Assignment 6

Jiarong Ye

October 1, 2018

import packages

In [1]:
```
import datascience as ds
from datascience import *
import numpy as np
from graphviz import Source
import pandas as pd
import re, string
import nltk
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
from sklearn.ensemble import AdaBoostClassifier
from xgboost import XGBClassifier as XGBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree
from sklearn.metrics import confusion_matrix, precision_score,
                        recall_score, f1_score, accuracy_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold

%matplotlib inline
```

In [2]:
```
def process_text(data):
    cleaned_text = [
        re.sub('\s+', ' ',
               re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|[0-9]+|http.+?", '',
               tweets.lower()).strip(string.punctuation).strip()) for tweets in data
        ]
    return cleaned_text
```

In [53]:
```
tagged = pd.read_csv('tagged.csv', sep=',', index_col=False)
tagged.text = process_text(tagged.text)
tagged = tagged.drop(tagged.columns[0], axis=1).reset_index(drop=True)
tagged = tagged.drop(tagged[tagged.sentiment==-1].index).reset_index(drop=True)
tagged = tagged.iloc[tagged.text.drop_duplicates().index]
tagged.to_csv('relevant_tagged.csv')
```

tweets data loaded into Jupyter Notebook as Table object

```
In [4]: df = ds.Table.read_table('relevant_tagged.csv', sep=',')
        df
```

```
Out[4]: Unnamed: 0 | user_id             | user_name                      | tweet_time
        0          | 802657195661742080  | Christine Warren               | Wed Sep 12 01:38:14 +0000 20
        1          | 1039245812230893570 | Trumpservative                 | Wed Sep 12 01:38:16 +0000 20
        2          | 282084840           | Darrel Sheldon #MAGAVETERAN    | Wed Sep 12 01:38:18 +0000 20
        3          | 62315639            | Queer Liberal Voting Snowflake | Wed Sep 12 01:38:18 +0000 20
        4          | 340428574           | DelcoGal                       | Wed Sep 12 01:38:21 +0000 20
        5          | 1603928228          | Julz                           | Wed Sep 12 01:38:22 +0000 20
        6          | 1865678516          | Barbara Kuczinski              | Wed Sep 12 01:38:22 +0000 20
        7          | 59288409            | Josh Steed PhD                 | Wed Sep 12 01:38:25 +0000 20
        8          | 325172419           | Mrs. Linz                      | Wed Sep 12 01:38:25 +0000 20
        9          | 2283127514          | Garrett ODowd                  | Wed Sep 12 01:38:26 +0000 20
        ... (1002 rows omitted)
```

StratifiedKFold

```
In [5]: X = list(df['text'])
        y = list(df['sentiment'])
```

Check whether the data distribution is balanced

```
In [6]: def check(sentiment, index, note='training'):
            if sentiment==0:
                label = 'neutral'
            elif sentiment==1:
                label = 'positive'
            else:
                label = 'negative'
            print('There are {} '.format(df.take(index).where('sentiment',
                  are.equal_to(sentiment)).size[0][0])+label+' tweets in the '+note+' set.')
```

Model Building

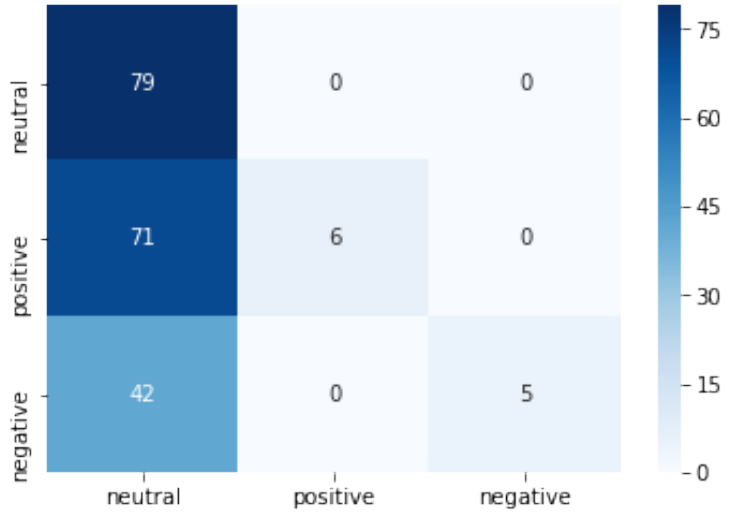
```
In [7]: def custom_split(train_index, test_index):
            trainingset = df.take(train_index)
            testingset = df.take(test_index)

            X_train= list(trainingset['text'])
            y_train= list(trainingset['sentiment'])
            X_test= list(testingset['text'])
            y_test= list(testingset['sentiment'])

            return X_train, X_test, y_train, y_test
```

```
In [8]: def word_vectorizer(X_train, X_test):
            vect = CountVectorizer(
                 analyzer="word", ngram_range=([1,2]), tokenizer=nltk.word_tokenize,
                preprocessor=None, stop_words='english', max_features=3000)
        #     vect = TfidfVectorizer(sublinear_tf=True, min_df=10, norm='l1', encoding='latin-1',
        #                            ngram_range=(1,2), stop_words='english')
            X_train_vect = vect.fit_transform(X_train).todense()
            X_test_vect = vect.transform(X_test)
            return X_train_vect, X_test_vect, vect.get_feature_names()

In [32]: def classifier(X_train, y_train, X_test, fold, feature_names):
             clf = DecisionTreeClassifier(criterion = 'entropy',
                                          random_state = 100,
                                          max_depth = 5,
                                          min_samples_leaf = 2)

             clf.fit(X_train, y_train)
             try:
                 dot_data = tree.export_graphviz(clf, out_file=None,
                                                 feature_names=feature_names)
                 graph = Source(dot_data)
                 graph.render('SentientClassifier-Fold_{}'.format(fold))
             except Exception as e:
                 print(e)
             predicted_y_test = clf.predict(X_test)
             return predicted_y_test

In [33]: def eval_results(predicted_y_test, y_test):
             print('\n Testing precision are: ', predicted_y_test, '\n')
             precision_s = precision_score(y_test, predicted_y_test, average='weighted')
             recall_s = recall_score(y_test, predicted_y_test, average='weighted')
             f1_s = f1_score(y_test, predicted_y_test, average='weighted')
             cm = confusion_matrix(y_test, predicted_y_test)

             print("Precision Score:", precision_s)
             print("Recall Score:", recall_s)
             print("f1 Score:", f1_s)
             print('confusion_matrix is: \n', cm, '\n')
             return precision_s, recall_s, f1_s, cm

In [47]: accuracy_ = []
         precision = []
         recall=[]
         f1 = []

         def k_fold_evaluate(X, y, n_splits):
             # initialization
             classes = ['neutral', 'positive', 'negative']
```

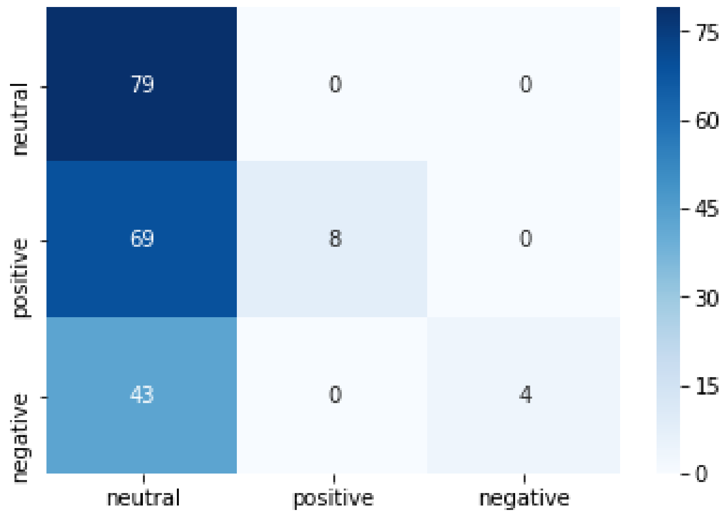
```
            fold = 1
            skf = StratifiedKFold(n_splits=n_splits, random_state=1, shuffle= True)

            # 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(3)))
                    list(map(lambda x: check(x, test_index, note='testing'), range(3)))

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

                X_train_vect, X_test_vect, feature_names = word_vectorizer(X_train, X_test)
                predicted_y_test = classifier(X_train=X_train_vect, y_train=y_train,
                                              X_test=X_test_vect, fold=fold, feature_names=feature_name
                precision_s, recall_s, f1_s, cm = eval_results(predicted_y_test, y_test)

                precision.append(precision_s)
                recall.append(recall_s)
                f1.append(f1_s)

                sns.heatmap(cm, annot=True, cmap='Blues', yticklabels=classes, xticklabels=classes)
                plt.show()

                fold += 1
```

```
In [48]: k_fold_evaluate(X, y, n_splits=5)
```

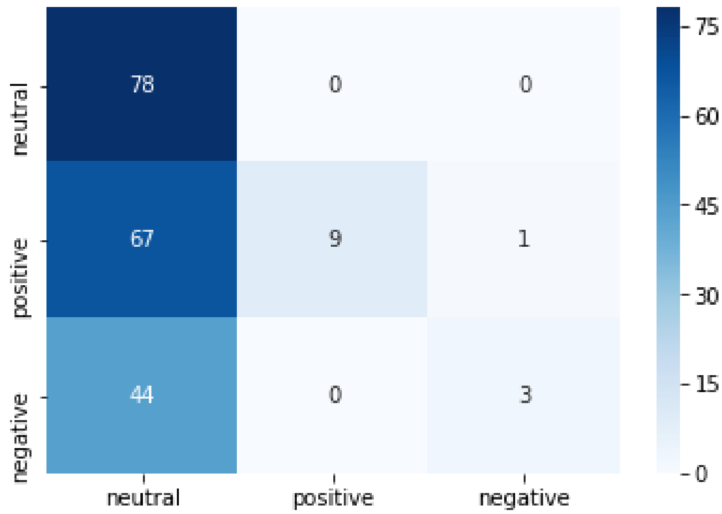
```
There are 313 neutral tweets in the training set.
There are 308 positive tweets in the training set.
There are 188 negative tweets in the training set.
There are 79 neutral tweets in the testing set.
There are 77 positive tweets in the testing set.
There are 47 negative tweets in the testing set.

 Testing precision are:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 0 0 0 0 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0
 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Precision Score: 0.7709616174055828
Recall Score: 0.4433497536945813
f1 Score: 0.32625631649925796
confusion_matrix is:
 [[79  0  0]
 [71  6  0]
 [42  0  5]]
```



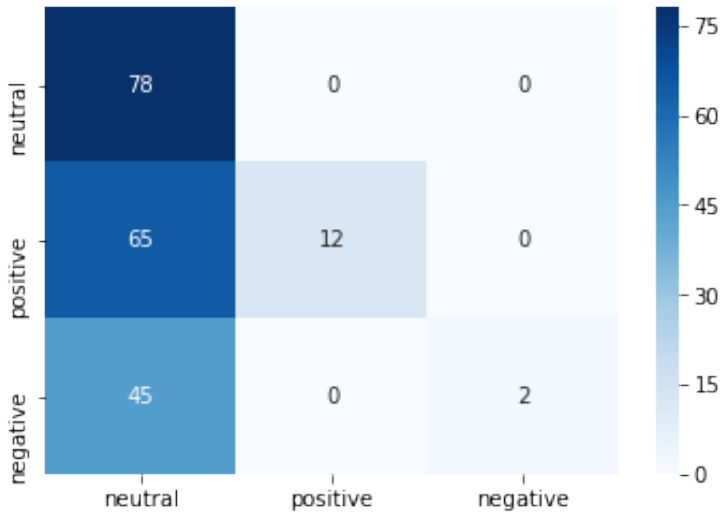

```
 Testing precision are:  [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 2 0 0 0 0 0 0 0 1 0 0 0]

Precision Score: 0.7717999638923994
Recall Score: 0.4482758620689655
f1 Score: 0.3354497354497354
confusion_matrix is:
 [[79  0  0]
 [69  8  0]
 [43  0  4]]
```



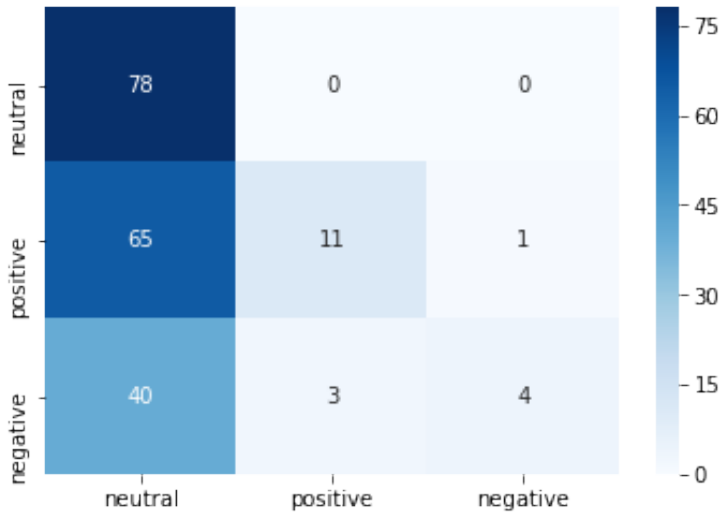

```
 Testing precision are:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Precision Score: 0.7150518623290901
Recall Score: 0.44554455445544555
f1 Score: 0.33276596308088363
confusion_matrix is:
 [[78  0  0]
 [67  9  1]
 [44  0  3]]
```




```
 Testing precision are:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Precision Score: 0.7740678323151464
Recall Score: 0.45544554455445546
f1 Score: 0.3482432699997694
confusion_matrix is:
 [[78  0  0]
 [65 12  0]
 [45  0  2]]
```




```
 Testing precision are:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 2 0 0 1 0 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 0 0
 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]

Precision Score: 0.6502272358383379
Recall Score: 0.4603960396039604
f1 Score: 0.3587467500065656
confusion_matrix is:
 [[78  0  0]
 [65 11  1]
 [40  3  4]]
```

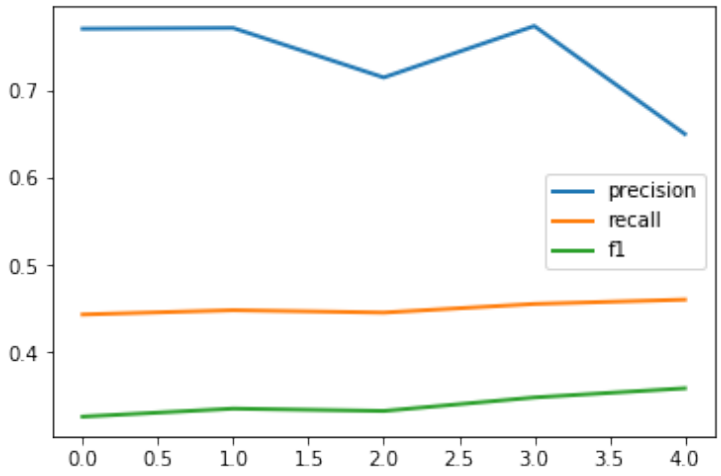


Discuss the result of the k-fold cross validation (using the list of precision, recall, and f1 score)

```
In [49]:  metrics_df = pd.DataFrame(
                {'precision': precision,
                'recall':recall,
                'f1':f1}
            )
          print(metrics_df)
          metrics_df.plot(linewidth=2)
```

```
   precision    recall        f1
0   0.770962  0.443350  0.326256
1   0.771800  0.448276  0.335450
2   0.715052  0.445545  0.332766
3   0.774068  0.455446  0.348243
4   0.650227  0.460396  0.358747
```

```
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5fbe53358>
```



- First we might need to specify that in this multi-label classification task, the calculation of TP, FP, TN, FN could be slightly different compared with binary task, thus the formulas for precision, recall, f1 score are different as well.

  here:

  * True positive: confusion-matrix(x, x)
  * False positive: $sum(\text{confusion-matrix}(:, x), axis = 0) - \text{confusion-matrix}(x, x)$
  * False negative: $sum(\text{confusion-matrix}(x, :), axis = 1) - \text{confusion-matrix}(x, x)$

  since:

  * $precision = \frac{TP}{TP+FP}$
  * $recall = \frac{TP}{TP+FN}$
  * $f1 = \frac{2TP}{2TP+FP+FN}$

  then we could analyze the measurement:

  for instance, in fold 1, confusion matrix is:

  ```
  [[79  0  0]
  [69  8  0]
  [43  0  4]]
  ```

  TP, FP, FN for [neutral, positive, negative]:

  * $TP = [79, 8, 4]$
  * $FP = [112, 0, 0]$
  * $FN = [0, 69, 43])$

  for neutral:

  * $precision = \frac{79}{79+112} = 0.413$

* $recall = \frac{79}{79+0} = 1$
* $f1 = \frac{2 \cdot 79}{2 \cdot 79+112+0} = 0.585$

for positive:

* $precision = \frac{8}{8+0} = 1$
* $recall = \frac{8}{8+69} = 0.104$
* $f1 = \frac{2 \cdot 8}{2 \cdot 8+0+69} = 0.188$

for negative:

* $precision = \frac{4}{4+0} = 1$
* $recall = \frac{4}{4+43} = 0.085$
* $f1 = \frac{2 \cdot 4}{2 \cdot 4+0+43} = 0.156$

Thus, from the table, graph and the calculation above, we can observe that:

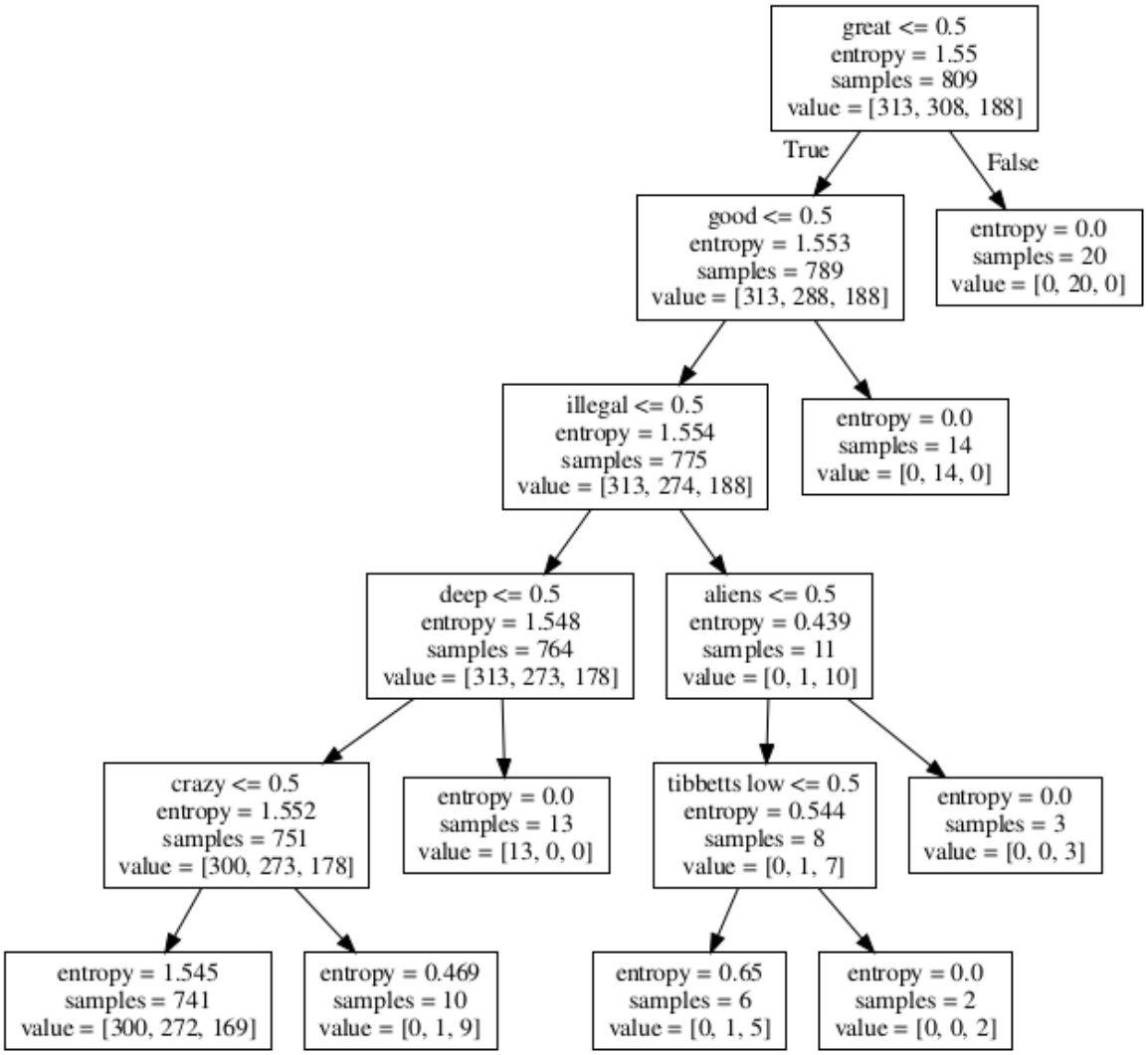
although the total precision score is higher than recall, however from the measurement calculation of each class we could see that it is due to the reason that the precision of *positive* and *negative* are much higher, while their recall are much lower, indicating the fact that:

* FP of class *neutral* it too high
* FN of classes *positive, negative* are too high

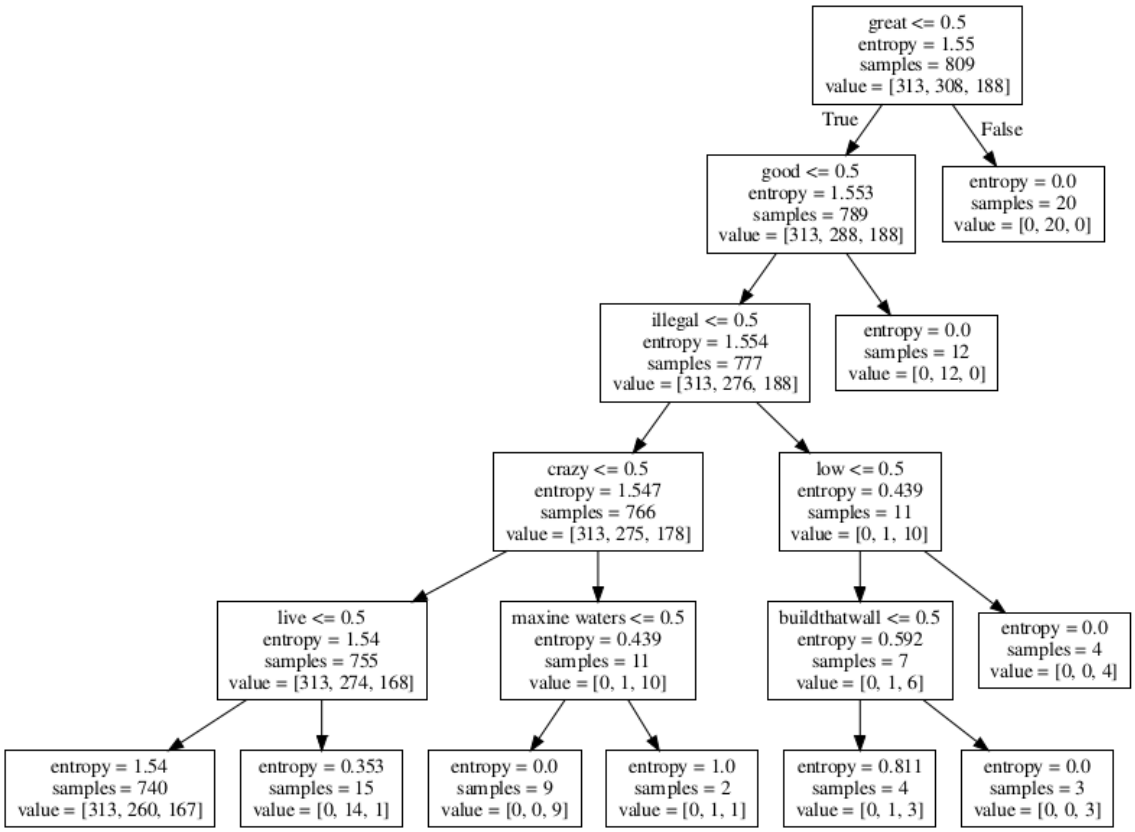
Note: the cause for this problem is probably because the number of *neutral* tweets is larger while the other two, especially *negative* is smaller.

Display 5 DTrees

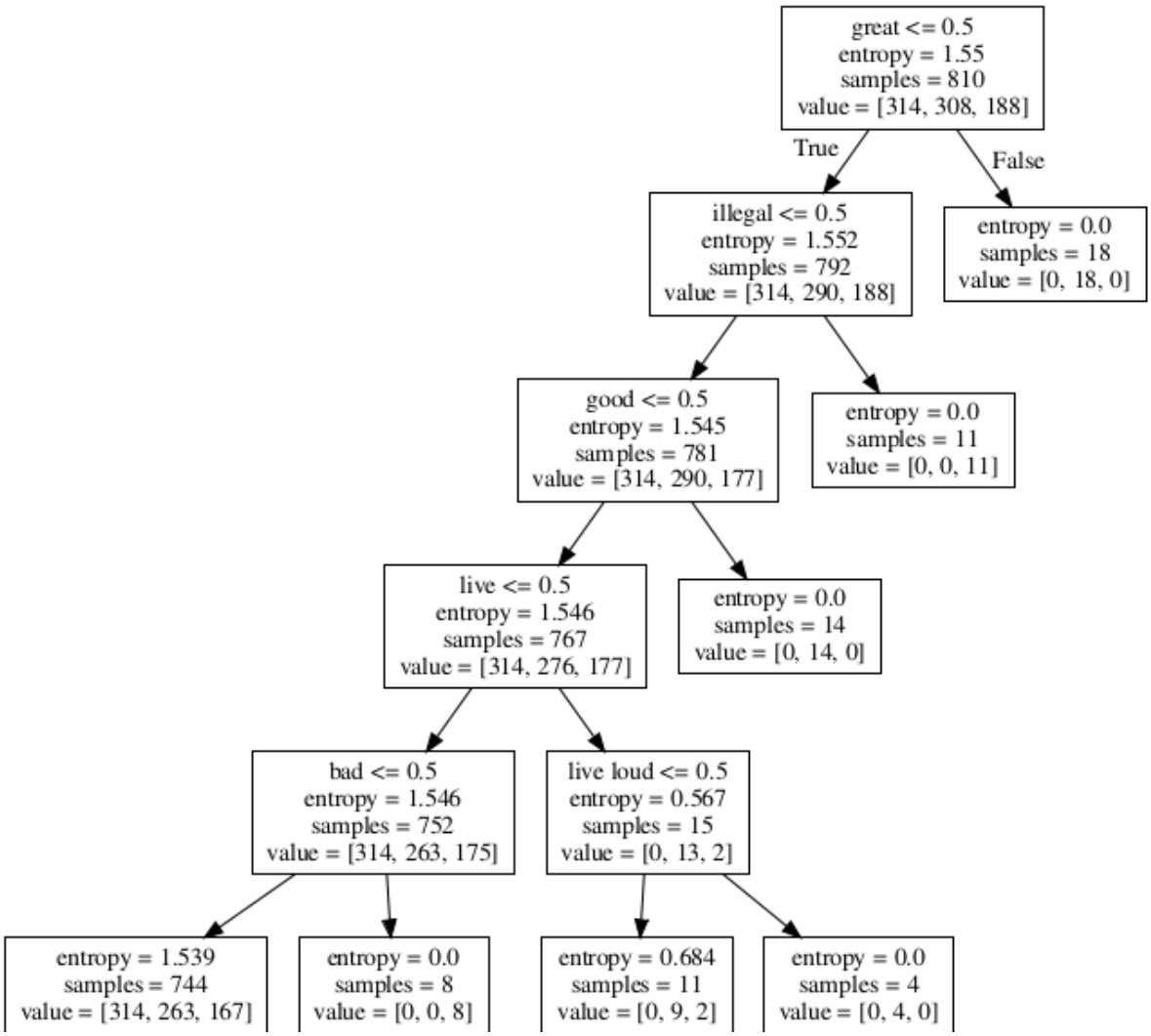
- fold = 1



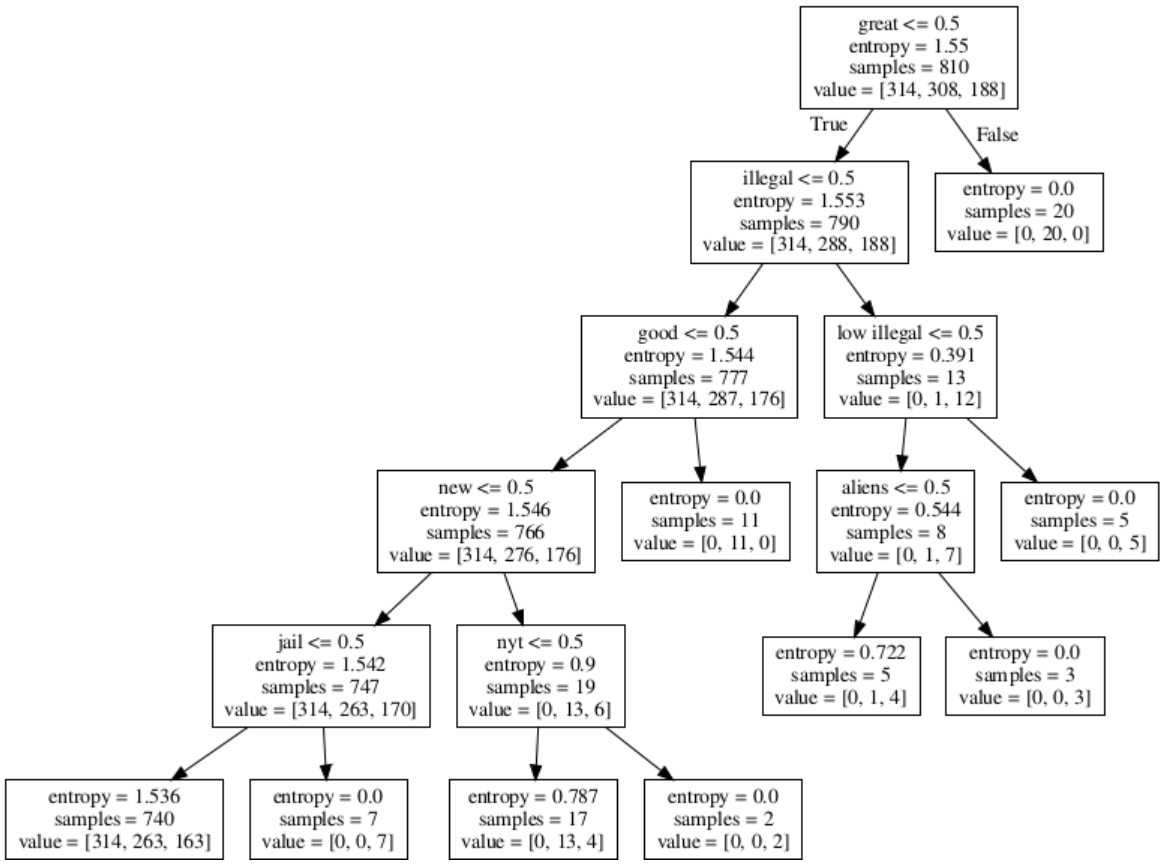
- fold = 2


great <= 0.5
entropy = 1.55
samples = 809
value = [313, 308, 188]
True
False
good <= 0.5
entropy = 1.553
samples = 789
value = [313, 288, 188]
entropy = 0.0
samples = 20
value = [0, 20, 0]
illegal <= 0.5
entropy = 1.554
samples = 777
value = [313, 276, 188]
entropy = 0.0
samples = 12
value = [0, 12, 0]
crazy <= 0.5
entropy = 1.547
samples = 766
value = [313, 275, 178]
low <= 0.5
entropy = 0.439
samples = 11
value = [0, 1, 10]
live <= 0.5
entropy = 1.54
samples = 755
value = [313, 274, 168]
maxine waters <= 0.5
entropy = 0.439
samples = 11
value = [0, 1, 10]
buildthatwall <= 0.5
entropy = 0.592
samples = 7
value = [0, 1, 6]
entropy = 0.0
samples = 4
value = [0, 0, 4]
entropy = 1.54
samples = 740
value = [313, 260, 167]
entropy = 0.353
samples = 15
value = [0, 14, 1]
entropy = 0.0
samples = 9
value = [0, 0, 9]
entropy = 1.0
samples = 2
value = [0, 1, 1]
entropy = 0.811
samples = 4
value = [0, 1, 3]
entropy = 0.0
samples = 3
value = [0, 0, 3]

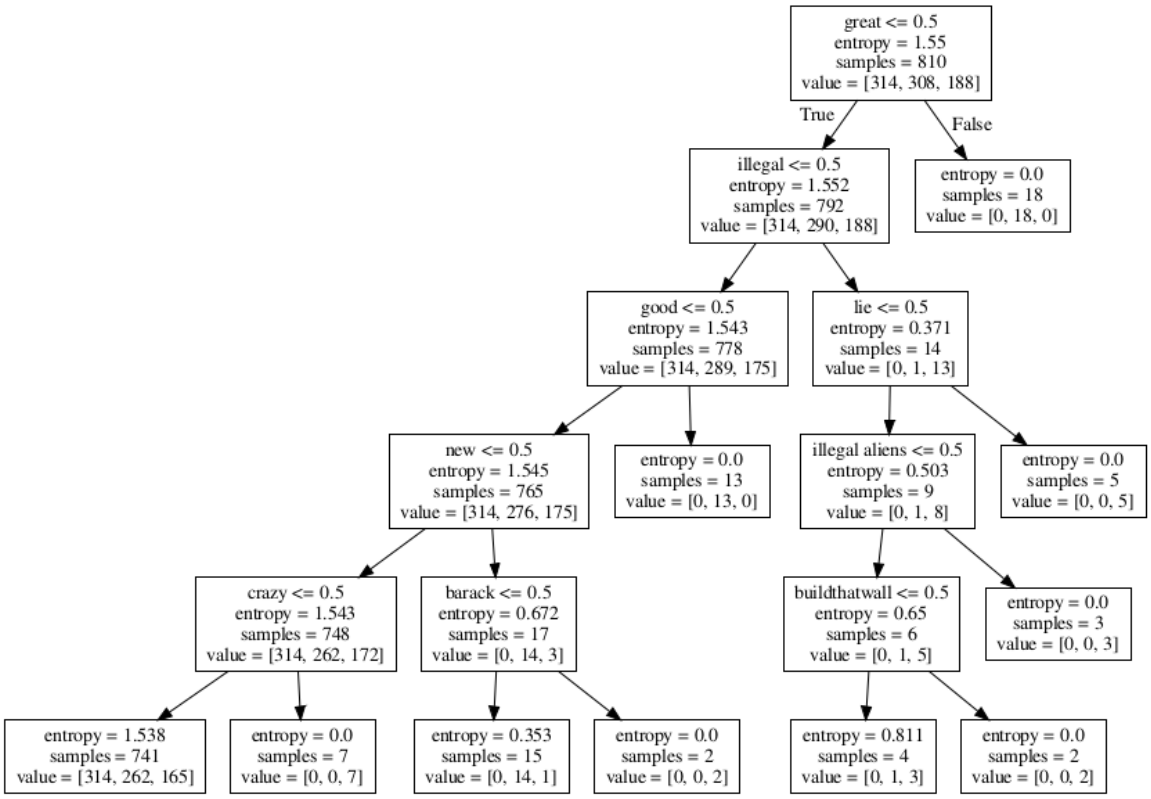
- fold = 3


great <= 0.5
entropy = 1.55
samples = 810
value = [314, 308, 188]
True
False
illegal <= 0.5
entropy = 1.552
samples = 792
value = [314, 290, 188]
entropy = 0.0
samples = 18
value = [0, 18, 0]
good <= 0.5
entropy = 1.545
samples = 781
value = [314, 290, 177]
entropy = 0.0
samples = 11
value = [0, 0, 11]
live <= 0.5
entropy = 1.546
samples = 767
value = [314, 276, 177]
entropy = 0.0
samples = 14
value = [0, 14, 0]
bad <= 0.5
entropy = 1.546
samples = 752
value = [314, 263, 175]
live loud <= 0.5
entropy = 0.567
samples = 15
value = [0, 13, 2]
entropy = 1.539
samples = 744
value = [314, 263, 167]
entropy = 0.0
samples = 8
value = [0, 0, 8]
entropy = 0.684
samples = 11
value = [0, 9, 2]
entropy = 0.0
samples = 4
value = [0, 4, 0]

- fold = 4


great <= 0.5
entropy = 1.55
samples = 810
value = [314, 308, 188]
True
False
illegal <= 0.5
entropy = 1.553
samples = 790
value = [314, 288, 188]
entropy = 0.0
samples = 20
value = [0, 20, 0]
good <= 0.5
entropy = 1.544
samples = 777
value = [314, 287, 176]
low illegal <= 0.5
entropy = 0.391
samples = 13
value = [0, 1, 12]
new <= 0.5
entropy = 1.546
samples = 766
value = [314, 276, 176]
entropy = 0.0
samples = 11
value = [0, 11, 0]
aliens <= 0.5
entropy = 0.544
samples = 8
value = [0, 1, 7]
entropy = 0.0
samples = 5
value = [0, 0, 5]
jail <= 0.5
entropy = 1.542
samples = 747
value = [314, 263, 170]
nyt <= 0.5
entropy = 0.9
samples = 19
value = [0, 13, 6]
entropy = 0.722
samples = 5
value = [0, 1, 4]
entropy = 0.0
samples = 3
value = [0, 0, 3]
entropy = 1.536
samples = 740
value = [314, 263, 163]
entropy = 0.0
samples = 7
value = [0, 0, 7]
entropy = 0.787
samples = 17
value = [0, 13, 4]
entropy = 0.0
samples = 2
value = [0, 0, 2]

- fold = 5



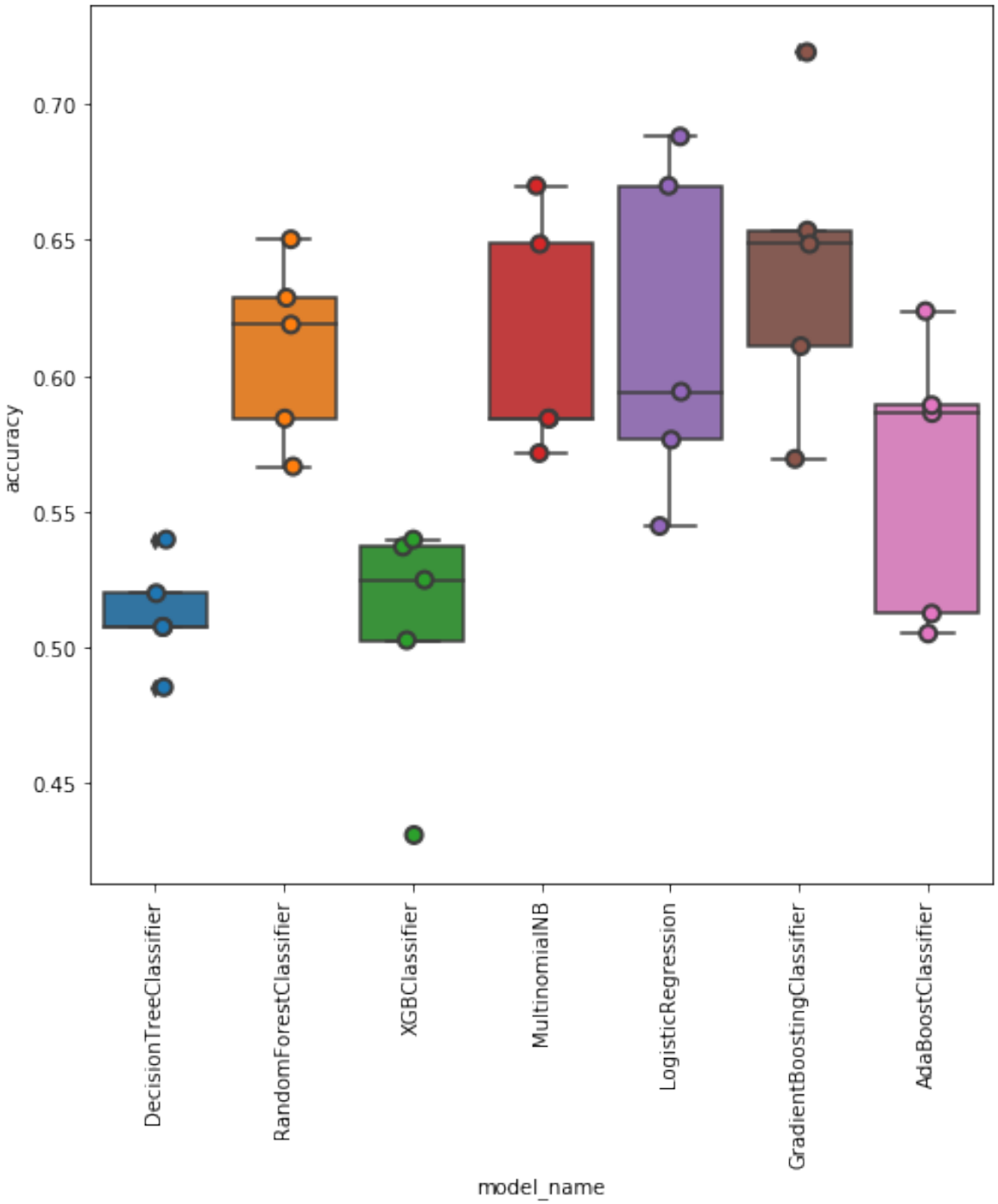
- Decision Trees of different folds share some similar features, for instance:
    * $great \leq 5$ is the root node of all 5 trees
    * $illegal \leq 0.5, good \leq 0.5$ also appears in all 5 trees
    * $crazy \leq 0.5, alien \leq 0.5$ appears in not all 5 but also more than 1 tree

****Explore other possible classifiers for the task**

```
In [68]: vect = CountVectorizer(
                    analyzer="word", ngram_range=([1,2]), tokenizer=nltk.word_tokenize,
                   preprocessor=None, stop_words='english', max_features=3000)
         X_vect = vect.fit_transform(tagged.text).todense()
         y = tagged.sentiment

In [69]: def compare_classifiers(models, X, y):
             CV = 5
             cv_df = pd.DataFrame(index=range(CV * len(models)))
             entries = []
             for model in models:
                 model_name = model.__class__.__name__
                 accuracies = cross_val_score(model, X, y, scoring='accuracy', cv=CV)
                 for fold_idx, accuracy in enumerate(accuracies):
                     entries.append((model_name, fold_idx, accuracy))
```

```
    cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
    plt.figure(figsize=(8,8))
    sns.boxplot(x='model_name', y='accuracy', data=cv_df)
    sns.stripplot(x='model_name', y='accuracy', data=cv_df,
                  size=8, jitter=True, edgecolor="gray", linewidth=2)
    plt.xticks(rotation=90)
    plt.show()
```

```
In [97]: models = [
    DecisionTreeClassifier(criterion = 'entropy',
                                    random_state = 100,
                                    max_depth = 15,
                                    min_samples_leaf = 2),
    RandomForestClassifier(random_state=100,
                                   n_estimators=50,
                                   criterion='entropy',
                                   n_jobs=4),
    XGBoostClassifier(max_depth=8, n_estimators=5),
    MultinomialNB(),
    LogisticRegression(random_state=0),
    GradientBoostingClassifier(),
    AdaBoostClassifier(n_estimators=100, learning_rate=0.1),
]
compare_classifiers(models, X_vect, y)
```


0.70
0.65
0.60
0.55
0.50
0.45
accuracy
DecisionTreeClassifier
RandomForestClassifier
XGBClassifier
MultinomialNB
LogisticRegression
GradientBoostingClassifier
AdaBoostClassifier
model_name