Wright State University
Department of Computer Science and Engineering

CS7800 Spring 2022

Prasad

Assignment 2 (Due: April 9) (No late submissions accepted) (10 pts)

In this project, you will have the opportunity to implement various classifiers both manually and **using scikit-learn** APIs in Python. The project comprises two phases. During Phase I, you will construct classifiers and apply them to the modified 20 Newsgroup dataset. In Phase II, you will evaluate these classifiers using metrics such as accuracy and confusion matrix to determine their prediction quality.

To foster discussion and insight, teams of two or three members, as before, may collaborate on the project. Each team member is expected to contribute to all facets of the project, including design, implementation, documentation, and testing, for their own benefit.

Phase I

Scikit-learn offers a sophisticated range of APIs that facilitate the construction of models using regression, classification, and clustering techniques. The library has been widely adopted in various applications and domains for predictive tasks. With its user-friendly interface and extensive documentation, scikit-learn allows developers and data scientists to focus on the problem at hand and easily build accurate and reliable models.

Classification on Newsgroups

In this project, you will be working with a subset of the **20 Newsgroups** dataset, which is commonly used for experimentation in text-based machine learning techniques such as text classification and text clustering. The full dataset comprises 20,000 newsgroup documents, divided evenly among 20 different newsgroups. However, for this assignment, you will work with a pre-processed subset of 1000 documents and a vocabulary of 5,500 terms.

The dataset has already undergone text pre-processing, including parsing and converting text into a sequence of terms. The term-document matrix representation of the dataset is provided as input, and each document belongs to one of two classes: Hockey (class label 1) and Microsoft Windows (class label 0). The data has been split into training and test data with an 80-20 percent split. The training and test data, in **`term x document`** format, contain a row for each term in the vocabulary and a column for each document. The values in the table represent raw term occurrence counts. The data has already been preprocessed to extract tokens, remove stop words, and perform stemming, so the terms in the vocabulary are stems and not full terms. Please consult the readme.txt file in the distribution for further details.

In this assignment, your main goal is to implement and evaluate the performance of various classifiers on the 20 Newsgroups dataset. Specifically, you will use scikit-learn to apply Naive Bayes (NB) and Support Vector Machine (SVM) classifiers via APIs, as well as manually code the K-Nearest-Neighbor (kNN) classifier using cosine similarity as the distance metric. To parameterize kNN, you can choose a reasonable value for k, but you will not be required

to write an algorithm that finds the optimal k value. Instead, you should manually experiment with different k values to obtain good results. Note that for SVM, you should use the linear classifier in scikit-learn.

To ensure consistency and avoid potential issues, you are only allowed to use Pandas, NumPy, standard Python libraries, and Matplotlib in your programs. Your main objective is to evaluate the accuracy and confusion matrix for each classifier using the provided dataset. To calculate the confusion matrix, you can use scikit-learn's built-in function, so you do not need to write this part manually.

In summary, your task is to implement and evaluate three classifiers on the 20 Newsgroups dataset, namely NB, SVM, and kNN using cosine similarity as the "distance" metric. You should experiment with different values of k for kNN and report the accuracy and confusion matrix for each classifier using scikit-learn's built-in functions. Your implementation should only use Pandas, NumPy, standard Python libraries, and Matplotlib to ensure consistency and avoid potential issues and point reductions in grading.

Milestones for Phase I:

- Familiarize yourself with the scikit-learn library and its various APIs for regression, classification, and clustering techniques.
- Understand the problem statement and dataset details for the 20 Newsgroups dataset
- Implement the K-Nearest-Neighbor (kNN) classifier manually using Python and NumPy libraries.
- Apply the Naive Bayes (NB) and Support Vector Machine (SVM) classifiers using scikit-learn APIs.
- Evaluate the performance of each classifier in terms of accuracy and confusion matrix using the provided dataset.
- Use Matplotlib libraries to visualize the results and provide insights into the performance of each classifier.
- Document the implementation and results in a well-organized report, including a detailed explanation of the approach, code snippets, and visualizations. Report should be called Evaluation.pdf

Phase I resources

- Dataset: 2Newsgroup.zip provided by the professor
- Relevant tutorials and descriptions
  - https://scikit-learn.org/stable/supervised_learning.html
  - https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
  - https://scikit-learn.org/stable/datasets/real_world.html

Phase I environment setup

- Python 3 or higher
- Install scikit-learn:
  - https://www.activestate.com/resources/quick-reads/how-to-install-scikit-learn/
- Name your well-documented Python script must be named **assignment2.py.**
- You are required to use a Python virtual environment.
  - For those unfamiliar with Python virtual environments here are two reference articles explaining how to use and activate:
    - https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/26/python-virtual-env/
    - https://realpython.com/lessons/creating-virtual-environment/

- When you are ready to submit the completed assignment to Pilot, you need to create a **requirements.txt** file listing all the libraries used in your development.
- Use the following command only after you have activated your Python environment.
  - source venv/bin/activate
  - pip list --format=freeze requirements.txt
  - deactivate
  - https://openclassrooms.com/en/courses/6900846-set-up-a-python-environment/6990546-manage-virtual-environments-using-requirements-files

Phase II

To better understand and quantify the quality of classifier predictions, various evaluation metrics are used. These metrics provide insights into how well a classifier is performing and can be used to compare different classifiers or to fine-tune the parameters of a specific classifier. Some commonly used evaluation metrics for classifier predictions include accuracy, precision, recall, and F1-score. Accuracy measures the proportion of correct predictions out of all predictions made. It is the most basic metric and is often used as a starting point for evaluation. However, it can be misleading when dealing with imbalanced datasets.

- **Accuracy** measures the proportion of correct predictions out of all predictions made. It is the most basic metric and is often used as a starting point for evaluation. However, it can be misleading when dealing with imbalanced datasets.
- **Precision** measures the proportion of true positive predictions out of all positive predictions made. It is a useful metric when the cost of false positives is high.
- **Recall** measures the proportion of true positive predictions out of all actual positive cases in the dataset. It is a useful metric when the cost of false negatives is high.
- **F1-score** is the harmonic mean of precision and recall. It provides a balanced evaluation of a classifier's performance and is particularly useful when dealing with imbalanced datasets.

For this section of the assignment, please provide a brief overview of the evaluation metrics utilized to clarify and quantify the accuracy of the classifier's predictions.

Milestones for Phase II

- For each classifier, at the minimum, provide confusion matrix, accuracy, precision, recall and • Recall
- Compare and contrast K-Nearest-Neighbor (kNN), Naive Bayes and SVM.

Deliverables

Please upload one tar archive file call `assignment2.tgz` per team, which should include all the files created during Phases 1 and 2

- **Code and accompanying documentation:** Include well-documented source code for the entire project.
- **Evaluation information:** Provide comparative analysis of the different classifier performance using the chosen evaluation metrics in a PDF document called **Evaluation**

- **ReadMe.txt:** Briefly explain your application, how to launch the application, any external libraries used, all team members names and UIDs, and any other relevant information. The more information you provide in this document the easier it is to grade and give partial credit if something does not work. *Make sure to include names and email addresses of all team members*.
- **requirements.txt:** All the libraries used in your virtual Python environment must be stated

- **Input/Output files:** Read the input data in the order given: trainMatrixModified.txt, trainClasses.txt, testMatrixModified.txt, testClasses.txt, modifiedterms.txt and generate output data per classifier as: kNN.txt, NB.txt, SVM.txt, ...
- **Application execution:** To ensure successful execution of your Python program, please follow these guidelines:

    1. Ensure that your Python program reads inputs internally in the order outlined in the assignment.
    2. Your Python program must be executable from the command-line.
    3. Use the command "`python3 assignment2.py`" to run your code.
    4. Avoid using absolute paths for input or data files. Instead, use local and relative paths from your working directory.
    5. We recommend each team member participates in the development and testing of the application separately, on multiple computers/installations, to ensure everything works and there are no hard-coded elements.
    6. Do not use Jupyter notebook; your application must be launchable from the command-line with the syntax provided above.
    7. Any deviation from these guidelines will result in a 25% penalty.

    Additionally, make sure all the outputs are saved in your (working) directory.

- **File Location:** To ensure proper submission of your assignment, please adhere to the following guidelines:

    1. All files, including any generated output files, must be placed in your working directory.
    2. Please create a subdirectory within your working directory called "backup" and keep verbatim copies of all classifier outputs in that directory.

- **TAR archive:** Build the project file you submit using the following command to tar up your working directory:

  - `tar -zcvf assignment2.tgz yourWorkDirectoryName/`

- **Uploading to Pilot:** To submit your assignment, followthese steps:

  1. Compress your assignment as an archive file named "assignment2.tgz".
  2. Upload the archive file to Pilot > Dropbox > Assignment 2 folder no later than April 9th.
  3. Ensure that your team submits only one file and includes the names and UIDs of all team members in the dropbox.

  If required, be prepared to demo your program to the instructors and answer questions related to its design, implementation, and comparative evaluation.

- **Caution:** Points will be deducted if you do not follow the naming conventions and use standard file extensions, or if your program does not run in the terminal/command line. **Use/monitor *dis- cord* to seek clarifications and for updates.** If there are several small changes due to these discussions, I may even email an updated assignment consolidating all the changes.

Grading Criteria

You must obtain a PASS on this assignment to PASS the course. At the minimum, your code should compile, process the dataset, and return reasonable results for at least one classifier.

Assignments are designed to help you learn the core concepts and are the primary course "homework". Corrupt files or other computer problems will not be considered a valid excuse to extend the deadline. It is your responsibility to regularly back-up your work. We strongly suggest that you save your work to multiple locations to aid in the recovery of corrupt files. If you have questions regarding the project, we (the GTA, the grader and the instructor) are there to help.

Assignments that are submitted late will incur a penalty of 25% reduction on total grade per day the assignment is late. The project must be turned in on Pilot as described in the project description to receive full credit. Assignments emailed to the GTA, or professor will receive an immediate 25% reduction in total grade because the whole purpose of Pilot is to streamline communication.

Cheating:

Please do not copy other team's work, do not copy other projects found on the Internet or use any outside resource without proper citation. In short, plagiarism will get you a zero on this assignment and potentially an F grade in the class. We are not obsessed with looking for cheating, but if we see something suspicious, we will investigate and then refer it to the Office of Judicial Affairs. This is more work for us and is embarrassing for everyone. Again, please don't; this has been a problem in the past. If the rules are unclear or you are unsure of how they apply, ask the instructor, GTA, or grader beforehand. The academic integrity policy as available online.

No deadline extension will be given.