Import required libraries, fetch data from database into pandas.DataFrame and format data type for PCA

In [5]:
```
import numpy as np
import sqlite3
import matplotlib.pylab as plt
import datetime as DT
import seaborn as sns
np.set_printoptions(precision=5)
import pandas as pd  # data processing, CSV file I/O (e.g. pd.read_csv)
#### import the PCA library from scikit learn library
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D  # didn't get to use this
%matplotlib inline

# Normalizing (scaling) the data is VERY important - indeed can be important to many machine
# learning algorithms.  Take the original features and scale them so that they all have zero
# mean and unit variance
from sklearn import preprocessing

## pandas actually has a command to read_sql or read_sql_query and return a pandas.DataFrame
## coerce_float=True argument to force float data type
## Need to FIRST connect to the database by creating a connection object called conn.
conn = sqlite3.connect('database.sqlite')

# create a cursor object using the conn object method
# The cursor object has methods for accessing the data
# c = conn.cursor()  # this is not needed for the pd.read_sql IO tool

# Get the database table list from information in the sqlite_master table
# Follow convention to type SQL commands in all caps

# preview all tables in the database
print ('======')
print ('Tables in the database')
# Set the execute SQL command, Fetch and print all table names and info, and
# return a pandas DataFrame
df_tables = pd.read_sql("""SELECT * FROM sqlite_master WHERE type='table';""", conn)
print('df_tables shape: ', df_tables.shape)
print(df_tables)
print ('======')
print('Player_Attributes table:')
print(df_tables.sql[1]) # get sql that CREATE the Player_Atrribtues table
print ('======')
print('Player table:')
print(df_tables.sql[2]) # get sql that CREATE the Player table
print ('======')
print ('Player table')
df_Player = pd.read_sql("""SELECT * FROM Player """, conn)
print('df_Player.shape:', df_Player.shape)
print(df_Player.columns)
print(df_Player.head())
print ('======')
print ('Player_Attributes table')
df_Player_Attributes = pd.read_sql("""SELECT * FROM Player_Attributes""", conn)
print('df_Player_Attributes.shape:', df_Player_Attributes.shape)
print(df_Player_Attributes.columns)
print(df_Player_Attributes.head())
print ('======')

# acquire data from database using pd.read_sql_query(sql, , ,)
# build SQL to SELECT all columns from both Player and Player_Attributes tables
# for rows reocrds w/ matching player_fifa_api_id
sql="SELECT * FROM Player INNER JOIN Player_Attributes ON Player.player_fifa_api_id=Player_A
ttributes.player_fifa_api_id;"
df_all_col=pd.read_sql_query(sql, conn, coerce_float=True, params=None, parse_dates=['birthd
ay','date'], chunksize=None)
# calculate age of player at the time attributes were collected
df_all_col['age'] = (df_all_col.date - df_all_col.birthday).astype('timedelta64[Y]')

#Tally total score per player attribute category
df_all_col['total_attack'] = df_all_col.crossing + df_all_col.finishing + df_all_col.heading
 accuracy + \
```

```
======
Tables in the database
df_tables shape:  (8, 5)
    type               name           tbl_name  rootpage  \
0  table    sqlite_sequence    sqlite_sequence         4
1  table  Player_Attributes  Player_Attributes        11
2  table             Player             Player        14
3  table              Match              Match        18
4  table             League             League        24
5  table            Country            Country        26
6  table               Team               Team        29
7  table    Team_Attributes    Team_Attributes         2

                                                 sql
0             CREATE TABLE sqlite_sequence(name,seq)
1  CREATE TABLE "Player_Attributes" (\n\t`id`\tIN...
2  CREATE TABLE `Player` (\n\t`id`\tINTEGER PRIMA...
3  CREATE TABLE `Match` (\n\t`id`\tINTEGER PRIMAR...
4  CREATE TABLE `League` (\n\t`id`\tINTEGER PRIMA...
5  CREATE TABLE `Country` (\n\t`id`\tINTEGER PRIM...
6  CREATE TABLE "Team" (\n\t`id`\tINTEGER PRIMARY...
7  CREATE TABLE `Team_Attributes` (\n\t`id`\tINTE...
======
Player_Attributes table:
CREATE TABLE "Player_Attributes" (
        `id`    INTEGER PRIMARY KEY AUTOINCREMENT,
        `player_fifa_api_id`    INTEGER,
        `player_api_id` INTEGER,
        `date`  TEXT,
        `overall_rating`        INTEGER,
        `potential`     INTEGER,
        `preferred_foot`        TEXT,
        `attacking_work_rate`   TEXT,
        `defensive_work_rate`   TEXT,
        `crossing`      INTEGER,
        `finishing`     INTEGER,
        `heading_accuracy`      INTEGER,
        `short_passing` INTEGER,
        `volleys`       INTEGER,
        `dribbling`     INTEGER,
        `curve` INTEGER,
        `free_kick_accuracy`    INTEGER,
        `long_passing`  INTEGER,
        `ball_control`  INTEGER,
        `acceleration`  INTEGER,
        `sprint_speed`  INTEGER,
        `agility`       INTEGER,
        `reactions`     INTEGER,
        `balance`       INTEGER,
        `shot_power`    INTEGER,
        `jumping`       INTEGER,
        `stamina`       INTEGER,
        `strength`      INTEGER,
        `long_shots`    INTEGER,
        `aggression`    INTEGER,
        `interceptions` INTEGER,
        `positioning`   INTEGER,
        `vision`        INTEGER,
        `penalties`     INTEGER,
        `marking`       INTEGER,
        `standing_tackle`       INTEGER,
        `sliding_tackle`        INTEGER,
        `gk_diving`     INTEGER,
        `gk_handling`   INTEGER,
        `gk_kicking`    INTEGER,
        `gk_positioning`        INTEGER,
        `gk_reflexes`   INTEGER,
        FOREIGN KEY(`player_fifa_api_id`) REFERENCES `Player`(`player_fifa_api_id`),
        FOREIGN KEY(`player_api_id`) REFERENCES `Player`(`player_api_id`)
)
```

data cleaning

In [6]:
```
df_all_col.replace(r'\s+', np.nan, regex=True, inplace = True)
df_all_col.dropna(axis=0, how='any', inplace=True)  #drop row (sample) with any NA entry
df_all_col.sort_values('player_name',axis=0, inplace=True)
df_all_col.drop_duplicates(inplace=True)
df_all_col.to_csv('df_all_col.csv')

print ('df_all_col.shape: ', df_all_col.shape)
print(df_all_col.shape)
print(df_all_col.info())
print(df_all_col.head())
print(df_all_col.tail())
print(df_all_col['defensive_work_rate'][0:60]) # need more data cleaning for col before plot
ing
print ('======')

df_unscaled_data = df_all_col[numeric_col]
print('df_unscaled_data.columns:', df_unscaled_data.columns)
print('df_unscaled_data.shape:', df_unscaled_data.shape)
print('df_unscaled_data.info: ', df_unscaled_data.info())
```

```
df_all_col.shape:  (10898, 57)
(10898, 57)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10898 entries, 1045 to 183615
Data columns (total 57 columns):
id                     10898 non-null int64
player_api_id          10898 non-null int64
player_name            10898 non-null object
player_fifa_api_id     10898 non-null int64
birthday               10898 non-null datetime64[ns]
height                 10898 non-null float64
weight                 10898 non-null int64
id                     10898 non-null int64
player_fifa_api_id     10898 non-null int64
player_api_id          10898 non-null int64
date                   10898 non-null datetime64[ns]
overall_rating         10898 non-null float64
potential              10898 non-null float64
preferred_foot         10898 non-null object
attacking_work_rate    10898 non-null object
defensive_work_rate    10898 non-null object
crossing               10898 non-null float64
finishing              10898 non-null float64
heading_accuracy       10898 non-null float64
short_passing          10898 non-null float64
volleys                10898 non-null float64
dribbling              10898 non-null float64
curve                  10898 non-null float64
free_kick_accuracy     10898 non-null float64
long_passing           10898 non-null float64
ball_control           10898 non-null float64
acceleration           10898 non-null float64
sprint_speed           10898 non-null float64
agility                10898 non-null float64
reactions              10898 non-null float64
balance                10898 non-null float64
shot_power             10898 non-null float64
jumping                10898 non-null float64
stamina                10898 non-null float64
strength               10898 non-null float64
long_shots             10898 non-null float64
aggression             10898 non-null float64
interceptions          10898 non-null float64
positioning            10898 non-null float64
vision                 10898 non-null float64
penalties              10898 non-null float64
marking                10898 non-null float64
standing_tackle        10898 non-null float64
sliding_tackle         10898 non-null float64
gk_diving              10898 non-null float64
gk_handling            10898 non-null float64
gk_kicking             10898 non-null float64
gk_positioning         10898 non-null float64
gk_reflexes            10898 non-null float64
age                    10898 non-null float64
total_attack           10898 non-null float64
total_skill            10898 non-null float64
total_movement         10898 non-null float64
total_power            10898 non-null float64
total_mentality        10898 non-null float64
total_defending        10898 non-null float64
total_goalkeeping      10898 non-null float64
dtypes: datetime64[ns](2), float64(44), int64(7), object(4)
memory usage: 4.8+ MB
None
      id  player_api_id player_name  player_fifa_api_id   birthday  height  \
1045  67          40938        Abel               17880 1978-12-22   177.8
1046  67          40938        Abel               17880 1978-12-22   177.8
1047  67          40938        Abel               17880 1978-12-22   177.8
1048  67          40938        Abel               17880 1978-12-22   177.8
```

Preprocess data and conduct PCA - principal component analysis

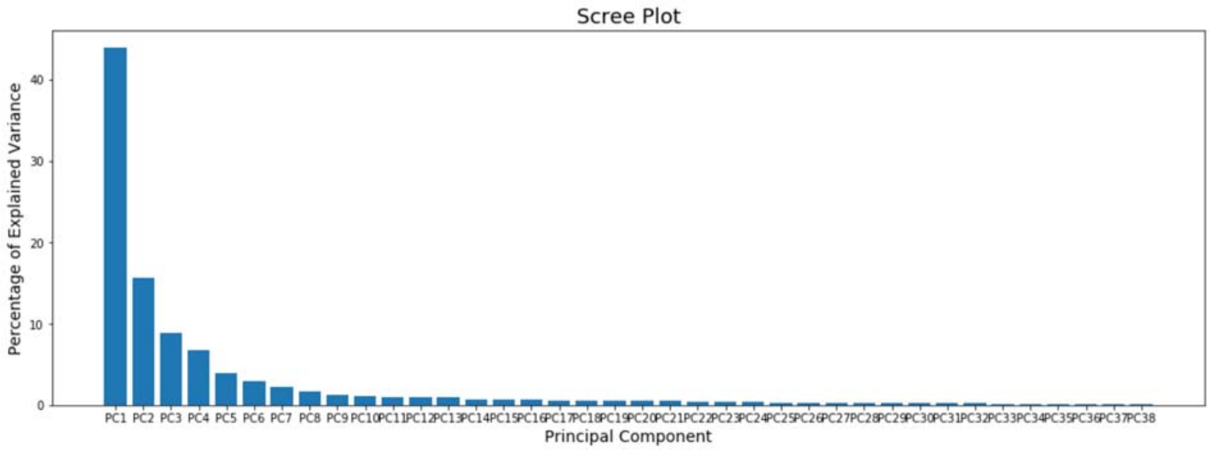
In [9]:
```
scaled_data = preprocessing.scale(df_unscaled_data) #center and scale the data
print('scaled data:')
print (scaled_data)  # preview scaled data

# create a PCA object.
# sklean uses this PCA object that can be trained using one dataset and applied to another d
ataset
pca = PCA()
print(type(pca))
# do PCA math, calculate loading scores and the variation each PCA accounts for
pca.fit(scaled_data)
# generate coordinates for a PCA graph based on the loading scores and the scaled data
pca_data = pca.transform(scaled_data)
```

```
scaled data:
[[ 1.20435 -0.50782 -0.12955 ... -0.64338 -0.19821 -0.36672]
 [ 1.20435 -0.50782 -0.12955 ...  2.36822  0.20348  0.18149]
 [ 0.98217 -0.50782 -0.12955 ...  2.36822  0.20348  0.18149]
 ...
 [-1.68402 -0.88666 -1.24685 ... -0.64338 -0.31297 -0.47637]
 [-1.68402 -0.88666 -1.24685 ... -0.64338 -0.31297 -0.47637]
 [-3.2393  -0.88666 -1.24685 ... -0.64338 -0.31297 -0.47637]]
<class 'sklearn.decomposition.pca.PCA'>
```

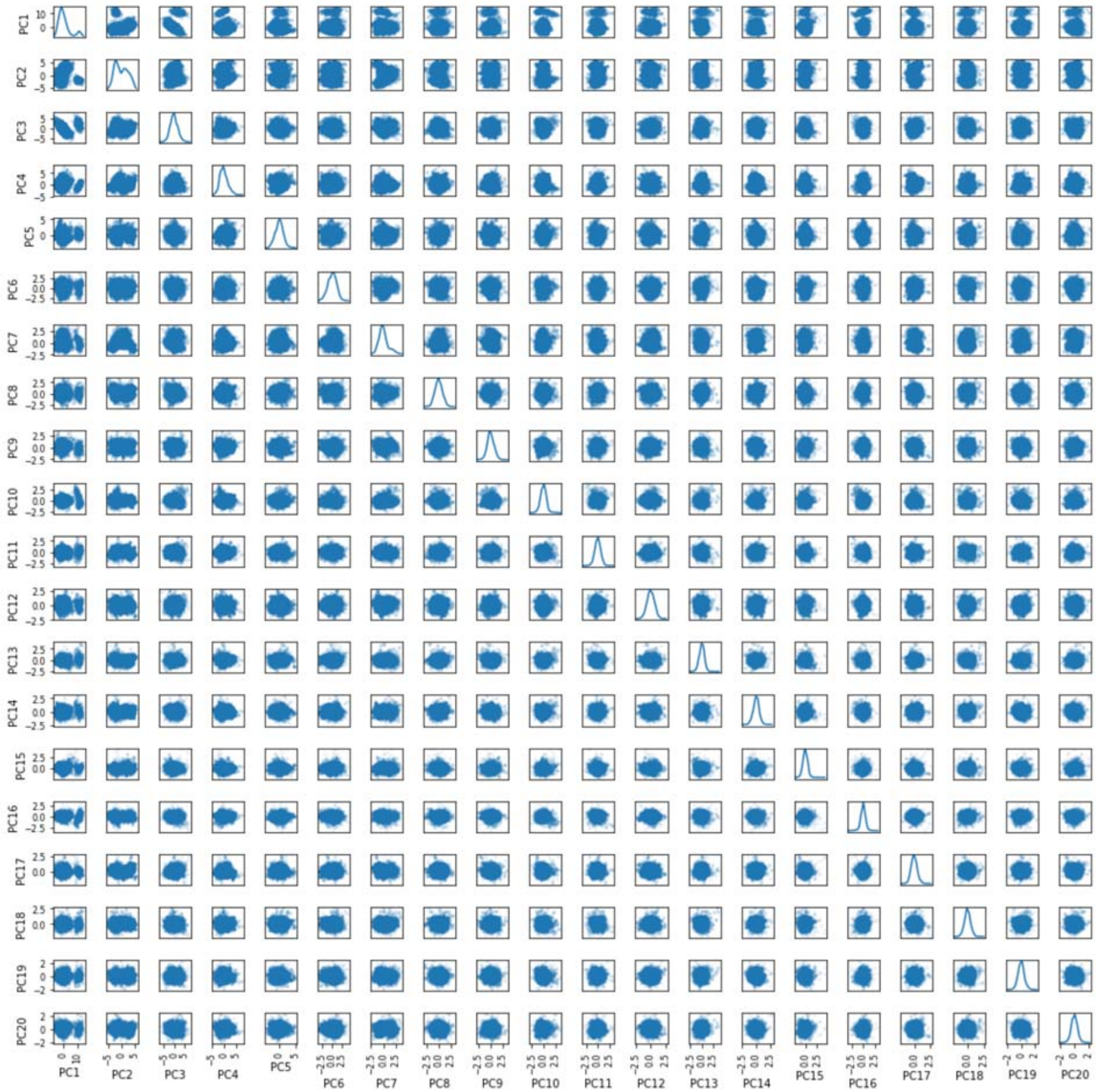
Present Explained Variance, Scree Plot and Principal Components Scatter Matrix

In [11]:
```
# pca.explained_variance_ratio_ is <class 'numpy.ndarray'>.
# It calculates the percentage of variance that each principal component accoutns for
per_var = np.round(pca.explained_variance_ratio_*100, decimals =1)
print('======================')
print('percent of explained variance: ')
print(per_var)
PC_labels = ['PC'+ str(x) for x in range(1,len(per_var)+1)] # labels for the Scree Plot: PC1
, PC2 ...
print(' ')
print('======================')

# create Scree Plot
plt.figure(figsize=(18, 6))
plt.bar(x=range(1, len(per_var)+1), height=per_var, tick_label=PC_labels)
plt.ylabel('Percentage of Explained Variance', fontsize='14')
plt.xlabel('Principal Component', fontsize='14')
plt.title('Scree Plot', fontsize='18')
plt.show()
plt.close()

# put pca_data with DataFrame with PC_labels
pca_df = pd.DataFrame(pca_data, index=None, columns=PC_labels)
print(pca_df.head()) # preview transformed and scaled
print('======================')


print('Principal Components Scatter Matrix')
df_pc_matrix= pca_df[['PC'+ str(x) for x in range(1,21)]] # scatter matrix for PC1, PC2, ...
, PC15
pd.plotting.scatter_matrix(df_pc_matrix, alpha=0.1, figsize=(14, 14), diagonal='kde',range_p
adding =0.1)
plt.tight_layout()
plt.show()
plt.close()
```

```
=======================
percent of explained variance:
[43.9 15.7  8.9  6.8  3.9  3.   2.2  1.7  1.3  1.1  1.   1.   0.9  0.7
  0.7  0.7  0.6  0.6  0.5  0.5  0.5  0.4  0.4  0.4  0.3  0.3  0.3  0.3
  0.2  0.2  0.2  0.2  0.1  0.1  0.1  0.1  0.1  0.1]

=======================
```




```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0 -2.640114  1.968095  0.650296 -1.085030  0.432893 -0.997913 -0.036752
1 -1.974144  1.547071  1.945571 -1.698880  0.785256 -0.431110  2.178955
2 -1.998243  1.632706  1.964671 -1.818000  0.655679 -0.269227  2.168876
3 -1.777293  1.508709  1.666837 -1.760435  0.469343 -0.019567  2.297794
4 -2.056297  0.869395  0.466741 -1.179464  0.879790 -0.124301  2.485314

        PC8       PC9      PC10    ...         PC29      PC30      PC31  \
0 -0.345769 -0.212905  0.717640    ...    -0.243024 -0.082784 -0.287740
1 -0.524871  0.683914  0.138753    ...    -0.313755 -0.136859 -0.142348
2 -0.553466  0.642269  0.157086    ...    -0.304040  0.008732 -0.030220
3 -0.364947  0.543994  0.246732    ...    -0.335973  0.075856  0.170031
4  0.184533  0.201187 -0.439294    ...    -0.316171  0.099965  0.113237

       PC32      PC33      PC34      PC35      PC36      PC37      PC38
0 -0.044248 -0.119196 -0.127400 -0.024028  0.054763  0.303422  0.202196
1 -0.096422 -0.082920  0.120610  0.091425 -0.042255  0.005217 -0.040199
2 -0.022378 -0.068867  0.080547  0.128276  0.035120  0.007889 -0.050469
3  0.044900 -0.031943  0.052600  0.120342  0.048383  0.012409 -0.034910
4  0.045786 -0.056746  0.502708  0.322174 -0.167724 -0.178028 -0.046963

[5 rows x 38 columns]
=======================
Principal Components Scatter Matrix
```



Take a Look at the Loading Scores of PC1, PC2 and PC3 to determine which feature has the largest influence in each principal component

In [12]:
```
loading_scores_PC1 = pd.Series(pca.components_[0],index=numeric_col)
loading_scores_PC1_sorted = loading_scores_PC1.abs().sort_values(ascending=False)
print('Sorted PC1 Loading Scores (abs)')
print('PC1 sorted components: ', loading_scores_PC1_sorted.index)
print(loading_scores_PC1_sorted)
print('=======')

loading_scores_PC2 = pd.Series(pca.components_[1],index=numeric_col)
loading_scores_PC2_sorted = loading_scores_PC2.abs().sort_values(ascending=False)
print('Sorted PC2 Loading Scores (abs)')
print('PC2 sorted components: ', loading_scores_PC2_sorted.index)
print(loading_scores_PC2_sorted)
print('=======')

loading_scores_PC3 = pd.Series(pca.components_[2],index=numeric_col)
loading_scores_PC3_sorted = loading_scores_PC3.abs().sort_values(ascending=False)
print('Sorted PC3 Loading Scores (abs)')
print('PC3 sorted components: ', loading_scores_PC3_sorted.index)
print(loading_scores_PC3_sorted)
```

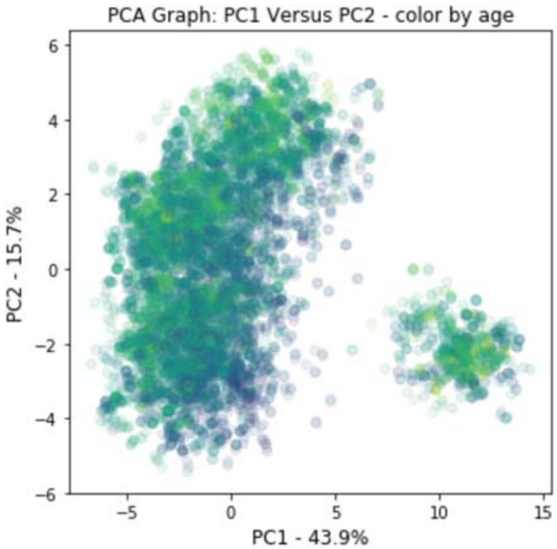
```
Sorted PC1 Loading Scores (abs)
PC1 sorted components:  Index(['ball_control', 'dribbling', 'short_passing', 'crossing', '
curve',
       'long_shots', 'positioning', 'shot_power', 'vision', 'gk_diving',
       'free_kick_accuracy', 'gk_reflexes', 'gk_handling', 'gk_positioning',
       'volleys', 'finishing', 'acceleration', 'penalties', 'sprint_speed',
       'long_passing', 'agility', 'stamina', 'balance', 'height',
       'heading_accuracy', 'weight', 'gk_kicking', 'reactions', 'aggression',
       'overall_rating', 'potential', 'interceptions', 'standing_tackle',
       'sliding_tackle', 'marking', 'strength', 'age', 'jumping'],
      dtype='object')
ball_control          0.233895
dribbling             0.226609
short_passing         0.220168
crossing              0.213529
curve                 0.211442
long_shots            0.211257
positioning           0.204851
shot_power            0.199655
vision                0.197614
gk_diving             0.196600
free_kick_accuracy    0.196366
gk_reflexes           0.195234
gk_handling           0.195011
gk_positioning        0.193662
volleys               0.193077
finishing             0.191773
acceleration          0.185480
penalties             0.183370
sprint_speed          0.182038
long_passing          0.177636
agility               0.168734
stamina               0.158548
balance               0.153811
height                0.134852
heading_accuracy      0.132158
weight                0.125473
gk_kicking            0.121982
reactions             0.105162
aggression            0.087644
overall_rating        0.083033
potential             0.080174
interceptions         0.071469
standing_tackle       0.065230
sliding_tackle        0.059237
marking               0.052163
strength              0.028138
age                   0.005573
jumping               0.001561
dtype: float64
=======
Sorted PC2 Loading Scores (abs)
PC2 sorted components:  Index(['marking', 'standing_tackle', 'sliding_tackle', 'intercepti
ons',
       'aggression', 'strength', 'heading_accuracy', 'agility', 'height',
       'weight', 'finishing', 'stamina', 'volleys', 'balance',
       'gk_positioning', 'gk_handling', 'acceleration', 'gk_reflexes',
       'positioning', 'gk_diving', 'jumping', 'dribbling', 'age', 'curve',
       'gk_kicking', 'sprint_speed', 'long_passing', 'penalties', 'long_shots',
       'vision', 'short_passing', 'free_kick_accuracy', 'overall_rating',
       'shot_power', 'crossing', 'reactions', 'potential', 'ball_control'],
      dtype='object')
marking               0.360160
standing_tackle       0.357693
sliding_tackle        0.351805
interceptions         0.323485
aggression            0.310414
strength              0.269961
heading_accuracy      0.231016
agility               0.172402
```

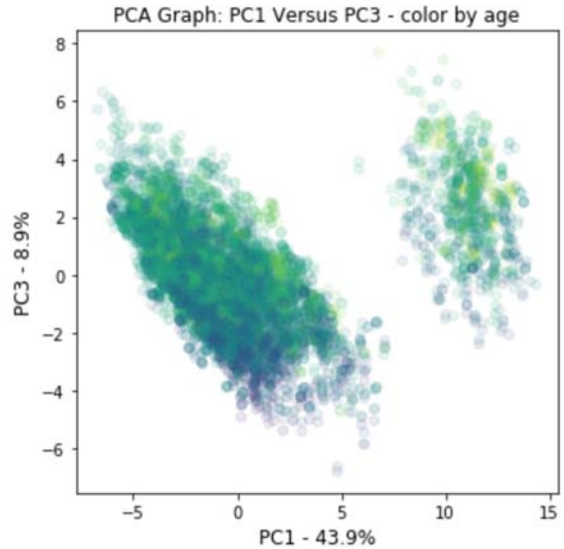
Present graphs pertinent to the first three principal componets:

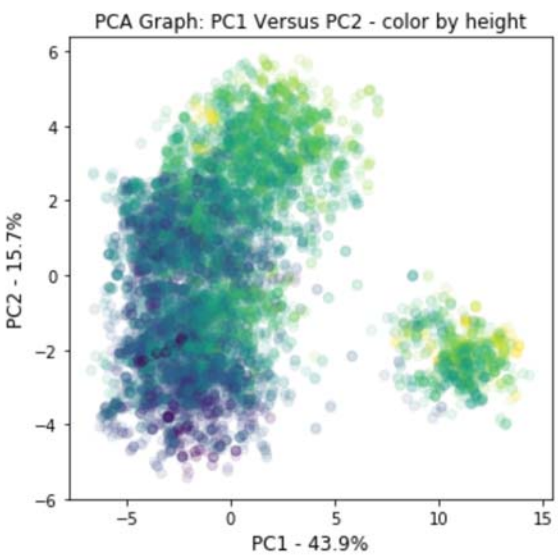
Note: When plotting PC1 versus PC2 AND PC1 versus PC3, two clusters are displayed.

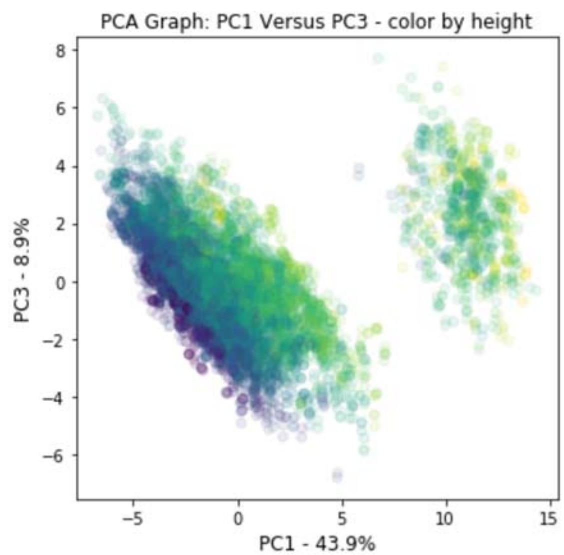
Next, we will visualize further in PC1, PC2 and PC3.

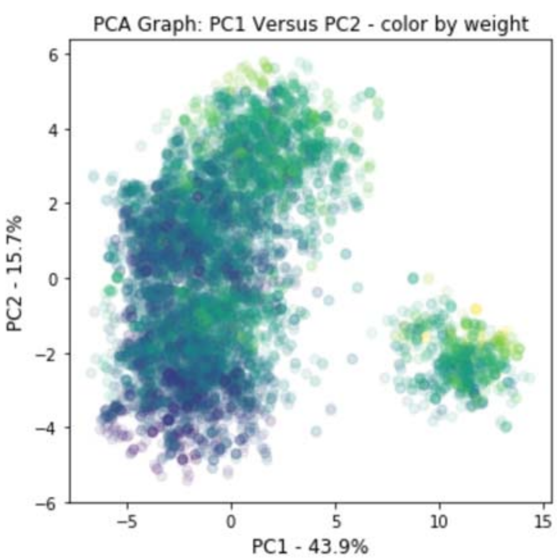
In [13]:
```
# draw PCA 2D plot: PC1 Vs PC2 and PC1 Vs PC3
def color_plot (i):
    plt.figure(figsize=(10, 5))
    plt.subplot(1,2,1)

    plt.scatter(pca_df.PC1, pca_df.PC2, c=scaled_data[:,i], alpha=0.1)
    plt.title("PCA Graph: PC1 Versus PC2 - color by " + numeric_col[i], fontsize='12')
    plt.xlabel('PC1 - {0}%'.format(per_var[0]), fontsize='12')
    plt.ylabel('PC2 - {0}%'.format(per_var[1]), fontsize='12')
    plt.tight_layout()

    plt.subplot(1,2,2)
    plt.scatter(pca_df.PC1, pca_df.PC3, c=scaled_data[:,i], alpha=0.1)
    plt.title("PCA Graph: PC1 Versus PC3 - color by " + numeric_col[i], fontsize='12')
    plt.xlabel('PC1 - {0}%'.format(per_var[0]), fontsize='12')
    plt.ylabel('PC3 - {0}%'.format(per_var[2]), fontsize='12')
    plt.tight_layout()
    plt.show()
    plt.close()

for j in range(0,38):
    color_plot (j)
```

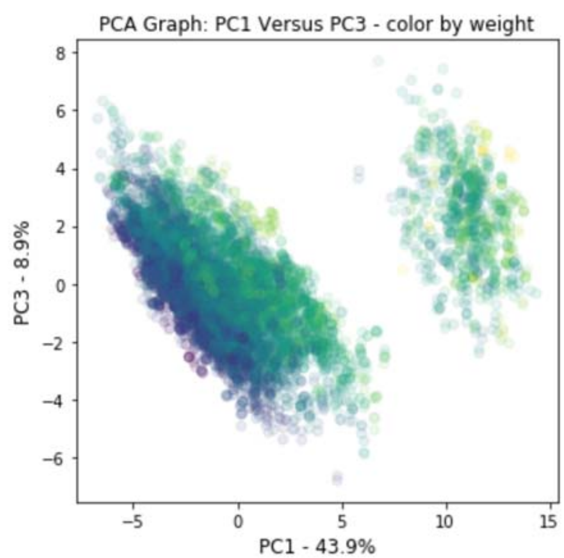

PCA Graph: PC1 Versus PC2 - color by age
PC2 - 15.7%
PC1 - 43.9%

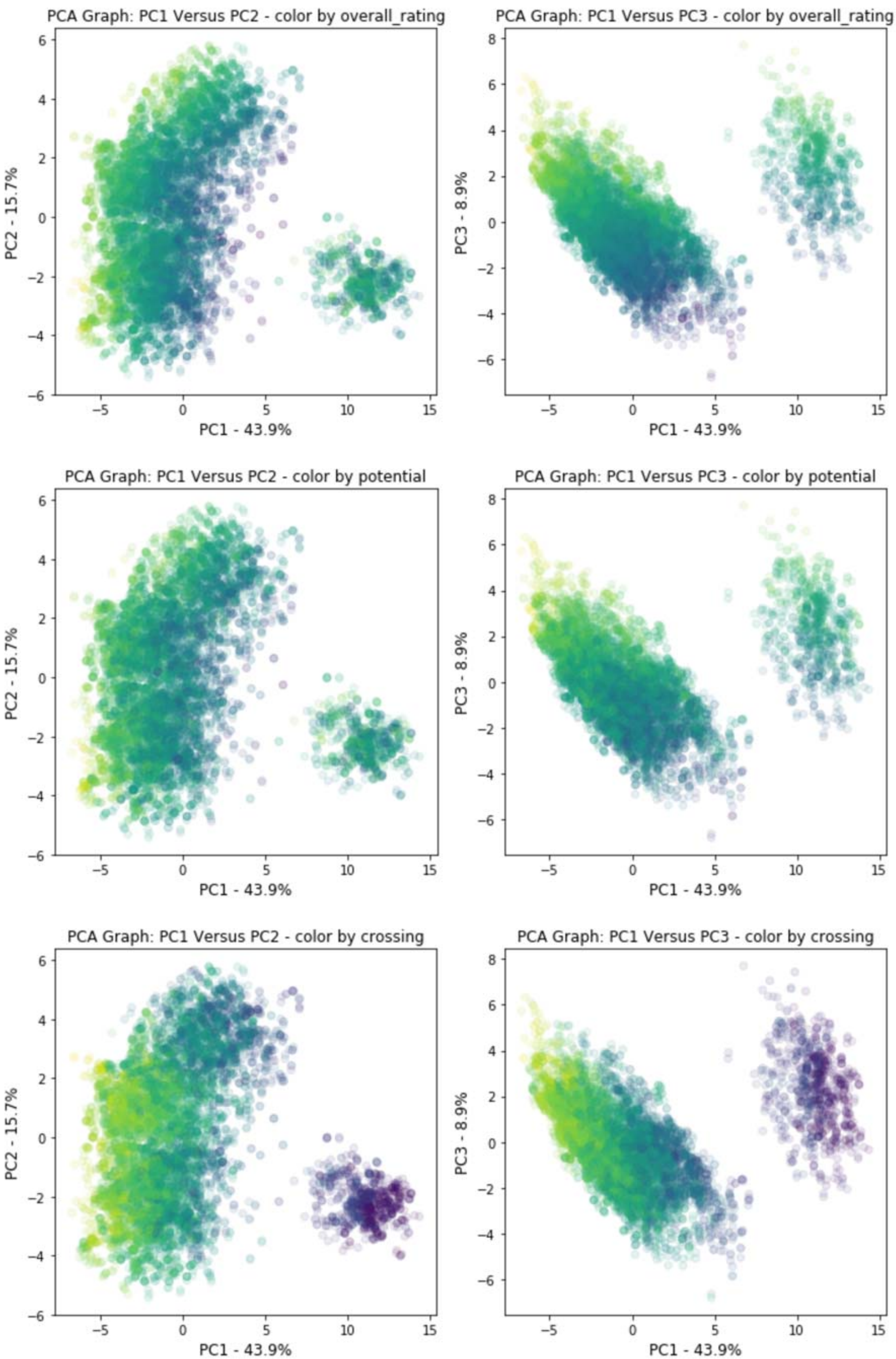

PCA Graph: PC1 Versus PC3 - color by age
PC3 - 8.9%
PC1 - 43.9%

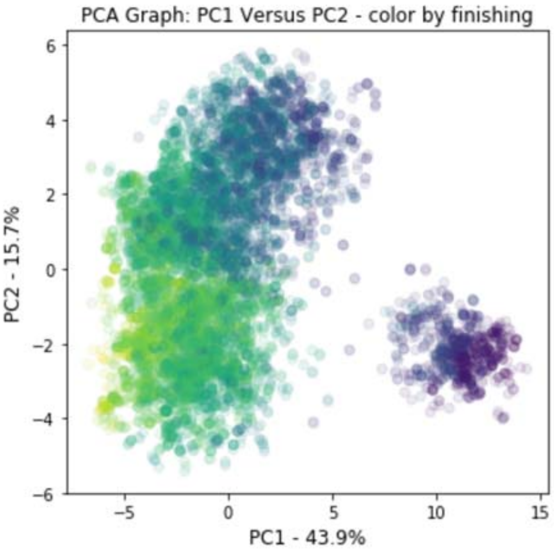

PCA Graph: PC1 Versus PC2 - color by height
PC2 - 15.7%
PC1 - 43.9%

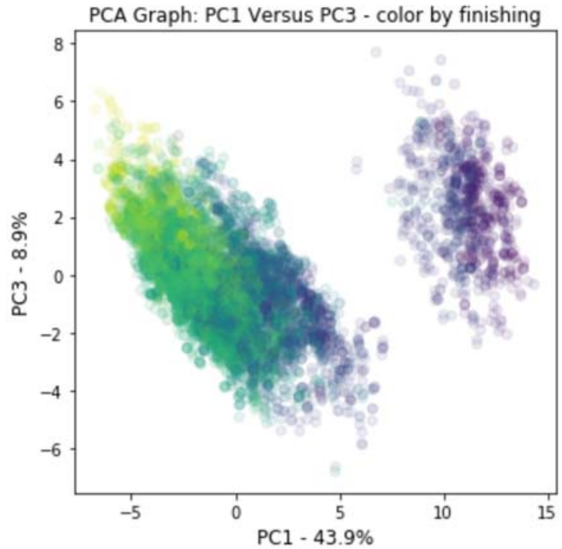

PCA Graph: PC1 Versus PC3 - color by height
PC3 - 8.9%
PC1 - 43.9%

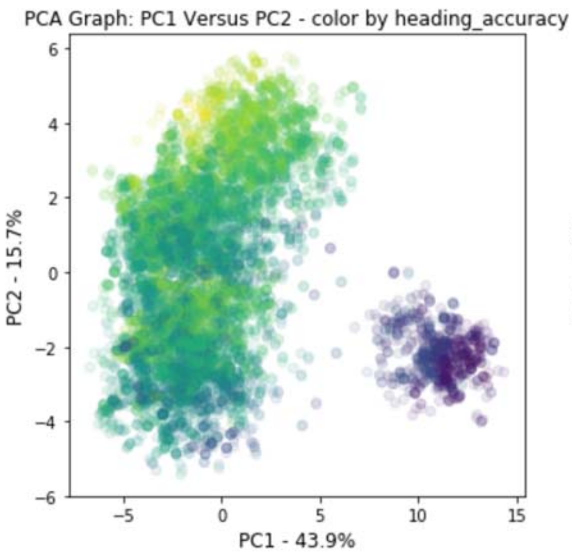

PCA Graph: PC1 Versus PC2 - color by weight
PC2 - 15.7%
PC1 - 43.9%

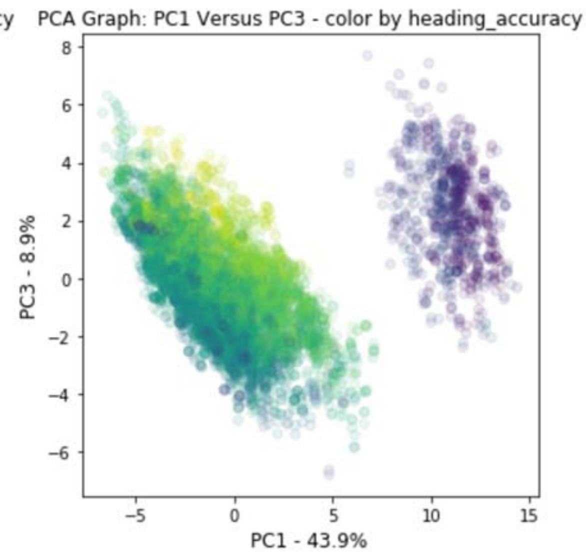

PCA Graph: PC1 Versus PC3 - color by weight
PC3 - 8.9%
PC1 - 43.9%

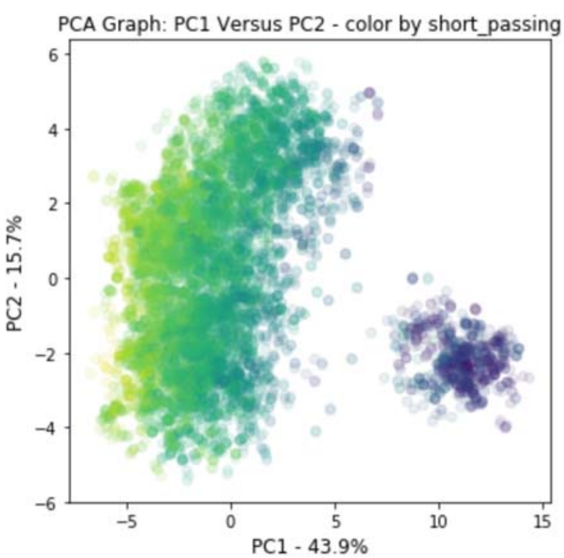

PCA Graph: PC1 Versus PC2 - color by overall_rating
PCA Graph: PC1 Versus PC3 - color by overall_rating
PCA Graph: PC1 Versus PC2 - color by potential
PCA Graph: PC1 Versus PC3 - color by potential
PCA Graph: PC1 Versus PC2 - color by crossing
PCA Graph: PC1 Versus PC3 - color by crossing
PC1 - 43.9%
PC2 - 15.7%
PC3 - 8.9%

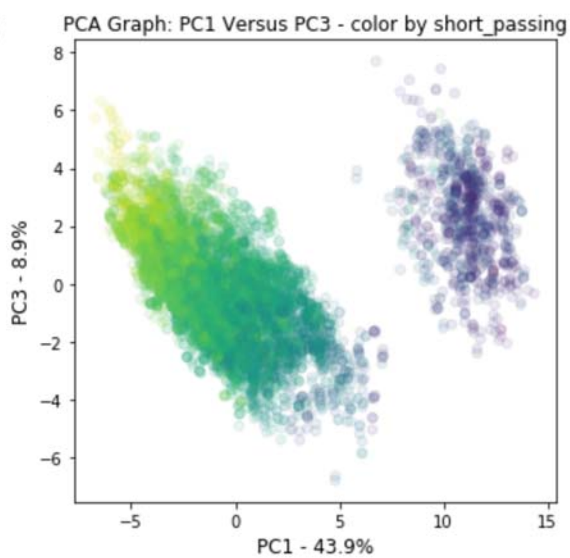

PCA Graph: PC1 Versus PC2 - color by finishing
PC2 - 15.7%
PC1 - 43.9%

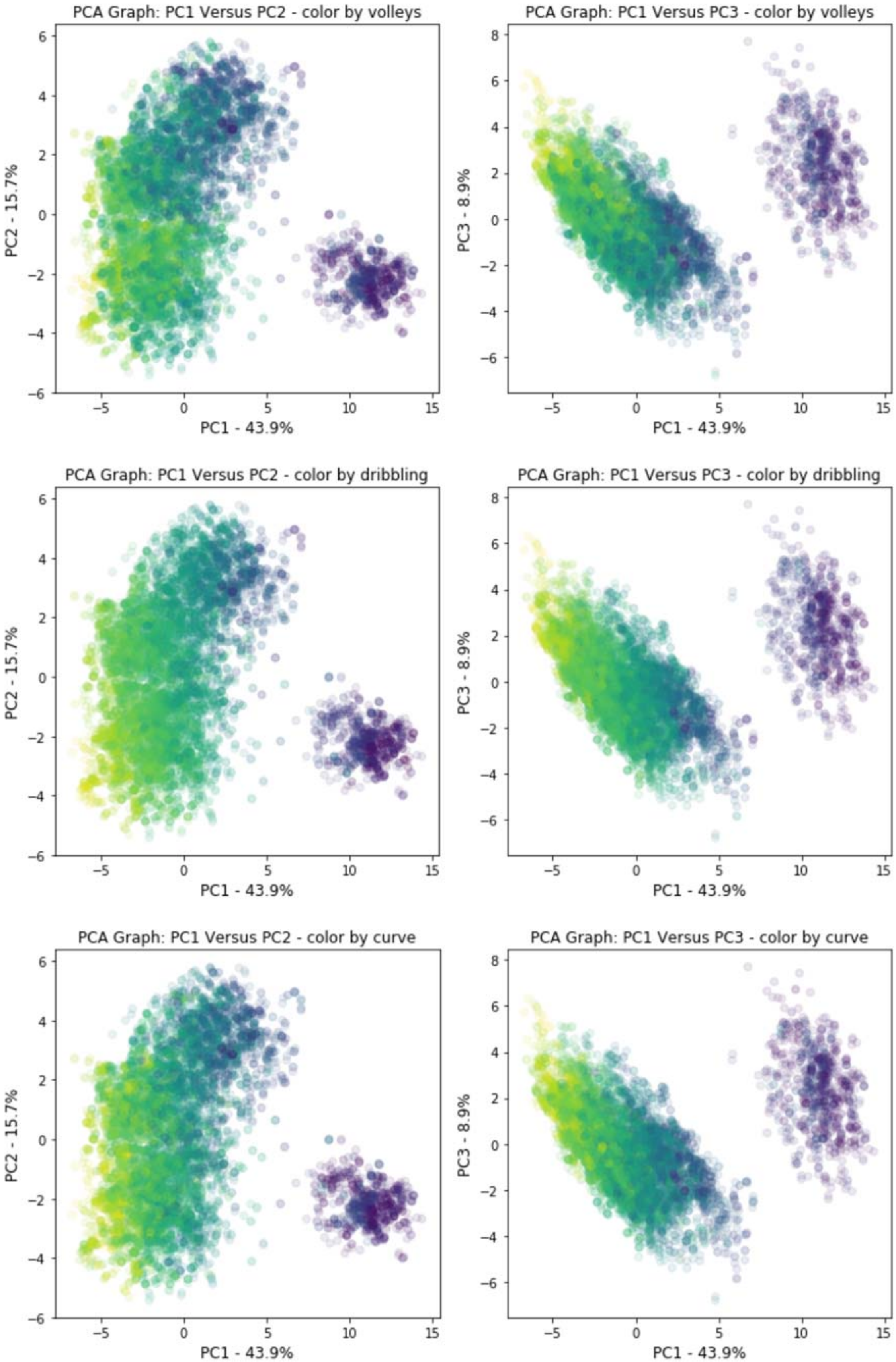

PCA Graph: PC1 Versus PC3 - color by finishing
PC3 - 8.9%
PC1 - 43.9%

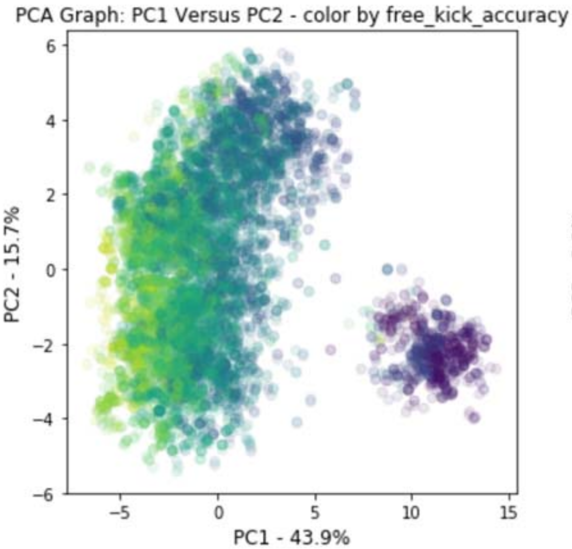

PCA Graph: PC1 Versus PC2 - color by heading_accuracy
PC2 - 15.7%
PC1 - 43.9%

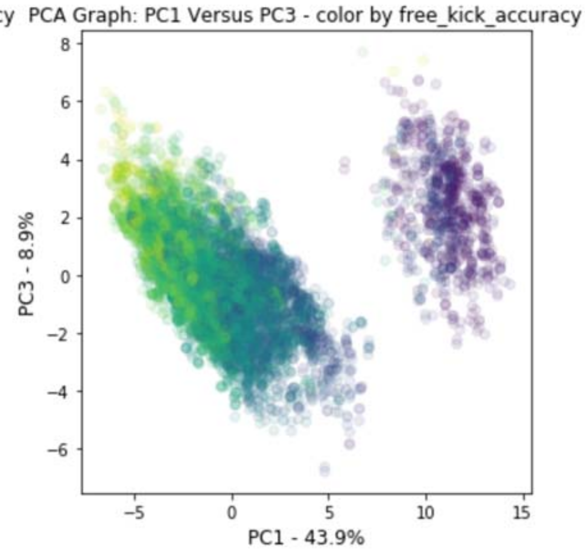

PCA Graph: PC1 Versus PC3 - color by heading_accuracy
PC3 - 8.9%
PC1 - 43.9%

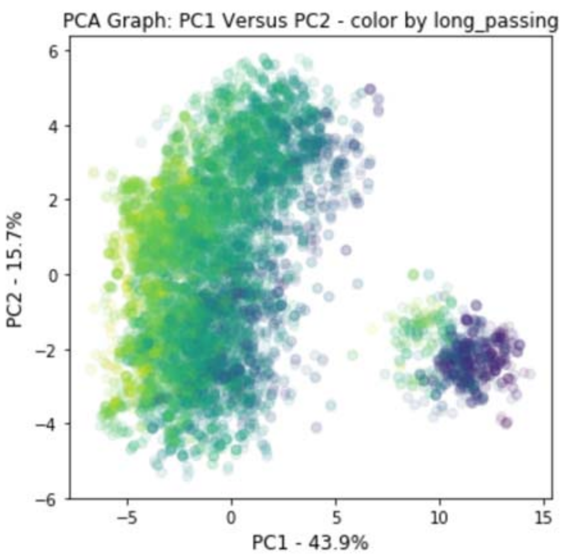

PCA Graph: PC1 Versus PC2 - color by short_passing
PC2 - 15.7%
PC1 - 43.9%

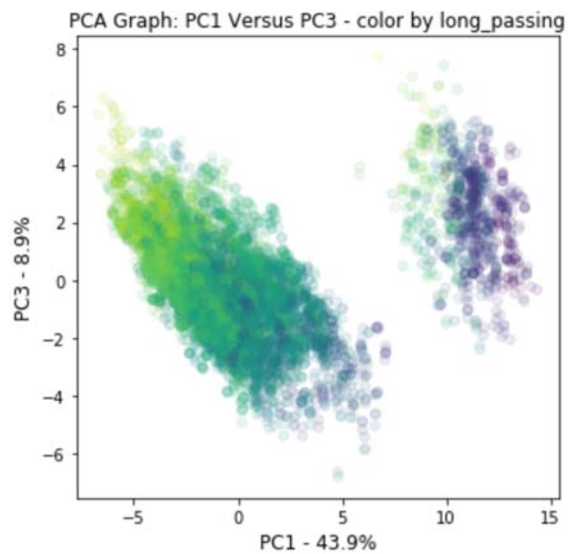

PCA Graph: PC1 Versus PC3 - color by short_passing
PC3 - 8.9%
PC1 - 43.9%

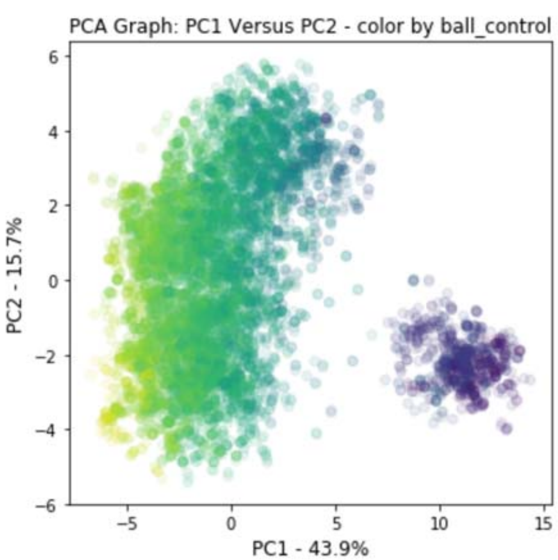

PCA Graph: PC1 Versus PC2 - color by volleys
PC2 - 15.7%
PC1 - 43.9%
PCA Graph: PC1 Versus PC3 - color by volleys
PC3 - 8.9%
PC1 - 43.9%
PCA Graph: PC1 Versus PC2 - color by dribbling
PC2 - 15.7%
PC1 - 43.9%
PCA Graph: PC1 Versus PC3 - color by dribbling
PC3 - 8.9%
PC1 - 43.9%
PCA Graph: PC1 Versus PC2 - color by curve
PC2 - 15.7%
PC1 - 43.9%
PCA Graph: PC1 Versus PC3 - color by curve
PC3 - 8.9%
PC1 - 43.9%

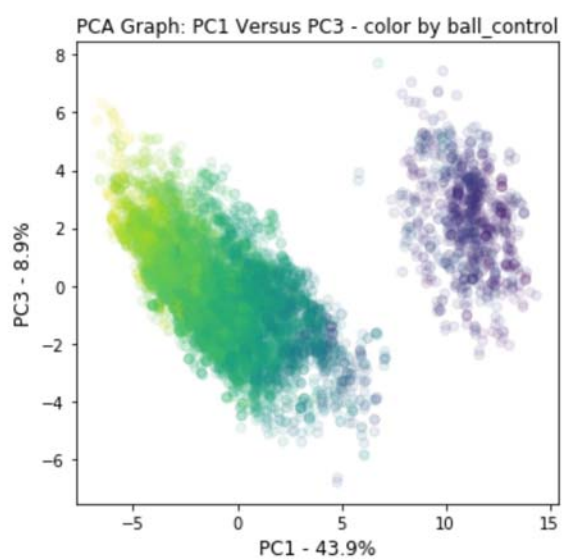

PCA Graph: PC1 Versus PC2 - color by free_kick_accuracy
PC2 - 15.7%
PC1 - 43.9%

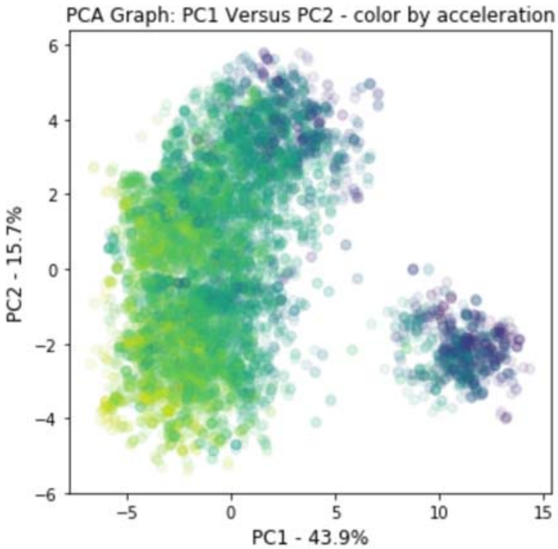

PCA Graph: PC1 Versus PC3 - color by free_kick_accuracy
PC3 - 8.9%
PC1 - 43.9%

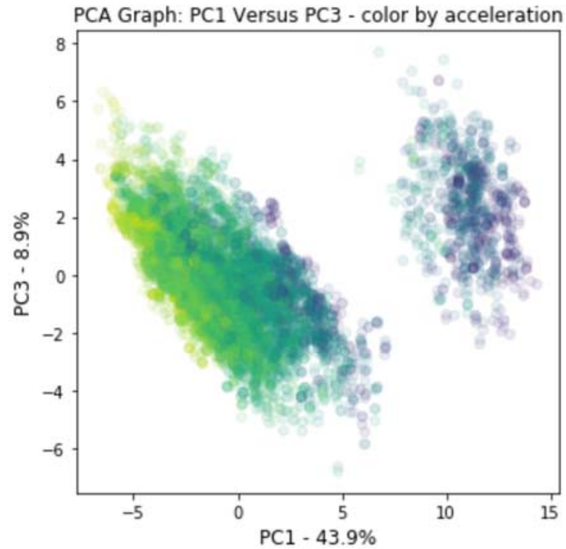

PCA Graph: PC1 Versus PC2 - color by long_passing
PC2 - 15.7%
PC1 - 43.9%

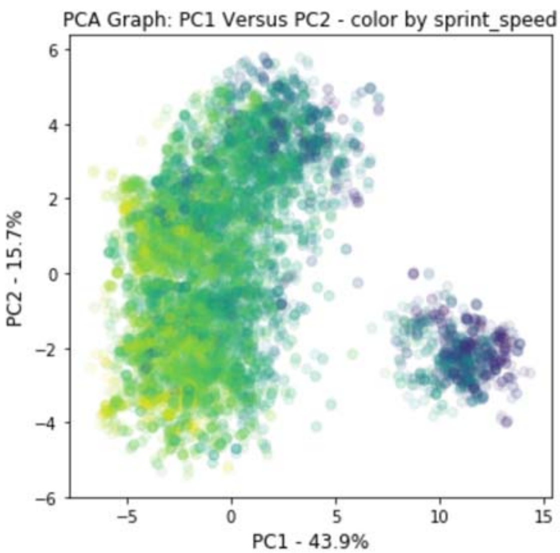

PCA Graph: PC1 Versus PC3 - color by long_passing
PC3 - 8.9%
PC1 - 43.9%

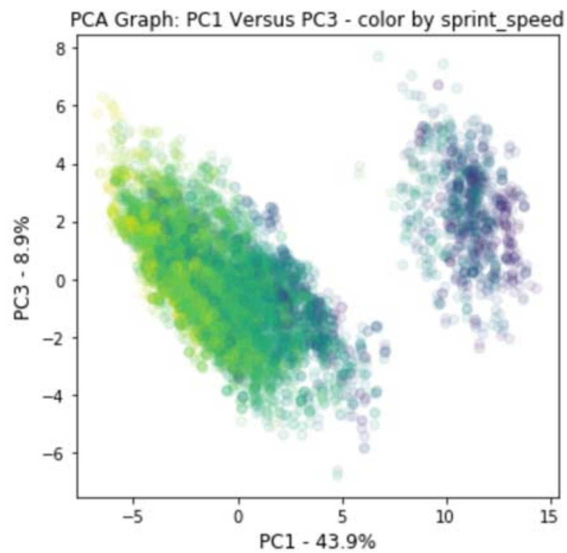

PCA Graph: PC1 Versus PC2 - color by ball_control
PC2 - 15.7%
PC1 - 43.9%

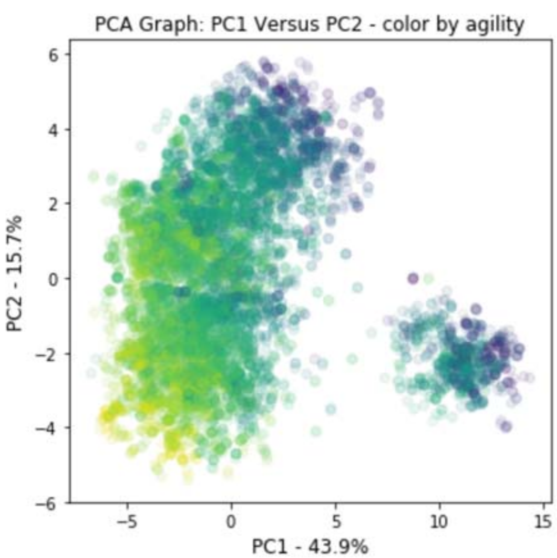

PCA Graph: PC1 Versus PC3 - color by ball_control
PC3 - 8.9%
PC1 - 43.9%

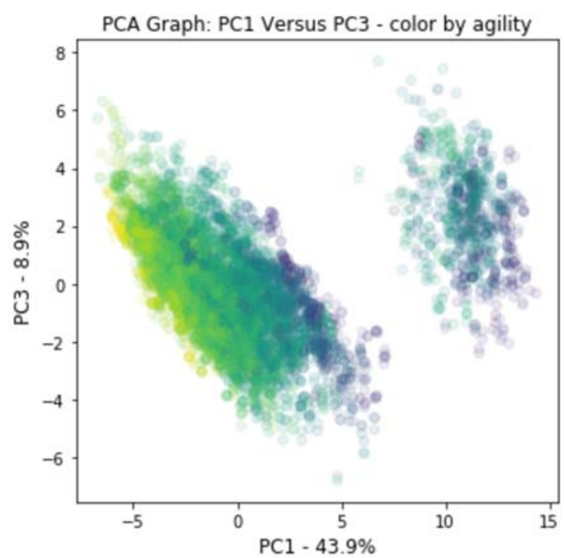

PCA Graph: PC1 Versus PC2 - color by acceleration
PC2 - 15.7%
PC1 - 43.9%

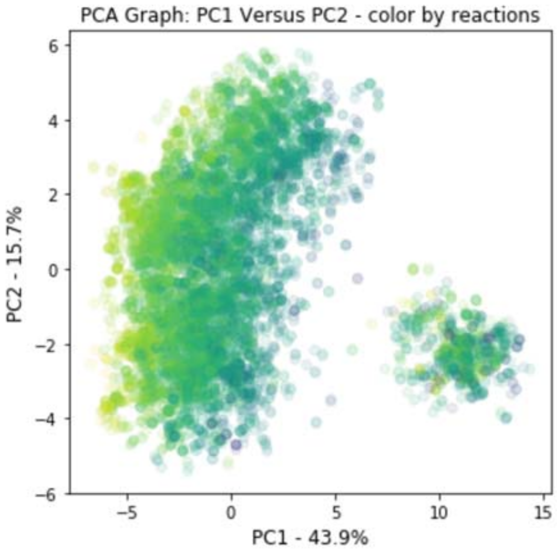

PCA Graph: PC1 Versus PC3 - color by acceleration
PC3 - 8.9%
PC1 - 43.9%

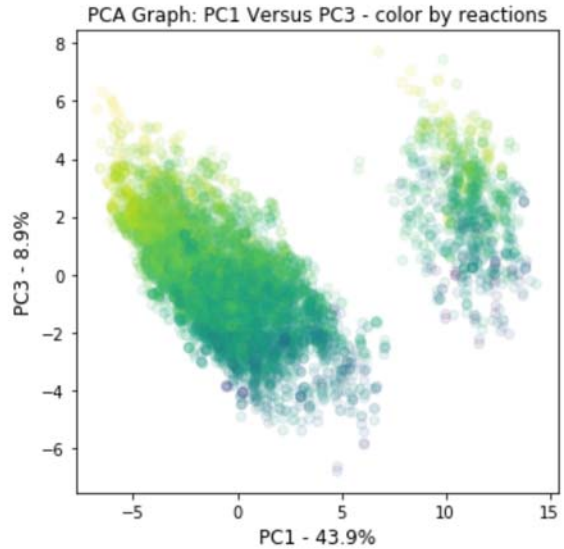

PCA Graph: PC1 Versus PC2 - color by sprint_speed
PC2 - 15.7%
PC1 - 43.9%

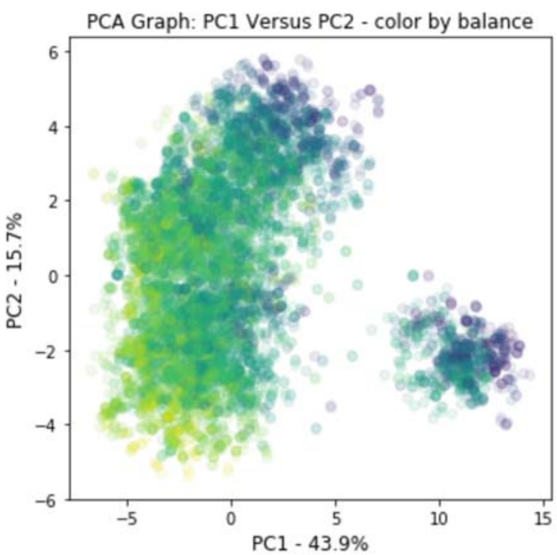

PCA Graph: PC1 Versus PC3 - color by sprint_speed
PC3 - 8.9%
PC1 - 43.9%

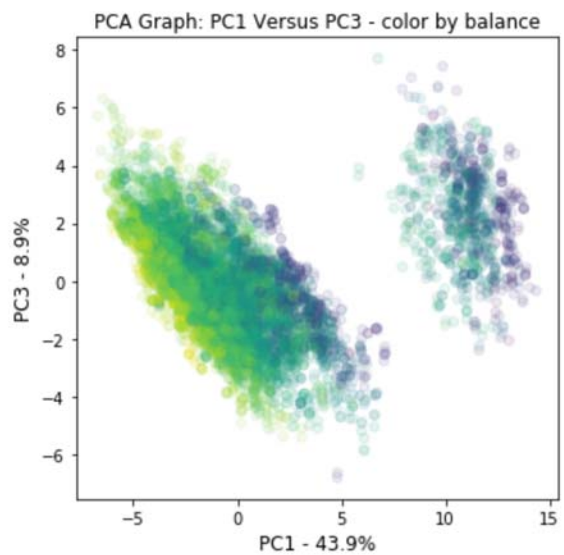

PCA Graph: PC1 Versus PC2 - color by agility
PC2 - 15.7%
PC1 - 43.9%

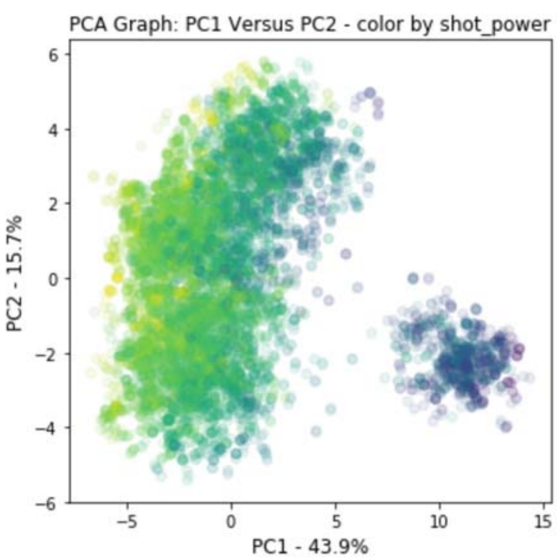

PCA Graph: PC1 Versus PC3 - color by agility
PC3 - 8.9%
PC1 - 43.9%

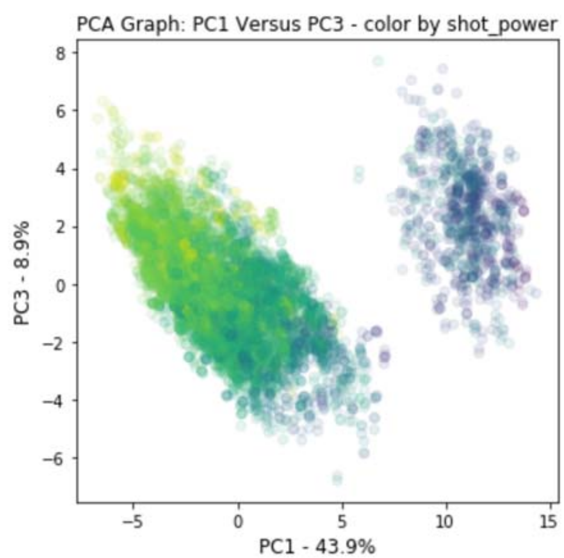

PCA Graph: PC1 Versus PC2 - color by reactions
PC2 - 15.7%
PC1 - 43.9%

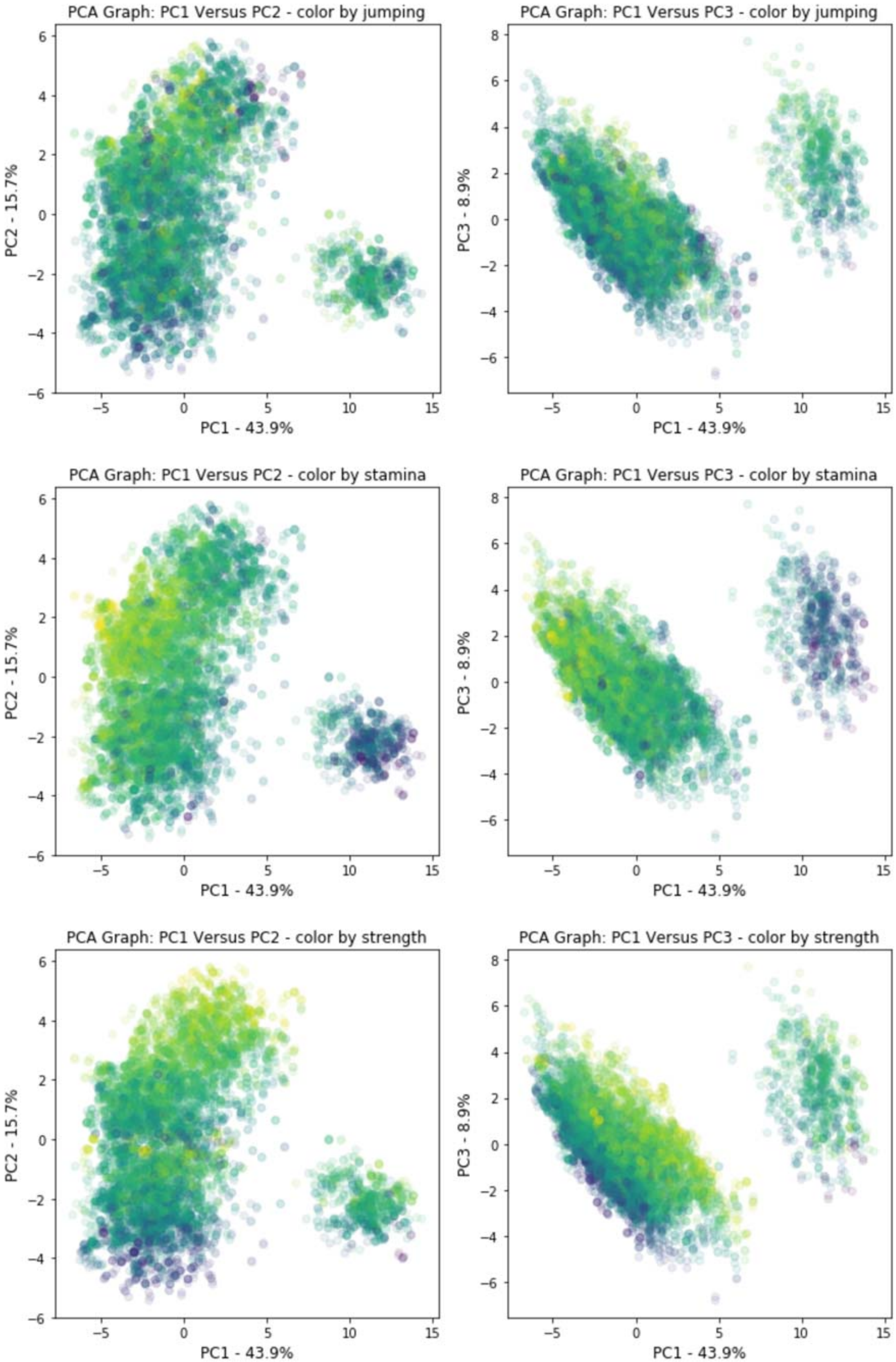

PCA Graph: PC1 Versus PC3 - color by reactions
PC3 - 8.9%
PC1 - 43.9%

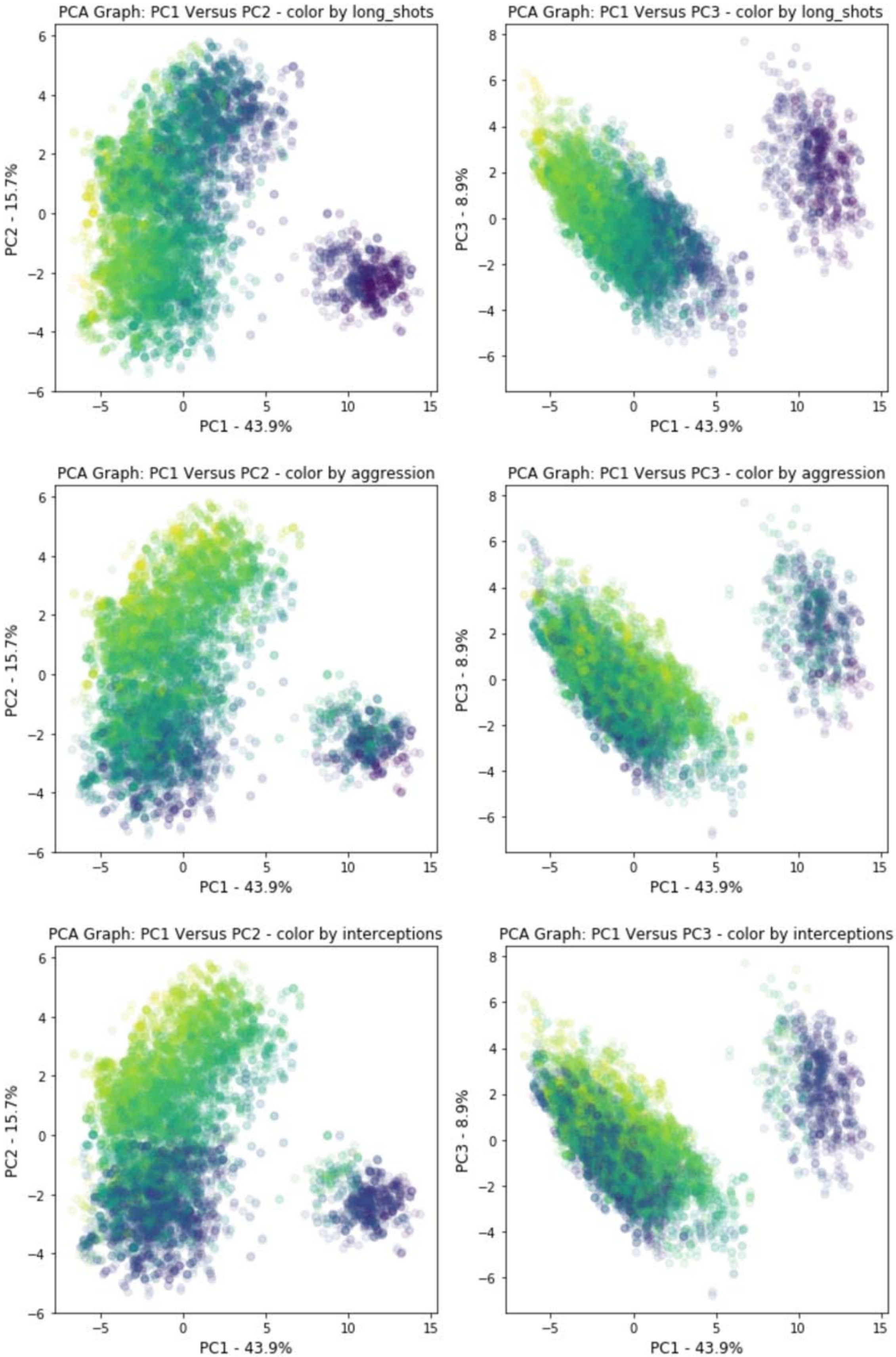

PCA Graph: PC1 Versus PC2 - color by balance
PC2 - 15.7%
PC1 - 43.9%

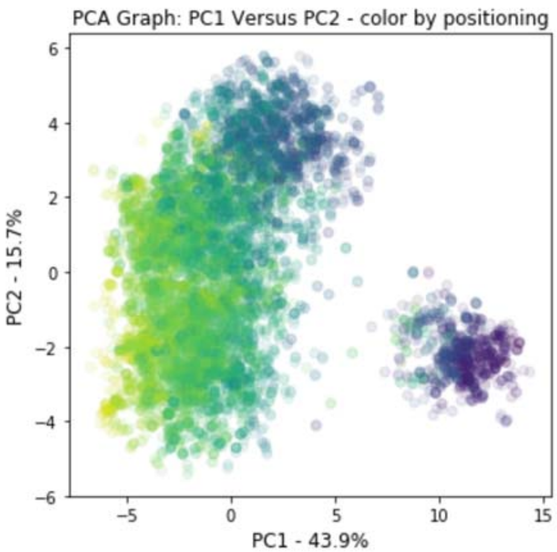

PCA Graph: PC1 Versus PC3 - color by balance
PC3 - 8.9%
PC1 - 43.9%

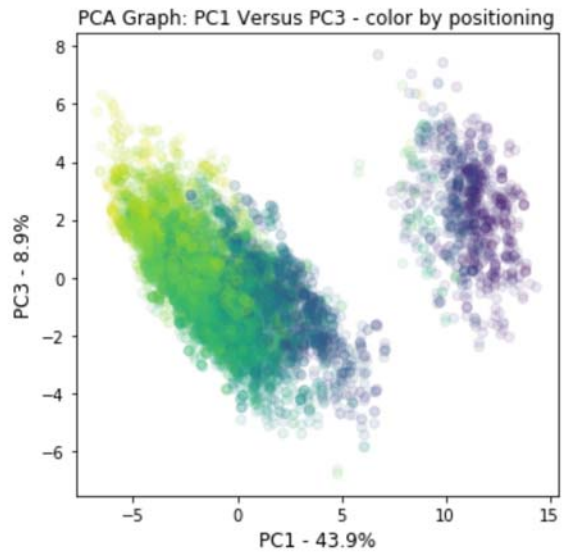

PCA Graph: PC1 Versus PC2 - color by shot_power
PC2 - 15.7%
PC1 - 43.9%

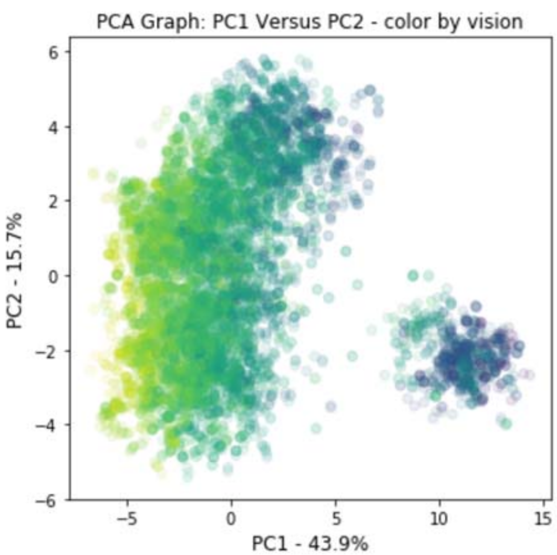

PCA Graph: PC1 Versus PC3 - color by shot_power
PC3 - 8.9%
PC1 - 43.9%

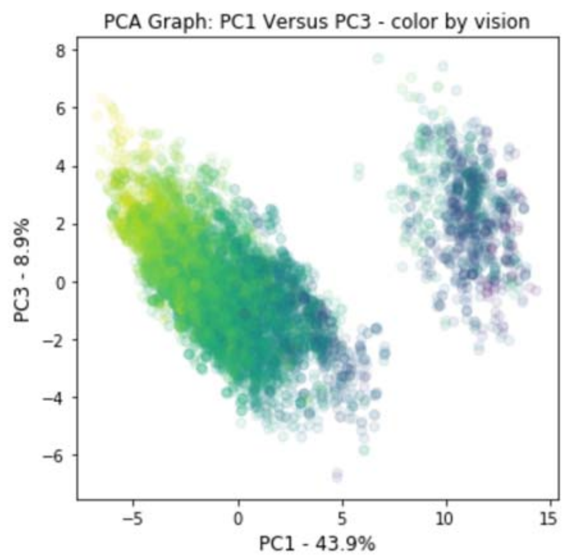

PCA Graph: PC1 Versus PC2 - color by jumping
PCA Graph: PC1 Versus PC3 - color by jumping
PCA Graph: PC1 Versus PC2 - color by stamina
PCA Graph: PC1 Versus PC3 - color by stamina
PCA Graph: PC1 Versus PC2 - color by strength
PCA Graph: PC1 Versus PC3 - color by strength
PC1 - 43.9%
PC2 - 15.7%
PC3 - 8.9%

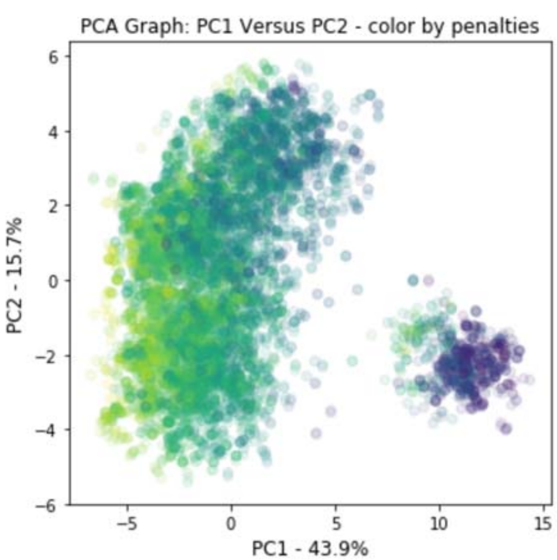

PCA Graph: PC1 Versus PC2 - color by long_shots
PCA Graph: PC1 Versus PC3 - color by long_shots
PCA Graph: PC1 Versus PC2 - color by aggression
PCA Graph: PC1 Versus PC3 - color by aggression
PCA Graph: PC1 Versus PC2 - color by interceptions
PCA Graph: PC1 Versus PC3 - color by interceptions
PC1 - 43.9%
PC2 - 15.7%
PC3 - 8.9%

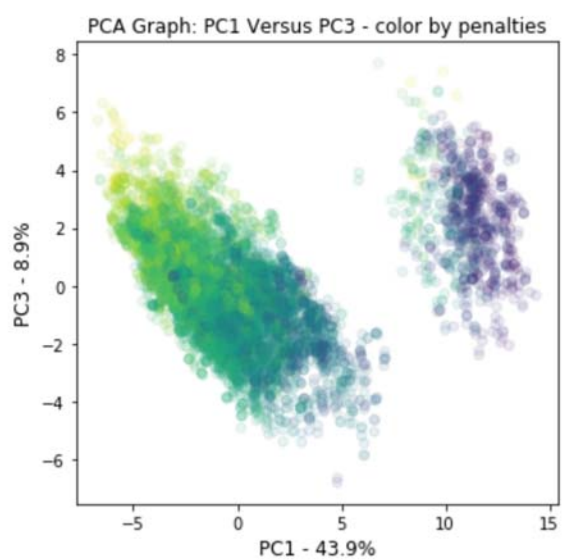

PCA Graph: PC1 Versus PC2 - color by positioning
PC2 - 15.7%
PC1 - 43.9%

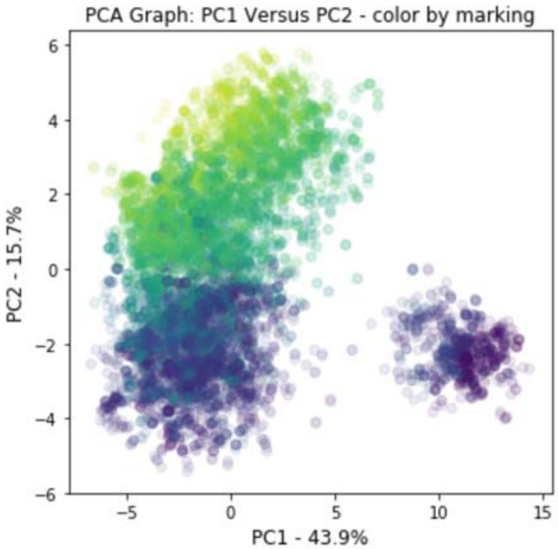

PCA Graph: PC1 Versus PC3 - color by positioning
PC3 - 8.9%
PC1 - 43.9%

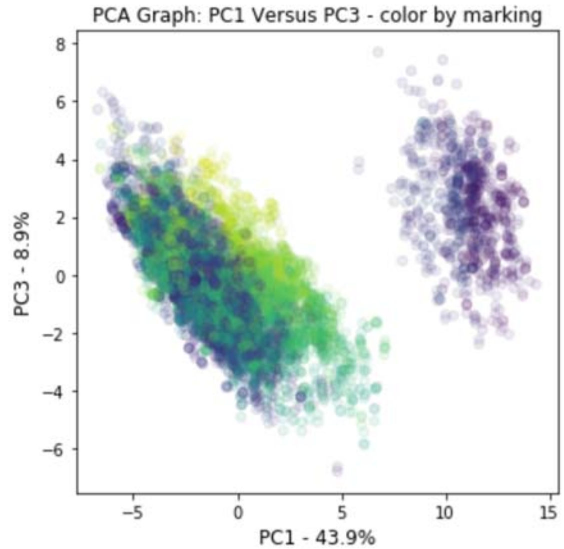

PCA Graph: PC1 Versus PC2 - color by vision
PC2 - 15.7%
PC1 - 43.9%

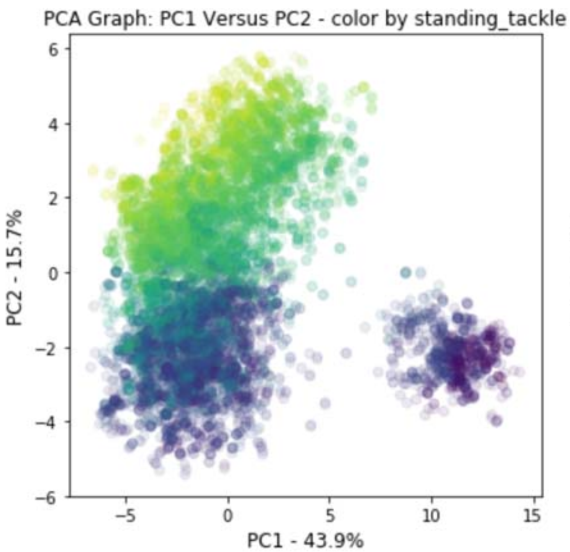

PCA Graph: PC1 Versus PC3 - color by vision
PC3 - 8.9%
PC1 - 43.9%

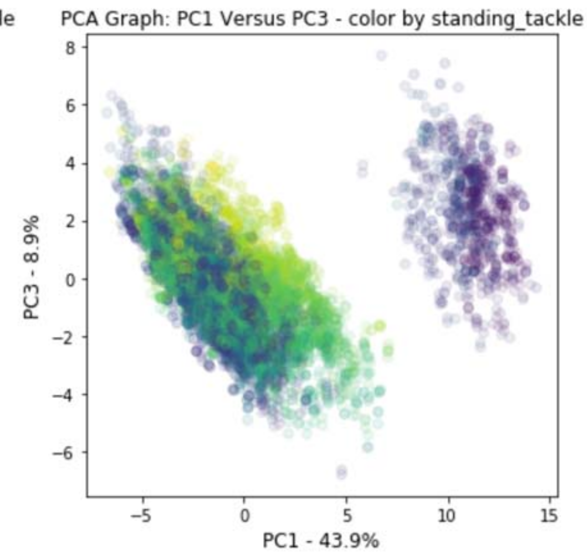

PCA Graph: PC1 Versus PC2 - color by penalties
PC2 - 15.7%
PC1 - 43.9%

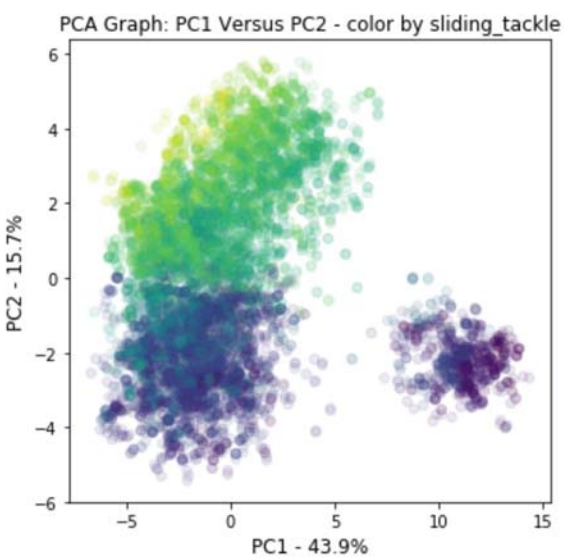

PCA Graph: PC1 Versus PC3 - color by penalties
PC3 - 8.9%
PC1 - 43.9%

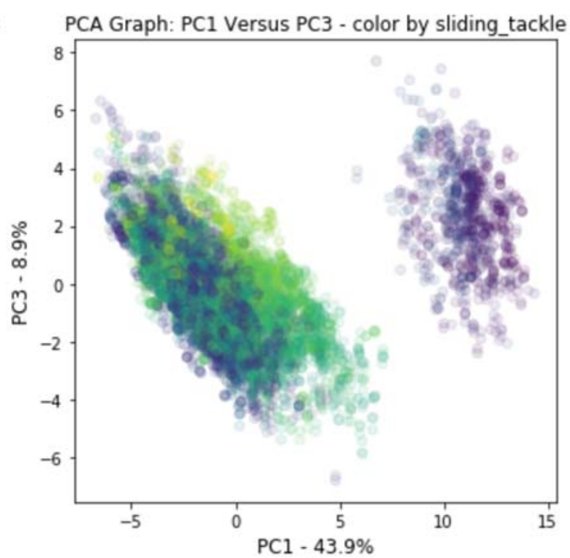

PCA Graph: PC1 Versus PC2 - color by marking
PC2 - 15.7%
PC1 - 43.9%

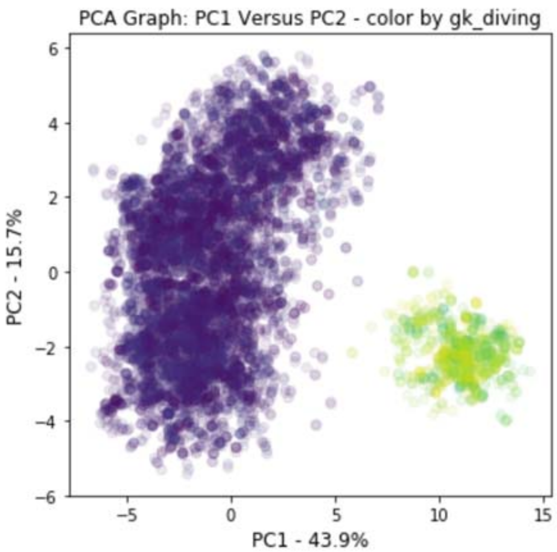

PCA Graph: PC1 Versus PC3 - color by marking
PC3 - 8.9%
PC1 - 43.9%

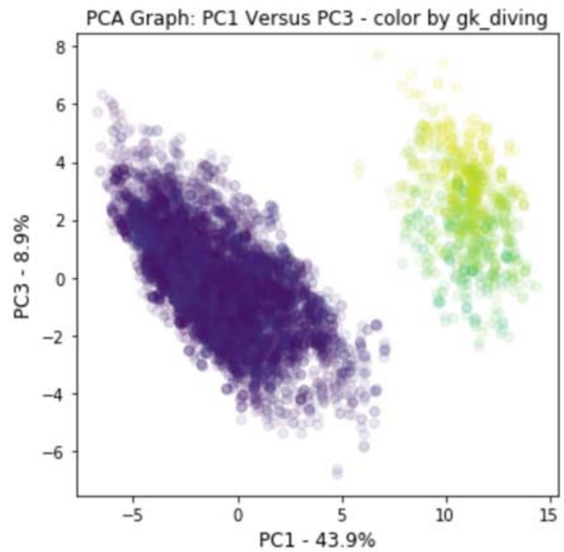

PCA Graph: PC1 Versus PC2 - color by standing_tackle
PC2 - 15.7%
PC1 - 43.9%

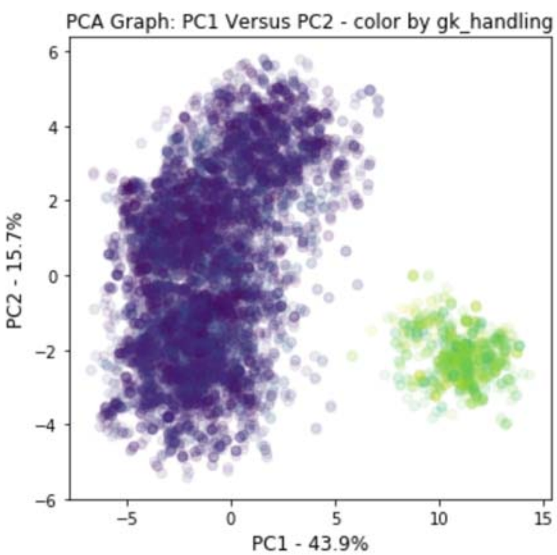

PCA Graph: PC1 Versus PC3 - color by standing_tackle
PC3 - 8.9%
PC1 - 43.9%

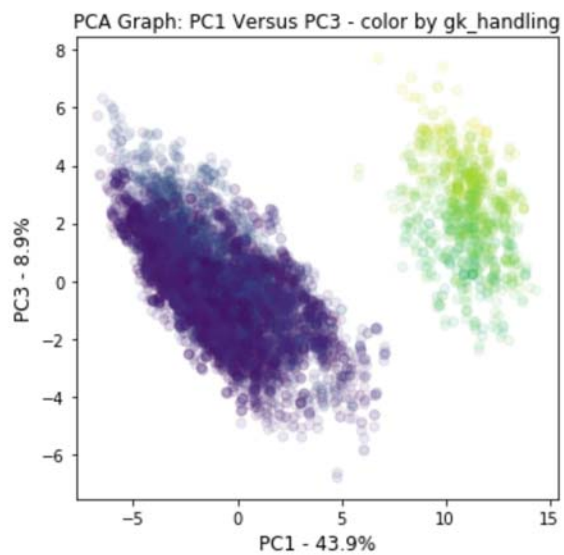

PCA Graph: PC1 Versus PC2 - color by sliding_tackle
PC2 - 15.7%
PC1 - 43.9%

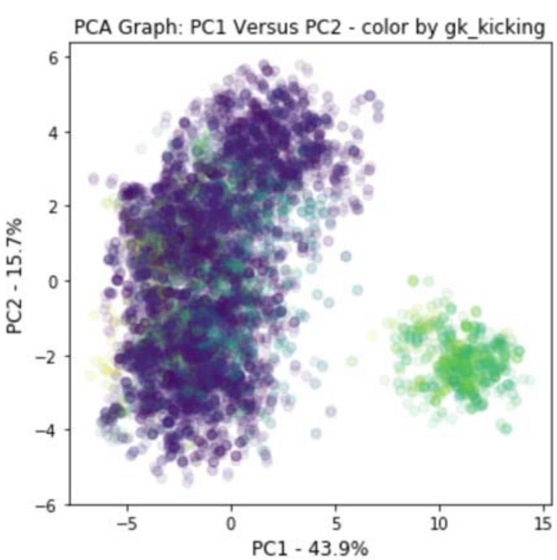

PCA Graph: PC1 Versus PC3 - color by sliding_tackle
PC3 - 8.9%
PC1 - 43.9%

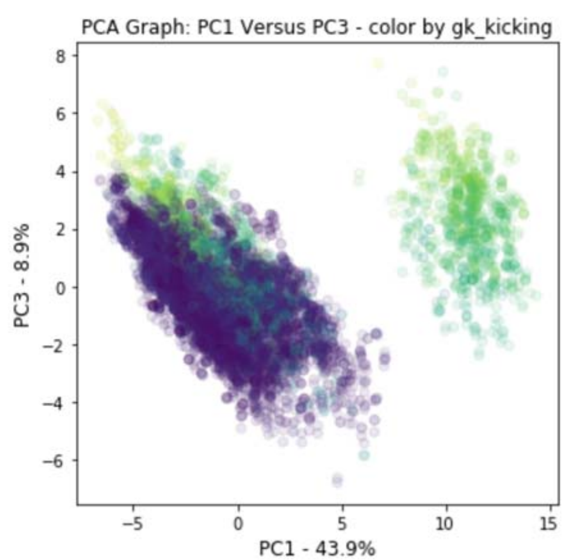

PCA Graph: PC1 Versus PC2 - color by gk_diving
PC2 - 15.7%
PC1 - 43.9%

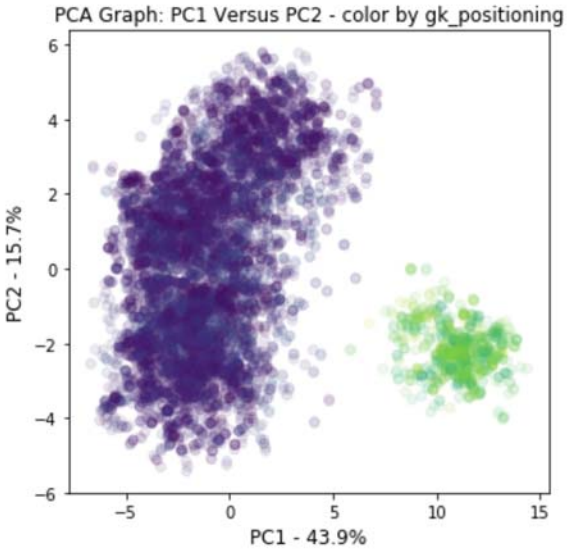

PCA Graph: PC1 Versus PC3 - color by gk_diving
PC3 - 8.9%
PC1 - 43.9%

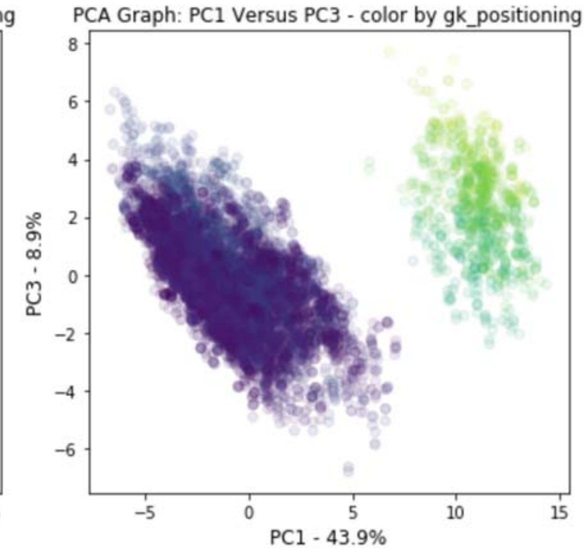

PCA Graph: PC1 Versus PC2 - color by gk_handling
PC2 - 15.7%
PC1 - 43.9%

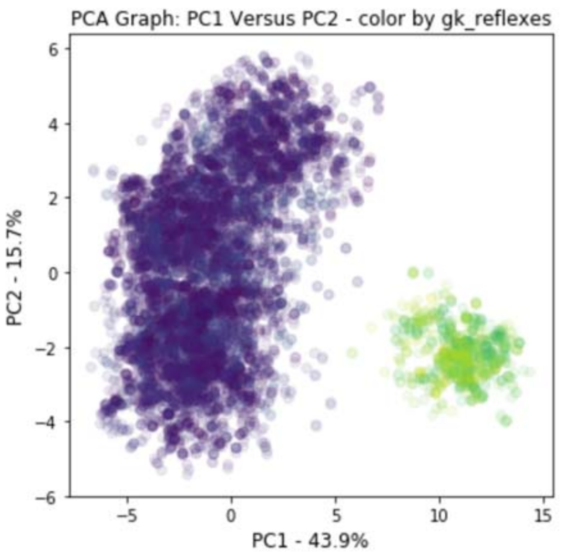

PCA Graph: PC1 Versus PC3 - color by gk_handling
PC3 - 8.9%
PC1 - 43.9%

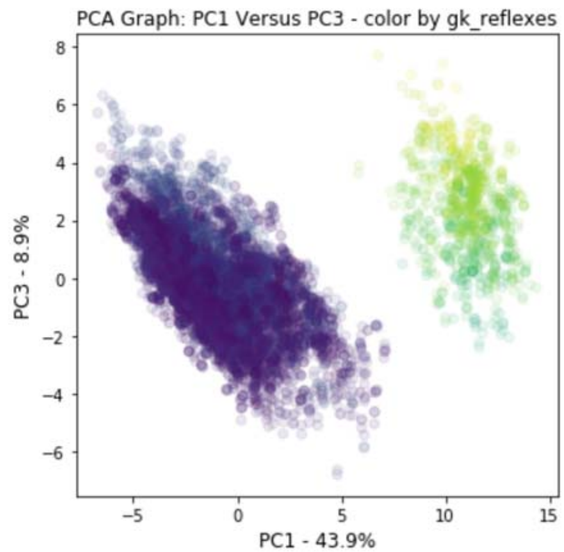

PCA Graph: PC1 Versus PC2 - color by gk_kicking
PC2 - 15.7%
PC1 - 43.9%


PCA Graph: PC1 Versus PC3 - color by gk_kicking
PC3 - 8.9%
PC1 - 43.9%


PCA Graph: PC1 Versus PC2 - color by gk_positioning
PC2 - 15.7%
PC1 - 43.9%

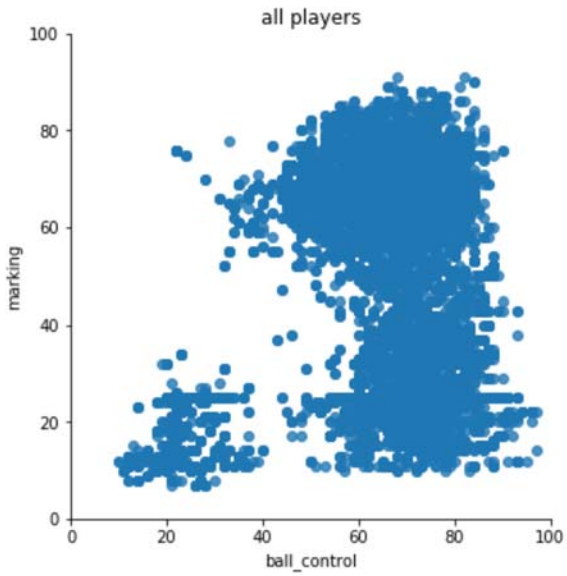

PCA Graph: PC1 Versus PC3 - color by gk_positioning
PC3 - 8.9%
PC1 - 43.9%

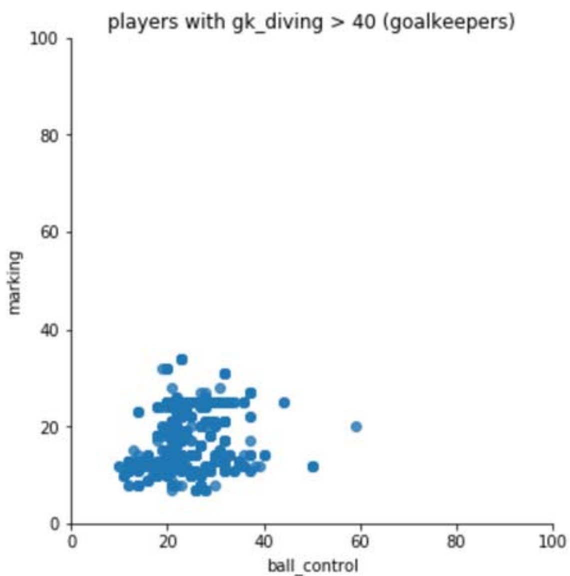

PCA Graph: PC1 Versus PC2 - color by gk_reflexes
PC2 - 15.7%
PC1 - 43.9%

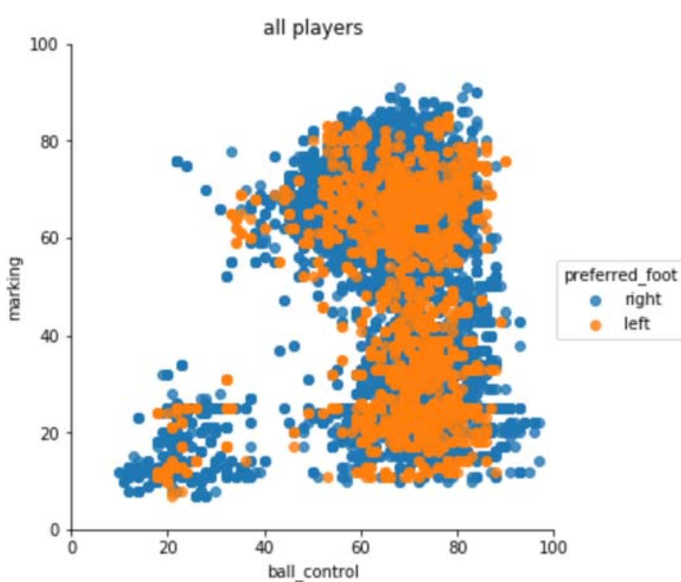

PCA Graph: PC1 Versus PC3 - color by gk_reflexes
PC3 - 8.9%
PC1 - 43.9%

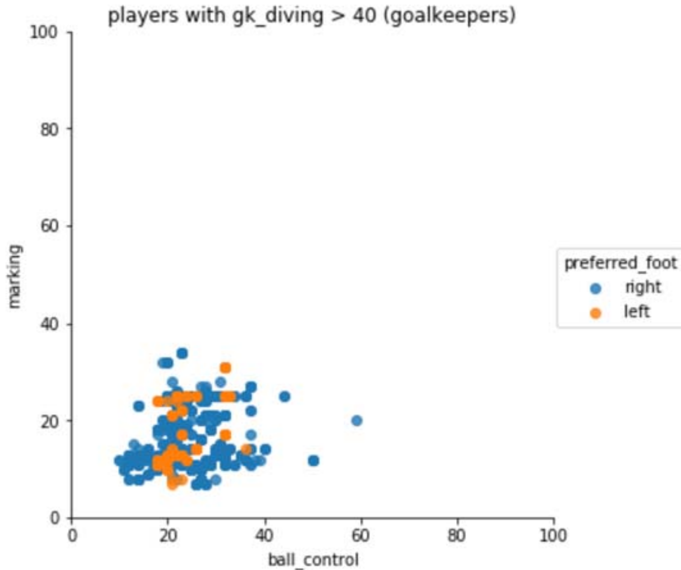
In [17]:
```
#'attacking_work_rate', 'defensive_work_rate']
#plt.title('players with gk_diving > 40',loc='center')
def plot (df_all, df_sub, hue_col):
    # first plot
    vis1=sns.lmplot(x='ball_control', y='marking',  hue=hue_col, sharex=False, data=df_all,
scatter=True, fit_reg=False, units=None, order=1, legend=True)
    plt.title('all players')
    plt.xlim(0,100)
    plt.ylim(0,100)
    plt.show()
    plt.close()
    # second plot: goalkeepers only
    vis2=sns.lmplot(x='ball_control', y='marking',  hue=hue_col, sharex=False, data=df_sub,
scatter=True, fit_reg=False, units=None, order=1, legend=True)
    plt.title('players with gk_diving > 40 (goalkeepers)')
    plt.xlim(0,100)
    plt.ylim(0,100)
    plt.show()
    plt.close()

print('gk_diving > 40 (goalkeepers)')
df_goalkeepers=df_all_col.loc[df_all_col['gk_diving']>40]
#print(df_goalkeepers.head())
plot(df_all_col, df_goalkeepers, None)

# color by lefty and righty
plot(df_all_col, df_goalkeepers, 'preferred_foot')
# plot lefty only
print('plot preferred left foot')
df1=df_all_col.loc[df_all_col['preferred_foot']=='left']
df2=df_goalkeepers.loc[df_goalkeepers['preferred_foot']=='left']
plot(df1,df2,None)

# color by 'attaching_work_rate'
df3=df_all_col[df_all_col['attacking_work_rate'].isin (['low','medium','high'])]
df4=df_goalkeepers[df_goalkeepers['attacking_work_rate'].isin (['low','medium','high'])]
plot(df3, df4, 'attacking_work_rate')

#plot jointplot with goal keeper attributes:
#'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning', 'gk_reflexes'
def joint_plot (df, title) :
    vis=sns.jointplot(x='ball_control',y='gk_diving', data=df, xlim=(0,100), ylim=(0,100), s
tat_func=None)
    plt.title(title, loc='left')
    plt.show()
    plt.close()
    vis=sns.jointplot(x='ball_control',y='gk_handling', data=df, xlim=(0,100), ylim=(0,100),
stat_func=None)
    plt.title(title, loc='left')
    plt.show()
    plt.close()
    vis=sns.jointplot(x='ball_control',y='gk_kicking', data=df, xlim=(0,100), ylim=(0,100),
stat_func=None)
    plt.title(title, loc='left')
    plt.show()
    plt.close()
    vis=sns.jointplot(x='ball_control',y='gk_positioning', data=df, xlim=(0,100), ylim=(0,10
0), stat_func=None)
    plt.title(title, loc='left')
    plt.show()
    plt.close()
    vis=sns.jointplot(x='ball_control',y='gk_reflexes', data=df, xlim=(0,100), ylim=(0,100),
stat_func=None)
    plt.title(title, loc='left')
    plt.show()
    plt.close()


    vis=sns.jointplot(x='marking',y='gk_diving', data=df, xlim=(0,100), ylim=(0,100), stat_f
unc=None)
```

```
gk_diving > 40 (goalkeepers)
```

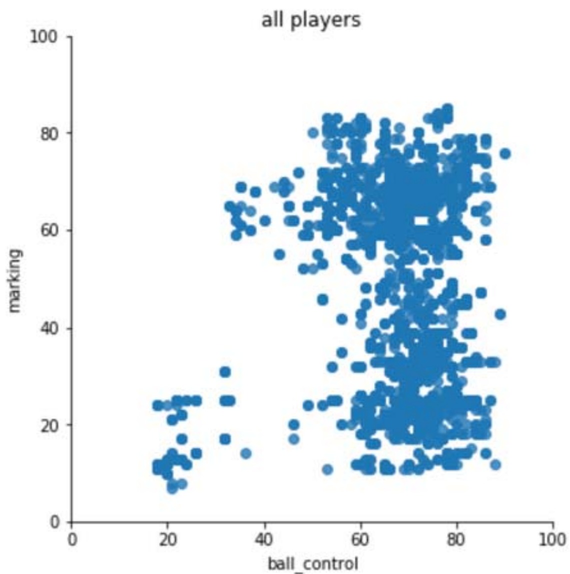


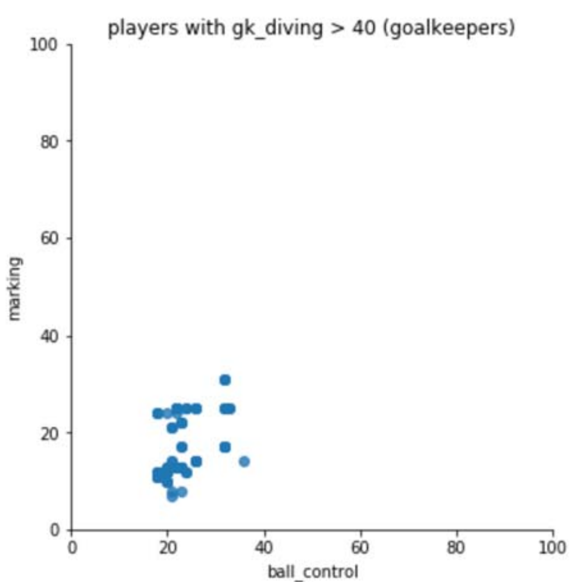


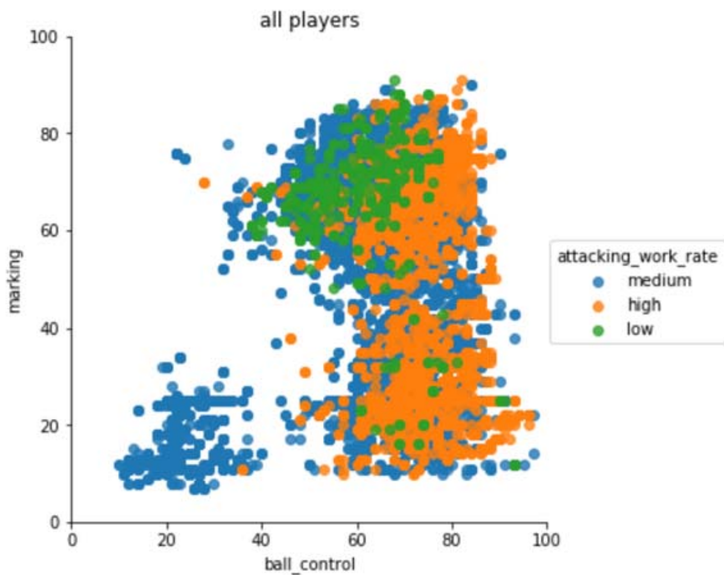


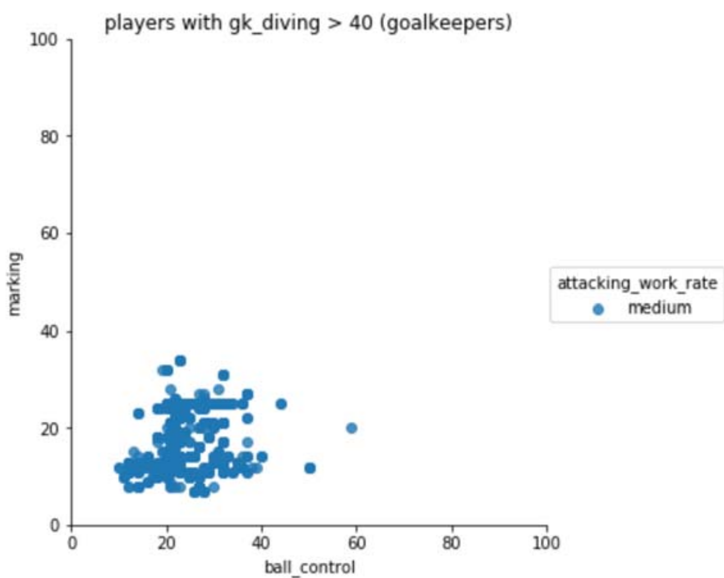


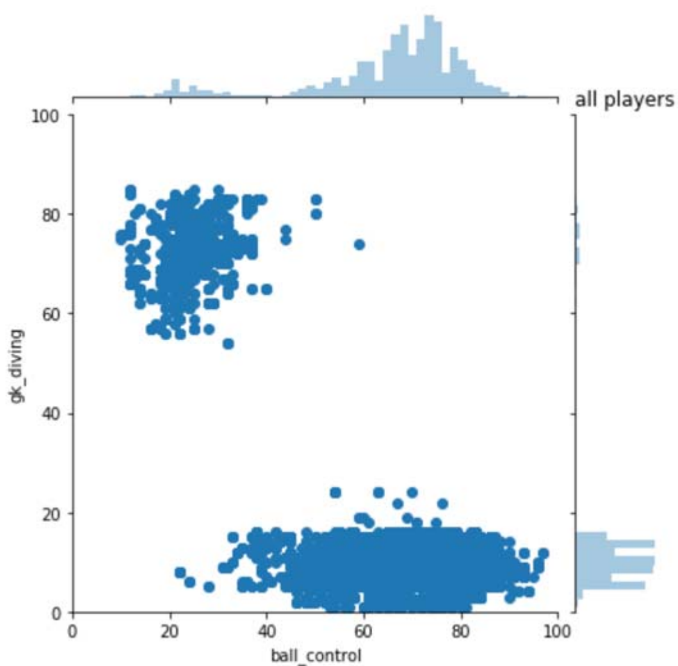

```
plot preferred left foot
```

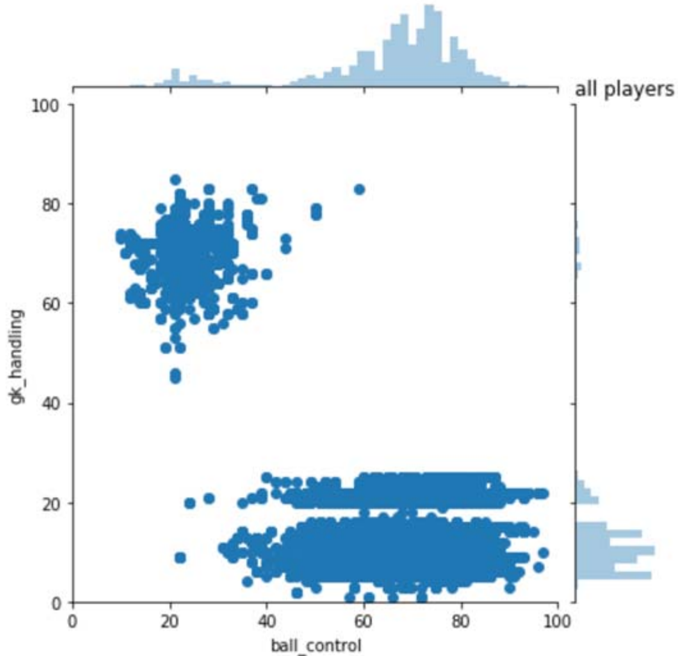


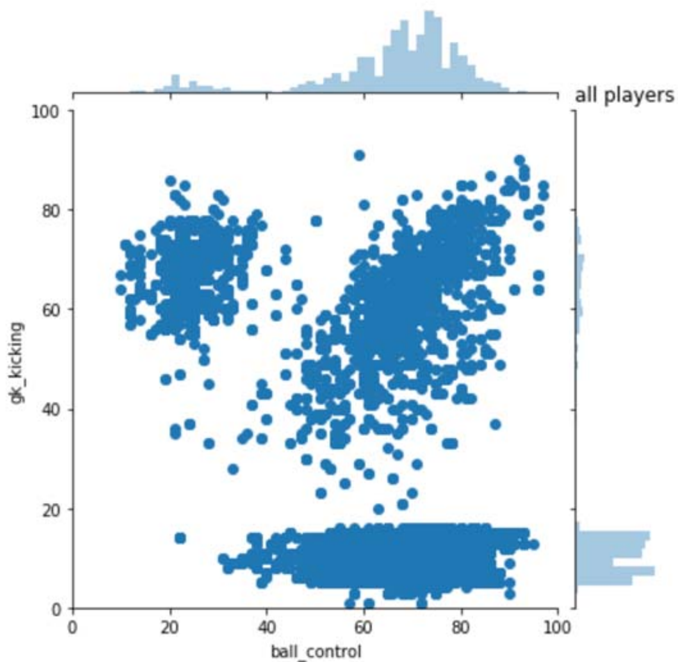


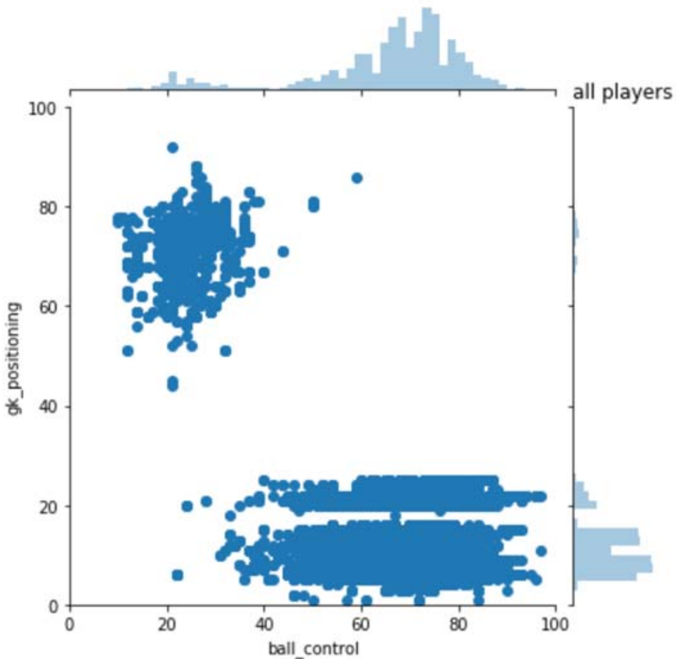

all players
marking
ball_control
attacking_work_rate
medium
high
low
0
20
40
60
80
100

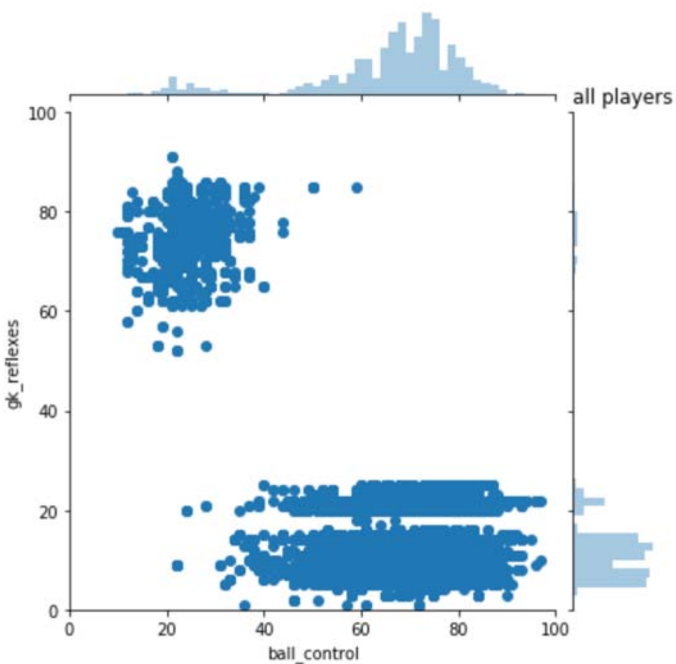

players with gk_diving > 40 (goalkeepers)
marking
ball_control
attacking_work_rate
medium
0
20
40
60
80
100

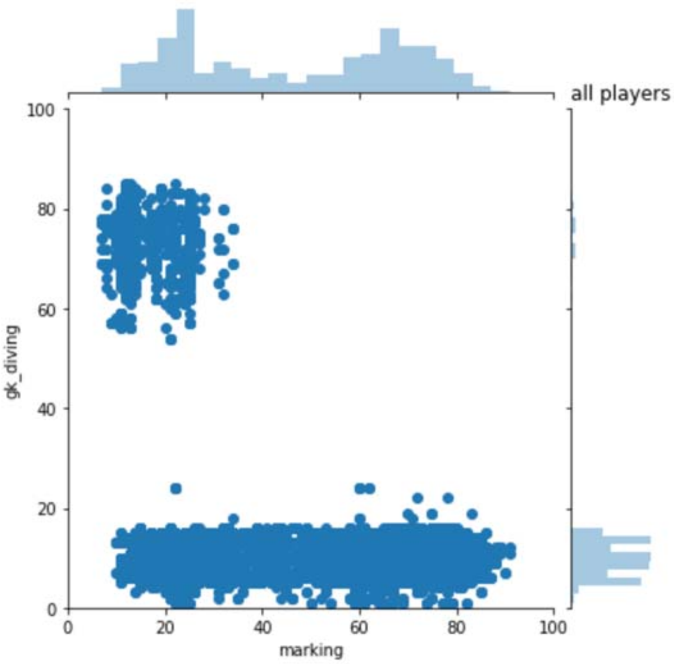

all players
gk_diving
ball_control
0
20
40
60
80
100

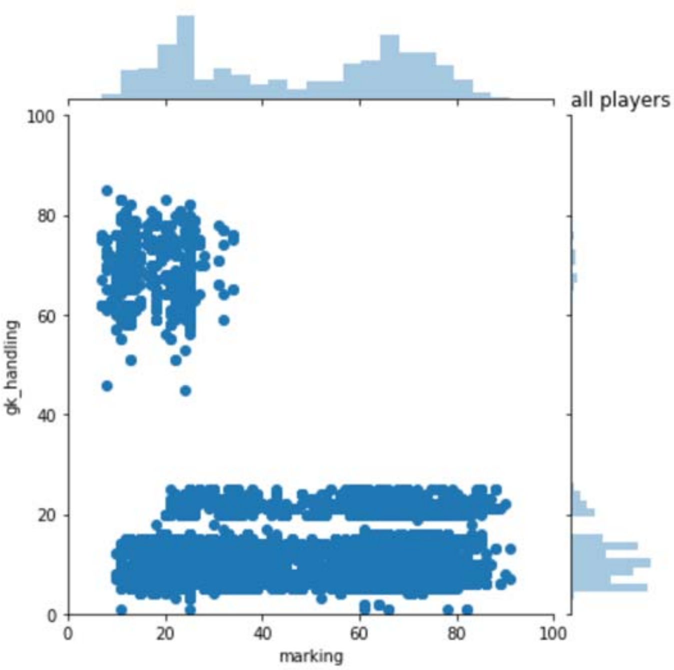

all players
100
80
60
40
20
0
gk_handling
0
20
40
60
80
100
ball_control

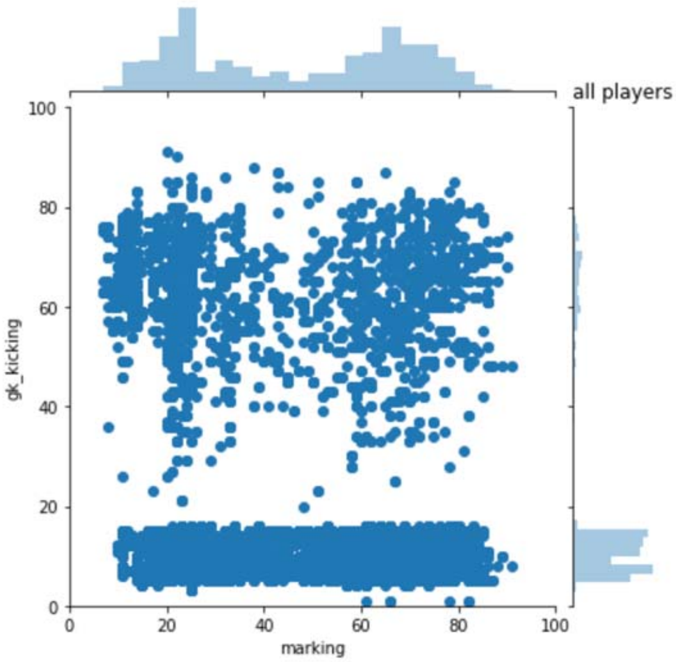

all players
100
80
60
40
20
0
gk_kicking
0
20
40
60
80
100
ball_control

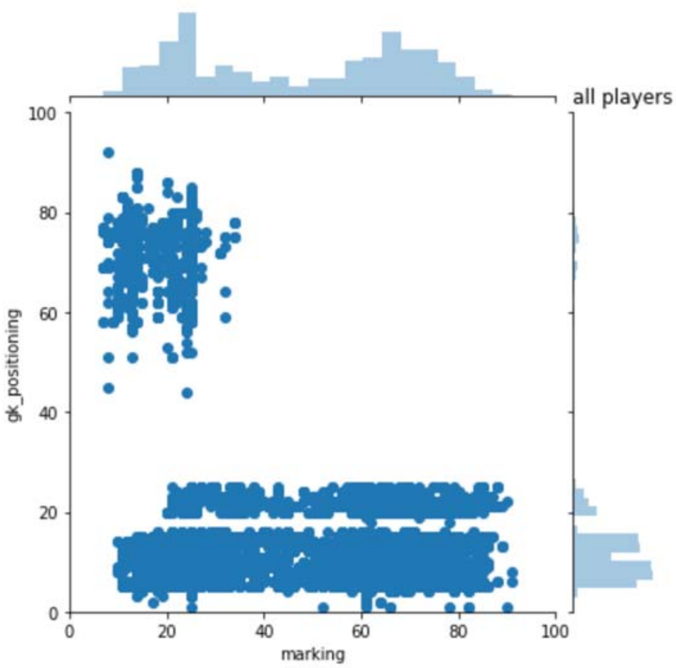

all players
gk_positioning
ball_control
0
20
40
60
80
100

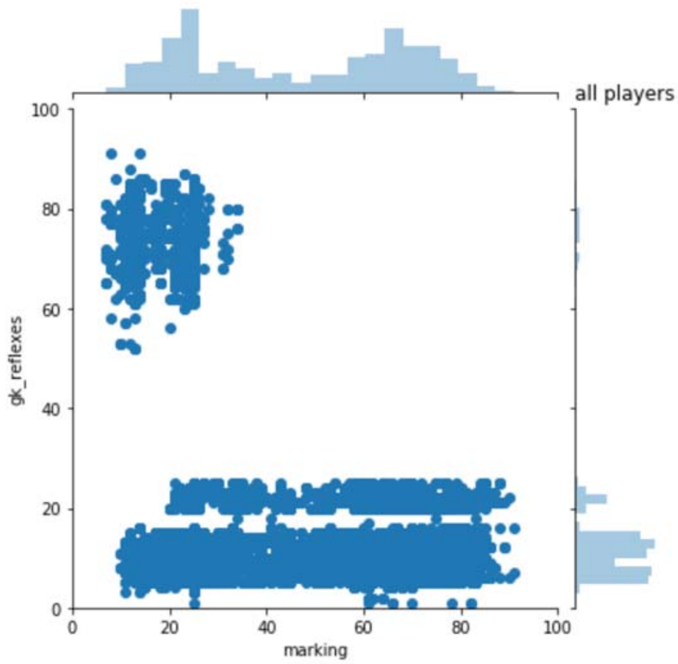

all players
gk_reflexes
ball_control
0
20
40
60
80
100

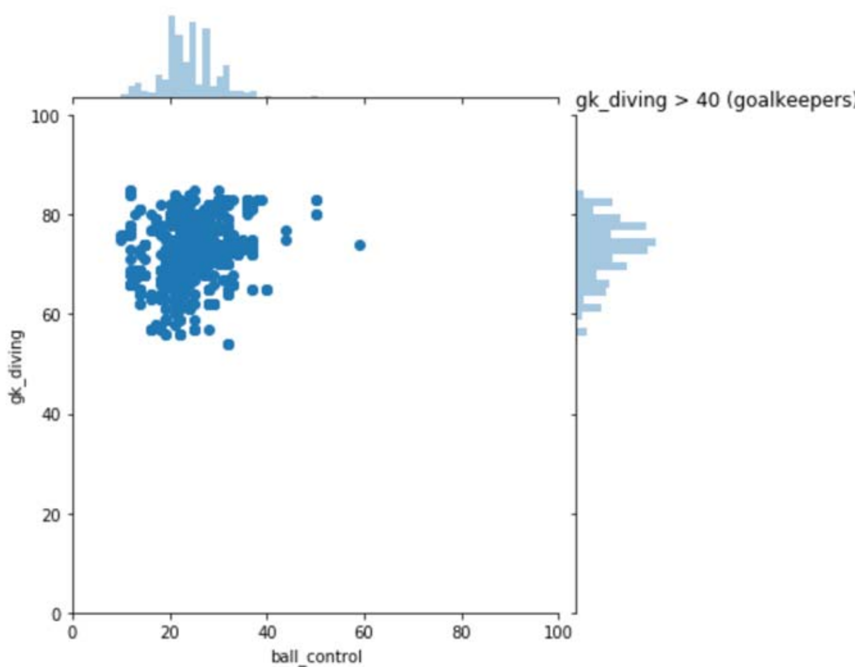

all players
gk_diving
marking
0
20
40
60
80
100

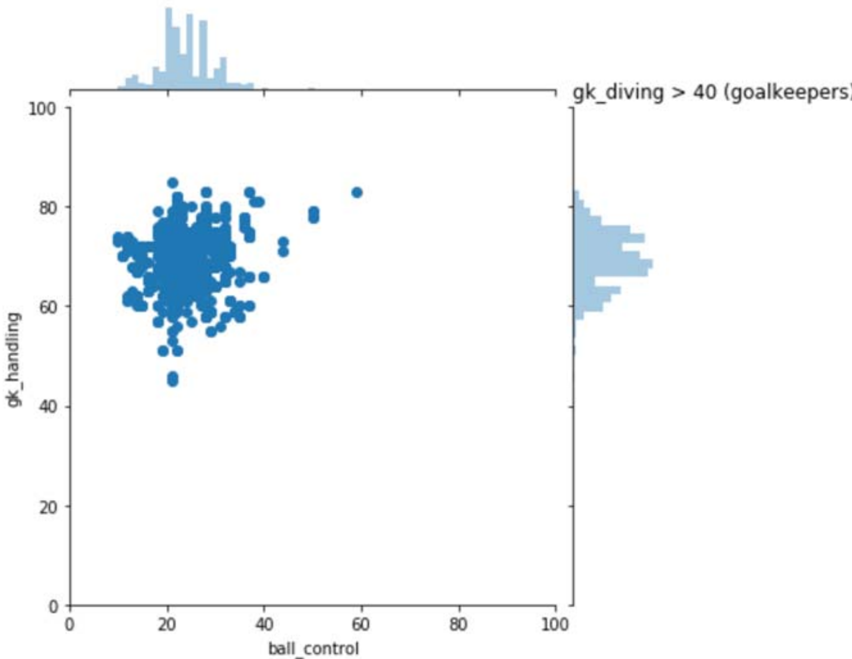

all players
gk_handling
marking
0
20
40
60
80
100

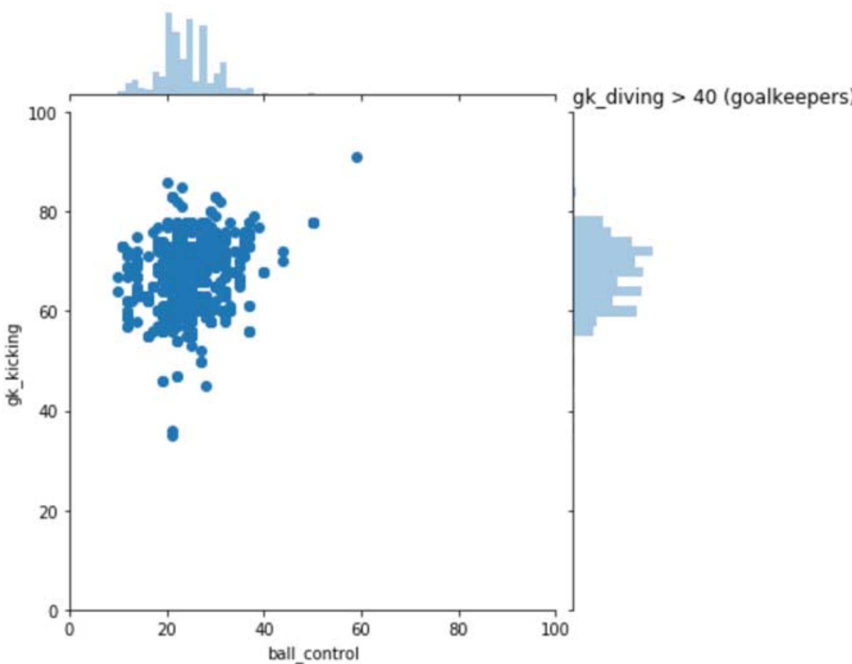

all players
100
80
60
40
20
0
gk_kicking
0
20
40
60
80
100
marking

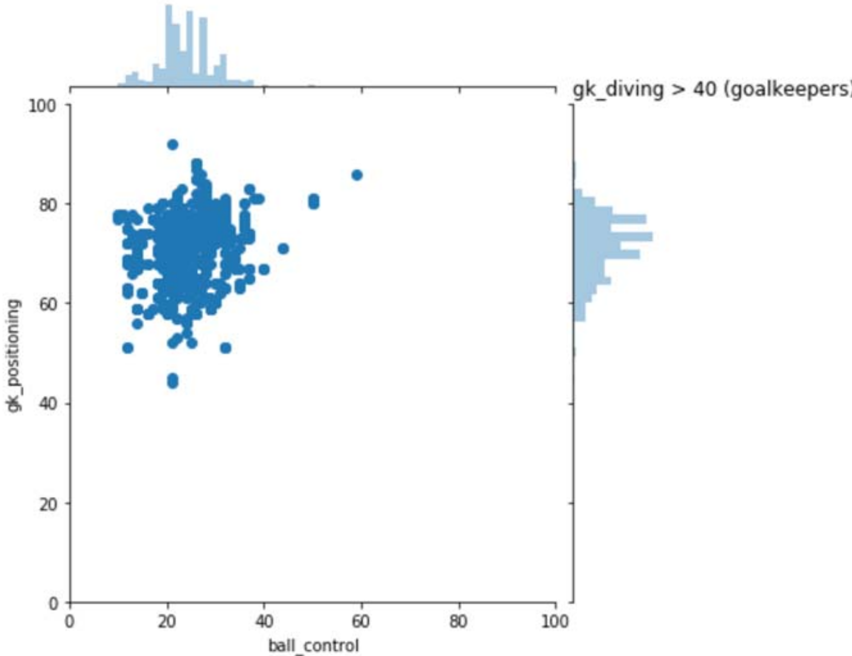

all players
100
80
60
40
20
0
gk_positioning
0
20
40
60
80
100
marking

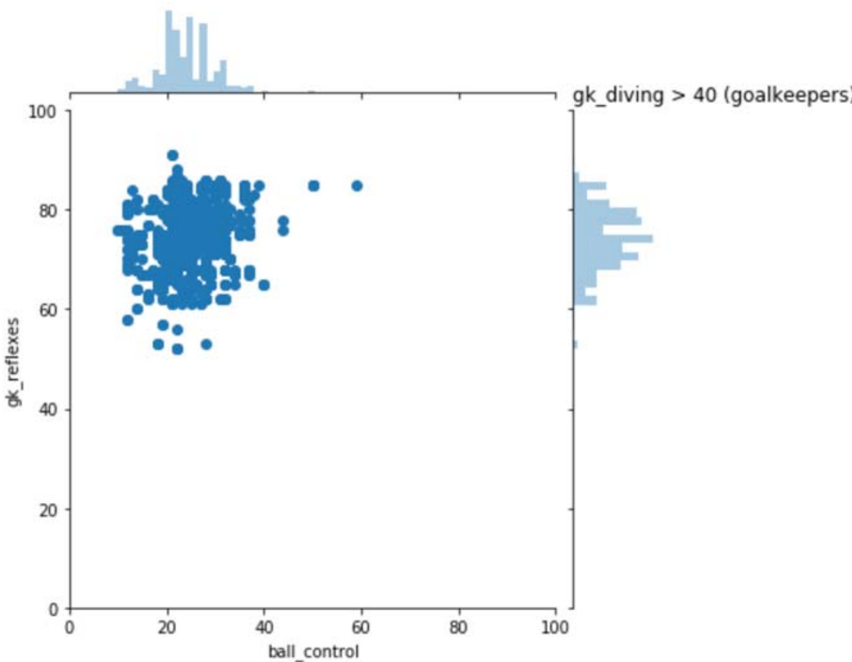

all players
gk_reflexes
marking
0
20
40
60
80
100

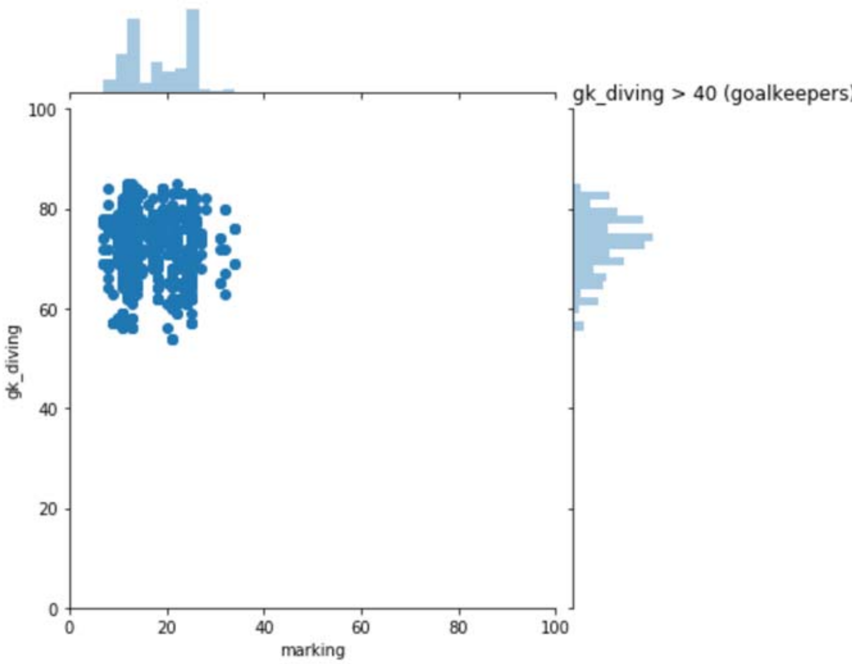

gk_diving > 40 (goalkeepers)
gk_diving
ball_control
0
20
40
60
80
100

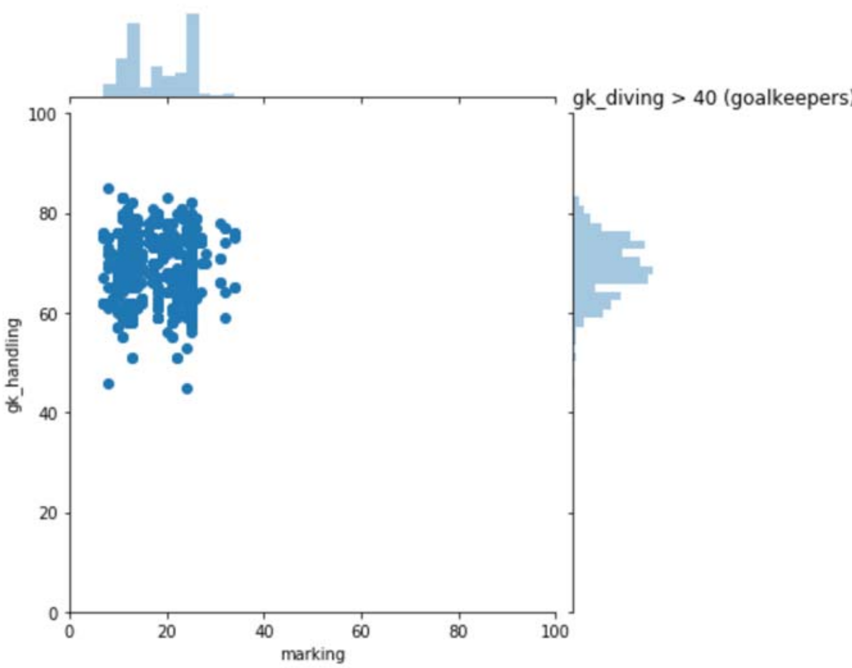

gk_diving > 40 (goalkeepers)
gk_handling
ball_control
0
20
40
60
80
100

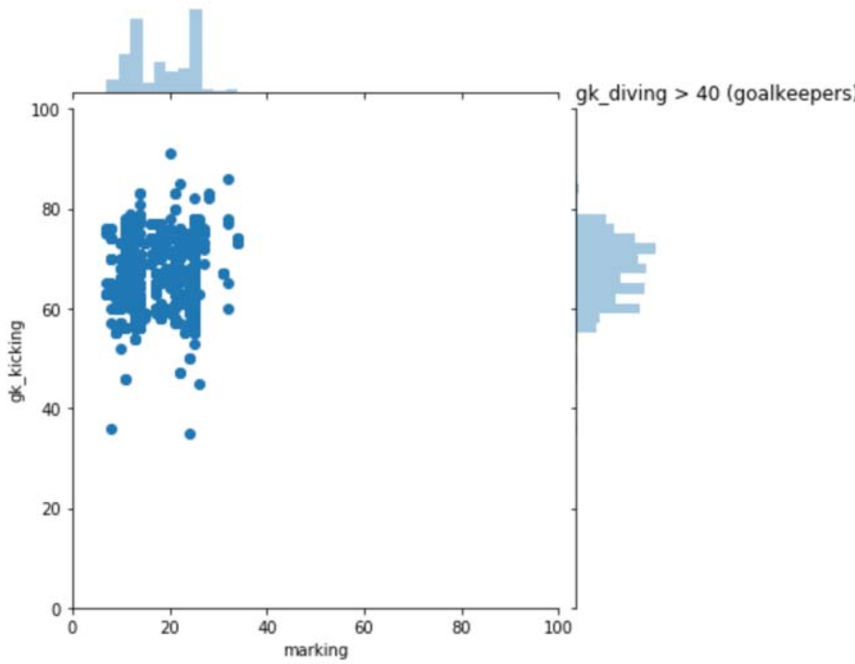

gk_diving > 40 (goalkeepers)
gk_kicking
ball_control
0
20
40
60
80
100

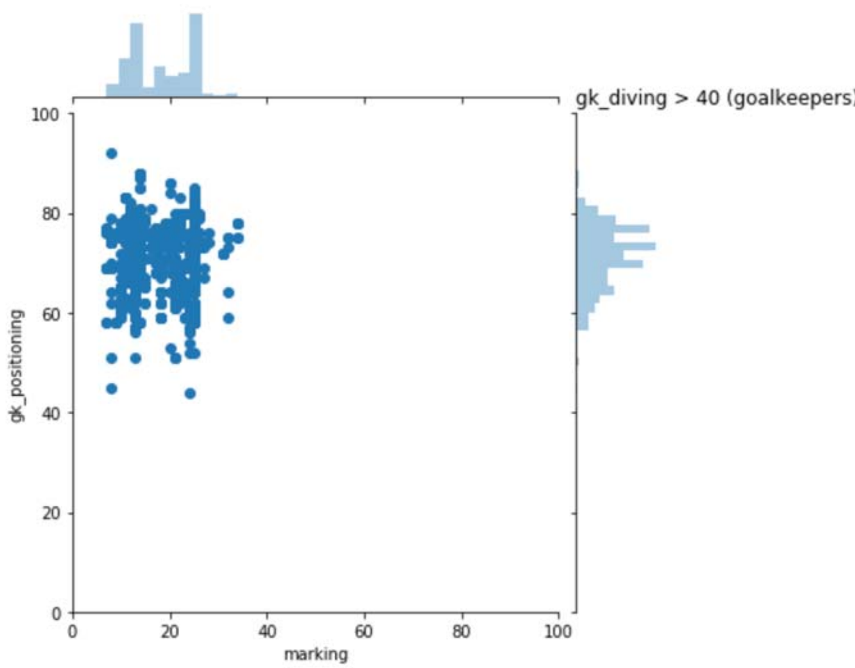

gk_diving > 40 (goalkeepers)
gk_positioning
ball_control
0
20
40
60
80
100

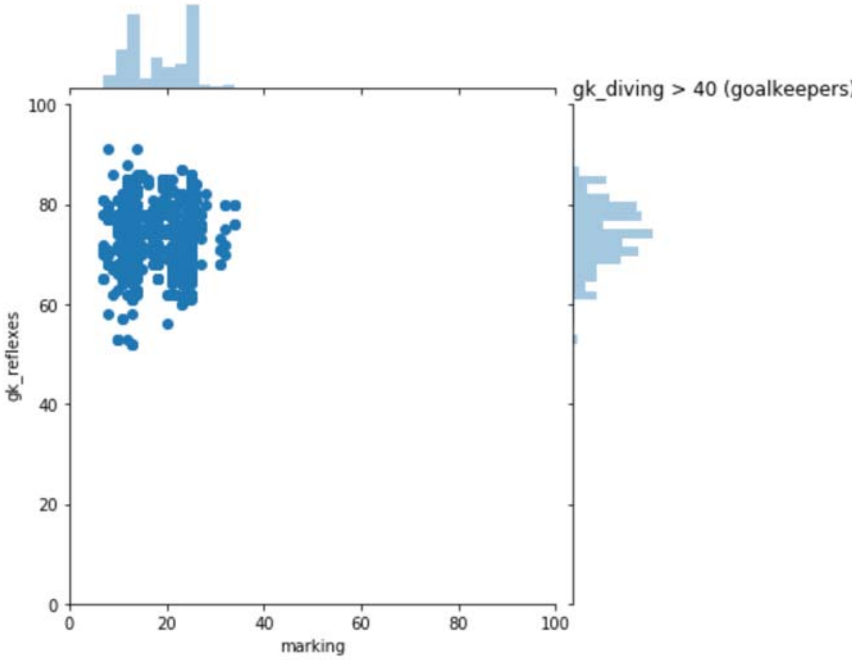

gk_diving > 40 (goalkeepers)
gk_reflexes
ball_control
0
20
40
60
80
100


gk_diving > 40 (goalkeepers)
gk_diving
marking
0
20
40
60
80
100

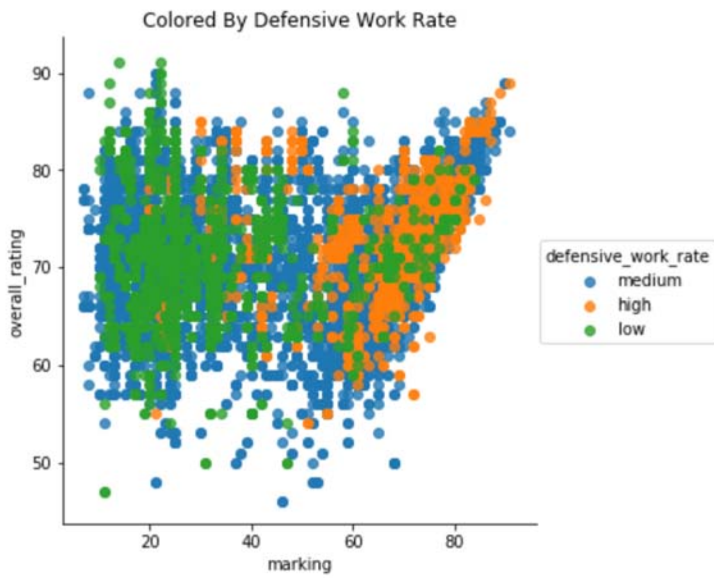

gk_diving > 40 (goalkeepers)
gk_handling
marking
0
20
40
60
80
100

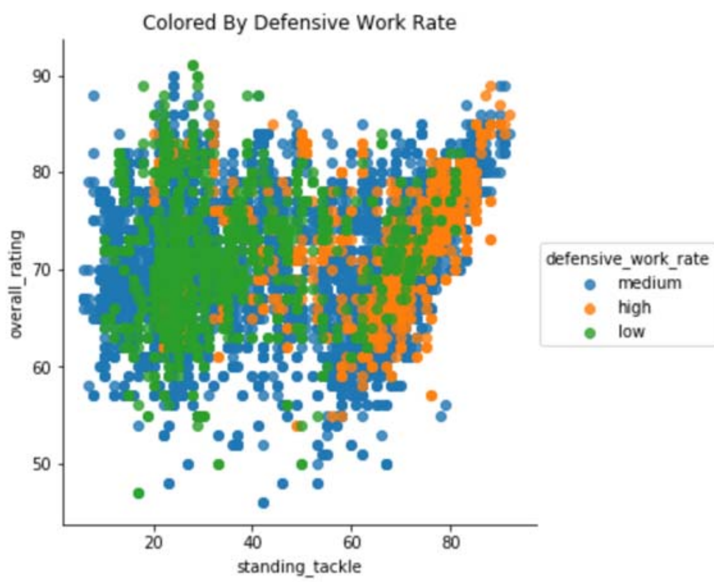

gk_diving > 40 (goalkeepers)
gk_kicking
marking
0
20
40
60
80
100

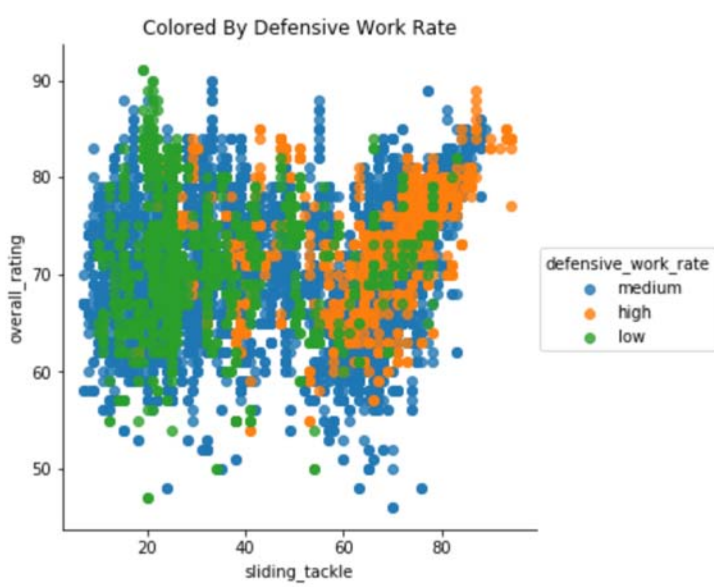

gk_diving > 40 (goalkeepers)
gk_positioning
marking
0
20
40
60
80
100

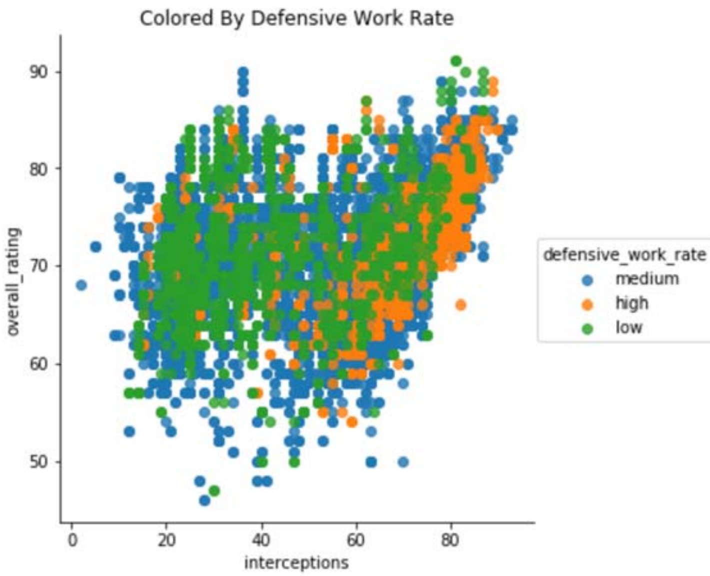

gk_diving > 40 (goalkeepers)
gk_reflexes
marking
0
20
40
60
80
100

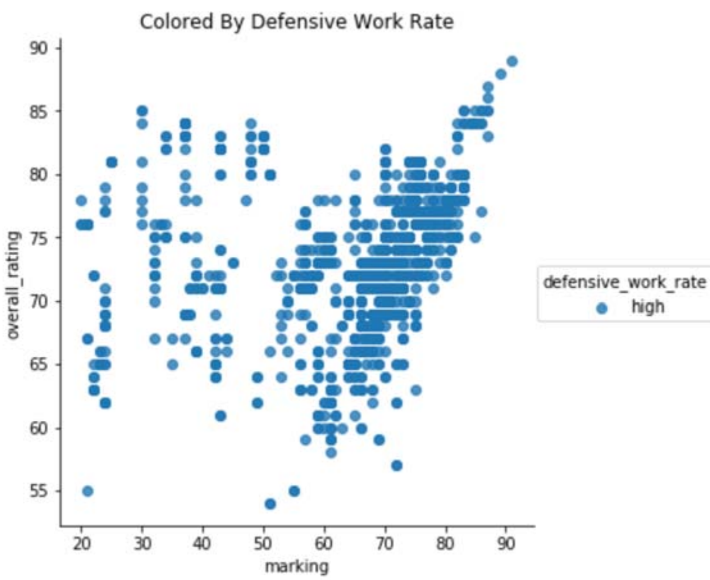
In [42]:
```
import seaborn as sns
df1=df_all_col[df_all_col['defensive_work_rate'].isin (['low','medium','high'])]
df2=df_all_col[df_all_col['defensive_work_rate'].isin (['high'])]
df3=df_all_col[df_all_col['defensive_work_rate'].isin (['medium'])]
df4=df_all_col[df_all_col['defensive_work_rate'].isin (['low'])]

def lmplot (df):
    vis=sns.lmplot(x='marking', y='overall_rating',  hue='defensive_work_rate', sharex=False
, data=df, \
                   scatter=True, fit_reg=False, units=None, order=1, legend=True)
    plt.title('Colored By Defensive Work Rate')
    plt.show()
    plt.close()

    vis=sns.lmplot(x='standing_tackle', y='overall_rating',  hue='defensive_work_rate', shar
ex=False, data=df, \
                   scatter=True, fit_reg=False, units=None, order=1, legend=True)
    plt.title('Colored By Defensive Work Rate')
    plt.show()
    plt.close()

    vis=sns.lmplot(x='sliding_tackle', y='overall_rating',  hue='defensive_work_rate', share
x=False, data=df, \
                   scatter=True, fit_reg=False, units=None, order=1, legend=True)
    plt.title('Colored By Defensive Work Rate')
    plt.show()
    plt.close()

    vis=sns.lmplot(x='interceptions', y='overall_rating',  hue='defensive_work_rate', sharex
=False, data=df, \
                   scatter=True, fit_reg=False, units=None, order=1, legend=True)
    plt.title('Colored By Defensive Work Rate')
    plt.show()
    plt.close()

lmplot(df1) #color by 'defensive_work_rate'].isin (['low','medium','high'])
lmplot(df2) #color by 'defensive_work_rate'].isin (['high'])
lmplot(df3) #color by 'defensive_work_rate'].isin (['medium'])
lmplot(df4) #color by 'defensive_work_rate'].isin (['low'])
```

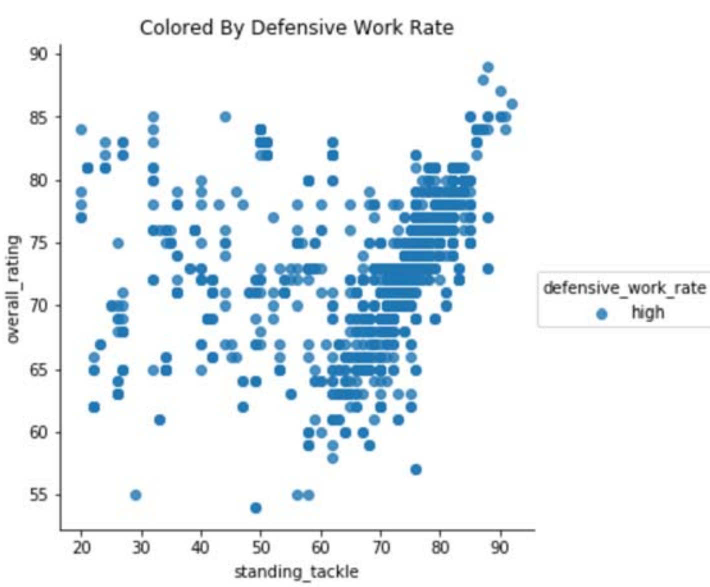

Colored By Defensive Work Rate
overall_rating
marking
defensive_work_rate
medium
high
low
90
80
70
60
50
20
40
60
80

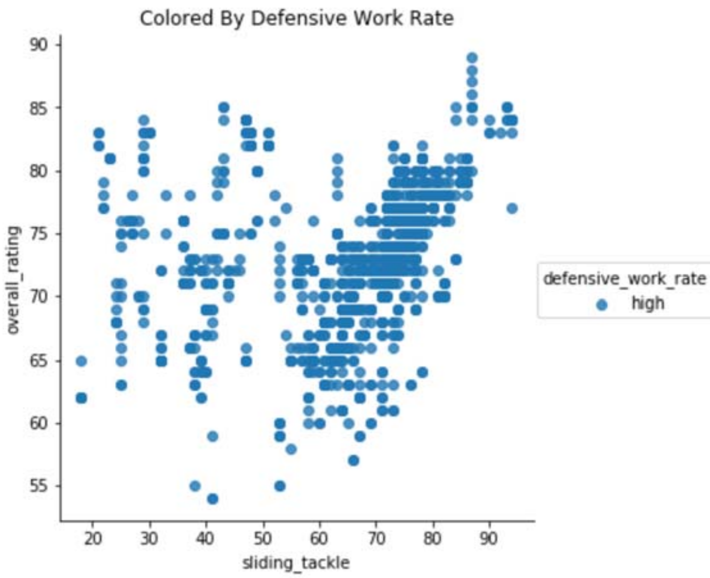

Colored By Defensive Work Rate
overall_rating
standing_tackle
defensive_work_rate
medium
high
low
90
80
70
60
50
20
40
60
80

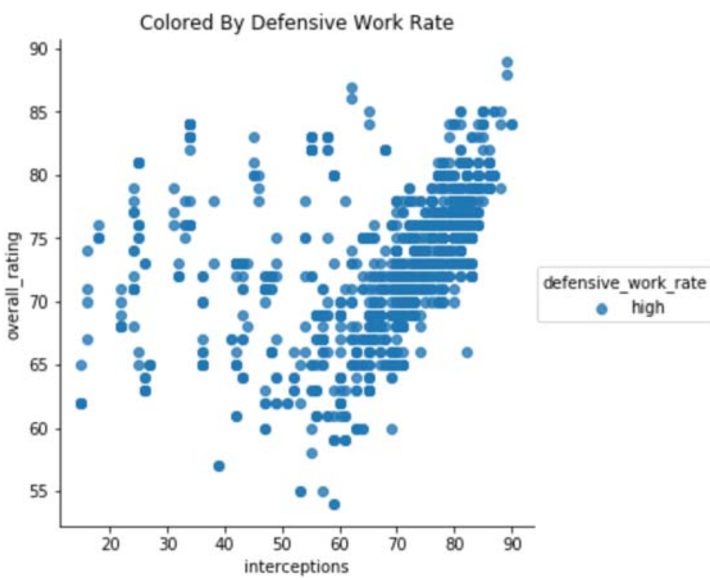

Colored By Defensive Work Rate
overall_rating
sliding_tackle
defensive_work_rate
medium
high
low
90
80
70
60
50
20
40
60
80

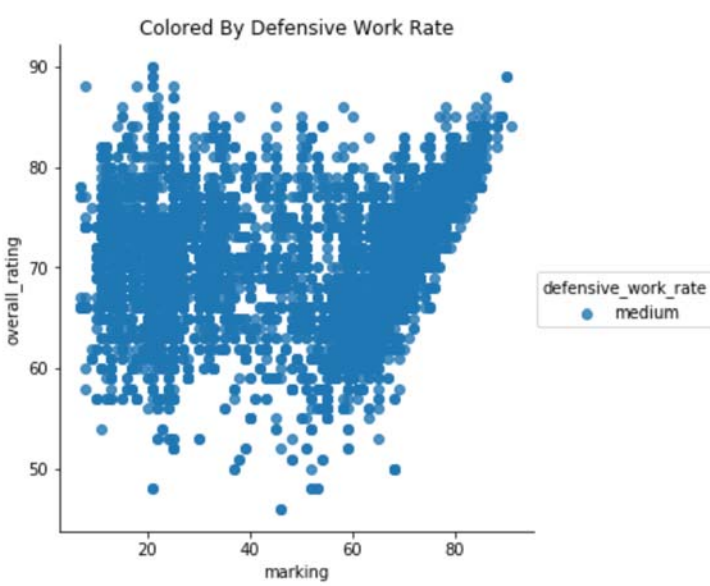

Colored By Defensive Work Rate
overall_rating
interceptions
defensive_work_rate
medium
high
low

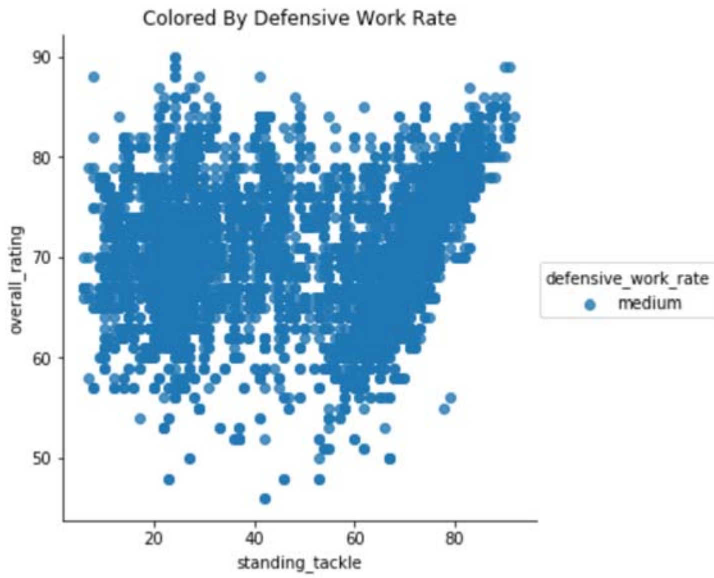

Colored By Defensive Work Rate
overall_rating
marking
defensive_work_rate
high

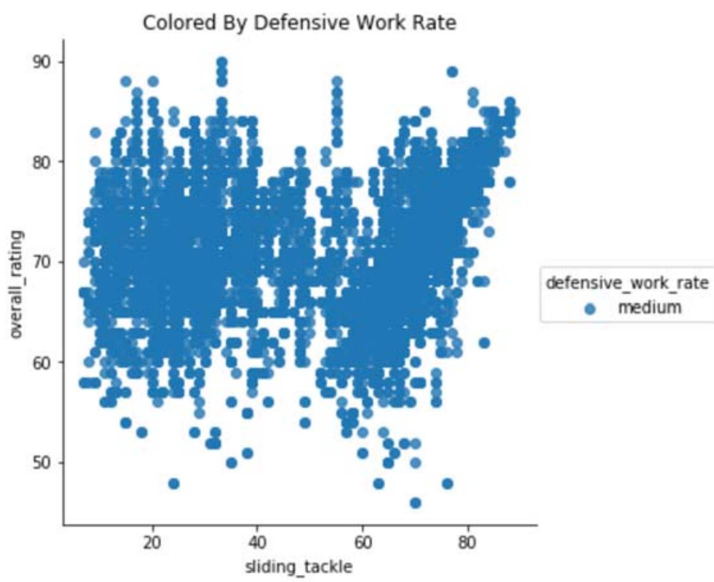

Colored By Defensive Work Rate
overall_rating
standing_tackle
defensive_work_rate
high

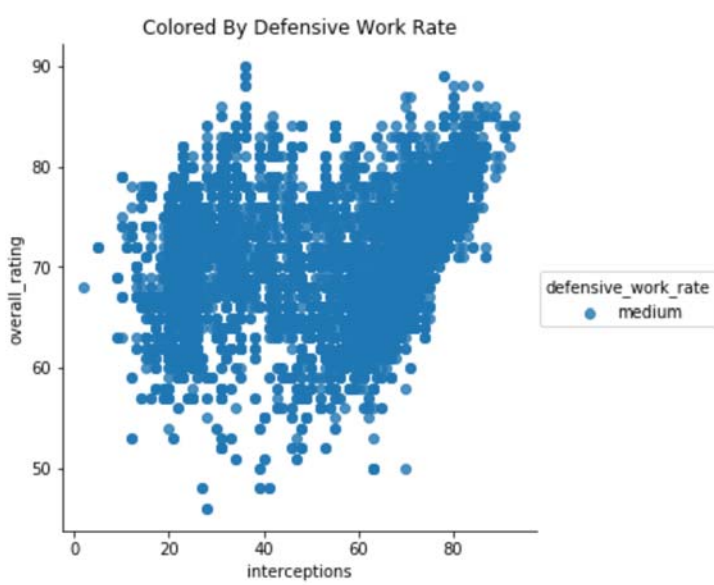

Colored By Defensive Work Rate
overall_rating
sliding_tackle
defensive_work_rate
high

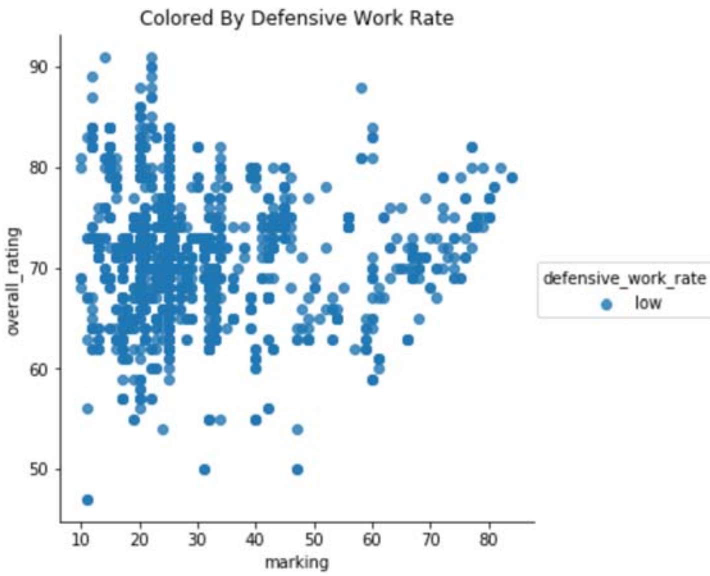

Colored By Defensive Work Rate
overall_rating
interceptions
defensive_work_rate
high

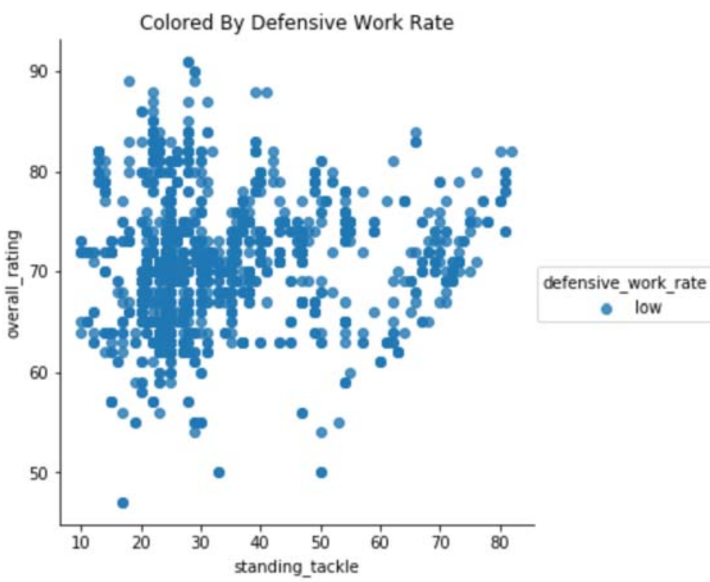

Colored By Defensive Work Rate
overall_rating
marking
defensive_work_rate
medium

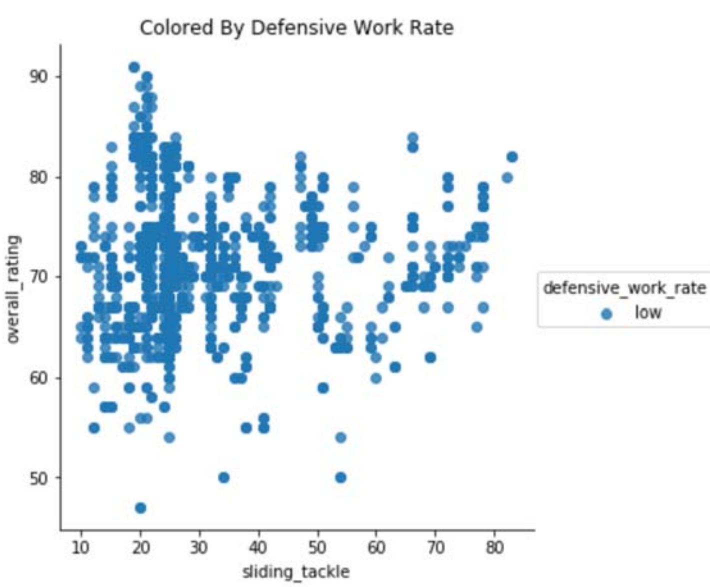

Colored By Defensive Work Rate
90
80
70
60
50
overall_rating
20
40
60
80
standing_tackle
defensive_work_rate
medium

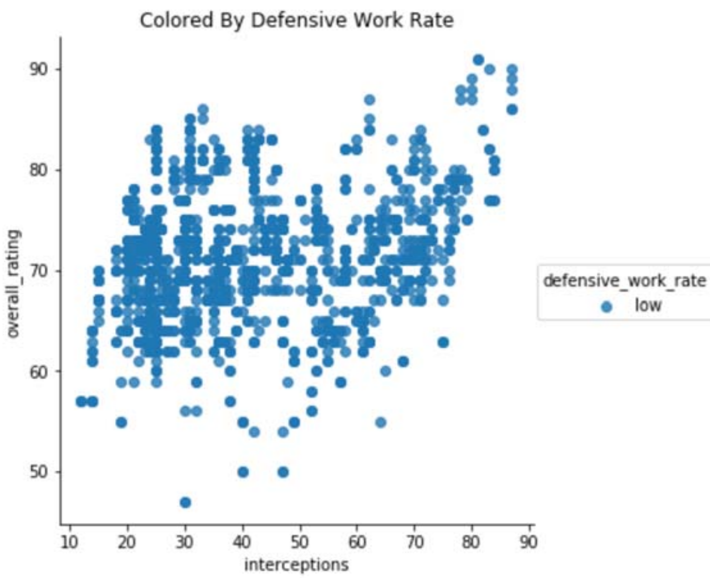

Colored By Defensive Work Rate
90
80
70
60
50
overall_rating
20
40
60
80
sliding_tackle
defensive_work_rate
medium


Colored By Defensive Work Rate
90
80
70
60
50
overall_rating
0
20
40
60
80
interceptions
defensive_work_rate
medium

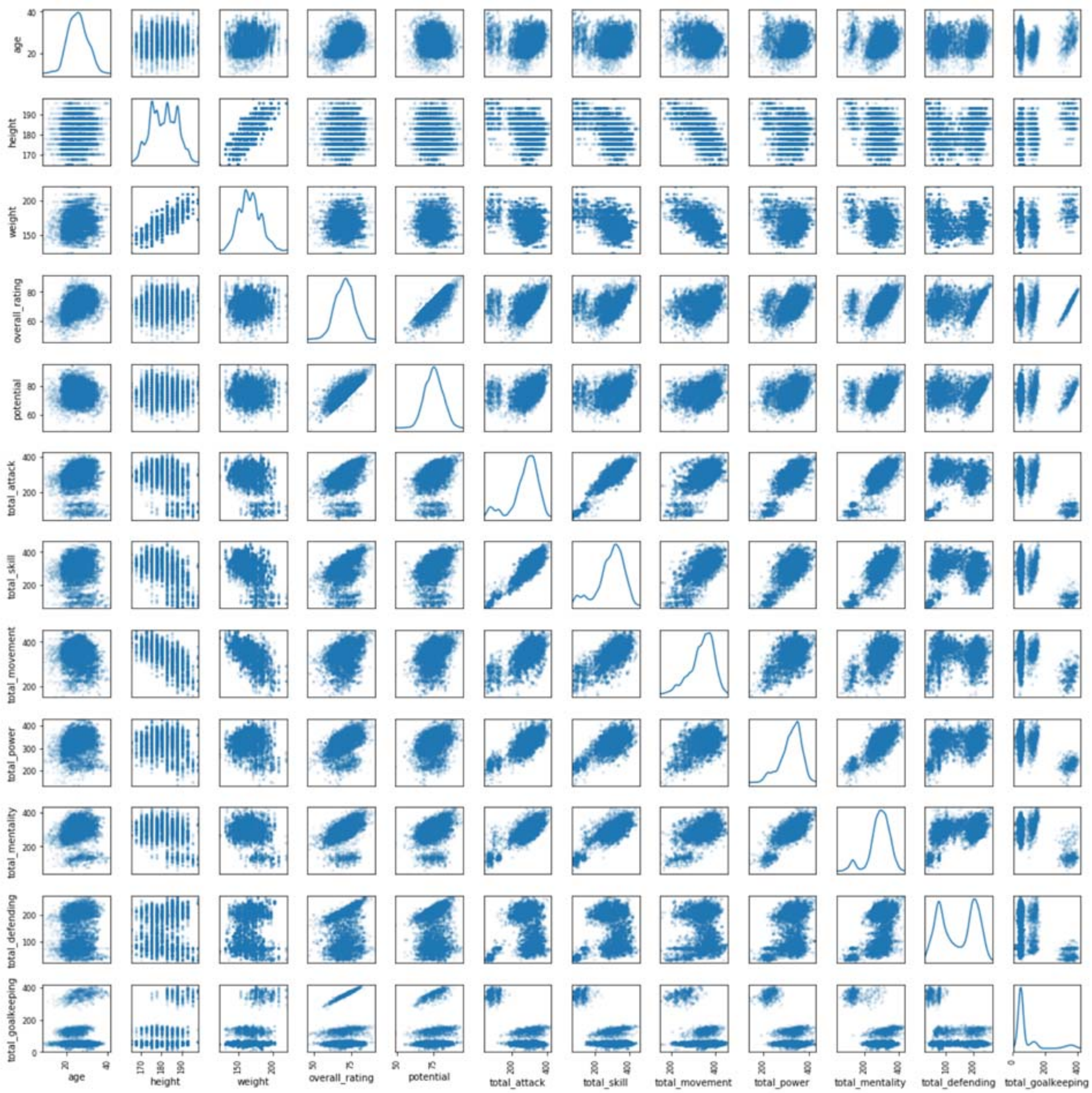

Colored By Defensive Work Rate
overall_rating
90
80
70
60
50
10
20
30
40
50
60
70
80
marking
defensive_work_rate
low


Colored By Defensive Work Rate
overall_rating
90
80
70
60
50
10
20
30
40
50
60
70
80
standing_tackle
defensive_work_rate
low


Colored By Defensive Work Rate
overall_rating
90
80
70
60
50
10
20
30
40
50
60
70
80
sliding_tackle
defensive_work_rate
low

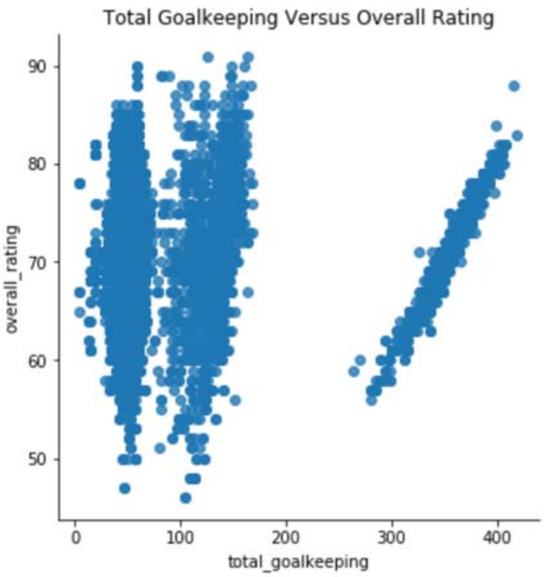

Colored By Defensive Work Rate
overall_rating
interceptions
90
80
70
60
50
10
20
30
40
50
60
70
80
90
defensive_work_rate
low

In [53]:
```
df_totals=df_all_col[numeric_few_col]
pd.plotting.scatter_matrix(df_totals, alpha=0.1, figsize=(16, 16), diagonal='kde',range_padd
ing =0.01)
plt.tight_layout()
plt.show()
plt.close()

total_cols = numeric_few_col + ['player_fifa_api_id', 'player_name']
df_t=df_all_col[total_cols]
df_t.to_csv('player_total_attributes.csv')
print(df_t.shape)
print (df_t.head())
```


age
height
weight
overall_rating
potential
total_attack
total_skill
total_movement
total_power
total_mentality
total_defending
total_goalkeeping

```
(10898, 15)
       age  height  weight  overall_rating  potential  total_attack  \
1045  31.0   177.8     165            73.0       75.0         311.0
1046  31.0   177.8     165            72.0       75.0         307.0
1047  30.0   177.8     165            73.0       75.0         305.0
1048  28.0   177.8     165            73.0       75.0         298.0
1049  28.0   177.8     165            70.0       72.0         288.0

      total_skill  total_movement  total_power  total_mentality  \
1045        319.0           375.0        355.0            351.0
1046        318.0           375.0        355.0            372.0
1047        316.0           377.0        357.0            372.0
1048        311.0           372.0        355.0            370.0
1049        335.0           366.0        346.0            356.0

      total_defending  total_goalkeeping  player_fifa_api_id  \
1045            219.0               42.0               17880
1046            216.0              143.0               17880
1047            221.0              143.0               17880
1048            221.0              141.0               17880
1049            212.0              102.0               17880

      player_fifa_api_id player_name
1045               17880        Abel
1046               17880        Abel
1047               17880        Abel
1048               17880        Abel
1049               17880        Abel
```

In [54]:
```
vis=sns.lmplot(x='total_goalkeeping', y='overall_rating',  hue = None, sharex=False, data=df
_totals, \
               scatter=True, fit_reg=False, units=None, order=1, legend=True)
plt.title('Total Goalkeeping Versus Overall Rating')
plt.show()
plt.close()

df_totals_gk = df_totals[df_totals['total_goalkeeping'] > 200]
corr_gk=df_totals_gk[['total_goalkeeping','overall_rating']].corr()
print ('A Closer Look at the Goalkeeper Subgroup on the Far Right')
print(' total_goalkeeping > 200 ')
print(' ')
print ('Correlation Coefficient between Overall Rating and Total Goalkeeping Attribute')
print(corr_gk)
```




```
A Closer Look at the Goalkeeper Subgroup on the Far Right
 total_goalkeeping > 200

Correlation Coefficient between Overall Rating and Total Goalkeeping Attribute
                   total_goalkeeping  overall_rating
total_goalkeeping           1.000000        0.978269
overall_rating              0.978269        1.000000
```

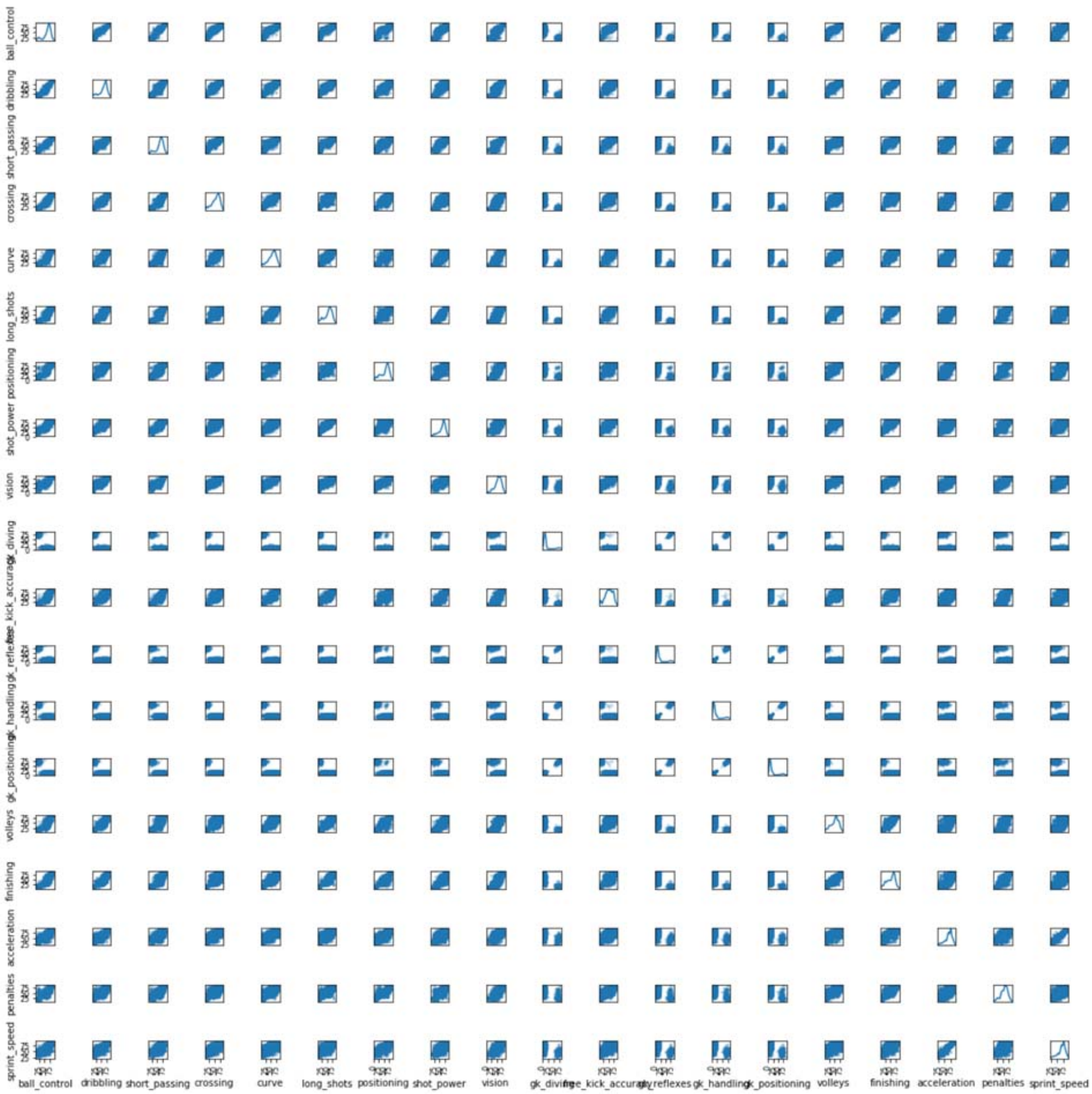
Interpret correlation coefficient

Exactly –1. A perfect downhill (negative) linear relationship
• –0.70. A strong downhill (negative) linear relationship
• –0.50. A moderate downhill (negative) relationship
• –0.30. A weak downhill (negative) linear relationship
• 0. No linear relationship

In [55]:
```
# save correlation coefficient for dataset to csv
df_corr = df_unscaled_data.corr()
df_corr.to_csv('df_corr.csv')
```

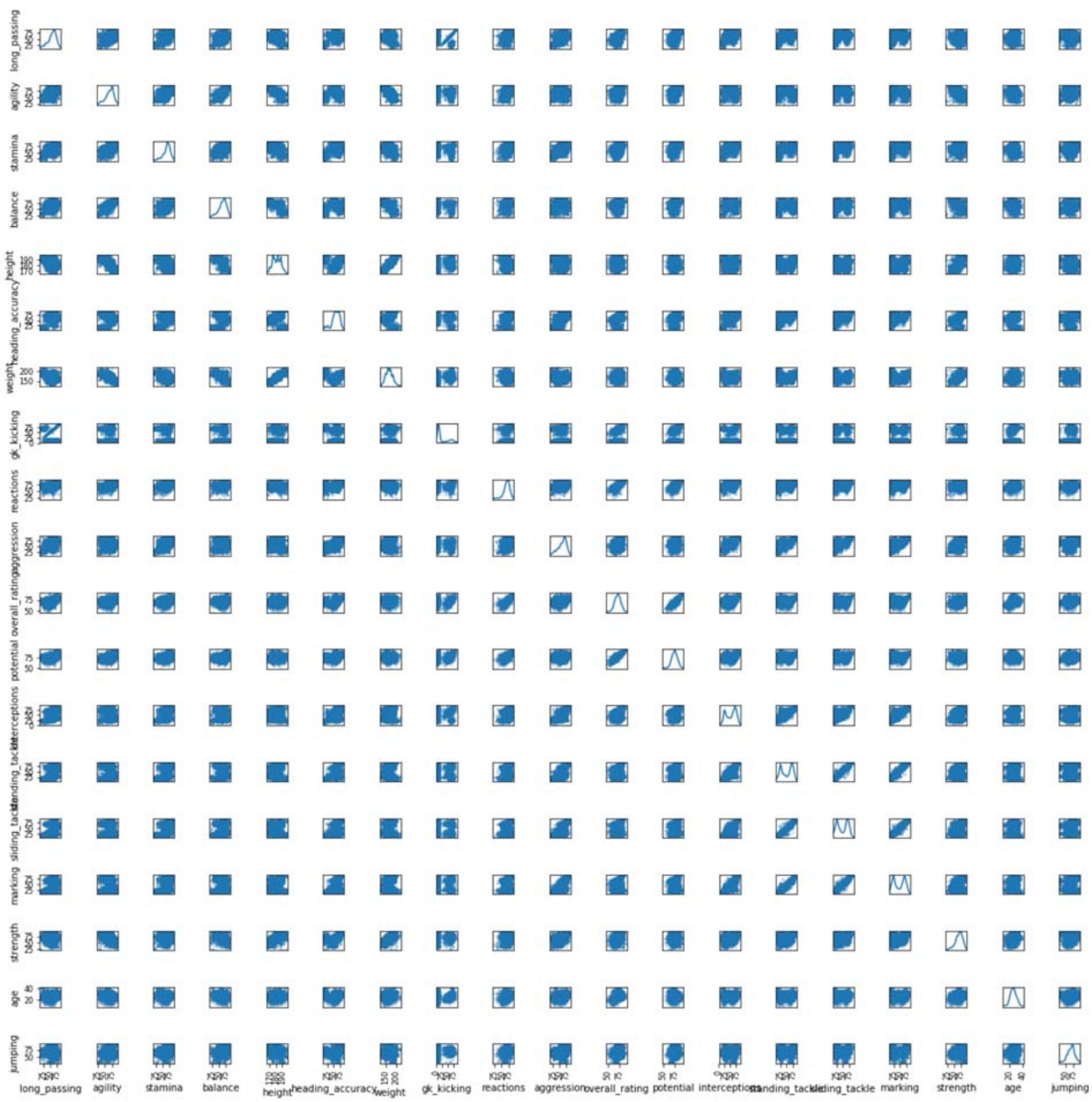
In [31]:
```
print('Unscaled Data Scatter Matrix 1')
print('PC1 sorted components: ', loading_scores_PC1_sorted.index[0:19])
#col_of_interest = ['ball_control', 'dribbling', 'short_passing', 'crossing', 'curve','long_
shots', 'positioning', 'shot_power', 'vision', 'gk_diving','free_kick_accuracy', 'gk_reflexe
s', 'gk_handling']
col_of_interest = loading_scores_PC1_sorted.index[0:19]
df_col_of_interest= df_unscaled_data[col_of_interest] # scatter matrix for columns of intere
st


pd.plotting.scatter_matrix(df_col_of_interest, alpha=0.1, figsize=(16, 16), diagonal='kde',r
ange_padding =0.01)
plt.tight_layout()
plt.show()
plt.close()
```

```
Unscaled Data Scatter Matrix 1
PC1 sorted components:  Index(['ball_control', 'dribbling', 'short_passing', 'crossing', '
curve',
       'long_shots', 'positioning', 'shot_power', 'vision', 'gk_diving',
       'free_kick_accuracy', 'gk_reflexes', 'gk_handling', 'gk_positioning',
       'volleys', 'finishing', 'acceleration', 'penalties', 'sprint_speed'],
      dtype='object')
```



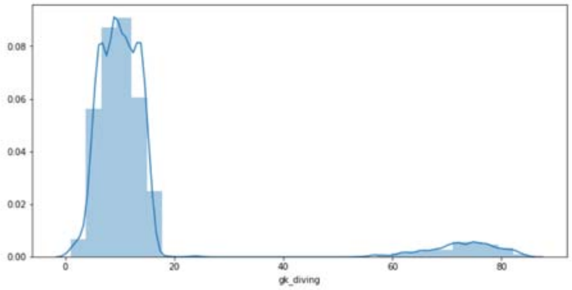
In [39]:
```
print('Unscaled Data Scatter Matrix 2')
print('PC1 sorted components: ', loading_scores_PC1_sorted.index[19:38])
#col_of_interest = ['ball_control', 'dribbling', 'short_passing', 'crossing', 'curve','long_
shots', 'positioning', 'shot_power', 'vision', 'gk_diving','free_kick_accuracy', 'gk_reflexe
s', 'gk_handling']
col_of_interest = loading_scores_PC1_sorted.index[19:38]
df_col_of_interest= df_unscaled_data[col_of_interest] # scatter matrix for columns of intere
st

pd.plotting.scatter_matrix(df_col_of_interest, alpha=0.1, figsize=(16, 16), diagonal='kde',r
ange_padding =0.01)
plt.tight_layout()
plt.show()
plt.close()
```

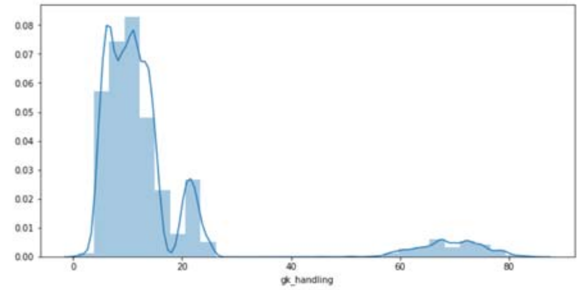
```
Unscaled Data Scatter Matrix 2
PC1 sorted components:  Index(['long_passing', 'agility', 'stamina', 'balance', 'height',
       'heading_accuracy', 'weight', 'gk_kicking', 'reactions', 'aggression',
       'overall_rating', 'potential', 'interceptions', 'standing_tackle',
       'sliding_tackle', 'marking', 'strength', 'age', 'jumping'],
      dtype='object')
```

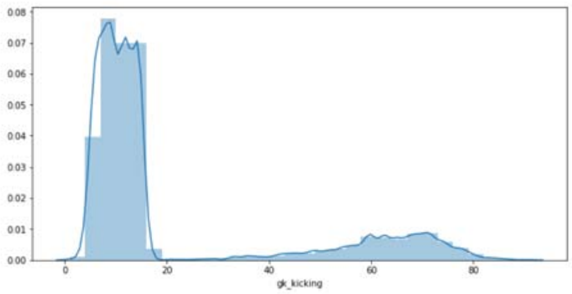


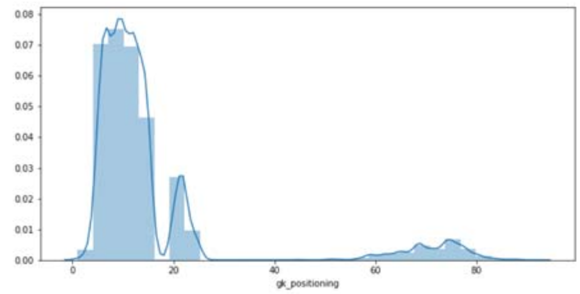
In [36]:
```
# create distribution plot for all features

final_col = ['player_fifa_api_id','preferred_foot','attacking_work_rate', 'defensive_work_ra
te'] + numeric_col
print(final_col)
df_final=df_all_col[final_col]
'''
df_final=def_all_col['player_fifa_api_id', 'preferred_foot', 'attacking_work_rate', 'defensi
ve_work_rate', 'age', \
                     'height', 'weight', 'overall_rating', 'potential', 'crossing', 'finishi
ng', 'heading_accuracy', \
                     'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy',
'long_passing', \
                     'ball_control', 'acceleration', 'sprint_speed', 'agility', 'reactions',
'balance', 'shot_power', \
                     'jumping', 'stamina', 'strength', 'long_shots', 'aggression', 'intercep
tions', 'positioning', \
                     'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle',
'gk_diving', 'gk_handling', \
                     'gk_kicking', 'gk_positioning', 'gk_reflexes']
'''
print(df_final.head())
print(len(final_col))
df_final.to_csv("df_final.csv")

# distplot for goalkeeping attributes
fig = plt.figure(figsize=(24,18))
ax1 = fig.add_subplot(321)
ax2 = fig.add_subplot(322)
ax3 = fig.add_subplot(323)
ax4 = fig.add_subplot(324)
ax5 = fig.add_subplot(325)
vis1=sns.distplot (df_all_col['gk_diving'], bins=30, ax=ax1)
vis2=sns.distplot (df_all_col['gk_handling'], bins=30, ax=ax2)
vis3=sns.distplot (df_all_col['gk_kicking'], bins=30, ax=ax3)
vis4=sns.distplot (df_all_col['gk_positioning'], bins=30, ax=ax4)
vis5=sns.distplot (df_all_col['gk_reflexes'], bins=30, ax=ax5)
plt.show()
plt.close()

# distplot for defending attributes
fig = plt.figure(figsize=(18,5))
ax6 = fig.add_subplot(131)
ax7 = fig.add_subplot(132)
ax8 = fig.add_subplot(133)
vis6=sns.distplot (df_all_col['marking'], bins=30, ax=ax6)
vis7=sns.distplot (df_all_col['standing_tackle'], bins=30, ax=ax7)
vis8=sns.distplot (df_all_col['sliding_tackle'], bins=30, ax=ax8)
plt.show()
plt.close()

#  distplot
for i in range (4,42) :
    sns.distplot (df_all_col[final_col[i]], bins=30)
    plt.title('Distribution Plot')
    plt.show()
    plt.close()
```

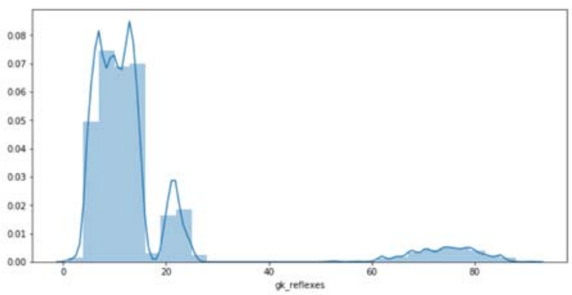
```
['player_fifa_api_id', 'preferred_foot', 'attacking_work_rate', 'defensive_work_rate', 'ag
e', 'height', 'weight', 'overall_rating', 'potential', 'crossing', 'finishing', 'heading_a
ccuracy', 'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy', 'long_pa
ssing', 'ball_control', 'acceleration', 'sprint_speed', 'agility', 'reactions', 'balance',
'shot_power', 'jumping', 'stamina', 'strength', 'long_shots', 'aggression', 'interceptions
', 'positioning', 'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle', '
gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning', 'gk_reflexes']
      player_fifa_api_id  player_fifa_api_id preferred_foot  \
1045               17880               17880          right
1046               17880               17880          right
1047               17880               17880          right
1048               17880               17880          right
1049               17880               17880          right

     attacking_work_rate defensive_work_rate   age  height  weight  \
1045                None                   o  31.0   177.8     165
1046                None                   o  31.0   177.8     165
1047                None                   o  30.0   177.8     165
1048                None                   o  28.0   177.8     165
1049                None                   o  28.0   177.8     165

      overall_rating  potential     ...       vision  penalties  marking  \
1045            73.0       75.0     ...         75.0       66.0     73.0
1046            72.0       75.0     ...         75.0       75.0     72.0
1047            73.0       75.0     ...         75.0       75.0     74.0
1048            73.0       75.0     ...         75.0       76.0     74.0
1049            70.0       72.0     ...         75.0       83.0     70.0

      standing_tackle  sliding_tackle  gk_diving  gk_handling  gk_kicking  \
1045             74.0            72.0        7.0          5.0         7.0
1046             72.0            72.0        9.0         20.0        74.0
1047             75.0            72.0        9.0         20.0        74.0
1048             75.0            72.0        9.0         20.0        72.0
1049             70.0            72.0        9.0         10.0        63.0

      gk_positioning  gk_reflexes
1045            13.0         10.0
1046            20.0         20.0
1047            20.0         20.0
1048            20.0         20.0
1049             8.0         12.0

[5 rows x 43 columns]
42
```

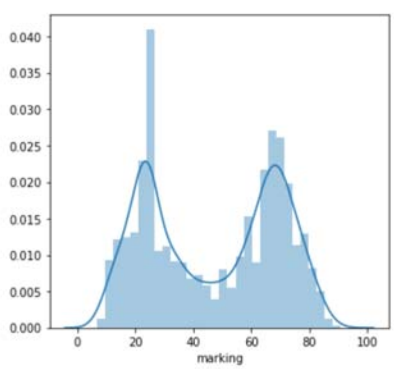

gk_diving

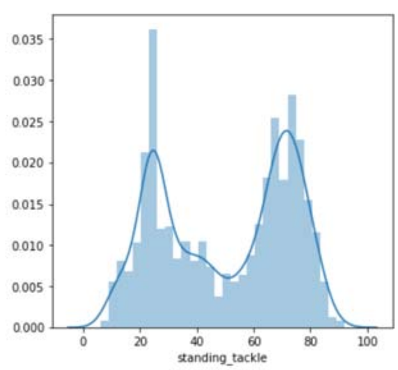

gk_handling

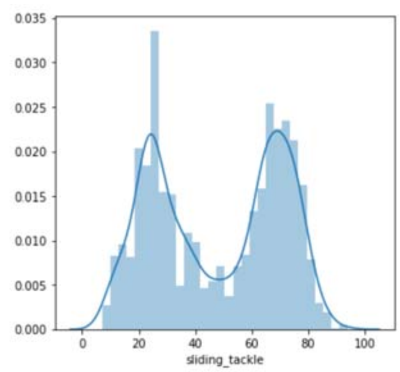

gk_kicking

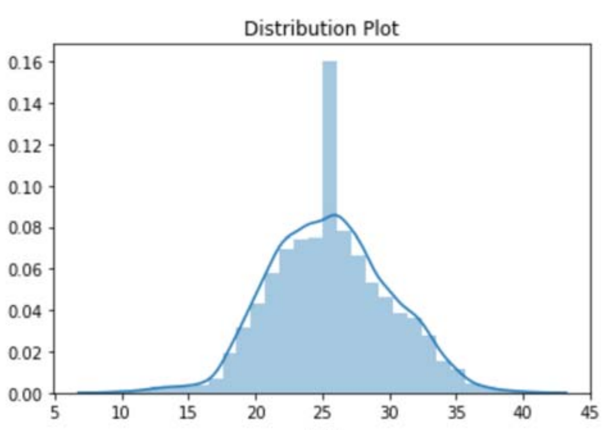

gk_positioning

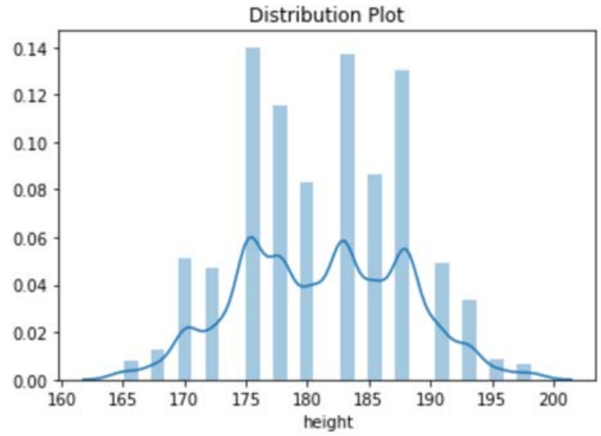

gk_reflexes

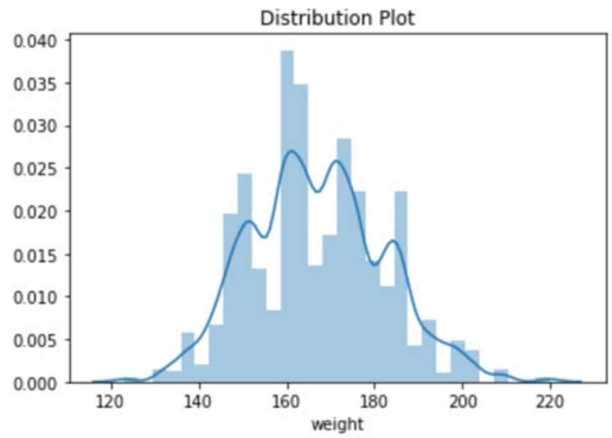

marking

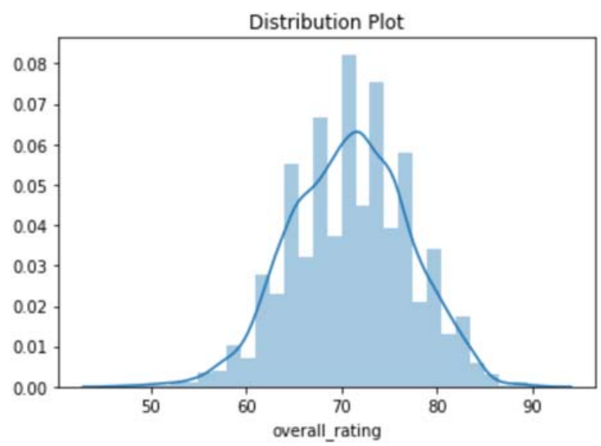

standing_tackle

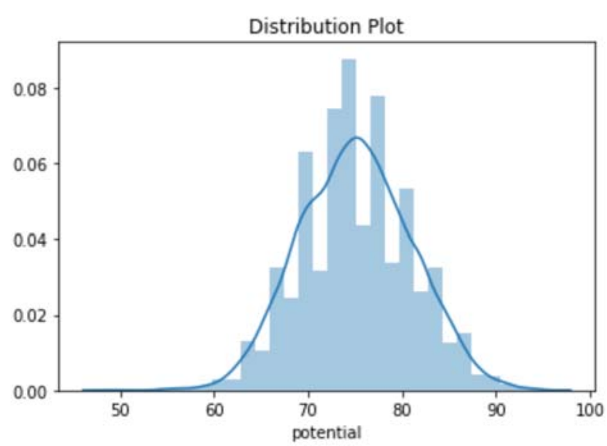

sliding_tackle

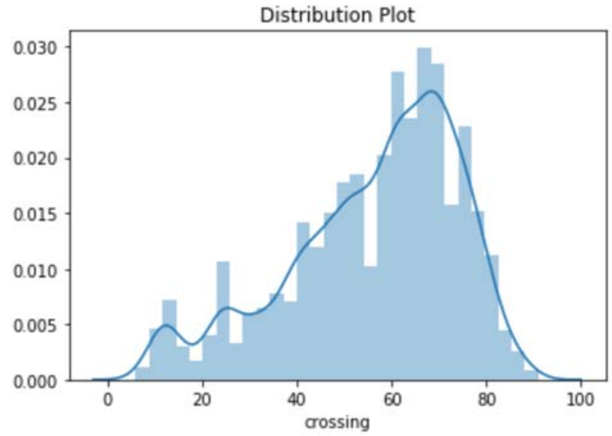

Distribution Plot
age

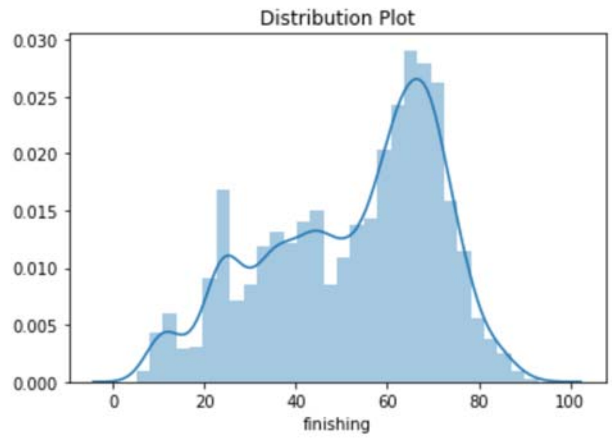

Distribution Plot
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
160
165
170
175
180
185
190
195
200
height

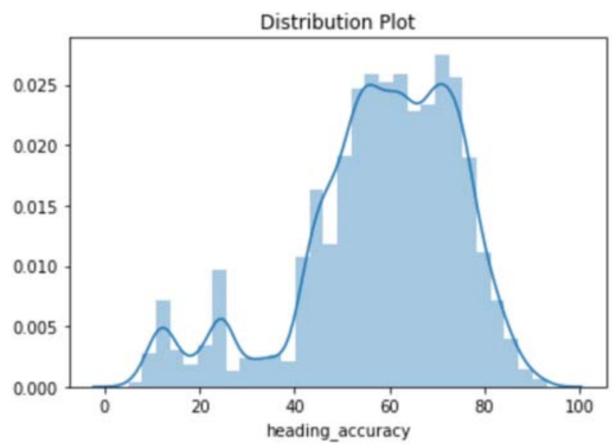

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
120
140
160
180
200
220
weight

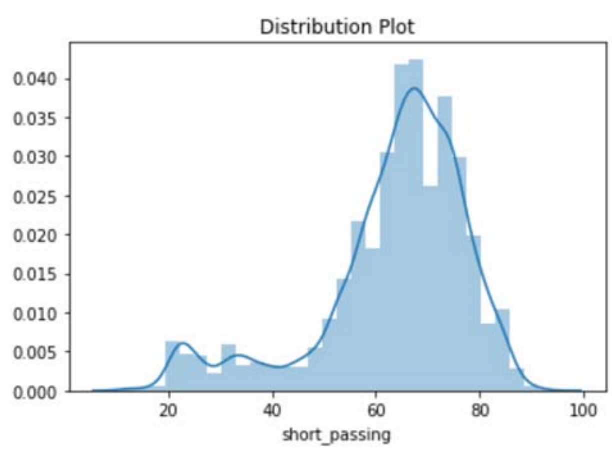

Distribution Plot
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
60
70
80
90
overall_rating

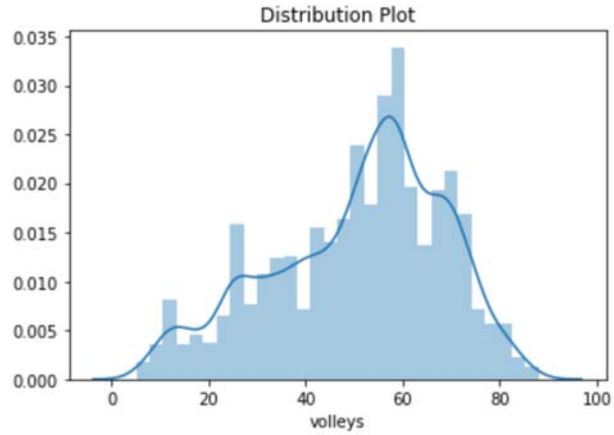

Distribution Plot
0.08
0.06
0.04
0.02
0.00
50
60
70
80
90
100
potential

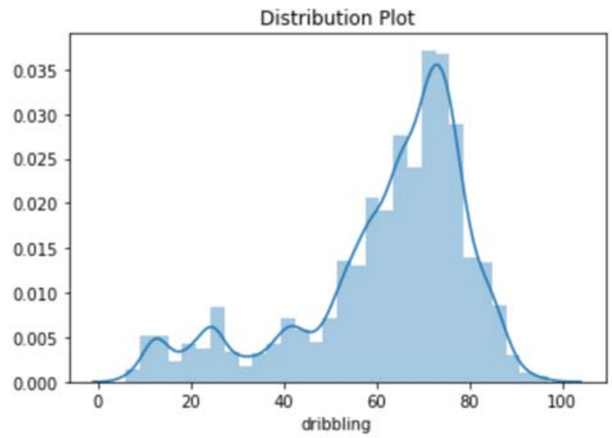

Distribution Plot
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
crossing

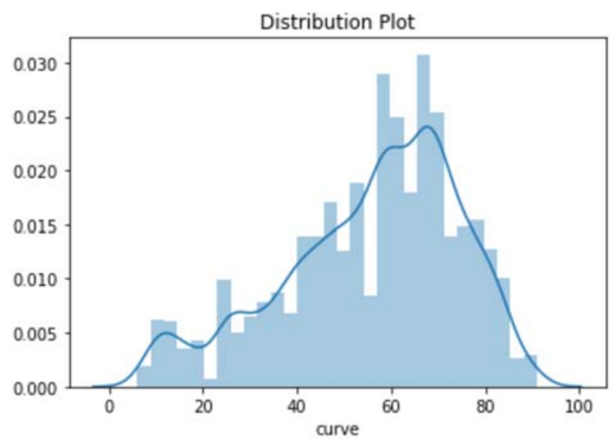

Distribution Plot
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
finishing

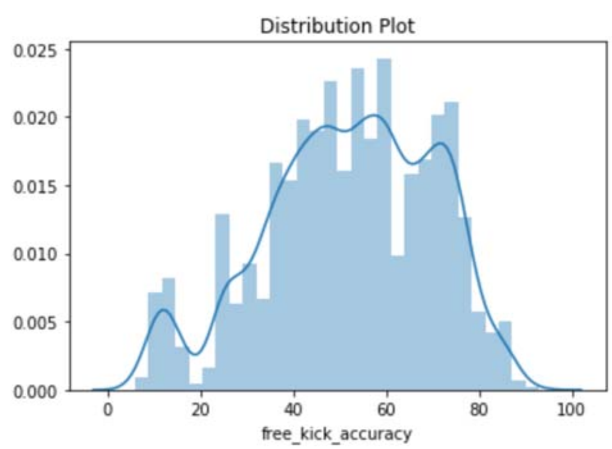

Distribution Plot
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
heading_accuracy

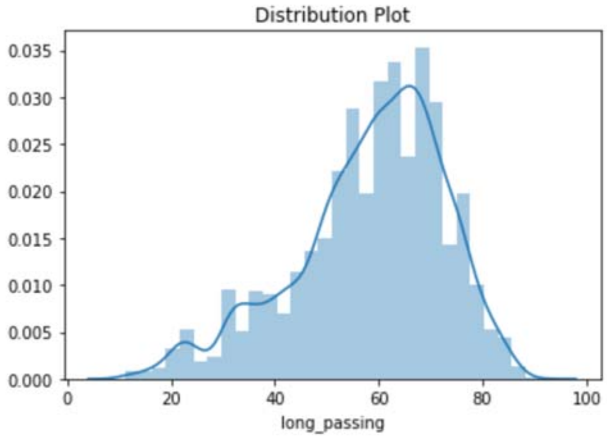

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
short_passing

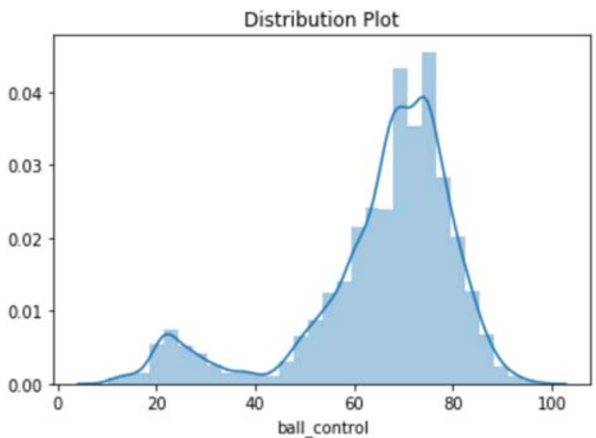

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
volleys

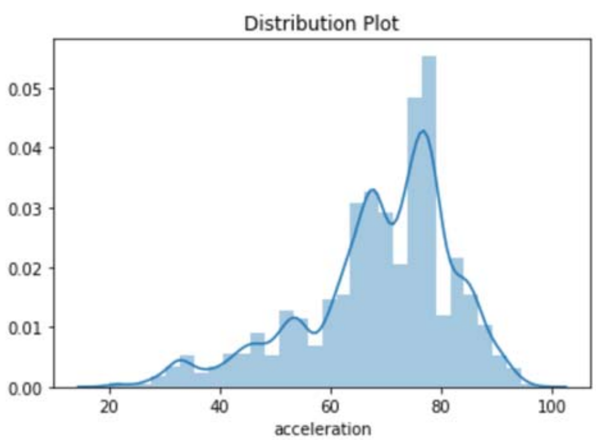

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
dribbling

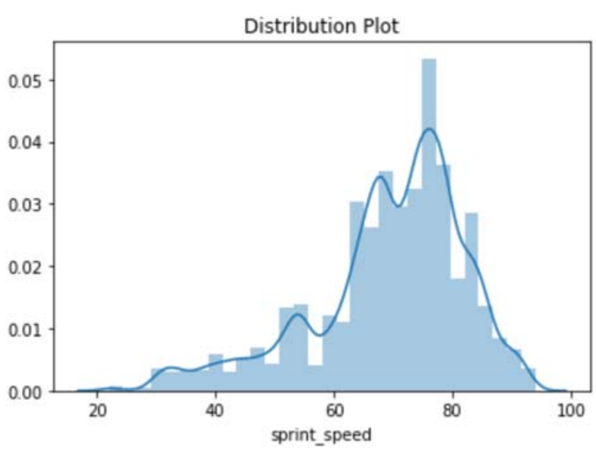

Distribution Plot
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
curve

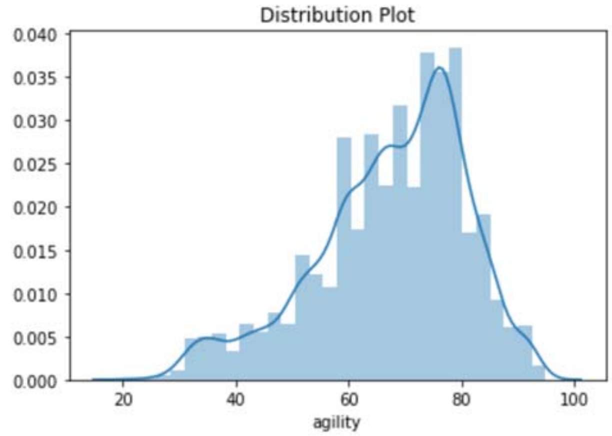

Distribution Plot
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
free_kick_accuracy

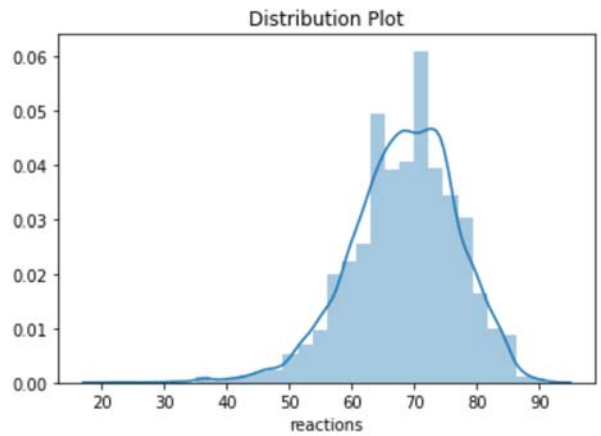

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
long_passing

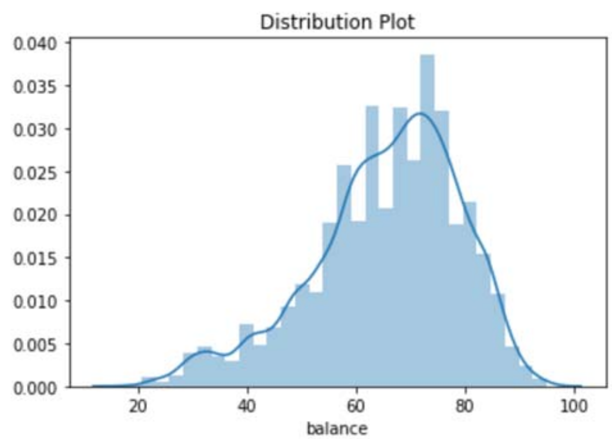

Distribution Plot
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
ball_control

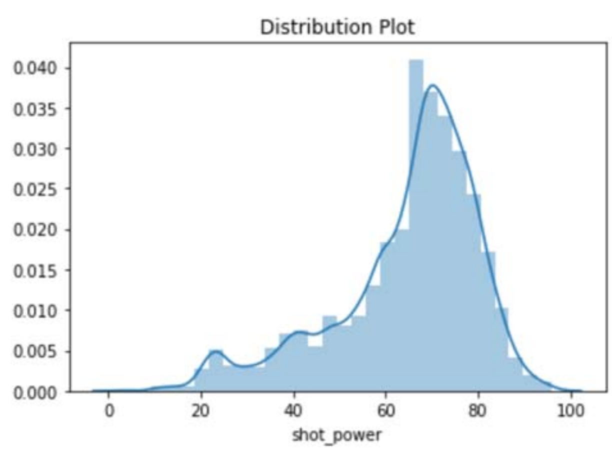

Distribution Plot
0.05
0.04
0.03
0.02
0.01
0.00
20
40
60
80
100
acceleration

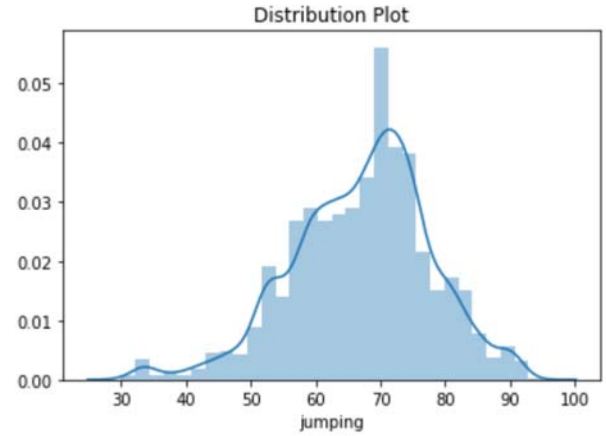

Distribution Plot
0.05
0.04
0.03
0.02
0.01
0.00
20
40
60
80
100
sprint_speed

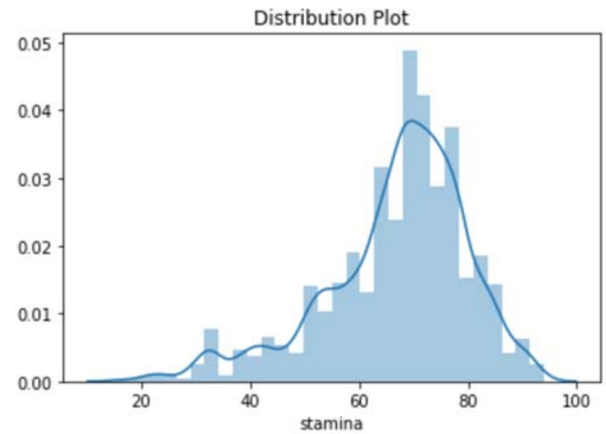

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
agility

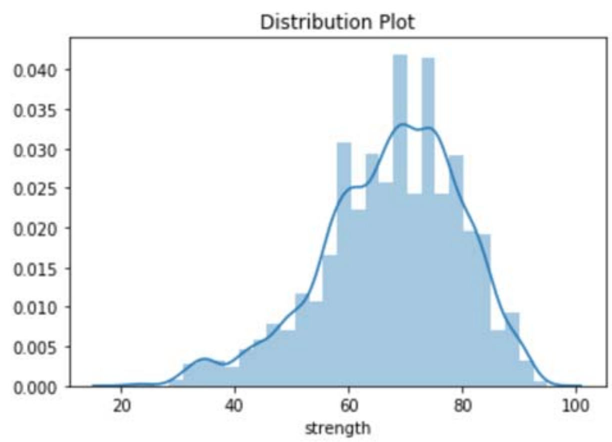

Distribution Plot
0.06
0.05
0.04
0.03
0.02
0.01
0.00
20
30
40
50
60
70
80
90
reactions

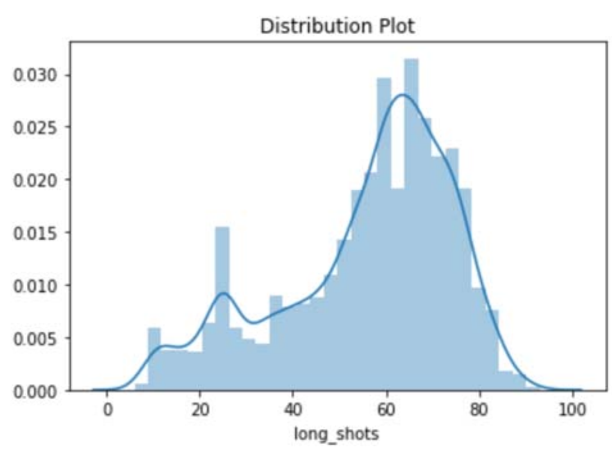

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
balance

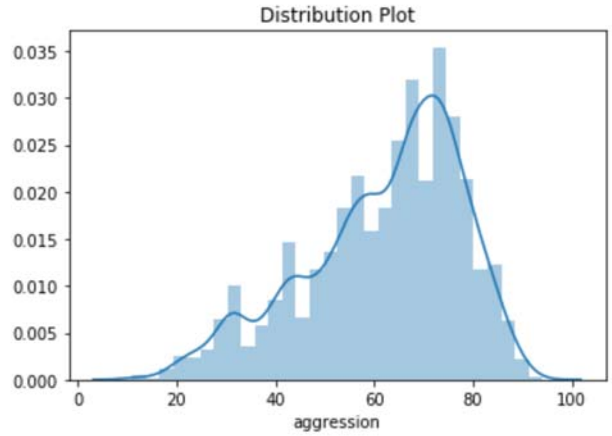

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
shot_power

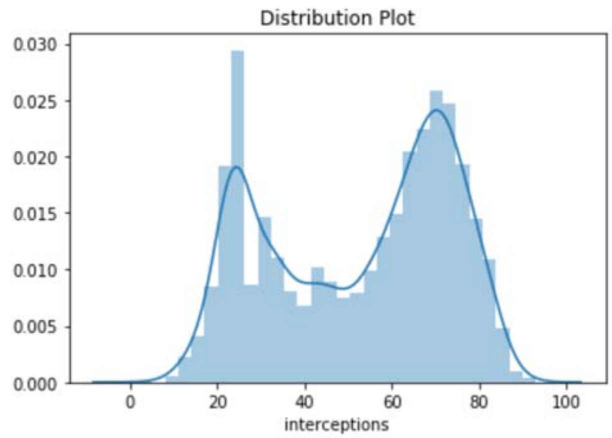

Distribution Plot
0.05
0.04
0.03
0.02
0.01
0.00
30
40
50
60
70
80
90
100
jumping

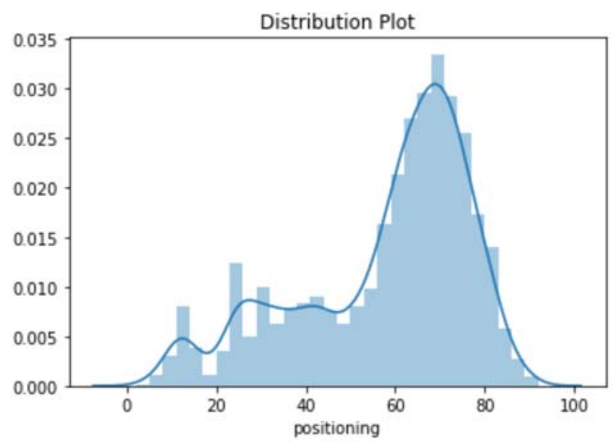

Distribution Plot
0.05
0.04
0.03
0.02
0.01
0.00
20
40
60
80
100
stamina

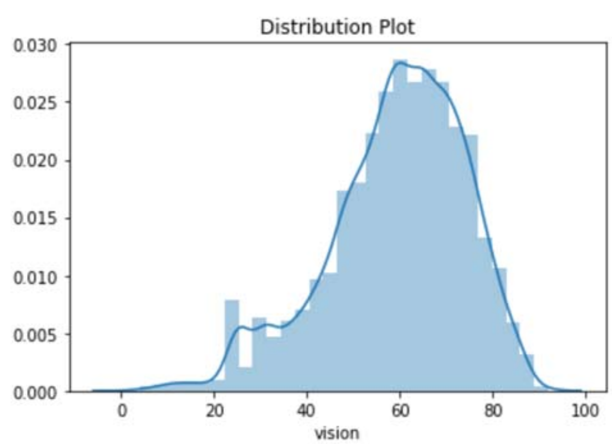

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
strength

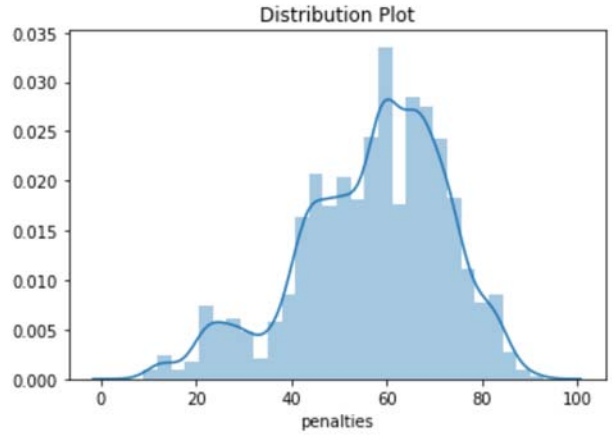

Distribution Plot
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
long_shots

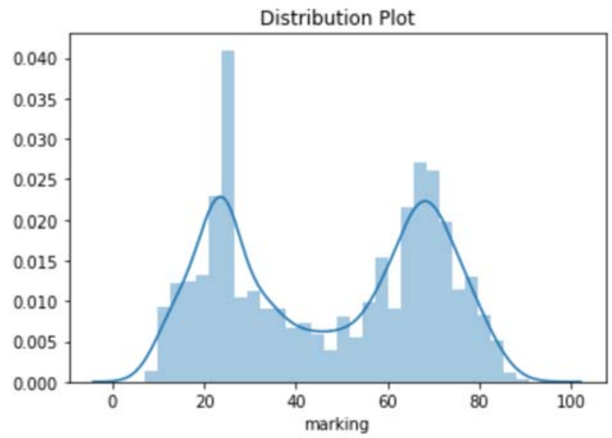

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
aggression

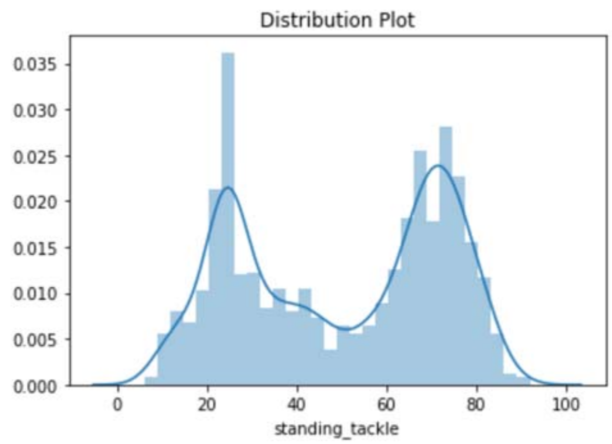

Distribution Plot
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
interceptions

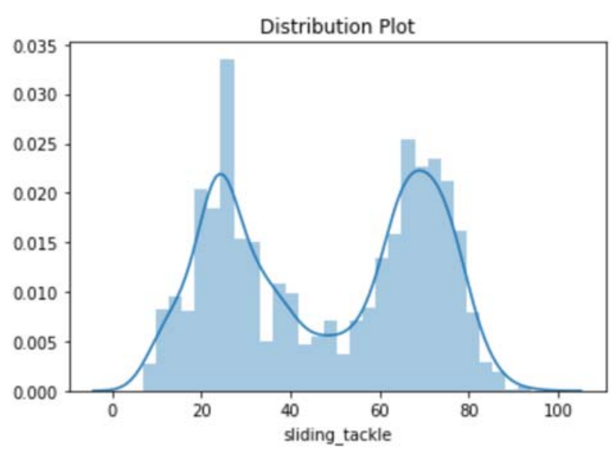

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
positioning

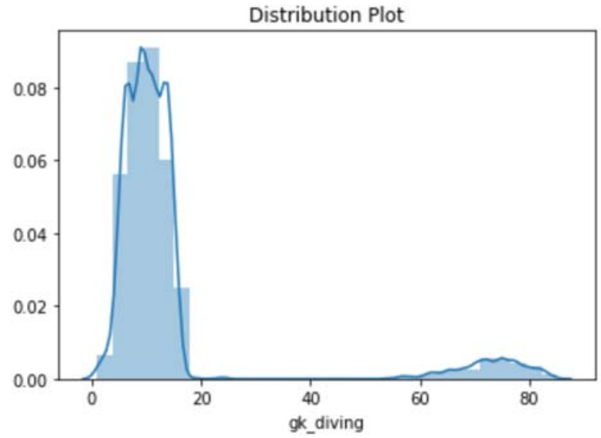

Distribution Plot
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
vision

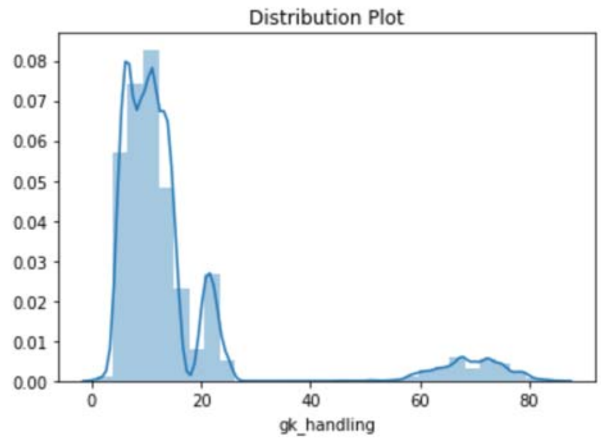

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
penalties

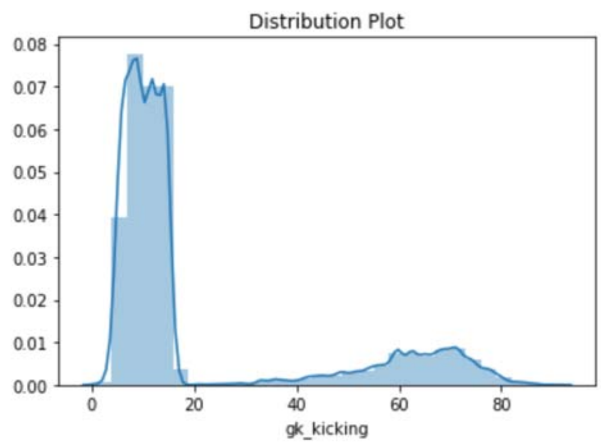

Distribution Plot
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
marking

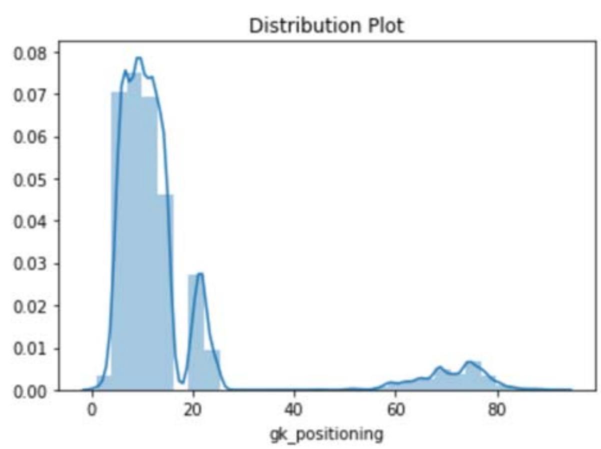

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
standing_tackle

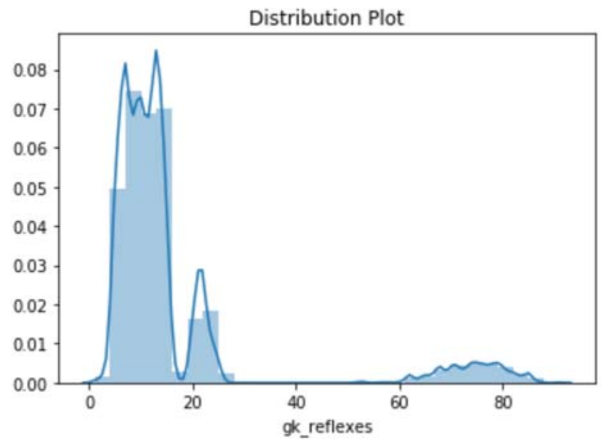

Distribution Plot
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
sliding_tackle


Distribution Plot
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
gk_diving


Distribution Plot
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
gk_handling

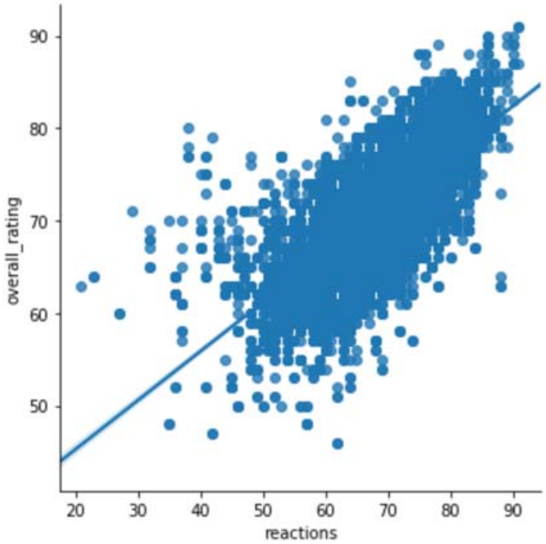

Distribution Plot
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
gk_kicking


Distribution Plot
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
gk_positioning

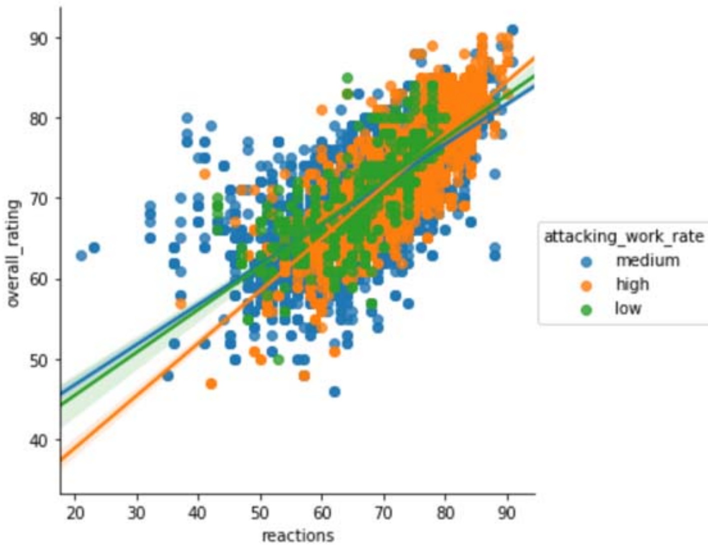

Distribution Plot
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
gk_reflexes

In [40]:
```
print('A Closer Look at Overall Rating Versus Reactions Attributes')
print(' ')
corr1=df_all_col[['reactions','overall_rating']].corr()
print(corr1)
vis1=sns.lmplot( x='reactions', y='overall_rating', hue=None, sharex=False, data=df_all_col,
scatter=True, fit_reg=True, units=None, order=1, legend=True)
plt.show()


df11=df_all_col[df_all_col['attacking_work_rate'].isin (['low','medium','high'])]
corr11=df11[['reactions','overall_rating']].corr()
print(corr11)
vis11=sns.lmplot( x='reactions', y='overall_rating', hue='attacking_work_rate', sharex=False
, data=df11, scatter=True, fit_reg=True, units=None, order=1, legend=True)
plt.show()


df12=df_all_col[df_all_col['defensive_work_rate'].isin (['low','medium','high'])]
corr12=df12[['reactions','overall_rating']].corr()
print(corr12)
vis12=sns.lmplot( x='reactions', y='overall_rating', hue='defensive_work_rate', sharex=False
, data=df12, scatter=True, fit_reg=True, units=None, order=1, legend=True)
plt.show()


df13=df_all_col[df_all_col['gk_diving'] > 40]
corr13=df13[['reactions','overall_rating']].corr()
print('gk_diving > 40 ')
print(corr13)
vis13=sns.lmplot( x='reactions', y='overall_rating', hue=None, sharex=False, data=df13, scat
ter=True, fit_reg=True, units=None, order=1, legend=True)
plt.show()
plt.close()


df14=df_all_col[df_all_col['gk_diving'] < 41]
corr14=df14[['reactions','overall_rating']].corr()
print('gk_diving < 41')
print(corr14)
vis14=sns.lmplot( x='reactions', y='overall_rating', hue=None, sharex=False, data=df14, scat
ter=True, fit_reg=True, units=None, order=1, legend=True)
plt.show()
plt.close()
```

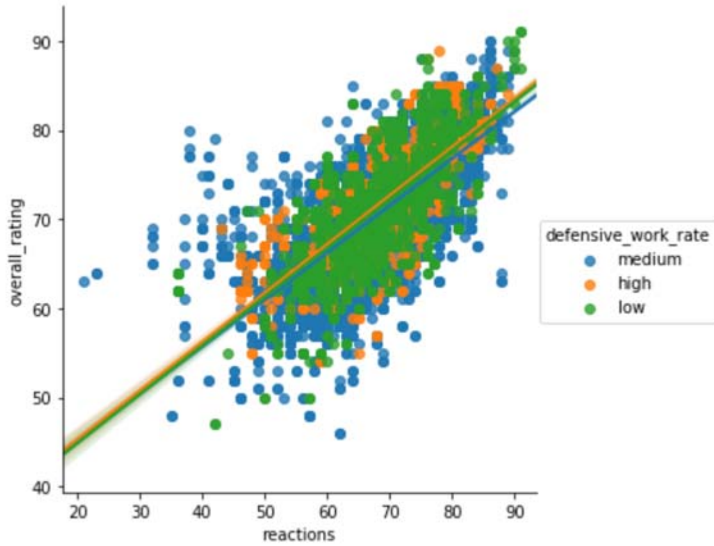
```
A Closer Look at Overall Rating Versus Reactions Attributes

               reactions  overall_rating
reactions        1.00000         0.72483
overall_rating   0.72483         1.00000
```



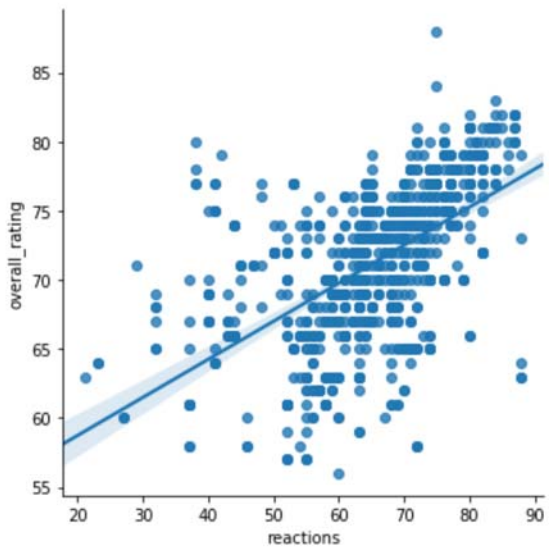

```
               reactions  overall_rating
reactions        1.00000         0.72456
overall_rating   0.72456         1.00000
```



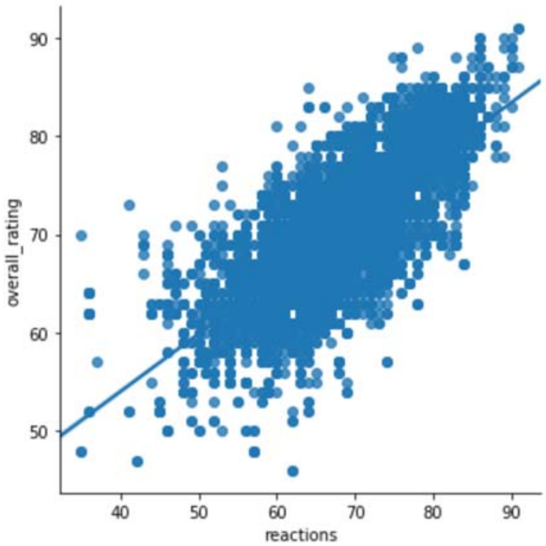

```
               reactions  overall_rating
reactions       1.000000        0.725001
overall_rating  0.725001        1.000000
```




```
gk_diving > 40
                reactions  overall_rating
reactions         1.00000         0.56092
overall_rating    0.56092         1.00000
```




```
gk_diving < 41
                reactions  overall_rating
reactions        1.000000        0.759507
overall_rating   0.759507        1.000000
```


overall_rating
reactions
90
80
70
60
50
40
50
60
70
80
90

Discussion

Clustters

PCA results shows that the first three principal components out of 38 dimensions, PC1, PC2 and PC2, explain 43.9%, 15.7% and 8.9%, respectively, which is a total of 68.5% of the variance in the final dataset of soccer players. Loading scores of all player attributes in the three principal components show that no one single player attribute contributes significantly more than any other attributes to these three principal components. Scatter matrix plots of first 20 principal components indicate presence of two distinct large and small subgroups of players in the PC1 dimension. Scatter plots, distribution plots, joinplots and correlation coefficient analysis explained the clusters and substantiated the answers to the research questions.

Some of the attributes, marking, standing tackle and sliding tackle, display bimodal distribution, which suggests that these attributes divide the players into two subgroups. Ditribution plots basing on goalkeeper attributes like gk_diving, gk_reflexes, gk_handling, gk_positioning and gk_kicking also show a large cluster with distinctively low scores and a much smaller subgroup with relatively high scores.

Correlation

For correlation of overall rating with a single player attribute, reaction attribute has the strongest positive linear correlation. (coeff = 0.7248)

Player attributes are tallied by the following attribute categories for further analysis:

1. Total Attack: crossing, finishing, heading accuracy, short passing and volleys
2. Total Skill: dribbling, curve, free kicking accuracy, long passing and ball control
3. Total Movement: acceleration, sprint speed, agility, reactions and balance
4. Total Power: shot power, jumping, stamina, strength and long shots
5. Total Mentality: aggression, interceptions, positioning, vision and penalties
6. Total Defending: marking, standing tackle and sliding tackle
7. Total Goalkeeping: gk_diving, dg_handling, gk_kicking, gk_positioning and gk_reflexes

Analysis shows that overall rating very strong correlation (coeff = 0.978269) with total goalkeeping attribute categpry of the goalkeeper subgroup. This subgroup of players have total scores of all goalkeeper attributes greater than 200. It is reasonanle to beieve that this is a subgroup group of goalkeepers, who receive specialized drills on attributes that are important for goalkeeper position. As for the other two subgroups with total goalkeeping scores less than 200, overall rating and total goalkeeping scores do not correlate at all.

List of Loading Scores for all Player Attributes in PC1 Dimension

A closer look at the loading values for PC1 shows that, basically, no skill is much more influential than others.

```
Sorted Loading Scores for all attributes in PC1 dimension:
ball_control 0.233895
dribbling 0.226609
short_passing 0.220168
crossing 0.213529
curve 0.211442
long_shots 0.211257
positioning 0.204851
shot_power 0.199655
vision 0.197614
gk_diving 0.196600
free_kick_accuracy 0.196366
gk_reflexes 0.195234
gk_handling 0.195011
gk_positioning 0.193662
volleys 0.193077
finishing 0.191773
acceleration 0.185480
penalties 0.183370
sprint_speed 0.182038
long_passing 0.177636
agility 0.168734
```