Project 3: Text Classification-Spam and Ham and TF-IDF

Due date: November 12, 2023

Honour Statement

In doing this project, you must adhere to the following honour statement:

Red River College is committed to protecting the integrity of our curriculum ensuring the college continues to add value to our students and industry, while ensuring that students have opportunities to pursue their marks fairly, honestly and ethically.

This includes, but is not limited to, the fact that no collaboration, plagiarism, cheating, unauthorized collaboration, or false representation is permitted on assessments and is a violation of the S4 – Academic Integrity policy.

I understand I am subject to all of the same academic honesty requirements that apply during an in person assessment or online assessment. I understand that by beginning an exam, I accept and agree not to commit any violation of academic integrity.

I understand that there are consequences for violating the policy and as a Red River College student, I will not participate in or condone academic dishonesty.

Project 3: Spam Classifier and TF-IDF scores

Objective: The goal of the project is two fold: successfully implement a spam classifier to the YouTube comments from 5 different YouTube videos; and create code to calculate TF-IDF scores for a list of sentences.

Instructions:

1. From the UCI ML Repository, you will need the following files:

   https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection

2. Alternatively, and easier, the same files are in the LEARN shell.

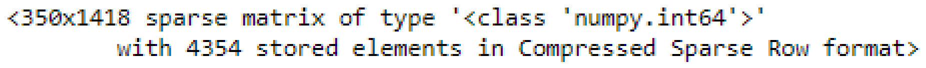
Task 1: Data Exploration

1.1 The Data:

- Read in just the Psy file to your Jupyter Notebook. (1 point)
- Display the head and tail of the file. (1 point)
- Give the number of rows in the file. (1 point)
- Display the number of spam and ham comments for the file. (2 points)
- Discuss the balance of the dataset and the balance of the classes spam/ham. Based on this, what sort of classifiers are appropriate for the task? (3 marks)

1.2 Data Transformation:

- Create a bag-of-words for the Psy file. The fit and transform should be done on the 'CONTENT' of your data. (2 points)
- Display the matrix information. For example. (2 points)



```
<350x1418 sparse matrix of type '<class 'numpy.int64'>'
        with 4354 stored elements in Compressed Sparse Row format>
```

- How many different words are in your bag of words? (1 point)
- Display the 349th comment in the Psy file. (2 points)
- Using 'analyze' ,or something similar, give the breakdown of the 349th comment. (2 points)

2. **Training and Testing Sets**
   - Shuffle your dataset(frac=1). (1 point)
   - Create your training and testing splits by using the first 300 entries for training and the remaining for testing. Name them appropriately. (3 points)
   - Create your training and testing attributes BOW. Name them appropriately. (2 points)
   - Create your training and testing labels. Name them appropriately. (2 points)
   - Output the matrix information of d_train_att and d_test_att. What are the dimensions of the matrices? (3 points)

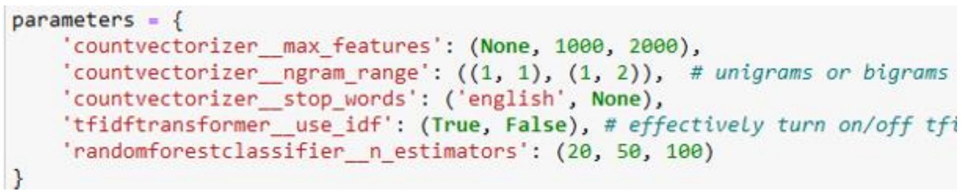
3. **Random Forest Classifier**:
   - Implement a Random Forest classifier. With 50 trees, output. (3 points
   - Train the classifier on the training data and test its performance on the testing data. (2 points)
   - Print the training and testing accuracies. (2 points)
   - Cross validate using 3 folds. Output the accuracies of the folds. (3 points)
   - Generate a confusion matrix for the Random Forest classifier's predictions on the test data. (3 points)

- Visualize this matrix (you may need to implement or use a utility function for visualization). (3 points)

4. **Pipeline Data** Recall that there are five comment files, we would like to compile all of them into a single file to use in the pipeline we create in what follows.
    - Concatenate your 5 files into one using .concat. (3 points)
    - Provided you have concatenated the files correctly the length of your new file should be close to 2000. Output the length. (1 point)
    - Check the spam-to-ham ratio in the new file; they should be balanced- about 1000 each. Output the number of spams and the number of hams. (2 points)
    - Shuffle the new data and create content and label sets, and name them appropriately. (3 points)

5. **Pipeline Creation**:
    - Read the sci-kit learn documentation on Pipelines. Name 3 advantages to using a pipeline. You can use any reference you wish to answer this question, just be sure to include the reference in your response. (5 points)
    - Create a two-step pipeline with a bag-of-words step and a random forest step. (3 points)
    - Output the pipeline to display its steps. (1 point)
    - Fit your pipeline with the first 1500 entries of the content and labels. Output. (3 points)
    - Use .score to score your pipeline. (2 points)
    - Use your pipeline to predict whether the following two comments are spam or ham. (5 points)
        1. “what a neat video”
        2. “plz subscribe to my channel”
    - Cross-validate using your pipeline. Use cv=3. Print out the accuracies. (3 points)

6. **Pipeline 2 Creation**:
    - Create a second pipeline named *pipeline2* which includes a TfidfTransformer step.(3 points)
    - Cross validate *pipeline2* with 3 folds. Output the accuracy.(3 points)
    - Use the following parameter dictionary to perform a grid search. (3 points)



```
parameters = {
    'countvectorizer__max_features': (None, 1000, 2000),
    'countvectorizer__ngram_range': ((1, 1), (1, 2)),  # unigrams or bigrams
    'countvectorizer__stop_words': ('english', None),
    'tfidftransformer__use_idf': (True, False), # effectively turn on/off tfi
    'randomforestclassifier__n_estimators': (20, 50, 100)
}
```

- Perform the grid search, you can reduce n_estimators to 2 values and max_features to 1000 and 2000 to speed things up. (3 points)
- Print out the best parameter settings. (1 point)

7. **TF Calculation**: Write python code that takes as an input a list of sentences and outputs the TF for every term in every sentence.(7 points)

   Input: list of sentences
   Output: the term frequency for every word in every sentence

   Test Cases:

   ```
   sentences = [
                   "Python is a great programming language",
                   "Python can be used for a wide variety of programming tasks",
                   "It's easy to learn Python"
                 ]


   sentences= [ “I love math”,
                   “Math is great”
                   “ Math rules”
                 ]
   ```

8. **IDF Calculation:** Write Python code that calculates the inverse document frequency for a list of sentences. (8 points)

   Inputs: list of sentences
   Outputs: inverse document frequency for each word in the list of sentences.

   Test Cases:

   ```
   sentences = [
                   "Python is a great programming language",
                   "Python can be used for a wide variety of programming tasks",
                   "It's easy to learn Python"
                 ]


   sentences= [ “I love math”,
                   “Math is great”
                   “ Math rules”
                 ]
   ```

Grading Rubric

Project 3 Grading Rubric

Task 1: Data Exploration (Total: 10 Points)

- **1.1 The Data (8 Points)**
  - Read in just the Psy file to your Jupyter Notebook. (1 point)
  - Display the head and tail of the file. (1 point)
  - Give the number of rows in the file. (1 point)
  - Display the number of spam and ham comments for the file. (2 points)
  - Discuss the balance of the dataset and the balance of the classes spam/ham. (3 points)
- **1.2 Data Transformation (7 Points)**
  - Create a bag-of-words for the Psy file. (2 points)
  - Display the matrix information. (2 points)
  - How many different words are in your bag of words? (1 point)
  - Display the 349th comment in the Psy file. (2 points)
  - Breakdown of the 349th comment using ‘analyze’. (2 points)

Task 2: Training and Testing Sets (Total: 11 Points)

- Shuffle your dataset. (1 point)
- Create your training and testing splits. (3 points)
- Create your training and testing attributes BOW. (2 points)
- Create your training and testing labels. (2 points)
- Output the matrix information of d_train_att and d_test_att. (3 points)

Task 3: Random Forest Classifier (Total: 16 Points)

- Implement a Random Forest classifier with 50 trees. (3 points)
- Train the classifier on the training data and test its performance. (2 points)
- Print the training and testing accuracies. (2 points)
- Cross-validate using 3 folds. (3 points)
- Generate a confusion matrix. (3 points)
- Visualize the confusion matrix. (3 points)

Task 4: Pipeline Data (Total: 9 Points)

- Concatenate the 5 files into one. (3 points)
- Output the length of the new concatenated file. (1 point)
- Check and output the spam to ham ratio. (2 points)
- Shuffle the new data and create content and label sets. (3 points)

Task 5: Pipeline Creation (Total: 19 Points)

- List 3 advantages of using a pipeline with references. (5 points)
- Create a two-step pipeline. (3 points)
- Output the pipeline to display its steps. (1 point)
- Fit the pipeline with the first 1500 entries. (3 points)
- Score the pipeline. (2 points)
- Use the pipeline to predict spam/ham for two comments. (5 points)
- Cross-validate using the pipeline. (3 points)

Task 6: Pipeline 2 Creation (Total: 9 Points)

- Create a second pipeline with a TfidfTransformer step. (3 points)
- Cross validate pipeline2. (3 points)
- Use the parameter dictionary to perform a grid search. (3 points)
- Print out the best parameter settings. (1 point)

Task 7: TF Calculation (Total: 8 Points)

- Write Python code to output the TF for every term in every sentence. (8 points)

Task 8: IDF Calculation (Total: 8 Points)

- Write Python code to calculate and output the IDF for each word in the list of sentences. (8 points)

Total Possible Points: 90 Points

Notes:

- The rubric is structured to assign points to specific deliverables within each task.
- The rubric assumes a total of 90 points for the project, with each bullet point corresponding to a subtask carrying a designated point value.
- Clear and thorough execution of each task is required for full points.
- Partial points may be awarded for incomplete or partially correct submissions as deemed appropriate by the instructor.
- In case your grid searches are taking too much time you can reduce the parameters.

Submission: Submit your Jupyter notebook with your working code and answers to the Dropbox before the due date.