In [1]:

```
#importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import numpy as np
```

In [2]:

```
#importing data
df = pd.read_csv('Footballdata.csv')
#checking info to see what columns (features) are present
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17588 entries, 0 to 17587
Data columns (total 53 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Name                17588 non-null  object 
 1   Nationality         17588 non-null  object 
 2   National_Position   1075 non-null   object 
 3   National_Kit        1075 non-null   float64
 4   Club                17588 non-null  object 
 5   Club_Position       17587 non-null  object 
 6   Club_Kit            17587 non-null  float64
 7   Club_Joining        17587 non-null  object 
 8   Contract_Expiry     17587 non-null  float64
 9   Rating              17588 non-null  int64  
 10  Height              17588 non-null  object 
 11  Weight              17588 non-null  object 
 12  Preffered_Foot      17588 non-null  object 
 13  Birth_Date          17588 non-null  object 
 14  Age                 17588 non-null  int64  
 15  Preffered_Position  17588 non-null  object 
 16  Work_Rate           17588 non-null  object 
 17  Weak_foot           17588 non-null  int64  
 18  Skill_Moves         17588 non-null  int64  
 19  Ball_Control        17588 non-null  int64  
 20  Dribbling           17588 non-null  int64  
 21  Marking             17588 non-null  int64  
 22  Sliding_Tackle      17588 non-null  int64  
 23  Standing_Tackle     17588 non-null  int64  
 24  Aggression          17588 non-null  int64  
 25  Reactions           17588 non-null  int64  
 26  Attacking_Position  17588 non-null  int64
 27  Interceptions       17588 non-null  int64
 28  Vision              17588 non-null  int64
 29  Composure           17588 non-null  int64
 30  Crossing            17588 non-null  int64
 31  Short_Pass          17588 non-null  int64
 32  Long_Pass           17588 non-null  int64
 33  Acceleration        17588 non-null  int64
 34  Speed               17588 non-null  int64
 35  Stamina             17588 non-null  int64
 36  Strength            17588 non-null  int64
 37  Balance             17588 non-null  int64
 38  Agility             17588 non-null  int64
 39  Jumping             17588 non-null  int64
 40  Heading             17588 non-null  int64
 41  Shot_Power          17588 non-null  int64
 42  Finishing           17588 non-null  int64
 43  Long_Shots          17588 non-null  int64
 44  Curve               17588 non-null  int64
 45  Freekick_Accuracy   17588 non-null  int64
 46  Penalties           17588 non-null  int64
 47  Volleys             17588 non-null  int64
 48  GK_Positioning      17588 non-null  int64
 49  GK_Diving           17588 non-null  int64
 50  GK_Kicking          17588 non-null  int64
 51  GK_Handling         17588 non-null  int64
 52  GK_Reflexes         17588 non-null  int64
dtypes: float64(3), int64(38), object(12)
memory usage: 7.1+ MB
```

In [3]:

```
#dropping unnecessary columns
df2 = df.drop(['Name', 'Nationality', 'National_Kit', 'Club', 'Club_Kit', 'Club_Joining', 'Contract_Expiry',
               'National_Position', 'Rating', 'Preffered_Foot', 'Birth_Date', 'Age', 'Preffered_Position', 'Work_Rate',
              'Height', 'Weight', 'GK_Positioning', 'GK_Diving', 'GK_Kicking', 'GK_Handling', 'GK_Reflexes'],
              axis=1)
```

In [4]:

```
#checking all the unique positions for players
pd.unique(df['Club_Position'])
```

Out[4]:

```
array(['LW', 'RW', 'ST', 'GK', 'Sub', 'RCM', 'CAM', 'LCB', 'LCM', 'RS',
       'RB', 'RCB', 'LM', 'LDM', 'RM', 'LB', 'CDM', 'RDM', 'LF', 'CB',
       'LAM', 'Res', 'CM', 'LS', 'RF', 'RWB', 'RAM', 'LWB', nan, 'CF'],
      dtype=object)
```

List of all positions I need: Forward - LW, RW, ST, RS, LF, LS, RF, CF Mid - RCM, CAM, LCM, LM, LDM, RM, CDM, RDM, LAM, CM, RAM

In [5]:

```
#replacing the values of all unnecessary positions (not midfielders or forwards) and dropping them
df_removal = df2.replace(['GK', 'Sub', 'LCB', 'RB', 'RCB', 'LB', 'CB', 'Res', 'RWB', 'LWB'] , np.nan)
df_removal = df_removal.dropna().reset_index(drop=True)
```

In [6]:

```
#changing the name of all forward positions to forward and all midfielder positions to midfielder
df_update = df_removal.replace(['LW', 'RW', 'ST', 'RS', 'LF', 'LS', 'RF', 'CF'], 'Forward')
df_update = df_update.replace(['RCM', 'CAM', 'LCM', 'LM', 'LDM', 'RM', 'CDM', 'RDM', 'LAM', 'CM', 'RAM'], 'Midfielder')
#ensuring that the replacemnet was successful
pd.unique(df_update['Club_Position'])
```

Out[6]:

```
array(['Forward', 'Midfielder'], dtype=object)
```

In [7]:

```
#changing the club_positiosn to binary values of 1 and 0 as Logistic Regression model cannot take string values
le = LabelEncoder()
df_update['Club_Position'] = le.fit_transform(df_update['Club_Position'])
#ensuring if label encoding was successful
pd.unique(df_update['Club_Position'])
```

Out[7]:

```
array([0, 1])
```

In [8]:

```
#converting the columns headings for the dataframe into a list
column_list = list(df_update.columns.values)
column_list[1:]
```

Out[8]:

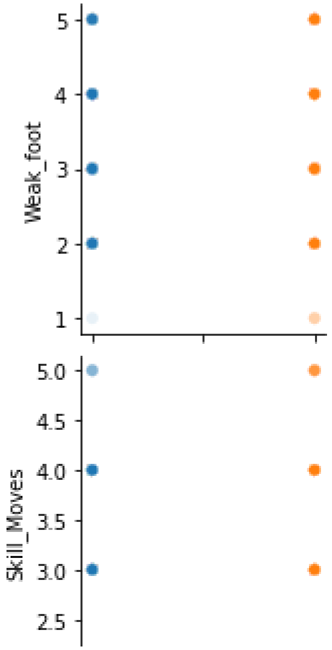
```
['Weak_foot',
 'Skill_Moves',
 'Ball_Control',
 'Dribbling',
 'Marking',
 'Sliding_Tackle',
 'Standing_Tackle',
 'Aggression',
 'Reactions',
 'Attacking_Position',
 'Interceptions',
 'Vision',
 'Composure',
 'Crossing',
 'Short_Pass',
 'Long_Pass',
 'Acceleration',
 'Speed',
 'Stamina',
 'Strength',
 'Balance',
 'Agility',
 'Jumping',
 'Heading',
 'Shot_Power',
 'Finishing',
 'Long_Shots',
 'Curve',
 'Freekick_Accuracy',
 'Penalties',
 'Volleys']
```

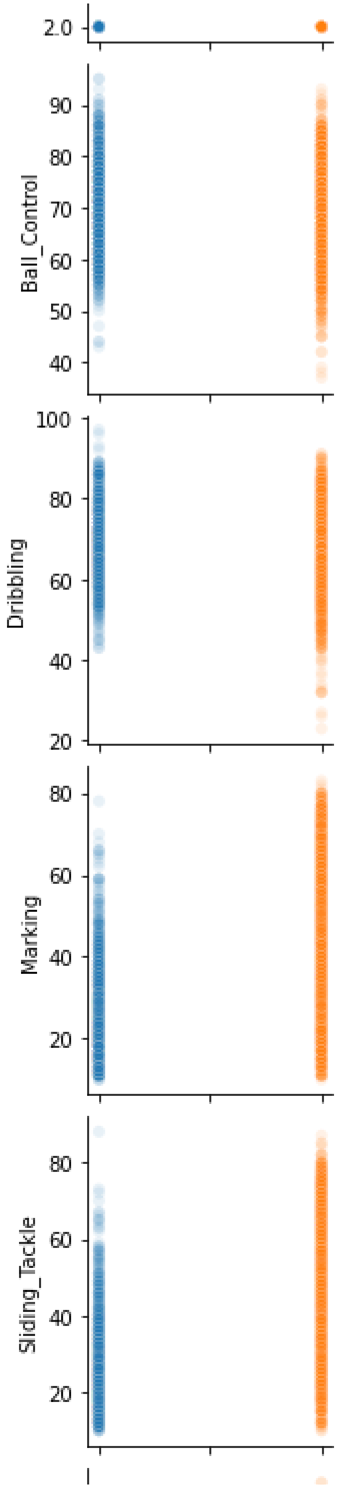
In [9]:

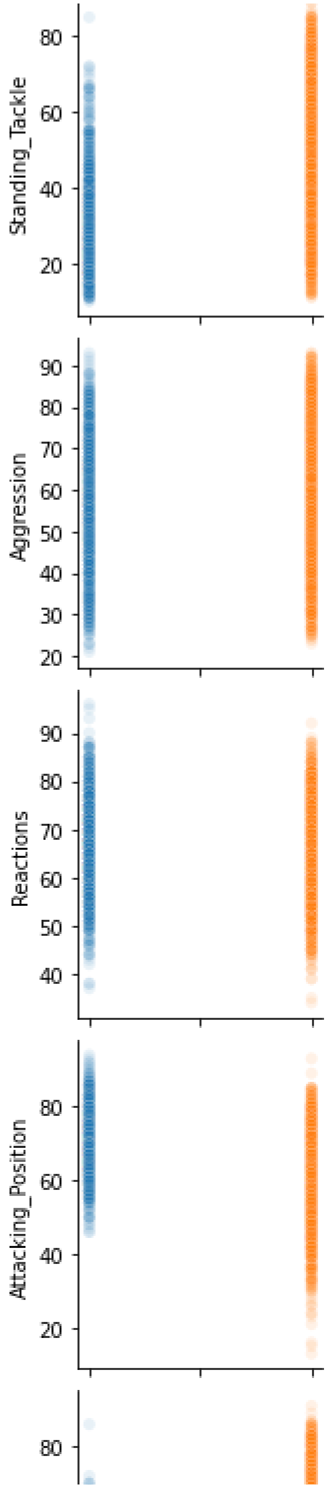
```
#generating pariplot to check how different features differntiate for forwards and midfielders
sns.pairplot(
    df_update,
    x_vars=['Club_Position'],
    y_vars=column_list[1:],
    hue='Club_Position',
    kind='scatter',
    plot_kws={'alpha':0.1}
)
```

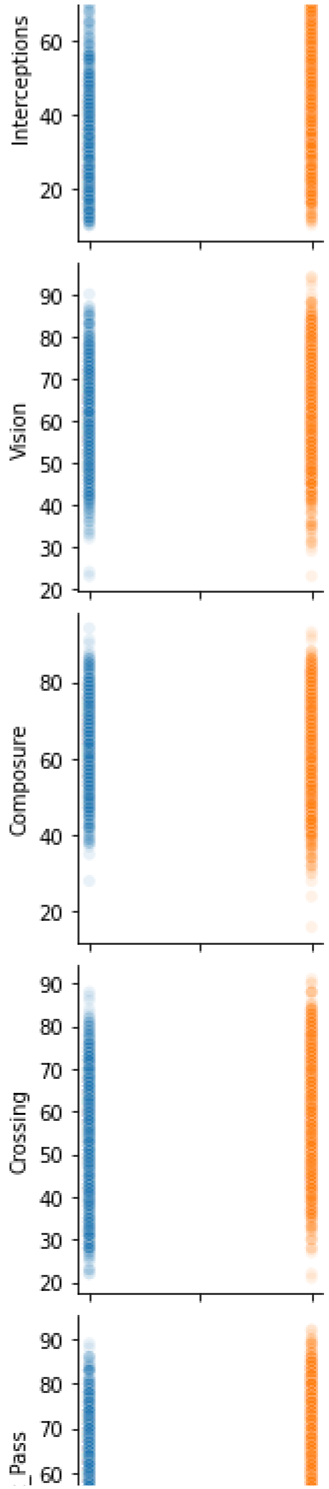
Out[9]:

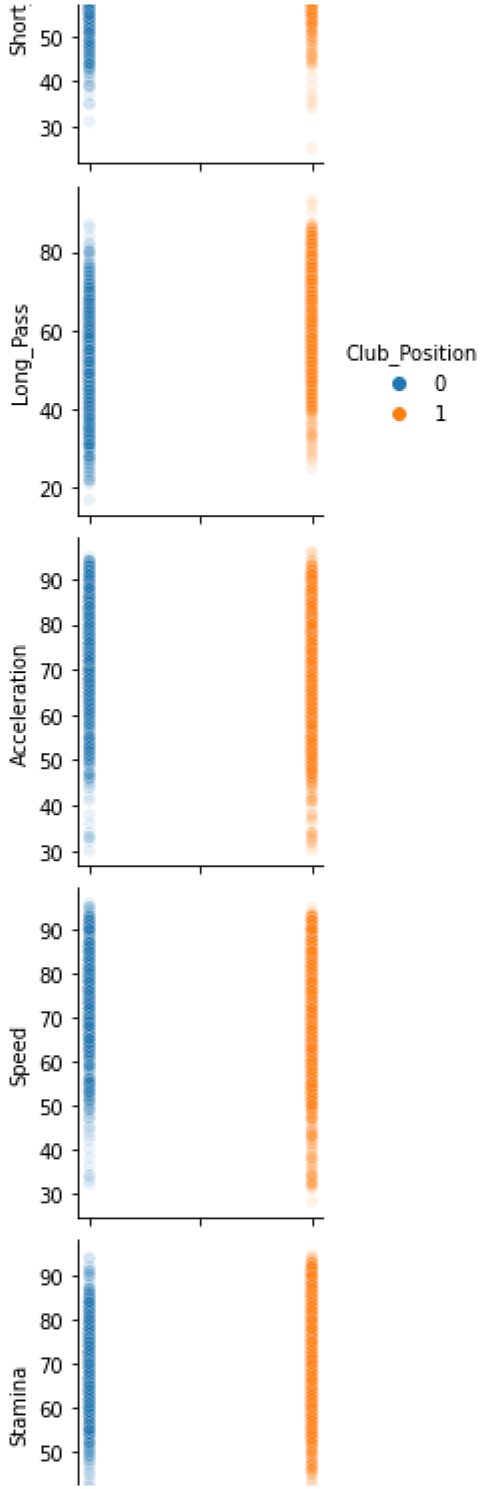
```
<seaborn.axisgrid.PairGrid at 0x207f6513580>
```

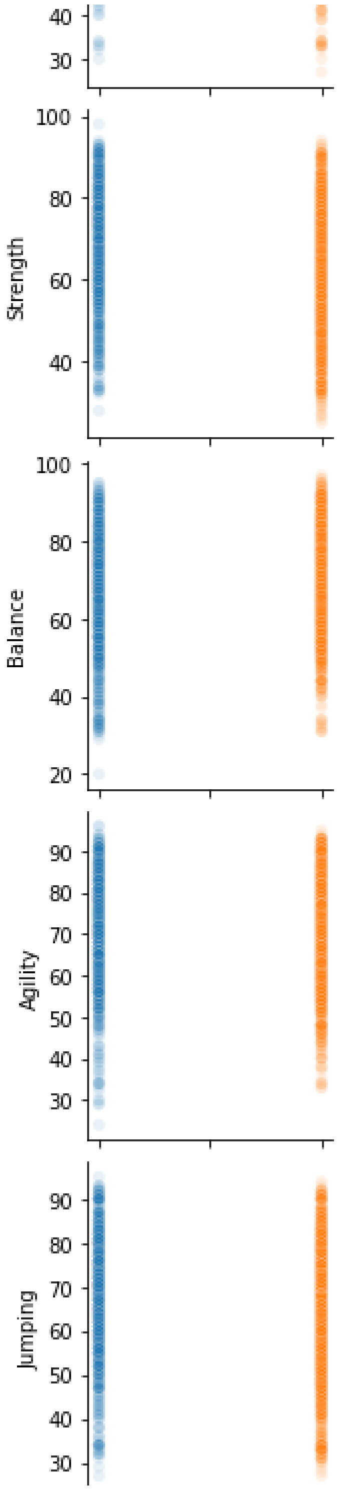


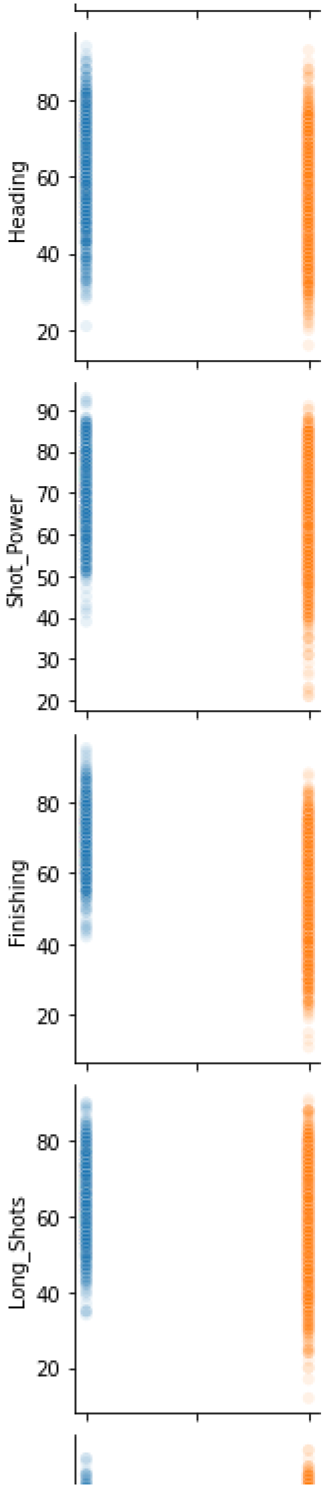

2.0
Ball_Control
90
80
70
60
50
40
Dribbling
100
80
60
40
20
Marking
80
60
40
20
Sliding_Tackle
80
60
40
20

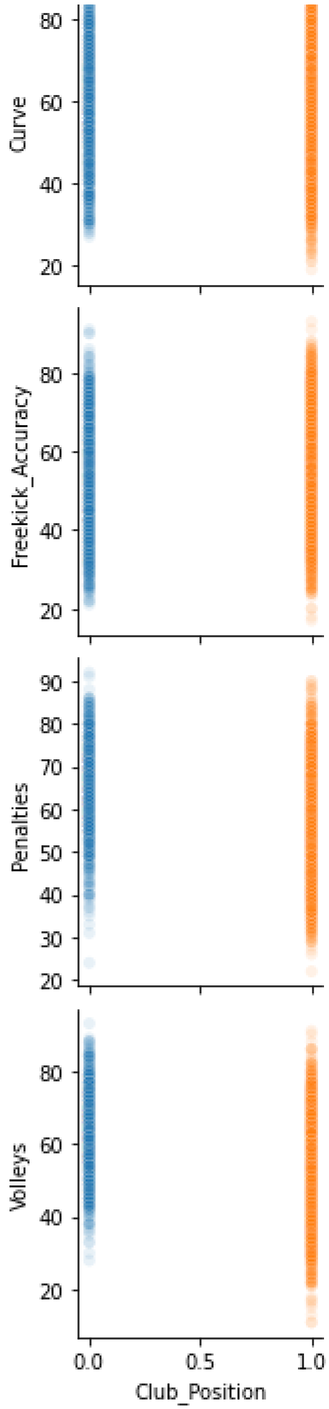

Standing_Tackle
80
60
40
20
Aggression
90
80
70
60
50
40
30
20
Reactions
90
80
70
60
50
40
Attacking_Position
80
60
40
20
80


Interceptions
60
40
20
Vision
90
80
70
60
50
40
30
20
Composure
80
60
40
20
Crossing
90
80
70
60
50
40
30
20
Pass
90
80
70
60


Short
50
40
30
Long_Pass
80
60
40
20
Club_Position
0
1
Acceleration
90
80
70
60
50
40
30
Speed
90
80
70
60
50
40
30
Stamina
90
80
70
60
50


40
30
100
80
60
40
Strength
100
80
60
40
20
Balance
90
80
70
60
50
40
30
Agility
90
80
70
60
50
40
30
Jumping


Heading
80
60
40
20
Shot_Power
90
80
70
60
50
40
30
20
Finishing
80
60
40
20
Long_Shots
80
60
40
20



Posssible Correlations: Dribbling, Marking, Sliding_Tackle, Standing_Tackle, Attacking_Position, Interceptions, Short_Pass, Long_Pass, Speed, Balance, Agility, Shot_Power, Finishing, Penalties, Volleys, Curve

In [10]:

```
#making the final dataframe that only has the features that I need
df_trial_1 = df_update[['Club_Position', 'Dribbling', 'Marking', 'Sliding_Tackle', 'Standing_Tackle', 'Attacking_Positior
                        'Interceptions', 'Short_Pass', 'Long_Pass', 'Speed', 'Balance', 'Agility', 'Shot_Power',
                        'Finishing', 'Penalties', 'Volleys', 'Curve']]
df_trial_1 = df_trial_1.rename(columns = {'Club_Position':'Position'})
```

In [11]:

```
#assigning the columns to X and y
X = df_trial_1[df_trial_1.columns[1:]]
y = df_trial_1[['Position']]

#splitting the data into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#fitting to the logistic regression model
clf = LogisticRegression(random_state=123, solver='lbfgs', max_iter=10000)
clf.fit(X_train, y_train.values.ravel())

#checking the accuracy score of my ML pipeline
print(f'Model Accuracy: {clf.score(X_test, y_test)}')
```

```
Model Accuracy: 0.8745046235138706
```

In [12]:

```
#showing predictions and actual values side by side
predictions = le.inverse_transform(clf.predict(X_test))
actual_values = le.inverse_transform(np.concatenate(y_test.values, axis=0))
data = {
    'Predictions': predictions,
    'Actual Values': actual_values
}
df_predictions = pd.DataFrame(data)
df_predictions
```

Out[12]:

	Predictions	Actual Values
0	Forward	Forward
1	Forward	Forward
2	Midfielder	Midfielder

	Predictions	Actual Values
3	Midfielder	Midfielder
4	Midfielder	Midfielder
...	...	...
752	Midfielder	Midfielder
753	Midfielder	Midfielder
754	Midfielder	Midfielder
755	Midfielder	Midfielder
756	Midfielder	Midfielder

757 rows × 2 columns