Predicting swing decisions

1. Load the training data and add ‘is\_swing’ by parsing the data in ‘pitch\_call’
2. Add engineered features, such as the distance of the pitch from the center of the zone, the location of the pitch without any break, and boolean columns for bucketing the location and count.
   1. The xgboost models perform better when the target can be largely explained by a monotonic dependence on just a few features.
   2. I chose an xgboost model because it is particularly well suited to highly nonlinear problems with many interactions and access to a large dataset.
3. Split the training data into a training set and a validation set
4. Set the features that will be used in the pitch characteristic-only model, including monotonic constraints on ‘dist\_from\_center’ and ‘no\_break\_dist’ which are the distance from the center of the zone with and without break respectively.
   1. More features were tested than shown in the notebook. Other than the features describing the count and pitch location, the next few most important features for both fastball and secondary offerings were included in the final model.
5. Tune the hyperparameters of the pitch characteristics-only xgboost models using cross-validation on the training set.
   1. There are 8 separate models trained, one for each combination of fastball/secondary and matchup platoon.
   2. The cross validation was done using R vs R fastballs, and I confirmed that the optimal values were reasonable for the other models.
   3. A variety of different parameter grids were used during tuning; a final example grid is shown in the notebook.
6. Calculate the average difference between a batter’s swing rate and the expected swing rate based on the pitch characteristics-only models for different counts and pitch locations. The difference is calculated separately for each platoon and pitch group. We’ll call these numbers a batter’s “swing peripherals” from now on.
   1. The count buckets were chosen based on when hitters may significantly change their approach
      1. First Pitch (0-0)
      2. Even (1-0, 0-1, 1-1, 2-1)
      3. Three Balls No Strikes (3-0)
      4. Hitter’s Count (2-0, 3-1)
      5. Two Strikes (0-2, 1-2, 2-2)
      6. Full Count (3-2)
   2. The pitch location buckets are
      1. Up (> 6 inches above the center of the zone)
      2. Down (> 6 inches below the center of the zone)
      3. Inside (> 6 inches towards the hitter from the center of the zone)
      4. Outside (> 6 inches away from the hitter from the center of the zone)
      5. Middle (within the central 1 square foot of the strike zone)
7. Regress the swing peripherals to account for sample size differences. The proper regression amount is estimated from the ratio of the variance of the differences themselves to the variance of the average difference of the hitters, adjusted and weighting for sample size
8. Include the regressed batter swing peripherals as features in the final xgboost model.
   1. The features corresponding to the count and x- and y- location of the pitch are given their own features. This is mostly what the model focuses on, but all the swing peripherals are provided for context.
   2. Monotonic constraints are placed on all of the swing peripheral features to prevent overfitting. Basically, if a swing peripheral is high, then the model can only use that information to increase the probability of swing, never decrease it.
   3. This is the step where I planned on including the level as a feature. It was important not to include it in the pitch characteristic-only model, as that would make the swing peripherals relative to the level, and I wanted them to all be on the same scale. However, when I included one-hot encoded features for the level, they were by far the least significant features in the model. It appears that the batter swing peripherals were sufficient to encode the differences in swing-decision talent that exists across levels.
9. Tune the hyperparameters for the final xgboost models by maximizing accuracy on the held-out validation set across all 8 models.
   1. Because the swing peripherals are not out-of-sample for the training set, there is a tendency for them to overfit even with the monotonic constraints. I found that the most accurate results could be obtained by weighting the final model that includes swing peripherals at 75% and the original pitch characteristic-only model at 25%.
10. Retrain all models using the entire training set.
11. Load the test set and add all engineered features, including joining in the swing peripherals.
12. Predict whether the batter swung at each pitch in the test set using a 75/25 split of the swing peripheral-inclusive and pitch characteristic-only models.

Potential improvements:

1. Using a more robust feature addition or feature elimination strategy using cross-validation, rather than including the top few features from among the less important features.
2. Make the swing-peripherals out-of-sample in the training set by splitting the data and using the peripherals calculated from one half of the data as features in the other half. Its possible this is worse, because you can only work with half the data, but it would be worth trying.
3. Test features describing the typical stuff of the pitcher, especially average fastball characteristics, and the framing ability of the catcher to see if they improve accuracy.
4. With a dataset that made it easier to determine pitch order, information about the previous pitch, both the characteristics and outcome, may also help the model.