Be sure that the features are normalized though. Regression coefficients can only be compared when they’re multiplying the same range of numbers. The advantage of eliminating features that aren’t important is that you’re eliminating features that probably aren’t causally related to the result, and which are therefore just adding noise to our data, which we don’t want to fit to, and which is counterproductive to fit to. But in general, you have to be careful when eliminating features. Can always just see if it helps to eliminate a feature, and if not, then restore it. Another thing to keep in mind when doing feature elimination is that if we should eliminate poorly predictive features one by one and retest the model. When you eliminate a super poorly predictive feature, a priorly poorly predicting feature may become important. Finally, some guy points out that we can combine models’ feature\_importance predictions (random forest regressor, gradient boosting regressor, lasso regressor, etc.) and use a simple voting threshold procedure to determine which feature to keep or drop.

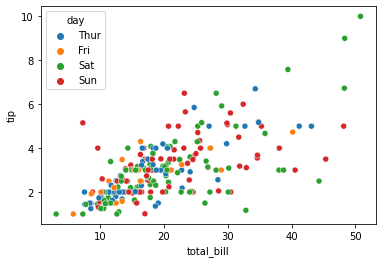
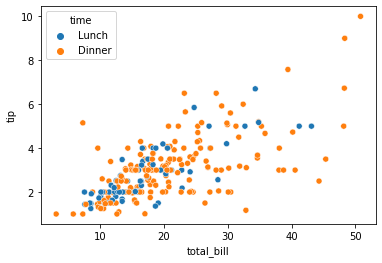
**Example**

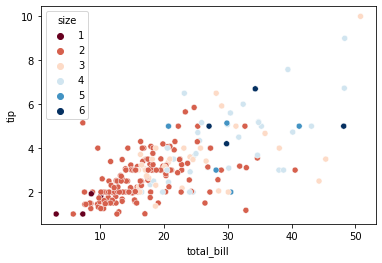
For example, I took the tips dataset,



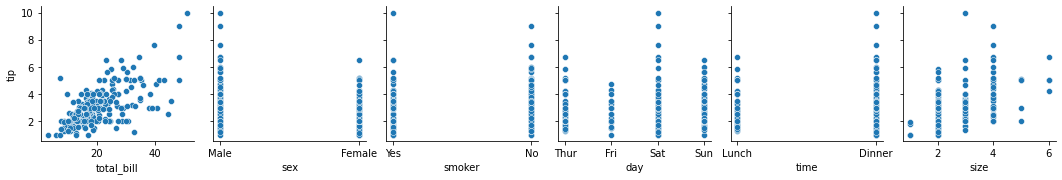
and tried to predict the *tip* column from the other columns. I graphed tip vs. total\_bill in seaborn, and changed the colors of the plots according to the values of the other columns.

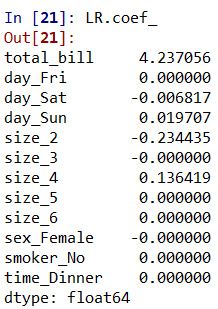
 



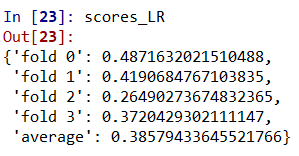
The graphs suggest that total\_bill is the driving factor, as when we look at tip vs. total\_bill with any column value held constant, we don’t seem to get anything substantially different than what we get with any other column value held constant.



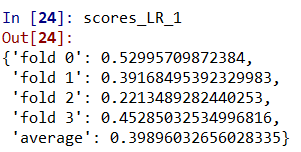
When we do Lasso regression (α = 0.02, the best value), we find the following coefficients,



Can see that only total\_bill seems to matter. When do cross-fold validation on the entire dataset, we get:



When we do it on just the total\_bill column, we get:



which illustrates that discarding columns can often be the way to go.