Assignment 1

UMC 203: Artificial Intelligence and Machine Learning

March 2025

No copying is allowed. **Thorough plagiarism check will be run.** Refer: CSA misconduct policy You are given three questions, each of which requires Python programming. You may use jupyter notebook to write your Python programs, although not necessary. If you are using a notebook, please convert it to a Python file before you submit it. We expect you to implement your own programs. Code generated by non-humans will be non-acceptable ;). Shenanigans will beget Shenanigans.

SUBMISSION INSTRUCTIONS

1. You should submit **two files** (NOT a zip file) with the following naming convention.
   - ▷ `AIML_2025_A1_LastFiveDigitsOfSRNumber.pdf` → Answers to all the problems.
   - ▷ `AIML_2025_A1_LastFiveDigitsOfSRNumber.py` → Code for the all the problems.

   For example, if the last five digits of your SR Number is 20000, then you should submit four files: `AIML_2025_A1_20000.pdf, AIML_2025_A1_20000.py`.
2. **Any deviation from the above rule will incur serious penalty!**
3. For the coding questions, you are asked to report some values, e.g., the number of iterations. These values should be reported in the `.pdf` file you submit.
4. At the top of the `.pdf` file you submit, write your name and SR Number.
5. Reports must be typed neatly in LaTeX
6. You will be using the `numpy`, `pandas` and `sklearn` libraries for this assignment.
7. All datasets are available under `Files>Assignment 1`

You will need to use **Python 3.12.8**. You are encouraged to use the GPU-laden machines in the UGC Lab.

1 Fisher Linear Discriminant (10 marks)

In class, you have been introduced to the Fisher Linear Discriminant (FLD) for classification of linearly separable data. In this exercise, we will consider the practical challenges in implementing a multi-class FLD. We use the CelebA dataset [Liu et al., 2015]. With the oracle, you will each be given a unique pair of binary attributes of the celebrities in CelebA. You will build a *multi-class FLD* for the four possible classes. For example, if your pair is (No Beard, Young), then, you will learn classifiers to classify between the 4 classes: Beard+Old, Beard+Young, No Beard+Old, No Beard+Young. Each class will contain around 5000 sample RGB images of size 32x32. See Duda et al. [2006, Section 4.11], for some of the details of Multiple Discriminant analysis.

1. Estimation is a core branch of statistics. The FLD 'estimates' the class conditional mean and variance with the samples in the dataset. We study how the estimates change with the number of samples chosen. For each class, study the change in the estimates with n=50, 100, 500, 1000, 2000, 4000. **Report** this with a plot of the ℓ_2 norms of the mean vectors for each class and the Frobenius Norm of the covariance matrices for each class.

   (3 marks)

2. You will now implement the multi-class FLD for the provided dataset.

   (a) Find the weights of the FLD for each of the 4 classes with n=2500, 3500, 4000, 4500, 5000 samples with 20 different subsets (except 5000). **Plot** box plots for the multi-class objective value for different values of n. **Plot** the projection of points of each class onto the 'linear' discriminant. Based on the above plot, choose *suitable* choices for the threshold for each classifier. **Report** the chosen thresholds and draw insight on the projected points.

   (5 marks)

   (b) **Report** the accuracy of the model on the test set with the classifiers with different number of samples. You will not be graded solely based on the performance of the classifier, but on the soundness and correctness of the design of the classifier based on the multi-class FLD. **(2 marks)**

Details of the Oracle functions:

- **Prerequisites**: Ensure you have downloaded the CelebA dataset and placed the data in a folder named `CelebA` in your project directory. This will be around **14GB** compressed, only retain the folders `Anno` and `Img/img_align_celeba`. Unzip `Img/img_align_celeba.zip` into a folder `CelebA/Img/img_align_celeba`.

- `q1_fish_train_test_data(srn)`: Takes a 5 digit integer, which is your SR number as input. Please use your SR number. The output is a 5-tuple: `(('attribute 1', 'attribute 2'), [train image tensors...], [train labels...], [test image tensor...], [test labels...])`. For example: (('No Beard','Young'), [im1, im2, ...(list of 20000 images of size 3x32x32)], [0,1,2,3,0,3,1,...(list of 20000 labels from 0,1,2,3)], [test1,...(list of 1000 images of size 3x32x32)], [1,2,...(list of 1000 labels from 0,1,2,3)]). You are highly encouraged to actually take a look at these images. **Note:** The images are normalized to [0,1]; you will need to multiply by 255 and convert to uint8 to view the images correctly. Label 0 refers to both attributes False, 1 is only the second attribute is True, and so on. These will serve as training images for your learning algorithm. The train data will be used to answer Questions 1 and 2a. The test data will be used to answer Question 2b.

For students of history, read Belhumeur et al. [1997].

2 Bayes Classification (10 marks)

In a standard binary classification setting, a classifier h assigns each input x to a class in $\{0, 1\}$. However, there are situations in which the classifier might be highly uncertain and could benefit from not making a prediction at all. One way to address this is via **classification with a reject option**. In this framework, we allow the classifier to output a special label, reject, when it deems the input too ambiguous. This approach can be useful in safety-critical applications (e.g., medical diagnostics), where making no decision can be preferable to making a wrong one. Let $\eta(x) = P(Y = 1|X = x)$ denote the posterior probability that $X = x$

belongs to class 1. The *(standard)* Bayes classifier assigns a sample x to class 1 if $\eta(x) > \frac{1}{2}$ and to class 0 otherwise. The Bayes classifier minimizes the probability of misclassification under the assumption that η is known. In the reject-option setting, we introduce a threshold parameter $\epsilon \in \left(0, \frac{1}{2}\right)$ and define the **Modified Bayes Classifier** h_ϵ as follows:

$$h_\epsilon(x) = \begin{cases} 1 & \text{if } \eta(x) \geq \frac{1}{2} + \epsilon \\ 0 & \text{if } \eta(x) \leq \frac{1}{2} - \epsilon \\ \mathsf{reject} & \text{otherwise} \end{cases}$$

We evaluate its performance using:

- the error probability $P(h(X) \neq Y(X) \mid h(X) \neq \mathsf{reject})$ where $Y(X)$ is the true class label.
- the probability of rejection $P(h(X) = \mathsf{reject})$

You are provided 2400 samples each from 2 classes of the **Extended MNIST** [Cohen et al., 2017] dataset and asked to learn a classifier that minimizes the misclassification loss.

1. Train the Modified Bayes Classifier for different values of $\epsilon \in \{0.01, 0.1, 0.25, 0.4\}$ on training data and evaluate how decision confidence affects the performance on test data. **Report**:

   - Misclassification Loss among the non-rejected samples
   - Number of rejected samples

   Report a plot showing how the misclassification loss changes with respect to ϵ. **(1 Marks)**

2. You are provided a dataset with a 50-50 split between priors. Subsample the dataset to create datasets with a 60-40, 80-20, 90-10, 99-1 split. Now compute the Modified Bayes Classifier under these modified priors for $\epsilon = \{0.1, 0.25, 0.4\}$. **Report**:

   - Misclassification Loss among the non rejected samples for each ϵ and varied splits.
   - Number of rejected samples.

   Report a plot showing how the misclassification loss changes with respect to ϵ for varied splits. **(3 marks)**

3. **K-fold cross validation**
   K-fold cross validation is a technique to evaluate the performance and robustness of a model by partitioning the train dataset into equally sized subsets ('folds'). The train dataset is split into K folds and the model is trained K times, using $K - 1$ folds for training and one fold for validation, and then averaging the performance metrics across the K models. Refer Wikipedia for more details.

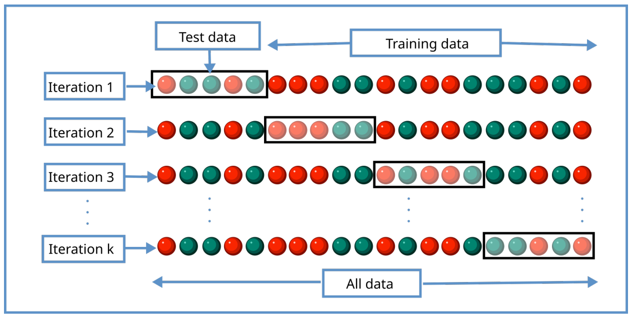

Figure 1: K-Fold Cross Validation

   (a) Perform K-Fold Cross Validation with $K = 5$ and run Modified Bayes classifier with $\epsilon = 0.25$. For each fold, construct the confusion matrix on the validation set. Then, **report** the following performance metrics (you may treat 'reject' as a third category in the confusion matrix, or simply exclude it and focus on the 2×2 confusion of 'predicted 0 vs. predicted 1' whenever the classifier does not reject):

i. Recall: $\frac{TP}{TP+FN}$

ii. Precision: $\frac{TP}{TP+FP}$

iii. Accuracy: Proportion of correctly classified instances among the total instances

iv. F1-score: Harmonic mean of precision and recall, providing a balanced measure of model performance

You are allowed to use scikit-learn for **K-fold cross validation**. **(5 Marks)**

(b) Apply the trained model to the test data and **Report**:

- Number of rejected samples.
- Report the misclassification loss among the non-rejected samples **(1 Marks)**

Details of the Oracle functions:

- **Prerequisites:** Ensure you have downloaded the Extended Mnist Dataset and placed the `emnist-balanced-test.csv` and `emnist-balanced-train.csv` in a folder named `EMNIST` in your project Directory.
- `q2_train_test_emnist(srn, train_csv_path, test_csv_path)`: Takes a 5 digit integer, which is your SR number, the path to your `emnist-balanced-train.csv` and `emnist-balanced-test.csv` as input. The output is a tuple of training and testing data where both of the entries are `numpy.ndarray`. You are encouraged to look at the output of the function and take a look at these images.

3 Decision Trees (10 marks)

Decision trees are hierarchical models used for classification and regression. They recursively partition the data into subsets based on feature values, forming a tree-like structure. You can read more about them in Quinlan [1986, 1993], Breiman et al. [1984]. *Decision trees will be discussed in the tutorial.* In this question, you will implement a decision tree using `DecisionTreeClassifier` in `scikit-learn`.

Instructions:

1. Download the UCI Heart Disease Dataset. This dataset contains 303 instances with 14 features, including age, cholesterol, blood pressure, and chest pain type, with the goal of predicting presence of heart disease (target variable: 1 = Disease, 0 = No Disease).
2. Do any necessary data cleanup. Split the dataset into train (80%) and test (20%) sets.
3. Query the oracle for the `criterion`, `splitter`, and `max_depth` hyperparameters.
4. Train a `DecisionTreeClassifier` on the train set.

Report the following:

1. Visualize the decision tree you trained using `dtreeviz`. **(4 marks)**
2. Report the **precision**, **accuracy**, **recall**, and **F1 score** of the above decision tree. **(4 marks)**
3. What is the most important feature for predicting heart disease according to your tree? **(2 marks)**

Details of oracle

- `q3_hyper(srn)`: Takes a 5 digit integer, which is your SR number as input. Please use your SR number. The output is a 3-tuple `(criterion,splitter,max_depth)` to be used as your input for the function `DecisionTreeClassifier`.

References

P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997. doi: 10.1109/34.598228.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole, 1984. ISBN 978-0-412-04841-8.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andr van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017. doi: 10.1109/IJCNN.2017.7966217.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-India, 2006. ISBN 9788126511167. URL `https://books.google.co.in/books?id=NR-SzW2t7WYC`.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi: 10.1007/BF00116251.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN 978-1-55860-238-0.