

➤ Sentiment Analysis of the IMDB comments dataset

Author: M. Kolaksazov

Abstract:

This is a task for analysis of the sentiment of comments from the IMDB.

```
from zipfile import ZipFile
file_name = "aclImdb.zip"
with ZipFile(file_name, "r") as zip:
    zip.extractall()
    print("Done")
```



Done

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.externals import joblib
```

```
↳ /usr/local/lib/python3.6/dist-packages/sklearn/externals/joblib/__init__.py:15: Depre
warnings.warn(msg, category=DeprecationWarning)
```

```
nltk.download("stopwords")
from nltk.corpus import stopwords
```

```
↳ [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
```

```
import os
import glob
import re

BASE_DATA_FOLDER = "aclImdb"
TRAIN_DATA_FOLDER = os.path.join(BASE_DATA_FOLDER, "train")

ROWS = 12500+12500
COLS = 2

reviews_train = np.ndarray(shape=(ROWS, COLS), dtype=object)
reviews_train = list(reviews_train)

for i, file_path in enumerate(glob.glob(os.path.join(TRAIN_DATA_FOLDER, '*', '*.txt'))):
    if re.split(r"\\", str(file_path))[2] != 'unsup':
        for line in open(str(file_path), 'r', encoding="utf8"):
            reviews_train[i][0] = str(line.strip())
```

```
reviews_train[i][1] = str(re.split('_|\\.', str(file_path))[1])
```

```
print(reviews_train[20000])
```

 ["This Documentary (Now available free on Video.Google.Com) is a fantastic demonstrat
'10']

```
BASE_DATA_FOLDER = "aclImdb"
Test_DATA_FOLDER = os.path.join(BASE_DATA_FOLDER, "test")
```

```
ROWS = 12500+12500
COLS = 2
```

```
reviews_test = np.ndarray(shape=(ROWS, COLS), dtype=object)
reviews_test = list(reviews_test)
```

```
for i, file_path in enumerate(glob.glob(os.path.join(Test_DATA_FOLDER, '*', '*.txt'))):
    #if re.split(r"\\", str(file_path))[2] != 'unsup':
        for line in open(str(file_path), 'r', encoding="utf8"):
            reviews_test[i][0] = str(line.strip())
            reviews_test[i][1] = str(re.split(' |\\.', str(file_path))[1])
```

```
print(reviews_test[20000])
```

 ["'In the Line of Fire' is one of those Hollywood films that shows up on tv quite a b
'9']

```
replace_no_space = re.compile(r"(\.)|(\;)|(\:)|(!)|(\')|(\?)|(\,)|(\")|(\(|\)|)|(\[|\]|)|(\{|\})")
replace_with_space = re.compile("<br\s*/><br\s*/>|(\-)|(\\/)")
```

```
def preprocess_reviews(reviews):
    reviews = [replace_no_space.sub("", str(line).lower()) for line in reviews]
    reviews = [replace_with_space.sub(" ", line) for line in reviews]
    return reviews
```

```
reviews_train_clean = preprocess_reviews(np.asarray(reviews_train)[:,:0])
reviews_test_clean = preprocess_reviews(np.asarray(reviews_test)[:,:0])
```

```
print(reviews_train_clean[20000])
```

 this documentary now available free on [videogoogle.com](https://www.videogoogle.com) is a fantastic demonstration of

```
print(reviews_test_clean[20000])
```

 in the line of fire is one of those hollywood films that shows up on tv quite a bit b

```
reviews_train_label = np.asarray(reviews_train)[:,-1]
reviews_test_label = np.asarray(reviews_test)[:,-1]
```

```
english_stop_words = stopwords.words('english')
def remove_stop_words(corpus):
    removed_stop_words = []
    for review in corpus:
```


```
removed_stop_words.append(' '.join([word for word in review.split() if word not in englis
return removed_stop_words
```

```
reviews_train_ready = remove_stop_words(reviews_train_clean)
reviews_test_ready = remove_stop_words(reviews_test_clean)
```


```
print(reviews_train_ready[20000])
```

 documentary available free videogooglegom fantastic demonstration power ordinary peop

```
print(reviews_test_ready[20000])
```


 line fire one hollywood films shows tv quite bit although ive seen times usually end

```
tfidf_vectorizer = TfidfVectorizer()
tfidf_vectorizer.fit(reviews_train_ready)
```


 TfidfVectorizer(analyzer='word', binary=False, decode_error='strict', dtype=<class 'numpy.float64'>, encoding='utf-8', input='content', lowercase=True, max_df=1.0, max_features=None, min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=True, stop_words=None, strip_accents=None, sublinear_tf=False, token_pattern='(?u)\\b\\w+\\b', tokenizer=None, use_idf=True, vocabulary=None)

```
reviews_features = tfidf_vectorizer.transform(reviews_train_ready)
x_test = tfidf_vectorizer.transform(reviews_test_ready)
```


```
labels = [0 if i < 12500 else 1 for i in range(25000)]
x_train, x_val, y_train, y_val = train_test_split(reviews_features, labels, train_size=0.8, randc
x_train.shape
x_train
```

 <20000x92694 sparse matrix of type '<class 'numpy.float64'>' with 2027185 stored elements in Compressed Sparse Row format>

```
x_train[0]
```


 <1x92694 sparse matrix of type '<class 'numpy.float64'>' with 164 stored elements in Compressed Sparse Row format>

```
k_fold = StratifiedKFold(n_splits = 5)
k_fold.split(x_train, y_train)
```

 <generator object _BaseKFold.split at 0x0000000015DFE408>

```
def model_logistic_regression_tuned(x, y):
    grid_search = GridSearchCV(LogisticRegression(), tuned_params)
    model_logistic_regression = grid_search.fit(x, y)
    return model_logistic_regression.best_estimator_
```


```
tuned_params = [{"C": [0.05, 0.1, 0.3, 0.5, 1]}]
model_logistic_regression_tuned(x_train, y_train)
```

 C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\model_selection_split.py:2053: FutureWarning: warn(CV_WARNING, FutureWarning)

C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning

```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)
```


```
tuned_params = [{"C": [1, 2, 3, 4, 5]}]
model_logistic_regression_tuned(x_train, y_train)
```

 C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\model_selection_split.py:2053: FutureWarning: warn(CV_WARNING, FutureWarning)

C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning

```
LogisticRegression(C=3, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)
```

```
tuned_params = [{"C": [3.5, 4, 4.5], "max_iter": [5, 10, 15]}]
model_logistic_regression_tuned(x_train, y_train)
```


 C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\model_selection_split.py:2053: FutureWarning: warn(CV_WARNING, FutureWarning)

C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning

C:\Users\Marko\Anaconda3\lib\site-packages\sklearn\svm\base.py:931: ConvergenceWarning: "the number of iterations.", ConvergenceWarning)

```
LogisticRegression(C=4.5, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=5, multi_class='warn', n_jobs=None,
                    penalty='l2', random_state=None, solver='warn', tol=0.0001,
                    verbose=0, warm_start=False)
```


```
model_logistic_regression = LogisticRegression(C = 4, max_iter = 10)
model_logistic_regression.fit(x_train, y_train)
```



```
LogisticRegression(C=4, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=10, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)
```

```
accuracy_train = accuracy_score(y_train, model_logistic_regression.predict(x_train))
accuracy_validation = accuracy_score(y_val, model_logistic_regression.predict(x_val))
```

```
print("Training score:", round(accuracy_train, 2))
print("Validation score:", round(accuracy_validation, 2))
```

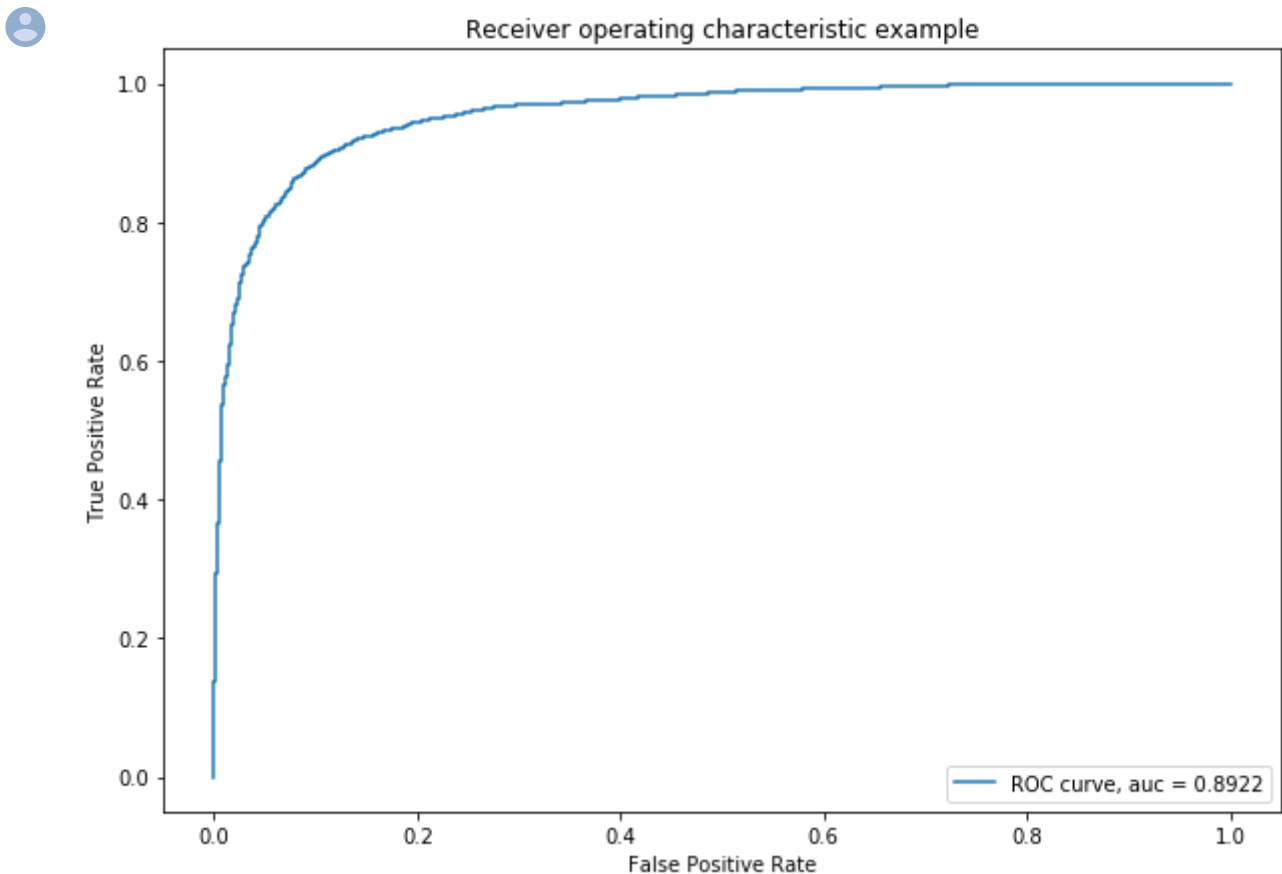
 Training score: 0.98
Validation score: 0.89

```
final_accuracy = accuracy_score(labels, model_logistic_regression.predict(x_test))
print("Testing score:", round(final_accuracy, 2))
```

Testing score: 0.88

```
def receiver_perating_characteristic(model, x, y, accuracy):
    plt.figure(figsize = (10, 7))
    y_pred_proba = model.predict_proba(x)[::, 1]
    fpr, tpr, _ = roc_curve(y, y_pred_proba)
    auc = roc_auc_score(y, y_pred_proba)
    plt.plot(fpr, tpr, label = "ROC curve, auc = " + str(accuracy))
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc = 4)
    plt.show()
```

```
receiver_perating_characteristic(model_logistic_regression, x_val,
y_val, accuracy_validation)
```



```
joblib.dump(model_logistic_regression, 'Movie_Reviews.pkl')
```

['Movie_Reviews.pkl']


```
reviews_alita = []
for line in open("alita.txt", 'r'):
    reviews_alita.append(line.strip())

reviews_alita_clean = preprocess_reviews(reviews_alita)
reviews_alita_ready = remove_stop_words(reviews_alita_clean)
reviews_alita_ready
```


```
['n»iin 2563 three hundred years war fall mysterious scientist dr dyson ido christoph  
'year 2563 300 years fall sky city zalem elysium rules factory town iron city make w  
'james cameron really busy avatar 2 5 know wasnt supposed trilogy find time produce  
'film based upon 660 years science fiction flicks youve already seen names changed s  
'movies james cameron stealing movies terminator stealing westworld gunslinger many  
'like wacky awkward cheesiness rob rodriguezs movies wont disappointed youre looking  
'plot sucks feels dated finished half way story set sequels film making exploitation  
'biggest issue clear plot nothing actually happens one minute becoming bounty hunter
```

```
reiews_alita_ready = tfidf_vectorizer.transform(reviews_alita_ready)
```

```
reiews_alita_ready.shape
```

 (8, 92694)

```
clf2 = joblib.load('Movie_Reviews.pkl')  
# Predict data set using loaded model  
predict = clf2.predict(reiews_alita_ready)  
# 8,10,8,7,1,2,2,3  
predict
```

 array([1, 1, 1, 0, 0, 0, 0, 0])