CSC 503/SENG 474: Assignment 3

Due on: Friday, Dec 2nd at 23:59 PST
Where: Brightspace (`https://bright.uvic.ca/d2l/home/244947`)

Instructions:

- You must complete this assignment *entirely* on your own. In other words, you should come up with the solution *yourself*, write the code *yourself,* conduct the experiments *yourself*, analyze the results *yourself*, and finally, write it all solely by *yourself*. The university policies on academic dishonesty (a.k.a. cheating) will be taken very seriously.

- This does not mean that you need to go to a cave and self-isolate while preparing the assignment. You are allowed to have high-level discussions with your classmates about the course material. You are also more than welcome to use Piazza or come to office hours and ask questions. If in doubt, ask!— we are here to help.

- If you are still stuck, you can use books and *published* online material (i.e., material that has a fixed URL). However, **you must *explicitly* credit all sources**. **You are also not allowed to copy-paste online materials**. This includes "slightly" adapting the online code to fit your problem. Woe to you if you are caught doing this!

  - Why "if stuck"? Assignments are designed to develop your practical ML skills and make you strong. If you do the assignments well, the project will feel like a piece of cake. So, give your best. But, on the other hand, do not waste a whole week on a single question: if you are stuck on a question for a few days, ask (us) for help!

- **No extensions will be granted this time!** Urgent accommodations might be negotiated if explicitly approved by the instructor at least 7 days before the deadline.

- Remember: **you will need to gather at least one-third of all points during the assignments to pass the course. If you don't, you will get an F!**

- Make sure to follow the technical requirements outlined below. TAs can (and will) take 50% off your grade if you disregard some of them.

- Be sure that your answers are clear and easy for TAs to understand. They can penalize you if your solutions lack clarity or are convoluted (in a non-algebraic way), even if they are nominally correct.

- We will try to grade your assignments within seven (7) days of the initial submission deadline.

- If you think there is a problem with your grade, you have one week to raise concern after the grades go public. Grading TAs will be holding office hours during those seven days to address any such problems. After that, your grade is set in stone.

Technical matters:

- You must type up your analysis and solutions *electronically* and submit them as a self-containing Jupyter notebook. Jupyter notebooks can contain code, its output, and images. They can also be used to type math and proofs in LaTeX mode.

  - You ***must*** use LaTeX mode to type formulas. Typing `a^2=sqrt(3)+b1` is a pretty good way to lose 50% of your grade for no good reason.

- Each problem should be submitted as a separate file.
- Each file should be named `VNumber_SurnameInitial_N.ipynb`, where `N` is two digit-padded problem number.
  - **Correct:** `V00000000_SmithJ_05.ipynb`.
  - **Incorrect:** `JohnSmith_V12345 Problem 1.ipynb`, `prob1.pdf` etc.
- Zip all `ipynb` files and submit them as `assignment3.zip` to the Brightspace. Do not put Jupyter files into a directory—they have to be in the root directory of the submitted ZIP file. Do not submit RAR, TAR, 7zip, SHAR and whatnot; just use good ol' ZIP. Do not include other files.
  - You can validate your submission by downloading `https://gist.github.com/inumanag/56ae2ca04b7357530c3aeb2b63699fed` and running it as follows: `python3 validate.py assignment3.zip`. This script will complain if you violate submission rules. Use it to polish your submission—if you submit something that the script does not like, you risk losing 50% of your grade.
- The first cell of each Jupyter notebook must start with your name and V number. See the attached notebook for the details.
- Your notebook should be organized sequentially according to the problem statement. Use sections (with the appropriate numbers and labels) within the notebook. Figures and relevant code should be placed in the proper location in the document.
- Notebook code ***must be runnable***! Ideally, all answers will be the output of a code cell.
- Make sure that all images are *embedded* within the notebook (i.e., we cannot see an image that points to `C:\WIN311\THISSU~1.BMP`).
- You must use **Python 3** to complete the assignments. Feel free to use NumPy and `pandas` as you find it fit. Use SciPy, `scikit-learn`, and other non-standard libraries **only** when explicitly allowed to do so.
- Your first executable cell should set the random seed to 1337 to ensure the reproducibility of your results. For Numpy/SciPy and `pandas`, use `numpy.random.seed(1337)`; otherwise, use `random.seed(1337)`.
- Document your code! Use either Markdown cells or Python comments to let us know what you have done!
- Finally, **be concise**! We do not appreciate long essays that amount to basically nothing.

This assignment consists of 4 problems. Some are intended only for graduate students (those taking CSC 503), and are labelled as such. Some contain bonus sections: you can use bonus points to improve your overall homework score. Bonus points cannot be transferred to other assignments or the final project. Any graduate-level problem counts as a bonus problem for the undergraduate students.

Some problems are purposefully open-ended. Whatever you think a correct answer is, make sure to support it with code and data.

Problem 1. Multi-dimensional coat [3 points]

The good ol' MNIST dataset is nowadays deemed too boring and too easy. Thus a replacement was born—the Fashion-MNIST. This dataset throws away the digits and instead uses shirts, sandals and other fashion items. Let's play with it.

1. [**Dataset; 0.5 points**] Get the Fashion-MNIST dataset from `https://github.com/zalando research/fashion-mnist`. For this assignment, you can get either the training or the test data. As either dataset is quite large, downsample it to 5,000 elements. Make sure that each class contains 500 items.

2. [**Dimensionality; 1 point**] Run PCA on this dataset. Plot the top 4 components. What are they standing for, if anything? How many components do you need to explain 75% of the variance? Let this number be d. Pick some examples (say, 3 of them) randomly from the dataset, and show the difference between their projection to d-dimensional space and the original image.

3. [**Plot; 0.5 points**] Now select the two topmost principal components. Use them to plot your dataset as a 2D scatterplot. Use the label vectors to colour different classes on the plot as well. Would you be able to distinguish classes without knowing the colours in advance?

4. [**SNE; 1 point**] Now run t-SNE with parameters of your choice on this dataset. Again, plot the result of t-SNE as a scatterplot, and use the label vectors to colour different classes. Can you distinguish classes now? Is this plot more informative than the PCA-based one?

You are free to use any library of your choice (e.g. `sklearn`) to complete this problem. Discuss the results of all steps, and use plots to support your findings.

Problem 2. Bundles [4 points]

In this problem, we will use the d-dimensional PCA projections of Fashion-MNIST images obtained in the previous problem. Implement the following clustering algorithms on this projected dataset:

1. [**Lloyd; 2.5 points**] Implement Lloyd's algorithm from scratch. Use the Euclidean distance, and use the following two initialization techniques:

   - **random initialization** [1 point]: the centers are initialized to a set of k distinct examples from the dataset drawn uniformly at random; and
   - **k-means++ initialization** [1.5 points]: check the slides for details. You must implement this step yourself as well.

   For each version of Lloyd's algorithm, try different values of k (from 2 to 15). For each value of k, you can run the clustering multiple times and pick the best result. Make a plot of the cost (sum of the squared errors from each point to its cluster center) as k increases. Using the plot, decide how many clusters to use, and explain your choice (hint: the lecture slides might be helpful!). How well does the best clustering correspond to the true labels? Do classes always entirely fall within a single cluster (if $k \leq 10$) or not? What happens if $k = 10$?

2. [**Hague; 1.5 points**] Run hierarchical agglomerative clustering on this dataset. You should use Euclidean distance for the dissimilarity measure between the two examples. For the dissimilarity measure between clusters, you should use both single linkage and average linkage. You do not need to implement this yourself: feel free to use sklearn. What is happening here? Do clusters correspond to the true labels or not? Plot the dendrogram, somehow decide what the final clustering is (by making a cut in the dendrogram), and explain your reasoning behind this decision.

Discuss the results of all steps, and use plots to support your discussion.

Problem 3. Associations [3 points]

- [**Manual; 1.5 points**] Consider the following data set describing the courses taken by UVic students during the Fall 2025:

Student ID	**Courses**
X1	{ 101, 503, 330, 482 }
X2	{ 503, bioinformatics, 330 }
X3	{ 101, 503, 330, 482 }
X4	{ 101, bioinformatics, 330, 482 }
X5	{ 503, bioinformatics, 330, 482 }
X6	{ 503, 330, 482 }
X7	{ bioinformatics, 330 }
X8	{ 101, 503, bioinformatics }
X9	{ 101, 330, 482 }
XA	{ 503, 330 }

Suppose the Apriori algorithm is applied to the data set shown in the table with `minsup`=30%.

1. (0.5 points) Draw an itemset lattice for this data set, and label each node in the lattice with:
   - **N** if the itemset is not considered to be a candidate by the Apriori algorithm;
   - **F** if the candidate itemset is found to be frequent by the Apriori algorithm; and
   - **I** if the candidate itemset is found to be infrequent after support counting.
2. (0.1 point) What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?
3. (0.1 point) What is the pruning ratio of the Apriori algorithm on this data set? Pruning ratio is defined as the percentage of itemsets not considered a candidate because they are either not generated during candidate generation or are pruned during the candidate pruning step.
4. (0.1 point) What is the false alarm rate (i.e., percentage of candidate itemsets that are found to be infrequent after performing support counting)?
5. (0.7 points) Build an FP-Tree on this dataset and then mine the frequent itemsets via FP-Growth algorithm with `minsup`=30%.

This subproblem (Manual) should be done by hand on a piece of paper (or, even better, in a Jupyter notebook). If submitting a paper scan, add `VNumber_Surname Initial_3.pdf` (PDFs only!) to the final ZIP archive.

- [**Retail; 1.5 points**] Now, get the online retail dataset from `https://archive.ics.uci.edu/ml/datasets/Online+Retail+II`, and use the Apriori algorithm implemented in `apyori` package to mine the top 5 associations for each country in the dataset. What are your findings?

  Note: `apyori` returns the associations as a generator. Do not completely consume generators unless you really have to (otherwise it will run for a very, very long time). Use your Python-fu to get the top associations instead. You can use other packages instead as well.