TraditionalML

August 18, 2019

[8]:
```
import sys
sys.path.append('/usr/local/bin/python2.7')


import numpy as np
import os, sys, getopt, pickle, csv, sklearn
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import classification_report, f1_score, accuracy_score,␣
 ↪confusion_matrix, make_scorer, recall_score, precision_score,␣
 ↪classification_report, precision_recall_fscore_support
from sklearn.pipeline import Pipeline
#from sklearn.grid_search import GridSearchCV
from sklearn.ensemble  import GradientBoostingClassifier, RandomForestClassifier
from sklearn.model_selection import StratifiedKFold, cross_val_score,␣
 ↪train_test_split, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.utils import shuffle
#from textblob import TextBlob
import random
import matplotlib.pyplot as plt
from sklearn import metrics
from collections import Counter
import argparse
from sklearn.model_selection import cross_validate
from sklearn.metrics import roc_auc_score
import preprocessor as p
```

[9]:
```
models = [ 'svm', 'naive', 'lr', 'random_forest']
NO_OF_FOLDS = 10
MODEL_TYPE = "all"
HASH_REMOVE = None
```

[10]:
```
def load_data(filename):
    data = pickle.load(open(filename, 'rb'))
    x_text = []
    labels = []
```

```
    for i in range(len(data)):
        if(HASH_REMOVE):
            x_text.append(p.tokenize((data[i]['text']).encode('utf-8')))
        else:
            x_text.append(data[i]['text'])
        labels.append(data[i]['label'])
    return x_text,labels

def get_filename(dataset):
    global N_CLASS, HASH_REMOVE
    if(dataset=="twitter"):
        filename = "/home/sujeendra/Desktop/data/data/twitter_data.pkl"
        N_CLASS = 3
        HASH_REMOVE = False
    elif(dataset=="formspring"):
        N_CLASS = 2
        filename = "/home/sujeendra/Desktop/data/data/formspring_data.pkl"
        HASH_REMOVE = False
    elif(dataset=="wiki"):
        N_CLASS = 2
        filename = "/home/sujeendra/Desktop/data/data/wiki_data.pkl"
        HASH_REMOVE = False
    return filename
```

[11]:

```
def get_scores(y_true, y_pred):
#     if(data=="wiki"):
#         auc = roc_auc_score(y_true,y_pred)
#         print('Test ROC AUC: %.3f' %auc)
#     print(":: Confusion Matrix")
#     print(confusion_matrix(y_true, y_pred))
#     print(":: Classification Report")
#     print(classification_report(y_true, y_pred))
    return np.array([
            precision_score(y_true, y_pred, average=None),
            recall_score(y_true, y_pred,  average=None),
            f1_score(y_true, y_pred, average=None)])

def print_scores(scores):
    for i in range(N_CLASS):
        if(i!=0):
            print("Precision Class %d (avg): %0.3f (+/- %0.3f)" % (i,scores[:,␣
 ↪i].mean(), scores[:, i].std() * 2))
            print ("Recall Class %d (avg): %0.3f (+/- %0.3f)" % (i,scores[:, ␣
 ↪N_CLASS+i].mean(), scores[:,N_CLASS+i].std() * 2))
            print ("F1_score Class %d (avg): %0.3f (+/- %0.3f)" % (i,scores[:,␣
 ↪N_CLASS*2+i].mean(), scores[:,  N_CLASS*2+i].std() * 2))
```

[13]:
```
def classification_model(X, Y, model_type):
    X, Y = shuffle(X, Y, random_state=42)
    print ("Model Type:", model_type)
    kf = KFold(n_splits=NO_OF_FOLDS)
    scores = []
    for train_index, test_index in kf.split(X):
        Y = np.asarray(Y)
        model = get_model(model_type)
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = Y[train_index], Y[test_index]
        model.fit(X_train,y_train)
        y_pred = model.predict(X_test)
        curr_scores = get_scores(y_test, y_pred)
        scores.append(np.hstack(curr_scores))
    print_scores(np.array(scores))
```

[14]:
```
def get_model(m_type):
    if m_type == 'lr':
        logreg = LogisticRegression(class_weight="balanced")
    elif m_type == 'naive':
        logreg =  MultinomialNB()
    elif m_type == "random_forest":
        logreg = RandomForestClassifier(n_estimators=100, n_jobs=-1)
    elif m_type == "svm":
        logreg = LinearSVC(class_weight="balanced")
    else:
        print( "ERROR: Please specify a correst model")
        return None
    return logreg
```

[15]:
```
def train(x_text, labels, MODEL_TYPE):

    if(WORD):
        print("Using word based features")
        bow_transformer = CountVectorizer(analyzer="word",max_features =␣
↪10000,stop_words='english').fit(x_text)
        comments_bow = bow_transformer.transform(x_text)
        tfidf_transformer = TfidfTransformer(norm = 'l2').fit(comments_bow)
        comments_tfidf = tfidf_transformer.transform(comments_bow)
        features = comments_tfidf
    else:
        print("Using char n-grams based features")
        bow_transformer = CountVectorizer(max_features = 10000, ngram_range =␣
↪(1,2)).fit(x_text)
        comments_bow = bow_transformer.transform(x_text)
        tfidf_transformer = TfidfTransformer(norm = 'l2').fit(comments_bow)
        comments_tfidf = tfidf_transformer.transform(comments_bow)
        features = comments_tfidf
```

```
    if(data == "twitter"):
        dict1 = {'racism':0,'sexism':1,'none':2}
        labels = np.array([dict1[b] for b in labels])

    from collections import Counter
    print(Counter(labels))

    if(MODEL_TYPE != "all"):
        classification_model(features, labels, MODEL_TYPE)
    else:
        for model_type in models:
            classification_model(features, labels, model_type)
```

[16]:
```
data = "formspring"
WORD =  False
x_text, labels = load_data(get_filename(data))
print ("Data loaded!")
train(x_text, labels, MODEL_TYPE)
```

```
Data loaded!
Using char n-grams based features
Counter({0: 11997, 1: 776})
Model Type: svm
Precision Class 1 (avg): 0.466 (+/- 0.109)
Recall Class 1 (avg): 0.503 (+/- 0.122)
F1_score Class 1 (avg): 0.483 (+/- 0.104)
Model Type: naive
Precision Class 1 (avg): 0.850 (+/- 0.640)
Recall Class 1 (avg): 0.015 (+/- 0.015)
F1_score Class 1 (avg): 0.030 (+/- 0.028)
Model Type: lr

/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: F-score
is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

Precision Class 1 (avg): 0.410 (+/- 0.099)
Recall Class 1 (avg): 0.626 (+/- 0.131)
```

```
F1_score Class 1 (avg): 0.495 (+/- 0.104)
Model Type: random_forest
Precision Class 1 (avg): 0.816 (+/- 0.187)
Recall Class 1 (avg): 0.171 (+/- 0.060)
F1_score Class 1 (avg): 0.281 (+/- 0.086)
```

[18]:
```
data = "formspring"
WORD = True
x_text, labels = load_data(get_filename(data))
print ("Data loaded!")
train(x_text, labels, MODEL_TYPE)
```

```
Data loaded!
Using word based features
Counter({0: 11997, 1: 776})
Model Type: svm
Precision Class 1 (avg): 0.415 (+/- 0.089)
Recall Class 1 (avg): 0.525 (+/- 0.132)
F1_score Class 1 (avg): 0.463 (+/- 0.100)
Model Type: naive
Precision Class 1 (avg): 0.575 (+/- 0.950)
Recall Class 1 (avg): 0.013 (+/- 0.029)
F1_score Class 1 (avg): 0.025 (+/- 0.055)
Model Type: lr

/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: F-score
is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: F-score
is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
```

```
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: F-score
is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: F-score
is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/home/sujeendra/miniconda3/envs/tf/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

Precision Class 1 (avg): 0.407 (+/- 0.079)
Recall Class 1 (avg): 0.617 (+/- 0.127)
F1_score Class 1 (avg): 0.489 (+/- 0.084)
Model Type: random_forest
Precision Class 1 (avg): 0.709 (+/- 0.263)
Recall Class 1 (avg): 0.165 (+/- 0.068)
F1_score Class 1 (avg): 0.266 (+/- 0.101)
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: