Complete Final project

December 11, 2018

1 West Nile Virus Prediction

West Nile virus is spread to humans from infected mosquito bites. West Nile virus is present throughout the world, from Africa and Europe, to North and South America. While most cases of the virus do not show symptoms those that do have very severe fever, headaches, nausea and vomiting, often requiring them to miss days of work. In about 10% of cases, that show symptoms, patients will die, in 2012 the United States had one its worst outbreaks leading to over 250 deaths. To prevent these cases countries have developed many control strategies, including spraying insecticides.

Spraying insecticides kills the adult mosquitoes, our primary concern is the adult female mosquito, who bites humans for blood. There are periods where there will be very few adult mosquitoes present, but the juvenile stages, often in water, will present and unreachable by spraying. Due to this fact we can hope to optimize our spraying by using years of trap collection data, along with locations and weather data, to find the best time for spraying to occur.

The dataset I have chosen in found on Kaggle, called West Nile Virus Prediction. It has a dataset containing <100,000 mosquito samples, and their status of virus present. Then a data set with weather conditions, temperature, sun, rain, sunset time, sunrise, and so on. I hope to use this data set to be able to predict the next outbreak of West Nile

The data has already been split into a test and training set. In the training set mosquitoes have been trapped at trap site across chicago, with longitude and latitude points. Then the mosquito traps were emptied and tested for the presence of west nile virus, the mosquitoes are identified down to the species level. The test sets include the years 2007, 2009, 2011, and 2013. While the test set does not include the presence of west nile virus in the mosquiotes collected for the years 2008, 2010, 2012, and 2014. The point of the data is predict the number of cases in these years in the test set, based off the data in the training set.

First, I would like to do some EDA, by creating a map of the number of mosquitoes collected from traps across the city by different years.

```
In [1]: import pandas as pd
        import numpy as np

        import matplotlib.pyplot as pl
        #%matplotlib inline

        from sklearn.neighbors import KernelDensity
        from __future__ import print_function
        import datetime
```

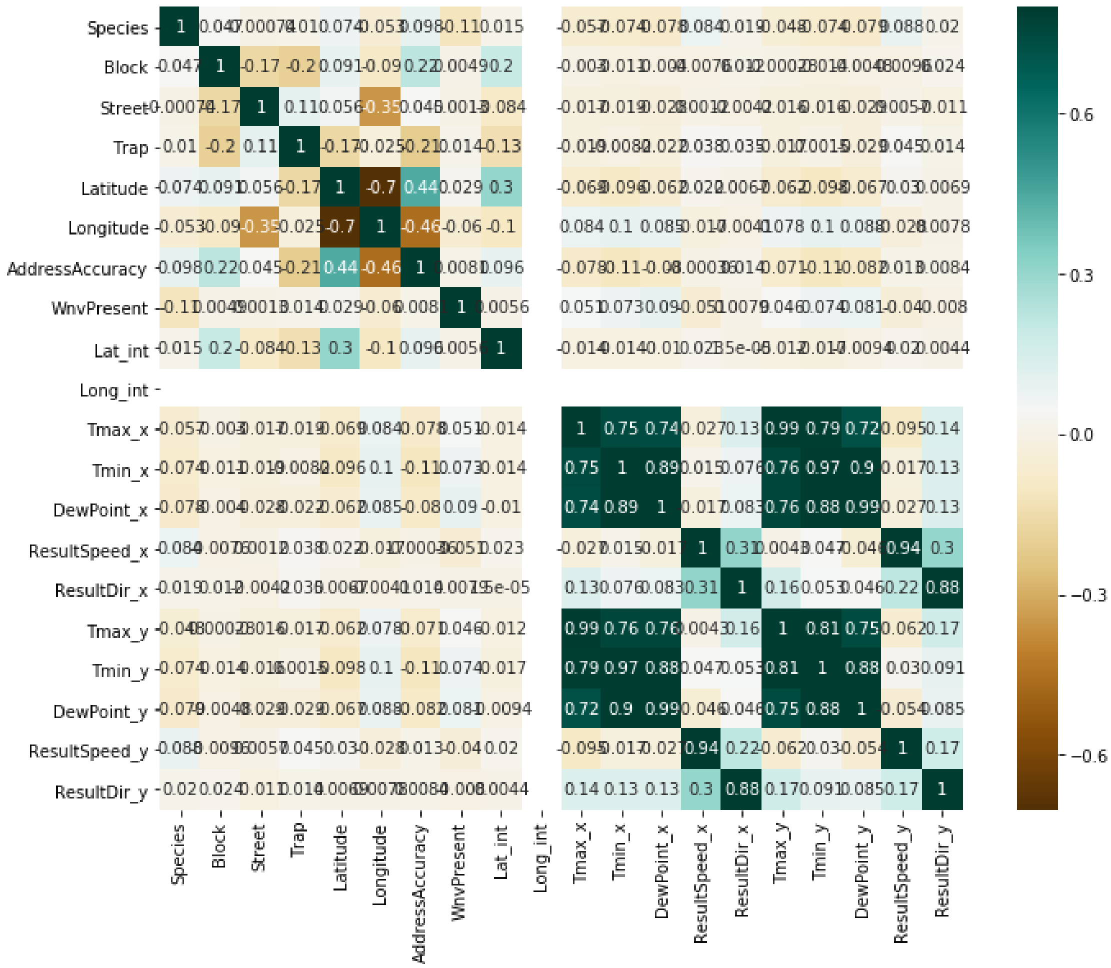
```
from sklearn.cross_validation import train_test_split
import csv
from sklearn import metrics
from sklearn.utils import shuffle
```

```
ImportErrorTraceback (most recent call last)

<ipython-input-1-de8e27beaba7> in <module>()
  2 import numpy as np
  3
----> 4 import xelatex
  5
  6 import matplotlib.pyplot as pl


ImportError: No module named xelatex
```

1.1 EDA

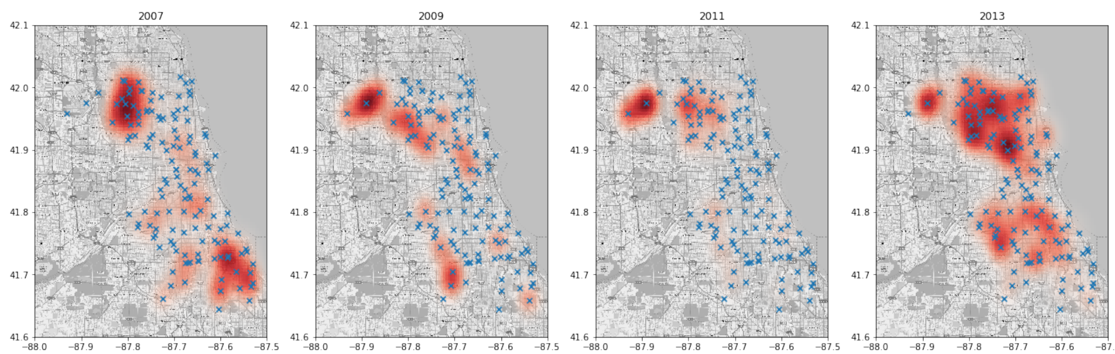
```
In [3]: mapdata = np.loadtxt("C:\Users\deanm\OneDrive\Documents\University of Idaho\Classes\Fal
        traps = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
        species = pd.np.unique(traps['Species'])
```

```
In [17]: import seaborn as sns
         hmap = train.corr()
         pl.subplots(figsize=(12, 9))
         sns.heatmap(hmap, vmax=.8,annot=True,cmap="BrBG", square=True);
```




```
In [4]: alpha_cm = pl.cm.Reds
        alpha_cm._init()
        alpha_cm._lut[:-3,-1] = abs(np.logspace(0, 1, alpha_cm.N) / 10 - 1)[::-1]
        aspect = mapdata.shape[0] * 1.0 / mapdata.shape[1]
        lon_lat_box = (-88, -87.5, 41.6, 42.1)

        pl.figure(figsize=(18,6))
        for year, subplot in zip([2007, 2009, 2011, 2013], [141, 142, 143, 144]):
            sightings = traps[(traps['WnvPresent'] > 0) & (traps['Date'].apply(lambda x: x.yea
            sightings = sightings.groupby(['Date', 'Trap','Longitude', 'Latitude']).max()['Wnv
            X = sightings[['Longitude', 'Latitude']].values
            kd = KernelDensity(bandwidth=0.02)
            kd.fit(X)

            xv,yv = np.meshgrid(np.linspace(-88, -87.5, 100), np.linspace(41.6, 42.1, 100))
            gridpoints = np.array([xv.ravel(),yv.ravel()]).T
            zv = np.exp(kd.score_samples(gridpoints).reshape(100,100))
```

```
        pl.subplot(subplot)
        pl.gca().set_title(year)
        pl.imshow(mapdata,
                   cmap=pl.get_cmap('gray'),
                   extent=lon_lat_box,
                   aspect=aspect)
        pl.imshow(zv,
                   origin='lower',
                   cmap=alpha_cm,
                   extent=lon_lat_box,
                   aspect=aspect)
        pl.tight_layout()
        locations = traps[['Longitude', 'Latitude']].drop_duplicates().values
        pl.scatter(locations[:,0], locations[:,1], marker='x')

    pl.savefig('heatmap.png')
```

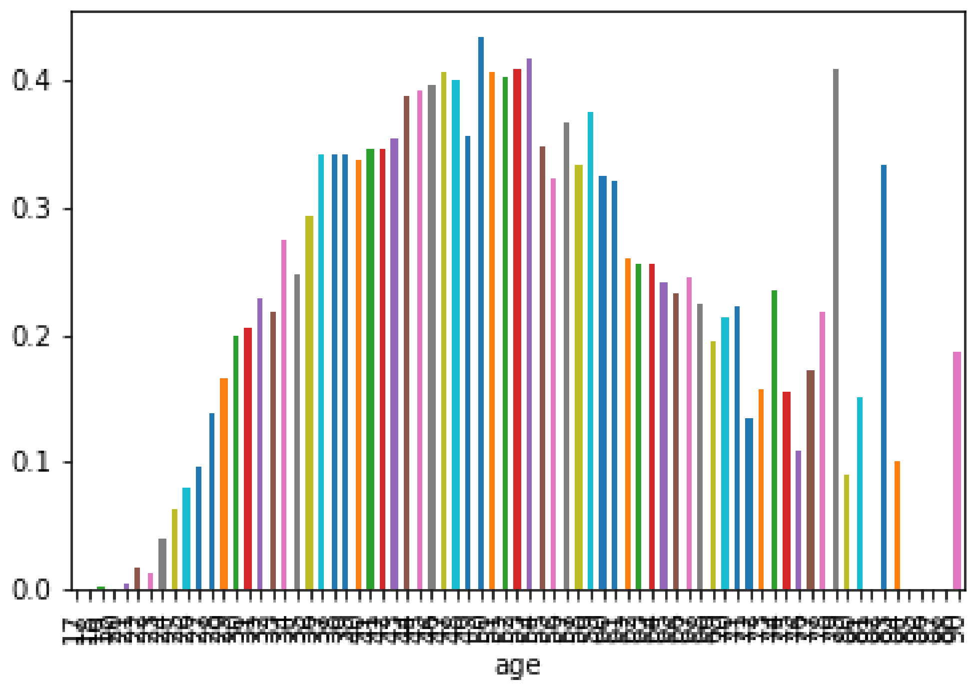


From these maps is would appear that 2011 did not have a lot of WNV, while there was more in 2007 and 2009. Based on these maps 2013 seems to be an outbreak year, meaning this year may prove to be useful in models we build.

```
In [5]: alpha_cm = pl.cm.Reds
        alpha_cm._init()
        alpha_cm._lut[:-3,-1] = abs(np.logspace(0, 1, alpha_cm.N) / 10 - 1)[::-1]
        alpha_mcm = pl.cm.Greens
        alpha_mcm._init()
        alpha_mcm._lut[:-3,-1] = abs(np.logspace(0, 1, alpha_cm.N) / 10 - 1)[::-1]
        aspect = mapdata.shape[0] * 1.0 / mapdata.shape[1]
        lon_lat_box = (-88, -87.5, 41.6, 42.1)

        subplot = 0
        numSpcs = len(species)
        pl.figure(figsize=(18,6*numSpcs))
        for spcsIndx in range(numSpcs):
            for year in [2007, 2009, 2011, 2013]:
```

```
        subplot += 1
        sightings = traps[(traps['Species'] == species[spcsIndx])
                          & (traps['WnvPresent'] > 0)
                          & (traps['Date'].apply(lambda x: x.year) == year)]
        sightings = sightings.groupby(['Date', 'Trap', 'Longitude', 'Latitude', 'Specie
        mSightings = traps[(traps['Species'] == species[spcsIndx])
                          & (traps['Date'].apply(lambda x: x.year) == year)]
        mSightings = mSightings.groupby(['Date', 'Trap', 'Longitude', 'Latitude', 'Spe
        if(len(mSightings) <= 0):
            print("SKIPPING [" + str(subplot) + "]:" + str(year) + " (" + species[spcsI
            continue

        mX = mSightings[['Longitude', 'Latitude']].values
        mkd = KernelDensity(bandwidth=0.02)
        mkd.fit(mX)
        mxv,myv = np.meshgrid(np.linspace(-88, -87.5, 100), np.linspace(41.6, 42.1, 10
        mGridpoints = np.array([mxv.ravel(),myv.ravel()]).T
        mzv = np.exp(mkd.score_samples(mGridpoints).reshape(100,100))

        pl.subplot(numSpcs, 4, subplot)
        pl.gca().set_title(str(year) + " (" + species[spcsIndx] + ")")
        pl.imshow(mapdata,
                   cmap=pl.get_cmap('gray'),
                   extent=lon_lat_box,
                   aspect=aspect)
        pl.imshow(mzv,
                   origin='lower',
                   cmap=alpha_mcm,
                   extent=lon_lat_box,
                   aspect=aspect)
        if(len(sightings) > 0):
            X = sightings[['Longitude', 'Latitude']].values
            kd = KernelDensity(bandwidth=0.02)
            kd.fit(X)
            xv,yv = np.meshgrid(np.linspace(-88, -87.5, 100), np.linspace(41.6, 42.1,
            gridpoints = np.array([xv.ravel(),yv.ravel()]).T
            zv = np.exp(kd.score_samples(gridpoints).reshape(100,100))
            pl.imshow(zv,
                       origin='lower',
                       cmap=alpha_cm,
                       extent=lon_lat_box,
                       aspect=aspect)
        print("         [" + str(subplot) + "]:" + str(year) + " (" + species[spcsIndx
        pl.tight_layout()
        locations = traps[['Longitude', 'Latitude']].drop_duplicates().values
        pl.scatter(locations[:,0], locations[:,1], marker='x')

pl.savefig('heatmap.png')
```

```
SKIPPING [1]:2007 (CULEX ERRATICUS)                No sightings
SKIPPING [2]:2009 (CULEX ERRATICUS)                No sightings
SKIPPING [3]:2011 (CULEX ERRATICUS)                No sightings
         [4]:2013 (CULEX ERRATICUS)
         [5]:2007 (CULEX PIPIENS)
         [6]:2009 (CULEX PIPIENS)
         [7]:2011 (CULEX PIPIENS)
         [8]:2013 (CULEX PIPIENS)
         [9]:2007 (CULEX PIPIENS/RESTUANS)
         [10]:2009 (CULEX PIPIENS/RESTUANS)
         [11]:2011 (CULEX PIPIENS/RESTUANS)
         [12]:2013 (CULEX PIPIENS/RESTUANS)
         [13]:2007 (CULEX RESTUANS)
         [14]:2009 (CULEX RESTUANS)
         [15]:2011 (CULEX RESTUANS)
         [16]:2013 (CULEX RESTUANS)
         [17]:2007 (CULEX SALINARIUS)
         [18]:2009 (CULEX SALINARIUS)
         [19]:2011 (CULEX SALINARIUS)
         [20]:2013 (CULEX SALINARIUS)
SKIPPING [21]:2007 (CULEX TARSALIS)                No sightings
         [22]:2009 (CULEX TARSALIS)
         [23]:2011 (CULEX TARSALIS)
SKIPPING [24]:2013 (CULEX TARSALIS)                No sightings
         [25]:2007 (CULEX TERRITANS)
         [26]:2009 (CULEX TERRITANS)
         [27]:2011 (CULEX TERRITANS)
         [28]:2013 (CULEX TERRITANS)
```



As we can see in the above figure different species of mosquitoes prefer different areas of the city. This to us indicates species is an important determinenistic feature for our model. Since in the WNV by year map we don't see an virus in parts of the cities that has lots of mosquitoes. So, certain mosquito species are more likely to carry the disease than other, and in fact this is a well established in vector ecology, some mosquitoes are better hosts than others.

Since mosquito species and location seem to be a good indicator of where disease is, I plan to create a few simple models with just this data set, excluding the weather information for now. I will use models we have learned in class, neural network, decision tree, random forest, and support vector machine.

I will be using 10-fold cross validation for each model. Then calculating the accuracy score as the mean of these ten models, along with the mean AUC. Accuracy is defined as the precent of correct predictions the model makes.

1.2 Basic Models

```
In [14]: train = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fa
         train =train.drop(train.columns[3:7], axis=1)

         features_categorical = ["Species"]
         for c in features_categorical:
             train[c] = pd.Categorical(train[c]).codes
```

```
In [15]: train
```

```
Out[15]:           Date                                            Address  Species  \
        0     2007-05-29  4100 North Oak Park Avenue, Chicago, IL 60634,...        2
```

1	2007-05-29	4100 North Oak Park Avenue, Chicago, IL 60634,...	3
2	2007-05-29	6200 North Mandell Avenue, Chicago, IL 60646, USA	3
3	2007-05-29	7900 West Foster Avenue, Chicago, IL 60656, USA	2
4	2007-05-29	7900 West Foster Avenue, Chicago, IL 60656, USA	3
5	2007-05-29	1500 West Webster Avenue, Chicago, IL 60614, USA	3
6	2007-05-29	2500 West Grand Avenue, Chicago, IL 60654, USA	3
7	2007-05-29	1100 Roosevelt Road, Chicago, IL 60608, USA	2
8	2007-05-29	1100 Roosevelt Road, Chicago, IL 60608, USA	3
9	2007-05-29	1100 West Chicago Avenue, Chicago, IL 60642, USA	3
10	2007-05-29	2100 North Stave Street, Chicago, IL 60647, USA	2
11	2007-05-29	2200 North Cannon Drive, Chicago, IL 60614, USA	2
12	2007-05-29	2200 North Cannon Drive, Chicago, IL 60614, USA	3
13	2007-05-29	2200 West 113th Street, Chicago, IL 60643, USA	2
14	2007-05-29	2200 West 113th Street, Chicago, IL 60643, USA	3
15	2007-05-29	1100 South Peoria Street, Chicago, IL 60608, USA	3
16	2007-05-29	1700 West 95th Street, Chicago, IL 60643, USA	3
17	2007-05-29	2200 West 89th Street, Chicago, IL 60643, USA	3
18	2007-05-29	2200 West 89th Street, Chicago, IL 60643, USA	1
19	2007-05-29	North Streeter Drive, Chicago, IL 60611, USA	2
20	2007-05-29	North Streeter Drive, Chicago, IL 60611, USA	3
21	2007-05-29	6500 North Oak Park Avenue, Chicago, IL 60631,...	2
22	2007-05-29	7500 North Oakley Avenue, Chicago, IL 60645, USA	2
23	2007-05-29	1500 North Long Avenue, Chicago, IL 60651, USA	3
24	2007-05-29	8900 South Carpenter Street, Chicago, IL 60620...	3
25	2007-06-05	4100 North Oak Park Avenue, Chicago, IL 60634,...	2
26	2007-06-05	4100 North Oak Park Avenue, Chicago, IL 60634,...	3
27	2007-06-05	4100 North Oak Park Avenue, Chicago, IL 60634,...	1
28	2007-06-05	7900 West Foster Avenue, Chicago, IL 60656, USA	2
29	2007-06-05	7900 West Foster Avenue, Chicago, IL 60656, USA	3
...	...	...	...
10476	2013-09-26	South Cottage Grove Avenue, Chicago, IL, USA	2
10477	2013-09-26	South Cottage Grove Avenue, Chicago, IL, USA	3
10478	2013-09-26	South Cottage Grove Avenue, Chicago, IL, USA	1
10479	2013-09-26	5800 North Pulaski Road, Chicago, IL 60646, USA	1
10480	2013-09-26	4000 East 130th Street, Chicago, IL 60633, USA	2
10481	2013-09-26	4000 East 130th Street, Chicago, IL 60633, USA	3
10482	2013-09-26	4000 East 130th Street, Chicago, IL 60633, USA	1
10483	2013-09-26	9100 West Higgins Road, Rosemont, IL 60018, USA	3
10484	2013-09-26	ORD Terminal 5, O'Hare International Airport, ...	2
10485	2013-09-26	ORD Terminal 5, O'Hare International Airport, ...	2
10486	2013-09-26	ORD Terminal 5, O'Hare International Airport, ...	1
10487	2013-09-26	ORD Terminal 5, O'Hare International Airport, ...	1
10488	2013-09-26	ORD Terminal 5, O'Hare International Airport, ...	1
10489	2013-09-26	ORD Terminal 5, O'Hare International Airport, ...	1
10490	2013-09-26	4800 West Montana Street, Chicago, IL 60639, USA	2
10491	2013-09-26	5100 North Mont Clare Avenue, Chicago, IL 6065...	2
10492	2013-09-26	5100 North Mont Clare Avenue, Chicago, IL 6065...	1
10493	2013-09-26	8200 South Kostner Avenue, Chicago, IL 60652, USA	2

10494	2013-09-26	East 91st Place, Chicago, IL, USA	2
10495	2013-09-26	East 91st Place, Chicago, IL, USA	1
10496	2013-09-26	1700 West Addison Street, Chicago, IL 60613, USA	2
10497	2013-09-26	West Garfield Boulevard, Chicago, IL, USA	2
10498	2013-09-26	1300 North Laramie Avenue, Chicago, IL 60651, USA	2
10499	2013-09-26	1300 North Laramie Avenue, Chicago, IL 60651, USA	1
10500	2013-09-26	3900 North Springfield Avenue, Chicago, IL 606...	2
10501	2013-09-26	5100 West 72nd Street, Chicago, IL 60638, USA	2
10502	2013-09-26	5800 North Ridge Avenue, Chicago, IL 60660, USA	2
10503	2013-09-26	1700 North Ashland Avenue, Chicago, IL 60622, USA	2
10504	2013-09-26	7100 North Harlem Avenue, Chicago, IL 60631, USA	2
10505	2013-09-26	4200 West 65th Street, Chicago, IL 60621, USA	2

	Latitude	Longitude	AddressAccuracy	NumMosquitos	WnvPresent
0	41.954690	-87.800991	9	1	0
1	41.954690	-87.800991	9	1	0
2	41.994991	-87.769279	9	1	0
3	41.974089	-87.824812	8	1	0
4	41.974089	-87.824812	8	4	0
5	41.921600	-87.666455	8	2	0
6	41.891118	-87.654491	8	1	0
7	41.867108	-87.654224	8	1	0
8	41.867108	-87.654224	8	2	0
9	41.896282	-87.655232	8	1	0
10	41.919343	-87.694259	8	1	0
11	41.921965	-87.632085	8	2	0
12	41.921965	-87.632085	8	3	0
13	41.688324	-87.676709	8	1	0
14	41.688324	-87.676709	8	1	0
15	41.862292	-87.648860	8	1	0
16	41.720848	-87.666014	9	3	0
17	41.731922	-87.677512	8	5	0
18	41.731922	-87.677512	8	1	0
19	41.891126	-87.611560	5	1	0
20	41.891126	-87.611560	5	2	0
21	41.999129	-87.795585	8	1	0
22	42.017430	-87.687769	8	1	0
23	41.907645	-87.760886	8	1	0
24	41.732984	-87.649642	8	1	0
25	41.954690	-87.800991	9	3	0
26	41.954690	-87.800991	9	5	0
27	41.954690	-87.800991	9	1	0
28	41.974089	-87.824812	8	1	0
29	41.974089	-87.824812	8	2	0
...	...	...	...	...	...
10476	41.750498	-87.605294	5	2	0
10477	41.750498	-87.605294	5	1	0
10478	41.750498	-87.605294	5	1	0

```
10479  41.984809 -87.728492            8             5            0
10480  41.659112 -87.538693            8             5            0
10481  41.659112 -87.538693            8             5            0
10482  41.659112 -87.538693            8             4            0
10483  41.992478 -87.862995            8             1            0
10484  41.974689 -87.890615            9            39            1
10485  41.974689 -87.890615            9             4            0
10486  41.974689 -87.890615            9            16            0
10487  41.974689 -87.890615            9             9            0
10488  41.974689 -87.890615            9            11            0
10489  41.974689 -87.890615            9             1            0
10490  41.925198 -87.746381            8             1            0
10491  41.973845 -87.805059            9            11            0
10492  41.973845 -87.805059            9             1            0
10493  41.743402 -87.731435            8             3            0
10494  41.728495 -87.600963            5             7            0
10495  41.728495 -87.600963            5             1            0
10496  41.947227 -87.671457            9             3            0
10497  41.793818 -87.654234            5             8            0
10498  41.904194 -87.756155            9            13            0
10499  41.904194 -87.756155            9             5            0
10500  41.951866 -87.725057            8             3            0
10501  41.763733 -87.742302            8             6            1
10502  41.987280 -87.666066            8             5            0
10503  41.912563 -87.668055            9             1            0
10504  42.009876 -87.807277            9             5            0
10505  41.776428 -87.627096            8             1            0

[10506 rows x 8 columns]
```

1.3 Logestic Regression

In [16]:
```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10)

train = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fa
train =train.drop(train.columns[3:7], axis=1)

features_categorical = ["Species"]
for c in features_categorical:
    train[c] = pd.Categorical(train[c]).codes

X = train.iloc[:,2:7].values
Y = train.iloc[:,7].values
```

In [41]:
```
train
```

Out[41]:
```
array([    0,     1,     2, ..., 10099, 10114, 10117], dtype=int64)
```

In [17]:
```
from sklearn.linear_model import LogisticRegression
clf_Log = LogisticRegression(solver='liblinear', max_iter=100,
                             random_state=42,verbose=2,class_weight='balanced')

for train_indices, test_indices in kf.split(X):
    clf_Log.fit(X[train_indices], Y[train_indices])
    print(clf_Log.score(X[test_indices], Y[test_indices]))
```

```
[LibLinear]0.8049476688867745
[LibLinear]0.5480494766888677
[LibLinear]0.5927687916270219
[LibLinear]0.7516650808753568
[LibLinear]0.7573739295908658
[LibLinear]0.8829686013320647
[LibLinear]0.7761904761904762
[LibLinear]0.8961904761904762
[LibLinear]0.6971428571428572
[LibLinear]0.680952380952381
```

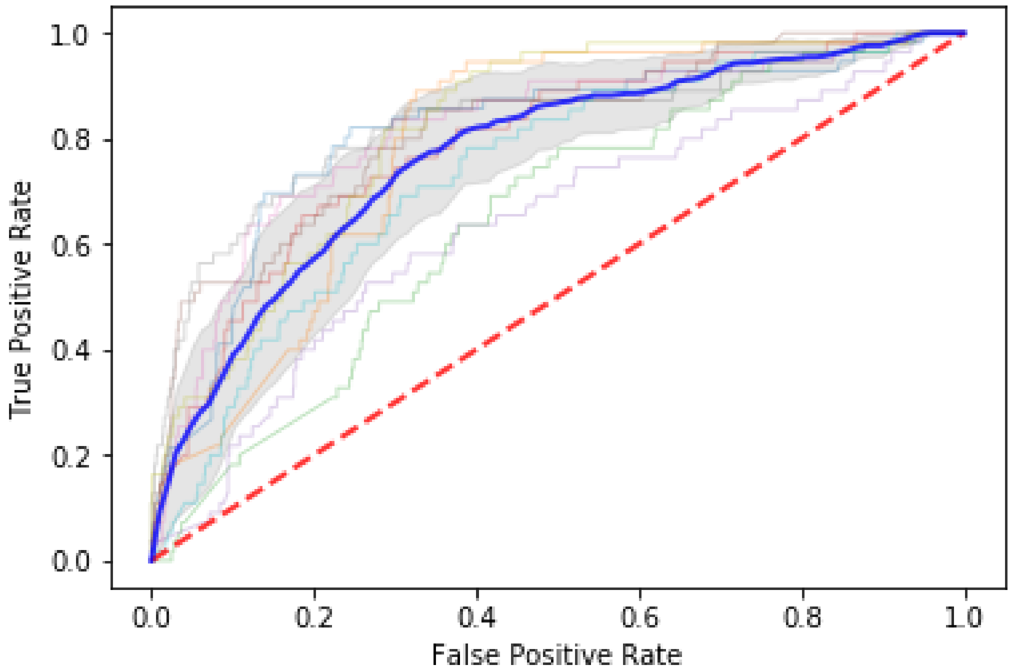
In [19]:
```
for train_indices, test_indices in kf.split(X):
    clf_Log.fit(X[train_indices], Y[train_indices])
print(clf_Log.score(X[train_indices], Y[train_indices]))
```

```
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```

In [30]:
```
print(__doc__)

import numpy as np
from scipy import interp
import matplotlib.pyplot as plt

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold


cv = StratifiedKFold(n_splits=10)
classifier = LogisticRegression(solver='liblinear', max_iter=100,
                             random_state=42,verbose=2,class_weight='balanced')

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
```

```python
        # Compute ROC curve and area the curve
        fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
        tprs.append(interp(mean_fpr, fpr, tpr))
        tprs[-1][0] = 0.0
        roc_auc = auc(fpr, tpr)
        aucs.append(roc_auc)
        plt.plot(fpr, tpr, lw=1, alpha=0.3,
                 label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

        i += 1
    plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
             label='Chance', alpha=.8)

    mean_tpr = np.mean(tprs, axis=0)
    mean_tpr[-1] = 1.0
    mean_auc = auc(mean_fpr, mean_tpr)
    std_auc = np.std(aucs)
    plt.plot(mean_fpr, mean_tpr, color='b',
             label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
             lw=2, alpha=.8)

    std_tpr = np.std(tprs, axis=0)
    tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
    tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
    plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                     label=r'$\pm$ 1 std. dev.')

    plt.xlim([-0.05, 1.05])
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.show()

    print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
Automatically created module for IPython interactive environment
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```




```
Mean ROC (AUC = 0.77 $\pm$ 0.07)
```

Each ROC curve will have 11 curves, with the dark blue being the average of the other 10 curves from the 10-fold cross validation. AUC score is also averaged over this 10-fold cross validation.

For this data set in particular it is important to look at the AUC curve since the data is heavily skewed to one direction. So if the accuracy is 95%, but the AUC is 0.40 it most likely means the model is really good at predicting true negatives, but not so good at predicting true positives.

1.4 Decision Tree

```
In [21]: from sklearn.tree import DecisionTreeClassifier
         clf_tree = DecisionTreeClassifier(random_state=10)

         for train_indices, test_indices in kf.split(X):
             clf_tree.fit(X[train_indices], Y[train_indices])
             print(clf_tree.score(X[test_indices], Y[test_indices]))
```

```
0.9600380589914367
0.8810656517602283
0.8591817316841104
0.9638439581351094
0.9590865842055185
0.9743101807802094
```

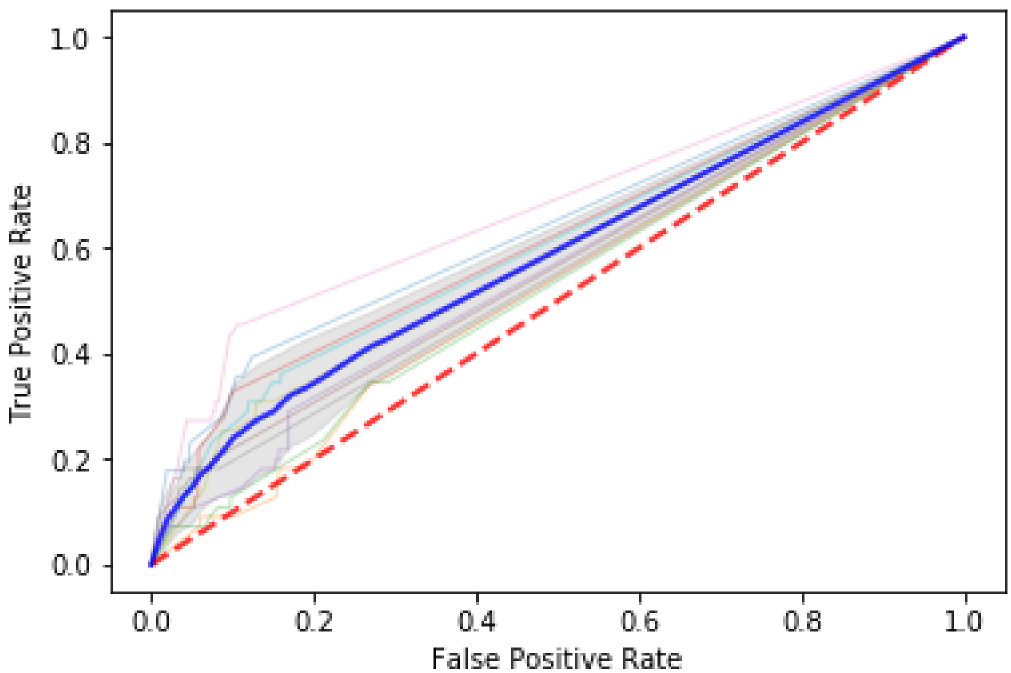
```
0.9495238095238095
0.96
0.9095238095238095
0.819047619047619
```

```
In [22]: print(clf_tree.score(X[train_indices], Y[train_indices]))
```

```
0.9773688663282571
```

```
In [29]: cv = StratifiedKFold(n_splits=10)
         classifier = DecisionTreeClassifier(random_state=10)

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)

         mean_tpr = np.mean(tprs, axis=0)
         mean_tpr[-1] = 1.0
         mean_auc = auc(mean_fpr, mean_tpr)
         std_auc = np.std(aucs)
         plt.plot(mean_fpr, mean_tpr, color='b',
                  label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
                  lw=2, alpha=.8)

         std_tpr = np.std(tprs, axis=0)
         tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
         tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
         plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                          label=r'$\pm$ 1 std. dev.')
```

```
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.59 $\pm$ 0.05)
```

1.5 Naive Bayes

```
In [23]: from sklearn.naive_bayes import GaussianNB, BernoulliNB
         clf_NB = GaussianNB()

         for train_indices, test_indices in kf.split(X):
             clf_NB.fit(X[train_indices], Y[train_indices])
             print(clf_NB.score(X[test_indices], Y[test_indices]))
```

```
0.9828734538534729
0.928639391056137
0.8705994291151284
0.994291151284491
0.9257849666983825
```

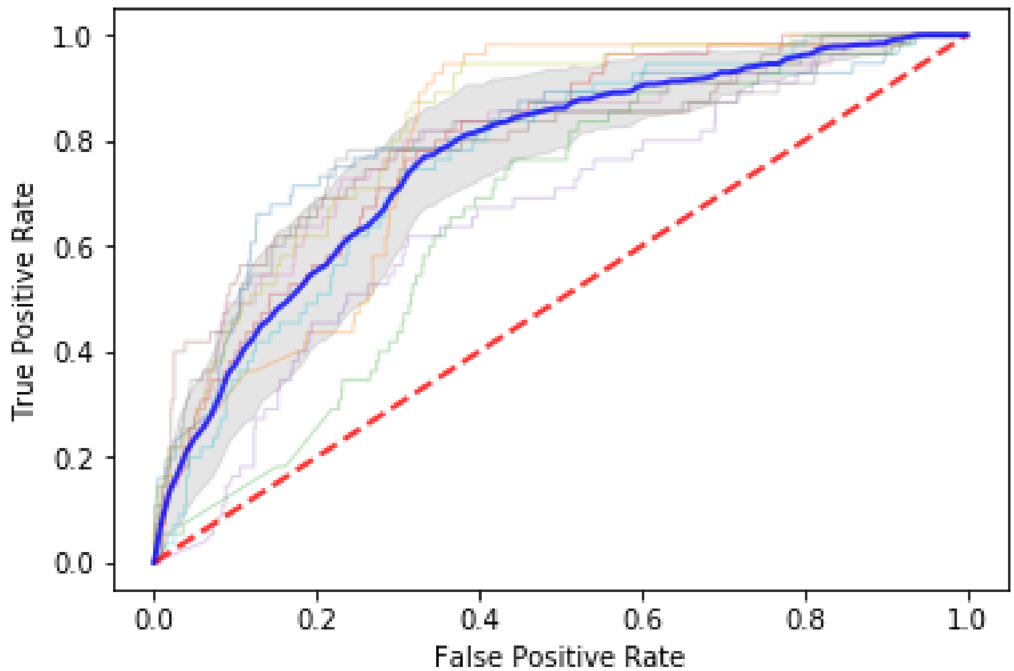
```
0.9857278782112274
0.9723809523809523
0.9714285714285714
0.9352380952380952
0.8333333333333334
```

```
In [24]:   print(clf_NB.score(X[train_indices], Y[train_indices]))
```

```
0.9602368866328257
```

```
In [28]: cv = StratifiedKFold(n_splits=10)
         classifier = GaussianNB()

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)

         mean_tpr = np.mean(tprs, axis=0)
         mean_tpr[-1] = 1.0
         mean_auc = auc(mean_fpr, mean_tpr)
         std_auc = np.std(aucs)
         plt.plot(mean_fpr, mean_tpr, color='b',
                  label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
                  lw=2, alpha=.8)

         std_tpr = np.std(tprs, axis=0)
         tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
         tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
         plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                          label=r'$\pm$ 1 std. dev.')
```

```
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.77 $\pm$ 0.06)
```

1.6 SVM

```
In [25]: from sklearn.svm import SVC
         clf_SVM = SVC()

         for train_indices, test_indices in kf.split(X):
             clf_SVM.fit(X[train_indices], Y[train_indices])
             print(clf_SVM.score(X[test_indices], Y[test_indices]))
```

```
0.9838249286393911
0.9257849666983825
0.8715509039010466
0.994291151284491
```

```
0.9952426260704091
0.9866793529971456
0.9742857142857143
0.9714285714285714
0.939047619047619
0.8333333333333334
```

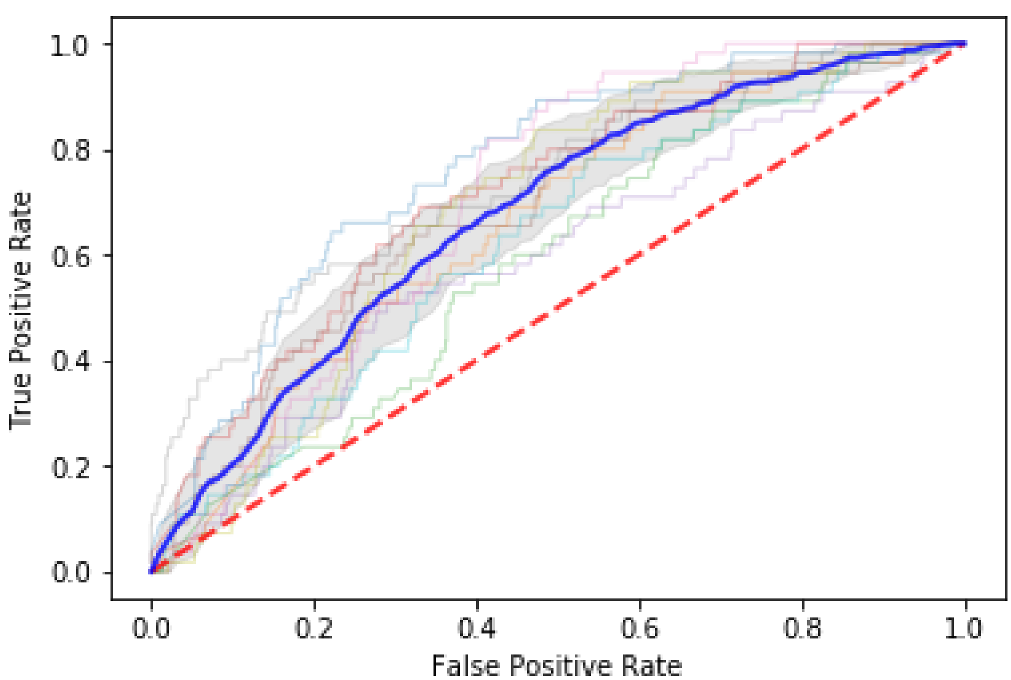
```
In [26]: print(clf_SVM.score(X[train_indices], Y[train_indices]))
```

```
0.9602368866328257
```

```
In [31]: print(__doc__)

         import numpy as np
         from scipy import interp
         import matplotlib.pyplot as plt

         from sklearn import svm, datasets
         from sklearn.metrics import roc_curve, auc
         from sklearn.model_selection import StratifiedKFold


         cv = StratifiedKFold(n_splits=10)
         classifier = svm.SVC(kernel='linear', probability=True,
                              random_state=10)

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)
```

```python
mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
Automatically created module for IPython interactive environment
```



```
Mean ROC (AUC = 0.68 $\pm$ 0.06)
```

1.7 Random Forest

```
In [27]: from sklearn.ensemble import RandomForestClassifier
         clf_RM = RandomForestClassifier(n_estimators = 100, criterion='entropy', random_state=
         for train_indices, test_indices in kf.split(X):
             clf_RM.fit(X[train_indices], Y[train_indices])
             print(clf_RM.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: Deprecation
  from numpy.core.umath_tests import inner1d
```

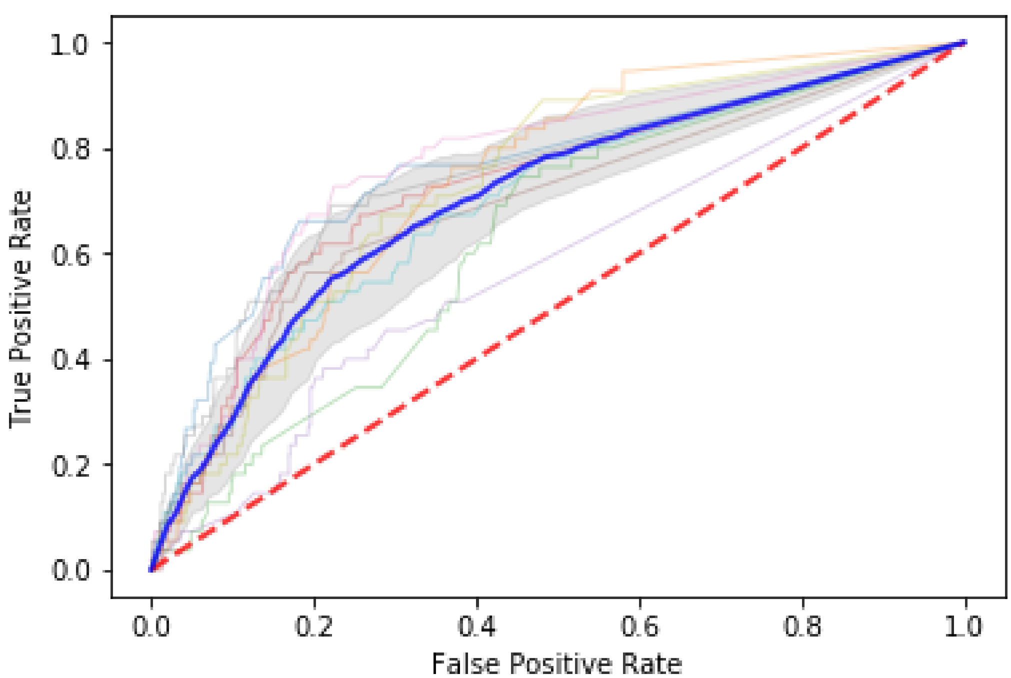
```
0.9705042816365367
0.8972407231208372
0.8658420551855376
0.9676498572787822
0.9714557564224549
0.9752616555661275
0.9485714285714286
0.9647619047619047
0.9266666666666666
0.8314285714285714
```

```
In [28]: print(clf_RM.score(X[train_indices], Y[train_indices]))
```

```
0.9773688663282571
```

```
In [36]: cv = StratifiedKFold(n_splits=10)
         classifier = RandomForestClassifier(n_estimators = 100, criterion='entropy', random_s

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
```

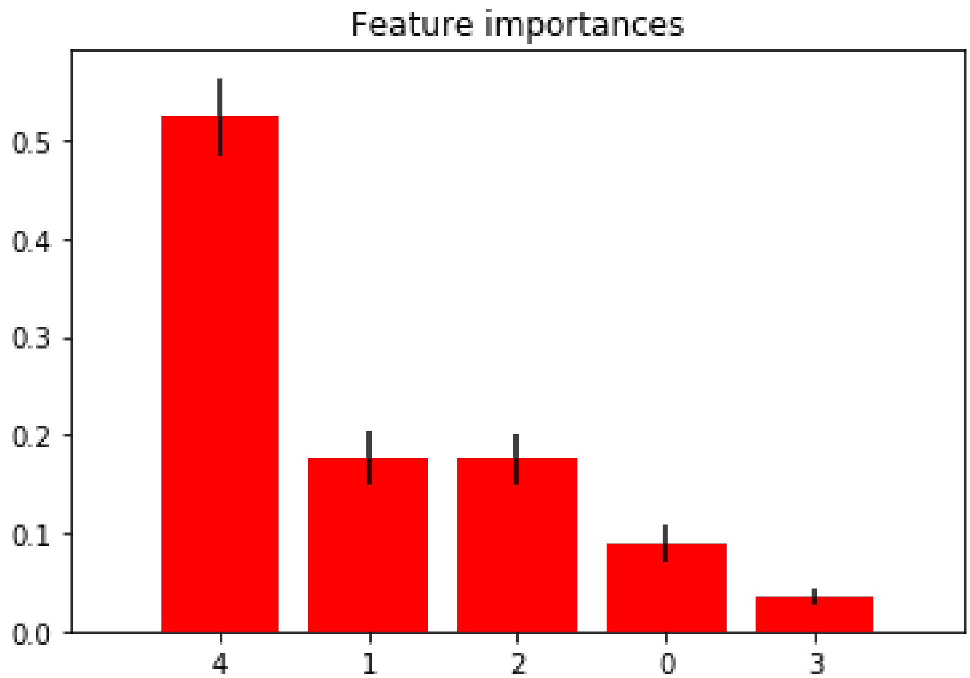
```
                label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```



```
Mean ROC (AUC = 0.71 $\pm$ 0.06)
```

```
In [35]: importances = clf_RM.feature_importances_
         std = np.std([tree.feature_importances_ for tree in clf_RM.estimators_],
                      axis=0)
         indices = np.argsort(importances)[::-1]

         # Print the feature ranking
         print("Feature ranking:")

         for f in range(X.shape[1]):
             print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

         # Plot the feature importances of the forest
         plt.figure()
         plt.title("Feature importances")
         plt.bar(range(X.shape[1]), importances[indices],
                color="r", yerr=std[indices], align="center")
         plt.xticks(range(X.shape[1]), indices)
         plt.xlim([-1, X.shape[1]])
         plt.show()
         col = list(X)
         print("\n Lowest scores:", col[10],col[20],col[21])
```

```
Feature ranking:
1. feature 4 (0.524265)
2. feature 1 (0.177247)
3. feature 2 (0.175415)
4. feature 0 (0.088704)
5. feature 3 (0.034370)
```




```
Lowest scores: [  2.        41.919343 -87.694259   8.          1.      ] [  3.        41.89112
```

Based on this feature importance from random forrest models it appears theat feature 4, number of mosquitoes collected in a trap, is very important >50% in the model. However I think this feature may be biased in the model, the more mosquitoes collected in a trap the more likely it is there will be WNV present. I have decided to remove this feature in all further models since it doesn't hold any relavancy in truely predicting WNV.

1.8 Neural Network

```
In [14]: from sklearn import neural_network

         nns=neural_network.MLPClassifier(hidden_layer_sizes=(150, ),
                                          activation='relu', solver='adam', alpha=0.0001,
                                          batch_size='auto', learning_rate='constant',
                                          learning_rate_init=0.01, power_t=0.5, max_iter=2000,
                                          shuffle=True, random_state=None, tol=0.0001, verbose
                                          warm_start=False, momentum=0.9, nesterovs_momentum=T
                                          early_stopping=False, validation_fraction=0.1, beta_
                                          beta_2=0.999, epsilon=1e-08)

         for train_indices, test_indices in kf.split(X):
             nns.fit(X[train_indices], Y[train_indices])
             print(nns.score(X[test_indices], Y[test_indices]))
```

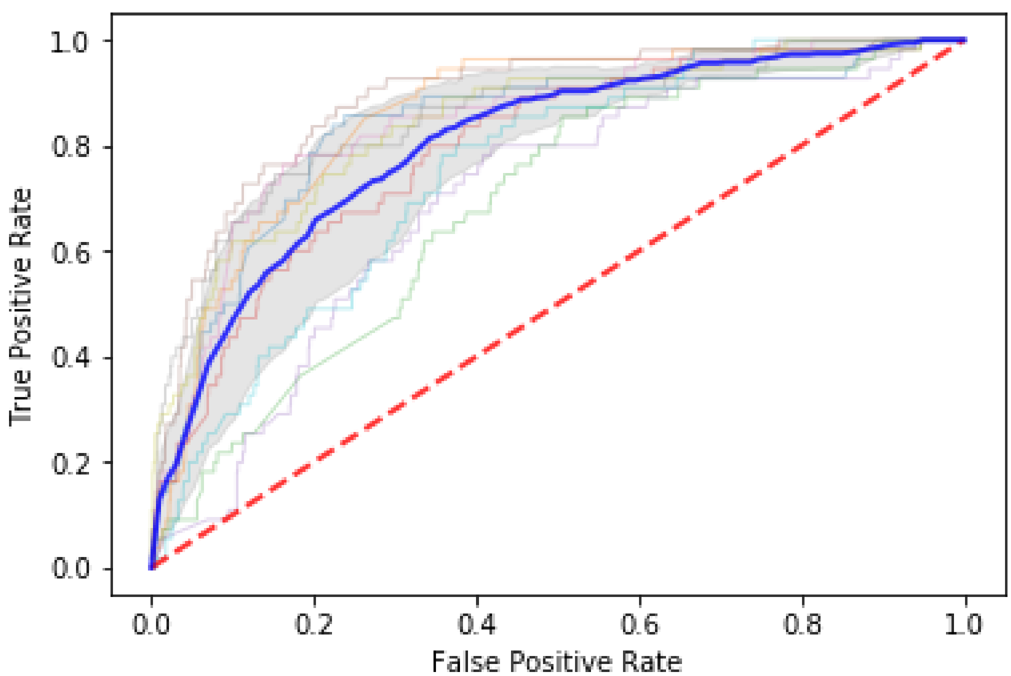
```
NameErrorTraceback (most recent call last)

<ipython-input-14-3e91ed11f891> in <module>()
 10                                   beta_2=0.999, epsilon=1e-08)
 11
---> 12 for train_indices, test_indices in kf.split(X):
 13     nns.fit(X[train_indices], Y[train_indices])
 14     print(nns.score(X[test_indices], Y[test_indices]))


NameError: name 'kf' is not defined
```

In [30]:
```
print(nns.score(X[train_indices], Y[train_indices]))
```

```
0.9602368866328257
```

In [44]:
```
cv = StratifiedKFold(n_splits=10)
classifier = neural_network.MLPClassifier(hidden_layer_sizes=(150, ),
                                    activation='relu', solver='adam', alpha=0.0001,
                                    batch_size='auto', learning_rate='constant',
                                    learning_rate_init=0.01, power_t=0.5, max_iter=2000,
                                    shuffle=True, random_state=None, tol=0.0001, verbose=
                                    warm_start=False, momentum=0.9, nesterovs_momentum=T
                                    early_stopping=False, validation_fraction=0.1, beta_
                                    beta_2=0.999, epsilon=1e-08)

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
```

```python
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```



```
Mean ROC (AUC = 0.80 $\pm$ 0.07)
```

For my initial simple models the accuracy score for most them were above 90%, with AUC scores >70. However as mentioned above I think one of the the features needs to be removed from the models.

Next, I plan to add in all the weather data and see how the model works with that. I think there will need to be some feature selection to optimize the model. There are a few features that does not have any importance in disease transmission, such as lake depth and water temperature, that may not be helpful in fitting a model.

2 Adding weather data to the models

```
In [25]: train = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
         test = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
         weather = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/F
```

```
In [26]: weather_stn1 = weather[weather['Station']==1]
         weather_stn2 = weather[weather['Station']==2]
         weather_stn1 = weather_stn1.drop('Station', axis=1)
         weather_stn2 = weather_stn2.drop('Station', axis=1)
         weather = weather_stn1.merge(weather_stn2, on='Date')
```

```
In [27]: weather = weather.replace('M', -1)
         weather = weather.replace('-', -1)
         weather = weather.replace('T', -1)
         weather = weather.replace(' T', -1)
         weather = weather.replace('  T', -1)
```

```
In [28]: def create_month(x):
             return x.split('-')[1]

         def create_day(x):
             return x.split('-')[2]

         train['month'] = train.Date.apply(create_month)
         train['day'] = train.Date.apply(create_day)
         test['month'] = test.Date.apply(create_month)
         test['day'] = test.Date.apply(create_day)
```

```
In [29]: train['Lat_int'] = train.Latitude.apply(int)
         train['Long_int'] = train.Longitude.apply(int)
         test['Lat_int'] = test.Latitude.apply(int)
         test['Long_int'] = test.Longitude.apply(int)
```

```
In [30]: train = train.drop(['Address', 'AddressNumberAndStreet', 'NumMosquitos'], axis = 1)
         test = test.drop(['Id', 'Address', 'AddressNumberAndStreet'], axis = 1)
```

```
In [31]: train = train.merge(weather, on='Date')
         test = test.merge(weather, on='Date')
```

```
In [32]: train = train.drop(['Date'], axis = 1)
         test = test.drop(['Date'], axis = 1)
```

```
In [33]: from sklearn import ensemble, preprocessing
         lbl = preprocessing.LabelEncoder()
         lbl.fit(list(train['Species'].values) + list(test['Species'].values))
         train['Species'] = lbl.transform(train['Species'].values)
         test['Species'] = lbl.transform(test['Species'].values)

         lbl.fit(list(train['Street'].values) + list(test['Street'].values))
         train['Street'] = lbl.transform(train['Street'].values)
         test['Street'] = lbl.transform(test['Street'].values)

         lbl.fit(list(train['Trap'].values) + list(test['Trap'].values))
         train['Trap'] = lbl.transform(train['Trap'].values)
         test['Trap'] = lbl.transform(test['Trap'].values)
```

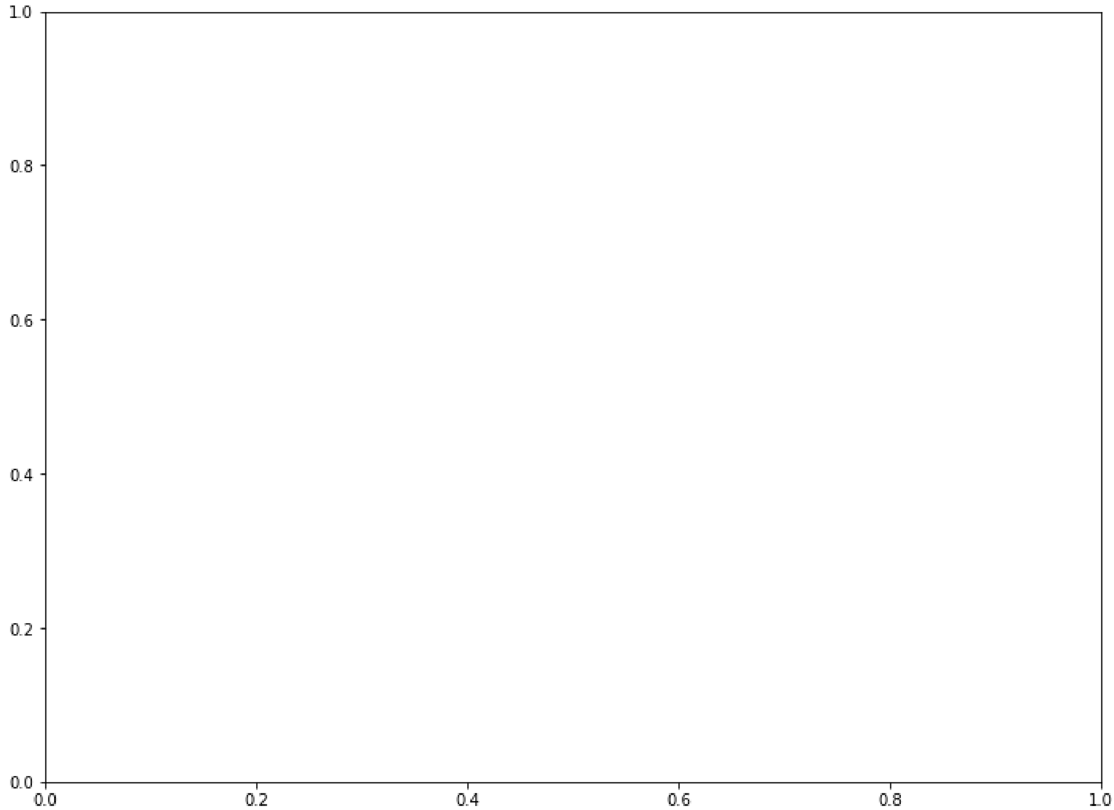
```
In [34]: train = train.ix[:,(train != -1).any(axis=0)]
         test = test.ix[:,(test != -1).any(axis=0)]
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

```
In [13]: hmap = train.corr()
         pl.subplots(figsize=(12, 9))
         sns.heatmap(hmap, vmax=.8,annot=True,cmap="BrBG", square=True);
```

```
        NameErrorTraceback (most recent call last)
```

```
    <ipython-input-13-2f89cf6a9273> in <module>()
      1 hmap = train.corr()
      2 pl.subplots(figsize=(12, 9))
----> 3 sns.heatmap(hmap, vmax=.8,annot=True,cmap="BrBG", square=True);


    NameError: name 'sns' is not defined
```



In [23]:
```
list(train.columns.values)
```

Out[23]:
```
['Species',
 'Block',
 'Street',
 'Trap',
 'Latitude',
 'Longitude',
 'AddressAccuracy',
 'WnvPresent',
 'month',
 'day',
```

```
 'Lat_int',
 'Long_int',
 'Tmax_x',
 'Tmin_x',
 'Tavg_x',
 'Depart_x',
 'DewPoint_x',
 'WetBulb_x',
 'Heat_x',
 'Cool_x',
 'Sunrise_x',
 'Sunset_x',
 'CodeSum_x',
 'Depth_x',
 'SnowFall_x',
 'PrecipTotal_x',
 'StnPressure_x',
 'SeaLevel_x',
 'ResultSpeed_x',
 'ResultDir_x',
 'AvgSpeed_x',
 'Tmax_y',
 'Tmin_y',
 'Tavg_y',
 'DewPoint_y',
 'WetBulb_y',
 'Heat_y',
 'Cool_y',
 'CodeSum_y',
 'PrecipTotal_y',
 'StnPressure_y',
 'SeaLevel_y',
 'ResultSpeed_y',
 'ResultDir_y',
 'AvgSpeed_y']
```

In [20]:
```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10)



for n,v in train.items():
    if v.dtype == "object":
        train[n] = v.factorize()[0]


test = train[['WnvPresent']]
train = train.drop(['WnvPresent'], axis = 1)
```

```
X = train.iloc[:,:].values
Y = test.iloc[:,:].values
```

3 Feature Selection

3.1 Random Forest

```
In [21]: from sklearn.ensemble import RandomForestClassifier
         clf_RM = RandomForestClassifier(n_estimators = 100, criterion='entropy', random_state=
         for train_indices, test_indices in kf.split(X):
             clf_RM.fit(X[train_indices], Y[train_indices])
             print(clf_RM.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.


0.9838249286393911


C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.


0.9257849666983825


C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.


0.8715509039010466


C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.


0.994291151284491


C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.


0.9952426260704091
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.9866793529971456
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.9742857142857143
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.9714285714285714
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.9419047619047619
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.8333333333333334
```

In [43]:
```
print(clf_RM.score(X[train_indices], Y[train_indices]))
```

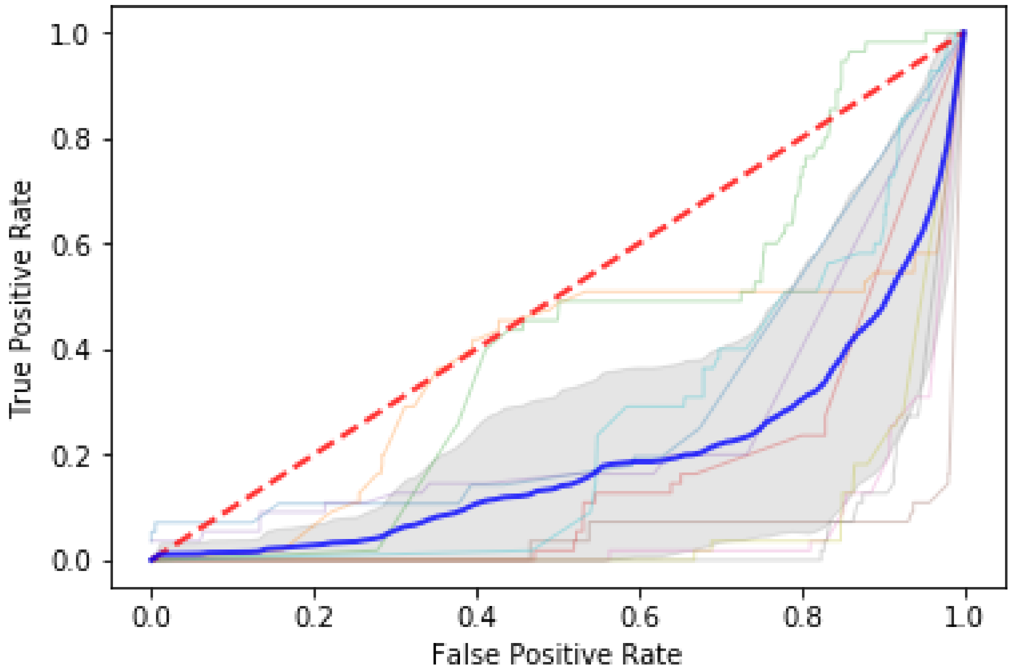
```
0.9831852791878173
```

In [22]:
```
print(__doc__)

import numpy as np
from scipy import interp
import matplotlib.pyplot as plt

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold
```

```python
cv = StratifiedKFold(n_splits=10)
classifier = RandomForestClassifier(n_estimators = 100, criterion='entropy', random_s

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()


print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
Automatically created module for IPython interactive environment
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:21: DataConversionWarning: A c
```



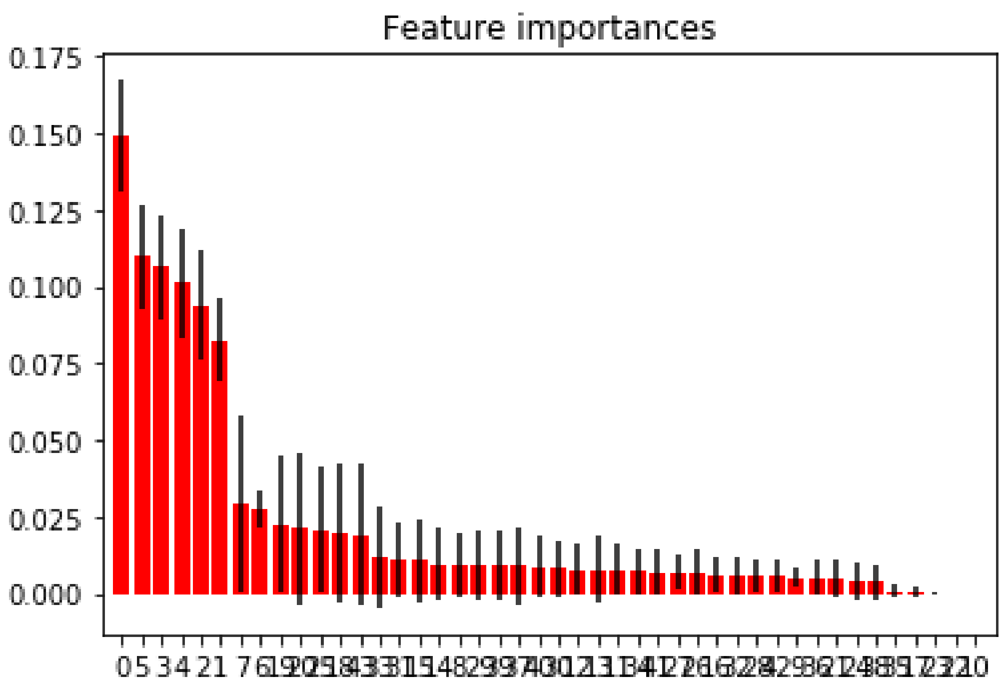

```
Mean ROC (AUC = 0.19 $\pm$ 0.13)
```

In [61]:
```
importances = clf_RM.feature_importances_
std = np.std([tree.feature_importances_ for tree in clf_RM.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")
```

```python
    for f in range(X.shape[1]):
        print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

    # Plot the feature importances of the forest
    plt.figure()
    plt.title("Feature importances")
    plt.bar(range(X.shape[1]), importances[indices],
           color="r", yerr=std[indices], align="center")
    plt.xticks(range(X.shape[1]), indices)
    plt.xlim([-1, X.shape[1]])
    plt.show()
    col = list(X)
    print("\n Lowest scores:", col[10],col[20],col[21])
```

```
Feature ranking:
1. feature 0 (0.149209)
2. feature 5 (0.109899)
3. feature 3 (0.106528)
4. feature 4 (0.101055)
5. feature 2 (0.094005)
6. feature 1 (0.082675)
7. feature 7 (0.029198)
8. feature 6 (0.027509)
9. feature 19 (0.022585)
10. feature 20 (0.021180)
11. feature 25 (0.020829)
12. feature 18 (0.019897)
13. feature 43 (0.018959)
14. feature 33 (0.011953)
15. feature 31 (0.010735)
16. feature 15 (0.010704)
17. feature 14 (0.009575)
18. feature 8 (0.009394)
19. feature 29 (0.009330)
20. feature 39 (0.009260)
21. feature 37 (0.008953)
22. feature 40 (0.008713)
23. feature 30 (0.008046)
24. feature 12 (0.007914)
25. feature 13 (0.007879)
26. feature 11 (0.007738)
27. feature 34 (0.007226)
28. feature 41 (0.007060)
29. feature 27 (0.006996)
30. feature 26 (0.006994)
31. feature 16 (0.005877)
32. feature 32 (0.005826)
33. feature 28 (0.005820)
```

```
34. feature 42 (0.005702)
35. feature 9 (0.005294)
36. feature 36 (0.005053)
37. feature 21 (0.004749)
38. feature 24 (0.004271)
39. feature 38 (0.004000)
40. feature 35 (0.000810)
41. feature 17 (0.000585)
42. feature 23 (0.000014)
43. feature 22 (0.000000)
44. feature 10 (0.000000)
```




```
Lowest scores: [  2.          21.          45.          39.          41.919343 -87.694259
   8.          0.          0.         41.         -87.         88.
  60.          0.          0.         58.           0.          0.
   0.          0.          0.          0.           0.          0.
   0.          0.          0.          5.8         18.          0.
  88.         65.          0.         59.           0.          0.
   0.          0.          0.          0.           0.          5.8
  16.          0.       ] [  3.          53.          46.          91.          41.891126 -87.61156
   5.          0.          0.         41.         -87.         88.
  60.          0.          0.         58.           0.          0.
   0.          0.          0.          0.           0.          0.
```

```
  0.          0.          0.          5.8        18.          0.
 88.         65.          0.         59.          0.          0.
  0.          0.          0.          0.          0.          5.8
 16.          0.       ] [  2.        65.        36.         96.         41.999129 -87.795585
  8.          0.          0.         41.        -87.         88.
 60.          0.          0.         58.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          5.8        18.          0.
 88.         65.          0.         59.          0.          0.
  0.          0.          0.          0.          0.          5.8
 16.          0.       ]
```

So from this feature importance ranking it appears that the species of mosqiuto is the most important feature, which I hypothesized from the above figures showing that some species did not carry WNV in any of the years in the data set. Next, there are some weather related features such as temperature min and max, along with longitude and latitude. When I optimize the model I will select the top 15 features.

Let's continue with the current set of features

3.2 Logistic Regression

```
In [44]: for train_indices, test_indices in kf.split(X):
             clf_Log.fit(X[train_indices], Y[train_indices])
             print(clf_Log.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


[LibLinear]0.8981921979067554
[LibLinear]0.6898192197906755
[LibLinear]0.4719314938154139
[LibLinear]0.45670789724072314
[LibLinear]0.9181731684110371
[LibLinear]0.7411988582302569
[LibLinear]0.6619047619047619
[LibLinear]0.5961904761904762
[LibLinear]0.7790476190476191
[LibLinear]0.4076190476190476
```

```
In [45]: print(clf_Log.score(X[train_indices], Y[train_indices]))
```

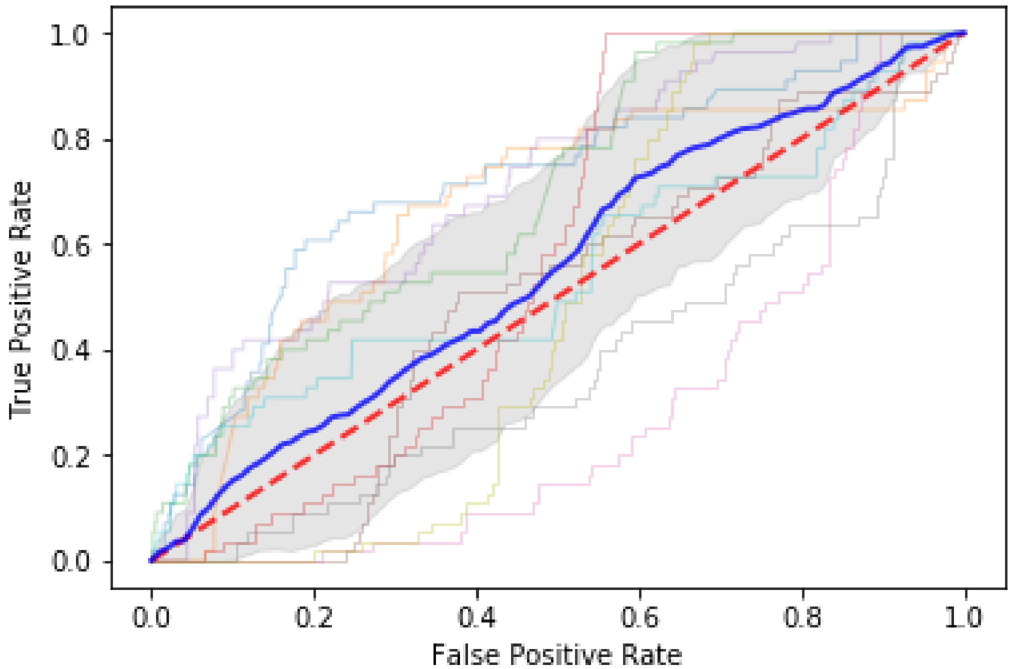
```
0.705160744500846
```

```
In [72]: print(__doc__)
```

```python
import numpy as np
from scipy import interp
import matplotlib.pyplot as plt

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold


cv = StratifiedKFold(n_splits=10)
classifier = LogisticRegression(solver='liblinear', max_iter=100,
                             random_state=42,verbose=2,class_weight='balanced')

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
```

```
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()


print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
Automatically created module for IPython interactive environment
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```




```
Mean ROC (AUC = 0.56 $\pm$ 0.14)
```

3.3 Neural Network

```
In [46]: for train_indices, test_indices in kf.split(X):
             nns.fit(X[train_indices], Y[train_indices])
             print(nns.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:912
  y = column_or_1d(y, warn=True)


0.9838249286393911
0.9257849666983825
```

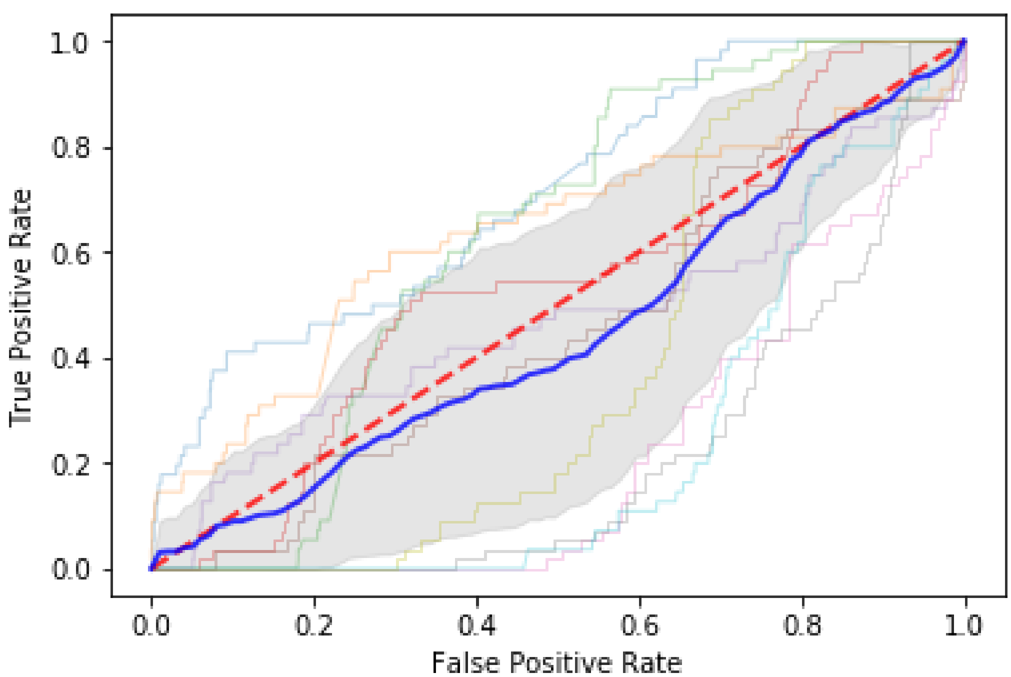
```
0.8715509039010466
0.7516650808753568
0.9952426260704091
0.9571836346336822
0.9733333333333334
0.9714285714285714
0.939047619047619
0.8333333333333334
```

```
In [47]: print(nns.score(X[train_indices], Y[train_indices]))
```

```
0.9602368866328257
```

```
In [73]: cv = StratifiedKFold(n_splits=10)
         classifier = neural_network.MLPClassifier(hidden_layer_sizes=(150, ),
                                          activation='relu', solver='adam', alpha=0.0001,
                                          batch_size='auto', learning_rate='constant',
                                          learning_rate_init=0.01, power_t=0.5, max_iter=2000,
                                          shuffle=True, random_state=None, tol=0.0001, verbose=
                                          warm_start=False, momentum=0.9, nesterovs_momentum=T
                                          early_stopping=False, validation_fraction=0.1, beta_
                                          beta_2=0.999, epsilon=1e-08)


         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)

         mean_tpr = np.mean(tprs, axis=0)
         mean_tpr[-1] = 1.0
         mean_auc = auc(mean_fpr, mean_tpr)
```

```
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()


print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.45 $\pm$ 0.17)
```

3.4 D Tree

In [48]:
```
from sklearn.tree import DecisionTreeClassifier
clf_tree = DecisionTreeClassifier(random_state=10)

for train_indices, test_indices in kf.split(X):
    clf_tree.fit(X[train_indices], Y[train_indices])
    print(clf_tree.score(X[test_indices], Y[test_indices]))
```

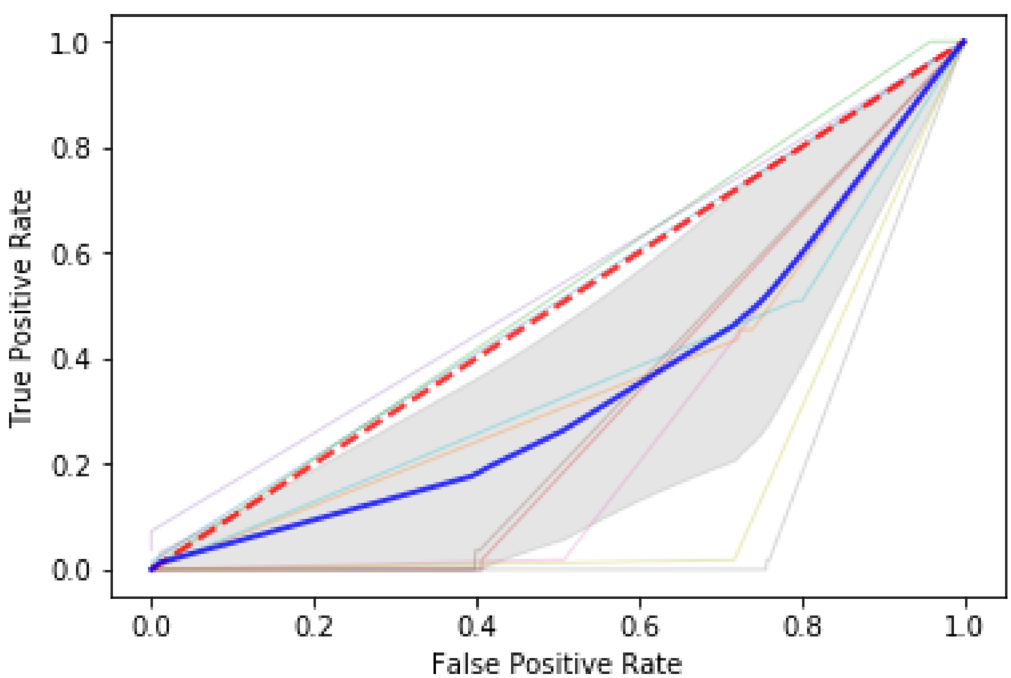
```
0.9647954329210275
0.8953377735490009
0.8068506184586108
0.9134157944814463
0.9952426260704091
0.8877259752616555
0.9371428571428572
0.9161904761904762
0.9238095238095239
0.8019047619047619
```

In [49]:
```
print(clf_tree.score(X[train_indices], Y[train_indices]))
```

```
0.9831852791878173
```

In [75]:
```
cv = StratifiedKFold(n_splits=10)
classifier = DecisionTreeClassifier(random_state=10)

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)
```

```python
mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.34 $\pm$ 0.14)
```

3.5 Naive Bayes

```
In [50]: from sklearn.naive_bayes import GaussianNB, BernoulliNB
         clf_NB = GaussianNB()

         for train_indices, test_indices in kf.split(X):
             clf_NB.fit(X[train_indices], Y[train_indices])
             print(clf_NB.score(X[test_indices], Y[test_indices]))
```

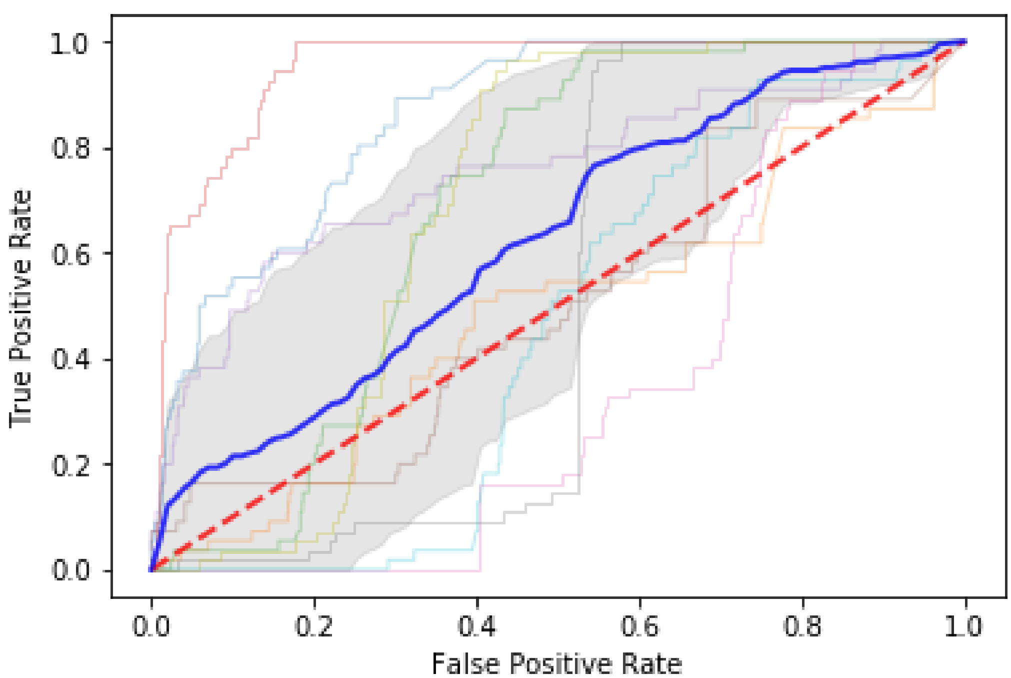
```
0.2702188392007612
0.09990485252140818
0.24643196955280686
0.4871550903901047
0.4015223596574691
0.5071360608943863
0.2276190476190476
0.5876190476190476
0.6447619047619048
0.4361904761904762
```

```
In [51]: print(clf_NB.score(X[train_indices], Y[train_indices]))
```

```
0.557741116751269
```

```
In [77]: cv = StratifiedKFold(n_splits=10)
         classifier = GaussianNB()

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)
```

```python
mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.62 $\pm$ 0.18)
```

3.6 Support Vector Machines

In [19]:
```
from sklearn.svm import SVC
clf_SVM = SVC()

for train_indices, test_indices in kf.split(X):
    clf_SVM.fit(X[train_indices], Y[train_indices])
    print(clf_SVM.score(X[test_indices], Y[test_indices]))
```

```
NameErrorTraceback (most recent call last)

<ipython-input-19-e710bf6b20fc> in <module>()
  2 clf_SVM = SVC()
  3
----> 4 for train_indices, test_indices in kf.split(X):
  5     clf_SVM.fit(X[train_indices], Y[train_indices])
  6     print(clf_SVM.score(X[test_indices], Y[test_indices]))


NameError: name 'kf' is not defined
```

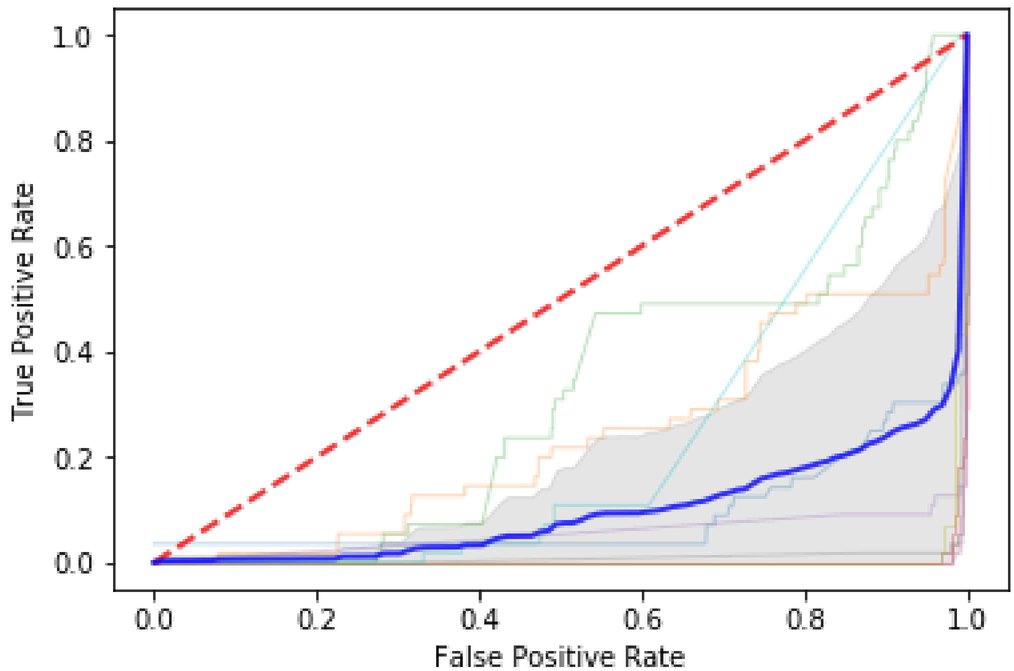
In [53]:
```
print(clf_SVM.score(X[train_indices], Y[train_indices]))
```

```
0.9677453468697124
```

In [23]:
```
cv = StratifiedKFold(n_splits=10)
classifier =  svm.SVC(kernel='rbf', probability=True,
                      random_state=10)

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))
```

```
            i += 1
        plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                 label='Chance', alpha=.8)

        mean_tpr = np.mean(tprs, axis=0)
        mean_tpr[-1] = 1.0
        mean_auc = auc(mean_fpr, mean_tpr)
        std_auc = np.std(aucs)
        plt.plot(mean_fpr, mean_tpr, color='b',
                 label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
                 lw=2, alpha=.8)

        std_tpr = np.std(tprs, axis=0)
        tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
        tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
        plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                         label=r'$\pm$ 1 std. dev.')

        plt.xlim([-0.05, 1.05])
        plt.ylim([-0.05, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.show()

        print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
```




```
Mean ROC (AUC = 0.10 $\pm$ 0.12)
```

With all the weather data added to the model all the models have preformed much worse than my simplified model. This may be due to overfitting of the model, so I will use the top 20 features from the random forrest feature selection.

3.7 Model optimization

```
In [35]: from sklearn.model_selection import KFold
         kf = KFold(n_splits=10)



         for n,v in train.items():
             if v.dtype == "object":
                 train[n] = v.factorize()[0]


         test = train[['WnvPresent']]
         train = train.drop(['WnvPresent','AddressAccuracy','Lat_int','Long_int','Sunrise_x','S

         X = train.iloc[:,:].values
         Y = test.iloc[:,:].values
```

```
In [36]: from sklearn.ensemble import RandomForestClassifier
         clf_RM = RandomForestClassifier(n_estimators = 100, criterion='entropy', random_state=
         for train_indices, test_indices in kf.split(X):
             clf_RM.fit(X[train_indices], Y[train_indices])
             print(clf_RM.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.

0.9838249286393911

C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.

0.9257849666983825

C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.

0.8705994291151284

C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.

0.994291151284491

C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.

0.9952426260704091

C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.

0.9866793529971456

C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.9742857142857143
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.9714285714285714
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.94
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A co
  after removing the cwd from sys.path.
```

```
0.8352380952380952
```

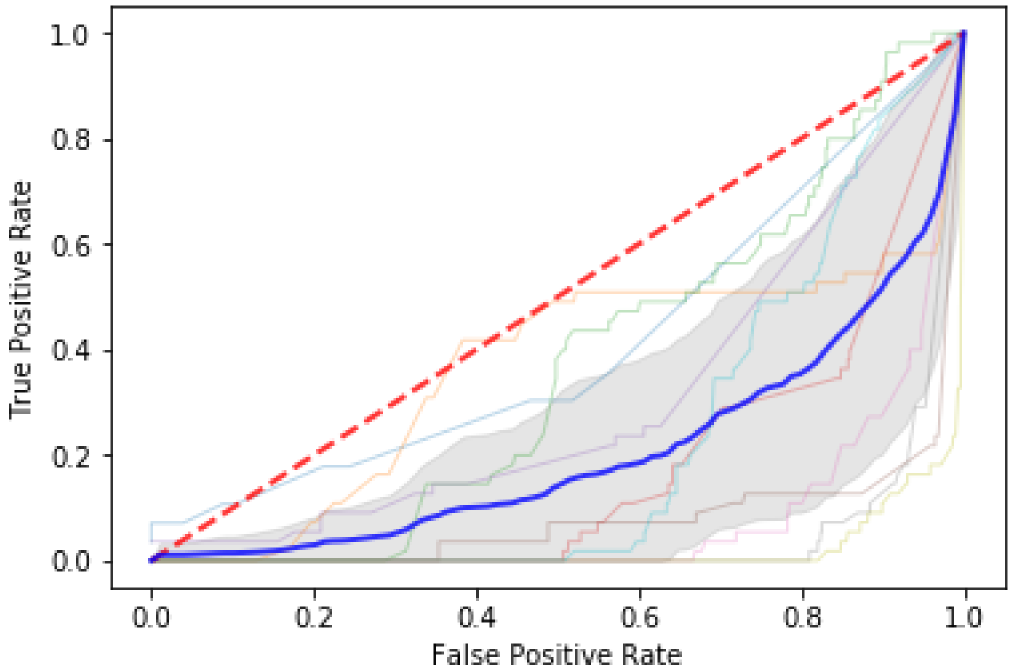
In [37]:
```
print(clf_RM.score(X[train_indices], Y[train_indices]))
```

```
0.9831852791878173
```

In [78]:
```
import numpy as np
from scipy import interp
import matplotlib.pyplot as plt

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold


cv = StratifiedKFold(n_splits=10)
classifier = RandomForestClassifier(n_estimators = 100, criterion='entropy', random_st

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
```

```
        fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
        tprs.append(interp(mean_fpr, fpr, tpr))
        tprs[-1][0] = 0.0
        roc_auc = auc(fpr, tpr)
        aucs.append(roc_auc)
        plt.plot(fpr, tpr, lw=1, alpha=0.3,
                 label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

        i += 1
    plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
             label='Chance', alpha=.8)

    mean_tpr = np.mean(tprs, axis=0)
    mean_tpr[-1] = 1.0
    mean_auc = auc(mean_fpr, mean_tpr)
    std_auc = np.std(aucs)
    plt.plot(mean_fpr, mean_tpr, color='b',
             label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
             lw=2, alpha=.8)

    std_tpr = np.std(tprs, axis=0)
    tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
    tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
    plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                     label=r'$\pm$ 1 std. dev.')

    plt.xlim([-0.05, 1.05])
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.show()


    print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

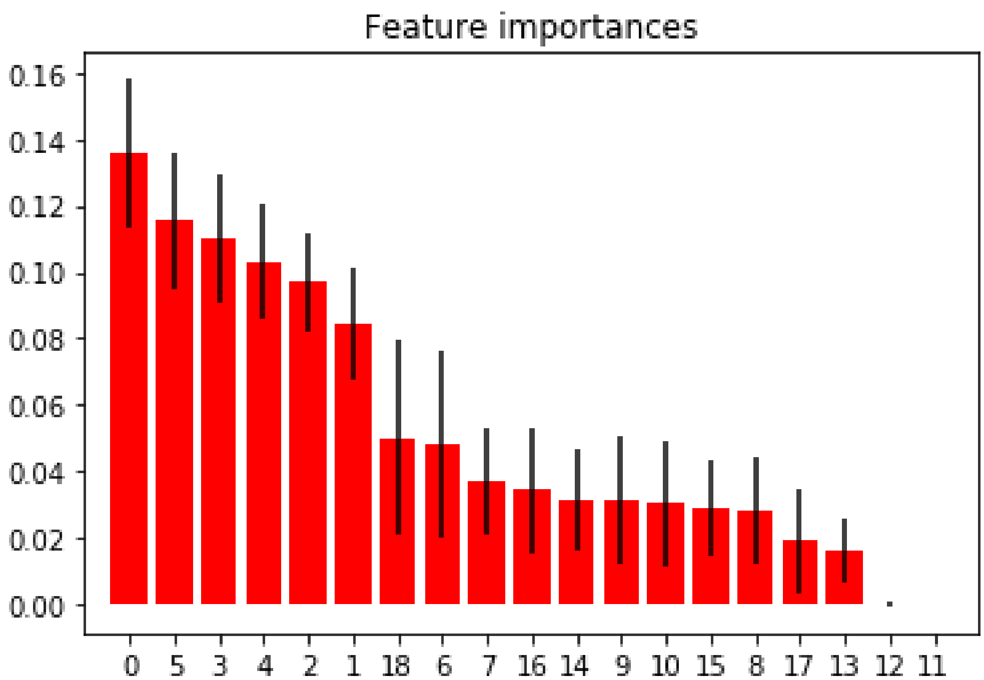
```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:19: DataConversionWarning: A c
```




```
Mean ROC (AUC = 0.21 $\pm$ 0.14)
```

```
In [18]: importances = clf_RM.feature_importances_
         std = np.std([tree.feature_importances_ for tree in clf_RM.estimators_],
                      axis=0)
         indices = np.argsort(importances)[::-1]

         # Print the feature ranking
         print("Feature ranking:")

         for f in range(X.shape[1]):
             print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

         # Plot the feature importances of the forest
         plt.figure()
         plt.title("Feature importances")
         plt.bar(range(X.shape[1]), importances[indices],
                color="r", yerr=std[indices], align="center")
         plt.xticks(range(X.shape[1]), indices)
         plt.xlim([-1, X.shape[1]])
         plt.show()
         col = list(X)
         print("\n Lowest scores:", col[10],col[20],col[21])
```

```
Feature ranking:
1. feature 0 (0.135563)
2. feature 5 (0.115265)
3. feature 3 (0.110013)
4. feature 4 (0.103012)
5. feature 2 (0.096822)
6. feature 1 (0.084450)
7. feature 18 (0.050129)
8. feature 6 (0.048161)
9. feature 7 (0.036738)
10. feature 16 (0.034494)
11. feature 14 (0.031653)
12. feature 9 (0.031196)
13. feature 10 (0.030321)
14. feature 15 (0.028859)
15. feature 8 (0.028219)
16. feature 17 (0.018877)
17. feature 13 (0.016123)
18. feature 12 (0.000104)
19. feature 11 (0.000000)
```




```
 Lowest scores: [  2.          21.          45.          39.          41.919343 -87.694259
   0.           0.          88.          60.           0.           0.
```

```
0.          0.          0.          88.         65.          0.
0.      ] [  3.          53.          46.          91.          41.891126 -87.61156
0.          0.         88.          60.          0.          0.
0.          0.          0.          88.         65.          0.
0.      ] [  2.          65.          36.          96.          41.999129 -87.795585
0.          0.         88.          60.          0.          0.
0.          0.          0.          88.         65.          0.
0.      ]
```

Species of mosquito is still the most important feature.

3.8 Logestic Regression

```
In [69]: from sklearn.linear_model import LogisticRegression
         clf_Log = LogisticRegression(solver='liblinear', max_iter=100,
                                      random_state=42,verbose=2,class_weight='balanced')

         for train_indices, test_indices in kf.split(X):
             clf_Log.fit(X[train_indices], Y[train_indices])
             print(clf_Log.score(X[test_indices], Y[test_indices]))
```

```
[LibLinear]0.7050428163653664
[LibLinear]0.25118934348239774
[LibLinear]0.44814462416745954
[LibLinear]0.49096098953377737
[LibLinear]0.8553758325404377
[LibLinear]0.5499524262607041
[LibLinear]0.7076190476190476
[LibLinear]0.6838095238095238
[LibLinear]0.9114285714285715
[LibLinear]0.42095238095238097
```

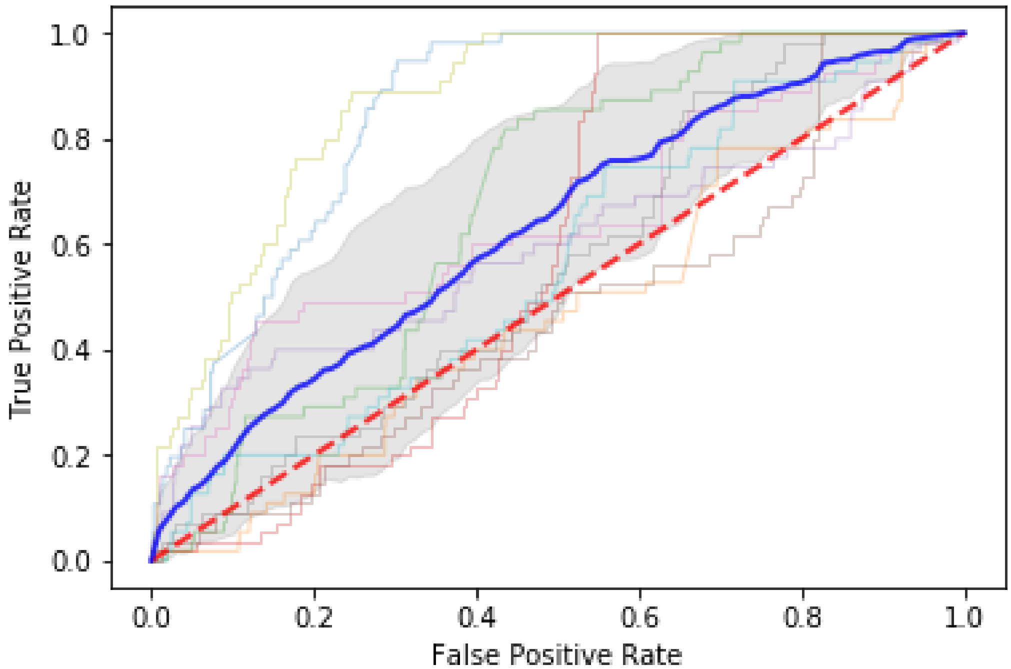
```
In [70]:  print(clf_Log.score(X[train_indices], Y[train_indices]))
```

```
0.6253172588832487
```

```
In [22]: cv = StratifiedKFold(n_splits=10)
         classifier = LogisticRegression(solver='liblinear', max_iter=100,
                                         random_state=42,verbose=2,class_weight='balanced')

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
```

```python
        probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
        # Compute ROC curve and area the curve
        fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
        tprs.append(interp(mean_fpr, fpr, tpr))
        tprs[-1][0] = 0.0
        roc_auc = auc(fpr, tpr)
        aucs.append(roc_auc)
        plt.plot(fpr, tpr, lw=1, alpha=0.3,
                 label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

        i += 1
    plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
             label='Chance', alpha=.8)

    mean_tpr = np.mean(tprs, axis=0)
    mean_tpr[-1] = 1.0
    mean_auc = auc(mean_fpr, mean_tpr)
    std_auc = np.std(aucs)
    plt.plot(mean_fpr, mean_tpr, color='b',
             label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
             lw=2, alpha=.8)

    std_tpr = np.std(tprs, axis=0)
    tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
    tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
    plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                     label=r'$\pm$ 1 std. dev.')

    plt.xlim([-0.05, 1.05])
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.show()


    print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
[LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLinear][LibLi
```




```
Mean ROC (AUC = 0.63 $\pm$ 0.13)
```

3.9 Naive Bayes

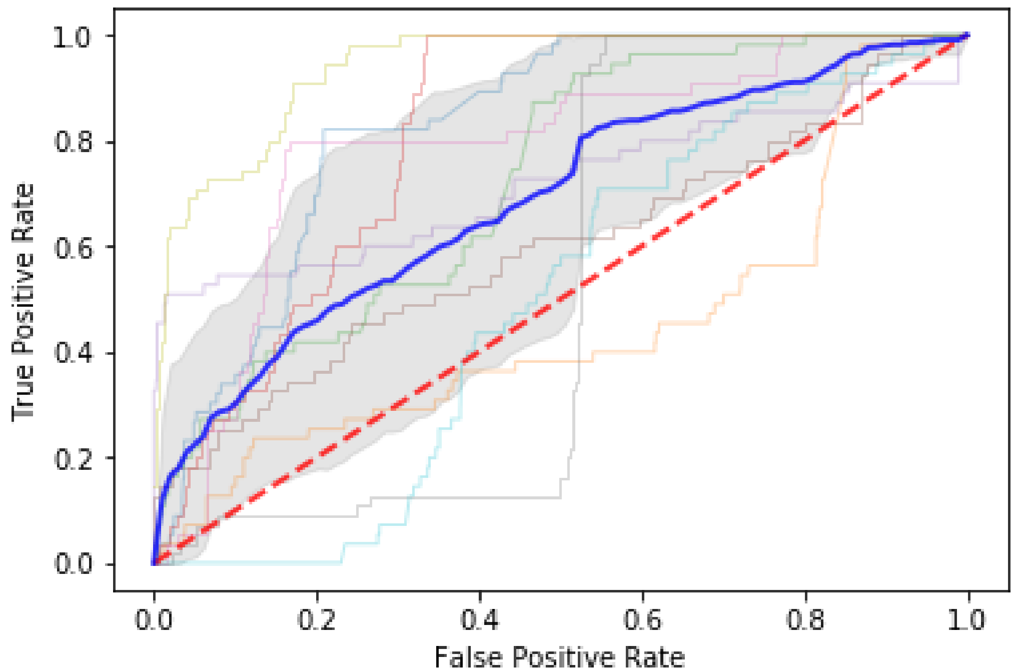
```
In [71]: from sklearn.naive_bayes import GaussianNB, BernoulliNB
         clf_NB = GaussianNB()

         for train_indices, test_indices in kf.split(X):
             clf_NB.fit(X[train_indices], Y[train_indices])
             print(clf_NB.score(X[test_indices], Y[test_indices]))
```

```
0.3139866793529971
0.1065651760228354
0.2521408182683159
0.6460513796384396
0.5375832540437678
0.5842055185537584
0.35714285714285715
0.6180952380952381
0.6533333333333333
0.5457142857142857
```

```
In [72]: print(clf_NB.score(X[train_indices], Y[train_indices]))
```

```
0.5328891708967851
```

```
In [24]: cv = StratifiedKFold(n_splits=10)
         classifier = GaussianNB()

         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)

         mean_tpr = np.mean(tprs, axis=0)
         mean_tpr[-1] = 1.0
         mean_auc = auc(mean_fpr, mean_tpr)
         std_auc = np.std(aucs)
         plt.plot(mean_fpr, mean_tpr, color='b',
                  label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
                  lw=2, alpha=.8)

         std_tpr = np.std(tprs, axis=0)
         tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
         tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
         plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                          label=r'$\pm$ 1 std. dev.')

         plt.xlim([-0.05, 1.05])
         plt.ylim([-0.05, 1.05])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.show()


         print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.69 $\pm$ 0.16)
```

3.10 Neural Network

```
In [73]: from sklearn import neural_network

         nns=neural_network.MLPClassifier(hidden_layer_sizes=(150, ),
                                          activation='relu', solver='adam', alpha=0.0001,
                                          batch_size='auto', learning_rate='constant',
                                          learning_rate_init=0.01, power_t=0.5, max_iter=2000,
                                          shuffle=True, random_state=None, tol=0.0001, verbose=
                                          warm_start=False, momentum=0.9, nesterovs_momentum=T
                                          early_stopping=False, validation_fraction=0.1, beta_
                                          beta_2=0.999, epsilon=1e-08)

         for train_indices, test_indices in kf.split(X):
             nns.fit(X[train_indices], Y[train_indices])
             print(nns.score(X[test_indices], Y[test_indices]))
```

```
0.9838249286393911
0.9257849666983825
0.8715509039010466
0.994291151284491
0.9952426260704091
```

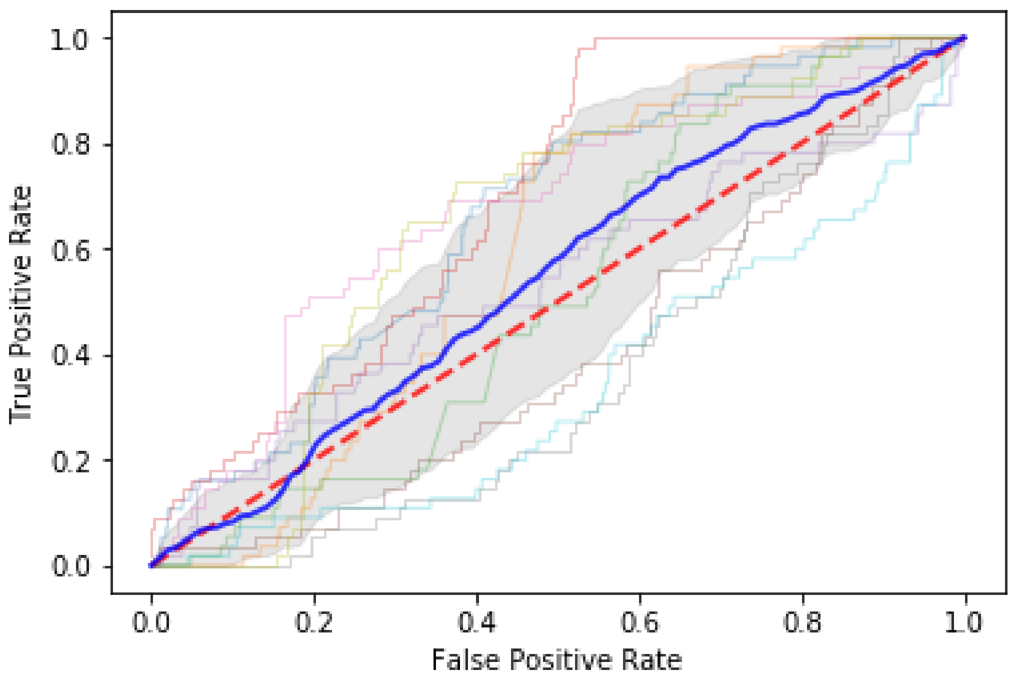
```
0.9276879162702188
0.9638095238095238
0.9714285714285714
0.939047619047619
0.8333333333333334
```

```
In [74]: print(nns.score(X[train_indices], Y[train_indices]))
```

```
0.9602368866328257
```

```
In [26]: cv = StratifiedKFold(n_splits=10)
         classifier =  neural_network.MLPClassifier(hidden_layer_sizes=(150, ),
                                         activation='relu', solver='adam', alpha=0.0001,
                                         batch_size='auto', learning_rate='constant',
                                         learning_rate_init=0.01, power_t=0.5, max_iter=2000,
                                         shuffle=True, random_state=None, tol=0.0001, verbose=
                                         warm_start=False, momentum=0.9, nesterovs_momentum=T
                                         early_stopping=False, validation_fraction=0.1, beta_
                                         beta_2=0.999, epsilon=1e-08)


         tprs = []
         aucs = []
         mean_fpr = np.linspace(0, 1, 100)

         i = 0
         for train, test in cv.split(X, Y):
             probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
             # Compute ROC curve and area the curve
             fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
             tprs.append(interp(mean_fpr, fpr, tpr))
             tprs[-1][0] = 0.0
             roc_auc = auc(fpr, tpr)
             aucs.append(roc_auc)
             plt.plot(fpr, tpr, lw=1, alpha=0.3,
                      label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

             i += 1
         plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                  label='Chance', alpha=.8)

         mean_tpr = np.mean(tprs, axis=0)
         mean_tpr[-1] = 1.0
         mean_auc = auc(mean_fpr, mean_tpr)
         std_auc = np.std(aucs)
         plt.plot(mean_fpr, mean_tpr, color='b',
                  label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
```

```
            lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()


print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.54 $\pm$ 0.13)
```

3.11 SVM

```
In [38]: from sklearn.svm import SVC
         clf_SVM = svm.SVC(kernel='rbf', probability=True,
                                     random_state=10)
```

```
for train_indices, test_indices in kf.split(X):
    clf_SVM.fit(X[train_indices], Y[train_indices])
    print(clf_SVM.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


0.9838249286393911


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


0.9257849666983825


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


0.8715509039010466


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


0.994291151284491


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


0.9952426260704091


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)


0.9866793529971456


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
```

```
0.9742857142857143


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarni
  y = column_or_1d(y, warn=True)


0.9714285714285714


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarni
  y = column_or_1d(y, warn=True)


0.939047619047619


C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarni
  y = column_or_1d(y, warn=True)


0.8333333333333334
```

```
In [ ]: cv = StratifiedKFold(n_splits=10)
        classifier =  svm.SVC(kernel='linear', probability=True,
                             random_state=10)

        tprs = []
        aucs = []
        mean_fpr = np.linspace(0, 1, 100)

        i = 0
        for train, test in cv.split(X, Y):
            probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
            # Compute ROC curve and area the curve
            fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
            tprs.append(interp(mean_fpr, fpr, tpr))
            tprs[-1][0] = 0.0
            roc_auc = auc(fpr, tpr)
            aucs.append(roc_auc)
            plt.plot(fpr, tpr, lw=1, alpha=0.3,
                     label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

            i += 1
        plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
                 label='Chance', alpha=.8)

        mean_tpr = np.mean(tprs, axis=0)
```

```
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarn
  y = column_or_1d(y, warn=True)
```

3.12 DT

In [75]:
```
from sklearn.tree import DecisionTreeClassifier
clf_tree = DecisionTreeClassifier(random_state=10)

for train_indices, test_indices in kf.split(X):
    clf_tree.fit(X[train_indices], Y[train_indices])
    print(clf_tree.score(X[test_indices], Y[test_indices]))
```

```
0.9628924833491912
0.9143672692673644
0.840152235965747
0.9086584205518554
```

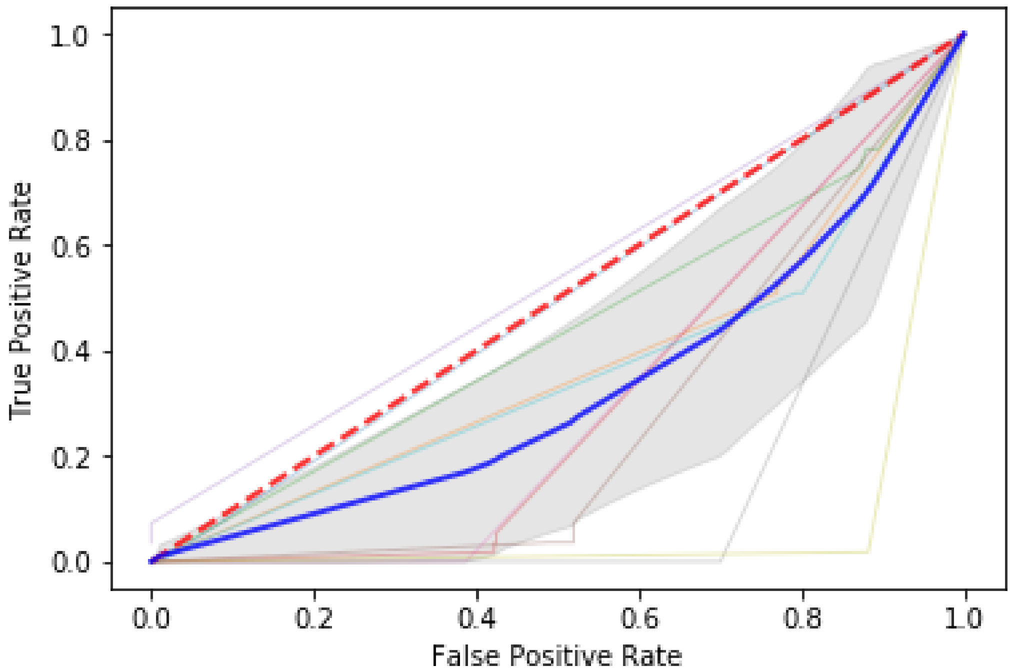
```
0.9952426260704091
0.9533777354900095
0.9085714285714286
0.9238095238095239
0.9276190476190476
0.7904761904761904
```

In [76]:
```python
print(clf_tree.score(X[train_indices], Y[train_indices]))
```

```
0.9831852791878173
```

In [79]:
```python
cv = StratifiedKFold(n_splits=10)
classifier =  DecisionTreeClassifier(random_state=10)

tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, Y):
    probas_ = classifier.fit(X[train], Y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(Y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
```

```
                    label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print(r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc))
```




```
Mean ROC (AUC = 0.33 $\pm$ 0.14)
```

Overall the models have preformed better, higher accuracy scores and higher AUC scores, when the number of features are reduced to the top 20 features. Now I will use logestic regression model, highest AUC score, to build maps predicting the location of outbreaks.

3.13 Predicting unknowns

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import ensemble, preprocessing

        train = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fall
        test = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fall
        weather = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fa
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: Deprecation
  from numpy.core.umath_tests import inner1d
```

```
In [2]: labels = train.WnvPresent.values

        weather = weather.drop('CodeSum', axis=1)

        weather_stn1 = weather[weather['Station']==1]
        weather_stn2 = weather[weather['Station']==2]
        weather_stn1 = weather_stn1.drop('Station', axis=1)
        weather_stn2 = weather_stn2.drop('Station', axis=1)
        weather = weather_stn1.merge(weather_stn2, on='Date')

In [3]: weather = weather.replace('M', -1)
        weather = weather.replace('-', -1)
        weather = weather.replace('T', -1)
        weather = weather.replace(' T', -1)
        weather = weather.replace('  T', -1)

In [4]: def create_month(x):
            return x.split('-')[1]

        def create_day(x):
            return x.split('-')[2]

        train['month'] = train.Date.apply(create_month)
        train['day'] = train.Date.apply(create_day)
        test['month'] = test.Date.apply(create_month)
        test['day'] = test.Date.apply(create_day)

In [5]: train['Lat_int'] = train.Latitude.apply(int)
        train['Long_int'] = train.Longitude.apply(int)
        test['Lat_int'] = test.Latitude.apply(int)
        test['Long_int'] = test.Longitude.apply(int)

In [6]: train = train.merge(weather, on='Date')
        test = test.merge(weather, on='Date')

In [7]: lbl = preprocessing.LabelEncoder()
        lbl.fit(list(train['Species'].values) + list(test['Species'].values))
        train['Species'] = lbl.transform(train['Species'].values)
        test['Species'] = lbl.transform(test['Species'].values)

        lbl.fit(list(train['Street'].values) + list(test['Street'].values))
        train['Street'] = lbl.transform(train['Street'].values)
        test['Street'] = lbl.transform(test['Street'].values)

        lbl.fit(list(train['Trap'].values) + list(test['Trap'].values))
```

```
        train['Trap'] = lbl.transform(train['Trap'].values)
        test['Trap'] = lbl.transform(test['Trap'].values)
```

In [8]:
```
train = train.ix[:,(train != -1).any(axis=0)]
test = test.ix[:,(test != -1).any(axis=0)]
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

In [10]:
```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10)



for n,v in train.items():
    if v.dtype == "object":
        train[n] = v.factorize()[0]


test1 = train[['WnvPresent']]
train = train.drop(['WnvPresent','AddressAccuracy','Lat_int','Long_int','Sunrise_x','S

X = train.iloc[:,:].values
Y = test1.iloc[:,:].values
```

In [11]:
```
X = train.iloc[:,:].values
Y = test1.iloc[:,:].values
```

In [12]:
```
from sklearn.linear_model import LogisticRegression
clf_Log = LogisticRegression(solver='liblinear', max_iter=100,
                             random_state=10,verbose=2,class_weight='balanced')

for train_indices, test_indices in kf.split(X):
    clf_Log.fit(X[train_indices], Y[train_indices])
    print(clf_Log.score(X[test_indices], Y[test_indices]))
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarni
  y = column_or_1d(y, warn=True)
```

```
[LibLinear]0.8829686013320647
[LibLinear]0.6279733587059942
[LibLinear]0.6527117031398668
[LibLinear]0.7078972407231209
[LibLinear]0.8877259752616555
[LibLinear]0.8392007611798288
[LibLinear]0.799047619047619
[LibLinear]0.7228571428571429
[LibLinear]0.7904761904761904
[LibLinear]0.4228571428571429
```

```
In [30]: test = test.drop(['AddressAccuracy','Lat_int','Long_int','Sunrise_x','Sunset_x','Heat_
```

```
        KeyErrorTraceback (most recent call last)

        <ipython-input-30-98c1412cbcb6> in <module>()
          1 test = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/I
    ----> 2 test = test.drop(['AddressAccuracy','Lat_int','Long_int','Sunrise_x','Sunset_x','He

        C:\Users\deanm\Anaconda2\lib\site-packages\pandas\core\frame.pyc in drop(self, labels,
       3695                                             index=index, columns=columns,
       3696                                             level=level, inplace=inplace,
    -> 3697                                             errors=errors)
       3698
       3699     @rewrite_axis_style_signature('mapper', [('copy', True),

        C:\Users\deanm\Anaconda2\lib\site-packages\pandas\core\generic.pyc in drop(self, labels
       3109         for axis, labels in axes.items():
       3110             if labels is not None:
    -> 3111                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
       3112
       3113         if inplace:

        C:\Users\deanm\Anaconda2\lib\site-packages\pandas\core\generic.pyc in _drop_axis(self,
       3141                 new_axis = axis.drop(labels, level=level, errors=errors)
       3142             else:
    -> 3143                 new_axis = axis.drop(labels, errors=errors)
```

```
   3144                 result = self.reindex(**{axis_name: new_axis})
   3145


    C:\Users\deanm\Anaconda2\lib\site-packages\pandas\core\indexes\base.pyc in drop(self, l
   4402                 if errors != 'ignore':
   4403                     raise KeyError(
-> 4404                         '{} not found in axis'.format(labels[mask]))
   4405                 indexer = indexer[~mask]
   4406             return self.delete(indexer)


    KeyError: "['Lat_int' 'Long_int' 'Sunrise_x' 'Sunset_x' 'Heat_x' 'Heat_y' 'Cool_x'\n 'C
```

```
In [15]: for n,v in test.items():
             if v.dtype == "object":
                 test[n] = v.factorize()[0]

In [16]: Y2 = test.iloc[:,:].values

In [17]: results=clf_Log.predict(Y2)

In [18]: results = pd.DataFrame(results)

In [19]: my_columns = ["result"]
         results.columns = my_columns

In [31]: results
```

Out[31]:

	result
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1

```
18           1
19           1
20           1
21           1
22           1
23           1
24           1
25           1
26           1
27           1
28           1
29           1
...        ...
116263       1
116264       1
116265       1
116266       1
116267       1
116268       1
116269       1
116270       1
116271       1
116272       1
116273       1
116274       1
116275       1
116276       1
116277       1
116278       1
116279       1
116280       1
116281       1
116282       1
116283       1
116284       1
116285       1
116286       1
116287       1
116288       1
116289       1
116290       1
116291       1
116292       1

[116293 rows x 1 columns]
```

```
In [32]: test = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
```

```
In [33]: df_out = pd.merge(test,results,how = 'left',left_index = True, right_index = True)
```

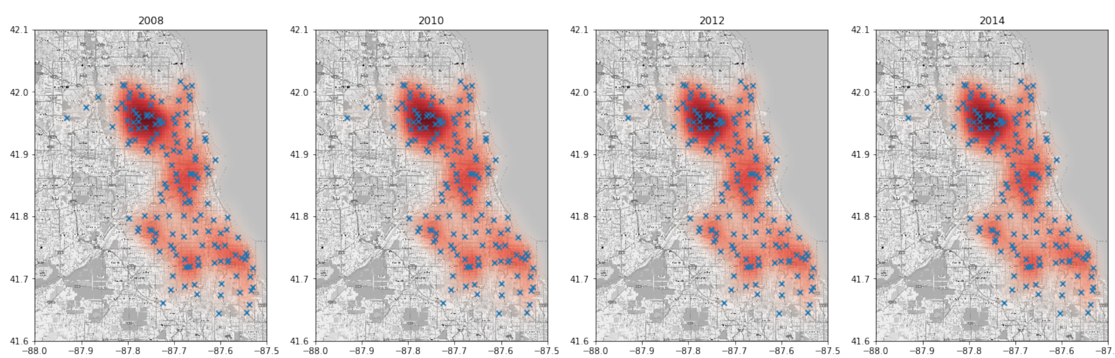
```
In [34]: df_out.to_csv('results.csv')

In [35]: traps = pd.read_csv('C:/Users/deanm/Documents/University of Idaho/Classes/Fall 2018/S
         species = pd.np.unique(traps['Species'])
         mapdata = np.loadtxt("C:\Users\deanm\OneDrive\Documents\University of Idaho\Classes\F

In [36]: traps

Out[36]:               Date   Trap  Longitude   Latitude  result                 Species
         0      2008-06-11   T002 -87.800991  41.954690       1  CULEX PIPIENS/RESTUANS
         1      2008-06-11   T002 -87.800991  41.954690       1          CULEX RESTUANS
         2      2008-06-11   T002 -87.800991  41.954690       1           CULEX PIPIENS
         3      2008-06-11   T002 -87.800991  41.954690       1        CULEX SALINARIUS
         4      2008-06-11   T002 -87.800991  41.954690       1         CULEX TERRITANS
         5      2008-06-11   T002 -87.800991  41.954690       1          CULEX TARSALIS
         6      2008-06-11   T002 -87.800991  41.954690       1       UNSPECIFIED CULEX
         7      2008-06-11   T002 -87.800991  41.954690       1         CULEX ERRATICUS
         8      2008-06-11   T007 -87.769279  41.994991       1  CULEX PIPIENS/RESTUANS
         9      2008-06-11   T007 -87.769279  41.994991       1          CULEX RESTUANS
         10     2008-06-11   T007 -87.769279  41.994991       1           CULEX PIPIENS
         11     2008-06-11   T007 -87.769279  41.994991       1        CULEX SALINARIUS
         12     2008-06-11   T007 -87.769279  41.994991       1         CULEX TERRITANS
         13     2008-06-11   T007 -87.769279  41.994991       1          CULEX TARSALIS
         14     2008-06-11   T007 -87.769279  41.994991       1       UNSPECIFIED CULEX
         15     2008-06-11   T007 -87.769279  41.994991       1         CULEX ERRATICUS
         16     2008-06-11   T015 -87.824812  41.974089       1  CULEX PIPIENS/RESTUANS
         17     2008-06-11   T015 -87.824812  41.974089       1          CULEX RESTUANS
         18     2008-06-11   T015 -87.824812  41.974089       1           CULEX PIPIENS
         19     2008-06-11   T015 -87.824812  41.974089       1        CULEX SALINARIUS
         20     2008-06-11   T015 -87.824812  41.974089       1         CULEX TERRITANS
         21     2008-06-11   T015 -87.824812  41.974089       1          CULEX TARSALIS
         22     2008-06-11   T015 -87.824812  41.974089       1       UNSPECIFIED CULEX
         23     2008-06-11   T015 -87.824812  41.974089       1         CULEX ERRATICUS
         24     2008-06-11   T045 -87.666455  41.921600       1  CULEX PIPIENS/RESTUANS
         25     2008-06-11   T045 -87.666455  41.921600       1          CULEX RESTUANS
         26     2008-06-11   T045 -87.666455  41.921600       1           CULEX PIPIENS
         27     2008-06-11   T045 -87.666455  41.921600       1        CULEX SALINARIUS
         28     2008-06-11   T045 -87.666455  41.921600       1         CULEX TERRITANS
         29     2008-06-11   T045 -87.666455  41.921600       1          CULEX TARSALIS
         ...           ...    ...        ...        ...     ...                     ...
         116263 2014-10-02   T238 -87.707394  41.753391       1           CULEX PIPIENS
         116264 2014-10-02   T238 -87.707394  41.753391       1        CULEX SALINARIUS
         116265 2014-10-02   T238 -87.707394  41.753391       1         CULEX TERRITANS
         116266 2014-10-02   T238 -87.707394  41.753391       1          CULEX TARSALIS
         116267 2014-10-02   T238 -87.707394  41.753391       1       UNSPECIFIED CULEX
         116268 2014-10-02   T238 -87.707394  41.753391       1         CULEX ERRATICUS
         116269 2014-10-02  T065A -87.749149  41.777689       1  CULEX PIPIENS/RESTUANS
         116270 2014-10-02  T065A -87.749149  41.777689       1          CULEX RESTUANS
```

```
116271 2014-10-02  T065A -87.749149  41.777689       1           CULEX PIPIENS
116272 2014-10-02  T065A -87.749149  41.777689       1        CULEX SALINARIUS
116273 2014-10-02  T065A -87.749149  41.777689       1         CULEX TERRITANS
116274 2014-10-02  T065A -87.749149  41.777689       1          CULEX TARSALIS
116275 2014-10-02  T065A -87.749149  41.777689       1       UNSPECIFIED CULEX
116276 2014-10-02  T065A -87.749149  41.777689       1         CULEX ERRATICUS
116277 2014-10-02  T094B -87.669539  41.719140       1  CULEX PIPIENS/RESTUANS
116278 2014-10-02  T094B -87.669539  41.719140       1          CULEX RESTUANS
116279 2014-10-02  T094B -87.669539  41.719140       1           CULEX PIPIENS
116280 2014-10-02  T094B -87.669539  41.719140       1        CULEX SALINARIUS
116281 2014-10-02  T094B -87.669539  41.719140       1         CULEX TERRITANS
116282 2014-10-02  T094B -87.669539  41.719140       1          CULEX TARSALIS
116283 2014-10-02  T094B -87.669539  41.719140       1       UNSPECIFIED CULEX
116284 2014-10-02  T094B -87.669539  41.719140       1         CULEX ERRATICUS
116285 2014-10-02  T054C -87.633590  41.925652       1  CULEX PIPIENS/RESTUANS
116286 2014-10-02  T054C -87.633590  41.925652       1          CULEX RESTUANS
116287 2014-10-02  T054C -87.633590  41.925652       1           CULEX PIPIENS
116288 2014-10-02  T054C -87.633590  41.925652       1        CULEX SALINARIUS
116289 2014-10-02  T054C -87.633590  41.925652       1         CULEX TERRITANS
116290 2014-10-02  T054C -87.633590  41.925652       1          CULEX TARSALIS
116291 2014-10-02  T054C -87.633590  41.925652       1       UNSPECIFIED CULEX
116292 2014-10-02  T054C -87.633590  41.925652       1         CULEX ERRATICUS

[116293 rows x 6 columns]
```

In [38]:
```
import matplotlib.pyplot as pl
from sklearn.neighbors import KernelDensity
from __future__ import print_function
import datetime
from sklearn.cross_validation import train_test_split
import csv
from sklearn import metrics
from sklearn.utils import shuffle

alpha_cm = pl.cm.Reds
alpha_cm._init()
alpha_cm._lut[:-3,-1] = abs(np.logspace(0, 1, alpha_cm.N) / 10 - 1)[::-1]
aspect = mapdata.shape[0] * 1.0 / mapdata.shape[1]
lon_lat_box = (-88, -87.5, 41.6, 42.1)

pl.figure(figsize=(18,6))
for year, subplot in zip([2008, 2010, 2012, 2014], [141, 142, 143, 144]):
    sightings = traps[(traps['result'] > 0) & (traps['Date'].apply(lambda x: x.year) =
    sightings = sightings.groupby(['Date', 'Trap','Longitude', 'Latitude']).max()['res
    X = sightings[['Longitude', 'Latitude']].values
    kd = KernelDensity(bandwidth=0.02)
    kd.fit(X)
```

```
            xv,yv = np.meshgrid(np.linspace(-88, -87.5, 100), np.linspace(41.6, 42.1, 100))
            gridpoints = np.array([xv.ravel(),yv.ravel()]).T
            zv = np.exp(kd.score_samples(gridpoints).reshape(100,100))
            pl.subplot(subplot)
            pl.gca().set_title(year)
            pl.imshow(mapdata,
                      cmap=pl.get_cmap('gray'),
                      extent=lon_lat_box,
                      aspect=aspect)
            pl.imshow(zv,
                      origin='lower',
                      cmap=alpha_cm,
                      extent=lon_lat_box,
                      aspect=aspect)
            pl.tight_layout()
            locations = traps[['Longitude', 'Latitude']].drop_duplicates().values
            pl.scatter(locations[:,0], locations[:,1], marker='x')

        pl.savefig('heatmapresult.png')
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning:
  "This module will be removed in 0.20.", DeprecationWarning)
```



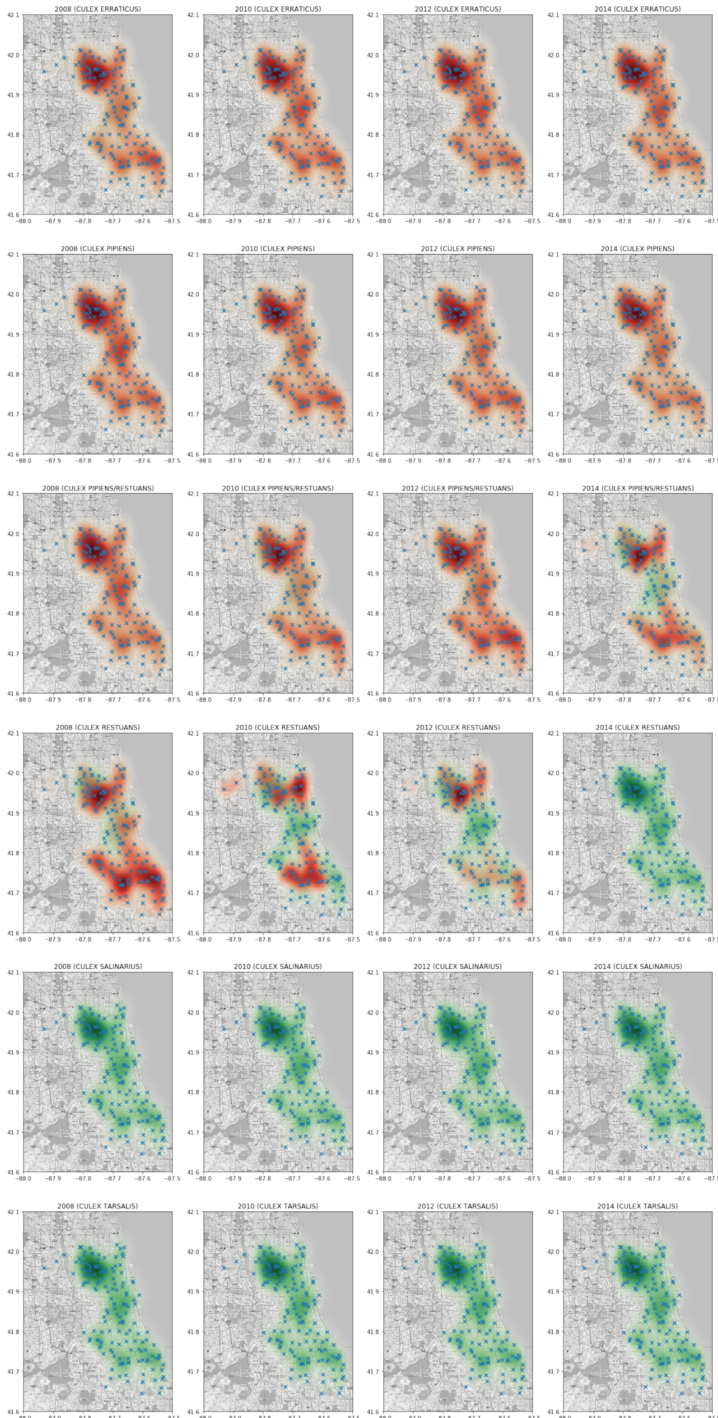
These maps do not appear to be ideal. All the years have been predicted to be outbreak years with the same amount of WNV present in each year. Lets break the maps into species and see if that helps with our understanding of what is going on.

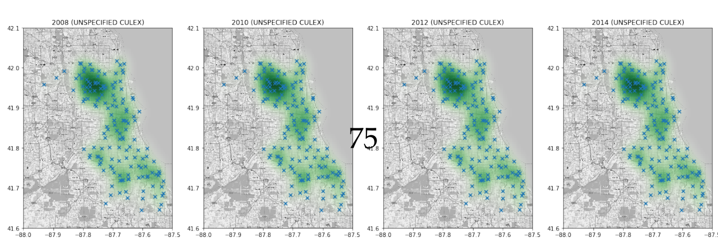
```
In [13]: alpha_cm = pl.cm.Reds
         alpha_cm._init()
         alpha_cm._lut[:-3,-1] = abs(np.logspace(0, 1, alpha_cm.N) / 10 - 1)[::-1]
         alpha_mcm = pl.cm.Greens
         alpha_mcm._init()
         alpha_mcm._lut[:-3,-1] = abs(np.logspace(0, 1, alpha_cm.N) / 10 - 1)[::-1]
         aspect = mapdata.shape[0] * 1.0 / mapdata.shape[1]
         lon_lat_box = (-88, -87.5, 41.6, 42.1)
```

```
subplot = 0
numSpcs = len(species)
pl.figure(figsize=(18,6*numSpcs))
for spcsIndx in range(numSpcs):
    for year in [2008, 2010, 2012, 2014]:
        subplot += 1
        sightings = traps[(traps['Species'] == species[spcsIndx])
                          & (traps['result'] > 0)
                          & (traps['Date'].apply(lambda x: x.year) == year)]
        sightings = sightings.groupby(['Date', 'Trap', 'Longitude', 'Latitude', 'Speci
        mSightings = traps[(traps['Species'] == species[spcsIndx])
                          & (traps['Date'].apply(lambda x: x.year) == year)]
        mSightings = mSightings.groupby(['Date', 'Trap', 'Longitude', 'Latitude', 'Spe
        if(len(mSightings) <= 0):
            print("SKIPPING [" + str(subplot) + "]:" + str(year) + " (" + species[spcs
            continue

        mX = mSightings[['Longitude', 'Latitude']].values
        mkd = KernelDensity(bandwidth=0.02)
        mkd.fit(mX)
        mxv,myv = np.meshgrid(np.linspace(-88, -87.5, 100), np.linspace(41.6, 42.1, 10
        mGridpoints = np.array([mxv.ravel(),myv.ravel()]).T
        mzv = np.exp(mkd.score_samples(mGridpoints).reshape(100,100))

        pl.subplot(numSpcs, 4, subplot)
        pl.gca().set_title(str(year) + " (" + species[spcsIndx] + ")")
        pl.imshow(mapdata,
                  cmap=pl.get_cmap('gray'),
                  extent=lon_lat_box,
                  aspect=aspect)
        pl.imshow(mzv,
                  origin='lower',
                  cmap=alpha_mcm,
                  extent=lon_lat_box,
                  aspect=aspect)
        if(len(sightings) > 0):
            X = sightings[['Longitude', 'Latitude']].values
            kd = KernelDensity(bandwidth=0.02)
            kd.fit(X)
            xv,yv = np.meshgrid(np.linspace(-88, -87.5, 100), np.linspace(41.6, 42.1,
            gridpoints = np.array([xv.ravel(),yv.ravel()]).T
            zv = np.exp(kd.score_samples(gridpoints).reshape(100,100))
            pl.imshow(zv,
                      origin='lower',
                      cmap=alpha_cm,
                      extent=lon_lat_box,
                      aspect=aspect)
```

```
        print("         [" + str(subplot) + "]:" + str(year) + " (" + species[spcsInd
        pl.tight_layout()
        locations = traps[['Longitude', 'Latitude']].drop_duplicates().values
        pl.scatter(locations[:,0], locations[:,1], marker='x')

pl.savefig('heatmap.png')
```

```
[1]:2008 (CULEX ERRATICUS)
[2]:2010 (CULEX ERRATICUS)
[3]:2012 (CULEX ERRATICUS)
[4]:2014 (CULEX ERRATICUS)
[5]:2008 (CULEX PIPIENS)
[6]:2010 (CULEX PIPIENS)
[7]:2012 (CULEX PIPIENS)
[8]:2014 (CULEX PIPIENS)
[9]:2008 (CULEX PIPIENS/RESTUANS)
[10]:2010 (CULEX PIPIENS/RESTUANS)
[11]:2012 (CULEX PIPIENS/RESTUANS)
[12]:2014 (CULEX PIPIENS/RESTUANS)
[13]:2008 (CULEX RESTUANS)
[14]:2010 (CULEX RESTUANS)
[15]:2012 (CULEX RESTUANS)
[16]:2014 (CULEX RESTUANS)
[17]:2008 (CULEX SALINARIUS)
[18]:2010 (CULEX SALINARIUS)
[19]:2012 (CULEX SALINARIUS)
[20]:2014 (CULEX SALINARIUS)
[21]:2008 (CULEX TARSALIS)
[22]:2010 (CULEX TARSALIS)
[23]:2012 (CULEX TARSALIS)
[24]:2014 (CULEX TARSALIS)
[25]:2008 (CULEX TERRITANS)
[26]:2010 (CULEX TERRITANS)
[27]:2012 (CULEX TERRITANS)
[28]:2014 (CULEX TERRITANS)
[29]:2008 (UNSPECIFIED CULEX)
[30]:2010 (UNSPECIFIED CULEX)
[31]:2012 (UNSPECIFIED CULEX)
[32]:2014 (UNSPECIFIED CULEX)
```


2008 (CULEX ERRATICUS)
2010 (CULEX ERRATICUS)
2012 (CULEX ERRATICUS)
2014 (CULEX ERRATICUS)
2008 (CULEX PIPIENS)
2010 (CULEX PIPIENS)
2012 (CULEX PIPIENS)
2014 (CULEX PIPIENS)
2008 (CULEX PIPIENS/RESTUANS)
2010 (CULEX PIPIENS/RESTUANS)
2012 (CULEX PIPIENS/RESTUANS)
2014 (CULEX PIPIENS/RESTUANS)
2008 (CULEX RESTUANS)
2010 (CULEX RESTUANS)
2012 (CULEX RESTUANS)
2014 (CULEX RESTUANS)
2008 (CULEX SALINARIUS)
2010 (CULEX SALINARIUS)
2012 (CULEX SALINARIUS)
2014 (CULEX SALINARIUS)
2008 (CULEX TARSALIS)
2010 (CULEX TARSALIS)
2012 (CULEX TARSALIS)
2014 (CULEX TARSALIS)


2008 (UNSPECIFIED CULEX)
2010 (UNSPECIFIED CULEX)
2012 (UNSPECIFIED CULEX)
2014 (UNSPECIFIED CULEX)

It appears that th emodel has predicted that Culex erraticus will be a carrier of WNV, even though it was not a carrier of WNV in the training years. This is an interesting result, I think further work would need to be done to fix this issue.

All other species of mosquitoes that did not carry WNV in the training set, were predicted to not carry WNV in the test set, which is interesting that these species were predicted correctly.

The Culex pippiens mosquito has the best looking map of outbreaks, with some years having more west nile than others. This model looks much more like the test data set.

I would also like to state that I'm surprised a guassian model did so poorly, since epidemiologists use models that have a guessian distribution to predict disease and population growth. I would have thought this model would have preformed well since it is the standard model distribution in the field.

Overall, I think more work would need to be done to properlly predict the outbreaks of WNV.

4 Cluster analysis

Now I plan to analyze the data with clustering. Lets see if there are clusters of dates where WNV could be present. For this part of the analysis I will be dropping the WNV present column from all data, since clustering is usually thought of as a unsupervised learning techinque.

```
In [10]: train = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
         test = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fall
         weather = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/F


         labels = train.WnvPresent.values


         weather = weather.drop('CodeSum', axis=1)


         weather_stn1 = weather[weather['Station']==1]
         weather_stn2 = weather[weather['Station']==2]
         weather_stn1 = weather_stn1.drop('Station', axis=1)
         weather_stn2 = weather_stn2.drop('Station', axis=1)
         weather = weather_stn1.merge(weather_stn2, on='Date')


         weather = weather.replace('M', -1)
         weather = weather.replace('-', -1)
         weather = weather.replace('T', -1)
         weather = weather.replace(' T', -1)
         weather = weather.replace('  T', -1)


         def create_month(x):
```

```
        return x.split('-')[1]

    def create_day(x):
        return x.split('-')[2]

    train['month'] = train.Date.apply(create_month)
    train['day'] = train.Date.apply(create_day)
    test['month'] = test.Date.apply(create_month)
    test['day'] = test.Date.apply(create_day)

    train['Lat_int'] = train.Latitude.apply(int)
    train['Long_int'] = train.Longitude.apply(int)
    test['Lat_int'] = test.Latitude.apply(int)
    test['Long_int'] = test.Longitude.apply(int)

    train = train.drop(['Address', 'AddressNumberAndStreet','WnvPresent', 'NumMosquitos']
    test = test.drop(['Id', 'Address', 'AddressNumberAndStreet'], axis = 1)

    train = train.merge(weather, on='Date')
    test = test.merge(weather, on='Date')
    train = train.drop(['Date'], axis = 1)
    test = test.drop(['Date'], axis = 1)

    lbl = preprocessing.LabelEncoder()
    lbl.fit(list(train['Species'].values) + list(test['Species'].values))
    train['Species'] = lbl.transform(train['Species'].values)
    test['Species'] = lbl.transform(test['Species'].values)

    lbl.fit(list(train['Street'].values) + list(test['Street'].values))
    train['Street'] = lbl.transform(train['Street'].values)
    test['Street'] = lbl.transform(test['Street'].values)

    lbl.fit(list(train['Trap'].values) + list(test['Trap'].values))
    train['Trap'] = lbl.transform(train['Trap'].values)
    test['Trap'] = lbl.transform(test['Trap'].values)

    train = train.ix[:,(train != -1).any(axis=0)]
    test = test.ix[:,(test != -1).any(axis=0)]
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:63: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:64: DeprecationWarning:
.ix is deprecated. Please use
```

```
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

```
In [11]: from sklearn import preprocessing
         import numpy as np
         X_scaled = preprocessing.scale(test)

         X_scaled
```

```
Out[11]: array([[-0.64920977, -0.00527255, -0.81593499, ...,  1.12634419,
                 -0.03993339,  0.94414545],
                [-0.21112813, -0.00527255, -0.81593499, ...,  1.12634419,
                 -0.03993339,  0.94414545],
                [-1.08729142, -0.00527255, -0.81593499, ...,  1.12634419,
                 -0.03993339,  0.94414545],
                ...,
                [ 0.66503516, -0.80962834, -1.26951524, ...,  0.36586998,
                 -0.14007513, -0.05781679],
                [ 1.54119844, -0.80962834, -1.26951524, ...,  0.36586998,
                 -0.14007513, -0.05781679],
                [-1.52537306, -0.80962834, -1.26951524, ...,  0.36586998,
                 -0.14007513, -0.05781679]])
```

I would like to reduce the dimensions of the data before continuing with clustering.

```
In [12]: import numpy as np
         from sklearn.decomposition import PCA
         X = X_scaled
         pca = PCA(n_components=2)
         pca.fit(X)
         PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
           svd_solver='auto', tol=0.0, whiten=False)
         print(pca.explained_variance_ratio_)

         print(pca.singular_values_)
```

```
[0.3434767 0.1110375]
[1264.02429517  718.69002161]
```

```
In [13]: X_pca = pca.transform(X)
```

```
In [14]: projected = pca.fit_transform(X)
         print(X)
         print(projected.shape)
```

```
[[-0.64920977 -0.00527255 -0.81593499 ...  1.12634419 -0.03993339
   0.94414545]
 [-0.21112813 -0.00527255 -0.81593499 ...  1.12634419 -0.03993339
   0.94414545]
 [-1.08729142 -0.00527255 -0.81593499 ...  1.12634419 -0.03993339
   0.94414545]
 ...
 [ 0.66503516 -0.80962834 -1.26951524 ...  0.36586998 -0.14007513
  -0.05781679]
 [ 1.54119844 -0.80962834 -1.26951524 ...  0.36586998 -0.14007513
  -0.05781679]
 [-1.52537306 -0.80962834 -1.26951524 ...  0.36586998 -0.14007513
  -0.05781679]]
(116293L, 2L)
```

Checking to make sure we still have all our rows, and we do.

4.1 K-Means

First, I have choosen to do a kmeans cluster. This is one of the simpler types of clustering, it bases the groups off of samilarity in the features. The algorithm will optimize the centroid for the clusters, giving us our groups.

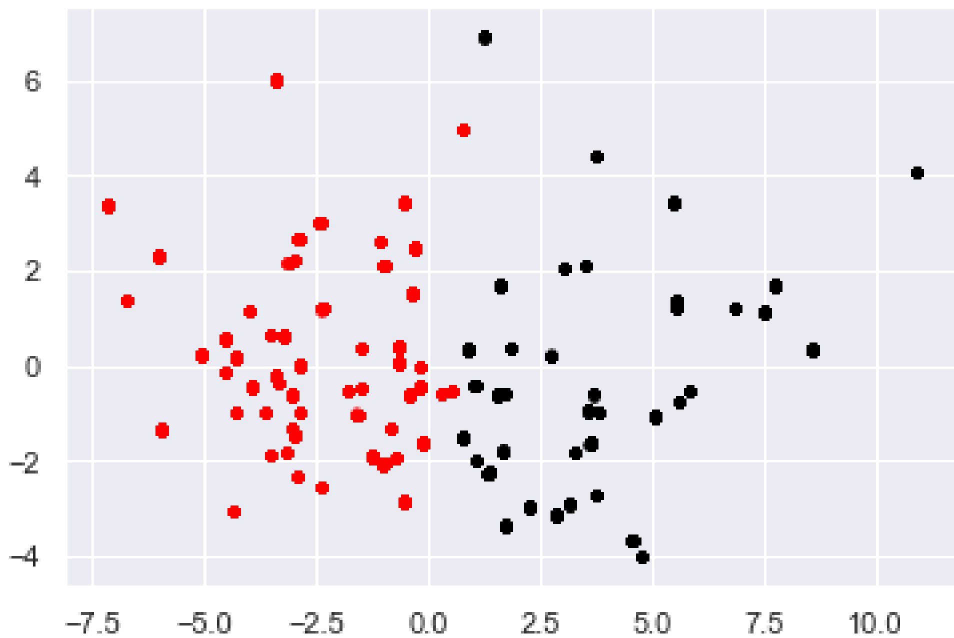
```
In [15]: from sklearn.cluster import KMeans
```

```
In [16]: kmeans = KMeans(n_clusters=2)

         kmeans.fit(X_pca)

         print(kmeans.cluster_centers_)

         y_km = kmeans.fit_predict(X_pca)
```

```
[[-2.50384839  0.13657815]
 [ 3.77875978 -0.20612111]]
```

```
In [17]: pl.scatter(X_pca[y_km ==0,0], X_pca[y_km == 0,1], s=10, c='red')
         pl.scatter(X_pca[y_km ==1,0], X_pca[y_km == 1,1], s=10, c='black')
```

```
Out[17]: <matplotlib.collections.PathCollection at 0xd5039e8>
```



I'm not sure why I've lost many of my data points in this figure. I confirmed above that we have not lost any rows of data. I've tried other coding to plot more of the points but they do not seem to be working.

It does appear that when we have two clsuters defined that the data does fall nicely into two groups, WNV present and not present. This could indicate that there are days that have the right conditions versus other days do not have conditions for the virus. Lets move onto another type of clustering

4.2 H. Clustering

Heirarchacal clustering working by spliting clusters based on dissimilarity between two sets. Most often the dissimilarity is determined by euclidean distance, but any form of matrix disctance can be used.

```
In [ ]: #Caused computer to crash
        from scipy.cluster.hierarchy import dendrogram, linkage
        from matplotlib import pyplot as plt

        linked = linkage(X_pca, 'single')

        labelList = range(1, 11)

        plt.figure(figsize=(10, 7))
        dendrogram(linked,
                   orientation='top',
                   labels=labelList,
```

```
                distance_sort='descending',
                show_leaf_counts=True)
plt.show()
```

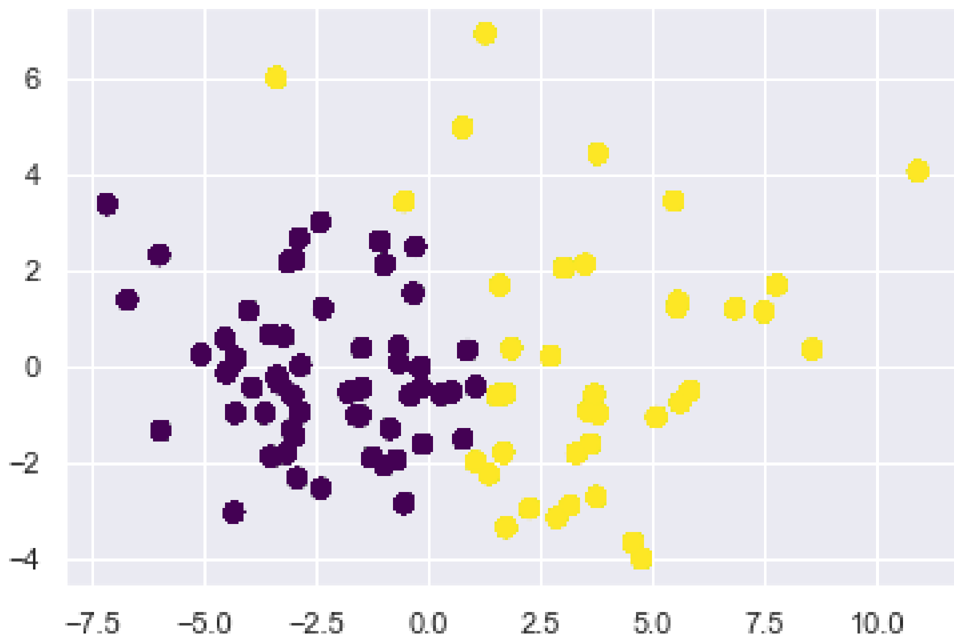
Was unable to run the above code without having my computer crash, tried multiple times.

5 Gausian Model based clustering

```
In [18]: %matplotlib inline
         import matplotlib.pyplot as plt
         import seaborn as sns; sns.set()
         import numpy as np
         from sklearn.cluster import KMeans
         from scipy.spatial.distance import cdist
```

```
In [21]: from sklearn.mixture import GMM
         gmm = GMM(n_components=2).fit(X_pca)
         labels = gmm.predict(X_pca)
         plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, s=40, cmap='viridis');
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```



Again, I'm not sucre why I'm losing so many points in these figures, this is clearly not 100K data points.

```
In [22]: n_components = np.arange(1, 21)
         models = [GMM(n, covariance_type='full', random_state=0).fit(X)
                   for n in n_components]

         plt.plot(n_components, [m.bic(X) for m in models], label='BIC')
         plt.plot(n_components, [m.aic(X) for m in models], label='AIC')
         plt.legend(loc='best')
         plt.xlabel('n_components');
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

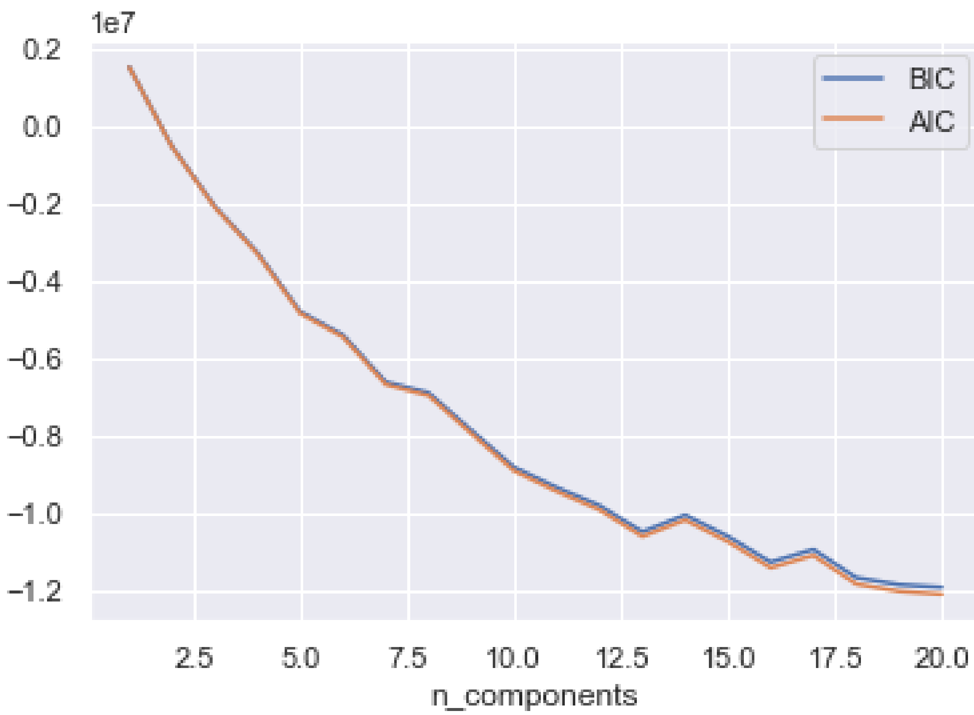
```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:58: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\deanm\Anaconda2\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning
  warnings.warn(msg, category=DeprecationWarning)
```



6 Association

Now I will analyze the data in way to see if there are any rules that dictate when WNV can and can't be present. This is refered to as association, it is often used with super markets, to target customers who buy items together frequently, such as butter and popcorn also buy Coke.

To continue I need to transform many of my continous values into categorical. I will be using the quantiles of values to do this. So, for examples, the bottom quartile of temperature will be "cold", then "cool", "warm, and the top quartile will be "hot".

I will also only be using the top 15 features that were decided on previoulsy in my feature selection.

```
In [81]: train = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
         test = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/Fal
         weather = pd.read_csv('C:/Users/deanm/OneDrive/Documents/University of Idaho/Classes/F
```

```python
labels = train.WnvPresent.values


weather = weather.drop('CodeSum', axis=1)


weather_stn1 = weather[weather['Station']==1]
weather_stn2 = weather[weather['Station']==2]
weather_stn1 = weather_stn1.drop('Station', axis=1)
weather_stn2 = weather_stn2.drop('Station', axis=1)
weather = weather_stn1.merge(weather_stn2, on='Date')


weather = weather.replace('M', -1)
weather = weather.replace('-', -1)
weather = weather.replace('T', -1)
weather = weather.replace(' T', -1)
weather = weather.replace('  T', -1)


def create_month(x):
    return x.split('-')[1]

def create_day(x):
    return x.split('-')[2]

train['month'] = train.Date.apply(create_month)
train['day'] = train.Date.apply(create_day)
test['month'] = test.Date.apply(create_month)
test['day'] = test.Date.apply(create_day)

train['Lat_int'] = train.Latitude.apply(int)
train['Long_int'] = train.Longitude.apply(int)
test['Lat_int'] = test.Latitude.apply(int)
test['Long_int'] = test.Longitude.apply(int)

train = train.drop(['Address', 'AddressNumberAndStreet', 'NumMosquitos'], axis = 1)
test = test.drop(['Id', 'Address', 'AddressNumberAndStreet'], axis = 1)

train = train.merge(weather, on='Date')
test = test.merge(weather, on='Date')
train = train.drop(['Date'], axis = 1)
test = test.drop(['Date'], axis = 1)


lbl.fit(list(train['Street'].values) + list(test['Street'].values))
train['Street'] = lbl.transform(train['Street'].values)
```

```python
test['Street'] = lbl.transform(test['Street'].values)

lbl.fit(list(train['Trap'].values) + list(test['Trap'].values))
train['Trap'] = lbl.transform(train['Trap'].values)
test['Trap'] = lbl.transform(test['Trap'].values)

train = train.ix[:,(train != -1).any(axis=0)]
test = test.ix[:,(test != -1).any(axis=0)]
```

```
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:59: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
C:\Users\deanm\Anaconda2\lib\site-packages\ipykernel_launcher.py:60: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

In [52]: train

Out[52]:

	Species	Block	Street	Trap	Latitude	Longitude	\
0	CULEX PIPIENS/RESTUANS	41	36	1	41.954690	-87.800991	
1	CULEX RESTUANS	41	36	1	41.954690	-87.800991	
2	CULEX RESTUANS	62	30	8	41.994991	-87.769279	
3	CULEX PIPIENS/RESTUANS	79	120	15	41.974089	-87.824812	
4	CULEX RESTUANS	79	120	15	41.974089	-87.824812	
5	CULEX RESTUANS	15	138	34	41.921600	-87.666455	
6	CULEX RESTUANS	25	123	35	41.891118	-87.654491	
7	CULEX PIPIENS/RESTUANS	11	134	37	41.867108	-87.654224	
8	CULEX RESTUANS	11	134	37	41.867108	-87.654224	
9	CULEX RESTUANS	11	117	38	41.896282	-87.655232	
10	CULEX PIPIENS/RESTUANS	21	45	39	41.919343	-87.694259	
11	CULEX PIPIENS/RESTUANS	22	18	41	41.921965	-87.632085	
12	CULEX RESTUANS	22	18	41	41.921965	-87.632085	
13	CULEX PIPIENS/RESTUANS	22	95	68	41.688324	-87.676709	
14	CULEX RESTUANS	22	95	68	41.688324	-87.676709	
15	CULEX RESTUANS	11	81	75	41.862292	-87.648860	
16	CULEX RESTUANS	17	112	77	41.720848	-87.666014	
17	CULEX RESTUANS	22	111	80	41.731922	-87.677512	
18	CULEX PIPIENS	22	111	80	41.731922	-87.677512	
19	CULEX PIPIENS/RESTUANS	53	46	91	41.891126	-87.611560	

```
20              CULEX RESTUANS     53       46     91  41.891126 -87.611560
21       CULEX PIPIENS/RESTUANS    65       36     96  41.999129 -87.795585
22       CULEX PIPIENS/RESTUANS    75       37    101  42.017430 -87.687769
23              CULEX RESTUANS     15       29    106  41.907645 -87.760886
24              CULEX RESTUANS     89       57    112  41.732984 -87.649642
25       CULEX PIPIENS/RESTUANS    41       36      1  41.954690 -87.800991
26              CULEX RESTUANS     41       36      1  41.954690 -87.800991
27               CULEX PIPIENS     41       36      1  41.954690 -87.800991
28       CULEX PIPIENS/RESTUANS    79      120     15  41.974089 -87.824812
29              CULEX RESTUANS     79      120     15  41.974089 -87.824812
...                        ...    ...      ...    ...        ...        ...
10476    CULEX PIPIENS/RESTUANS    10       62     84  41.750498 -87.605294
10477           CULEX RESTUANS     10       62     84  41.750498 -87.605294
10478            CULEX PIPIENS     10       62     84  41.750498 -87.605294
10479            CULEX PIPIENS     58       40     21  41.984809 -87.728492
10480    CULEX PIPIENS/RESTUANS    40        5    129  41.659112 -87.538693
10481           CULEX RESTUANS     40        5    129  41.659112 -87.538693
10482            CULEX PIPIENS     40        5    129  41.659112 -87.538693
10483           CULEX RESTUANS     91      127     10  41.992478 -87.862995
10484    CULEX PIPIENS/RESTUANS    10      132    147  41.974689 -87.890615
10485    CULEX PIPIENS/RESTUANS    10      132    147  41.974689 -87.890615
10486            CULEX PIPIENS     10      132    147  41.974689 -87.890615
10487            CULEX PIPIENS     10      132    147  41.974689 -87.890615
10488            CULEX PIPIENS     10      132    147  41.974689 -87.890615
10489            CULEX PIPIENS     10      132    147  41.974689 -87.890615
10490    CULEX PIPIENS/RESTUANS    48      129    130  41.925198 -87.746381
10491    CULEX PIPIENS/RESTUANS    51       34    131  41.973845 -87.805059
10492            CULEX PIPIENS     51       34    131  41.973845 -87.805059
10493    CULEX PIPIENS/RESTUANS    82       72    133  41.743402 -87.731435
10494    CULEX PIPIENS/RESTUANS    65        9    135  41.728495 -87.600963
10495            CULEX PIPIENS     65        9    135  41.728495 -87.600963
10496    CULEX PIPIENS/RESTUANS    17      113    132  41.947227 -87.671457
10497    CULEX PIPIENS/RESTUANS    90      122    134  41.793818 -87.654234
10498    CULEX PIPIENS/RESTUANS    13       26    138  41.904194 -87.756155
10499            CULEX PIPIENS     13       26    138  41.904194 -87.756155
10500    CULEX PIPIENS/RESTUANS    39       44    136  41.951866 -87.725057
10501    CULEX PIPIENS/RESTUANS    51      108     27  41.763733 -87.742302
10502    CULEX PIPIENS/RESTUANS    58       42    139  41.987280 -87.666066
10503    CULEX PIPIENS/RESTUANS    17       12    140  41.912563 -87.668055
10504    CULEX PIPIENS/RESTUANS    71       21    141  42.009876 -87.807277
10505    CULEX PIPIENS/RESTUANS    42      106    143  41.776428 -87.627096

       AddressAccuracy  WnvPresent month day    ...      DewPoint_y  \
0                    9           0    05  29    ...              59
1                    9           0    05  29    ...              59
2                    9           0    05  29    ...              59
3                    8           0    05  29    ...              59
4                    8           0    05  29    ...              59
```

5	8	0	05	29	...	59
6	8	0	05	29	...	59
7	8	0	05	29	...	59
8	8	0	05	29	...	59
9	8	0	05	29	...	59
10	8	0	05	29	...	59
11	8	0	05	29	...	59
12	8	0	05	29	...	59
13	8	0	05	29	...	59
14	8	0	05	29	...	59
15	8	0	05	29	...	59
16	9	0	05	29	...	59
17	8	0	05	29	...	59
18	8	0	05	29	...	59
19	5	0	05	29	...	59
20	5	0	05	29	...	59
21	8	0	05	29	...	59
22	8	0	05	29	...	59
23	8	0	05	29	...	59
24	8	0	05	29	...	59
25	9	0	06	05	...	47
26	9	0	06	05	...	47
27	9	0	06	05	...	47
28	8	0	06	05	...	47
29	8	0	06	05	...	47
...	...	...	...	..	...	...
10476	5	0	09	26	...	52
10477	5	0	09	26	...	52
10478	5	0	09	26	...	52
10479	8	0	09	26	...	52
10480	8	0	09	26	...	52
10481	8	0	09	26	...	52
10482	8	0	09	26	...	52
10483	8	0	09	26	...	52
10484	9	1	09	26	...	52
10485	9	0	09	26	...	52
10486	9	0	09	26	...	52
10487	9	0	09	26	...	52
10488	9	0	09	26	...	52
10489	9	0	09	26	...	52
10490	8	0	09	26	...	52
10491	9	0	09	26	...	52
10492	9	0	09	26	...	52
10493	8	0	09	26	...	52
10494	5	0	09	26	...	52
10495	5	0	09	26	...	52
10496	9	0	09	26	...	52
10497	5	0	09	26	...	52

```
10498              9           0    09  26    ...             52
10499              9           0    09  26    ...             52
10500              8           0    09  26    ...             52
10501              8           1    09  26    ...             52
10502              8           0    09  26    ...             52
10503              9           0    09  26    ...             52
10504              9           0    09  26    ...             52
10505              8           0    09  26    ...             52

       WetBulb_y  Heat_y  Cool_y PrecipTotal_y StnPressure_y  SeaLevel_y  \
0             66       0      12          0.00         29.44       30.09
1             66       0      12          0.00         29.44       30.09
2             66       0      12          0.00         29.44       30.09
3             66       0      12          0.00         29.44       30.09
4             66       0      12          0.00         29.44       30.09
5             66       0      12          0.00         29.44       30.09
6             66       0      12          0.00         29.44       30.09
7             66       0      12          0.00         29.44       30.09
8             66       0      12          0.00         29.44       30.09
9             66       0      12          0.00         29.44       30.09
10            66       0      12          0.00         29.44       30.09
11            66       0      12          0.00         29.44       30.09
12            66       0      12          0.00         29.44       30.09
13            66       0      12          0.00         29.44       30.09
14            66       0      12          0.00         29.44       30.09
15            66       0      12          0.00         29.44       30.09
16            66       0      12          0.00         29.44       30.09
17            66       0      12          0.00         29.44       30.09
18            66       0      12          0.00         29.44       30.09
19            66       0      12          0.00         29.44       30.09
20            66       0      12          0.00         29.44       30.09
21            66       0      12          0.00         29.44       30.09
22            66       0      12          0.00         29.44       30.09
23            66       0      12          0.00         29.44       30.09
24            66       0      12          0.00         29.44       30.09
25            52       8       0          0.27         29.16       29.78
26            52       8       0          0.27         29.16       29.78
27            52       8       0          0.27         29.16       29.78
28            52       8       0          0.27         29.16       29.78
29            52       8       0          0.27         29.16       29.78
...          ...     ...     ...           ...           ...         ...
10476         58       0       0          0.00         29.40       30.04
10477         58       0       0          0.00         29.40       30.04
10478         58       0       0          0.00         29.40       30.04
10479         58       0       0          0.00         29.40       30.04
10480         58       0       0          0.00         29.40       30.04
10481         58       0       0          0.00         29.40       30.04
10482         58       0       0          0.00         29.40       30.04
```

10483	58	0	0	0.00	29.40	30.04
10484	58	0	0	0.00	29.40	30.04
10485	58	0	0	0.00	29.40	30.04
10486	58	0	0	0.00	29.40	30.04
10487	58	0	0	0.00	29.40	30.04
10488	58	0	0	0.00	29.40	30.04
10489	58	0	0	0.00	29.40	30.04
10490	58	0	0	0.00	29.40	30.04
10491	58	0	0	0.00	29.40	30.04
10492	58	0	0	0.00	29.40	30.04
10493	58	0	0	0.00	29.40	30.04
10494	58	0	0	0.00	29.40	30.04
10495	58	0	0	0.00	29.40	30.04
10496	58	0	0	0.00	29.40	30.04
10497	58	0	0	0.00	29.40	30.04
10498	58	0	0	0.00	29.40	30.04
10499	58	0	0	0.00	29.40	30.04
10500	58	0	0	0.00	29.40	30.04
10501	58	0	0	0.00	29.40	30.04
10502	58	0	0	0.00	29.40	30.04
10503	58	0	0	0.00	29.40	30.04
10504	58	0	0	0.00	29.40	30.04
10505	58	0	0	0.00	29.40	30.04

	ResultSpeed_y	ResultDir_y	AvgSpeed_y
0	5.8	16	7.4
1	5.8	16	7.4
2	5.8	16	7.4
3	5.8	16	7.4
4	5.8	16	7.4
5	5.8	16	7.4
6	5.8	16	7.4
7	5.8	16	7.4
8	5.8	16	7.4
9	5.8	16	7.4
10	5.8	16	7.4
11	5.8	16	7.4
12	5.8	16	7.4
13	5.8	16	7.4
14	5.8	16	7.4
15	5.8	16	7.4
16	5.8	16	7.4
17	5.8	16	7.4
18	5.8	16	7.4
19	5.8	16	7.4
20	5.8	16	7.4
21	5.8	16	7.4
22	5.8	16	7.4

```
23          5.8     16     7.4
24          5.8     16     7.4
25          6.2      3     8.1
26          6.2      3     8.1
27          6.2      3     8.1
28          6.2      3     8.1
29          6.2      3     8.1
...         ...    ...     ...
10476       4.1      9     4.6
10477       4.1      9     4.6
10478       4.1      9     4.6
10479       4.1      9     4.6
10480       4.1      9     4.6
10481       4.1      9     4.6
10482       4.1      9     4.6
10483       4.1      9     4.6
10484       4.1      9     4.6
10485       4.1      9     4.6
10486       4.1      9     4.6
10487       4.1      9     4.6
10488       4.1      9     4.6
10489       4.1      9     4.6
10490       4.1      9     4.6
10491       4.1      9     4.6
10492       4.1      9     4.6
10493       4.1      9     4.6
10494       4.1      9     4.6
10495       4.1      9     4.6
10496       4.1      9     4.6
10497       4.1      9     4.6
10498       4.1      9     4.6
10499       4.1      9     4.6
10500       4.1      9     4.6
10501       4.1      9     4.6
10502       4.1      9     4.6
10503       4.1      9     4.6
10504       4.1      9     4.6
10505       4.1      9     4.6

[10506 rows x 43 columns]
```

In [27]:
```
import numpy as np
import pandas as pd

from scipy.stats.mstats import mquantiles
```

```
AttributeErrorTraceback (most recent call last)

<ipython-input-27-3cb893059400> in <module>()
  4 from scipy.stats.mstats import mquantiles
  5
----> 6 df = test.DataFrame({'value': np.random.randint(1, 80, 20)})
  7 df['Tmax_x'] = pd.cut(df.value,
  8                       bins=[0, 5, 31, 51, 80],


C:\Users\deanm\Anaconda2\lib\site-packages\pandas\core\generic.pyc in __getattr__(self
   4374             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   4375                 return self[name]
-> 4376             return object.__getattribute__(self, name)
   4377
   4378     def __setattr__(self, name, value):


AttributeError: 'DataFrame' object has no attribute 'DataFrame'
```

```
In [82]: test = train.drop(['AddressAccuracy','Lat_int','Long_int','Sunrise_x','Sunset_x','Hea
```

```
In [54]: test
```

Out[54]:

	Species	Block	Street	Trap	Latitude	Longitude	\
0	CULEX PIPIENS/RESTUANS	41	36	1	41.954690	-87.800991	
1	CULEX RESTUANS	41	36	1	41.954690	-87.800991	
2	CULEX RESTUANS	62	30	8	41.994991	-87.769279	
3	CULEX PIPIENS/RESTUANS	79	120	15	41.974089	-87.824812	
4	CULEX RESTUANS	79	120	15	41.974089	-87.824812	
5	CULEX RESTUANS	15	138	34	41.921600	-87.666455	
6	CULEX RESTUANS	25	123	35	41.891118	-87.654491	
7	CULEX PIPIENS/RESTUANS	11	134	37	41.867108	-87.654224	
8	CULEX RESTUANS	11	134	37	41.867108	-87.654224	
9	CULEX RESTUANS	11	117	38	41.896282	-87.655232	
10	CULEX PIPIENS/RESTUANS	21	45	39	41.919343	-87.694259	
11	CULEX PIPIENS/RESTUANS	22	18	41	41.921965	-87.632085	
12	CULEX RESTUANS	22	18	41	41.921965	-87.632085	
13	CULEX PIPIENS/RESTUANS	22	95	68	41.688324	-87.676709	
14	CULEX RESTUANS	22	95	68	41.688324	-87.676709	
15	CULEX RESTUANS	11	81	75	41.862292	-87.648860	
16	CULEX RESTUANS	17	112	77	41.720848	-87.666014	
17	CULEX RESTUANS	22	111	80	41.731922	-87.677512	
18	CULEX PIPIENS	22	111	80	41.731922	-87.677512	
19	CULEX PIPIENS/RESTUANS	53	46	91	41.891126	-87.611560	
20	CULEX RESTUANS	53	46	91	41.891126	-87.611560	

```
21      CULEX PIPIENS/RESTUANS     65      36    96  41.999129 -87.795585
22      CULEX PIPIENS/RESTUANS     75      37   101  42.017430 -87.687769
23              CULEX RESTUANS     15      29   106  41.907645 -87.760886
24              CULEX RESTUANS     89      57   112  41.732984 -87.649642
25      CULEX PIPIENS/RESTUANS     41      36     1  41.954690 -87.800991
26              CULEX RESTUANS     41      36     1  41.954690 -87.800991
27               CULEX PIPIENS     41      36     1  41.954690 -87.800991
28      CULEX PIPIENS/RESTUANS     79     120    15  41.974089 -87.824812
29              CULEX RESTUANS     79     120    15  41.974089 -87.824812
...                        ...    ...     ...   ...        ...        ...
10476   CULEX PIPIENS/RESTUANS     10      62    84  41.750498 -87.605294
10477           CULEX RESTUANS     10      62    84  41.750498 -87.605294
10478            CULEX PIPIENS     10      62    84  41.750498 -87.605294
10479            CULEX PIPIENS     58      40    21  41.984809 -87.728492
10480   CULEX PIPIENS/RESTUANS     40       5   129  41.659112 -87.538693
10481           CULEX RESTUANS     40       5   129  41.659112 -87.538693
10482            CULEX PIPIENS     40       5   129  41.659112 -87.538693
10483           CULEX RESTUANS     91     127    10  41.992478 -87.862995
10484   CULEX PIPIENS/RESTUANS     10     132   147  41.974689 -87.890615
10485   CULEX PIPIENS/RESTUANS     10     132   147  41.974689 -87.890615
10486            CULEX PIPIENS     10     132   147  41.974689 -87.890615
10487            CULEX PIPIENS     10     132   147  41.974689 -87.890615
10488            CULEX PIPIENS     10     132   147  41.974689 -87.890615
10489            CULEX PIPIENS     10     132   147  41.974689 -87.890615
10490   CULEX PIPIENS/RESTUANS     48     129   130  41.925198 -87.746381
10491   CULEX PIPIENS/RESTUANS     51      34   131  41.973845 -87.805059
10492            CULEX PIPIENS     51      34   131  41.973845 -87.805059
10493   CULEX PIPIENS/RESTUANS     82      72   133  41.743402 -87.731435
10494   CULEX PIPIENS/RESTUANS     65       9   135  41.728495 -87.600963
10495            CULEX PIPIENS     65       9   135  41.728495 -87.600963
10496   CULEX PIPIENS/RESTUANS     17     113   132  41.947227 -87.671457
10497   CULEX PIPIENS/RESTUANS     90     122   134  41.793818 -87.654234
10498   CULEX PIPIENS/RESTUANS     13      26   138  41.904194 -87.756155
10499            CULEX PIPIENS     13      26   138  41.904194 -87.756155
10500   CULEX PIPIENS/RESTUANS     39      44   136  41.951866 -87.725057
10501   CULEX PIPIENS/RESTUANS     51     108    27  41.763733 -87.742302
10502   CULEX PIPIENS/RESTUANS     58      42   139  41.987280 -87.666066
10503   CULEX PIPIENS/RESTUANS     17      12   140  41.912563 -87.668055
10504   CULEX PIPIENS/RESTUANS     71      21   141  42.009876 -87.807277
10505   CULEX PIPIENS/RESTUANS     42     106   143  41.776428 -87.627096

        WnvPresent month day  Tmax_x  Tmin_x Depart_x Depth_x SnowFall_x  \
0                0    05  29      88      60       10       0        0.0
1                0    05  29      88      60       10       0        0.0
2                0    05  29      88      60       10       0        0.0
3                0    05  29      88      60       10       0        0.0
4                0    05  29      88      60       10       0        0.0
5                0    05  29      88      60       10       0        0.0
```

6	0	05	29	88	60	10	0	0.0
7	0	05	29	88	60	10	0	0.0
8	0	05	29	88	60	10	0	0.0
9	0	05	29	88	60	10	0	0.0
10	0	05	29	88	60	10	0	0.0
11	0	05	29	88	60	10	0	0.0
12	0	05	29	88	60	10	0	0.0
13	0	05	29	88	60	10	0	0.0
14	0	05	29	88	60	10	0	0.0
15	0	05	29	88	60	10	0	0.0
16	0	05	29	88	60	10	0	0.0
17	0	05	29	88	60	10	0	0.0
18	0	05	29	88	60	10	0	0.0
19	0	05	29	88	60	10	0	0.0
20	0	05	29	88	60	10	0	0.0
21	0	05	29	88	60	10	0	0.0
22	0	05	29	88	60	10	0	0.0
23	0	05	29	88	60	10	0	0.0
24	0	05	29	88	60	10	0	0.0
25	0	06	05	64	47	-9	0	0.0
26	0	06	05	64	47	-9	0	0.0
27	0	06	05	64	47	-9	0	0.0
28	0	06	05	64	47	-9	0	0.0
29	0	06	05	64	47	-9	0	0.0
...	...	...	..	...	...	...	...	...
10476	0	09	26	75	50	3	0	0.0
10477	0	09	26	75	50	3	0	0.0
10478	0	09	26	75	50	3	0	0.0
10479	0	09	26	75	50	3	0	0.0
10480	0	09	26	75	50	3	0	0.0
10481	0	09	26	75	50	3	0	0.0
10482	0	09	26	75	50	3	0	0.0
10483	0	09	26	75	50	3	0	0.0
10484	1	09	26	75	50	3	0	0.0
10485	0	09	26	75	50	3	0	0.0
10486	0	09	26	75	50	3	0	0.0
10487	0	09	26	75	50	3	0	0.0
10488	0	09	26	75	50	3	0	0.0
10489	0	09	26	75	50	3	0	0.0
10490	0	09	26	75	50	3	0	0.0
10491	0	09	26	75	50	3	0	0.0
10492	0	09	26	75	50	3	0	0.0
10493	0	09	26	75	50	3	0	0.0
10494	0	09	26	75	50	3	0	0.0
10495	0	09	26	75	50	3	0	0.0
10496	0	09	26	75	50	3	0	0.0
10497	0	09	26	75	50	3	0	0.0
10498	0	09	26	75	50	3	0	0.0

10499	0	09	26	75	50	3	0	0.0
10500	0	09	26	75	50	3	0	0.0
10501	1	09	26	75	50	3	0	0.0
10502	0	09	26	75	50	3	0	0.0
10503	0	09	26	75	50	3	0	0.0
10504	0	09	26	75	50	3	0	0.0
10505	0	09	26	75	50	3	0	0.0

	PrecipTotal_x	AvgSpeed_x	Tmax_y	Tmin_y	PrecipTotal_y	AvgSpeed_y
0	0.00	6.5	88	65	0.00	7.4
1	0.00	6.5	88	65	0.00	7.4
2	0.00	6.5	88	65	0.00	7.4
3	0.00	6.5	88	65	0.00	7.4
4	0.00	6.5	88	65	0.00	7.4
5	0.00	6.5	88	65	0.00	7.4
6	0.00	6.5	88	65	0.00	7.4
7	0.00	6.5	88	65	0.00	7.4
8	0.00	6.5	88	65	0.00	7.4
9	0.00	6.5	88	65	0.00	7.4
10	0.00	6.5	88	65	0.00	7.4
11	0.00	6.5	88	65	0.00	7.4
12	0.00	6.5	88	65	0.00	7.4
13	0.00	6.5	88	65	0.00	7.4
14	0.00	6.5	88	65	0.00	7.4
15	0.00	6.5	88	65	0.00	7.4
16	0.00	6.5	88	65	0.00	7.4
17	0.00	6.5	88	65	0.00	7.4
18	0.00	6.5	88	65	0.00	7.4
19	0.00	6.5	88	65	0.00	7.4
20	0.00	6.5	88	65	0.00	7.4
21	0.00	6.5	88	65	0.00	7.4
22	0.00	6.5	88	65	0.00	7.4
23	0.00	6.5	88	65	0.00	7.4
24	0.00	6.5	88	65	0.00	7.4
25	0.42	7.6	63	51	0.27	8.1
26	0.42	7.6	63	51	0.27	8.1
27	0.42	7.6	63	51	0.27	8.1
28	0.42	7.6	63	51	0.27	8.1
29	0.42	7.6	63	51	0.27	8.1
...	...	...	...	...	...	...
10476	0.00	4.2	75	55	0.00	4.6
10477	0.00	4.2	75	55	0.00	4.6
10478	0.00	4.2	75	55	0.00	4.6
10479	0.00	4.2	75	55	0.00	4.6
10480	0.00	4.2	75	55	0.00	4.6
10481	0.00	4.2	75	55	0.00	4.6
10482	0.00	4.2	75	55	0.00	4.6
10483	0.00	4.2	75	55	0.00	4.6

```
10484          0.00      4.2    75    55        0.00      4.6
10485          0.00      4.2    75    55        0.00      4.6
10486          0.00      4.2    75    55        0.00      4.6
10487          0.00      4.2    75    55        0.00      4.6
10488          0.00      4.2    75    55        0.00      4.6
10489          0.00      4.2    75    55        0.00      4.6
10490          0.00      4.2    75    55        0.00      4.6
10491          0.00      4.2    75    55        0.00      4.6
10492          0.00      4.2    75    55        0.00      4.6
10493          0.00      4.2    75    55        0.00      4.6
10494          0.00      4.2    75    55        0.00      4.6
10495          0.00      4.2    75    55        0.00      4.6
10496          0.00      4.2    75    55        0.00      4.6
10497          0.00      4.2    75    55        0.00      4.6
10498          0.00      4.2    75    55        0.00      4.6
10499          0.00      4.2    75    55        0.00      4.6
10500          0.00      4.2    75    55        0.00      4.6
10501          0.00      4.2    75    55        0.00      4.6
10502          0.00      4.2    75    55        0.00      4.6
10503          0.00      4.2    75    55        0.00      4.6
10504          0.00      4.2    75    55        0.00      4.6
10505          0.00      4.2    75    55        0.00      4.6

[10506 rows x 20 columns]
```

In [46]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori
```

In [90]:
```
for elem in test['Species'].unique():
    test[str(elem)] = test['Species'] == elem
```

In [94]:
```
cat_names = {'01':'Jan', '02':'Feb', '03':'Mar','04':'Apr','05':'May','06':'June','07
for elem in test['month'].unique():
    test[cat_names[elem]] = test['month'] == elem
```

In [132]:
```
cat_names = {40-50:'cold', 70:'Hot', 64:'warm',88:'Hot',92:'Very Hot',91-100:'Boilin
             83:'Very Hot',91:'Very Hot', 90:'Very Hot', 87:'Very Hot',84:'Very Hot',
             60:'warm',61:'warm',62:'warm',63:'warm',64:'warm',65:'warm',66:'warm',67
             70:'Hot',71:'Hot',72:'Hot',73:'Hot',74:'Hot',75:'Hot',76:'Hot',77:'Hot',
             86:'Very Hot',
              90:'Very Hot',91:'Very Hot',92:'Very Hot',93:'Very Hot',94:'Very Hot',9
             50:'cool',51:'cool',52:'cool',53:'cool',54:'cool',55:'cool',56:'cool',57
 for elem in test['Tmax_x'].unique():
     test[cat_names[elem]] = test['Tmax_x'] == elem
```

In [135]:
```
cat_names = {40-50:'cold', 70:'Hot', 64:'warm',88:'Hot',92:'Very Hot',91-100:'Boilin
             83:'Very Hot',91:'Very Hot', 90:'Very Hot', 87:'Very Hot',84:'Very Hot',
             60:'warm',61:'warm',62:'warm',63:'warm',64:'warm',65:'warm',66:'warm',67
             70:'Hot',71:'Hot',72:'Hot',73:'Hot',74:'Hot',75:'Hot',76:'Hot',77:'Hot',
```

```
                 86:'Very Hot',
                  90:'Very Hot',91:'Very Hot',92:'Very Hot',93:'Very Hot',94:'Very Hot',95
                 50:'cool',51:'cool',52:'cool',53:'cool',54:'cool',55:'cool',56:'cool',57
                 40:'cold',41:'cold',42:'cold',43:'cold',44:'cold',45:'cold',46:'cold',47
        for elem in test['Tmin_x'].unique():
            test[cat_names[elem]] = test['Tmin_x'] == elem
```

```
In [143]: cat_names = {'0.00':'dry','0.42':'some','0.16':'some','1.55':'wet','-1':'dry'}
          for elem in test['PrecipTotal_x'].unique():
              test[cat_names[elem]] = test['PrecipTotal_x'] == elem
```

```
        KeyErrorTraceback (most recent call last)

        <ipython-input-143-f79e373e2287> in <module>()
          1 cat_names = {'0.00':'dry','0.42':'some','0.16':'some','1.55':'wet','-1':'dry'}
          2 for elem in test['PrecipTotal_x'].unique():
    ----> 3     test[cat_names[elem]] = test['PrecipTotal_x'] == elem


        KeyError: -1
```

```
In [72]: for elem in test['PrecipTotal_x'].unique():
             test[str(elem)] = test['PrecipTotal_x'] == elem

In [144]: test=test.drop(['Latitude','Longitude','month','day','Block'], axis=1)

In [145]: test=test.drop(['Depart_x','Depth_x','Trap','SnowFall_x','AvgSpeed_x','PrecipTotal_x

In [146]: test=test.drop(['Species','Street','AvgSpeed_y'], axis=1)

In [147]: test
```

```
Out[147]:        WnvPresent    May   June  CULEX PIPIENS/RESTUANS  CULEX RESTUANS  \
          0               0   True  False                    True           False
          1               0   True  False                   False            True
          2               0   True  False                   False            True
          3               0   True  False                    True           False
          4               0   True  False                   False            True
          5               0   True  False                   False            True
          6               0   True  False                   False            True
          7               0   True  False                    True           False
          8               0   True  False                   False            True
          9               0   True  False                   False            True
          10              0   True  False                    True           False
          11              0   True  False                    True           False
```

12	0	True	False	False	True
13	0	True	False	True	False
14	0	True	False	False	True
15	0	True	False	False	True
16	0	True	False	False	True
17	0	True	False	False	True
18	0	True	False	False	False
19	0	True	False	True	False
20	0	True	False	False	True
21	0	True	False	True	False
22	0	True	False	True	False
23	0	True	False	False	True
24	0	True	False	False	True
25	0	False	True	True	False
26	0	False	True	False	True
27	0	False	True	False	False
28	0	False	True	True	False
29	0	False	True	False	True
...	...	...	...	...	...
10476	0	False	False	True	False
10477	0	False	False	False	True
10478	0	False	False	False	False
10479	0	False	False	False	False
10480	0	False	False	True	False
10481	0	False	False	False	True
10482	0	False	False	False	False
10483	0	False	False	False	True
10484	1	False	False	True	False
10485	0	False	False	True	False
10486	0	False	False	False	False
10487	0	False	False	False	False
10488	0	False	False	False	False
10489	0	False	False	False	False
10490	0	False	False	True	False
10491	0	False	False	True	False
10492	0	False	False	False	False
10493	0	False	False	True	False
10494	0	False	False	True	False
10495	0	False	False	False	False
10496	0	False	False	True	False
10497	0	False	False	True	False
10498	0	False	False	True	False
10499	0	False	False	False	False
10500	0	False	False	True	False
10501	1	False	False	True	False
10502	0	False	False	True	False
10503	0	False	False	True	False
10504	0	False	False	True	False

```
10505           0  False  False                 True           False

       CULEX PIPIENS  CULEX SALINARIUS  CULEX TERRITANS  CULEX TARSALIS  \
0              False             False            False           False
1              False             False            False           False
2              False             False            False           False
3              False             False            False           False
4              False             False            False           False
5              False             False            False           False
6              False             False            False           False
7              False             False            False           False
8              False             False            False           False
9              False             False            False           False
10             False             False            False           False
11             False             False            False           False
12             False             False            False           False
13             False             False            False           False
14             False             False            False           False
15             False             False            False           False
16             False             False            False           False
17             False             False            False           False
18              True             False            False           False
19             False             False            False           False
20             False             False            False           False
21             False             False            False           False
22             False             False            False           False
23             False             False            False           False
24             False             False            False           False
25             False             False            False           False
26             False             False            False           False
27              True             False            False           False
28             False             False            False           False
29             False             False            False           False
...              ...               ...              ...             ...
10476          False             False            False           False
10477          False             False            False           False
10478           True             False            False           False
10479           True             False            False           False
10480          False             False            False           False
10481          False             False            False           False
10482           True             False            False           False
10483          False             False            False           False
10484          False             False            False           False
10485          False             False            False           False
10486           True             False            False           False
10487           True             False            False           False
10488           True             False            False           False
10489           True             False            False           False
```

```
10490          False           False           False           False
10491          False           False           False           False
10492           True           False           False           False
10493          False           False           False           False
10494          False           False           False           False
10495           True           False           False           False
10496          False           False           False           False
10497          False           False           False           False
10498          False           False           False           False
10499           True           False           False           False
10500          False           False           False           False
10501          False           False           False           False
10502          False           False           False           False
10503          False           False           False           False
10504          False           False           False           False
10505          False           False           False           False

      CULEX ERRATICUS  ...   very hot    Hot  Very hot  Veryhot  Very Hot  \
0               False  ...      False  False     False    False     False
1               False  ...      False  False     False    False     False
2               False  ...      False  False     False    False     False
3               False  ...      False  False     False    False     False
4               False  ...      False  False     False    False     False
5               False  ...      False  False     False    False     False
6               False  ...      False  False     False    False     False
7               False  ...      False  False     False    False     False
8               False  ...      False  False     False    False     False
9               False  ...      False  False     False    False     False
10              False  ...      False  False     False    False     False
11              False  ...      False  False     False    False     False
12              False  ...      False  False     False    False     False
13              False  ...      False  False     False    False     False
14              False  ...      False  False     False    False     False
15              False  ...      False  False     False    False     False
16              False  ...      False  False     False    False     False
17              False  ...      False  False     False    False     False
18              False  ...      False  False     False    False     False
19              False  ...      False  False     False    False     False
20              False  ...      False  False     False    False     False
21              False  ...      False  False     False    False     False
22              False  ...      False  False     False    False     False
23              False  ...      False  False     False    False     False
24              False  ...      False  False     False    False     False
25              False  ...      False  False     False    False     False
26              False  ...      False  False     False    False     False
27              False  ...      False  False     False    False     False
28              False  ...      False  False     False    False     False
29              False  ...      False  False     False    False     False
```

```
...              ...  ...        ...    ...       ...     ...       ...
10476          False  ...      False  False     False   False     False
10477          False  ...      False  False     False   False     False
10478          False  ...      False  False     False   False     False
10479          False  ...      False  False     False   False     False
10480          False  ...      False  False     False   False     False
10481          False  ...      False  False     False   False     False
10482          False  ...      False  False     False   False     False
10483          False  ...      False  False     False   False     False
10484          False  ...      False  False     False   False     False
10485          False  ...      False  False     False   False     False
10486          False  ...      False  False     False   False     False
10487          False  ...      False  False     False   False     False
10488          False  ...      False  False     False   False     False
10489          False  ...      False  False     False   False     False
10490          False  ...      False  False     False   False     False
10491          False  ...      False  False     False   False     False
10492          False  ...      False  False     False   False     False
10493          False  ...      False  False     False   False     False
10494          False  ...      False  False     False   False     False
10495          False  ...      False  False     False   False     False
10496          False  ...      False  False     False   False     False
10497          False  ...      False  False     False   False     False
10498          False  ...      False  False     False   False     False
10499          False  ...      False  False     False   False     False
10500          False  ...      False  False     False   False     False
10501          False  ...      False  False     False   False     False
10502          False  ...      False  False     False   False     False
10503          False  ...      False  False     False   False     False
10504          False  ...      False  False     False   False     False
10505          False  ...      False  False     False   False     False

        cool   cold    dry   some    wet
0      False  False   True  False  False
1      False  False   True  False  False
2      False  False   True  False  False
3      False  False   True  False  False
4      False  False   True  False  False
5      False  False   True  False  False
6      False  False   True  False  False
7      False  False   True  False  False
8      False  False   True  False  False
9      False  False   True  False  False
10     False  False   True  False  False
11     False  False   True  False  False
12     False  False   True  False  False
13     False  False   True  False  False
14     False  False   True  False  False
```

```
15     False  False   True  False  False
16     False  False   True  False  False
17     False  False   True  False  False
18     False  False   True  False  False
19     False  False   True  False  False
20     False  False   True  False  False
21     False  False   True  False  False
22     False  False   True  False  False
23     False  False   True  False  False
24     False  False   True  False  False
25     False  False  False  False  False
26     False  False  False  False  False
27     False  False  False  False  False
28     False  False  False  False  False
29     False  False  False  False  False
...      ...    ...    ...    ...    ...
10476  False  False   True  False  False
10477  False  False   True  False  False
10478  False  False   True  False  False
10479  False  False   True  False  False
10480  False  False   True  False  False
10481  False  False   True  False  False
10482  False  False   True  False  False
10483  False  False   True  False  False
10484  False  False   True  False  False
10485  False  False   True  False  False
10486  False  False   True  False  False
10487  False  False   True  False  False
10488  False  False   True  False  False
10489  False  False   True  False  False
10490  False  False   True  False  False
10491  False  False   True  False  False
10492  False  False   True  False  False
10493  False  False   True  False  False
10494  False  False   True  False  False
10495  False  False   True  False  False
10496  False  False   True  False  False
10497  False  False   True  False  False
10498  False  False   True  False  False
10499  False  False   True  False  False
10500  False  False   True  False  False
10501  False  False   True  False  False
10502  False  False   True  False  False
10503  False  False   True  False  False
10504  False  False   True  False  False
10505  False  False   True  False  False

[10506 rows x 26 columns]
```

In [116]: list(test)

```
Out[116]: ['Species',
 'Block',
 'Street',
 'Trap',
 'Latitude',
 'Longitude',
 'WnvPresent',
 'month',
 'day',
 'Tmax_x',
 'Tmin_x',
 'Depart_x',
 'Depth_x',
 'SnowFall_x',
 'PrecipTotal_x',
 'AvgSpeed_x',
 'Tmax_y',
 'Tmin_y',
 'PrecipTotal_y',
 'AvgSpeed_y',
 'May',
 'June',
 'CULEX PIPIENS/RESTUANS',
 'CULEX RESTUANS',
 'CULEX PIPIENS',
 'CULEX SALINARIUS',
 'CULEX TERRITANS',
 'CULEX TARSALIS',
 'CULEX ERRATICUS',
 'July',
 'Aug',
 'Sept',
 'Oct',
 'hot',
 'warm',
 'very hot',
 'Hot',
 'Very hot',
 'Veryhot']
```

In [150]:
```
frequent_itemsets = apriori(test, min_support=0.001, use_colnames=True)
```

In [152]:
```
from mlxtend.frequent_patterns import association_rules

rules=association_rules(frequent_itemsets, metric="confidence", min_threshold=0.1)
```

In [153]:
```
rules
```

```
Out[153]:                                           antecedents  \
0                                  (dry, CULEX TERRITANS)
1                                 (CULEX TERRITANS, warm)
2                                             (dry, Oct)
3                           (CULEX PIPIENS/RESTUANS, Oct)
4                                                  (Oct)
5         (dry, CULEX PIPIENS/RESTUANS, WnvPresent, Aug)
6      (dry, CULEX PIPIENS/RESTUANS, very hot, WnvPre...
7                        (dry, WnvPresent, Aug, very hot)
8      (CULEX PIPIENS/RESTUANS, WnvPresent, Aug, very...
9                            (dry, very hot, WnvPresent)
10        (CULEX PIPIENS/RESTUANS, very hot, WnvPresent)
11                           (WnvPresent, Aug, very hot)
12                                (very hot, WnvPresent)
13                                                 (wet)
14                                                 (Oct)
15                         (CULEX PIPIENS/RESTUANS, warm)
16                                          (warm, June)
17                                      (Sept, very hot)
18                          (WnvPresent, CULEX RESTUANS)
19                                            (very hot)
20                                                 (May)
21                                                (some)
22                                            (very hot)
23                                          (July, some)
24                                 (some, CULEX PIPIENS)
25                                           (dry, June)
26                         (CULEX PIPIENS/RESTUANS, June)
27                                                (June)
28                                           (dry, Sept)
29                                 (dry, CULEX RESTUANS)
..                                                   ...
930                                (hot, CULEX RESTUANS)
931                                          (dry, July)
932                                           (dry, hot)
933                                          (July, hot)
934                                                (hot)
935                            (Veryhot, CULEX RESTUANS)
936                                           (dry, hot)
937                         (CULEX PIPIENS/RESTUANS, hot)
938                                                (hot)
939                                          (dry, July)
940                               (July, CULEX PIPIENS)
941                                   (CULEX SALINARIUS)
942                        (CULEX PIPIENS/RESTUANS, cold)
943                                         (cold, Sept)
944                                               (cold)
945                                   (CULEX SALINARIUS)
```

```
946                     (CULEX PIPIENS/RESTUANS)
947                                       (July)
948            (July, Very hot, CULEX RESTUANS)
949                       (July, Very hot, wet)
950                 (July, CULEX RESTUANS, wet)
951             (Very hot, CULEX RESTUANS, wet)
952                            (July, Very hot)
953                                 (July, wet)
954                  (Very hot, CULEX RESTUANS)
955                             (Very hot, wet)
956                       (CULEX RESTUANS, wet)
957                                  (Very hot)
958                                       (wet)
959                                      (some)

                          consequents  antecedent support  \
0                              (warm)            0.012184
1                               (dry)            0.001523
2            (CULEX PIPIENS/RESTUANS)            0.020084
3                               (dry)            0.012088
4       (dry, CULEX PIPIENS/RESTUANS)            0.026271
5                          (very hot)            0.009899
6                               (Aug)            0.001523
7            (CULEX PIPIENS/RESTUANS)            0.001713
8                               (dry)            0.001333
9       (CULEX PIPIENS/RESTUANS, Aug)            0.001999
10                         (dry, Aug)            0.001523
11      (dry, CULEX PIPIENS/RESTUANS)            0.001713
12 (dry, CULEX PIPIENS/RESTUANS, Aug)            0.001999
13                             (July)            0.012279
14           (CULEX PIPIENS/RESTUANS)            0.026271
15                             (June)            0.024272
16           (CULEX PIPIENS/RESTUANS)            0.005901
17                    (CULEX PIPIENS)            0.004759
18                              (dry)            0.004664
19           (CULEX PIPIENS/RESTUANS)            0.063868
20                              (dry)            0.007995
21                         (very hot)            0.022844
22                             (some)            0.063868
23                    (CULEX PIPIENS)            0.004283
24                             (July)            0.004474
25           (CULEX PIPIENS/RESTUANS)            0.058252
26                              (dry)            0.067009
27      (dry, CULEX PIPIENS/RESTUANS)            0.149534
28                   (CULEX RESTUANS)            0.153912
29                             (Sept)            0.142014
..                                ...                 ...
930                            (June)            0.018846
```

931	(hot)	0.116695
932	(July)	0.028270
933	(dry)	0.030935
934	(dry, July)	0.053684
935	(Aug)	0.003807
936	(CULEX PIPIENS/RESTUANS)	0.028270
937	(dry)	0.022844
938	(dry, CULEX PIPIENS/RESTUANS)	0.053684
939	(CULEX PIPIENS)	0.116695
940	(dry)	0.033124
941	(Sept)	0.008186
942	(Sept)	0.003236
943	(CULEX PIPIENS/RESTUANS)	0.005806
944	(CULEX PIPIENS/RESTUANS, Sept)	0.005806
945	(July)	0.008186
946	(July)	0.452313
947	(CULEX PIPIENS/RESTUANS)	0.248049
948	(wet)	0.003903
949	(CULEX RESTUANS)	0.012279
950	(Very hot)	0.003807
951	(July)	0.003807
952	(CULEX RESTUANS, wet)	0.021036
953	(Very hot, CULEX RESTUANS)	0.012279
954	(July, wet)	0.003998
955	(July, CULEX RESTUANS)	0.012279
956	(July, Very hot)	0.003807
957	(July, CULEX RESTUANS, wet)	0.023225
958	(July, Very hot, CULEX RESTUANS)	0.012279
959	(June)	0.022844

	consequent support	support	confidence	lift	leverage	conviction
0	0.050733	0.001237	0.101562	2.001905	0.000619	1.056576
1	0.563392	0.001237	0.812500	1.442157	0.000379	2.328574
2	0.452313	0.009233	0.459716	1.016366	0.000149	1.013701
3	0.563392	0.009233	0.763780	1.355680	0.002422	1.848306
4	0.256235	0.009233	0.351449	1.371592	0.002501	1.146812
5	0.063868	0.001333	0.134615	2.107704	0.000700	1.081752
6	0.357034	0.001333	0.875000	2.450746	0.000789	5.143727
7	0.452313	0.001333	0.777778	1.719557	0.000558	2.464592
8	0.563392	0.001333	1.000000	1.774962	0.000582	inf
9	0.155625	0.001333	0.666667	4.283792	0.001021	2.533124
10	0.206453	0.001333	0.875000	4.238243	0.001018	6.348372
11	0.256235	0.001333	0.777778	3.035414	0.000894	3.346945
12	0.090710	0.001333	0.666667	7.349423	0.001151	2.727870
13	0.248049	0.012279	1.000000	4.031466	0.009233	inf
14	0.452313	0.012088	0.460145	1.017315	0.000206	1.014508
15	0.149534	0.003331	0.137255	0.917887	-0.000298	0.985768
16	0.452313	0.003331	0.564516	1.248065	0.000662	1.257652

```
17               0.256901  0.001237    0.260000   1.012064  0.000015    1.004188
18               0.563392  0.002475    0.530612   0.941817 -0.000153    0.930164
19               0.452313  0.027127    0.424739   0.939038 -0.001761    0.952067
20               0.563392  0.007995    1.000000   1.774962  0.003491         inf
21               0.063868  0.006663    0.291667   4.566692  0.005204    1.321598
22               0.022844  0.006663    0.104322   4.566692  0.005204    1.090968
23               0.256901  0.001047    0.244444   0.951513 -0.000053    0.983514
24               0.248049  0.001047    0.234043   0.943535 -0.000063    0.981714
25               0.452313  0.025604    0.439542   0.971766 -0.000744    0.977214
26               0.563392  0.025604    0.382102   0.678217 -0.012148    0.706602
27               0.256235  0.025604    0.171229   0.668249 -0.012711    0.897431
28               0.260803  0.025795    0.167594   0.642608 -0.014346    0.888025
29               0.211117  0.025795    0.181635   0.860352 -0.004187    0.963974
..                    ...       ...         ...        ...       ...         ...
930              0.149534  0.005140    0.272727   1.823853  0.002322    1.169391
931              0.053684  0.017038    0.146003   2.719699  0.010773    1.108103
932              0.248049  0.017038    0.602694   2.429739  0.010026    1.892623
933              0.563392  0.017038    0.550769   0.977594 -0.000390    0.971901
934              0.116695  0.017038    0.317376   2.719699  0.010773    1.293984
935              0.357034  0.003807    1.000000   2.800853  0.002448         inf
936              0.452313  0.011993    0.424242   0.937940 -0.000794    0.951246
937              0.563392  0.011993    0.525000   0.931855 -0.000877    0.919174
938              0.256235  0.011993    0.223404   0.871874 -0.001762    0.957725
939              0.256901  0.014087    0.120718   0.469900 -0.015892    0.845120
940              0.563392  0.014087    0.425287   0.754869 -0.004575    0.759697
941              0.211117  0.001428    0.174419   0.826169 -0.000300    0.955548
942              0.211117  0.003236    1.000000   4.736700  0.002553         inf
943              0.452313  0.003236    0.557377   1.232282  0.000610    1.237367
944              0.099753  0.003236    0.557377   5.587599  0.002657    2.033892
945              0.248049  0.002760    0.337209   1.359448  0.000730    1.134523
946              0.248049  0.113840    0.251684   1.014653  0.001644    1.004857
947              0.452313  0.113840    0.458941   1.014653  0.001644    1.012250
948              0.012279  0.003807    0.975610  79.455474  0.003759   40.496573
949              0.260803  0.003807    0.310078   1.188932  0.000605    1.071420
950              0.023225  0.003807    1.000000  43.057377  0.003719         inf
951              0.248049  0.003807    1.000000   4.031466  0.002863         inf
952              0.003807  0.003807    0.180995  47.538462  0.003727    1.216346
953              0.003998  0.003807    0.310078  77.563677  0.003758    1.443644
954              0.012279  0.003807    0.952381  77.563677  0.003758   20.742147
955              0.093566  0.003807    0.310078   3.314013  0.002658    1.313821
956              0.021036  0.003807    1.000000  47.538462  0.003727         inf
957              0.003807  0.003807    0.163934  43.057377  0.003719    1.191525
958              0.003903  0.003807    0.310078  79.455474  0.003759    1.443782
959              0.149534  0.006663    0.291667   1.950509  0.003247    1.200658

[960 rows x 9 columns]
```

When using association it is important to understand what some of the terms are. So, confidence is the reliability of a rule, so the higher the confidence the more likely Y is to occur with the set of X. It is essientaly the conditional probability that Y occurs given X. Support is the fequency that item sets appear in the data set, so we want a higher support to eliminate rules that may happen by chance. I needed to set the support threshold to be relatively low since the data set is very large. Lift is the ratio of support if X and Y are independent. If a rule has a lift above 1 we can consider this rule ot be useful, while a lift below 1 means the presence of one item may have a negative effect on another.

In [155]:
```
rules[rules['consequents'] == {'WnvPresent'}]
```

Out[155]:

	antecedents	consequents	antecedent support
36	(CULEX PIPIENS, Aug)	(WnvPresent)	0.133448
128	(dry, cool)	(WnvPresent)	0.014944
258	(CULEX PIPIENS/RESTUANS, Aug)	(WnvPresent)	0.155625
268	(cool)	(WnvPresent)	0.014944
483	(Aug)	(WnvPresent)	0.357034
531	(dry, CULEX PIPIENS, Aug)	(WnvPresent)	0.070721
606	(Aug, cool)	(WnvPresent)	0.014944
721	(dry, CULEX PIPIENS/RESTUANS, Aug)	(WnvPresent)	0.090710
752	(dry, Aug, cool)	(WnvPresent)	0.014944

	consequent support	support	confidence	lift	leverage	conviction
36	0.052446	0.017228	0.129101	2.461594	0.010229	1.088018
128	0.052446	0.001904	0.127389	2.428936	0.001120	1.085883
258	0.052446	0.016276	0.104587	1.994179	0.008114	1.058231
268	0.052446	0.001904	0.127389	2.428936	0.001120	1.085883
483	0.052446	0.035884	0.100507	1.916373	0.017159	1.053430
531	0.052446	0.007329	0.103634	1.976003	0.003620	1.057106
606	0.052446	0.001904	0.127389	2.428936	0.001120	1.085883
721	0.052446	0.009899	0.109129	2.080780	0.005142	1.063626
752	0.052446	0.001904	0.127389	2.428936	0.001120	1.085883

These are the rules that invole Wnv presence. We see that often when it is august, and Culex pipiens mosquitoes WNV is present. These rules confirm the above results that species is important at determing the presence of WNV.

In this project I have completed my goal of predicting outbreaks of west nile virus in chicago. In the feature selection we have shown that species of mosquito appears to be the most important for the modeling determing the presence of the virus, and this seems to be confirmed in the rule set. The rule with the higherst confidence, lift, and support shows that Culex pippiens mosquitoes in august have WNV. These results confirm a lot of the written literature about species being better or worse carries for the virus.

The clustering analysis showed two clear groups, which may show days where WNV is more likely to be present, versus days when WNV is less likely to be present.

I think with more time I could have gotten the predictive models a better accuracy and AUC score.

In the future I think more data could be added to the set overall, such as the virus in human and bird populations, or humidity of the city. This model may be a good start for predicting the future outbreaks of disease and could help city officials spray insecticides in areas predicted to have high levels of virus.