# CS235 Project Description

Mariam Salloum

## 1 Description

In this project, you are going to work in groups of two on a number of data mining techniques that we cover in this class and apply them to a real-world imbalanced dataset.

Please the starter code template ([CS235F25]_Course_Project.ipynb) for your ipynb and be sure to include your name(s) in the filename and within the notebook.

**Programming language:** We will be using Python and Jupyter Notebooks.

**Dataset:** We will be using the Wisconsin Breast Cancer Diagnostic dataset from the UCI data repository: `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic`

**Implementation from scratch:** "Implementation" means writing the code for the method from scratch. You may use packages like Pandas, NumPy etc, but only for their basic functionality (e.g., loading and manipulating data). If you find a resource online that describes the implementation, you may use it as inspiration and guide but anything you submit must be your own original implementation. Verbatim (or nearly verbatim) copies will not be allowed or tolerated and will be reported (see the academic integrity section below).

**Deliverables:** For each phase, your final deliverable will be a Jupyter Notebook (.ipynb file) which should contain the following:

- Your name and SID at the top of the notebook.
- For every question:
  - A markup block with the question number and title, and a brief description of how you went about implementing the solution to the question.
  - One or more code blocks with your original code that implements the functionality required in each question. The code has to contain descriptive comments throughout (not just at the beginning of a block).
  - The ultimate output for each question should be one or more plots that display the required result for that question. All plots must have axis labels, a descriptive title, and a legend (when appropriate).

## 2 Phase 1 - Supervised Techniques [50/100]

In this first phase we are going to focus on supervised techniques and their interaction with dimensionality reduction, feature selection, and data augmentation.

For all results reported in this phase, when referring to "performance" you should measure the average +/- standard deviation of the F1 score calculated using stratified 10-fold cross-validation (the reason for stratification is because our two classes are imbalanced). You may use existing Sklearn functionality to compute the F1 score and to perform cross-validation. When plotting the performance, you should always produce error-bars. Whenever your figure contains more than one line/graph (e.g., performance of two classifiers), you should always make sure those are easily distinguishable (use different colors and markers) and make sure you include a legend.

1. Implementing simple classifiers [15pts]

   **What to implement:** In this question you should implement (1) a decision tree classifier that uses the Information Gain splitting criterion and (2) a Naive Bayes classifier which uses Gaussian modeling for continuous features.
   **What to plot:** You should produce a bar-chart that shows the performance of your classifiers on the dataset.

2. Dimensionality reduction with the Singular Value Decomposition [10pts]

   **What to implement:** The Singular Value Decomposition (SVD) is an extremely useful tool from linear algebra that can help us approximate a data matrix (such as the instance-by-feature matrix that represents our data) into a smaller dimension, hence reduce the dimensionality of the data. In this question you should use the SVD (you may use the existing NumPy library) to approximate the data in different ranks. Important: because we are dealing with supervised learning, given a train/test split, in order to avoid data leakage you should always make sure that you only compute the SVD on the train split and project the test instances to that space, in the same way that the Latent Semantic Analysis paper [1] does.
   **What to plot:** You should produce a figure that shows the performance of your classifiers as a function of the SVD approximation rank

3. Feature selection with randomization [15pts]

   **What to implement:** Randomization is a very powerful tool in helping us understand whether a piece of data contains useful information for a given task or whether the information contained is close to random. In this question you should implement the following feature selection technique which randomizes each feature of the data and measures how predictive it is for our task.

   In order to avoid data leakage, for this question only you should conduct the feature selection on a 20% stratified random sample of the entire dataset, leaving the rest 80as the data on which you will report the performance on your plot.

   - For every feature, generate a copy of the dataset where that feature values are randomized (i.e., randomly permuted)

- Train two versions of the model, one with the actual data and one with the data that contains the randomized feature.
- Test on part of the hold out that is for validation and use the difference in performance as a score for that feature.
- Steps b and c should be done as part of 5-fold cross-validation train/test splits on the feature selection sample and produce an average F1 score for each of the two models (regular and "randomized feature")
- If randomization of the feature results in a drop of performance, the percentage of the drop can be used to characterize how important this feature is for the task
- Repeat for all features and rank features according to the above metric, from most predictive to least predictive

**What to plot (2 figures in total):** A bar chart that shows each feature and the calculated importance

4. Data augmentation using SMOTE [10pts]

   **What to implement:** As we saw in class, SMOTE is one of the earliest and very successful data augmentation (or minority class oversampling) techniques. You can use the pseudocode included in the original SMOTE paper [2] as a basis for your implementation.
   **What to plot:** The performance of your classifiers as a function of the percentage of oversampled minority samples (100%, 200%, 300%) for k = 1, and k = 5. This plot should contain 2 lines per classifier (one for each k).

## 3 Phase 2 - Unsupervised Techniques [50/100]

In this second phase we are going to focus on unsupervised techniques including different clustering paradigms (partitioning, density-based, and graph-based).

For all clustering results reported in this phase, when referring to "performance" you should run the clustering algorithm 10 times with different random initializations and report the average +/- standard deviation of the Silhouette coefficient. You may use the existing Silhouette implementation of Sklearn. When plotting the performance, you should always produce error-bars. Whenever your figure contains more than one line/graph (e.g., performance of two classifiers), you should always make sure those are easily distinguishable (use different colors and markers) and make sure you include a legend.

1. **k-means clustering [10pts]**

   **What to implement:** You should implement Lloyd's algorithm for k-means clustering and the k-means++ initialization algorithm as described in [5]. Your code should have an option to use either fully random or k-means++ initialization.

   **What to plot:** The performance of k-means for k ranging from 1 to 5 when using completely random initialization and when using k-means++

2. **Density-based clustering with DBSCAN [15pts]**
   **What to implement:** You should implement the DBSCAN algorithm

   **What to plot:** The performance of DBSCAN as a function of MinPts (taking values [5, 10, 15, 20] and for (1) Eps = 0.1 and (2) Eps = 0.2 (two lines in total)

3. **Graph-based clustering with Spectral Clustering [10pts]**

   **What to implement:** You should implement the version of Spectral Clustering (titled "Unnormalized spectral clustering") shown in Page 6 of [3]. You should implement the Gaussian similarity function as described in Section 2 [3].

   **What to plot:** The performance of spectral clustering as a function of k ranging from 1 to 5, and for sigma equal to (1) 0.1, (2) 1, and (3) 10 (three lines in total)

4. **Anomaly detection with the Isolation Forest** [15pts]
   **What to implement:** You should implement the Isolation Forest anomaly detection algorithm as described in the original paper [4].

   **What to plot:** The performance of k-means with k-means++ and k=2 on the data after removing the top [1%, 5%, 10%, 15%] of anomalies as determined by the Isolation Forest.

## 4 References

1. Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. "Indexing by latent semantic analysis." Journal of the American society for information science 41, no. 6 (1990): 391-407.

2. Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

3. Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17 (2007): 395-416.

4. Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." In 2008 eighth ieee international conference on data mining, pp. 413-422. IEEE, 2008.

5. Arthur, D. and Vassilvitskii, S., 2007, January. k-means++: The advantages of careful seeding. In Soda (Vol. 7, pp. 1027-1035).

## 5 Academic Integrity

You must only submit your original work. If you receive help by any external sources (other than the TA or the instructor), you must properly credit those sources and describe the exact amount of overlap, and if the help is significant, the appropriate grade reduction will be applied. For example,. if your entire code is taken from an online source, which you properly credit, you will not receive any credit, but this will not be regarded as plagiarism, since you properly credited the source. If you fail to credit all relevant sources, the instructor and the TAs are obligated to take the appropriate actions outlined here. Always cite your sources! Never copy any text, figure, method, or any part of a paper/book/source code/any intellectual work verbatim from that source, unless there is attribution. For code, you have to specify exactly where a given snippet of code came from. Plagiarism is a very serious offense that can get a researcher banned from publishing for a number of years, and is taken very seriously. Instances of plagiarism will absolutely not be tolerated in this class. In addition to UCR's academic integrity website, we encourage you to read carefully the ACM and IEEE policies on plagiarism and academic misconduct.

## 6 What about AI-powered tools?

The answer is the same as above. If you received help by someone else other than the TA or the instructor, including AI-powered tools (ChatGPT, Copilot and so on), you should clearly describe the help you received and document how you produced your own implementation based on that help. If we deem that the help received is substantial, appropriate grade reduction may be applied.

Participating in the class implies your agreement to and understanding of the UCR academic integrity policies above.