MLB Stolen Base Predictions

By: Waleed Almousa & Colin Macy

Introduction

Questions:

- Can we use a machine learning model to accurately predict the outcome of a stolen base attempt?
- What factors influence the success of a stolen base attempt the most?

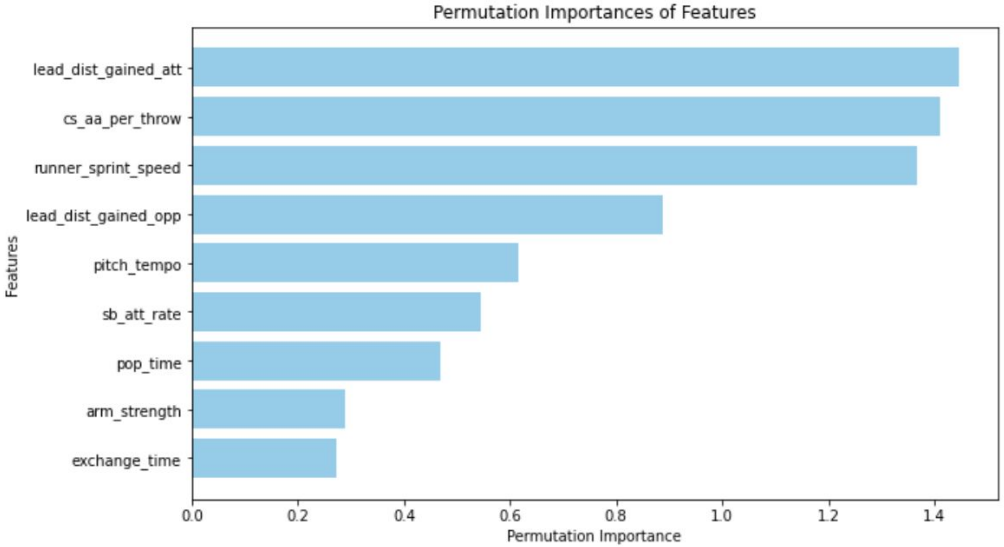
Dataset

- Custom dataset from a combination of MLB statcast resources
- Retrieved stolen base attempts with API call to baseballsavant.mlb.com
- Dataset is 2789 rows x 15 columns
  - Target variable: Successful_sb
  - 8 predictors shown below
  - 6 miscellaneous variables used for merging datasets (pitcher ID, catcher ID, etc.)

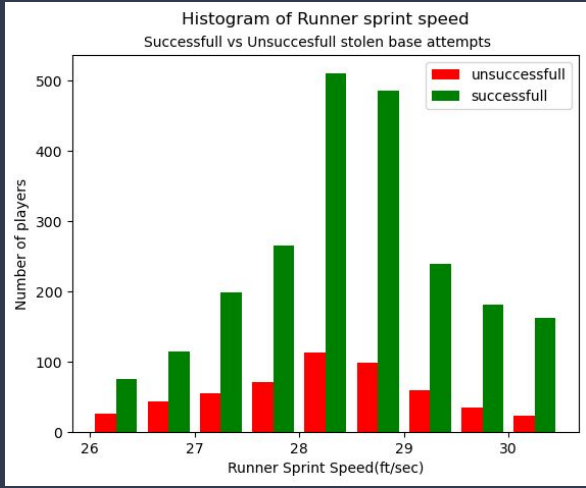
successful_sb	runner_sprint_speed	pitch_tempo	sb_att_rate	lead_dist_gained_opp	pop_time	exchange_time	arm_strength	cs_aa_per_throw
1	28.0	17.433	0.023847	1.847397	1.988417	0.644583	78.32061	0.175065
0	27.8	17.433	0.023847	1.847397	1.988417	0.644583	78.32061	0.175065
1	29.2	17.433	0.023847	1.847397	1.988417	0.644583	78.32061	0.175065
1	30.4	15.792	0.014737	1.715638	1.988417	0.644583	78.32061	0.175065
1	27.1	17.923	0.012097	3.617790	1.988417	0.644583	78.32061	0.175065

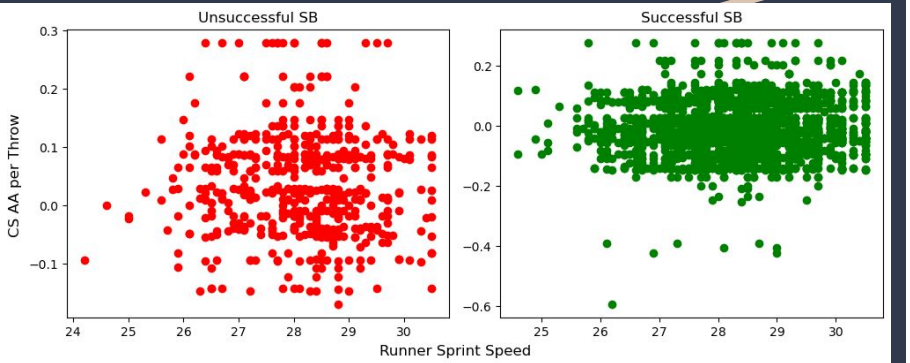
Feature Choice

- Started with eight potential predictor variables
- Used permutation importance to narrow it down to four:
  - Runner_sprint_speed
  - Lead_dist_gained_opp
  - Lead_dist_gained_att
  - Cs_aa_per_throw (Caught stealing above average)



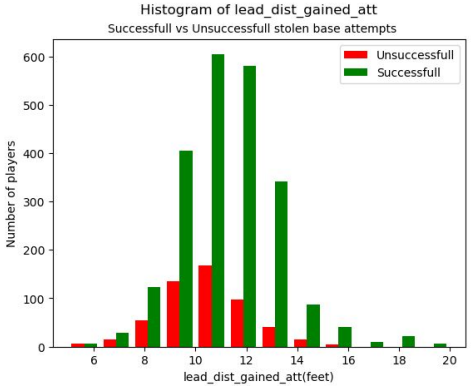
Exploratory Plots

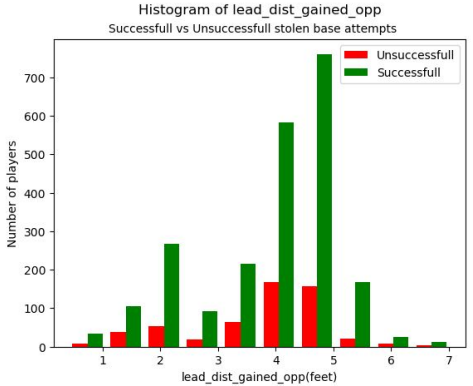




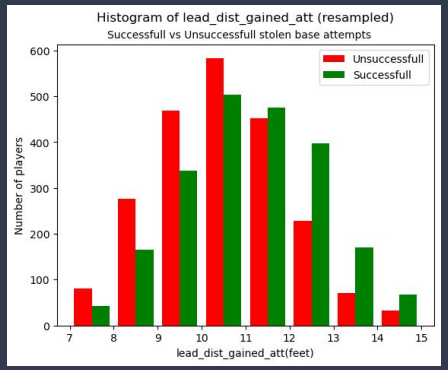
The Dataset is imbalanced

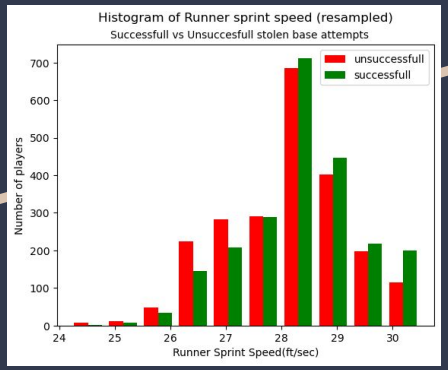
- 81% of observations were successes
- Resampling/data manipulation Was needed



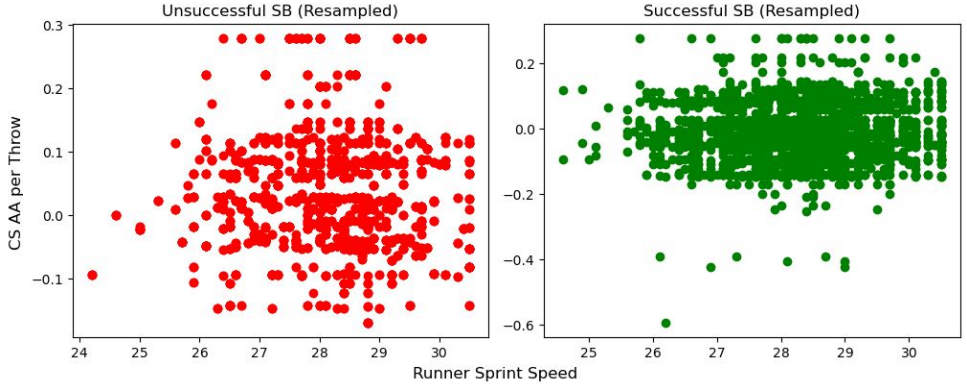


Over/Undersampling

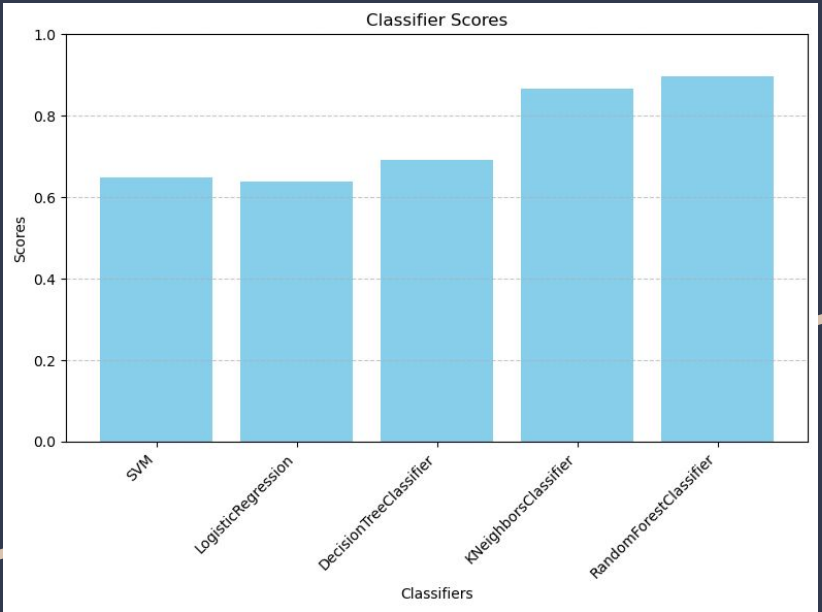




- Using this data set, any model can predict 1 for every observation and get an 81% accuracy score
  - This model would be useless!
- Oversampling solved this by creating more “fake” observations that had an outcome of 0
- Using oversampling produced higher accuracy score than undersampling
  - Undersampling would require for 1000+ observations to be removed



Model Selection



SVM('C': 1, 'Kernel': 'Linear') scored .648

LogisticRegression('C': 1, 'max_iter': 5000) scored .639

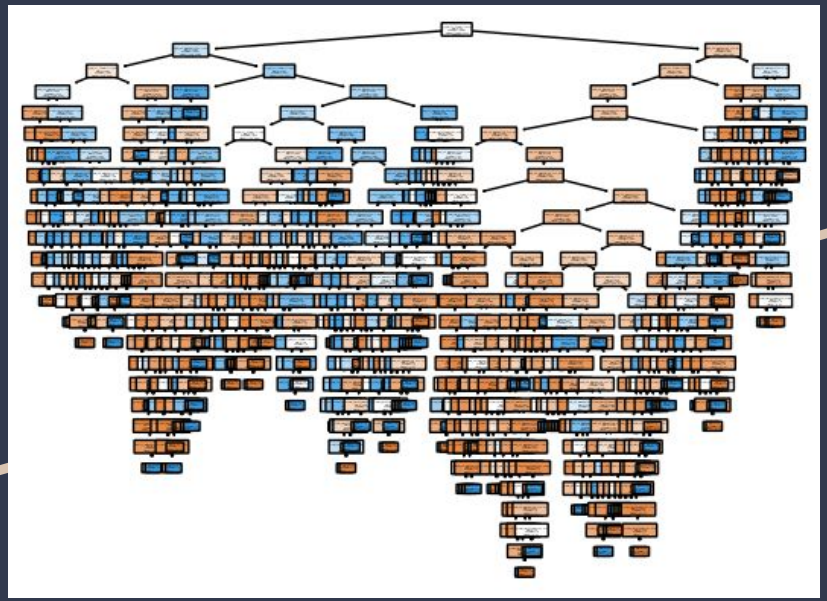
DecisionTreeClassifier('criterion': 'Entropy', 'max_depth': 7) scored .692

KNeighborsClassifier('metric': 'euclidean', 'n_neighbors': 1) scored .867

RandomForestClassifier('max_depth': 100, 'n_estimators': 100) scored .896

- Utilized GridSearchCV on a range of different models with a range of different hyperparameters to get the best fits
- The first three performed worse than guessing a successful stolen base for every attempt
- KNeighborsClassifier and RandomForestClassifier performed better than guessing a successful stolen base for every attempt

Random Forest Classifier



- Subject to less noise
- Reduced Overfitting
- Accuracy score of **89%**
  - **8%** more accurate than the success rate of stolen bases throughout the year
- Recall of **78%**
- Precision score of **97%**
  - Precision is most important in this context
  - When the model predicts a success, it will be correct 97% of the time
- Model Can be used by coaches to minimize risk and give teams a competitive edge

Summary

Obtain Dataset

- Compiled dataset from variety of different MLB statistics resources (found in repository)

Feature Choice and Oversampling

- Narrowed down features with permutation importance
- Data was oversampled to avoid creating a useless model

Random Forest model predicts with **89% accuracy** and **97%** precision

- MLB teams could use this model to decide when they should have runners steal bases
- Teams that utilize this model would gain a competitive advantage over others in terms of stolen base successes