Table of Contents

[Motivation 2](#_Toc469965341)

[Data Preparation 2](#_Toc469965342)

[Transformations 2](#_Toc469965343)

[Dummy Variables 4](#_Toc469965344)

[Feature Selection 5](#_Toc469965345)

[ANOVA for Univariate Feature Selection 5](#_Toc469965346)

[Mutual Information 7](#_Toc469965347)

[Multi-Collinearity 8](#_Toc469965348)

[Model Fitting 11](#_Toc469965349)

[Backward Stepwise Regression 11](#_Toc469965350)

[Regularization 14](#_Toc469965351)

[Model Interpretation 15](#_Toc469965352)

[Odds Ratio 15](#_Toc469965353)

[Prediction Simulation 18](#_Toc469965354)

[Appendix 1: Non-Linear Features 20](#_Toc469965355)

[Mutual Information 20](#_Toc469965356)

[Weekday Percent 20](#_Toc469965357)

[Appendix 2: Regularization 24](#_Toc469965358)

[Full Model (Over-Fit Model) 24](#_Toc469965359)

[L2 Regularization 27](#_Toc469965360)

[Appendix 3: Bias/Variance 29](#_Toc469965361)

[Out of Box Model 29](#_Toc469965362)

[Learning Curves 30](#_Toc469965363)

[Validation Curves 31](#_Toc469965364)

[Counter-Example 32](#_Toc469965365)

[Appendix 4: Feature Impact Analysis 33](#_Toc469965366)

# Motivation

Our motivation in this exercise is slightly different from the last two. Whereas previously the goal was to target a high f1 or accuracy score, here we’re focusing on interpretation. As a result we’ll spend more time focusing on interpreting model output instead of making a better model.

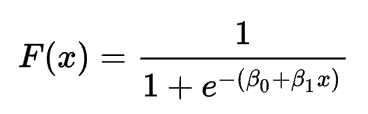
# Data Preparation

### Transformations

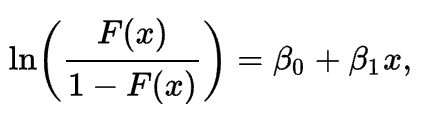
As before, we’re going to convert data types and impute null values to means. However, **we won’t log- or exp-transform our features** because doing so makes interpretation confusing. Take this fake example where we relate sugar consumption to the risk of diabetes:



We can make a statement like, “a an increase of sugar consumption from 31 to 33 lbs leads to a 10% increase in the risk of diabetes.” But the statement changes at various levels of x, because the curve is not linear. The logistic or sigmoid function on which this curve is based is this:



If you transform the above equation, you get a linear equation set against the “log odds” of an outcome:



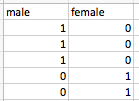
Interpreting the log odds, you can make an alternative statement like, “a one unit increase in annual sugar consumption leads to one unit increase in the log odds of having diabetes,” because the relationship between x and log odds (blue line) is linear. A one unit increase in x always has the same impact on the log odds of an outcome.



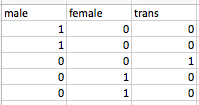
We will go into log-odds in more detail in the model interpretation section of this guide.

### Dummy Variables

It’s also worth noting that since we’re focused on interpreting the model, we’re going to do one-hot/dummy encoding but remove one of the categories. The reason for this include interpretation, [multicollinearity](https://en.wikipedia.org/wiki/Multicollinearity#Remedies_for_multicollinearity), and the [dummy variable trap](https://en.wikipedia.org/wiki/Dummy_variable_(statistics)). Think about the case where there’s just two categories, male and female. We represent both by selecting just one of the fields, because using both is redundant (when a person isn’t male, they’re necessarily female):



Now let’s say we introduced a new category, transgender, the same logic applies. When a person isn’t male or transgender, they’re necessarily female. Having three variables representing three categories leads to the above problems.

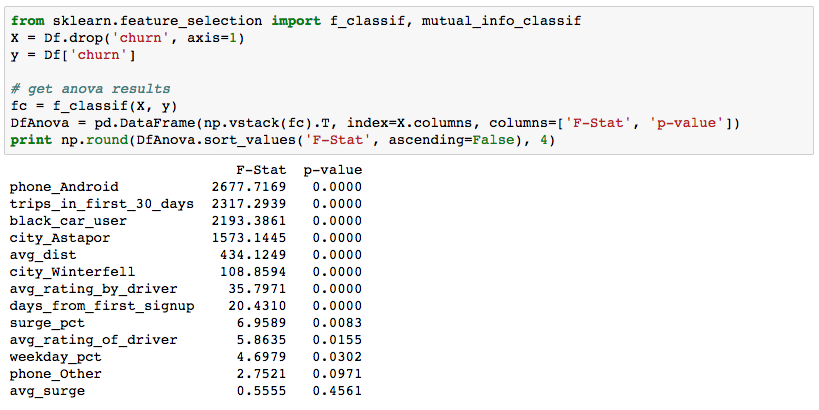


This video provides further explanation of why you drop one of the categories when doing dummy encoding: <https://www.youtube.com/watch?v=9yTui_LoSOc>.

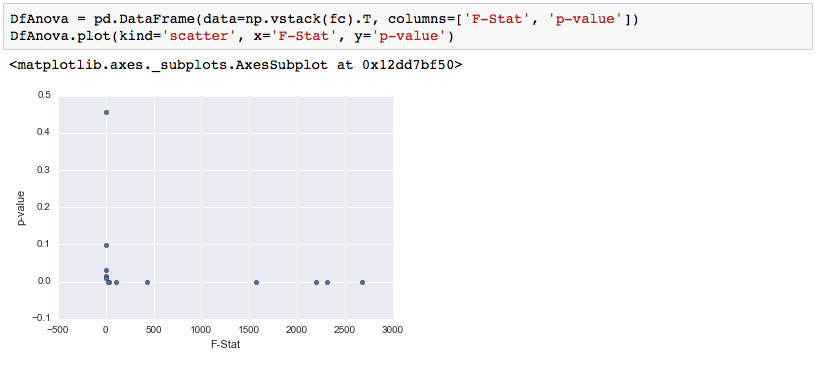
# Feature Selection

### ANOVA for Univariate Feature Selection

One measure of the linear strength of a relationship is the F-Statistic from ANOVA. It tells you the strength of the relationship between both categorical and continuous variables against your target class. If you suspect a non-linear relationship, you can use [mutual\_info\_classif](http://scikit-learn.org/stable/auto_examples/feature_selection/plot_f_test_vs_mi.html#sphx-glr-auto-examples-feature-selection-plot-f-test-vs-mi-py) instead.



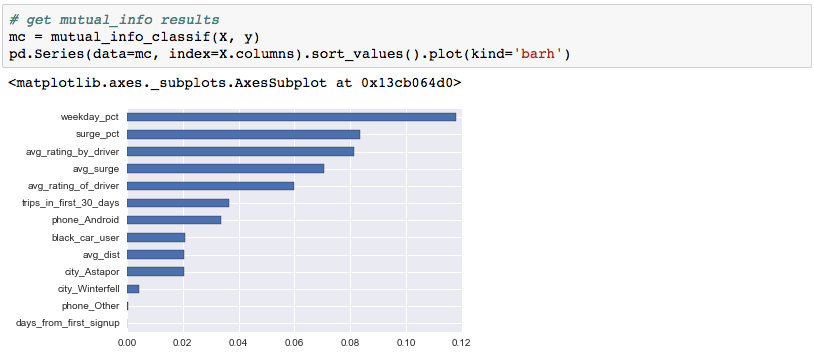
The results show an F-Statistic which is a test statistic on the F-Distribution, and p-value, which tells you how significant a feature is or isn’t. High F-Statistics are good and low p-values are good (usually below 0.005).



Here we see phone\_Android, trips, black\_car, city\_Astapor, avg\_dist are strongly related to churn (very high F-Stat). Avg\_surge, phone\_other, weekday\_pct, and avg\_rating\_of\_driver are not strongly related (high p-value and low F-Stat).

### Mutual Information

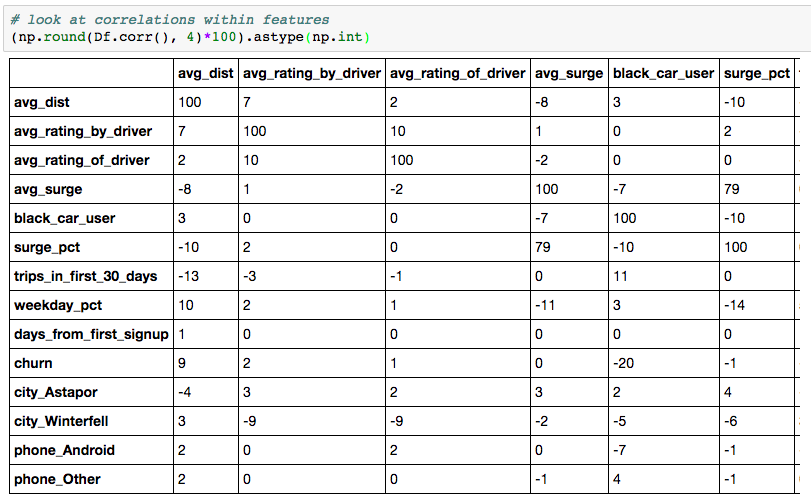
Another feature selection technique that captures non-linear relationships (good for Random Forest and Neural Nets) is Mutual Information. You can read more at the link above, but it’s worth noting that Mutual Information and ANOVA point to very different features.



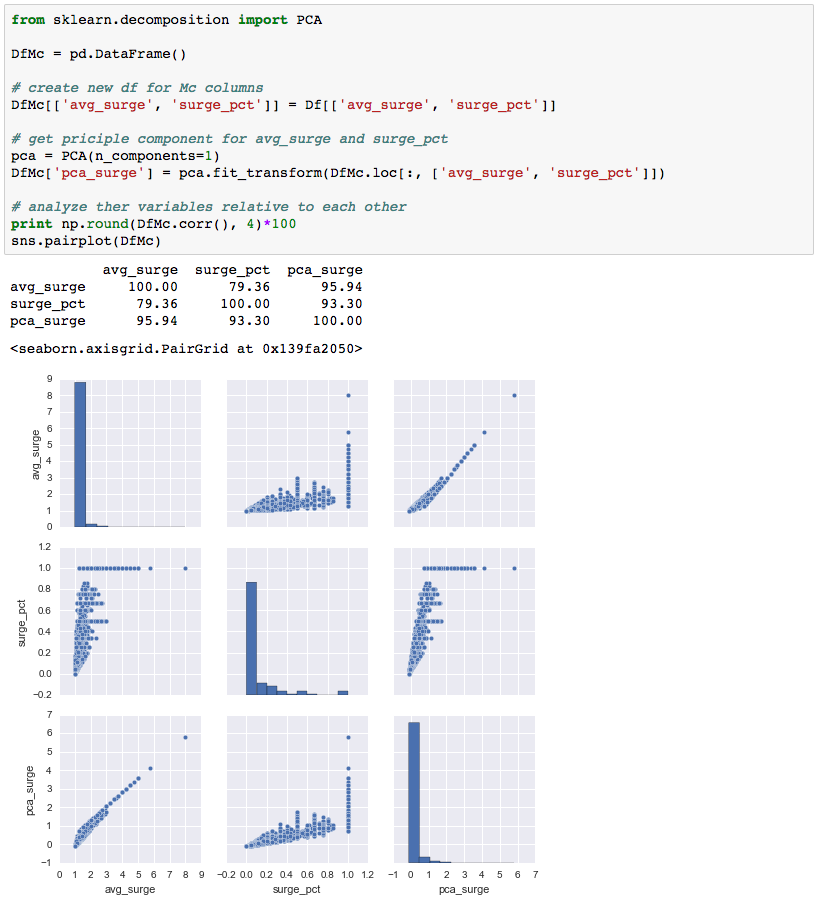
We will go into more detail on how to capture these non-linear relationships in Logistic Regression at the end.

### Multi-Collinearity

In model building there’s an issue called multi-collinearity, in which two highly correlated variables “steal signal” from each other. Because the variables are so related to each other, the model doesn’t know whether to attribute predictive value to one or the other. If a customer never has surge pricing, then avg\_surge = 1 and surge\_pct = 0. You can see the correlation coefficient is very high between these two at 79%:



One way to address multicollinearity is to create a feature that that combines the two original ones, as with principle components analysis which reduces the two features to just one shared one. When you do this, you find the PCA feature is highly correlated with both surge features, so serves as a potentially good go-between:



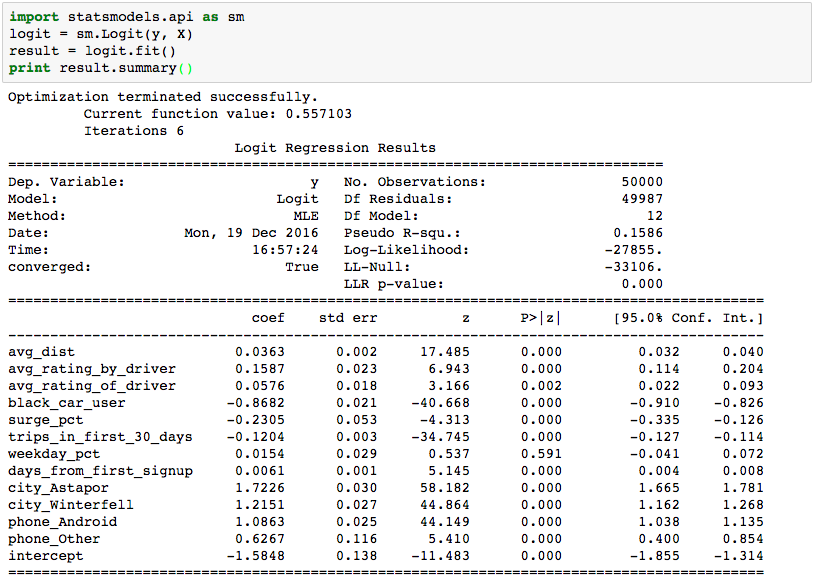
But since avg\_surge is not strongly related to churn and the principle component of avg\_surge and surge\_pct isn’t either, we will drop it instead of creating a feature that captures both surge variables:



# Model Fitting

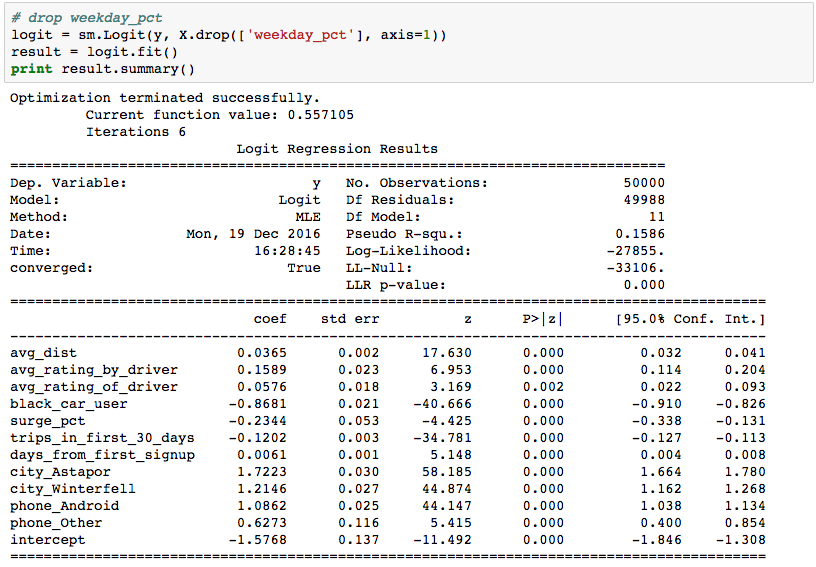
### Backward Stepwise Regression

Backward Stepwise Regression starts with the **full model**, where all features included, then takes features away one-by-one until all of the features fit a certain criteria. **Instead of using scikit-learn, we will use statsmodels** because it has many more diagnostics for model interpretation.

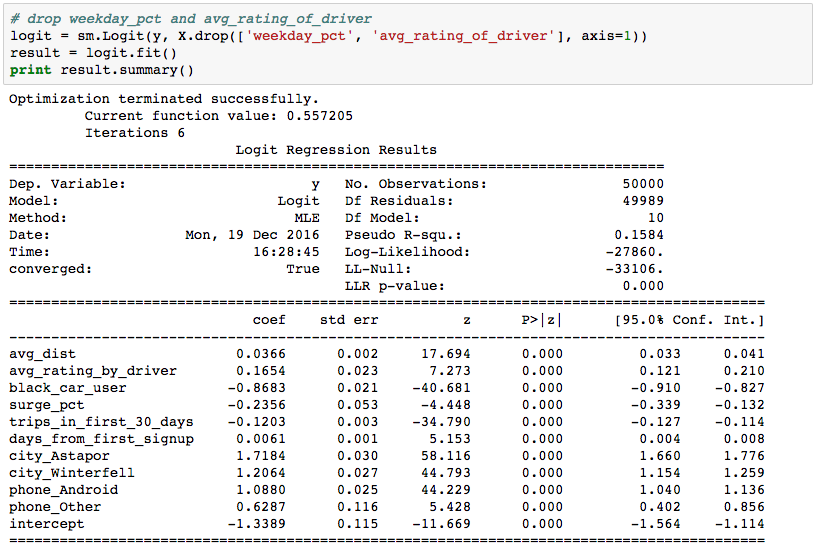


**LLR p-value** tells you whether or not the model is statistically significant and is derived from Log-Likelihood and LL-Null. Since the value is very close to 0, we know this model is predictive.

The **Z-scores** and **p-values** for each of the coefficients tells you how significant each feature is. High Z’s and low p’s (close to 0) are very good. Immediately we notice weekday\_pct has a very high p-value, so we drop it:



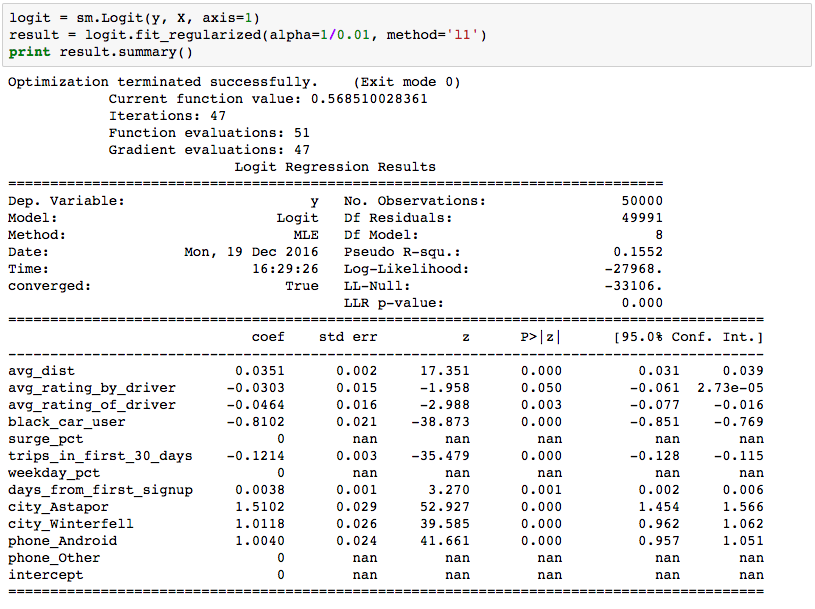
Notice that LLR p-value is unchanged and Log-Likelihood is exactly the same, so dropping the feature didn’t have a material impact on our model. Let’s try this one more time with avg\_rating\_of\_driver:



Notice that Log-Likelihood is mostly unchanged (only 5 higher). Also notice the most significant features, the ones with the highest Z-scores, are **city, phone, black\_car\_user, and trips\_in\_first\_30\_days.**

### Regularization

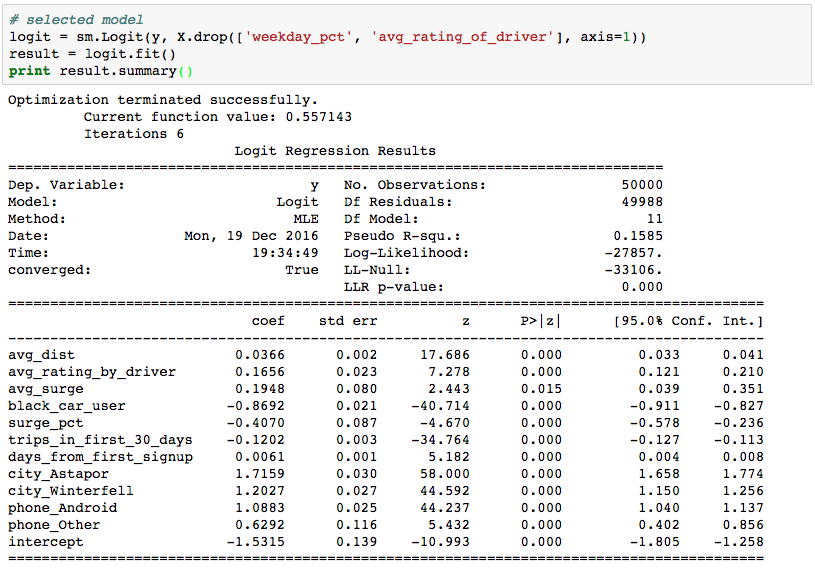
Another approach we can take to feature selection is regularization. Instead of manually adding and removing features, we can penalize the cost function to restrict the weight it assigns to features. We will go into much more detail with this in Appendix 2, but for now you can see that the regularized model still favors the features we originally identified as important using **ANOVA**:



# Model Interpretation

### Odds Ratio

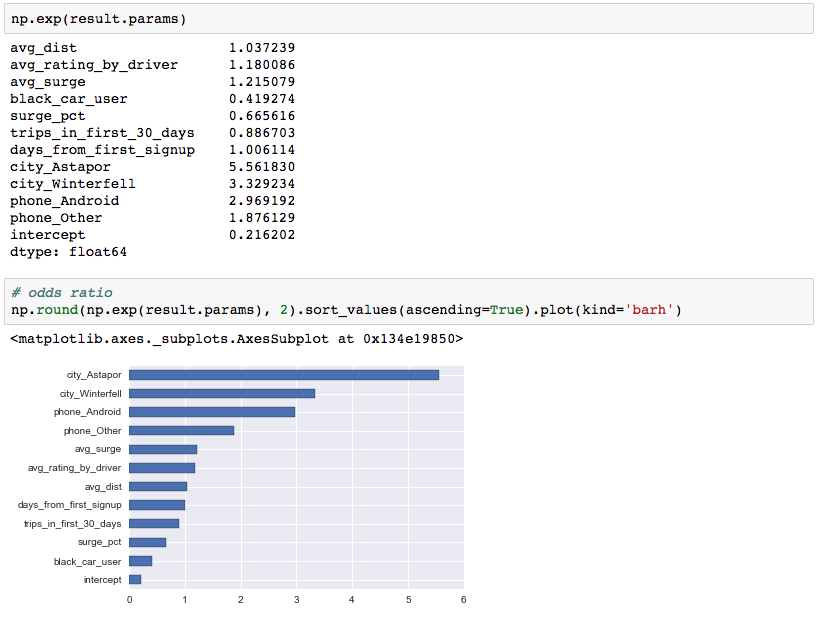
The most important tool for interpreting Logistic Regression model output is the **odds ratio**. The Logistic Function isn’t linear, but the log-odds of it is. You should read about the Odds Ratio in detail [here](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/odds_ratio.htm).



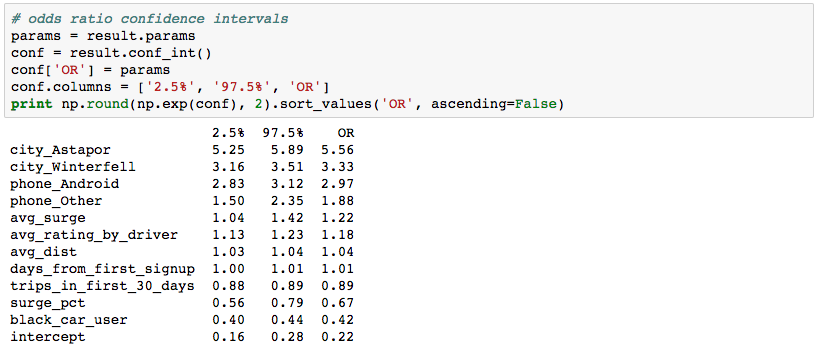
phone\_Android has coefficient = 1.0883. The way to read this is, all other factors held equal, Android users are exp(1.0883) more likely to churn than iPhone users (the base class in this example). That is, their odds of churning are 2.969 versus iPhone users.

trips\_in\_first\_30\_days has a slightly different interpretation since it’s a continuous variable. The way to read this is, all other factors held equal, a one unit increase in trips leads to an exp(-0.1202) increase in the likelihood of churn. This translates to 0.8867 increase or -0.1133 decrease in the odds of churning.

You can get a sense of which features most affect the odds of churn by looking at the full set of odds ratios. Keep in mind binary features can only affect odds by 1x, whereas continuous ones such as trips\_in\_first\_30\_days have a multiplicative effect. For better understanding of what this means, compare difference in p(churn) between different levels of trips and phone\_Android in the Prediction Simulation section.

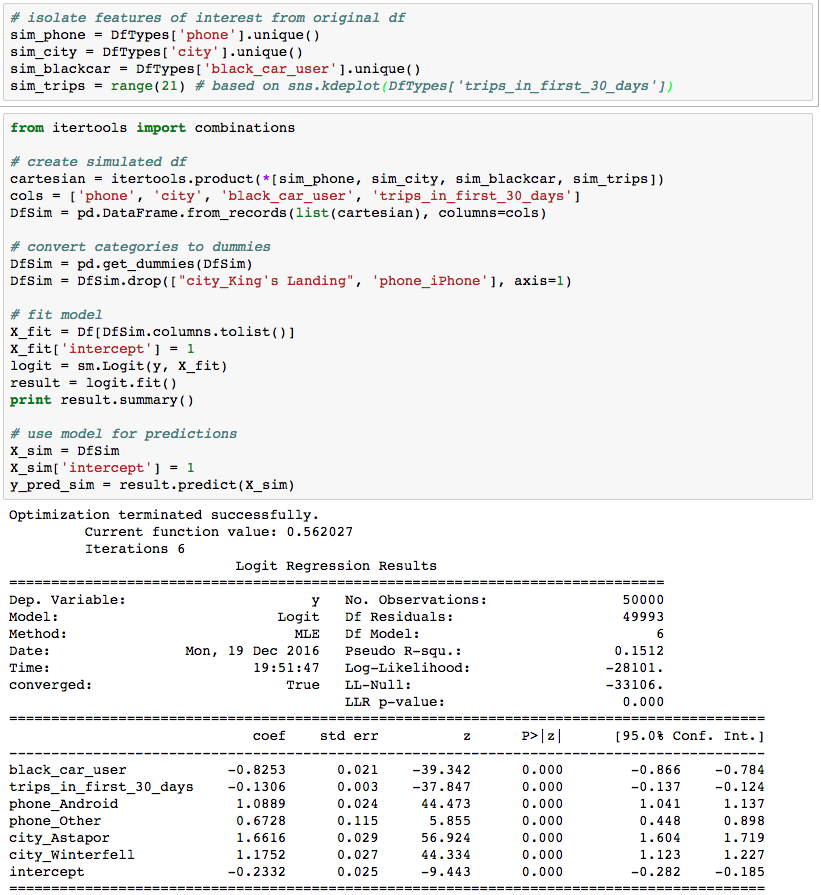


You can also get 95% confidence intervals of the odds ratio estimates:

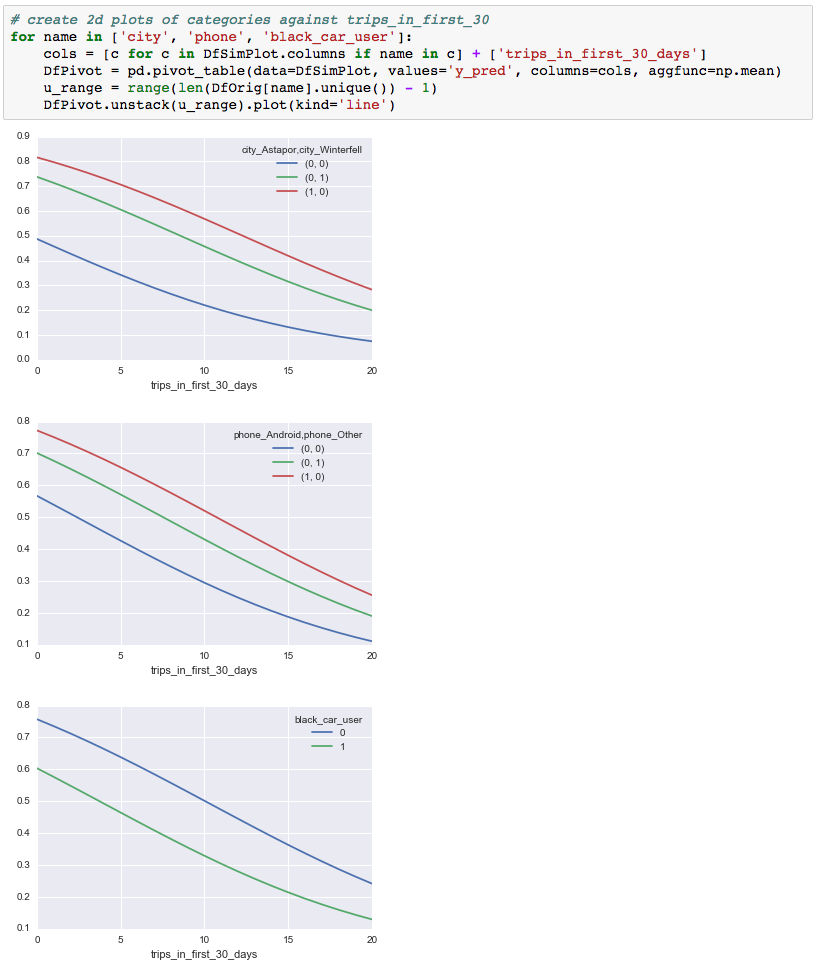


### Prediction Simulation

Another less confusing but also less descriptive tool for understanding the factors that influence churn is by simulation. Using the **itertools** package, we can create the simulated data using a fea key features (like black\_car\_user, city, phone, and trips\_in\_first\_30\_days), and plot them.



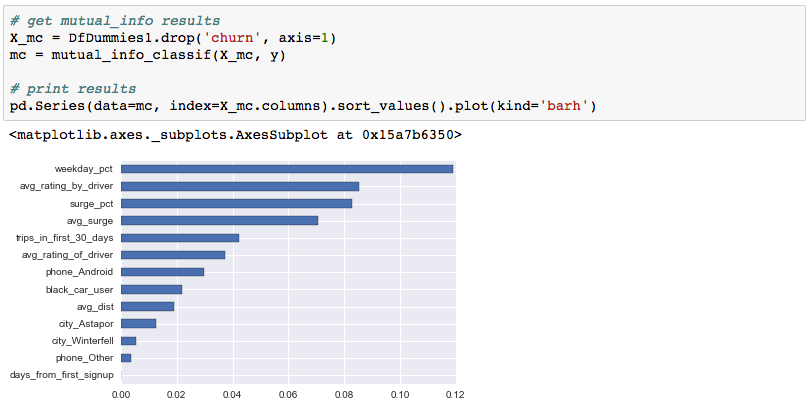
By pivoting the predictions dataframe on trips\_in\_first\_30\_days and our categorical fields of interest, we can see how much our churn predictions change for different levels of each:



# Appendix 1: Non-Linear Features

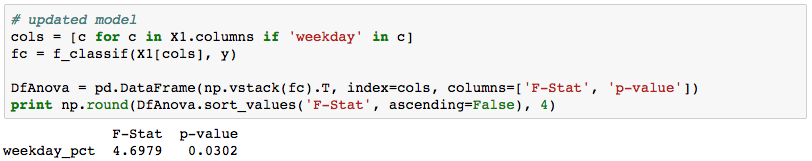
### Mutual Information

Earlier in this exercise we looked at the direct relationship between each feature and churn by using ANOVA and Mutual Information. **ANOVA** captures the strength of the linear relationship between the two, whereas **Mutual Information** captures the non-linear relationship. While in our first pass we ignored the fact that some features had very high Mutual Information Scores but low ANOVA F-Statistics, we will now look into those other features to see if we can describe their relationship to churn.

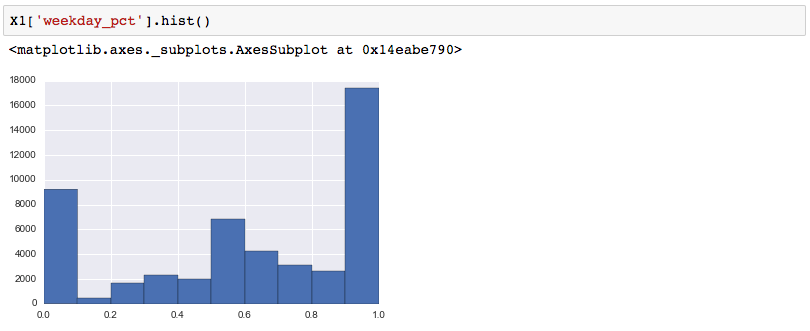


### Weekday Percent

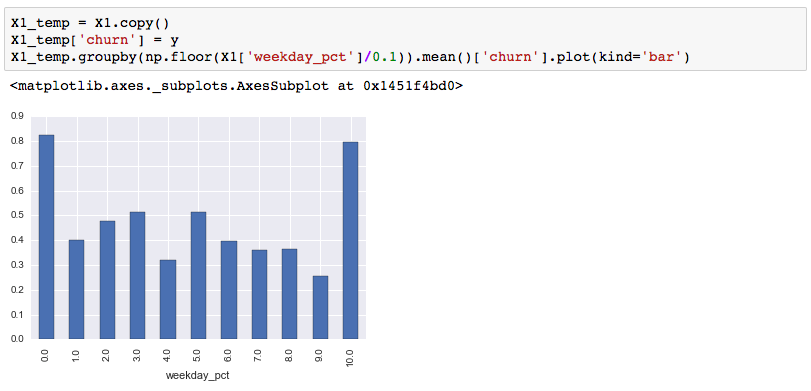
Revisiting ANOVA with weekday\_pct we find an F-Stat and p-value of 4.7 and 0.03, respectively:



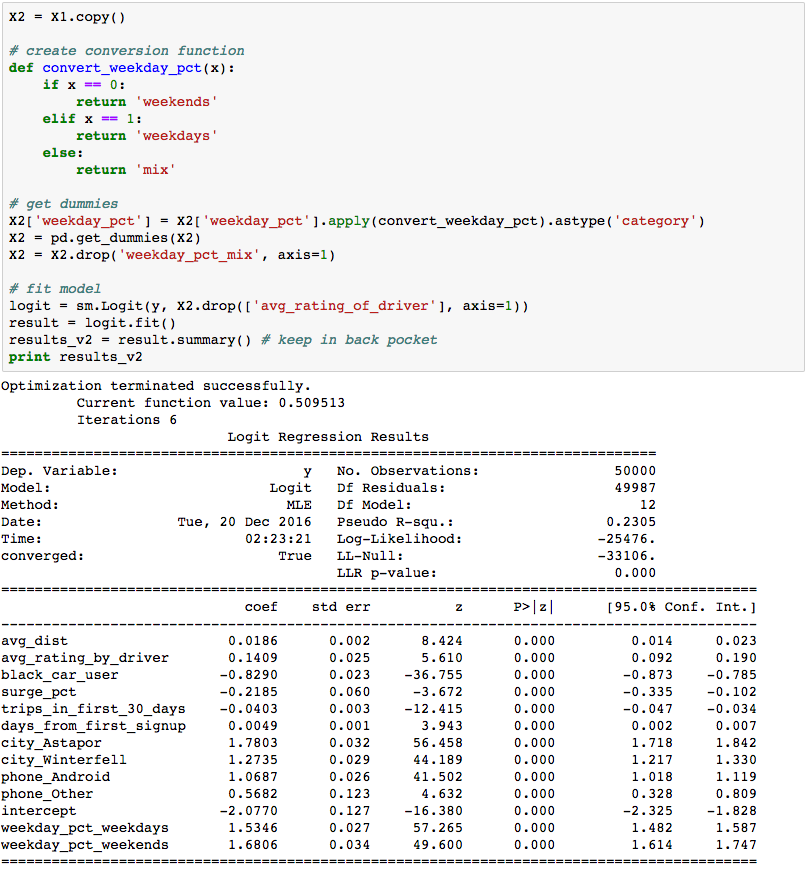
However, the Mutual Information Score suggests there’s a very strong non-linear relationship between weekday\_pct and churn. Looking at the distribution of weekday\_pct it’s clear most values are 0 or 1.



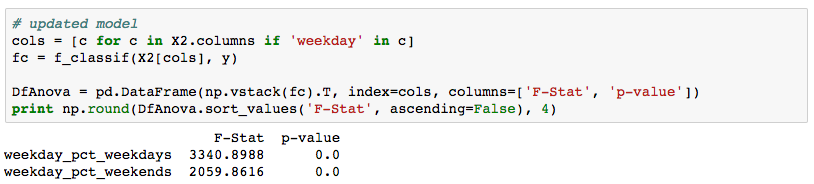
A comparison of churn at each level of of weekday\_pct further supports that not only is the distribution different, but % churn for each group is different:



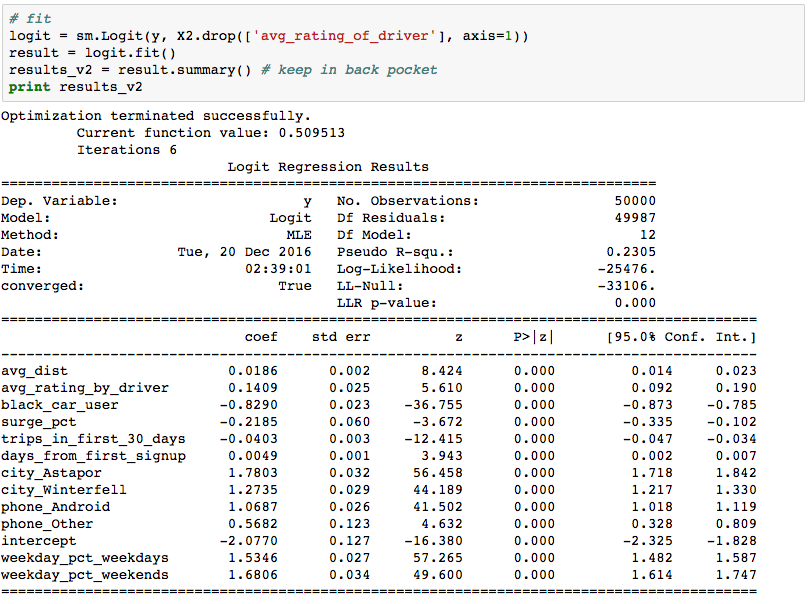
By transforming weekday\_pct to a categorical variable, we might be able to get more information out of it. My hypothesis is that regular customers are rarely 100% weekday or weekend users, but that the customers who have 0 or 1 scores only took 1 weekday or weekend ride, ever, hence the higher churn rate:



This hypothesis pays of huge. Not only are the F-statistics associated with this new dummy variable large:



The log-likelihood of the new model shows a big improvement:

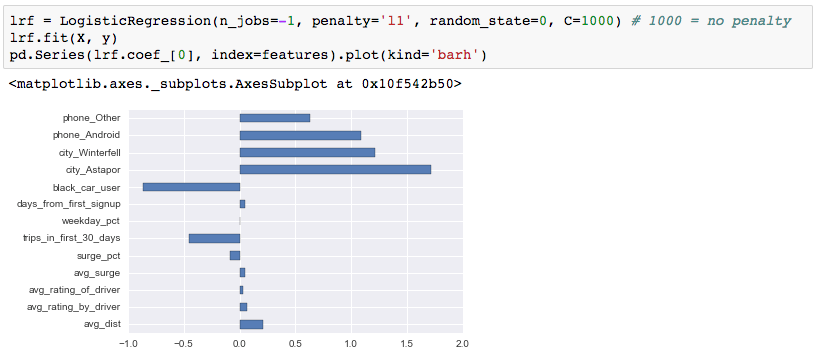


And Accuracy and F1 take a big jump from 71.9% to 75.5% and 79.1% to 80.9%, because we have found a better way to describe the relationship between features and target through thorough, investigative feature engineering.

# Appendix 2: Regularization

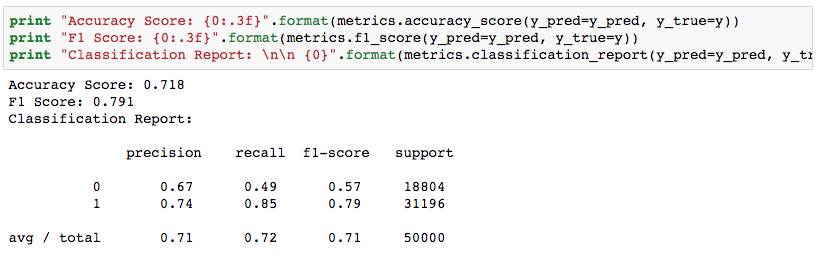
### Full Model (Over-Fit Model)

Taking a look at model coefficients, we see every feature has a large effect, with binary variables having a particularly large one:



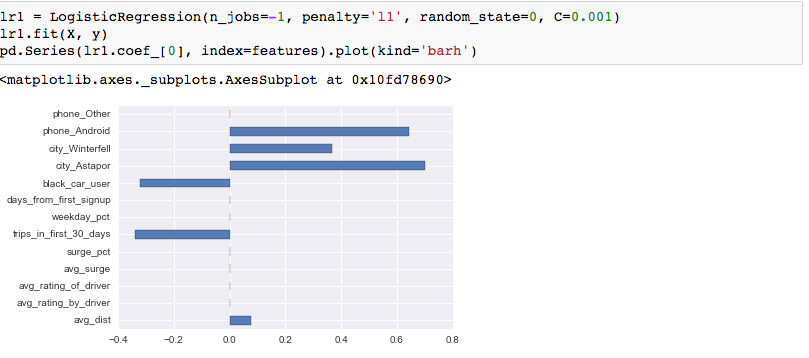
It’s important to note that in the unpenalized model, every feature has “a say in the outcome.” So every feature has a coefficient. It’s also important to note that since the continuous features are on a larger scale than the binary ones, which are on a 0-1 scale, they have smaller coefficients.

Accuracy and F1 for this model are 71.8% and 79.1%, respectively. It’s also very worth noting that precision (accuracy of guesses) is 74% whereas recall (% of target class observations you identify) is 85%. This will be important as we regularize.



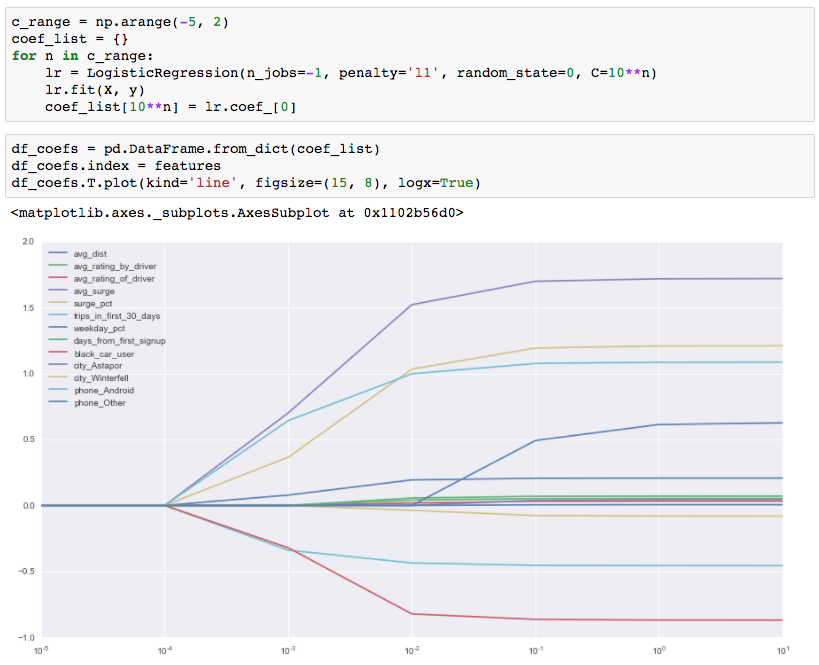
L1 Regularization

The goal of the regularization parameter is to stop the model you’re using from overfitting the data. L1 regularization in particular, aggressively penalizes coefficients such that some get suppressed to 0. In the above case we had a model where every feature had a coefficient and every coefficient was high. By penalizing the model for giving every feature a high coefficient, we can avoid overfitting, and can accomplish this by making the parameter C relatively low:



Settling on some value like C=0.001, we can see the factors that positively correlate with churn are: living in Astapor and Winterfell, having an Android phone, and taking longer distance rides. Features that negatively correlate are: using a black car and taking more trips in the first 30 days. Based on these findings alone, you can start to make suggestions on how to aggressively promote for certain cities, investigate the Android app experience, incentivize more trips in the first 30 days, or give discounts for longer distance rides… But back to regularization.

More generally, we can see the effect of changing the parameter C for each feature by iterating through C values:



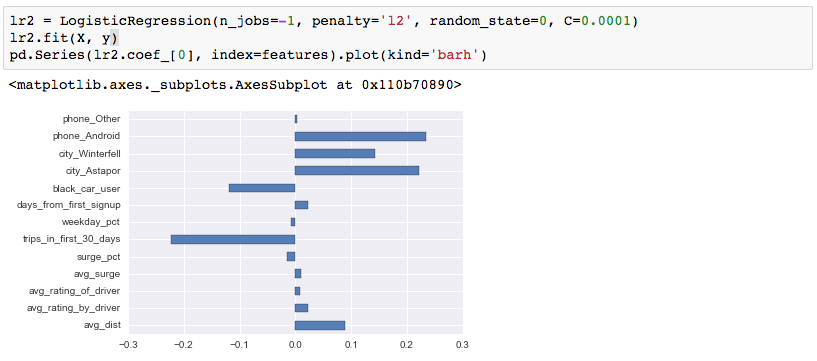
In the above example, the regularization parameter C takes strong effect between 10^-4 and 10^-3, or 0.001 and 0.01.

Overall, this model is 1% less accurate than the overfit model, but has 0.5% better F1. Most notably though, precision fell to 70% and recall went up to 93%. How do you interpret that?

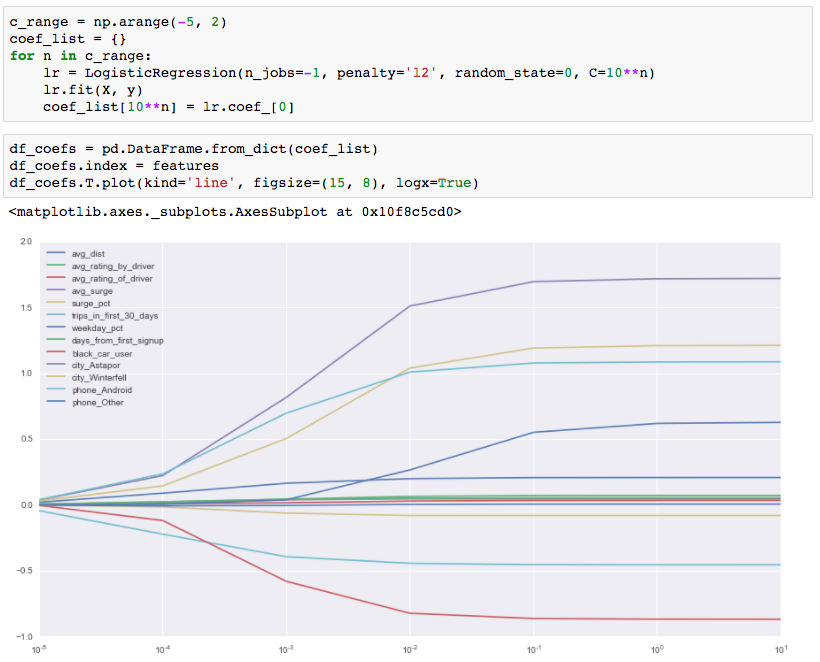


### L2 Regularization

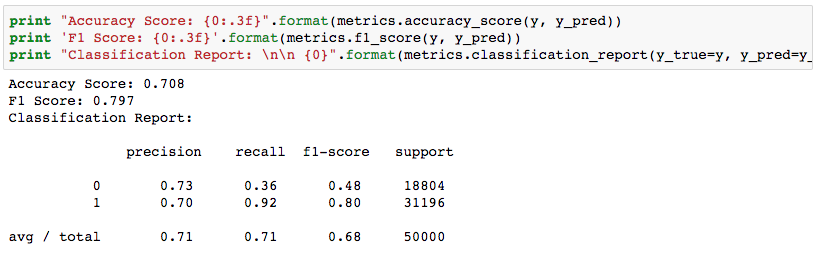
Similar to L1 Regularization, the L2 penalizes a model for overfitting, but does so more “smoothly.” Instead of forcing some coefficients to 0, it just depresses all coefficients to some lower value simultaneously. You can see none of the coefficients have been zeroed out:



And that coefficients smoothly climb into their full-model values as penalization gets smaller:



As with L1 Regularization, precision is lower and recall is higher than with the full model.



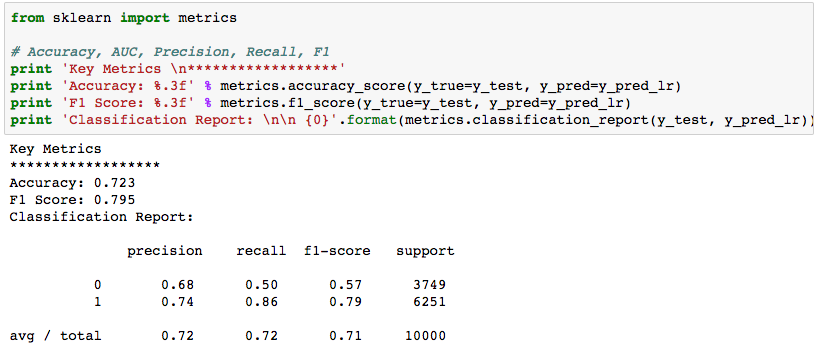
# Appendix 3: Bias/Variance

### Out of Box Model

As before, running the model on a single test/train split of 20%, we find that the model is around 72% accurate between test and train.



It also has an F1 of 79.5% and a very high recall—it captures 86% of all cases of churn:



### Learning Curves

Learning Curves are much more meaningful for a model like Logistic Regression because we expect test and training accuracy to be near each other. Versus Random Forest which explicitly over-fits on test data. Here we find that after 15k training examples, differences between train and test accuracy converge:



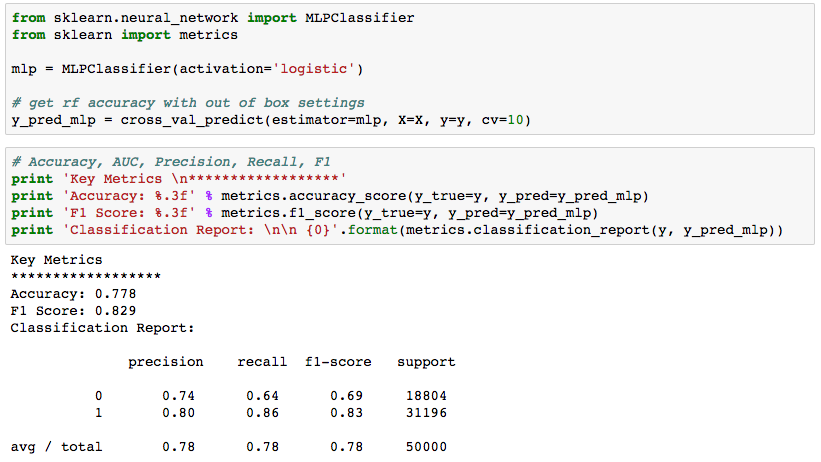
### Validation Curves

Similarly, the validation curve hits peak accuracy at C=10^-1, and there isn’t much difference in test/train for various levels of C. Meaning there’s good bias-variance tradeoff:



### Counter-Example

Grid search is good for finding optimal parameters, but it’s not the best for accuracy. If seeking more accurate results, **the best thing you can do is find a better model or create an ensemble of models to represent your problem**. After you have found the best model, the next thing to focus on is feature engineering. To make the point, if we wanted an accurate model, we could have skipped Logistic Regression and used a Neural Network instead:



Accuracy is 5.9% better and F1 is 3.8% better, right out of the box.

# Appendix 4: Feature Impact Analysis

[Defer to Regression Lesson]