# Assignment

In this project, you will implement several different classifiers in Python using **scikitlearn** APIs. The project has two phases. In Phase I, you will build classifiers and run them on the same modified 20 Newsgroup dataset. In Phase II, you will evaluate these classifiers to elucidate their quality of predictions (e.g., using accuracy and confusion matrix).

The project may be done in a team of two or three members like before, to promote discussions and insights. If a fourth member is included because someone cannot find a teammate, the team must also implement and evaluate two of the classifiers (e.g., kNN and Rocchio) manually from scratch as required below. All the team members are expected to contribute to all aspects of the project: design, implementation, documentation, and testing, for their own good.

# Phase I

**scikit-learn** provides a mature set of APIs for building models using regression, classification and clustering techniques, and has been used extensively for prediction tasks.

## Classification on Newsgroups

For this project, you will use a subset of the 20 Newsgroups dataset. The full data set contains 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups and has been used for experiments in text applications of machine learning techniques, such as text classification and text clustering. This assignment dataset contains a pre-processed subset of 1000 documents and a vocabulary (dictionary) of 5,500 terms. As you are already familiar with the text preprocessing pipeline for parsing and converting text into sequence of terms, we are providing associated term-document matrix representation of the dataset as input. Each document belongs to one of two classes Hockey (class label 1) and Microsoft Windows (class label 0). The data has already been split (80%, 20%) into training data and test data. The class labels for training data and test data are also provided in separate files. The training data and test data, in term x document format, contains 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, not full terms). Please be sure to read the readme.txt file in the distribution. Your task is to exercise several different classifiers available in **scikit-learn** for classification on the given 2Newsgroups dataset. Specifically, you must use K-Nearest-Neighbor (kNN), Centroid-based Rocchio Method, Naive Bayes (NB) and Support Vector Machine (SVM) classifiers. You may additionally use Pandas, NumPy, standard Python libraries, and Matplotlib in your programs.

## Milestones for Phase I

* Read and understand the input dataset format.
* Implement a program to exercise the different classifiers given.
* For a four member team: implement K-Nearest-Neighbor (kNN) and Centroidbased Rocchio classifiers from scratch (that is, without using scikit-learn APIs) in Python.

## Phase I resources

* Dataset: [2Newsgroups.zip](https://web1.cs.wright.edu/~tkprasad/courses/cs7800/asg-R22/2Newsgroups.zip)
* Relevant tutorials and descriptions
* See:<https://scikit-learn.org/stable/supervised_learning.html>
* See: [https://scikithttps://scikit-learn.org/stable/tutorial/text\_analytics/working\_with\_text\_data.htmllearn.org/stable/tutorial/text\_analytics/working\_with\_text\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)
* See:<https://scikit-learn.org/stable/datasets/real_world.html>

## Phase I environment setup

* Python 3 or higher
* **Do not use Jupiter notebook**
* Install scikit-learn:
* See: [https://www.activestate.com/resources/quick-reads/how-to-install-scikithttps://www.activestate.com/resources/quick-reads/how-to-install-scikit-learn/learn/](https://www.activestate.com/resources/quick-reads/how-to-install-scikit-learn/)
* Name your well-documented Python script as **assignment2.py**.
* Students 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:
* See: [https://uoa-eresearch.github.io/eresearchhttps://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/26/python-virtual-env/cookbook/recipe/2014/11/26/python-virtual-env/](https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/26/python-virtual-env/)
* See:<https://realpython.com/lessons/creating-virtual-environment/>
* Once you finish this assignment and are ready to submit it to pilot, you need to create a **requirements.txt** file with all the libraries used in your development. Use the following command only after you’ve activated your Python environment.
* source venv/bin/activate
* pip list --format=freeze > requirements.txt
* deactivate
* See: [https://openclassrooms.com/en/courses/6900846-set-up-a-python- environment/6990546-manage-virtual-environments-using-requirements- files](https://openclassrooms.com/en/courses/6900846-set-up-a-python-environment/6990546-manage-virtual-environments-using-requirements-files)

# Phase II

Briefly discuss the evaluation metrics to be used to elucidate and quantify the quality of classifier pre- dictions.

## Milestones for Phase II

* For each classifier, at the minimum, provide confusion matrix and accuracy.
* For a four-member team: additionally, evaluate and compare K-Nearest-Neighbor (kNN) and Centroid-based Rocchio classifiers implemented from scratch with those implemented using scikit- learn APIs in Python.

## Deliverables

**TURN IN**: Upload one tar archive **file** per team that contains the following files:

* **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.
* **README.txt:** This document should briefly explain your application, how to launch the application, any external libraries used, what version **of Python 3 used**, all team members names and UIDs, and any other relevant information. The more information you provide in this document the easier it is for us to grade and give extra credit if something does not work on our system. Additionally, all team member names, and email addresses must be included in this document.
* **Application execution**: Make sure that your Python program runs from the commandline. We will use the following command to execute your code: **python3 assignment2.py**
* Do not use any **absolute path** for input files or any data files. All paths should be local to your working directory. We suggest testing your application on one of your teammates computers to make sure everything works, and you did not hardcode anything. Points will be deducted if you use absolute paths for your input and output files.
* Do not use Jupyter notebook, as your application must launch from the command-line with the above given syntax.
* **Input and output files**: All file(s) must be placed in your working directory, all generated output file(s) must also be placed in your working directory, and no subdirectories.
* **requirements.txt:** all the libraries used in your virtual Python environment.
* **TAR archive**: Submit your application using the following command exactly as written to tar up your working directory: “tar -zcvf assignment2.tgz yourWorkDirectoryName/”
* **Upload** the archive asg1.zip onto Pilot $>$ Dropbox $>$ Assignment 2 folder. (Only one submission per team.) You are also expected to demo your program to us (if necessary) and be prepared to answer questions about its design, implementation, and comparative evaluation.