In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.style as style
style.use('fivethirtyeight')
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, roc_auc_score
, roc_curve, confusion_matrix, classification_report, accuracy_score
from sklearn.utils.multiclass import type_of_target
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
# read in our data
df = pd.read_csv('/Users/calebdill/Library/Mobile Documents/com~apple~Cl
oudDocs/Syracuse/Data Analytics/Project/Combine Data.csv')

####### Data Cleansing ########

# view first five rows
df.head()
```

Out[2]:

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year	P
0	John Abraham	OLB	76	252	4.55	NaN	NaN	NaN	NaN	NaN	2000	Abra
1	Shaun Alexander	RB	72	218	4.58	NaN	NaN	NaN	NaN	NaN	2000	Alex
2	Darnell Alford	OT	76	334	5.56	25.0	23.0	94.0	8.48	4.98	2000	Alfol
3	Kyle Allamon	TE	74	253	4.97	29.0	NaN	104.0	7.29	4.49	2000	
4	Rashard Anderson	CB	74	206	4.55	34.0	NaN	123.0	7.18	4.15	2000	Andel

In [3]:
```
# view the data types of each column
df.dtypes
```

Out[3]:
```
Player        object
Pos           object
Ht             int64
Wt             int64
Forty        float64
Vertical     float64
BenchReps    float64
BroadJump    float64
Cone         float64
Shuttle      float64
Year           int64
Pfr_ID        object
AV           float64
Team          object
Round        float64
Pick         float64
dtype: object
```

In [4]:
```
# count the number of rows missing data in each column
df.apply(lambda x: x.isnull().sum(), axis = 'rows')
```

Out[4]:
```
Player          0
Pos             0
Ht              0
Wt              0
Forty         172
Vertical     1422
BenchReps    2006
BroadJump    1464
Cone         2225
Shuttle      2155
Year            0
Pfr_ID       1323
AV              0
Team         2480
Round        2480
Pick         2480
dtype: int64
```

In [5]:
```
## At first look, we should have enough data to just remove those with m
issing values as opposed to trying to fill them.
## However, after further analysis it appears the missing data is highly
dependent on position.
## For example, it makes sense that QBs wouldn't participate in as many
 events at the combine, as that is common.
## How to handle these missing values? 2 options: replace with position
 median or replace with 0.
## For now, let's try the position median approach.
## Assuming Team, Round, and Pick are all blank means the player did not
get drafted. How to handle?
## Going to do 2 things:
## 1. Round, pick, and team will be set as "undrafted" for the missing v
alues
## 2. We will create an additional column which indicates whether or not
a player went undrafted

# First, let's go ahead and convert round and pick to category types
df[['Round', 'Pick']] = df[['Round', 'Pick']].astype('category')
df.dtypes
```

Out[5]:
```
Player         object
Pos            object
Ht              int64
Wt              int64
Forty         float64
Vertical      float64
BenchReps     float64
BroadJump     float64
Cone          float64
Shuttle       float64
Year            int64
Pfr_ID         object
AV            float64
Team           object
Round        category
Pick         category
dtype: object
```

In [6]:
```
# Replace missing values for Round, Team, and Pick with "undrafted"
df[['Team', 'Round', 'Pick']] = df[['Team', 'Round', 'Pick']].replace(np
.NaN, 'Undrafted')
```

In [7]:
```
# view players which went undrafted
df.loc[df['Team'] == 'Undrafted']
```

Out[7]:

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year
3	Kyle Allamon	TE	74	253	4.97	29.0	NaN	104.0	7.29	4.49	2000
5	Jake Arians	K	70	202	NaN	NaN	NaN	NaN	NaN	NaN	2000
7	Corey Atkins	OLB	72	237	4.72	31.0	21.0	112.0	7.96	4.39	2000
8	Kyle Atteberry	K	72	167	NaN	NaN	NaN	NaN	NaN	NaN	2000
10	John Baker	P	75	227	NaN	NaN	NaN	NaN	NaN	NaN	2000
11	Mark Baniewicz	OT	78	312	5.34	28.0	20.0	96.0	7.72	4.73	2000
15	Andrew Bayes	P	74	200	NaN	NaN	NaN	NaN	NaN	NaN	2000
16	Terrance Beadles	OG	75	312	5.19	NaN	29.0	NaN	NaN	NaN	2000
19	Matt Beck	ILB	75	234	4.65	NaN	NaN	NaN	NaN	NaN	2000
24	Michael Boireau	DE	76	274	5.09	29.0	26.0	105.0	7.68	4.49	2000
26	Carl Bradley	DT	74	300	5.03	NaN	NaN	NaN	NaN	NaN	2000
28	Robert Brannon	DT	76	309	5.19	28.5	28.0	99.0	7.82	4.70	2000
29	Travis Brown-01	QB	75	218	5.01	29.0	NaN	101.0	7.54	4.87	2000
32	Keith Brown	RB	71	216	4.66	30.0	NaN	117.0	6.64	4.08	2000
33	Demario Brown	RB	72	217	4.72	31.0	NaN	107.0	7.17	4.39	2000
35	Lamont Bryant	DE	75	260	4.91	NaN	17.0	NaN	NaN	NaN	2000
38	Bill Burke	QB	76	206	5.03	28.5	NaN	107.0	7.46	4.42	2000
40	David Byrd	CB	71	202	4.53	36.0	14.0	117.0	7.18	4.39	2000
43	Giovanni Carmazzi	QB	75	224	4.74	36.5	NaN	119.0	6.94	4.23	2000
46	Kwame Cavil	WR	74	208	4.54	39.5	NaN	118.0	NaN	NaN	2000
47	Chris Chaloupka	QB	74	216	4.96	NaN	NaN	NaN	NaN	NaN	2000
50	Chrys Chukwuma	RB	73	232	4.56	NaN	NaN	115.0	NaN	NaN	2000
51	Joey Chustz	OT	79	304	5.08	31.0	28.0	103.0	7.65	4.65	2000
52	Pedro Cirino	FS	71	197	4.59	NaN	NaN	NaN	NaN	NaN	2000
53	John Saint Clair	C	77	302	5.20	NaN	25.0	NaN	NaN	NaN	2000

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year
60	Chris Coleman	WR	72	211	4.61	32.5	NaN	110.0	6.79	4.10	2000
63	Kurth Connell	OT	76	291	5.49	33.0	27.0	103.0	8.14	4.84	2000
64	Rameel Connor	DE	75	276	5.00	35.5	25.0	121.0	7.65	4.45	2000
65	D.J. Cooper	DT	75	277	5.22	28.0	NaN	108.0	7.41	4.49	2000
66	Ashley Cooper	SS	70	208	4.67	NaN	16.0	117.0	NaN	NaN	2000
...	...	...	...	...	...	...	...	...	...	...	...
6188	Fred Warner	OLB	75	236	4.64	38.5	21.0	119.0	6.90	4.28	2018
6189	Chris Warren	RB	74	247	4.69	33.0	25.0	121.0	6.98	4.18	2018
6190	James Washington	WR	71	213	4.54	34.5	14.0	120.0	7.11	4.32	2018
6191	Armani Watts	S	71	205	NaN	35.0	13.0	120.0	7.25	4.37	2018
6192	Jester Weah	WR	74	211	4.43	38.0	15.0	129.0	7.24	4.41	2018
6193	Toby Weathersby	OT	76	317	5.34	24.5	28.0	106.0	8.55	5.38	2018
6194	Damon Webb	S	71	195	4.62	NaN	17.0	NaN	NaN	NaN	2018
6195	David Wells	TE	78	256	4.75	33.0	20.0	116.0	7.50	4.59	2018
6196	Sean Welsh	C	75	306	5.43	25.0	20.0	103.0	7.90	4.81	2018
6197	Karaun White	WR	73	206	4.52	33.5	24.0	118.0	7.16	4.57	2018
6198	Kyzir White	S	74	216	NaN	35.5	21.0	113.0	NaN	NaN	2018
6199	Mike White	QB	77	224	5.09	27.0	NaN	96.0	7.40	4.40	2018
6200	Jordan Whitehead	S	71	195	NaN	NaN	21.0	NaN	NaN	NaN	2018
6201	Jojo Wicker	DE	75	296	5.16	30.0	25.0	105.0	NaN	NaN	2018
6202	Jake Wieneke	WR	76	221	4.67	34.0	9.0	114.0	7.24	4.37	2018
6203	Jordan Wilkins	RB	73	216	NaN	36.0	16.0	117.0	NaN	4.27	2018
6204	Connor Williams	OT	77	296	5.05	34.0	26.0	112.0	7.83	4.63	2018
6205	Darrel Williams	RB	72	225	4.72	32.0	22.0	109.0	NaN	4.21	2018
6206	Cedrick Wilson-02	WR	74	197	4.55	37.0	9.0	121.0	6.89	4.23	2018

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year
6207	Eddy Wilson	DT	76	301	5.37	NaN	26.0	96.0	NaN	NaN	2018
6208	Jeff Wilson	RB	71	210	NaN	NaN	15.0	NaN	NaN	NaN	2018
6209	Javon Wims	WR	75	215	4.53	33.5	NaN	113.0	7.00	NaN	2018
6210	Anthony Winbush	EDGE	73	249	NaN	NaN	25.0	110.0	NaN	NaN	2018
6211	Ryan Winslow	P	77	217	NaN	NaN	NaN	NaN	NaN	NaN	2018
6212	Logan Woodside	QB	73	213	4.79	31.5	NaN	103.0	6.94	4.15	2018
6213	Chris Worley	ILB	74	238	4.86	29.5	15.0	NaN	NaN	NaN	2018
6214	Isaiah Wynn	G	75	313	NaN	NaN	NaN	NaN	NaN	NaN	2018
6215	Isaac Yiadom	CB	73	190	4.52	NaN	8.0	120.0	NaN	4.18	2018
6216	Kenny Young	ILB	73	236	4.60	36.0	23.0	117.0	7.38	4.48	2018
6217	Trevon Young	EDGE	76	258	4.78	33.0	25.0	114.0	6.99	4.40	2018

2480 rows × 16 columns

In [8]:
```
# Add column indicating whether or not a player was drafted
df['Player Drafted'] = np.where(df['Team'] == 'Undrafted', 0, 1)
```

In [9]:
```
# Get median value for each position for our columns containing continuo
us data
dfPosMed = df.groupby('Pos').median()[['Forty', 'Vertical', 'BenchReps',
'BroadJump', 'Cone', 'Shuttle']]
```

In [10]:
```
# for anything we still don't have a value for, replace with 0
dfPosMed = dfPosMed.replace(np.NaN, 0)
```

In [11]:
```
# rename columns and join them into our original data set
dfPosMed = dfPosMed.rename(columns = {'Forty' : 'Median Forty', 'Vertica
l' : 'Median Vertical',
'BenchReps' : 'Median BenchReps', 'BroadJump': 'Median BroadJump','Cone'
: 'Median Cone', 'Shuttle' : 'Median Shuttle'})
```

In [12]:
```
dfPosMed = dfPosMed.reset_index()

df = df.merge(dfPosMed, on = 'Pos', how = 'left')
```

In [13]:
```
# replace missing values with median
df['Forty'] = df['Forty'].replace(np.NaN, df['Median Forty'])
df['Vertical'] = df['Vertical'].replace(np.NaN, df['Median Vertical'])
df['BenchReps'] = df['BenchReps'].replace(np.NaN, df['Median BenchReps'
])
df['BroadJump'] = df['BroadJump'].replace(np.NaN, df['Median BroadJump'
])
df['Cone'] = df['Cone'].replace(np.NaN, df['Median Cone'])
df['Shuttle'] = df['Shuttle'].replace(np.NaN, df['Median Shuttle'])
```

In [14]:
```
# check for other nulls
df.apply(lambda x: x.isnull().sum(), axis = 'rows')
```

Out[14]:
```
Player                 0
Pos                    0
Ht                     0
Wt                     0
Forty                  0
Vertical               0
BenchReps              0
BroadJump              0
Cone                   0
Shuttle                0
Year                   0
Pfr_ID              1323
AV                     0
Team                   0
Round                  0
Pick                   0
Player Drafted         0
Median Forty           0
Median Vertical        0
Median BenchReps       0
Median BroadJump       0
Median Cone            0
Median Shuttle         0
dtype: int64
```

In [15]:
```
# explore null Pfr_IDs
df.loc[df['Pfr_ID'].isnull()]
# Can't drop null Pfr_IDs since this would drop our Undrafted players an
d skew the data
# this is just an ID, so we can remove the column altogether and just cr
eate an index for each player
```

Out[15]:

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	...	
3	Kyle Allamon	TE	74	253	4.97	29.00	20.0	104.0	7.290	4.49	...	Ur
8	Kyle Atteberry	K	72	167	4.92	33.50	13.5	113.5	0.000	0.00	...	Ur
11	Mark Baniewicz	OT	78	312	5.34	28.00	20.0	96.0	7.720	4.73	...	Ur
15	Andrew Bayes	P	74	200	4.90	30.50	16.0	114.0	7.280	4.41	...	Ur
16	Terrance Beadles	OG	75	312	5.19	27.50	29.0	100.0	7.910	4.78	...	Ur
19	Matt Beck	ILB	75	234	4.65	33.00	22.0	115.0	7.190	4.30	...	Ur
24	Michael Boireau	DE	76	274	5.09	29.00	26.0	105.0	7.680	4.49	...	Ur
26	Carl Bradley	DT	74	300	5.03	29.50	27.0	105.0	7.695	4.64	...	Ur
28	Robert Brannon	DT	76	309	5.19	28.50	28.0	99.0	7.820	4.70	...	Ur
32	Keith Brown	RB	71	216	4.66	30.00	19.0	117.0	6.640	4.08	...	Ur
33	Demario Brown	RB	72	217	4.72	31.00	19.0	107.0	7.170	4.39	...	Ur
38	Bill Burke	QB	76	206	5.03	28.50	19.0	107.0	7.460	4.42	...	Ur
40	David Byrd	CB	71	202	4.53	36.00	14.0	117.0	7.180	4.39	...	Ur
43	Giovanni Carmazzi	QB	75	224	4.74	36.50	19.0	119.0	6.940	4.23	...	Ur
47	Chris Chaloupka	QB	74	216	4.96	31.00	19.0	110.0	7.130	4.33	...	Ur
50	Chrys Chukwuma	RB	73	232	4.56	34.00	19.0	115.0	7.060	4.24	...	Ur
51	Joey Chustz	OT	79	304	5.08	31.00	28.0	103.0	7.650	4.65	...	Ur
52	Pedro Cirino	FS	71	197	4.59	35.50	16.0	120.0	7.040	4.19	...	Ur
53	John Saint Clair	C	77	302	5.20	27.50	25.0	102.0	7.685	4.63	...	Ur
63	Kurth Connell	OT	76	291	5.49	33.00	27.0	103.0	8.140	4.84	...	Ur
64	Rameel Connor	DE	75	276	5.00	35.50	25.0	121.0	7.650	4.45	...	Ur
65	D.J. Cooper	DT	75	277	5.22	28.00	27.0	108.0	7.410	4.49	...	Ur
66	Ashley Cooper	SS	70	208	4.67	35.25	16.0	117.0	7.000	4.20	...	Ur

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	...	
69	Sedrick Curry	CB	73	197	4.44	39.00	15.0	125.0	6.340	3.94	...	Ur
70	Chris Daniels	WR	75	217	4.74	33.00	14.0	115.0	7.200	4.24	...	Ur
71	Brandon Daniels	WR	70	211	4.54	35.00	14.0	120.0	6.950	4.21	...	Ur
72	Joe Dean Davenport	TE	79	268	5.06	30.00	13.0	108.0	7.330	4.67	...	Ur
73	Darren Davis	RB	68	189	4.81	32.00	13.0	109.0	7.040	4.47	...	Ur
74	Adam Davis	OG	76	309	5.66	26.00	25.0	92.0	8.510	5.00	...	Ur
78	Mike Derouselle	DT	74	314	5.28	27.50	27.0	96.0	7.670	4.80	...	Ur
...	...	...	...	...	...	...	...	...	...	...	...	
6144	Greg Senat	OT	78	302	5.38	26.00	19.0	106.0	7.410	4.71	...	Ur
6147	Nathan Shepherd	DT	77	315	5.09	31.00	27.0	112.0	7.500	4.53	...	Ur
6148	Nic Shimonek	QB	75	220	4.88	28.50	19.0	101.0	7.280	4.32	...	Ur
6149	Andre Smith-02	ILB	72	237	4.63	33.00	19.0	115.0	7.190	4.30	...	Ur
6152	Trequan Smith	WR	74	203	4.49	37.50	12.0	130.0	6.970	4.50	...	Ur
6156	Taylor Stallworth	DT	74	312	5.28	23.50	18.0	101.0	7.950	4.75	...	Ur
6164	Rod Taylor	OT	75	320	5.24	30.50	24.0	99.0	7.860	4.75	...	Ur
6168	Jordan Thomas-01	CB	72	186	4.64	38.00	4.0	124.0	6.280	3.94	...	Ur
6169	Jordan Thomas-02	TE	77	265	4.74	27.00	16.0	111.0	7.500	4.75	...	Ur
6171	Roc Thomas	RB	70	198	4.56	35.50	16.0	125.0	7.060	4.24	...	Ur
6173	Henre Toliver	CB	73	185	4.63	32.50	6.0	117.0	7.010	4.15	...	Ur
6174	Kevin Toliver	CB	75	204	4.49	33.00	10.0	120.0	6.940	4.16	...	Ur
6175	Brett Toth	OT	78	291	5.26	28.00	24.0	102.0	7.860	4.75	...	Ur
6176	Johnny Townsend	P	73	210	4.90	30.50	16.0	114.0	7.280	4.41	...	Ur
6177	Shane Tripucka	P	75	211	4.90	30.50	16.0	114.0	7.280	4.41	...	Ur

	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	...	
6179	Marquez Valdes-Scantling	WR	76	206	4.37	30.50	15.0	124.0	6.950	4.21	...	Ur
6181	Azeem Victor	ILB	73	240	4.72	32.50	19.0	115.0	7.200	4.40	...	Ur
6182	Dmontre Wade	CB	72	200	4.57	35.50	15.0	120.0	6.940	4.40	...	Ur
6184	Trey Walker	S	74	200	4.51	33.00	14.0	119.0	7.000	4.28	...	Ur
6189	Chris Warren	RB	74	247	4.69	33.00	25.0	121.0	6.980	4.18	...	Ur
6195	David Wells	TE	78	256	4.75	33.00	20.0	116.0	7.500	4.59	...	Ur
6197	Karaun White	WR	73	206	4.52	33.50	24.0	118.0	7.160	4.57	...	Ur
6198	Kyzir White	S	74	216	4.52	35.50	21.0	113.0	6.890	4.28	...	Ur
6199	Mike White	QB	77	224	5.09	27.00	19.0	96.0	7.400	4.40	...	Ur
6202	Jake Wieneke	WR	76	221	4.67	34.00	9.0	114.0	7.240	4.37	...	Ur
6207	Eddy Wilson	DT	76	301	5.37	29.50	26.0	96.0	7.695	4.64	...	Ur
6208	Jeff Wilson	RB	71	210	4.56	34.00	15.0	118.0	7.060	4.24	...	Ur
6210	Anthony Winbush	EDGE	73	249	4.76	33.00	25.0	110.0	7.140	4.28	...	Ur
6211	Ryan Winslow	P	77	217	4.90	30.50	16.0	114.0	7.280	4.41	...	Ur
6216	Kenny Young	ILB	73	236	4.60	36.00	23.0	117.0	7.380	4.48	...	Ur

1323 rows × 23 columns

In [16]:
```
# create index
df = df.reset_index()
```

In [17]:
```
# drop Pfr_ID column
df = df.drop('Pfr_ID', axis = 1)
df.head()
```

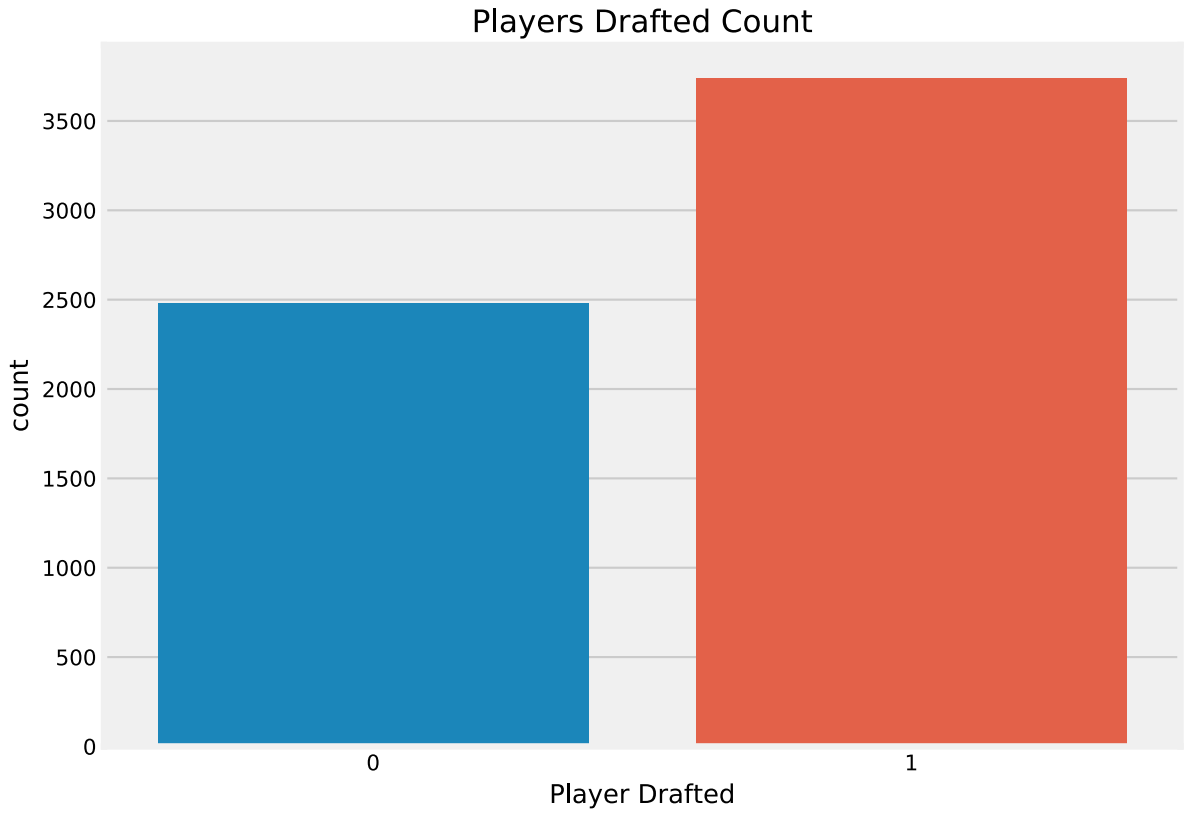
Out[17]:

	index	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	...	Team
0	0	John Abraham	OLB	76	252	4.55	34.5	22.0	117.0	7.11	...	New York Jets
1	1	Shaun Alexander	RB	72	218	4.58	34.0	19.0	118.0	7.06	...	Seattle Seahawks
2	2	Darnell Alford	OT	76	334	5.56	25.0	23.0	94.0	8.48	...	Kansas City Chiefs
3	3	Kyle Allamon	TE	74	253	4.97	29.0	20.0	104.0	7.29	...	Undrafted
4	4	Rashard Anderson	CB	74	206	4.55	34.0	15.0	123.0	7.18	...	Carolina Panthers

5 rows × 23 columns

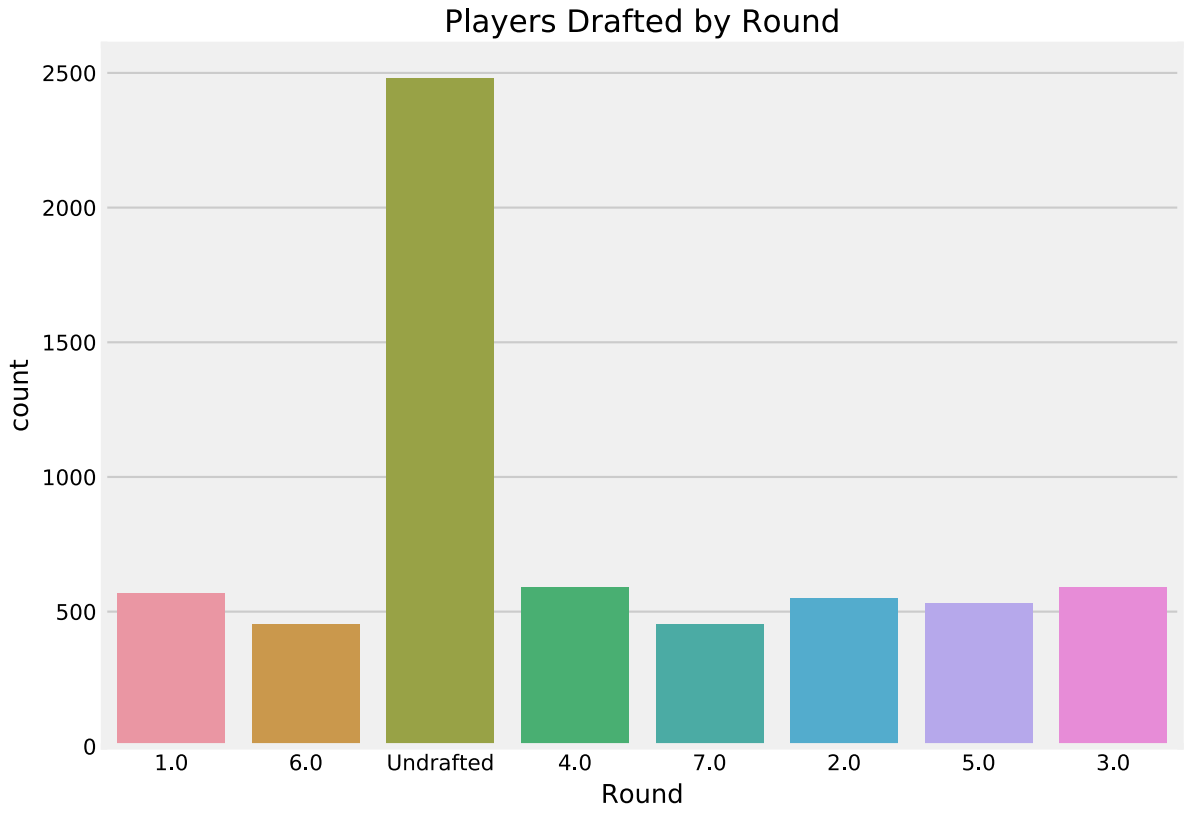
In [18]:
```
## Good news - we managed to keep all of our original data without dropp
ing records

####### EDA #########

# show the number of players drafted vs undrafted
plt.figure(figsize=(8,6))
sns.countplot(x = 'Player Drafted', data = df).set_title('Players Drafte
d Count')
## looks like most players got drafted. A little unbalanced,  but not te
rrible
```

Out[18]:
```
Text(0.5, 1.0, 'Players Drafted Count')
```



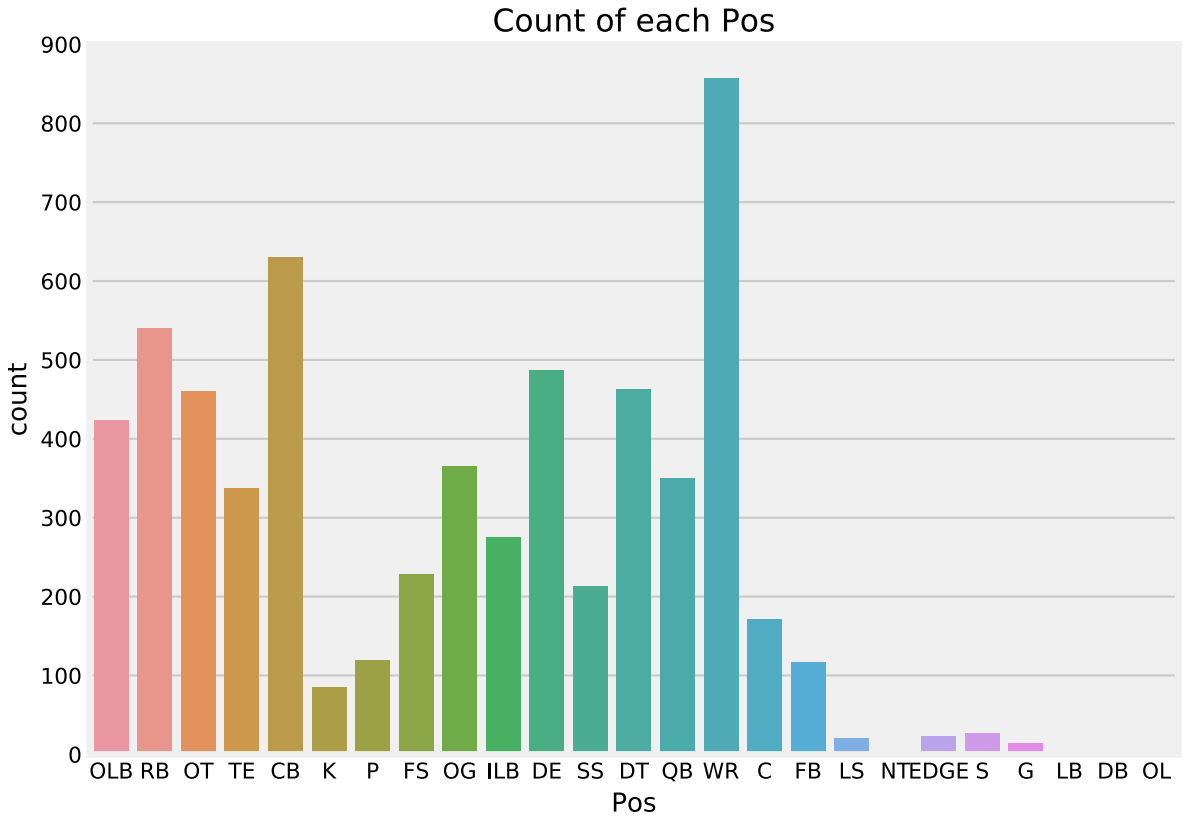
In [19]:
```
# let's look at the number of players drafted by round
plt.figure(figsize=(8,6))
sns.countplot(x = 'Round', data = df).set_title('Players Drafted by Roun
d')
## undrafted is the leader here, which makes sense
## everything else is about even, wiith rounds 3 and 4 being the most co
mmon for drafted players
```

Out[19]:
```
Text(0.5, 1.0, 'Players Drafted by Round')
```



In [20]:
```
# finally, counts by position
plt.figure(figsize=(8,6))
sns.countplot(x = 'Pos', data = df).set_title('Count of each Pos')
## interesting insights here - a lot of CBs and WRs as part of our data
 set, regardless of draft position.
## Let's dig deeper and look at draft rate per position
```

Out[20]:
```
Text(0.5, 1.0, 'Count of each Pos')
```



In [21]:
```
# groupby position and get the count of each position
dfPosCount = df.groupby('Pos').count()
dfPosCount = dfPosCount.reset_index()
dfPosCount = dfPosCount[['Pos', 'Player']]
dfPosCount = dfPosCount.rename(columns = {'Player': 'Position Count'})
dfPosCount
```

Out[21]:

	Pos	Position Count
0	C	171
1	CB	630
2	DB	2
3	DE	487
4	DT	463
5	EDGE	23
6	FB	117
7	FS	229
8	G	14
9	ILB	276
10	K	85
11	LB	3
12	LS	20
13	NT	3
14	OG	365
15	OL	2
16	OLB	424
17	OT	460
18	P	120
19	QB	350
20	RB	540
21	S	27
22	SS	213
23	TE	337
24	WR	857

In [22]:
```
### An argument could be made that we need to consolidate some of these
 positions
### For example, SS and FS are both Safeties, while DB could be a catch
 all for Safeties and CBs
### For the purposes of this analysis, we will not make any changes to t
he positions
### It is possible a more "general" position impacts draft grade as oppo
sed to a more "specific" position
### so let's see how everything plays out with the original data

# groupby position and get the number of drafted players at each positio
n
dfPosDrafted = df.groupby('Pos').sum()['Player Drafted']
dfPosDrafted = dfPosDrafted.reset_index()
dfPosDrafted = dfPosDrafted.rename(columns = {'Player Drafted': 'Players
Drafted'})
dfPosDrafted
```

Out[22]:

	Pos	Players Drafted
0	C	96
1	CB	424
2	DB	0
3	DE	325
4	DT	316
5	EDGE	0
6	FB	66
7	FS	149
8	G	0
9	ILB	171
10	K	25
11	LB	0
12	LS	5
13	NT	0
14	OG	203
15	OL	0
16	OLB	303
17	OT	294
18	P	30
19	QB	194
20	RB	302
21	S	0
22	SS	139
23	TE	205
24	WR	491

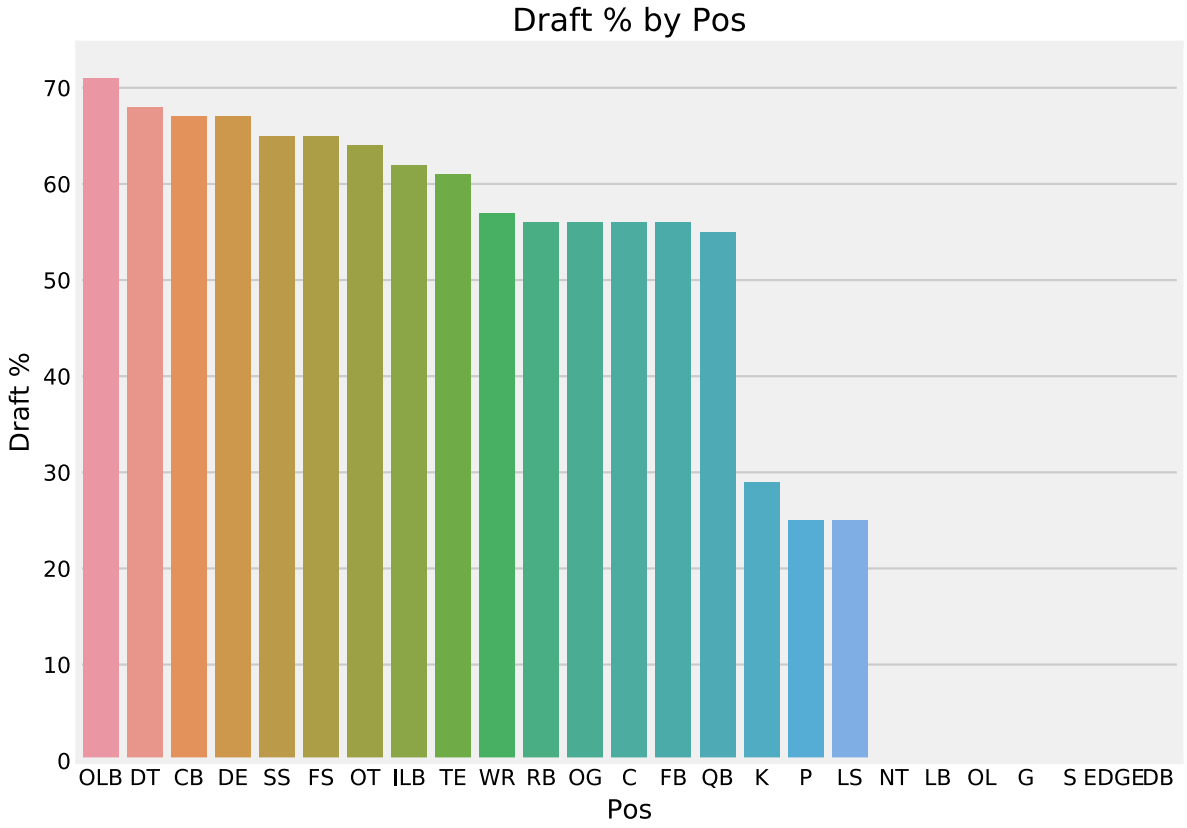
In [23]:
```
dfPos = dfPosCount.merge(dfPosDrafted, on = 'Pos', how = 'left')
dfPos['Draft %'] = dfPos['Players Drafted'] / dfPos['Position Count'] *
100
dfPos['Draft %'] = round(dfPos['Draft %'])
dfPos = dfPos.sort_values(by = 'Draft %', ascending = False)
dfPos
```

Out[23]:

	Pos	Position Count	Players Drafted	Draft %
16	OLB	424	303	71.0
4	DT	463	316	68.0
1	CB	630	424	67.0
3	DE	487	325	67.0
22	SS	213	139	65.0
7	FS	229	149	65.0
17	OT	460	294	64.0
9	ILB	276	171	62.0
23	TE	337	205	61.0
24	WR	857	491	57.0
20	RB	540	302	56.0
14	OG	365	203	56.0
0	C	171	96	56.0
6	FB	117	66	56.0
19	QB	350	194	55.0
10	K	85	25	29.0
18	P	120	30	25.0
12	LS	20	5	25.0
13	NT	3	0	0.0
11	LB	3	0	0.0
15	OL	2	0	0.0
8	G	14	0	0.0
21	S	27	0	0.0
5	EDGE	23	0	0.0
2	DB	2	0	0.0

In [24]:
```
# visualize draft %
plt.figure(figsize=(8,6))
sns.barplot(y = 'Draft %', x = 'Pos', data = dfPos).set_title('Draft % by Pos')

## Looks like OLB had the highest draft % with a few positions having no players drafted at all
```

Out[24]:
```
Text(0.5, 1.0, 'Draft % by Pos')
```



In [25]:
```
#### Model Building ####
## Our goal is going to be to pick whether or not a player is drafted
## Going to remove columns which may mess up our model, such as "Pick" a
nd "Round"
## These columns are being removed because they are not something we cou
ld actually use to predict our target variable
## If we know which pick someone was drafted in, then there would be no
 need to predict the round!

# remove "Pick", "Round", "AV", and "Team" columns
df = df.drop(['Pick', 'Round', 'AV', 'Team'], axis = 1)
df.head()
```

Out[25]:

	index	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year
0	0	John Abraham	OLB	76	252	4.55	34.5	22.0	117.0	7.11	4.29	2000
1	1	Shaun Alexander	RB	72	218	4.58	34.0	19.0	118.0	7.06	4.24	2000
2	2	Darnell Alford	OT	76	334	5.56	25.0	23.0	94.0	8.48	4.98	2000
3	3	Kyle Allamon	TE	74	253	4.97	29.0	20.0	104.0	7.29	4.49	2000
4	4	Rashard Anderson	CB	74	206	4.55	34.0	15.0	123.0	7.18	4.15	2000

In [26]:
```
# remove Median columns as they are no longer needed
dfNew = df.drop(['Median Forty', 'Median Vertical', 'Median BenchReps',
'Median BroadJump',
'Median Cone', 'Median Shuttle'], axis = 1)
dfNew.head()
```

Out[26]:

	index	Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year
0	0	John Abraham	OLB	76	252	4.55	34.5	22.0	117.0	7.11	4.29	2000
1	1	Shaun Alexander	RB	72	218	4.58	34.0	19.0	118.0	7.06	4.24	2000
2	2	Darnell Alford	OT	76	334	5.56	25.0	23.0	94.0	8.48	4.98	2000
3	3	Kyle Allamon	TE	74	253	4.97	29.0	20.0	104.0	7.29	4.49	2000
4	4	Rashard Anderson	CB	74	206	4.55	34.0	15.0	123.0	7.18	4.15	2000

In [27]:
```
# convert year and index to type object
dfNew[['Year', 'index']] = dfNew[['Year', 'index']].astype(object)
dfNew.dtypes
```

Out[27]:
```
index              object
Player             object
Pos                object
Ht                  int64
Wt                  int64
Forty             float64
Vertical          float64
BenchReps         float64
BroadJump         float64
Cone              float64
Shuttle           float64
Year               object
Player Drafted      int64
dtype: object
```

In [28]:
```
### Model 1: Random Forest ###

## split our label and features
labels = np.array(dfNew['Player Drafted'])
labels
```

Out[28]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [29]:
```
# dropping index and Player as well
dfFeatures = dfNew.drop(['index', 'Player', 'Player Drafted', 'Year'], a
xis = 1)
dfFeatures.head()
```

Out[29]:

	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle
0	OLB	76	252	4.55	34.5	22.0	117.0	7.11	4.29
1	RB	72	218	4.58	34.0	19.0	118.0	7.06	4.24
2	OT	76	334	5.56	25.0	23.0	94.0	8.48	4.98
3	TE	74	253	4.97	29.0	20.0	104.0	7.29	4.49
4	CB	74	206	4.55	34.0	15.0	123.0	7.18	4.15

In [30]:
```
# convert Pos to data type category
dfFeatures[['Pos']] = dfFeatures[['Pos']].astype(object)
dfFeatures.dtypes
```

Out[30]:
```
Pos           object
Ht             int64
Wt             int64
Forty        float64
Vertical     float64
BenchReps    float64
BroadJump    float64
Cone         float64
Shuttle      float64
dtype: object
```

In [31]:
```
# One-hot encode our categorical variables
dfDummy = pd.get_dummies(dfFeatures)
dfDummy.head()
```

Out[31]:

	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Pos_C	Pos_CB	...	Pos_OL
0	76	252	4.55	34.5	22.0	117.0	7.11	4.29	0	0	...	0
1	72	218	4.58	34.0	19.0	118.0	7.06	4.24	0	0	...	0
2	76	334	5.56	25.0	23.0	94.0	8.48	4.98	0	0	...	0
3	74	253	4.97	29.0	20.0	104.0	7.29	4.49	0	0	...	0
4	74	206	4.55	34.0	15.0	123.0	7.18	4.15	0	1	...	0

5 rows × 33 columns

In [32]:
```
# view all of our columns
dfDummy.columns
```

Out[32]:
```
Index(['Ht', 'Wt', 'Forty', 'Vertical', 'BenchReps', 'BroadJump', 'Con
e',
       'Shuttle', 'Pos_C', 'Pos_CB', 'Pos_DB', 'Pos_DE', 'Pos_DT', 'Pos
_EDGE',
       'Pos_FB', 'Pos_FS', 'Pos_G', 'Pos_ILB', 'Pos_K', 'Pos_LB', 'Pos_
LS',
       'Pos_NT', 'Pos_OG', 'Pos_OL', 'Pos_OLB', 'Pos_OT', 'Pos_P', 'Pos
_QB',
       'Pos_RB', 'Pos_S', 'Pos_SS', 'Pos_TE', 'Pos_WR'],
      dtype='object')
```

In [33]:
```
# save feature names
featureNames = list(dfDummy.columns)
```

In [34]:
```
# convert features to np array
features = np.array(dfDummy)
features.dtype
```

Out[34]:
```
dtype('float64')
```

In [35]:
```
# split data into training and test sets
trainFeatures, testFeatures, trainLabels, testLabels = train_test_split(
    features, labels, test_size = 0.25, random_state = 42
)
```

In [36]:
```
# define our random forest model
rfModel = RandomForestClassifier(n_estimators = 100, random_state = 42,
max_features = 'sqrt', n_jobs = -1, verbose = 1)
```

In [37]:
```
# train model
rfModel.fit(trainFeatures, trainLabels)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    0.3s finished
```

Out[37]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gi
ni',
            max_depth=None, max_features='sqrt', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
            oob_score=False, random_state=42, verbose=1, warm_start=Fal
se)
```

In [38]:
```
# get predictions with training data
rfPredictionsTrain = rfModel.predict(trainFeatures)
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.1s finished
```

In [39]:
```
# get predictions with test data
rfPredictionsTest = rfModel.predict(testFeatures)
```

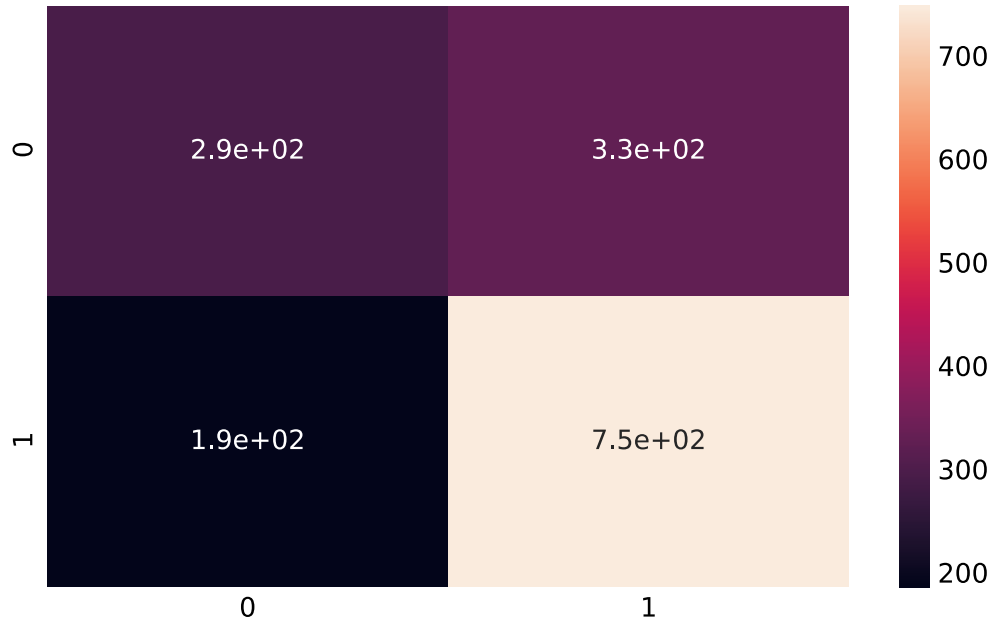
```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent
workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.1s finished
```

In [40]:
```
# create confusion matrix for test data
confMat = confusion_matrix(testLabels, rfPredictionsTest)
print(confMat)
```

```
[[294 327]
 [186 748]]
```

In [41]:
```
sns.heatmap(confMat, annot = True)
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa6de591dd8>
```



In [44]:
```
precision_score(testLabels, rfPredictionsTest)
```

Out[44]:
```
0.695813953488372
```

In [45]:
```
# which features are most important?
featImportance = pd.DataFrame({'Feature': featureNames,
'Importance': rfModel.feature_importances_}).sort_values('Importance', a
scending = False)
featImportance
### importance is greater for the actual drills and not position - inter
esting!
```

Out[45]:

	Feature	Importance
2	Forty	0.175282
1	Wt	0.153659
6	Cone	0.106100
7	Shuttle	0.101561
4	BenchReps	0.101057
3	Vertical	0.098857
5	BroadJump	0.095118
0	Ht	0.082513
32	Pos_WR	0.006764
13	Pos_EDGE	0.006240
29	Pos_S	0.005544
24	Pos_OLB	0.005190
28	Pos_RB	0.005166
11	Pos_DE	0.005021
9	Pos_CB	0.004665
27	Pos_QB	0.004337
31	Pos_TE	0.004192
12	Pos_DT	0.004130
25	Pos_OT	0.003973
17	Pos_ILB	0.003873
15	Pos_FS	0.003718
26	Pos_P	0.003677
30	Pos_SS	0.003248
22	Pos_OG	0.003223
14	Pos_FB	0.003031
16	Pos_G	0.002799
8	Pos_C	0.002742
18	Pos_K	0.001628
20	Pos_LS	0.001049
10	Pos_DB	0.000649
19	Pos_LB	0.000495
21	Pos_NT	0.000348
23	Pos_OL	0.000150

In [46]:
```
## Model 2: Logistic Regression ##

# define logReg model
lrModel = LogisticRegression()
```

In [47]:
```
# fit the model
lrModel.fit(trainFeatures, trainLabels)
```

```
/Users/calebdill/anaconda3/lib/python3.7/site-packages/sklearn/linear_m
odel/logistic.py:433: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[47]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [48]:
```
# get predictions with test data
lrPredictionsTest = lrModel.predict(testFeatures)
```

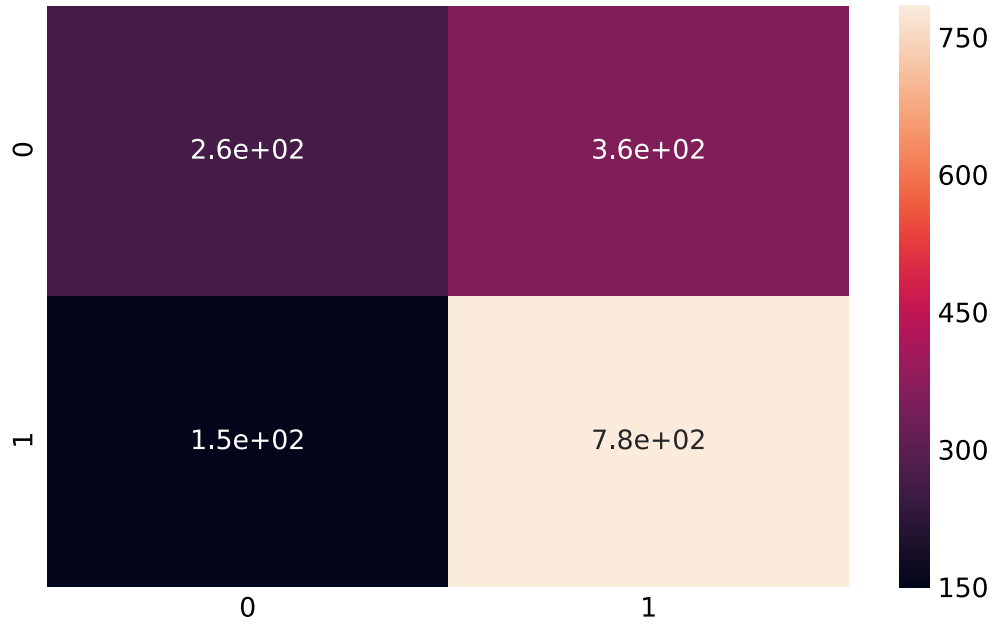
In [49]:
```
# get probabilities with test data
lrProbsTest = lrModel.predict_proba(testFeatures)
```

In [50]:
```
# create confusion matrix for test data
confMat = confusion_matrix(testLabels, lrPredictionsTest)
print(confMat)
```

```
[[265 356]
 [150 784]]
```

In [51]:
```
sns.heatmap(confMat, annot = True)
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa6e0b9ad30>



In [52]:
```
# overall performance
print('Accuracy:', recall_score(testLabels, lrPredictionsTest))
```

```
Accuracy: 0.8394004282655246
```

In [53]:
```
print('Recall:', recall_score(testLabels, lrPredictionsTest))
```

```
Recall: 0.8394004282655246
```

In [54]:
```
print('Precision:', precision_score(testLabels, lrPredictionsTest))
```

```
Precision: 0.6877192982456141
```

In [55]:
```
# feature importance
lrFeatImp = pd.DataFrame({'Feature': featureNames,
'Coefficient': [i for i in lrModel.coef_[0]]}).sort_values('Coefficient'
, ascending = False)
# interesting - a little different than our rf model!
```

In [56]:
```
# visualize coefficients
plt.figure(figsize=(8,6))
sns.barplot(y = 'Feature', x = 'Coefficient', data = lrFeatImp).set_titl
e('Logistic Regression Model Featue Coefficients')
```

Out[56]: Text(0.5, 1.0, 'Logistic Regression Model Featue Coefficients')