Text classification on movie reviews

I have used NLTK Naïve Bays and Random Forest model in this task and found that random forest has performed better. It has given me 85% accuracy, Precision, Recall and F1-Score. I have also used confusion matrix to see where this model went wrong.

Now let see what all I have done-

Dataset and its Collection:

We will be using movie reviews corpus, which contains 2000 movie reviews. 1000 among them are classified as positive and another 1000 are classified as negative review.

This corpus is already available in NLTK library and can easily imported by using below command-

```
from nltk.corpus import movie_reviews
```

Below are few commands to work with this movie corpus:

movie_reviews.categories() → Tells you sentiment category (Positive, Negative)

movie_reviews.fileids('pos') → Give you all the fileid containing positive reviews

movie_reviews.words() → Tells you all the words used in corpus

movie_reviews.words(fileid) → Tell you words used in a review under specified fileid

Data Preprocessing:

Reviews in this corpus are not cleaned. It contains many stop words and symbols. We will have to remove them for better model performance. Also words like boy and boys should be treated in the same way. To cater this case we will apply stemming on each word of corpus. Eventually we will shuffle whole dataset of reviews randomly so that when we are dividing them into train and test we have more chances of having pos and neg reviews in equal proportion in both train and test dataset.

Below is the picture of an un-cleaned review.

```
print(documents[0])
```

```
(['plot', ':', 'two', 'teen', 'couples', 'go', 'to', 'a', 'church', 'party', ',', 'drink', 'and', 'then', 'drive', '.', 'they',
'get', 'into', 'an', 'accident', '.', 'one', 'of', 'the', 'guys', 'dies', ',', 'but', 'his', 'girlfriend', 'continues', 'to',
'see', 'him', 'in', 'her', 'life', ',', 'and', 'has', 'nightmares', '.', 'what', "'", 's', 'the', 'deal', '?', 'watch', 'the',
'movie', 'and', '"', 'sorta', '"', 'find', 'out', '.', '.', '.', 'critique', ':', 'a', 'mind', '-', 'fuck', 'movie', 'for', 'th
e', 'teen', 'generation', 'that', 'touches', 'on', 'a', 'very', 'cool', 'idea', ',', 'but', 'presents', 'it', 'in', 'a', 'ver
y', 'bad', 'package', '.', 'which', 'is', 'what', 'makes', 'this', 'review', 'an', 'even', 'harder', 'one', 'to', 'write', ',',
'since', 'i', 'generally', 'applaud', 'films', 'which', 'attempt', 'to', 'break', 'the', 'mold', ',', 'mess', 'with', 'your',
'head', 'and', 'such', '(', 'lost', 'highway', '&', 'memento', ')', ',', 'but', 'there', 'are', 'good', 'and', 'bad', 'ways',
'of', 'making', 'all', 'types', 'of', 'films', ',', 'and', 'these', 'folks', 'just', 'didn', "'", 't', 'snag', 'this', 'one',
'correctly', '.', 'they', 'seem', 'to', 'have', 'taken', 'this', 'pretty', 'neat', 'concept', ',', 'but', 'executed', 'it', 'te
rribly', '.', 'so', 'what', 'are', 'the', 'problems', 'with', 'the', 'movie', '?', 'well', ',', 'its', 'main', 'problem', 'is',
'that', 'it', "'", 's', 'simply', 'too', 'jumbled', '.', 'it', 'starts', 'off', '"', 'normal', '"', 'but', 'then', 'downshift
s', 'into', 'this', '"', 'fantasy', '"', 'world', 'in', 'which', 'you', ',', 'as', 'an', 'audience', 'member', ',', 'have', 'n
o', 'idea', 'what', "'", 's', 'going', 'on', '.', 'there', 'are', 'dreams', ',', 'there', 'are', 'characters', 'coming', 'bac
k', 'from', 'the', 'dead', ',', 'there', 'are', 'others', 'who', 'look', 'like', 'the', 'dead', ',', 'there', 'are', 'strange',
'apparitions', ',', 'there', 'are', 'disappearances', ',', 'there', 'are', 'a', 'looooot', 'of', 'chase', 'scenes', ',', 'ther
e', 'are', 'tons', 'of', 'weird', 'things', 'that', 'happen', ',', 'and', 'most', 'of', 'it', 'is', 'simply', 'not', 'explaine
d', '.', 'now', 'i', 'personally', 'don', "'", 't', 'mind', 'trying', 'to', 'unravel', 'a', 'film', 'every', 'now', 'and', 'the
n', ',', 'but', 'when', 'all', 'it', 'does', 'is', 'give', 'me', 'the', 'same', 'clue', 'over', 'and', 'over', 'again', ',',
'i', 'get', 'kind', 'of', 'fed', 'up', 'after', 'a', 'while', ',', 'which', 'is', 'this', 'film', "'", 's', 'biggest', 'proble
m', '.', 'it', "'", 's', 'obviously', 'got', 'this', 'big', 'secret', 'to', 'hide', ',', 'but', 'it', 'seems', 'to', 'want', 't
```

After Pre-processing we have got below result:

Below is the picture of cleaned review

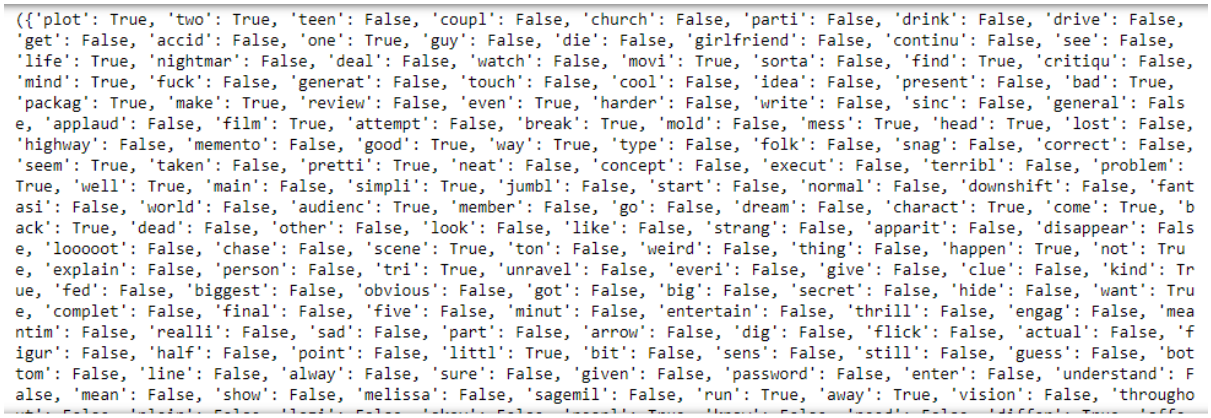
```
(['plot', 'two', 'teen', 'coupl', 'church', 'parti', 'drink', 'drive', 'get', 'accid', 'one', 'guy', 'die', 'girlfriend', 'cont
inu', 'see', 'life', 'nightmar', 'deal', 'watch', 'movi', 'sorta', 'find', 'critiqu', 'mind', 'fuck', 'movi', 'teen', 'genera
t', 'touch', 'cool', 'idea', 'present', 'bad', 'packag', 'make', 'review', 'even', 'harder', 'one', 'write', 'sinc', 'general',
'applaud', 'film', 'attempt', 'break', 'mold', 'mess', 'head', 'lost', 'highway', 'memento', 'good', 'bad', 'way', 'make', 'typ
e', 'film', 'folk', 'snag', 'one', 'correct', 'seem', 'taken', 'pretti', 'neat', 'concept', 'execut', 'terribl', 'problem', 'mo
vi', 'well', 'main', 'problem', 'simpli', 'jumbl', 'start', 'normal', 'downshift', 'fantasi', 'world', 'audienc', 'member', 'id
ea', 'go', 'dream', 'charact', 'come', 'back', 'dead', 'other', 'look', 'like', 'dead', 'strang', 'apparit', 'disappear', 'looo
oot', 'chase', 'scene', 'ton', 'weird', 'thing', 'happen', 'simpli', 'not', 'explain', 'person', 'mind', 'tri', 'unravel', 'fil
m', 'everi', 'give', 'clue', 'get', 'kind', 'fed', 'film', 'biggest', 'problem', 'obvious', 'got', 'big', 'secret', 'hide', 'se
em', 'want', 'hide', 'complet', 'final', 'five', 'minut', 'make', 'thing', 'entertain', 'thrill', 'even', 'engag', 'meantim',
'not', 'realli', 'sad', 'part', 'arrow', 'dig', 'flick', 'like', 'actual', 'figur', 'half', 'way', 'point', 'strang', 'start',
'make', 'littl', 'bit', 'sens', 'still', 'make', 'film', 'entertain', 'guess', 'bottom', 'line', 'movi', 'like', 'alway', 'mak
e', 'sure', 'audienc', 'even', 'given', 'secret', 'password', 'enter', 'world', 'understand', 'mean', 'show', 'melissa', 'sagem
il', 'run', 'away', 'vision', 'minut', 'throughout', 'movi', 'plain', 'lazi', 'okay', 'get', 'peopl', 'chase', 'know', 'reall
i', 'need', 'see', 'give', 'differ', 'scene', 'offer', 'insight', 'strang', 'go', 'movi', 'appar', 'studio', 'took', 'film', 'a
way', 'director', 'chop', 'show', 'might', 'pretti', 'decent', 'teen', 'mind', 'fuck', 'movi', 'somewher', 'guess', 'suit', 'de
cid', 'turn', 'music', 'video', 'littl', 'edg', 'would', 'make', 'sens', 'actor', 'pretti', 'good', 'part', 'although', 'wes',
'bentley', 'seem', 'play', 'exact', 'charact', 'american', 'beauti', 'new', 'neighborhood', 'biggest', 'kudo', 'sagemil', 'hol
d', 'throughout', 'entir', 'film', 'actual', 'feel', 'charact', 'unravel', 'overal', 'film', 'stick', 'entertain', 'confus', 'r
```

Data Modelling for NLTK Naïve base classifier

Obviously before feeding data into classifier we will have to format it in a way which classifier can understand. To do that, I have created many variables, but following two are the most important and worth understanding.

Word_Feature: We will create a list of 8000 most frequent words in the corpus. Obviously, we will not be including stop words and special symbol in it and we will only be considering those words which are greater than 2 in length. This will give us a list of relevant words in the corpus.

Featureset: We will create a feature set for each review using Word_Feature. We will check which all words of Word_Feature exist in review as well and assign True to it if it exist otherwise it will be assigned False. Same process we will repeat for all reviews in our dataset. So eventually the size of Feature set obtained by this process will be N x M (N is total no of reviews and M is length of Word_Feature).



Train Test Split :

We will divide the Featuresets in train and test dataset in ratio of 9:1. Note that as we have already shuffled out dataset in the start so we will not be shuffling it again.

devide training and testing dataset in 90:10 ratio

```
train_length=int(len(featuresets)*.90)
```

```
training_set = featuresets[:train_length]
testing_set = featuresets[train_length:]
```

```
print(len(training_set))
print(len(testing_set))
```

```
1800
200
```

Separate ground truth from test_data set:

We will have to separate ground truth labels from test dataset so that once the model is built, we can predict the test reviews sentiments and match them against ground truth labels to verify model performance.

Classification Models:
we will be applying NLTK naïve bays classifier and Scikit learn Random Forest classifier both one by one and check which one performs better

NLTK Naïve bays classifier

Frist will be applying NLTK naïve bays classifier. NLTK naïve bays expect data to be in format that we have in our train and test dataset created above. We will start training it by feeding training set using below command

classifier = nltk.NaiveBayesClassifier.train(training_set)

Once it is trained, we will test it on our test dataset by predicting sentiments for each test review.
Below command can be run to get the predicted lables-
tested_label=classifier.classify_many(testing_set_content)

Performance Evaluation:

I have used below performance metric to verify the performance of the model. I have also mentioned the score that I have received against each metric:

Accuracy	82.50%
Precision	82.50%
Recall	82.50%
F1-Score	82.50%

Confusion Matrix
```
[[88 16]
 [19 77]]
```

Note : These all metrics are available in scikit learn, so we can easily import and use them.

Feature Importance Using Naïve Bays

If you want to check most informative feature in featureset, just type in below command.

classifier.show_most_informative_features(20).

Above command will give top 20 most informative feature in featureset

Important Features as per this model

```
classifier.show_most_informative_features(20)
```

```
Most Informative Features
                  ideolog = True              pos : neg    =     15.9 : 1.0
                     plod = True              neg : pos    =     14.8 : 1.0
                   seagal = True              neg : pos    =     14.8 : 1.0
                    coyot = True              neg : pos    =     13.5 : 1.0
                     lore = True              pos : neg    =     13.2 : 1.0
                   misfir = True              neg : pos    =     12.8 : 1.0
                    pixar = True              pos : neg    =     12.5 : 1.0
                  incoher = True              neg : pos    =     12.1 : 1.0
                      cun = True              pos : neg    =     11.8 : 1.0
                    elmor = True              pos : neg    =     11.8 : 1.0
                    jumbo = True              neg : pos    =     11.5 : 1.0
                  anecdot = True              pos : neg    =     11.1 : 1.0
                   osment = True              pos : neg    =     11.1 : 1.0
                  redford = True              pos : neg    =     11.1 : 1.0
                   annual = True              pos : neg    =     11.1 : 1.0
                     3000 = True              neg : pos    =     10.9 : 1.0
                   predat = True              neg : pos    =     10.5 : 1.0
                   polley = True              pos : neg    =     10.5 : 1.0
                    mulan = True              pos : neg    =     10.3 : 1.0
                   hudson = True              neg : pos    =     10.1 : 1.0
```

Random Forest Classifier with BI-Gram Countvectorizer / TF-IDF

I have used Scikit learns Random Forest classifier, which expect data to be in numeric format. To do that, I have first combined the tokenized words of reviews and then fed them in Count Vectorizer and TF-IDF both one by one to check which produces better result.

Below is an image which shows how a review looks after combining.

try combining all the word tokens. we are doing so because scikit learn TF-IDF and Count vectrizer accepts data in this format

```
: " ".join(cleaned_documents[0][0])

: 'film magnolia compar simpl flower titl movi poster suggest dozen charact introduc develop like petal flower come stem flower b
  egin develop grow farther farther apart numer charact film close connect matter differ one anoth social humili kind cop age gam
  e show host one charact suffer kind pain serious lone seek perfect companion end loneli whether companion distant famili member
  spous charact beg one spend whole film search perfect one magnolia clever well thought film prodigi director writer paul thoma
  anderson boogi night fame detail charact analys power script make magnolia memor howev realli impress even though ton charact s
  ever unrel stori film never confus anderson control transit one stori anoth stori master also know exact cut anoth subplot sinc
  not one magnolia scene drag long three hour film not deliv even one uninterest scene definit worth recommend fascin begin riski
  yet surpris satisfi end film get bore obvious anderson put lot time connect subplot ultim game six degre separ everi charact co
  nnect everi charact two degre exampl cop john reilli connect game show host phillip baker hall hall daughter melora walter went
  date reilli fact charact differ other yet similar show univers pain class race gender suffer anderson explan prove point accur
  convinc mistak anderson made tri connect charact make actor sing aime mann song simultan foolish not unbeliev hilari erron also
  broadway music feel movi far apart typic broadway music get perform magnolia perfect classic exampl flawless cast flawless ense
  mbl act anderson allow mani hollywood best support actor blossom best charact act main charact stori perfect abl reveal emot at
  titud brief time screen movi star featur length film worth note tom cruis step usual superfici star role funni depress tour for
  c perform probabl notic actor film award giver role much differ impress past howev actor film could nomin best support actor wi
  thout complaint problem act charact felt serious edit melinda dillon charact phillip baker hall wife seem thrown toward end fel
  t littl sympathi not seen enough earlier film magnolia use sever effect metaphor describ charact anderson talent make audienc s
  ympathet toward numer peopl hour see anderson abl perfect portray mani differ type peopl show keen sens divers divers make dire
  ctor success long period time predict lot divers success anderson futur'
```

First, I have trained random forest with vectors that I received with count vectorizer and this model produced the better results than others. It got 85% accuracy, precision, recall and F1-score.

Second, I have again trained random forest with TF-IDF vectors and it performed almost equal to Naïve Bays. I got 82.5% accuracy, precision, recall and F1-score with this model.

Next page contains an image of the scores received with random forest and CountVectorizer

Feature Importance Using Random Forest

Random forest offers importance score against each attribute. We can take advantage of it and know which features were most important. To know the importance of each feature, type in below command

feat=clf.feature_importances_

where clf is the trained random forest model. We can sort the result coming from this command and select top n number of features.

We then fed this result in below command to know real feature names

tf_idf_vect.get_feature_names()[index] (please refer ipynb file for more detail)

accuracy_score

```
accuracy_score(pred,test_Y)
```

```
0.855
```

precision_score

```
precision_score(test_Y, pred,average='weighted')
```

```
0.8576465229274217
```

recall_score

```
recall_score(test_Y, pred,average='weighted')
```

```
0.855
```

f1_score

```
f1_score(test_Y, pred,average='weighted')
```

```
0.854574075468856
```

confusion_matrix

```
cm = confusion_matrix(pred, test_Y)
print(cm)
```

```
[[78 12]
 [20 90]]
```

Feture Importence using Random Forest

```
print('Feature','  :  ','Importence Score')
print('-'*35)
for i in range(len(max_index_list)):
    index=max_index_list[i]
    print(features[index], '  :  ',importence_value_list[i])
```

```
Feature   :   Importence Score
-----------------------------------
noth   :   0.003272005088999703
bad   :   0.002389649241501629
suppos   :   0.0022976954961258836
wast   :   0.002171306580406732
plot   :   0.0018140104066909357
insult   :   0.0017961682887674749
terribl   :   0.0017723512127874963
worst   :   0.0017275067598446475
stupid   :   0.0015413351052759306
perfect   :   0.0014811130327953036
script   :   0.0014120562687246364
bore   :   0.001358597813049846
anyway   :   0.0013547853744818967
bad movi   :   0.0013370619023370279
tri   :   0.0013245794093002872
excel   :   0.0012570122681253412
clich   :   0.0012198713845420182
mayb   :   0.0012092612633621991
memor   :   0.0012052454236444554
movi   :   0.001130658594963466
```