Brief Description

In this notebook, we will predict scores based on the other parameters using Linear Regression

Import Libraries

In [42]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib
%matplotlib inline
from sklearn.model_selection import train_test_split
```

Import the data

spi_matches.csv contains match-by-match SPI ratings and forecasts back to 2016.

In [43]:
```
#Source of Data: [https://projects.fivethirtyeight.com/soccer-api/club/spi_matches.csv]
df_unrefined = pd.read_csv('spi_matches.csv')
df_unrefined.head()
```

Out[43]:

	date	league_id	league	team1	team2	spi1	spi2	prob1	prob2	probtie	...
0	2016-08-12	1843	French Ligue 1	Bastia	Paris Saint-Germain	51.16	85.68	0.0463	0.8380	0.1157	...
1	2016-08-12	1843	French Ligue 1	AS Monaco	Guingamp	68.85	56.48	0.5714	0.1669	0.2617	...
2	2016-08-13	2411	Barclays Premier League	Hull City	Leicester City	53.57	66.81	0.3459	0.3621	0.2921	...
3	2016-08-13	2411	Barclays Premier League	Crystal Palace	West Bromwich Albion	55.19	58.66	0.4214	0.2939	0.2847	...
4	2016-08-13	2411	Barclays Premier League	Everton	Tottenham Hotspur	68.02	73.25	0.3910	0.3401	0.2689	...

5 rows × 22 columns

Remove unwanted features

In [44]:
```
df = df_unrefined.drop(columns=['date','league_id','league','team1'
,'team2'])
df.head()
```

Out[44]:

	spi1	spi2	prob1	prob2	probtie	proj_score1	proj_score2	importance1	importance
0	51.16	85.68	0.0463	0.8380	0.1157	0.91	2.36	32.4	67.
1	68.85	56.48	0.5714	0.1669	0.2617	1.82	0.86	53.7	22.
2	53.57	66.81	0.3459	0.3621	0.2921	1.16	1.24	38.1	22.
3	55.19	58.66	0.4214	0.2939	0.2847	1.35	1.14	43.6	34.
4	68.02	73.25	0.3910	0.3401	0.2689	1.47	1.38	31.9	48.

There are 32292 records in the dataset. Let's see how many of them contain NaN

In [45]:
```
df.isna().sum()
```

Out[45]:
```
spi1              0
spi2              0
prob1             0
prob2             0
probtie           0
proj_score1       0
proj_score2       0
importance1    8385
importance2    8385
score1         4113
score2         4113
xg1           16835
xg2           16835
nsxg1         16835
nsxg2         16835
adj_score1    16835
adj_score2    16835
dtype: int64
```

In [46]:
```
df=df.dropna()
df.head()
```

Out[46]:

	spi1	spi2	prob1	prob2	probtie	proj_score1	proj_score2	importance1	importance
0	51.16	85.68	0.0463	0.8380	0.1157	0.91	2.36	32.4	67.
1	68.85	56.48	0.5714	0.1669	0.2617	1.82	0.86	53.7	22.
2	53.57	66.81	0.3459	0.3621	0.2921	1.16	1.24	38.1	22.
3	55.19	58.66	0.4214	0.2939	0.2847	1.35	1.14	43.6	34.
4	68.02	73.25	0.3910	0.3401	0.2689	1.47	1.38	31.9	48.

In [47]:
```
df.isna().sum()
```

Out[47]:
```
spi1            0
spi2            0
prob1           0
prob2           0
probtie         0
proj_score1     0
proj_score2     0
importance1     0
importance2     0
score1          0
score2          0
xg1             0
xg2             0
nsxg1           0
nsxg2           0
adj_score1      0
adj_score2      0
dtype: int64
```

In [48]:
```
dataframe_minus_score = df.drop(columns=['score1','score2'])
```

Split the test and train data

In [49]:
```
T1X_train, T1X_test, T1Y_train, T1Y_test = train_test_split(datafra
me_minus_score, df.score1, random_state=1)
```

In [50]:
```
from sklearn.linear_model import LinearRegression
T1_LR = LinearRegression()
T1_LR.fit(T1X_train,T1Y_train)
```

Out[50]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [51]:
```
T1Predict_train = T1_LR.predict(T1X_train)
T1Predict_test = T1_LR.predict(T1X_test)
```

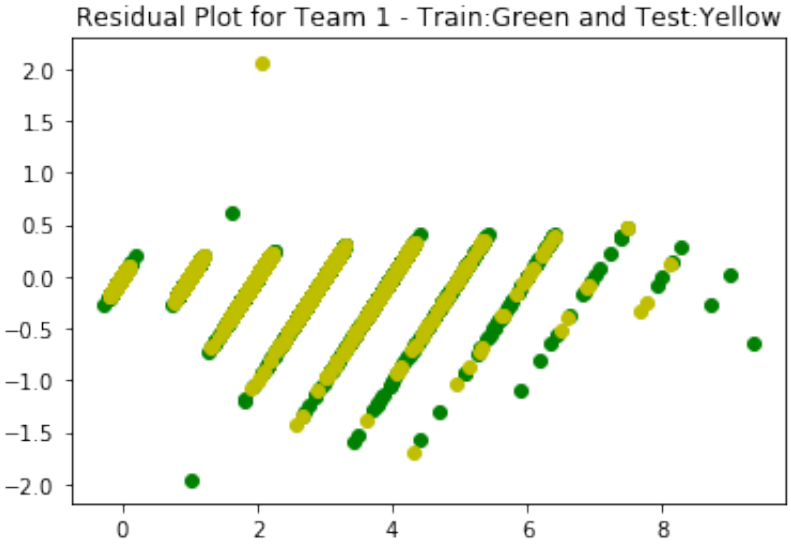
Print Linear Regression Mean Square Error for Team 1

In [52]:
```
LRTrainT1_Error = round(np.mean(np.subtract(T1Y_train.values,T1Pred
ict_train) ** 2),3)
print(LRTrainT1_Error)
LRTestT1_Error = round(np.mean(np.subtract(T1Y_test.values,T1Predic
t_test) ** 2),3)
print(LRTestT1_Error)
```

```
0.028
0.03
```

In [53]:
```
plt.title('Residual Plot for Team 1 - Train:Green and Test:Yellow')
plt.scatter(T1Predict_train,np.subtract(T1Predict_train,T1Y_train.v
alues),c='g')
plt.scatter(T1Predict_test,np.subtract(T1Predict_test,T1Y_test.valu
es),c='y')
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x1a1a2875f8>
```



Print Linear Regression Mean Square Error for Team 2

In [54]:
```
T2X_train, T2X_test, T2Y_train, T2Y_test = train_test_split(datafra
me_minus_score, df.score2, random_state=1)
T2_LR = LinearRegression()
T2_LR.fit(T2X_train,T2Y_train)
```

Out[54]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

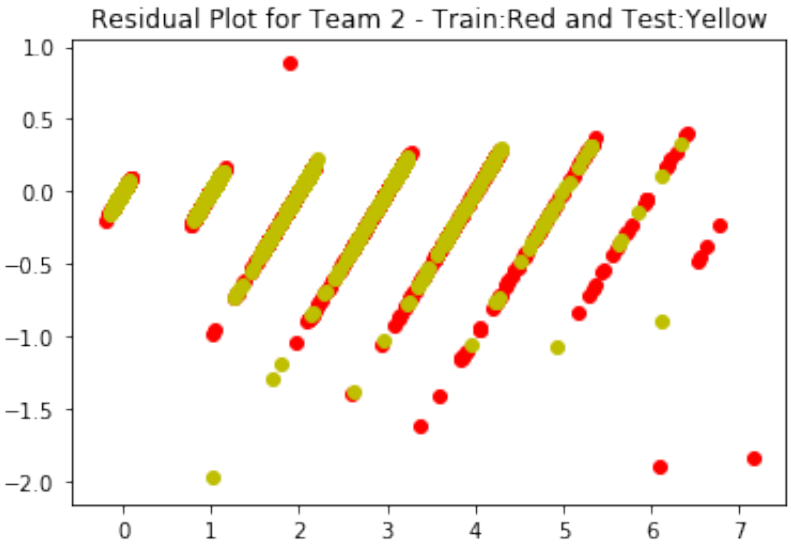
In [55]:
```
T2Predict_train = T2_LR.predict(T2X_train)
T2Predict_test = T2_LR.predict(T2X_test)
LRTrainT2_Error = round(np.mean(np.subtract(T2Y_train.values,T2Pred
ict_train) ** 2),3)
print(LRTrainT2_Error)
LRTestT2_Error = round(np.mean(np.subtract(T2Y_test.values,T2Predic
t_test) ** 2),3)
print(LRTestT2_Error)
```

```
0.017
0.018
```

Residual Plot

In [56]:
```
plt.title('Residual Plot for Team 2 - Train:Red and Test:Yellow')
#plt.scatter(T2Y_train,np.subtract(T1Predict_train,T2Y_train.values
),c='b')
plt.scatter(T2Predict_train,np.subtract(T2Predict_train,T2Y_train.v
alues),c='r')
plt.scatter(T2Predict_test,np.subtract(T2Predict_test,T2Y_test.valu
es),c='y')
```

Out[56]:
```
<matplotlib.collections.PathCollection at 0x1a1a22f5f8>
```



Analysing the work done

SEEN DATA (Training Set)

I have chosen a team(team 1) and summarized the actual score, predicted score, the error that came up with linear regression and the root mean square error too.

In [57]:
```
Team1_Score_predictor_seen = pd.DataFrame({ "Actual_Score": T1Y_tra
in, "Predicted_score": T1Predict_train})
Team1_Score_predictor_seen['Error']= abs(Team1_Score_predictor_seen
.Predicted_score-Team1_Score_predictor_seen.Actual_Score)
Team1_Score_predictor_seen['RMS_Error']= np.sqrt(((Team1_Score_pred
ictor_seen.Predicted_score-Team1_Score_predictor_seen.Actual_Score)
**2).mean())
Team1_Score_predictor_seen.head()
```

Out[57]:

	Actual_Score	Predicted_score	Error	RMS_Error
15337	3.0	3.138893	0.138893	0.168435
8677	1.0	1.037737	0.037737	0.168435
11113	3.0	3.082295	0.082295	0.168435
8501	1.0	0.927809	0.072191	0.168435
18456	0.0	0.014899	0.014899	0.168435

Analysis of errors

In [67]:
```
m=Team1_Score_predictor_seen['Error'].min()
k=Team1_Score_predictor_seen['Error'].max()
n=Team1_Score_predictor_seen['Error'].mean()
print(f"Min Error on prediction of seen data for team 1 scores: {m}
")
print(f"Max Error on prediction of seen data for team 1 scores: {k}
")
print(f"Mean Error on prediction of seen data for team 1 scores: {n
}")
```

```
Min Error on prediction of seen data for team 1 scores: 2.49974125
5590493e-05
Max Error on prediction of seen data for team 1 scores: 1.96709978
44090934
Mean Error on prediction of seen data for team 1 scores: 0.1066628
6142508104
```

UNSEEN DATA (Test Set)

I have chosen a team(team 1) and summarized the actual score, predicted score, the error that came up with linear regression and the root mean square error too.

In [59]:
```
Team1_Score_predictor_unseen = pd.DataFrame({ "Actual_Score": T1Y_test, "Predicted_score": T1Predict_test})
Team1_Score_predictor_unseen['Error']= abs(Team1_Score_predictor_unseen.Predicted_score-Team1_Score_predictor_unseen.Actual_Score)
Team1_Score_predictor_unseen['RMS_Error']= np.sqrt(((Team1_Score_predictor_unseen.Predicted_score-Team1_Score_predictor_unseen.Actual_Score)**2).mean())
Team1_Score_predictor_unseen.head()
```

Out[59]:

	Actual_Score	Predicted_score	Error	RMS_Error
700	3.0	3.100006	0.100006	0.174641
17213	2.0	2.146320	0.146320	0.174641
17376	3.0	3.141952	0.141952	0.174641
15318	1.0	1.093803	0.093803	0.174641
25988	2.0	2.138932	0.138932	0.174641

Analysis of errors

In [60]:
```
a=Team1_Score_predictor_unseen['Error'].min()
b=Team1_Score_predictor_unseen['Error'].max()
c=Team1_Score_predictor_unseen['Error'].mean()
print(f"Min Error on prediction of unseen data for team 1 scores: {a}")
print(f"Max Error on prediction of unseen data for team 1 scores: {b}")
print(f"Mean Error on prediction of unseen data for team 1 scores: {c}")
```

```
Min Error on prediction of unseen data for team 1 scores: 7.395870
38012246e-06
Max Error on prediction of unseen data for team 1 scores: 2.055267
187256323
Mean Error on prediction of unseen data for team 1 scores: 0.10746
5800088584
```

In [ ]:

In [ ]: