Final Project Code

Importing data set:

```
import os
import pandas as pd
import pyarrow as pa

file_path = (
    'data/statcast_pitch_swing_data_20240402_20240630.arrow'
)

table = pa.ipc.open_file(file_path).read_all()

df = table.to_pandas()

print(df.head())
```

```
  pitch_type   game_date  release_speed  release_pos_x  release_pos_z  \
0         FF  2024-04-02           95.0          -2.01           5.22
1         CH  2024-04-02           88.5          -2.09           4.95
2         SI  2024-04-02           95.0          -2.02           5.12
3         SI  2024-04-02           90.7          -1.26           5.13
4         FF  2024-04-02           95.4          -1.95           5.12

       player_name  batter  pitcher     events    description  ...  \
0  Rocchio, Brayan  677587   622491     single  hit_into_play  ...
1  Rocchio, Brayan  677587   622491                    foul  ...
2  Rocchio, Brayan  677587   622491             called_strike  ...
3   Ohtani, Shohei  660271   657277       walk           ball  ...
4   Hedges, Austin  595978   622491  strikeout       foul_tip  ...

  post_home_score post_bat_score post_fld_score if_fielding_alignment  \
0               0              4              0         Infield shade
1               0              4              0              Standard
2               0              4              0              Standard
```

```
3                5              5              2         Infield shade
4                0              4              0              Standard

   of_fielding_alignment spin_axis delta_home_win_exp delta_run_exp bat_speed  \
0               Standard     239.0             -0.006         0.388       NaN
1               Standard     253.0              0.000        -0.045       NaN
2               Standard     238.0              0.000        -0.042       NaN
3               Standard     234.0              0.009         0.082       NaN
4               Standard     238.0              0.008        -0.215       NaN

  swing_length
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN

[5 rows x 94 columns]
```

Data Cleaning

Renaming 'type' to 'category' to limit confusion:

```
df.rename(columns={'type': 'category'}, inplace=True)
```

Finding variables with high missing percentages and removing them from the dataframe:

```
missing_percentage = df.isnull().mean() * 100
print("Missing Percentage:\n", missing_percentage)
```

```
Missing Percentage:
 pitch_type             0.000000
game_date               0.000000
release_speed           0.073069
release_pos_x           0.072780
release_pos_z           0.072780
                          ...
spin_axis               0.635668
delta_home_win_exp      0.000000
delta_run_exp           0.005776
bat_speed              54.996101
swing_length           55.000144
Length: 94, dtype: float64
```

```
high_missing_columns = missing_percentage[missing_percentage >= 65].index
print(high_missing_columns)
```

```
Index(['spin_dir', 'spin_rate_deprecated', 'break_angle_deprecated',
       'break_length_deprecated', 'hit_location', 'on_3b', 'on_2b', 'on_1b',
       'hc_x', 'hc_y', 'tfs_deprecated', 'tfs_zulu_deprecated', 'umpire',
       'sv_id', 'hit_distance_sc', 'launch_speed', 'launch_angle',
       'estimated_ba_using_speedangle', 'estimated_woba_using_speedangle',
       'woba_value', 'woba_denom', 'babip_value', 'iso_value',
       'launch_speed_angle'],
      dtype='object')
```

```
df = df.drop(high_missing_columns, axis=1)
print(df.columns)
```

```
Index(['pitch_type', 'game_date', 'release_speed', 'release_pos_x',
       'release_pos_z', 'player_name', 'batter', 'pitcher', 'events',
       'description', 'zone', 'des', 'game_type', 'stand', 'p_throws',
       'home_team', 'away_team', 'category', 'bb_type', 'balls', 'strikes',
       'game_year', 'pfx_x', 'pfx_z', 'plate_x', 'plate_z', 'outs_when_up',
       'inning', 'inning_topbot', 'fielder_2', 'vx0', 'vy0', 'vz0', 'ax', 'ay',
       'az', 'sz_top', 'sz_bot', 'effective_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',
       'fielder_9', 'release_pos_y', '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_axis', 'delta_home_win_exp', 'delta_run_exp', 'bat_speed',
       'swing_length'],
      dtype='object')
```

```
df = df.drop(['bat_speed', 'swing_length'], axis=1)
```

Finding and removing redundant variables:

```
print(df.game_date)
```

```
0         2024-04-02
1         2024-04-02
2         2024-04-02
3         2024-04-02
```

```
4         2024-04-02
             ...
346245    2024-06-30
346246    2024-06-30
346247    2024-06-30
346248    2024-06-30
346249    2024-06-30
Name: game_date, Length: 346250, dtype: object
```

```
print(df.game_year)
```

```
0         2024
1         2024
2         2024
3         2024
4         2024
          ...
346245    2024
346246    2024
346247    2024
346248    2024
346249    2024
Name: game_year, Length: 346250, dtype: int32
```

```
are_fielder2_equal = (df['fielder_2'] == df['fielder_2_1']).all()
print(are_fielder2_equal)
```

```
True
```

```
df = df.drop(['game_year', 'fielder_2_1'], axis=1)
print(df)
```

```
       pitch_type   game_date  release_speed  release_pos_x  release_pos_z  \
0              FF  2024-04-02           95.0          -2.01           5.22
1              CH  2024-04-02           88.5          -2.09           4.95
2              SI  2024-04-02           95.0          -2.02           5.12
3              SI  2024-04-02           90.7          -1.26           5.13
4              FF  2024-04-02           95.4          -1.95           5.12
...           ...         ...            ...            ...            ...
346245         SL  2024-06-30           85.9           1.63           5.68
346246         SL  2024-06-30           89.9          -1.12           6.41
346247         CU  2024-06-30           73.7          -2.44           5.80
```

```
346248         FF  2024-06-30           94.1           2.97           5.97
346249         FF  2024-06-30           99.8          -0.40           6.01

             player_name  batter  pitcher     events      description  ...  \
0         Rocchio, Brayan  677587   622491     single    hit_into_play  ...
1         Rocchio, Brayan  677587   622491                       foul  ...
2         Rocchio, Brayan  677587   622491              called_strike  ...
3          Ohtani, Shohei  660271   657277       walk             ball  ...
4          Hedges, Austin  595978   622491  strikeout         foul_tip  ...
...                   ...     ...      ...        ...              ...  ...
346245        Cave, Jake  595909   694363              called_strike  ...
346246     Jiménez, Eloy  650391   641755               blocked_ball  ...
346247         Rice, Ben  700250   670102              called_strike  ...
346248  Merrifield, Whit  593160   677053                       ball  ...
346249  De La Cruz, Elly  682829   664854            swinging_strike  ...

        fld_score post_away_score post_home_score post_bat_score  \
0               0               4               0              4
1               0               4               0              4
2               0               4               0              4
3               2               2               5              5
4               0               4               0              4
...           ...             ...             ...            ...
346245          2               3               2              3
346246          3               3               3              3
346247          1               8               1              8
346248          6               6               5              5
346249          2               0               2              0

       post_fld_score if_fielding_alignment of_fielding_alignment spin_axis  \
0                   0         Infield shade              Standard     239.0
1                   0              Standard              Standard     253.0
2                   0              Standard              Standard     238.0
3                   2         Infield shade              Standard     234.0
4                   0              Standard              Standard     238.0
...               ...                   ...                   ...       ...
346245              2              Standard              Standard     144.0
346246              3              Standard              Standard     200.0
346247              1         Infield shade              Standard      47.0
346248              6              Standard              Standard     139.0
346249              2         Infield shade              Standard     206.0

       delta_home_win_exp  delta_run_exp
0                  -0.006          0.388
1                   0.000         -0.045
```

```
2                    0.000         -0.042
3                    0.009          0.082
4                    0.008         -0.215
...                    ...            ...
346245               0.000         -0.067
346246               0.000          0.057
346247               0.000         -0.033
346248               0.000          0.032
346249               0.000         -0.033

[346250 rows x 66 columns]
```

Finding and removing invalid values for pitch speed:

```
out_of_range = df[(df['release_speed'] < 60) | (df['release_speed'] > 105)]
unique_out_of_range = out_of_range['release_speed'].unique()

print("Unique values outside the valid range:")
print(unique_out_of_range)
```

```
Unique values outside the valid range:
[39.9 50.3 52.3 58.7 55.  57.5 58.5 54.6 56.5 59.4 46.1 53.2 50.6 49.7
 45.5 35.1 37.8 39.  40.7 40.  41.1 43.3 42.7 46.  59.5 53.9 58.8 56.6
 43.2 53.4 56.4 59.7 47.2 58.  59.1 58.1 59.2 57.2 55.9 51.8 57.7 50.4
 56.  54.4 53.1 54.  52.8 54.1 53.3 53.6 51.  58.6 55.3 55.4 57.1 55.8
 54.3 59.9 58.2 53.8 57.4 58.9 57.  57.6 56.1 42.1 56.7 54.2 55.7 56.3
 56.8 59.  54.8 58.3 42.4 51.9 49.5 55.1 54.9 53.5 38.6 50.9 54.7 51.3
 38.7 55.6 38.9 51.7 42.6 53.  52.6 52.9 51.1 46.9 49.8 59.3 44.2 50.5
 48.3 49.9 47.7 53.7 47.5 48.9 50.2 45.9 40.3 41.7 56.2 36.5 35.7 34.3
 35.6 36.6 54.5 55.2 36.8 41.2 37.6 43.7 40.8 39.8 48.7 57.8 48.2 48.
 46.5 49.3 47.1 47.8 49.6 50.1 57.3 44.1 44.4 44.5 43.8 42.2 42.8 43.
 43.9 43.5 41.5 41.9 45.6 45.8 59.8 52.2 40.5 59.6 37.2 42.3 31.9 48.6
 40.1 40.2]
```

```
df.drop(
    df[(df['release_speed'] < 60) | (df['release_speed'] > 105)].index,
    inplace=True
)
print(df)
```

```
       pitch_type   game_date  release_speed  release_pos_x  release_pos_z  \
0              FF  2024-04-02           95.0          -2.01           5.22
1              CH  2024-04-02           88.5          -2.09           4.95
```

```
2              SI  2024-04-02          95.0          -2.02          5.12
3              SI  2024-04-02          90.7          -1.26          5.13
4              FF  2024-04-02          95.4          -1.95          5.12
...           ...         ...           ...            ...           ...
346245         SL  2024-06-30          85.9           1.63          5.68
346246         SL  2024-06-30          89.9          -1.12          6.41
346247         CU  2024-06-30          73.7          -2.44          5.80
346248         FF  2024-06-30          94.1           2.97          5.97
346249         FF  2024-06-30          99.8          -0.40          6.01

            player_name  batter  pitcher     events      description  ...  \
0        Rocchio, Brayan  677587   622491     single    hit_into_play  ...
1        Rocchio, Brayan  677587   622491                       foul  ...
2        Rocchio, Brayan  677587   622491              called_strike  ...
3         Ohtani, Shohei  660271   657277       walk             ball  ...
4         Hedges, Austin  595978   622491  strikeout         foul_tip  ...
...                  ...     ...      ...        ...              ...  ...
346245       Cave, Jake  595909   694363              called_strike  ...
346246    Jiménez, Eloy  650391   641755               blocked_ball  ...
346247        Rice, Ben  700250   670102              called_strike  ...
346248  Merrifield, Whit  593160   677053                       ball  ...
346249  De La Cruz, Elly  682829   664854            swinging_strike  ...

        fld_score post_away_score post_home_score post_bat_score  \
0               0               4               0              4
1               0               4               0              4
2               0               4               0              4
3               2               2               5              5
4               0               4               0              4
...           ...             ...             ...            ...
346245          2               3               2              3
346246          3               3               3              3
346247          1               8               1              8
346248          6               6               5              5
346249          2               0               2              0

       post_fld_score if_fielding_alignment of_fielding_alignment spin_axis  \
0                   0         Infield shade              Standard     239.0
1                   0              Standard              Standard     253.0
2                   0              Standard              Standard     238.0
3                   2         Infield shade              Standard     234.0
4                   0              Standard              Standard     238.0
...               ...                   ...                   ...       ...
346245              2              Standard              Standard     144.0
346246              3              Standard              Standard     200.0
```

```
346247              1         Infield shade              Standard      47.0
346248              6              Standard              Standard     139.0
346249              2         Infield shade              Standard     206.0

       delta_home_win_exp  delta_run_exp
0                  -0.006          0.388
1                   0.000         -0.045
2                   0.000         -0.042
3                   0.009          0.082
4                   0.008         -0.215
...                   ...            ...
346245              0.000         -0.067
346246              0.000          0.057
346247              0.000         -0.033
346248              0.000          0.032
346249              0.000         -0.033

[345954 rows x 66 columns]
```

Ensuring that the possible home and away teams align:

```
unique_home_teams = df['home_team'].unique()
print(f"Unique values in 'home_team':")
print(unique_home_teams)
```

```
Unique values in 'home_team':
['SEA' 'LAD' 'OAK' 'AZ' 'MIA' 'CWS' 'TB' 'CHC' 'MIL' 'HOU' 'SD' 'PHI'
 'BAL' 'WSH' 'MIN' 'KC' 'NYM' 'STL' 'SF' 'CIN' 'ATL' 'NYY' 'TEX' 'PIT'
 'DET' 'LAA' 'COL' 'TOR' 'CLE' 'BOS']
```

```
unique_away_teams = df['away_team'].unique()
print(f"Unique values in 'away_team':")
print(unique_away_teams)
```

```
Unique values in 'away_team':
['CLE' 'SF' 'BOS' 'NYY' 'LAA' 'ATL' 'TEX' 'COL' 'MIN' 'TOR' 'STL' 'CIN'
 'KC' 'PIT' 'CWS' 'DET' 'MIA' 'SD' 'NYM' 'AZ' 'HOU' 'BAL' 'PHI' 'SEA'
 'OAK' 'TB' 'LAD' 'CHC' 'MIL' 'WSH']
```

Removing any remaining rows with missing data:

```
clean_df = df.dropna()
print(clean_df)
```

```
       pitch_type   game_date  release_speed  release_pos_x  release_pos_z  \
0              FF  2024-04-02           95.0          -2.01           5.22
1              CH  2024-04-02           88.5          -2.09           4.95
2              SI  2024-04-02           95.0          -2.02           5.12
3              SI  2024-04-02           90.7          -1.26           5.13
4              FF  2024-04-02           95.4          -1.95           5.12
...           ...         ...            ...            ...            ...
346245         SL  2024-06-30           85.9           1.63           5.68
346246         SL  2024-06-30           89.9          -1.12           6.41
346247         CU  2024-06-30           73.7          -2.44           5.80
346248         FF  2024-06-30           94.1           2.97           5.97
346249         FF  2024-06-30           99.8          -0.40           6.01

            player_name  batter  pitcher     events      description  ...  \
0        Rocchio, Brayan  677587   622491     single    hit_into_play  ...
1        Rocchio, Brayan  677587   622491                     foul  ...
2        Rocchio, Brayan  677587   622491              called_strike  ...
3         Ohtani, Shohei  660271   657277       walk             ball  ...
4         Hedges, Austin  595978   622491  strikeout         foul_tip  ...
...                  ...     ...      ...        ...              ...  ...
346245        Cave, Jake  595909   694363              called_strike  ...
346246     Jiménez, Eloy  650391   641755               blocked_ball  ...
346247         Rice, Ben  700250   670102              called_strike  ...
346248  Merrifield, Whit  593160   677053                     ball  ...
346249  De La Cruz, Elly  682829   664854            swinging_strike  ...

        fld_score post_away_score post_home_score post_bat_score  \
0               0               4               0              4
1               0               4               0              4
2               0               4               0              4
3               2               2               5              5
4               0               4               0              4
...           ...             ...             ...            ...
346245          2               3               2              3
346246          3               3               3              3
346247          1               8               1              8
346248          6               6               5              5
346249          2               0               2              0

       post_fld_score if_fielding_alignment of_fielding_alignment spin_axis  \
0                   0         Infield shade              Standard     239.0
```

```
1                    0              Standard              Standard     253.0
2                    0              Standard              Standard     238.0
3                    2         Infield shade              Standard     234.0
4                    0              Standard              Standard     238.0
...                ...                   ...                   ...       ...
346245               2              Standard              Standard     144.0
346246               3              Standard              Standard     200.0
346247               1         Infield shade              Standard      47.0
346248               6              Standard              Standard     139.0
346249               2         Infield shade              Standard     206.0

       delta_home_win_exp  delta_run_exp
0                  -0.006          0.388
1                   0.000         -0.045
2                   0.000         -0.042
3                   0.009          0.082
4                   0.008         -0.215
...                   ...            ...
346245              0.000         -0.067
346246              0.000          0.057
346247              0.000         -0.033
346248              0.000          0.032
346249              0.000         -0.033

[343602 rows x 66 columns]
```

Data Exploration

Printing the unique categories and events:

```
unique_category = clean_df['category'].unique()
print(f"Unique values in 'category':")
print(unique_category)
```

```
Unique values in 'category':
['X' 'S' 'B']
```

```
unique_events = clean_df['events'].unique()
print(f"Unique values in 'events':")
print(unique_events)
```

```
Unique values in 'events':
['single' '' 'walk' 'strikeout' 'field_out' 'home_run' 'force_out'
```

```
 'double' 'field_error' 'grounded_into_double_play' 'hit_by_pitch'
 'catcher_interf' 'triple' 'sac_fly' 'double_play' 'sac_bunt'
 'fielders_choice' 'caught_stealing_home' 'fielders_choice_out'
 'caught_stealing_2b' 'strikeout_double_play' 'stolen_base_2b'
 'caught_stealing_3b' 'other_out' 'pickoff_caught_stealing_home'
 'pickoff_caught_stealing_3b' 'pickoff_3b' 'sac_fly_double_play'
 'pickoff_1b' 'triple_play']
```

Designating the unique categories and events as either good or bad based on the pitcher's perspective, creating the binary 'pitch_outcome' variable:

```
bad_events = ['single', 'walk', 'home_run', 'double', 'field_error',
              'hit_by_pitch', 'catcher_interf', 'triple', 'sac_fly',
              'sac_bunt', 'stolen_base_2b']
good_events = ['strikeout', 'field_out', 'force_out',
               'grounded_into_double_play', 'double_play', 'fielders_choice',
               'caught_stealing_home', 'fielders_choice_out',
               'caught_stealing_2b', 'strikeout_double_play',
               'caught_stealing_3b', 'other_out',
               'pickoff_caught_stealing_home', 'pickoff_caught_stealing_3b',
               'pickoff_3b', 'sac_fly_double_play', 'pickoff_1b', 'triple_play']

def classify_pitch_outcome(row):
    if row['events'] in bad_events or row['category'] == 'B':
        return '0'
    elif row['events'] in good_events or row['category'] == 'S':
        return '1'
    else:
        return 'None'

clean_df['pitch_outcome'] = clean_df.apply(classify_pitch_outcome, axis=1)

clean_df['pitch_outcome'].head()
```

```
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\1514293199.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind

0    0
1    1
2    1
3    0
```

```
4    1
Name: pitch_outcome, dtype: object
```

Removing pitch types that are rare/unuseful for the pitch outcome analysis:

```
clean_df = clean_df.drop(
    clean_df[clean_df['pitch_type'].isin(['PO', 'EP', 'FA', 'CS'])].index
)
print(clean_df)
```

```
       pitch_type   game_date  release_speed  release_pos_x  release_pos_z  \
0              FF  2024-04-02           95.0          -2.01           5.22
1              CH  2024-04-02           88.5          -2.09           4.95
2              SI  2024-04-02           95.0          -2.02           5.12
3              SI  2024-04-02           90.7          -1.26           5.13
4              FF  2024-04-02           95.4          -1.95           5.12
...           ...         ...            ...            ...            ...
346245         SL  2024-06-30           85.9           1.63           5.68
346246         SL  2024-06-30           89.9          -1.12           6.41
346247         CU  2024-06-30           73.7          -2.44           5.80
346248         FF  2024-06-30           94.1           2.97           5.97
346249         FF  2024-06-30           99.8          -0.40           6.01

            player_name  batter  pitcher     events      description  ...  \
0        Rocchio, Brayan  677587   622491     single    hit_into_play  ...
1        Rocchio, Brayan  677587   622491                       foul  ...
2        Rocchio, Brayan  677587   622491              called_strike  ...
3         Ohtani, Shohei  660271   657277       walk             ball  ...
4         Hedges, Austin  595978   622491  strikeout         foul_tip  ...
...                  ...     ...      ...        ...              ...  ...
346245        Cave, Jake  595909   694363              called_strike  ...
346246     Jiménez, Eloy  650391   641755               blocked_ball  ...
346247         Rice, Ben  700250   670102              called_strike  ...
346248  Merrifield, Whit  593160   677053                       ball  ...
346249  De La Cruz, Elly  682829   664854             swinging_strike  ...

        post_away_score post_home_score post_bat_score post_fld_score  \
0                     4               0              4               0
1                     4               0              4               0
2                     4               0              4               0
3                     2               5              5               2
4                     4               0              4               0
...                 ...             ...            ...             ...
346245                3               2              3               2
```

```
346246                3              3              3              3
346247                8              1              8              1
346248                6              5              5              6
346249                0              2              0              2

       if_fielding_alignment of_fielding_alignment spin_axis  \
0              Infield shade              Standard     239.0
1                   Standard              Standard     253.0
2                   Standard              Standard     238.0
3              Infield shade              Standard     234.0
4                   Standard              Standard     238.0
...                      ...                   ...       ...
346245              Standard              Standard     144.0
346246              Standard              Standard     200.0
346247         Infield shade              Standard      47.0
346248              Standard              Standard     139.0
346249         Infield shade              Standard     206.0

       delta_home_win_exp delta_run_exp  pitch_outcome
0                  -0.006         0.388              0
1                   0.000        -0.045              1
2                   0.000        -0.042              1
3                   0.009         0.082              0
4                   0.008        -0.215              1
...                   ...           ...            ...
346245              0.000        -0.067              1
346246              0.000         0.057              0
346247              0.000        -0.033              1
346248              0.000         0.032              0
346249              0.000        -0.033              1

[343397 rows x 67 columns]
```

```
outcome_counts = clean_df['pitch_outcome'].value_counts()
print(outcome_counts)
```

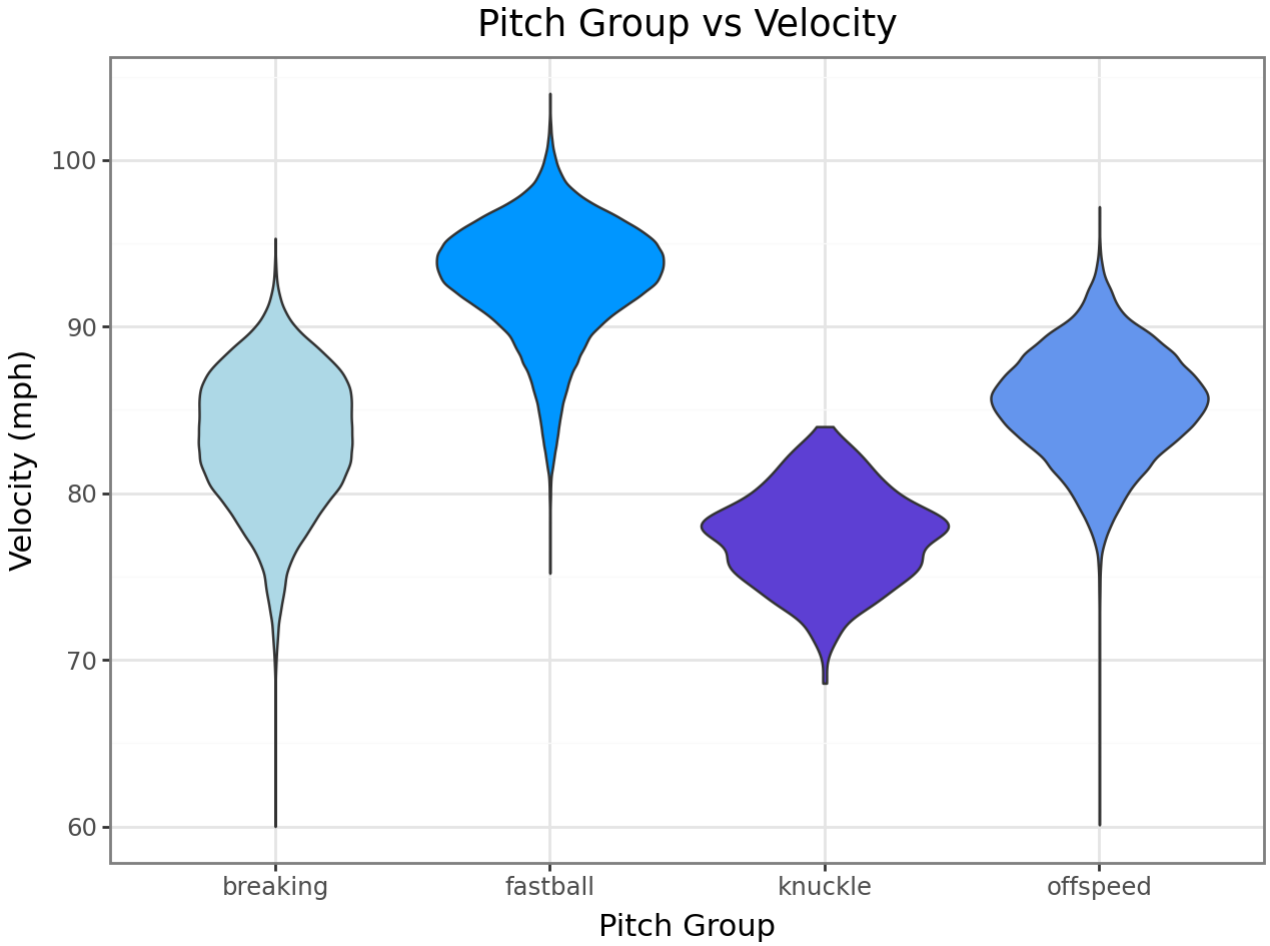
```
pitch_outcome
1    200320
0    143077
Name: count, dtype: int64
```

Engineering the 'pitch_group' variable where 'pitch_types' are sorted based on goal:

```
pitch_group_mapping = {
    'FC': 'fastball', 'FF': 'fastball', 'FS': 'fastball', 'SI': 'fastball',
    'FO': 'fastball', 'SL': 'breaking', 'ST': 'breaking', 'CU': 'breaking',
    'SC': 'breaking', 'KC': 'breaking', 'SV': 'breaking', 'CH': 'offspeed',
    'KN': 'knuckle'
}

clean_df['pitch_group'] = clean_df['pitch_type'].apply(
    lambda x: pitch_group_mapping.get(x, 'unknown')
)
```

Creating a violin plot that charts velocity based on pitch group:

```
from plotnine import *

clean_filtered_df = clean_df[clean_df['pitch_group'] != 'unknown']

colors = ['#ADD8E6', '#0096FF', '#5D3FD3', '#6495ED']

violin_plot = (
    ggplot(clean_filtered_df, aes(x='pitch_group', y='release_speed', fill='pitch_group'))
    + geom_violin(show_legend=False)
    + scale_fill_manual(values=colors)
    + labs(title='Pitch Group vs Velocity', x='Pitch Group', y='Velocity (mph)')
    + theme_bw()
)

violin_plot.show()
```



Demonstrating what the pitch zone looks like:

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, 3, figsize=(6, 8), gridspec_kw={'wspace': 0, 'hspace': 0})

fig.suptitle("Strike Zone From the Catcher's Perspective", fontsize=16)

axs[0, 0].text(0.5, 0.5, '1', fontsize=20, ha='center', va='center')
axs[0, 1].text(0.5, 0.5, '2', fontsize=20, ha='center', va='center')
axs[0, 2].text(0.5, 0.5, '3', fontsize=20, ha='center', va='center')

axs[1, 0].text(0.5, 0.5, '4', fontsize=20, ha='center', va='center')
axs[1, 1].text(0.5, 0.5, '5', fontsize=20, ha='center', va='center')
axs[1, 2].text(0.5, 0.5, '6', fontsize=20, ha='center', va='center')

axs[2, 0].text(0.5, 0.5, '7', fontsize=20, ha='center', va='center')
axs[2, 1].text(0.5, 0.5, '8', fontsize=20, ha='center', va='center')
```

```
axs[2, 2].text(0.5, 0.5, '9', fontsize=20, ha='center', va='center')

for ax in axs.flat:
    ax.set_xticks([])
    ax.set_yticks([])

for ax in axs.flat:
    for _, spine in ax.spines.items():
        spine.set_visible(True)
        spine.set_linewidth(1)
        spine.set_edgecolor('black')

plt.tight_layout(pad=0)
plt.subplots_adjust(top=0.9)
plt.show()
```

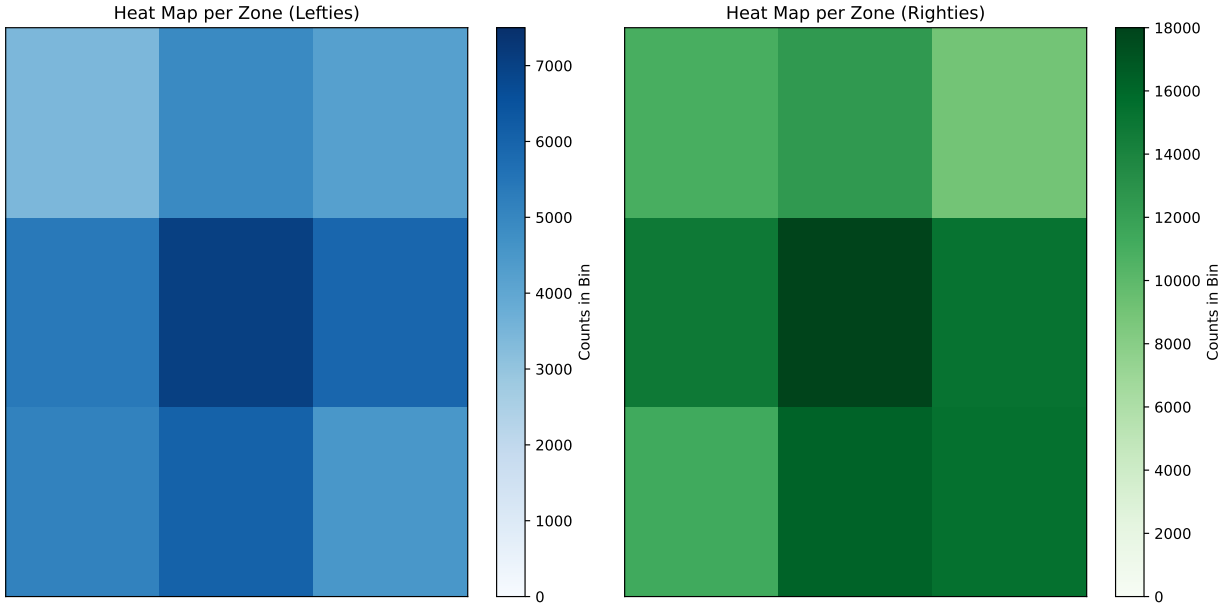
Strike Zone From the Catcher's Perspective

1	2	3
4	5	6
7	8	9

Heat map showing frequency of pitch per zone for left-handed and right-handed pitchers:

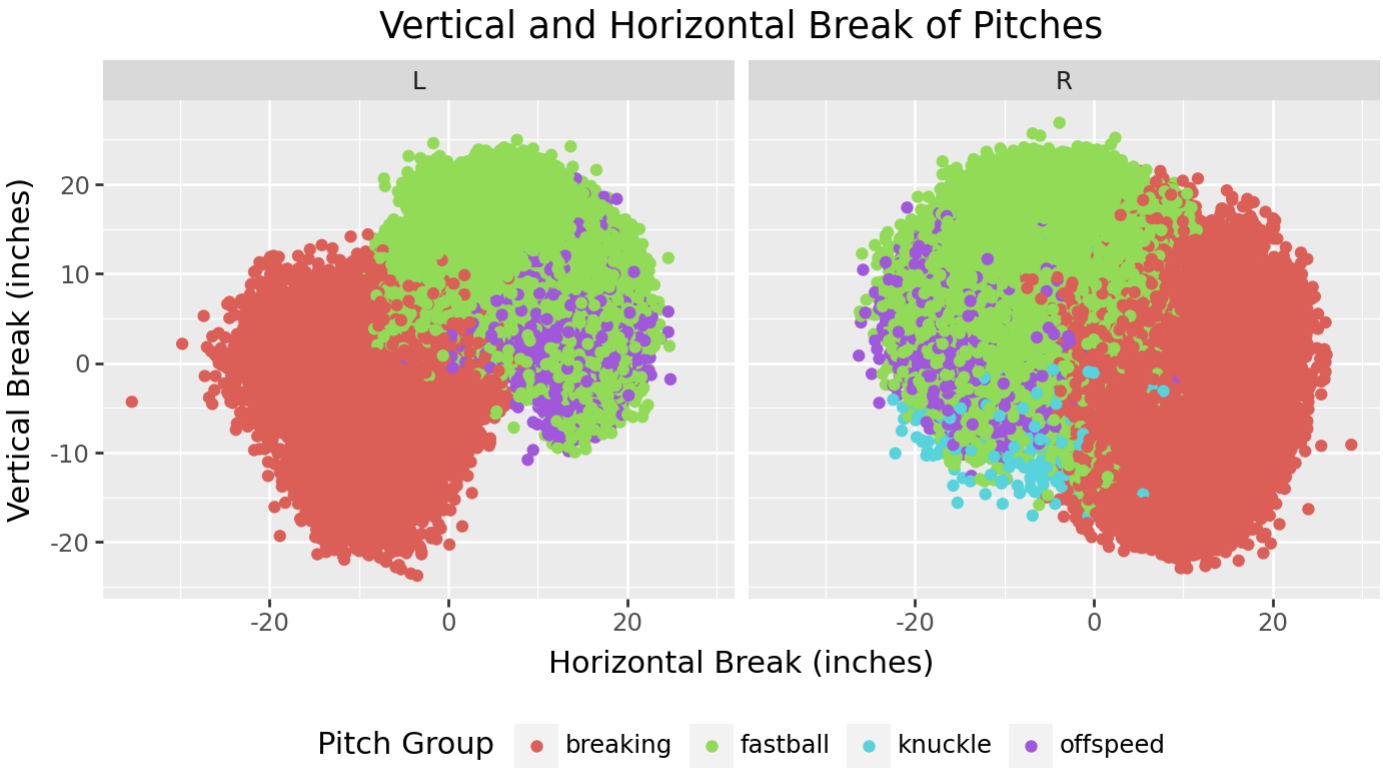
```
def assign_x_coord(row):
    if row.zone in [1, 4, 7]:
        return 1
    if row.zone in [2, 5, 8]:
        return 2
    if row.zone in [3, 6, 9]:
        return 3

def assign_y_coord(row):
    if row.zone in [1, 2, 3]:
        return 3
    if row.zone in [4, 5, 6]:
        return 2
    if row.zone in [7, 8, 9]:
        return 1

clean_df_zones = clean_df.copy().loc[df.zone <= 9]

clean_df_zones['zone_x'] = clean_df_zones.apply(assign_x_coord, axis=1)
clean_df_zones['zone_y'] = clean_df_zones.apply(assign_y_coord, axis=1)

clean_df_lefties = clean_df_zones[clean_df_zones['p_throws'] == 'L']
clean_df_righties = clean_df_zones[clean_df_zones['p_throws'] == 'R']

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.hist2d(
    clean_df_lefties.zone_x, clean_df_lefties.zone_y, bins=3, cmap='Blues',
    vmin=0, vmax=7500
)
plt.title('Heat Map per Zone (Lefties)')
plt.gca().get_xaxis().set_visible(False)
plt.gca().get_yaxis().set_visible(False)
cb_left = plt.colorbar()
cb_left.set_label('Counts in Bin')


plt.subplot(1, 2, 2)
plt.hist2d(
    clean_df_righties.zone_x, clean_df_righties.zone_y, bins=3, cmap='Greens',
    vmin=0, vmax=18000
)
plt.title('Heat Map per Zone (Righties)')
```

```
plt.gca().get_xaxis().set_visible(False)
plt.gca().get_yaxis().set_visible(False)
cb_right = plt.colorbar()
cb_right.set_label('Counts in Bin')

plt.tight_layout()
plt.show()
```



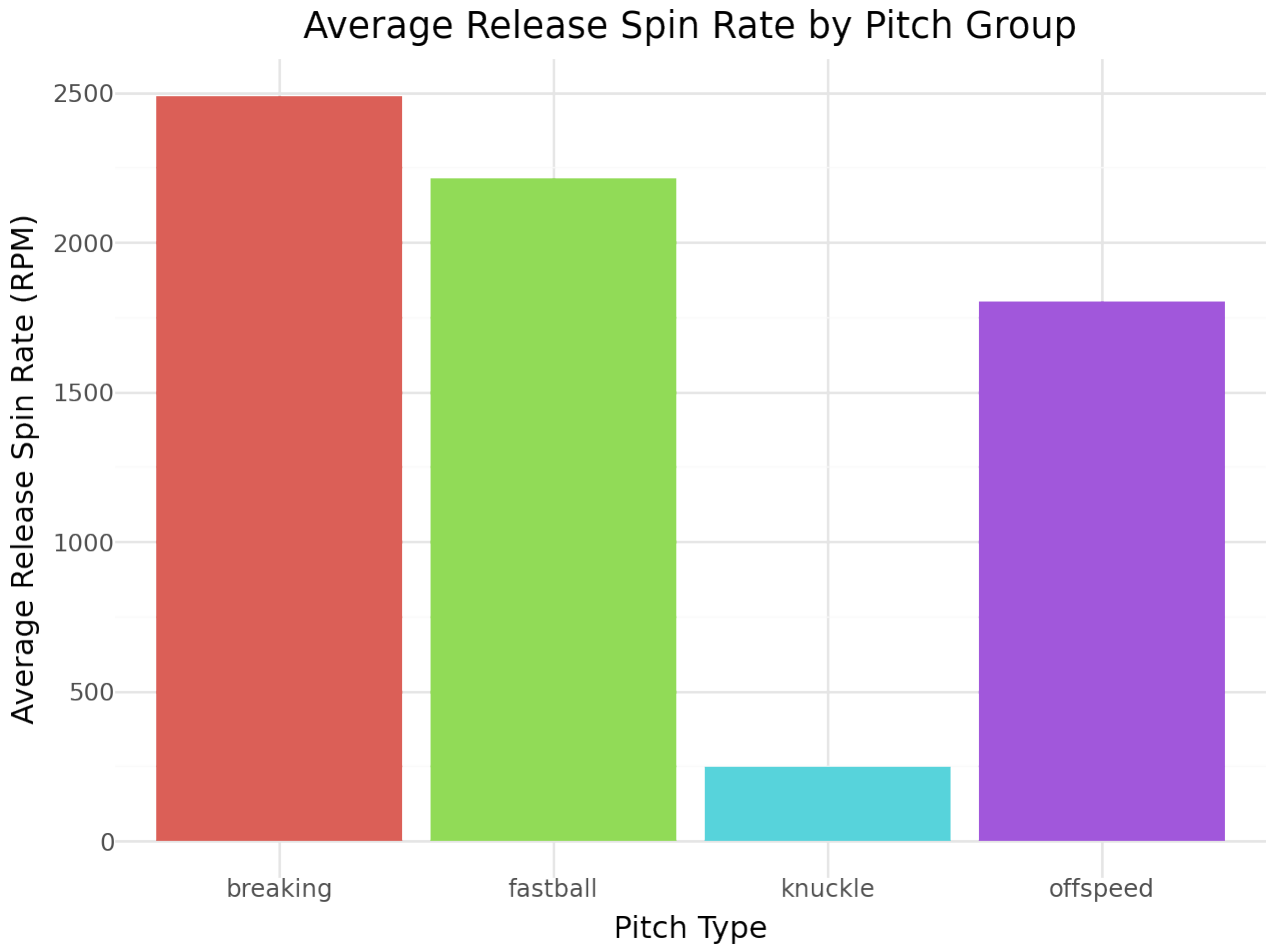
Scatter plot showing vertical and horizontal break for right-handed and left-handed pitchers, color-coded by pitch group:

```
clean_df['pfx_x'] = clean_df['pfx_x'] * 12
clean_df['pfx_z'] = clean_df['pfx_z'] * 12

(ggplot(clean_df, aes(x='pfx_x', y='pfx_z', color='pitch_group')) +
    geom_point() +
    labs(title='Vertical and Horizontal Break of Pitches',
         x='Horizontal Break (inches)',
         y='Vertical Break (inches)',
         color='Pitch Group') +
    coord_fixed(ratio=1) +
    theme(figure_size=(7, 5), legend_position='bottom') +
    facet_wrap('~p_throws')
)
```



Bar plot showing average spin rate for each pitch group:

```
avg_spin_rate_by_pitch_group = clean_filtered_df.groupby('pitch_group')[
    'release_spin_rate'].mean().reset_index()

(ggplot(avg_spin_rate_by_pitch_group,
        aes(x='pitch_group', y='release_spin_rate', fill='pitch_group'))
 + geom_bar(stat='identity', show_legend=False)
 + labs(
     title='Average Release Spin Rate by Pitch Group',
     x='Pitch Type',
     y='Average Release Spin Rate (RPM)'
   )
 + theme_minimal()
)
```



Determining whether data is normally distributed:

```
from scipy.stats import shapiro

numeric_columns = clean_df.select_dtypes(include=['float64', 'int64']).columns

for column in numeric_columns:
    stat, p_value = shapiro(clean_df[column])
    print(f"Shapiro-Wilk test for {column}: Statistic={stat}, P-value={p_value}")

    if p_value < 0.05:
        print(f"{column} is NOT normally distributed (reject H0).")
    else:
        print(f"{column} is normally distributed (fail to reject H0).")
```

```
C:\Users\18607\AppData\Local\Programs\Python\Python312\Lib\site-packages\scipy\stats\_axis_nan_po
```

```
Shapiro-Wilk test for release_speed: Statistic=0.9701733737511701, P-value=8.663200591804365e-103
release_speed is NOT normally distributed (reject H0).
Shapiro-Wilk test for release_pos_x: Statistic=0.8762952538548088, P-value=4.962795546422406e-144
release_pos_x is NOT normally distributed (reject H0).
Shapiro-Wilk test for release_pos_z: Statistic=0.9144160504216262, P-value=1.1294021103232072e-13
release_pos_z is NOT normally distributed (reject H0).
Shapiro-Wilk test for zone: Statistic=0.8966125285483152, P-value=1.9310570216949392e-138
zone is NOT normally distributed (reject H0).
Shapiro-Wilk test for pfx_x: Statistic=0.9619261837407397, P-value=2.294666647703834e-109
pfx_x is NOT normally distributed (reject H0).
Shapiro-Wilk test for pfx_z: Statistic=0.9683092419645413, P-value=2.1076030985318253e-104
pfx_z is NOT normally distributed (reject H0).
Shapiro-Wilk test for plate_x: Statistic=0.9996202228285507, P-value=3.512993589748462e-20
plate_x is NOT normally distributed (reject H0).
Shapiro-Wilk test for plate_z: Statistic=0.9991295918924639, P-value=8.510742439497825e-31
plate_z is NOT normally distributed (reject H0).
Shapiro-Wilk test for vx0: Statistic=0.936654681555992, P-value=9.511215345338106e-124
vx0 is NOT normally distributed (reject H0).
Shapiro-Wilk test for vy0: Statistic=0.9702744542901001, P-value=1.0657903139189927e-102
vy0 is NOT normally distributed (reject H0).
Shapiro-Wilk test for vz0: Statistic=0.9955082712499248, P-value=7.9542941062316125e-59
vz0 is NOT normally distributed (reject H0).
Shapiro-Wilk test for ax: Statistic=0.9735965124304796, P-value=1.3939490407933937e-99
ax is NOT normally distributed (reject H0).
Shapiro-Wilk test for ay: Statistic=0.9941126796590561, P-value=2.405069053473543e-64
ay is NOT normally distributed (reject H0).
Shapiro-Wilk test for az: Statistic=0.9794047505775959, P-value=3.3029061798519583e-93
az is NOT normally distributed (reject H0).
Shapiro-Wilk test for sz_top: Statistic=0.9978745614544096, P-value=6.926248981348034e-45
sz_top is NOT normally distributed (reject H0).
Shapiro-Wilk test for sz_bot: Statistic=0.9978675745365625, P-value=6.075793937482741e-45
sz_bot is NOT normally distributed (reject H0).
Shapiro-Wilk test for effective_speed: Statistic=0.9734425890658718, P-value=9.829350452994871e-1
effective_speed is NOT normally distributed (reject H0).
Shapiro-Wilk test for release_spin_rate: Statistic=0.9359555645846115, P-value=4.551938588479931e
release_spin_rate is NOT normally distributed (reject H0).
Shapiro-Wilk test for release_extension: Statistic=0.9947531897347226, P-value=5.779806431006556e
release_extension is NOT normally distributed (reject H0).
Shapiro-Wilk test for release_pos_y: Statistic=0.9986522811556826, P-value=2.3739250167464235e-37
release_pos_y is NOT normally distributed (reject H0).
Shapiro-Wilk test for spin_axis: Statistic=0.9604286698530877, P-value=2.006343423164368e-110
spin_axis is NOT normally distributed (reject H0).
Shapiro-Wilk test for delta_home_win_exp: Statistic=0.44492758722732173, P-value=3.36693544569397
delta_home_win_exp is NOT normally distributed (reject H0).
Shapiro-Wilk test for delta_run_exp: Statistic=0.6627796140180917, P-value=2.4245423639969912e-17
```

```
delta_run_exp is NOT normally distributed (reject H0).
```

Because data is non-parametric, use Kruskal-Wallis test to determine if there is a statistically significant difference in velocity based on pitch group:

```
from scipy.stats import kruskal

pitch_groups = clean_df['pitch_group'].unique()

grouped_data = [
    clean_df[clean_df['pitch_group'] == group]['release_speed'].dropna()
    for group in pitch_groups
]

stat, p_value = kruskal(*grouped_data)

print(f"Kruskal-Wallis test result: Statistic={stat}, P-value={p_value}")

if p_value < 0.05:
    print("There is a significant difference in pitch speeds between pitch groups "
          "(reject H0).")
else:
    print("There is no significant difference in pitch speeds between pitch groups "
          "(fail to reject H0).")
```

```
Kruskal-Wallis test result: Statistic=210487.8574218729, P-value=0.0
There is a significant difference in pitch speeds between pitch groups (reject H0).
```

Kruskal-Wallis test to see if there is a statistically significant difference in horizontal and vertical break based on pitch group:

```
grouped_horizontal = [
    clean_df[clean_df['pitch_group'] == group]['pfx_x'].dropna()
    for group in pitch_groups
]

grouped_vertical = [
    clean_df[clean_df['pitch_group'] == group]['pfx_z'].dropna()
    for group in pitch_groups
]

stat_x, p_value_x = kruskal(*grouped_horizontal)
print(f"Kruskal-Wallis test for horizontal break: Statistic={stat_x}, "
      f"P-value={p_value_x}")
```

```
stat_z, p_value_z = kruskal(*grouped_vertical)
print(f"Kruskal-Wallis test for vertical break: Statistic={stat_z}, P-value={p_value_z}")

if p_value_x < 0.05:
    print("There is a significant difference in horizontal breaks between pitch groups "
          "(reject H0).")
else:
    print("There is no significant difference in horizontal breaks between pitch groups "
          "(fail to reject H0).")

if p_value_z < 0.05:
    print("There is a significant difference in vertical breaks between pitch groups "
          "(reject H0).")
else:
    print("There is no significant difference in vertical breaks between pitch groups "
          "(fail to reject H0).")
```

```
Kruskal-Wallis test for horizontal break: Statistic=39725.47250271622, P-value=0.0
Kruskal-Wallis test for vertical break: Statistic=172196.66805453913, P-value=0.0
There is a significant difference in horizontal breaks between pitch groups (reject H0).
There is a significant difference in vertical breaks between pitch groups (reject H0).
```

Kruskal-Wallis test to determine if there is a statistically significant difference in spin rate based on pitch group:

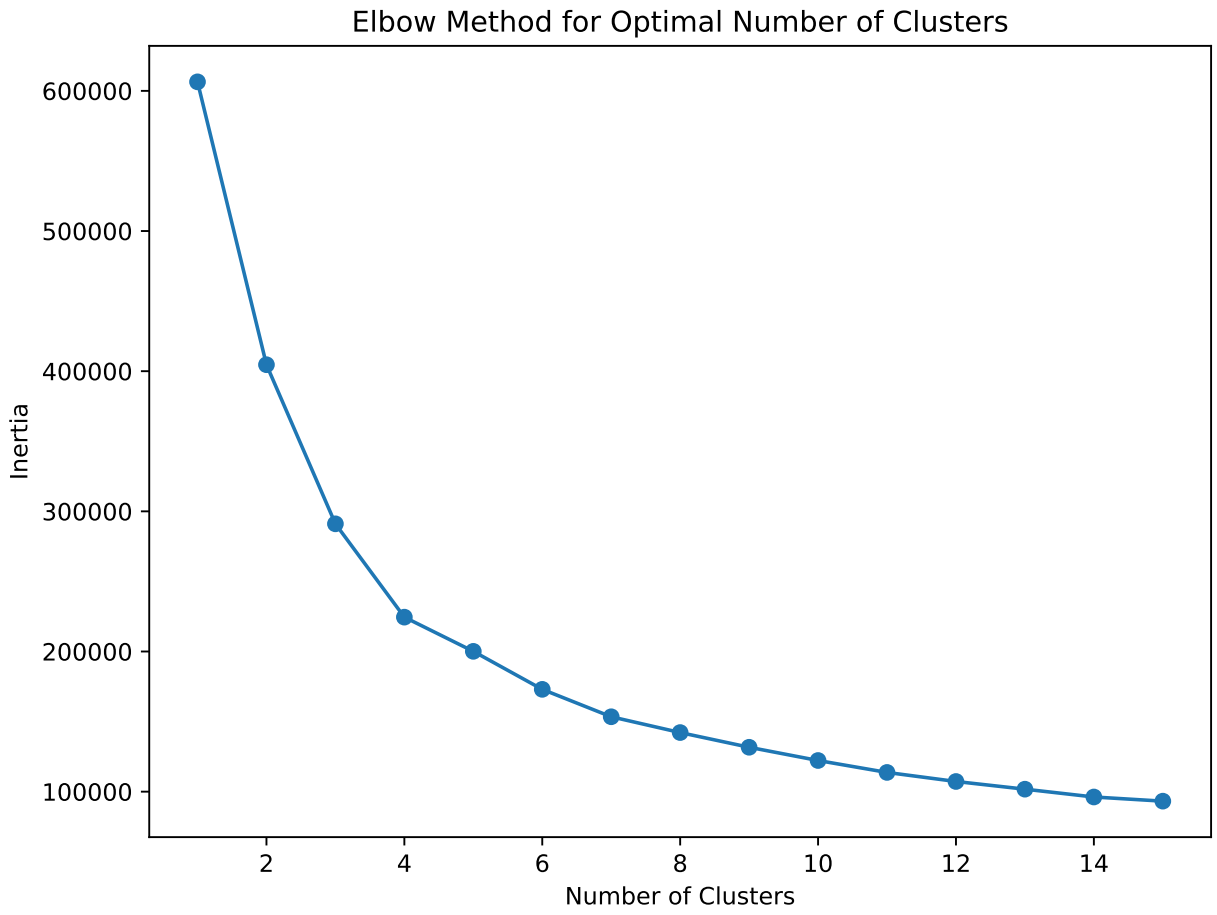
```
grouped_spin_rate = [
    clean_df[clean_df['pitch_group'] == group]['release_spin_rate'].dropna()
    for group in pitch_groups
]

stat, p_value = kruskal(*grouped_spin_rate)

print(f"Kruskal-Wallis test result for release spin rate: Statistic={stat}, "
      f"P-value={p_value}")

if p_value < 0.05:
    print("There is a significant difference in release spin rates between pitch "
          "groups (reject H0).")
else:
    print("There is no significant difference in release spin rates between pitch "
          "groups (fail to reject H0).")
```

```
Kruskal-Wallis test result for release spin rate: Statistic=108414.15977712892, P-value=0.0
There is a significant difference in release spin rates between pitch groups (reject H0).
```

Data Analysis

Determine necessary number of clusters to limit inertia while standardizing the data to prepare for k-means clustering:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

clean_df = clean_df[clean_df['pitch_group'] == 'fastball']

features = clean_df[['release_speed', 'pfx_x', 'pfx_z']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

inertia = []
k_range = range(1, 16)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=1918)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 6))
plt.plot(k_range, inertia, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```



Determine the variance explained by each PCA component:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_components = pca.fit_transform(features)

print("Explained variance ratio by each component:")
print(pca.explained_variance_ratio_)

print("\nPrincipal components (directions):")
print(pca.components_)
```

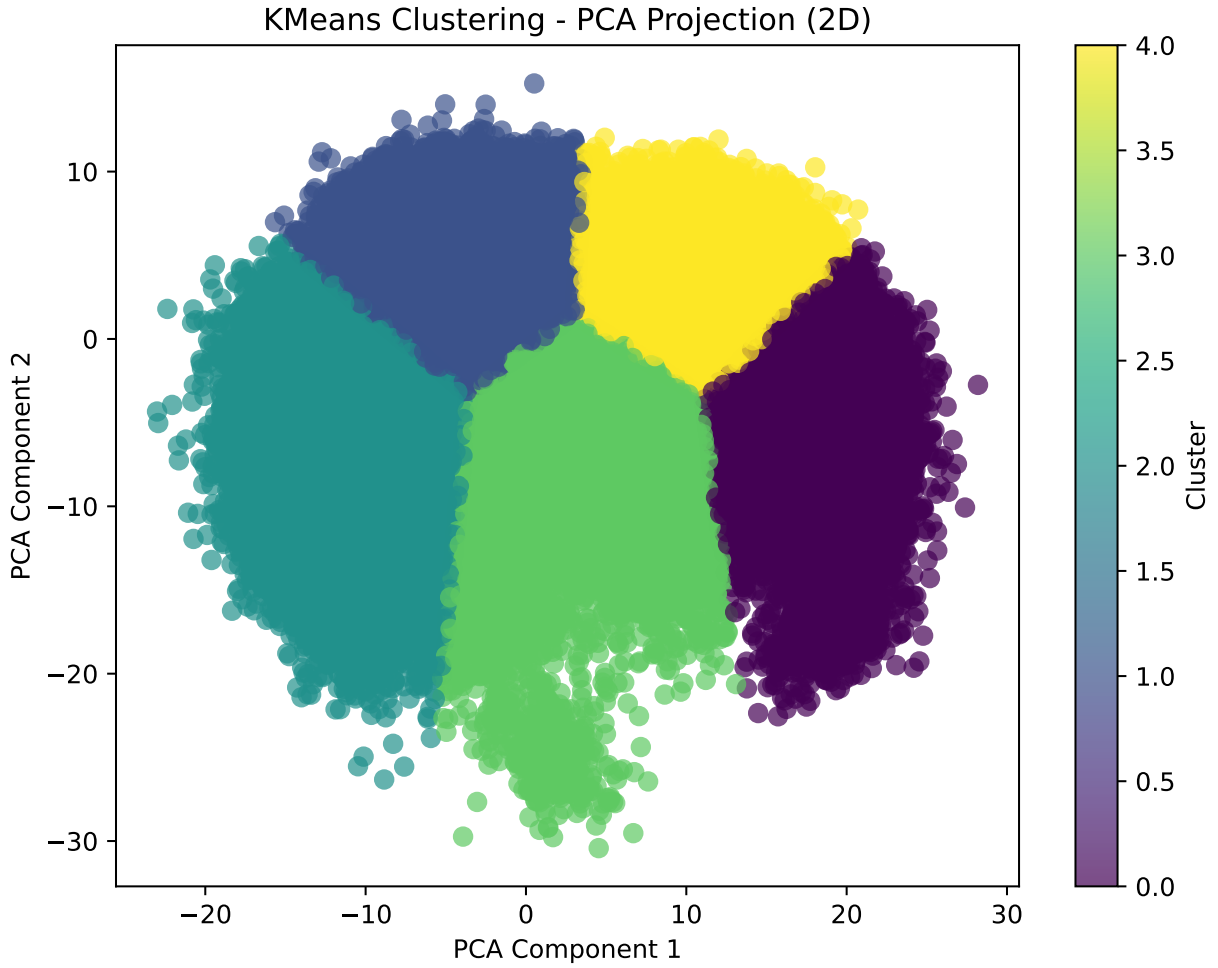
```
Explained variance ratio by each component:
[0.69114799 0.25398751]

Principal components (directions):
[[-0.05013017  0.99642904  0.06794214]
```

```
 [ 0.35713866 -0.0456452   0.93293542]]
```

Visualization of 5 created clusters based on elbow method:

```
kmeans = KMeans(n_clusters=5, random_state=1918)
clean_df['cluster'] = kmeans.fit_predict(pca_components)

plt.figure(figsize=(8, 6))
plt.scatter(pca_components[:, 0], pca_components[:, 1],
            c=clean_df['cluster'], cmap='viridis',
            s=50, alpha=0.7)

plt.title('KMeans Clustering - PCA Projection (2D)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

```
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\2566601178.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
```



Using k-means clustering to group pitchers based on velocity, horizontal break, and vertical break tendencies, then ranking those clusters to create deciles:

Those rankings are then used to determine pitchers prioritizing movement with the most pitches in the top 30% for vertical and horizontal break but in the bottom 20% for velocity:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

features = clean_df[['release_speed', 'pfx_x', 'pfx_z']]

features = pd.get_dummies(features, drop_first=True)

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

```
pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_features)

kmeans = KMeans(n_clusters=5, random_state=1918)
clean_df['cluster'] = kmeans.fit_predict(pca_components)

clean_df.loc[:, 'velocity_rank'] = clean_df.groupby('cluster')['release_speed'].rank(pct=True)
clean_df.loc[:, 'hbreak_rank'] = clean_df.groupby('cluster')['pfx_x'].rank(pct=True)
clean_df.loc[:, 'vbreak_rank'] = clean_df.groupby('cluster')['pfx_z'].rank(pct=True)

clean_df.loc[:, 'vbreak_decile'] = (clean_df['vbreak_rank'] * 10).astype(int)
clean_df.loc[:, 'velocity_decile'] = (clean_df['velocity_rank'] * 10).astype(int)
clean_df.loc[:, 'hbreak_decile'] = (clean_df['hbreak_rank'] * 10).astype(int)

clean_df['velocity_improvement_candidate'] = (
    (clean_df['vbreak_decile'] >= 3) &
    (clean_df['hbreak_decile'] >= 3) &
    (clean_df['velocity_decile'] <= 2)
)

velocity_improvement_candidates = clean_df[clean_df['velocity_improvement_candidate'] == True].co

velocity_improvement_candidates_sorted = velocity_improvement_candidates.sort_values(by='pitcher'

top_5_velocity_improvement_pitchers = velocity_improvement_candidates_sorted['pitcher'].value_cou

print("Top 5 Pitchers Prioritizing Movement:")
print(top_5_velocity_improvement_pitchers)
```

```
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
```

```
Top 5 Pitchers Prioritizing Movement:
pitcher
676710    734
684007    673
665871    610
594902    588
641927    504
Name: count, dtype: int64


C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
C:\Users\18607\AppData\Local\Temp\ipykernel_12672\3805862247.py:27: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
```

542881: Tyler Anderson 596295: Austin Gomber 676710: Kutter Crawford 594902: Ben Lively 684007: Shota Imanaga

Determine pitchers prioritizing velocity with the most pitches in the top 20% for velocity but the bottom 30% for both horizonal and vertical break:

```
clean_df['break_improvement_candidate'] = (
    (clean_df['vbreak_decile'] <= 3) &
    (clean_df['hbreak_decile'] <= 3) &
    (clean_df['velocity_decile'] >= 8)
```

```
)

break_improvement_candidates = clean_df[clean_df[
    'break_improvement_candidate'] == True].copy()

break_improvement_candidates_sorted = break_improvement_candidates.sort_values(
    by='pitcher')

top_5_break_improvement_pitchers = break_improvement_candidates_sorted[
    'pitcher'].value_counts().head(5)

print("\nTop 5 Pitchers Prioritizing Velocity:")
print(top_5_break_improvement_pitchers)
```

```
Top 5 Pitchers Prioritizing Velocity:
pitcher
667755    363
694973    336
678394    255
657044    250
687330    239
Name: count, dtype: int64

C:\Users\18607\AppData\Local\Temp\ipykernel_12672\2766881952.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
```

667755: Jose Soriano 665625: Elvis Peguero 666974: Yennier Cano 694973: Paul Skenes 656557: Tanner Houck

Sort the variables into categorical and numerical groups:

```
categorical_col = [
    'pitch_type',
    'stand',
    'p_throws',
    'home_team',
    'away_team',
    'bb_type',
    'inning_topbot',
    'if_fielding_alignment',
```

```
    'of_fielding_alignment',
    'pitch_group'
]

numerical_col = [
    'release_speed', 'release_pos_x', 'release_pos_z', 'batter', 'pitcher',
    'zone', 'balls', 'strikes', 'pfx_x', 'pfx_z', 'plate_x', 'plate_z',
    'outs_when_up', 'inning', 'sz_top', 'sz_bot', 'release_spin_rate',
    'release_extension', 'game_pk', 'release_pos_y', 'at_bat_number',
    'pitch_number', 'home_score', 'away_score', 'bat_score', 'fld_score', 'spin_axis'
]
```

Scale both the numerical and categorical variables and enter them into a preprocessor:

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

numerical_transformer = StandardScaler()

categorical_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_col),
        ('num', numerical_transformer, numerical_col)
    ]
)
```

Use a pipeline to turn the preprocessed variables into a LASSO logistic model; label X and Y variables and sort the data into training and testing sets before fitting the model:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(penalty='l1', solver='liblinear',
                                      max_iter=1000))
])

X = X = clean_df[numerical_col + categorical_col]
y = clean_df['pitch_outcome']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1918)

pipeline.fit(X_train, y_train)
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('cat', OneHotEncoder(),
                                                  ['pitch_type', 'stand',
                                                   'p_throws', 'home_team',
                                                   'away_team', 'bb_type',
                                                   'inning_topbot',
                                                   'if_fielding_alignment',
                                                   'of_fielding_alignment',
                                                   'pitch_group']),
                                                 ('num', StandardScaler(),
                                                  ['release_speed',
                                                   'release_pos_x',
                                                   'release_pos_z', 'batter',
                                                   'pitcher', 'zone', 'balls',
                                                   'strikes', 'pfx_x', 'pfx_z',
                                                   'plate_x', 'plate_z',
                                                   'outs_when_up', 'inning',
                                                   'sz_top', 'sz_bot',
                                                   'release_spin_rate',
                                                   'release_extension',
                                                   'game_pk', 'release_pos_y',
                                                   'at_bat_number',
                                                   'pitch_number', 'home_score',
                                                   'away_score', 'bat_score',
                                                   'fld_score',
                                                   'spin_axis'])])),
                ('classifier',
                 LogisticRegression(max_iter=1000, penalty='l1',
                                    solver='liblinear'))])
```

Determine the intercept and coefficients from the LASSO logistic model:

```
model = pipeline.named_steps['classifier']
intercept = model.intercept_
coefficients = model.coef_[0]

intercept, coefficients
```

```
(array([0.04917749]),
 array([ 6.47497386e-02, -8.00332654e-02,  5.07378770e-01,  1.73919960e-01,
         0.00000000e+00,  1.75574813e-01,  2.63615087e-01,  6.31418576e-02,
         0.00000000e+00,  0.00000000e+00, -1.17045754e-01,  1.40553358e-02,
         4.45796226e-02,  3.03782948e-02, -7.46869066e-02,  4.26391895e-02,
        -4.62976922e-02,  7.84564864e-02,  7.35028642e-02,  1.38936963e-02,
        -3.01683546e-02, -7.34585708e-03,  4.35773985e-02, -6.37709987e-02,
        -3.63595851e-02,  9.39811232e-03,  4.81238480e-03, -6.97506347e-03,
        -6.52865215e-02, -7.28302257e-02, -5.09484625e-02,  0.00000000e+00,
         4.03698770e-02, -9.36832000e-02, -7.89961538e-02,  1.89547554e-03,
         1.82884622e-02,  1.08291163e-02, -3.28502894e-02,  3.38405770e-03,
        -1.09632999e-03,  1.53553856e-02,  3.35321253e-02, -6.12948336e-03,
        -4.81040400e-03,  3.71106106e-02,  3.82470269e-02, -3.63659012e-03,
         8.73857055e-02,  0.00000000e+00, -1.58052845e-03, -5.54021577e-02,
        -8.54961474e-03,  2.77059196e-02, -2.51098631e-02,  2.28637512e-02,
        -3.51806547e-02, -1.16495098e-02, -2.96365472e-02,  1.94286169e-02,
         9.55463422e-02,  1.70927812e-02,  9.76869732e-03, -7.28898409e-03,
         1.91963731e-02,  6.30496939e-03,  0.00000000e+00, -1.76853977e-02,
         0.00000000e+00,  4.51042422e-01, -3.06597030e-01,  5.84034377e-02,
        -2.06127651e+00,  3.65548999e+00,  0.00000000e+00,  1.76101722e-02,
        -8.35101998e-03,  0.00000000e+00,  1.14479606e-02,  0.00000000e+00,
        -1.56696593e-01,  8.30249339e-02,  6.95145324e-02,  2.17120694e-02,
         3.11911280e-02, -3.38070300e-02, -3.17769764e-03,  1.63596200e-02,
        -1.55986722e-02, -1.65301288e+00,  2.97927414e-02,  1.17286548e-02,
         1.95422633e-02, -5.93662774e-03,  1.38573127e-01, -3.81993911e-01,
         9.70392824e-03,  9.86595462e-02,  2.54080118e-02, -8.20165898e-03,
         2.26221661e-02,  8.71117046e-02, -3.09792505e-02,  9.35840667e-02,
        -1.11059751e-01,  1.20372990e-01,  4.47419526e-04,  8.53176733e-03,
         5.47473728e-04,  2.77548317e-02,  2.29568361e-02]))
```

Test the model using the test data set, print the validation considerations for the model based on this test:

```
from sklearn.metrics import recall_score, precision_score, f1_score, accuracy_score, confusion_ma

y_pred = pipeline.predict(X_test)

y_test = y_test.astype(int)
y_pred = y_pred.astype(int)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='binary')
precision = precision_score(y_test, y_pred, average='binary')
f1 = f1_score(y_test, y_pred, average='binary')
cm = confusion_matrix(y_test, y_pred)
```

```
print(f"Accuracy: {accuracy}")
print(f"Recall: {recall}")
print(f"Precision: {precision}")
print(f"F1 Score: {f1}")
print("Confusion Matrix:")
print(cm)
```

```
Accuracy: 0.7990106356665843
Recall: 0.7733980023839861
Precision: 0.878062438751225
F1 Score: 0.8224135670265309
Confusion Matrix:
[[13488  2613]
 [ 5513 18816]]
```

Tune the parameters using a 'grid_search' to ensure the model is optimized, then print the validation values using the best parameters:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'classifier__penalty': ['l1'],
    'classifier__solver': ['liblinear', 'saga'],
    'classifier__max_iter': [1000, 10000]
}

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', verbose=1, n_jobs=-1)

grid_search.fit(X_train, y_train)

print(f"Best parameters found: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_}")

best_model = grid_search.best_estimator_

y_pred_best = best_model.predict(X_test)
y_pred_best = y_pred_best.astype(int)

accuracy_best = accuracy_score(y_test, y_pred_best)
recall_best = recall_score(y_test, y_pred_best, average='binary')
precision_best = precision_score(y_test, y_pred_best, average='binary')
f1_best = f1_score(y_test, y_pred_best, average='binary')
cm_best = confusion_matrix(y_test, y_pred_best)
```

```
print(f"Accuracy: {accuracy_best}")
print(f"Recall: {recall_best}")
print(f"Precision: {precision_best}")
print(f"F1 Score: {f1_best}")
print("Confusion Matrix:")
print(cm_best)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameters found: {'classifier__max_iter': 10000, 'classifier__penalty': 'l1', 'classifier__
Best cross-validation score: 0.7990700184040946
Accuracy: 0.7990106356665843
Recall: 0.7733980023839861
Precision: 0.878062438751225
F1 Score: 0.8224135670265309
Confusion Matrix:
[[13488  2613]
 [ 5513 18816]]
```

Finding the coefficients of the pitch group feature within the LASSO logistic model to determine their importance in creating predictions:

```
pipeline.fit(X_train, y_train)

categorical_transformer = pipeline.named_steps['preprocessor'].transformers_[0][1]

categorical_feature_names = categorical_transformer.get_feature_names_out(categorical_col)

all_feature_names = numerical_col + list(categorical_feature_names)

feature_importance = pd.DataFrame({
    'Feature': all_feature_names,
    'Coefficient': coefficients
})

feature_importance['Abs_Coefficient'] = feature_importance['Coefficient'].abs()

pitch_group_features = [
    feature for feature in categorical_feature_names if 'pitch_group' in feature
]

feature_importance_pitch_group = feature_importance[
    feature_importance['Feature'].isin(pitch_group_features)
]
```

```
print(feature_importance_pitch_group[['Feature', 'Coefficient', 'Abs_Coefficient']])
```

```
                  Feature  Coefficient  Abs_Coefficient
110  pitch_group_fastball     0.022957         0.022957
```