In [1]:
```
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
```

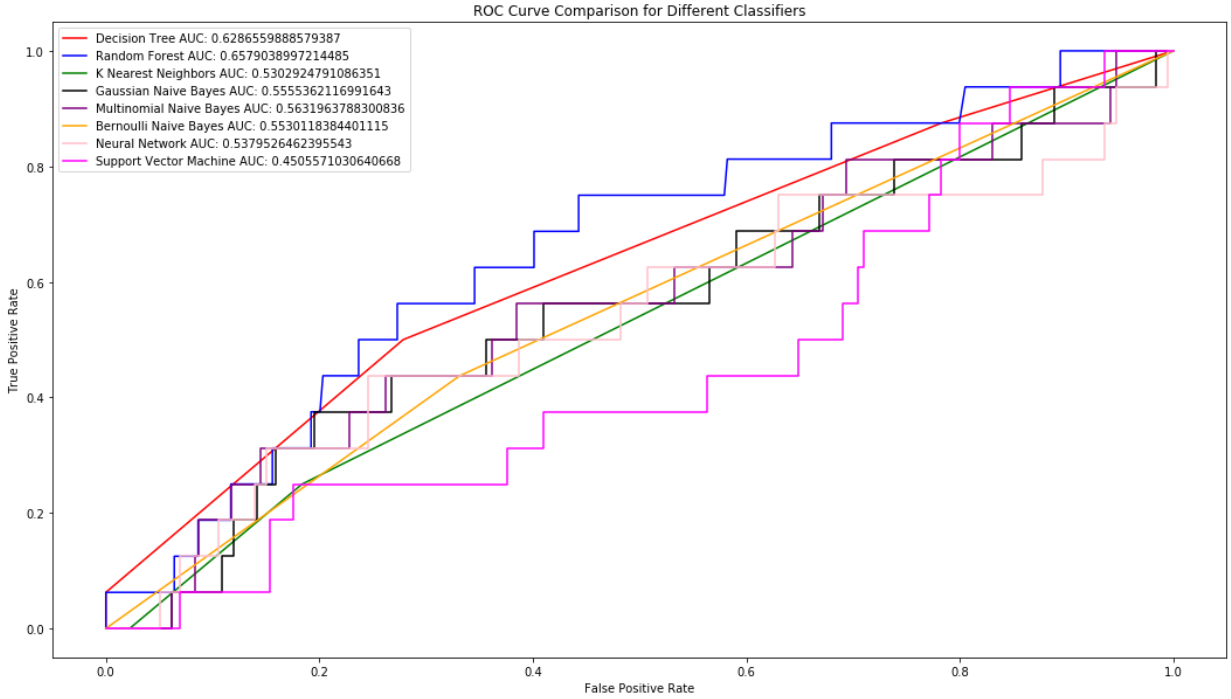
In [2]:
```
features = np.genfromtxt("./Aggregated_Data.csv", delimiter=",", usecols=(1
targets = np.genfromtxt("./Aggregated_Data.csv", delimiter=",", usecols=8)

trainingFeatures, testingFeatures, trainingTargets, testingTargets = train_
```

In [10]:
```
#follows the Confusion Matrix information page on Scikit Learn to visualize
#https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusi
def plotMatrix(confusionMatrix, title, color):
    classLabels = ["No Outbreak", "Outbreak"]
    plt.rcParams['figure.figsize'] = (5.0, 5.0)
    plt.imshow(confusionMatrix, cmap=plt.cm.Blues)
    plt.xticks(ticks=[0,1], labels=classLabels)
    plt.yticks(ticks=[0,1], labels=classLabels)
    plt.ylabel("Predicted Labels")
    plt.xlabel("True Labels")
    plt.title(title)
    maximum = confusionMatrix.max()/2
    plt.text(0, 0, str(confusionMatrix[0][0]), horizontalalignment="center"
    plt.text(0, 1, str(confusionMatrix[0][1]), horizontalalignment="center"
    plt.text(1, 0, str(confusionMatrix[1][0]), horizontalalignment="center"
    plt.text(1, 1, str(confusionMatrix[1][1]), horizontalalignment="center"
    plt.show()
```

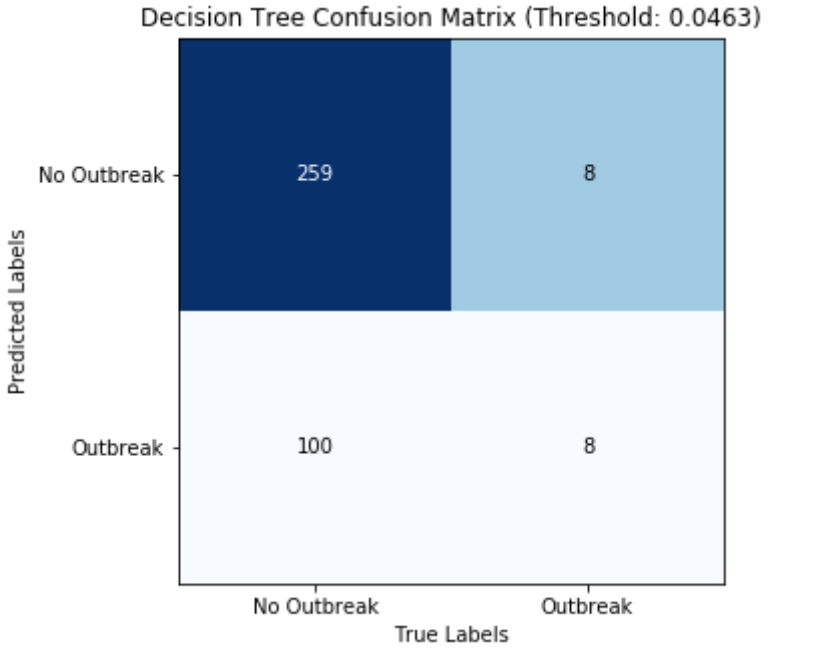
In [9]:
```
def thresholdBasedClassification(threshold, classifier, label, testingFeatu
    predictions = []
    probabilities = classifier.predict_proba(testingFeatures)
    for j in range(0, probabilities.shape[0]):
        if (probabilities[j][1] <= threshold):
            predictions.append(0)
        else:
            predictions.append(1)
    ##https://scikit-learn.org/stable/modules/generated/sklearn.metrics.con
    #uses the confusion_matrix documentation on Scikit Learn to calculate t
    confusionMatrix = confusion_matrix(testingTargets, predictions)
    #uses the Confusion Matrix information page on Scikit Learn to find out
    #https://scikit-learn.org/stable/auto_examples/model_selection/plot_con
    normalizedConfusionMatrix = confusionMatrix.astype('float') / confusion
    plotMatrix(confusionMatrix, label + (" Confusion Matrix (Threshold: %.4
    plotMatrix(normalizedConfusionMatrix, label + (" Normalized Confusion M
```

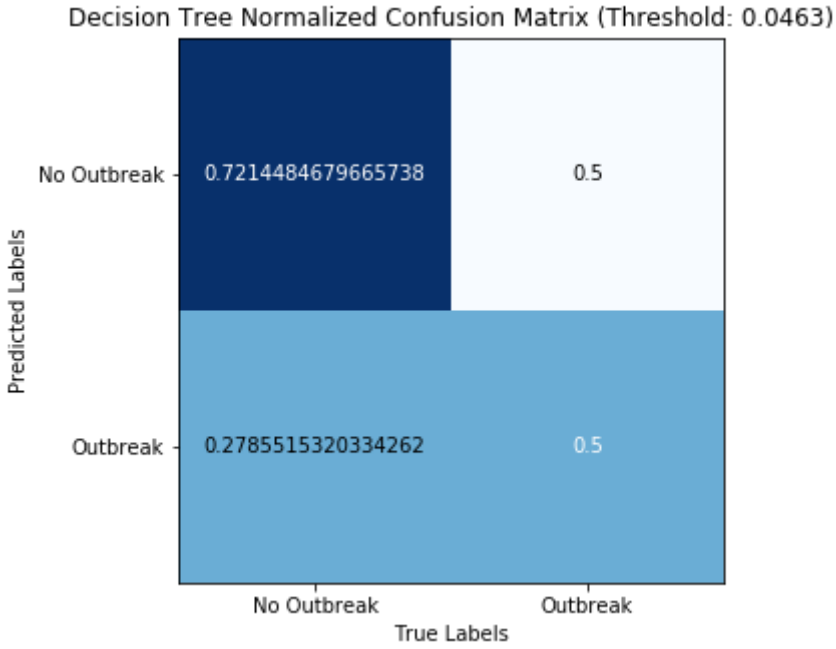
In [5]:

```
#The citations for the classifiers can be found in their respective files
decisionTree = DecisionTreeClassifier(criterion = "entropy", max_depth = 4)
decisionTree.fit(trainingFeatures, trainingTargets)

randomForest = RandomForestClassifier(n_estimators=100, max_depth=4, random
randomForest.fit(trainingFeatures, trainingTargets)

knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(trainingFeatures, trainingTargets)

gaussianNaiveBayes = GaussianNB()
gaussianNaiveBayes.fit(trainingFeatures, trainingTargets)
multinomialNaiveBayes = MultinomialNB()
multinomialNaiveBayes.fit(trainingFeatures, trainingTargets)
bernoulliNaiveBayes = BernoulliNB(binarize=0.1)
bernoulliNaiveBayes.fit(trainingFeatures, trainingTargets)

network = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(3,)
network.fit(trainingFeatures, trainingTargets)

svm = SVC(kernel='poly', degree=3, random_state=0, gamma='auto', probabilit
svm.fit(trainingFeatures, trainingTargets)

classifiers = [decisionTree, randomForest, knn, gaussianNaiveBayes, multinc
colors = ['red', 'blue', 'green', 'black', 'purple', 'orange', 'pink', 'mag
labels = ["Decision Tree", "Random Forest", 'K Nearest Neighbors', "Gaussia

thresholds = []

plt.rcParams['figure.figsize'] = (18.0, 10.0)
for i in range(0,8):
    outbreakProbabilities = classifiers[i].predict_proba(testingFeatures)[:
    #uses the documentation on Scikit Learn for roc_curve to calculate the
    #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_
    fpr, tpr, tresh = roc_curve(testingTargets, outbreakProbabilities)
    thresholds.append(np.median(outbreakProbabilities))
    #uses the documentation on Scikit Learn for auc on how to calculate auc
    #https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.
    area = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], label=labels[i]+" AUC: "+str(area))
    plt.legend()
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC Curve Comparison for Different Classifiers")
plt.show()
```

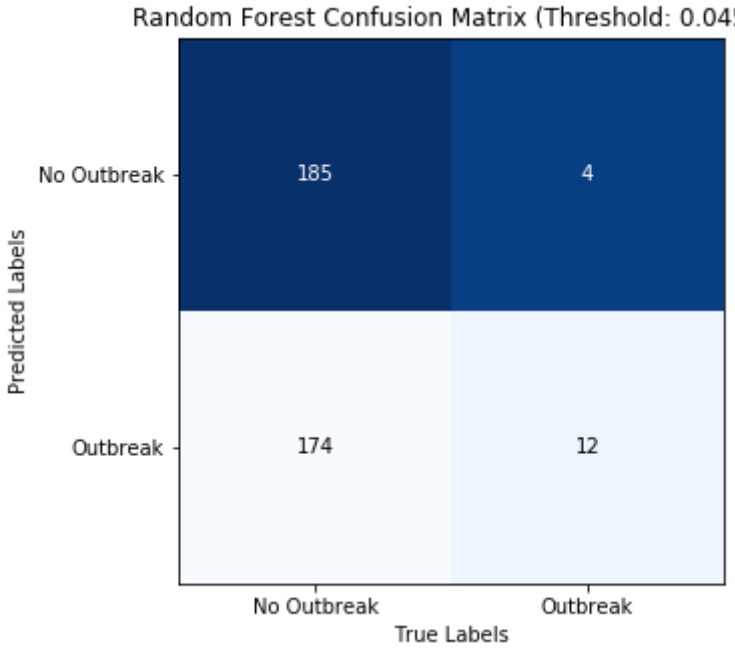

ROC Curve Comparison for Different Classifiers
Decision Tree AUC: 0.6286559888579387
Random Forest AUC: 0.6579038997214485
K Nearest Neighbors AUC: 0.5302924791086351
Gaussian Naive Bayes AUC: 0.5555362116991643
Multinomial Naive Bayes AUC: 0.5631963788300836
Bernoulli Naive Bayes AUC: 0.5530118384401115
Neural Network AUC: 0.5379526462395543
Support Vector Machine AUC: 0.4505571030640668
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

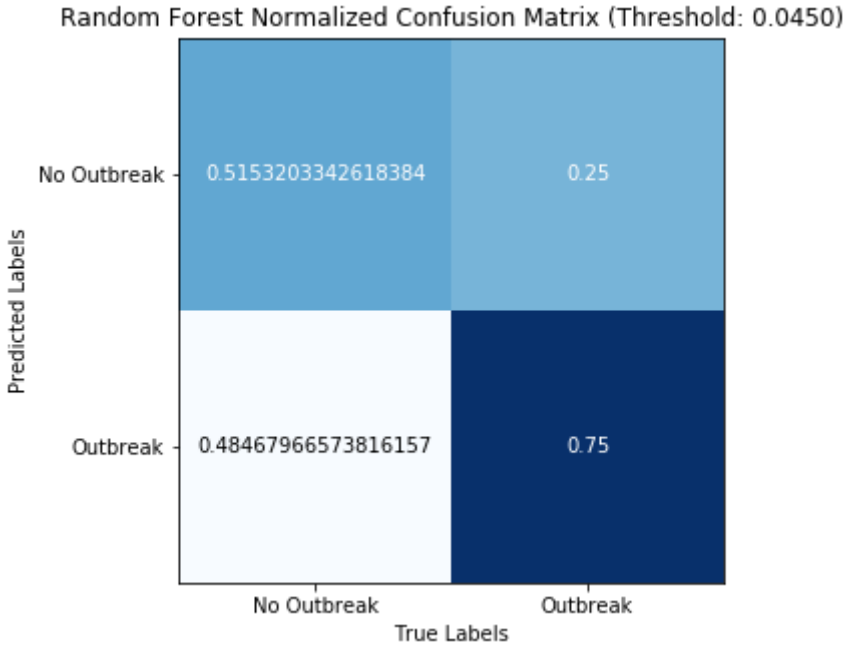
In [11]:
```
thresholdBasedClassification(thresholds[0], classifiers[0],labels[0], testi
```



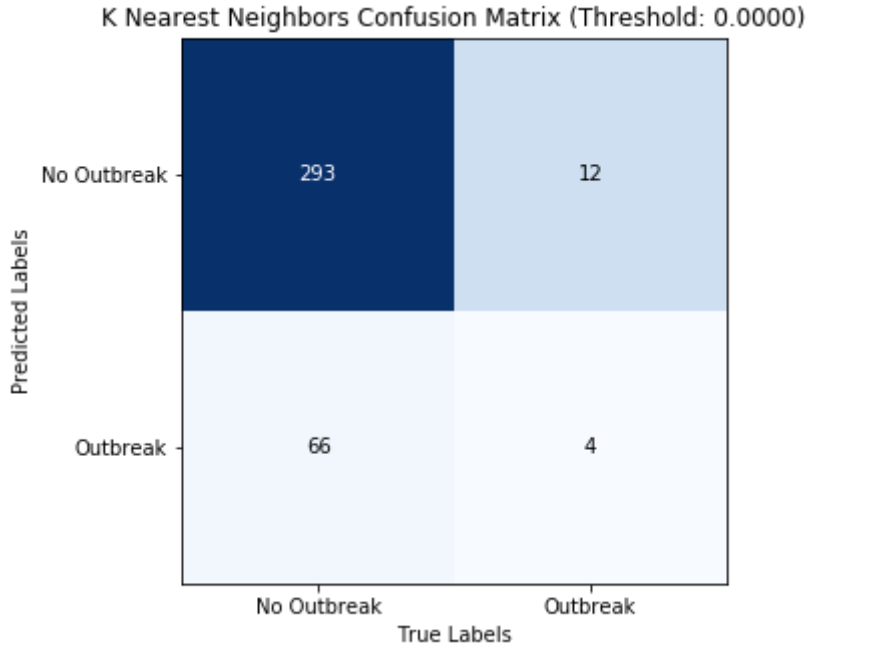


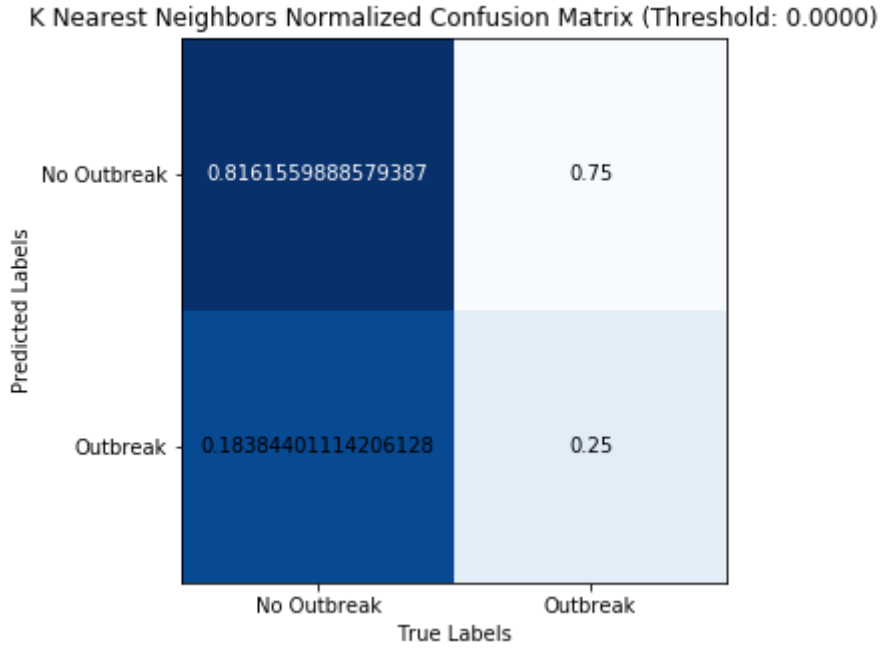
In [12]:
```
thresholdBasedClassification(thresholds[1], classifiers[1],labels[1], testi
```



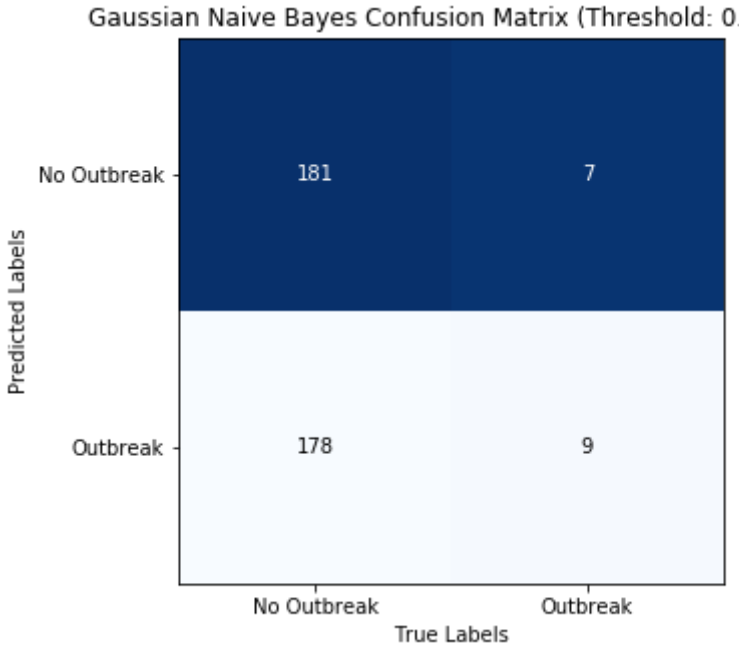


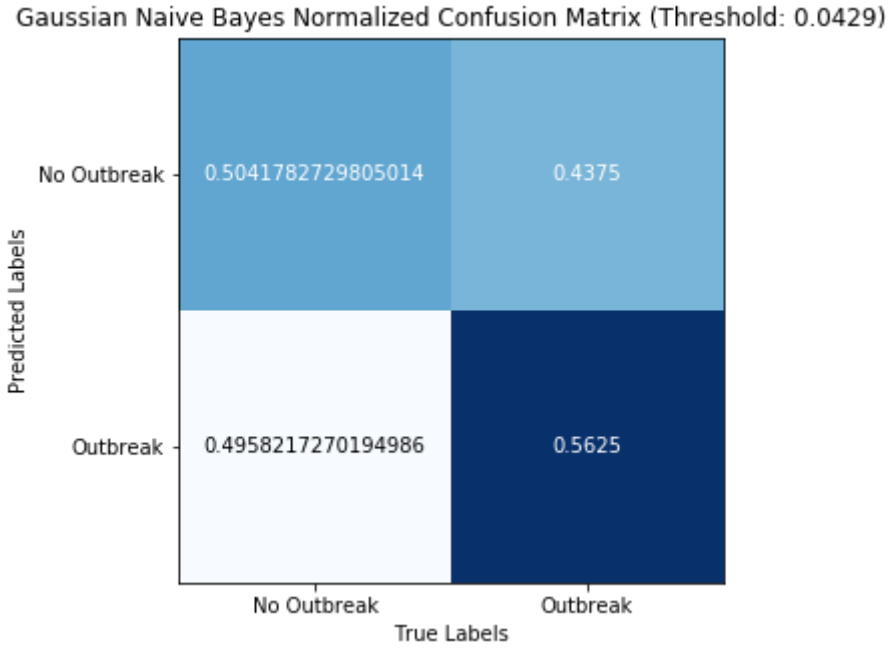
In [13]:
```
thresholdBasedClassification(thresholds[2], classifiers[2],labels[2], testi
```



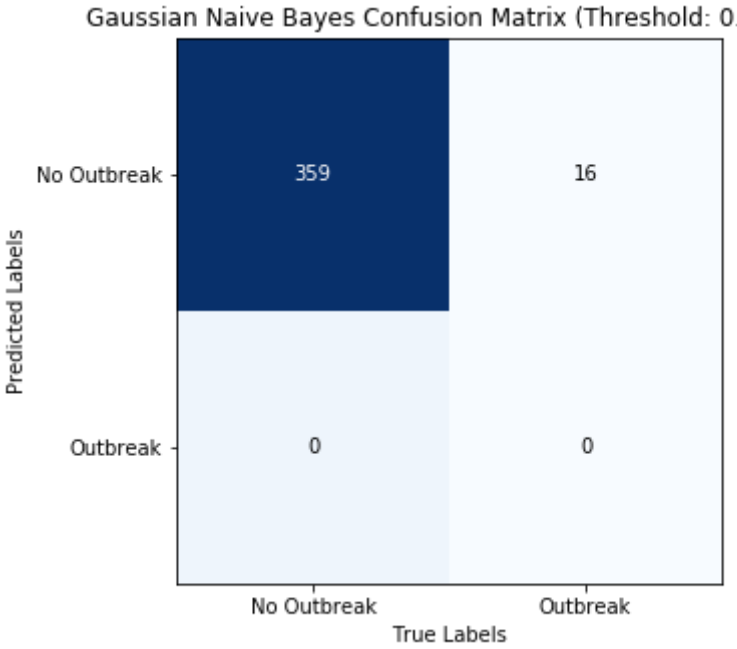


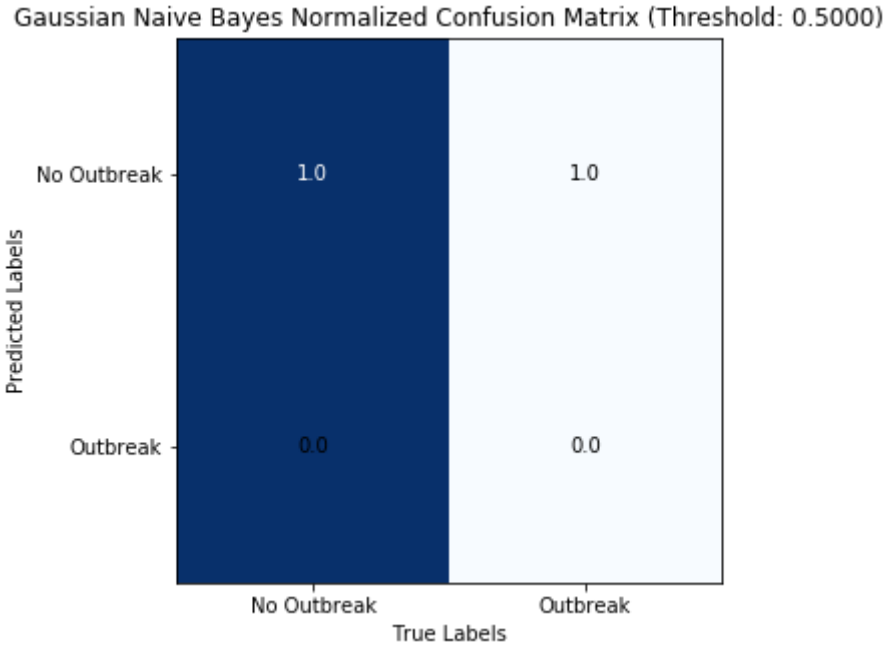
In [14]:
```
thresholdBasedClassification(thresholds[3], classifiers[3],labels[3], testi
```



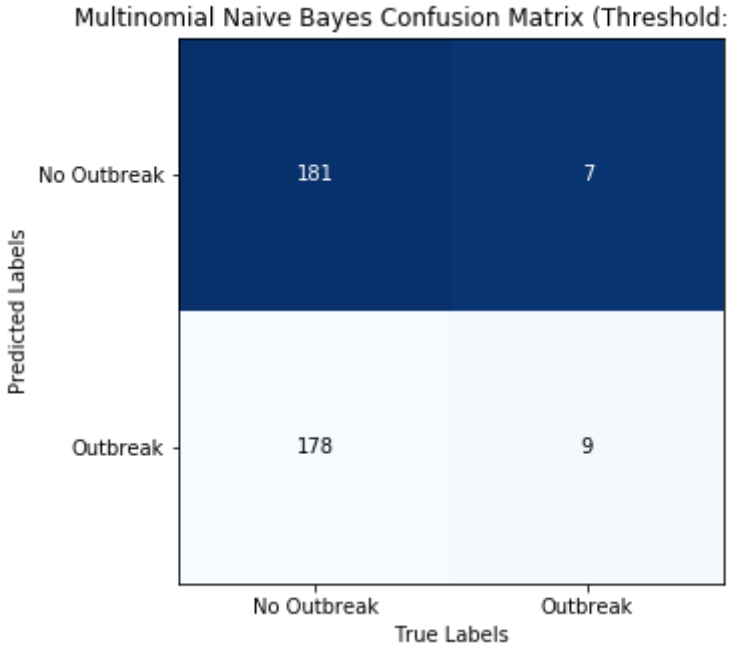


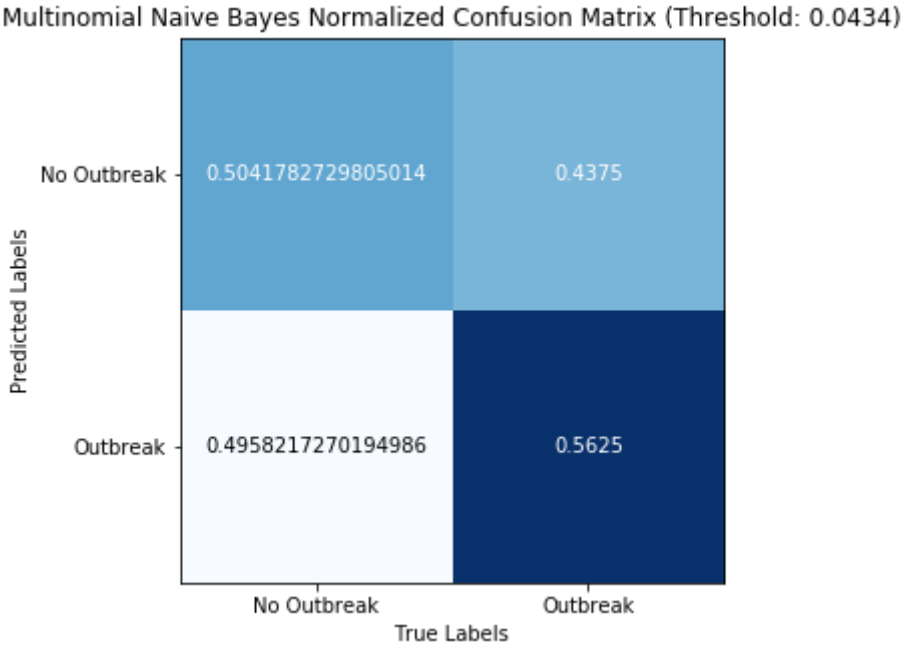
In [22]:
```
thresholdBasedClassification(0.5, classifiers[3],labels[3], testingFeatures
```



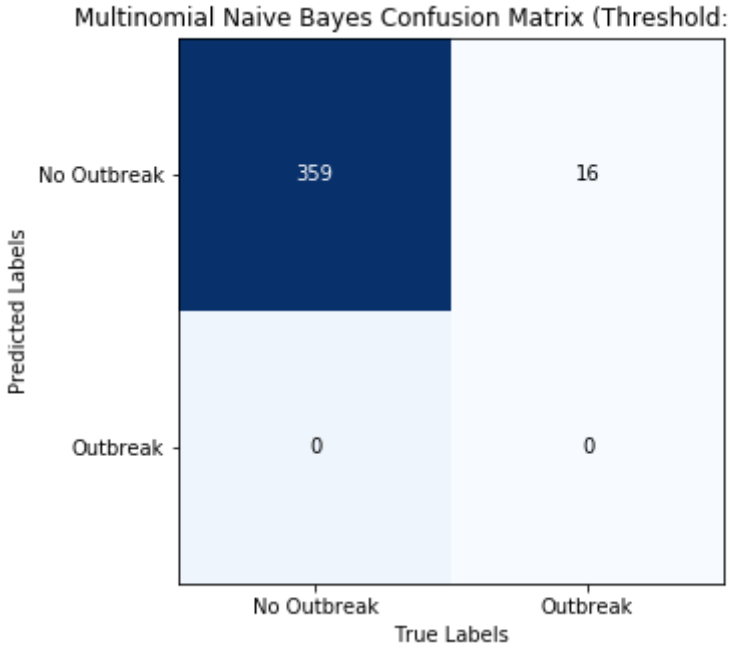


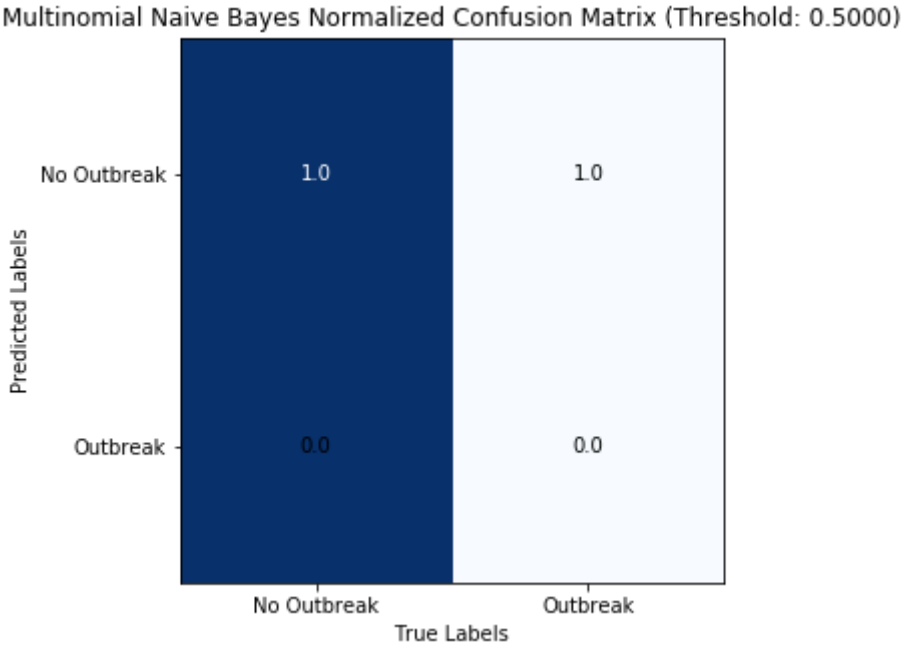
In [15]:
```
thresholdBasedClassification(thresholds[4], classifiers[4],labels[4], testi
```



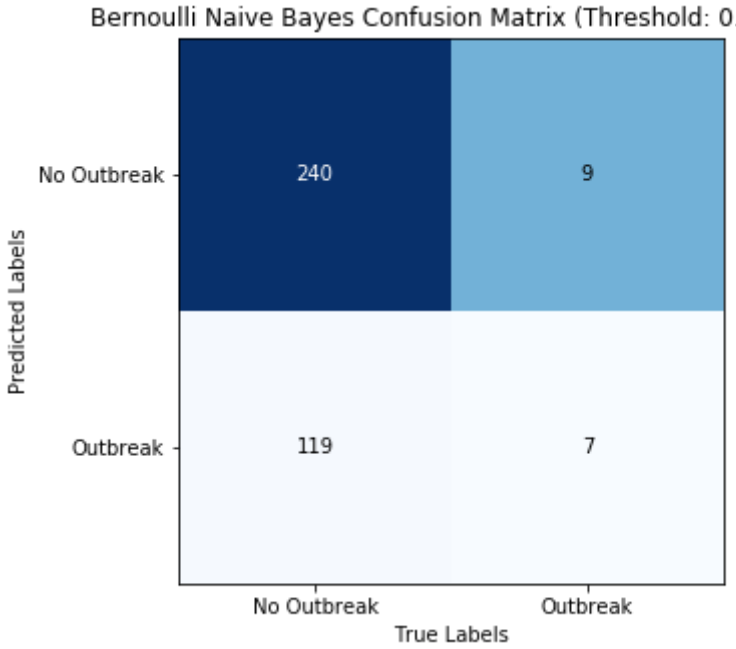


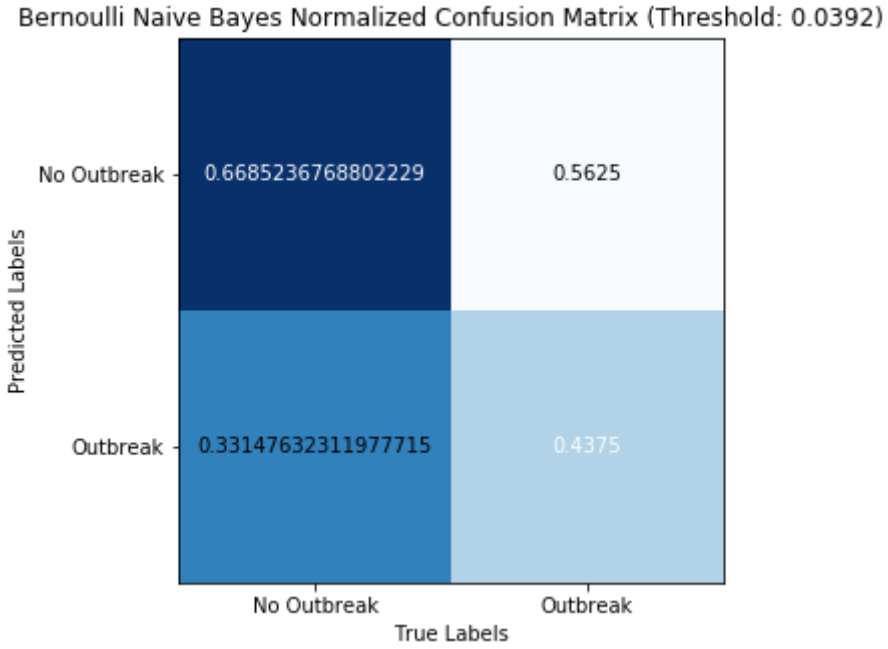
In [23]:
```
thresholdBasedClassification(0.5, classifiers[4],labels[4], testingFeatures
```



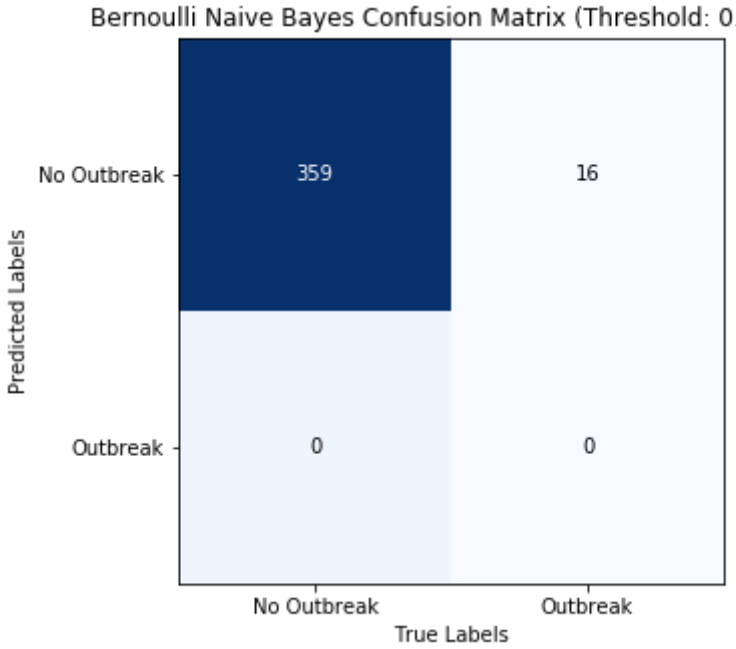


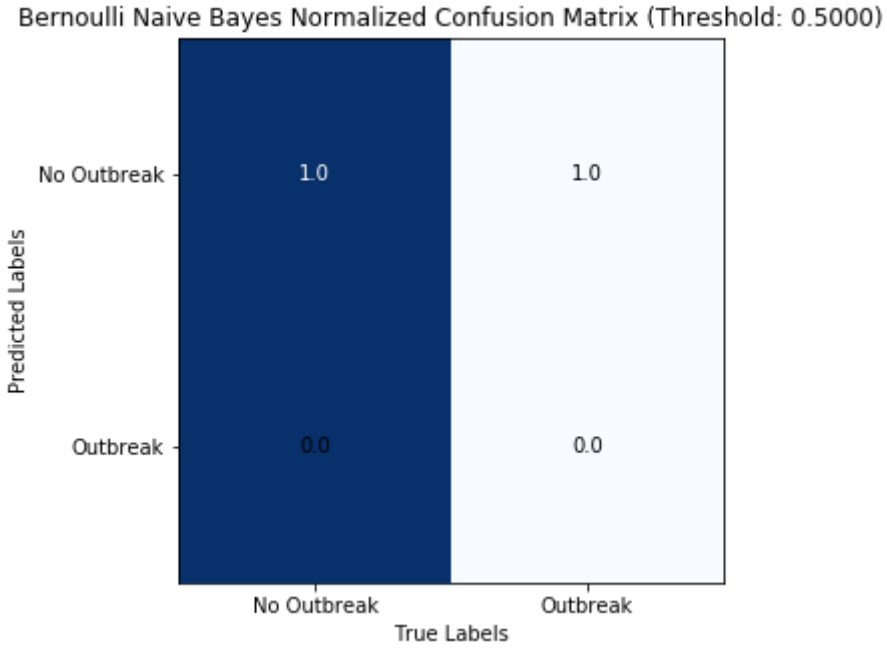
In [16]:
```
thresholdBasedClassification(thresholds[5], classifiers[5],labels[5], testi
```



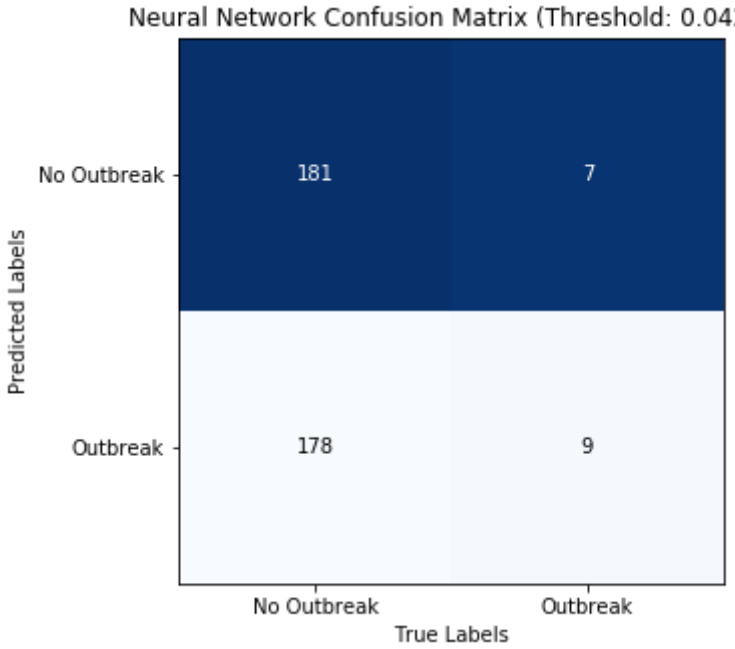


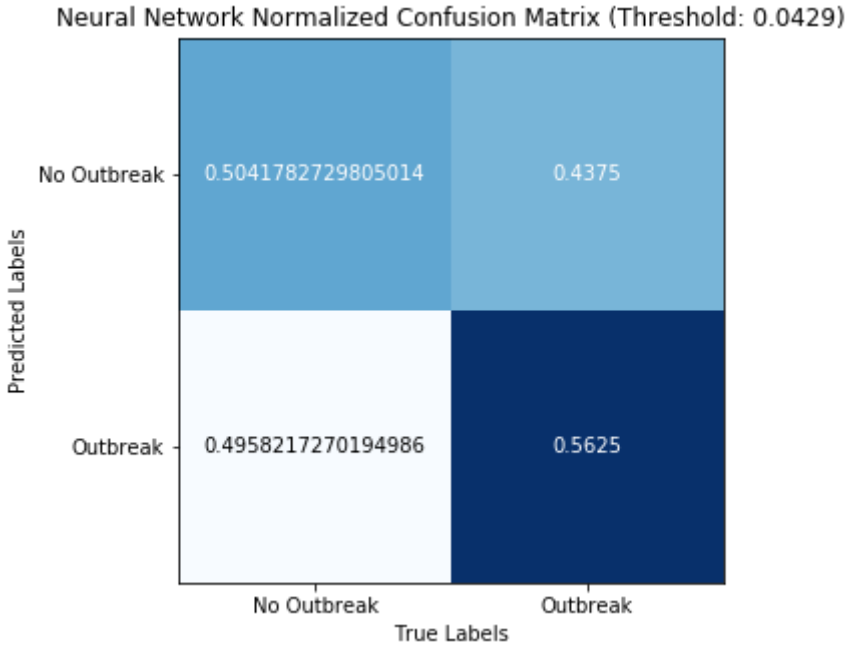
In [19]:
```
thresholdBasedClassification(0.5, classifiers[5],labels[5], testingFeatures
```



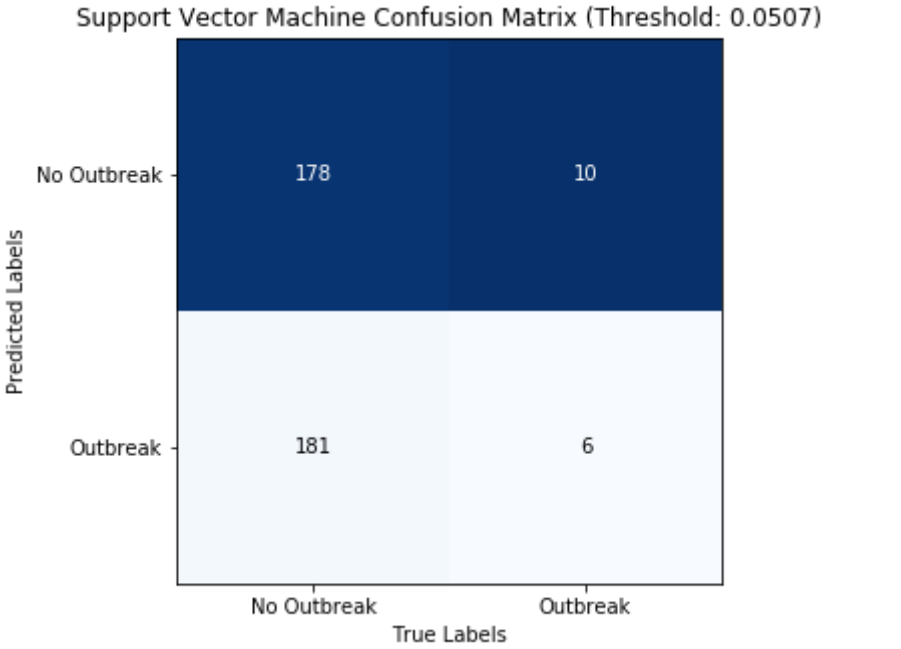


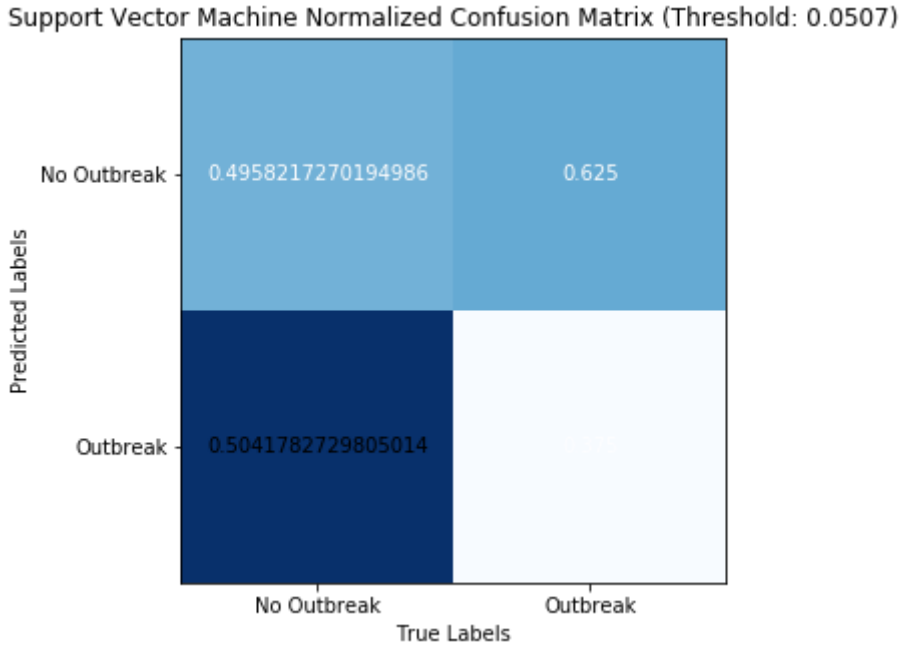
In [17]:
```
thresholdBasedClassification(thresholds[6], classifiers[6],labels[6], testi
```





In [18]:
```
thresholdBasedClassification(thresholds[7], classifiers[7],labels[7], testi
```





In [ ]:
```

```