Load the old model developed in Part I

In [1]:
```
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [2]:
```
import pandas as pd
df = pd.read_csv("nba.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 486 entries, 0 to 485
Data columns (total 52 columns):
Number                          486 non-null int64
Season_Start                    486 non-null int64
Player_Name                     486 non-null object
Salary                          484 non-null float64
Position                        486 non-null object
Age                             486 non-null int64
Team                            486 non-null object
Games_Played                    486 non-null int64
Games_Started                   486 non-null int64
Minutes_Played                  486 non-null int64
Player_Efficiency_Rating        486 non-null float64
True_Shooting_Pct               485 non-null float64
Three_Point_Attempt_Rate        485 non-null float64
Free_Throw_Rate                 485 non-null float64
Offensive_Rebound_Pct           486 non-null float64
Defensive_Rebound_Pct           486 non-null float64
Total_Rebound_Pct               486 non-null float64
Assists_Pct                     486 non-null float64
Steals_Pct                      486 non-null float64
Blocks_Pct                      486 non-null float64
Turnovers_Pct                   485 non-null float64
Usage_Pct                       486 non-null float64
Offensive_Win_Shares            486 non-null float64
Defensive_Win_Shares            486 non-null float64
Win_Shares                      486 non-null float64
Win_Shares/48                   486 non-null float64
Offensive_Box_Plus_Minus        486 non-null float64
Defensive_Box_Plus_Minus        486 non-null float64
Box_Plus_Minus                  486 non-null float64
Value_Over_Replacement_Player   486 non-null float64
Field_Goals                     486 non-null int64
Field_Goals_Attempted           486 non-null int64
Field_Goal_Pct                  485 non-null float64
Three_Pointers                  486 non-null int64
Three_Pointers_Attempted        486 non-null int64
Three_Pointers_Pct              450 non-null float64
Two_Pointers                    486 non-null int64
Two_Pointers_Attempted          486 non-null int64
Two_Pointers_Pct                484 non-null float64
Effective_Field_Goal_Pct        485 non-null float64
Free_Throws                     486 non-null int64
Free_Throws_Attempted           486 non-null int64
Free_Throws_Pct                 471 non-null float64
Offensive_Rebounds              486 non-null int64
Defensive_Rebounds              486 non-null int64
Total_Rebounds                  486 non-null int64
Assists                         486 non-null int64
Steals                          486 non-null int64
Blocks                          486 non-null int64
Turnovers                       486 non-null int64
```

```
Personal_Fouls                  486 non-null int64
Points                          486 non-null int64
dtypes: float64(26), int64(23), object(3)
memory usage: 197.5+ KB
```

This code chunk will change all of object classes to numerical values

In [3]:
```
# Data transformation
# Convert categorical values to numeric using label encoder
from sklearn import preprocessing
from collections import defaultdict
d = defaultdict(preprocessing.LabelEncoder)
# Encoding the categorical variable
fit = df.select_dtypes(include=['object']).apply(lambda x: d[x.name].fit_transfor

#Convert the categorical columns based on encoding
for i in list(d.keys()):
    df[i] = d[i].transform(df[i])
```

The `features` DataFrame is everything but Salary (which is what we want to predict)

Salary will be its own DataFrame called `labels`

In [4]:
```
features = df[df.columns.difference(["Salary"])]
labels = df["Salary"]
```

There are some null values that we need to replace with 0

In [5]:
```
pd.set_option('display.max_rows', 500)
labels.isnull()
```

Out[5]:
```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
```

In [6]:
```
labels = labels.fillna(0)
```

In [7]:
```
pd.set_option('display.max_columns', 500)
features.isnull().values.any()
```

Out[7]:
```
True
```

In [8]:
```
features.isnull().sum()
```

Out[8]:
```
Age                              0
Assists                          0
Assists_Pct                      0
Blocks                           0
Blocks_Pct                       0
Box_Plus_Minus                   0
Defensive_Box_Plus_Minus         0
Defensive_Rebound_Pct            0
Defensive_Rebounds               0
Defensive_Win_Shares             0
Effective_Field_Goal_Pct         1
Field_Goal_Pct                   1
Field_Goals                      0
Field_Goals_Attempted            0
Free_Throw_Rate                  1
Free_Throws                      0
Free_Throws_Attempted            0
Free_Throws_Pct                 15
Games_Played                     0
Games_Started                    0
Minutes_Played                   0
Number                           0
Offensive_Box_Plus_Minus         0
Offensive_Rebound_Pct            0
Offensive_Rebounds               0
Offensive_Win_Shares             0
Personal_Fouls                   0
Player_Efficiency_Rating         0
Player_Name                      0
Points                           0
Position                         0
Season_Start                     0
Steals                           0
Steals_Pct                       0
Team                             0
Three_Point_Attempt_Rate         1
Three_Pointers                   0
Three_Pointers_Attempted         0
Three_Pointers_Pct              36
Total_Rebound_Pct                0
Total_Rebounds                   0
True_Shooting_Pct                1
Turnovers                        0
Turnovers_Pct                    1
Two_Pointers                     0
Two_Pointers_Attempted           0
Two_Pointers_Pct                 2
Usage_Pct                        0
Value_Over_Replacement_Player    0
Win_Shares                       0
Win_Shares/48                    0
dtype: int64
```

In [9]:
```
features = features.fillna(0)
features.isnull().sum()
```

Out[9]:
```
Age                              0
Assists                          0
Assists_Pct                      0
Blocks                           0
Blocks_Pct                       0
Box_Plus_Minus                   0
Defensive_Box_Plus_Minus         0
Defensive_Rebound_Pct            0
Defensive_Rebounds               0
Defensive_Win_Shares             0
Effective_Field_Goal_Pct         0
Field_Goal_Pct                   0
Field_Goals                      0
Field_Goals_Attempted            0
Free_Throw_Rate                  0
Free_Throws                      0
Free_Throws_Attempted            0
Free_Throws_Pct                  0
Games_Played                     0
Games_Started                    0
Minutes_Played                   0
Number                           0
Offensive_Box_Plus_Minus         0
Offensive_Rebound_Pct            0
Offensive_Rebounds               0
Offensive_Win_Shares             0
Personal_Fouls                   0
Player_Efficiency_Rating         0
Player_Name                      0
Points                           0
Position                         0
Season_Start                     0
Steals                           0
Steals_Pct                       0
Team                             0
Three_Point_Attempt_Rate         0
Three_Pointers                   0
Three_Pointers_Attempted         0
Three_Pointers_Pct               0
Total_Rebound_Pct                0
Total_Rebounds                   0
True_Shooting_Pct                0
Turnovers                        0
Turnovers_Pct                    0
Two_Pointers                     0
Two_Pointers_Attempted           0
Two_Pointers_Pct                 0
Usage_Pct                        0
Value_Over_Replacement_Player    0
Win_Shares                       0
Win_Shares/48                    0
dtype: int64
```

Now we can subset the data to include only the relevant features defined in Part I of this project. These were:

Age, Player_Efficiency_Rating, Blocks, Turnovers_Pct, Steals, Assists_Pct, Games_Started, Games_Played, Total_Rebounds, Field_Goals, Defensive_Box_Plus_Minus, Defensive_Rebound_Pct, Usage_Pct, and Free_Throws

In [10]:
```
nba_subset = df[["Salary", "Age", "Player_Efficiency_Rating", "Blocks", "Turnover
nba_subset.head(5)
```

Out[10]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct	Games_St
0	1312611.0	24	8.4	13	16.4	1	3.8	
1	2116955.0	32	9.5	9	17.2	25	20.7	
2	5504420.0	21	14.4	40	8.5	64	10.5	
3	365289.0	22	-2.2	0	0.0	0	22.6	
4	2022240.0	25	14.4	7	11.4	8	8.0	

We now need to get our dataset ready for our model

The *first line* of code below creates an object of the target variable called "target_column".
The *second line* gives us the list of all the features, excluding the target variable of Salary.
The *third line* normalizes the predictors. This is done because the units of the variables differ significantly and may influence the modeling process. To prevent this, we will do normalization via scaling of the predictors between 0 and 1.
The *fourth line* displays the summary of the normalized data. We can see that all the independent variables have now been scaled between 0 and 1. The target variable (Salary) remains unchanged.

In [11]:
```
import warnings
warnings.filterwarnings("ignore")

target_column = ['Salary']
predictors = list(set(list(nba_subset.columns))-set(target_column))
nba_subset[predictors] = nba_subset[predictors]/nba_subset[predictors].max()
nba_subset.describe()
```

Out[11]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals
count	4.840000e+02	486.000000	486.000000	486.000000	485.000000	486.000000
mean	6.717244e+06	0.660134	0.413358	0.112342	0.293786	0.248395
std	7.376188e+06	0.108630	0.182934	0.136705	0.122302	0.208235
min	1.722400e+04	0.475000	-0.558730	0.000000	0.000000	0.000000
25%	1.471382e+06	0.575000	0.311111	0.023364	0.220183	0.082803
50%	3.343830e+06	0.650000	0.406349	0.070093	0.284404	0.210191
75%	1.004073e+07	0.725000	0.501587	0.149533	0.353211	0.361465
max	3.468255e+07	1.000000	1.000000	1.000000	1.000000	1.000000

Now to make the TEST and TRAINING datasets. We will be using a 70/30 train/test ratio

In [12]:
```
nba_subset.isnull().sum()
```

Out[12]:
```
Salary                    2
Age                       0
Player_Efficiency_Rating  0
Blocks                    0
Turnovers_Pct             1
Steals                    0
Assists_Pct               0
Games_Started             0
Games_Played              0
Total_Rebounds            0
Field_Goals               0
Defensive_Box_Plus_Minus  0
Defensive_Rebound_Pct     0
Usage_Pct                 0
Free_Throws               0
dtype: int64
```

In [13]:
```
nba_subset = nba_subset.fillna(0)
```

We now separate our data into TRAINING and TEST subsets. This will be at a 70/30 ratio.

In [14]:
```
X = nba_subset[predictors].values
y = nba_subset[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_
print(X_train.shape); print(X_test.shape)
```

```
(340, 14)
(146, 14)
```

Now we can run the Xgboost Regression model

In [15]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(X_train, y_train, verbose=False)
y_train_pred1 = xgb_model1.predict(X_train)
y_pred1 = xgb_model1.predict(X_test)

print('Train r2 score: ', r2_score(y_train_pred1, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, y_train)
test_mse1 = mean_squared_error(y_pred1, y_test)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
[13:19:54] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9221592427857032
Test r2 score:  0.6506232636580878
Train RMSE: 1818765.6973
Test RMSE: 4530414.8808
```

Loading the new data from our web scraping application

In [16]:
```
# This is a snippet for basic libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Loading the scraped DataFrame

In [17]:
```
final = pd.read_csv("final.csv")
final = final.dropna()
final.head(3)
```

Out[17]:

	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P
1	Arron Afflalo	SG	24.0	DEN	82.0	75.0	2221.0	272.0	585.0	0.465	108.0	249.0	0.434	164.0
3	LaMarcus Aldridge	PF	24.0	POR	78.0	78.0	2922.0	579.0	1169.0	0.495	5.0	16.0	0.313	574.0
5	Joe Alexander	SF	23.0	CHI	8.0	0.0	29.0	1.0	6.0	0.167	0.0	1.0	0.000	1.0

In [18]:
```
final.shape
```

Out[18]:
```
(3638, 50)
```

Changing the column names to the names we used in our previous model

In [19]:
```
final = final.rename(columns={'Player': 'Player_Name',
                              'Pos': 'Position',
                              'Age': 'Age',
                              'Tm': 'Team',
                              'G': 'Games_Played',
                              'GS': 'Games_Started',
                              'MP': 'Minutes_Played',
                              'FG': 'Field_Goals',
                              'FGA': 'Field_Goals_Attempted',
                              'FG%': 'Field_Goal_Pct',
                              '3P': 'Three_Pointers',
                              '3PA': 'Three_Pointers_Attempted',
                              '3P%': 'Three_Pointers_Pct',
                              '2P': 'Two_Pointers',
                              '2PA': 'Two_Pointers_Attempted',
                              '2P%': 'Two_Pointers_Pct',
                              'eFG%': 'Effective_Field_Goal_Pct',
                              'FT': 'Free_Throws',
                              'FTA': 'Free_Throws_Attempted',
                              'FT%': 'Free_Throws_Pct',
                              'ORB': 'Offensive_Rebounds',
                              'DRB': 'Defensive_Rebounds',
                              'TRB': 'Total_Rebounds',
                              'AST': 'Assists',
                              'STL': 'Steals',
                              'BLK': 'Blocks',
                              'TOV': 'Turnovers',
                              'PF': 'Personal_Fouls',
                              'PER': 'Player_Efficiency_Rating',
                              'TS%': 'True_Shooting_Pct',
                              '3PAr': 'Three_Point_Attempt_Rate',
                              'FTr': 'Free_Throw_Rate',
                              'ORB%': 'Offensive_Rebound_Pct',
                              'DRB%': 'Defensive_Rebound_Pct',
                              'TRB%': 'Total_Rebound_Pct',
                              'AST%': 'Assists_Pct',
                              'STL%': 'Steals_Pct',
                              'BLK%': 'Blocks_Pct',
                              'TOV%': 'Turnovers_Pct',
                              'USG%': 'Usage_Pct',
                              'OWS': 'Offensive_Win_Shares',
                              'DWS': 'Defensive_Win_Shares',
                              'WS': 'Win_Shares',
                              'WS/48': 'Win_Shares/48',
                              'OBPM': 'Offensive_Box_Plus_Minus',
                              'DBPM': 'Defensive_Box_Plus_Minus',
                              'BPM': 'Box_Plus_Minus',
                              'VORP': 'Value_Over_Replacement_Player',
                              'Year': 'Year',
                              'Salary': 'Salary'})
```

Separating the DataFrames into years

This code chunk will change all of object classes to numerical values

In [20]:
```
final_10 = final[final.Year.eq(2010)]
final_11 = final[final.Year.eq(2011)]
final_12 = final[final.Year.eq(2012)]
final_13 = final[final.Year.eq(2013)]
final_14 = final[final.Year.eq(2014)]
final_15 = final[final.Year.eq(2015)]
final_16 = final[final.Year.eq(2016)]
final_17 = final[final.Year.eq(2017)]
final_18 = final[final.Year.eq(2018)]
final_19 = final[final.Year.eq(2019)]
```

We need to make an individual dataframe of just the player names that we will be using later

In [21]:
```
names_10_actual = final_10["Player_Name"]
names_11_actual = final_11["Player_Name"]
names_12_actual = final_12["Player_Name"]
names_13_actual = final_13["Player_Name"]
names_14_actual = final_14["Player_Name"]
names_15_actual = final_15["Player_Name"]
names_16_actual = final_16["Player_Name"]
names_17_actual = final_17["Player_Name"]
names_18_actual = final_18["Player_Name"]
names_19_actual = final_19["Player_Name"]
```

In [22]:
```
# Data transformation
# Convert categorical values to numeric using label encoder
from sklearn import preprocessing
from collections import defaultdict
d = defaultdict(preprocessing.LabelEncoder)
# Encoding the categorical variable
fit = final.select_dtypes(include=['object']).apply(lambda x: d[x.name].fit_trans

#Convert the categorical columns based on encoding
for i in list(d.keys()):
    final[i] = d[i].transform(final[i])
```

Separating the DataFrames into years

In [23]:
```
final_10 = final[final.Year.eq(2010)]
final_11 = final[final.Year.eq(2011)]
final_12 = final[final.Year.eq(2012)]
final_13 = final[final.Year.eq(2013)]
final_14 = final[final.Year.eq(2014)]
final_15 = final[final.Year.eq(2015)]
final_16 = final[final.Year.eq(2016)]
final_17 = final[final.Year.eq(2017)]
final_18 = final[final.Year.eq(2018)]
final_19 = final[final.Year.eq(2019)]
```

We need to make an individual dataframe of just the player names that we will be using later

In [24]:
```
names_10 = final_10["Player_Name"]
names_11 = final_11["Player_Name"]
names_12 = final_12["Player_Name"]
names_13 = final_13["Player_Name"]
names_14 = final_14["Player_Name"]
names_15 = final_15["Player_Name"]
names_16 = final_16["Player_Name"]
names_17 = final_17["Player_Name"]
names_18 = final_18["Player_Name"]
names_19 = final_19["Player_Name"]
```

The `features` DataFrame is everything but Salary (which is what we want to predict)

Salary will be its own DataFrame called `labels`

In [25]:
```
features_10 = final_10[final_10.columns.difference(["Salary"])]
features_11 = final_11[final_11.columns.difference(["Salary"])]
features_12 = final_12[final_12.columns.difference(["Salary"])]
features_13 = final_13[final_13.columns.difference(["Salary"])]
features_14 = final_14[final_14.columns.difference(["Salary"])]
features_15 = final_15[final_15.columns.difference(["Salary"])]
features_16 = final_16[final_16.columns.difference(["Salary"])]
features_17 = final_17[final_17.columns.difference(["Salary"])]
features_18 = final_18[final_18.columns.difference(["Salary"])]
features_19 = final_19[final_19.columns.difference(["Salary"])]

labels_10 = final_10["Salary"]
labels_11 = final_11["Salary"]
labels_12 = final_12["Salary"]
labels_13 = final_13["Salary"]
labels_14 = final_14["Salary"]
labels_15 = final_15["Salary"]
labels_16 = final_16["Salary"]
labels_17 = final_17["Salary"]
labels_18 = final_18["Salary"]
labels_19 = final_19["Salary"]
```

The web scraping script automatically ignored players who did not have a reported salary. Therefore, we do not need to check for zero values in the Salary column.

Checking the features column for zero values

In [26]:
```
pd.set_option('display.max_columns', 500)
print(features_10.isnull().values.any())
print(features_11.isnull().values.any())
print(features_12.isnull().values.any())
print(features_13.isnull().values.any())
print(features_14.isnull().values.any())
print(features_15.isnull().values.any())
print(features_16.isnull().values.any())
print(features_17.isnull().values.any())
print(features_18.isnull().values.any())
print(features_19.isnull().values.any())
```

```
False
False
False
False
False
False
False
False
False
False
```

Since we already know the features we are going to include in our model, we can go ahead and make a subset with just those features.

In [27]:
```
nba_subset_10 = final_10[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_11 = final_11[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_12 = final_12[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_13 = final_13[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_14 = final_14[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_15 = final_15[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_16 = final_16[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_17 = final_17[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_18 = final_18[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
nba_subset_19 = final_19[["Salary", "Age", "Player_Efficiency_Rating", "Blocks",
```

We now need to get our dataset ready for our model

The *first line* of code below creates an object of the target variable called "target_column".
The *second line* gives us the list of all the features, excluding the target variable of Salary.
The *third line* normalizes the predictors. This is done because the units of the variables differ significantly and may influence the modeling process. To prevent this, we will do normalization via scaling of the predictors between 0 and 1.
The *fourth line* displays the summary of the normalized data. We can see that all the independent variables have now been scaled between 0 and 1. The target variable (Salary) remains unchanged

In [28]:
```
import warnings
warnings.filterwarnings("ignore")

target_column_10 = ['Salary']
target_column_11 = ['Salary']
target_column_12 = ['Salary']
target_column_13 = ['Salary']
target_column_14 = ['Salary']
target_column_15 = ['Salary']
target_column_16 = ['Salary']
target_column_17 = ['Salary']
target_column_18 = ['Salary']
target_column_19 = ['Salary']

predictors_10 = list(set(list(nba_subset_10.columns))-set(target_column_10))
predictors_11 = list(set(list(nba_subset_11.columns))-set(target_column_11))
predictors_12 = list(set(list(nba_subset_12.columns))-set(target_column_12))
predictors_13 = list(set(list(nba_subset_13.columns))-set(target_column_13))
predictors_14 = list(set(list(nba_subset_14.columns))-set(target_column_14))
predictors_15 = list(set(list(nba_subset_15.columns))-set(target_column_15))
predictors_16 = list(set(list(nba_subset_16.columns))-set(target_column_16))
predictors_17 = list(set(list(nba_subset_17.columns))-set(target_column_17))
predictors_18 = list(set(list(nba_subset_18.columns))-set(target_column_18))
predictors_19 = list(set(list(nba_subset_19.columns))-set(target_column_19))

nba_subset_10[predictors_10] = nba_subset_10[predictors_10]/nba_subset_10[predict
nba_subset_11[predictors_11] = nba_subset_11[predictors_11]/nba_subset_11[predict
nba_subset_12[predictors_12] = nba_subset_12[predictors_12]/nba_subset_12[predict
nba_subset_13[predictors_13] = nba_subset_13[predictors_13]/nba_subset_13[predict
nba_subset_14[predictors_14] = nba_subset_14[predictors_14]/nba_subset_14[predict
nba_subset_15[predictors_15] = nba_subset_15[predictors_15]/nba_subset_15[predict
nba_subset_16[predictors_16] = nba_subset_16[predictors_16]/nba_subset_16[predict
nba_subset_17[predictors_17] = nba_subset_17[predictors_17]/nba_subset_17[predict
nba_subset_18[predictors_18] = nba_subset_18[predictors_18]/nba_subset_18[predict
nba_subset_19[predictors_19] = nba_subset_19[predictors_19]/nba_subset_19[predict

#Just showing one of the datasets to make sure the normalization worked
nba_subset_19.describe()
```

Out[28]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals
count	3.980000e+02	398.000000	398.000000	398.000000	398.000000	398.000000
mean	7.067359e+06	0.622577	0.409578	0.129681	0.450433	0.247960
std	8.103553e+06	0.102421	0.147215	0.142415	0.140954	0.188784
min	4.737000e+03	0.452381	0.027356	0.000000	0.000000	0.000000
25%	1.512601e+06	0.547619	0.319149	0.035176	0.358209	0.100000
50%	3.369804e+06	0.619048	0.386018	0.085427	0.432836	0.211765
75%	1.000000e+07	0.690476	0.495441	0.170854	0.529851	0.347059
max	3.745715e+07	1.000000	1.000000	1.000000	1.000000	1.000000

Running 2010 Data in the Model

In [29]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_10, labels_10, verbose=False)
y_train_pred1 = xgb_model1.predict(features_10)
y_pred1 = xgb_model1.predict(features_10)

print('Train r2 score: ', r2_score(y_train_pred1, labels_10))
print('Test r2 score: ', r2_score(labels_10, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_10)
test_mse1 = mean_squared_error(y_pred1, labels_10)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
[13:21:13] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9319146481117234
Test r2 score:  0.9455469368276412
Train RMSE: 1099626.5634
Test RMSE: 1099626.5634
```

In [30]:
```
y_train_xgb = pd.DataFrame(labels_10, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_10, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[30]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct	C
0	1086240.0	0.615385	0.350482	0.131579	0.346535	0.243386	0.204846	
1	5844827.0	0.615385	0.585209	0.210526	0.244224	0.354497	0.218062	
2	2583360.0	0.589744	0.090032	0.004386	0.000000	0.005291	0.204846	
3	1300000.0	0.794872	0.189711	0.021930	0.504950	0.058201	0.112335	
4	2500000.0	0.717949	0.456592	0.083333	0.587459	0.312169	0.286344	

In [31]:
```
final_df = pd.concat([y_predict_xgb, final_10], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_10_actual], axis=1)
final_df = final_df.dropna()
final_df.head(5)
```

Out[31]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field_
1	6664083.500	10.0	24.0	82.0	75.0	2221.0	272.0	
3	1692061.875	2.0	24.0	78.0	78.0	2922.0	579.0	
5	2098974.500	7.0	23.0	8.0	0.0	29.0	1.0	
7	4331724.500	2.0	31.0	51.0	3.0	456.0	46.0	
9	1338712.875	10.0	28.0	54.0	8.0	889.0	129.0	

In [32]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [33]:
```
cols = list(final_df.columns.values)
```

In [34]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[34]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
1	Arron Afflalo	1086240.0	6664083	-5577843.500	24.0	82.0	75.0	
3	LaMarcus Aldridge	5844827.0	1692061	4152765.125	24.0	78.0	78.0	
5	Joe Alexander	2583360.0	2098974	484385.500	23.0	8.0	0.0	
7	Malik Allen	1300000.0	4331724	-3031724.500	31.0	51.0	3.0	
9	Tony Allen	2500000.0	1338712	1161287.125	28.0	54.0	8.0	

In [35]:
```
final_df_overpaid_10 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_10 = final_df_overpaid_10.head(10)
top_10_overpaid_10
```

Out[35]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Min
195	Tim Duncan	22183220.0	880920	2.130230e+07	33.0	78.0	77.0	
109	Kobe Bryant	23034375.0	3125619	1.990876e+07	31.0	73.0	73.0	
91	Elton Brand	14858472.0	907392	1.395108e+07	30.0	76.0	57.0	
133	Vince Carter	16123250.0	3793071	1.233018e+07	33.0	75.0	74.0	
21	Carmelo Anthony	15779912.0	4990881	1.078903e+07	25.0	69.0	69.0	
337	Andre Iguodala	12200000.0	1443273	1.075673e+07	26.0	82.0	82.0	
321	Josh Howard	10890000.0	242632	1.064737e+07	29.0	35.0	12.0	
87	Chris Bosh	15779912.0	5605472	1.017444e+07	25.0	70.0	70.0	
73	Chauncey Billups	12100000.0	2014230	1.008577e+07	33.0	73.0	73.0	
167	Baron Davis	12100000.0	2086629	1.001337e+07	30.0	75.0	73.0	

In [36]:
```
final_df_underpaid_10 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_10 = final_df_underpaid_10.head(10)
top_10_underpaid_10
```

Out[36]:

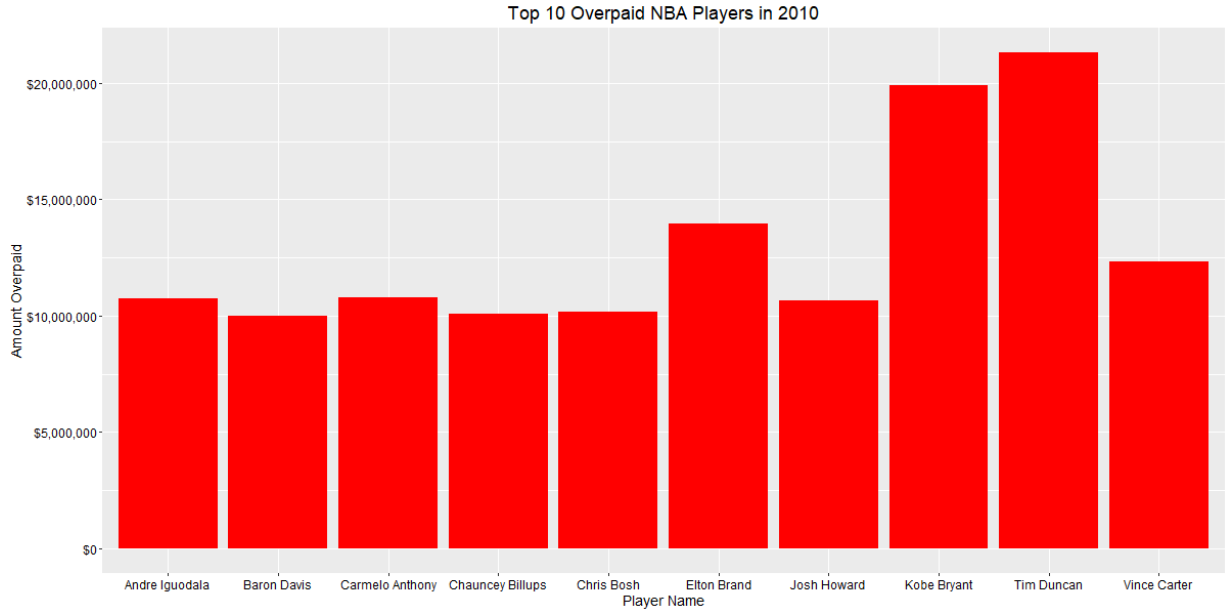
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
97	Aaron Brooks	1118520.0	21628846	-20510326.0	25.0	82.0	82.0	
239	Rudy Gay	3280997.0	19312470	-16031473.0	23.0	80.0	80.0	
11	Rafer Alston	475875.0	15781898	-15306023.0	33.0	52.0	38.0	
261	Stephen Graham	825497.0	15574602	-14749105.0	27.0	70.0	8.0	
307	Jordan Hill	2483280.0	16427102	-13943822.0	22.0	47.0	0.0	
159	Dante Cunningham	457588.0	14191066	-13733478.0	22.0	63.0	2.0	
175	Carlos Delfino	3500000.0	15601446	-12101446.0	27.0	75.0	66.0	
325	Lester Hudson	457588.0	12029192	-11571604.0	25.0	25.0	0.0	
45	J.J. Barea	1657500.0	12947800	-11290300.0	25.0	78.0	18.0	
139	Wilson Chandler	1255440.0	11741122	-10485682.0	22.0	65.0	64.0	

Making the Plots

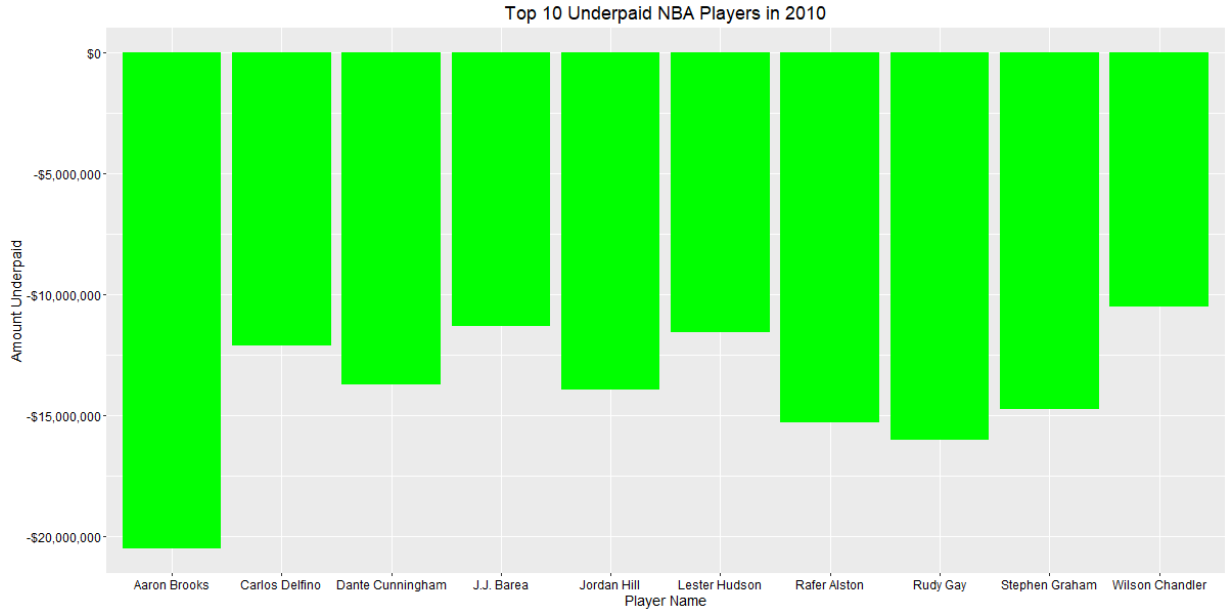
In [37]:
```
from rpy2.robjects.packages import importr
import rpy2.robjects as robjects
import rpy2.robjects.packages as rpackages
from rpy2.robjects.vectors import StrVector
utils = importr('utils')
utils.install_packages('ggplot2')
utils.install_packages('ggthemes')
utils.install_packages('gridExtra')
utils.install_packages('grid')
utils.install_packages('forcats')
utils.install_packages('magrittr')
utils.install_packages('dplyr')
%reload_ext rpy2.ipython
%R library(ggplot2)
%R library(gridExtra)
%R library(grid)
%R library(forcats)
%R library(magrittr)
%R library(dplyr)
```

Out[37]:
```
array(['dplyr', 'magrittr', 'forcats', 'grid', 'gridExtra', 'ggplot2',
       'tools', 'stats', 'graphics', 'grDevices', 'utils', 'datasets',
       'methods', 'base'], dtype='<U9')
```

In [38]:
```
%%R -i top_10_overpaid_10 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_10 <- ggplot(top_10_overpaid_10, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_10.png")
player_overpaid_bar_10
```



In [39]:
```
%%R -i top_10_underpaid_10 -w 1200 -h 600 -u px

player_underpaid_bar_10 <- ggplot(top_10_underpaid_10, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_10.png")
player_underpaid_bar_10
```



Running 2011 Data in the Model

In [40]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_11, labels_11, verbose=False)
y_train_pred1 = xgb_model1.predict(features_11)
y_pred1 = xgb_model1.predict(features_11)

print('Train r2 score: ', r2_score(y_train_pred1, labels_11))
print('Test r2 score: ', r2_score(labels_11, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_11)
test_mse1 = mean_squared_error(y_pred1, labels_11)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
[13:23:30] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9375518247968413
Test r2 score:  0.9490464658809084
Train RMSE: 1070874.0781
Test RMSE: 1070874.0781
```

In [41]:
```
y_train_xgb = pd.DataFrame(labels_11, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_11, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[41]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct
0	1959577.0	0.657895	0.498168	0.156566	0.195745	0.180851	0.231423
1	11244000.0	0.657895	0.787546	0.474747	0.185106	0.436170	0.214437
2	3000000.0	0.763158	0.670330	0.222222	0.278723	0.686170	0.231423
3	2563320.0	0.526316	0.351648	0.126263	0.389362	0.313830	0.135881
4	4533300.0	0.842105	0.695971	0.292929	0.240426	0.122340	0.084926

In [42]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_11.reset_index(drop=True, inplace=True)
names_11_actual.reset_index(drop=True, inplace=True)
```

In [43]:
```
final_df = pd.concat([y_predict_xgb, final_11], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_11_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[43]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field_
0	2785246.50	10	25.0	69.0	69.0	2324.0	312.0	
1	11683302.00	2	25.0	81.0	81.0	3211.0	707.0	
2	4954868.50	10	29.0	72.0	31.0	1494.0	251.0	
3	2312129.25	7	20.0	81.0	14.0	1452.0	160.0	
4	4380841.50	0	32.0	45.0	0.0	732.0	82.0	

In [44]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [45]:
```
cols = list(final_df.columns.values)
```

In [46]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[46]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
0	Arron Afflalo	1959577.0	2785246	-825669.50	25.0	69.0	69.0	
1	LaMarcus Aldridge	11244000.0	11683302	-439302.00	25.0	81.0	81.0	
2	Tony Allen	3000000.0	4954868	-1954868.50	29.0	72.0	31.0	
3	Al-Farouq Aminu	2563320.0	2312129	251190.75	20.0	81.0	14.0	
4	Chris Andersen	4533300.0	4380841	152458.50	32.0	45.0	0.0	

In [47]:
```
final_df_overpaid_11 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_11 = final_df_overpaid_11.head(10)
top_10_overpaid_11
```

Out[47]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
199	Rashard Lewis	19573711.0	15005214	4568497.0	31.0	57.0	52.0	
190	Andrei Kirilenko	17823000.0	13955712	3867288.0	29.0	64.0	62.0	
269	Brandon Roy	13603750.0	9749412	3854338.0	26.0	47.0	23.0	
211	Kenyon Martin	16545454.0	12784600	3760854.0	33.0	48.0	48.0	
261	Michael Redd	18300000.0	14722574	3577426.0	31.0	10.0	0.0	
52	Caron Butler	10561960.0	7054976	3506984.0	30.0	29.0	29.0	
59	Vince Carter	17522375.0	14455542	3066833.0	34.0	73.0	63.0	
241	Mehmet Okur	9945000.0	6894919	3050080.5	31.0	13.0	0.0	
110	T.J. Ford	8500000.0	5680021	2819979.0	27.0	41.0	3.0	
63	Josh Childress	6500000.0	4018967	2481032.5	27.0	54.0	3.0	

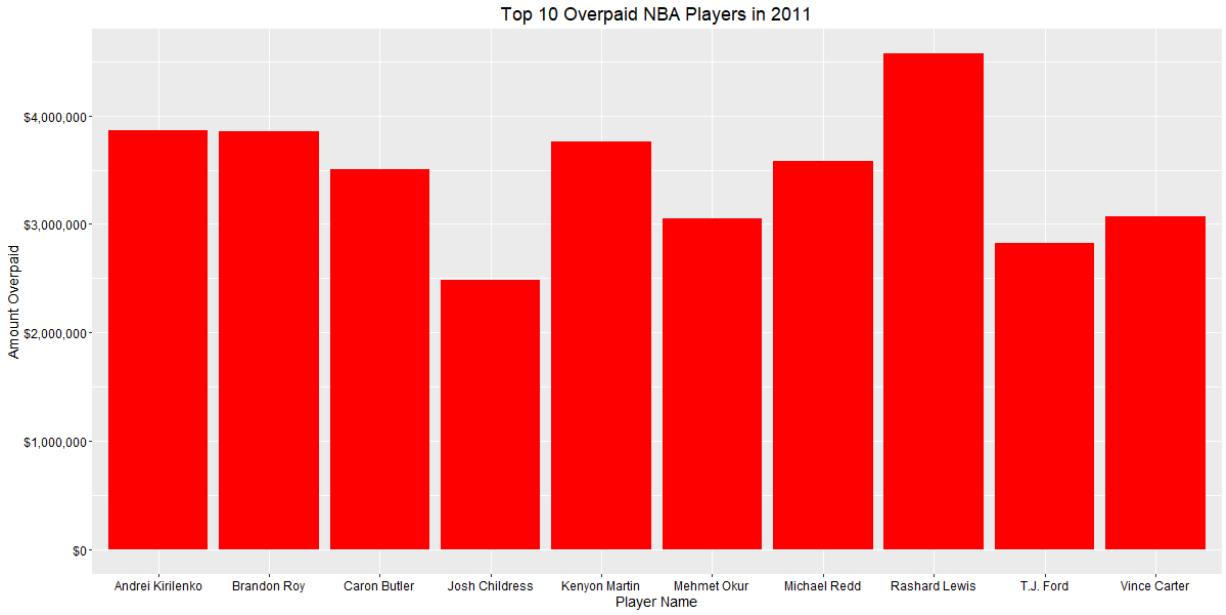
In [48]:
```
final_df_underpaid_11 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_11 = final_df_underpaid_11.head(10)
top_10_underpaid_11
```

Out[48]:

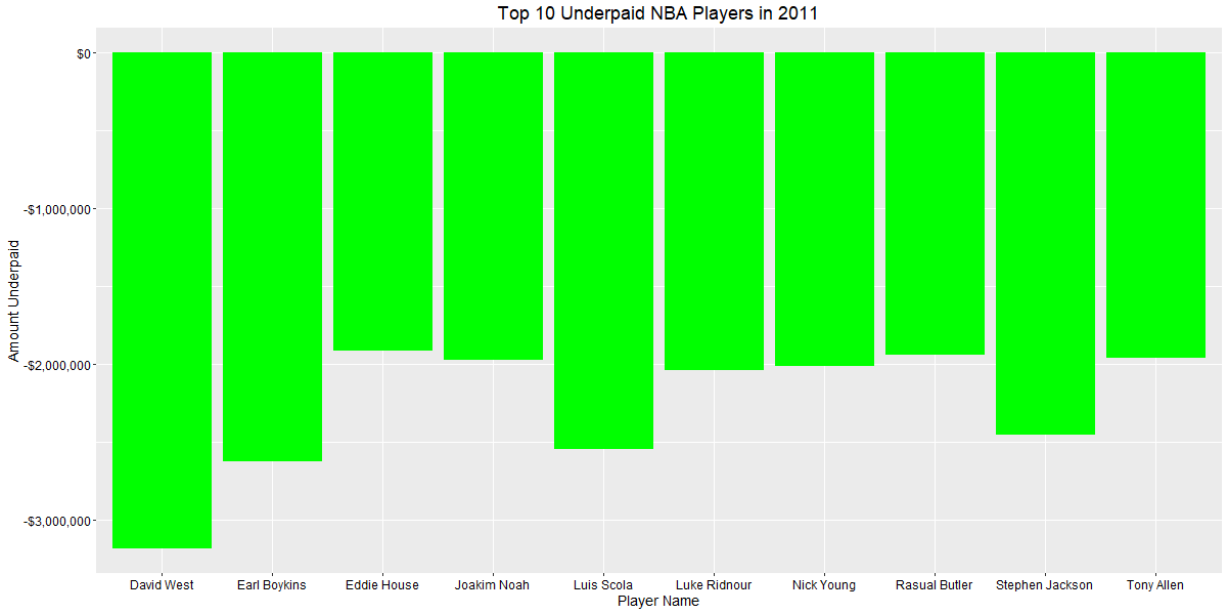
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
320	David West	8287500.0	11467776	-3180276.00	30.0	70.0	70.0	
42	Earl Boykins	854389.0	3472965	-2618576.75	34.0	57.0	0.0	
273	Luis Scola	7775378.0	10318822	-2543444.00	30.0	74.0	74.0	
169	Stephen Jackson	8453250.0	10901166	-2447916.00	32.0	67.0	67.0	
264	Luke Ridnour	4000000.0	6038604	-2038604.50	29.0	71.0	66.0	
338	Nick Young	2630503.0	4642066	-2011563.50	25.0	64.0	40.0	
235	Joakim Noah	3128536.0	5101314	-1972778.50	25.0	48.0	48.0	
2	Tony Allen	3000000.0	4954868	-1954868.50	29.0	72.0	31.0	
53	Rasual Butler	211084.0	2146462	-1935378.00	31.0	47.0	2.0	
158	Eddie House	1352181.0	3261679	-1909498.00	32.0	56.0	1.0	

Making the Plots

In [49]:
```
%%R -i top_10_overpaid_11 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_11 <- ggplot(top_10_overpaid_11, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_11.png")
player_overpaid_bar_11
```



In [50]:
```
%%R -i top_10_underpaid_11 -w 1200 -h 600 -u px

player_underpaid_bar_11 <- ggplot(top_10_underpaid_11, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_11.png")
player_underpaid_bar_11
```



Running 2012 Data in the Model

In [51]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_12, labels_12, verbose=False)
y_train_pred1 = xgb_model1.predict(features_12)
y_pred1 = xgb_model1.predict(features_12)

print('Train r2 score: ', r2_score(y_train_pred1, labels_12))
print('Test r2 score: ', r2_score(labels_12, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_12)
test_mse1 = mean_squared_error(y_pred1, labels_12)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
[13:24:51] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9489241895501423
Test r2 score:  0.9567032901339774
Train RMSE: 989150.0014
Test RMSE: 989150.0014
```

In [52]:
```
y_train_xgb = pd.DataFrame(labels_12, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_12, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[52]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct
0	7750000.0	0.702703	0.478827	0.053942	0.276163	0.236842	0.211429
1	14000000.0	0.702703	0.739414	0.186722	0.276163	0.335526	0.251429
2	3150000.0	0.810811	0.511401	0.136929	0.441860	0.684211	0.158095
3	75793.0	0.702703	0.283388	0.000000	0.151163	0.046053	0.083810
4	2755560.0	0.567568	0.345277	0.141079	0.523256	0.388158	0.137143

In [53]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_12.reset_index(drop=True, inplace=True)
names_12_actual.reset_index(drop=True, inplace=True)
```

In [54]:
```
final_df = pd.concat([y_predict_xgb, final_12], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_12_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[54]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field
0	6.831878e+06	10	26.0	62.0	62.0	2086.0	329.0	
1	1.363243e+07	2	26.0	55.0	55.0	1994.0	483.0	
2	4.919110e+06	10	30.0	58.0	57.0	1525.0	210.0	
3	5.423358e+05	10	26.0	4.0	0.0	67.0	6.0	
4	2.444576e+06	7	21.0	66.0	21.0	1477.0	150.0	

In [55]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [56]:
```
cols = list(final_df.columns.values)
```

In [57]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[57]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minut
0	Arron Afflalo	7750000.0	6831878	9.181220e+05	26.0	62.0	62.0	
1	LaMarcus Aldridge	14000000.0	13632426	3.675740e+05	26.0	55.0	55.0	
2	Tony Allen	3150000.0	4919110	-1.769110e+06	30.0	58.0	57.0	
3	Morris Almond	75793.0	542335	-4.665428e+05	26.0	4.0	0.0	
4	Al-Farouq Aminu	2755560.0	2444576	3.109835e+05	21.0	66.0	21.0	

In [58]:
```
final_df_overpaid_12 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_12 = final_df_overpaid_12.head(10)
top_10_overpaid_12
```

Out[58]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
212	Rashard Lewis	21136631.0	15162632	5973999.00	32.0	28.0	15.0	
181	Richard Jefferson	9282000.0	5920181	3361818.50	31.0	63.0	44.0	
128	Ben Gordon	11600000.0	8833972	2766028.00	28.0	52.0	21.0	
22	Michael Beasley	6262347.0	3846490	2415856.75	23.0	47.0	7.0	
51	Caron Butler	8000000.0	5736456	2263544.00	31.0	63.0	63.0	
160	Kirk Hinrich	8000000.0	5780448	2219552.00	31.0	48.0	31.0	
66	Wilson Chandler	5516664.0	3334044	2182619.75	24.0	8.0	6.0	
359	Thaddeus Young	8039130.0	5979937	2059193.00	23.0	63.0	1.0	
225	Kevin Martin	11519840.0	9481712	2038128.00	28.0	40.0	40.0	
254	Lamar Odom	8900000.0	6942651	1957349.00	32.0	50.0	4.0	

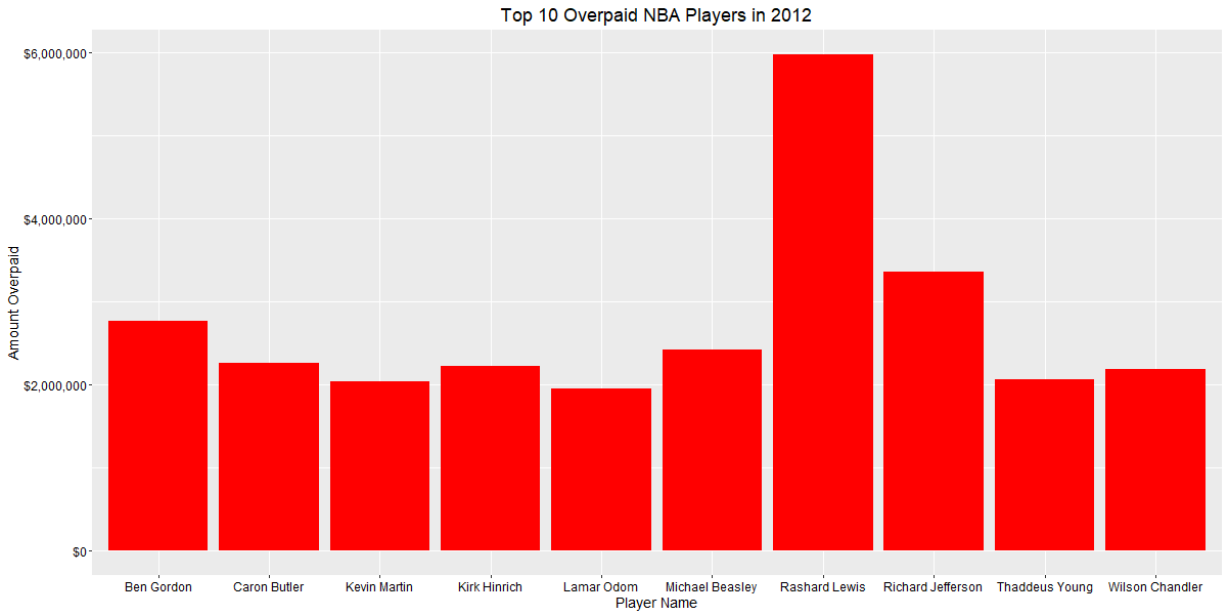
In [59]:
```
final_df_underpaid_12 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_12 = final_df_underpaid_12.head(10)
top_10_underpaid_12
```

Out[59]:

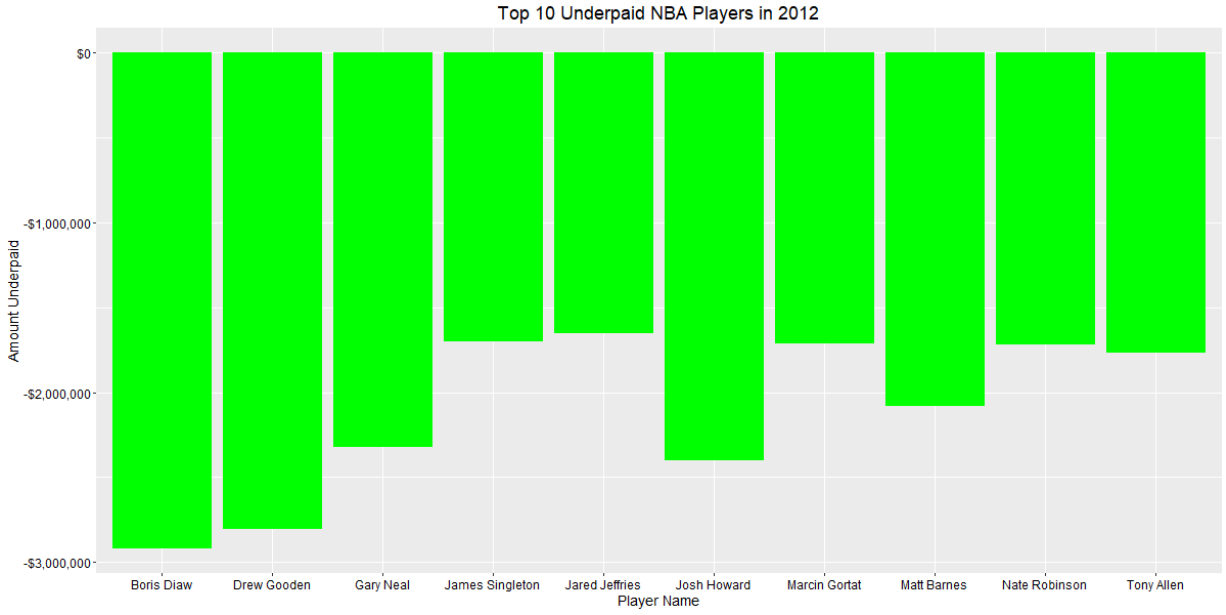
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
89	Boris Diaw	241158.0	3161587	-2920429.25	29.0	57.0	35.0	
127	Drew Gooden	6226200.0	9032490	-2806290.00	30.0	56.0	46.0	
166	Josh Howard	2150000.0	4553835	-2403835.00	31.0	43.0	18.0	
249	Gary Neal	854389.0	3176236	-2321847.50	27.0	56.0	7.0	
17	Matt Barnes	1906200.0	3990187	-2083987.00	31.0	63.0	16.0	
2	Tony Allen	3150000.0	4919110	-1769110.00	30.0	58.0	57.0	
278	Nate Robinson	785487.0	2503038	-1717551.25	27.0	51.0	9.0	
130	Marcin Gortat	7258960.0	8972970	-1714010.00	27.0	66.0	66.0	
294	James Singleton	151585.0	1853364	-1701779.50	30.0	12.0	0.0	
182	Jared Jeffries	854389.0	2504021	-1649632.00	30.0	39.0	4.0	

Making the Plots

In [60]:
```
%%R -i top_10_overpaid_12 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_12 <- ggplot(top_10_overpaid_12, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_12.png")
player_overpaid_bar_12
```



In [61]:
```
%%R -i top_10_underpaid_12 -w 1200 -h 600 -u px

player_underpaid_bar_12 <- ggplot(top_10_underpaid_12, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_12.png")
player_underpaid_bar_12
```



Running 2013 Data in the Model

In [62]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_13, labels_13, verbose=False)
y_train_pred1 = xgb_model1.predict(features_13)
y_pred1 = xgb_model1.predict(features_13)

print('Train r2 score: ', r2_score(y_train_pred1, labels_13))
print('Test r2 score: ', r2_score(labels_13, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_13)
test_mse1 = mean_squared_error(y_pred1, labels_13)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
[13:26:16] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9274330993549923
Test r2 score:  0.9419926164016555
Train RMSE: 1132491.3098
Test RMSE: 1132491.3098
```

In [63]:
```
y_train_xgb = pd.DataFrame(labels_13, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_13, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[63]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct	Ga
0	665000.0	0.550	0.503165	0.061983	0.289963	0.074713	0.105477	
1	916099.0	0.650	0.424051	0.111570	0.243494	0.103448	0.168357	
2	7750000.0	0.675	0.411392	0.045455	0.224907	0.229885	0.296146	
3	13500000.0	0.675	0.645570	0.376033	0.165428	0.356322	0.263692	
4	3000000.0	0.575	0.363924	0.227273	0.191450	0.137931	0.135903	

In [64]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_13.reset_index(drop=True, inplace=True)
names_13_actual.reset_index(drop=True, inplace=True)
```

In [65]:
```
final_df = pd.concat([y_predict_xgb, final_13], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_13_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[65]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field
0	7.410186e+05	2	22.0	29.0	0.0	342.0	42.0	
1	2.792544e+06	2	26.0	52.0	5.0	713.0	72.0	
2	7.626414e+06	7	27.0	64.0	64.0	2307.0	397.0	
3	1.325341e+07	2	27.0	74.0	74.0	2790.0	638.0	
4	2.217858e+06	0	23.0	79.0	37.0	1669.0	206.0	

In [66]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [67]:
```
cols = list(final_df.columns.values)
```

In [68]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[68]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minut
0	Quincy Acy	665000.0	741018	-7.601856e+04	22.0	29.0	0.0	
1	Jeff Adrien	916099.0	2792544	-1.876445e+06	26.0	52.0	5.0	
2	Arron Afflalo	7750000.0	7626414	1.235860e+05	27.0	64.0	64.0	
3	LaMarcus Aldridge	13500000.0	13253410	2.465900e+05	27.0	74.0	74.0	
4	Lavoy Allen	3000000.0	2217858	7.821420e+05	23.0	79.0	37.0	

In [69]:
```
final_df_overpaid_13 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_13 = final_df_overpaid_13.head(10)
top_10_overpaid_13
```

Out[69]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
169	Richard Jefferson	10164000.0	4580900	5583100.0	32.0	56.0	1.0	
178	Joe Johnson	19752645.0	15045588	4707057.0	31.0	72.0	72.0	
157	Kris Humphries	12000000.0	8052070	3947929.5	27.0	65.0	21.0	
316	Tyrus Thomas	8000000.0	4531414	3468586.0	26.0	26.0	2.0	
218	Corey Maggette	10924138.0	7477913	3446225.0	33.0	18.0	0.0	
164	Stephen Jackson	10059750.0	6902868	3156882.0	34.0	55.0	6.0	
122	Danny Granger	13058606.0	9941856	3116750.0	29.0	5.0	0.0	
223	Kevin Martin	12439675.0	9529776	2909899.0	29.0	77.0	0.0	
334	Luke Walton	6091363.0	3288226	2803136.5	32.0	50.0	0.0	
326	Beno Udrih	7372000.0	4943816	2428184.0	30.0	66.0	9.0	

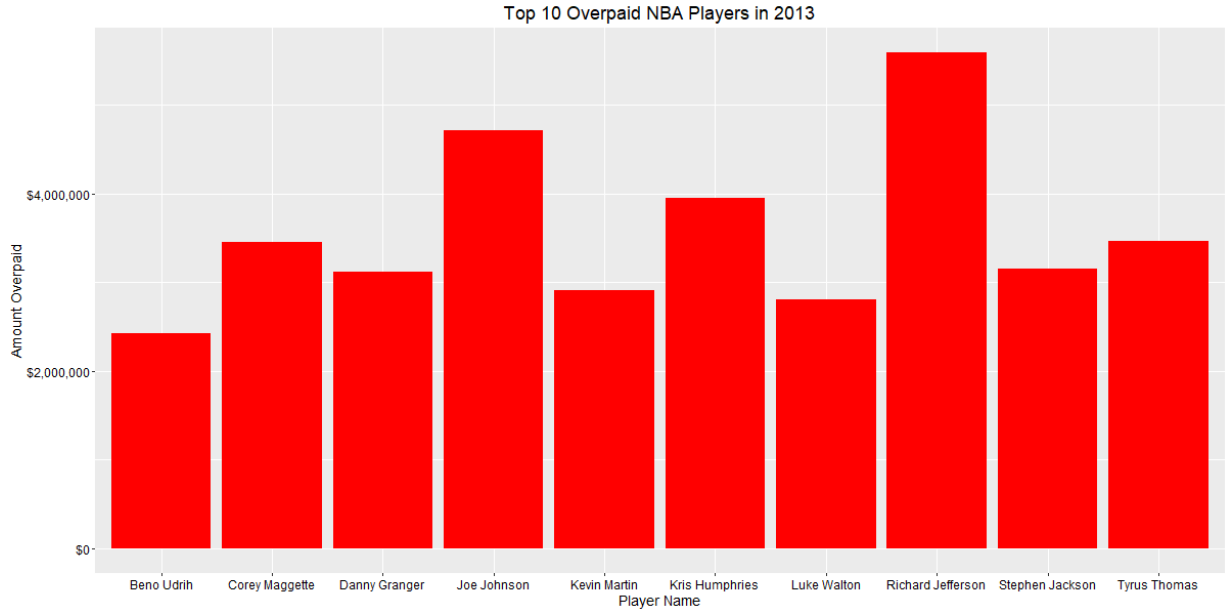
In [70]:
```
final_df_underpaid_13 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_13 = final_df_underpaid_13.head(10)
top_10_underpaid_13
```

Out[70]:

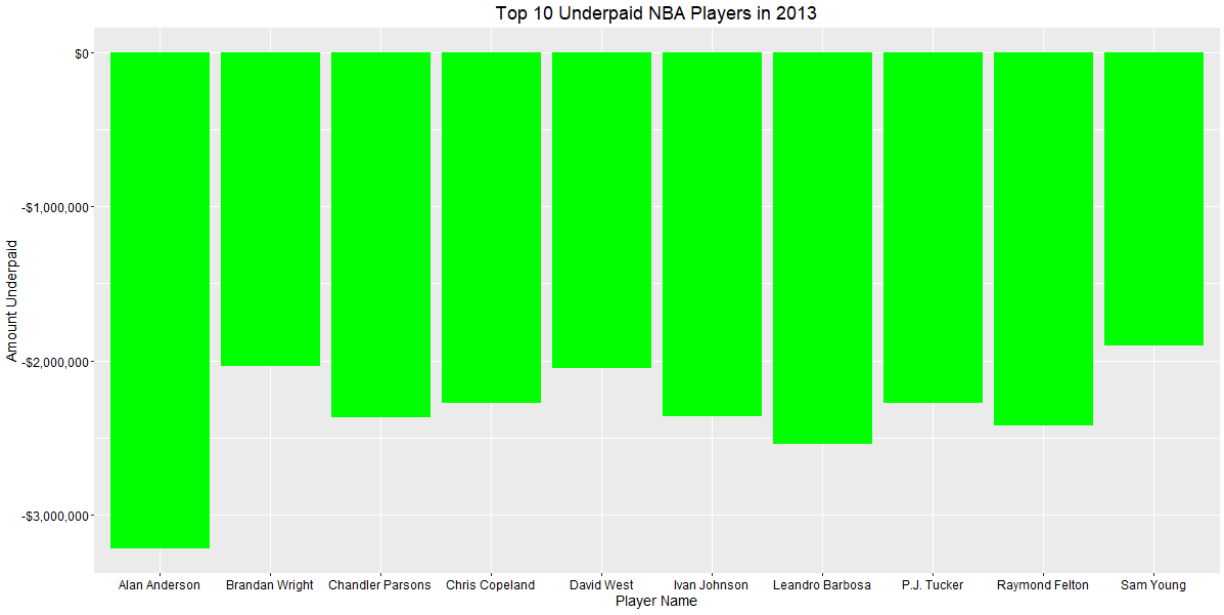
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
8	Alan Anderson	854389.0	4076822	-3222433.0	30.0	65.0	2.0	
17	Leandro Barbosa	854389.0	3397026	-2542637.5	30.0	41.0	2.0	
102	Raymond Felton	3480453.0	5900611	-2420158.0	28.0	68.0	68.0	
256	Chandler Parsons	888250.0	3254866	-2366616.0	24.0	76.0	76.0	
176	Ivan Johnson	962195.0	3327435	-2365240.5	28.0	69.0	5.0	
69	Chris Copeland	473604.0	2749921	-2276317.0	28.0	56.0	13.0	
323	P.J. Tucker	884293.0	3158692	-2274399.5	27.0	79.0	45.0	
340	David West	10000000.0	12048276	-2048276.0	32.0	73.0	73.0	
352	Brandan Wright	992680.0	3028285	-2035605.0	25.0	64.0	16.0	
356	Sam Young	402065.0	2308570	-1906505.5	27.0	56.0	3.0	

Making the Plots

In [71]:
```
%%R -i top_10_overpaid_13 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_13 <- ggplot(top_10_overpaid_13, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_13.png")
player_overpaid_bar_13
```



In [72]:
```
%%R -i top_10_underpaid_13 -w 1200 -h 600 -u px

player_underpaid_bar_13 <- ggplot(top_10_underpaid_13, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_13.png")
player_underpaid_bar_13
```



Running 2014 Data in the Model

In [73]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_14, labels_14, verbose=False)
y_train_pred1 = xgb_model1.predict(features_14)
y_pred1 = xgb_model1.predict(features_14)

print('Train r2 score: ', r2_score(y_train_pred1, labels_14))
print('Test r2 score: ', r2_score(labels_14, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_14)
test_mse1 = mean_squared_error(y_pred1, labels_14)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
[13:27:46] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9333043038662966
Test r2 score:  0.9463851960661243
Train RMSE: 1174768.8833
Test RMSE: 1174768.8833
```

In [74]:
```
y_train_xgb = pd.DataFrame(labels_14, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_14, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[74]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct
0	788872.0	0.605263	0.338926	0.118721	0.269702	0.120419	0.100204
1	7500000.0	0.736842	0.536913	0.013699	0.196147	0.183246	0.345603
2	14878000.0	0.736842	0.731544	0.310502	0.126095	0.329843	0.265849
3	3060000.0	0.631579	0.419463	0.150685	0.210158	0.125654	0.206544
4	4494383.0	0.842105	0.523490	0.086758	0.283713	0.471204	0.239264

In [75]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_14.reset_index(drop=True, inplace=True)
names_14_actual.reset_index(drop=True, inplace=True)
```

In [76]:
```
final_df = pd.concat([y_predict_xgb, final_14], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_14_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[76]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field_
0	1860835.00	7	23.0	63.0	0.0	847.0	66.0	
1	8343345.00	10	28.0	73.0	73.0	2552.0	464.0	
2	15580174.00	2	28.0	69.0	69.0	2498.0	652.0	
3	2559069.75	2	24.0	65.0	2.0	1072.0	134.0	
4	4226438.50	10	32.0	55.0	28.0	1278.0	204.0	

In [77]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [78]:
```
cols = list(final_df.columns.values)
```

In [79]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[79]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
0	Quincy Acy	788872.0	1860835	-1071963.00	23.0	63.0	0.0	
1	Arron Afflalo	7500000.0	8343345	-843345.00	28.0	73.0	73.0	
2	LaMarcus Aldridge	14878000.0	15580174	-702174.00	28.0	69.0	69.0	
3	Lavoy Allen	3060000.0	2559069	500930.25	24.0	65.0	2.0	
4	Tony Allen	4494383.0	4226438	267944.50	32.0	55.0	28.0	

In [80]:
```
final_df_overpaid_14 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_14 = final_df_overpaid_14.head(10)
top_10_overpaid_14
```

Out[80]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
161	Kris Humphries	12000000.0	7698955	4301045.00	28.0	69.0	30.0	
180	Joe Johnson	21466718.0	17681146	3785572.00	32.0	79.0	79.0	
126	Eric Gordon	14283844.0	10636072	3647772.00	25.0	64.0	64.0	
109	Landry Fields	6250000.0	2717337	3532662.25	25.0	30.0	2.0	
313	Marcus Thornton	8050000.0	4878960	3171039.50	26.0	72.0	27.0	
329	Deron Williams	18466130.0	15361248	3104882.00	29.0	64.0	58.0	
205	Jeremy Lin	8374646.0	5385457	2989189.00	25.0	71.0	33.0	
47	Chase Budinger	5000000.0	2087954	2912045.75	25.0	41.0	8.0	
163	Andre Iguodala	12868632.0	9986612	2882020.00	30.0	63.0	63.0	
156	Jrue Holiday	9713484.0	6856769	2856715.00	23.0	34.0	34.0	

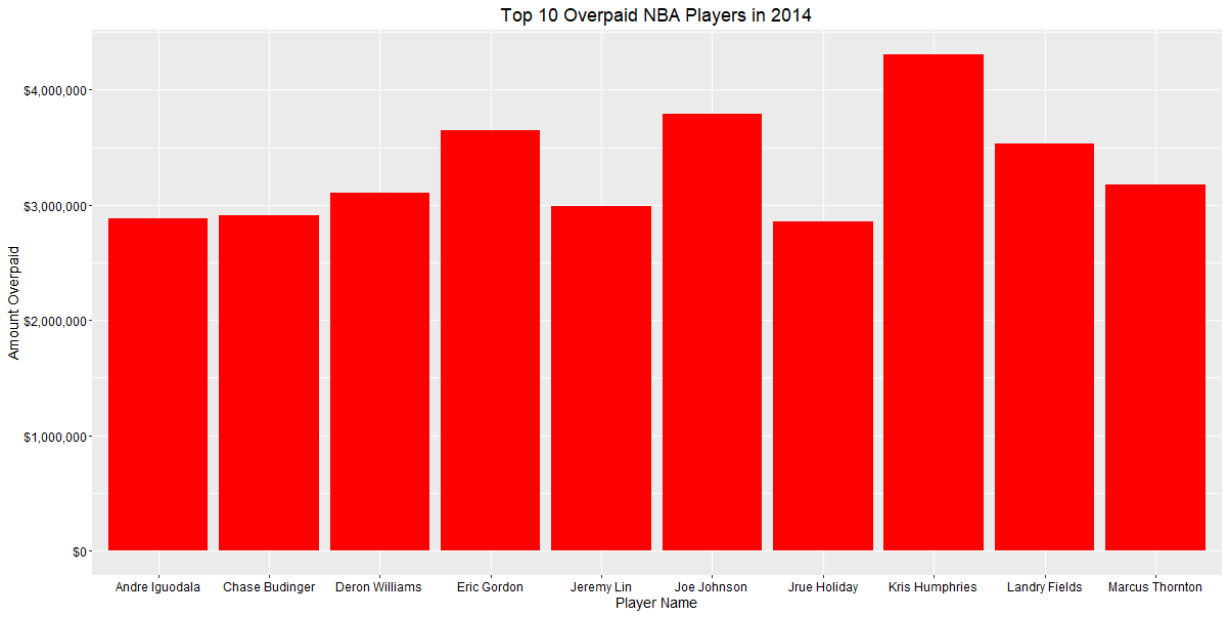
In [81]:
```
final_df_underpaid_14 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_14 = final_df_underpaid_14.head(10)
top_10_underpaid_14
```

Out[81]:

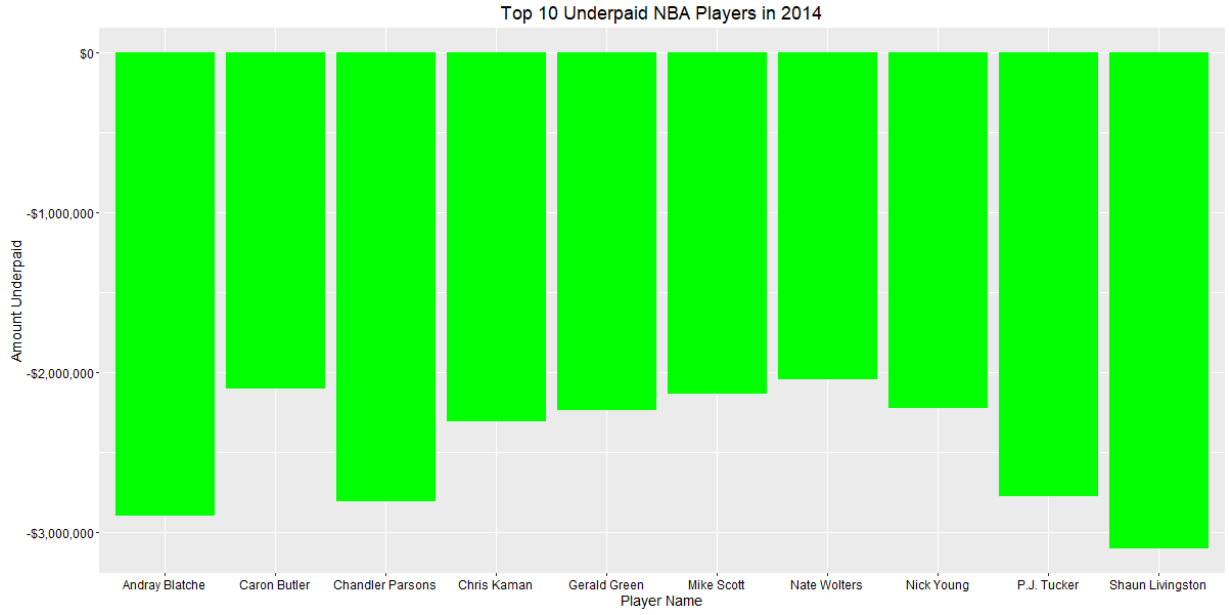
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
206	Shaun Livingston	884293.0	3988528	-3104235.50	28.0	76.0	54.0	
33	Andray Blatche	1375604.0	4272563	-2896959.00	27.0	73.0	7.0	
257	Chandler Parsons	926500.0	3738410	-2811910.00	25.0	74.0	74.0	
315	P.J. Tucker	884293.0	3663806	-2779513.00	28.0	81.0	81.0	
186	Chris Kaman	3183000.0	5491764	-2308764.50	31.0	39.0	13.0	
132	Gerald Green	3500000.0	5735202	-2235202.00	28.0	82.0	48.0	
340	Nick Young	1106942.0	3333801	-2226859.50	28.0	64.0	9.0	
284	Mike Scott	788872.0	2926426	-2137554.00	25.0	80.0	6.0	
51	Caron Butler	244489.0	2346278	-2101789.25	33.0	56.0	13.0	
336	Nate Wolters	500000.0	2548361	-2048361.00	22.0	58.0	31.0	

Making the Plots

In [82]:
```
%%R -i top_10_overpaid_14 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_14 <- ggplot(top_10_overpaid_14, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_14.png")
player_overpaid_bar_14
```



In [83]:
```
%%R -i top_10_underpaid_14 -w 1200 -h 600 -u px

player_underpaid_bar_14 <- ggplot(top_10_underpaid_14, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_14.png")
player_underpaid_bar_14
```



Running 2015 Data in the Model

In [84]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_15, labels_15, verbose=False)
y_train_pred1 = xgb_model1.predict(features_15)
y_pred1 = xgb_model1.predict(features_15)

print('Train r2 score: ', r2_score(y_train_pred1, labels_15))
print('Test r2 score: ', r2_score(labels_15, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_15)
test_mse1 = mean_squared_error(y_pred1, labels_15)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
[13:29:25] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.923539160055924
Test r2 score:  0.9395819510050532
Train RMSE: 1204873.8339
Test RMSE: 1204873.8339
```

In [85]:
```
y_train_xgb = pd.DataFrame(labels_15, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_15, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[85]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct	Ga
0	915243.0	0.631579	0.386364	0.110	0.436709	0.165644	0.183544	
1	1344120.0	0.526316	0.415584	0.035	0.401899	0.098160	0.213080	
2	2184960.0	0.552632	0.457792	0.430	0.531646	0.233129	0.116034	
3	7500000.0	0.763158	0.347403	0.035	0.338608	0.251534	0.172996	
4	2963232.0	0.605263	0.389610	0.080	0.503165	0.104294	0.175105	

In [86]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_15.reset_index(drop=True, inplace=True)
names_15_actual.reset_index(drop=True, inplace=True)
```

In [87]:
```
final_df = pd.concat([y_predict_xgb, final_15], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_15_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[87]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field_
0	1610782.875	2	24.0	68.0	22.0	1287.0	152.0	
1	1145078.000	10	20.0	30.0	0.0	248.0	35.0	
2	2481302.500	0	21.0	70.0	67.0	1771.0	217.0	
3	6948061.500	10	29.0	78.0	72.0	2502.0	375.0	
4	2143111.000	2	23.0	41.0	9.0	540.0	40.0	

In [88]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [89]:
```
cols = list(final_df.columns.values)
```

In [90]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[90]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes_
0	Quincy Acy	915243.0	1610782	-695539.875	24.0	68.0	22.0	
1	Jordan Adams	1344120.0	1145078	199042.000	20.0	30.0	0.0	
2	Steven Adams	2184960.0	2481302	-296342.500	21.0	70.0	67.0	
3	Arron Afflalo	7500000.0	6948061	551938.500	29.0	78.0	72.0	
4	Furkan Aldemir	2963232.0	2143111	820121.000	23.0	41.0	9.0	

In [91]:
```
final_df_overpaid_15 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_15 = final_df_overpaid_15.head(10)
top_10_overpaid_15
```

Out[91]:

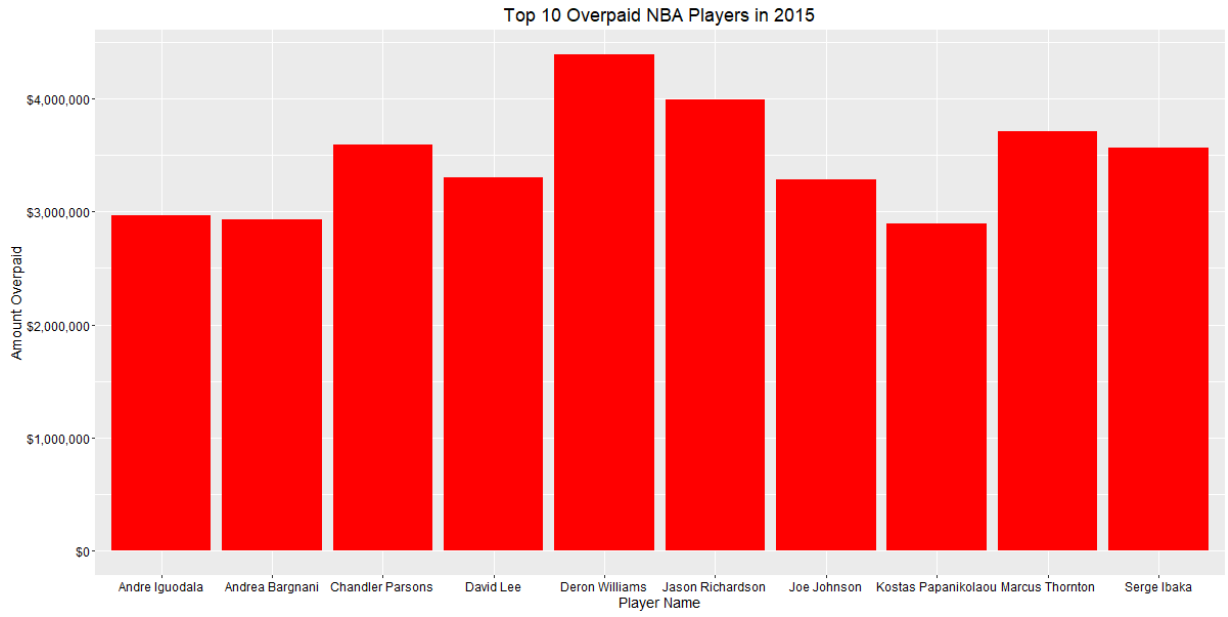
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
360	Deron Williams	19754465.0	15366410	4388055.00	30.0	68.0	55.0	
291	Jason Richardson	6601125.0	2612296	3988828.50	34.0	19.0	15.0	
339	Marcus Thornton	8575000.0	4865416	3709584.00	27.0	48.0	0.0	
273	Chandler Parsons	14700000.0	11105402	3594598.00	26.0	66.0	66.0	
176	Serge Ibaka	12350000.0	8782102	3567898.00	25.0	64.0	64.0	
215	David Lee	15012000.0	11713204	3298796.00	31.0	49.0	4.0	
193	Joe Johnson	23180790.0	19895778	3285012.00	33.0	80.0	80.0	
177	Andre Iguodala	12289544.0	9320008	2969536.00	31.0	77.0	0.0	
21	Andrea Bargnani	11500000.0	8569986	2930014.00	29.0	29.0	22.0	
269	Kostas Papanikolaou	4797664.0	1899827	2897836.25	24.0	43.0	1.0	

In [92]:
```
final_df_underpaid_15 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_15 = final_df_underpaid_15.head(10)
top_10_underpaid_15
```

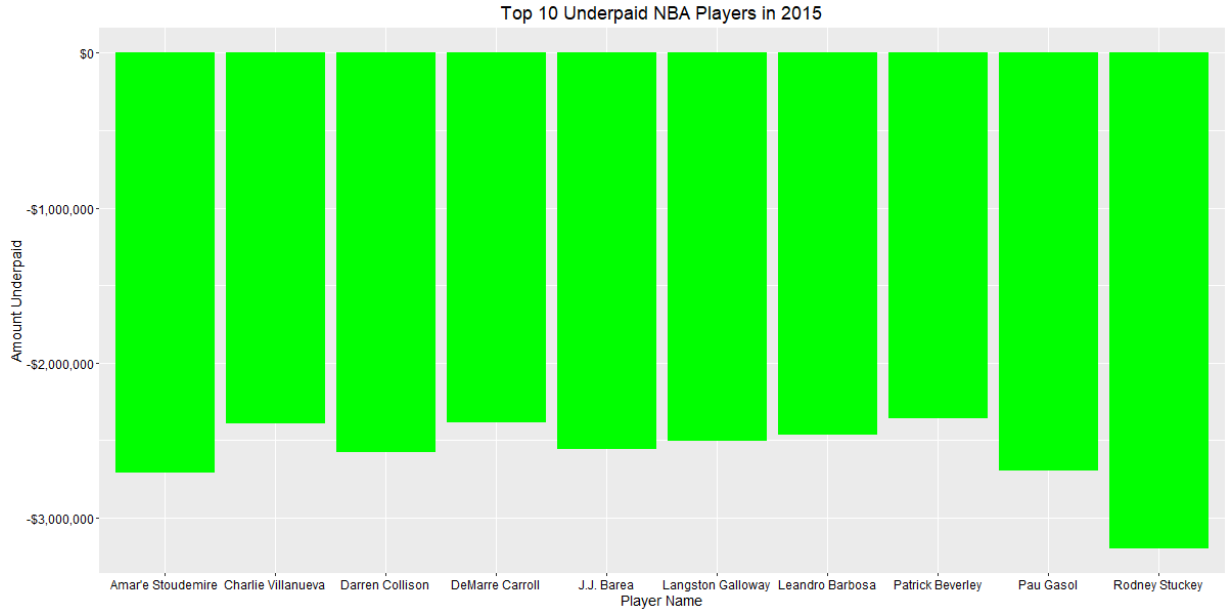
Out[92]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
327	Rodney Stuckey	915243.0	4114319	-3199076.25	28.0	71.0	36.0	
326	Amar'e Stoudemire	306876.0	3018201	-2711325.75	32.0	59.0	15.0	
128	Pau Gasol	7128000.0	9822810	-2694810.00	34.0	78.0	78.0	
73	Darren Collison	4797664.0	7372277	-2574613.50	27.0	45.0	45.0	
20	J.J. Barea	909859.0	3469649	-2559790.00	30.0	77.0	10.0	
125	Langston Galloway	235762.0	2737654	-2501892.50	23.0	45.0	41.0	
19	Leandro Barbosa	915243.0	3380046	-2464803.50	32.0	66.0	1.0	
344	Charlie Villanueva	915243.0	3308595	-2393352.25	30.0	64.0	1.0	
63	DeMarre Carroll	2442455.0	4830966	-2388511.50	28.0	70.0	69.0	
35	Patrick Beverley	915243.0	3275820	-2360577.00	26.0	56.0	55.0	

In [93]:
```
%%R -i top_10_overpaid_15 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_15 <- ggplot(top_10_overpaid_15, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_15.png")
player_overpaid_bar_15
```



In [94]:
```
%%R -i top_10_underpaid_15 -w 1200 -h 600 -u px

player_underpaid_bar_15 <- ggplot(top_10_underpaid_15, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_15.png")
player_underpaid_bar_15
```



Running 2016 Data in the Model

In [95]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_16, labels_16, verbose=False)
y_train_pred1 = xgb_model1.predict(features_16)
y_pred1 = xgb_model1.predict(features_16)

print('Train r2 score: ', r2_score(y_train_pred1, labels_16))
print('Test r2 score: ', r2_score(labels_16, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_16)
test_mse1 = mean_squared_error(y_pred1, labels_16)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
[13:31:02] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9457051573649677
Test r2 score:  0.9549776219832756
Train RMSE: 1147450.9442
Test RMSE: 1147450.9442
```

In [96]:
```
y_train_xgb = pd.DataFrame(labels_16, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_16, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[96]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct
0	981348.0	0.641026	0.466667	0.135593	0.337838	0.171598	0.083491
1	1404600.0	0.538462	0.549206	0.000000	0.662162	0.017751	0.605313
2	8000000.0	0.769231	0.346032	0.056497	0.293919	0.147929	0.187856
3	19689000.0	0.769231	0.711111	0.457627	0.263514	0.224852	0.153700
4	5158539.0	0.871795	0.409524	0.101695	0.432432	0.650888	0.130930

In [97]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_16.reset_index(drop=True, inplace=True)
names_16_actual.reset_index(drop=True, inplace=True)
```

In [98]:
```
final_df = pd.concat([y_predict_xgb, final_16], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_16_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[98]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field
0	1.673773e+06	2	25.0	59.0	29.0	876.0	119.0	
1	1.907881e+06	10	21.0	2.0	0.0	15.0	2.0	
2	8.419974e+06	10	30.0	71.0	57.0	2371.0	354.0	
3	1.902708e+07	2	30.0	74.0	74.0	2261.0	536.0	
4	5.582366e+06	10	34.0	64.0	57.0	1620.0	215.0	

In [99]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [100]:
```
cols = list(final_df.columns.values)
```

In [101]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[101]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
0	Quincy Acy	981348.0	1673773	-692425.125	25.0	59.0	29.0	
1	Jordan Adams	1404600.0	1907881	-503281.250	21.0	2.0	0.0	
2	Arron Afflalo	8000000.0	8419974	-419974.000	30.0	71.0	57.0	
3	LaMarcus Aldridge	19689000.0	19027076	661924.000	30.0	74.0	74.0	
4	Tony Allen	5158539.0	5582366	-423827.500	34.0	64.0	57.0	

In [102]:
```
final_df_overpaid_16 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_16 = final_df_overpaid_16.head(10)
top_10_overpaid_16
```

Out[102]:

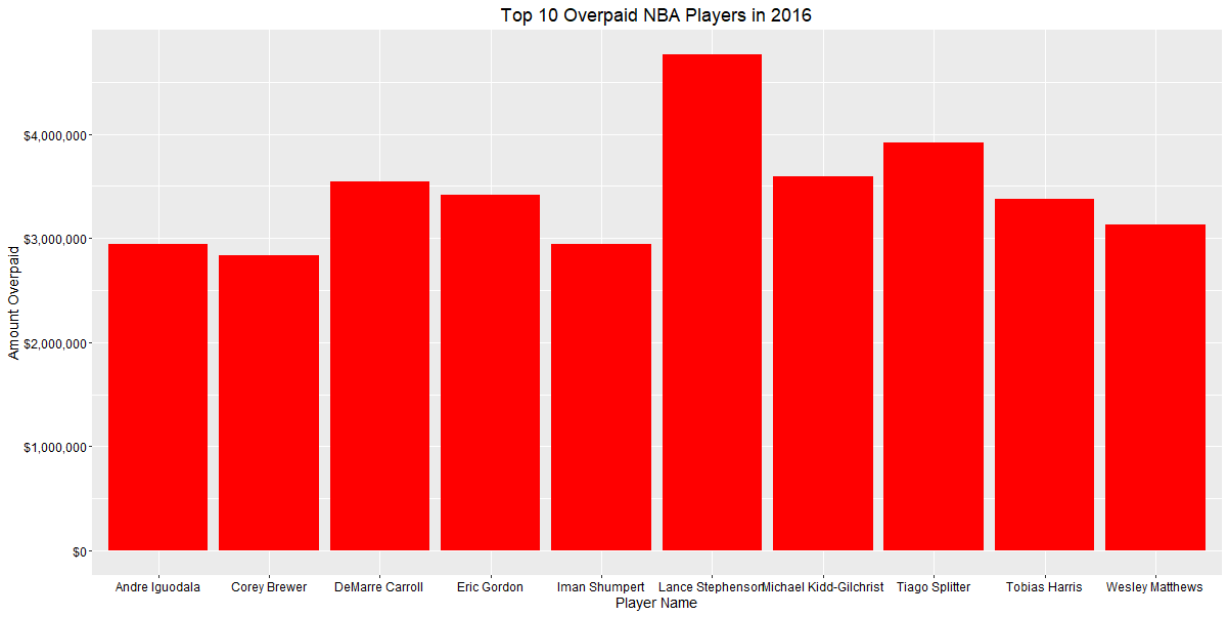
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
333	Lance Stephenson	9000000.0	4240104	4759895.5	25.0	69.0	13.0	
331	Tiago Splitter	9756250.0	5836750	3919499.5	31.0	36.0	2.0	
205	Michael Kidd-Gilchrist	6331404.0	2743396	3588007.5	22.0	7.0	7.0	
64	DeMarre Carroll	13600000.0	10054780	3545220.0	29.0	26.0	22.0	
130	Eric Gordon	15514031.0	12094824	3419207.0	27.0	45.0	44.0	
151	Tobias Harris	16000000.0	12624060	3375940.0	23.0	76.0	74.0	
234	Wesley Matthews	16407500.0	13280896	3126604.0	29.0	78.0	78.0	
175	Andre Iguodala	11710456.0	8767072	2943384.0	32.0	65.0	1.0	
320	Iman Shumpert	8988765.0	6045441	2943323.5	25.0	54.0	5.0	
48	Corey Brewer	8229375.0	5393711	2835663.5	29.0	82.0	12.0	

In [103]:
```
final_df_underpaid_16 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_16 = final_df_underpaid_16.head(10)
top_10_underpaid_16
```

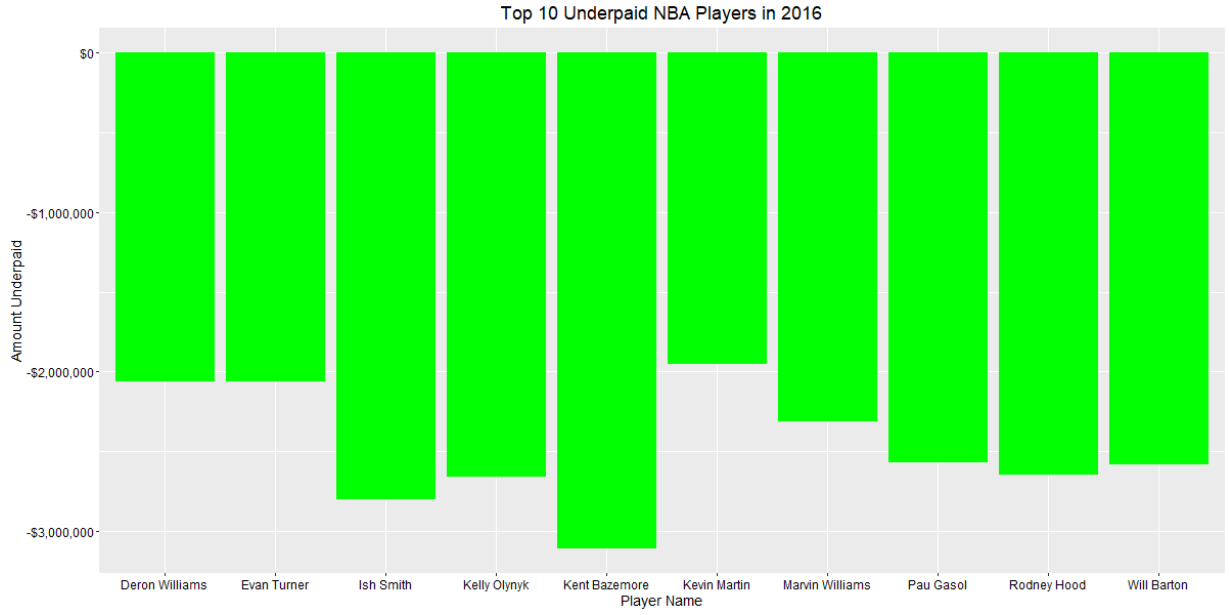
Out[103]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
31	Kent Bazemore	2000000.0	5111111	-3111111.50	26.0	75.0	68.0	
325	Ish Smith	947276.0	3748740	-2801464.00	27.0	77.0	53.0	
278	Kelly Olynyk	2165160.0	4826370	-2661210.00	24.0	69.0	8.0	
169	Rodney Hood	1348440.0	3997333	-2648893.00	23.0	79.0	79.0	
26	Will Barton	3533333.0	6115936	-2582603.50	25.0	82.0	1.0	
122	Pau Gasol	7448760.0	10021274	-2572514.00	35.0	72.0	72.0	
369	Marvin Williams	7000000.0	9312182	-2312182.00	29.0	81.0	81.0	
348	Evan Turner	3425510.0	5491414	-2065904.50	27.0	81.0	12.0	
365	Deron Williams	5378974.0	7443563	-2064589.50	31.0	65.0	63.0	
233	Kevin Martin	200600.0	2154428	-1953828.75	32.0	55.0	13.0	

In [104]:
```
%%R -i top_10_overpaid_16 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_16 <- ggplot(top_10_overpaid_16, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_16.png")
player_overpaid_bar_16
```



In [105]:
```
%%R -i top_10_underpaid_16 -w 1200 -h 600 -u px

player_underpaid_bar_16 <- ggplot(top_10_underpaid_16, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_16.png")
player_underpaid_bar_16
```



Running 2017 Data in the Model

In [106]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_17, labels_17, verbose=False)
y_train_pred1 = xgb_model1.predict(features_17)
y_pred1 = xgb_model1.predict(features_17)

print('Train r2 score: ', r2_score(y_train_pred1, labels_17))
print('Test r2 score: ', r2_score(labels_17, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_17)
test_mse1 = mean_squared_error(y_pred1, labels_17)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
[13:32:45] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9240670772224324
Test r2 score:  0.9398401824768344
Train RMSE: 1641826.5574
Test RMSE: 1641826.5574
```

In [107]:
```
y_train_xgb = pd.DataFrame(labels_17, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_17, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[107]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct	Ga
0	1790092.0	0.650	0.383117	0.070093	0.222477	0.089172	0.085515	
1	3140517.0	0.575	0.535714	0.364486	0.366972	0.566879	0.094241	
2	12500000.0	0.775	0.288961	0.028037	0.192661	0.133758	0.129145	
3	20575005.0	0.775	0.603896	0.411215	0.176606	0.292994	0.172775	
4	4000000.0	0.675	0.376623	0.112150	0.314220	0.114650	0.158813	

In [108]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_17.reset_index(drop=True, inplace=True)
names_17_actual.reset_index(drop=True, inplace=True)
```

In [109]:
```
final_df = pd.concat([y_predict_xgb, final_17], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_17_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[109]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field
0	2.033745e+06	2	26.0	38.0	1.0	558.0	70.0	
1	3.678990e+06	0	23.0	80.0	80.0	2389.0	374.0	
2	1.141594e+07	10	31.0	61.0	45.0	1580.0	185.0	
3	2.106197e+07	2	31.0	72.0	72.0	2335.0	500.0	
4	4.843621e+06	2	27.0	61.0	5.0	871.0	77.0	

In [110]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [111]:
```
cols = list(final_df.columns.values)
```

In [112]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[112]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
0	Quincy Acy	1790092.0	2033745	-243653.375	26.0	38.0	1.0	
1	Steven Adams	3140517.0	3678990	-538473.500	23.0	80.0	80.0	
2	Arron Afflalo	12500000.0	11415942	1084058.000	31.0	61.0	45.0	
3	LaMarcus Aldridge	20575005.0	21061966	-486961.000	31.0	72.0	72.0	
4	Lavoy Allen	4000000.0	4843621	-843621.000	27.0	61.0	5.0	

In [113]:
```
final_df_overpaid_17 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_17 = final_df_overpaid_17.head(10)
top_10_overpaid_17
```

Out[113]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
286	Chandler Parsons	22116750.0	14710280	7406470.0	28.0	34.0	34.0	
85	Luol Deng	18000000.0	11854422	6145578.0	31.0	56.0	49.0	
353	Evan Turner	16393443.0	11589294	4804149.0	28.0	65.0	12.0	
167	Al Horford	26540100.0	21786892	4753208.0	30.0	68.0	68.0	
92	Jared Dudley	10470000.0	5942127	4527873.0	31.0	64.0	7.0	
51	Alec Burks	10154495.0	5692048	4462446.5	25.0	42.0	0.0	
276	Dirk Nowitzki	25000000.0	20795476	4204524.0	38.0	54.0	54.0	
273	Joakim Noah	17000000.0	13034342	3965658.0	31.0	46.0	46.0	
204	Brandon Knight	12606250.0	8798514	3807736.0	25.0	54.0	5.0	
156	John Henson	12517606.0	8800004	3717602.0	26.0	58.0	39.0	

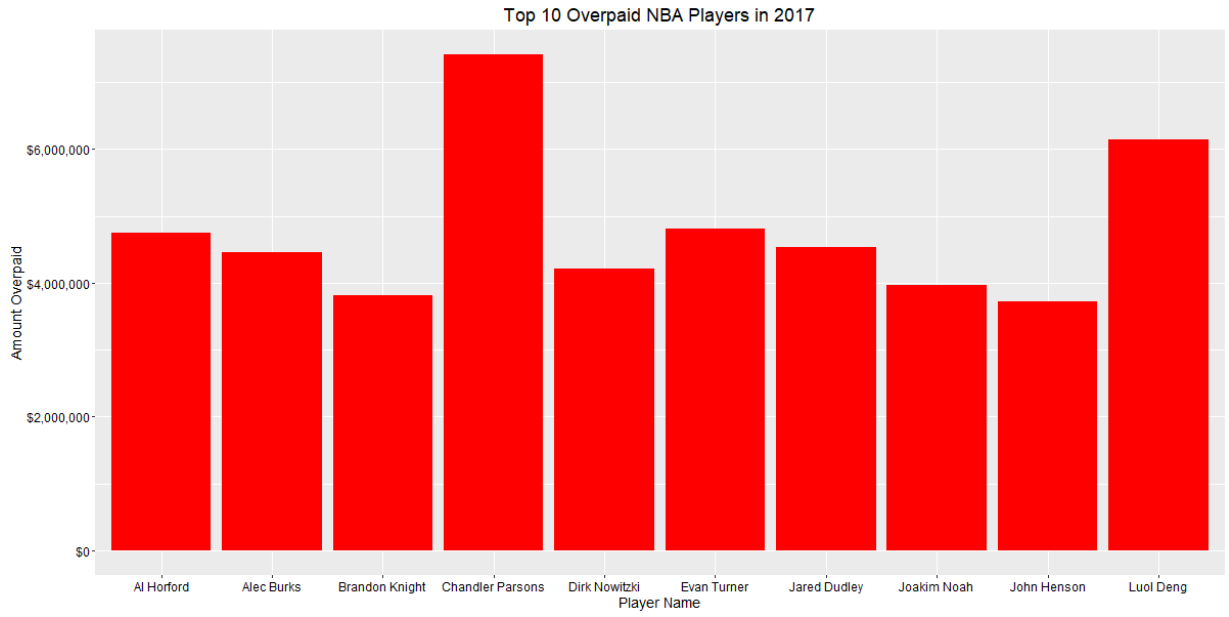
In [114]:
```
final_df_underpaid_17 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_17 = final_df_underpaid_17.head(10)
top_10_underpaid_17
```

Out[114]:

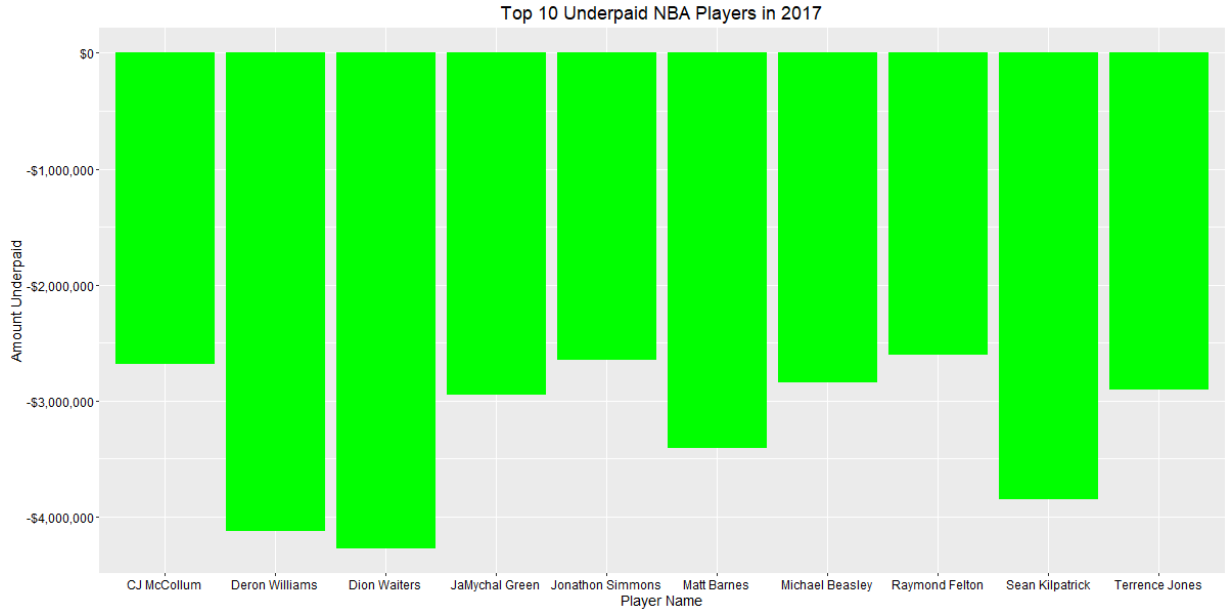
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
363	Dion Waiters	2898000.0	7173128	-4275128.50	25.0	46.0	43.0	
376	Deron Williams	259526.0	4387177	-4127651.00	32.0	64.0	44.0	
203	Sean Kilpatrick	980431.0	4833067	-3852636.00	27.0	70.0	24.0	
22	Matt Barnes	242224.0	3653303	-3411079.25	36.0	74.0	18.0	
137	JaMychal Green	980431.0	3926015	-2945584.50	26.0	77.0	75.0	
195	Terrence Jones	980431.0	3883318	-2902887.75	25.0	54.0	12.0	
30	Michael Beasley	1403611.0	4244590	-2840979.50	28.0	56.0	6.0	
238	CJ McCollum	3219579.0	5900217	-2680638.50	25.0	80.0	80.0	
328	Jonathon Simmons	874636.0	3519596	-2644960.50	27.0	78.0	8.0	
107	Raymond Felton	980431.0	3583678	-2603247.25	32.0	80.0	11.0	

Making the Plots

In [115]:
```
%%R -i top_10_overpaid_17 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_17 <- ggplot(top_10_overpaid_17, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_17.png")
player_overpaid_bar_17
```



In [116]:
```
%%R -i top_10_underpaid_17 -w 1200 -h 600 -u px

player_underpaid_bar_17 <- ggplot(top_10_underpaid_17, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_17.png")
player_underpaid_bar_17
```



Running 2018 Data in the Model

In [117]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_18, labels_18, verbose=False)
y_train_pred1 = xgb_model1.predict(features_18)
y_pred1 = xgb_model1.predict(features_18)

print('Train r2 score: ', r2_score(y_train_pred1, labels_18))
print('Test r2 score: ', r2_score(labels_18, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_18)
test_mse1 = mean_squared_error(y_pred1, labels_18)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
[13:34:30] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.9022915446689753
Test r2 score:  0.9262451282632693
Train RMSE: 2124853.1538
Test RMSE: 2124853.1538
```

In [118]:
```
y_train_xgb = pd.DataFrame(labels_18, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_18, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[118]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct
0	22471910.0	0.585366	0.691275	0.404145	0.391176	0.519774	0.110442
1	2955840.0	0.487805	0.526846	0.212435	0.400000	0.180791	0.220884
2	21461010.0	0.780488	0.838926	0.466321	0.200000	0.242938	0.226908
3	2034120.0	0.463415	0.587248	0.455959	0.444118	0.158192	0.108434
4	1471382.0	0.878049	0.291946	0.015544	0.467647	0.062147	0.092369

In [119]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_18.reset_index(drop=True, inplace=True)
names_18_actual.reset_index(drop=True, inplace=True)
```

In [120]:
```
final_df = pd.concat([y_predict_xgb, final_18], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_18_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[120]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field_
0	20231560.00	0	24.0	76.0	76.0	2487.0	448.0	
1	2535676.25	0	20.0	69.0	19.0	1368.0	174.0	
2	23167536.00	0	32.0	75.0	75.0	2509.0	687.0	
3	2367933.00	0	19.0	72.0	31.0	1441.0	234.0	
4	2608320.50	7	36.0	22.0	0.0	273.0	44.0	

In [121]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [122]:
```
cols = list(final_df.columns.values)
```

In [123]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[123]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
0	Steven Adams	22471910.0	20231560	2240350.00	24.0	76.0	76.0	
1	Bam Adebayo	2955840.0	2535676	420163.75	20.0	69.0	19.0	
2	LaMarcus Aldridge	21461010.0	23167536	-1706526.00	32.0	75.0	75.0	
3	Jarrett Allen	2034120.0	2367933	-333813.00	19.0	72.0	31.0	
4	Tony Allen	1471382.0	2608320	-1136938.50	36.0	22.0	0.0	

In [124]:
```
final_df_overpaid_18 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_18 = final_df_overpaid_18.head(10)
top_10_overpaid_18
```

Out[124]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
223	Paul Millsap	31269231.0	23643266	7625965.00	32.0	38.0	37.0	
248	Chandler Parsons	23112004.0	15572520	7539484.00	29.0	36.0	8.0	
167	Tyler Johnson	19245370.0	12092814	7152556.00	25.0	72.0	39.0	
68	Mike Conley	28530608.0	21993216	6537392.00	30.0	12.0	12.0	
140	Solomon Hill	12236535.0	5796587	6439947.50	26.0	12.0	1.0	
51	Kentavious Caldwell-Pope	17745894.0	11457120	6288774.00	24.0	74.0	74.0	
85	Gorgui Dieng	14112360.0	7875157	6237202.50	28.0	79.0	0.0	
127	Maurice Harkless	10162922.0	4146807	6016114.25	24.0	59.0	36.0	
21	Nicolas Batum	24000000.0	18267652	5732348.00	29.0	64.0	64.0	
19	Harrison Barnes	23112004.0	17803270	5308734.00	25.0	77.0	77.0	

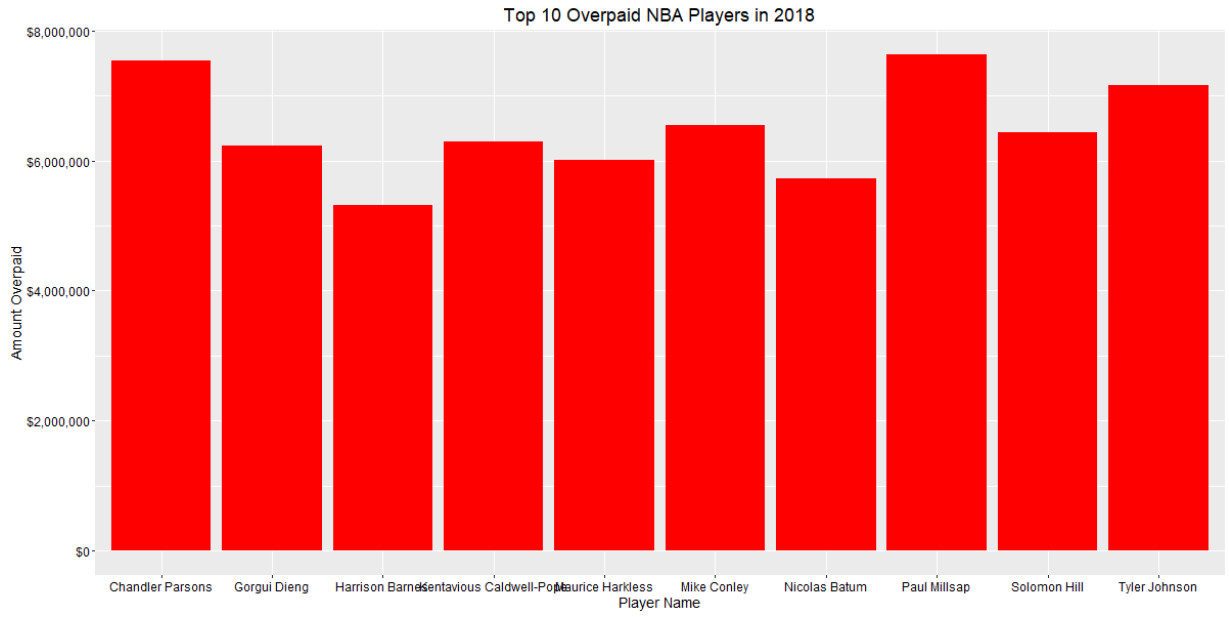
In [125]:
```
final_df_underpaid_18 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_18 = final_df_underpaid_18.head(10)
top_10_underpaid_18
```

Out[125]:

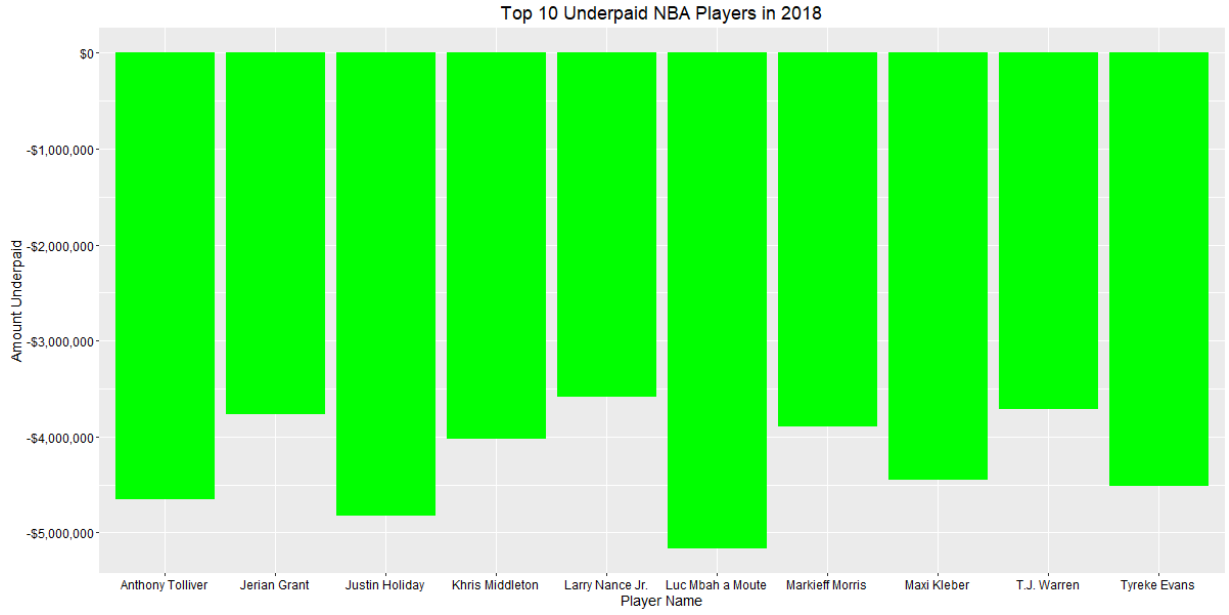
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
207	Luc Mbah a Moute	1471382.0	6637813	-5166431.0	31.0	61.0	15.0	
142	Justin Holiday	4384616.0	9208958	-4824342.0	28.0	72.0	72.0	
300	Anthony Tolliver	3290000.0	7945485	-4655485.0	32.0	79.0	14.0	
96	Tyreke Evans	3290000.0	7806513	-4516513.5	28.0	52.0	32.0	
176	Maxi Kleber	815615.0	5262059	-4446444.5	26.0	72.0	36.0	
219	Khris Middleton	1300000.0	5324410	-4024410.0	26.0	82.0	82.0	
227	Markieff Morris	8600000.0	12495510	-3895510.0	28.0	73.0	73.0	
119	Jerian Grant	2639314.0	6412118	-3772804.5	25.0	74.0	26.0	
313	T.J. Warren	3152931.0	6863812	-3710881.5	24.0	65.0	65.0	
234	Larry Nance Jr.	2272391.0	5853496	-3581105.0	25.0	66.0	27.0	

Making the Plots

In [126]:
```
%%R -i top_10_overpaid_18 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_18 <- ggplot(top_10_overpaid_18, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour="black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))

ggsave(file="player_overpaid_bar_18.png")
player_overpaid_bar_18
```



In [127]:
```
%%R -i top_10_underpaid_18 -w 1200 -h 600 -u px

player_underpaid_bar_18 <- ggplot(top_10_underpaid_18, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text=element_text(size=12, colour = "black"),
     axis.title.x=element_text(size=14),
     axis.title.y=element_text(size=14))
ggsave(file="player_underpaid_bar_18.png")
player_underpaid_bar_18
```



Running 2019 Data in the Model

In [128]:
```
import sys
!{sys.executable} -m pip install xgboost
from xgboost import XGBRegressor, plot_importance
xgb_model1 = XGBRegressor()
xgb_model1.fit(features_19, labels_19, verbose=False)
y_train_pred1 = xgb_model1.predict(features_19)
y_pred1 = xgb_model1.predict(features_19)

print('Train r2 score: ', r2_score(y_train_pred1, labels_19))
print('Test r2 score: ', r2_score(labels_19, y_pred1))
train_mse1 = mean_squared_error(y_train_pred1, labels_19)
test_mse1 = mean_squared_error(y_pred1, labels_19)
train_rmse1 = np.sqrt(train_mse1)
test_rmse1 = np.sqrt(test_mse1)
print('Train RMSE: %.4f' % train_rmse1)
print('Test RMSE: %.4f' % test_rmse1)
```

```
Requirement already satisfied: xgboost in c:\users\jeffery.peterson.tm\appdata
\local\continuum\anaconda3\lib\site-packages (0.90)
Requirement already satisfied: numpy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in c:\users\jeffery.peterson.tm\appdata\lo
cal\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
[13:36:29] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
Train r2 score:  0.8977170573157198
Test r2 score:  0.9229840607401892
Train RMSE: 2246050.5422
Test RMSE: 2246050.5422
```

In [129]:
```
y_train_xgb = pd.DataFrame(labels_19, columns=["Salary"])
y_predict_xgb = pd.DataFrame(y_train_pred1, columns=["Prediction"])

train_predictions_xgb = pd.DataFrame(np.concatenate([nba_subset_19, y_predict_xgb
train_predictions_xgb.head(5)
```

Out[129]:

	Salary	Age	Player_Efficiency_Rating	Blocks	Turnovers_Pct	Steals	Assists_Pct
0	270014.0	0.523810	0.231003	0.025126	0.735075	0.082353	0.416842
1	24157303.0	0.595238	0.562310	0.381910	0.470149	0.688235	0.138947
2	2955840.0	0.500000	0.544073	0.326633	0.638060	0.417647	0.298947
3	22347015.0	0.785714	0.696049	0.537688	0.328358	0.252941	0.244211
4	2074320.0	0.547619	0.227964	0.030151	0.518657	0.035294	0.187368

In [130]:
```
y_predict_xgb.reset_index(drop=True, inplace=True)
final_19.reset_index(drop=True, inplace=True)
names_19_actual.reset_index(drop=True, inplace=True)
```

In [131]:
```
final_df = pd.concat([y_predict_xgb, final_19], axis=1)
final_df = final_df.drop(['Player_Name'], axis=1)
final_df = final_df.drop(final_df.columns[3], axis=1)

final_df = pd.concat([final_df, names_19_actual], axis=1)
# final_df = final_df.dropna()
final_df.head(5)
```

Out[131]:

	Prediction	Position	Age	Games_Played	Games_Started	Minutes_Played	Field_Goals	Field_
0	1346993.25	5	22.0	34.0	1.0	428.0	38.0	
1	20528162.00	0	25.0	80.0	80.0	2669.0	481.0	
2	2871048.25	0	21.0	82.0	28.0	1913.0	280.0	
3	22380306.00	0	33.0	81.0	81.0	2687.0	684.0	
4	3076324.75	10	23.0	38.0	2.0	416.0	67.0	

In [132]:
```
final_df['Salary-Prediction'] = final_df['Salary'] - final_df['Prediction']
```

Rearranging the columns

In [133]:
```
cols = list(final_df.columns.values)
```

In [134]:
```
cols = ['Player_Name', 'Salary', 'Prediction', 'Salary-Prediction', 'Age',
 'Games_Played',
 'Games_Started',
 'Minutes_Played',
 'Field_Goals',
 'Field_Goals_Attempted',
 'Field_Goal_Pct',
 'Three_Pointers',
 'Three_Pointers_Attempted',
 'Three_Pointers_Pct',
 'Two_Pointers',
 'Two_Pointers_Attempted',
 'Two_Pointers_Pct',
 'Effective_Field_Goal_Pct',
 'Free_Throws',
 'Free_Throws_Attempted',
 'Free_Throws_Pct',
 'Offensive_Rebounds',
 'Defensive_Rebounds',
 'Total_Rebounds',
 'Assists',
 'Steals',
 'Blocks',
 'Turnovers',
 'Personal_Fouls',
 'Player_Efficiency_Rating',
 'Turnovers_Pct',
 'Assists_Pct',
 'Defensive_Box_Plus_Minus',
 'Defensive_Rebound_Pct',
 'Usage_Pct']

final_df = final_df[cols]
final_df['Prediction'] = final_df['Prediction'].astype('int64')

final_df.head(5)
```

Out[134]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
0	Jaylen Adams	270014.0	1346993	-1076979.25	22.0	34.0	1.0	
1	Steven Adams	24157303.0	20528162	3629141.00	25.0	80.0	80.0	
2	Bam Adebayo	2955840.0	2871048	84791.75	21.0	82.0	28.0	
3	LaMarcus Aldridge	22347015.0	22380306	-33291.00	33.0	81.0	81.0	
4	Grayson Allen	2074320.0	3076324	-1002004.75	23.0	38.0	2.0	

In [135]:
```
final_df_overpaid_19 = final_df.sort_values("Salary-Prediction", ascending = Fals
top_10_overpaid_19 = final_df_overpaid_19.head(10)
top_10_overpaid_19
```

Out[135]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minute
171	Gordon Hayward	31214295.0	20861014	10353281.0	28.0	72.0	18.0	
300	Chandler Parsons	24107258.0	16017678	8089580.0	30.0	25.0	3.0	
298	Jabari Parker	20000000.0	12211210	7788790.0	23.0	64.0	17.0	
203	Tyler Johnson	19356932.0	11852434	7504498.0	26.0	57.0	22.0	
266	Paul Millsap	29230769.0	22118664	7112105.0	33.0	70.0	65.0	
9	Ryan Anderson	20421546.0	13536294	6885252.0	30.0	25.0	8.0	
175	George Hill	19000000.0	12303144	6696856.0	32.0	60.0	13.0	
212	Michael Kidd-Gilchrist	13000000.0	7017022	5982977.5	25.0	64.0	3.0	
293	Victor Oladipo	21000000.0	15070530	5929470.0	26.0	36.0	36.0	
90	Allen Crabbe	18500000.0	12762518	5737482.0	26.0	43.0	20.0	

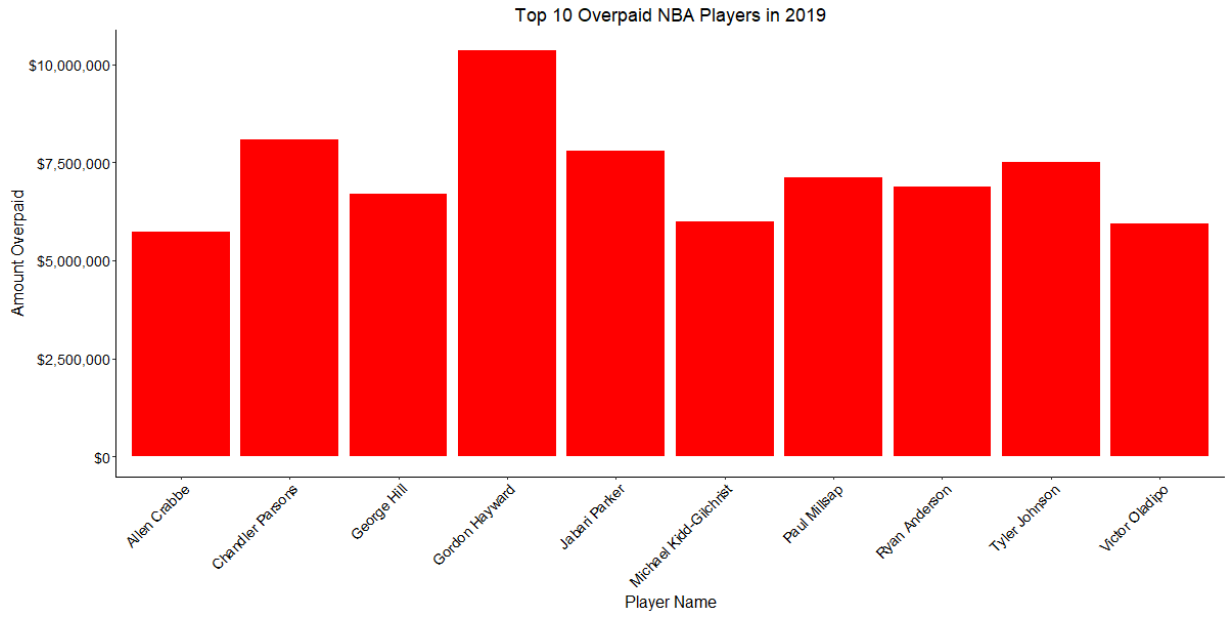
In [136]:
```
final_df_underpaid_19 = final_df.sort_values("Salary-Prediction", ascending = Tru
top_10_underpaid_19 = final_df_underpaid_19.head(10)
top_10_underpaid_19
```

Out[136]:

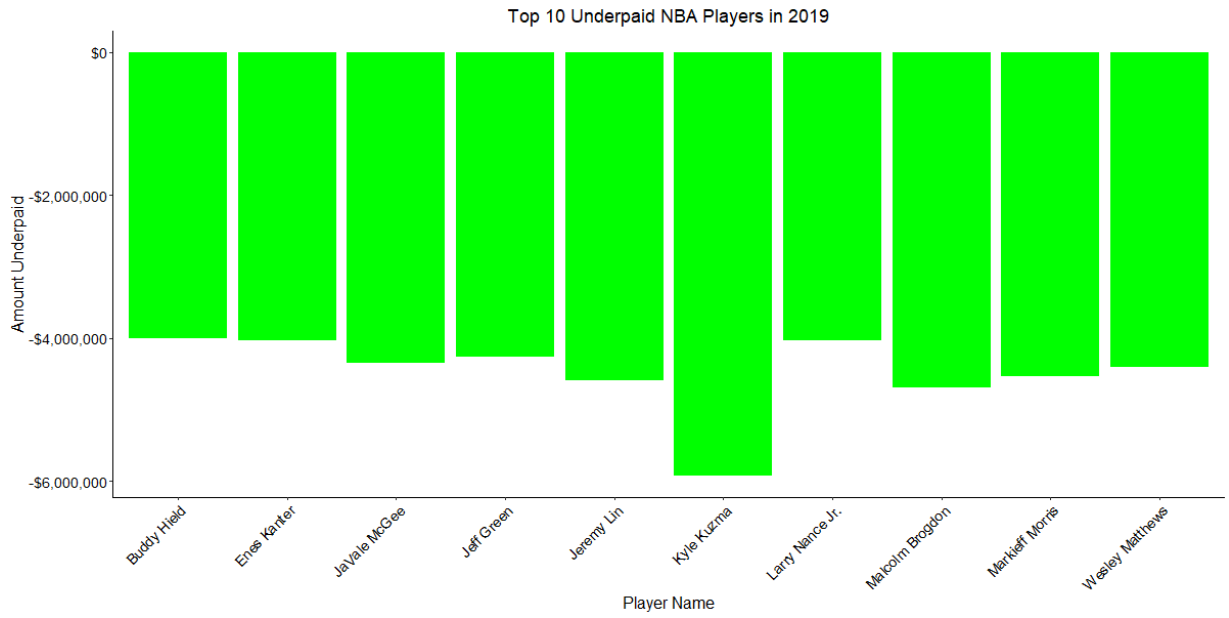
	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started	Minutes
219	Kyle Kuzma	1689840.0	7623985	-5934145.5	23.0	70.0	68.0	
52	Malcolm Brogdon	1544951.0	6238412	-4693461.0	26.0	64.0	64.0	
231	Jeremy Lin	487109.0	5085279	-4598170.0	30.0	74.0	4.0	
272	Markieff Morris	427288.0	4961320	-4534032.0	29.0	58.0	16.0	
246	Wesley Matthews	512746.0	4922741	-4409995.5	32.0	69.0	68.0	
252	JaVale McGee	1512601.0	5862542	-4349941.5	31.0	75.0	62.0	
155	Jeff Green	1512601.0	5773879	-4261278.5	32.0	77.0	44.0	
279	Larry Nance Jr.	2272391.0	6302699	-4030308.5	26.0	67.0	30.0	
210	Enes Kanter	487109.0	4515775	-4028666.0	26.0	67.0	31.0	
174	Buddy Hield	3844760.0	7853181	-4008421.0	26.0	82.0	82.0	

Making the Plots

In [137]:
```
%%R -i top_10_overpaid_19 -w 1200 -h 600 -u px
options(scipen=999)
player_overpaid_bar_19 <- ggplot(top_10_overpaid_19, aes(x = Player_Name, y=Salar
labs(x = "Player Name", y = "Amount Overpaid") + ggtitle("Top 10 Overpaid NBA Pla
scale_y_continuous(labels=scales::dollar_format(prefix="$")) + theme_classic() +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text.x=element_text(size=14, colour="black", angle = 45, hjust = 1.0),
      axis.text.y=element_text(size=14, colour="black"),
     axis.title.x=element_text(size=16),
     axis.title.y=element_text(size=16))

ggsave(file="player_overpaid_bar_19.png")
player_overpaid_bar_19
```



In [138]:
```
%%R -i top_10_underpaid_19 -w 1200 -h 600 -u px

player_underpaid_bar_19 <- ggplot(top_10_underpaid_19, aes(x = Player_Name, y=Sal
labs(x = "Player Name", y = "Amount Underpaid") + ggtitle("Top 10 Underpaid NBA P
scale_y_continuous(labels=scales::dollar_format(prefix="$")) + theme_classic() +
theme(plot.title = element_text(hjust = 0.5, size=18),
      axis.text.x=element_text(size=14, colour="black", angle = 45, hjust = 1.0),
      axis.text.y=element_text(size=14, colour="black"),
     axis.title.x=element_text(size=16),
     axis.title.y=element_text(size=16))

ggsave(file="player_underpaid_bar_19.png")
player_underpaid_bar_19
```



Summary of Model Generalization

Year	R^2	RMSE
2010	0.946	1,099,627
2011	0.949	1,070,874
2012	0.957	989,150
2013	0.942	1,132,491
2014	0.946	1,174,769
2015	0.940	1,204,874
2016	0.955	1,147,450
2017	0.940	1,641,827
2018	0.926	2,124,853
2019	0.923	2,246,051

Most Overpaid Players from 2010-2019

In [185]:
```
df_summary_overpaid = pd.concat([top_10_overpaid_10, top_10_overpaid_11, top_10_c
df_summary_overpaid
```

Out[185]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Starte
195	Tim Duncan	22,183,220.0	880920	21,302,299.5625	33.0	78.0	77.(
109	Kobe Bryant	23,034,375.0	3125619	19,908,755.25	31.0	73.0	73.(
91	Elton Brand	14,858,472.0	907392	13,951,079.1875	30.0	76.0	57.(
133	Vince Carter	16,123,250.0	3793071	12,330,178.75	33.0	75.0	74.(
21	Carmelo Anthony	15,779,912.0	4990881	10,789,031.0	25.0	69.0	69.(
337	Andre Iguodala	12,200,000.0	1443273	10,756,726.75	26.0	82.0	82.(
321	Josh Howard	10,890,000.0	242632	10,647,367.875	29.0	35.0	12.(
87	Chris Bosh	15,779,912.0	5605472	10,174,440.0	25.0	70.0	70.(
73	Chauncey Billups	12,100,000.0	2014230	10,085,770.0	33.0	73.0	73.(

In [186]:
```
df_summary_overpaid = df_summary_overpaid[["Player_Name", "Salary-Prediction"]]
df_summary_overpaid
```

Out[186]:

	Player_Name	Salary-Prediction
195	Tim Duncan	21,302,299.5625
109	Kobe Bryant	19,908,755.25
91	Elton Brand	13,951,079.1875
133	Vince Carter	12,330,178.75
21	Carmelo Anthony	10,789,031.0
337	Andre Iguodala	10,756,726.75
321	Josh Howard	10,647,367.875
87	Chris Bosh	10,174,440.0
73	Chauncey Billups	10,085,770.0
167	Baron Davis	10,013,370.125
199	Rashard Lewis	4,568,497.0
190	Andrei Kirilenko	3.867.288.0

In [187]:
```
years_total = df_summary_overpaid.groupby(['Player_Name']).count().sort_values('S
years_total
```

Out[187]:

Player_Name	Salary-Prediction
Andre Iguodala	4
Chandler Parsons	4
Joe Johnson	3
Paul Millsap	2
Caron Butler	2
Kris Humphries	2
Marcus Thornton	2
Michael Kidd-Gilchrist	2
Eric Gordon	2
Deron Williams	2
Rashard Lewis	2

In [188]:
```
#pd.options.display.float_format = '{:.0f}'.format
pd.options.display.float_format = '{:,}'.format
amount_total = df_summary_overpaid.groupby(['Player_Name']).sum().sort_values('Sa
amount_total
```

Out[188]:

Player_Name	Salary-Prediction
Chandler Parsons	26,630,132.0
Tim Duncan	21,302,299.5625
Kobe Bryant	19,908,755.25
Andre Iguodala	19,551,666.75
Vince Carter	15,397,011.75
Paul Millsap	14,738,070.0
Tyler Johnson	14,657,054.0
Elton Brand	13,951,079.1875
Joe Johnson	11,777,641.0
Carmelo Anthony	10,789,031.0
Josh Howard	10.647.367.875

In [202]:
```
overpaid_result = pd.merge(years_total,
                           amount_total[["Salary-Prediction"]],
                           on="Player_Name")
overpaid_result.columns = ["Years Overpaid", "Total Amount Overpaid"]
overpaid_result['Total Amount Overpaid'] = overpaid_result['Total Amount Overpaic
overpaid_result
```

Out[202]:

Player_Name	Years Overpaid	Total Amount Overpaid
Andre Iguodala	4	$19,551,666.75
Chandler Parsons	4	$26,630,132.00
Joe Johnson	3	$11,777,641.00
Paul Millsap	2	$14,738,070.00
Caron Butler	2	$5,770,528.00
Kris Humphries	2	$8,248,974.50
Marcus Thornton	2	$6,880,623.50
Michael Kidd-Gilchrist	2	$9,570,985.00
Eric Gordon	2	$7,066,979.00
Deron Williams	2	$7,492,937.00
Rashard Lewis	2	$10.542.496.00

Most Underpaid Players from 2010-2019

In [212]:
```
df_summary_underpaid = pd.concat([top_10_underpaid_10, top_10_underpaid_11, top_1
df_summary_underpaid
```

Out[212]:

	Player_Name	Salary	Prediction	Salary-Prediction	Age	Games_Played	Games_Started
97	Aaron Brooks	1,118,520.0	21628846	-20,510,326.0	25.0	82.0	82.0
239	Rudy Gay	3,280,997.0	19312470	-16,031,473.0	23.0	80.0	80.0
11	Rafer Alston	475,875.0	15781898	-15,306,023.0	33.0	52.0	38.0
261	Stephen Graham	825,497.0	15574602	-14,749,105.0	27.0	70.0	8.0
307	Jordan Hill	2,483,280.0	16427102	-13,943,822.0	22.0	47.0	0.0
159	Dante Cunningham	457,588.0	14191066	-13,733,478.0	22.0	63.0	2.0
175	Carlos Delfino	3,500,000.0	15601446	-12,101,446.0	27.0	75.0	66.0
325	Lester Hudson	457,588.0	12029192	-11,571,604.0	25.0	25.0	0.0
45	J.J. Barea	1,657,500.0	12947800	-11,290,300.0	25.0	78.0	18.0

In [213]:
```
df_summary_underpaid = df_summary_underpaid[["Player_Name", "Salary-Prediction"]]
df_summary_underpaid
```

Out[213]:

	Player_Name	Salary-Prediction
97	Aaron Brooks	-20,510,326.0
239	Rudy Gay	-16,031,473.0
11	Rafer Alston	-15,306,023.0
261	Stephen Graham	-14,749,105.0
307	Jordan Hill	-13,943,822.0
159	Dante Cunningham	-13,733,478.0
175	Carlos Delfino	-12,101,446.0
325	Lester Hudson	-11,571,604.0
45	J.J. Barea	-11,290,300.0
139	Wilson Chandler	-10,485,682.0
320	David West	-3,180,276.0
42	Earl Bovkins	-2.618.576.75

In [214]:
```
years_total_under = df_summary_underpaid.groupby(['Player_Name']).count().sort_va
years_total_under
```

Out[214]:

	Salary-Prediction
Player_Name	
Pau Gasol	2
Chandler Parsons	2
P.J. Tucker	2
J.J. Barea	2
Deron Williams	2
Raymond Felton	2
David West	2
Matt Barnes	2
Markieff Morris	2
Nick Young	2
Tony Allen	2

In [215]:
```
amount_total_under = df_summary_underpaid.groupby(['Player_Name']).sum().sort_val
amount_total_under
```

Out[215]:

Player_Name	Salary-Prediction
Aaron Brooks	-20,510,326.0
Rudy Gay	-16,031,473.0
Rafer Alston	-15,306,023.0
Stephen Graham	-14,749,105.0
Jordan Hill	-13,943,822.0
J.J. Barea	-13,850,090.0
Dante Cunningham	-13,733,478.0
Carlos Delfino	-12,101,446.0
Lester Hudson	-11,571,604.0
Wilson Chandler	-10,485,682.0
Markieff Morris	-8.429.542.0

In [217]:
```
underpaid_result = pd.merge(years_total_under,
                            amount_total_under[["Salary-Prediction"]],
                            on="Player_Name")
underpaid_result.columns = ["Years Underpaid", "Total Amount Underpaid"]
underpaid_result['Total Amount Underpaid'] = underpaid_result['Total Amount Under
underpaid_result.sort_values(['Years Underpaid', 'Total Amount Underpaid'], ascer
```

Out[217]:

Player_Name	Years Underpaid	Total Amount Underpaid
J.J. Barea	2	$-13,850,090.00
Tony Allen	2	$-3,723,978.50
Nick Young	2	$-4,238,423.00
Leandro Barbosa	2	$-5,007,441.00
Raymond Felton	2	$-5,023,405.25
P.J. Tucker	2	$-5,053,912.50
Chandler Parsons	2	$-5,178,526.00
David West	2	$-5,228,552.00
Pau Gasol	2	$-5,267,324.00
Matt Barnes	2	$-5,495,066.25
Deron Williams	2	$-6.192.240.50

In [ ]: