

1. Load the data from the “nflstats.csv” file into a DataFrame.

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
from yellowbrick.features import Rank2D
from yellowbrick.style import set_palette
from yellowbrick.features import ParallelCoordinates
import numpy as np

df = pd.read_csv('nflstats.csv')

# Reads in our data.
```

1. Display the dimensions of the file.

```
In [2]: print('Dimensions of file: ', df.shape)

# Prints the dimensions of the data.
```

Dimensions of file: (1578, 33)

1. Display the first 5 rows.

```
In [3]: df.head()

# Head shows the first 5 rows.
```

Out[3]:

	Year	League	TeamName	WonSB	W	L	T	DivPlace	DivMax	DivTotal	...	DefYardsRank
0	2019	NFL	Arizona Cardinals	0	5	10	1	4	4	4th of 4	...	32
1	2018	NFL	Arizona Cardinals	0	3	13	0	4	4	4th of 4	...	20
2	2017	NFL	Arizona Cardinals	0	8	8	0	3	4	3rd of 4	...	6
3	2016	NFL	Arizona Cardinals	0	7	8	1	2	4	2nd of 4	...	2
4	2015	NFL	Arizona Cardinals	0	13	3	0	1	4	1st of 4	...	5

5 rows × 33 columns



1. Look at summary information about your data (total, mean, min, max, freq, unique, etc.) Does this present any more questions for you? Does it lead you to a conclusion yet?

```
In [4]: print('Describe data:\n',df.describe())

print('Summary: \n',df.describe(include = ['O']))

# This lets us take a look at our categorical and numerical variables and their stats.
```

Describe data:

	Year	WonSB	W	L	T \
count	1578.000000	1578.000000	1578.000000	1578.000000	1578.000000
mean	1993.712928	0.033587	7.675539	7.675539	0.110266
std	15.502264	0.180220	3.079982	3.074418	0.369075
min	1966.000000	0.000000	0.000000	0.000000	0.000000
25%	1980.000000	0.000000	5.000000	5.000000	0.000000
50%	1994.500000	0.000000	8.000000	8.000000	0.000000
75%	2007.000000	0.000000	10.000000	10.000000	0.000000
max	2019.000000	1.000000	16.000000	16.000000	3.000000

	DivPlace	DivMax	PlayoffsResultNumerical	PointsFor \
count	1578.000000	1578.000000	1578.000000	1578.000000
mean	2.731939	4.477820	0.857414	323.847275
std	1.304985	0.600002	1.364213	74.217167
min	1.000000	4.000000	0.000000	103.000000
25%	2.000000	4.000000	0.000000	275.000000
50%	3.000000	4.000000	0.000000	321.000000
75%	4.000000	5.000000	2.000000	372.000000
max	8.000000	8.000000	5.000000	606.000000

	PointsAllowed	...	DefYardsRank	T/G	PointsRank \
count	1578.000000	...	1578.000000	1578.000000	1578.000000
mean	323.847275	...	14.821293	14.453739	14.791508
std	68.443814	...	8.679220	8.647435	8.683339
min	128.000000	...	1.000000	1.000000	1.000000
25%	281.000000	...	7.000000	7.000000	7.000000
50%	325.000000	...	14.000000	14.000000	14.000000
75%	371.000000	...	22.000000	22.000000	22.000000
max	533.000000	...	32.000000	32.000000	32.000000

	YardsRank	MaxTeams	MoV	SoS	SRS \
count	1578.000000	1578.000000	1578.000000	1578.000000	1578.000000
mean	14.825095	28.657795	0.000127	0.000127	0.000253
std	8.680247	4.567482	6.547290	1.610161	6.281144
min	1.000000	9.000000	-20.500000	-6.300000	-19.700000
25%	7.000000	28.000000	-4.400000	-1.100000	-4.200000
50%	14.000000	29.000000	0.000000	0.000000	-0.100000
75%	22.000000	32.000000	4.600000	1.100000	4.400000
max	32.000000	32.000000	19.700000	5.100000	20.100000

	OSRS	DSRS
count	1578.000000	1578.000000
mean	0.000380	-0.000824
std	4.139455	3.722509
min	-12.300000	-13.700000
25%	-2.900000	-2.500000
50%	-0.100000	0.150000
75%	2.700000	2.500000
max	15.900000	10.600000

[8 rows x 24 columns]

Summary:

	League	TeamName	DivTotal	PlayoffsResult	Coaches	AV \
count	1578	1578	1578	1578	1578	1578
unique	2	41	28	6	323	606
top	NFL	Dallas Cowboys	2nd of 4	Miss	Shula	Williams

freq	1540	54	222	1013	34	29
------	------	----	-----	------	----	----

	Passer	Rusher	Receiver
count	1578	1578	1578
unique	314	398	419
top	Manning	Smith	Johnson
freq	40	44	47

1. Make some histograms of your data (“A picture is worth a thousand words!”)

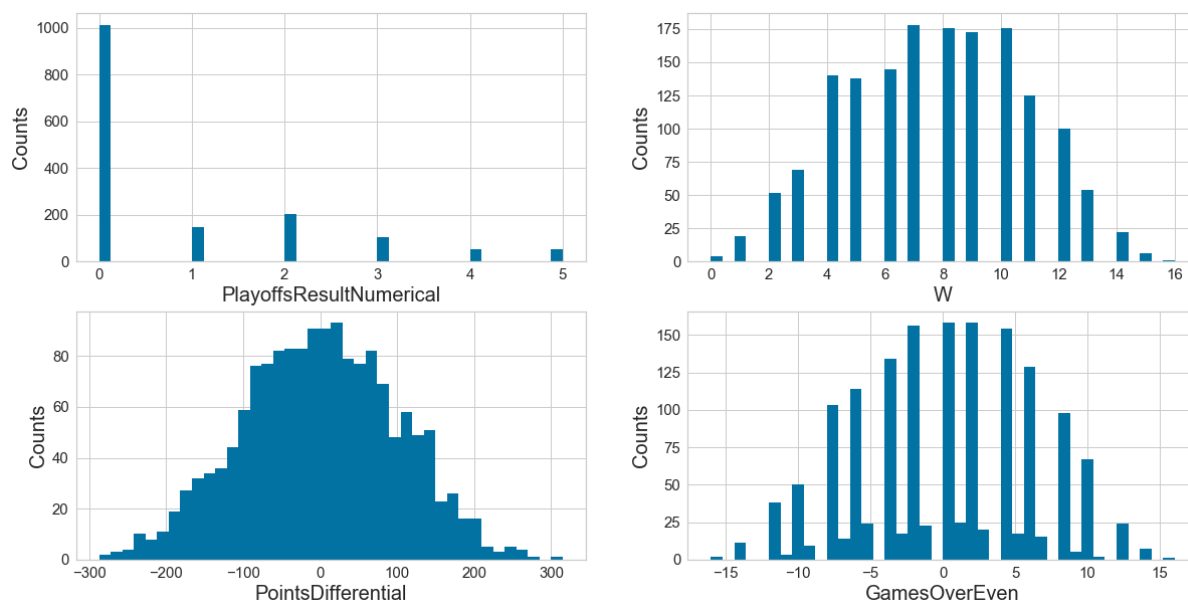
```
In [5]: plt.rcParams['figure.figsize'] = (20, 10)
fig, axes = plt.subplots(nrows = 2, ncols = 2)

df['GamesOverEven'] = df['W'] - df['L']

# Creates a new column for their win differential.

num_features = ['PlayoffsResultNumerical', 'W', 'PointsDifferential', 'GamesOverEven']
xaxes = num_features
yaxes = ['Counts', 'Counts', 'Counts', 'Counts']

axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(df[num_features[idx]].dropna(), bins=40)
    ax.set_xlabel(xaxes[idx], fontsize=20)
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)
```



1. Make some bar charts for variables with only a few options.

```

In [6]: plt.rcParams['figure.figsize'] = (20, 10)

# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 1)

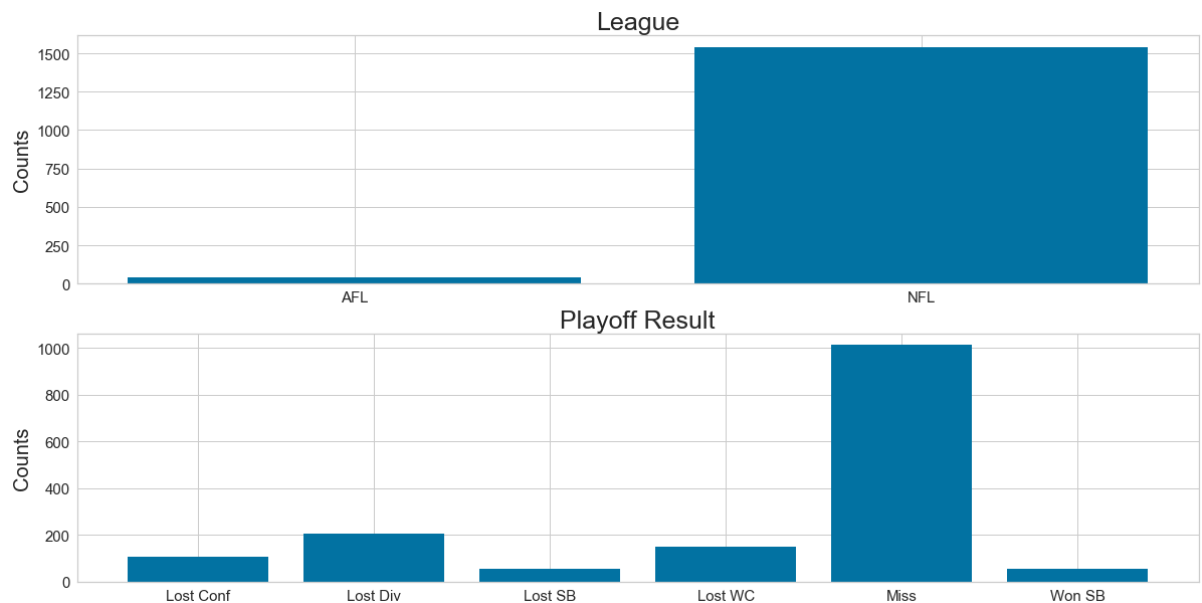
# make the data read to feed into the visulizer
X_League = df.replace({'League': {1: 'NFL', 0: 'AFL'}}).groupby('League').size()
().reset_index(name='Counts')['League']
Y_League = df.replace({'League': {1: 'NFL', 0: 'AFL'}}).groupby('League').size()
().reset_index(name='Counts')['Counts']

# make the bar plot
axes[0].bar(X_League, Y_League)
axes[0].set_title('League', fontsize=25)
axes[0].set_ylabel('Counts', fontsize=20)
axes[0].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
X_Playoff = df.replace({'PlayoffsResult': {0: 'Miss', 1: 'Lost WC', 2: 'Lost D
iv', 3: 'Lost Conf', 4: 'Lost SB', 5: 'Won SB'}}).groupby('PlayoffsResult').si
ze().reset_index(name='Counts')['PlayoffsResult']
Y_Playoff = df.replace({'PlayoffsResult': {0: 'Miss', 1: 'Lost WC', 2: 'Lost D
iv', 3: 'Lost Conf', 4: 'Lost SB', 5: 'Won SB'}}).groupby('PlayoffsResult').si
ze().reset_index(name='Counts')['Counts']

# make the bar plot
axes[1].bar(X_Playoff, Y_Playoff)
axes[1].set_title('Playoff Result', fontsize=25)
axes[1].set_ylabel('Counts', fontsize=20)
axes[1].tick_params(axis='both', labelsize=15)

```



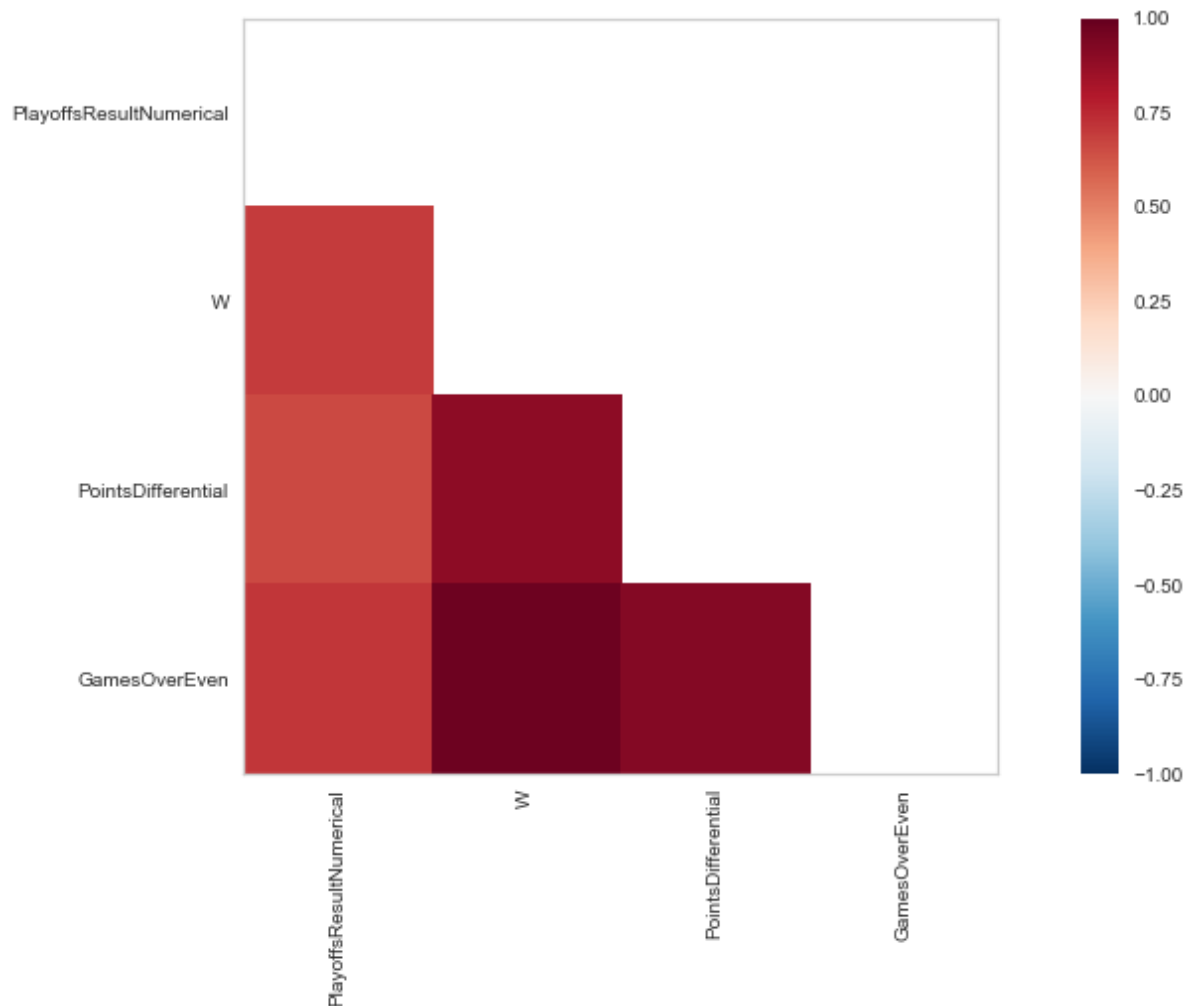
1. To see if the data is correlated, make some Pearson Ranking charts

```
In [7]: plt.rcParams['figure.figsize'] = (15, 7)

X = df[num_features].values

visualizer = Rank2D(features=num_features, algorithm='pearson')
visualizer.fit(X)           # Fit the data to the visualizer
visualizer.transform(X)     # Transform the data
```

```
Out[7]: array([[ 0,  5, -81, -5],
 [ 0,  3, -200, -10],
 [ 0,  8, -66,  0],
 ...,
 [ 0,  5, -109, -4],
 [ 0,  5,  -6, -1],
 [ 0,  7,  -4,  0]], dtype=int64)
```



1. Use Parallel Coordinates visualization to compare the distributions of numerical variables between team records and playoff success.

```

In [8]: plt.rcParams['figure.figsize'] = (15, 7)
plt.rcParams['font.size'] = 50

# setup the color for yellowbrick visualizer

set_palette('sns_bright')

classes = ['Missed Playoffs', 'Lost WC', 'Lost Divisional', 'Lost Conference',
'Lost SB', 'Won SB']

# copy data to a new dataframe
df_norm = df.copy()
# normalize data to 0-1 range
for feature in num_features:
    df_norm[feature] = (df[feature] - df[feature].mean(skipna=True)) / (df[feature].max(skipna=True) - df[feature].min(skipna=True))

X = df_norm[num_features].values
y = df.PlayoffsResultNumerical.values

visualizer = ParallelCoordinates(classes=classes, features=num_features)

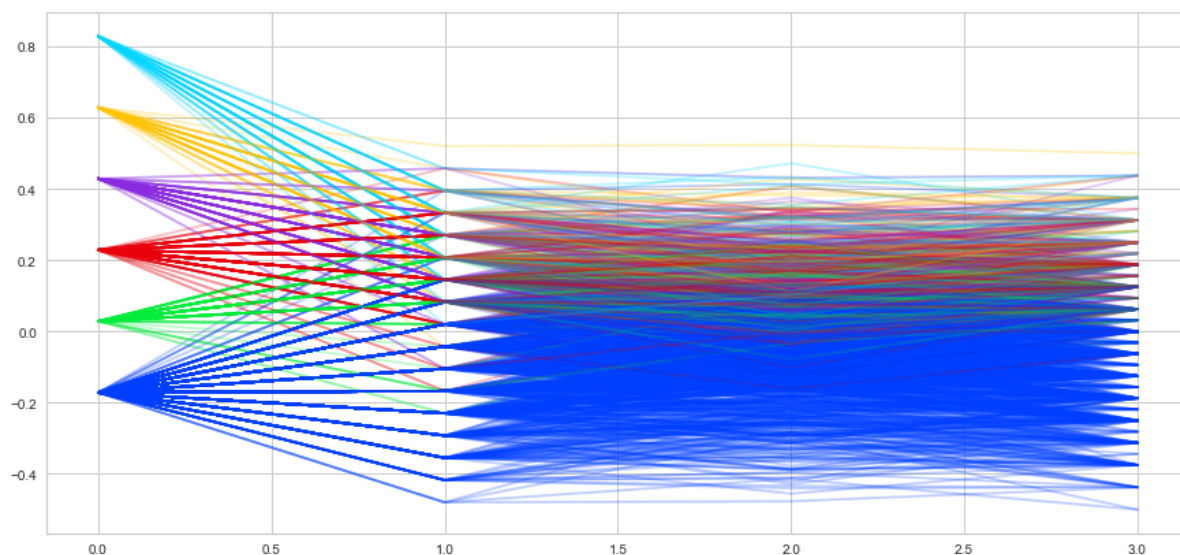
visualizer.fit(X, y) # Fit the data to the visualizer
visualizer.transform(X) # Transform the data

```

```

Out[8]: array([[ -0.17148289, -0.16722117, -0.1345515 , -0.15625   ],
               [ -0.17148289, -0.29222117, -0.33222591, -0.3125   ],
               [ -0.17148289,  0.02027883, -0.10963455,  0.         ],
               ...,
               [ -0.17148289, -0.16722117, -0.18106312, -0.125     ],
               [ -0.17148289, -0.16722117, -0.00996678, -0.03125   ],
               [ -0.17148289, -0.04222117, -0.00664452,  0.         ]])

```



1. Use Stack Bar Charts to compare teams who made it far in the playoffs to teams who didn't make it based on the other variables.

```

In [9]: plt.rcParams['figure.figsize'] = (20, 10)

fig, axes = plt.subplots(nrows = 1, ncols = 2)

# make the data read to feed into the visulizer
W_MadePlayoffs = df.replace({'WonSB': {1: 'Yes', 0: 'No'}})[df['WonSB']==1][
'W'].value_counts()
W_NoPlayoffs = df.replace({'WonSB': {1: 'Yes', 0: 'No'}})[df['WonSB']==0]['W']
.value_counts()
W_NoPlayoffs = W_NoPlayoffs.reindex(index = W_MadePlayoffs.index)
# make the bar plot

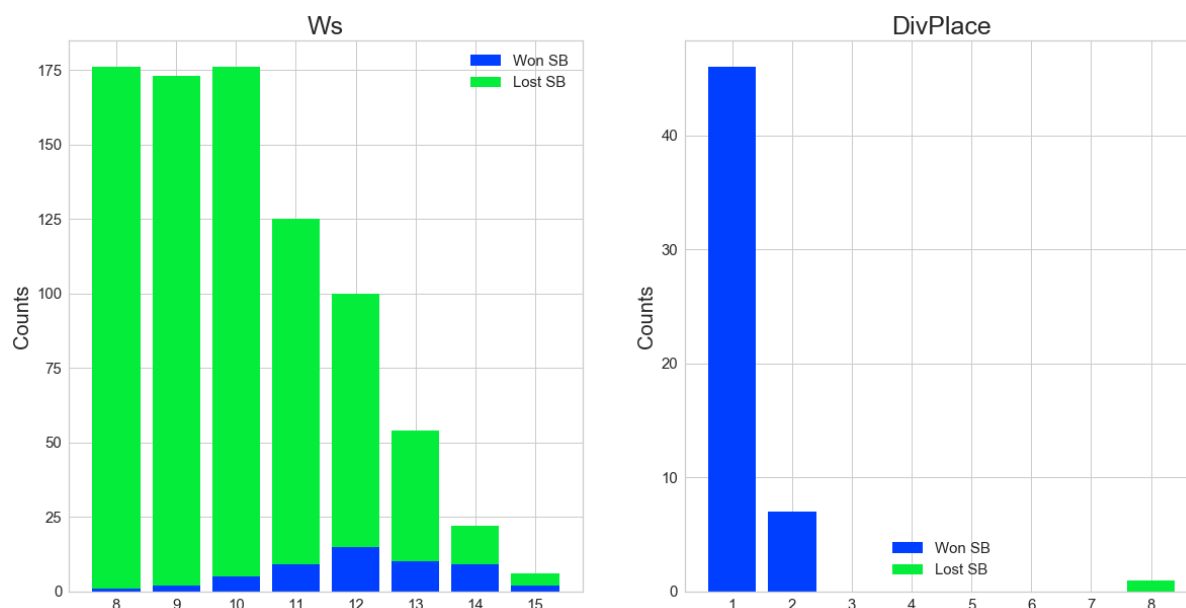
p1 = axes[0].bar(W_MadePlayoffs.index, W_MadePlayoffs.values)
p2 = axes[0].bar(W_NoPlayoffs.index, W_NoPlayoffs.values, bottom=W_MadePlayoff
s.values)
axes[0].set_title('Ws', fontsize=25)
axes[0].set_ylabel('Counts', fontsize=20)
axes[0].tick_params(axis='both', labelsize=15)
axes[0].legend((p1[0], p2[0]), ('Won SB', 'Lost SB'), fontsize = 15)

# make the data read to feed into the visulizer
DivPlace_MadePlayoffs= df.replace('WonSB')[df['WonSB']==1]['DivPlace'].value_c
ounts()
DivPlace_NoPlayoffs = df.replace('WonSB')[df['WonSB']==0]['DivPlace'].value_co
unts()
DivPlace_NoPlayoffs = DivPlace_NoPlayoffs.reindex(index = W_MadePlayoffs.index
)
# make the bar plot

p3 = axes[1].bar(DivPlace_MadePlayoffs.index, DivPlace_MadePlayoffs.values)
p4 = axes[1].bar(DivPlace_NoPlayoffs.index, DivPlace_NoPlayoffs.values)
axes[1].set_title('DivPlace', fontsize=25)
axes[1].set_ylabel('Counts', fontsize=20)
axes[1].tick_params(axis='both', labelsize=15)
axes[1].legend((p3[0], p4[0]), ('Won SB', 'Lost SB'), fontsize = 15)

```


Out[9]: <matplotlib.legend.Legend at 0x28dda53b3c8>



- Now it's time to reduce some of the features so we can concentrate on the things that matter! There features we will get rid of are: "DivTotal" (it's redundant), "Coaches", "AV", "Passer", "Rusher", "Receiver". (Names don't really tell us performance and there are hundreds of unique values.
 - We can also fill in missing values if there were any. In this case, we don't, but it would have been something to try. Maybe we can fill in the NAs for playoff results if we hadn't already.

```
In [10]: def fill_na_median(data, inplace=True):
          return data.fillna(data.median(), inplace=inplace)
          # This function fills NAs with the median.

          # fill with the most represented value
          def fill_na_most(data, inplace=True):
              return data.fillna('S', inplace=inplace)
          # This one fills it with the most representative value.
```

- If you go back and look at the histograms of MadePlayoffs, you'll see that it is very skewed. Most teams don't make the playoffs or even make it very far. Let's try a Log Transformation: it is a good method to use on highly skewed data.

```
In [11]: # Log-transformation
def log_transformation(data):
    return data.apply(np.log1p)

df['PlayoffResults_log1p'] = log_transformation(df['PlayoffsResultNumerical'])

# check the data
print(df.describe())
```

	Year	WonSB	W	L	T \
count	1578.000000	1578.000000	1578.000000	1578.000000	1578.000000
mean	1993.712928	0.033587	7.675539	7.675539	0.110266
std	15.502264	0.180220	3.079982	3.074418	0.369075
min	1966.000000	0.000000	0.000000	0.000000	0.000000
25%	1980.000000	0.000000	5.000000	5.000000	0.000000
50%	1994.500000	0.000000	8.000000	8.000000	0.000000
75%	2007.000000	0.000000	10.000000	10.000000	0.000000
max	2019.000000	1.000000	16.000000	16.000000	3.000000

	DivPlace	DivMax	PlayoffsResultNumerical	PointsFor \
count	1578.000000	1578.000000	1578.000000	1578.000000
mean	2.731939	4.477820	0.857414	323.847275
std	1.304985	0.600002	1.364213	74.217167
min	1.000000	4.000000	0.000000	103.000000
25%	2.000000	4.000000	0.000000	275.000000
50%	3.000000	4.000000	0.000000	321.000000
75%	4.000000	5.000000	2.000000	372.000000
max	8.000000	8.000000	5.000000	606.000000

	PointsAllowed	...	PointsRank	YardsRank	MaxTeams	MoV
count	1578.000000	...	1578.000000	1578.000000	1578.000000	1578.000000
mean	323.847275	...	14.791508	14.825095	28.657795	0.000127
std	68.443814	...	8.683339	8.680247	4.567482	6.547290
min	128.000000	...	1.000000	1.000000	9.000000	-20.500000
25%	281.000000	...	7.000000	7.000000	28.000000	-4.400000
50%	325.000000	...	14.000000	14.000000	29.000000	0.000000
75%	371.000000	...	22.000000	22.000000	32.000000	4.600000
max	533.000000	...	32.000000	32.000000	32.000000	19.700000

	SoS	SRS	OSRS	DSRS	GamesOverEven \
count	1578.000000	1578.000000	1578.000000	1578.000000	1578.000000
mean	0.000127	0.000253	0.000380	-0.000824	0.000000
std	1.610161	6.281144	4.139455	3.722509	6.010243
min	-6.300000	-19.700000	-12.300000	-13.700000	-16.000000
25%	-1.100000	-4.200000	-2.900000	-2.500000	-4.000000
50%	0.000000	-0.100000	-0.100000	0.150000	0.000000
75%	1.100000	4.400000	2.700000	2.500000	4.000000
max	5.100000	20.100000	15.900000	10.600000	16.000000

	PlayoffResults_log1p
count	1578.000000
mean	0.415090
std	0.594730
min	0.000000
25%	0.000000
50%	0.000000
75%	1.098612
max	1.791759

[8 rows x 26 columns]



1. Convert your categorical data into numbers (TeamName, Playoffs Result)

```
In [17]: #get the categorical data
cat_features = ['TeamName']
print(cat_features)
df_cat = df[cat_features]
df_cat = df_cat.replace({'TeamName': {1: 'Arizona Cardinals', 1: 'Phoenix Cardinals', 1: 'St. Louis Cardinals', 2: 'Atlanta Falcons', 3: 'Baltimore Ravens', 4: 'Buffalo Bills', 5: 'Carolina Panthers', 6: 'Chicago Bears', 7: 'Cincinnati Bengals', 8: 'Cleveland Browns', 9: 'Dallas Cowboys', 10: 'Denver Broncos', 11: 'Detroit Lions', 12: 'Green Bay Packers', 13: 'Houston Texans', 14: 'Indianapolis Colts', 14: 'Baltimore Colts', 15: 'Jacksonville Jaguars', 16: 'Kansas City Chiefs', 17: 'Los Angeles Chargers', 17: 'San Diego Chargers', 18: 'Los Angeles Rams', 18: 'St. Louis Rams', 19: 'Miami Dolphins', 20: 'Minnesota Vikings', 21: 'New England Patriots', 21: 'Boston Patriots', 22: 'New Orleans Saints', 23: 'New York Giants', 24: 'New York Jets', 25: 'Oakland Raiders', 25: 'Los Angeles Raiders', 26: 'Philadelphia Eagles', 27: 'Pittsburgh Steelers', 28: 'San Francisco 49ers', 29: 'Seattle Seahawks', 30: 'Tampa Bay Buccaneers', 31: 'Tennessee Titans', 31: 'Tennessee Oilers', 31: 'Houston Oilers', 32: 'Washington Redskins'}})
# One Hot Encoding
df_cat_dummies = pd.get_dummies(df_cat)
# check the data
print(df_cat_dummies)
```

```

['TeamName']
TeamName_Arizona Cardinals TeamName_Atlanta Falcons \
0 1 0
1 1 0
2 1 0
3 1 0
4 1 0
5 1 0
6 1 0
7 1 0
8 1 0
9 1 0
10 1 0
11 1 0
12 1 0
13 1 0
14 1 0
15 1 0
16 1 0
17 1 0
18 1 0
19 1 0
20 1 0
21 1 0
22 1 0
23 1 0
24 1 0
25 1 0
26 0 0
27 0 0
28 0 0
29 0 0
... ..
1548 0 0
1549 0 0
1550 0 0
1551 0 0
1552 0 0
1553 0 0
1554 0 0
1555 0 0
1556 0 0
1557 0 0
1558 0 0
1559 0 0
1560 0 0
1561 0 0
1562 0 0
1563 0 0
1564 0 0
1565 0 0
1566 0 0
1567 0 0
1568 0 0
1569 0 0
1570 0 0
1571 0 0

```

1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_Baltimore Colts	TeamName_Baltimore Ravens \
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	0
1549	0	0
1550	0	0
1551	0	0
1552	0	0
1553	0	0
1554	0	0
1555	0	0
1556	0	0
1557	0	0
1558	0	0
1559	0	0
1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0

1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_Boston Patriots	TeamName_Buffalo Bills	\
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	0	0	
9	0	0	
10	0	0	
11	0	0	
12	0	0	
13	0	0	
14	0	0	
15	0	0	
16	0	0	
17	0	0	
18	0	0	
19	0	0	
20	0	0	
21	0	0	
22	0	0	
23	0	0	
24	0	0	
25	0	0	
26	0	0	
27	0	0	
28	0	0	
29	0	0	
...	
1548	0	0	
1549	0	0	
1550	0	0	
1551	0	0	
1552	0	0	
1553	0	0	
1554	0	0	
1555	0	0	
1556	0	0	
1557	0	0	
1558	0	0	
1559	0	0	

1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0
1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_Carolina Panthers	TeamName_Chicago Bears \
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	0
1549	0	0
1550	0	0
1551	0	0
1552	0	0
1553	0	0

1554	0	0
1555	0	0
1556	0	0
1557	0	0
1558	0	0
1559	0	0
1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0
1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_Cincinnati Bengals	TeamName_Cleveland Browns	...	\
0	0	0	...	
1	0	0	...	
2	0	0	...	
3	0	0	...	
4	0	0	...	
5	0	0	...	
6	0	0	...	
7	0	0	...	
8	0	0	...	
9	0	0	...	
10	0	0	...	
11	0	0	...	
12	0	0	...	
13	0	0	...	
14	0	0	...	
15	0	0	...	
16	0	0	...	
17	0	0	...	
18	0	0	...	
19	0	0	...	
20	0	0	...	
21	0	0	...	
22	0	0	...	
23	0	0	...	
24	0	0	...	
25	0	0	...	
26	0	0	...	
27	0	0	...	
28	0	0	...	
29	0	0	...	
...	

1548	0	0	...
1549	0	0	...
1550	0	0	...
1551	0	0	...
1552	0	0	...
1553	0	0	...
1554	0	0	...
1555	0	0	...
1556	0	0	...
1557	0	0	...
1558	0	0	...
1559	0	0	...
1560	0	0	...
1561	0	0	...
1562	0	0	...
1563	0	0	...
1564	0	0	...
1565	0	0	...
1566	0	0	...
1567	0	0	...
1568	0	0	...
1569	0	0	...
1570	0	0	...
1571	0	0	...
1572	0	0	...
1573	0	0	...
1574	0	0	...
1575	0	0	...
1576	0	0	...
1577	0	0	...

	TeamName_Pittsburgh Steelers	TeamName_San Diego Chargers	\
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	0	0	
9	0	0	
10	0	0	
11	0	0	
12	0	0	
13	0	0	
14	0	0	
15	0	0	
16	0	0	
17	0	0	
18	0	0	
19	0	0	
20	0	0	
21	0	0	
22	0	0	
23	0	0	
24	0	0	

25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	0
1549	0	0
1550	0	0
1551	0	0
1552	0	0
1553	0	0
1554	0	0
1555	0	0
1556	0	0
1557	0	0
1558	0	0
1559	0	0
1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0
1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_San Francisco 49ers	TeamName_Seattle Seahawks \
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0

19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	0
1549	0	0
1550	0	0
1551	0	0
1552	0	0
1553	0	0
1554	0	0
1555	0	0
1556	0	0
1557	0	0
1558	0	0
1559	0	0
1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0
1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_St. Louis Cardinals	TeamName_St. Louis Rams \
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0

13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	0
1549	0	0
1550	0	0
1551	0	0
1552	0	0
1553	0	0
1554	0	0
1555	0	0
1556	0	0
1557	0	0
1558	0	0
1559	0	0
1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0
1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0

	TeamName_Tampa Bay Buccaneers	TeamName_Tennessee Oilers \
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0

7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	0
1549	0	0
1550	0	0
1551	0	0
1552	0	0
1553	0	0
1554	0	0
1555	0	0
1556	0	0
1557	0	0
1558	0	0
1559	0	0
1560	0	0
1561	0	0
1562	0	0
1563	0	0
1564	0	0
1565	0	0
1566	0	0
1567	0	0
1568	0	0
1569	0	0
1570	0	0
1571	0	0
1572	0	0
1573	0	0
1574	0	0
1575	0	0
1576	0	0
1577	0	0
TeamName_Tennessee Titans TeamName_Washington Redskins		
0	0	0

1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
...
1548	0	1
1549	0	1
1550	0	1
1551	0	1
1552	0	1
1553	0	1
1554	0	1
1555	0	1
1556	0	1
1557	0	1
1558	0	1
1559	0	1
1560	0	1
1561	0	1
1562	0	1
1563	0	1
1564	0	1
1565	0	1
1566	0	1
1567	0	1
1568	0	1
1569	0	1
1570	0	1
1571	0	1
1572	0	1
1573	0	1
1574	0	1

1575	0	1
1576	0	1
1577	0	1

[1578 rows x 41 columns]

1. Training - Split your data into two sets: Training and Testing.

```
In [18]: # here we will combine the numerical features and the dummie features together
features_model = ['W', 'PointsDifferential']
df_model_X = pd.concat([df[features_model], df_cat_dummies], axis=1)

# create a whole target dataset that can be used for train and validation data
splitting
df_model_y = df.replace({'WonSB': {0: 'No', 1: 'Yes'}})['WonSB']
# separate data into training and validation and check the details of the data
sets
# import packages
from sklearn.model_selection import train_test_split

# split the data
X_train, X_val, y_train, y_val = train_test_split(df_model_X, df_model_y, test
_size =0.3, random_state=15)
```

```
In [19]: print("No. of samples in training set: ", X_train.shape[0])
print("No. of samples in validation set:", X_val.shape[0])

# Playoff results.
print('\n')
print('Playoff results in the training set:')
print(y_train.value_counts())

print('\n')
print('Playoff results in the validation set:')
print(y_val.value_counts())
```

No. of samples in training set: 1104

No. of samples in validation set: 474

Playoff results in the training set:

No 1067

Yes 37

Name: WonSB, dtype: int64

Playoff results in the validation set:

No 458

Yes 16

Name: WonSB, dtype: int64

Evaluation – We are trying to predict if a team will win the Super Bowl. We will start with linear regression.

Metrics for the evaluation:

- Confusion Matrix (you should get 84% - pretty good) (84.3% yeah, pretty good)
- Precision, Recall & F1 score (all 3 were very good) (I can see that)
- ROC curve (the dotted line is the randomly guessed so anything above that is good metric) (Way above that line.)

```
In [20]: print(X_train)
```

	W	PointsDifferential	TeamName_Arizona Cardinals \
405	6	81	0
1258	10	147	0
1567	10	74	0
412	10	82	0
147	9	111	0
723	2	-124	0
669	8	97	0
1344	7	47	0
274	4	-73	0
884	11	118	0
1466	10	36	0
1129	9	23	0
1165	6	10	0
32	7	-6	0
827	6	-48	0
336	3	-258	0
296	12	119	0
1092	10	116	0
15	6	-38	1
58	8	-6	0
36	8	-54	0
1577	7	-4	0
1514	10	67	0
159	11	98	0
434	7	-89	0
1375	8	104	0
479	9	-12	0
1058	4	-110	0
185	9	103	0
739	4	-78	0
...
824	10	30	0
1039	5	-107	0
102	7	-3	0
1359	10	75	0
1181	5	-122	0
956	12	155	0
717	6	19	0
812	6	-106	0
19	3	-233	1
1063	6	-20	0
873	8	-48	0
196	8	7	0
143	7	-6	0
778	6	-37	0
318	7	-33	0
416	11	184	0
1047	3	-104	0
927	10	76	0
749	9	106	0
17	5	-155	1
221	7	-48	0
943	12	128	0
630	1	-238	0
85	5	-71	0
1223	7	36	0

667	5	-65	0
156	10	15	0
384	10	29	0
645	11	171	0
1480	8	-48	0
TeamName_Atlanta Falcons TeamName_Baltimore Colts \			
405	0	0	0
1258	0	0	0
1567	0	0	0
412	0	0	0
147	0	0	0
723	0	0	0
669	0	0	0
1344	0	0	0
274	0	0	0
884	0	0	0
1466	0	0	0
1129	0	0	0
1165	0	0	0
32	0	0	0
827	0	0	0
336	0	0	0
296	0	0	0
1092	0	0	0
15	0	0	0
58	1	0	0
36	0	0	0
1577	0	0	0
1514	0	0	0
159	0	0	0
434	0	0	0
1375	0	0	0
479	0	0	0
1058	0	0	0
185	0	0	0
739	0	0	0
...
824	0	0	0
1039	0	0	0
102	1	0	0
1359	0	0	0
1181	0	0	0
956	0	0	0
717	0	0	0
812	0	0	0
19	0	0	0
1063	0	0	0
873	0	0	0
196	0	0	0
143	0	0	0
778	0	0	0
318	0	0	0
416	0	0	0
1047	0	0	0
927	0	0	0
749	0	0	0

17	0	0
221	0	0
943	0	0
630	0	0
85	1	0
1223	0	0
667	0	0
156	0	0
384	0	0
645	0	1
1480	0	0

	TeamName_Baltimore Ravens	TeamName_Boston Patriots \
405	0	0
1258	0	0
1567	0	0
412	0	0
147	0	0
723	0	0
669	0	0
1344	0	0
274	0	0
884	0	0
1466	0	0
1129	0	0
1165	0	0
32	0	0
827	0	0
336	0	0
296	0	0
1092	0	0
15	0	0
58	0	0
36	0	0
1577	0	0
1514	0	0
159	0	0
434	0	0
1375	0	0
479	0	0
1058	0	0
185	0	0
739	0	0
...
824	0	0
1039	0	0
102	0	0
1359	0	0
1181	0	0
956	0	0
717	0	0
812	0	0
19	0	0
1063	0	0
873	0	0
196	0	0
143	0	0

778	0	0
318	0	0
416	0	0
1047	0	0
927	0	0
749	0	0
17	0	0
221	0	0
943	0	0
630	0	0
85	0	0
1223	0	0
667	0	0
156	0	0
384	0	0
645	0	0
1480	0	0

	TeamName_Buffalo Bills	TeamName_Carolina Panthers \
405	0	0
1258	0	0
1567	0	0
412	0	0
147	1	0
723	0	0
669	0	0
1344	0	0
274	0	0
884	0	0
1466	0	0
1129	0	0
1165	0	0
32	0	0
827	0	0
336	0	0
296	0	0
1092	0	0
15	0	0
58	0	0
36	0	0
1577	0	0
1514	0	0
159	1	0
434	0	0
1375	0	0
479	0	0
1058	0	0
185	1	0
739	0	0
...
824	0	0
1039	0	0
102	0	0
1359	0	0
1181	0	0
956	0	0
717	0	0

812	0	0
19	0	0
1063	0	0
873	0	0
196	0	1
143	1	0
778	0	0
318	0	0
416	0	0
1047	0	0
927	0	0
749	0	0
17	0	0
221	0	0
943	0	0
630	0	0
85	0	0
1223	0	0
667	0	0
156	1	0
384	0	0
645	0	0
1480	0	0

	TeamName_Chicago Bears	...	TeamName_Pittsburgh Steelers	\
405	0	...	0	
1258	0	...	0	
1567	0	...	0	
412	0	...	0	
147	0	...	0	
723	0	...	0	
669	0	...	0	
1344	0	...	0	
274	0	...	0	
884	0	...	0	
1466	0	...	0	
1129	0	...	0	
1165	0	...	0	
32	0	...	0	
827	0	...	0	
336	0	...	0	
296	0	...	0	
1092	0	...	0	
15	0	...	0	
58	0	...	0	
36	0	...	0	
1577	0	...	0	
1514	0	...	0	
159	0	...	0	
434	0	...	0	
1375	0	...	0	
479	0	...	0	
1058	0	...	0	
185	0	...	0	
739	0	...	0	
...	
824	0	...	0	

1039	0	...	0
102	0	...	0
1359	0	...	0
1181	0	...	0
956	0	...	0
717	0	...	0
812	0	...	0
19	0	...	0
1063	0	...	0
873	0	...	0
196	0	...	0
143	0	...	0
778	0	...	0
318	0	...	0
416	0	...	0
1047	0	...	0
927	0	...	0
749	0	...	0
17	0	...	0
221	1	...	0
943	0	...	0
630	0	...	0
85	0	...	0
1223	0	...	0
667	0	...	0
156	0	...	0
384	0	...	0
645	0	...	0
1480	0	...	0

	TeamName_San Diego Chargers	TeamName_San Francisco 49ers	\
405	0	0	
1258	0	0	
1567	0	0	
412	0	0	
147	0	0	
723	0	0	
669	0	0	
1344	0	1	
274	0	0	
884	0	0	
1466	0	0	
1129	0	0	
1165	0	0	
32	0	0	
827	0	0	
336	0	0	
296	0	0	
1092	0	0	
15	0	0	
58	0	0	
36	0	0	
1577	0	0	
1514	0	0	
159	0	0	
434	0	0	
1375	0	1	

479	0	0
1058	0	0
185	0	0
739	1	0
...
824	0	0
1039	0	0
102	0	0
1359	0	1
1181	0	0
956	0	0
717	0	0
812	0	0
19	0	0
1063	0	0
873	0	0
196	0	0
143	0	0
778	1	0
318	0	0
416	0	0
1047	0	0
927	0	0
749	1	0
17	0	0
221	0	0
943	0	0
630	0	0
85	0	0
1223	0	0
667	0	0
156	0	0
384	0	0
645	0	0
1480	0	0

	TeamName_Seattle Seahawks	TeamName_St. Louis Cardinals \
405	0	0
1258	0	0
1567	0	0
412	0	0
147	0	0
723	0	0
669	0	0
1344	0	0
274	0	0
884	0	0
1466	0	0
1129	0	0
1165	0	0
32	0	1
827	0	0
336	0	0
296	0	0
1092	0	0
15	0	0
58	0	0

36	0	1
1577	0	0
1514	0	0
159	0	0
434	0	0
1375	0	0
479	0	0
1058	0	0
185	0	0
739	0	0
...
824	0	0
1039	0	0
102	0	0
1359	0	0
1181	0	0
956	0	0
717	0	0
812	0	0
19	0	0
1063	0	0
873	0	0
196	0	0
143	0	0
778	0	0
318	0	0
416	0	0
1047	0	0
927	0	0
749	0	0
17	0	0
221	0	0
943	0	0
630	0	0
85	0	0
1223	0	0
667	0	0
156	0	0
384	0	0
645	0	0
1480	0	0
TeamName_St. Louis Rams TeamName_Tampa Bay Buccaneers \		
405	0	0
1258	0	0
1567	0	0
412	0	0
147	0	0
723	0	0
669	0	0
1344	0	0
274	0	0
884	0	0
1466	0	1
1129	0	0
1165	0	0
32	0	0

827	0	0
336	0	0
296	0	0
1092	0	0
15	0	0
58	0	0
36	0	0
1577	0	0
1514	0	0
159	0	0
434	0	0
1375	0	0
479	0	0
1058	0	0
185	0	0
739	0	0
...
824	0	0
1039	0	0
102	0	0
1359	0	0
1181	0	0
956	0	0
717	0	0
812	1	0
19	0	0
1063	0	0
873	0	0
196	0	0
143	0	0
778	0	0
318	0	0
416	0	0
1047	0	0
927	0	0
749	0	0
17	0	0
221	0	0
943	0	0
630	0	0
85	0	0
1223	0	0
667	0	0
156	0	0
384	0	0
645	0	0
1480	0	0

	TeamName_Tennessee Oilers	TeamName_Tennessee Titans \
405	0	0
1258	0	0
1567	0	0
412	0	0
147	0	0
723	0	0
669	0	0
1344	0	0

274	0	0
884	0	0
1466	0	0
1129	0	0
1165	0	0
32	0	0
827	0	0
336	0	0
296	0	0
1092	0	0
15	0	0
58	0	0
36	0	0
1577	0	0
1514	0	0
159	0	0
434	0	0
1375	0	0
479	0	0
1058	0	0
185	0	0
739	0	0
...
824	0	0
1039	0	0
102	0	0
1359	0	0
1181	0	0
956	0	0
717	0	0
812	0	0
19	0	0
1063	0	0
873	0	0
196	0	0
143	0	0
778	0	0
318	0	0
416	0	0
1047	0	0
927	0	0
749	0	0
17	0	0
221	0	0
943	0	0
630	0	0
85	0	0
1223	0	0
667	0	0
156	0	0
384	0	0
645	0	0
1480	0	1

TeamName_Washington Redskins

405	0
1258	0

1567	1
412	0
147	0
723	0
669	0
1344	0
274	0
884	0
1466	0
1129	0
1165	0
32	0
827	0
336	0
296	0
1092	0
15	0
58	0
36	0
1577	1
1514	0
159	0
434	0
1375	0
479	0
1058	0
185	0
739	0
...	...
824	0
1039	0
102	0
1359	0
1181	0
956	0
717	0
812	0
19	0
1063	0
873	0
196	0
143	0
778	0
318	0
416	0
1047	0
927	0
749	0
17	0
221	0
943	0
630	0
85	0
1223	0
667	0
156	0
384	0

645
1480

0
0

[1104 rows x 43 columns]

```
In [21]: from sklearn.linear_model import LogisticRegression

from yellowbrick.classifier import ConfusionMatrix
from yellowbrick.classifier import ClassificationReport
from yellowbrick.classifier import ROCAUC

# Instantiate the classification model
model = LogisticRegression()

#The ConfusionMatrix visualizer takes a model
classes = ['Yes', 'No']
cm = ConfusionMatrix(model, classes=classes, percent=False)

#Fit fits the passed model. This is unnecessary if you pass the visualizer a p
re-fitted model
cm.fit(X_train, y_train)

#To create the ConfusionMatrix, we need some test data. Score runs predict() o
n the data
#and then creates the confusion_matrix from scikit Learn.
cm.score(X_val, y_val)

# change fontsize of the labels in the figure
for label in cm.ax.texts:
    label.set_size(20)

#How did we do?
cm.poof()

# Precision, Recall, and F1 Score
# set the size of the figure and the font size
#%matplotlib inline
plt.rcParams['figure.figsize'] = (15, 7)
plt.rcParams['font.size'] = 20

# Instantiate the visualizer
visualizer = ClassificationReport(model, classes=classes)

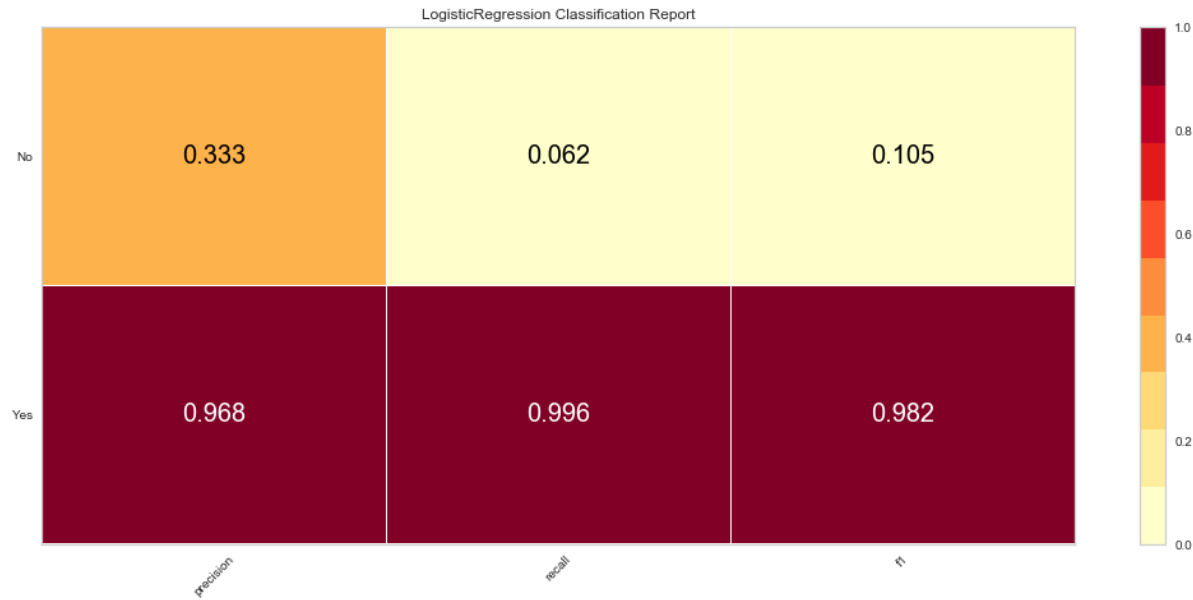
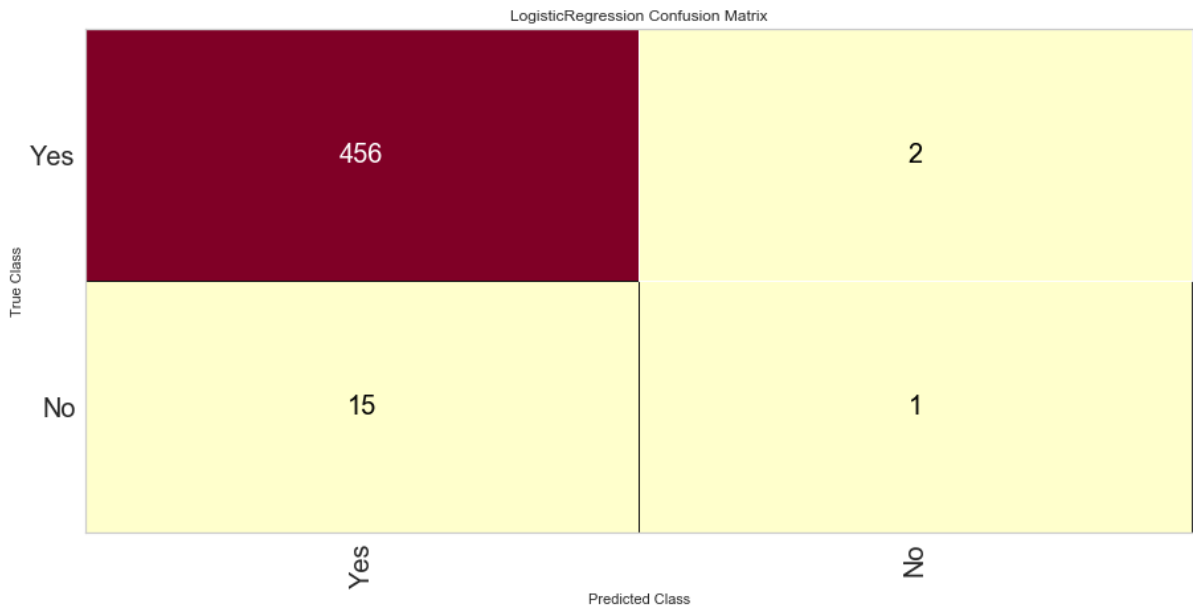
visualizer.fit(X_train, y_train) # Fit the training data to the visualizer
visualizer.score(X_val, y_val) # Evaluate the model on the test data
g = visualizer.poof()

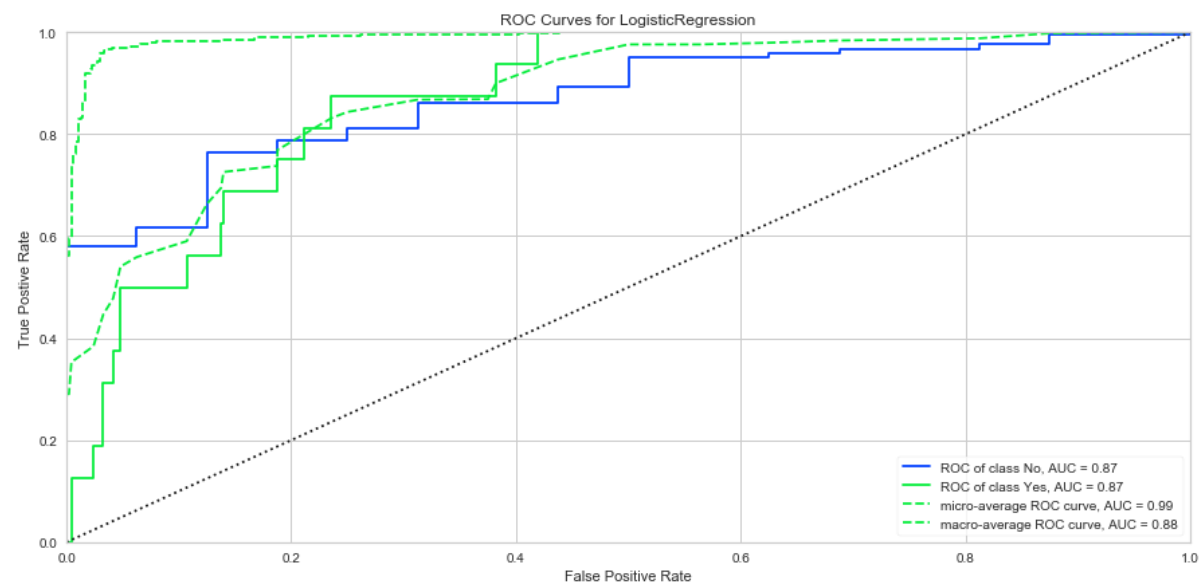
# ROC and AUC
#Instantiate the visualizer
visualizer = ROCAUC(model)

visualizer.fit(X_train, y_train) # Fit the training data to the visualizer
visualizer.score(X_val, y_val) # Evaluate the model on the test data
g = visualizer.poof()
```



```
C:\Users\Kyle Morris\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.  
FutureWarning)
```





In []: