2016 MLB Season

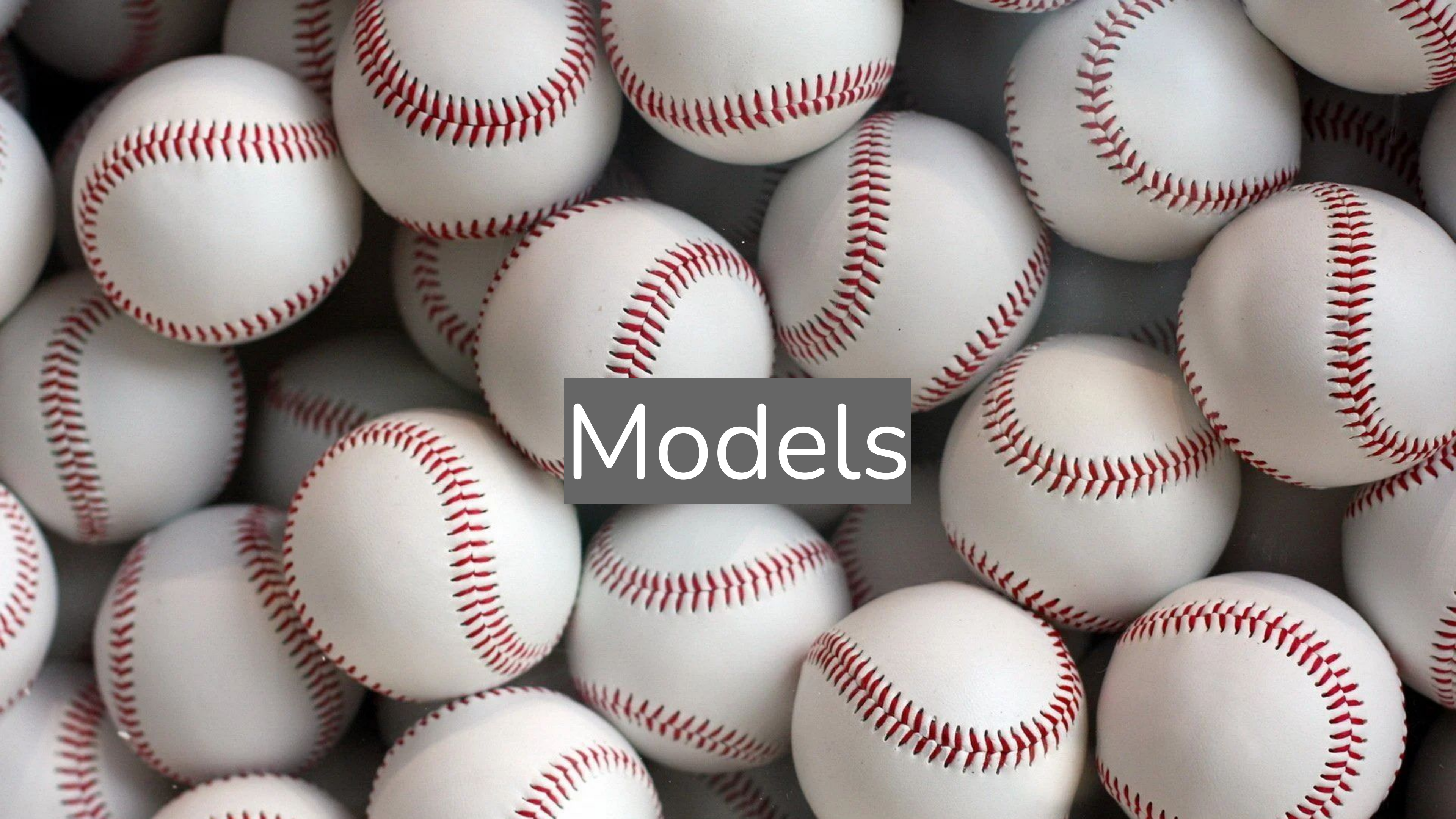
Kristin Fasiang, Ricardo German, Sid Mehrota, Melchor Ronquillo, Rob Sisto

Introduction

- The Dataset:
  - From Kaggle
  - Stats from each game played in the 2016 MLB season
    - Time, date, weather
    - Home team, away team
    - Runs, hits, errors during game by each team
- Our Expectations + Goals:
  - Predict who will win a game only on pre-game factors
    - Ignore runs, hits, errors
  - Difficult task
    - Goal of 55% prediction accuracy

Preprocessing

- The Base Model:
  - After reducing the Kaggle dataset to only include pre-game factors:
    - Attendance, away team, date, field type, game type (day or night), home team, start time, day of week, temperature, wind speed, wind direction, sky, and season (regular or post-season).
  - Best accuracy was 52%
- Our Additions:
  - Number of hits and runs for and against each team in the season so far
  - Winning % up to the day of the game
  - Home and away winning percentages splits
  - Ballpark factor
  - Run differential
  - Pythagorean Winning Percentage



Models

Ricardo’s Model

Dataset:

- Feature Selection
  - Environmental
  - Game percentages
  - Ballpark Factor
  - Regular Season
- Model
  - Logistic Regression
- Hyperparameters
  - C = 0.01
  - Penalty = “l2”
  - Solver = “saga”

Results:

- Accuracy
  - Training/Validate - 53%
  - Testing - 54%
- F1 Score
  - 0.69

Sid’s Model

- Dataset:
    - Kept Rob’s additions but then used L1 regularization to drop insignificant features
    - Normalization of feature data
    - Split data into 60% training, 20% development and 20% test set.
- Perceptron:
    - Ran a “grid search” to determine the best learning rate and used 50 epochs.
- Adaline:
    - Ran a “grid search” to determine the best learning rate and used 50 epochs.

```
Best Perceptron Accuracy on Dev: 0.5182926829268293
Best LR on Dev: 0.001
This is Perceptron accuracy on Test: 0.565922920892495
Best Adaline Accuracy on Dev: 0.5223577235772358
Best ETA on Dev: 0.0001
This is Adaline accuracy on Test: 0.5679513184584178
```

Kristin’s Model

- The Dataset:
    - Calculated differences
        - Win percentage to date, AVG runs by and against, and AVG hits by and against
    - Ordered team indices
        - By record in the 2015 season
- The Models:
    - Basic NN with Keras
        - 4 densely connected layers, 10 epochs
        - 53-55% accuracy depending on random seed
    - Grid search over SVM + Logistic Regression
        - Best was SVM
            - C = 0.1, kernel = 'linear', and tolerance = 1
        - Accuracy:
            - Train: 0.569   Test: 0.553

Melchor’s Model

- Dataset
  - Kept Rob’s additions
  - 80 / 20 split
  - Dropped insignificant variables
    - In-game statistics
    - Date, Time
  - Qualitative data changed to Categorical data type
    - Strings → numbers that represent each value respectively
  - Quantitative data scaled
- Model 1
  - Support vector classifier
  - 10-fold CV Grid Search
  - Best parameters:
    - C : 1
    - Gamma : 0.01
  - **Train Accuracy = 59%, Test Accuracy: 56%**
- Model 2
  - 90 / 10 split
  - Same exact process
  - Best Parameters:
    - C : 100
    - Gamma : 0.001
  - **Train Accuracy: 63%, Accuracy : 63%**
- Model 3
  - Applied parameters to 80 / 20 split (model 1)
  - **Accuracy : 57%**

```
[ ] best_SVC = SVC(C = 1, gamma = 0.01, kernel = 'rbf').fit(x_train, y_train)
    best_pred = best_SVC.predict(x_test)
    print(classification_report(y_test, best_pred))

    /usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataConversionWarnin
      y = column_or_1d(y, warn=True)
                  precision    recall  f1-score   support

               0       0.53      0.41      0.46       227
               1       0.58      0.70      0.63       265

        accuracy                           0.56       492
       macro avg       0.56      0.55      0.55       492
    weighted avg       0.56      0.56      0.55       492
```

```
best_SVC = SVC(C = 100, gamma = 0.001, kernel = 'rbf').fit(x_train_full, y_train_full)
best_pred = best_SVC.predict(x_test)
print(classification_report(y_test, best_pred))

/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataConversionWa
  y = column_or_1d(y, warn=True)
              precision    recall  f1-score   support

           0       0.62      0.49      0.55       168
           1       0.64      0.75      0.69       201

    accuracy                           0.63       369
   macro avg       0.63      0.62      0.62       369
weighted avg       0.63      0.63      0.63       369
```

```
[ ] '''did grid search on 90 / 10% train test split by accident, got 63% with parameters below,
       used the paramters on this proper 80 / 20% split and got a better score'''
    best_SVC2 = SVC(C = 100, gamma = 0.001, kernel = 'rbf').fit(x_train, y_train)
    best_pred2 = best_SVC2.predict(x_test)
    print(classification_report(y_test, best_pred2))

    /usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataConversionWarnin
      y = column_or_1d(y, warn=True)
                  precision    recall  f1-score   support

               0       0.55      0.39      0.46       227
               1       0.58      0.72      0.64       265

        accuracy                           0.57       492
       macro avg       0.56      0.56      0.55       492
    weighted avg       0.56      0.57      0.56       492
```

The Best Approach - SVM

- ➔ Kristin's SVM accuracy = 55.3%, Melchor's SVM accuracy = 57.0%
  - ◆ 1.7% difference
  - ◆ Similar performance
- ➔ Differences in Methods
  - ◆ Elimination of additional features
    - Original dataset = 42 columns
    - Processed dataset = 28 columns
  - ◆ Radial Kernel vs Linear Kernel
  - ◆ 10-fold vs 5-fold cross-validation
    - Higher fold → train on larger data → test on smaller data
  - ◆ 90/10 training/testing split on the data
    - Should not use due to overfitting
    - More data → better fitting → higher accuracy
    - Led to finding better parameters
- ➔ **Overall best method for classifying home team wins with this data: SVM**

```
Grid search results:
Classifier: SVM
Parameters: [0.1, 'linear', 1]
TRAIN SUCCESS 0.5697615423642821
TEST SUCCESS 0.5528455284552846
```

```
best_SVC2 = SVC(C = 100, gamma = 0.001, kernel = 'rbf').fit(x_train, y_train)
best_pred2 = best_SVC2.predict(x_test)
print(classification_report(y_test, best_pred2))

/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataC
  y = column_or_1d(y, warn=True)
              precision    recall  f1-score   support

           0       0.55      0.39      0.46       227
           1       0.58      0.72      0.64       265

    accuracy                           0.57       492
```

Improvements to Make

- ➔ Increasing the amount of data
  - ◆ Look at years prior to 2016
- ➔ Factoring in player stats
  - ◆ For the people in the starting lineups
  - ◆ ERAs, batting averages, etc.
- ➔ Exploring more with NNs
  - ◆ CNNs, optimization
- ➔ Testing only on data from later in the season
  - ◆ Stats used as features could be less accurate at the beginning of the season

Conclusions

- Predicting win/loss based on pre-game factors is difficult
  - Hard problem in general
  - Makes it hard for a model to do with a ton of accuracy
- Better than dummy classifiers
  - More accurate and consistent
- In all, there’s a reason a baseball game is played, not predicted
  - So many things can happen in-game to change odds
  - Hard to know who will win before the game is played

Questions?