Final Project Report

Introduction

For this project, the task is to use machine learning to analyze tweets from the past and then thus being able to predict whether a tweet is negative and non-negative in the future.

Data

The data we have initially are two csv files named 'compliant1700' and 'noncompliant1700'. In the pre-treatment process, I combined these two files into a new one and labeled them based on which csv file the tweet comes from. Then, additional efforts include lower-casing all the content, use of stop words, and generation of token. The language used is Python. Here, re package is used for regular expression and nltk.corpus package is used for stop words.

```
eng_stopwords = set(stopwords.words('english'))
def clean_text(text):
    text = re.sub(r'[^a-zA-Z]',' ',text) # use space to replace non-letter
    words = text.lower().split() # turn all letters to lowercase
    words = [w for w in words if w not in eng_stopwords]
    return  ' '.join(words)
```

```
data_dup = data.drop_duplicates(subset=['tweet']) # drop drop_duplicates
data_dup['clean_tweet'] = data_dup['tweet'].apply(clean_text)
print(data_dup['clean_tweet'].head(5))
print(len(data_dup['clean_tweet']))
```

After data treatment, I created a training dataset and test dataset in a 7:3 ratio to achieve sampling for future model use.

```
X_train, X_test, y_train, y_test = train_test_split(data_dup['clean_tweet'], data_dup['label'], test_size=0.3, random_state=42)
```

Method

I plan to use two methods to classify the target file. One is CountVectorizer and the other one is TfidfVectorizer. CountVectorizer majorly counts the occurrence of words in each type and is then able to predict the tweet to negative or non-negative based on selection the word within the content.

Convert Text to feature vector--1.CountVectorizer

```
vec = CountVectorizer(
    analyzer='word',
    ngram_range=(1,4),
    max_features=50000)
vec.fit(X_train)
```

TfidVectorizer is a widely used method in text classification and it can convert files into a matrix of TF-IDF features. I assume TfidVectorizer will do better because it not only includes the frequency count but also gives IDF values and Tf-idf scores.

```
tfid_stop_vec = TfidfVectorizer(analyzer='word', stop_words='english')
x_tfid_stop_train = tfid_stop_vec.fit_transform(X_train)
x_tfid_stop_test = tfid_stop_vec.transform(X_test)
print(x_tfid_stop_train.shape[0],x_tfid_stop_train.shape[1])#2372 5820
print(x_tfid_stop_test.shape[0],x_tfid_stop_test.shape[1])# 1017 5820
```

```
2372 5820
1017 5820
```

Model Selection
Models that I plan to use include:

a. Naive Bayes (from sklearn.naive_bayes import MultinomialNB)
b. Random Forest (from sklearn.ensemble import RandomForestClassifier)
c. GradientBoosting (from sklearn.ensemble import GradientBoostingClassifer)

Based on results generated by CountVectorizer the accuracy for them respectively are 73.25%, 69.71% and 69.71%. I also tried to tune the GradientBossting model, such as adjusting n_estimators, max_depth, min_samples_split, min_samples_leaf, but found that there is not obvious better performance.

```
'''
Naive bayes
'''
classfier = MultinomialNB()
classfier.fit(vec.transform(X_train),y_train)
print(classfier.score(vec.transform(X_test),y_test))
```

```
0.7325467059980334
```

```
'''
Random forests
'''
forest = RandomForestClassifier(n_estimators=100)
forest.fit(vec.transform(X_train),y_train)
print(forest.score(vec.transform(X_test),y_test))
```

```
0.6971484759095379
```

```
param = {'max_depth':2, 'eta':1, 'silent':0, 'objective':'binary:logistic' }
num_round =300
bst = xgb.train(param, dtrain, num_round)
```

```
'''
GradientBoosting
'''
clf=ensemble.GradientBoostingClassifier()
clf.fit(vec.transform(X_train),y_train)
print(clf.score(vec.transform(X_test),y_test))
```

```
0.6971484759095379
```

Based on results generated by TfidfVectorizer, the accuracy for them respectively are 73.45%, 70.71% and 67.05%.

```
# Naive bayes
mnb_tfid_stop = MultinomialNB()
mnb_tfid_stop.fit(x_tfid_stop_train, y_train)
mnb_tfid_stop_y_predict = mnb_tfid_stop.predict(x_tfid_stop_test)
print(mnb_tfid_stop.score(x_tfid_stop_test, y_test))
```

```
0.7345132743362832
```

```
'''
Random forests
'''
forest_tfid_stop = RandomForestClassifier(n_estimators=100)
forest_tfid_stop.fit(x_tfid_stop_train,y_train)
# forest_tfid_stop_y_predict = forest_tfid_stop.predict(x_tfid_stop_test)
# print(forest_tfid_stop.score(x_tfid_stop_test,y_test))
print(forest_tfid_stop.score(x_tfid_stop_test,y_test))
```

```
0.7099311701081613
```

```
# GradientBoosting(梯度提升树算法)
clf_tfid_stop = ensemble.GradientBoostingClassifier()
clf_tfid_stop.fit(x_tfid_stop_train,y_train)
print(clf_tfid_stop.score(x_tfid_stop_test,y_test))
```

```
0.6745329400196657
```

In conclusion, Naïve Bayes has the best performance among three models under TfidfVectorizer, and I will use it on the validation dataset.

Predication

First, we would load the validation dataset and remove duplicates:

```
data_pre = pd.read_csv("validation.csv",engine='python',encoding='utf-8')
```

```
data_pre_dup = data_pre.drop_duplicates(subset=['tweet']) # drop drop_duplicates
data_pre_dup['clean_tweet'] = data_pre_dup['tweet'].apply(clean_text)
print(data_pre_dup['clean_tweet'].head(5))
print(len(data_pre_dup['clean_tweet']))
```

Applying the model that has the best performance and export the results to a new csv file:

```
x_tfid_stop_pre = tfid_stop_vec.transform(data_pre_dup['clean_tweet'])
print(x_tfid_stop_pre.shape[0],x_tfid_stop_pre.shape[1])#4550 5820
test_pre = mnb_tfid_stop.predict(x_tfid_stop_pre)
```

```
4550 5820
```

```
# print(data_pre_dup.head(5))
output = pd.DataFrame({'id':data_pre_dup['id'],'label':test_pre,'tweet':data_pre_dup['tweet']})
output['label'][output['label'] == 'complaint'] = 1
output['label'][output['label'] == 'noncomplaint'] = 0
# result of prediction into csv
output.to_csv('mucong_zhou.csv',index=False)
```

the result csv file looks like this:

id	label	tweet
94	0	@monkbent as epic as this fail is (and it is epic) Im still just about to book @united - cheapest and only direct option (sydney<->san fran)
132	1	@AmericanAir your employee "Jeanne" at the O'hare airport priority checkin was just extremely rude to me and my father an executive platinum
162	1	@SouthwestAir Two of the most rude and unprofessional flight attendants I have seen. First and last time flying Southwest.
206	0	@IIJERiiCHOII @VirginAmerica you guys need to get your shit together.
214	1	@IIJERiiCHOII @VirginAmerica man, that really sucks...
332	0	@elyseco that's kinda funny u blowin up the twittersphere but that kinda sucks tho whatever happened w/ @united n all
357	1	@AmericanAir very frustrated that as a EXP, flying on AA Metal I can't put in for a comp. upgrade just because I purchased the ticket on US.

The precision for this predication is calculated by first summing up the second column and then dividing the sum by the total number of rows, which in this case is 3515/4550 = 77.25%