ECE656 W25 – Lab 4 – Data Mining

Due date: March 14th, 2025 (submit to dropbox in LEARN)

Group work policy: In this lab, students are expected to work in groups no larger than 3. Groups of 1 or 2 are fine.

Objective:

- develop a working knowledge of classification using decision trees
- gain familiarity with data mining using the Python scikit-learn library
- practice critical thinking skills

1 Data set

In this lab, you will perform data mining computations over data from the US Census Bureau. We will use a specific data set credited to Ronny Kohavi and Barry Becker: https://archive.ics.uci.edu/dataset/20/census+income. The data set comprises one table with fifteen columns: `age`, `workclass`, `fnlwgt`, `education`, `education-num`, `marital-status`, `occupation`, `relationship`, `race`, `sex`, `capital-gain`, `capital-loss`, `hours-per-week`, `native-country`, and `income`. Some columns represent integer features (e.g., `age`), while others are categorical (e.g., `education`). Some values are missing, and they are represented by question marks. The `fnlwgt` (final weight) column indicates the number of distinct individuals represented by a particular combination of feature values in one row of the data set; we will ignore this column for simplicity, and treat each row as representing a single instance. The data set is partitioned into distinct training and testing portions, in files `adult.data` and `adult.test`, respectively. The training portion comprises 2/3 of the data set, and the testing portion comprises the remaining 1/3.

2 Classification Task

You will analyze the census data and construct a suite of classifiers to predict `income`, which is a categorical feature with two values: $\leq$50K (low income) and >50K (high income).

Task A: construct decision tree classifiers of different sizes to predict `income` from the other features. In this task, you will explore the inherent trade-off between predictive performance (e.g., accuracy or F1 score) and model complexity (e.g., number of tree nodes) by selecting subsets of features and tuning the parameters of the training algorithm. **To complete this task, construct one decision tree for each of the following three cases, aiming to maximize the predictive power in each case:**

C1 *small* decision tree: at most 10 leaf nodes

C2 *medium* decision tree: at most 100 leaf nodes

C3 *large* decision tree: no limit on the number of leaf nodes

Task B: improve the predictive power of your *large* tree from Task A by applying one of the well-known ensemble methods. **To complete this task, construct one ensemble classifier, aiming to maximize the predictive power.**

Note: Task B is an open-ended exercise. Don't worry about finding the optimal ensemble technique, and don't go overboard with tuning. We suggest that you limit the time spent on this task to a few hours.

3 Methodology

For a classification problem, we begin with *feature selection*. In this phase, you will identify the specific features to be used as inputs to the machine learning model. Some features may be more suitable for a given classification problem, while others may be redundant or irrelevant, and can be dropped at this stage. The chosen features in this lab must be loaded from a file into a Python DataFrame. Furthermore, categorical features must be transformed to numerical for the specific decision tree learning algorithm implemented in the scikit-learn library.

The starter coded provided with Lab 4 takes care of loading the data set and encoding categorical features into integers. We recommend that you install Python version 3 on your own computer, along with the required packages: `numpy`, `pandas`, `scikit-learn`, `graphviz`, and `pydotplus`. Please see the comments at the top of the starter code file for further instructions on how to install missing packages.

After loading and transforming the data set, you will construct classifiers for Task A and Task B, which were defined earlier. Three decision trees are required for Task A, and one ensemble classifier is required for Task B. All classifiers must be trained on `adult.data` and tested on `adult.test`. For each classifier, you will compute the *confusion matrix*, and performance metrics like *accuracy* and *F1 score*. It is important to do the computation both on the training portion of the data set and on the testing portion for comparison, and to watch for signs of over-fitting (i.e., much better performance on training data than on testing data). You will then tune the classifiers by experimenting with different hyperparameters (e.g., impurity measure, maximum depth, maximum number of leaf nodes, etc.) to **optimize the predictive power on the testing portion of the data set**, and reflect on your design choices.

4 Deliverables

The expected submission for this lab is a collection of files, to be submitted electronically using the Lab 4 dropbox in LEARN. Each deliverable is described in more detail below.

1. **Report [80%]**
   Write a report on your findings, and submit it as a single PDF file named `lab4-report.pdf` that includes all supporting tables and figures. The report should be 1000–1500 words long (excluding title page, tables, figures, and illustrations), and include the following content:

   (a) Start with a title page that states the names, Waterloo emails, and student numbers of all group members, as well as the group number.

   (b) Compute the total number of instances belonging to each class in the data set. Also compute the proportion of instances belonging to each class.

   (c) What values of accuracy and F1 score can be achieved by a classifier that always outputs a hard-coded class label (e.g., low income)? In other words, how good can a trivial solution be in this classification problem?

   (d) Is accuracy or F1 score a better measure of predictive power in this lab? Explain.

   (e) For each of the two prediction tasks defined earlier, and for each classifier you construct, explain and justify the chosen set of features, and comment on how you tuned hyperparameter values. Additionally, present the confusion matrix as a table, compute both the accuracy and F1 score obtained on the training and testing data,

and quantify the relative predictive power of your chosen features. Is there evidence of over-fitting? Which features were the most useful for classification? Did you encounter any surprises?

(f) For the *small* and *medium* classifiers for Task A, present illustrations of the decision trees. The illustrations must show clearly the features used in different splits, as well a structural properties such as the depth, branching factor, and number of leaf nodes. Either embed scalable graphics directly in your report document, or submit the illustrations are separate files (e.g., in SVG format).

(g) For Task B, state which ensemble technique you chose and comment on the improvement in predictive power you were able to achieve. Discuss any additional hyperparameter tuning you did.

(h) Conclude the report with some critical thinking. What are the underlying factors that cause an individual to earn a high or low income in the real world? Comment on potential associations and cause-effect relationships between these factors and the features you extracted from the data set to train your models.

2. **Python code [20%]**
Submit the Python scripts used to construct and evaluate your classifiers. Name the files `lab4-taskA-small.py`, `lab4-taskA-medium.py`, `lab4-taskA-large.py`, and `lab4-taskB.py`.

Your submission will be checked for completeness, and also assessed for quality. Highest grades will be awarded to submissions that are technically deep and well polished.

5 Python reference material

Python Sklearn method	Description
clf = DecisionTreeClassifier()	This method constructs a decision tree classifier object.
model = clf.fit(X_train, y_train)	This method is used to train the decision tree and outputs a concrete model for classification. It takes two arguments: X_train refers to the feature values, and y_train refers to the class labels (i.e., ground truth). https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.fit
y_pred = model.predict(X_test)	This method is used to evaluate the output of a trained model. The argument X_test refers to the feature values from the testing portion of the data set. The output is a predicted class label for every instance in the testing portion of the data set. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.predict
confusion_matrix(y_test, y_pred)	Computes the confusion matrix given the correct class labels for the testing data in y_test and the predicted class labels for the testing data in y_pred. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Table 1: Python Sklearn methods for constructing and evaluating decision tree classifiers.

Hyperparameter name	Description
max_leaf_nodes	The maximum number of leaf nodes in the tree.
max_depth	The maximum depth of the tree.
min_samples_split	The minimum number of instances required at an internal node of the tree.
min_samples_leaf	The minimum number of instances required at a leaf node of the tree.

Table 2: Python Sklearn hyperparameters for decision tree classifiers.