Predicting Quality Starts in Tough Ballparks

Capstone by Tad Inglesby

Identify the Problem

- Unlike in other sports, baseball stadiums have unique factors such as that affect the outcome of games such as weather conditions and ballpark
- Coors Field (Denver)/Chase Field(Phoenix): Elevation and thin air
- Rogers Center (Toronto): Astro Turf
- Miller Park(Milwaukee): Field Dimensions
- Camden Yards and Globe Life Park: Summer humidity
- These conditions help create positive run-scoring environments for home and away hitters, adversely affect pitchers
- Ground Ball Pitchers → More success in these environments
- What explains quality starts, ground balls or something else?

Acquire the Data

- Averaged Park Factor Rankings by Runs from 2011-2016 and categorized top 10 as “Hard” Ballpark Dataset, and bottom 20 as “Easy” Ballpark
- Scraped starting pitching lines from box scores for every game from 2014-2016, grouping them into the hard or easy ballparks categories

MLB Park Factors - 2011

RK	PARK NAME	RUNS	
1	Globe Life Park in Arlington (Arlington, Texas)	1.409	1
2	Coors Field (Denver, Colorado)	1.347	1
3	Fenway Park (Boston, Massachusetts)	1.173	0
4	Rogers Centre (Toronto, Ontario)	1.152	1
5	Chase Field (Phoenix, Arizona)	1.146	1
6	Yankee Stadium (New York, New York)	1.131	1
7	Minute Maid Park (Houston, Texas)	1.100	1
8	Great American Ball Park (Cincinnati, Ohio)	1.082	1
9	Comerica Park (Detroit, Michigan)	1.061	0
10	Miller Park (Milwaukee, Wisconsin)	1.041	1

Baltimore Orioles Glossary · SHARE · Embed · CSV · Export · PRE · LINK · ?

Pitching	IP	H	R	ER	BB	SO	HR	ERA	BF	Pit	Str	Ctct	StS	StL	GB	FB	LD
Miguel Gonzalez, L (0-1)	5.2	3	1	1	5	5	0	1.59	23	98	59	35	9	15	10	3	1

Parse/Mine/Refine the Data

- Dropped one row of data with an infinite FIP value
- Used RegEx to eliminate Game Record in name column
- Dropped irrelevant columns and derived new target columns from existing ones

	Name	IP	H	R	ER	BB	K	HR	ERA	BF	...	StS	StL	GB	FB	LD	GB_rate	FB_rate	LD_rate	QS	FIP
0	Wade Miley	5.0	3	3	3	2	8	1	5.40	22	...	11.0	15.0	6.0	7.0	3.0	0.375000	0.437500	0.187500	0	3.800000
1	Trevor Cahill	4.0	8	5	5	4	1	0	11.25	23	...	3.0	10.0	8.0	9.0	6.0	0.347826	0.391304	0.260870	0	5.700000
2	Brandon McCarthy	6.2	6	5	5	1	4	1	6.75	27	...	7.0	20.0	11.0	11.0	5.0	0.407407	0.407407	0.185185	0	4.490323
3	Wade Miley	7.0	6	4	4	1	5	1	5.25	28	...	12.0	18.0	11.0	11.0	5.0	0.407407	0.407407	0.185185	0	4.057143
4	Trevor Cahill	6.0	4	2	2	3	3	0	6.30	25	...	7.0	16.0	9.0	10.0	5.0	0.375000	0.416667	0.208333	1	3.700000
5	Bronson Arroyo	4.1	5	2	2	2	3	1	4.15	20	...	2.0	17.0	6.0	9.0	4.0	0.315789	0.473684	0.210526	0	6.370732
6	Brandon McCarthy	7.0	10	6	6	1	4	1	7.78	31	...	10.0	12.0	18.0	8.0	8.0	0.529412	0.235294	0.235294	0	4.342857
7	Wade Miley	5.0	8	5	5	3	4	1	5.04	25	...	6.0	20.0	8.0	10.0	7.0	0.320000	0.400000	0.280000	0	6.000000
8	Trevor Cahill	4.0	5	7	6	5	8	2	9.17	22	...	16.0	17.0	5.0	4.0	1.0	0.500000	0.400000	0.100000	0	9.450000
9	Josh Collmenter	4.0	5	3	3	1	3	0	3.75	18	...	4.0	12.0	6.0	8.0	3.0	0.352941	0.470588	0.176471	0	2.450000

Parse/Mine/Refine the Data

Features

```
['Name', 'IP', 'H', 'R','ER','BB','K','HR','ERA','BF','Pit','Str', 'Ctct', 'StS', 'StL', 'GB', 'FB', 'LD']
```

```
['H', 'HR','BB','K', 'Ctct', 'StS', 'StL', 'GB', 'FB', 'LD']]
```

```
['BB','K', 'Ctct', 'StS', 'StL', 'GB', 'FB', 'LD']]
```

Categorical Target:
Quality Starts

```
#Create a column for Quality Starts
QS = []

for x,y in zip( baseball4['IP'].values,baseball4['ER'].values):
    if x >= 6.0:
        if y <= 3.0:
            QS.append(1)
        else:
            QS.append(0)
    else:
        QS.append(0)
```

Build a Model: Logistic Regression, Tough Parks

```
X = baseball6[['BB','K', 'Ctct', 'StS', 'StL', 'GB', 'FB', 'LD']]
```

```
cross_val_score(logreg, X_scaled, y2, cv=5)
```

```
array([ 0.7375    ,  0.75312856,  0.72696246,  0.73265074,  0.74601367])
```

```
baseball6_scores = cross_val_score(logreg, X_scaled, y2, cv=5, scoring='roc
print np.mean(baseball6_scores)
```

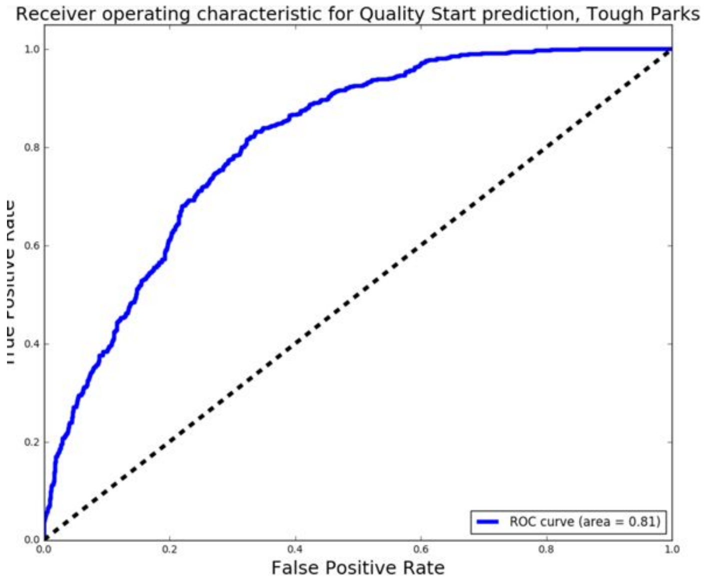
```
0.820343432046
```

```
#logreg scorewith IP, ER, Pit R, H,HR, BF, ERA, HR out, test size = 0.33,
```

```
0.73259820813232257
```

```
from sklearn.metrics import confusion_matrix
confusion = np.array(confusion_matrix(y_test, y_pred))
print(confusion)
```

```
[[596 197]
 [191 467]]
```



Build a Model: Logistic Regression, Easy Parks

```
#Easy Park Score, scaled, test size = 0.33, C = 0.1, penalty = 'l1'
```

0.73163841807909602

```
cross_val_score(logreg, X_scaled, y2, cv=5)
```

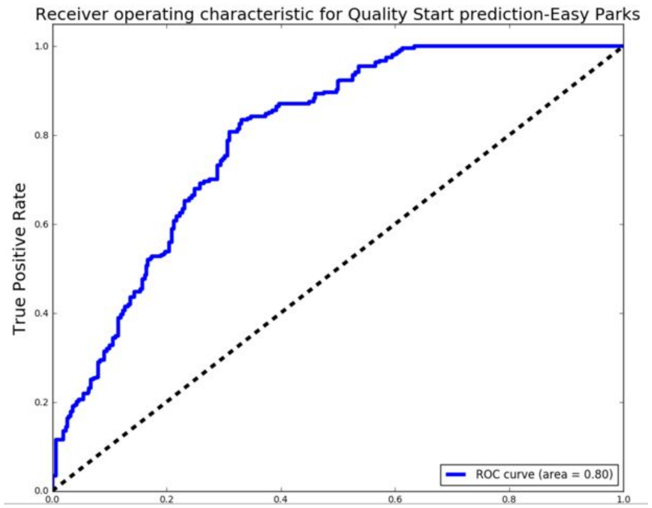
array([0.73706004, 0.72901554, 0.75544041, 0.72901554, 0.72746114])

```
easy_baseball4_scores = cross_val_score(logreg, X_scaled, y2, cv=5, scoring
print np.mean(easy_baseball4_scores)
```

0.794116798582

```
from sklearn.metrics import confusion_matrix
confusion = np.array(confusion_matrix(y_test, y_pred))
print(confusion)
```

```
[[1048  463]
 [ 392 1283]]
```



Decision Tree, Tough Ballparks

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

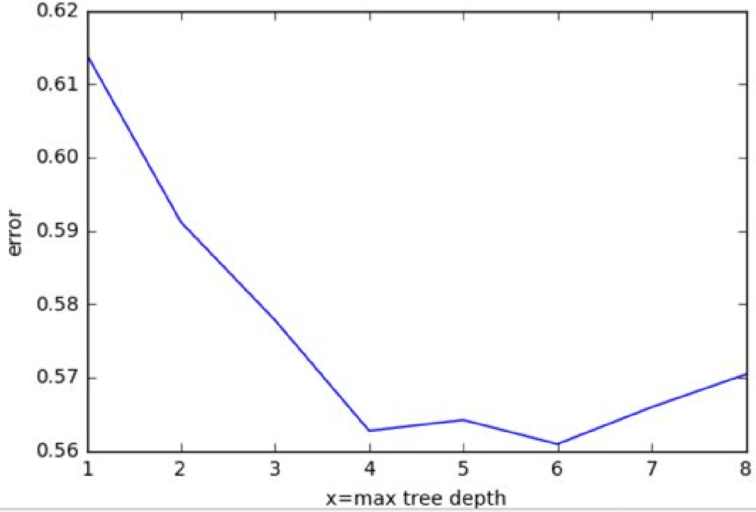
```
0.15745768125199611
```

```
rt = tree.DecisionTreeClassifier(
 min_samples_split=30, min_samples_leaf=5,
  random_state=0)
rt.fit(X_scaled,y2)
```

```
if current_score < best_score or best_score == 0:
    best_score = current_score
    best_depth = i
```

```
Best score:
0.56089626926
Best depth:
6

<matplotlib.text.Text at 0x7f1e64310150>
```



Decision Trees, Easy Ballparks

```
rt = tree.DecisionTreeClassifier(
 min_samples_split=30, min_samples_leaf=5,
  random_state=0)
rt.fit(X_scaled,y2)
```

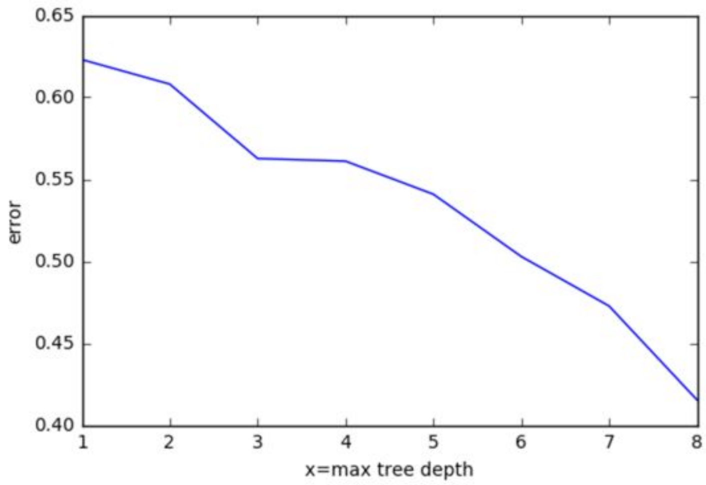
```
if current_score < best_score or best_score == 0:
    best_score = current_score
    best_depth = i
```

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
1.0
```

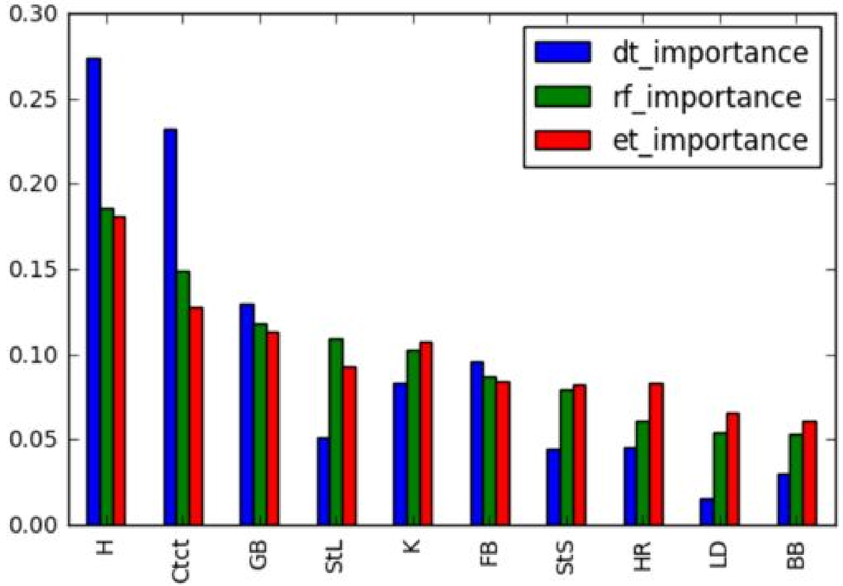
```
Best score:
0.41541799521
Best depth:
8

<matplotlib.text.Text at 0x7f1e57f796d0>
```



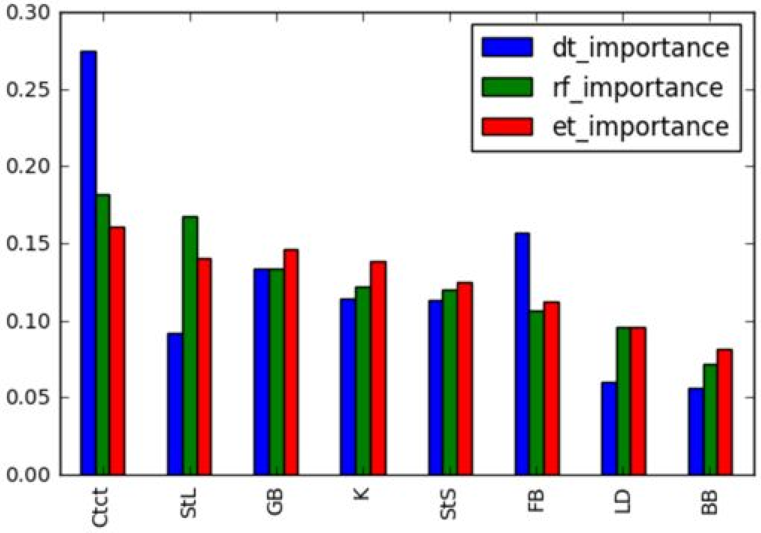
Feature Importances, Tough Parks: H, HR Included

	dt_importance	rf_importance	et_importance
H	0.273564	0.186264	0.181139
Ctct	0.232083	0.148891	0.128006
GB	0.129351	0.117725	0.112819
StL	0.051053	0.109865	0.092989
K	0.083339	0.102241	0.107014
FB	0.095482	0.086799	0.084656
StS	0.044298	0.079218	0.082585
HR	0.045493	0.061072	0.083599
LD	0.015563	0.054150	0.066287
BB	0.029775	0.053776	0.060905



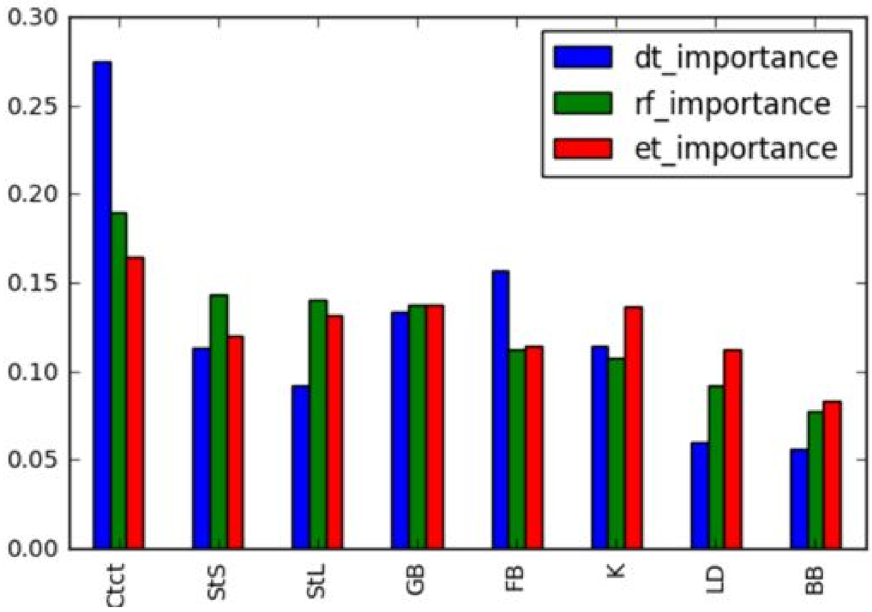
Tough Ballpark Feature Importances, no H or HR

	dt_importance	rf_importance	et_importance
Ctct	0.274694	0.182055	0.160359
StL	0.091903	0.167701	0.140590
GB	0.133592	0.133249	0.146049
K	0.113881	0.121727	0.138641
StS	0.112903	0.120510	0.125079
FB	0.156794	0.106501	0.112496
LD	0.059687	0.096271	0.095496
BB	0.056546	0.071985	0.081291



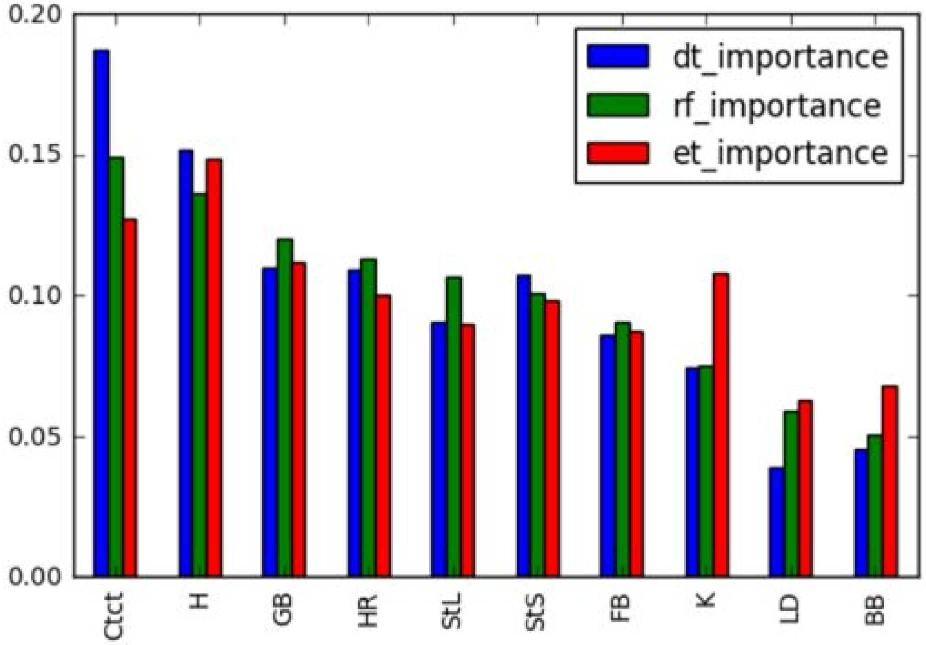
Feature Importances, Easy Parks, H & HR excluded

	dt_importance	rf_importance	et_importance
Ctct	0.274694	0.189599	0.164411
StS	0.112903	0.142824	0.120166
StL	0.091903	0.140711	0.131473
GB	0.133592	0.137693	0.137140
FB	0.156794	0.111928	0.114359
K	0.113881	0.107485	0.136465
LD	0.059687	0.092341	0.112603
BB	0.056546	0.077418	0.083382



Feature Importances, Easy Parks, H & HR Included

	dt_importance	rf_importance	et_importance
Ctct	0.187344	0.149181	0.126992
H	0.151826	0.136312	0.148496
GB	0.109906	0.120014	0.111774
HR	0.109295	0.113222	0.100333
StL	0.090671	0.106752	0.089735
StS	0.106841	0.100860	0.098009
FB	0.085581	0.090189	0.086942
K	0.074365	0.074598	0.107651
LD	0.038875	0.058456	0.062431
BB	0.045297	0.050417	0.067636



Present Results

- Strikes by Contact most “important feature in predicting Quality Starts
    - Ground Balls important, but not the most important in logistic regression and Decision Tree/Random Forest/Extra Trees Classification
- Future expansion on this study would account for quality of contact data
    - Hard, Medium, Soft