Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

For the classification we used the following models:

XGBOOST

1. Code:

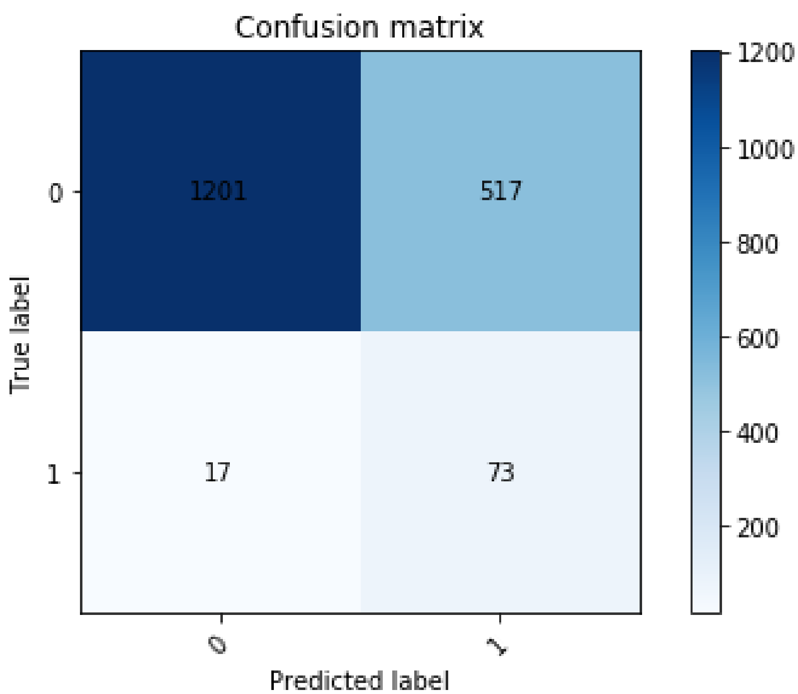
```
#XgBoost classifier
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
from imblearn.combine import SMOTETomek
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score

#plotting the confusion matrix
def plot_confusion_matrix(cm, target_names, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()

    width, height = cnf_matrix.shape

    for x in range(width):
        for y in range(height):
            plt.annotate(str(cm[x][y]), xy=(y, x),
                        horizontalalignment='center',
                        verticalalignment='center')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

#setting seed for reproducing
seed =123
#loading the datasets
target = pd.DataFrame(np.load('target_1980_2010.npy').astype(float))
data = pd.DataFrame(np.load('data_selected_1980_2010.npy').astype(float))
target = target.iloc[:, 1:]
#splitting the datasets in train and test sets
```

```
data_train, data_test, target_train, target_test = train_test_split(data, target, test_size=1803,
random_state=seed)
#generate minority class and undersample majority to balance the classes
sm = SMOTETomek()
data_train_s, target_train_s = sm.fit_sample(data_train, target_train)
data_train_s = pd.DataFrame(data_train_s)
target_train_s = pd.DataFrame(target_train_s)
#creating xgboost matrices
data_test = pd.DataFrame(data_test)
dtrain = xgb.DMatrix(data_train_s, label=target_train_s)
dtest = xgb.DMatrix(data_test)
type (dtrain)
train_labels = dtrain.get_label()
#parameters for xgboost
params = {
        'objective':'binary:logistic',
        'max_depth':5,
        'silent': 1,
        'n_estimators' : 1000,
        'learning_rate': 0.1,
        'min_child_weight': 1,
        'gamma' : 0,
        'subsample' : 0.8,
        'colsample_bytree':0.8,
        #'eta':1,
        'nthread':4,
       # 'max_delta_step' :1,
      'eval_metric':'auc'
      }
num_rounds = 20
#training and predicting the model
bst = xgb.train(params, dtrain, num_rounds)
y_test_preds = (bst.predict(dtest) > 0.49).astype(int)
cnf_matrix = confusion_matrix(target_test, y_test_preds)
plot_confusion_matrix(cnf_matrix, ['0', '1'])

# Compute ROC curve and area the curve
probas = bst.predict(dtest)
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(2):
    fpr[i], tpr[i], _ = roc_curve(target_test, probas)
    roc_auc[i] = auc(fpr[i], tpr[i])
```

Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

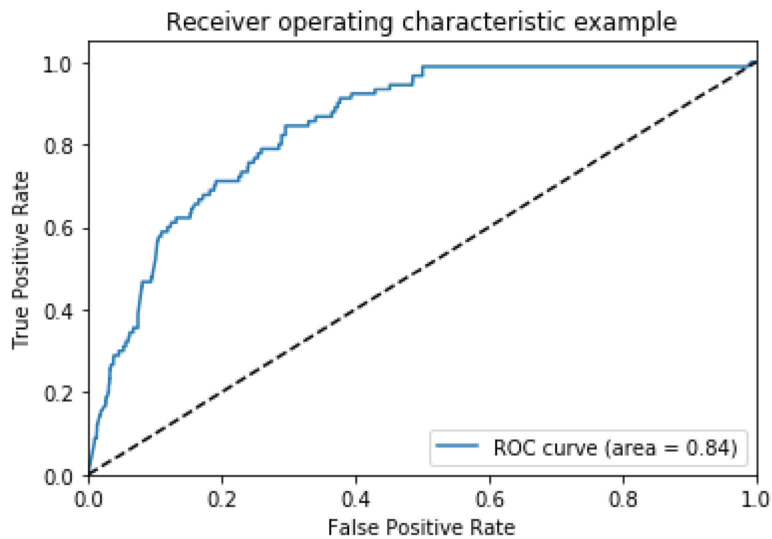
```
# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(target_test, probas)
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
plt.figure()
plt.plot(fpr[1], tpr[1], label='ROC curve (area = %0.2f)' % roc_auc[1])
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()

#printing the accuracy, precision and recall scores for the model
print('Accuracy:{0:.2f}'.format(accuracy_score(target_test, y_test_preds)))
print('Precision:{0:.2f}'.format(precision_score(target_test, y_test_preds)))
print('Recall:{0:.2f}'.format(recall_score(target_test, y_test_preds)))
```

2. Confusion Matrix



Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

3. ROC Curve for test data.



4. Accuracy, Precision, Recall
Accuracy: 0.70
Precision: 0.12
Recall: 0.81

Support Vector Machine (SVM)

1. Code

```
#SVM SVC classifier
from sklearn import preprocessing
from scipy import interp
import pylab as pl
from sklearn.metrics import accuracy_score, precision_score, recall_score
import numpy as np
from sklearn import svm
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

#plotting the confusion matrix
def plot_confusion_matrix(cm, target_names, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
```

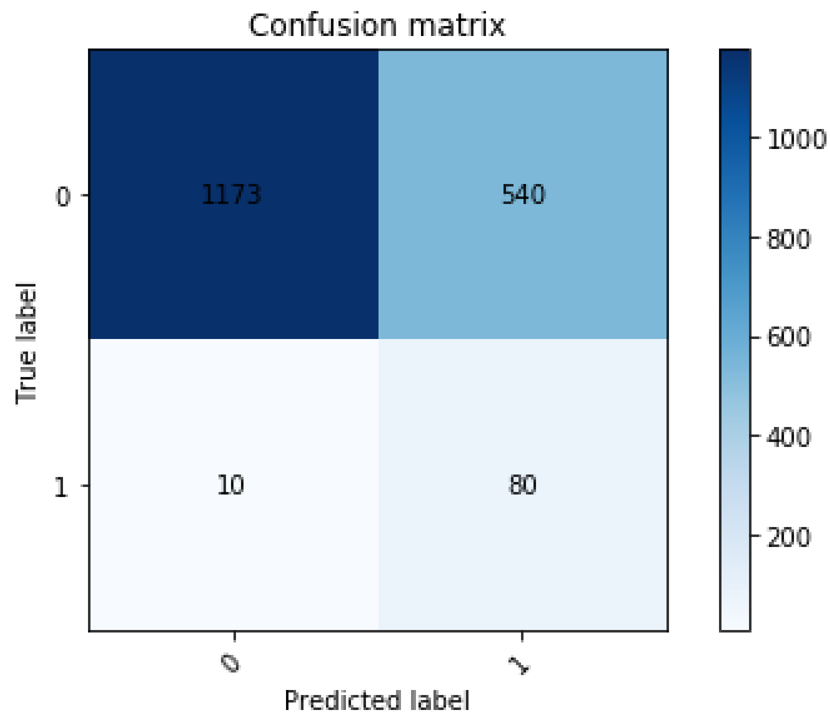
```
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()

    width, height = cnf_matrix.shape

    for x in range(width):
        for y in range(height):
            plt.annotate(str(cm[x][y]), xy=(y, x),
                         horizontalalignment='center',
                         verticalalignment='center')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

##loading the datasets
target = (np.load('target_1980_2010.npy').astype(float))
data = (np.load('data_selected_1980_2010.npy').astype(float))
target = target[:, 1:]
# setting seed to reproduce results
seed =123
# Normalizing the predictors
N_data = preprocessing.normalize(data)
#splitting the train and test data & training the model for prediction
data_train, data_test, target_train, target_test = train_test_split(N_data, target,
test_size=1803, random_state=seed)

# using SVM to train and predict the model
clf = svm.SVC(kernel='linear', C=26, class_weight={1: 14.4}, probability=True,
random_state=seed)
clf.fit(data_train, target_train)
predictions = clf.predict(data_test)

#calcualte confusion matrix for the prediction
cnf_matrix = confusion_matrix(target_test, predictions)
plot_confusion_matrix(cnf_matrix, ['0', '1'])

print('Accuracy:{0:.2f}'.format(accuracy_score(target_test, predictions)))
print('Precision:{0:.2f}'.format(precision_score(target_test, predictions)))
print('Recall:{0:.2f}'.format(recall_score(target_test, predictions)))

probabilities = clf.predict_proba(data_test)[:,1]
# Compute ROC curve and area the curve
fpr, tpr, _ = roc_curve(target_test, probabilities)
```

```
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for the test data')
plt.legend(loc="lower right")
plt.show()

#10 fold cross validation and ROC curves
target = np.reshape(target,[11300,])
cv = StratifiedKFold(n_splits = 10, random_state=seed)

# generating roc curves for different folds
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
i = 0
for train, test in cv.split(N_data, target):
   probas_ = clf.fit(N_data[train], target[train]).predict_proba(N_data[test])
   # Compute ROC curve and area the curve for different folds
   fpr, tpr, thresholds = roc_curve(target[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, label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))
   i += 1

pl.plot([0, 1], [0, 1], '--', color=(0.6, 0.6, 0.6), label='Luck')

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
pl.plot(mean_fpr, mean_tpr, 'k--', label='Mean ROC (area = %0.2f)' % mean_auc, lw=2)

pl.xlim([-0.05, 1.05])
pl.ylim([-0.05, 1.05])
pl.xlabel('False Positive Rate')
pl.ylabel('True Positive Rate')
pl.title('ROC curve for 10 fold cross validation')
```

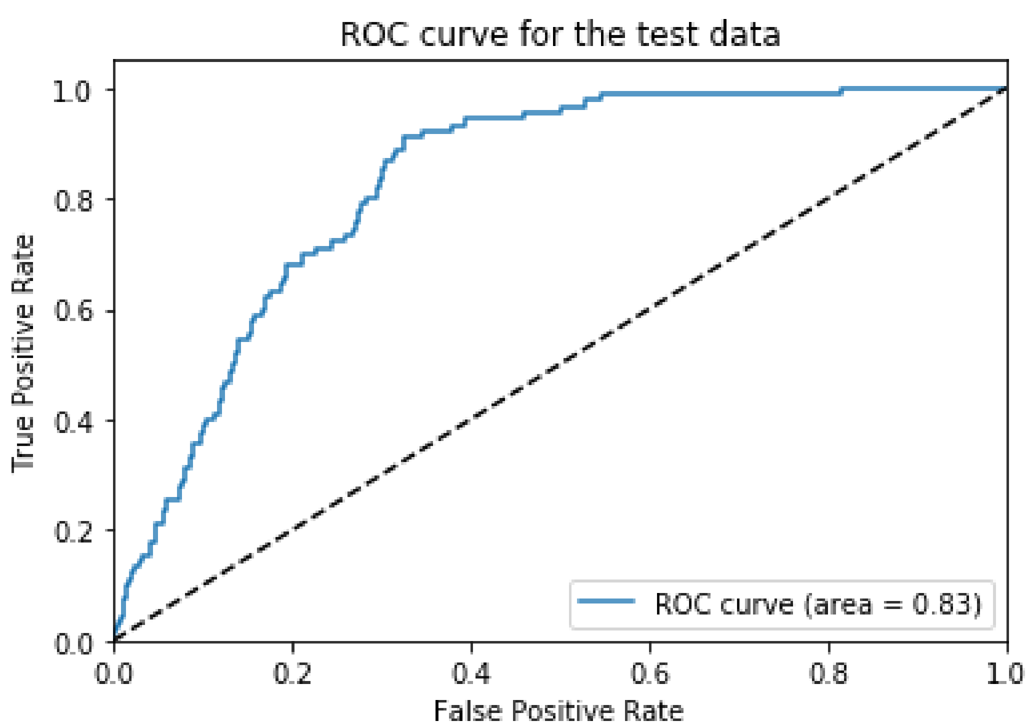
Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

```
pl.legend(loc="lower right")
pl.show()
```

2. Confusion Matrix

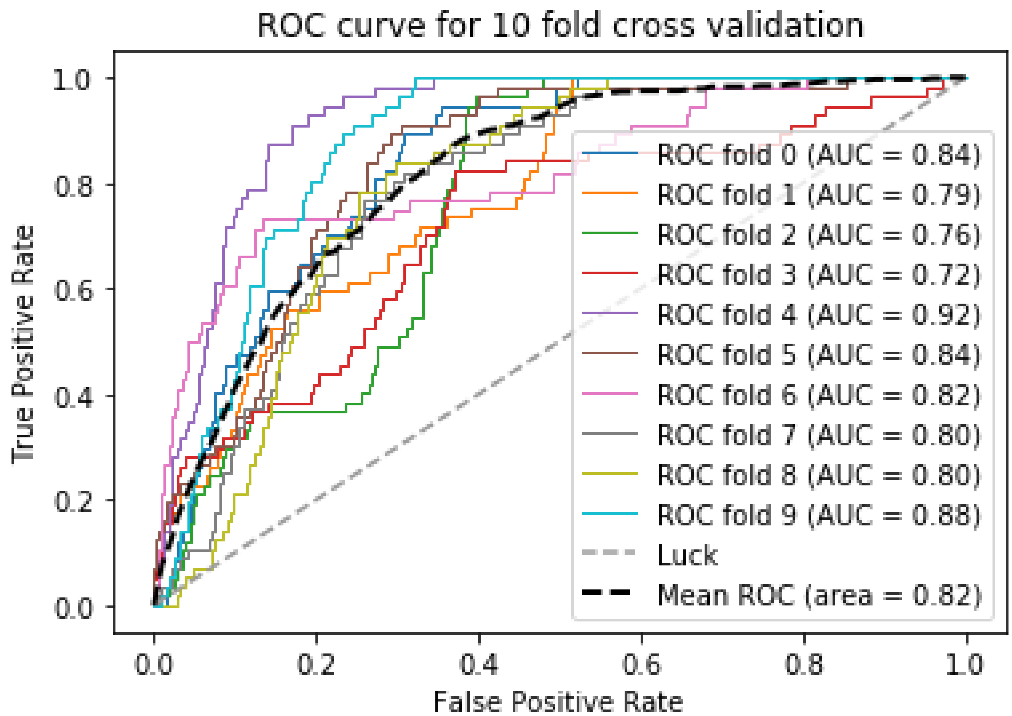


3. ROC Curve for test data.



Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

4. ROC Curve for 10 fold cross validation on whole data.



5. Accuracy, Precision, Recall
Accuracy:0.69
Precision:0.13
Recall:0.89

Naïve Bayes

1. Code

```
#Naive Bayes classifier
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import pylab as pl
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_curve, auc
from scipy import interp
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score
from imblearn.over_sampling import SMOTE
```

Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

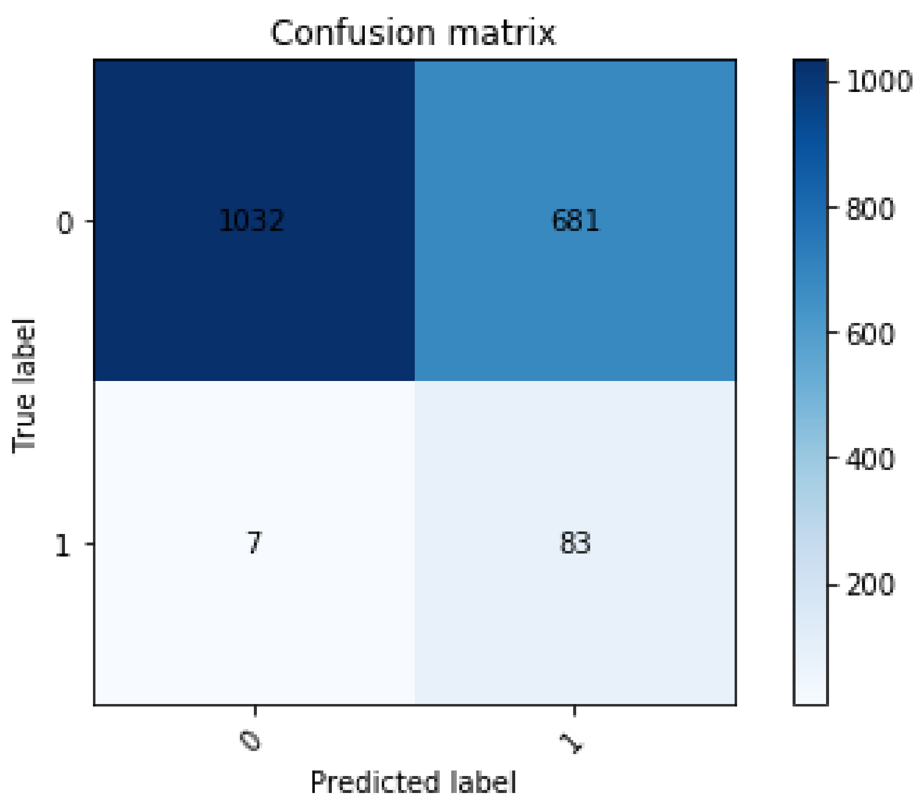
```
#plotting the confusion matrix
def plot_confusion_matrix(cm, target_names, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()

    width, height = cnf_matrix.shape

    for x in range(width):
        for y in range(height):
            plt.annotate(str(cm[x][y]), xy=(y, x),
                        horizontalalignment='center',
                        verticalalignment='center')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

##loading the datasets
target = (np.load('target_1980_2010.npy').astype(float))
data = (np.load('data_selected_1980_2010.npy').astype(float))
target = target[:, 1:]
# setting seed to reproduce results
seed =123
# Normalizing the predictors
N_data = preprocessing.normalize(data)
#splitting the train and test data & training the model for prediction
data_train, data_test, target_train, target_test = train_test_split(N_data, target,
test_size=1803, random_state=seed)

#Naive Bayes classifier
clf = GaussianNB()

#Dealing with imbalance data

#SMOTE -Synthetic Minority Oversampling Technique
data_train_smote, target_train_smote = SMOTE(random_state = seed).fit_sample(data_train,
target_train)
clf.fit(data_train_smote,target_train_smote)

predictions = clf.predict(data_test)
```

```
cnf_matrix = confusion_matrix(target_test, predictions)
plot_confusion_matrix(cnf_matrix, ['0', '1'])

print('Accuracy:{0:.2f}'.format(accuracy_score(target_test, predictions)))
print('Precision:{0:.2f}'.format(precision_score(target_test, predictions)))
print('Recall:{0:.2f}'.format(recall_score(target_test, predictions)))

probabilities = clf.predict_proba(data_test)[:,1]
# Compute ROC curve and area the curve
fpr, tpr, _ = roc_curve(target_test, probabilities)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for the test data')
plt.legend(loc="lower right")
plt.show()

#10 fold cross validation and ROC curves
target = np.reshape(target,[11300,])
cv = StratifiedKFold(n_splits = 10, random_state = seed)

# generating roc curves for different folds
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
i = 0
for train, test in cv.split(N_data, target):
   data_train_cv_smote, target_train_cv_smote = SMOTE(random_state =
seed).fit_sample(N_data[train], target[train])
   probas_ = clf.fit(data_train_cv_smote, target_train_cv_smote).predict_proba(N_data[test])
   # Compute ROC curve and area the curve for different folds
   fpr, tpr, thresholds = roc_curve(target[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, label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))
   i += 1
```

Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

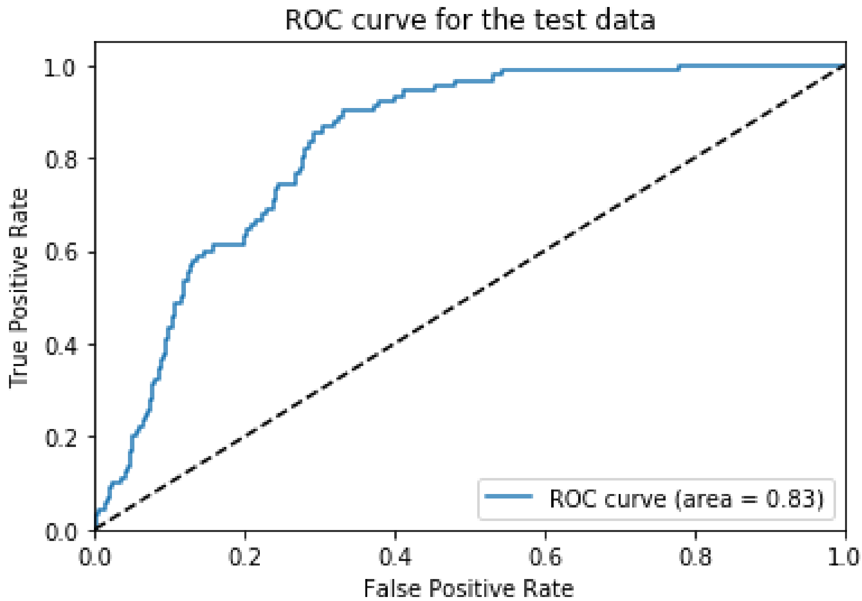
```
pl.plot([0, 1], [0, 1], '--', color=(0.6, 0.6, 0.6), label='Luck')

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
pl.plot(mean_fpr, mean_tpr, 'k--', label='Mean ROC (area = %0.2f)' % mean_auc, lw=2)

pl.xlim([-0.05, 1.05])
pl.ylim([-0.05, 1.05])
pl.xlabel('False Positive Rate')
pl.ylabel('True Positive Rate')
pl.title('ROC curve for 10 fold cross validation')
pl.legend(loc="lower right")
pl.show()
```

2. Confusion Matrix

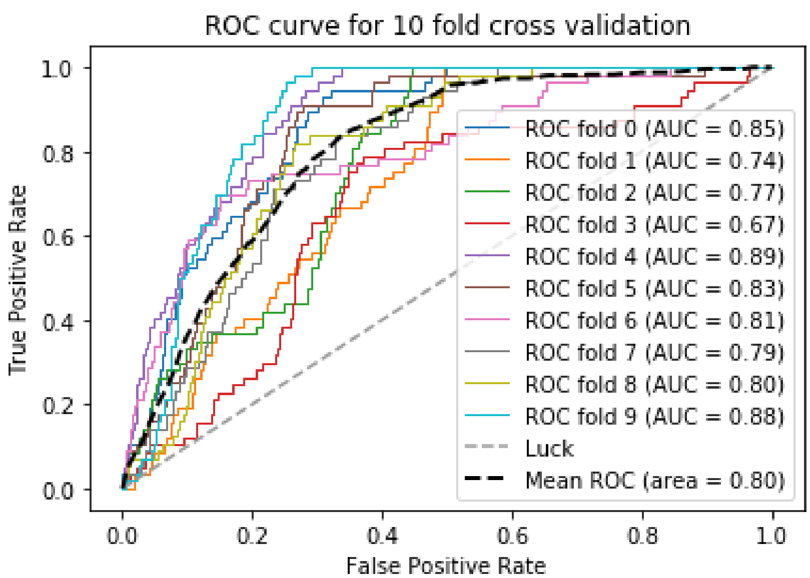


Team 7 Project Phase 3: Long-lead Forecasting of Extreme Precipitation

3. ROC Curve for test data



4. ROC Curve for 10 fold cross validation on whole data.



5. Accuracy, Precision, Recall

Accuracy:0.62

Precision:0.11

Recall:0.92

Team member names and emails:

Sonica Kalmangi - sonica.kalmangi001@umb.edu

Sidhraj Solanki - Sidhraj.solanki001@umb.edu

Jacob Robins - jacob.robins001@umb.edu

Swapnil Patil - swapnil.patil001@umb.edu