INF8245AE(Fall 2025) : Machine Learning — Given on: Oct 22^{nd}

Assignment #3 — Due on : Nov 14^{th}, 10 pm

- This is an individual assignment. You are not allowed to discuss the problems with other students.
- Make sure to write your code only in the `.py` files provided. Avoid creating new files. Do not rename the files or functions as it will interfere with the autograder's ability to evaluate your work correctly. Also, do not change the input or output structure of the functions.
- When Submitting to GradeScope, be sure to submit

  1. A `.zip` file containing all your Python codes (in `.py` format) to the `Assignment 3 - Code` section on Gradescope. Do not submit data, environment files, or any other files.
  2. A `.pdf` file that contains your answers to the questions to the `Assignment 3 - Report` entry.

- Part of this assignment will be auto-graded by Gradescope. You can use it as immediate feedback to improve your answers. You can resubmit as many times as you want. We provide some tests that you can use to check that your code will be executed properly on Gradescope. These are **not** meant to check the correctness of your answers. We encourage you to write your own tests for this.
- Please provide the title and the labels of the axes for all the figures you are plotting.
- If you have questions regarding the assignment, you can ask for clarifications in Piazza. You should use the corresponding tag for this assignment.
- Before starting the assignment, make sure that you have downloaded all the data and tests related to the assignment and put them in the appropriate locations.
- You **are allowed** to use **Scikit-learn** for training, testing, and evaluating algorithms such as Support Vector Machines and Decision Trees. Please do not use any other machine learning library for applying the classification algorithms. Appropriate instructions are provided for each question.
- You cannot use ChatGPT or any other code assistance tool for the programming part; however, if you use ChatGPT to edit grammar in your report, you have to explicitly state it in the report.

Dataset

We will be using the **Adult Income Dataset**, from the UCI Machine Learning Repository. This dataset was extracted from the 1994 US Census database by Ronny Kohavi and Barry Becker, and contains 48,842 samples. Each record represents an individual, described by a set of demographic and employment-related attributes, with the objective of predicting whether their annual income exceeds $\$50,000$. The dataset includes 14 features, which are a mix of continuous and categorical variables. These features capture information such as age, education, occupation, marital status, and hours worked per week. The target variable (or label) is binary, indicating whether the individual's income is greater than $\$50K$ ($> \$50K$) or less than or equal to $\$50K$ ($\leq \$50K$). For more details about the dataset and its attributes, please refer to the official description on the UCI Machine Learning Repository. For the purpose of this assignment, we will drop two unimportant/redundant features, **fnlwgt** and **education**, leaving us with 12 features in total.

Data Exploration and Preprocessing (20 Points)

1. In this assignment, you will employ several supervised learning algorithms to accurately classify whether an individual earns more than $50,000 per year based on demographic and employment-related attributes. You will then identify the most promising algorithm based on the preliminary results and further refine it to achieve optimal performance on the dataset. The dataset is often used for binary classification tasks and serves as a benchmark for evaluating algorithms in fairness-aware machine learning, as it includes sensitive attributes such as sex and race. The dataset can be loaded by running the `downloading_data()` function in **q1.py** file. You may want to print a few rows of the dataset to see the type of features and label. In **q1.py**, data is loaded in `adult_income`, `X` shows the feature matrix and `y` shows the labels.

   First, we need to perform an initial exploration of the dataset to understand its structure.

   (a) (8 Points) Show:

   - average age of the individuals
   - percentage of women
   - percentage of individuals earning more than $50K
   - percentage of missing values
   - a scatter plot with features "Hours-per-week" and "Age" on the two axes, with samples labeled according to this task (i.e., $> \$50K$ or $\leq \$50K$)

   Write your code in the `data_exploration()` function in **q1.py**. Provide the answers and the plot in the report and briefly describe what you observe.

   (b) (5 Points) Note that there are missing values in the dataset. Identify the features that have missing values, and perform a data imputation in the `data_imputation()`

function based on the most frequently occurring class for categorical features and the mean for numerical features. Ensure that the column names are retained.

Hint: Use `SimpleImputer` from `sklearn`.

(c) (5 Points) Convert categorical variables to numerical variables: Some features might neither be numerical nor ordinal, i.e., categorical variables. However, the learning algorithms we want to use work with numerical variables. One way to convert categorical to numerical variables is to use `OneHotEncoder` from `sklearn`. Identify categorical features from the dataset and write your code in the `feature_encoding()` function in **q1.py** to convert them to numerical features. Report the final number of features after preprocessing.

Hint: Convert only the categorical variables to OHE, and concatenate the resulting feature matrix with the original matrix after dropping the initial categorical columns.

(d) (2 Points) Moreover, the labels are also categorical since they are either "$\leq 50K$" or "$> 50K$". So convert "$\leq 50K$" to **0** and "$> 50K$" to **1**. Write your code in the `encode_label()` function to convert labels in **q1.py**.

Model Training and Evaluation (55 Points)

In this part, we want to create train and test sets and then pick some classification algorithms to train on the training set.

Now we need to have a training set to train the model on and a test set, which is separate from the training set to evaluate the model. For this we split the data (both features and their labels) into training and test sets.

1. (5 Points) Shuffle and Split Data: Use an 80-20 split ratio meaning 80% for training and 20% for test. For this, it is important to shuffle data before splitting or use a function that does it. Also, to have unique results, set random seed to 0. To do that, implement your code in the `data_splits()` function in **q2.py**. You need to set **random state of 0**.

   Hint: Use `train_test_split` from `sklearn`. You need to set random state of 0 in `train_test_split`.

2. (5 Points) Normalize features: normalization ensures that each feature is treated equally when applying supervised learners. Write your code in the `normalize_features()` function to normalize the variables in **q2.py**. You should calculate the normalization parameters only using the training data and then use the same parameters to normalize the test data.

   Hint: Use `MinMaxScaler` from `sklearn`.

3. (10 Points) Train models: Now we need to train a model on the training set to learn how to predict the labels from the features. We chose these three classification algorithms to model the data:

- (2 Points) Decision-Trees,
- (2 Points) Random-Forest,
- (2 Points) Support Vector Machines (SVM with Support Vector Classification).

For each of these methods you need to train models on all the samples of the training set. You need to set random state of 0 for each model. You should use the Scikit-Learn package. Implement your code in the `train_model()` function in **q2.py**

Note 1: Use the default settings for each model — you will tune the models in a later section.

Note 2: Depending on the algorithms, the code may take some time to run!

4. (35 Points) Model evaluation:

   Report: You need to write the results of these questions in your report in addition to the implementation.

   Here we want to use metrics to see first if the models with default parameters learned the data and then evaluate the performance of the methods.

   (a) (10 Points) Calculate accuracy and F1-score to evaluate the performance of each model with default values. Write your code in the `eval_model()` function in **q2.py**. First calculate the accuracy score for both the training set and test set. Then, calculate the F-score for both sets.

   (b) (10 points) You need to report these values for both train and test sets in your report. Justify your response.

   (c) (5 points) Use `classification_report()` and `confusion_matrix()` from Scikit-Learn. Implement your code in the `report_model()` function in **q2.py**.

   (d) (10 points) Write your results from `classification_report()` and `confusion_matrix()` in your report. Explain what they mean.

Model Tuning (45 Points)

Here you will fine-tune parameters of the three supervised learning models to find a better model for each of them. You will then perform a grid search optimization with cross-validation for the model over the entire training set (`X_train` and `y_train`).

1. (20 Points) Specify Hyper-Parameters:

   Report: You need to write the answer of this question in your report.

   For Random-Forest and SVM, write five significant hyper-parameters they have. Then for each of the hyper-parameters explain what they are and what are their impact on their model briefly.

2. (20 Points) Grid search: use grid search `(GridSearchCV)` for the hyper-parameters of the models with a good range of values. You are responsible for specifying a reasonable range for each hyper-parameter and find at least one setting which has higher performance than the default.

   (a) (3 points) Create a dictionary of the specified parameters below for each models with a good range of values. Here are the parameters for each model:

      1. Decision-Trees: `criterion`, `max_depth`, `min_sample_leaf`, `max_leaf_nodes`
      2. Random-Forest: `n_estimators`, `max_depth`, `bootstrap`
      3. SVM: `kernel`, `shrinking`, `C` (Regularization parameter), `tolerance`, `gamma`

   (b) (3 points) Initialize the classifier with 0 random state.

   (c) (2 points) Create a scorer, it should be accuracy.

   (d) (2 points) In the `perform_grid_search()` function write your code for cross-validation with 10-fold. You need to use `StratifiedKFold`.

   (e) (2 points) Perform grid search on the classifier using the `scorer` and `cross-validation`, and store it in `grid_search`.

   (f) (6 points) Fit the grid search object to the training data (`X_train`, `y_train`).

   (g) (2 points) Print the best parameters and best scores.

   Implement your codes in **q3.py**.

3. (5 Points) **Report:** explain why do we need cross-validation and why should we do stratified cross-validation in your report?

Report (10 Points)

You need to write the answers to these questions in your report based on the previous questions.

1. (6 Points) For each of the models, choose the hyper-parameter specified below and use the ranges you defined in the Model Tuning section to calculate the accuracy on the training set for different values within that range. Then make a plot for each of the models where the x-axis is the value of the hyper-parameter and the y-axis is the accuracy (connected line plot or bar plot). Justify the effect of the hyper-parameter values on the accuracy for each model separately.

   - Decision-Trees, for `max_depth`
   - Random-Forest, for `n_estimators`
   - SVM, for `kernel`

   Note: Use the same hyper-parameter ranges you defined in the Model Tuning section. For example, if you chose `max_depth` values {None, 5, 10, 15, 20} for Decision Trees in the tuning section, use those same values for this analysis.

2. (2 Points) With the best hyper-parameter values you found from the model tuning part, now find the accuracy of the test dataset. You need to find it for each model separately. Then plot it such that the x-axis is model names and the y-axis is the test accuracy.

3. (2 Points) What is the best model's accuracy on the test set across all models? Why do you think it is better? (explain briefly in one paragraph)