1 MLB Swing Classification

Author: Eric Wehmueller

1.1 Overview

This project is the final/Capstone project for Flatiron School's bootcamp program in Data Science. We have created a hypothetical situation as a Data Scientist and are hoping to provide value to our business for the scenario.

1.2 Business Problem

A hot topic in the 2021 Major League Baseball season surrounds discussion about certain substances being used by pitchers to increase their "spin rate"- an advanced metric now being recorded on every pitch by sophisticated cameras. The argument is that a higher spin rate on pitches gives better results, and this substance is legal and used by a high percentage of pitchers around the league. However, this is not the singular determining factor in throwing an effective pitch: namely, one that will cause a Major League better to swing and miss. Although typically regarded as an "old man's game", can we get a step ahead of the game and leverage this metric and a variety of other data on pitches to know what types of pitches will give us the best results?

We have been hired as a hypothetical member of the Cardinals baseball organization: a member of the coaching staff. As a coaching analyst, our job is to create a model that will give us insights into pitch quality and classify a pitch, given its metrics, as a "strike" or a red flag "hit" for our opponent.

1.3 Project Setup

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import metrics
```

executed in 1.43s, finished 23:20:27 2021-09-08

```
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import scipy.stats as stats
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, Gri
from sklearn.model_selection import RandomizedSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, plot_
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import export_graphviz, plot_tree
from IPython.display import Image
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, roc_curve, a
from sklearn.ensemble import RandomForestRegressor, Rand
from sklearn.tree import plot_tree
```

executed in 23ms, finished 23:38:39 2021-09-08

```
from pybaseball import playerid_lookup, statcast_batter,
```

executed in 428ms, finished 23:20:28 2021-09-08

1.4 Data Exploration

To start, let's see if we can get some immediate value in our current season against one particular player giving us trouble. So far in the 2021 season, Jesse Winker, a member of the Cincinnati Reds, has proven himself as an elite hitter. Since we are in the same division as this team (NL Central), he is going to be in the batters' box against our pitchers extremely often. If we can find a way to mitigate the damage he does against our ball club, that would be ideal. Let's work towards creating a model specifically for this.

```
player_info_df = playerid_lookup('winker','jesse')
player_info_df.head()
```

executed in 6.15s, finished 23:20:34 2021-09-08

```
Gathering player lookup table. This may take a moment.
```

	name_last	name_first	key_mlbam	key_retro	key_
0	winker	jesse	608385	winkj002	winke

```
jwinker_id = 608385
#statcast data (data per pitch, goes back to 2015)
df = statcast_batter('2016-08-01','2021-08-01', jwinker_
```

executed in 2.04s, finished 23:20:36 2021-09-08

```
Gathering Player Data
```

```
df.shape
```

executed in 14ms, finished 23:20:36 2021-09-08

```
(5805, 92)
```

```
print(df.columns.tolist())
```

executed in 13ms, finished 23:20:36 2021-09-08

```
['pitch_type', 'game_date', 'release_speed', 'release_pos_
x', 'release_pos_z', 'player_name', 'batter', 'pitcher',
'events', 'description', 'spin_dir', 'spin_rate_deprecate
d', 'break_angle_deprecated', 'break_length_deprecated',
'zone', 'des', 'game_type', 'stand', 'p_throws', 'home_tea
m', 'away_team', 'type', 'hit_location', 'bb_type', 'ball
s', 'strikes', 'game_year', 'pfx_x', 'pfx_z', 'plate_x',
'plate_z', 'on_3b', 'on_2b', 'on_1b', 'outs_when_up', 'inn
ing', 'inning_topbot', 'hc_x', 'hc_y', 'tfs_deprecated',
'tfs_zulu_deprecated', 'fielder_2', 'umpire', 'sv_id', 'vx
0', 'vy0', 'vz0', 'ax', 'ay', 'az', 'sz_top', 'sz_bot', 'h
it_distance_sc', 'launch_speed', 'launch_angle', 'effectiv
e_speed', 'release_spin_rate', 'release_extension', 'game_
pk', 'pitcher.1', 'fielder_2.1', 'fielder_3', 'fielder_4',
'fielder_5', 'fielder_6', 'fielder_7', 'fielder_8', 'field
er_9', 'release_pos_y', 'estimated_ba_using_speedangle',
'estimated_woba_using_speedangle', 'woba_value', 'woba_den
om', 'babip_value', 'iso_value', 'launch_speed_angle', 'at
_bat_number', 'pitch_number', 'pitch_name', 'home_score',
'away_score', 'bat_score', 'fld_score', 'post_away_score',
'post_home_score', 'post_bat_score', 'post_fld_score', 'if
_fielding_alignment', 'of_fielding_alignment', 'spin_axi
s', 'delta_home_win_exp', 'delta_run_exp']
```

```
pd.set_option('max_columns', 100)
pd.set_option('display.max_colwidth', None)
df.head(5)
```

executed in 75ms, finished 23:20:36 2021-09-08

	pitch_type	game_date	release_speed	release_pos
0	FS	2021-08-01	86.5	-1.24
1	FS	2021-08-01	88.3	-1.22
2	FC	2021-08-01	90.4	-1.37
3	SL	2021-08-01	86.2	-1.27
4	SI	2021-08-01	91.2	-1.23

Needs Exact Clarification:

pfx_x, pfx_z

plate_x, plate_z

vx0, vy0, vz0

ax, ay, az

sz_top, sz_bot

spin_axis

zone

Other potentially relevant features for our model:

---

pitch_type/pitch_name, pitch_number

release_speed

release_pos_x, release_pos_z

events, description

stand, p_throws

balls, strikes

release_spin_rate, release_extension

Documentation on the specifics of these metrics can be found at https://baseballsavant.mlb.com/csv-docs (https://baseballsavant.mlb.com/csv-docs)

```
#looking to remove "intent_ball", this is not a pitch th
#of making the hitter swing, so we could remove these
df['description'].unique().tolist()
```

executed in 14ms, finished 23:20:36 2021-09-08

```
['hit_into_play',
 'foul',
 'ball',
 'swinging_strike',
 'called_strike',
 'foul_tip',
 'blocked_ball',
 'swinging_strike_blocked',
 'hit_by_pitch',
 'foul_bunt',
 'missed_bunt']
```

From this list, we could make our target column for classification purposes. Namely, 1 for a swing and 0 for a non-swing to keep it simple. We could make this more sophisticated later if need be.. For example- can we get ahead in the count by forcing foul balls? Or maybe "hit_into_play" is not always bad- forcing an easy infield ground ball is also an easy way to get a batter out)

```
df['pitch_type'].unique().tolist()
```

executed in 14ms, finished 23:20:36 2021-09-08

```
['FS', 'FC', 'SL', 'SI', 'FF', 'CU', 'CH', 'KC', 'FT', na
n, 'FO', 'KN']
```

```
#checking for nulls in pitch location/speed data, this i
print(df['pitch_type'].isna().sum())
print(df['release_speed'].isna().sum())
print(df['plate_x'].isna().sum())
print(df['plate_z'].isna().sum())
```

executed in 14ms, finished 23:20:36 2021-09-08

```
51
48
49
49
```

```
df2 = df.loc[(df['pitch_type'].notnull()) & (df['release_
            & (df['plate_x'].notnull()) & (df['plate_z']
print(df2.shape)
df2.head()
```

executed in 61ms, finished 23:20:36 2021-09-08

```
(5754, 92)
```

	pitch_type	game_date	release_speed	release_pos
0	FS	2021-08-01	86.5	-1.24
1	FS	2021-08-01	88.3	-1.22
2	FC	2021-08-01	90.4	-1.37
3	SL	2021-08-01	86.2	-1.27
4	SI	2021-08-01	91.2	-1.23

Additionally, I am going to attempt to classify events as favorable or unfavorable for the pitcher, denoted by 1, 0, or -1.

I'll engineer a field "swing_for_stk_or_out" to denonte the difference between a swing, which could have a wide variety of outcomes (home run vs a called strike, for example). I'll only count swings that register a strike or an out as a 1 in this field, balls as a zero, and very negative pitcher outcomes as -1 (hits, walks, and sac flies). We'll have to look at both the 'description' column as well as the 'events' column to achieve this.

```
df2['swing_stk_or_out'] = 0
```

executed in 14ms, finished 23:20:36 2021-09-08

```
<ipython-input-13-ad41a3d6c6ec>:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a D
ataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/i
ndexing.html#returning-a-view-versus-a-copy)
  df2['swing_stk_or_out'] = 0
```

```
df2['events'].unique().tolist()
```

executed in 14ms, finished 23:20:36 2021-09-08

```
['double',
 nan,
 'strikeout',
 'field_out',
 'walk',
 'single',
 'home_run',
 'grounded_into_double_play',
 'fielders_choice',
 'field_error',
 'force_out',
 'hit_by_pitch',
 'fielders_choice_out',
 'triple',
 'sac_fly',
 'caught_stealing_2b',
 'strikeout_double_play',
 'sac_bunt']
```

```
df2.loc[df2['description'].isin(['hit_into_play', 'foul'
 'called_strike', 'foul_tip', 'swinging_strike_blocked',
 'missed_bunt']), 'swing_stk_or_out'] = 1
```

executed in 15ms, finished 23:20:36 2021-09-08

```
C:\Users\Darko\anaconda3\envs\learn-env\lib\site-packages
\pandas\core\indexing.py:1765: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  isetter(loc, value)
```

```
df2.loc[df2['events'].isin(['double', 'walk', 'single',
 'home_run', 'triple', 'sac_fly']), 'swing_stk_or_out']
```

executed in 14ms, finished 23:20:36 2021-09-08

```
df2.head(20)
```

executed in 90ms, finished 23:20:36 2021-09-08

14	SI	2021-08-01	91.3	-1.37

```
df2['pitch_type'].unique().tolist()
```

executed in 15ms, finished 23:20:36 2021-09-08

```
['FS', 'FC', 'SL', 'SI', 'FF', 'CU', 'CH', 'KC', 'FT', 'FO', 'KN']
```

I'm going to simplify this into 2 types of pitches- fastball or offspeed for our model.

```
fastballs = ['FC', 'SI', 'FF','FT']
offspeeds = ['FS', 'SL', 'CU', 'CH', 'KC', 'FO', 'KN']
```

executed in 14ms, finished 23:20:36 2021-09-08

```
df2['is_fastball'] = 0
```

executed in 14ms, finished 23:20:36 2021-09-08

```
<ipython-input-21-516ffa13e6da>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  df2['is_fastball'] = 0
```

```
df2.loc[df2['pitch_type'].isin(fastballs), 'is_fastball'
```

executed in 11ms, finished 23:20:36 2021-09-08

```
C:\Users\Darko\anaconda3\envs\learn-env\lib\site-packages
\pandas\core\indexing.py:1765: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  isetter(loc, value)
```

```
df2.head()
```

executed in 59ms, finished 23:20:36 2021-09-08

	pitch_type	game_date	release_speed	release_pos
0	FS	2021-08-01	86.5	-1.24
1	FS	2021-08-01	88.3	-1.22
2	FC	2021-08-01	90.4	-1.37
3	SL	2021-08-01	86.2	-1.27
4	SI	2021-08-01	91.2	-1.23

```
df2['is_pitcher_righty'] = 0
```

executed in 15ms, finished 23:20:36 2021-09-08

```
<ipython-input-24-7ac8acdb1d98>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  df2['is_pitcher_righty'] = 0
```

```
df2.loc[df2['p_throws'] == 'R', 'is_pitcher_righty'] = 1
```

executed in 14ms, finished 23:20:36 2021-09-08

```
C:\Users\Darko\anaconda3\envs\learn-env\lib\site-packages
\pandas\core\indexing.py:1765: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  isetter(loc, value)
```

```
model_df = df2[['is_fastball','is_pitcher_righty','relea
                'plate_x','plate_z','pfx_x','pfx_z','v
                'ax','ay','az','release_spin_rate',
                'strikes','balls','swing_stk_or_out']]
model_df.head(50)
```

executed in 45ms, finished 23:20:36 2021-09-08

	is_fastball	is_pitcher_righty	release_speed	plate_
0	0	1	86.5	0.28
1	0	1	88.3	-0.15
2	1	1	90.4	0.06
3	0	1	86.2	1.97
4	1	1	91.2	-0.64
5	1	1	93.1	0.00
6	0	1	84.9	-0.03
7	0	1	84.6	-0.20
8	1	1	92.8	-1.27
9	0	1	88.4	-1.40
10	0	1	85.6	-0.54
11	1	1	95.8	-1.52
12	0	1	84.8	-1.28
13	1	1	96.8	-1.61
14	1	1	91.3	-0.86
15	0	1	82.8	0.53
16	0	1	84.4	-0.94
17	1	1	95.5	-0.60
18	0	1	84.0	-0.81
19	1	1	92.6	0.43
20	1	1	95.1	-0.57
21	1	1	93.3	-1.39
22	0	1	83.0	0.76
23	1	1	95.3	-1.69
24	1	0	87.1	0.49
25	0	0	71.3	-1.28
26	0	0	70.1	-1.46
27	1	0	89.0	0.37
28	0	0	72.2	-0.04
29	1	0	88.6	-0.05
30	1	1	97.3	-0.36
31	0	1	91.2	-1.83
32	1	0	87.8	-0.52

	is_fastball	is_pitcher_righty	release_speed	plate_
33	0	1	91.9	-0.72
34	0	1	90.8	0.32
35	0	1	89.6	0.21
36	1	1	99.3	-2.11
37	0	1	89.8	-0.14
38	1	1	92.4	-1.04
39	1	1	92.0	-0.44
40	1	0	93.6	-0.36
41	1	0	93.4	0.20
42	1	1	93.1	0.04
43	0	1	75.9	-2.06
44	0	1	82.8	-0.67
45	0	1	91.6	-0.52
46	1	1	98.6	-0.93
47	0	1	87.1	-0.78
48	0	1	86.4	-0.25
49	1	1	94.5	0.78

```
model_df = model_df.iloc[::-1]
```

executed in 14ms, finished 23:20:36 2021-09-08

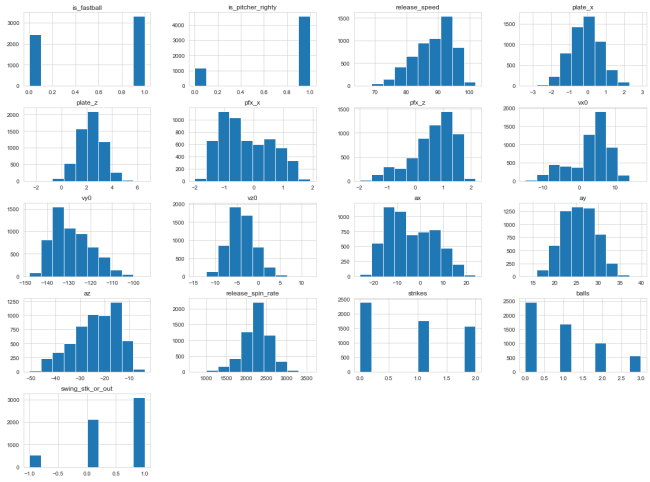
```
model_df.head(30)
```

executed in 29ms, finished 23:20:36 2021-09-08

	is_fastball	is_pitcher_righty	release_speed	plat
5804	0	1	82.8	0.01
5803	1	1	92.1	-0.2
5802	0	1	82.2	-1.1
5801	0	1	81.6	-0.9
5800	1	1	93.0	0.22
5799	1	1	93.9	-0.6
5798	1	1	91.1	-0.0
5797	0	1	76.1	0.91
5796	0	1	85.7	-0.4
5795	0	1	86.7	-0.7
5794	1	1	91.2	-0.2
5793	0	1	83.7	-1.4
5792	1	1	92.2	-0.1
5791	1	1	92.8	-0.8
5790	1	1	92.8	0.30
5789	1	1	92.6	0.29
5788	0	1	81.3	-1.0
5787	0	1	81.2	-0.0
5786	0	1	81.8	-0.6
5785	1	0	97.8	-2.0
5784	1	0	98.1	-0.4
5783	1	1	88.2	0.37
5782	1	1	93.0	-0.6
5781	1	1	88.4	-0.0
5780	1	1	91.3	0.82
5779	0	1	71.9	-1.0
5778	0	1	85.9	-0.4
5777	0	1	73.2	-0.8
5776	1	1	91.1	-0.0
5775	0	1	86.8	0.16

```
sns.set_style("whitegrid")
model_df.hist(figsize = (20,15));
```

executed in 2.29s, finished 23:20:39 2021-09-08




```
#checking for nulls/nans again here before modeling phas
model_df.isnull().sum(axis = 0)
```

executed in 15ms, finished 23:20:39 2021-09-08

```
is_fastball           0
is_pitcher_righty     0
release_speed         0
plate_x               0
plate_z               0
pfx_x                 0
pfx_z                 0
vx0                   0
vy0                   0
vz0                   0
ax                    0
ay                    0
az                    0
release_spin_rate    61
strikes               0
balls                 0
swing_stk_or_out      0
dtype: int64
```

```
print(model_df.shape)
model_df = model_df.dropna()
print(model_df.shape)

model_df.isnull().sum(axis = 0)
```

executed in 14ms, finished 23:20:39 2021-09-08

```
(5754, 17)
(5693, 17)
```

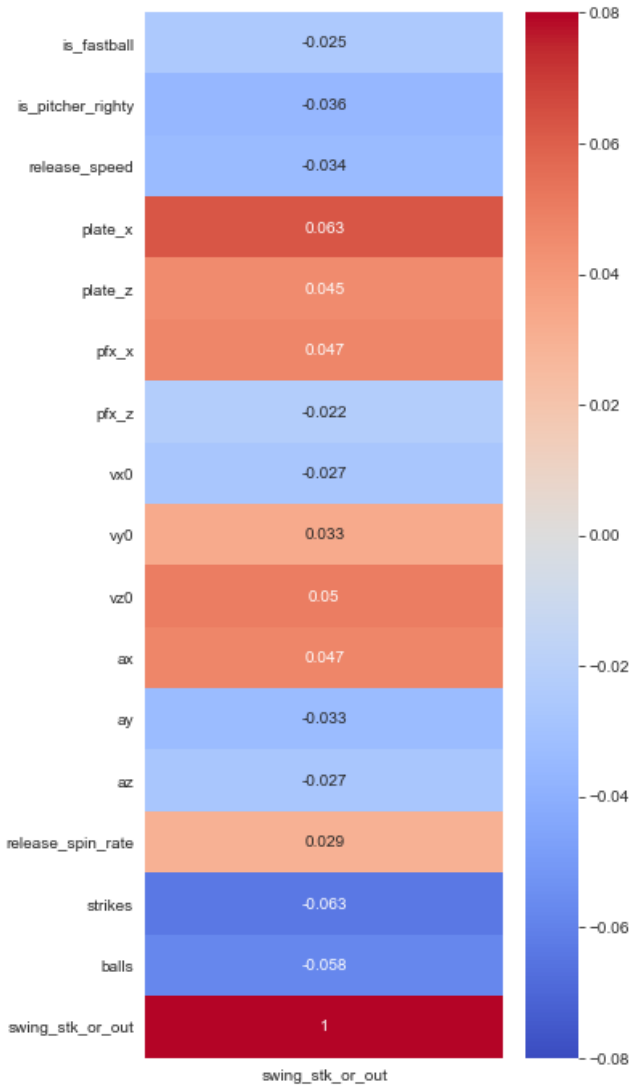
```
is_fastball          0
is_pitcher_righty    0
release_speed        0
plate_x              0
plate_z              0
pfx_x                0
pfx_z                0
vx0                  0
vy0                  0
vz0                  0
ax                   0
ay                   0
az                   0
release_spin_rate    0
strikes              0
balls                0
swing_stk_or_out     0
dtype: int64
```

Let's take a look at which features correlate the most to our target outcome ('swing_stk_or_out').

```
sns.set_style("whitegrid")

fig = plt.figure(figsize=(5, 12))
sns.heatmap(model_df.corr()[['swing_stk_or_out']], annot
            cmap="coolwarm", vmin=-0.08, vmax=0.08);
```

executed in 308ms, finished 23:20:39 2021-09-08



This is interesting to see, but I'm not sure how interpretable this is for some features. For example, increasingly negative values for 'pfx_x' indicate more movement from Right to Left, whereas higher positive values indicate more movement from Left to Right. We should probably be taking the absolute value of this field for being able to comment on Jesse Winker's ability to handle general horizonal movement, but losing information on the direction of the movement is not ideal here.

However, it is worth noting that higher "plate_x" and "plate_z" values tends towards positive pitcher outcomes, meaning that pitches up and/or inside (relative to our left-handed hitter Jesse Winker) more often result in a positive outcome for our pitcher.

Additionally, there is somewhat of a correlation between the handedness of the pitcher and the outcome. This indicates that lower values for "is_pitcher_righty" typically result in positive outcomes for our pitcher. In other words, left handed pitchers seem to have a better time against Jesse, who is also a left handed hitter. This is reassuring, as it is a commonly held belief that handedness mismatches tend to favor the batter. The data here confirms there is some value in this sentiment. This is why some players opt to switch-hit; even though they might be weaker on one side, they believe this handedness mismatch to be a more important determining factor in their success.

Additionally, there seems to also be a strong correlation between the number of balls or strikes and the outcome. This seems to indicate that Jesse may start to catch on to the pitch sequence during the at-bat, or be able to more easily predict the next pitch as an at-bat drags on. The best chance for success for our pitchers is to get him to guess early, as to not "show our hand" so to speak.

1.5 Basic Model

```
X1= model_df.drop('swing_stk_or_out', 1)
y1= model_df['swing_stk_or_out']

X_train, X_test, y_train, y_test = train_test_split(X1,y

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

executed in 15ms, finished 23:20:39 2021-09-08

```
((4269, 16), (1424, 16), (4269,), (1424,))
```

```
nb = GaussianNB()
nb.fit(X_train, y_train)
```

executed in 14ms, finished 23:20:39 2021-09-08

```
GaussianNB()
```

```
y_pred_train = nb.predict(X_train)
y_pred_test = nb.predict(X_test)
```

executed in 14ms, finished 23:20:39 2021-09-08

```
print(classification_report(y_test, y_pred_test))
```

executed in 14ms, finished 23:20:39 2021-09-08

```
              precision    recall  f1-score   support

          -1       0.27      0.24      0.25       142
           0       0.82      0.73      0.77       532
           1       0.74      0.81      0.78       750

    accuracy                           0.72      1424
   macro avg       0.61      0.59      0.60      1424
weighted avg       0.72      0.72      0.72      1424
```

```
print(classification_report(y_train, y_pred_train))
```

executed in 14ms, finished 23:20:39 2021-09-08

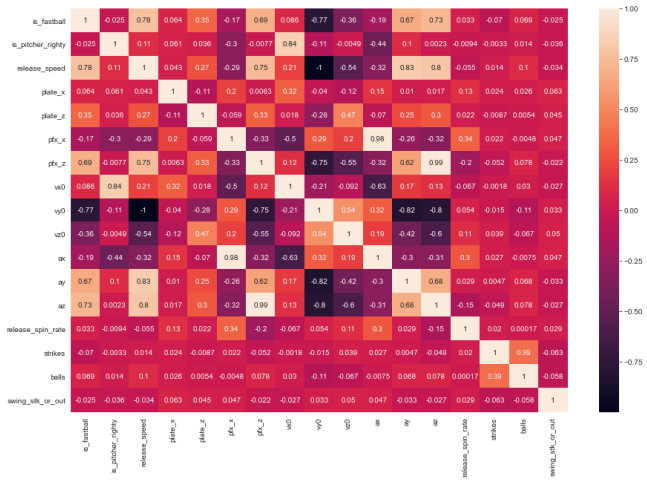
```
              precision    recall  f1-score   support

          -1       0.30      0.30      0.30       385
           0       0.83      0.75      0.79      1560
           1       0.77      0.82      0.80      2324

    accuracy                           0.75      4269
   macro avg       0.63      0.62      0.63      4269
weighted avg       0.75      0.75      0.75      4269
```

```
corr = model_df.corr()
plt.figure(figsize=(15,10))
ax= plt.subplot()
sns.heatmap(corr, ax=ax, annot=True);
```

executed in 2.05s, finished 23:20:41 2021-09-08



From this heatmap, I've gathered some useful information for our models moving forward. Firstly, "release_speed" is the same as "vy0", just in a different metric. Additionally, the acceleration fields ("ax", "ay", and "az" fields) are also highly correlated with the velocity fields. For this reason, I will be removing the acceleration fields from our model. Aside from this, the only other fields with high correlation with one another is the "is_fastball" and "vy0", namely the velocity, which totally makes sense. But I'll keep this for now as the value does not exceed 0.8 in either direction. Additionally, we have a class imbalance issue that I will try to address.

```
model_df = model_df.drop(['ax','ay','az','release_speed'
```

executed in 14ms, finished 23:20:41 2021-09-08

```
X_train = X_train.drop(['ax','ay','az','release_speed'],
```

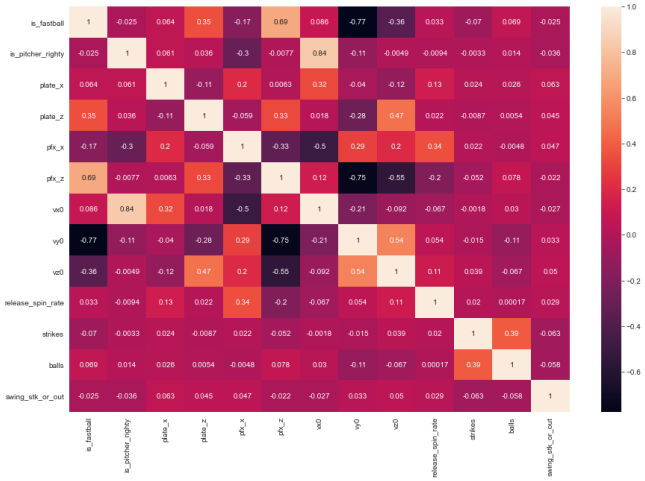
executed in 14ms, finished 23:20:41 2021-09-08

```
X_test = X_test.drop(['ax','ay','az','release_speed'], a
```

executed in 15ms, finished 23:20:41 2021-09-08

```
corr = model_df.corr()
plt.figure(figsize=(15,10))
ax= plt.subplot()
sns.heatmap(corr, ax=ax, annot=True);
```

executed in 1.33s, finished 23:20:43 2021-09-08




```
# y_score = nb.decision_function(X_test)
# fpr, tpr, thresholds = roc_curve(y_test, y_score)


# fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(8, 4)
# ax1.plot(fpr, tpr, color=BLUE, label='ROC Curve')
# ax1.plot([0,1], [0,1], color='red', ls=':') # Creating
# ax1.set(
#     title='ROC Curve',
#     xlabel='False Positive Rate',
#     ylabel='True Positive Rate'
# )
```

executed in 15ms, finished 23:20:43 2021-09-08

1.6 Model Iteration

I'm going to try a different approach now that we've removed some redundant features.

```
model_xgb = XGBClassifier(objective='multi:softmax', num
#scale_pos_weight
#model_xgb = XGBClassifier(scale_pos_weight=5)
# scale_pos_weight is the ratio of number of negative cl

model_xgb.fit(X_train, y_train)
scores = cross_val_score(model_xgb, X_train,y_train)
print('cross-val-scores')
print(scores)
print('mean')
print(round(scores.mean(), 5))
```

executed in 3.15s, finished 23:20:46 2021-09-08

```
cross-val-scores
[0.83021077 0.81733021 0.78922717 0.81498829 0.81594373]
mean
0.81354
```

```
pred_xgb_test = model_xgb.predict(X_test)
print(classification_report(y_test, pred_xgb_test))
```

executed in 27ms, finished 00:01:50 2021-09-09

```
              precision    recall  f1-score   support

          -1       0.72      0.23      0.35       142
           0       0.82      0.86      0.84       532
           1       0.78      0.85      0.82       750

    accuracy                           0.79      1424
   macro avg       0.77      0.65      0.67      1424
weighted avg       0.79      0.79      0.78      1424
```

```
pred_xgb_train = model_xgb.predict(X_train)
print(classification_report(y_train, pred_xgb_train))
```

executed in 41ms, finished 00:02:00 2021-09-09

```
              precision    recall  f1-score   support

          -1       1.00      0.98      0.99       385
           0       1.00      1.00      1.00      1560
           1       1.00      1.00      1.00      2324

    accuracy                           1.00      4269
   macro avg       1.00      0.99      1.00      4269
weighted avg       1.00      1.00      1.00      4269
```

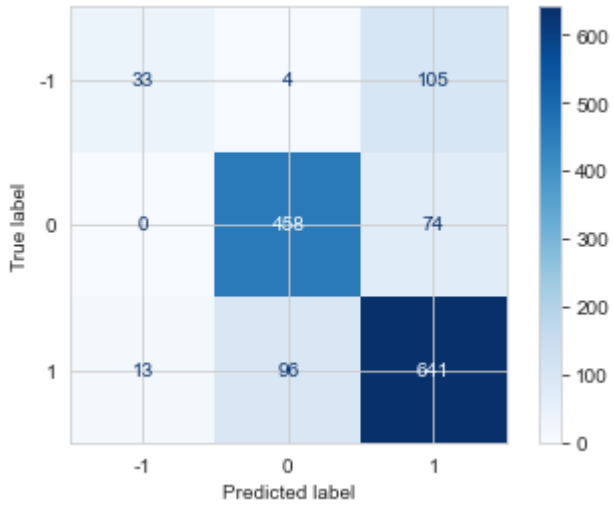
```
confusion_matrix(y_test,pred_xgb_test)
```

executed in 11ms, finished 00:02:09 2021-09-09

```
array([[ 33,   4, 105],
       [  0, 458,  74],
       [ 13,  96, 641]], dtype=int64)
```

```
sns.set_style("whitegrid")
plot_confusion_matrix(model_xgb, X_test, y_test, cmap='B
plt.show()
```

executed in 265ms, finished 00:08:11 2021-09-09

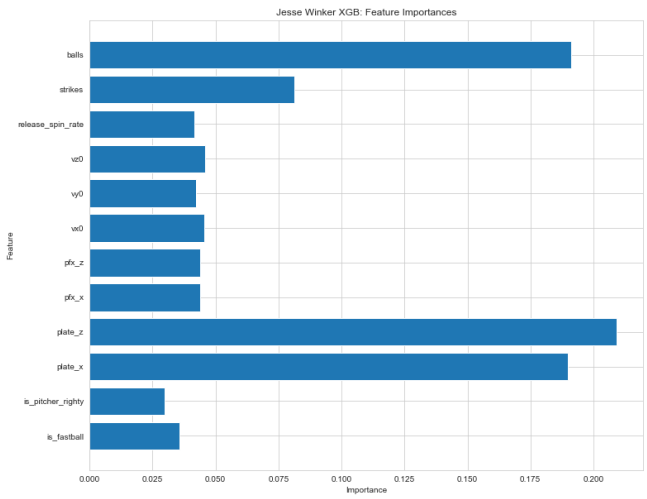


Definitely concerned that we are overfitting here with our use of XGBoost. We have nearly 100% across the board on our train data. Additionally, the "-1" outcomes are not fantastic. This means we are not predicting these very accurately. A recall of 0.23 means that this model is missing a lot of negative pitcher outcomes and the model is very "picky". It is predicting correctly on negative pitcher outcomes 72% of the time, but it is missing a LOT of negative pitcher outcome where it was predicted to be positive. However, since these events of "extremely negative outcomes for the pitcher" are much more sparse per pitch, I think this is acceptable as long as we are attempting to address the class imbalance.

```
sns.set_style("whitegrid")

fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111)
barlist = plt.barh(y=X_train.columns, width=model_xgb.fe
ax.set(
    title="Jesse Winker XGB: Feature Importances",
    ylabel="Feature",
    xlabel="Importance"
);
```

executed in 400ms, finished 02:39:18 2021-09-10



Moving forward, I'd like to take a look at the entire Reds starting hitting lineup and model it as well. From there, we can compare/contrast this with a single hitter's weaknesses and strengths.

1.7 Application To Entire Team

```
player_info__all_reds = playerid_lookup('india','jonatha

player_info_temp = playerid_lookup('winker','jesse')
player_info__all_reds= player_info__all_reds.append(play

player_info_temp = playerid_lookup('naquin','tyler')
player_info__all_reds= player_info__all_reds.append(play

player_info_temp = playerid_lookup('castellanos','nick')
player_info__all_reds= player_info__all_reds.append(play

player_info_temp = playerid_lookup('votto','joey')
player_info__all_reds= player_info__all_reds.append(play

player_info_temp = playerid_lookup('suarez','eugenio')
player_info__all_reds= player_info__all_reds.append(play

player_info_temp = playerid_lookup('farmer','kyle')
player_info__all_reds= player_info__all_reds.append(play

player_info_temp = playerid_lookup('barnhart','tucker')
player_info__all_reds= player_info__all_reds.append(play

player_info__all_reds.head(8)
```

executed in 152ms, finished 23:20:46 2021-09-08

	name_last	name_first	key_mlbam	key_retro	key_
0	india	jonathan	663697	indij001	indiaj
1	winker	jesse	608385	winkj002	winke
2	naquin	tyler	571980	naqut001	naqui
3	castellanos	nick	592206	castn001	caste
4	votto	joey	458015	vottj001	vottoj
5	suarez	eugenio	553993	suare001	suare
6	farmer	kyle	571657	farmk001	farme
7	barnhart	tucker	571466	barnt001	barnh

```
all_reds_df = pd.DataFrame()

for index, row in player_info__all_reds.iterrows():
    temp_id = row['key_mlbam']
    first_year = int(row['mlb_played_first'])
    if first_year < 2018:
        first_year = 2018 #4 year cap, so that players w
                          #are not over-influencing our
    start_date = str(first_year)+'-01-01'
    temp_df = statcast_batter(start_date,'2021-08-01', t
    all_reds_df= all_reds_df.append(temp_df, ignore_inde

print(all_reds_df.shape)
all_reds_df.head()
```

executed in 15.8s, finished 23:21:02 2021-09-08

```
Gathering Player Data
Gathering Player Data
Gathering Player Data
Gathering Player Data
Gathering Player Data
Gathering Player Data
Gathering Player Data
Gathering Player Data
(40524, 92)
```

	pitch_type	game_date	release_speed	release_pos
0	CU	2021-08-01	76.7	-1.30
1	SL	2021-08-01	87.2	-1.24
2	FS	2021-08-01	88.9	-1.28
3	SL	2021-08-01	87.1	-1.30
4	SL	2021-08-01	86.0	-1.25

40,000 pitches were given to the current active 2021 Reds lineup over the past 4 seasons. I am now going to clean this data and engineer our fields in the same way so that we can compare the models effectively.

```
print(all_reds_df['pitch_type'].isna().sum())
print(all_reds_df['release_speed'].isna().sum())
print(all_reds_df['plate_x'].isna().sum())
print(all_reds_df['plate_z'].isna().sum())
```

executed in 14ms, finished 23:21:02 2021-09-08

```
385
385
388
388
```

```
reds_cleaned_df = all_reds_df.loc[(all_reds_df['pitch_ty
            & (all_reds_df['plate_x'].notnull()) & (all_
print(reds_cleaned_df.shape)
reds_cleaned_df.head()
```

executed in 91ms, finished 23:21:02 2021-09-08

```
(40130, 92)
```

	pitch_type	game_date	release_speed	release_pos
0	CU	2021-08-01	76.7	-1.30
1	SL	2021-08-01	87.2	-1.24
2	FS	2021-08-01	88.9	-1.28
3	SL	2021-08-01	87.1	-1.30
4	SL	2021-08-01	86.0	-1.25

```
reds_cleaned_df['swing_stk_or_out'] = 0
```

executed in 14ms, finished 23:21:02 2021-09-08

```
<ipython-input-58-6e8988925a3a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  reds_cleaned_df['swing_stk_or_out'] = 0
```

```
reds_cleaned_df['events'].unique().tolist()
```

executed in 15ms, finished 23:21:02 2021-09-08

```
['strikeout',
 nan,
 'double',
 'field_out',
 'strikeout_double_play',
 'walk',
 'hit_by_pitch',
 'home_run',
 'grounded_into_double_play',
 'single',
 'force_out',
 'field_error',
 'caught_stealing_2b',
 'fielders_choice',
 'sac_bunt',
 'triple',
 'sac_fly',
 'fielders_choice_out',
 'double_play',
 'caught_stealing_3b',
 'other_out',
 'sac_fly_double_play']
```

```
reds_cleaned_df.loc[reds_cleaned_df['description'].isin(
 'called_strike', 'foul_tip', 'swinging_strike_blocked',
 'missed_bunt']), 'swing_stk_or_out'] = 1

reds_cleaned_df.loc[reds_cleaned_df['events'].isin(['dou
 'home_run', 'triple', 'sac_fly']), 'swing_stk_or_out']
```

executed in 14ms, finished 23:21:02 2021-09-08

```
C:\Users\Darko\anaconda3\envs\learn-env\lib\site-packages
\pandas\core\indexing.py:1765: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  isetter(loc, value)
```

```
reds_cleaned_df['is_fastball'] = 0

reds_cleaned_df.loc[reds_cleaned_df['pitch_type'].isin(f
```

executed in 15ms, finished 23:21:02 2021-09-08

```
<ipython-input-61-aa7fab14ad18>:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a D
ataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/i
ndexing.html#returning-a-view-versus-a-copy)
  reds_cleaned_df['is_fastball'] = 0
```

```
reds_cleaned_df['is_pitcher_righty'] = 0

reds_cleaned_df.loc[reds_cleaned_df['p_throws'] == 'R',
```

executed in 14ms, finished 23:21:02 2021-09-08

```
<ipython-input-62-45775579286b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Da
taFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy)
  reds_cleaned_df['is_pitcher_righty'] = 0
```

```
reds_cleaned_df = reds_cleaned_df.loc[(reds_cleaned_df['
```

executed in 29ms, finished 23:21:02 2021-09-08

Now that we have engineered some of these fields in the same way, let's just take the relevant fields for our model (minus the ones that are highly correlated with one another, removing redundant data) to get our dataframe 100% ready to be modeled.

```
reds_cleaned_df = reds_cleaned_df[['is_fastball','is_pit
                  'plate_x','plate_z','pfx_x','pfx_z','v
                  'release_spin_rate','strikes','balls',
print(reds_cleaned_df.shape)
```

executed in 15ms, finished 23:21:02 2021-09-08

```
(39667, 13)
```

```
reds_cleaned_df = reds_cleaned_df.iloc[::-1]
```

executed in 15ms, finished 23:21:02 2021-09-08

```
reds_cleaned_df.head(50)
```

executed in 30ms, finished 23:21:02 2021-09-08

	is_fastball	is_pitcher_righty	plate_x	plate_z	
40523	1	1	0.79	1.78	(
40522	1	1	-0.44	4.20	-
40521	1	1	-0.26	3.04	(
40520	1	1	-0.43	3.96	-
40519	1	1	-0.71	2.97	-
40518	0	1	-0.72	1.15	-
40517	1	1	-0.24	2.82	-
40516	1	1	1.60	1.23	(
40515	1	1	0.74	2.43	-
40514	1	1	-1.24	2.03	-
40513	0	1	-0.81	1.89	-
40512	1	1	-1.79	2.78	-
40511	1	1	-1.39	3.33	-
40510	1	1	0.07	2.28	-
40509	0	0	-1.13	1.51	(
40508	0	0	-0.15	0.14	'
40507	1	0	-0.50	1.70	'
40506	0	0	0.09	1.82	-
40505	0	1	-1.34	4.45	(
40504	0	1	-1.28	2.49	-
40503	0	1	-0.40	1.99	-
40502	1	1	0.61	4.00	-
40501	1	1	0.12	2.32	-
40500	0	1	-0.13	1.18	-
40499	1	1	0.03	1.71	-
40498	1	1	1.45	3.53	-
40497	0	1	-0.27	2.12	-
40496	0	1	-2.13	3.42	-
40495	0	1	0.12	2.53	(
40494	0	1	0.87	0.81	(
40493	0	1	-0.18	2.07	-
40492	0	1	-1.09	3.26	(
40491	1	1	0.34	2.53	(
40490	1	1	-0.03	2.93	-
40489	0	1	-0.74	2.18	(
40488	1	1	0.35	3.17	-

	is_fastball	is_pitcher_righty	plate_x	plate_z	
40487	1	1	0.53	1.20	(
40486	1	1	0.96	1.49	(
40485	1	1	0.78	2.41	-
40484	1	1	-0.70	3.85	-
40483	1	1	1.36	2.79	-
40482	1	1	-0.75	1.61	-
40481	1	1	-0.15	1.29	-
40480	1	1	-0.75	0.76	-
40479	1	0	-0.57	2.01	(
40478	1	1	-0.71	3.95	-
40477	1	1	-0.20	2.85	-
40476	1	1	-0.80	3.55	-
40475	1	0	-0.44	1.01	'
40474	0	0	0.50	2.27	-

1.8 Full Team Modeling

```
X1_reds= reds_cleaned_df.drop('swing_stk_or_out', 1)
y1_reds = reds_cleaned_df['swing_stk_or_out']

X_train_reds, X_test_reds, y_train_reds, y_test_reds = t

X_train_reds.shape, X_test_reds.shape, y_train_reds.shap
```

executed in 22ms, finished 23:21:03 2021-09-08

```
((29750, 12), (9917, 12), (29750,), (9917,))
```

```
reds_model_xgb = XGBClassifier(objective='multi:softmax'
reds_model_xgb.fit(X_train_reds, y_train_reds)
scores = cross_val_score(reds_model_xgb, X_train_reds,y_
print('cross-val-scores')
print(scores)
print('mean')
print(round(scores.mean(), 5))
```

executed in 17.4s, finished 23:21:20 2021-09-08

```
cross-val-scores
[0.80403361 0.8010084  0.80319328 0.80420168 0.80436975]
mean
0.80336
```

```
pred_xgb_reds = reds_model_xgb.predict(X_test_reds)
print(classification_report(y_test_reds, pred_xgb_reds))
```

executed in 60ms, finished 23:21:20 2021-09-08

```
              precision    recall  f1-score   support

          -1       0.72      0.22      0.33       851
           0       0.81      0.84      0.83      3459
           1       0.80      0.87      0.84      5607

    accuracy                           0.81      9917
   macro avg       0.78      0.64      0.67      9917
weighted avg       0.80      0.81      0.79      9917
```

```
pred_xgb_reds = reds_model_xgb.predict(X_train_reds)
print(classification_report(y_train_reds, pred_xgb_reds))
```

executed in 153ms, finished 23:21:20 2021-09-08

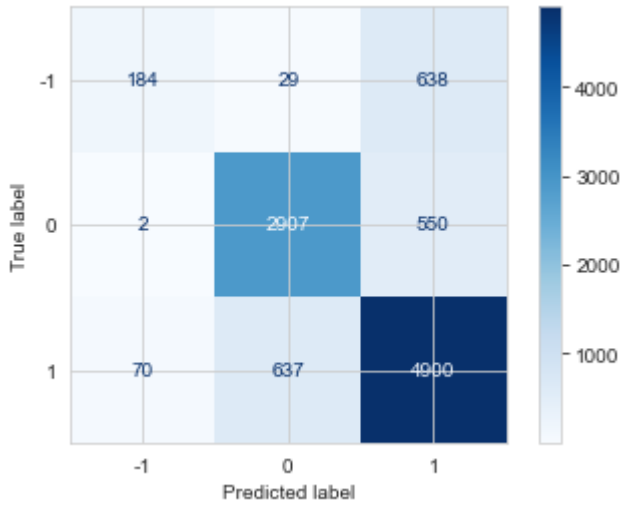
```
              precision    recall  f1-score   support

          -1       0.98      0.33      0.50      2519
           0       0.89      0.91      0.90     10613
           1       0.86      0.94      0.90     16618

    accuracy                           0.87     29750
   macro avg       0.91      0.73      0.76     29750
weighted avg       0.88      0.87      0.86     29750
```

```
sns.set_style("whitegrid")
plot_confusion_matrix(reds_model_xgb, X_test_reds, y_tes
plt.show()
```

executed in 289ms, finished 00:17:06 2021-09-09

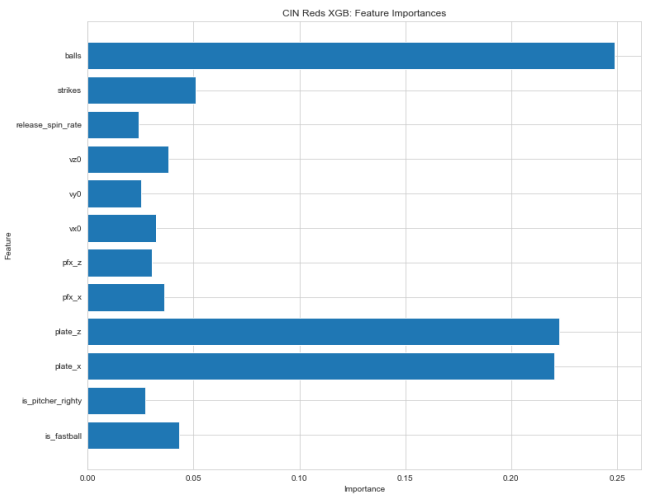


Our confusion matrix looks very similar to our last model, and we have similar precision and recall values for the "negative pitcher outcomes" class. However, this model is not overfitting nearly as much. Values around 87% in our training data indicate that we seem to have addressed the overfitting for this model. Unfortunately, the confusion matrix looks similar and we have similar issues in this model- although we have a high rate for predicting true positives, we seem to have issues predicting negative pitcher outcomes when we think they will result in a positive outcome.

```
sns.set_style("whitegrid")

fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111)
barlist = plt.barh(y=X_train_reds.columns, width=reds_mo
ax.set(
    title="CIN Reds XGB: Feature Importances",
    ylabel="Feature",
    xlabel="Importance"
);
```

executed in 369ms, finished 02:39:39 2021-09-10



The results are actually extremely similar. As we can see, there are 3 main determining factors that I can sum up in just a few words- location, location, location. The location of the pitch, namely the plate_z and plate_x values, are the 2nd highest determining factors in the outcome of a pitch having a positive or negative outcome for the pitcher. This is a similar feature importance value we saw just looking at Jesse Winker's model as well. Similarly, the # of balls in the count has the highest importance for the model against the Reds. This could mean several things. Firstly, if a pitcher has thrown a high number of balls in the at bat, the hitter

now has a HUGE advantage as he knows a pitch in the strike zone is coming- something he can potentially do damage with. Having a 0-0 count, a batter essentially has no idea what the pitcher could be going with in the at-bat. No previous pitches to predict off of, and the batter unsure if the pitcher wants to attack the strike zone early or get him chasing at balls early.

Additionally, we once again see that the handedness of the pitcher has the LOWEST importance in both Jesse Winker and the entire Red's team's model. However, because the batters in the Reds lineup are both right and left handed, this may seem to make sense. As for Jesse Winker, this may just speak to his ability to be able to hit extremely well against both right and left handed pitchers.

This has given me an idea. I want to turn this problem into a binary classification problem to potentially simplify the model. Having this "neutral" outcome of throwing a "ball" might be confusing our model, because really it's only a positive/negative outcome depending on the context. I'm going to drop the "neutral" outcome pitches and see what our model comes up with for a binary classification model for the entire Reds team.

1.9 "No Neutrals" Model

```
no_neutrals_df = reds_cleaned_df
no_neutrals_df.head(5)
```

executed in 15ms, finished 23:21:20 2021-09-08

	is_fastball	is_pitcher_righty	plate_x	plate_z	
40523	1	1	0.79	1.78	(
40522	1	1	-0.44	4.20	-
40521	1	1	-0.26	3.04	(
40520	1	1	-0.43	3.96	-
40519	1	1	-0.71	2.97	-

```
no_neutrals_df['favors_pitcher'] = no_neutrals_df['swing
no_neutrals_df = no_neutrals_df.drop('swing_stk_or_out',
```

executed in 14ms, finished 23:21:20 2021-09-08

```
no_neutrals_df.head(20)
```

executed in 29ms, finished 23:21:21 2021-09-08

	is_fastball	is_pitcher_righty	plate_x	plate_z	
40523	1	1	0.79	1.78	(
40522	1	1	-0.44	4.20	-
40521	1	1	-0.26	3.04	(
40520	1	1	-0.43	3.96	-
40519	1	1	-0.71	2.97	-
40518	0	1	-0.72	1.15	-
40517	1	1	-0.24	2.82	-
40516	1	1	1.60	1.23	(
40515	1	1	0.74	2.43	-
40514	1	1	-1.24	2.03	-
40513	0	1	-0.81	1.89	-
40512	1	1	-1.79	2.78	-
40511	1	1	-1.39	3.33	-
40510	1	1	0.07	2.28	-
40509	0	0	-1.13	1.51	(
40508	0	0	-0.15	0.14	'
40507	1	0	-0.50	1.70	'
40506	0	0	0.09	1.82	-
40505	0	1	-1.34	4.45	(
40504	0	1	-1.28	2.49	-

```
no_neutrals_df = no_neutrals_df[no_neutrals_df['favors_p
```

executed in 15ms, finished 23:21:21 2021-09-08

```
no_neutrals_df.shape
```

executed in 14ms, finished 23:21:21 2021-09-08

```
(25595, 13)
```

```
no_neutrals_df['favors_hitter_binary'] = 1
no_neutrals_df.loc[no_neutrals_df['favors_pitcher']==1,
```

executed in 14ms, finished 23:21:21 2021-09-08

```
no_neutrals_df = no_neutrals_df.drop('favors_pitcher',ax
```

executed in 15ms, finished 23:21:21 2021-09-08

```
no_neutrals_df.head(20)
```

executed in 30ms, finished 23:21:21 2021-09-08

	is_fastball	is_pitcher_righty	plate_x	plate_z	
40523	1	1	0.79	1.78	(
40521	1	1	-0.26	3.04	(
40519	1	1	-0.71	2.97	-
40518	0	1	-0.72	1.15	-
40517	1	1	-0.24	2.82	-
40515	1	1	0.74	2.43	-
40513	0	1	-0.81	1.89	-
40510	1	1	0.07	2.28	-
40509	0	0	-1.13	1.51	(
40507	1	0	-0.50	1.70	'
40506	0	0	0.09	1.82	-
40504	0	1	-1.28	2.49	-
40503	0	1	-0.40	1.99	-
40501	1	1	0.12	2.32	-
40500	0	1	-0.13	1.18	-
40499	1	1	0.03	1.71	-
40497	0	1	-0.27	2.12	-
40495	0	1	0.12	2.53	(
40493	0	1	-0.18	2.07	-
40491	1	1	0.34	2.53	(

```
X1_nn= no_neutrals_df.drop('favors_hitter_binary', 1)
y1_nn = no_neutrals_df['favors_hitter_binary']

X_train_nn, X_test_nn, y_train_nn, y_test_nn = train_tes

X_train_nn.shape, X_test_nn.shape, y_train_nn.shape, y_t
```

executed in 15ms, finished 23:21:21 2021-09-08

```
((19196, 12), (6399, 12), (19196,), (6399,))
```

```
no_neutrals_df['favors_hitter_binary'].value_counts()
```

executed in 14ms, finished 23:21:21 2021-09-08

```
0    22225
1     3370
Name: favors_hitter_binary, dtype: int64
```

```
nn_model_xgb = XGBClassifier(scale_pos_weight=6) #6x val
nn_model_xgb.fit(X_train_nn, y_train_nn)
scores = cross_val_score(nn_model_xgb, X_train_nn,y_trai
print('cross-val-scores')
print(scores)
print('mean')
print(round(scores.mean(), 5))
```

executed in 3.85s, finished 23:21:24 2021-09-08

```
cross-val-scores
[0.79322917 0.798385   0.80854389 0.81557697 0.81166971]
mean
0.80548
```

```
pred_xgb_nn = nn_model_xgb.predict(X_test_nn)
print(classification_report(y_test_nn, pred_xgb_nn))
```

executed in 44ms, finished 23:21:25 2021-09-08

```
              precision    recall  f1-score   support

           0       0.91      0.86      0.88      5588
           1       0.29      0.39      0.33       811

    accuracy                           0.80      6399
   macro avg       0.60      0.62      0.61      6399
weighted avg       0.83      0.80      0.81      6399
```

```
pred_xgb_nn = nn_model_xgb.predict(X_train_nn)
print(classification_report(y_train_nn, pred_xgb_nn))
```

executed in 61ms, finished 23:21:25 2021-09-08

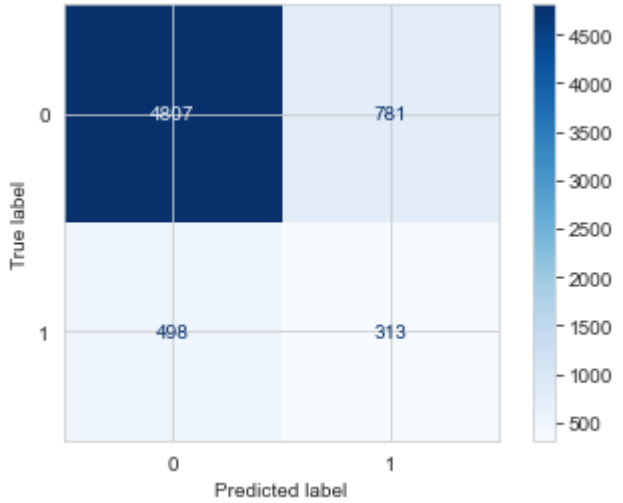
```
              precision    recall  f1-score   support

           0       0.99      0.92      0.95     16637
           1       0.64      0.93      0.76      2559

    accuracy                           0.92     19196
   macro avg       0.81      0.92      0.86     19196
weighted avg       0.94      0.92      0.93     19196
```

```
sns.set_style("whitegrid")
plot_confusion_matrix(nn_model_xgb, X_test_nn, y_test_nn
plt.show()
```

executed in 251ms, finished 00:22:50 2021-09-09

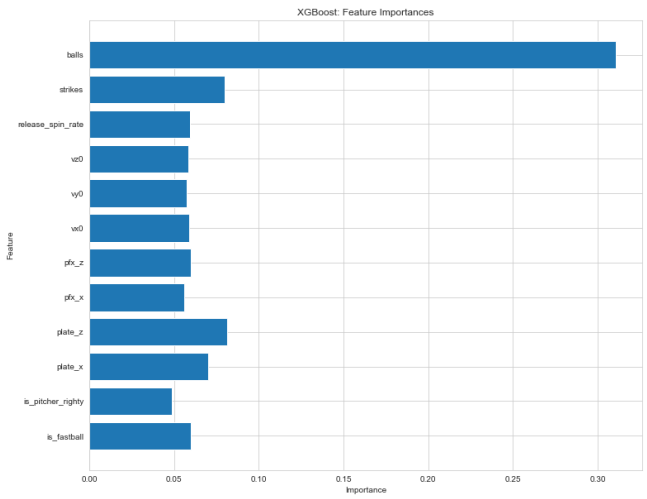


Although the confusion matrix here looks different, it is essentially just inverted. This is because we flipped one of our engineered fields into a positive hitter outcome instead, in order to address the binary class imbalance through the use of the "scale_pos_weight" parameter. If only there was a "scale_neg_weight", I could have avoiding needing to do this. Regardless, the same issue has carried over into our binary classification problem as well. In fact, it actually significantly hurt our ability to predict true positives in this model. Namely, our model is just NOT predicting negative hitter outcomes correctly in 70% of all cases, which is not ideal.

```
sns.set_style("whitegrid")

fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111)
barlist = plt.barh(y=X_train_nn.columns, width=nn_model_
ax.set(
    title="XGBoost: Feature Importances",
    ylabel="Feature",
    xlabel="Importance"
);
```

executed in 292ms, finished 23:21:25 2021-09-08



Definitely seeing a LOT of similarities in this model's feature importances and classification reports with very few differences. Still top 3 importances in "balls" and location data, just less importances on location than in previous models. However, the number of strikes in this model is one of the most important predictors for pitcher success, which makes sense, as 3 strikes is a strikeout. That's alright though. It was worth investigating the "scale_pos_weight" and the low precision, recall, and f1-scores for the minority class. Still predicting at about 81.2% of the time as well, essentially no improvement and losing information (the 3rd class of neutral outcome) in this binary classification.

1.10 Random Forest

```
reds_cleaned_df = reds_cleaned_df.drop('swing_stk_or_out
```

executed in 14ms, finished 23:21:25 2021-09-08

```
reds_cleaned_df.head(40)
```

executed in 29ms, finished 23:21:25 2021-09-08

	is_fastball	is_pitcher_righty	plate_x	plate_z
40523	1	1	0.79	1.78
40522	1	1	-0.44	4.20
40521	1	1	-0.26	3.04
40520	1	1	-0.43	3.96
40519	1	1	-0.71	2.97
40518	0	1	-0.72	1.15
40517	1	1	-0.24	2.82
40516	1	1	1.60	1.23
40515	1	1	0.74	2.43
40514	1	1	-1.24	2.03
40513	0	1	-0.81	1.89
40512	1	1	-1.79	2.78
40511	1	1	-1.39	3.33
40510	1	1	0.07	2.28
40509	0	0	-1.13	1.51
40508	0	0	-0.15	0.14
40507	1	0	-0.50	1.70
40506	0	0	0.09	1.82
40505	0	1	-1.34	4.45
40504	0	1	-1.28	2.49
40503	0	1	-0.40	1.99
40502	1	1	0.61	4.00
40501	1	1	0.12	2.32
40500	0	1	-0.13	1.18
40499	1	1	0.03	1.71
40498	1	1	1.45	3.53
40497	0	1	-0.27	2.12
40496	0	1	-2.13	3.42
40495	0	1	0.12	2.53
40494	0	1	0.87	0.81
40493	0	1	-0.18	2.07
40492	0	1	-1.09	3.26
40491	1	1	0.34	2.53
40490	1	1	-0.03	2.93
40489	0	1	-0.74	2.18
40488	1	1	0.35	3.17

	is_fastball	is_pitcher_righty	plate_x	plate_z
40487	1	1	0.53	1.20
40486	1	1	0.96	1.49
40485	1	1	0.78	2.41
40484	1	1	-0.70	3.85

```
bootstrap = [True, False]

param_grid = {
    "n_estimators": [25],
    "max_depth": [25],
    "min_samples_leaf": [2],
    "bootstrap": bootstrap,
}

rf = RandomForestClassifier()

rf_model = GridSearchCV(estimator=rf, param_grid=param_g
rf_model.fit(X_train_reds, y_train_reds)

#X_train_reds, X_test_reds, y_train_reds, y_test_reds =
```

executed in 4.10s, finished 23:38:53 2021-09-08

```
Fitting 2 folds for each of 2 candidates, totalling 4 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 c
oncurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:
2.3s
[Parallel(n_jobs=-1)]: Done   2 out of   4 | elapsed:
2.3s remaining:    2.3s
[Parallel(n_jobs=-1)]: Done   4 out of   4 | elapsed:
2.6s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   4 out of   4 | elapsed:
2.6s finished


GridSearchCV(cv=2, estimator=RandomForestClassifier(), n_j
obs=-1,
             param_grid={'bootstrap': [True, False], 'max_
depth': [25],
                         'min_samples_leaf': [2], 'n_estim
ators': [25]},
             verbose=10)
```

```
y_pred = rf_model.predict(X_test_reds)
y_pred_label = list(y_pred)

print("Training--set score for SVM: %f" % rf_model.score
print("Testing---set score for SVM: %f" % rf_model.score
```

executed in 230ms, finished 23:39:02 2021-09-08

```
Training set score for SVM: 0.938756
Testing  set score for SVM: 0.800645
```

```
# fig = plt.figure(figsize=(15, 10))
# plot_tree(rf_model.estimator,
#           feature_names=X1_reds.columns,
#           class_names=["Pitcher Result"],
#           filled=True, impurity=True,
#           rounded=True)
```

executed in 11ms, finished 23:50:21 2021-09-08

1.11 Results and Conclusion

What are the most important metrics that go into a pitch against the Reds? **The Current Ball Count and Pitch Location**

What is the least important metric that goes into a pitch against the Reds? **Spin Rate (surprisingly!!)**

To my surprise, the spin rate on a pitch seems to have little to do with the overall outcome of the pitch itself against both Jesse Winker and the Reds as a whole. Additionally, the current ball count seems to have the most impact on predicting the outcome, according to our model. This makes sense, as a high ball count indicates that the batter is in a highly favorable position. The pitcher now needs to successfully throw a ball in the strike zone, or suffer another negative pitcher outcome in the form of a walk. Pitch location also makes sense as an important predictive factor in our model. A pitch in the dirt is unlikely to be swung at, resulting in a neutral outcome. But a strike is within a certain combination of values. Pitches right down the middle are likely to be crushed, but strikes on the corners are more likely to result in a strike. Although the model has some recall issues, namely being unable to detect negative pitcher outcomes correctly, the overall accuracy is around 81%.

1.12 Future Work

I am now extremely curious as to whether this "spin rate" discussion just doesn't apply to the Reds, or if other teams are in the same boat. I'd love to experiment further with more modeling that allows me to view feature importances- models for specific teams to compare and contrast, or perhaps a model for all MLB starters. The Reds have a very good record this year, and this may be partly why- they're really able to handle balls with higher spin rate with less issues than others. Only more modeling and investigation will tell.

I'd also think that the player-specific models will do well in a game setting. For these, we could narrow down our models to just specific pitch types and locations, to be able to create a tool that will be able to tell us the success rate of our next pitch, giving us a live decision in a game scenario. I think that would be extremely valuable from an organization standpoint.