In [1]:
```
pip install pybaseball
```

```
Requirement already satisfied: pybaseball in /Users/kishan/Documents/Documen
ts/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (2.2.7)
Requirement already satisfied: numpy>=1.13.0 in /Users/kishan/Documents/Docu
ments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fro
m pybaseball) (1.26.1)
Requirement already satisfied: pandas>=1.0.3 in /Users/kishan/Documents/Docu
ments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fro
m pybaseball) (2.1.2)
Requirement already satisfied: beautifulsoup4>=4.4.0 in /Users/kishan/Docume
nts/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packa
ges (from pybaseball) (4.12.2)
Requirement already satisfied: requests>=2.18.1 in /Users/kishan/Documents/D
ocuments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (
from pybaseball) (2.31.0)
Requirement already satisfied: lxml>=4.2.1 in /Users/kishan/Documents/Docume
nts/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
pybaseball) (4.9.3)
Requirement already satisfied: pyarrow>=1.0.1 in /Users/kishan/Documents/Doc
uments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fr
om pybaseball) (14.0.0)
Requirement already satisfied: pygithub>=1.51 in /Users/kishan/Documents/Doc
uments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fr
om pybaseball) (2.1.1)
Requirement already satisfied: scipy>=1.4.0 in /Users/kishan/Documents/Docum
ents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
pybaseball) (1.11.3)
Requirement already satisfied: matplotlib>=2.0.0 in /Users/kishan/Documents/
Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages
(from pybaseball) (3.8.1)
Requirement already satisfied: tqdm>=4.50.0 in /Users/kishan/Documents/Docum
ents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
pybaseball) (4.66.1)
Requirement already satisfied: attrs>=20.3.0 in /Users/kishan/Documents/Docu
ments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fro
m pybaseball) (23.1.0)
Requirement already satisfied: soupsieve>1.2 in /Users/kishan/Documents/Docu
ments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fro
m beautifulsoup4>=4.4.0->pybaseball) (2.5)
Requirement already satisfied: contourpy>=1.0.1 in /Users/kishan/Documents/D
ocuments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (
from matplotlib>=2.0.0->pybaseball) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /Users/kishan/Documents/Docum
ents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
matplotlib>=2.0.0->pybaseball) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/kishan/Documents/
Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages
(from matplotlib>=2.0.0->pybaseball) (4.43.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/kishan/Documents/
Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages
(from matplotlib>=2.0.0->pybaseball) (1.4.5)
```

```
Requirement already satisfied: packaging>=20.0 in /Users/kishan/Documents/Do
cuments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (f
rom matplotlib>=2.0.0->pybaseball) (23.2)
Requirement already satisfied: pillow>=8 in /Users/kishan/Documents/Document
s/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from ma
tplotlib>=2.0.0->pybaseball) (10.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/kishan/Documents/D
ocuments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (
from matplotlib>=2.0.0->pybaseball) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /Users/kishan/Documen
ts/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packag
es (from matplotlib>=2.0.0->pybaseball) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in /Users/kishan/D
ocuments/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-
packages (from matplotlib>=2.0.0->pybaseball) (6.1.0)
Requirement already satisfied: pytz>=2020.1 in /Users/kishan/Documents/Docum
ents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
pandas>=1.0.3->pybaseball) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /Users/kishan/Documents/Doc
uments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fr
om pandas>=1.0.3->pybaseball) (2023.3)
Requirement already satisfied: pynacl>=1.4.0 in /Users/kishan/Documents/Docu
ments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fro
m pygithub>=1.51->pybaseball) (1.5.0)
Requirement already satisfied: pyjwt>=2.4.0 in /Users/kishan/Documents/Docum
ents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
pyjwt[crypto]>=2.4.0->pygithub>=1.51->pybaseball) (2.8.0)
Requirement already satisfied: typing-extensions>=4.0.0 in /Users/kishan/Doc
uments/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-pa
ckages (from pygithub>=1.51->pybaseball) (4.8.0)
Requirement already satisfied: urllib3>=1.26.0 in /Users/kishan/Documents/Do
cuments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (f
rom pygithub>=1.51->pybaseball) (2.0.7)
Requirement already satisfied: Deprecated in /Users/kishan/Documents/Documen
ts/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from p
ygithub>=1.51->pybaseball) (1.2.14)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/kishan/Doc
uments/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-pa
ckages (from requests>=2.18.1->pybaseball) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /Users/kishan/Documents/Docum
ents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
requests>=2.18.1->pybaseball) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /Users/kishan/Document
s/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-package
s (from requests>=2.18.1->pybaseball) (2023.7.22)
Requirement already satisfied: zipp>=3.1.0 in /Users/kishan/Documents/Docume
nts/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
importlib-resources>=3.2.0->matplotlib>=2.0.0->pybaseball) (3.17.0)
Requirement already satisfied: cryptography>=3.4.0 in /Users/kishan/Document
s/Documents/IST 707 Applied Machine Learning/venv/lib/python3.9/site-package
s (from pyjwt[crypto]>=2.4.0->pygithub>=1.51->pybaseball) (41.0.5)
Requirement already satisfied: cffi>=1.4.1 in /Users/kishan/Documents/Docume
nts/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from
```

```
pynacl>=1.4.0->pygithub>=1.51->pybaseball) (1.16.0)
Requirement already satisfied: six>=1.5 in /Users/kishan/Documents/Document
s/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from py
thon-dateutil>=2.7->matplotlib>=2.0.0->pybaseball) (1.16.0)
Requirement already satisfied: wrapt<2,>=1.10 in /Users/kishan/Documents/Doc
uments/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (fr
om Deprecated->pygithub>=1.51->pybaseball) (1.14.1)
Requirement already satisfied: pycparser in /Users/kishan/Documents/Document
s/IST 707 Applied Machine Learning/venv/lib/python3.9/site-packages (from cf
fi>=1.4.1->pynacl>=1.4.0->pygithub>=1.51->pybaseball) (2.21)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
#Import initial libraries

import pybaseball
import numpy as np
import pandas as pd
import matplotlib.pyplot as plot
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

```
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/urllib3/__init__.py:34: NotOpenSSLWarning: urllib3 v
2.0 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled wit
h 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
```

In [3]:
```
#We use pybaseball data and split it in regular and post season dataframes
df_reg = pybaseball.statcast(start_dt = '2023-03-30', end_dt = '2023-10-02')
df_post = pybaseball.statcast(start_dt = '2023-10-03', end_dt = '2023-11-01'
```

```
This is a large query, it may take a moment to complete
```

```
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/pybaseball/statcast.py:50: UserWarning:
That's a nice request you got there. It'd be a shame if something were to ha
ppen to it.
We strongly recommend that you enable caching before running this. It's as s
imple as `pybaseball.cache.enable()`.
Since the Statcast requests can take a *really* long time to run, if somethi
ng were to happen, like: a disconnect;
gremlins; computer repair by associates of Rudy Giuliani; electromagnetic in
terference from metal trash cans; etc.;
you could lose a lot of progress. Enabling caching will allow you to immedia
tely recover all the successful
subqueries if that happens.
  warnings.warn(_OVERSIZE_WARNING)
100%|██████████████████████████████████████████| 187/187 [01:28<00:00,  2.11i
t/s]
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/pybaseball/statcast.py:85: FutureWarning: The behavi
or of DataFrame concatenation with empty or all-NA entries is deprecated. In
a future version, this will no longer exclude empty or all-NA columns when d
etermining the result dtypes. To retain the old behavior, exclude the releva
nt entries before the concat operation.
  final_data = pd.concat(dataframe_list, axis=0).convert_dtypes(convert_stri
ng=False)
This is a large query, it may take a moment to complete
100%|██████████████████████████████████████████| 30/30 [00:04<00:00,  6.18it
/s]
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/pybaseball/statcast.py:85: FutureWarning: The behavi
or of DataFrame concatenation with empty or all-NA entries is deprecated. In
a future version, this will no longer exclude empty or all-NA columns when d
etermining the result dtypes. To retain the old behavior, exclude the releva
nt entries before the concat operation.
  final_data = pd.concat(dataframe_list, axis=0).convert_dtypes(convert_stri
ng=False)
```

In [4]:
```
#We handpicked columns that we felt did not convey much information or were
#After this, we ended up with 32 columns that we will use in our analyses

df_reg = df_reg.drop(['game_date', 'spin_dir', 'spin_rate_deprecated', 'brea
                  'des', 'game_year', 'inning_topbot', 'hc_x', 'hc_y', 'tfs_d
                  'umpire', 'sv_id', 'vx0', 'vy0', 'vz0', 'ax', 'ay', 'az', '
                  'release_extension', 'release_pos_y', 'babip_value', 'woba_
                  'bat_score', 'fld_score', 'post_away_score', 'post_home_sco
                  'spin_axis', 'release_speed', 'events', 'description', 'zon
                   'pfx_x', 'pfx_z', 'plate_x', 'plate_z','hit_distance_sc',
                   'estimated_woba_using_speedangle', 'launch_speed_angle', '
df_post = df_post.drop(['game_date', 'spin_dir', 'spin_rate_deprecated', 'br
                  'des', 'game_year', 'inning_topbot', 'hc_x', 'hc_y', 'tfs_d
                  'umpire', 'sv_id', 'vx0', 'vy0', 'vz0', 'ax', 'ay', 'az', '
                  'release_extension', 'release_pos_y', 'babip_value', 'woba_
                  'bat_score', 'fld_score', 'post_away_score', 'post_home_sco
                  'spin_axis', 'release_speed', 'events', 'description', 'zon
```

```
                    'pfx_x', 'pfx_z', 'plate_x', 'plate_z', 'hit_distance_sc',
                    'estimated_woba_using_speedangle', 'launch_speed_angle', '
```

In [5]:
```
df_reg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 717945 entries, 2437 to 4497
Data columns (total 34 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   release_pos_x          717673 non-null  Float64
 1   release_pos_z          717673 non-null  Float64
 2   player_name            717945 non-null  object 
 3   batter                 717945 non-null  Int64  
 4   pitcher                717945 non-null  Int64  
 5   stand                  717945 non-null  object 
 6   p_throws               717945 non-null  object 
 7   home_team              717945 non-null  object 
 8   away_team              717945 non-null  object 
 9   type                   717945 non-null  object 
 10  balls                  717945 non-null  Int64  
 11  strikes                717945 non-null  Int64  
 12  on_3b                  69419 non-null   Int64  
 13  on_2b                  137481 non-null  Int64  
 14  on_1b                  221293 non-null  Int64  
 15  outs_when_up           717945 non-null  Int64  
 16  inning                 717945 non-null  Int64  
 17  game_pk                717945 non-null  Int64  
 18  pitcher.1              717945 non-null  Int64  
 19  fielder_2.1            717945 non-null  Int64  
 20  fielder_3              717945 non-null  Int64  
 21  fielder_4              717945 non-null  Int64  
 22  fielder_5              717945 non-null  Int64  
 23  fielder_6              717945 non-null  Int64  
 24  fielder_7              717945 non-null  Int64  
 25  fielder_8              717945 non-null  Int64  
 26  fielder_9              717945 non-null  Int64  
 27  at_bat_number          717945 non-null  Int64  
 28  pitch_number           717945 non-null  Int64  
 29  pitch_name             717677 non-null  object 
 30  home_score             717945 non-null  Int64  
 31  away_score             717945 non-null  Int64  
 32  if_fielding_alignment  715432 non-null  object 
 33  of_fielding_alignment  715432 non-null  object 
dtypes: Float64(2), Int64(23), object(9)
memory usage: 208.8+ MB
```

In [6]:
```
#We use 14 pitchers that will also be playing in the postseason to filter ou

df_reg_filtered = df_reg[df_reg['pitcher'].isin([605400, 554430,624133,54313
df_post_filtered = df_post[df_post['pitcher'].isin([605400, 554430,624133,54
```

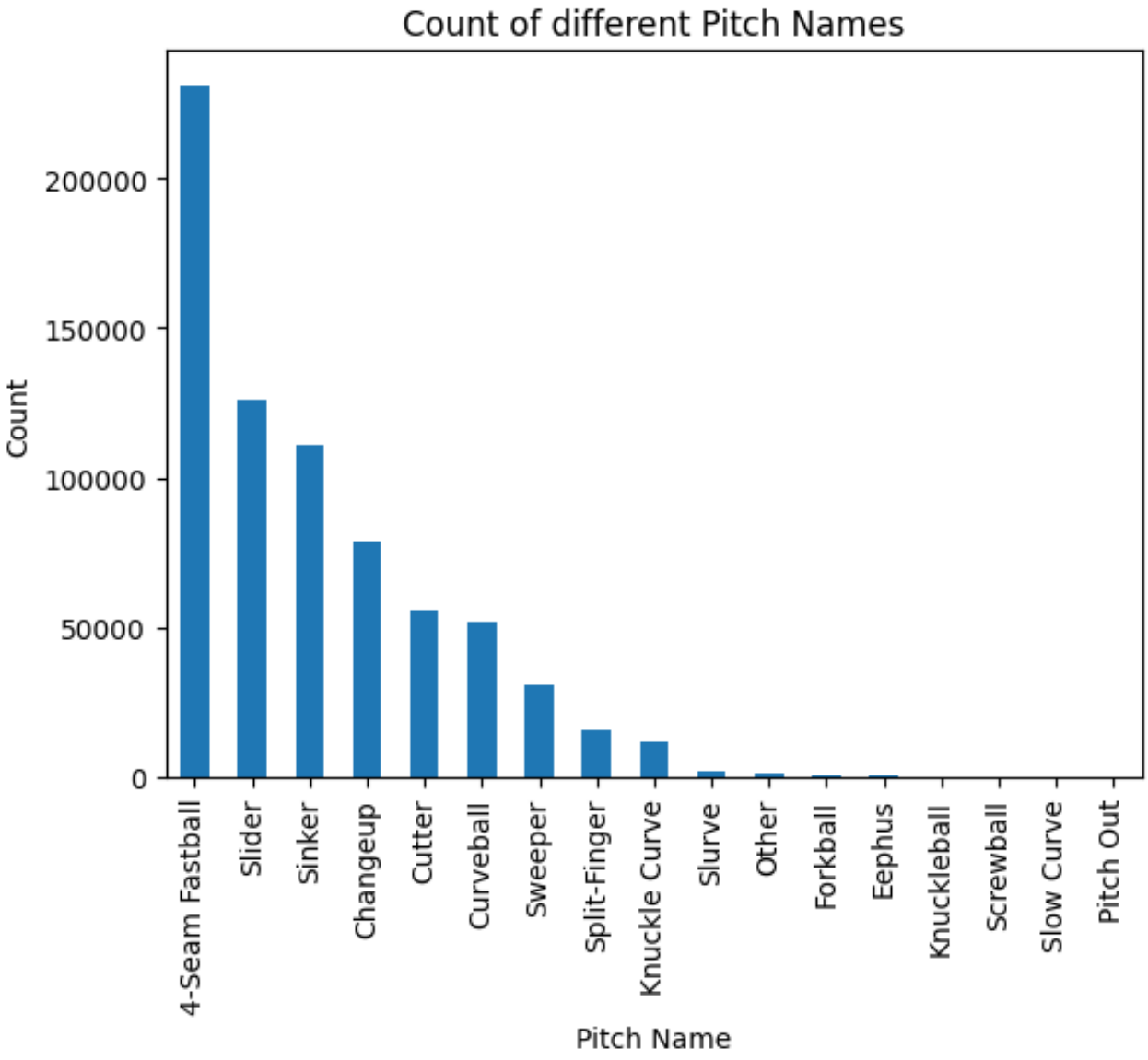
In [7]:
```
#Deleting two rows that had pitch_type as NaN which might cause an issue lat
```

```
df_reg_filtered = df_reg_filtered[~df_reg_filtered.pitch_name.isin([np.nan])
```

In [8]:
```
#First, we look at the different pitch types that these pitchers throw
pitch_name_counts = df_reg['pitch_name'].value_counts()
```

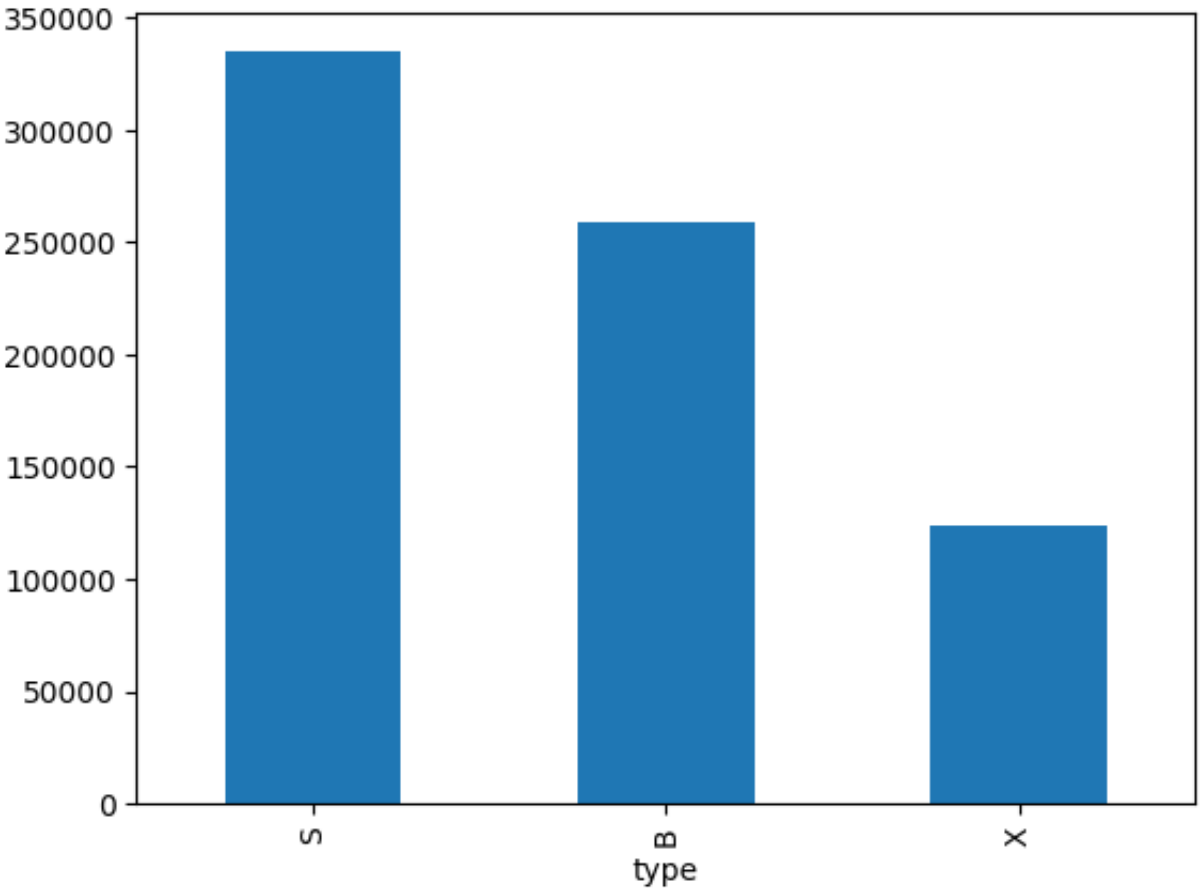
In [9]:
```
#We plot a count graph for the different types of pitches
pitch_name_counts.plot(kind="bar")
plot.title("Count of different Pitch Names")
plot.xlabel("Pitch Name")
plot.ylabel("Count")
```

Out[9]:
```
Text(0, 0.5, 'Count')
```



In [10]:
```
df_reg['type'].value_counts().plot(kind="bar")
```

Out[10]:
```
<Axes: xlabel='type'>
```



In [11]:
```
#Next, we divide the pitches for each individual pitcher

cross_tab_prop = pd.crosstab(index=df_reg_filtered['player_name'],
                             columns=df_reg_filtered['pitch_name'],
                             normalize="index")
cross_tab = pd.crosstab(index=df_reg_filtered['player_name'],
                        columns=df_reg_filtered['pitch_name'])
cross_tab
```

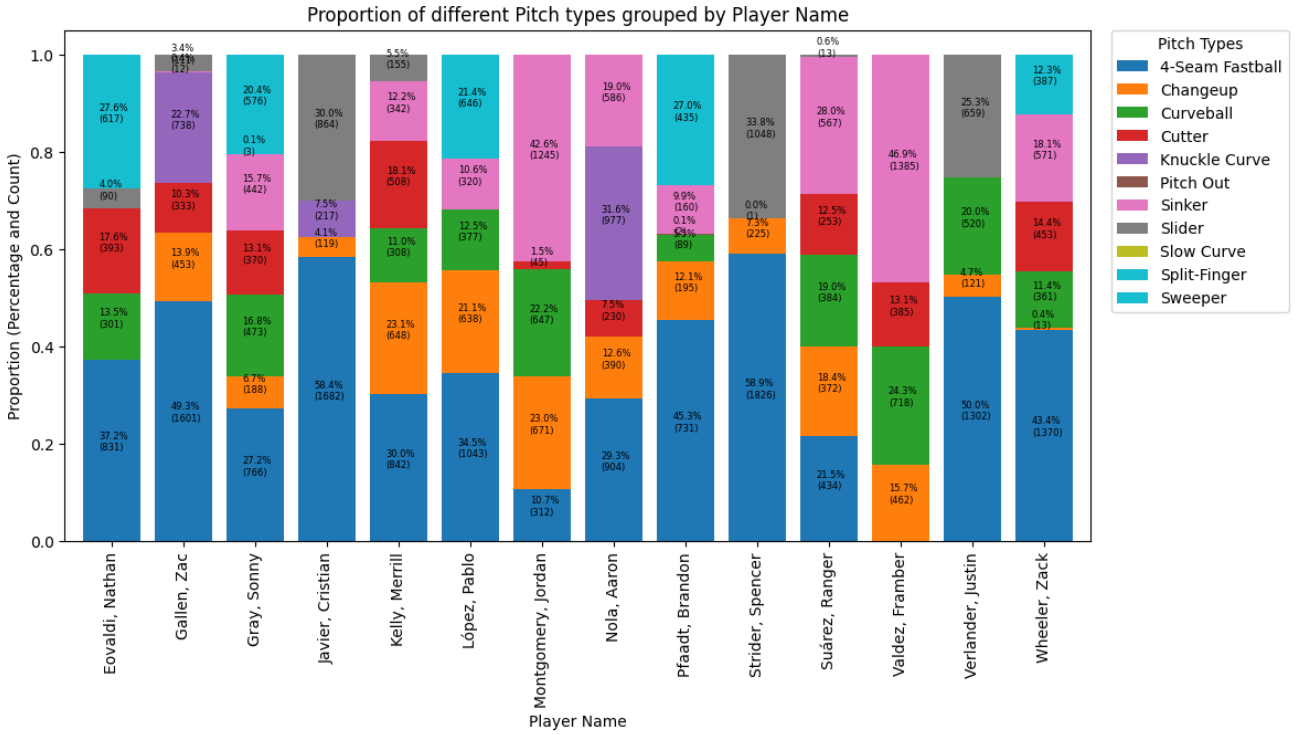
Out[11]:

pitch_name player_name	4-Seam Fastball	Changeup	Curveball	Cutter	Knuckle Curve	Pitch Out	Sinker	Slider	C
Eovaldi, Nathan	831	0	301	393	0	0	0	90	
Gallen, Zac	1601	453	0	333	738	0	12	111	
Gray, Sonny	766	188	473	370	0	0	442	0	
Javier, Cristian	1682	119	0	0	217	0	0	864	
Kelly, Merrill	842	648	308	508	0	0	342	155	
López, Pablo	1043	638	377	0	0	0	320	0	
Montgomery, Jordan	312	671	647	45	0	0	1245	0	
Nola, Aaron	904	390	0	230	977	0	586	0	
Pfaadt, Brandon	731	195	89	0	0	2	160	0	
Strider, Spencer	1826	225	0	0	0	1	0	1048	
Suárez, Ranger	434	372	384	253	0	0	567	13	
Valdez, Framber	0	462	718	385	0	0	1385	0	
Verlander, Justin	1302	121	520	0	0	0	0	659	
Wheeler, Zack	1370	13	361	453	0	0	571	0	

In [12]:

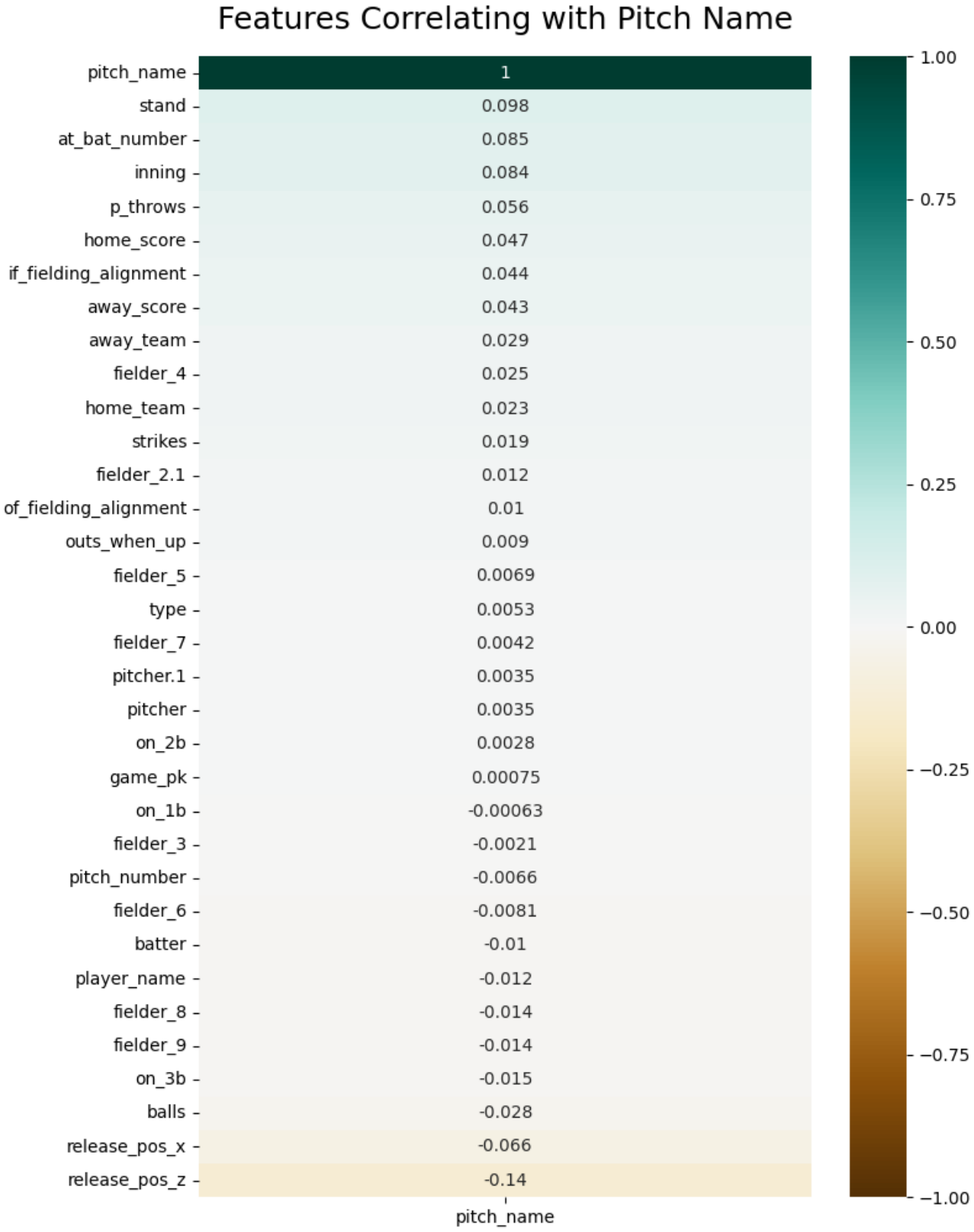
```
#Plotting the distribution of pitches for the 14 pitchers
cross_tab_prop.plot(kind='bar',
                    stacked=True,
                    colormap='tab10',
                    width=0.8,
                    figsize=(12, 6))

plot.title("Proportion of different Pitch types grouped by Player Name")
plot.xlabel("Player Name")
plot.ylabel("Proportion (Percentage and Count)")


for n, x in enumerate([*cross_tab.index.values]):
    for (proportion, count, y_loc) in zip(cross_tab_prop.loc[x],
```

```
                                        cross_tab.loc[x],
                                        cross_tab_prop.loc[x].cumsum()):
        if (proportion > 0):
            plot.text(x=n - 0.17,
                      y=(y_loc - proportion) + (proportion / 2),
                      s=f'{np.round(proportion * 100, 1)}%\n({count})',
                      color="black",
                      fontsize=6)
plot.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0, tit
plot.show()
```



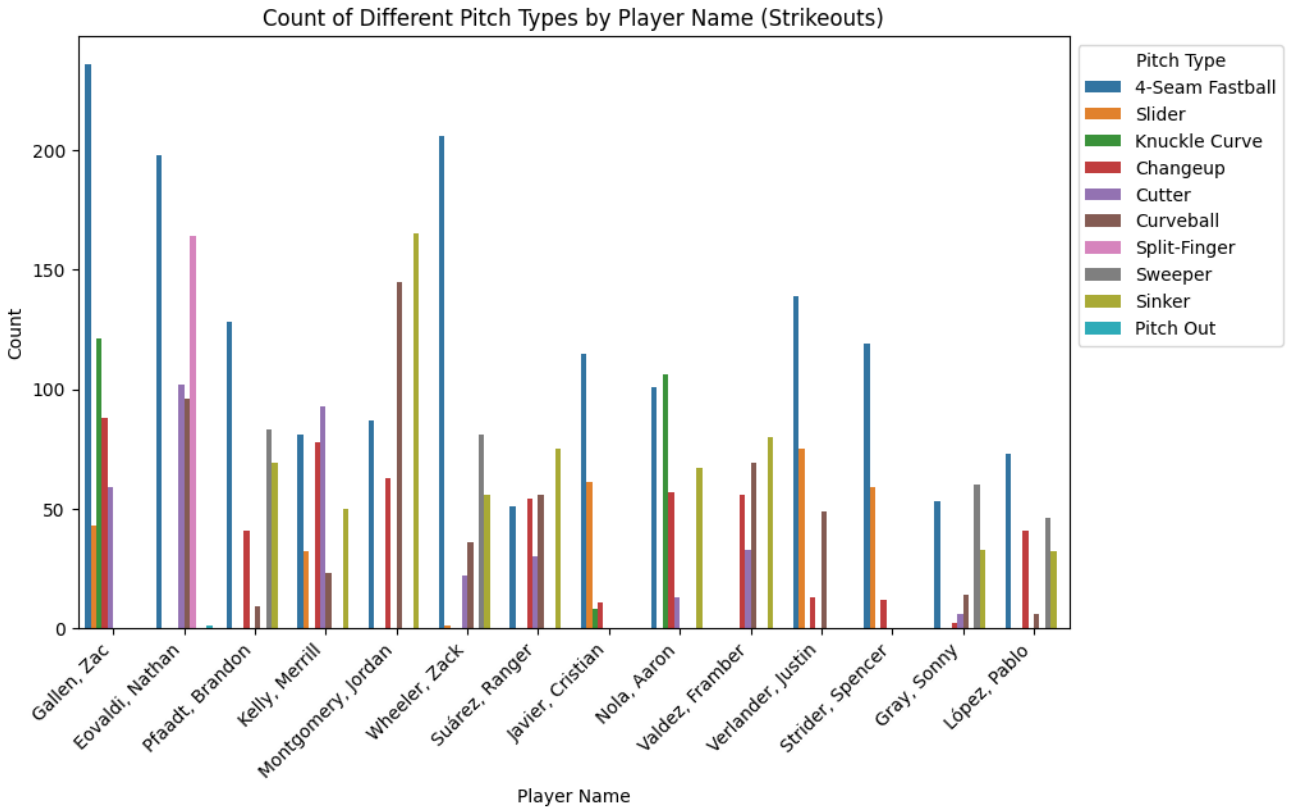
In [13]:
```
#We encode the data so that we can generate a heatmap for correlation of var
encoded_data = df_reg.copy()
le = LabelEncoder()
#Using Label Encoder, change all categorical data types to numerical
for col in ['pitch_name', 'player_name', 'stand', 'p_throws', 'home_team', '
            'type', 'pitch_name', 'if_fielding_alignment', 'of_fielding_ali
    encoded_data[col] = le.fit_transform(df_reg[col])
```

In [14]:
```
plot.figure(figsize=(8, 12))
heatmap = sns.heatmap(encoded_data.corr()[['pitch_name']].sort_values(by='pi
heatmap.set_title('Features Correlating with Pitch Name', fontdict={'fontsiz
```



In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
sns.countplot(x='player_name', hue='pitch_name', data=df_post_filtered)
plt.title('Count of Different Pitch Types by Player Name (Strikeouts)')
plt.xlabel('Player Name')
plt.ylabel('Count')
```

```
plt.legend(title='Pitch Type', bbox_to_anchor=(1, 1))
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [16]:
```
#Divide the data into training and testing sets (regular season for the form
X_train, y_train = df_reg_filtered.iloc[:,df_reg_filtered.columns!= 'pitch_r
X_test, y_test = df_post_filtered.iloc[:,df_post_filtered.columns!= 'pitch_r
```

In [17]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(38456, 33)
(38456, 1)
(4532, 33)
(4532, 1)
```

In [18]:
```
y_test
```

Out[18]:

	pitch_name
14	4-Seam Fastball
17	Slider
18	Slider
21	4-Seam Fastball
22	Knuckle Curve
...	...
987	Sinker
1048	4-Seam Fastball
1084	4-Seam Fastball
1119	4-Seam Fastball
1160	4-Seam Fastball

4532 rows × 1 columns

In [19]:
```
#Using Label Encoder, change all categorical data types to numerical
from sklearn.preprocessing import MinMaxScaler
le = LabelEncoder()

for col in ['player_name', 'stand', 'p_throws', 'home_team', 'away_team',
            'type', 'if_fielding_alignment', 'of_fielding_alignment']:
    X_train.loc[:,col] = le.fit_transform(X_train[col])
    X_test.loc[:,col] = le.fit_transform(X_test[col])
X_train.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_train[['rele
X_train.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_train[['rele
X_test.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_test[['releas
X_test.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_test[['releas

y_train = le.fit_transform(np.ravel(y_train))
y_test = le.fit_transform(np.ravel(y_test))
```

In [20]:
```
#Removing all nulls and filling them with 0. There are only 3 columns that c
X_train = X_train.fillna(0)
X_test = X_test.fillna(0)
```

In [21]:
```
#Now, we create a covariance matrix that will be used for PCA
covar_matrix_train = np.matmul(X_train.T , X_train)

print ( "The shape of variance matrix = ", covar_matrix_train.shape)
```

```
The shape of variance matrix =  (33, 33)
```

In [22]:
```
#all columns are object types by default which causes an error in the eigh f
covar_matrix_train = covar_matrix_train.apply(np.float64)
```

```
covar_matrix_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 33 entries, release_pos_x to of_fielding_alignment
Data columns (total 33 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   release_pos_x          33 non-null     float64
 1   release_pos_z          33 non-null     float64
 2   player_name            33 non-null     float64
 3   batter                 33 non-null     float64
 4   pitcher                33 non-null     float64
 5   stand                  33 non-null     float64
 6   p_throws               33 non-null     float64
 7   home_team              33 non-null     float64
 8   away_team              33 non-null     float64
 9   type                   33 non-null     float64
 10  balls                  33 non-null     float64
 11  strikes                33 non-null     float64
 12  on_3b                  33 non-null     float64
 13  on_2b                  33 non-null     float64
 14  on_1b                  33 non-null     float64
 15  outs_when_up           33 non-null     float64
 16  inning                 33 non-null     float64
 17  game_pk                33 non-null     float64
 18  pitcher.1              33 non-null     float64
 19  fielder_2.1            33 non-null     float64
 20  fielder_3              33 non-null     float64
 21  fielder_4              33 non-null     float64
 22  fielder_5              33 non-null     float64
 23  fielder_6              33 non-null     float64
 24  fielder_7              33 non-null     float64
 25  fielder_8              33 non-null     float64
 26  fielder_9              33 non-null     float64
 27  at_bat_number          33 non-null     float64
 28  pitch_number           33 non-null     float64
 29  home_score             33 non-null     float64
 30  away_score             33 non-null     float64
 31  if_fielding_alignment  33 non-null     float64
 32  of_fielding_alignment  33 non-null     float64
dtypes: float64(33)
memory usage: 9.8+ KB
```

In [23]:
```
from scipy.linalg import eigh

values_train, vectors_train = eigh(covar_matrix_train, subset_by_index=(31,3
print("Shape of eigen vectors = ",vectors_train.shape)
vectors_train = vectors_train.T
print("Updated shape of eigen vectors = ",vectors_train.shape)
```

```
Shape of eigen vectors =  (33, 2)
Updated shape of eigen vectors =  (2, 33)
```

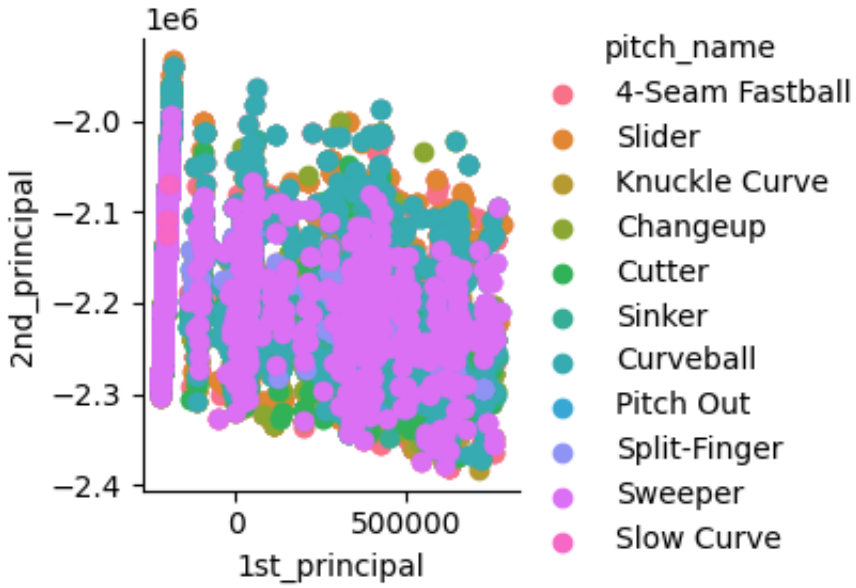
In [24]:
```
new_coordinates = np.matmul(vectors_train, X_train.T)
```

```
new_coordinates = np.vstack((new_coordinates, df_reg_filtered['pitch_name']))

# creating a new data frame for ploting the labeled points.
dataframe_pca = pd.DataFrame(data=new_coordinates, columns=("1st_principal",
print(dataframe_pca.head())
```

```
   1st_principal   2nd_principal       pitch_name
0  413464.045195 -2102242.806629  4-Seam Fastball
1  413464.045196 -2102242.806628  4-Seam Fastball
2  413464.045196 -2102242.806626  4-Seam Fastball
3  413464.045196 -2102242.806624  4-Seam Fastball
4  413464.045197 -2102242.806623  4-Seam Fastball
```

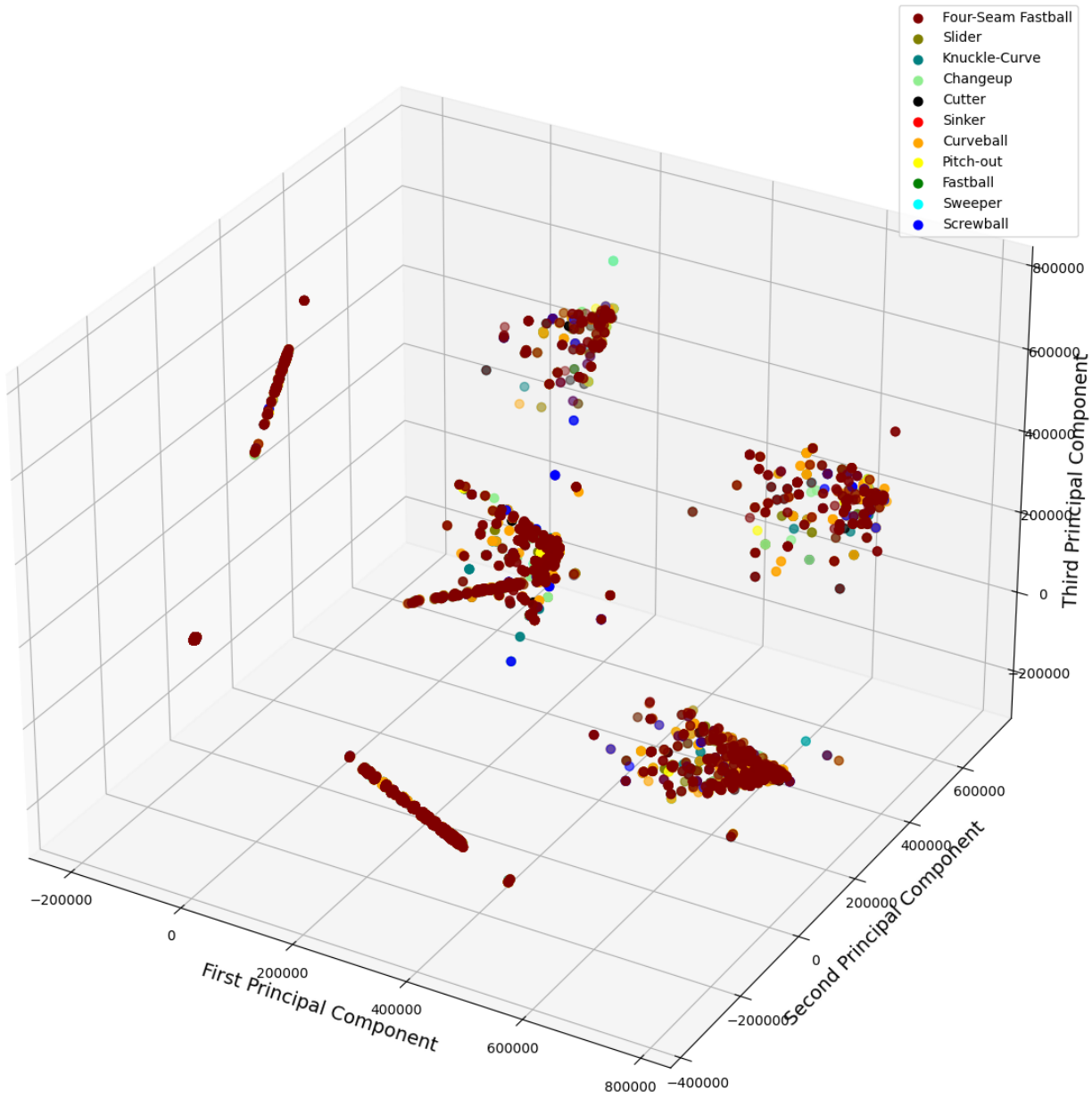
In [25]:
```
#2-D representation of the PCA

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(16, 16))
sns.FacetGrid(dataframe_pca, hue="pitch_name").map(plt.scatter, '1st_princip
plt.show()
```

```
<Figure size 1600x1600 with 0 Axes>
```



In [26]:
```
#3-D representation of the PCA

from sklearn.decomposition import PCA

pca = PCA(n_components=11)
pca.fit(X_train)
X_pca = pca.transform(X_train)

ex_variance=np.var(X_pca,axis=0)
ex_variance_ratio = ex_variance/np.sum(ex_variance)
ex_variance_ratio
```

```
Xax = X_pca[:,0]
Yax = X_pca[:,1]
Zax = X_pca[:,2]

cdict = {0:'maroon',1:'olive', 2:'teal',3:'lightgreen',4:'black', 5:'red',6:
labl = {0:'Four-Seam Fastball',1:'Slider', 2:'Knuckle-Curve',3:'Changeup',4:

fig = plt.figure(figsize=(15,15))
ax = fig.add_subplot(111, projection='3d')

fig.patch.set_facecolor('white')
for l in np.unique(y_train):
 ix=np.where(y_train==l)
 ax.scatter(Xax[ix], Yax[ix], Zax[ix], c=cdict[l], s=40, label=labl[l])
# for loop ends
ax.set_xlabel("First Principal Component", fontsize=14)
ax.set_ylabel("Second Principal Component", fontsize=14)
ax.set_zlabel("Third Principal Component", fontsize=14)

ax.legend()
plt.show()
```



In [27]:
```
from apyori import apriori
rules = apriori(transactions = np.array(df_reg_filtered.astype(str)), min_su
```

In [28]:
```
def inspect(output):
    Left_Hand_Side = [tuple(result[2][0][0])[0] for result in output]
    support = [result[1] for result in output]
    confidence = [result[2][0][2] for result in output]
    lift = [result[2][0][3] for result in output]
    Right_Hand_Side = [tuple(result[2][0][1])[0] for result in output]
    return list(zip(Left_Hand_Side, support, confidence, lift, Right_Hand_Si

output = list(rules)
output_data = pd.DataFrame(inspect(output), columns = ['Left_Hand_Side', 'Su
print(output_data)
```

	Left_Hand_Side	Support	Confidence	Lift	Right_Hand_Side
0	455117	0.107187	0.616236	5.560290	514888
1	455117	0.141746	0.814920	5.406930	547989
2	455117	0.171495	0.985947	5.399542	608324
3	455117	0.106797	0.613993	5.498770	643289
4	455117	0.161223	0.926895	5.456095	663656
...	...	...	...	...	...
5215	681082	0.113896	0.566404	4.088910	PHI
5216	681082	0.118525	0.589422	4.099622	PHI
5217	681082	0.189307	0.941420	4.126196	PHI
5218	681082	0.112986	0.561878	4.104001	PHI
5219	682998	0.180362	0.984388	4.443671	Standard

[5220 rows x 5 columns]

In [29]:
```
# importing the required module
from mlxtend.preprocessing import TransactionEncoder
# initializing the transactionEncoder
te = TransactionEncoder()
te_ary = te.fit(np.array(df_reg_filtered.astype(str))).transform(np.array(df
dataset = pd.DataFrame(te_ary, columns=te.columns_)
# dataset after encoded
dataset
```

Out[29]:

	-0.41	-0.42	-0.43	-0.45	-0.46	-0.48	-0.49	-0.51	-0.53	-0.54	...	Swe
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	False	False	False	False	False	False	...	
...	...	...	...	...	...	...	...	...	...	...	...	
38451	False	False	False	False	False	False	False	False	False	False	...	
38452	False	False	False	False	False	False	False	False	False	False	...	
38453	False	False	False	False	False	False	False	False	False	False	...	
38454	False	False	False	False	False	False	False	False	False	False	...	
38455	False	False	False	False	False	False	False	False	False	False	...	

38456 rows × 1902 columns

In [30]:
```
# importing the required module
from mlxtend.frequent_patterns import apriori, association_rules

# Extracting the most frequest itemsets via Mlxtend.
# The length column has been added to increase ease of filtering.
```

```
frequent_itemsets = apriori(dataset, min_support=0.1, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x:
# printing the frequent itemset
frequent_itemsets
```

Out[30]:

	support	itemsets	length
0	0.913226	(0)	1
1	0.914240	(1)	1
2	0.838153	(2)	1
3	0.509049	(3)	1
4	0.376742	(4)	1
...	...	...	...
15326	0.102299	(592663, 592206, 0, 607208, 681082, R, 664761,...	10
15327	0.103547	(665161, HOU, 1, 608324, <NA>, R, Standard, 66...	10
15328	0.101961	(592663, 592206, 607208, 1, 681082, R, Standar...	10
15329	0.103963	(665161, 663656, HOU, 1, 608324, R, Standard, ...	10
15330	0.102767	(592663, 592206, 607208, 1, 681082, R, 664761,...	10

15331 rows × 3 columns

In [31]:

```
frequent_itemsets[(frequent_itemsets['length'] >= 2) &
                  (frequent_itemsets['support'] >= 0.2) & (frequent_itemse
                  (frequent_itemsets['itemsets'].str.contains("4-Seam Fast
```

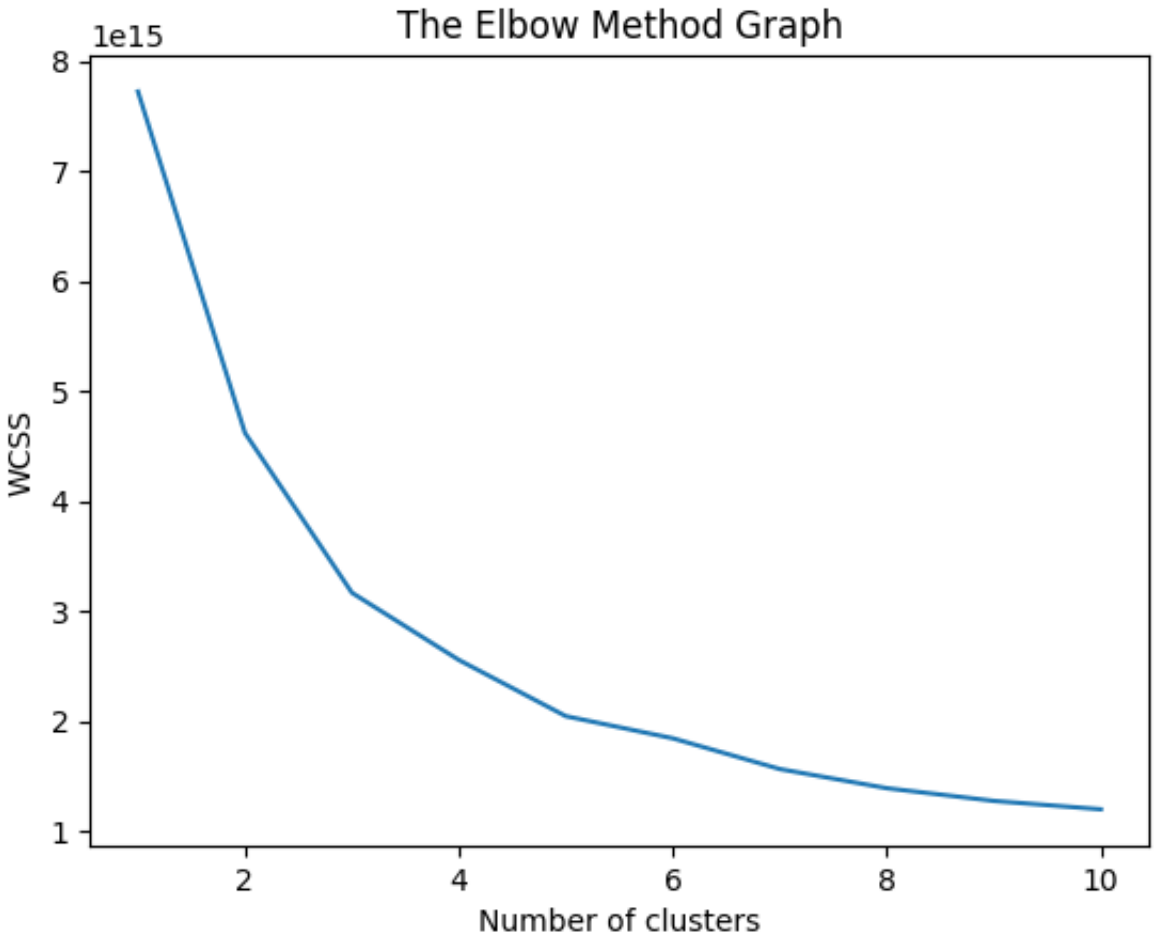
Out[31]:

	support	itemsets	length
55	0.325671	(4-Seam Fastball, 0)	2
99	0.323955	(4-Seam Fastball, 1)	2
143	0.290514	(2, 4-Seam Fastball)	2
202	0.349282	(4-Seam Fastball, <NA>)	2
205	0.353157	(4-Seam Fastball, R)	2
...	...	...	...
8538	0.230419	(0, 1, 4-Seam Fastball, R, Standard, 2)	6
8784	0.285495	(0, 1, 4-Seam Fastball, R, Standard, <NA>)	6
9261	0.253276	(0, 4-Seam Fastball, R, Standard, <NA>, 2)	6
10195	0.249818	(1, 4-Seam Fastball, R, Standard, <NA>, 2)	6
12010	0.226493	(0, 1, 4-Seam Fastball, R, Standard, <NA>, 2)	7

63 rows × 3 columns

In [ ]:
```
print(frequent_itemsets.nlargest(n = 5, columns = 'support'))
```

In [32]:
```
#Trying out the elbow curve to get the number of important features/clusters

from sklearn.cluster import KMeans

wcss=[]

for i in range(1,11):
    kmeans = KMeans(n_clusters=i, init ='k-means++', max_iter=300,  n_init=1
    kmeans.fit(X_train)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



MODELS

In [33]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

model = MultinomialNB()

model.fit(X_train,y_train)

y_train_pred = model.predict(X_train)
y_train_prob = model.predict_proba(X_train)
print("accuracy:", accuracy_score(y_train, y_train_pred))
```

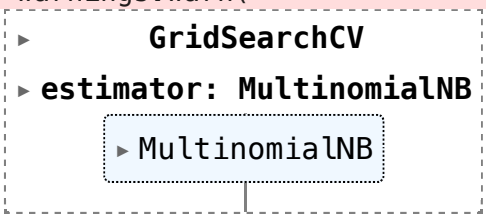
accuracy: 0.014042022051175369

In [34]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {'alpha': [0.0001, 0.001, 0.01, 0.1, 1.0]}
print("Parameter grid:\n{}".format(param_grid))

grid_search = GridSearchCV(MultinomialNB(), param_grid, cv=5,
                           return_train_score=True, scoring="accuracy")

grid_search.fit(X_train, y_train)
```

```
Parameter grid:
{'alpha': [0.0001, 0.001, 0.01, 0.1, 1.0]}
```

```
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/model_selection/_split.py:737: UserWarning:
The least populated class in y has only 3 members, which is less than n_spli
ts=5.
  warnings.warn(
```

Out[34]:



In [35]:

```
print("Best parameters: {}".format(grid_search.best_params_))
print("Best cross-validation score: {:.4f}".format(grid_search.best_score_))
```

```
Best parameters: {'alpha': 0.0001}
Best cross-validation score: 0.0141
```

In [36]:

```
import multiprocessing
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, precisi

nb = MultinomialNB(**grid_search.best_params_)
nb.fit(X_train, y_train)
test_score = nb.score(X_test, y_test)

y_pred = nb.predict(X_test)
y_pred_prob = nb.predict_proba(X_test)

print("Test accuracy:", accuracy_score(y_test, y_pred))
print("F1 score:", f1_score(y_test, y_pred, average='weighted'))
print("Precision score:", precision_score(y_test, y_pred, average='weighted'
print("Recall score:", recall_score(y_test, y_pred, average='weighted'))
```

```
Test accuracy: 0.02581641659311562
F1 score: 0.010299713480654222
Precision score: 0.03380975153576563
Recall score: 0.02581641659311562
```

```
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/metrics/_classification.py:1471: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/metrics/_classification.py:1471: UndefinedMe
tricWarning: Recall is ill-defined and being set to 0.0 in labels with no tr
ue samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [37]:

```
from sklearn.ensemble import RandomForestClassifier

param_grid = {
```

```
    'bootstrap': [True, False],
    'max_depth': [11,15,17,19],
    'min_samples_leaf': [5, 10, 20],
    'min_samples_split': [10, 15, 20]
}
scorers = {
    'accuracy_score': make_scorer(accuracy_score)
}

rf_vec_gs = GridSearchCV(estimator=RandomForestClassifier(), param_grid=para
        scoring=scorers, cv=10, n_jobs=multiprocessing.cpu_count(), verbose=

rf_vec_gs.fit(X_train, y_train)

rf_vec_best = rf_vec_gs.best_estimator_
rf_vec_best.fit(X_train, y_train)

print('Best model: %s' % str(rf_vec_gs.best_params_))
print('Best score: %f' % rf_vec_gs.best_score_)
print('Best test accuracy: %f' % rf_vec_best.score(X_test, y_test))
```

```
Fitting 10 folds for each of 72 candidates, totalling 720 fits
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/model_selection/_split.py:737: UserWarning:
The least populated class in y has only 3 members, which is less than n_spli
ts=10.
  warnings.warn(
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/joblib/externals/loky/process_executor.py:752: UserW
arning: A worker stopped while some jobs were given to the executor. This ca
n be caused by a too short worker timeout or by a memory leak.
  warnings.warn(
Best model: {'bootstrap': True, 'max_depth': 15, 'min_samples_leaf': 20, 'mi
n_samples_split': 15}
Best score: 0.486195
Best test accuracy: 0.442630
```

In [38]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {'n_neighbors': [2,4,7,11,15,18,20],
              'metric': ["euclidean", "manhattan", "minkowski"]}

knn_grid_search = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=
                           return_train_score=True, verbose=1, scoring='accur

knn_grid_search.fit(X_train, y_train)
print("Best parameters: {}".format(knn_grid_search.best_params_))
print("Best cross-validation score: {:.2f}".format(knn_grid_search.best_scor
```

```
Fitting 5 folds for each of 21 candidates, totalling 105 fits
```

```
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/model_selection/_split.py:737: UserWarning:
The least populated class in y has only 3 members, which is less than n_spli
ts=5.
  warnings.warn(
Best parameters: {'metric': 'manhattan', 'n_neighbors': 20}
Best cross-validation score: 0.30
```

In [39]:
```
optimal_knn_model = KNeighborsClassifier(**knn_grid_search.best_params_)
optimal_knn_model.fit(X_train, y_train)

y_pred = optimal_knn_model.predict(X_test)
```

In [40]:
```
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, precisi

print("Test accuracy:", accuracy_score(y_test, y_pred))
print("F1 score:", f1_score(y_test, y_pred, average='weighted'))
print("Precision score:", precision_score(y_test, y_pred, average='weighted'
print("Recall score:", recall_score(y_test, y_pred, average='weighted'))
```

```
Test accuracy: 0.3287731685789938
F1 score: 0.2701272552247107
Precision score: 0.26569524377909404
Recall score: 0.3287731685789938
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/metrics/_classification.py:1471: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/kishan/Documents/Documents/IST 707 Applied Machine Learning/venv/lib/
python3.9/site-packages/sklearn/metrics/_classification.py:1471: UndefinedMe
tricWarning: Recall is ill-defined and being set to 0.0 in labels with no tr
ue samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [42]:
```
import optuna
from sklearn.svm import SVC

def objective(trial):
    # Define the hyperparameters to search
    C = trial.suggest_float('C', 1e-4, 1e4)
    gamma = trial.suggest_float('gamma', 1e-4, 1e4)
    kernel = trial.suggest_categorical('kernel', ['rbf'])

    # Create the SVM classifier with the suggested hyperparameters
    clf = SVC(C=C, gamma=gamma, kernel=kernel)

    # Train the classifier and evaluate on the validation set
    clf.fit(X_train, y_train)
    accuracy = clf.score(X_test, y_test)

    return accuracy
```

```
# Run the optimization
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=10)

# Print the best hyperparameters and the corresponding accuracy
best_params = study.best_params
best_accuracy = study.best_value
print('Best Hyperparameters:', best_params)
print('Best Accuracy:', best_accuracy)
```

```
[I 2023-12-03 22:27:26,837] A new study created in memory with name: no-name
-a0d3ebd5-153c-4932-9d81-775dd0484a67
[I 2023-12-03 22:32:29,282] Trial 0 finished with value: 0.35017652250661957
and parameters: {'C': 9178.370146770083, 'gamma': 4423.787399043729, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 22:37:36,796] Trial 1 finished with value: 0.35017652250661957
and parameters: {'C': 8887.906518984144, 'gamma': 9781.59198639811, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 22:42:39,127] Trial 2 finished with value: 0.35017652250661957
and parameters: {'C': 3738.4275754521464, 'gamma': 8313.346516779042, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 22:47:41,349] Trial 3 finished with value: 0.35017652250661957
and parameters: {'C': 1624.1860998197994, 'gamma': 4212.212615812341, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 22:52:43,277] Trial 4 finished with value: 0.35017652250661957
and parameters: {'C': 618.391921643613, 'gamma': 8542.15187245932, 'kernel':
'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 22:57:43,648] Trial 5 finished with value: 0.35017652250661957
and parameters: {'C': 6730.700106298582, 'gamma': 4222.483566343328, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 23:02:44,250] Trial 6 finished with value: 0.35017652250661957
and parameters: {'C': 1411.6483741007348, 'gamma': 6280.727288509174, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 23:07:44,788] Trial 7 finished with value: 0.35017652250661957
and parameters: {'C': 5776.239048814928, 'gamma': 9505.940407374144, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 23:12:45,573] Trial 8 finished with value: 0.35017652250661957
and parameters: {'C': 6005.036674308265, 'gamma': 3566.0572289026345, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
[I 2023-12-03 23:17:45,960] Trial 9 finished with value: 0.35017652250661957
and parameters: {'C': 9931.256496019494, 'gamma': 2799.1959978800214, 'kerne
l': 'rbf'}. Best is trial 0 with value: 0.35017652250661957.
Best Hyperparameters: {'C': 9178.370146770083, 'gamma': 4423.787399043729, '
kernel': 'rbf'}
Best Accuracy: 0.35017652250661957
```

INDIVIDUAL PITCHERS

In [126…

```
nola_reg_filtered = df_reg_filtered[df_reg_filtered['pitcher'].isin([605400]
nola_post_filtered = df_post_filtered[df_post_filtered['pitcher'].isin([6054

gallen_reg_filtered = df_reg_filtered[df_reg_filtered['pitcher'].isin([66867
gallen_post_filtered = df_post_filtered[df_post_filtered['pitcher'].isin([66
```

```
montgomery_reg_filtered = df_reg_filtered[df_reg_filtered['pitcher'].isin([6
montgomery_post_filtered = df_post_filtered[df_post_filtered['pitcher'].isir
```

In [127…

```
X_train_nola, y_train_nola = nola_reg_filtered.iloc[:,nola_reg_filtered.colu
X_test_nola, y_test_nola = nola_post_filtered.iloc[:,nola_post_filtered.colu

X_train_gallen, y_train_gallen = gallen_reg_filtered.iloc[:,gallen_reg_filte
X_test_gallen, y_test_gallen = gallen_post_filtered.iloc[:,gallen_post_filte

X_train_montgomery, y_train_montgomery = montgomery_reg_filtered.iloc[:,mont
X_test_montgomery, y_test_montgomery = montgomery_post_filtered.iloc[:,montg
```

In [128…

```
#Using Label Encoder, change all categorical data types to numerical
from sklearn.preprocessing import MinMaxScaler
le = LabelEncoder()

for col in ['player_name', 'stand', 'p_throws', 'home_team', 'away_team',
            'type', 'if_fielding_alignment', 'of_fielding_alignment']:
    X_train_nola.loc[:,col] = le.fit_transform(X_train_nola[col])
    X_test_nola.loc[:,col] = le.fit_transform(X_test_nola[col])

    X_train_gallen.loc[:,col] = le.fit_transform(X_train_gallen[col])
    X_test_gallen.loc[:,col] = le.fit_transform(X_test_gallen[col])

    X_train_montgomery.loc[:,col] = le.fit_transform(X_train_montgomery[col]
    X_test_montgomery.loc[:,col] = le.fit_transform(X_test_montgomery[col])

X_train_nola.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_train_r
X_train_nola.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_train_r
X_test_nola.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_test_nol
X_test_nola.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_test_nol
X_train_nola = X_train_nola.fillna(0)
X_test_nola = X_test_nola.fillna(0)
y_train_nola = le.fit_transform(np.ravel(y_train_nola))

X_train_gallen.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_trair
X_train_gallen.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_trair
X_test_gallen.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_test_g
X_test_gallen.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_test_g
X_train_gallen = X_train_gallen.fillna(0)
X_test_gallen = X_test_gallen.fillna(0)
y_train_gallen = le.fit_transform(np.ravel(y_train_gallen))

X_train_montgomery.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_t
X_train_montgomery.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_t
X_test_montgomery.loc[:,'release_pos_x'] = MinMaxScaler().fit_transform(X_te
X_test_montgomery.loc[:,'release_pos_z'] = MinMaxScaler().fit_transform(X_te
X_train_montgomery = X_train_montgomery.fillna(0)
X_test_montgomery = X_test_montgomery.fillna(0)
y_train_montgomery = le.fit_transform(np.ravel(y_train_montgomery))
```

In [143…

```
X_test_nola.iloc[2]
```

Out[143…

```
release_pos_x            0.724638
release_pos_z            0.673077
player_name                   0.0
batter                   682998.0
pitcher                  605400.0
stand                         0.0
p_throws                      0.0
home_team                     0.0
away_team                     1.0
type                          2.0
balls                         2.0
strikes                       2.0
on_3b                         0.0
on_2b                         0.0
on_1b                         0.0
outs_when_up                  1.0
inning                        5.0
game_pk                  748539.0
pitcher.1                605400.0
fielder_2.1              592663.0
fielder_3                547180.0
fielder_4                681082.0
fielder_5                664761.0
fielder_6                607208.0
fielder_7                669016.0
fielder_8                679032.0
fielder_9                592206.0
at_bat_number                37.0
pitch_number                  8.0
home_score                    1.0
away_score                    3.0
if_fielding_alignment         1.0
of_fielding_alignment         0.0
Name: 12, dtype: Float64
```

In [139…

```
#Trying to predict a pitch type based on some input
pitch_to_predict = X_test_nola.iloc[2]

rf_nola = rf_vec_best.fit(X_train_nola.values, y_train_nola)
print("Nola: %s" % (df_reg_filtered['pitch_name'].unique()[rf_nola.predict([

rf_gallen = rf_vec_best.fit(X_train_gallen.values, y_train_gallen)
print("Gallen: %s" % (df_reg_filtered['pitch_name'].unique()[rf_gallen.predi

rf_montgomery = rf_vec_best.fit(X_train_montgomery.values, y_train_montgomer
print("Montgomery: %s" % (df_reg_filtered['pitch_name'].unique()[rf_montgome
```

```
Nola: Changeup
Gallen: 4-Seam Fastball
Montgomery: Cutter
```