

**Assignment 2 (Due: April 11) (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 classification 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: Construction and Application of Classifiers**

In the first phase of the project, you will focus on building classifiers and applying them to a tailored subset of the 20 Newsgroups dataset, a well-known collection perfect for text-based machine learning explorations, such as text classification and clustering. This dataset contains 20,000 documents evenly distributed across 20 newsgroups. You will use a pre-processed subset of 1,000 documents and a vocabulary comprising 5,500 terms. The dataset has undergone preliminary text processing, including parsing and conversion of text into a sequence of terms. The provided term-document matrix, which organizes the vocabulary as rows and documents as columns, shows term occurrence frequencies. Each document is categorized into one of two classes: Hockey (class label 1) or Microsoft Windows (class label 0), with the dataset divided into training and test segments in an 80-20 split. Preprocessing has simplified the terms in the vocabulary to their stems by tokenizing, removing stop words, and stemming. Refer to the provided `readme.txt` file for more details.

For data handling, `np.loadtxt` is recommended due to its capability in managing numerical data and features like `transpose()` that assist in organizing the dataset for analysis correctly.

Your primary objective is to implement and assess various classification algorithms on the 20 Newsgroups dataset, specifically utilizing K-Nearest-Neighbor (kNN) and Support Vector Machine (SVM) classifiers through scikit-learn APIs, and manually coding the Naive Bayes (NB) classifier. Your Multinomial Naive Bayes (MNB) text classifier must be built from scratch, exactly following the methodology outlined in Section 13.2 of your textbook and covered in class. This approach requires adherence to the provided pseudocode and guidelines, with a strict prohibition on using online resources, code snippets, or libraries for direct implementation or enhancement. Your implementation should strictly mirror the pseudocode discussed, and deviations will result in penalties. This requirement is to ensure the integrity of your academic work and highlight the importance of building machine learning algorithms from fundamental principles.

Here's a summary of the Multinomial Naive Bayes algorithm from Figure 13.2 in your textbook (to be adapted to the available dataset):

1. **Training the Classifier (TRAINMULTINOMIALNB):**

- Extract the vocabulary (unique words) from all documents in the training set ( $D$ ).
- Count the total number of documents ( $N$ ).
- For each class  $c$  in the set of classes  $C$ :
  - Count the number of documents in class  $c$  ( $N_c$ ).
  - Calculate the prior probability of class  $c$  ( $\text{prior}[c] = N_c / N$ ).
  - Concatenate the text of all documents in class  $c$ .
  - For each term  $t$  in the vocabulary:
    - Count the occurrences of  $t$  in the concatenated text for class  $c$  ( $T_{ct}$ ).
  - Calculate the conditional probability for each term  $t$  in class  $c$ , with add-one smoothing:  $\text{condprob}[t][c] = (T_{ct} + 1) / (\sum (T_{ct'} + 1))$  for all  $t'$  in the vocabulary.
- Return the vocabulary, priors, and conditional probabilities.

2. **Applying the Classifier (APPLYMULTINOMIALNB):**

- Extract the tokens from the document  $d$  to be classified, using the vocabulary.
- Initialize a score for each class  $c$  in  $C$ .
- For each class  $c$ :
  - Start with the log of the prior probability of  $c$  ( $\text{score}[c] = \log(\text{prior}[c])$ ).
  - For each term  $t$  in the document:
    - Add the log of the conditional probability of  $t$  given  $c$  to the score of  $c$  ( $\text{score}[c] += \log(\text{condprob}[t][c])$ ).
- Determine the class with the highest score and return it as the classification result.

It is crucial to pay careful attention to specific details, such as the application of add-one smoothing in the calculation of conditional probabilities and the utilization of logarithms to prevent underflow in probability computations.

For the k-Nearest Neighbors (kNN) classifier, you should start by selecting a reasonable value for  $k$  through experimentation. It's not necessary to find the optimal  $k$  value, but trying out different values will help you understand how  $k$  affects the classifier's performance. The similarity measure to use for kNN is cosine similarity, which is effective for high-dimensional data like text because it measures the cosine of the angle between two vectors, thus focusing on the orientation rather than the magnitude of the vectors. This approach is particularly suitable for text classification where your vectors are term-frequency vectors from the term-document matrix.

For the Support Vector Machine (SVM) classifier, you should use the linear SVM implementation provided by scikit-learn. The linear SVM is well-suited for text classification tasks, as text data often results in high-dimensional feature spaces where linear models can perform quite well.

In your project, you must restrict your primitives to NumPy for numerical calculations, standard Python libraries for general programming tasks, and Matplotlib for any data visualization needs. The use of NLTK or any non-standard libraries is not allowed and will result in a zero for the assignment. This restriction ensures that you engage with the data and the machine learning algorithms at a fundamental level, without relying on higher-level abstractions provided by specialized libraries.

Your primary task is to evaluate the performance of each classifier—kNN, SVM, and your

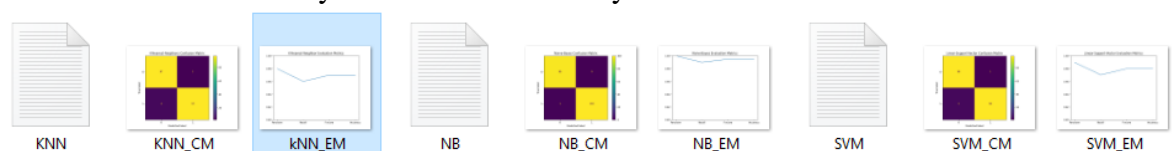
manually implemented Naive Bayes—on the provided dataset. Specifically, you should assess the accuracy and analyze the confusion matrix for each classifier. Accuracy will give you a straightforward metric of how often the classifier makes the correct prediction, while the confusion matrix will provide deeper insights into the types of errors the classifier is making, such as confusing one class for another.

To streamline your evaluation process, you can utilize scikit-learn's built-in functions for generating confusion matrices and calculating accuracy. These tools will allow you to focus on the implementation and comparison of the classifiers rather than on the mechanics of these evaluation metrics.

In summary, your project involves experimenting with  $k$  values for kNN using cosine similarity, employing a linear SVM from scikit-learn, and carefully implementing and testing your classifiers. By evaluating each classifier's accuracy and confusion matrix, you'll gain valuable insights into their strengths and weaknesses in the context of text classification on the 20 Newsgroups dataset.

### Milestones for Phase I:

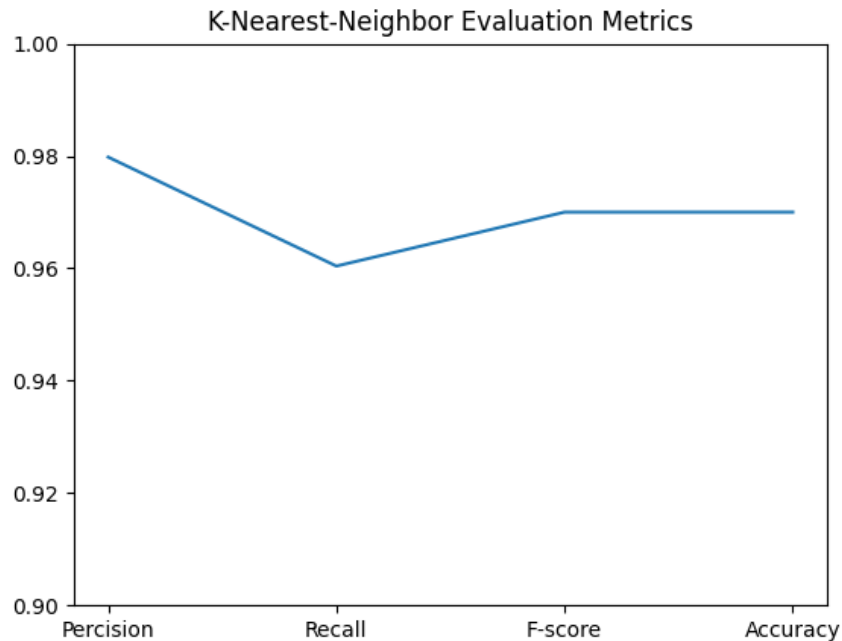
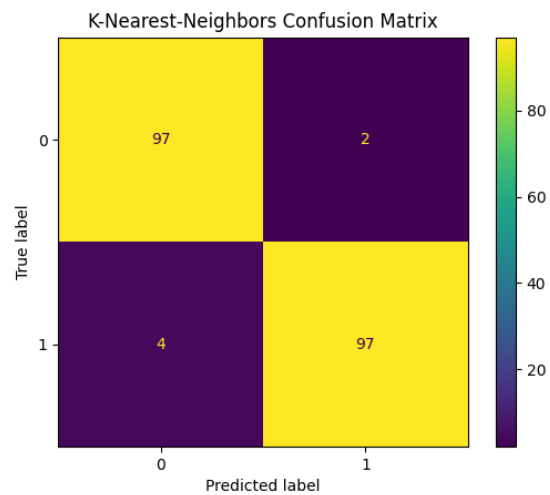
- Familiarize yourself with the scikit-learn library and its comprehensive APIs for regression, classification, and clustering techniques.
- Gain a deep understanding of the problem statement and the specifics of the 20 Newsgroups dataset.
- Thoroughly read Section 13.2 of the textbook to grasp the theoretical foundation, methodology, and implementation nuances of the Multinomial Naive Bayes (MNB) classifier.
- Manually implement the Multinomial Naive Bayes (MNB) classifier using Python and the NumPy library, adhering closely to the principles outlined in the textbook.
- Utilize scikit-learn APIs to apply the k-Nearest Neighbors (kNN) and Support Vector Machine (SVM) classifiers.
- Evaluate the performance of each classifier—MNB, kNN, and SVM—in terms of accuracy and confusion matrix using the provided dataset.
- Employ Matplotlib to create visualizations that highlight the results and shed light on the performance of each classifier.
- Compile the implementation and findings into a coherent report titled "Evaluation.pdf". This report should encompass detailed methodology, code snippets, and visualizations. It must include three confusion matrix graphs and three graphs illustrating the evaluation metrics for comprehensive analysis, and text document results.
- In total, you need to generate six graphs and three text documents. The figure below demonstrates exactly what they should be named.



- Ensure the presentation of the text documents adheres precisely to the specified format. Similarly, the confusion matrices and evaluation graphs must match the provided examples in form. Below are figures demonstrating the exact format required. Note that the outputs depicted in these examples might differ from your results. However, it is crucial not to deviate from this format, as a grading script will be used to assess your submissions.

### KNN - Notepad

```
File Edit Format View Help
Accuracy:0.97
Precision:0.9603960396039604
Recall:0.9797979797979798
F1:0.9700000000000001
Confusion Matrix:
97 2
4 97
```



### 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/supervised_learning.html)
  - [https://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)
  - [https://scikit-learn.org/stable/datasets/real\\_world.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-ereseach.github.io/ereseach-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 classifier performance, various evaluation metrics are used. These metrics can be used to compare different classifiers or to fine-tune the parameters of a specific classifier. Some commonly used evaluation metrics for classifier performance 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.
- **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.
- **F-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 and precision-recall trade-offs.

For this section of the assignment, please provide a brief overview of the classifier's performance on different classes explaining it in terms of evaluation metrics.

### Milestones for Phase II

- For each classifier, at the minimum, provide confusion matrix, accuracy, precision, recall and F-score.
- Compare and contrast K-Nearest-Neighbor (kNN), Naive Bayes and SVM on the given dataset.

## 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.pdf
- **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.

- **Archive:** Only archive your files in a zip format.
- **Virtual Environment:** Ensure that your virtual environment folder is not included in your projects submissions. Including it can significantly increase the size of your file, so make sure to remove it or exclude it from the compression process. Points will be deducted for including this folder. The purpose of having a `requirements.txt` document is so that we can generate our own environment if needed.
- **Uploading to Pilot: To submit your assignment, follow these steps:**
  1. Before compressing your assignment, make sure to remove the environment (venv) folder.
  2. Compress your assignment into an archive file named "assignment2.zip".
  3. Upload the archive file to Pilot > Dropbox > Assignment 2 folder no later than April 9th.
  4. 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 discord 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.
- Below, we provide an illustration of your folder structure in its initial state, showing all the existing files.

 .vscode	3/11/2024 3:48 PM	File folder	
 __pycache__	3/11/2024 3:48 PM	File folder	
 2Newsgroups	3/4/2024 3:01 PM	File folder	
 venv	3/11/2024 3:48 PM	File folder	
 assignment2	3/11/2024 6:30 PM	Python File	6 KB

- **Environment setup requirements:** Any deviation from what was discussed in this section will result in a 30% reduction for each deviation in your total grade. Therefore, I strongly suggest you make sure you follow this document closely and ask questions on *discord* for clarification.

## 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, do not blindly use AI assistance (ChatGPT/Copilot), 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 is available [online](#).

No deadline extension will be given.