CMT307 Part2
C1966881
Shenglei Fang

Sentiment analysis --- IMDB

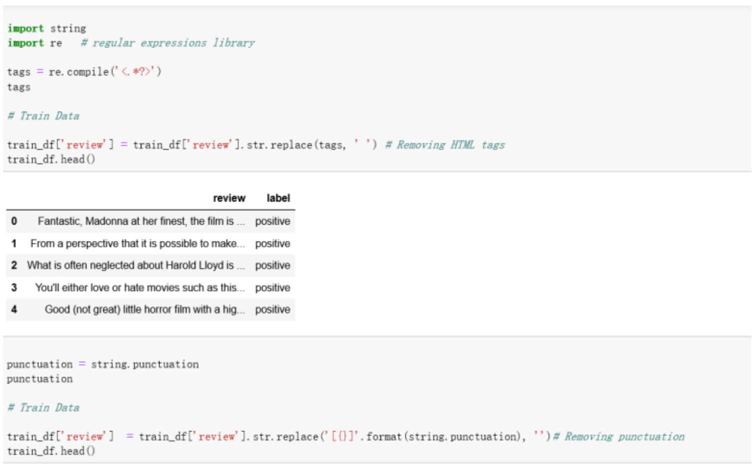
Introduction

As is known to all, when a new movie is released, a large number of audiences tend to browse the movie evaluation websites to get the movie evaluation and decide whether to see the movie based on it. When the audience likes the movie, they will post positive feedback on the website; otherwise, they will post negative reviews (Topal and Ozsoyoglu, 2016). Sentiment analysis is to extract opinions and subjective knowledge from user-generated text content (Sharma and Dey, 2012). This essay is to make sentiment analysis through the training model for movie reviews from IMDb, the world's most popular content source for movie, TV and celebrity content. First, python is used for data preprocessing, and the data is cleaned to remove the interference vocabulary. Then, Build models for training, and finally compare the accuracy of different models.

Preprocessing

Data preprocessing is an important step in the data mining process. There are some problem about sentiment analysis context (Nassr et al., 2019). Reviews contains a lot of meaningless data that can interfere with results, such as urls, punctuation and stopwords. Meanwhile，some nouns and verbs without emotional tendency would also interfere with the accuracy of the model. Therefore, these interference data should be removed in the process of data preprocessing.

Since the train and test divided the positive and negative data into two TXT files, it was necessary to combine the two files into a CSV file. At the same time, it was necessary to label each review with the corresponding positive or negative labels for the subsequent modeling. There are 15000 reviews in Train and 5000 in Test. After merging the files, we can start preprocessing the data. First, use the regular expression to match the HTML in the review, removing the match from the text. Punctuation is also a meaningless field and should be cleaned up. Then, import the preliminary cleaned data into the newly created csv file.

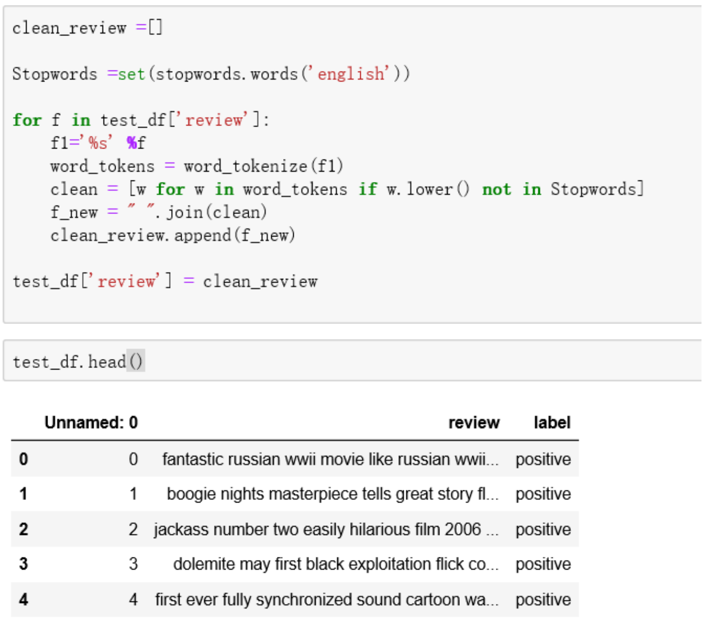


```python
import string
import re   # regular expressions library

tags = re.compile('<.*?>')
tags

# Train Data

train_df['review'] = train_df['review'].str.replace(tags, ' ') # Removing HTML tags
train_df.head()
```

```
                                         review     label
0     Fantastic, Madonna at her finest, the film is ...  positive
1   From a perspective that it is possible to make...  positive
2  What is often neglected about Harold Lloyd is ...  positive
3    You'll either love or hate movies such as this...  positive
4       Good (not great) little horror film with a hig...  positive
```

```python
punctuation = string.punctuation
punctuation

# Train Data

train_df['review']  = train_df['review'].str.replace('[{}]'.format(string.punctuation), '')# Removing punctuation
train_df.head()
```

Next step, stop words should be cleaned up. Stop words are words, which are filtered out in preprocessing of natural language data (Rajaraman and Ullman, 2011). Stop words are generally the most common words in a language. I called the NLTK library, there is stopwords list in this library. Read each vocabulary in reviews one by one, match the vocabulary in the Stopwords list, and output the non-listed words to a new csv file.

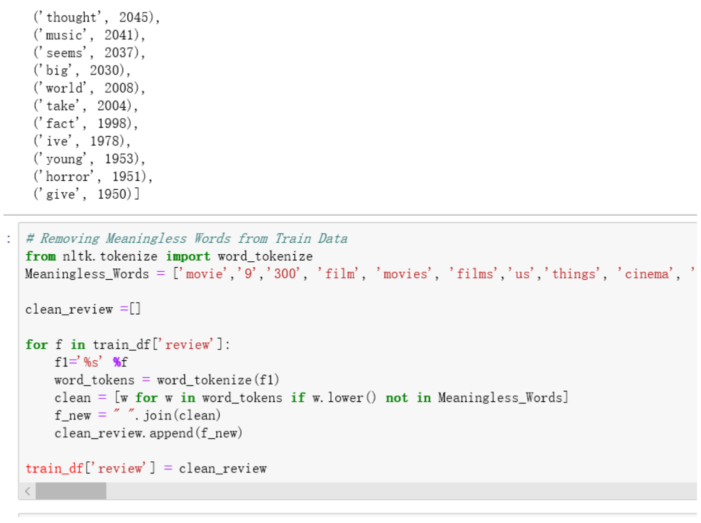


```python
clean_review =[]

Stopwords =set(stopwords.words('english'))

for f in test_df['review']:
    f1='%s' %f
    word_tokens = word_tokenize(f1)
    clean = [w for w in word_tokens if w.lower() not in Stopwords]
    f_new = " ".join(clean)
    clean_review.append(f_new)

test_df['review'] = clean_review
```

```python
test_df.head()
```

```
   Unnamed: 0                                           review     label
0           0  fantastic russian wwii movie like russian wwii...  positive
1           1    boogie nights masterpiece tells great story fl...  positive
2           2  jackass number two easily hilarious film 2006 ...  positive
3           3     dolemite may first black exploitation flick co...  positive
4           4  first ever fully synchronized sound cartoon wa...  positive
```

After the above two steps of cleaning, the data preprocessing has been basically completed, but I found that there are still a lot of meaningless high-frequency words in the text. These emotionless tendencies will interfere with the accuracy of the model, so it should also be filtered. First find these high-frequency words, use word frequency counting, count the 200 words that appear most frequently in reviews, filter out meaningless words by myself, and build a Meaningless_Words list. Finally perform the same steps as cleaning stop words to delete meaningless words and output the final csv file.



```
('thought', 2045),
('music', 2041),
('seems', 2037),
('big', 2030),
('world', 2008),
('take', 2004),
('fact', 1998),
('ive', 1978),
('young', 1953),
('horror', 1951),
('give', 1950)]
```

```
# Removing Meaningless Words from Train Data
from nltk.tokenize import word_tokenize
Meaningless_Words = ['movie','9','300', 'film', 'movies', 'films','us','things', 'cinema', '

clean_review =[]

for f in train_df['review']:
    f1='%s' %f
    word_tokens = word_tokenize(f1)
    clean = [w for w in word_tokens if w.lower() not in Meaningless_Words]
    f_new = " ".join(clean)
    clean_review.append(f_new)

train_df['review'] = clean_review
```

Examine the results of data preprocessing, re-output the top ten high-frequency words, and find that there are basically no meaningless words. The data preprocessing is completed.

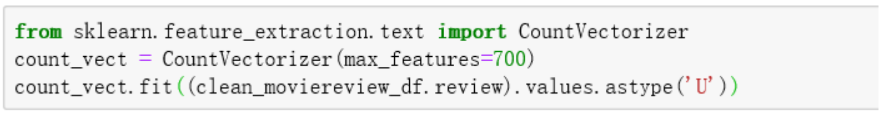
Model

After the data is cleaned, the model is built. Firstly, the characteristics are selected. In this paper, frequency of words, frequency of adjectives only and frequency of adjectives and verbs are selected to establish the model. In terms of model selection, I tried different models and compared the output results to determine which model had the highest accuracy.

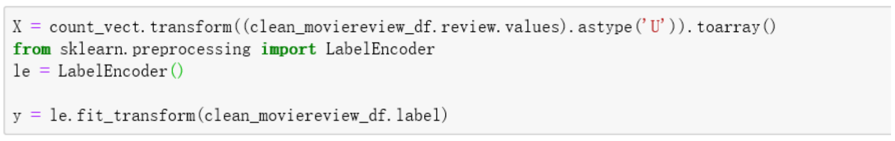
1. frequency of words:

There are four models built, including Logistic Regression, Decision Tree, Random Forest and SVM.

Call the sklearn library and statistical word frequency.

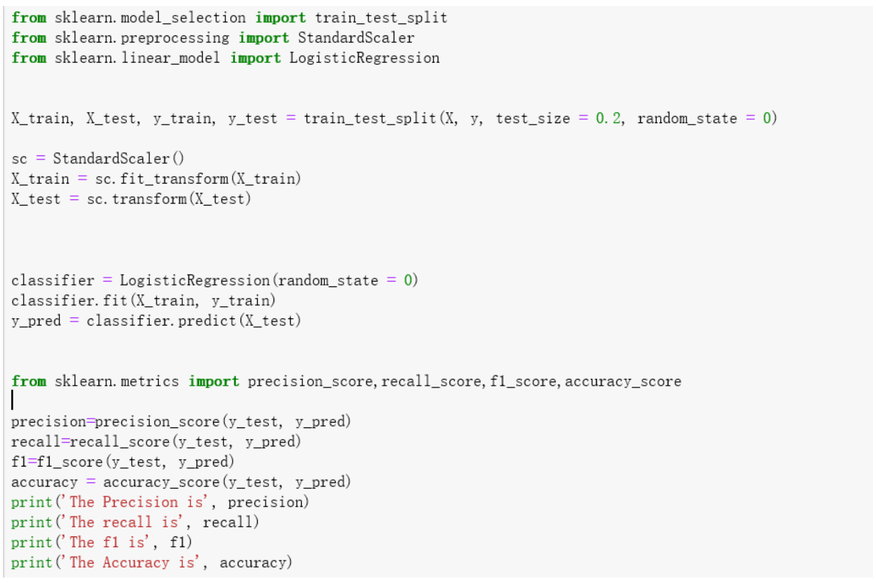


```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(max_features=700)
count_vect.fit((clean_moviereview_df.review).values.astype('U'))
```



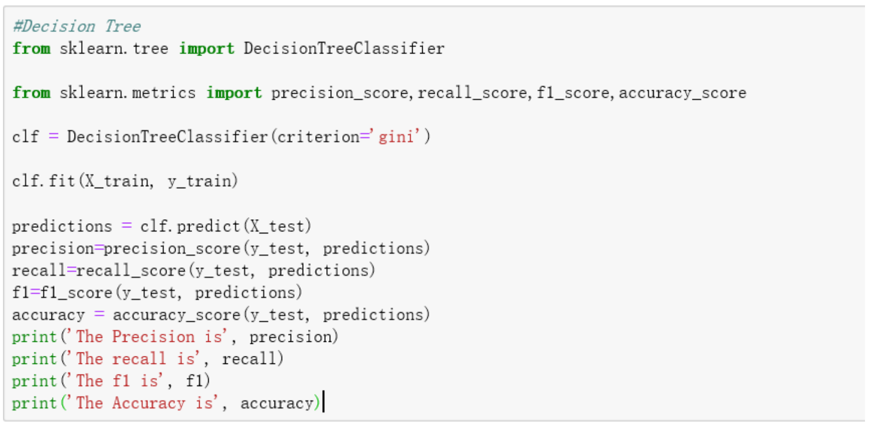
```
X = count_vect.transform((clean_moviereview_df.review.values).astype('U')).toarray()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

y = le.fit_transform(clean_moviereview_df.label)
```

Logistic Regression:



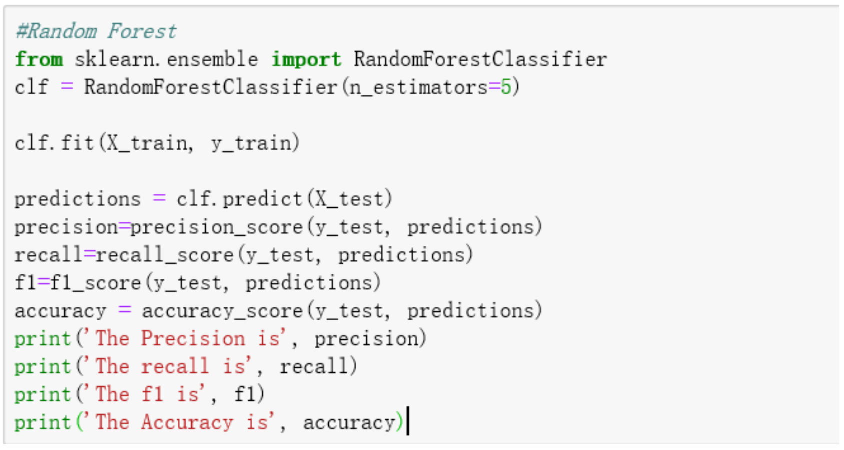
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression


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

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)



classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)


from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score

precision=precision_score(y_test, y_pred)
recall=recall_score(y_test, y_pred)
f1=f1_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print('The Precision is', precision)
print('The recall is', recall)
print('The f1 is', f1)
print('The Accuracy is', accuracy)
```

Decision Tree:



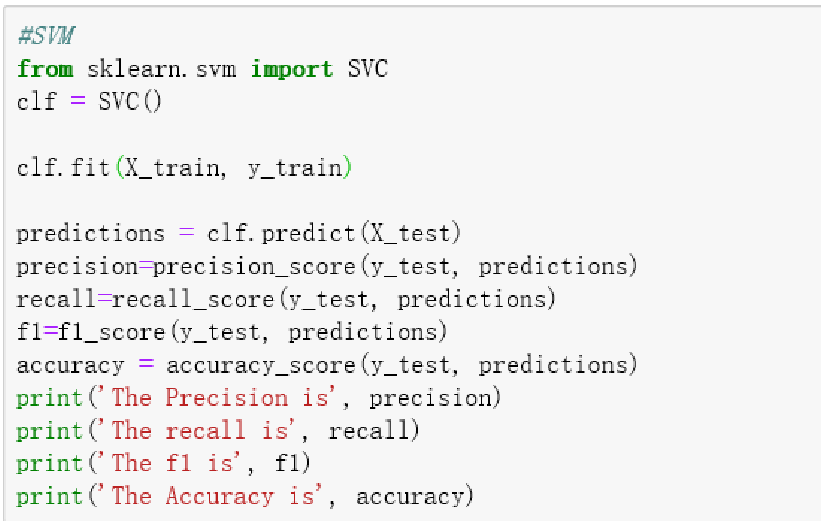
```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score

clf = DecisionTreeClassifier(criterion='gini')

clf.fit(X_train, y_train)

predictions = clf.predict(X_test)
precision=precision_score(y_test, predictions)
recall=recall_score(y_test, predictions)
f1=f1_score(y_test, predictions)
accuracy = accuracy_score(y_test, predictions)
print('The Precision is', precision)
print('The recall is', recall)
print('The f1 is', f1)
print('The Accuracy is', accuracy)
```

Random Forest:



```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=5)

clf.fit(X_train, y_train)

predictions = clf.predict(X_test)
precision=precision_score(y_test, predictions)
recall=recall_score(y_test, predictions)
f1=f1_score(y_test, predictions)
accuracy = accuracy_score(y_test, predictions)
print('The Precision is', precision)
print('The recall is', recall)
print('The f1 is', f1)
print('The Accuracy is', accuracy)
```

SVM:



```
#SVM
from sklearn.svm import SVC
clf = SVC()

clf.fit(X_train, y_train)

predictions = clf.predict(X_test)
precision=precision_score(y_test, predictions)
recall=recall_score(y_test, predictions)
f1=f1_score(y_test, predictions)
accuracy = accuracy_score(y_test, predictions)
print('The Precision is', precision)
print('The recall is', recall)
print('The f1 is', f1)
print('The Accuracy is', accuracy)
```

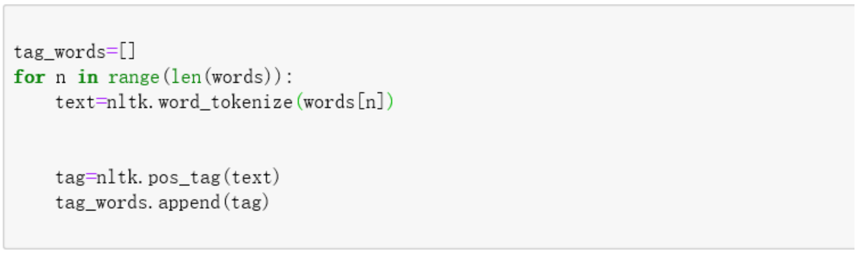
Frequency of words	precision	recall	F-measure	accuracy
Logistic Regression	0.83237	0.86133	0.8466048	0.843
Decision Tree	0.71571	0.71322	0.7144635	0.71325
Random Forest	0.7652	0.76938	0.7672862	0.76525
SVM	0.82292	0.87078	0.8461724	0.84075

We can get the result from above accuracy scores, it has been observed that SVM (0.84075) and Logistic Regression (0.843) have almost same accuracies and have higher accuracies than Decision Tree and Random Forest.

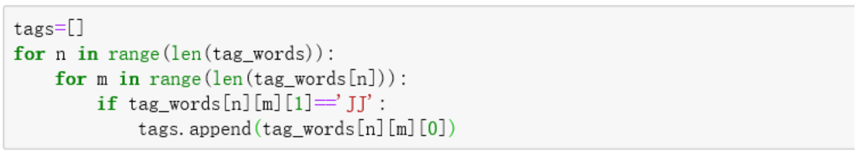
2. frequency of adjectives only:

There are three models built, including Decision Tree, Random Forest and SVM.

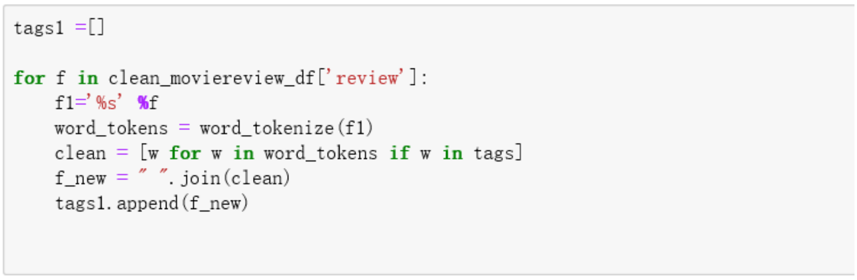
First tag the lexical parts of speech in reviews. Then, select adjectives to write into the text and perform word frequency statistics.



```
tag_words=[]
for n in range(len(words)):
    text=nltk.word_tokenize(words[n])


    tag=nltk.pos_tag(text)
    tag_words.append(tag)
```



```
tags=[]
for n in range(len(tag_words)):
    for m in range(len(tag_words[n])):
        if tag_words[n][m][1]=='JJ':
            tags.append(tag_words[n][m][0])
```



```
tags1 =[]

for f in clean_moviereview_df['review']:
    f1='%s' %f
    word_tokens = word_tokenize(f1)
    clean = [w for w in word_tokens if w in tags]
    f_new = " ".join(clean)
    tags1.append(f_new)
```

Frequency of adjectives	precision	recall	F-measure	accuracy
Decision Tree	0.7194	0.70974	0.7145359	0.71475
Random Forest	0.7653	0.75199	0.7585861	0.75925
SVM	0.79776	0.88618	0.8396515	0.82975

The SVM model has the highest accuracy (0.82975) among three models.

3. frequency adjectives and verbs:

There are three models built, including Decision Tree, Random Forest and SVM.

Frequency of adjectives and verbs	precision	recall	F-measure	accuracy
Decision Tree	0.7163193	0.7002982	0.7082181	0.70975
Random Forest	0.7511177	0.7514911	0.7513043	0.74975
SVM	0.7977629	0.8861829	0.8396515	0.82975

The SVM model has the highest accuracy (0.82975) among three models.

Conclusion

By comparison, it is found that the frequency of words can get the highest accuracy. The accuracy of SVM is higher than that of decision tree and random forest in three different features. Word frequency as a feature and SVM as a model is one of the best choices in all the above cases.

Defects of the algorithm:

-When meaningless words are counted, the statistics are incomplete. There is no data set like stop words, so you could try to delete all nouns and then calculate the word frequency. The deleted numbers are omitted in the preprocessing. These meaningless words may Interfering with the results.

- After part-of-speech tagging, it takes a lot of time to match the adjective set with the vocabulary of the data set one by one. Perhaps a better algorithm can be used to count the vocabulary of the labeled adjectives.

-During data preprocessing, stemming and lemmatization should be applied. Stemming and Lemmatization are Text Normalization techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing. Stemming and Lemmatization can eliminate the interference caused by some derived words, and can better remove some interference when further processing noise words.

References

Nassr, Z., Sael, N. and Benabbou, F. et al. 2019. A comparative study of sentiment analysis approaches. *In Proceedings of the 4th International Conference on Smart City Applications* p. 91. ACM.

Rajaraman, A., and Ullman, J. D. 2011. *Mining of massive datasets*. Cambridge University Press.

Sharma, A. and Dey, S. 2012. A comparative study of feature selection and machine learning techniques for sentiment analysis. *In Proceedings of the 2012 ACM research in applied computation symposium* pp. 1-7. ACM.

Topal, K. and Ozsoyoglu, G. 2016. Movie review analysis: Emotion analysis of IMDb movie reviews. *In Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* pp. 1170-1176. IEEE Press