Final Project Report

Introduction

For this project, the objective is to analyze tweets from the past dataset in order to predict the tweets' emotion, i.e., negative or non-negative through using machine learning.

Data

First of all, I used two csv files as training data, which are "noncompliant1700" and "complaint1700" including positive and negative emotions. With combining these two datasets, and marked the labels as 1 and 0 respectively, prepare for model training.

```
if __name__ == "__main__":
    df_1 = pd.read_csv("noncomplaint1700.csv", sep=",")
    df_1['label'] = 1
    df_0 = pd.read_csv("complaint1700.csv", sep=",")
    df_0['label'] = 0
    df = pd.concat([df_1, df_0])

    df = preprocessing(df)
    classify(df)
```

Data cleaning

Then, I preprocess the tweet field in the training data, remove links, remove text independent symbols, and remove Ait, using regex, lower casing all the content, to prevent the interference of irrelevant words, such as @AmericanAir http://t.co/pgvBilTGO3.

```
NEGATIVE_WORDS = set(opinion_lexicon.negative())
HTTP_PATTERN = re.compile('((http|https)\:\/\/)?[a-zA-Z0-9\.\/\?\:@\-_=#]+\.([a-zA-Z]){2,6}([a-zA-Z0-9\.\&\/\?\:@\-_=#])
MENTION_PATTERN = re.compile('(@|#)\w+')
CHARS_PATTERN = re.compile('[^a-zA-z\s]|^b\s+')


def preprocessing(df):
    def handle_links(text):
        text = re.sub(HTTP_PATTERN, ' ', text)
        return text

    def handle_mentions(text):
        text = re.sub(MENTION_PATTERN, ' ', text)
        return text

    def handle_special_chars(text):
        text = re.sub(CHARS_PATTERN, '', text)
        return text

    def process(text):
        text = handle_links(text)
        text = handle_mentions(text)
        text = handle_special_chars(text)
        text = text.lower()
        return text

    df['tweet_'] = df['tweet'].apply(process)
    return df
```

In the third step, I used TFIDF to turn the tweet document into a word frequency matrix for the convenience of model calculation

```
vectorizer = TfidfVectorizer(binary=False, max_features=2500, min_df=5, max_df=0.8, stop_words=stopwords.words('english'))
tfidf_model = vectorizer.fit(df['tweet_'].values)
X = tfidf_model.transform(df['tweet_'].values).toarray()
```

In order to train the model, I divide the data into training set and verification set, the proportion is 8:2

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Method

I used TfidfVectorizer method to classify the target file.

Model

1. **Random Forest**

I used Random forest model in the sklearn toolkit, and used 200 classifiers to make the prediction better, and model precision on positive tweet is 0.76, recall is 0.68

```
text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
predictions = text_classifier.predict(X_test)

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.69	0.77	0.73	325
1	0.76	0.68	0.72	355
micro avg	0.72	0.72	0.72	680
macro avg	0.73	0.72	0.72	680
weighted avg	0.73	0.72	0.72	680

2. **Support Machine Vector (SVM)**

Based on the SVM model that I used to test, the accuracy is 0.72.

```
text_classifier_2 = svm.SVC(kernel='linear',C=10)
clf.fit(df[['X1','X2']], df['class'])
    text_classifier_2.fit(X_train, y_train)
    predictions = text_classifier_2.predict(X_test)
```

3. Naive Bayes

Based on Naive Bayes Model, the accuracy is about 0.74.

```
text_classifier_3 = MultinomialNB()
text_classifier_3.fit(df[['X1','X2']], df['class'])
    text_classifier_3.fit(X_train, y_train)
    predictions = text_classifier_3.predict(X_test)
```

Based on these three models' accuracy, I chose the Random Forest model since it has the highest accuracy.

Prediction

I used opinion lexicon in nltk toolkit to select negative emotion words. If a tweet contains negative emotional words, it will be regarded as negative, and the rest as non-negative. My temp file has 4555 lines, of which non negative has 1482 lines, I used the trained model to predict the non-negative tweets, and 80% of the labels were 1 (accuracy)

Calculation of final accuracy: Sum of the second column divided by the total number of rows which is 1185/1482 = 0.80.

```
def output(tfidf_model, text_classifier):
    df = pd.read_excel("temp.xlsx", sep=",")
    df = preprocessing(df)
    m = df['tweet_'].apply(is_non_negative)
    df = df[m].reset_index()
    X = tfidf_model.transform(df['tweet_'].values).toarray()

    predictions = text_classifier.predict(X)
    df['label'] = pd.Series(predictions)
    df = df[['id', 'label', 'tweet']]
    def replace_quotation(x):
        x = x.replace("'", '').replace('"', '')
        return x
    df['tweet'] = df['tweet'].apply(replace_quotation)
    df.to_csv("result.csv", sep=",", index=False, quoting=csv.QUOTE_NONE, quotechar="",  escapechar="\\")
```