Project

December 14, 2018

```
In [3]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import sklearn
        from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import train_test_split
        from pandas import Series, DataFrame
        from pylab import rcParams
        from sklearn import preprocessing
        from sklearn import metrics
        from sklearn.metrics import classification_report
```

```
In [4]: DOHMH_BEACH = pd.read_csv('DOHMH_Beach_Water_Quality_Data.csv')
```

```
In [5]: display(DOHMH_BEACH.head())
```

```
In [6]: Weather = pd.read_csv('2008-2018 Weather Data.csv')
```

```
In [7]: display(Weather.head())
```

```
In [8]: Weather.DATE.value_counts()
```

```
Out[8]: 9/28/2012     19
        9/27/2012     19
        2/2/2018      18
        12/1/2017     18
        11/15/2017    18
        10/5/2017     18
        12/20/2017    18
        5/1/2018      18
        5/10/2018     18
        10/13/2018    18
        11/14/2017    18
        9/6/2017      18
        9/21/2012     18
        1/31/2018     18
        3/16/2018     18
```

```
12/31/2013    18
6/12/2018     18
5/9/2013      18
10/13/2017    18
5/5/2013      18
9/20/2012     18
11/17/2017    18
1/17/2018     18
3/1/2018      18
3/20/2018     18
5/17/2013     18
3/9/2018      18
9/29/2017     18
8/1/2012      18
5/4/2018      18
              ..
1/20/2008      7
1/18/2008      7
1/3/2008       7
2/6/2008       7
1/17/2008      7
6/8/2008       7
1/9/2008       7
1/29/2008      7
2/5/2008       7
1/4/2008       7
2/24/2008      7
1/11/2008      7
2/2/2008       7
2/22/2008      7
1/24/2008      7
1/30/2008      7
1/28/2008      7
1/1/2008       7
1/16/2008      7
2/4/2008       7
2/3/2008       7
1/6/2008       7
1/15/2008      7
7/28/2011      7
1/25/2008      7
1/2/2008       7
2/14/2008      7
1/23/2008      7
2/15/2008      7
2/8/2008       7
Name: DATE, Length: 3950, dtype: int64
```

```
In [9]: Weather.NAME.value_counts()
```

```
Out[9]: HARRISON, NJ US                                    3950
        NY CITY CENTRAL PARK, NY US                        3949
        JFK INTERNATIONAL AIRPORT, NY US                   3949
        LA GUARDIA AIRPORT, NY US                          3949
        NEWARK LIBERTY INTERNATIONAL AIRPORT, NJ US        3949
        TETERBORO AIRPORT, NJ US                           3943
        KEARNY 1.7 NW, NJ US                               3451
        PALISADES PARK 0.2 WNW, NJ US                      3399
        NORTH ARLINGTON 0.7 WNW, NJ US                     2994
        STATEN ISLAND 1.4 SE, NY US                        2477
        MIDDLE VILLAGE 0.5 SW, NY US                       2440
        STATEN ISLAND 4.5 SSE, NY US                       2429
        MAPLEWOOD TWP 0.9 SE, NJ US                        2128
        BRONX, NY US                                       1921
        SANDY HOOK, NJ US                                  1403
        BROOKLYN 3.1 NW, NY US                             1145
        WOOD RIDGE 0.6 NNW, NJ US                           949
        CARTERET 0.6 WSW, NJ US                             777
        BLOOMFIELD 1.7 S, NJ US                             776
        LINDEN 2.2 NW, NJ US                                605
        UNION TWP 1.1 NW, NJ US                             562
        RUTHERFORD 1.2 N, NJ US                             425
        SADDLE ROCK 3.4 WSW, NY US                          415
        NORTH ARLINGTON 0.7 NE, NJ US                       284
        WOOD RIDGE 0.6 SE, NJ US                            277
        ROSELLE PARK 0.5 ENE, NJ US                         229
        BROOKLYN 2.4 SW, NY US                              170
        JACKSON HEIGHTS 0.3 WSW, NY US                      157
        KENILWORTH 0.8 SSE, NJ US                           108
        Name: NAME, dtype: int64
```

```
In [10]: DOHMH_BEACH_Manhattan = DOHMH_BEACH[DOHMH_BEACH["Beach Name"]=="MANHATTAN BEACH"]
```

```
In [11]: Weather_JFK = Weather[Weather["NAME"]=="JFK INTERNATIONAL AIRPORT, NY US"]
```

```
In [12]: DOHMH_BEACH_MANHATTAN_COLS = DOHMH_BEACH_Manhattan[["Sample Date","Enterococci Result
```

```
In [13]: DOHMH_BEACH_MANHATTAN_COLS.columns= ['DATE','Enterococci']
```

```
In [14]: display(DOHMH_BEACH_MANHATTAN_COLS.head())
```

```
In [15]: DOHMH_BEACH_MANHATTAN_COLS['DATE'] = pd.to_datetime(DOHMH_BEACH_MANHATTAN_COLS['DATE'
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':
```

```
In [16]: DOHMH_BEACH_MANHATTAN_COLS.sort_values(by='DATE',inplace=True)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':
```

```
In [20]: df_weatherdata = Weather_JFK[['DATE','PRCP']]
```

```
In [21]: df_weatherdata.columns=['DATE','PRCP']
```

```
In [22]: df_weatherdata['DATE']=pd.to_datetime(df_weatherdata['DATE'])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':
```

```
In [23]: df_weatherdata.sort_values(by='DATE',inplace=True)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':
```

```
In [24]: df_weatherdata.DATE.value_counts()
```

```
Out[24]: 2015-10-18    1
         2011-06-07    1
         2018-02-20    1
         2013-07-23    1
         2018-01-14    1
         2010-02-12    1
         2015-02-13    1
         2013-06-19    1
         2013-03-01    1
         2018-06-11    1
         2016-03-19    1
```

```
2012-10-15    1
2010-07-04    1
2008-03-22    1
2017-11-15    1
2018-02-15    1
2014-07-28    1
2009-07-23    1
2017-07-28    1
2016-06-06    1
2010-08-11    1
2008-06-09    1
2012-03-27    1
2012-12-25    1
2017-12-11    1
2012-11-22    1
2016-08-24    1
2014-05-13    1
2008-02-17    1
2009-02-23    1
             ..
2014-01-13    1
2015-09-23    1
2010-07-11    1
2013-05-08    1
2012-10-23    1
2012-08-19    1
2011-10-04    1
2012-08-16    1
2009-03-14    1
2008-01-22    1
2009-05-21    1
2017-08-30    1
2014-01-15    1
2014-07-08    1
2014-06-21    1
2014-09-26    1
2013-05-15    1
2008-05-03    1
2016-04-07    1
2010-07-23    1
2008-04-10    1
2008-04-29    1
2008-05-01    1
2009-06-24    1
2009-12-11    1
2012-04-05    1
2011-01-27    1
2016-10-02    1
```

```
         2012-11-26    1
         2014-04-09    1
         Name: DATE, Length: 3949, dtype: int64
```

```
In [25]: df_weatherdata.sort_values(by='DATE',inplace=True)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':
```

```
In [26]: df_weatherdata.shape
```

```
Out[26]: (3949, 2)
```

```
In [27]: display(df_weatherdata.head())
```

```
In [28]: DOHMH_BEACH_MANHATTAN_COLS[DOHMH_BEACH_MANHATTAN_COLS.DATE == '06/29/2009']
```

```
In [29]: Weather[Weather.DATE == '06/29/2009']
```

```
In [30]: df_merge = pd.merge(DOHMH_BEACH_MANHATTAN_COLS,df_weatherdata,how='inner',on='DATE')
```

```
In [31]: df_merge.head()
```

```
In [32]: df_merge[df_merge['PRCP'] == 0]
```

```
In [33]: df_merge.PRCP.value_counts()
```

```
Out[33]: 0.00    480
         0.02     21
         0.01     12
         0.05     12
         0.10      9
         0.09      9
         0.78      9
         0.16      6
         0.20      6
         0.12      6
         0.11      6
         0.87      3
         1.12      3
         0.95      3
         0.13      3
         0.98      3
         0.29      3
         0.50      3
         0.57      3
```

```
        0.37      3
        0.67      3
        1.38      3
        0.04      3
        1.22      3
        0.22      3
        0.76      3
        0.65      3
        0.58      3
        0.61      3
        0.23      3
        0.40      3
        0.26      3
        0.48      3
        1.08      3
        0.39      3
        0.18      3
        1.03      3
        0.44      3
        1.05      3
        0.60      3
        0.27      3
        1.09      3
        0.07      3
        0.47      3
        1.54      3
        0.43      3
        0.90      3
        0.51      3
        Name: PRCP, dtype: int64
```

```
In [34]: df_merge.head(10)
```

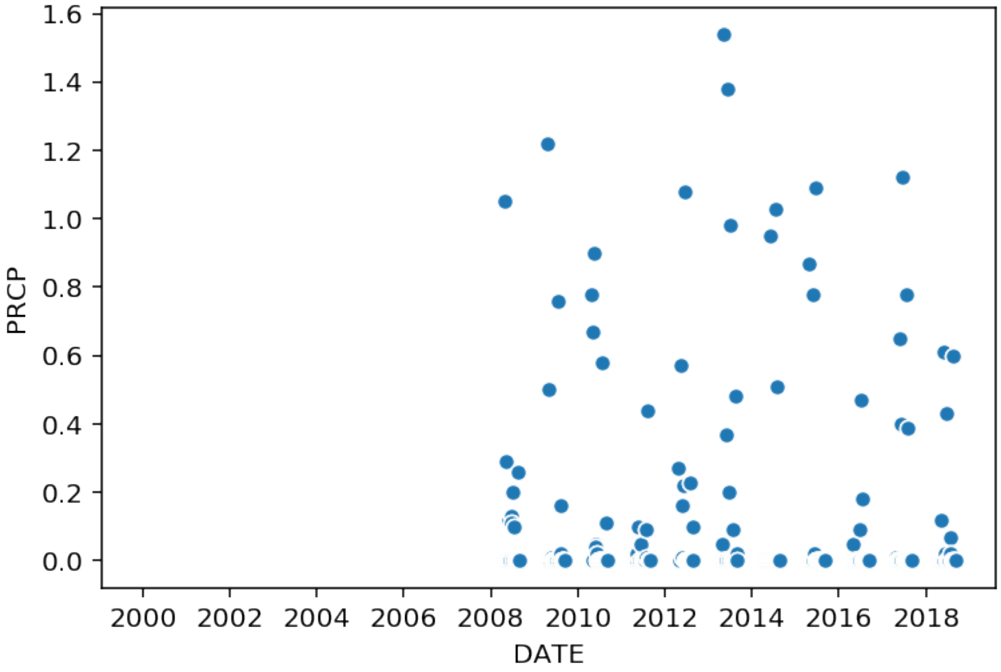
```
In [74]: a =type(DOHMH_BEACH_MANHATTAN_COLS.loc[1,'DATE'])
```

```
In [0]: display(df_merge.head())
```

```
In [0]: PRCP_Enterococci = sns.scatterplot(x='PRCP',y='Enterococci',data=df_merge)
```

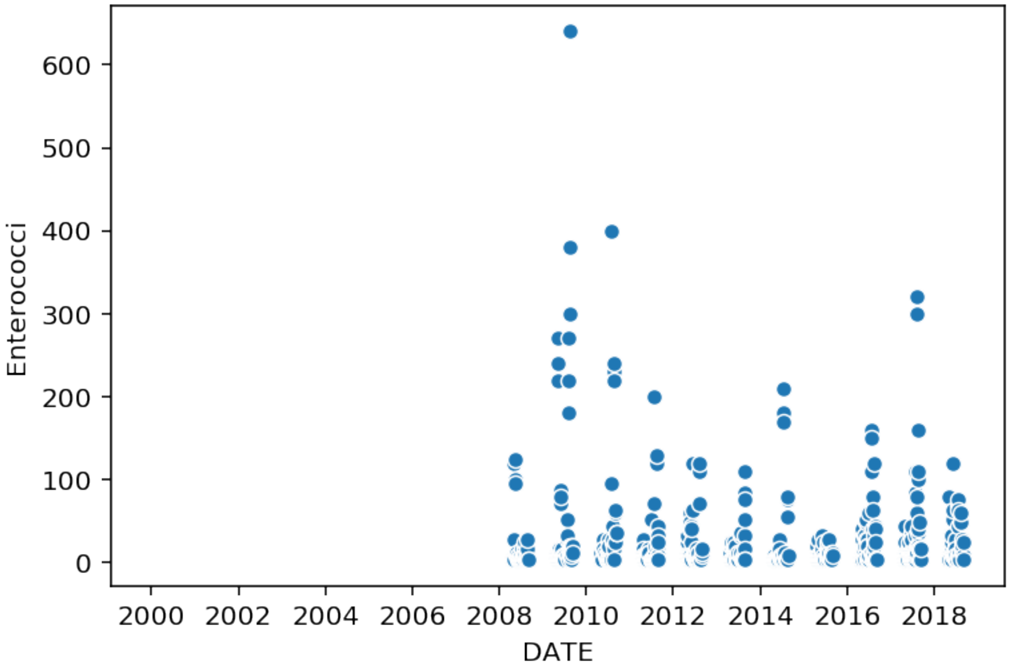
```
In [37]: DATE_PRCP = sns.scatterplot(x = 'DATE',y = 'PRCP',data = df_merge)
```

```
Out[37]:
```



```
In [38]: DATE_ENTEROCOCCI = sns.scatterplot(x ='DATE',y ='Enterococci', data = df_merge)
```

```
Out[38]:
```



```
In [39]: %matplotlib inline
         rcParams['figure.figsize'] = 10, 8
         sns.set_style('whitegrid')
```

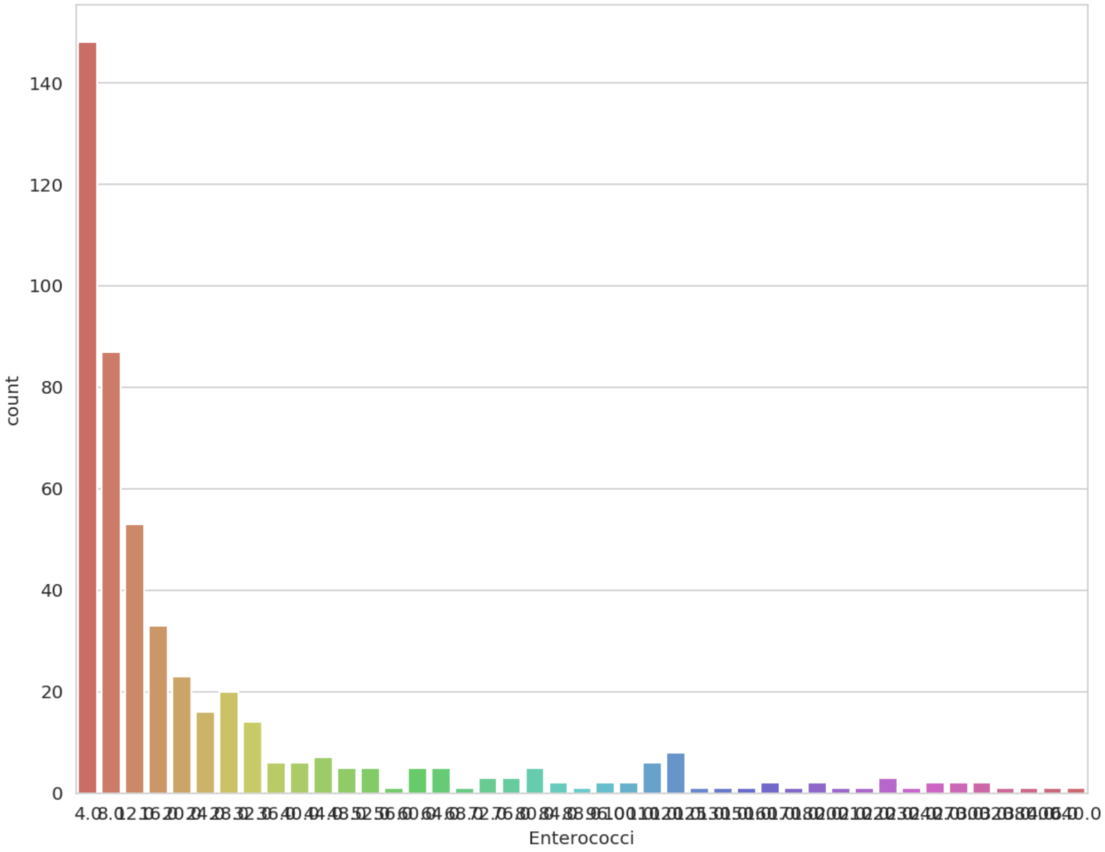
```
In [40]: df_merge.head()
```

```
In [41]: df_merge.columns = ['DATE','Enterococci','PRCP']
```

```
In [42]: sns.countplot(x = 'Enterococci', data = df_merge, palette = 'hls')
```

```
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6bf337afd0>
```

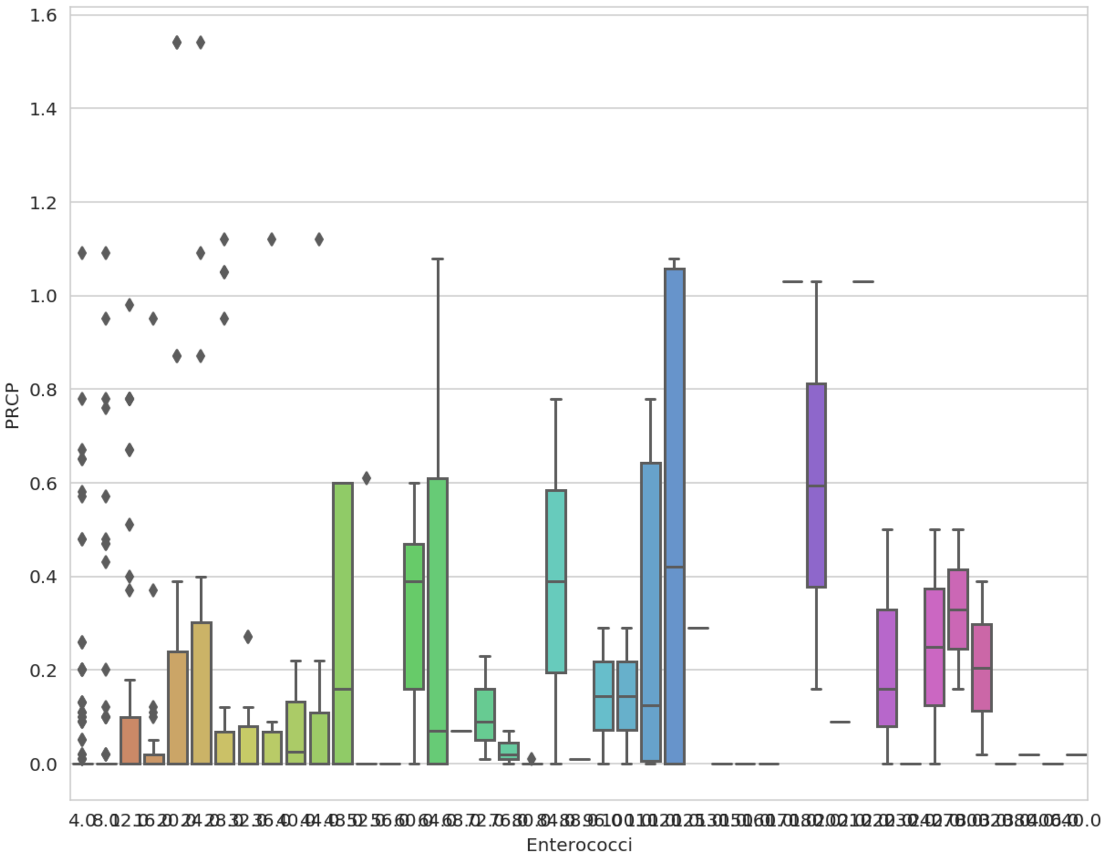
```
Out[42]:
```



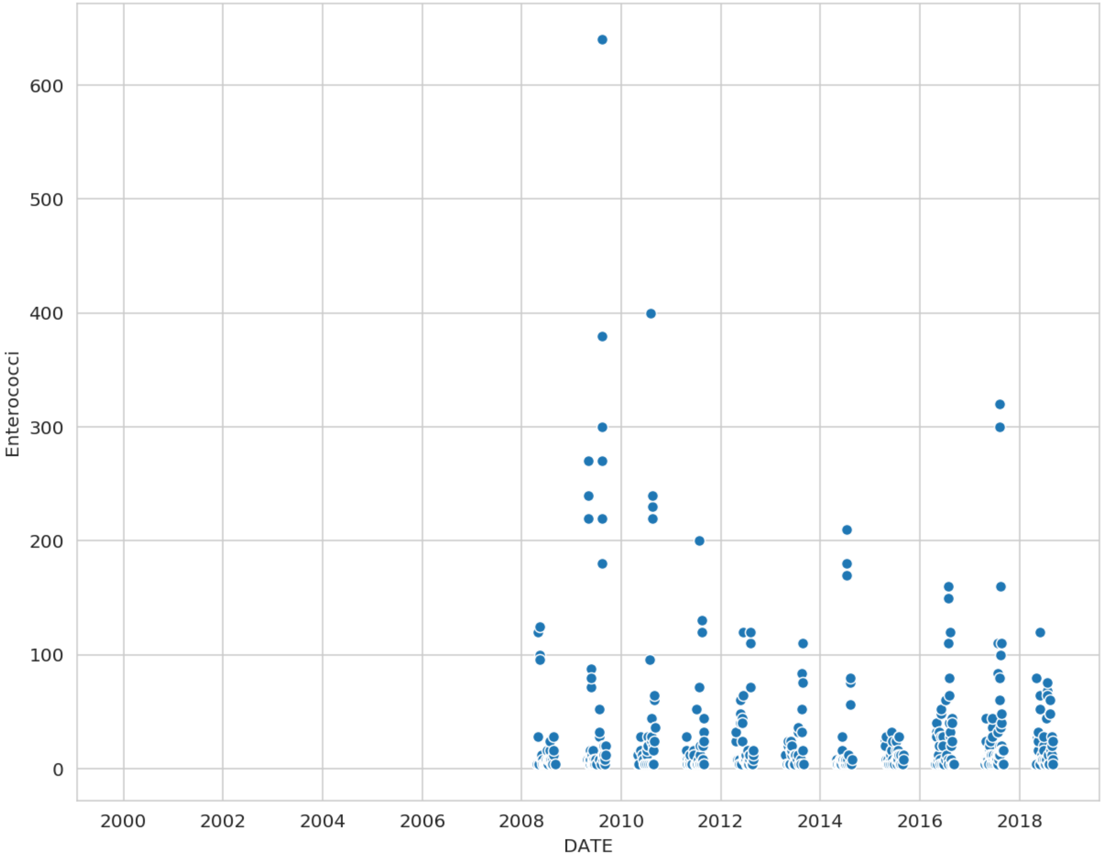
```
In [43]: sns.boxplot(x = 'Enterococci', y = 'PRCP', data = df_merge, palette = 'hls')
```

```
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6bf33b81d0>
```

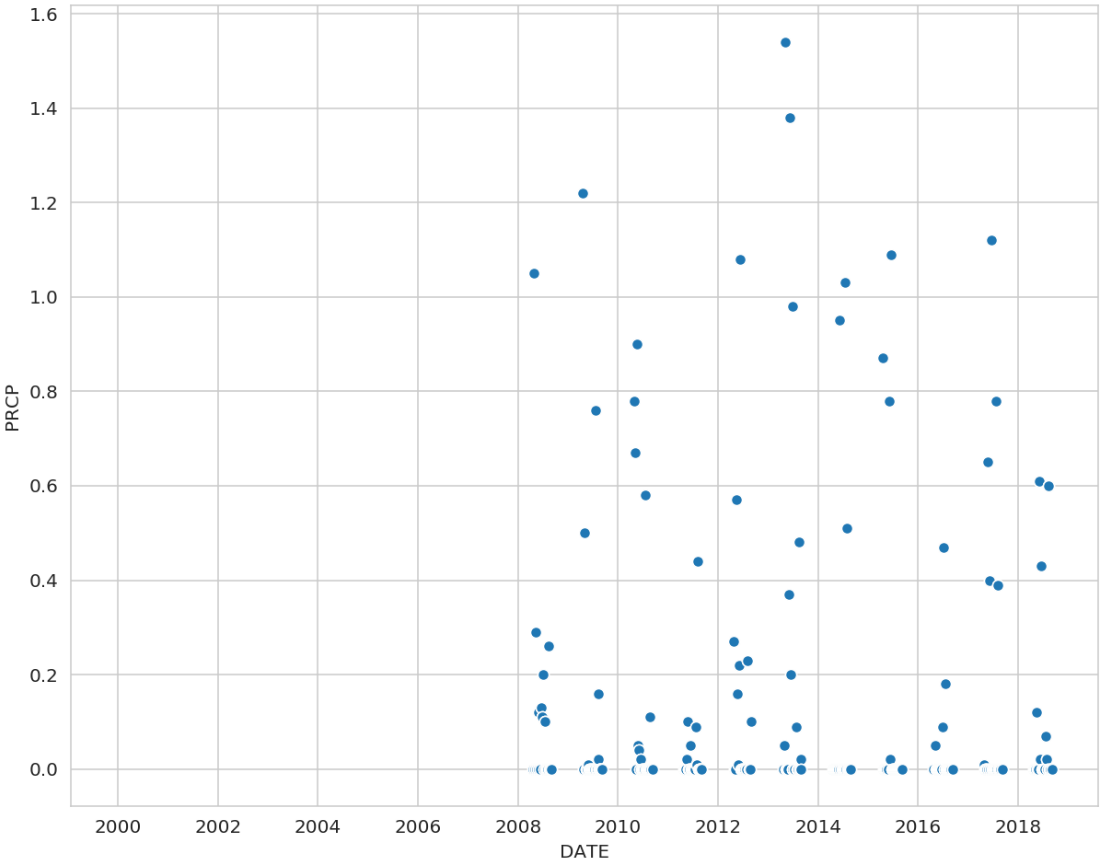
```
Out[43]:
```



```
In [44]: sns.scatterplot(x='DATE', y='Enterococci', data = df_merge, palette = 'hls')

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6bf29d4f60>

Out[44]:
```



```
In [45]: sns.scatterplot(x='DATE',y='PRCP',data = df_merge)

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6bf29e0d68>

Out[45]:
```



```
In [46]: df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 687 entries, 0 to 686
Data columns (total 3 columns):
DATE            687 non-null datetime64[ns]
Enterococci     491 non-null float64
PRCP            687 non-null float64
dtypes: datetime64[ns](1), float64(2)
memory usage: 41.5 KB
```

```
In [47]: df_merge.describe()
```

0.1 LOGISTIC REGRESSION

```
In [48]: df_merge.isnull().sum()
```

```
Out[48]: DATE             0
         Enterococci    196
```

```
        PRCP                0
        dtype: int64
```

```
In [49]: df_merge.dropna(inplace=True)
```

```
In [50]: def eWarn(m):
             if m > 104:
                 return 1
             else:
                 return 0
```

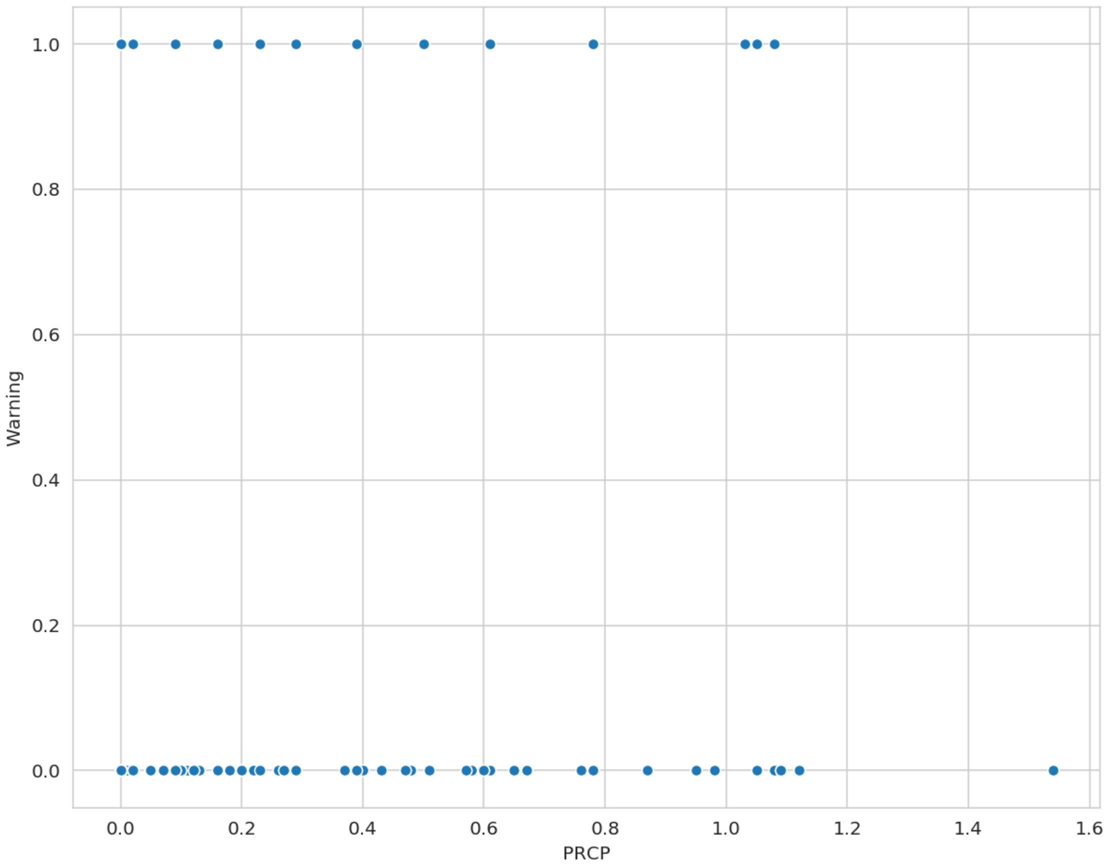
```
In [51]: df_merge['Warning'] = df_merge.Enterococci.apply(eWarn)
```

```
In [52]: df_merge.head()
```

```
In [53]: sns.scatterplot(x='PRCP',y='Warning',data=df_merge)
```

```
Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6bf036a0f0>
```

Out[53]:



```
In [54]: X = np.array(df_merge.PRCP)
         y = df_merge.Warning
```

```
In [55]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state
```

```
In [56]: logmodel = LogisticRegression(solver='liblinear')
```

```
In [57]: logmodel.fit(X_train.reshape(-1,1),y_train)
```

```
Out[57]: 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='liblinear',
                   tol=0.0001, verbose=0, warm_start=False)
```

```
In [58]: b0 = logmodel.intercept_
```

```
In [59]: b1 = logmodel.coef_
```

```
In [60]: b0
```

```
Out[60]: array([-2.45540583])
```

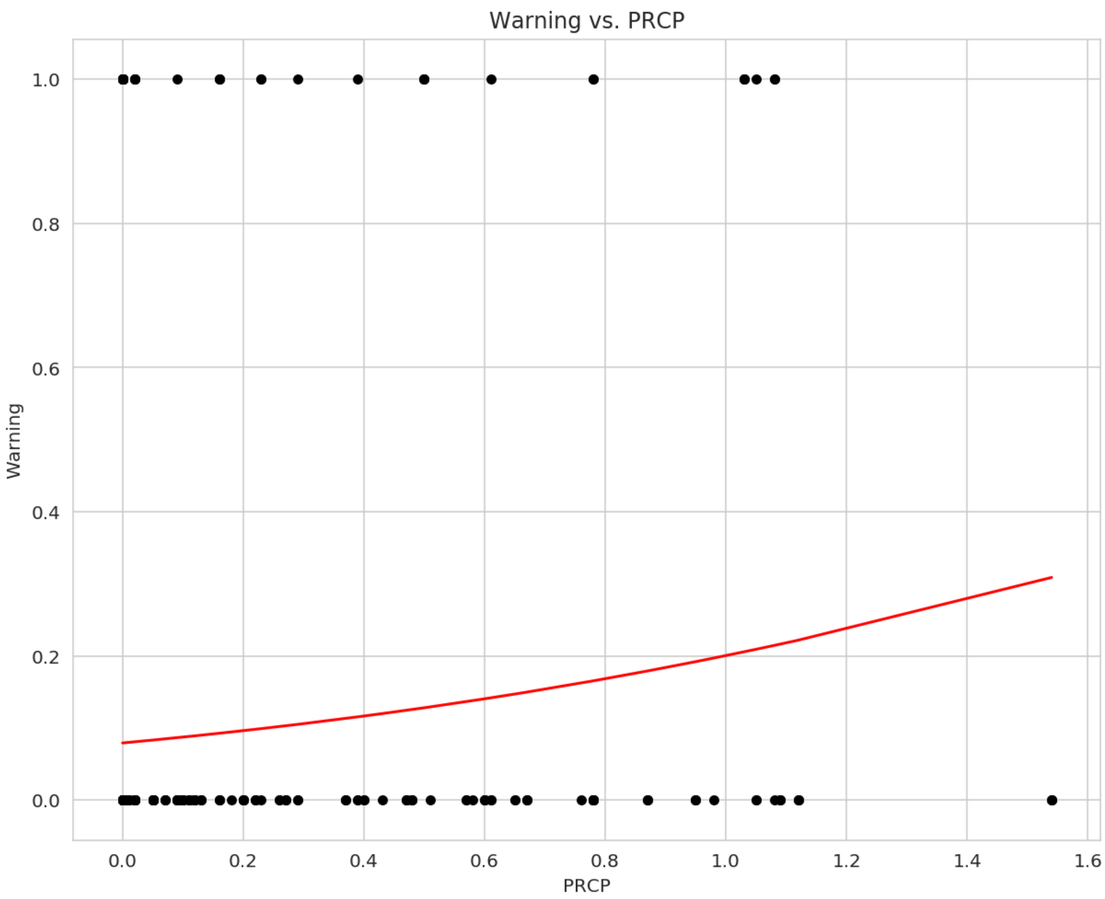
```
In [61]: b1
```

```
Out[61]: array([[1.07108309]])
```

```
In [63]: X2 = sorted(X)
         df_merge.plot.scatter(x='PRCP',y='Warning',color='black')
         plt.plot(X2,1/(1+np.exp(-b0-b1*X2)).reshape(-1,1),'r')
         plt.title('Warning vs. PRCP')
```

```
Out[63]: Text(0.5,1,'Warning vs. PRCP')
```

```
Out[63]:
```



```
In [66]: Predict = logmodel.predict(X_test.reshape(-1,1))
```

```
In [68]: print(classification_report(y_test,Predict))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       153
           1       0.00      0.00      0.00        10

   micro avg       0.94      0.94      0.94       163
   macro avg       0.47      0.50      0.48       163
weighted avg       0.88      0.94      0.91       163



/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:1143: UndefinedMetric
  'precision', 'predicted', average, warn_for)
```

```
In [0]: X_train.isnull().sum()
In [90]: y_train.isnull().sum()

Out[90]: Enterococci    1693
         dtype: int64

In [91]: y_train.dropna(inplace=True)

/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  if __name__ == '__main__':


In [92]: df_merge.isnull().sum()

Out[92]: DATE           0
         Enterococci    0
         PRCP           0
         dtype: int64

In [93]: df_merge.dropna(inplace=True)

In [94]: df_merge.isnull().sum()

Out[94]: DATE           0
         Enterococci    0
         PRCP           0
         dtype: int64

In [95]: y_train.isnull().sum()

Out[95]: Enterococci    0
         dtype: int64

In [0]:
```