Scouting Potential Premier League Signings with Machine Learning

Aaron Kochman

Project Overview

- Project Question: How can teams better manage player inventory and always have replacement options with valuation in mind?
  - Can machine learning models be used to manage squads?
- Target League: English Premier League (EPL)
  - Most valued league in global football at $10.63bn (https://www.transfermarkt.us/wettbewerbe/europa)
- Data Sources:
  - Transfermarkt, SoFIFA (EA Sports FIFA), Fantasy Premier League
- Prediction models:
  - Linear Regression: XGBoost, Sklearn

Business Case: Premier League Scouting

- Can machine learning models be used to produce a scouting report for team management?
  - Scout players with similar skills of existing players
  - Predict transfer values for scouted players
  - Recommend players to improve squad

Premier League

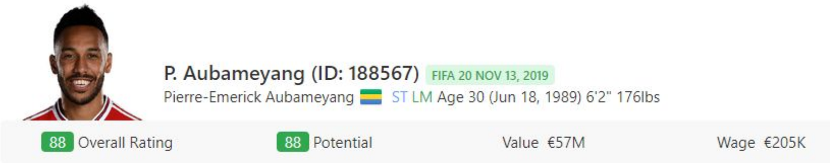
Hypotheses and Prediction Models

Null Hypothesis (H0): There is no relationship between player values and skill level.

Alternative Hypothesis (Ha): There is some relationship between player values and skill level.

Data Sources

- Data Sources:
  - Transfermarkt
  - SoFIFA (EA Sports FIFA)
  - Fantasy Premier League



Data Wrangling

- Data Sources:
  - Transfermarkt
    - Beautiful Soup HTML wrangling
  - SoFIFA (EA Sports FIFA)
    - CSV provided by Kaggle user stefanoleone992, web scraped (https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset)
  - Fantasy Premier League
    - Fantasy Premier League API request
      - JSON parsing

```python
if __name__ == "__main__":
    scraper = PageScraper()
    soup = scraper( LEAGUES_URL)
    LeagueTables = soup.find("table", class_="items").find("tbody")
    Leagues = LeagueTables.find_all("a", href=re.compile("wettbewerb/[A-Z]{2}1"), title=re.compile("\w"))
    Leagues = Leagues[:N_LEAGUES]
    LeagueUrlDic = { league.text : BASE_URL + league["href"] for league in Leagues}
    LeaguesData = []
    for leagueName, leagueUrl in LeagueUrlDic.items():
        print( "Scraping the %s..." %leagueName)
        LeaguesData.append( League( leagueName, leagueUrl, scraper))

    #flattening all players information to pandas.DataFrame and exporting to csv
    PlayerProfiles = [player.PlayerData for league in LeaguesData for team in league.TeamsData for player in team.PlayersData]
    df = pd.DataFrame( PlayerProfiles)
    df.to_csv("transfer.csv", index=False)
```

```
Scraping the Premier League...
['17/18', 'Jul 1, 2017', 'Benfica', 'Man City', '22,00 mil. €', '40,00 mil. €']
['15/16', 'Jul 1, 2015', 'Rio Ave FC', 'Benfica', '1,20 mil. €', '500 K €']
['12/13', 'Jul 1, 2012', 'GD Ribeirão', 'Rio Ave FC', 0, 'Free transfer']
['11/12', 'Jul 1, 2011', 'Benfica U19', 'GD Ribeirão', 0, 'Free transfer']
['10/11', 'Jul 1, 2010', 'Benfica U17', 'Benfica U19', 0, 0]
['09/10', 'Jan 1, 2010', 'São Paulo U17', 'Benfica U17', 0, '?']
        Ederson done
```

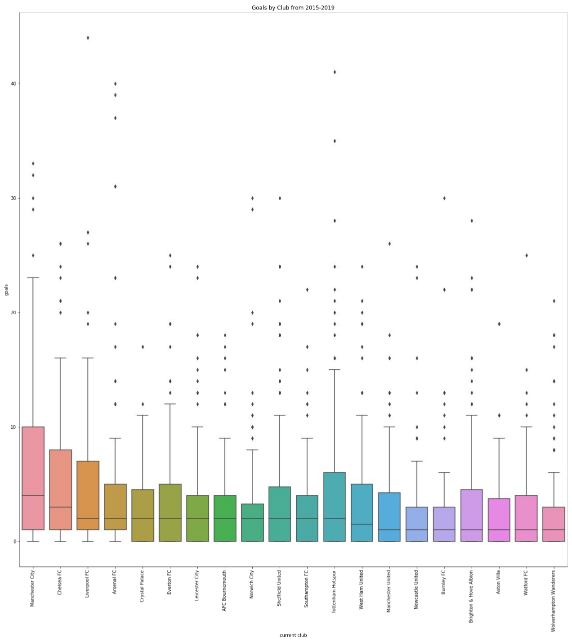

```
▼ root: {} 8 keys
  ▶ events: [] 38 items
  ▶ game_settings: {} 22 keys
  ▶ phases: [] 11 items
  ▶ teams: [] 20 items
    total_players: 6801617
  ▼ elements: [] 541 items
```

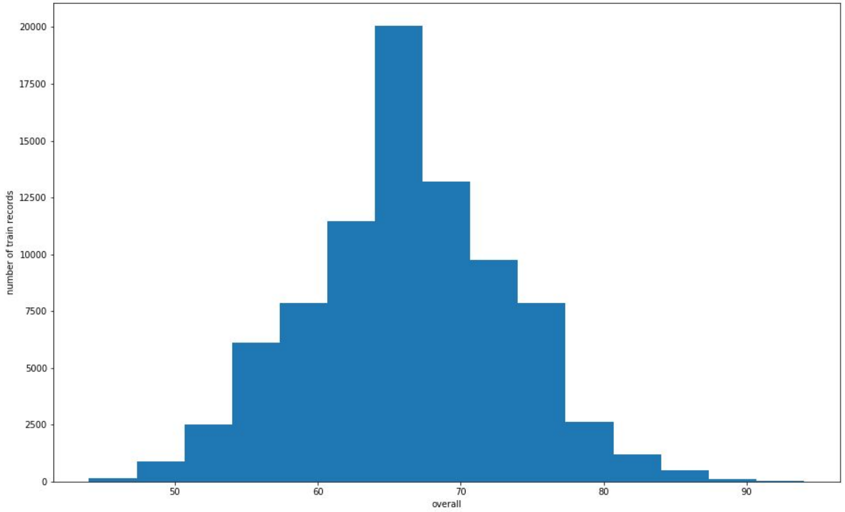
Data Compiling and Matching Player Names with FuzzyWuzzy

```
# List for dicts for easy dataframe creation
dict_list = []
# iterating over our players without salaries found above
for name in df_fifa.short_name:
    # Use our method to find best match, we can set a threshold here
    match = match_name(name, df_prem_field_players.name, 60)

    # New dict for storing data
    dict_ = {}
    dict_.update({"fifa_name" : name})
    dict_.update({"transfermarkt_name" : match[0]})
    dict_.update({"score" : match[1]})
    dict_list.append(dict_)

merge_table = pd.DataFrame(dict_list)
# Display results
merge_table.head()
```

	fifa_name	transfermarkt_name	score
0	K. De Bruyne	Kevin De Bruyne	81
1	V. van Dijk	Virgil van Dijk	77
2	M. Salah	Mohamed Salah	67
3	H. Kane	Harry Kane	71
4	Alisson		-1

Exploratory Analysis





Data Preparation for Regression

- Only used FIFA dataset to avoid losing players due to matching errors between datasets
- FIFA dataset consisted of 6 PCA components instead of all FIFA metrics
- Parsed current Arsenal players out from dataset and used KDTree to find nearest neighbors of Arsenal players

```
df_arsenal = df.query('club == "Arsenal" & year == "18/19"')
```

```
#Importing KDTree

from sklearn.neighbors import KDTree

kdt = KDTree(df[['pc1', 'pc2', 'pc3', 'pc4', 'pc5', 'pc6']])
```

```
#Using KDTree to find 4 players similar to that of Arsenal Players

dist, idx = kdt.query(df_arsenal_pca, k=5)
```

```
idx = idx.flatten()
```

```
idx
```

```
array([14820, 14889, 15003, 14915, 14908, 14830, 14974, 14859, 14993,
       15204, 14878, 14883, 14897, 14943, 14940, 14889, 15111, 14915,
       14820, 14912, 14898, 14831, 15025, 14843, 14980, 14907, 14903,
       15008, 14913, 15041, 14991, 15135, 15157, 15352, 15700, 14997,
       15527, 15024, 15115, 15239, 15013, 14944, 14909, 15697, 15138,
       15019, 14884, 15083,  9858, 15113, 15078,  9962, 15587, 15995,
       16075, 15148, 17032,    44, 15136, 15470, 15171,     5, 15058,
       16143, 15540, 15201, 15404, 15233, 15600, 15210, 15228, 15201,
       15404,  6407, 15600, 15431, 15285, 15297, 15882, 15445, 15467,
       15613, 16096, 15364, 17551, 15825, 14936, 15748, 16636,  6408,
       15900, 17200, 17675, 10291, 10263, 16711, 10667, 17384, 17540,
       17508, 17816, 17862, 22010, 20312, 23475, 18347, 22452,   244,
        6940, 21828, 20274, 10521, 22576, 19323, 19441, 20763, 25431,
       21302, 11613, 23731, 20847, 19945, 20999, 22850, 21927, 21742,
       22575, 20018, 21115, 19622, 23848, 12554, 25731, 21845, 12453,
       24700,   947,  1004, 24101, 12073, 25893, 26094, 26864, 28034,
        8589, 27417, 12956, 25909, 26289, 13652, 27438,  2006, 13037,
       26876, 26913, 28119, 27667, 28288, 27858,  1863, 29246,  2150,
       30674, 29889,  2653], dtype=int64)
```

Nearest Neighbors

- Cleaned up NN output from KDTree and included Arsenal player in data frame
- Dropped Arsenal player if NN was another Arsenal player

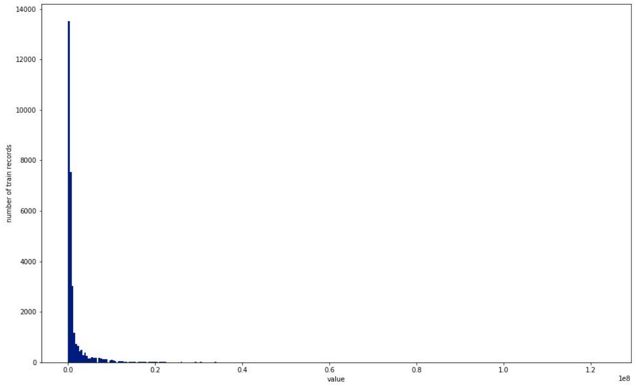
	pc1	pc2	pc3	pc4	pc5	pc6	value_eur	short_name	club	transfer
67812	-6.90	-2.46	2.86	-1.86	1.56	-0.24	50500000	P. Aubameyang	Arsenal	P. Aubameyang
67890	-6.70	-2.19	2.54	-1.34	1.36	-0.42	36500000	A. Lacazette	Arsenal	P. Aubameyang
68017	-6.17	-2.19	2.40	-1.98	1.13	-0.36	26000000	C. Bakambu	Beijing Sinobo Guoan FC	P. Aubameyang
67923	-7.08	-2.47	2.41	-1.07	1.15	0.02	35000000	M. Depay	Olympique Lyonnais	P. Aubameyang
67916	-6.77	-2.35	1.78	-1.93	1.88	-0.39	41000000	Gabriel Jesus	Manchester City	P. Aubameyang
67825	-5.92	-3.68	2.94	2.00	2.28	0.14	43500000	M. Özil	Arsenal	M. Özil
67986	-5.94	-3.28	2.47	1.70	2.09	0.28	30000000	Luis Alberto	Lazio	M. Özil
67856	-6.26	-3.93	1.93	2.39	2.48	-0.22	17000000	F. Ribéry	FC Bayern München	M. Özil
68007	-5.68	-2.75	1.91	1.37	2.23	0.05	27000000	E. Forsberg	RB Leipzig	M. Özil
68241	-5.95	-3.26	1.70	2.05	1.46	-0.24	13000000	Nani	Sporting CP	M. Özil
67878	6.99	0.12	2.62	-1.18	4.47	0.05	27000000	B. Leno	Arsenal	B. Leno
67884	6.80	-0.24	2.48	-1.22	4.63	0.11	26000000	W. Szczęsny	Juventus	B. Leno
67902	7.10	-0.28	2.80	-1.27	4.32	0.02	19000000	S. Ruffier	AS Saint-Étienne	B. Leno
67952	7.33	-0.27	2.78	-0.93	4.55	0.14	13000000	S. Mandanda	Olympique de Marseille	B. Leno
67949	6.76	-0.29	2.31	-1.61	4.66	0.12	6000000	Pepe Reina	Milan	B. Leno

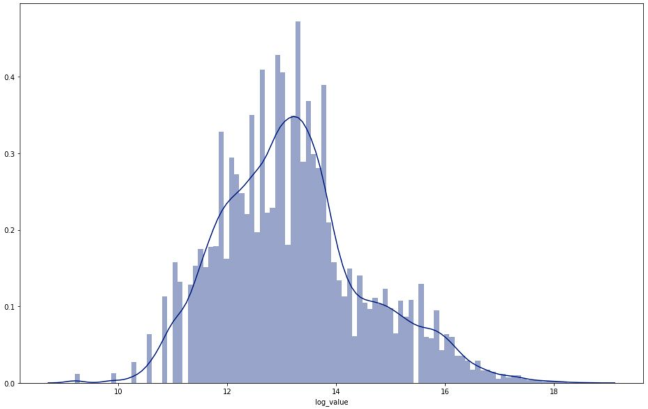
```
nn_df = nn_df.loc[nn_df['club']!='Arsenal']
nn_df
```

	pc1	pc2	pc3	pc4	pc5	pc6	value_eur	short_name	club	transfer
68017	-6.17	-2.19	2.40	-1.98	1.13	-0.36	26000000	C. Bakambu	Beijing Sinobo Guoan FC	P. Aubameyang
67923	-7.08	-2.47	2.41	-1.07	1.15	0.02	35000000	M. Depay	Olympique Lyonnais	P. Aubameyang
67916	-6.77	-2.35	1.78	-1.93	1.88	-0.39	41000000	Gabriel Jesus	Manchester City	P. Aubameyang
67986	-5.94	-3.28	2.47	1.70	2.09	0.28	30000000	Luis Alberto	Lazio	M. Özil
67856	-6.26	-3.93	1.93	2.39	2.48	-0.22	17000000	F. Ribéry	FC Bayern München	M. Özil
68007	-5.68	-2.75	1.91	1.37	2.23	0.05	27000000	E. Forsberg	RB Leipzig	M. Özil
68241	-5.95	-3.26	1.70	2.05	1.46	-0.24	13000000	Nani	Sporting CP	M. Özil
67884	6.80	-0.24	2.48	-1.22	4.63	0.11	26000000	W. Szczęsny	Juventus	B. Leno
67902	7.10	-0.28	2.80	-1.27	4.32	0.02	19000000	S. Ruffier	AS Saint-Étienne	B. Leno
67952	7.33	-0.27	2.78	-0.93	4.55	0.14	13000000	S. Mandanda	Olympique de Marseille	B. Leno
67949	6.76	-0.29	2.31	-1.61	4.66	0.12	6000000	Pepe Reina	Milan	B. Leno

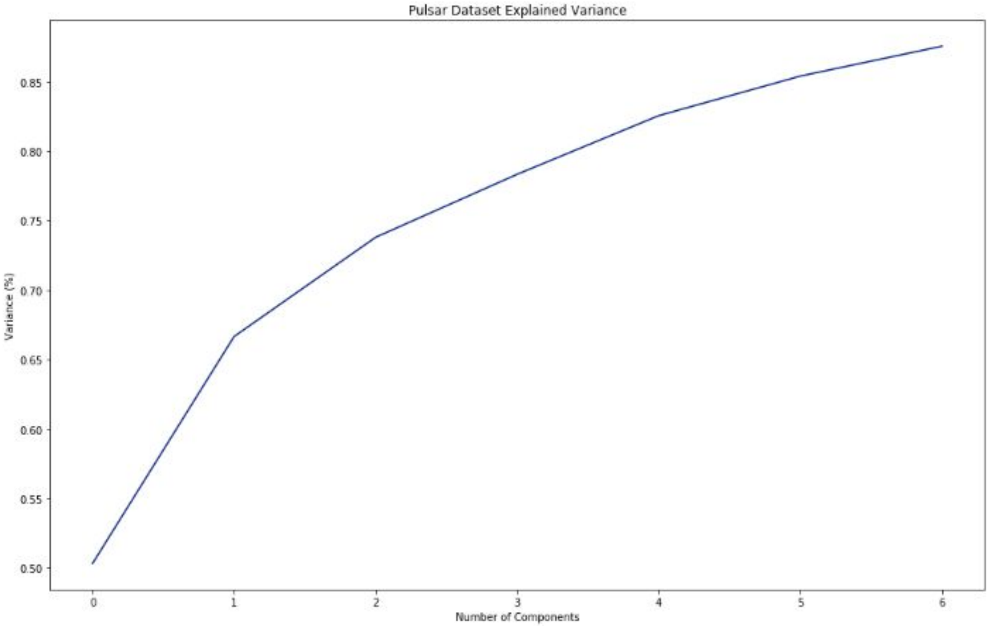
Transformations - Natural Log

- Transformed the entire training dataset with the natural log to produce a normal distribution.





Principal Component Analysis




```
# Separating out the features
x = df.loc[:, features].values
# Separating out the target
y = df.loc[:,['player_position_value']].values
# Standardizing the features
x = StandardScaler().fit_transform(x)
```

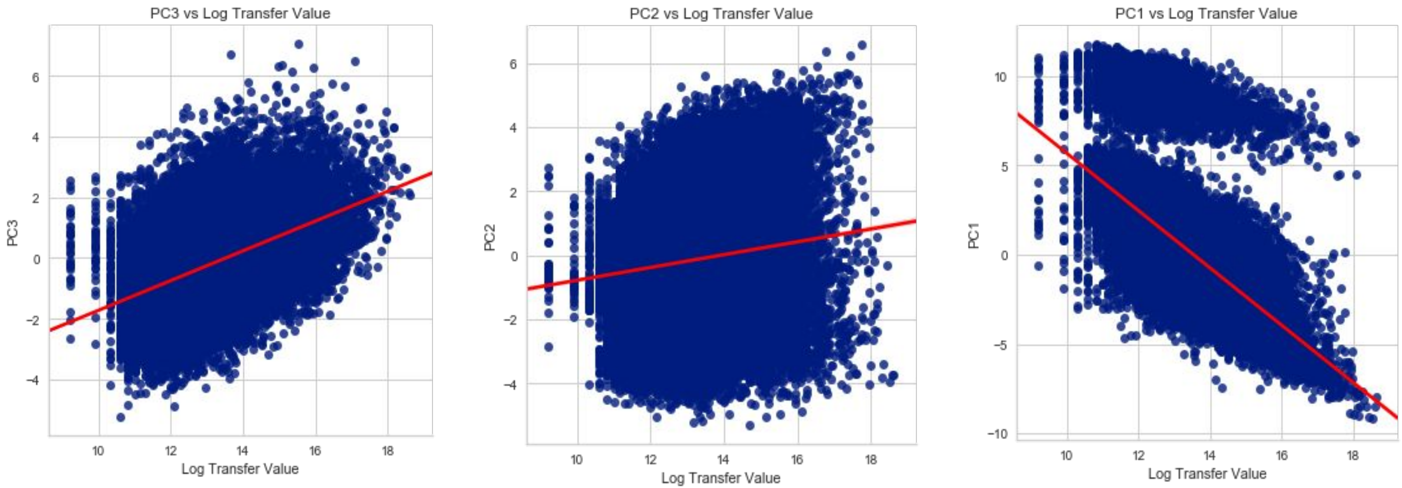
```
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['pc1', 'pc2', 'pc3', 'pc4', 'pc5', 'pc6', 'pc7'])
principalDf.head()
```

	pc1	pc2	pc3	pc4	pc5	pc6	pc7
0	-9.673541	-3.264636	2.994756	0.261758	3.335638	-0.052000	0.332797
1	-8.996034	-2.390310	4.182478	-2.734787	2.145011	-0.223099	-0.184125
2	-8.799608	-3.958752	2.177315	0.404965	3.250459	-0.191954	0.780470
3	5.379114	0.150147	3.657284	-1.635847	5.435309	0.400977	0.608621
4	-8.583742	-3.494537	2.120919	0.221559	2.952953	-0.217184	1.246865

```
pca.explained_variance_ratio_.cumsum()
```

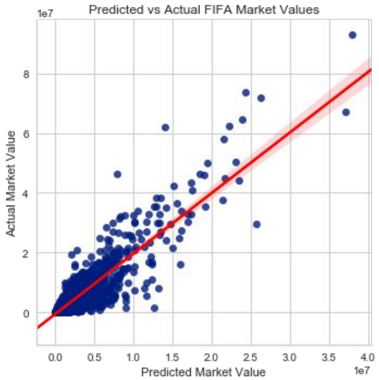
```
array([0.50318034, 0.66650386, 0.73790251, 0.78323356, 0.82541323,
       0.85386522, 0.87537854])
```

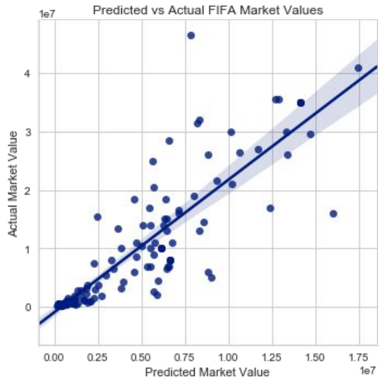
PCA Components and Log Values



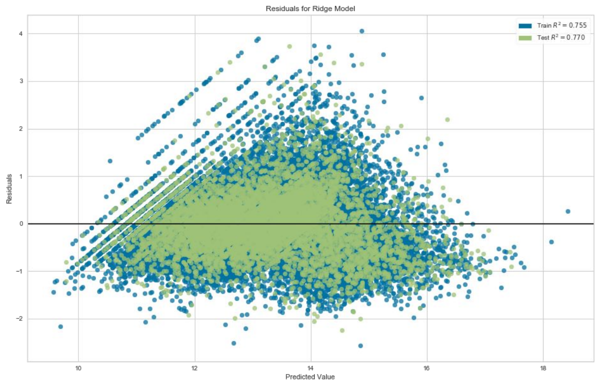
Scikit-learn Linear Regression

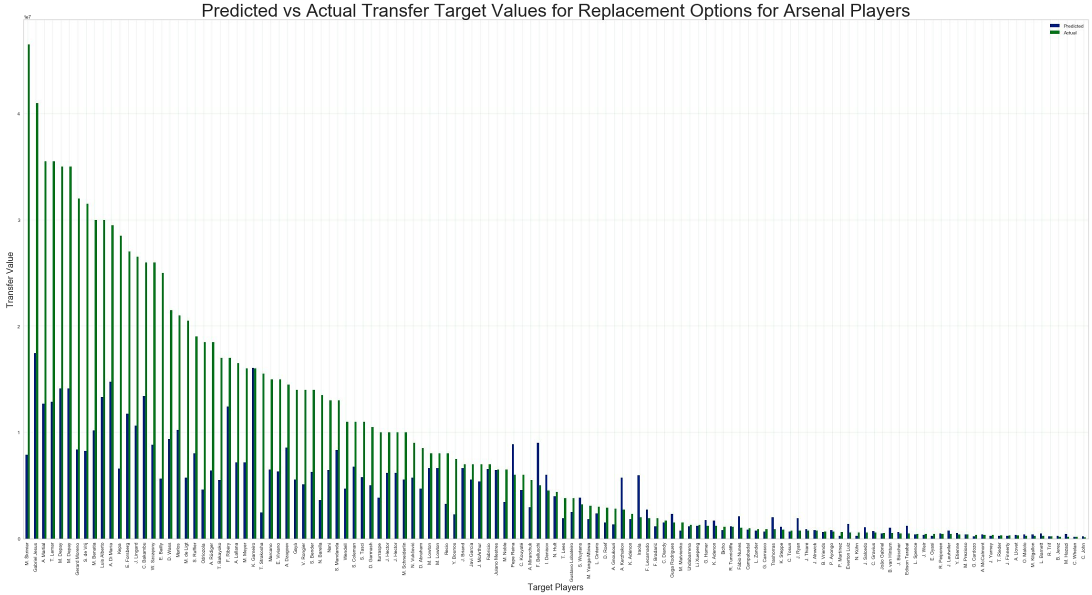
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_value   R-squared:                       0.771
Model:                            OLS   Adj. R-squared:                  0.770
Method:                 Least Squares   F-statistic:                     3523.
Date:                Fri, 15 Nov 2019   Prob (F-statistic):               0.00
Time:                        16:23:18   Log-Likelihood:                -6263.9
No. Observations:                6302   AIC:                         1.254e+04
Df Residuals:                    6295   BIC:                         1.259e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         13.4756      0.009   1504.356      0.000      13.458      13.493
pc1           -0.1896      0.002    -87.258      0.000      -0.194      -0.185
pc2            0.0565      0.004     14.598      0.000       0.049       0.064
pc3            0.4217      0.006     74.585      0.000       0.411       0.433
pc4           -0.1353      0.007    -18.930      0.000      -0.149      -0.121
pc5            0.5879      0.008     78.044      0.000       0.573       0.603
pc6            0.0158      0.012      1.277      0.202      -0.008       0.040
==============================================================================
Omnibus:                      636.284   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1332.066
Skew:                          -0.643   Prob(JB):                    5.57e-290
Kurtosis:                       4.849   Cond. No.                         6.07
==============================================================================
```



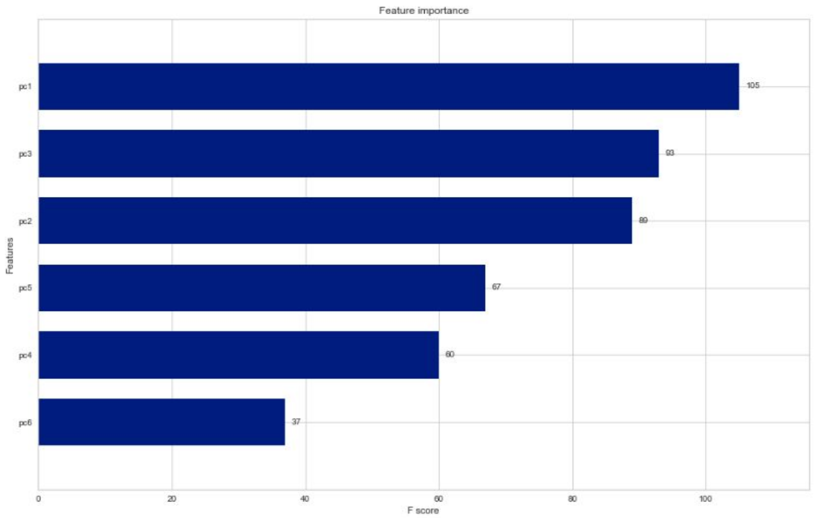


Scikit-learn Linear Regression





XGBoost Linear Regression




```
import xgboost as xgb
Train_Master = df_pca.drop(['value_eur', 'club', 'short_name'], axis=1)
Test_Master = df_pca.drop(['log_value','value_eur', 'club', 'short_name'], axis=1)
```

```
Train_Master.shape, Test_Master.shape
```

```
((31023, 7), (31023, 6))
```

```
Train, Test = train_test_split(Train_Master[0:100000], test_size = 0.2)
```

```
X_train = Train.drop(['log_value'], axis=1)
Y_train = Train["log_value"]
X_test = Test.drop(['log_value'], axis=1)
Y_test = Test["log_value"]
```

```
Test_Selection = Test_Master.loc[Test_Master.index.isin(selection)]
```

```
dtrain = xgb.DMatrix(X_train, label=Y_train)
dvalid = xgb.DMatrix(X_test, label=Y_test)
dtest = xgb.DMatrix(Test_Selection)
watchlist = [(dtrain, 'train'), (dvalid, 'valid')]
```

Scouting Report

Scikit-learn Regression

Target	Club	Arsenal Player	Predicted Value	FIFA Value
L. Cinterio	Deportes Iquique	A. Iwobi	1,546,289.62	3,000,000.00
A. Miranchuk	Lokomotiv Moscow	A. Iwobi	2,392,648.75	5,500,000.00
Gustavo Lobateiro	Internacional	A. Iwobi	1,546,289.62	3,800,000.00
F. Lecarnado	CD Palestino	A. Iwobi	2,260,594.50	1,900,000.00
A. Martial	Manchester United	A. Lacazette	21,959,924.00	35,500,000.00
M. Depay	Olympique Lyonnais	A. Lacazette	25,139,656.00	35,000,000.00
K. Gameiro	Valencia CF	A. Lacazette	19,790,724.00	16,000,000.00
J. Ryan	Blackpool	A. Maitland-Niles	1,734,610.25	725,000.00
Fábio Nunes	CD Tondela	A. Maitland-Niles	2,736,068.50	1,000,000.00
G. Hamer	PEC Zwolle	A. Maitland-Niles	2,317,341.25	1,200,000.00
Guga Rodrigues	Famalicão	A. Maitland-Niles	2,317,341.25	1,500,000.00
A. Dzagoev	PFC CSKA Moscow	A. Ramsey	10,294,118.00	14,500,000.00
A. Lallana	Liverpool	A. Ramsey	10,294,118.00	16,500,000.00
F. Belluschi	San Lorenzo de Almagro	A. Ramsey	10,294,118.00	5,000,000.00
D. Wass	Valencia CF	A. Ramsey	16,501,585.00	21,500,000.00
W. Szczęsny	Juventus	B. Leno	11,982,933.00	26,000,000.00
Pepe Reina	Milan	B. Leno	8,682,200.00	6,000,000.00
S. Ruffier	AS Saint-Étienne	B. Leno	14,531,549.00	19,000,000.00
S. Mandanda	Olympique de Marseille	B. Leno	14,531,549.00	13,000,000.00
Y. Etienne	KSV Cercle Brugge	C. Bramall	410,872.50	400,000.00
R. Peiponen	HJK Helsinki	C. Bramall	419,706.91	425,000.00
M. Hazazi	Al Fateh	C. Bramall	414,493.28	180,000.00
O. Malolo	HJK Helsinki	C. Bramall	378,391.78	270,000.00
C. Diandy	Sporting de Charleroi	C. Jenkinson	1,092,502.25	1,700,000.00
A. Gnoukouri	Inter	C. Jenkinson	606,718.94	2,800,000.00
Éverton Luiz	SPAL	C. Jenkinson	727,804.00	575,000.00
F. Bradarić	Cagliari	C. Jenkinson	553,731.38	1,900,000.00

XGBoost Regression

Target	Club	Arsenal Player	Predicted	FIFA Value
L. Cinterio	Deportes Iquique	A. Iwobi	2,344,928.43	3,000,000.00
Gustavo Lobateiro	Internacional	A. Iwobi	2,515,089.87	3,800,000.00
A. Miranchuk	Lokomotiv Moscow	A. Iwobi	2,937,542.59	5,500,000.00
F. Lecarnado	CD Palestino	A. Iwobi	2,704,041.23	1,900,000.00
A. Martial	Manchester United	A. Lacazette	12,690,951.66	35,500,000.00
M. Depay	Olympique Lyonnais	A. Lacazette	14,143,666.62	35,000,000.00
K. Gameiro	Valencia CF	A. Lacazette	16,049,719.37	16,000,000.00
J. Ryan	Blackpool	A. Maitland-Niles	1,904,390.74	725,000.00
Fábio Nunes	CD Tondela	A. Maitland-Niles	2,071,167.30	1,000,000.00
G. Hamer	PEC Zwolle	A. Maitland-Niles	1,726,154.43	1,200,000.00
Guga Rodrigues	Famalicão	A. Maitland-Niles	2,299,463.34	1,500,000.00
F. Belluschi	San Lorenzo de Almagro	A. Ramsey	9,028,382.92	5,000,000.00
A. Lallana	Liverpool	A. Ramsey	7,177,730.19	16,500,000.00
A. Dzagoev	PFC CSKA Moscow	A. Ramsey	8,572,936.88	14,500,000.00
D. Wass	Valencia CF	A. Ramsey	9,345,069.79	21,500,000.00
S. Ruffier	AS Saint-Étienne	B. Leno	8,021,038.78	19,000,000.00
W. Szczęsny	Juventus	B. Leno	8,847,357.80	26,000,000.00
S. Mandanda	Olympique de Marseille	B. Leno	8,335,649.69	13,000,000.00
Pepe Reina	Milan	B. Leno	8,866,799.44	6,000,000.00
R. Peiponen	HJK Helsinki	C. Bramall	479,100.56	425,000.00
Y. Etienne	KSV Cercle Brugge	C. Bramall	518,463.33	400,000.00
O. Malolo	HJK Helsinki	C. Bramall	382,448.54	270,000.00
M. Hazazi	Al Fateh	C. Bramall	406,941.62	180,000.00
C. Diandy	Sporting de Charleroi	C. Jenkinson	1,528,691.78	1,700,000.00
F. Bradarić	Cagliari	C. Jenkinson	1,142,106.63	1,900,000.00
A. Gnoukouri	Inter	C. Jenkinson	1,326,691.13	2,800,000.00
Éverton Luiz	SPAL	C. Jenkinson	1,351,597.84	575,000.00

Conclusion

- Reject the null hypothesis that there is no relationship between player values and skill level
- Inflation among high valued players
    - Running regression on quality tiers of players could eliminate this factor