In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [1]:
```
import json
with open("../hw2/hw2.json") as fp:
    movie_info = json.load(fp)
fp.close()
# print(movie_info[0])
```

In [2]:
```
import jieba
import re
from zhon.hanzi import punctuation
from nltk.corpus import stopwords
punctuations = list(punctuation)
stopword = [line.strip().replace('\n','') for line in open('../hw1/stopwords
punctuations = [line.strip().replace('\n','') for line in open('../hw1/punct
jieba.load_userdict('../hw1/userDict.txt')
stop_words = stopwords.words("chinese")
```

```
Building prefix dict from the default dictionary ...
Loading model from cache /var/folders/2m/2yp8xh891bd67xxdsy56_6gc0000gn/T/ji
eba.cache
Loading model cost 0.744 seconds.
Prefix dict has been built successfully.
```

In [3]:
```
def preprocess(data):
    data = re.sub('[^\u4e00-\u9fa5]','',data)
    info = data.replace('\n','').replace('\t','').replace('\u3000','')
    info = jieba.lcut(info)
    Info = [i for i in list(info) if i not in stopword]
    Info = [i for i in list(info) if i not in stop_words]
    return Info
```

In [4]:
```
DATA =[]
labels = []
for movie in movie_info[:6000]:
    info = preprocess(movie['intro'])
    info = [" ".join(info)]
    info.insert(0,movie['label[class]'][0] if len(movie['label[class]'])>0 e
    labels.append(movie['label[class]'][0] if len(movie['label[class]'])>0 e
    DATA.append(info)
```

In [5]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
train_x = [data[1] for data in DATA]
train_y = [data[0] for data in DATA]
TFIDF = TfidfVectorizer(token_pattern =r"(?u)\b\w\w+\b")
train_X = TFIDF.fit_transform(train_x)
train_X = TFIDF.transform(train_x)
```

In [6]:
```
label_class = list(dict.fromkeys(labels))
print(label_class)
```

```
['劇情', '動作', '喜劇', '冒險', '奇幻', '愛情', '溫馨/家庭', '科幻', '犯罪', '戰
爭', '紀錄片', '懸疑/驚悚', '動畫', '恐怖', '勵志', '歷史/傳記', '音樂/歌舞', 'NA']
```

In [15]:
```
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
import numpy as np
svm_clf = SVC()
print(f"SVM score, cross validated:   {np.mean(cross_val_score(svm_clf, trai
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
SVM score, cross validated:   0.498
```

In [16]:
```
from sklearn.naive_bayes import MultinomialNB
naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(train_X, train_y)
print(f"NAIVE BAYES score, cross validated:   {np.mean(cross_val_score(naive
```

```
NAIVE BAYES score, cross validated:   0.421
```

In [17]:
```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors=3)
KNN.fit(train_X, train_y)
print(f"KNN score, cross validated:   {np.mean(cross_val_score(KNN, train_X,
```

```
KNN score, cross validated:   0.427
```

In [18]:
```
from sklearn import ensemble
RF = ensemble.RandomForestClassifier(n_estimators = 100)
RF.fit(train_X, train_y)
print(f"RF score, cross validated:   {np.mean(cross_val_score(RF, train_X, t
```

```
RF score, cross validated:   0.504
```

In [ ]:
```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js