Mariners Challenge

March 16, 2020

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import xgboost as xgb
from xgboost import XGBClassifier

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, precision_recall_curve, auc,␣
↪matthews_corrcoef, accuracy_score

from joblib import load
```

[2]:
```
train_df = pd.read_csv('../Data/2020-train.csv')
```

[3]:
```
train_df.head()
```

[3]:
```
  pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
0   d7e3acce        Right  32678d8d       Right   a4833794  f88d09f4
1   44ec1bf5        Right  81d51733        Left   f60d6ea5  b67d862c
2   44d87ee6         Left  8eefccb7       Right   a9b8b538  13993d26
3   ff6adae0        Right  8f8ab5af       Right   e569ec39  0d8ba4bb
4   c70c96e5        Right  10874746       Right   a5ce1bf6  94a4c552

  catcher_id  inning  top_bottom  outs  …  zone_speed  vert_approach_angle  \
0   83cdf9ff       3           1   0.0  …   86.024200             -4.37258
1   a126f66f       6           2   0.0  …   89.458199             -4.90467
2   9db4e46f       5           2   2.0  …   75.593597             -6.00728
3   bbbfd290       5           1   2.0  …   76.396400             -9.50640
4   75087ec8       8           1   2.0  …   83.215302             -4.53233

   horz_approach_angle  zone_time       x55  y55      z55 pitch_type  \
0             1.429580   0.404622 -0.059343   55  6.03322         FA
1            -2.148410   0.385719 -2.148680   55  6.23380         FA
2            -0.122044   0.463953  1.300450   55  6.14750         CH
```

```
3           -2.581980   0.458471 -1.659590   55   6.60043          CU
4           -0.268188   0.415965 -1.526170   55   4.77332          FA

     pitch_call  pitch_id
0        InPlay  42fce2f6
1        InPlay  3e9cda86
2    BallCalled  f129a6cd
3        InPlay  03e9bc05
4  StrikeCalled  48feb675

[5 rows x 36 columns]
```

[4]:
```python
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 582205 entries, 0 to 582204
Data columns (total 36 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   pitcher_id          582205 non-null  object
 1   pitcher_side        582205 non-null  object
 2   batter_id           582205 non-null  object
 3   batter_side         582205 non-null  object
 4   stadium_id          582205 non-null  object
 5   umpire_id           582205 non-null  object
 6   catcher_id          582205 non-null  object
 7   inning              582205 non-null  int64
 8   top_bottom          582205 non-null  int64
 9   outs                582053 non-null  float64
 10  balls               582205 non-null  int64
 11  strikes             582205 non-null  int64
 12  release_speed       582093 non-null  float64
 13  vert_release_angle  582093 non-null  float64
 14  horz_release_angle  582093 non-null  float64
 15  spin_rate           573194 non-null  float64
 16  spin_axis           582093 non-null  float64
 17  tilt                580953 non-null  object
 18  rel_height          582093 non-null  float64
 19  rel_side            582093 non-null  float64
 20  extension           582093 non-null  float64
 21  vert_break          582093 non-null  float64
 22  induced_vert_break  582093 non-null  float64
 23  horz_break          582093 non-null  float64
 24  plate_height        582139 non-null  float64
 25  plate_side          582139 non-null  float64
 26  zone_speed          582093 non-null  float64
 27  vert_approach_angle 582093 non-null  float64
```

```
 28  horz_approach_angle  582093 non-null  float64
 29  zone_time            582093 non-null  float64
 30  x55                  582093 non-null  float64
 31  y55                  582205 non-null  int64
 32  z55                  582093 non-null  float64
 33  pitch_type           581720 non-null  object
 34  pitch_call           582205 non-null  object
 35  pitch_id             582205 non-null  object
dtypes: float64(20), int64(5), object(11)
memory usage: 159.9+ MB
```

1 Data Cleaning

First let's get rid of the null values floating around in the data set. There are a total of 582,205 entries in the initial data set, some columns are full but some columns have null values in them. Starting with release speed, it does not seem like there are many null values, let's take a look.

[5]:
```python
train_df[train_df['release_speed'].isnull()]
```

[5]:
```
       pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
3405     8759809c        Right  00bde845       Right   80756f45  9c6cbb5e
12334    f6d227a5        Right  69426d29        Left   a3f610ed  9c6cbb5e
13895    b74a40d9        Right  00bde845       Right   a3f610ed  9c6cbb5e
16260    47032f76         Left  20bf9444       Right   c9712626  9c6cbb5e
17008    96a1cebe        Right  76c0475e       Right   a5ce1bf6  9c6cbb5e
…             …            …        …           …          …         …
564267   b3336756         Left  bbbfd290       Right   9b5daeaf  9c6cbb5e
564950   c3ededfb        Right  8f8ab5af       Right   b20853fa  9c6cbb5e
567947   ad6bf2a7        Right  066e327d       Right   fe6b0f40  9c6cbb5e
571936   0a606b2d        Right  336a9f05        Left   1a39a252  9c6cbb5e
573697   f556cf83         Left  30300714       Right   b20853fa  9c6cbb5e

       catcher_id  inning  top_bottom  outs  …  zone_speed  \
3405     9c6cbb5e       5           2   0.0  …         NaN
12334    9c6cbb5e       5           2   2.0  …         NaN
13895    9c6cbb5e       3           1   1.0  …         NaN
16260    9c6cbb5e      16           2   1.0  …         NaN
17008    9c6cbb5e       9           2   0.0  …         NaN
…             …     …          …     …   …  …
564267   9c6cbb5e       9           1   2.0  …         NaN
564950   9c6cbb5e       1           1   2.0  …         NaN
567947   9c6cbb5e       5           2   0.0  …         NaN
571936   9c6cbb5e       9           1   1.0  …         NaN
573697   9c6cbb5e       7           1   0.0  …         NaN

        vert_approach_angle  horz_approach_angle  zone_time  x55  y55  z55  \
3405                    NaN                  NaN        NaN  NaN   55  NaN
```

```
12334                   NaN                     NaN        NaN  NaN    55  NaN
13895                   NaN                     NaN        NaN  NaN    55  NaN
16260                   NaN                     NaN        NaN  NaN    55  NaN
17008                   NaN                     NaN        NaN  NaN    55  NaN
…                     …                     …         …   …    …   …
564267                  NaN                     NaN        NaN  NaN    55  NaN
564950                  NaN                     NaN        NaN  NaN    55  NaN
567947                  NaN                     NaN        NaN  NaN    55  NaN
571936                  NaN                     NaN        NaN  NaN    55  NaN
573697                  NaN                     NaN        NaN  NaN    55  NaN

       pitch_type    pitch_call  pitch_id
3405          NaN  StrikeCalled  1d66612a
12334         NaN        InPlay  beb842a8
13895         NaN        InPlay  e271ef9d
16260          FA        InPlay  49f26761
17008          SL    BallCalled  ca8c6341
…           …           …        …
564267         CH        InPlay  8d21a585
564950        NaN    BallCalled  142a06b3
567947        NaN  StrikeCalled  5326a5f6
571936        NaN    BallCalled  3c4e2fe4
573697        NaN  StrikeCalled  763a95f4

[112 rows x 36 columns]
```

Only 112, I feel comfortable dropping these and it not affecting the integrity of the data.

[6]:
```python
train_df = train_df.drop(train_df[train_df['release_speed'].isnull()].index)
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 582093 entries, 0 to 582204
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   pitcher_id            582093 non-null  object
 1   pitcher_side          582093 non-null  object
 2   batter_id             582093 non-null  object
 3   batter_side           582093 non-null  object
 4   stadium_id            582093 non-null  object
 5   umpire_id             582093 non-null  object
 6   catcher_id            582093 non-null  object
 7   inning                582093 non-null  int64
 8   top_bottom            582093 non-null  int64
 9   outs                  581941 non-null  float64
 10  balls                 582093 non-null  int64
 11  strikes               582093 non-null  int64
```

```
 12  release_speed        582093 non-null  float64
 13  vert_release_angle   582093 non-null  float64
 14  horz_release_angle   582093 non-null  float64
 15  spin_rate            573194 non-null  float64
 16  spin_axis            582093 non-null  float64
 17  tilt                 580953 non-null  object
 18  rel_height           582093 non-null  float64
 19  rel_side             582093 non-null  float64
 20  extension            582093 non-null  float64
 21  vert_break           582093 non-null  float64
 22  induced_vert_break   582093 non-null  float64
 23  horz_break           582093 non-null  float64
 24  plate_height         582093 non-null  float64
 25  plate_side           582093 non-null  float64
 26  zone_speed           582093 non-null  float64
 27  vert_approach_angle  582093 non-null  float64
 28  horz_approach_angle  582093 non-null  float64
 29  zone_time            582093 non-null  float64
 30  x55                  582093 non-null  float64
 31  y55                  582093 non-null  int64
 32  z55                  582093 non-null  float64
 33  pitch_type           581674 non-null  object
 34  pitch_call           582093 non-null  object
 35  pitch_id             582093 non-null  object
dtypes: float64(20), int64(5), object(11)
memory usage: 164.3+ MB
```

Now let's take a look at outs. Again, not many null values in it, let's see how many there are.

[7]:
```
train_df[train_df['outs'].isnull()]
```

[7]:
```
       pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
6689     a6118212        Right  5dce2d1c       Right   99faafae  c16da957
14410    5b740fab         Left  e2e2a336       Right   d0e0eb76  c9752165
19161    161160dd        Right  f338e9d3        Left   1a39a252  c229ef9e
23843    f6d227a5        Right  e9553a98       Right   d0e0eb76  26a1bb6b
27156    264562c6        Right  29d12af7        Left   a5ce1bf6  9806dfbc
…             …          …         …         …          …         …
559476   b3d5c0a9        Right  cf690f2f        Left   0c59f5af  8f1ef267
569804   91700130        Right  de9d396f        Left   99faafae  667d5752
569836   91700130        Right  de9d396f        Left   99faafae  667d5752
570351   d5ef78cb        Right  781ec6be       Right   aa998b21  a9ad7586
574039   b3d5c0a9        Right  cf690f2f        Left   0c59f5af  8f1ef267

       catcher_id  inning  top_bottom  outs  …  zone_speed  \
6689     fd37f21c       7           1   NaN  …   75.911797
14410    5a42193e       5           2   NaN  …   72.887398
19161    41ac8158       4           2   NaN  …   84.766296
```

```
23843    0ffec018     6          1    NaN  …   74.321999
27156    a3a2988b    11          2    NaN  …   86.122299
…           …      …        …    …   …          …
559476   fa18ff59     5          2    NaN  …   80.375298
569804   41ac8158     7          1    NaN  …   84.243103
569836   41ac8158     7          1    NaN  …   82.846298
570351   a3a2988b     6          1    NaN  …   85.266098
574039   fa18ff59     5          2    NaN  …   67.233902

        vert_approach_angle  horz_approach_angle  zone_time      x55  y55  \
6689               -10.27540            -1.215380   0.457509 -1.44079   55
14410               -7.79636             0.764407   0.476133  3.30888   55
19161               -5.21754             0.053372   0.417402 -1.30256   55
23843               -9.18817            -4.715870   0.470084 -2.66638   55
27156               -7.29354            -4.722650   0.411571 -2.93787   55
…                      …                  …           …        …    …
559476              -7.07432            -2.223690   0.430519 -1.15777   55
569804              -5.38618             0.594491   0.419565 -1.64532   55
569836              -6.36660            -1.029200   0.425654 -1.62191   55
570351              -4.60246            -1.696110   0.401914 -1.92533   55
574039             -11.05840            -2.251130   0.515505 -1.19462   55

            z55 pitch_type      pitch_call  pitch_id
6689    6.19782         SL  StrikeSwinging  8f7e287a
14410   5.70064         CH      BallCalled  fdaab946
19161   5.76697         FA      BallCalled  479ee407
23843   6.70092         CH  StrikeSwinging  b01234ef
27156   6.37265         SL  StrikeSwinging  f7771432
…          …        …             …        …
559476  6.08150         SL        FoulBall  d29aa13d
569804  6.31180         FA      BallCalled  d711100f
569836  6.24978         FA    StrikeCalled  5f05e674
570351  5.80506         FA  StrikeSwinging  24ac5f78
574039  5.88421         CU        FoulBall  3a28f2f5

[152 rows x 36 columns]
```

Only 152, that's good. I'll drop these as well since there's no real way to know how many outs there were on a given pitch.

[8]:
```
train_df = train_df.drop(train_df[train_df['outs'].isnull()].index)
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 581941 entries, 0 to 582204
Data columns (total 36 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
```

```
 0   pitcher_id           581941 non-null  object
 1   pitcher_side         581941 non-null  object
 2   batter_id            581941 non-null  object
 3   batter_side          581941 non-null  object
 4   stadium_id           581941 non-null  object
 5   umpire_id            581941 non-null  object
 6   catcher_id           581941 non-null  object
 7   inning               581941 non-null  int64
 8   top_bottom           581941 non-null  int64
 9   outs                 581941 non-null  float64
 10  balls                581941 non-null  int64
 11  strikes              581941 non-null  int64
 12  release_speed        581941 non-null  float64
 13  vert_release_angle   581941 non-null  float64
 14  horz_release_angle   581941 non-null  float64
 15  spin_rate            573044 non-null  float64
 16  spin_axis            581941 non-null  float64
 17  tilt                 580802 non-null  object
 18  rel_height           581941 non-null  float64
 19  rel_side             581941 non-null  float64
 20  extension            581941 non-null  float64
 21  vert_break           581941 non-null  float64
 22  induced_vert_break   581941 non-null  float64
 23  horz_break           581941 non-null  float64
 24  plate_height         581941 non-null  float64
 25  plate_side           581941 non-null  float64
 26  zone_speed           581941 non-null  float64
 27  vert_approach_angle  581941 non-null  float64
 28  horz_approach_angle  581941 non-null  float64
 29  zone_time            581941 non-null  float64
 30  x55                  581941 non-null  float64
 31  y55                  581941 non-null  int64
 32  z55                  581941 non-null  float64
 33  pitch_type           581522 non-null  object
 34  pitch_call           581941 non-null  object
 35  pitch_id             581941 non-null  object
dtypes: float64(20), int64(5), object(11)
memory usage: 164.3+ MB
```

Now let's take a look at spin rate. It certainly looks like there are a lot of data points missing in this column, so dropping all the values may not be the best idea. Let's see how many there are.

[9]:
```python
train_df[train_df['spin_rate'].isnull()]
```

[9]:
```
       pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
43       4c807a49         Left  210e8d5b       Right   402559d3  4ff102e5
148      cb113772        Right  96339e13        Left   aa998b21  c683b9a6
161      af6d3149        Right  6c43d395       Right   03722f5d  d057fd71
```

237	eccb6087	Right	08b0b39d	Left	a5ce1bf6	9806dfbc
304	e332e67d	Right	34a8f234	Left	402559d3	fbbea103
…	…	…	…	…	…	…
581856	00f5fb90	Right	b4efd4bf	Right	f682daed	cac8185e
581859	fa9b0925	Right	c1ec06e6	Right	5025d8df	7675ce83
581927	09da5d7a	Left	073c2b16	Right	b20853fa	4db7bcbc
581992	be5181f0	Right	566220c7	Right	03722f5d	a86853a2
582189	a2f05755	Right	e7a70ed1	Left	854c6c72	16750c18

	catcher_id	inning	top_bottom	outs	…	zone_speed	\
43	fbc0970f	6	2	2.0	…	77.869301	
148	a3a2988b	7	1	0.0	…	73.582497	
161	e9aa50df	9	1	1.0	…	76.742798	
237	b1499101	9	1	1.0	…	79.280602	
304	e4fac104	5	2	2.0	…	80.649803	
…	…	…	…	…	…	…	
581856	e4fac104	4	2	1.0	…	77.819801	
581859	054f7d9f	6	1	0.0	…	77.702797	
581927	9db4e46f	9	1	0.0	…	76.172997	
581992	5b8927f6	2	2	2.0	…	77.229103	
582189	daa1322d	4	2	2.0	…	77.375603	

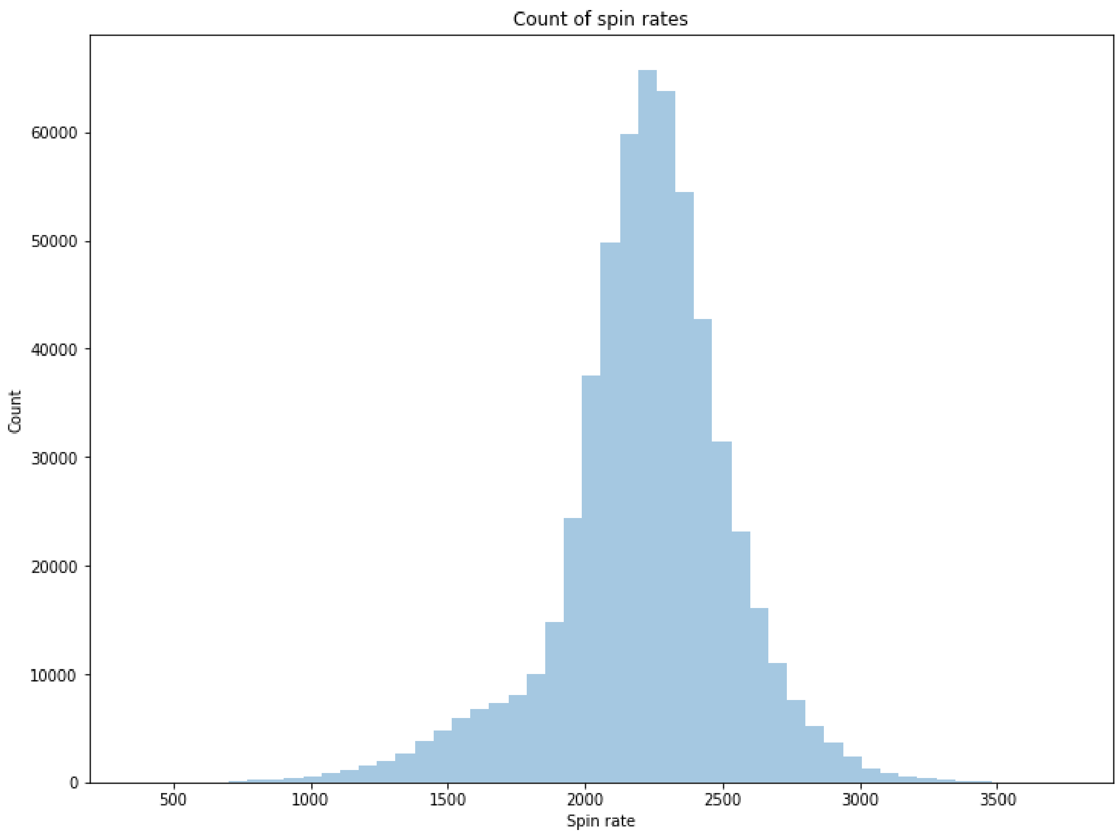
	vert_approach_angle	horz_approach_angle	zone_time	x55	y55	\
43	-6.96547	2.47110	0.444589	2.32704	55	
148	-9.61581	-2.45899	0.474079	-1.42864	55	
161	-6.86927	-3.62681	0.461172	-3.17556	55	
237	-6.75395	-3.49224	0.438489	-1.96139	55	
304	-8.90885	-2.33627	0.433380	-1.18638	55	
…	…	…	…	…	…	
581856	-8.53157	-2.76430	0.446821	-2.01818	55	
581859	-10.84580	-3.94847	0.452332	-1.45562	55	
581927	-8.79066	3.64529	0.459917	2.00739	55	
581992	-7.42066	-3.82928	0.458138	-1.49327	55	
582189	-8.75378	-2.46304	0.450004	-1.84355	55	

	z55	pitch_type	pitch_call	pitch_id
43	5.75515	SL	InPlay	1769b4d5
148	5.84820	SL	BallCalled	ef2a6b0e
161	4.88357	SL	StrikeCalled	f31fc865
237	5.75645	SL	StrikeCalled	3588e119
304	6.42062	SL	InPlay	6fa4b697
…	…	…	…	…
581856	5.79492	SL	InPlay	fc151903
581859	6.02164	CU	BallCalled	941e91ca
581927	6.51264	SL	StrikeSwinging	3f7dd184
581992	5.40118	SL	StrikeSwinging	40cc471a
582189	5.49470	SL	InPlay	059d09ff

```
[8897 rows x 36 columns]
```

Almost 9000 rows. Dropping that many data points could have an adverse effect on the data set and any potential modeling we do. Let's take a look at a countplot and see the distribution of the spin rate column.

[10]:
```
plt.figure(figsize=(12,9))
sns.distplot(train_df[train_df['spin_rate'].notnull()]['spin_rate'], hist=True,␣
↪kde=False)
plt.xlabel('Spin rate')
plt.ylabel('Count')
plt.title('Count of spin rates')
plt.show()
```



It looks like a normal shaped curve, which is expected from this type of data. Filling the missing values with the average would be the best way to go. That keeps the normal distribution of this variable intact and shouldn't have any adverse effect on any future modeling.

[11]:
```
train_df['spin_rate'].describe()
```

[11]:
```
count    573044.000000
mean       2220.693335
std         311.989506
min         362.382996
25%        2072.879883
50%        2238.449951
75%        2400.790039
max        3752.239990
Name: spin_rate, dtype: float64
```

[12]:
```
train_df['spin_rate'].fillna(train_df['spin_rate'].mean(), inplace=True)
```

[13]:
```
train_df['spin_rate'].describe()
```

[13]:
```
count    581941.000000
mean       2220.693335
std         309.595392
min         362.382996
25%        2076.080078
50%        2233.979980
75%        2397.649902
max        3752.239990
Name: spin_rate, dtype: float64
```

Tilt is the next column to have missing values, let's take a look at how many.

[14]:
```
train_df[train_df['tilt'].isnull()]
```

[14]:
```
       pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
378      cd483725        Right  192899a6       Right   c9712626  1869cf54
770      22b76a09         Left  d11080ae        Left   5025d8df  598ea1b2
1339     cd483725        Right  deb2ab32       Right   0b15e1ca  eb059a22
2117     28e273c4         Left  4ac005f3       Right   d0d69f32  46051258
3399     193d153f         Left  ed874f19       Right   c9712626  eb059a22
…             …            …         …           …          …         …
580815   98eaf8b2        Right  f57085ec       Right   99faafae  373947e5
581637   44ec1bf5            R  b3dac04c           L   cfe02944  aea4dd5a
581716   60a6f8df        Right  cddcbd8f        Left   0faa3b2d  4581c636
581733   f45c0602        Right  ad84b429        Left   fe6b0f40  bb04ea23
582161   1fb18290         Left  e14059d7       Right   6a69d99b  ff7406e8

       catcher_id  inning  top_bottom  outs  …  zone_speed  \
378      00ae6fb5       3           1   1.0  …   73.954399
770      a421b54b       3           2   0.0  …   67.967903
1339     00ae6fb5       6           2   1.0  …   75.172897
2117     4fedda83       3           1   1.0  …   75.375900
3399     4f9cd7f9       4           1   2.0  …   75.718102
```

```
…          …    …         …    …  …         …
580815   ccd72da8      9          2   2.0  …   79.889999
581637   a126f66f      6          2   1.0  …   88.644798
581716   a421b54b      7          1   0.0  …   79.617599
581733   dc18f830      1          1   2.0  …   82.575500
582161   bbbfd290      8          2   2.0  …   71.934700

        vert_approach_angle  horz_approach_angle  zone_time       x55  y55  \
378                -9.60881            -1.961030   0.477100 -1.181070   55
770               -11.73220             1.604870   0.512340  1.140470   55
1339               -9.79158            -1.867760   0.471512 -0.749778   55
2117               -8.49879             2.576980   0.459569  2.623400   55
3399               -9.39808             2.573850   0.462214  2.150780   55
…                       …                    …         …         …    …
580815             -8.93767            -1.194220   0.435881 -1.759100   55
581637             -5.31458            -2.812660   0.390148 -2.398420   55
581716             -6.18068            -2.975590   0.435664 -2.043060   55
581733             -5.96087            -0.656144   0.416289 -2.912520   55
582161            -13.77550             1.224220   0.486536  1.447500   55

            z55 pitch_type      pitch_call  pitch_id
378     6.13692         CU          InPlay  d30f5214
770     6.32628         CU      BallCalled  a7209bc8
1339    5.86303         CU  StrikeSwinging  6b05ddc4
2117    6.40245         CU          InPlay  3face29a
3399    6.07629         CU  StrikeSwinging  e1c3703d
…           …        …             …         …
580815  6.48687         CU  StrikeSwinging  672aa57b
581637  6.25742         FA      BallCalled  9e8fb97a
581716  5.01476         SL    StrikeCalled  1d36d6f5
581733  5.64645         CH        FoulBall  2e8f759b
582161  6.16835         CU  StrikeSwinging  7db0fd25

[1139 rows x 36 columns]
```

1139 rows with null values. That's not much in the grand scheme of the data set, only about 0.2% of the entire data set. Dropping these would not hurt in the long run.

[15]:
```python
train_df = train_df.drop(train_df[train_df['tilt'].isnull()].index)
```

[16]:
```python
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 580802 entries, 0 to 582204
Data columns (total 36 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   pitcher_id           580802 non-null  object
```

```
 1   pitcher_side         580802 non-null  object
 2   batter_id            580802 non-null  object
 3   batter_side          580802 non-null  object
 4   stadium_id           580802 non-null  object
 5   umpire_id            580802 non-null  object
 6   catcher_id           580802 non-null  object
 7   inning               580802 non-null  int64
 8   top_bottom           580802 non-null  int64
 9   outs                 580802 non-null  float64
 10  balls                580802 non-null  int64
 11  strikes              580802 non-null  int64
 12  release_speed        580802 non-null  float64
 13  vert_release_angle   580802 non-null  float64
 14  horz_release_angle   580802 non-null  float64
 15  spin_rate            580802 non-null  float64
 16  spin_axis            580802 non-null  float64
 17  tilt                 580802 non-null  object
 18  rel_height           580802 non-null  float64
 19  rel_side             580802 non-null  float64
 20  extension            580802 non-null  float64
 21  vert_break           580802 non-null  float64
 22  induced_vert_break   580802 non-null  float64
 23  horz_break           580802 non-null  float64
 24  plate_height         580802 non-null  float64
 25  plate_side           580802 non-null  float64
 26  zone_speed           580802 non-null  float64
 27  vert_approach_angle  580802 non-null  float64
 28  horz_approach_angle  580802 non-null  float64
 29  zone_time            580802 non-null  float64
 30  x55                  580802 non-null  float64
 31  y55                  580802 non-null  int64
 32  z55                  580802 non-null  float64
 33  pitch_type           580383 non-null  object
 34  pitch_call           580802 non-null  object
 35  pitch_id             580802 non-null  object
dtypes: float64(20), int64(5), object(11)
memory usage: 164.0+ MB
```

Finally, pitch_type is the last column to have null values in it. Let's take a look.

[17]:
```
train_df[train_df['pitch_type'].isnull()]
```

[17]:
```
        pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
634       bff0f759         Left  192899a6       Right   d0d69f32  9c6cbb5e
3126      900e6090        Right  699983d6        Left   78aaa563  9c6cbb5e
3263      7bdd4794        Right  44924919       Right   83508f28  9c6cbb5e
3485      7bdd4794        Right  0b8c61b3        Left   83508f28  9c6cbb5e
4655      57613174        Right  00bde845       Right   78aaa563  9c6cbb5e
```

```
…          …          …       …          …       …       …
576312   b48cf592        Right  a3b17b9b       Left   cfe02944  9c6cbb5e
578590   b4eadd6d        Right  0ae0de45      Right   0a0cfe0d  9c6cbb5e
579026   57613174        Right  6b115fe9       Left   fe6b0f40  9c6cbb5e
579999   d629b647        Right  699983d6       Left   78aaa563  9c6cbb5e
580965   57613174        Right  fd347bb1       Left   78aaa563  9c6cbb5e

       catcher_id  inning  top_bottom  outs  …  zone_speed  \
634      9c6cbb5e       5           2   0.0  …   67.892998
3126     9c6cbb5e       8           2   1.0  …   81.340797
3263     9c6cbb5e       9           1   0.0  …   86.714600
3485     9c6cbb5e       8           1   1.0  …   85.167503
4655     9c6cbb5e      12           1   1.0  …   85.837502
…          …     …          …   …  …          …
576312   9c6cbb5e       8           2   1.0  …   74.434799
578590   9c6cbb5e       8           1   1.0  …   87.362602
579026   9c6cbb5e       8           2   0.0  …   77.284798
579999   9c6cbb5e      14           2   2.0  …   83.999802
580965   9c6cbb5e      14           1   0.0  …   86.906998

        vert_approach_angle  horz_approach_angle  zone_time      x55  y55  \
634                -7.08524             0.721203   0.516315  2.17234   55
3126               -5.03602            -2.062410   0.432663 -2.91834   55
3263               -3.29493            -1.750810   0.402108 -1.73126   55
3485               -4.46310            -1.934850   0.410087 -1.83128   55
4655               -4.32100            -1.389630   0.402448 -1.74871   55
…                         …                    …          …        …  …
576312             -1.24151            -1.315220   0.465865 -1.59554   55
578590             -4.12619            -1.554950   0.396194 -1.33279   55
579026             -7.36536            -1.514690   0.460128 -1.91246   55
579999             -5.26501             0.381528   0.411825 -1.65040   55
580965             -3.75225            -0.884140   0.397061 -1.94958   55

            z55 pitch_type    pitch_call  pitch_id
634     6.50402        NaN  StrikeCalled  34257ee5
3126    5.70209        NaN        InPlay  972b06d9
3263    5.63128        NaN    BallCalled  44187be2
3485    5.61829        NaN  StrikeCalled  bf482788
4655    5.99225        NaN        InPlay  f41ccb06
…           …          …             …         …
576312  1.38558        NaN      FoulBall  17b5318a
578590  6.22610        NaN      FoulBall  d20f5e52
579026  5.90440        NaN      FoulBall  528ecba1
579999  6.96849        NaN      FoulBall  e0d8dea9
580965  5.92467        NaN        InPlay  fba61f04

[419 rows x 36 columns]
```

Only 419 rows, these can be dropped.

[18]:
```
train_df = train_df.drop(train_df[train_df['pitch_type'].isnull()].index)
train_df = train_df.reset_index().drop('index', axis=1)
```

[19]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 580383 entries, 0 to 580382
Data columns (total 36 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   pitcher_id           580383 non-null  object
 1   pitcher_side         580383 non-null  object
 2   batter_id            580383 non-null  object
 3   batter_side          580383 non-null  object
 4   stadium_id           580383 non-null  object
 5   umpire_id            580383 non-null  object
 6   catcher_id           580383 non-null  object
 7   inning               580383 non-null  int64
 8   top_bottom           580383 non-null  int64
 9   outs                 580383 non-null  float64
 10  balls                580383 non-null  int64
 11  strikes              580383 non-null  int64
 12  release_speed        580383 non-null  float64
 13  vert_release_angle   580383 non-null  float64
 14  horz_release_angle   580383 non-null  float64
 15  spin_rate            580383 non-null  float64
 16  spin_axis            580383 non-null  float64
 17  tilt                 580383 non-null  object
 18  rel_height           580383 non-null  float64
 19  rel_side             580383 non-null  float64
 20  extension            580383 non-null  float64
 21  vert_break           580383 non-null  float64
 22  induced_vert_break   580383 non-null  float64
 23  horz_break           580383 non-null  float64
 24  plate_height         580383 non-null  float64
 25  plate_side           580383 non-null  float64
 26  zone_speed           580383 non-null  float64
 27  vert_approach_angle  580383 non-null  float64
 28  horz_approach_angle  580383 non-null  float64
 29  zone_time            580383 non-null  float64
 30  x55                  580383 non-null  float64
 31  y55                  580383 non-null  int64
 32  z55                  580383 non-null  float64
 33  pitch_type           580383 non-null  object
 34  pitch_call           580383 non-null  object
 35  pitch_id             580383 non-null  object
```

```
dtypes: float64(20), int64(5), object(11)
memory usage: 159.4+ MB
```

[20]:
```
train_df.head(10)
```

[20]:
```
  pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
0   d7e3acce        Right  32678d8d       Right   a4833794  f88d09f4
1   44ec1bf5        Right  81d51733        Left   f60d6ea5  b67d862c
2   44d87ee6         Left  8eefccb7       Right   a9b8b538  13993d26
3   ff6adae0        Right  8f8ab5af       Right   e569ec39  0d8ba4bb
4   c70c96e5        Right  10874746       Right   a5ce1bf6  94a4c552
5   98f8936a        Right  a58e31f3       Right   9b5daeaf  0dad94e8
6   28e273c4         Left  9a2db1f2       Right   d0d69f32  caf1f50b
7   4f3062b6         Left  7e2bb9a9       Right   c9712626  33bb973b
8   afae9816         Left  ffe7832e        Left   d0d69f32  f88d09f4
9   61ab8c67        Right  daa1322d       Right   f682daed  c4c41d26

  catcher_id  inning  top_bottom  outs  …  zone_speed  vert_approach_angle  \
0   83cdf9ff       3           1   0.0  …   86.024200             -4.37258
1   a126f66f       6           2   0.0  …   89.458199             -4.90467
2   9db4e46f       5           2   2.0  …   75.593597             -6.00728
3   bbbfd290       5           1   2.0  …   76.396400             -9.50640
4   75087ec8       8           1   2.0  …   83.215302             -4.53233
5   68d1111a       7           2   0.0  …   80.265404             -8.24794
6   4fedda83       3           1   1.0  …   87.948799             -4.76645
7   20bf9444       6           1   1.0  …   76.352798            -10.25710
8   4fedda83       2           1   1.0  …   78.281097             -4.85101
9   41ac8158       4           1   1.0  …   86.078400             -6.09955

   horz_approach_angle  zone_time       x55  y55      z55 pitch_type  \
0             1.429580   0.404622 -0.059343   55  6.03322         FA
1            -2.148410   0.385719 -2.148680   55  6.23380         FA
2            -0.122044   0.463953  1.300450   55  6.14750         CH
3            -2.581980   0.458471 -1.659590   55  6.60043         CU
4            -0.268188   0.415965 -1.526170   55  4.77332         FA
5             0.780148   0.438111 -2.075230   55  5.79080         CH
6             0.696210   0.390590  2.569990   55  6.09316         FA
7             4.681720   0.464021  2.497290   55  6.22659         SL
8             1.945590   0.440157  2.559950   55  5.91159         FA
9             0.425454   0.411268 -0.876224   55  6.53540         FA

    pitch_call  pitch_id
0       InPlay  42fce2f6
1       InPlay  3e9cda86
2   BallCalled  f129a6cd
3       InPlay  03e9bc05
4  StrikeCalled  48feb675
```

```
5  StrikeSwinging  419540c7
6  StrikeSwinging  cf85249d
7      BallCalled  c9423da3
8        FoulBall  51ad39b4
9        FoulBall  b89e4ec3

[10 rows x 36 columns]
```

No more null values in the data set now, now we can move on to creating the target variable for modeling.

2 Data Wrangling

In the test set, the target variable is called "is_strike", and we don't have a column like that here in the training set. However, we do have a "pitch_call" column, which we can use to create the "is_strike" column. Along with called strikes and swinging strikes, any ball batted in to play or any foul balls are also counted as strikes. Using this, we can build the "is_strike" column using a simple for loop.

[21]:
```
train_df['pitch_call'].unique()
```

[21]:
```
array(['InPlay', 'BallCalled', 'StrikeCalled', 'StrikeSwinging',
       'FoulBall', 'HitByPitch', 'BallIntentional'], dtype=object)
```

[22]:
```
is_strike_list = ['InPlay', 'StrikeCalled', 'StrikeSwinging', 'FoulBall']
train_df['is_strike'] = train_df['pitch_call'].apply(lambda x: 1 if x in␣
↪is_strike_list else 0)
```

We can easily assign that list to a new column in the data set.

[23]:
```
train_df.head(10)
```

[23]:
```
  pitcher_id pitcher_side batter_id batter_side stadium_id umpire_id  \
0   d7e3acce        Right  32678d8d       Right   a4833794  f88d09f4
1   44ec1bf5        Right  81d51733        Left   f60d6ea5  b67d862c
2   44d87ee6         Left  8eefccb7       Right   a9b8b538  13993d26
3   ff6adae0        Right  8f8ab5af       Right   e569ec39  0d8ba4bb
4   c70c96e5        Right  10874746       Right   a5ce1bf6  94a4c552
5   98f8936a        Right  a58e31f3       Right   9b5daeaf  0dad94e8
6   28e273c4         Left  9a2db1f2       Right   d0d69f32  caf1f50b
7   4f3062b6         Left  7e2bb9a9       Right   c9712626  33bb973b
8   afae9816         Left  ffe7832e        Left   d0d69f32  f88d09f4
9   61ab8c67        Right  daa1322d       Right   f682daed  c4c41d26

  catcher_id  inning  top_bottom  outs  …  vert_approach_angle  \
0   83cdf9ff       3           1   0.0  …             -4.37258
1   a126f66f       6           2   0.0  …             -4.90467
```

```
2   9db4e46f        5            2   2.0  ...            -6.00728
3   bbbfd290        5            1   2.0  ...            -9.50640
4   75087ec8        8            1   2.0  ...            -4.53233
5   68d1111a        7            2   0.0  ...            -8.24794
6   4fedda83        3            1   1.0  ...            -4.76645
7   20bf9444        6            1   1.0  ...           -10.25710
8   4fedda83        2            1   1.0  ...            -4.85101
9   41ac8158        4            1   1.0  ...            -6.09955

   horz_approach_angle  zone_time       x55  y55       z55  pitch_type  \
0             1.429580   0.404622 -0.059343   55   6.03322          FA
1            -2.148410   0.385719 -2.148680   55   6.23380          FA
2            -0.122044   0.463953  1.300450   55   6.14750          CH
3            -2.581980   0.458471 -1.659590   55   6.60043          CU
4            -0.268188   0.415965 -1.526170   55   4.77332          FA
5             0.780148   0.438111 -2.075230   55   5.79080          CH
6             0.696210   0.390590  2.569990   55   6.09316          FA
7             4.681720   0.464021  2.497290   55   6.22659          SL
8             1.945590   0.440157  2.559950   55   5.91159          FA
9             0.425454   0.411268 -0.876224   55   6.53540          FA

       pitch_call  pitch_id  is_strike
0          InPlay  42fce2f6          1
1          InPlay  3e9cda86          1
2      BallCalled  f129a6cd          0
3          InPlay  03e9bc05          1
4    StrikeCalled  48feb675          1
5  StrikeSwinging  419540c7          1
6  StrikeSwinging  cf85249d          1
7      BallCalled  c9423da3          0
8        FoulBall  51ad39b4          1
9        FoulBall  b89e4ec3          1

[10 rows x 37 columns]
```

Now we have our data set with the target variable, let's take a look at the "is_strike" column and its value counts.

[24]:
```
train_df['is_strike'].value_counts()
```

[24]:
```
1    369807
0    210576
Name: is_strike, dtype: int64
```

Interesting. We have an imbalanced classification problem here, with the majority class being almost twice as large as the minority class. That has implications for modeling in the future, namely being careful about what classification model is used for this problem. We also may need to use some resampling methods if the model is choosing the majority class by an overwhelming margin.

Before we get into modeling however, there was something I noticed with the "tilt" column. It has two different types of string data packed into the column. We'll need to fix that column to get it all into one data format.

[25]:
```
train_df['tilt'].unique()
```

[25]:
```
array(['1:00', '12:15', '11:15', '7:45', '2:15', '2:45', '10:30', '4:45',
       '11:00', '1:30', '1:15', '12:45', '5:15', '10:45', '6:30', '8:00',
       '4:00', '2:00', '7:30', '3:30', '12:00', '1:45', '9:00', '10:00',
       '11:30', '32400 secs', '12:30', '9:15', '11:45', '9:45', '10:15',
       '3:00', '42300 secs', '5:00', '7:15', '7:00', '6:45', '9:30',
       '8:45', '3:45', '43200 secs', '6:15', '2:30', '4:15', '35100 secs',
       '5:45', '5:30', '8:15', '8:30', '29700 secs', '44100 secs', '3:15',
       '4:30', '14400 secs', '6300 secs', '45000 secs', '38700 secs',
       '34200 secs', '36000 secs', '45900 secs', '36900 secs',
       '3600 secs', '18000 secs', '7200 secs', '40500 secs', '5400 secs',
       '8100 secs', '15300 secs', '27900 secs', '4500 secs', '23400 secs',
       '25200 secs', '41400 secs', '30600 secs', '6:00', '9900 secs',
       '33300 secs', '37800 secs', '13500 secs', '27000 secs',
       '39600 secs', '12600 secs', '17100 secs', '16200 secs',
       '11700 secs', '9000 secs', '18900 secs', '26100 secs',
       '22500 secs', '20700 secs', '24300 secs', '31500 secs',
       '21600 secs', '10800 secs', '19800 secs', '28800 secs'],
      dtype=object)
```

I'm choosing to turn all of the "1:00", "12:15" format into a seconds-from-midnight integer, that will be the easiest way to get all of the column into one data format and data type.

[26]:
```
train_df['tilt'] = train_df['tilt'].map(lambda x: sum(a*int(t) for a, t in␣
 ↪zip([3600, 60], x.split(':'))) \
                                             if ':' in x else int(x[:-5]))
```

[27]:
```
train_df.to_csv('../Data/model_data.csv')
```

3 Predictive Modeling

Now that our data set clean and how we want it, we can get into some predictive modeling. Seeing as this is a binary classification problem, we'll need to use a classification algorithm. I'm choosing to use gradient boosting here because I've used it in the past and have gotten good results with it in a timely manner. Logistic regression would be faster, but would give us a less accurate model than a gradient boosting model.

I'm going to do the hyperparameter tuning in its own dedicated notebook, then load the trained model into this notebook after it's been fitted with all the correct hyperparameters.

[28]:
```
X = train_df.drop(['pitcher_id', 'batter_id', 'stadium_id', 'umpire_id',␣
↪'catcher_id', 'pitch_call', 'is_strike', 'pitch_id'], axis=1)
y = train_df['is_strike']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
↪random_state=34)
```

Getting dummy variabled for some of the categorical variables would be good for modeling. It will allow us to see how much importance was placed on these features by the model.

[29]:
```
X_train = pd.get_dummies(X_train, prefix=['pitcher', 'batter', 'is'],␣
↪columns=['pitcher_side', 'batter_side', 'pitch_type'])
X_test = pd.get_dummies(X_test, prefix=['pitcher', 'batter', 'is'],␣
↪columns=['pitcher_side', 'batter_side', 'pitch_type'])
```

[30]:
```
X_train.head()
```

[30]:
```
        inning  top_bottom  outs  balls  strikes  release_speed  \
362341       8           1   1.0      3        2      86.056702
392255       3           2   0.0      0        1      93.787697
520345       3           1   0.0      0        0      90.838699
120374       3           1   1.0      1        1      89.893600
194486       7           1   1.0      0        0      94.105202

        vert_release_angle  horz_release_angle    spin_rate   spin_axis  …  \
362341           -0.754319            -2.89459  2425.870117  205.197006  …
392255           -1.202140            -3.24475  2496.909912  210.024994  …
520345           -1.730600             1.75061  2230.830078  158.481995  …
120374           -2.494800            -3.26674  2175.179932 -150.666000  …
194486           -2.047340            -2.19327  2071.290039  227.212997  …

        pitcher_Left  pitcher_Right  batter_Left  batter_Right  is_CH  is_CU  \
362341             0              1            0             1      0      0
392255             0              1            1             0      0      0
520345             1              0            0             1      0      0
120374             0              1            0             1      0      0
194486             0              1            1             0      0      0

        is_FA  is_KN  is_SL  is_XX
362341      0      0      1      0
392255      1      0      0      0
520345      1      0      0      0
120374      0      0      1      0
194486      1      0      0      0

[5 rows x 36 columns]
```

Now that we have our modeling data set, let's get into modeling the data, first we will look at a non-tuned XGBoost model and see how it performs on the data, then compare that to the tuned model from the "Mariners Machine Learning Model" notebook.

[ ]:
```
base_xgb = XGBClassifier()
base_xgb.fit(X_train, y_train)
xgb_base_pred = base_xgb.predict(X_test)
```

[ ]:
```
print(f'Base XGB Classifier Test Accuracy: {round(accuracy_score(y_test,␣
↪xgb_base_pred) * 100, 2)}')
print('Base XGB Classifier Classification Report')
print(classification_report(y_test, xgb_base_pred))
print('\n')
print(f'Base XGB Classifier MCC Score: {matthews_corrcoef(y_test,␣
↪xgb_base_pred)}')

xgb_probs = base_xgb.predict_proba(X_test)
xgb_probs = xgb_probs[:, 1]

xgb_precision, xgb_recall, _ = precision_recall_curve(y_test, xgb_probs)

no_skill = len(y_test[y_test == 1]) / len(y_test)
plt.figure(figsize=(12,9))
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot(xgb_recall, xgb_precision, marker='.', label='Gradient Boosting')
plt.title('Precision Recall Curve for a Base XGB Classifier')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()

print(f'Base XGB Classifier AUC Score: {auc(xgb_recall, xgb_precision)}')
```

By itself, it's a good model. 86% accuracy and a solid precision/recall on both classes is a good start. As well as a .706 MCC score and very high AUC score. Looks like the base XGBoost is a good starting out point for the hyperparameter tuning we did in the other notebook. Let's load that in and take a look at the same metrics as above.

[ ]:
```
tuned_xgb = load('xgboost_model.pkl')
xgb_tuned_pred = tuned_xgb.predict(X_test)
```

[ ]:
```
print(f'Tuned XGB Classifier Test Accuracy: {round(accuracy_score(y_test,␣
↪xgb_tuned_pred) * 100, 2)}')
print('Tuned XGB Classifier Classification Report')
print(classification_report(y_test, xgb_tuned_pred))
print('\n')
print(f'Tuned XGB Classifier MCC Score: {matthews_corrcoef(y_test,␣
↪xgb_tuned_pred)}')
```

```
xgb_probs = tuned_xgb.predict_proba(X_test)
xgb_probs = xgb_probs[:, 1]

xgb_precision, xgb_recall, _ = precision_recall_curve(y_test, xgb_probs)

no_skill = len(y_test[y_test == 1]) / len(y_test)
plt.figure(figsize=(12,9))
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot(xgb_recall, xgb_precision, marker='.', label='Gradient Boosting')
plt.title('Precision Recall Curve for a Tuned XGB Classifier')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()

print(f'Tuned XGB Classifier AUC Score: {auc(xgb_recall, xgb_precision)}')
```

It did better, even if it was only slightly. Precision and recall scores on both classes improved, accuracy went up, and MCC and AUC scores went up as well. The hyper-parameter tuning we did in the other notebook worked well.

3.1 Making predictions on test set

We have a good model trained, now we can make predictions on the testing set. First we need to load it in and clean it the way we cleaned the training set.

[ ]:
```
test_df = pd.read_csv('../Data/2020-test.csv')
```

[ ]:
```
def clean_and_wrangle(df):

    df = df.drop(df[df['release_speed'].isnull()].index)
    df = df.drop(df[df['outs'].isnull()].index)
    df['spin_rate'].fillna(df['spin_rate'].mean(), inplace=True)
    df = df.drop(df[df['tilt'].isnull()].index)
    df = df.drop(df[df['pitch_type'].isnull()].index)
    df = df.reset_index().drop('index', axis=1)

    df['tilt'] = df['tilt'].map(lambda x: sum(a*int(t) for a, t in zip([3600,␣
 ↪60], x.split(':'))) \
                                          if ':' in x else int(x[:-5]))
    return df
```

[ ]:
```
test_df = clean_and_wrangle(test_df)
```

[ ]:
```
test_df.head(10)
```

[ ]:
```
test_df.info()
```

No null values (except our target variable "is_strike"). We'll use our trained model from up above and make predictions on the entire testing set and insert those predictions into the data set.

[ ]:
```
X = test_df.drop(['pitcher_id', 'batter_id', 'stadium_id', 'umpire_id',␣
↪'catcher_id', 'is_strike', 'pitch_id'], axis=1)

X = pd.get_dummies(X, prefix=['pitcher', 'batter', 'is'],␣
↪columns=['pitcher_side', 'batter_side', 'pitch_type'])
```

[ ]:
```
predictions = tuned_xgb.predict(X)
```

[ ]:
```
test_df['is_strike'] = predictions
test_df.head(10)
```

[ ]:
```
test_df.to_csv('../Data/predicted_test_set.csv')
```

4 Conclusion

Overall we created a good model to predict if a specific pitch was going to be a strike or not. We started off by cleaning the data set and making sure that no null values were in the table. Next we had to create the target variable from a column that already existed in the data set. Finally had to wrangle some data to get it all in the correct format to be suitable to run a machine learning model on. Using gradient boosting, tuned a number of hyper-parameters, and made predictions full training set. After making predictions, we needed to make sure the model was performing well, and took a look at a number of different metrics for model performance. Accuracy, the classification report, AUC score, and Matthew's Correlation Coefficient all agreed that this model we built was well suited for predicting strikes. We also took a look at the most important features of the model, and came away with plate side and plate height being the most important features.

If I had more time and resources to dedicate to this project, I would have tuned some more of the XGBoost hyper-parameters to make this model even more accurate. Having only tuned five parameters, there could be some more room for improvement, but the time it could have taken to do so may have outweighed the gains produce by finding more optimal parameters.