PD1

Jiarong Ye, Yuan Meng

November 10, 2018

import packages

```
In [205]: import datascience as ds
          from datascience import *
          import numpy as np
          from graphviz import Source
          import pandas as pd
          import seaborn as sns
          from sklearn.pipeline import Pipeline
          from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
          from sklearn.tree import DecisionTreeClassifier
          from sklearn.ensemble import RandomForestClassifier
          from sklearn import tree
          from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score,
                                      accuracy_score, classification_report
          import matplotlib.pyplot as plt
          from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
          from sklearn.externals import joblib
          %matplotlib inline
```

tweets data loaded into Jupyter Notebook as Table object

```
In [26]: df = ds.Table.read_table('Climate1SupportiveLevel.csv', sep=',')
         df

Out[26]: Unnamed: 0 | ID                       | Text
         0          | 962_Cleand_Climate1.csv  | RT @kasserolees: Energy is the #1 contributer t
         1          | 885_Cleand_Climate1.csv  | RT @edelman_barbara: @msnbc why don t you have
         2          | 680_Cleand_Climate1.csv  | RT @OtagoGrad: @anthonyfurey @OskieOckham The d
         3          | 1152_Cleand_Climate1.csv | The Dow just recorded its 3rd worst day ever. T
         4          | 731_Cleand_Climate1.csv  | RT @SimonBanksHB: I am not going to rule out th
         5          | 1075_Cleand_Climate1.csv | RT @sydneyleemarco: nothing like an 80 degree o
         6          | 85_Cleand_Climate1.csv   | @MerlenesMemos @CNN It's not an act of god. Cli
         7          | 654_Cleand_Climate1.csv  | RT @MikeLevinCA: When asked about climate chang
         8          | 916_Cleand_Climate1.csv  | RT @gq_jayq: Bet I got 11 years to run it up ht
         9          | 372_Cleand_Climate1.csv  | No they care about the oil billionaires
         ... (1273 rows omitted)
```

Preprocess

```
In [276]: X = list(df['Text'])
          y = list(df['SupportiveLabel'])
```

Check whether the data distribution is balanced

```
In [89]: def check(sentiment, index, note='training'):
             if sentiment==0:
                 label = 'not supportive'
             else:
                 label = 'supportive'
             print('There are {} '.format(df.take(index).where('SupportiveLabel',
                   are.equal_to(sentiment)).size[0][0])+label+' tweets in the '+note+' set.')
```

Model Building

```
In [90]: def custom_split(train_index, test_index):
             trainingset = df.take(train_index)
             testingset = df.take(test_index)

             X_train= list(trainingset['Text'])
             y_train= list(trainingset['SupportiveLabel'])
             X_test= list(testingset['Text'])
             y_test= list(testingset['SupportiveLabel'])

             return X_train, X_test, y_train, y_test
```

classifier

```
In [291]: def classifier(X_train, y_train, X_test, fold, max_depth, min_samples_leaf):
              # token_pattern='(([#@]|[0-9]|[a-z]|[A-Z])+)'
              clf = Pipeline(
                  [
                      ('vect', CountVectorizer(token_pattern="(?!RT|rt|\d+)[@#]*[\w\'_-]{2,100}"
                                               analyzer = 'word',
                                               stop_words='english',
                                               min_df = 3)),
                      ('clf', DecisionTreeClassifier(criterion='entropy',
                                                     random_state = 100,
                                                     max_depth = max_depth,
                                                     min_samples_leaf = min_samples_leaf))
                  ])
              clf.fit(X_train, y_train)
              feature_names = clf.named_steps['vect'].get_feature_names()
              try:
                  dot_data = tree.export_graphviz(clf.named_steps['clf'], out_file=None,
                                                  feature_names=feature_names)
                  graph = Source(dot_data)
```

```
                graph.render('ClimateClassifier-Fold_{}'.format(fold))
            except Exception as e:
                print(e)
            predicted_y_train = clf.predict(X_train)
            predicted_y_test = clf.predict(X_test)
            # save as pickle
            joblib.dump(clf, 'ClimateTeam7PD1.pkl')
            return predicted_y_train, predicted_y_test
```

```
In [283]: c=CountVectorizer(token_pattern="(?!RT|rt|\d+)[@#]*[\w\'_-]{2,100}",
                                                     analyzer = 'word',
                                                     stop_words='english',
                                                     min_df = 3)
          c.fit(X, y)
          c.get_feature_names()
```

```
Out[283]: ['#1o5c',
           '#actonclimate',
           '#auspol',
           '#cdnpoli',
           '#climate',
           '#climateaction',
           '#climatebreakdown',
           '#climatechange',
           '#climatechangeisreal',
           '#climateimpactsvic',
           '#dems',
           '#emissions',
           '#energy',
           '#environment',
           '#florida',
                        ....
```

evaluation

```
In [295]: def eval_results(predicted_y_train, y_train, predicted_y_test, y_test):
              accuracy_s = accuracy_score(y_test, predicted_y_test)
              precision_s = precision_score(y_test, predicted_y_test)
              recall_s = recall_score(y_test, predicted_y_test)
              f1_s = f1_score(y_test, predicted_y_test)
              cm_train = confusion_matrix(y_train, predicted_y_train)
              cm_test = confusion_matrix(y_test, predicted_y_test)

              print('Accuracy Score:', accuracy_s)
              print("Precision Score:", precision_s)
              print("Recall Score:", recall_s)
              print("f1 Score:", f1_s)
              print('confusion_matrix of training set is: \n', cm_train, '\n')
```

```
            print('confusion_matrix of testing set is: \n', cm_test, '\n')
            print(classification_report(y_test, predicted_y_test))

            classes = ['not supportive', 'supportive']
            sns.heatmap(cm_train, annot=True, cmap='Blues', yticklabels=classes,
                                                           xticklabels=classes)
            plt.title('confusion matrix of training set')
            plt.show()
            sns.heatmap(cm_test, annot=True, cmap='Blues', yticklabels=classes,
                                                           xticklabels=classes)
            plt.title('confusion matrix of testing set')
            plt.show()
            return accuracy_s, precision_s, recall_s, f1_s
```

k-fold

```
In [293]: def k_fold_evaluate(X, y, max_depth, min_samples_leaf):
            # initialization
            accuracy = []
            precision = []
            recall=[]
            f1 = []
            fold = 1
            skf = StratifiedKFold(n_splits=5, random_state=1, shuffle= True)

            # build model and collect results
            for train_index, test_index in skf.split(X, y):
                if fold==1:
                    list(map(lambda x: check(x, train_index), range(2)))
                    list(map(lambda x: check(x, test_index, note='testing'), range(2)))

                X_train, X_test, y_train, y_test = custom_split(train_index, test_index)

                predicted_y_train, predicted_y_test =
                                                   classifier(X_train=X_train, y_train=y
                                                              X_test=X_test, fold=fold,
                                                              max_depth = max_depth,
                                                              min_samples_leaf = min_sampl

                print('\nFold: {}'.format(fold))
                accuracy_s, precision_s, recall_s, f1_s = eval_results(predicted_y_train,
                                                              y_train, predicted_y_test, y_

                accuracy.append(accuracy_s)
                precision.append(precision_s)
                recall.append(recall_s)
                f1.append(f1_s)
                metrics_df = pd.DataFrame(
                            {
                                'accuracy': accuracy,
                                'precision': precision,
                                'recall':recall,
                                'f1':f1
                            }
                        )
                fold += 1
            return metrics_df
```

Tasks

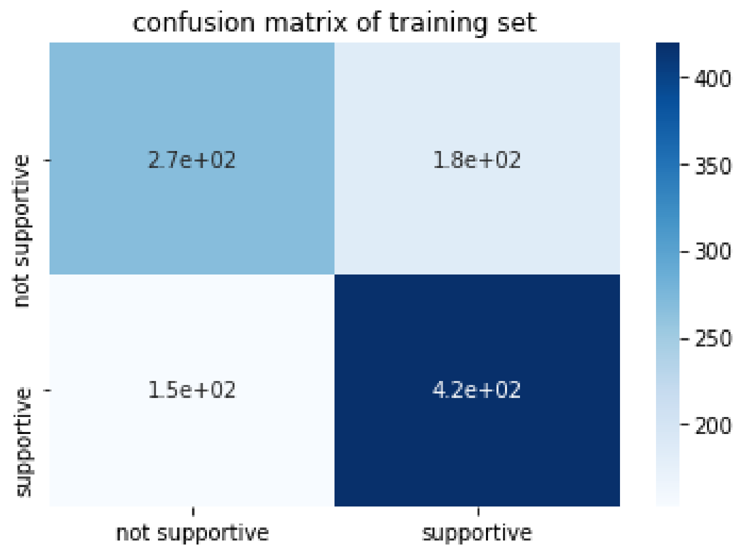
a) A description of model parameters you tried and the associated Stratified k-fold cross validation results for each model parameter choice

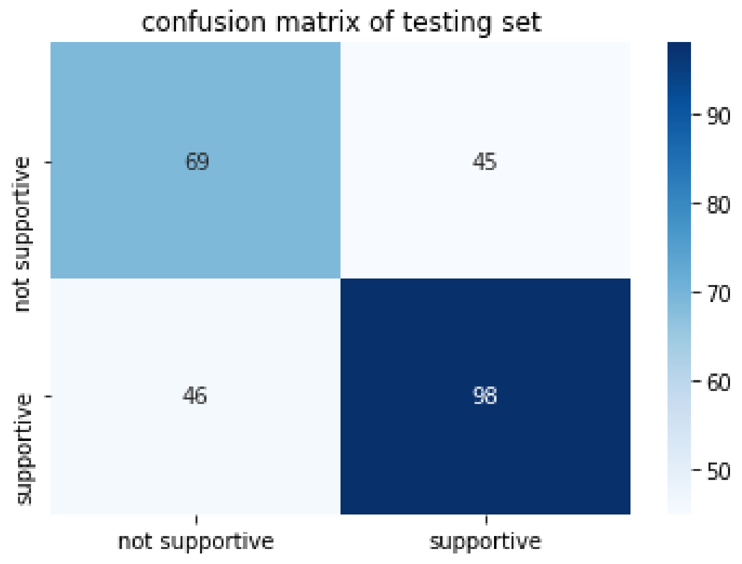
b) Describe the model parameters you chose and the rationale of your decision.

c) Double check overfitting risk: Compare the model's confusion matrix for training data vs the model's confusion matrix for testing data.

```
In [296]: k_fold_evaluate(X, y, max_depth=5, min_samples_leaf=2)
```

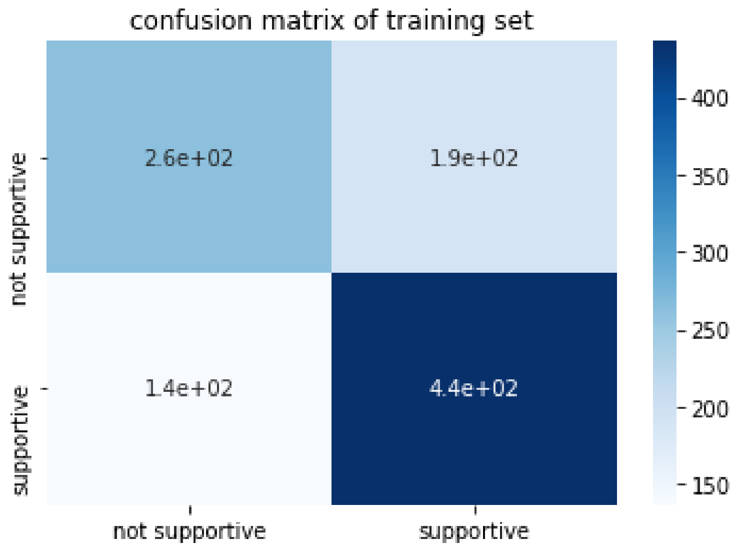
```
There are 453 not supportive tweets in the training set.
There are 572 supportive tweets in the training set.
There are 114 not supportive tweets in the testing set.
There are 144 supportive tweets in the testing set.

Fold: 1
Accuracy Score: 0.6472868217054264
Precision Score: 0.6853146853146853
Recall Score: 0.6805555555555556
f1 Score: 0.6829268292682927
confusion_matrix of training set is:
 [[268 185]
 [153 419]]

confusion_matrix of testing set is:
 [[69 45]
 [46 98]]

              precision    recall  f1-score   support

           0       0.60      0.61      0.60       114
           1       0.69      0.68      0.68       144

   micro avg       0.65      0.65      0.65       258
   macro avg       0.64      0.64      0.64       258
weighted avg       0.65      0.65      0.65       258
```

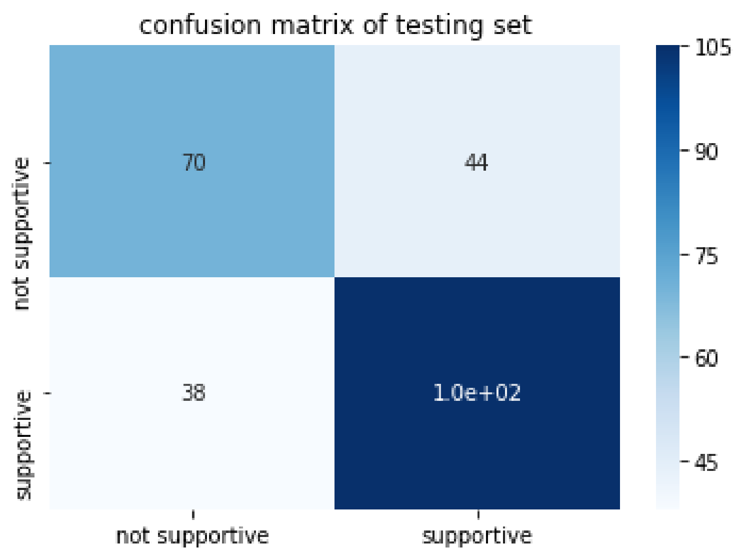




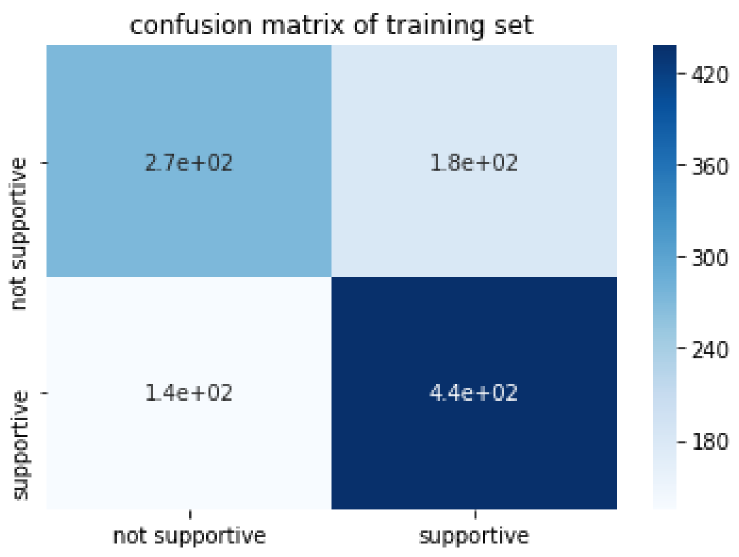
```
Fold: 2
Accuracy Score: 0.6809338521400778
Precision Score: 0.7046979865771812
```

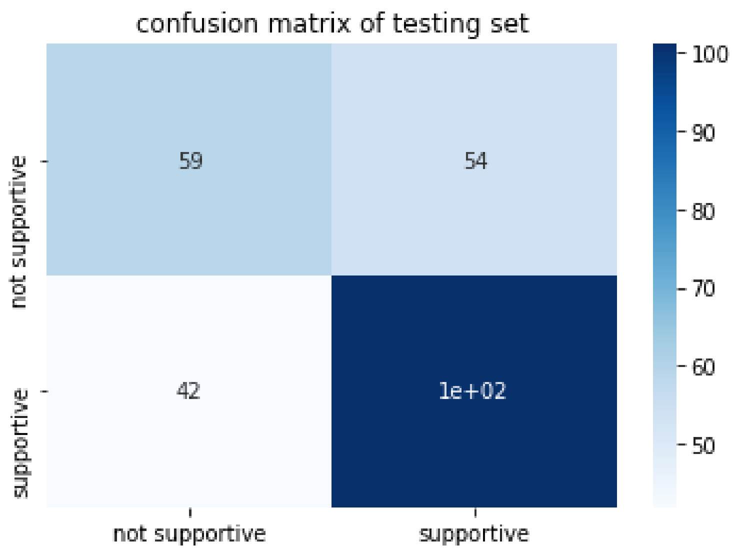
```
Recall Score: 0.7342657342657343
f1 Score: 0.7191780821917809
confusion_matrix of training set is:
 [[263 190]
 [137 436]]

confusion_matrix of testing set is:
 [[ 70  44]
 [ 38 105]]

              precision    recall  f1-score   support

           0       0.65      0.61      0.63       114
           1       0.70      0.73      0.72       143

   micro avg       0.68      0.68      0.68       257
   macro avg       0.68      0.67      0.67       257
weighted avg       0.68      0.68      0.68       257
```





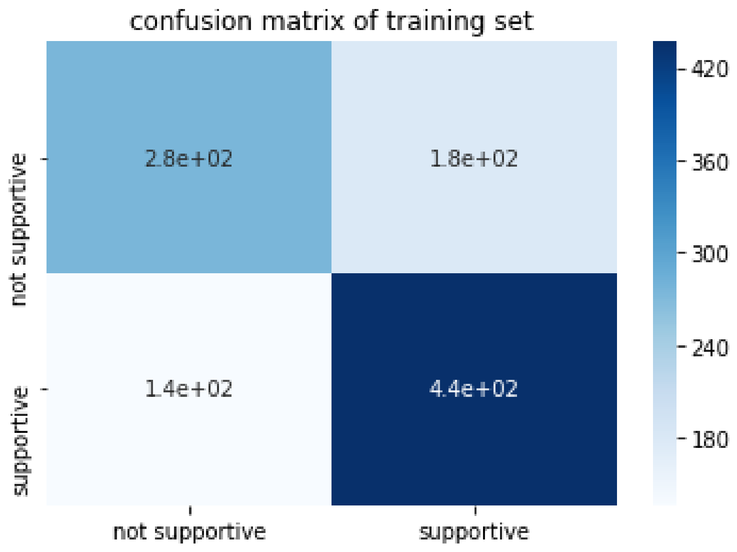
```
Fold: 3
Accuracy Score: 0.625
Precision Score: 0.6516129032258065
Recall Score: 0.7062937062937062
f1 Score: 0.6778523489932886
confusion_matrix of training set is:
 [[273 181]
 [135 438]]

confusion_matrix of testing set is:
 [[ 59  54]
 [ 42 101]]

              precision    recall  f1-score   support

           0       0.58      0.52      0.55       113
           1       0.65      0.71      0.68       143

   micro avg       0.62      0.62      0.62       256
   macro avg       0.62      0.61      0.61       256
weighted avg       0.62      0.62      0.62       256
```

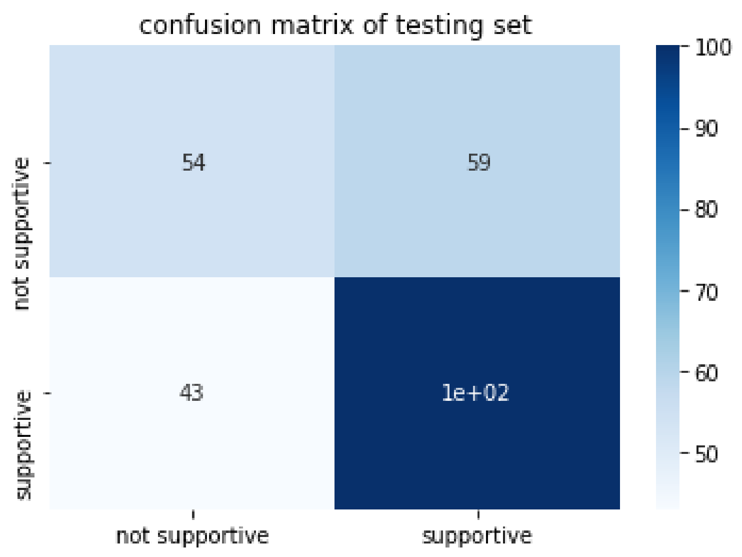




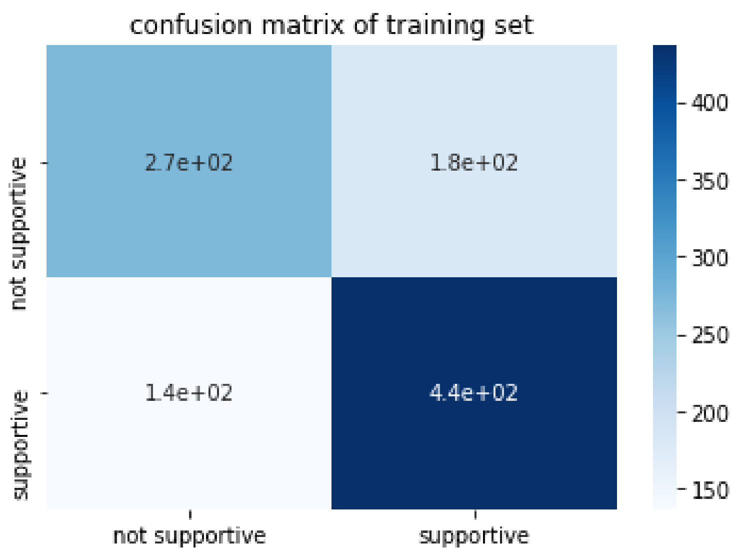
```
Fold: 4
Accuracy Score: 0.6015625
Precision Score: 0.6289308176100629
Recall Score: 0.6993006993006993
f1 Score: 0.6622516556291391
confusion_matrix of training set is:
 [[275 179]
 [136 437]]
```

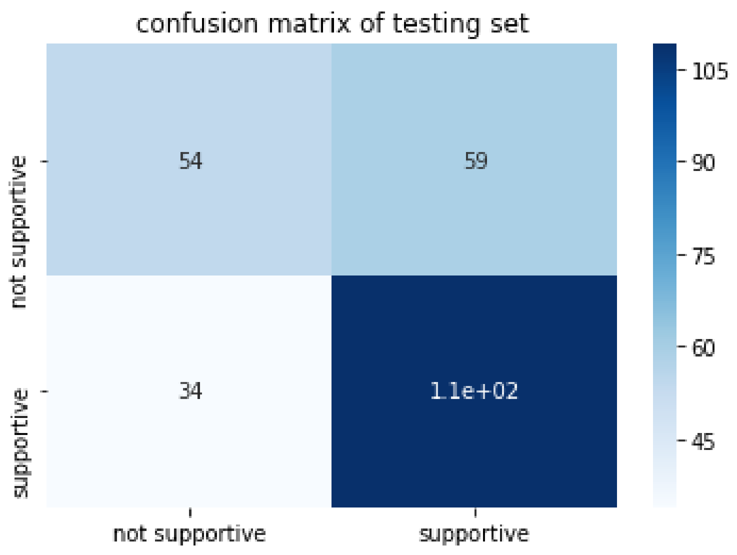
```
confusion_matrix of testing set is:
 [[ 54  59]
 [ 43 100]]

              precision    recall  f1-score   support

           0       0.56      0.48      0.51       113
           1       0.63      0.70      0.66       143

   micro avg       0.60      0.60      0.60       256
   macro avg       0.59      0.59      0.59       256
weighted avg       0.60      0.60      0.60       256
```





```
Fold: 5
Accuracy Score: 0.63671875
Precision Score: 0.6488095238095238
Recall Score: 0.7622377622377622
f1 Score: 0.7009646302250804
confusion_matrix of training set is:
 [[272 182]
 [137 436]]

confusion_matrix of testing set is:
 [[ 54  59]
 [ 34 109]]

              precision    recall  f1-score   support

           0       0.61      0.48      0.54       113
           1       0.65      0.76      0.70       143

   micro avg       0.64      0.64      0.64       256
   macro avg       0.63      0.62      0.62       256
weighted avg       0.63      0.64      0.63       256
```

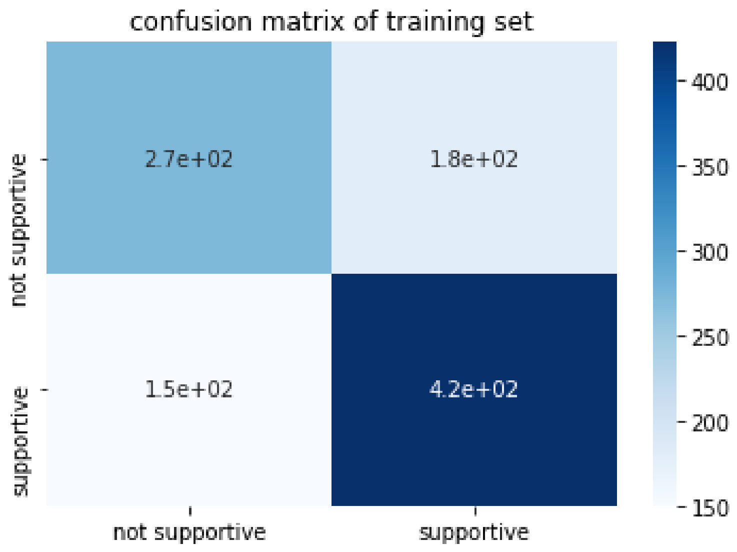


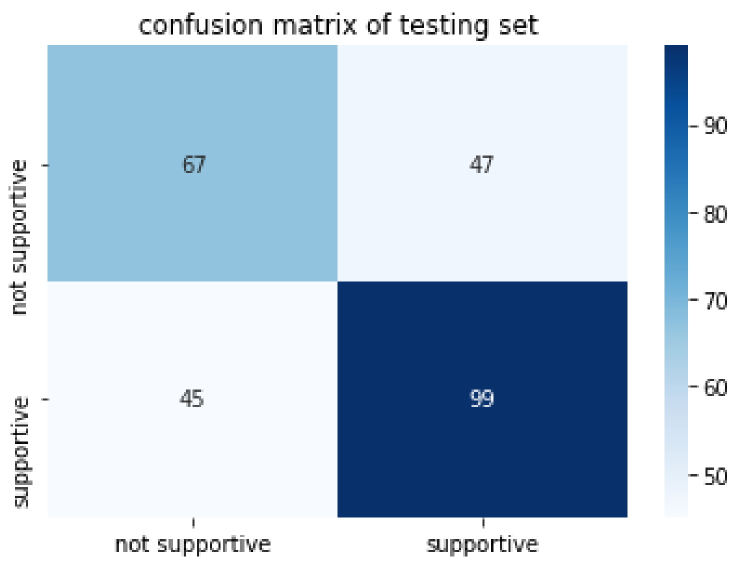


```
Out[296]:    accuracy  precision    recall        f1
          0  0.647287   0.685315  0.680556  0.682927
          1  0.680934   0.704698  0.734266  0.719178
          2  0.625000   0.651613  0.706294  0.677852
          3  0.601562   0.628931  0.699301  0.662252
          4  0.636719   0.648810  0.762238  0.700965
```

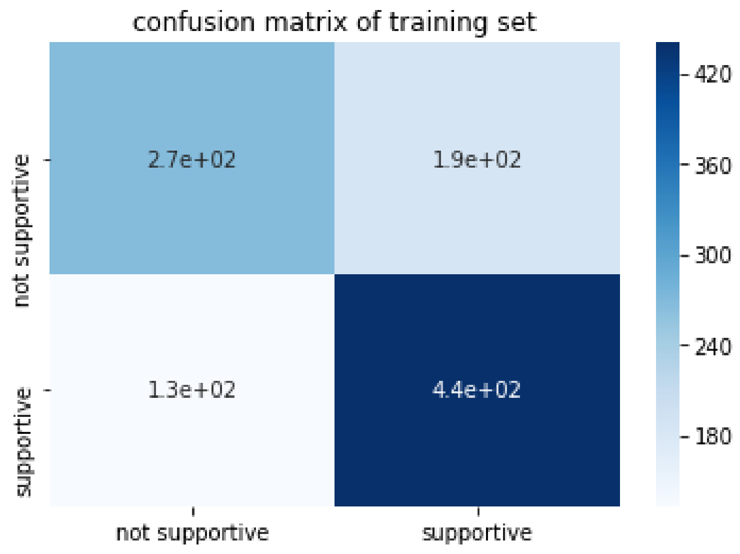
```
In [297]: k_fold_evaluate(X, y, max_depth=6, min_samples_leaf=2)
```

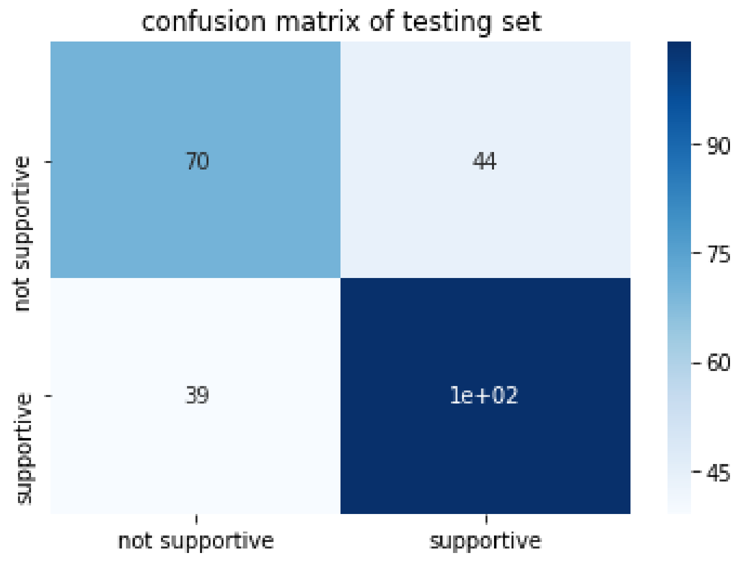
```
There are 453 not supportive tweets in the training set.
There are 572 supportive tweets in the training set.
There are 114 not supportive tweets in the testing set.
There are 144 supportive tweets in the testing set.

Fold: 1
Accuracy Score: 0.6434108527131783
Precision Score: 0.678082191780822
Recall Score: 0.6875
f1 Score: 0.6827586206896552
confusion_matrix of training set is:
 [[273 180]
 [150 422]]

confusion_matrix of testing set is:
 [[67 47]
 [45 99]]

              precision    recall  f1-score   support

           0       0.60      0.59      0.59       114
           1       0.68      0.69      0.68       144

   micro avg       0.64      0.64      0.64       258
   macro avg       0.64      0.64      0.64       258
weighted avg       0.64      0.64      0.64       258
```





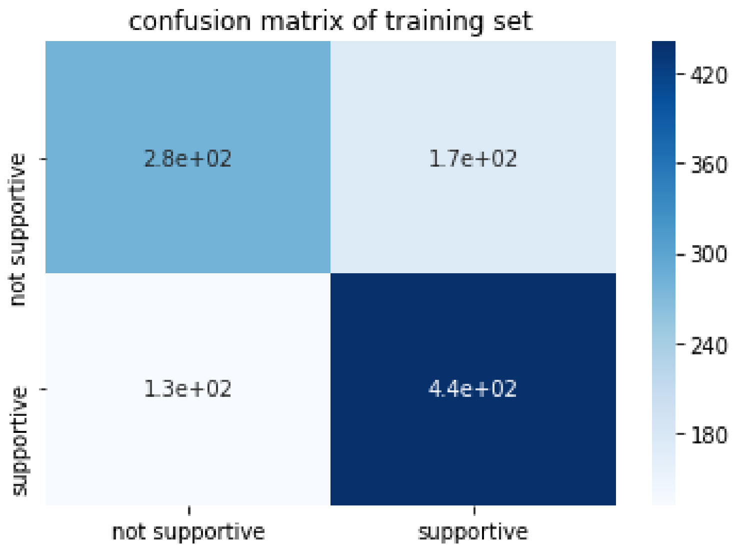
```
Fold: 2
Accuracy Score: 0.6770428015564203
Precision Score: 0.7027027027027027
Recall Score: 0.7272727272727273
f1 Score: 0.7147766323024054
confusion_matrix of training set is:
 [[267 186]
 [133 440]]

confusion_matrix of testing set is:
 [[ 70  44]
 [ 39 104]]

              precision    recall  f1-score   support

           0       0.64      0.61      0.63       114
           1       0.70      0.73      0.71       143

   micro avg       0.68      0.68      0.68       257
   macro avg       0.67      0.67      0.67       257
weighted avg       0.68      0.68      0.68       257
```

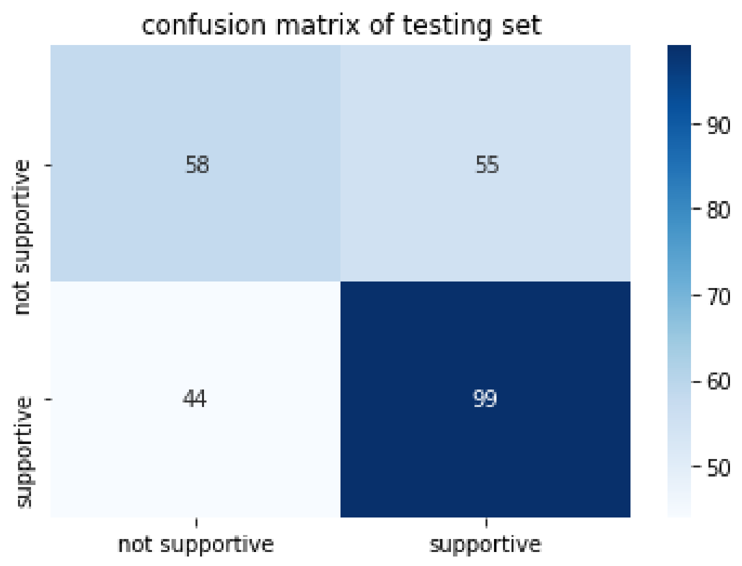




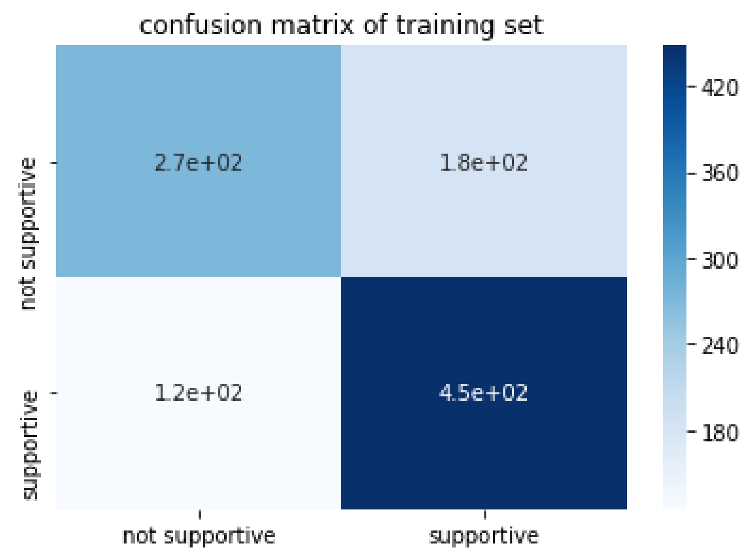
```
Fold: 3
Accuracy Score: 0.61328125
Precision Score: 0.6428571428571429
Recall Score: 0.6923076923076923
f1 Score: 0.6666666666666666
confusion_matrix of training set is:
 [[280 174]
 [132 441]]
```

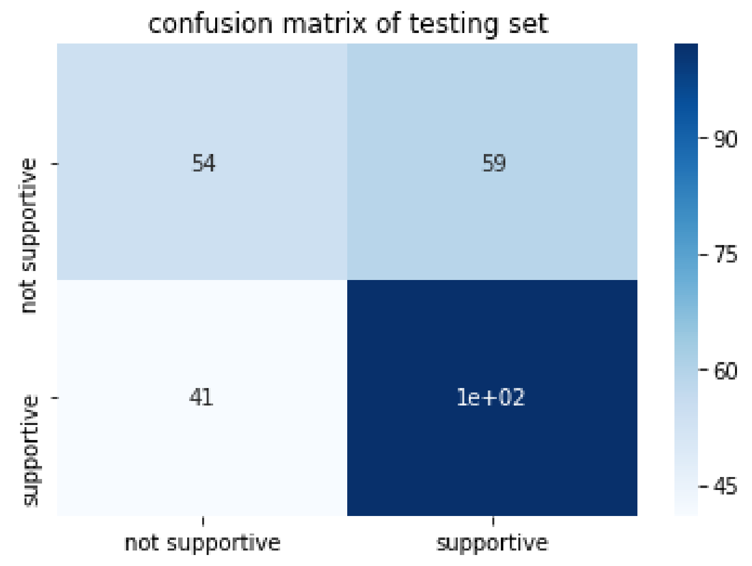
```
confusion_matrix of testing set is:
 [[58 55]
 [44 99]]

              precision    recall  f1-score   support

           0       0.57      0.51      0.54       113
           1       0.64      0.69      0.67       143

   micro avg       0.61      0.61      0.61       256
   macro avg       0.61      0.60      0.60       256
weighted avg       0.61      0.61      0.61       256
```





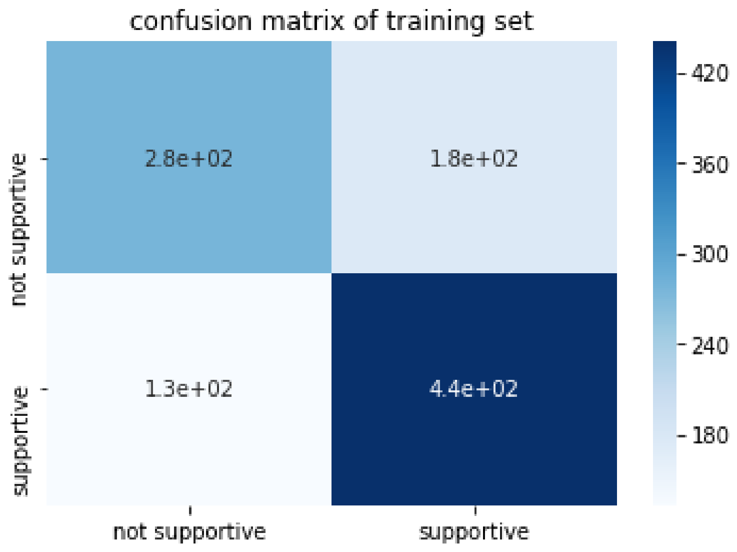
```
Fold: 4
Accuracy Score: 0.609375
Precision Score: 0.6335403726708074
Recall Score: 0.7132867132867133
f1 Score: 0.6710526315789473
confusion_matrix of training set is:
 [[272 182]
 [125 448]]

confusion_matrix of testing set is:
 [[ 54  59]
 [ 41 102]]

              precision    recall  f1-score   support

           0       0.57      0.48      0.52       113
           1       0.63      0.71      0.67       143

   micro avg       0.61      0.61      0.61       256
   macro avg       0.60      0.60      0.60       256
weighted avg       0.60      0.61      0.60       256
```

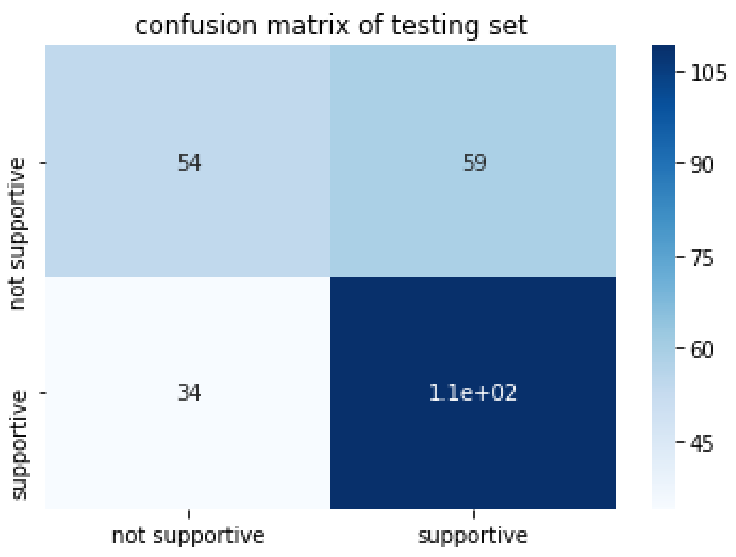




```
Fold: 5
Accuracy Score: 0.63671875
Precision Score: 0.6488095238095238
Recall Score: 0.7622377622377622
f1 Score: 0.7009646302250804
confusion_matrix of training set is:
 [[278 176]
 [133 440]]
```

```
confusion_matrix of testing set is:
 [[ 54  59]
 [ 34 109]]

              precision    recall  f1-score   support

           0       0.61      0.48      0.54       113
           1       0.65      0.76      0.70       143

   micro avg       0.64      0.64      0.64       256
   macro avg       0.63      0.62      0.62       256
weighted avg       0.63      0.64      0.63       256
```



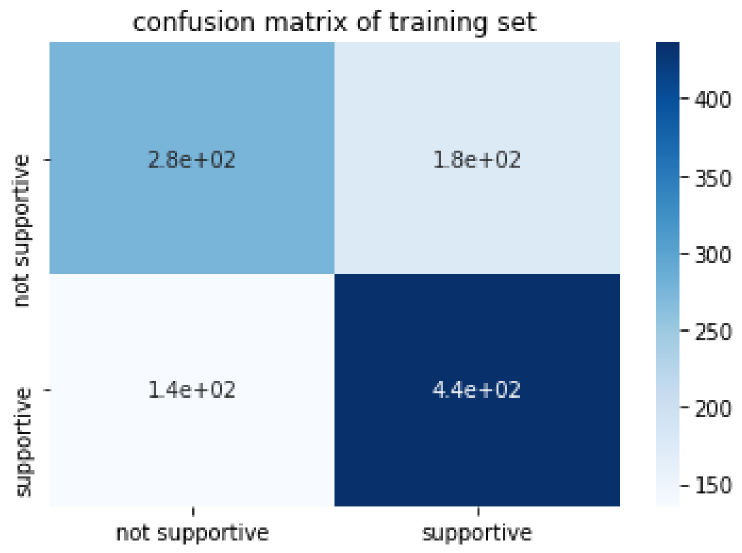


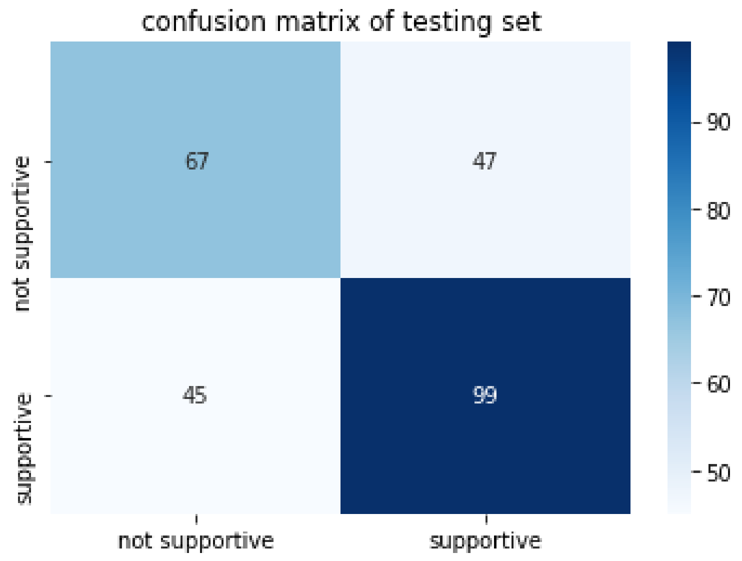
```
Out[297]:    accuracy  precision    recall        f1
          0  0.643411   0.678082  0.687500  0.682759
          1  0.677043   0.702703  0.727273  0.714777
          2  0.613281   0.642857  0.692308  0.666667
          3  0.609375   0.633540  0.713287  0.671053
          4  0.636719   0.648810  0.762238  0.700965
```

```
In [298]: k_fold_evaluate(X, y, max_depth=7, min_samples_leaf=1)
```

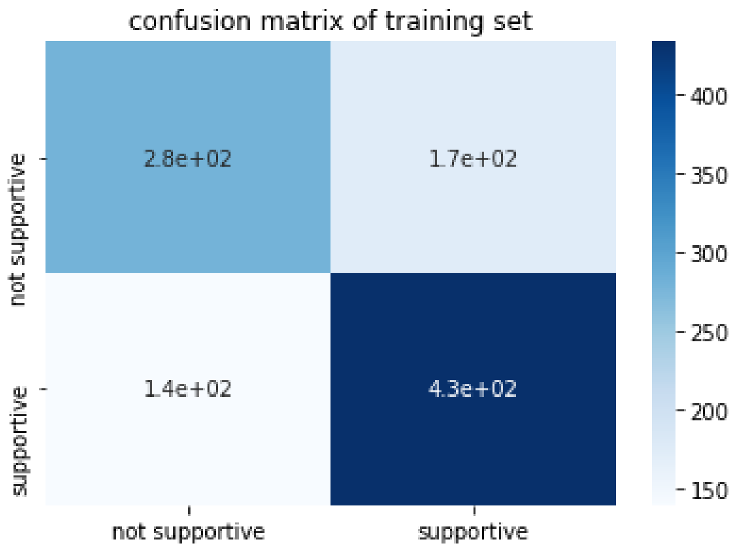
```
There are 453 not supportive tweets in the training set.
There are 572 supportive tweets in the training set.
There are 114 not supportive tweets in the testing set.
There are 144 supportive tweets in the testing set.

Fold: 1
Accuracy Score: 0.6434108527131783
Precision Score: 0.678082191780822
Recall Score: 0.6875
f1 Score: 0.6827586206896552
confusion_matrix of training set is:
 [[276 177]
 [136 436]]

confusion_matrix of testing set is:
 [[67 47]
 [45 99]]

              precision    recall  f1-score   support
```

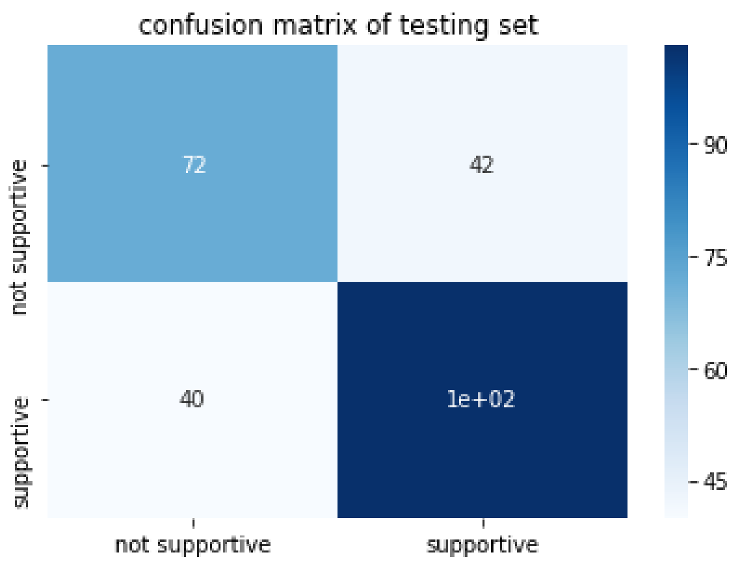
```
           0       0.60      0.59      0.59       114
           1       0.68      0.69      0.68       144

   micro avg       0.64      0.64      0.64       258
   macro avg       0.64      0.64      0.64       258
weighted avg       0.64      0.64      0.64       258
```



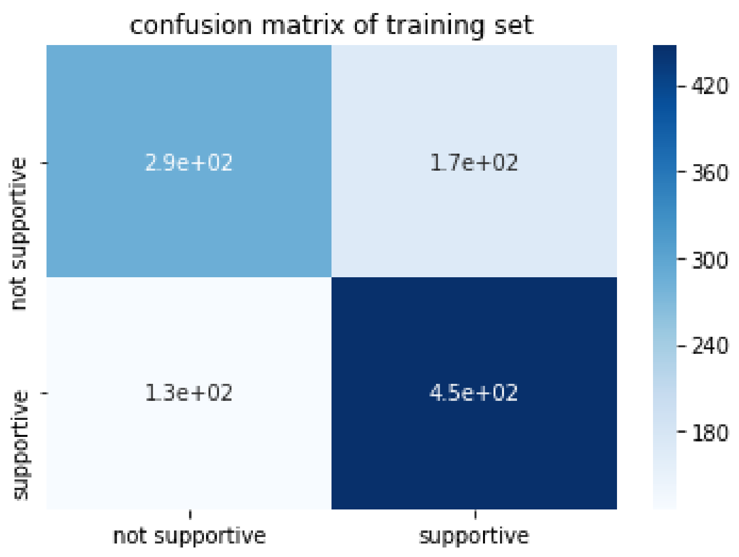


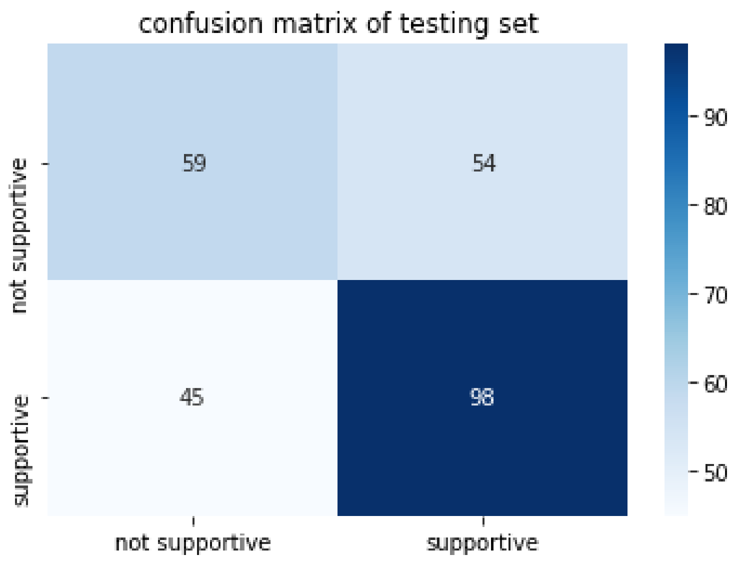
```
Fold: 2
Accuracy Score: 0.6809338521400778
Precision Score: 0.7103448275862069
Recall Score: 0.7202797202797203
f1 Score: 0.7152777777777778
confusion_matrix of training set is:
 [[280 173]
 [139 434]]

confusion_matrix of testing set is:
 [[ 72  42]
 [ 40 103]]

              precision    recall  f1-score   support

           0       0.64      0.63      0.64       114
           1       0.71      0.72      0.72       143

   micro avg       0.68      0.68      0.68       257
   macro avg       0.68      0.68      0.68       257
weighted avg       0.68      0.68      0.68       257
```





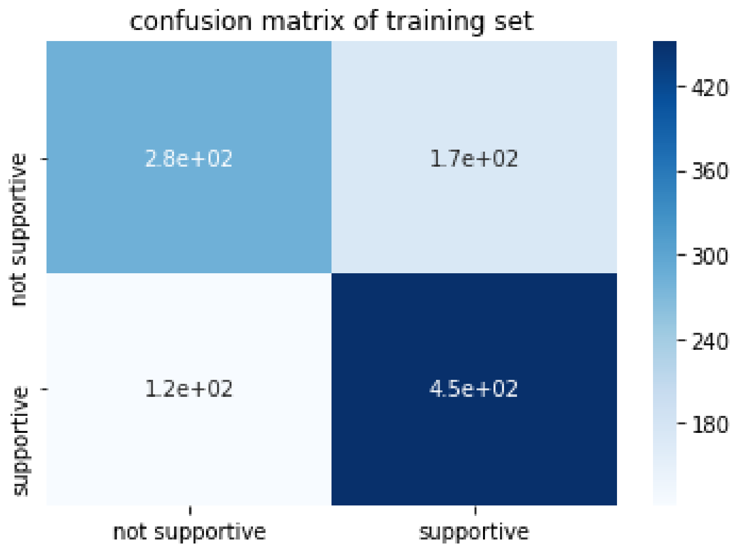
```
Fold: 3
Accuracy Score: 0.61328125
Precision Score: 0.6447368421052632
Recall Score: 0.6853146853146853
f1 Score: 0.6644067796610169
confusion_matrix of training set is:
 [[286 168]
 [126 447]]

confusion_matrix of testing set is:
 [[59 54]
 [45 98]]

              precision    recall  f1-score   support

           0       0.57      0.52      0.54       113
           1       0.64      0.69      0.66       143

   micro avg       0.61      0.61      0.61       256
   macro avg       0.61      0.60      0.60       256
weighted avg       0.61      0.61      0.61       256
```

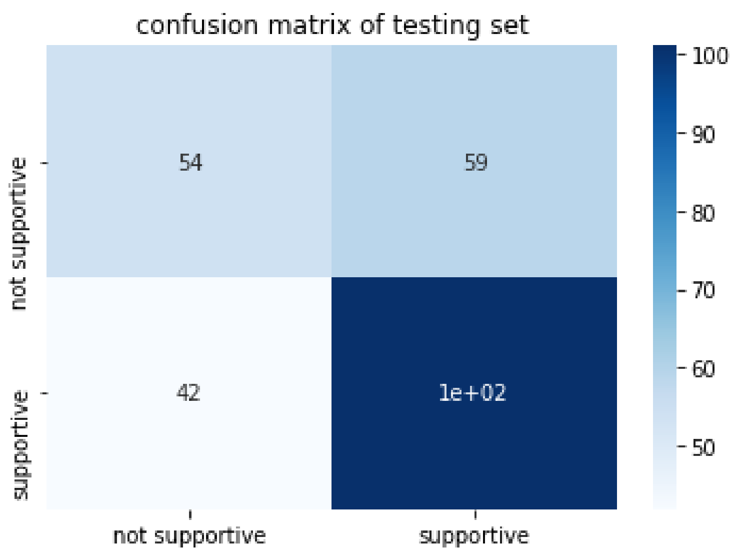




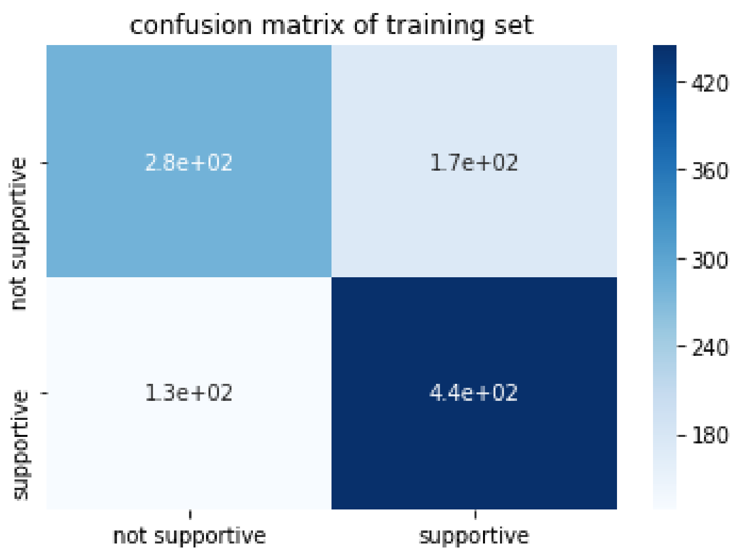
```
Fold: 4
Accuracy Score: 0.60546875
Precision Score: 0.63125
Recall Score: 0.7062937062937062
f1 Score: 0.6666666666666665
confusion_matrix of training set is:
 [[283 171]
 [122 451]]
```

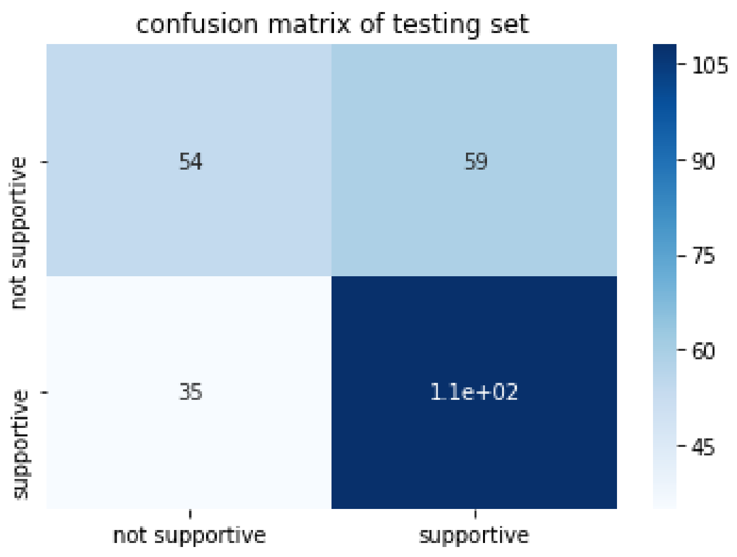
```
confusion_matrix of testing set is:
 [[ 54  59]
 [ 42 101]]

              precision    recall  f1-score   support

           0       0.56      0.48      0.52       113
           1       0.63      0.71      0.67       143

   micro avg       0.61      0.61      0.61       256
   macro avg       0.60      0.59      0.59       256
weighted avg       0.60      0.61      0.60       256
```





```
Fold: 5
Accuracy Score: 0.6328125
Precision Score: 0.6467065868263473
Recall Score: 0.7552447552447552
f1 Score: 0.6967741935483871
confusion_matrix of training set is:
 [[281 173]
 [129 444]]

confusion_matrix of testing set is:
 [[ 54  59]
 [ 35 108]]

              precision    recall  f1-score   support

           0       0.61      0.48      0.53       113
           1       0.65      0.76      0.70       143

   micro avg       0.63      0.63      0.63       256
   macro avg       0.63      0.62      0.62       256
weighted avg       0.63      0.63      0.63       256
```



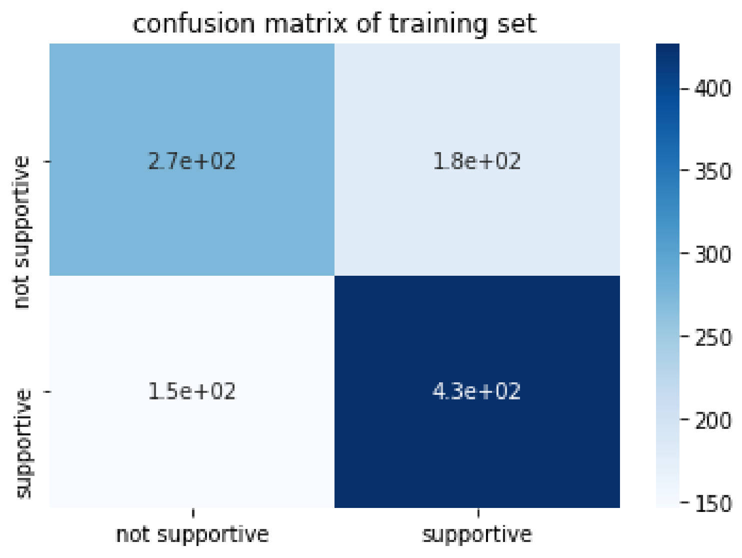


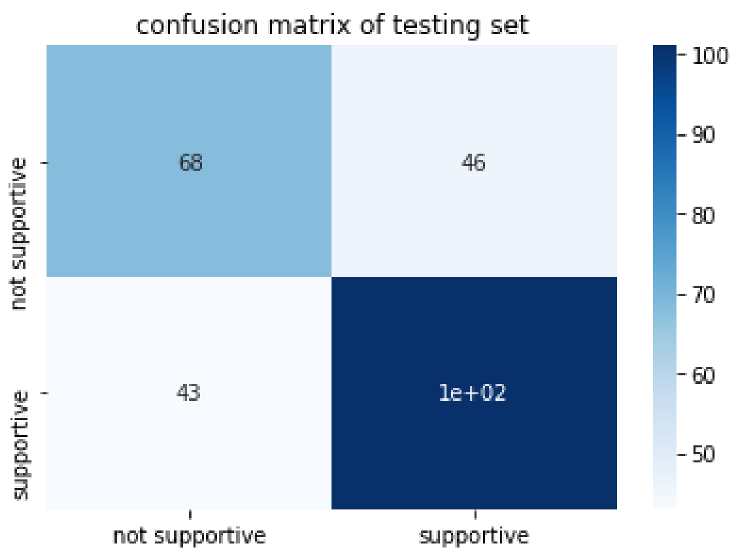
```
Out[298]:    accuracy  precision    recall        f1
          0  0.643411   0.678082  0.687500  0.682759
          1  0.680934   0.710345  0.720280  0.715278
          2  0.613281   0.644737  0.685315  0.664407
          3  0.605469   0.631250  0.706294  0.666667
          4  0.632812   0.646707  0.755245  0.696774
```

chosen parameters: `max_depth=7, min_samples_leaf=2`

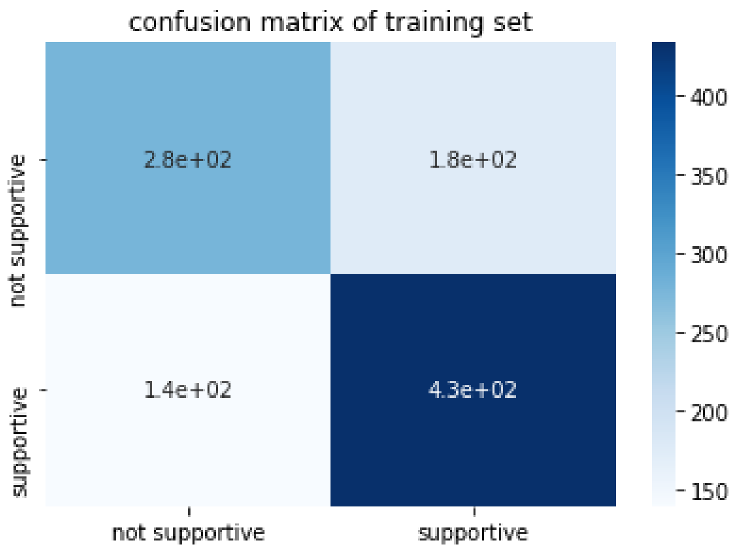
Because the 5-fold f1 score of max_depth=7, min_samples_leaf=2 are better.

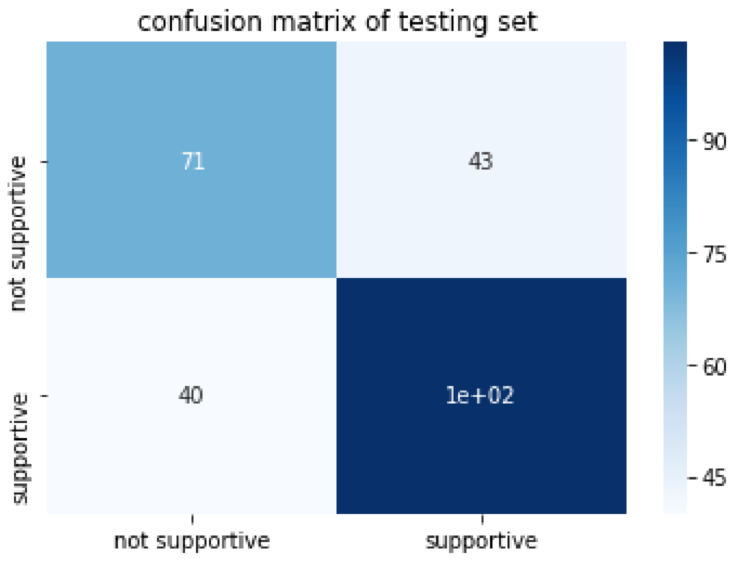
```
In [301]: k_fold_evaluate(X, y, max_depth=7, min_samples_leaf=2)

There are 453 not supportive tweets in the training set.
There are 572 supportive tweets in the training set.
There are 114 not supportive tweets in the testing set.
There are 144 supportive tweets in the testing set.

Fold: 1
Accuracy Score: 0.6550387596899225
Precision Score: 0.6870748299319728
Recall Score: 0.7013888888888888
f1 Score: 0.6941580756013744
confusion_matrix of training set is:
 [[273 180]
 [146 426]]

confusion_matrix of testing set is:
 [[ 68  46]
 [ 43 101]]

              precision    recall  f1-score   support

           0       0.61      0.60      0.60       114
           1       0.69      0.70      0.69       144

   micro avg       0.66      0.66      0.66       258
   macro avg       0.65      0.65      0.65       258
weighted avg       0.65      0.66      0.65       258
```





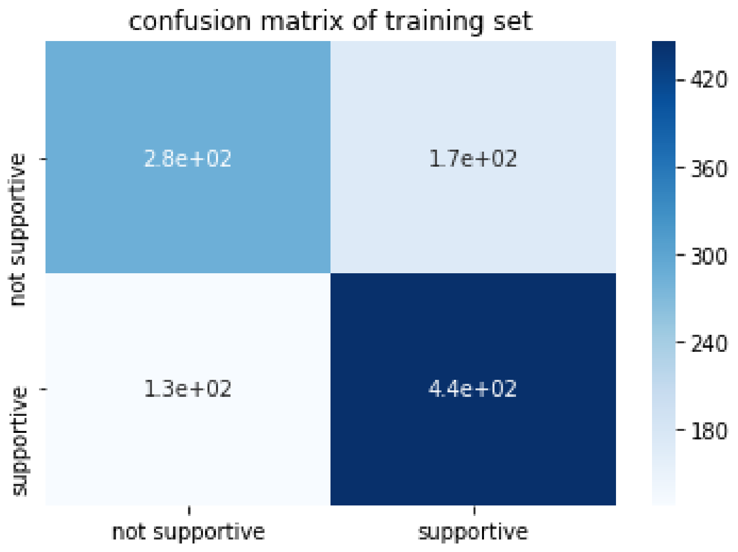
```
Fold: 2
Accuracy Score: 0.6770428015564203
Precision Score: 0.7054794520547946
Recall Score: 0.7202797202797203
f1 Score: 0.71280276816609
confusion_matrix of training set is:
 [[278 175]
 [139 434]]

confusion_matrix of testing set is:
 [[ 71  43]
 [ 40 103]]

              precision    recall  f1-score   support

           0       0.64      0.62      0.63       114
           1       0.71      0.72      0.71       143

   micro avg       0.68      0.68      0.68       257
   macro avg       0.67      0.67      0.67       257
weighted avg       0.68      0.68      0.68       257
```

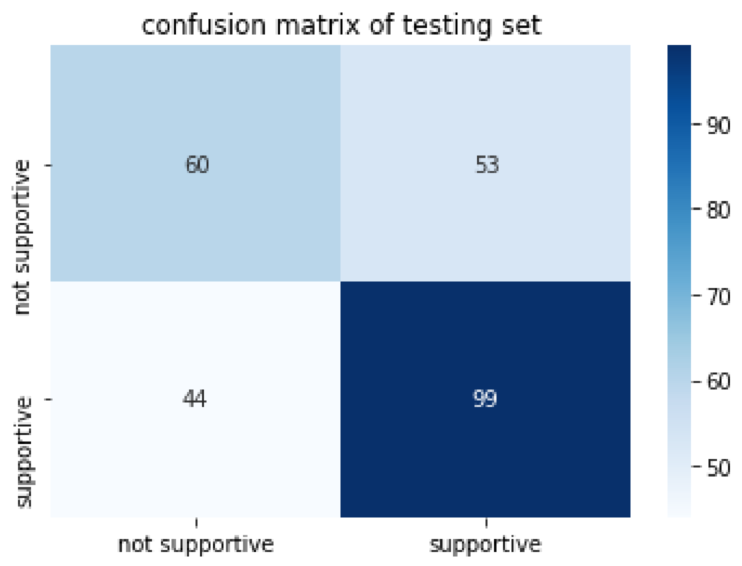




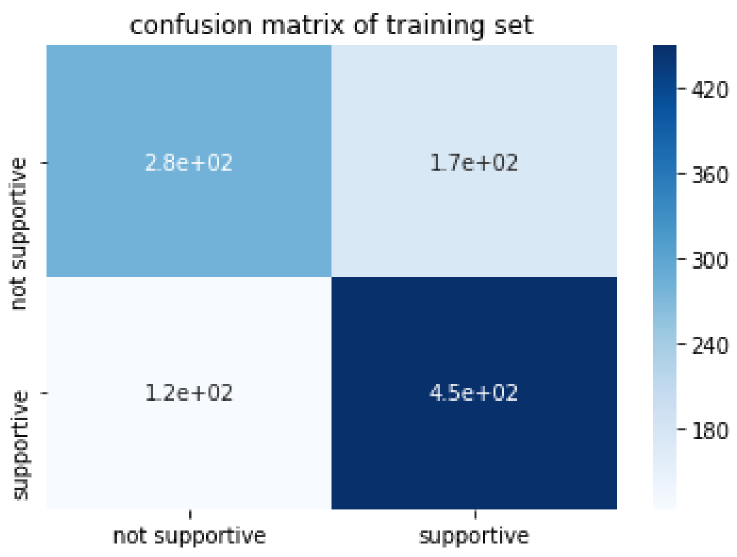
```
Fold: 3
Accuracy Score: 0.62109375
Precision Score: 0.6513157894736842
Recall Score: 0.6923076923076923
f1 Score: 0.6711864406779661
confusion_matrix of training set is:
 [[285 169]
 [128 445]]
```

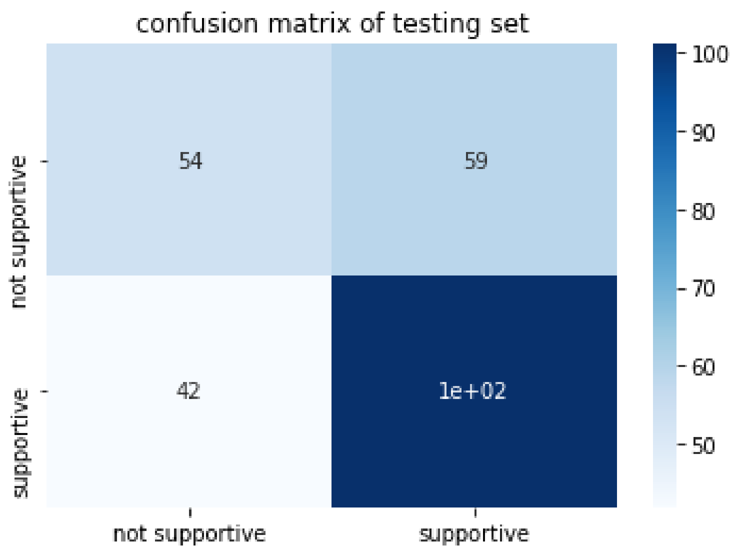
```
confusion_matrix of testing set is:
 [[60 53]
 [44 99]]

              precision    recall  f1-score   support

           0       0.58      0.53      0.55       113
           1       0.65      0.69      0.67       143

   micro avg       0.62      0.62      0.62       256
   macro avg       0.61      0.61      0.61       256
weighted avg       0.62      0.62      0.62       256
```





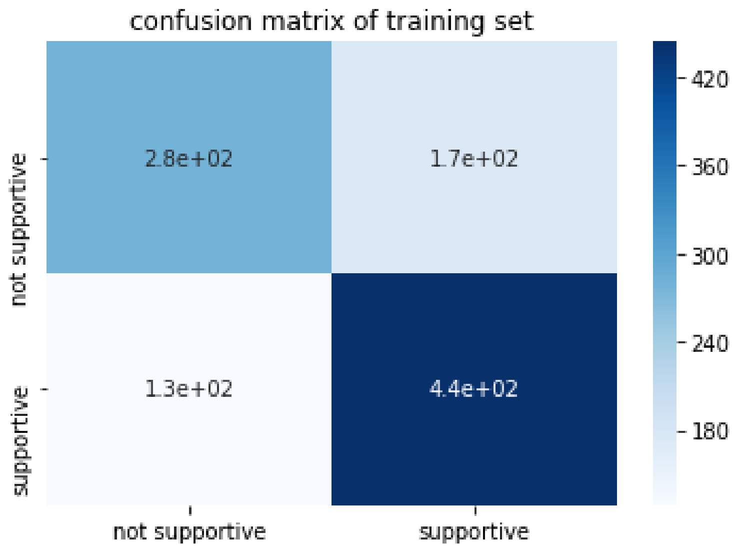
```
Fold: 4
Accuracy Score: 0.60546875
Precision Score: 0.63125
Recall Score: 0.7062937062937062
f1 Score: 0.6666666666666665
confusion_matrix of training set is:
 [[281 173]
 [124 449]]

confusion_matrix of testing set is:
 [[ 54  59]
 [ 42 101]]

              precision    recall  f1-score   support

           0       0.56      0.48      0.52       113
           1       0.63      0.71      0.67       143

   micro avg       0.61      0.61      0.61       256
   macro avg       0.60      0.59      0.59       256
weighted avg       0.60      0.61      0.60       256
```

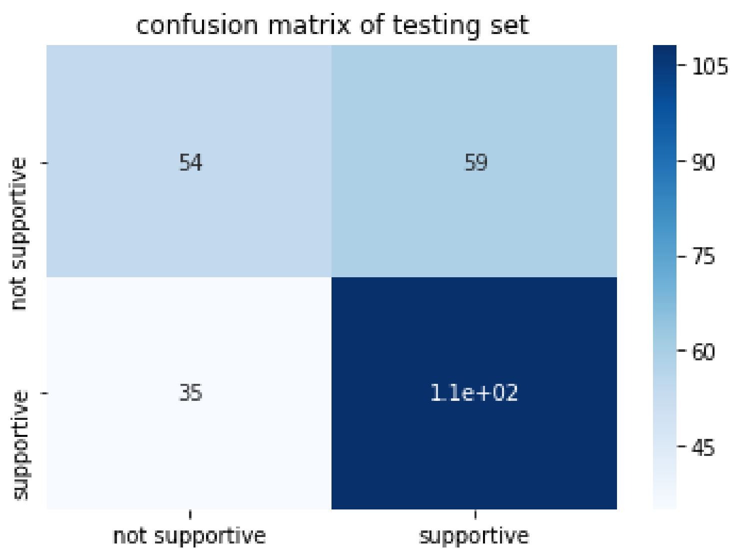




```
Fold: 5
Accuracy Score: 0.6328125
Precision Score: 0.6467065868263473
Recall Score: 0.7552447552447552
f1 Score: 0.6967741935483871
confusion_matrix of training set is:
 [[280 174]
 [129 444]]
```

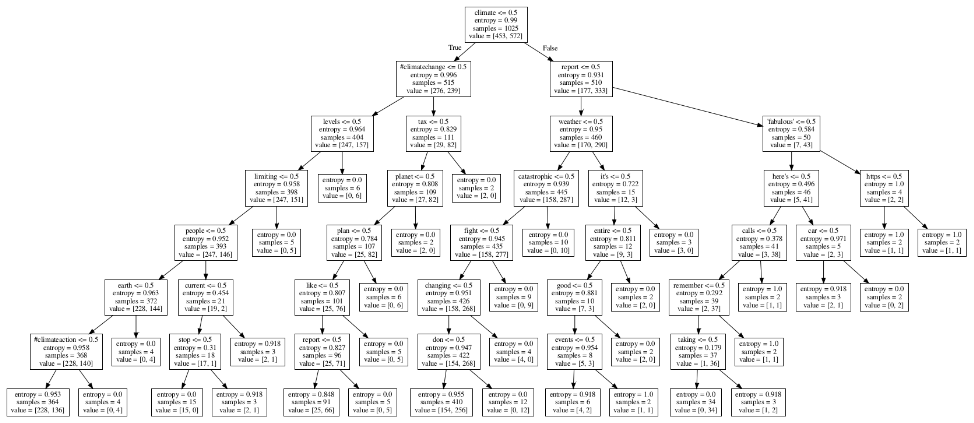
```
confusion_matrix of testing set is:
 [[ 54  59]
 [ 35 108]]

              precision    recall  f1-score   support

           0       0.61      0.48      0.53       113
           1       0.65      0.76      0.70       143

   micro avg       0.63      0.63      0.63       256
   macro avg       0.63      0.62      0.62       256
weighted avg       0.63      0.63      0.63       256
```



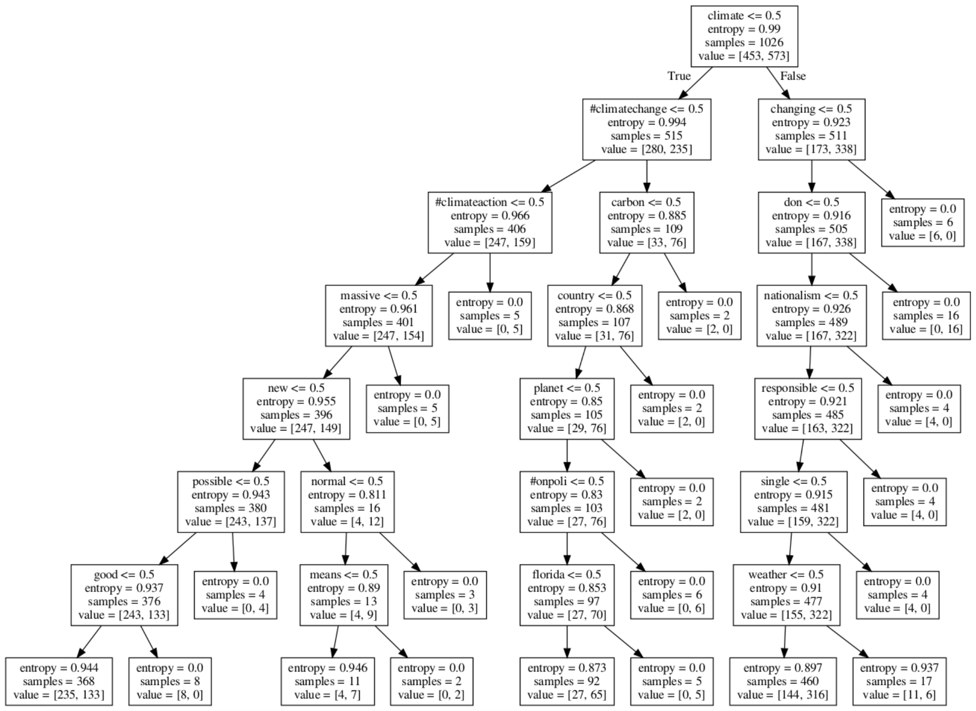


```
Out[301]:    accuracy  precision    recall        f1
          0  0.655039   0.687075  0.701389  0.694158
          1  0.677043   0.705479  0.720280  0.712803
          2  0.621094   0.651316  0.692308  0.671186
          3  0.605469   0.631250  0.706294  0.666667
          4  0.632812   0.646707  0.755245  0.696774
```

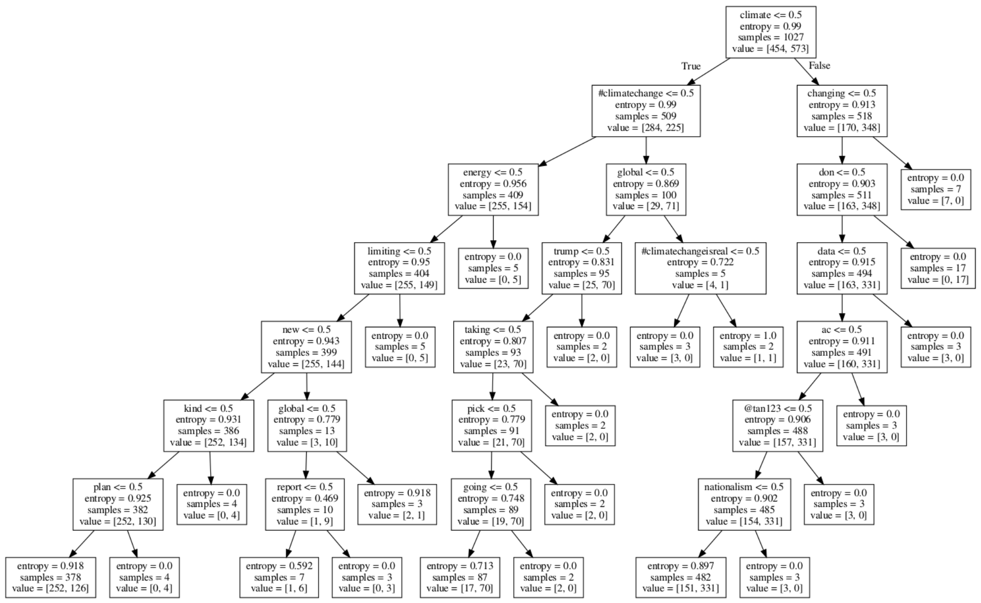
Fold 1


climate <= 0.5 entropy = 0.99 samples = 1025 value = [453, 572]
True
False
#climatechange <= 0.5 entropy = 0.996 samples = 515 value = [276, 239]
report <= 0.5 entropy = 0.931 samples = 510 value = [177, 333]
levels <= 0.5 entropy = 0.964 samples = 404 value = [247, 157]
tax <= 0.5 entropy = 0.829 samples = 111 value = [29, 82]
weather <= 0.5 entropy = 0.95 samples = 460 value = [170, 290]
'fabulous' <= 0.5 entropy = 0.584 samples = 50 value = [7, 43]
limiting <= 0.5 entropy = 0.958 samples = 398 value = [247, 151]
entropy = 0.0 samples = 6 value = [0, 6]
planet <= 0.5 entropy = 0.808 samples = 109 value = [27, 82]
entropy = 0.0 samples = 2 value = [2, 0]
catastrophic <= 0.5 entropy = 0.939 samples = 445 value = [158, 287]
it's <= 0.5 entropy = 0.722 samples = 15 value = [12, 3]
here's <= 0.5 entropy = 0.496 samples = 46 value = [5, 41]
https <= 0.5 entropy = 1.0 samples = 4 value = [2, 2]
people <= 0.5 entropy = 0.952 samples = 393 value = [247, 146]
entropy = 0.0 samples = 5 value = [0, 5]
plan <= 0.5 entropy = 0.784 samples = 107 value = [25, 82]
entropy = 0.0 samples = 2 value = [2, 0]
fight <= 0.5 entropy = 0.945 samples = 435 value = [158, 277]
entropy = 0.0 samples = 10 value = [0, 10]
entire <= 0.5 entropy = 0.811 samples = 12 value = [9, 3]
entropy = 0.0 samples = 3 value = [3, 0]
calls <= 0.5 entropy = 0.378 samples = 41 value = [3, 38]
car <= 0.5 entropy = 0.971 samples = 5 value = [2, 3]
entropy = 1.0 samples = 2 value = [1, 1]
entropy = 1.0 samples = 2 value = [1, 1]
earth <= 0.5 entropy = 0.963 samples = 372 value = [228, 144]
current <= 0.5 entropy = 0.454 samples = 21 value = [19, 2]
like <= 0.5 entropy = 0.807 samples = 101 value = [25, 76]
entropy = 0.0 samples = 6 value = [0, 6]
changing <= 0.5 entropy = 0.951 samples = 426 value = [158, 268]
entropy = 0.0 samples = 9 value = [0, 9]
good <= 0.5 entropy = 0.881 samples = 10 value = [7, 3]
entropy = 0.0 samples = 2 value = [2, 0]
remember <= 0.5 entropy = 0.292 samples = 39 value = [2, 37]
entropy = 1.0 samples = 2 value = [1, 1]
entropy = 0.918 samples = 3 value = [2, 1]
entropy = 0.0 samples = 2 value = [0, 2]
#climateaction <= 0.5 entropy = 0.958 samples = 368 value = [228, 140]
entropy = 0.0 samples = 4 value = [0, 4]
stop <= 0.5 entropy = 0.31 samples = 18 value = [17, 1]
entropy = 0.918 samples = 3 value = [2, 1]
report <= 0.5 entropy = 0.827 samples = 96 value = [25, 71]
entropy = 0.0 samples = 5 value = [0, 5]
don <= 0.5 entropy = 0.947 samples = 422 value = [154, 268]
entropy = 0.0 samples = 4 value = [4, 0]
events <= 0.5 entropy = 0.954 samples = 8 value = [5, 3]
entropy = 0.0 samples = 2 value = [2, 0]
taking <= 0.5 entropy = 0.179 samples = 37 value = [1, 36]
entropy = 1.0 samples = 2 value = [1, 1]
entropy = 0.953 samples = 364 value = [228, 136]
entropy = 0.0 samples = 4 value = [0, 4]
entropy = 0.0 samples = 15 value = [15, 0]
entropy = 0.918 samples = 3 value = [2, 1]
entropy = 0.848 samples = 91 value = [25, 66]
entropy = 0.0 samples = 5 value = [0, 5]
entropy = 0.955 samples = 410 value = [154, 256]
entropy = 0.0 samples = 12 value = [0, 12]
entropy = 0.918 samples = 6 value = [4, 2]
entropy = 1.0 samples = 2 value = [1, 1]
entropy = 0.0 samples = 34 value = [0, 34]
entropy = 0.918 samples = 3 value = [1, 2]

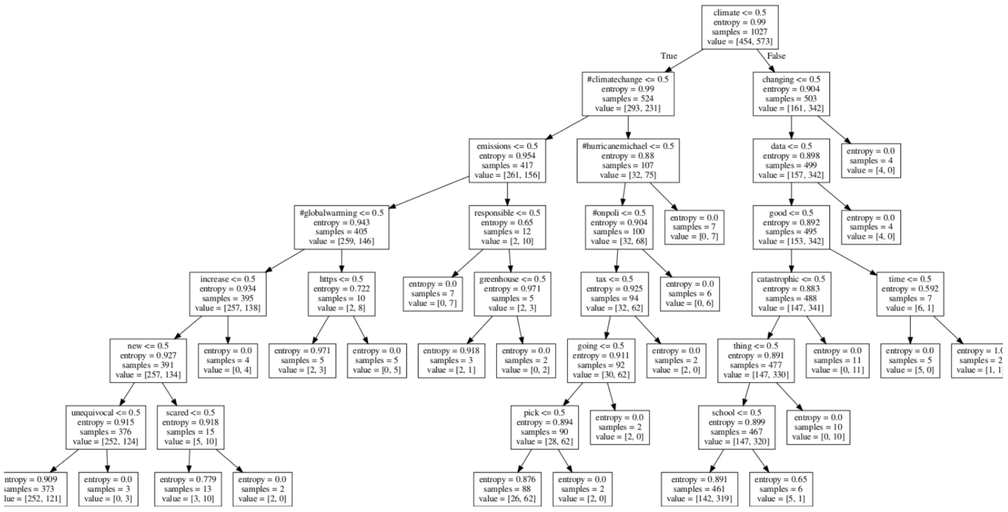
Fold 2


climate <= 0.5 entropy = 0.99 samples = 1026 value = [453, 573]
True
False
#climatechange <= 0.5 entropy = 0.994 samples = 515 value = [280, 235]
changing <= 0.5 entropy = 0.923 samples = 511 value = [173, 338]
#climateaction <= 0.5 entropy = 0.966 samples = 406 value = [247, 159]
carbon <= 0.5 entropy = 0.885 samples = 109 value = [33, 76]
don <= 0.5 entropy = 0.916 samples = 505 value = [167, 338]
entropy = 0.0 samples = 6 value = [6, 0]
massive <= 0.5 entropy = 0.961 samples = 401 value = [247, 154]
entropy = 0.0 samples = 5 value = [0, 5]
country <= 0.5 entropy = 0.868 samples = 107 value = [31, 76]
entropy = 0.0 samples = 2 value = [2, 0]
nationalism <= 0.5 entropy = 0.926 samples = 489 value = [167, 322]
entropy = 0.0 samples = 16 value = [0, 16]
new <= 0.5 entropy = 0.955 samples = 396 value = [247, 149]
entropy = 0.0 samples = 5 value = [0, 5]
planet <= 0.5 entropy = 0.85 samples = 105 value = [29, 76]
entropy = 0.0 samples = 2 value = [2, 0]
responsible <= 0.5 entropy = 0.921 samples = 485 value = [163, 322]
entropy = 0.0 samples = 4 value = [4, 0]
possible <= 0.5 entropy = 0.943 samples = 380 value = [243, 137]
normal <= 0.5 entropy = 0.811 samples = 16 value = [4, 12]
#onpoli <= 0.5 entropy = 0.83 samples = 103 value = [27, 76]
entropy = 0.0 samples = 2 value = [2, 0]
single <= 0.5 entropy = 0.915 samples = 481 value = [159, 322]
entropy = 0.0 samples = 4 value = [4, 0]
good <= 0.5 entropy = 0.937 samples = 376 value = [243, 133]
entropy = 0.0 samples = 4 value = [0, 4]
means <= 0.5 entropy = 0.89 samples = 13 value = [4, 9]
entropy = 0.0 samples = 3 value = [0, 3]
florida <= 0.5 entropy = 0.853 samples = 97 value = [27, 70]
entropy = 0.0 samples = 6 value = [0, 6]
weather <= 0.5 entropy = 0.91 samples = 477 value = [155, 322]
entropy = 0.0 samples = 4 value = [4, 0]
entropy = 0.944 samples = 368 value = [235, 133]
entropy = 0.0 samples = 8 value = [8, 0]
entropy = 0.946 samples = 11 value = [4, 7]
entropy = 0.0 samples = 2 value = [0, 2]
entropy = 0.873 samples = 92 value = [27, 65]
entropy = 0.0 samples = 5 value = [0, 5]
entropy = 0.897 samples = 460 value = [144, 316]
entropy = 0.937 samples = 17 value = [11, 6]

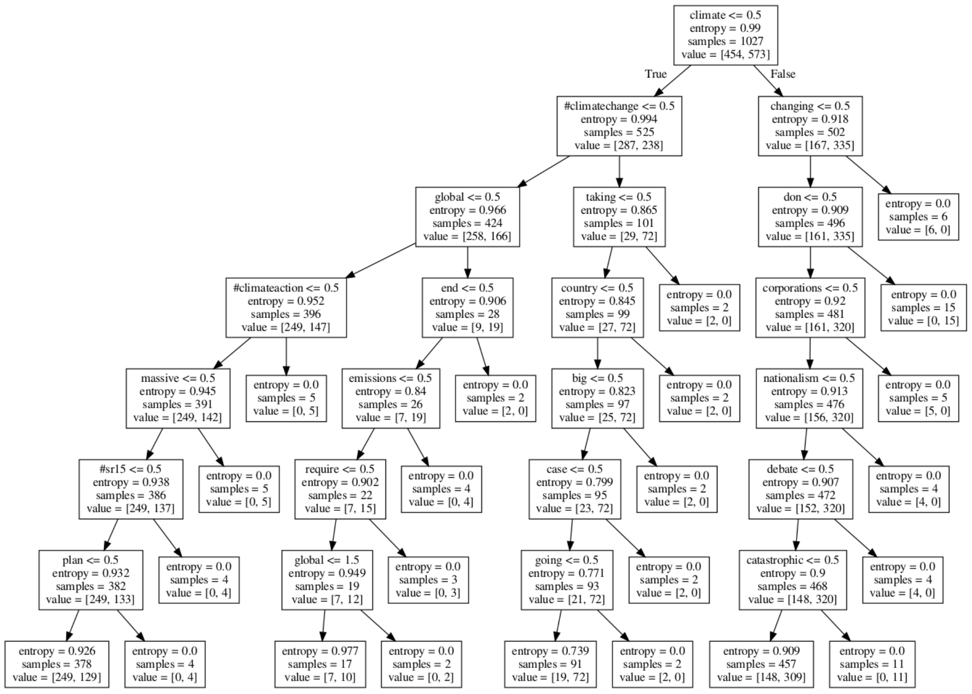
Fold 3


climate <= 0.5 entropy = 0.99 samples = 1027 value = [454, 573]
True
False
#climatechange <= 0.5 entropy = 0.99 samples = 509 value = [284, 225]
changing <= 0.5 entropy = 0.913 samples = 518 value = [170, 348]
energy <= 0.5 entropy = 0.956 samples = 409 value = [255, 154]
global <= 0.5 entropy = 0.869 samples = 100 value = [29, 71]
don <= 0.5 entropy = 0.903 samples = 511 value = [163, 348]
entropy = 0.0 samples = 7 value = [7, 0]
limiting <= 0.5 entropy = 0.95 samples = 404 value = [255, 149]
entropy = 0.0 samples = 5 value = [0, 5]
trump <= 0.5 entropy = 0.831 samples = 95 value = [25, 70]
#climatechangeisreal <= 0.5 entropy = 0.722 samples = 5 value = [4, 1]
data <= 0.5 entropy = 0.915 samples = 494 value = [163, 331]
entropy = 0.0 samples = 17 value = [0, 17]
new <= 0.5 entropy = 0.943 samples = 399 value = [255, 144]
entropy = 0.0 samples = 5 value = [0, 5]
taking <= 0.5 entropy = 0.807 samples = 93 value = [23, 70]
entropy = 0.0 samples = 2 value = [2, 0]
entropy = 0.0 samples = 3 value = [3, 0]
entropy = 1.0 samples = 2 value = [1, 1]
ac <= 0.5 entropy = 0.911 samples = 491 value = [160, 331]
entropy = 0.0 samples = 3 value = [3, 0]
kind <= 0.5 entropy = 0.931 samples = 386 value = [252, 134]
global <= 0.5 entropy = 0.779 samples = 13 value = [3, 10]
pick <= 0.5 entropy = 0.779 samples = 91 value = [21, 70]
entropy = 0.0 samples = 2 value = [2, 0]
@tan123 <= 0.5 entropy = 0.906 samples = 488 value = [157, 331]
entropy = 0.0 samples = 3 value = [3, 0]
plan <= 0.5 entropy = 0.925 samples = 382 value = [252, 130]
entropy = 0.0 samples = 4 value = [0, 4]
report <= 0.5 entropy = 0.469 samples = 10 value = [1, 9]
entropy = 0.918 samples = 3 value = [2, 1]
going <= 0.5 entropy = 0.748 samples = 89 value = [19, 70]
entropy = 0.0 samples = 2 value = [2, 0]
nationalism <= 0.5 entropy = 0.902 samples = 485 value = [154, 331]
entropy = 0.0 samples = 3 value = [3, 0]
entropy = 0.918 samples = 378 value = [252, 126]
entropy = 0.0 samples = 4 value = [0, 4]
entropy = 0.592 samples = 7 value = [1, 6]
entropy = 0.0 samples = 3 value = [0, 3]
entropy = 0.713 samples = 87 value = [17, 70]
entropy = 0.0 samples = 2 value = [2, 0]
entropy = 0.897 samples = 482 value = [151, 331]
entropy = 0.0 samples = 3 value = [3, 0]

Fold 4



Fold 5



reload pickle

```
In [300]: clf2 = joblib.load('ClimateTeam7PD1.pkl')
          y_pred = clf2.predict(X)
          f1_score(y, y_pred)
```

```
Out[300]: 0.7355096602265155
```