Mini-project 2 README

This zip file contains 5 files: Bernoulli Naive bayes, N grams, merge predictions, Kaggle accuracy, and cross validation.

General Instruction:
When initiating the data sets, we specified the method as “reddit_data = pd.read_csv("reddit_train.csv")” and “reddit_data_test = pd.read_csv("reddit_test.csv")”. Hence if file with such name is not found, it will render the file not executable. Therefore, to ensure that the code runs smoothly, please make sure that the files “reddit_train.csv” and “reddit_test.csv” are accessible for the code. It is achievable by either having both files located in the same folder as the python file to be used, or to modify the code such that it accesses the path of the location of the files. If file of other name is to be run, either rename them or modify the code. Our code only work specifically with this mini project. The cvs file has 3 columns, namely ID, comments and class. There is no empty row. The classes are composed of 20 specific classes. Any file with other format will not work.

Code files:

N grams

This file tests the affect of changing the n-gram parameter using logistic regression model to evaluate accuracy. It outputs an accuracy percentage as well as a metrics classification report.

Libraries used:

- Pandas
- Numpy
- sklearn.model_selection: train_test_split
- sklearn.linear_model: LogisticRegression
- sklearn: metrics
- sklearn.feature_extraction.text: CountVectorizer
- sklearn.preprocessing: normalize
- nltk
- nltk.corpus: stopwords
- nltk.tokenize: word_tokenize

Instructions:
Run each cell in order. The cells are in the following order:

1. initialize data by opening the data file and storing them
2. Preprocess the data by removing stop words and normalizing them
3. Prints an accuracy percentage as well as a metrics classification report

Kaggle accuracy

It contains all the models we used to output our prediction in a csv file.

Libraries used:

- Pandas
- Numpy
- re
- sklearn.model_selection: train_test_split, SelectKBest, chi2, f_classif, f_regression, mutual_info_classif
- sklearn.feature_extraction.text: TfidfVectorizer
- sklearn.naive_bayes: MultinomialNB
- sklearn: metrics, linear_model
- sklearn.svm: LinearSVCLinearSVC
- sklearn.metrics: accuracy_score
- sklearn.neural_network: MLPClassifier
- sklearn.preprocessing: normalize
- nltk
- nltk.corpus: stopwords
- nltk.tokenize: WordPunctTokenizer
- nltk.stem.porter: PorterStemmer
- nltk.stem.wordnet: WordNetLemmatizer

Instructions:

Run each cell in order. The cells are in the following order:

1. Initiating training data and separating them into training set and test sets
2. Preprocess and normalize data by lemmatizing, removing punctuation and stems etc.
3. Feature selection
4. Splitting training set and test set
5. Multinomial Naive Bayes
6. Linear model
7. Linear SVC
8. MLP
9. Initializing test data set
10. Output the prediction results as a csv file. Please rename the file manually if needed

Cell 5 – 8 initializes the parameters for these models. Delete the "#" before the last line of each cell to print prediction results for each individual model on the dataset. Finally, the last cell uses hard voting classifier to takes the class more agreed on by the four models and output the result in a csv file. Renaming might be needed in certain cases (See "Merge prediction").

Merge predictions

sum up the predictions using hard voting classifier. It takes the prediction that is agreed the most over different models and outputs a cvs file storing the predicted classes as strings. This is what we used to generate the final prediction results for the Kaggle competition.

Libraries used:

- Pandas
- Numpy

Instructions:

This file takes as input 4 prediction files, namely "prediction_voted.csv", "prediction_voted.csv2", "prediction_voted3.csv" and "prediction_voted4.csv". The last column of the csv file holds the predictions. We ran "Kaggle accuracy" 4 times, adjusting the name of the output file each time generate the 4 files needed as input. The output is named "merged_prediction.csv" in default. You can change it according do your need.

Cross Validation

It outputs graphs that plots iteration of k-fold validation where k=5 against runtime and iteration against accuracy for the following models: Logistic regression, MNB, linear model, linear SVC. It prints the accuracy, mean accuracy and runtime neural network. It also prints the accuracy of 5 iterations of k-folds validation where k=5 and their average accuracy.

Libraries used:

- Pandas
- Numpy
- sklearn.model_selection: train_test_split
- sklearn.linear_model: LogisticRegression
- sklearn: metrics
- sklearn.feature_extraction.text: CountVectorizer
- sklearn.preprocessing: normalize
- nltk
- nltk.corpus: stopwords
- nltk.tokenize: word_tokenize
- nltk.stem.porter: PorterStemmer
- sklearn.feature_extraction.text: TfidfVectorizer
- nltk.stem.wordnet: WordNetLemmatizer
- import re
- sklearn.preprocessing: normalize
- sklearn.feature_selection: SelectKBest, chi2, f_classif, f_regression, mutual_info_classif
- sklearn.linear_model: LogisticRegression
- sklearn.model_selection: cross_val_score
- timeit

- matplotlib.pyplot as plt
- %matplotlib inline
- sklearn.naive_bayes: MultinomialNB
- sklearn.metrics: accuracy_score
- sklearn: linear_model
- sklearn.svm: LinearSVC
- sklearn: metrics
- sklearn.neural_network: MLPClassifier

Instructions:

Run the cells in order. The cells are in the order of:

1. open the csv files and store them properly.
2. Preprocess the data by lemmatization, removing stopwords, punctuations and etc.
3. Implement feature selection
4. Run logistic regression
5. Run multinomial naïve bayes
6. Run linear model
7. Run linear svc
8. Run multilayer perceptron neural network (MLP)

MLP needs to be terminated manually after 3 to 4 minutes or else it would keeps on running. Leaving it to run for 3 to 4 minutes should be sufficient for what we need. However, if an error message appears, run and again and wait for a longer period of time.
You might need to write the following code: "nltk.download('wordnet')" if it is not downloaded already.

<u>Bernoulli Naive bayes</u>

It outputs prediction as well as the runtime of Bernoulli naïve bayes implemented from scratch.

Libraries used:

- Numpy
- sklearn.feature_extraction.text: CountVectorizer
- nltk
- nltk.corpus: stopwords
- nltk.tokenize: WordPunctTokenizer
- nltk.stem.porter: PorterStemmer
- nltk.stem.wordnet: WordNetLemmatizer
- re
- sklearn.feature_selection: SelectKBest, chi2, f_classif, f_regression, mutual_info_classif
- sklearn.model_selection: train_test_split
- timeit

Instructions:

Run the cells in order. The cells are in the order of:

1. open the csv files and store them properly. Define the algorithm for Bernoulli.
2. Preprocess the data using lemmatization, removing stem and removing punctuation.
3. Implement feature selections
4. Split test and train set
5. Transform the set into numpy arrays
6. Run the model and record and print runtime
7. Print prediction

You might need to write the following code: “nltk.download('wordnet')” if it is not downloaded.