ML_Project_Fake_News_Detector

September 18, 2020

1 Initialization tweaks

- Initiating GoogleDrive mounting
- Necessary Library's imports
- Load Dataset [**Optional**]

[ ]:
```
#-Google Drive Access-Granting
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
# files paths
path = '/content/drive/My Drive/ThesisFile/datasets/ml-project/
 ↪fake_real-news-dataset/'
if path[-1] != '/':
  path += '/'
print('Good to go  ')
```

```
Good to go
```

[ ]:
```
#-Library imports
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import statistics
import math
import nltk
import string
from time import time
import seaborn as sns
from sklearn import preprocessing
from sklearn import linear_model
from sklearn import svm
from sklearn import metrics
from sklearn.utils import validation
```

```
from sklearn.utils.extmath import density
from sklearn.cluster import KMeans
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression, PassiveAggressiveClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import normalize
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, pairwise_distances,␣
 ↪mean_absolute_error, classification_report, confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
from seaborn import heatmap
from math import exp
from random import randint, seed
import itertools
from statsmodels.graphics.mosaicplot import mosaic
from matplotlib.patches import Patch
from collections import deque

%matplotlib inline
```

[ ]:
```
def nclass_classification_mosaic_plot(n_classes, results):
    """
    build a mosaic plot from the results of a classification

    parameters:
    n_classes: number of classes
    results: results of the prediction in form of an array of arrays

    In case of 3 classes the prdiction could look like
    [[10, 2, 4],
     [1, 12, 3],
     [2, 2, 9]
    ]
    where there is one array for each class and each array holds the
    predictions for each class [class 1, class 2, class 3].
```

```
    This is just a prototype including colors for 6 classes.
    """
    class_lists = [range(n_classes)]*2
    mosaic_tuples = tuple(itertools.product(*class_lists))

    res_list = results[0]
    for i, l in enumerate(results):
        if i == 0:
            pass
        else:
            tmp = deque(l)
            tmp.rotate(-i)
            res_list.extend(tmp)
    data = {t:res_list[i] for i,t in enumerate(mosaic_tuples)}

    fig, ax = plt.subplots(figsize=(11, 10))
    plt.rcParams.update({'font.size': 16})

    font_color = '#2c3e50'
    pallet = [
        '#6a89cc',
        '#4a69bd',
        '#1e3799',
        '#0c2461',
        '#82ccdd',
        '#60a3bc',
    ]
    colors = deque(pallet[:n_classes])
    all_colors = []
    for i in range(n_classes):
        if i > 0:
            colors.rotate(-1)
        all_colors.extend(colors)

    props = {(str(a), str(b)):{'color':all_colors[i]} for i,(a, b) in␣
↪enumerate(mosaic_tuples)}

    labelizer = lambda k: ''

    p = mosaic(data, labelizer=labelizer, properties=props, ax=ax)

    title_font_dict = {
        'fontsize': 20,
        'color' : font_color,
    }
    axis_label_font_dict = {
        'fontsize': 16,
```

```
        'color' : font_color,
    }

    ax.tick_params(axis = "x", which = "both", bottom = False, top = False)
    ax.axes.yaxis.set_ticks([])
    ax.tick_params(axis='x', which='major', labelsize=14)

    ax.set_title('Classification Report', fontdict=title_font_dict, pad=25)
    ax.set_xlabel('Observed Class', fontdict=axis_label_font_dict, labelpad=10)
    ax.set_ylabel('Predicted Class', fontdict=axis_label_font_dict, labelpad=35)

    legend_elements = [Patch(facecolor=all_colors[i], label='Class {}'.
↪format(i)) for i in range(n_classes)]
    ax.legend(handles=legend_elements, bbox_to_anchor=(1,1.018), fontsize=16)

    plt.tight_layout()
    plt.show()
```

[ ]:
```
# Loading Dataset
file0 = 'news.csv'
file1 = 'Fake.csv'
file2 = 'True.csv'

df0 = pd.read_csv(path + file0)
df1 = pd.read_csv(path + file1)
df2 = pd.read_csv(path + file2)

print(df0.columns, df0, sep='\n', end='\n\n')
print(df1.columns, df1, sep='\n', end='\n\n')
print(df2.columns, df2, sep='\n', end='\n\n')

df1['label'] = 'FAKE'
df2['label'] = 'REAL'

dataset = pd.concat([df0[['title','text','label']] ,␣
↪df1[['title','text','label']] , df2[['title','text','label']]],␣
↪ignore_index=True)

dataset = dataset.sample(frac=1).reset_index(drop=True)
print(dataset.columns, dataset.shape)
dataset.tail()

# print('Dataset Attributes::', dataset.columns)
# print('Dataset Dim::', dataset.shape)
# print(dataset.head())
# print('\nData Output::\n', dataset.label)
```

```
Index(['Unnamed: 0', 'title', 'text', 'label'], dtype='object')
      Unnamed: 0  … label
0           8476  …  FAKE
1          10294  …  FAKE
2           3608  …  REAL
3          10142  …  FAKE
4            875  …  REAL
…           …  …   …
6330        4490  …  REAL
6331        8062  …  FAKE
6332        8622  …  FAKE
6333        4021  …  REAL
6334        4330  …  REAL

[6335 rows x 4 columns]

Index(['title', 'text', 'subject', 'date'], dtype='object')
                                                  title  …                date
0       Donald Trump Sends Out Embarrassing New Year'…  …  December 31, 2017
1       Drunk Bragging Trump Staffer Started Russian …  …  December 31, 2017
2       Sheriff David Clarke Becomes An Internet Joke…  …  December 30, 2017
3       Trump Is So Obsessed He Even Has Obama's Name…  …  December 29, 2017
4       Pope Francis Just Called Out Donald Trump Dur…  …  December 25, 2017
…                                                   …  …                …
23476  McPain: John McCain Furious That Iran Treated …  …   January 16, 2016
23477  JUSTICE? Yahoo Settles E-mail Privacy Class-ac…  …   January 16, 2016
23478  Sunnistan: US and Allied 'Safe Zone' Plan to T…  …   January 15, 2016
23479  How to Blow $700 Million: Al Jazeera America F…  …   January 14, 2016
23480  10 U.S. Navy Sailors Held by Iranian Military …  …   January 12, 2016

[23481 rows x 4 columns]

Index(['title', 'text', 'subject', 'date'], dtype='object')
                                                  title  …
date
0       As U.S. budget fight looms, Republicans flip t…  …  December 31, 2017
1       U.S. military to accept transgender recruits o…  …  December 29, 2017
2       Senior U.S. Republican senator: 'Let Mr. Muell…  …  December 31, 2017
3       FBI Russia probe helped by Australian diplomat…  …  December 30, 2017
4       Trump wants Postal Service to charge 'much mor…  …  December 29, 2017
…                                                   …  …
…
21412  'Fully committed' NATO backs new U.S. approach…  …    August 22, 2017
21413  LexisNexis withdrew two products from Chinese …  …    August 22, 2017
21414  Minsk cultural hub becomes haven from authorities  …    August 22, 2017
21415  Vatican upbeat on possibility of Pope Francis …  …    August 22, 2017
21416  Indonesia to buy $1.14 billion worth of Russia…  …    August 22, 2017
```

```
[21417 rows x 4 columns]

Index(['title', 'text', 'label'], dtype='object') (51233, 3)
```

[ ]:
```
                                                   title  … label
51228  Erdogan, Putin to discuss Syria, Jerusalem dur…  …  REAL
51229  LET THE BOYCOTTS BEGIN: KEURIG COFFEE and 4 Ma…  …  FAKE
51230  Egypt blocks Human Rights Watch website amid w…  …  REAL
51231  Minnesota governor to undergo surgery for canc…  …  REAL
51232   Bernie Sanders Will Visit The Vatican To Spea…  …  FAKE

[5 rows x 3 columns]
```

2 Analyzing Dataset

- Determining the Labels(outputs) and Inputs

[ ]:
```
# Outputs or Labels
labels = dataset.label;
print(labels)

# Sample Data
df = dataset['text']
df
```

```
0        FAKE
1        REAL
2        REAL
3        REAL
4        FAKE
        …
51228    REAL
51229    FAKE
51230    REAL
51231    REAL
51232    FAKE
Name: label, Length: 51233, dtype: object
```

[ ]:
```
0        While many in the United States have come to t…
1        MANCHESTER, England (Reuters) - British Prime …
2        ANKARA (Reuters) - Turkey received bids last F…
3        WASHINGTON (Reuters) - Two former top U.S. int…
4        What s it going to take for someone to step in…
                              …
51228    ANKARA (Reuters) - Russian President Vladimir …
51229    Before Keurig, and other major companies commi…
51230    CAIRO (Reuters) - Egypt has blocked the websit…
```

```
51231    (Reuters) - Minnesota Governor Mark Dayton wil…
51232    Presidential Candidate Bernie Sanders, who fac…
Name: text, Length: 51233, dtype: object
```

3 Splitting the dataset

- Split the dataset into training and testing sets.

[ ]:
```
# Split Dataset with a 8:2 ratio for training and testing
#         sampple data:: the 'text' column only.
x_train,x_test,y_train,y_test = train_test_split(df, labels, test_size=0.2,
                                                 random_state=7)
print(x_train)
print(y_train)

print(x_test)
```

```
6890     Police in this country had a great run. Let s …
45936    SYDNEY (Reuters) - Australians turned in 51,00…
22230    A Wisconsin judge has refused to order local o…
40082    LIMA (Reuters) - Brazilian builder Odebrecht […
42905    Many Americans have suspected that the rise of…
                              …
13927    The NAACP staged a sit-in at the office of Ala…
919      WASHINGTON (Reuters) - U.S. Congressman Steve …
38467    BEIRUT (Reuters) - The Syrian government rejec…
10742
49689    Sean Spicer gave Hillary Clinton a little dig …
Name: text, Length: 40986, dtype: object
6890     FAKE
45936    REAL
22230    FAKE
40082    REAL
42905    FAKE
         …
13927    FAKE
919      REAL
38467    REAL
10742    FAKE
49689    FAKE
Name: label, Length: 40986, dtype: object
23499    Private jets, lots of cash, presidential suite…
5965     Trump administration officials are mulling an …
47211    **Want FOX News First in your inbox every day?…
25930     Members Of The Intelligence Committee Simply …
20885    TWITTER IS ABUZZ OVER THE FLY THAT LANDED ON H…
                              …
```

```
28054    WASHINGTON (Reuters) - A group of Democratic l…
12110    KABUL (Reuters) - Vice President Mike Pence ma…
36322    West Virginia has been devastated by a loss of…
44194    B..b..but That can t be right Maybe we should …
44946    WASHINGTON (Reuters) - U.S. President Donald T…
Name: text, Length: 10247, dtype: object
```

4 Dataset Preprocessing using a TfidfVectorizer

Let's initialize a ***TfidfVectorizer*** with stop words from the English language and a maximum document frequency of 0.7 (terms with a higher document frequency will be discarded). Stop words are the most common words in a language that are to be filtered out before processing the natural language data. And a *TfidfVectorizer* turns a collection of raw documents into a matrix of TF-IDF features.

- **TF (Term Frequency)**: The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.
- **IDF (Inverse Document Frequency):** Words that occur many times a document, but also occur many times in many others, may be irrelevant. IDF is a measure of how significant a term is in the entire corpus.

[ ]:
```
# Initialize a TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Fit and transform train set, transform test set
t0 = time()
tfidf_train = tfidf_vectorizer.fit_transform(x_train)
tfidf_test = tfidf_vectorizer.transform(x_test)
preprocess_time = time() - t0
print(tfidf_test)
```

```
  (0, 128393)   0.09160219142317329
  (0, 123421)   0.050692675829555334
  (0, 120018)   0.13358732532461629
  (0, 115093)   0.0572543300124491
  (0, 114933)   0.09845238614140546
  (0, 110795)   0.16255093777486046
  (0, 109086)   0.039520328193992676
  (0, 108049)   0.12782150832497632
  (0, 107958)   0.1345110883906628
  (0, 105019)   0.06957679124535648
  (0, 100453)   0.11024026684077948
  (0, 100003)   0.06632491121109792
  (0, 97064)    0.06050491380951155
  (0, 96725)    0.11157827801659405
  (0, 96543)    0.0747981500020641
  (0, 95274)    0.1397670183327725
```

```
(0, 93563)    0.09798703217176453
(0, 92821)    0.10223071197467873
(0, 91620)    0.06732768161628239
(0, 91280)    0.048005949675386045
(0, 91183)    0.28908697090133667
(0, 85962)    0.15111257704319278
(0, 85626)    0.08882542864779046
(0, 84505)    0.06068187171259706
(0, 80052)    0.05359030768161428
:     :
(10246, 61957)        0.09524361237419485
(10246, 57904)        0.10579831006675786
(10246, 57355)        0.2059306774710701
(10246, 57349)        0.11435602692754665
(10246, 57118)        0.10385007438687317
(10246, 55620)        0.12558886057179253
(10246, 50944)        0.1456763258561254
(10246, 50680)        0.1477767435225432
(10246, 49653)        0.13879756022869624
(10246, 44876)        0.2116990616578496
(10246, 44873)        0.1721598615396174
(10246, 37026)        0.05354472949786926
(10246, 34395)        0.15432588922068582
(10246, 33043)        0.08903448147152995
(10246, 29124)        0.10468837600568447
(10246, 29118)        0.167842154618641
(10246, 28995)        0.14385537407476698
(10246, 28993)        0.27219953112183265
(10246, 21445)        0.1303388370844514
(10246, 18643)        0.09024454380991116
(10246, 9465) 0.07757296809157904
(10246, 9298) 0.10567525369416872
(10246, 9126) 0.09530504316714665
(10246, 7233) 0.1478275291099032
(10246, 1)    0.0805323369953822
```

5 Learning Model Selection

Choose a Learning Model, Methodology or Schema for training the dataset.

Here, as it's a classification problem, we are using a ***PassiveAggressiveClassifier*** due to the fact that we have vectorized the sample data during the preprocessing step using a *TfidfVectorizer*.

For explaination purpose, **Passive Aggressive algorithms** are online learning algorithms. Such an algorithm remains passive for a correct classification outcome, and turns aggressive in the event of a miscalculation, updating and adjusting. Unlike most other algorithms, it does not converge. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector.

[ ]:
```
# Initialize a PassiveAggressiveClassifier
model = PassiveAggressiveClassifier(max_iter=50)
model
```

[ ]:
```
PassiveAggressiveClassifier(C=1.0, average=False, class_weight=None,
                            early_stopping=False, fit_intercept=True,
                            loss='hinge', max_iter=50, n_iter_no_change=5,
                            n_jobs=None, random_state=None, shuffle=True,
                            tol=0.001, validation_fraction=0.1, verbose=0,
                            warm_start=False)
```

6 Fitting the Model and Predicting its outcomes.

Here we will fit the model with the trained vectorized sample data from `tfidf_train` of the **TfidfVectorizer** and the `y_train` of the initial sample data.

[ ]:
```
# Fitting the Model with the tfidf_train & y_train.
t0 = time()
history = model.fit(tfidf_train,y_train)
train_time = time() - t0


# Predict on the test set tfidf_test from the TfidfVectorizer
t0 = time()
y_pred = model.predict(tfidf_test)
test_time = time() - t0
print('Predicted Outcomes -->', y_pred, y_pred.shape)

#           and

# Calculate the accuracy with accuracy_score()
score = accuracy_score(y_test,y_pred)
# score = model.score(tfidf_test,y_pred)
print(f'\nModel-Accuracy: {round(score*100,2)}%')
```

```
Predicted Outcomes --> ['FAKE' 'FAKE' 'REAL' … 'FAKE' 'REAL' 'REAL'] (10247,)

Model-Accuracy: 97.47%
```

7 Report and Visualization on the selected Model

From the Previous Step, after training and testing the model using *PassiveAggressiveClassifier* we got an *accuracy* of more than **97%** on the testing sample.

For more insights, we will print out the **confusion matrix** to view the number of false and true negatives and positives.

We will also print out the **heat map** generated from the confusion matrix.

[ ]:
```
# Model Report
report = classification_report(y_test,y_pred, labels=['FAKE','REAL'])

print('----------------=====Model Report=====----------------\n')
print(report)
print('-----------------------------------------------------\n')


# Confusion-Matrix
conf_mat = confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])

print('::Confusion-Matrix::')
print('',conf_mat)
print('\n-----------------------------------------------------\n')


# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision: \t%.2f\n" % metrics.precision_score(y_test, y_pred,␣
 ↪average='weighted'))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall: \t%.2f\n" % metrics.recall_score(y_test, y_pred,␣
 ↪average='weighted'))

print("Density: \t%f\n" % density(model.coef_))

print("Dimensionality: %d\n" % model.coef_.shape[1])

print('-----------------------------------------------------\n')

print("Train Time: \t%.3fs\n" % train_time)

print("Test Time:  \t%.3fs\n" % test_time)

print(f'Data-Preprocessing Time: {round(preprocess_time, 3)}s\n')

print('-----------------------------------------------------\n')

print(f'Accuracy Score: {round(score*100,2)}%\n')

print('-----------------------------------------------------\n')
```

```
----------------=====Model Report=====----------------

              precision    recall  f1-score   support

        FAKE       0.97      0.98      0.98      5293
```

```
        REAL       0.97      0.97      0.97      4954

    accuracy                           0.97     10247
   macro avg       0.97      0.97      0.97     10247
weighted avg       0.97      0.97      0.97     10247

----------------------------------------------------

::Confusion-Matrix::
 [[5168  125]
 [ 134 4820]]

----------------------------------------------------

Precision:      0.97

Recall:         0.97

Density:        0.605463

Dimensionality: 129377

----------------------------------------------------

Train Time:     0.536s

Test Time:      0.005s

Data-Preprocessing Time: 12.507s

----------------------------------------------------

Accuracy Score: 97.47%

----------------------------------------------------
```

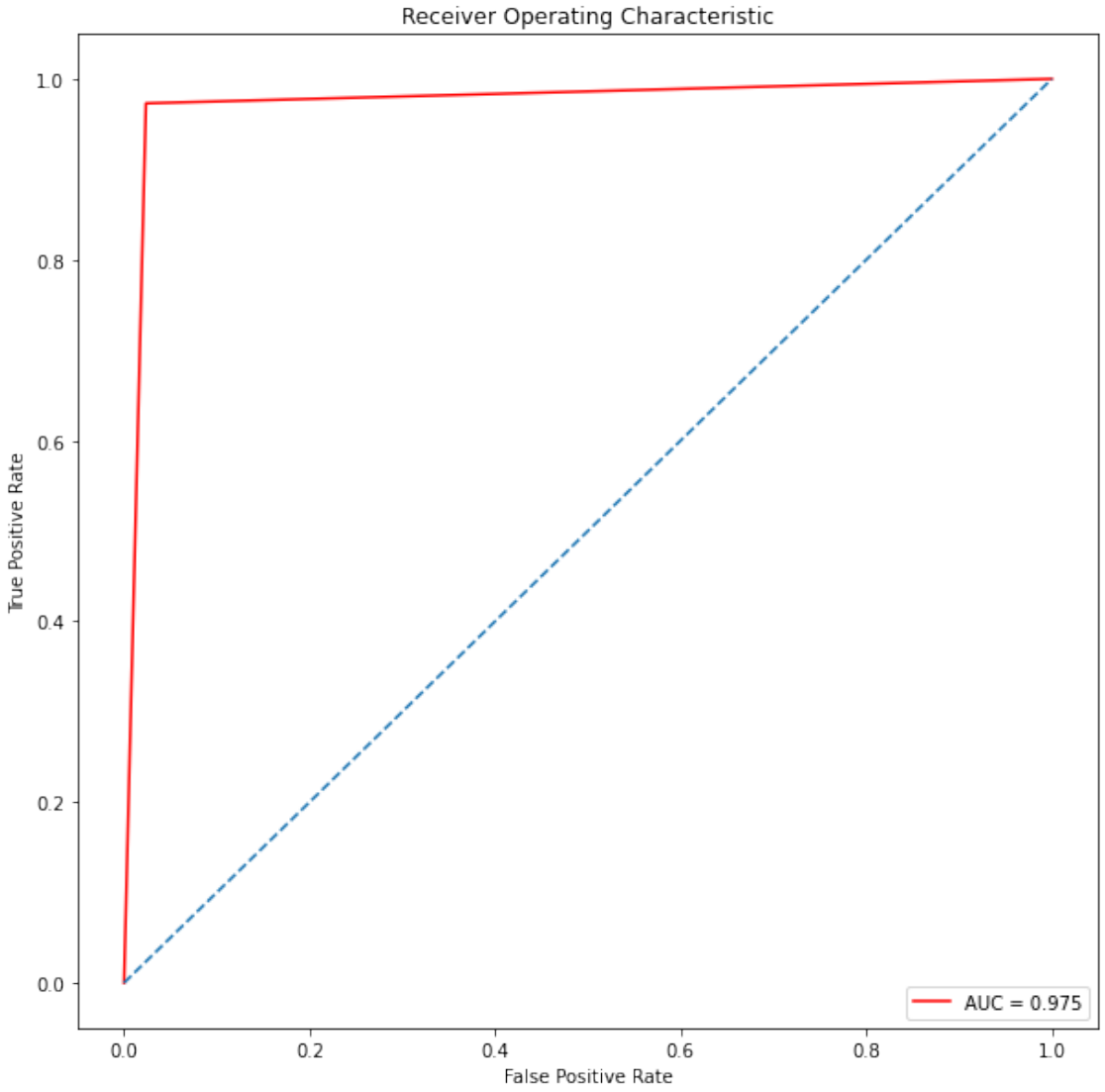
[ ]:
```python
# transform
y_tt = []
y_pr = []
for yt in y_test:
  if yt == 'REAL':
    y_tt.append(1)
  elif yt == 'FAKE':
    y_tt.append(0)

for yp in y_pred:
```

```
  if yp == np.str_('REAL'):
    y_pr.append(1)
  elif yp == np.str_('FAKE'):
    y_pr.append(0)

# print(len(y_tt), len(y_pr))
false_positive_rate, true_positive_rate, thresholds = roc_curve(np.
 ↪array(y_tt),np.array(y_pr))
roc_auc = auc(false_positive_rate, true_positive_rate)
print('ROC-AUC =', roc_auc)
```

```
ROC-AUC = 0.9746675269269267
```

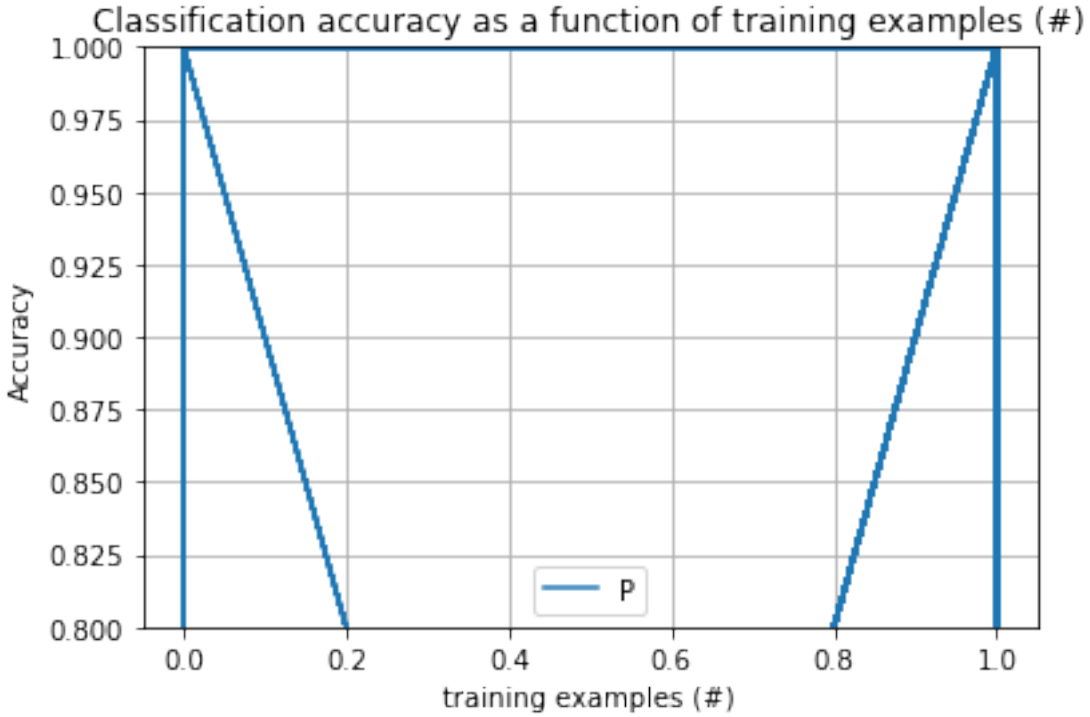
[ ]:
```
plt.figure(figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate,true_positive_rate, color='red',label = 'AUC = %0.
 ↪3f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],linestyle='--')
plt.axis('tight')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
```

[ ]:
```
Text(0.5, 0, 'False Positive Rate')
```

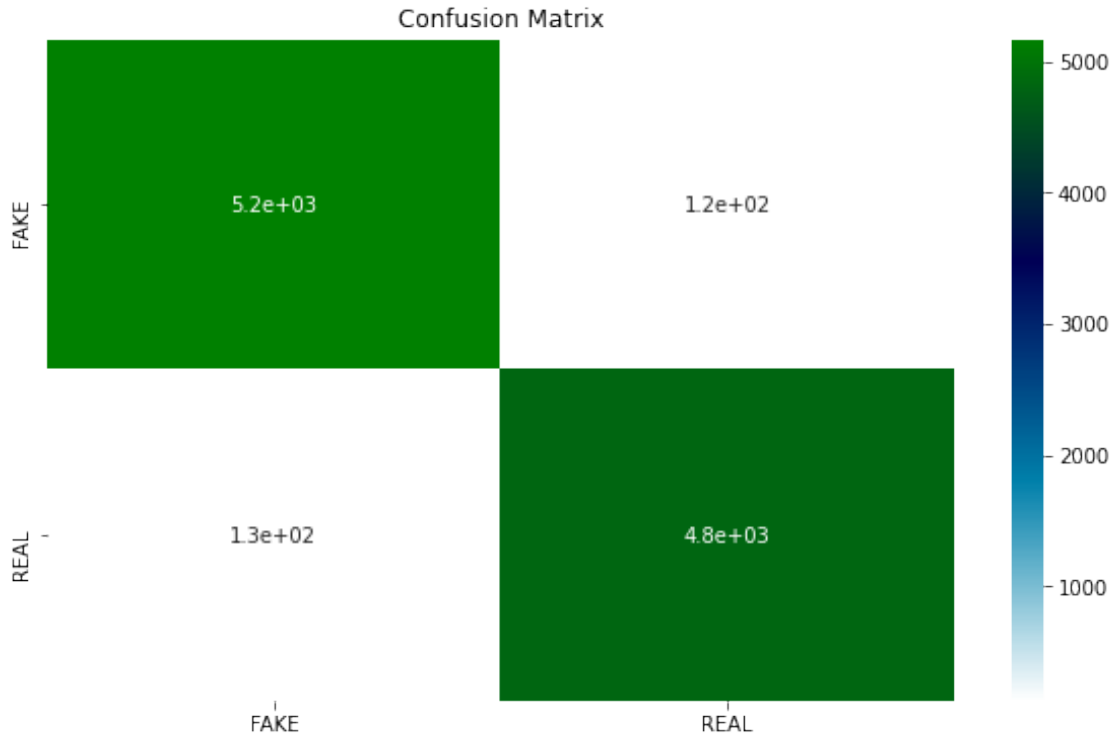


[ ]:
```
def plot_accuracy(x, y, x_legend):
    """Plot accuracy as a function of x."""
    x = np.array(x)
    y = np.array(y)
    plt.title('Classification accuracy as a function of %s' % x_legend)
    plt.xlabel('%s' % x_legend)
    plt.ylabel('Accuracy')
    plt.grid(True)
    plt.plot(x, y)

plt.figure()
plot_accuracy(y_tt,y_pr,"training examples (#)")
ax = plt.gca()
```

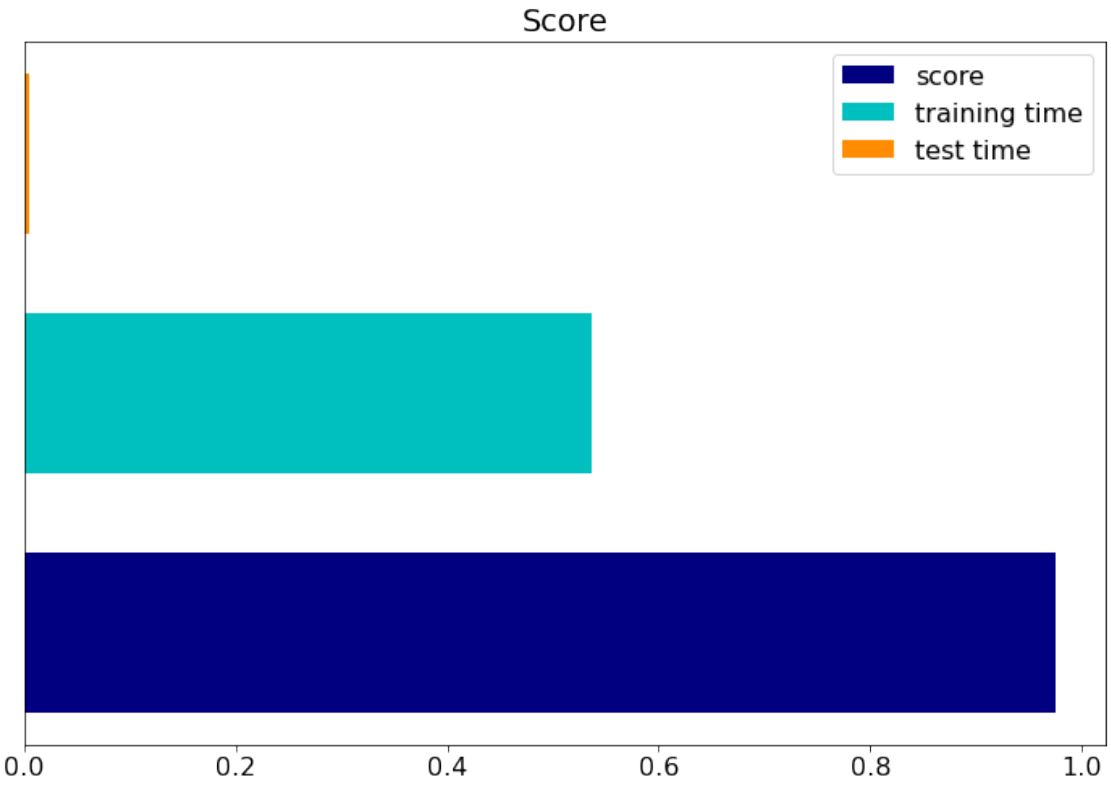
```
ax.set_ylim((0.8, 1))
plt.legend("PAC", loc='best')
plt.show()
```



[ ]:
```
# Visualization of confusion-matrix with a heat-map
fig = plt.figure(figsize=(10,6))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

sns.heatmap(conf_mat, annot=True, cmap=plt.cm.ocean_r,
            xticklabels=['FAKE','REAL'],
            yticklabels=['FAKE','REAL'])
plt.show()
```

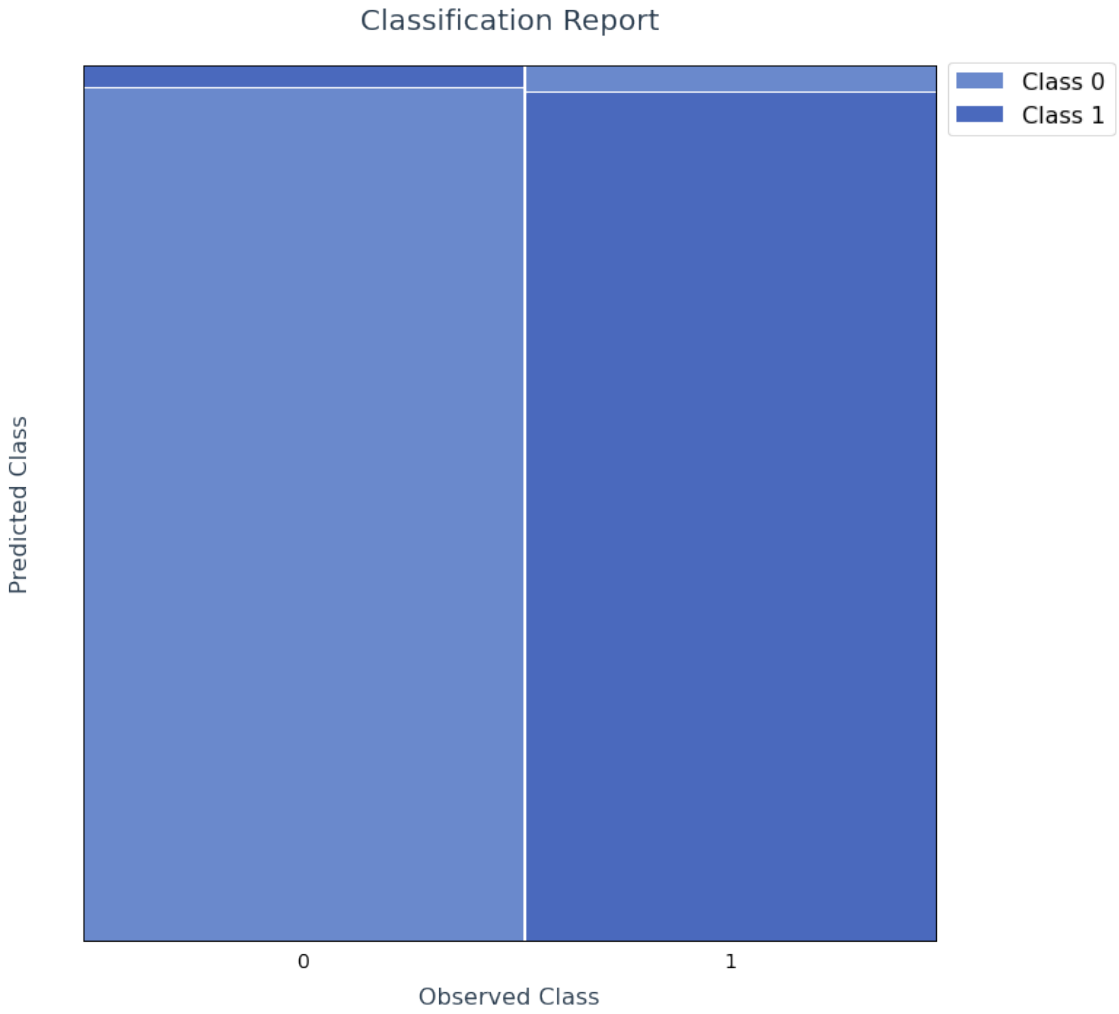


[ ]:
```
indices = np.arange(1)
plt.figure(figsize=(12, 8))
plt.title("Score")
plt.barh(indices, score, .2, label="score", color='navy')
plt.barh(indices + .3, train_time, .2, label="training time",
         color='c')
plt.barh(indices + .6, test_time, .2, label="test time", color='darkorange')
plt.yticks(())
plt.legend(loc='best')
# plt.text(-.4, indices[0], 'Passive-Aggressive Classifier')
plt.show()
```



[ ]:
```
# Converting numpyarray to list
results = list()
for line in conf_mat:
  results.append([data for data in line])
print(type(results), results)

# visualize the results
nclass_classification_mosaic_plot(len(results), results)
```

```
<class 'list'> [[5168, 125], [134, 4820]]
```



8 Testing The Model with user input

```
[ ]: text_data = "In recent weeks, Microsoft has detected cyberattacks targeting␣
     ↪people and organizations involved in the upcoming presidential election,␣
     ↪including unsuccessful attacks on people associated with both the Trump and␣
     ↪Biden campaigns, as detailed below. We have and will continue to defend our␣
     ↪democracy against these attacks through notifications of such activity to␣
     ↪impacted customers, security features in our products and services, and␣
     ↪legal and technical disruptions. The activity we are announcing today makes␣
     ↪clear that foreign activity groups have stepped up their efforts targeting␣
     ↪the 2020 election as had been anticipated, and is consistent with what the U.
     ↪S. government and others have reported. We also report here on attacks␣
     ↪against other institutions and enterprises worldwide that reflect similar␣
     ↪adversary activity." #@param {type:"raw"}
     print(text_data)
```

```
x = [text_data,]

tfidf_test = tfidf_vectorizer.transform(x)

pred = model.predict(tfidf_test)

pred

print('This news is', pred[0])
```

```
In recent weeks, Microsoft has detected cyberattacks targeting people and
organizations involved in the upcoming presidential election, including
unsuccessful attacks on people associated with both the Trump and Biden
campaigns, as detailed below. We have and will continue to defend our democracy
against these attacks through notifications of such activity to impacted
customers, security features in our products and services, and legal and
technical disruptions. The activity we are announcing today makes clear that
foreign activity groups have stepped up their efforts targeting the 2020
election as had been anticipated, and is consistent with what the U.S.
government and others have reported. We also report here on attacks against
other institutions and enterprises worldwide that reflect similar adversary
activity.
This news is REAL
```