|  |
| --- |
| Turnkey Intelligence |
| Advanced Prospecting |
| Prospector Model Prototyping Document |

|  |
| --- |
| Anders Bergren  8-3-2017 |

Contents

[Objectives 2](#_Toc489541312)

[Capacity Model 2](#_Toc489541313)

[Current Model Overview 2](#_Toc489541314)

[New Model Overview 2](#_Toc489541315)

[Data Preprocessing 3](#_Toc489541316)

[Model Algorithm Tuning and Selection 4](#_Toc489541317)

[Prediction and Binning 5](#_Toc489541318)

[Future Possibilities 5](#_Toc489541319)

[Current vs Prototype Model Performance Metrics 5](#_Toc489541320)

[Recommendations 5](#_Toc489541321)

# Objectives

Prospector, being a large source of revenue for Turnkey, has been threatened in recent years by the arrival of other advanced models in the sports industry. After becoming a fully automated process at Turnkey it provides an almost passive source of revenue. Unfortunately, this automation also removed the need to understand the models at their core. I’ve taken it upon myself to uncover the design principals of the Prospector models, view them in the light of modern machine learning, and redesign their them to better suit our needs and the needs of our clients.

The goal of this document is to describe and validate new approaches to Prospector modeling. These methods incorporate well-established and modern methods of machine learning while attempting to remain within Turnkey’s capacity to run and maintain them. Increases in predictive accuracy as well as new modeling features will establish these methods as improvements over the current implementation. The reader is expected to have a general understanding of machine learning and our current modeling efforts.

# Capacity Model

## Current Model Overview

The Capacity Model is useful for gauging a person’s propensity to spend money on a sporting event based purely on demographic and psychographic information. Spend is binned into a 1 to 5 star rank indicating spend propensity and the rank is then predicted. The model will intentionally not use information known about behavior related to the sporting event in question to simulate cases when this information would not be known. This model also ignores cases where spend did not occur to act as a simulation for how much a person would spend if they had already chosen to spend at all.

Versions of the data preprocessing stages are iterated upon, with each iteration receiving its own model build. Iterated preprocessing steps include variations in bin splits, p-value limits, and imputation. Once these iterations are trained and tested on a subset of the data not used for training, an analyst steps in to select the highest performing model. Selection is based on the mean spend gradually increasing with each predicted spend as well as an adequately sized population within each bin (close to the original bin distribution).

## New Model Overview

The new Capacity Model keeps the beginning and end similar to the current version, but everything in between has been updated. Instead of predicting propensity to spend in a 1 to 5 star rank, this model predicts actual spend so that all models stand on the same footing for comparing model accuracy. Technically it predicts the log of spend to normalize the distribution and better handle a large amount of extreme spend outliers.

The modeling process will shift its computational demand from iterating over preprocessing stages to iterating model algorithms and algorithm hyperparameters. These new iterations will create an immense number of possible models, but because they all predict exactly the same target variable we can mathematically select the most accurate model for scoring. This not only adds an academic rigor to the model selection process, but reduced the demand on an analyst to select a model themselves.

The predicted spends can then be converted into 1 to 5 star bins for client use. Because the most accurate model has already been selected to predict spend, the bins are now purely descriptive and hold no predictive weight. This is a major advantage that opens possibilities for bin selection curated to a client’s needs. For example, if a client does not have the resources to contact all the people who scored 5 stars in their database where do they start? We can adjust our binning system in this situation to score more conservatively and reveal the gradient previously covered by a single bin *without* changing the model!

The machine learning process from beginning to end currently uses the MLR framework. MLR is a package available in R that unifies and streamlines the major statistical packages for R and allows for a more understandable pipeline while enabling parallel computing. Utilizing this framework also allows for more modular model processes should we ever decide to improve or replace certain stages.

Parallel computation will provide a major speed advantage over the current process and allows the code to take advantage of higher computational capacities than before. With this increase in speed, a dramatically larger numbers of model iterations are possible, which translates into higher accuracy and better test validation. A current bottleneck of this model design will be the hardware available to run it on. Whereas the previous model could not take advantage of more CPU cores, this version will use however much computational capacity is provided to it and convert that into improved model accuracy. I recommend creating a budget of $700-$1000 to increase out computational capacity since, for example, 8-core CPUs have become increasingly cheap over the past few years. For actual pricing and ordering I would defer to the CTO’s recommendation.

## Data Preprocessing

Cases with 0 spend or expired PersonicX data still removed. The data is preprocessed in several new ways. Identifying or redundant fields are always removed. Then the data is formatted into numeric (ratio scaled) or factor (discrete codes) variables based on their type. Several variables are formatted in their own specific way to optimize the information available to the model. Variables with more than 60% of their data missing are removed from the model. The remaining variables eventually have their missing values imputed as a median of the available data. The median creates an outlier resistant midpoint for numeric variables and, if effect, a mode for factor variables which have been converted to binary flags at this point. In the end, missing variables never given their own code to be used as information in the model.

These methods of feature selection and imputation are overly simplistic in their current state and serve as more of a placeholder. Eventually the optimal missing value threshold will be found as part of the model tuning process and imputation will eventually be modelled instead of computing the simple median. Each of these improvements require significantly more computational capacity.

Additional preprocessing steps include automatically merging extremely small variable factor levels into a single “other” level, converting factors into dummy variables, scaling all variables to a 0-1 scale, and removing outliers from the log spend target by deleting cases 3 inter-quartile ranges outside the actual inter-quartile range (IQR being the distance between the 25th and 75th percentiles). This method of outlier removal is extremely forgiving in that it removes cases which are obvious outliers even after being log normalized, but if desired the number of IQRs outside of the actual IQR can be manually adjusted. This process cannot be included in model tuning because it changes the predicted variable and would invalidate the model performance comparisons. I use this method because it is a common method of outlier removal, but I would be open to using Michaels more customized method of outlier removal involving distances between extremely high spends.

## Model Algorithm Tuning and Selection

Several model algorithms can be used for model building. The process will iteratively try each algorithm provided and select the one which produces the most accurate results. Algorithms I suggest using include ridge regression, adaptive splines, random forests, and most of all: gradient boosted trees. The R package *xgboost* for gradient boosted trees is currently making waves in the machine learning community and from testing I can tell is an extremely well performing algorithm. MLR is built to accommodate each of these models and can easily be told to try new algorithms as we learn about them. These advanced algorithms contain settings called hyperparameters. If an algorithm is used for model building without specifying these hyperparameters a standard set will be chosen that may work but will not be tailored to our dataset. What we first need to do is tune our model algorithms to their optimal hyperparameter points. This occurs before comparing the algorithms directly to each other and makes up the bulk of the modeling processes computational demand.

MLR, thankfully, has a built-in process to automatically tune our algorithms. First, the hyperparameters to tune are identified and ranges are provided for the tuning process to iterate over. This requires some understanding of each algorithm and the effect each hyperparameter will have, but once a good general set is decided upon tuning will run automatically without manual input. Then a tuning method is chosen. Two possible options here include creating a matrix of evenly spaced hyperparameter values within their provided ranges or randomly generating values over their provided ranges. Either way, this is where we want to ramp up model iteration to a density that takes 12-48 hours to run. The number of iterations possible is dependent on our hardware’s capability, but the higher the better the model will be.

The final set of tuned hyperparameters is decided on by a process called *k-fold* *cross validation*. Cross validation is an extremely popular method in machine learning to prevent overfitting on the training date and improve model accuracy. In brief, 5-fold cross validation is a process by which each model iteration is trained on 4/5th of a dataset 5 times. The 1/5th not used to train becomes that iteration’s test set. Each time a different 1/5th is used as the train set until after 5 times the entire set has been used to both train and test the model. The test performance measures are finally averaged into that iterations overall performance. A cross validated performance measure is proven to be a much more realistic metric than a standard train/test method. This cross validated measure can then be compared to each other iteration’s cross validated measure to find the optimal set of hyperparameters. 5-fold cross validation is about 5 times as computationally intensive as normal (1-fold) testing, but the increase in accuracy is significant. Going even further is a process called nested cross validation which can be 9+ times as computationally intensive but again provides greater accuracy. For now, I have implemented a standard 5-fold cross validation.

Once the final set of hyperparameters has been found for each algorithm the models are run one final time (again using cross validation), this time iterating over the different algorithms with their optimized hyperparameters. From this, performance measures can be benchmarked and a final model selected.

## Prediction and Binning

Once the objectively most accurate model has been determined scoring can take place. The final model will be run on new data (using the same preprocessing except outlier removal) and spend will be predicted. After the predictions are made, spend will be binned using the same percentile ranges the client has become used to from their previous models. These typically are 20/40/60/80 or 30/60/80/90 percentile splits. With the new process, any binning combination is possible. I’ve found that, even at 100 bins representing 1 predicted percentile each, the average actual spend per bin does not “clump” and has no percentiles out of order.

## Future Possibilities

Even with the planned long-term implementation above there are additional pieces of Prospector we can improve.

A final stage in the modeling process I believe can be improved is the model memo. This is what we use to describe the model we’ve built to the client. It includes performance metrics and feature importance from the selected model. I think this process can be greatly improved if automated report generation is used here. If reports are generated automatically we can use a larger and more detailed variety of metrics and data visualizations to help the client understand the model they’ve purchased.

Some clients are also interested in using their own data in the model. This will always require a degree of customization in the preprocessing stage, but adding “extra” data (or supplementary data if they wish to use their own instead of PersonicX) will be completely possible in this framework. That said, preparing outside data sources will always require a time investment from an analyst familiar with feature engineering and should be priced appropriately.

## Current vs Prototype Model Performance Metrics

Even in the new Capacity Model’s prototype phase it has begun to outperform our current modeling process. The gradient boosted tree algorithm from the xgboost R package has been an especially strong predictor. Below are a series of performance measures from various client models comparing our current performance to the prototype model’s performance to highlight the improvement.

[Placeholder]

# Recommendations

* Approve time to develop the Capacity Model into a production level platform
* Approve time to prototype a new Priority model process, with eventual plans to bring to production level
* Consider new possibilities for delivering predictions outside of the 5-star scoring system such as a finer detailed 1-100 score or a list of contacts ordered by their propensity to spend.
* Create a budget of $700-$1000 for new hardware to test more computationally demanding models (I defer to the CTO for an actual budget recommendation and implementation)

This document was composed to communicate a path forward for our modeling capabilities. Successful execution of this plan requires a significant time investment to take a proof-of-concept script to a production level platform. Therefore, a plan should be agreed upon by high level managers of Turnkey before additional resources are spent developing the next generation of Prospector.