INSTALL IT!

Estimating User conversion probability

Background

In the mobile software industry advertising is a vital tool to increase the user base. A common model for ad placement is known as real-time bidding. In real-time bidding, ad-networks send present companies with the choice of showing their ad by bidding for the ad space. Every time they receive a bid request a decision needs to be made on whether they want to show the ad to the user and how much they are willing to pay. As a Data Science fellow at Insight Data Science, New York, I was fortunate to be paired with one such mobile gaming company that is interested in increasing the effectiveness of their bid strategy.

Objective: Estimate the probability that a user installs the software when they are shown an ad.

The objective is clear, given some data that is presented at the time of the bid request, should the company place the bid? The answer to whether one can predict buyer behavior is a challenging one especially if I am tasked to define it in absolute numbers of installs. If you don’t believe me then take a look at the amount of hoarders we have, if ads were 100% effective you and me would most certainly be part of the club. For me it was important to realize that the ad business is naturally a low conversion problem, so why not look at the relative increment that results from implementing my model? In the ad industry such measures are known as the gains and lifts. The lift in particular defines how much better your model is in predicting conversions versus just randomly picking.

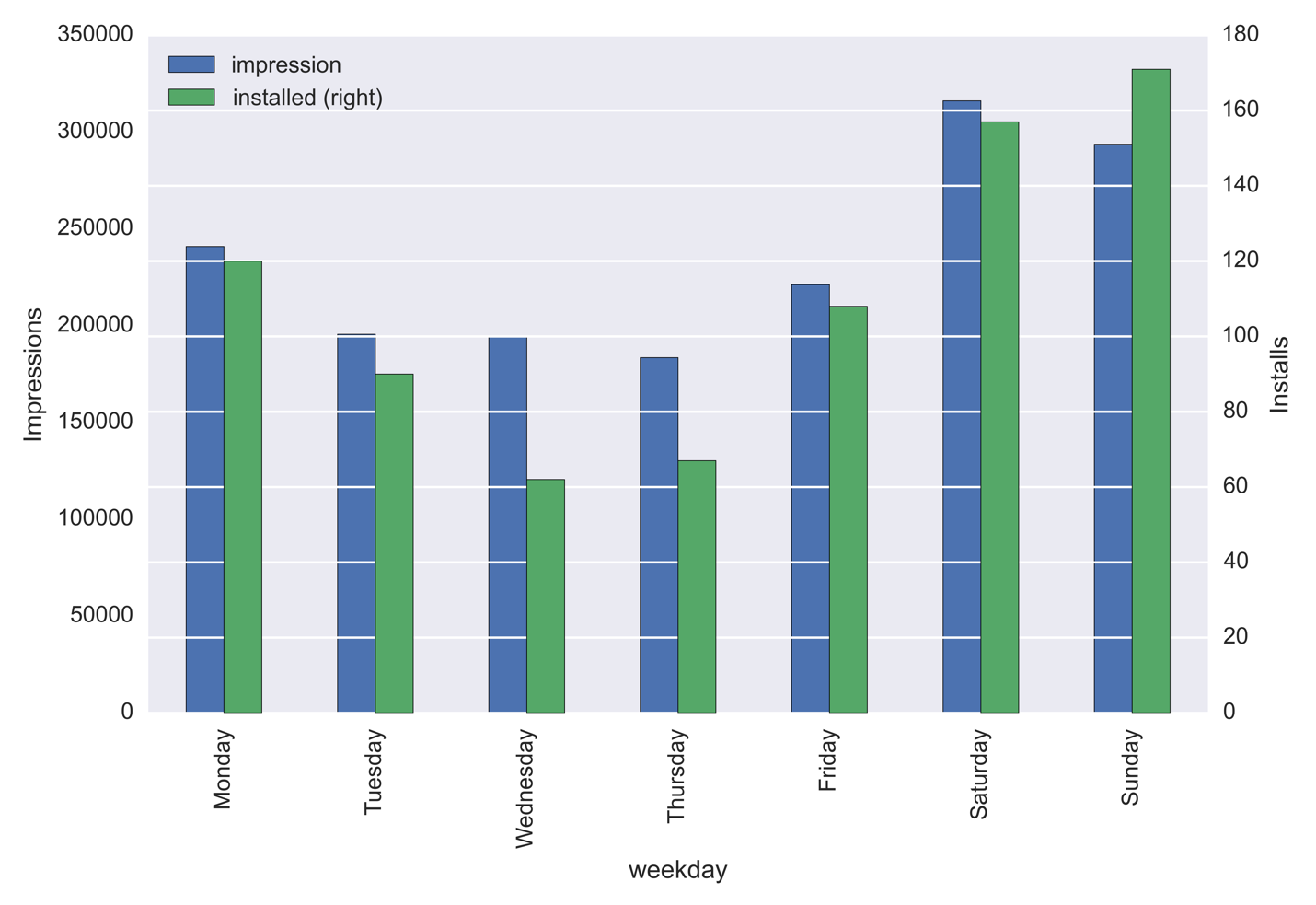
The Data and Feature Exploration

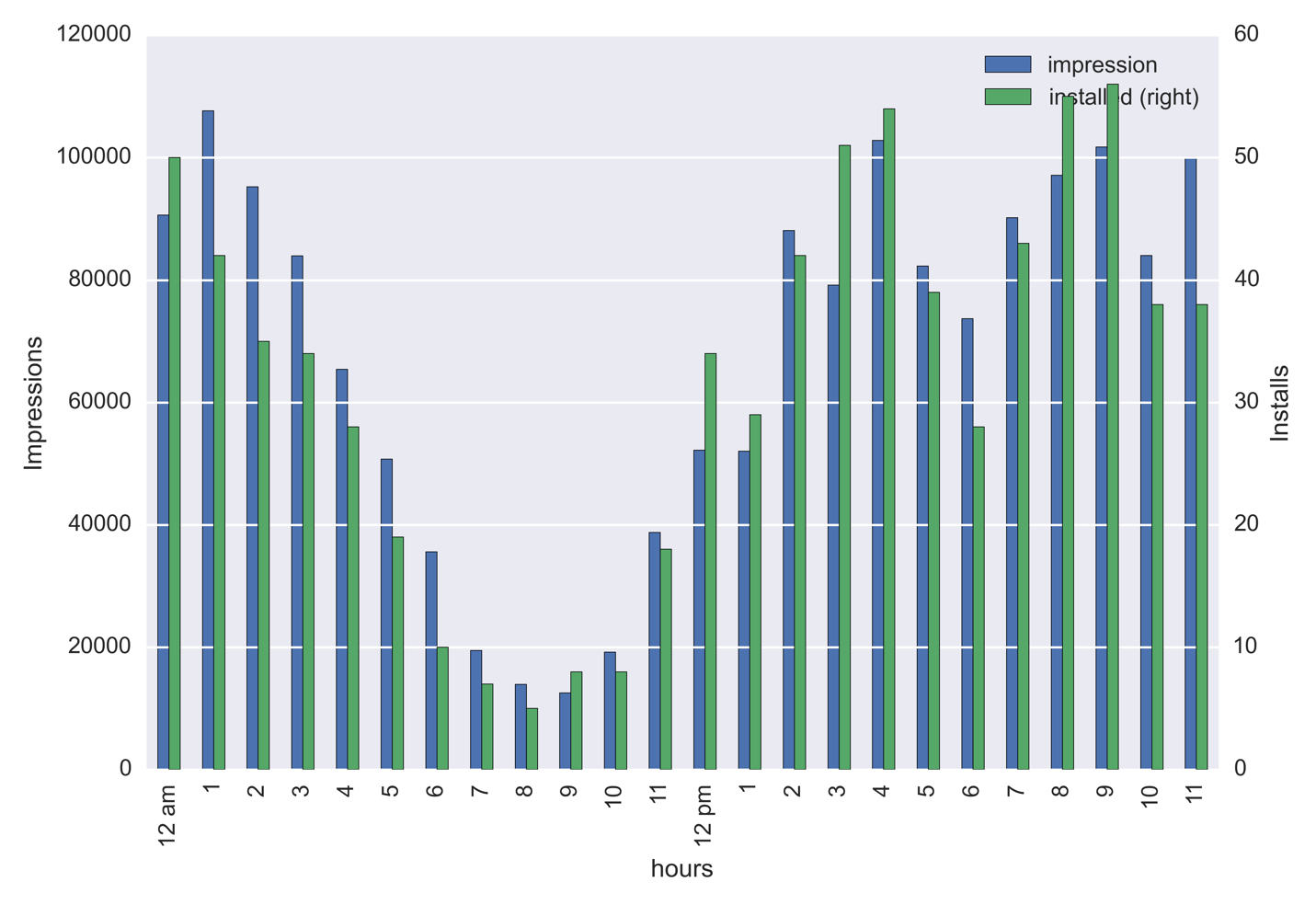
For this work the company supplied the data in the form of basic user information (such as model, city, country) as well as date and time in which the ad was shown, and other information concerning the publishers and ad venues. The data immediately presented challenges, for one it was severely imbalanced, the percentage of installs was a mere 0.05%, this number pales in comparison to the number of times a user gets shown an ad.

Categorical Variables

With the exception of the continuous time variable, all the other variables presented are categorical. To handle these, I used the python package patsy to create the dummy variables that are essentially binary variables that represent each category of the feature. To assure some variable independence one of the columns of the dummy set is removed. Due to computational reasons, the number of dummy variables is 1506, the dataset is down-sampled to 100,000 samples, at the same time to ensure the transferability of the model the class imbalance is preserved.

Data visualization, although a challenge because of the categorical nature of the features, serves as a sanity check and a common assumption is that people have more time in the weekends and this is shown with the increased weekend install rate.





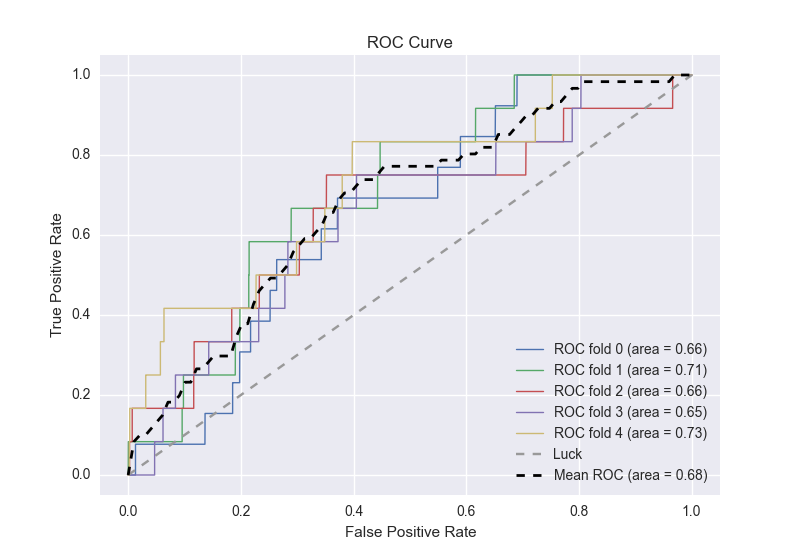
One neat way to look at which features to consider in your model, especially when your predictors are categorical, is to use the Random Forest classifier algorithm to obtain the feature importance. For this data set in particular features such as days, hours, publisher, app name, ad venues, app category, country and phone models proved to be the ones that carried a larger weight on the prediction.

Classification

The goal here is to use the data provided to classify the user as installers or non-installers. For this process I use machine learning algorithms for classification. I utilize the logistic regression classifier contained in the python package scikit-learn. Other classification algorithms such as Random Forests and Gradient Boost Classifier were considered, however the trained model was unable to overcome the severity of the class imbalance leading to poor outcome prediction.

Evaluation of the Model

To evaluate the effectiveness of the model, I looked into several metrics. It is standard approach to look at the area under the curve of the receiver operating characteristic (ROC) curve. The ROC shows the relationship between true predicted outcomes versus the number of false predicted outcomes, using a 5 fold cross-validation the area is on average 0.68, larger than 0.5, which generally means that the model is performing better than random. Other metrics that I looked at were precision and recall, however it was clear from the start that these metrics only care for the absolute numbers of installs and for the case of classifying outcomes in severe imbalanced dataset this is not the most prudent thing to do.



To effectively evaluate my model, I turned to lift, a metric that is often used in the advertisement industry to evaluate ad campaigns. It is calculated as the ratio between the results obtained with and without the predictive model. To obtain the lift for my model I first ranked the actual outcomes by the predicted probabilities. I then calculated the install rate as the number of installs in a fraction of the data divided by the total number of installs. This is then compared to the baseline assumption that for each percent of users who are shown an impression the same percentage installs. Looking at the lift graph my model outperforms the current situation by 300% in the users who my prediction ranks as most likely to install.

How is this helpful for the company?

Using lift and gains as the effectiveness metric, my model can help the company make better informed decisions that result in getting a more consistent number of users who installed. While the result is not an absolute number of installs it is very helpful to know that using the probability model that I created outperforms the current setup for the top 60% of the user base by a minimum of 200%.