Relatório ex06

Foi feito o que foi pedido na Parte 1 usando as funções CountVectorizer e TfidfTransformer.

Na **Parte 2**, rodando o Naive Bayes na matriz binária, obtive a acurácia de **0.799601196411**.
Rodando Logistic Regression obtive as seguintes acurácias,
na matriz binária: **0.822532402792**
e na matriz de frequência de termos: **0.830508474576**.

Na **Parte 3**, reduzindo a matriz de frequências para 99% da variância original através do PCA e rodando os dados no **SVM com kernel rbf** e no **Random Forest Regression** obtive as seguintes acurácias:

SVM com kernel rbf: 0.473579262213
Random Forest Regression: 0.339458467028

O código se encontra abaixo.

```python
from sklearn import datasets
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.naive_bayes import MultinomialNB
from sklearn.decomposition import PCA
from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV

from nltk.stem.porter import *

import os
import string
import pandas

def createFrequencyMatrix(matrix):
    tf_transformer = TfidfTransformer(use_idf=False).fit(matrix)
    frequencyMatrix = tf_transformer.transform(matrix)

    return frequencyMatrix

def getIndexes(matrix_data, matrix_target):
    #fold
    skf = StratifiedKFold(n_splits=5)
    for train_index, test_index in skf.split(matrix_data, matrix_target):
        break

    return train_index, test_index

def main():

    # read files
    all_files = datasets.load_files(os.getcwd() + '/filesk', encoding='utf-8')
    categories = pandas.read_csv('category.tab.txt')

    # convert data alphabets to integers
    stemmer = PorterStemmer()

    newData = []
    for text in all_files.data:
        text = string.join(map(lambda v: stemmer.stem(v), text.split(' ')))
        newData += [text]

    all_files.data = newData

    # remove stop_words, punctuation and convert to lowercase
    counterVectorizer = CountVectorizer(encoding='ascii', strip_accents='ascii',
stop_words='english', lowercase=True, min_df=2, binary=True)
```

```
    matrix = counterVectorizer.fit_transform(all_files.data)
    target = all_files.target

    # get train and test indexes
    train_index, test_index = getIndexes(matrix, target)

    #train naive classifier
    train_data, train_result = matrix[train_index], all_files.target[train_index]
    test_data, test_result = matrix[test_index], all_files.target[test_index]
    clf = MultinomialNB().fit(train_data, train_result)
    clf_score = clf.score(test_data, test_result)

    print 'Naive Classifier score on bin matrix: ', clf_score

    lr = LogisticRegression(C=10000)
    lr.fit(train_data, train_result)
    lr_score = lr.score(test_data, test_result)

    print 'Linear Regression score on bin matrix: ', lr_score

    # get frequency matrix
    frequencyMatrix = createFrequencyMatrix(matrix)

    # get data train and test
    train_data, train_result = frequencyMatrix[train_index],
all_files.target[train_index]
    test_data, test_result = frequencyMatrix[test_index],
all_files.target[test_index]

    lr = LogisticRegression(C=10000)
    lr.fit(train_data, train_result)
    fm_lr_score = lr.score(test_data, test_result)

    print 'Linear Regression score on frequency matrix: ', fm_lr_score

    # Reduce frequency matrix with PCA
    print 'Starting PCA...'
    pca = PCA(n_components=0.99)
    reduced_freq_matrix = pca.fit_transform(frequencyMatrix.toarray())

    # run SVM with RBF on reduced matrix
    svc = SVC()
    #train and test with data
    svc.fit(train_data, train_result)
    clf_score = svc.score(test_data, test_result)

    print 'SVM with rbf kernel score on frequency matrix: ', clf_score

    # run RFR on reduced matrix
    rfr = RandomForestRegressor()
    #train and test with data
    rfr.fit(train_data, train_result)
    rfr_score = rfr.score(test_data, test_result)

    print 'Random Forest Regression score on frequency matrix: ', rfr_score

if __name__ == "__main__":
    main()
```