Homework 2, Machine Learning, Fall 2023

***IMPORTANT* Homework Submission Instructions**

1. All homeworks must be submitted in one PDF file to Gradescope.
2. Please make sure to select the corresponding HW pages on Gradescope for each question.
3. For all theory problems, please type the answer and proof using Latex or Markdown, and export the tex or markdown into a PDF file.
4. For all coding components, complete the solutions with a Jupyter notebook/Google Colab, and export the notebook (including both code and outputs) into a PDF file. Concatenate the theory solutions PDF file with the coding solutions PDF file into one PDF file which you will submit.
5. Failure to adhere to the above submission format may result in penalties.

All homework assignments must be your independent work product, no collaboration is allowed. You may check your assignment with others or the internet after you have done it, but you must actually do it by yourself. **Please copy the following statements at the top of your assignment file**:

Agreement 1) This assignment represents my own work. I did not work on this assignment with others. All coding was done by myself.

1 Data Science Pipeline - Theory 5pts (Stephen)

In this problem, you will be provided with a scenario that follows the data science pipeline. However, the steps are out of order. It is your task to put the scenario back together so it follows a typical pipeline. Dr. Rudin presented lots of different pipelines in class, but in this question we will use the terminology and structure of the Computing Community Consortium (CCC) Big Data Pipeline. As a reminder, this pipeline is separated into 5 main components:

- Acquisition/Recording
- Extraction/Cleaning/Annotation
- Integration/Aggregation/Representation
- Analysis/Modeling
- Interpretation

Put the scenario sections in the correct order. Include a very brief (1-2 sentences) justification for each section.

Scenario:[1]

The Lorax (he speaks for the trees) must convince the Once-ler to stop cutting down Truffula Trees to knit his Thneeds™! The Lorax believes that showing the Once-ler solid, data-driven evidence for the business's unsustainability will finally change his mind. Friends of the Lorax – the Bar-Ba-Loots, Swomee-Swans, and Humming-Fish – have graciously offered their help.

(i) Using the raw image data of the forest, the Bar-Ba-Loots count the number of trees cut each day, as well as the total number of trees left in the Truffula forest. They make sure to take newly grown trees into account. They include information on Truffula tree tuft color and quality. Any ambiguous or dubious information is thrown out.

(ii) Every day, the Swomee-Swans fly over the Truffula forest and take high-definition photos of the treetops. They try to be consistent, taking the pictures at the same locations and times each day. Everyone makes mistakes though, and birds aren't very good at handling cameras mid-flight.

(iii) The Lorax compiles all the evidence, looks at the most important features predicting forest devastation, and comes up with a compelling plan to make the Thneed business sustainable, thus saving the Truffula Trees from decimation and allowing the Once-ler to continue business. Win-win!

(iv) The Swomee-Swans look at the data as a time series and try to predict when the Truffula trees of various colors and qualities will cease to exist based on current trends. Furthermore, they adjust parameters to find situations where sustainable business is possible.

(v) The Humming-Fish type away on their fin-top computers and shell-phones, working to get all the data in an easily-accessible, computer-readable format.

2 Information Theory 15pts - Theory (Harry & Eric)

We saw in class how the notion of entropy can be used to keep decision trees simple. In this problem, we will show another application of information theory by proving a result on data processing. We begin with some definitions.

[1] https://yale.learningu.org/download/91736886-e31e-47c0-8a3b-f1c1843a6f7c/H3146_The%20Lorax_Storybook.pdf

For this section, let X, Y, Z be arbitrary random variables defined over the sample spaces $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ respectively. We will simplify notation by writing $X = x$ as just x, and we will do the same for $Y = y$ and $Z = z$. Recall the entropy for the random variable X:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = E[-\log p(x)]$$

We can similarly define the joint entropy and the conditional entropy between two random variables X, Y:

$$H(X, Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log p(x, y) = E[-\log p(x, y)]$$

$$H(X|Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(y)} = E[-\log p(x|y)]$$

We can also define mutual information between two random variables X, Y as the decrease in entropy when we condition X by Y (or vice versa):

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Finally, we can also condition the mutual entropy between X and Y on the random variable Z. This is defined as follows:

$$I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = H(Y|Z) - H(Y|X, Z)$$

(a) Show that the mutual information between two random variables is always non-negative. That is, show that $I(X; Y) \geq 0$ and $I(X; Y|Z) \geq 0$. Hint: use Jensen's Inequality.

(b) Recall the chain rule of probability: $P(y, z) = P(y|z)P(z)$. Prove an analogous chain rule for mutual information:

$$I(X; Y, Z) = I(X; Y|Z) + I(X; Z) = I(X; Z|Y) + I(X; Y)$$

Note that $I(X; Y, Z)$ is the mutual information between X and the joint distribution of Y and Z. Hint: the proof is short!

(c) Suppose that $P(z|y, x) = P(z|y)$. We can equivalently model this relationship as the Markov Chain $X \to Y \to Z$ where the value of Z depends directly on that of Y. Show that $I(X; Y) \geq I(X; Z)$.

Note: The idea behind this theorem is to conceptualize Y as some data about X, and Z as some sort of processed version of Y without observing X. The theorem then shows that any processing of Y cannot increase the information one has about X. In other words, it is imperative that any data you collect is good before you attempt to run an analysis!

3 Build Decision Tree by Hand 10pts - Theory (Stark)

In class, we mainly used the information gain (overall reduction in entropy) as the criterion for choosing which feature to split on. Another option would be to use the Gini index.

Remember that the splitting criteria is a measure of how "impure" the node is. If the nodes are "pure," it means that the fraction of positives in the node is either close to 1 or close to 0. (Either almost all the observations are positive or they are almost all negative.) If there are 2 classes, the Gini index for $(p,\, 1-p)$ is the following, where p is the fraction of positives in the node and $1-p$ is the fraction of negatives:

$$\text{Gini index}(p, 1-p) = 1 - p^2 - (1-p)^2 = 2p(1-p).$$

You can see that if p is close to either 0 or 1 the Gini index will be very low. We will choose to split the node with the best reduction in Gini index, averaged across the leaves (children) of the possible split. Let us denote N as the number of observations in the node we are considering to split. Define p to be the fraction of positives in the node we are considering to split. Denote p_c as the fraction of positives in the c^{th} branch of the potential split, and $1-p_c$ as the fraction of negatives in the c^{th} branch of the potential split. Denote N_c as the number of observations falling into the c^{th} branch of the potential split. Then:

$$\text{Gini Reduction} = \text{Gini index}(p, 1-p) - \sum_{\text{children c}} \frac{N_c}{N} \text{ Gini index}(p_c, 1-p_c).$$

(a) The Ferrari F1 team hired you as a new analyst! You were given the following table of the past race history of the team. You were asked to use the Gini index to build a decision tree to predict race wins. First, you will need to figure out which feature to split first. (Show all work leading to your answer)

Rain	Good Strategy	Qualifying	Win Race
1	0	0	0
1	0	0	0
1	0	1	0
0	0	1	1
0	0	0	0
0	1	1	1
1	0	1	0
0	1	0	1
0	0	1	1
0	0	1	1

(b) If you had split according to information gain, would you choose the same feature to split first?

4 Branch and Bound for Optimal Decision Trees 15pts - Theory (Jon)

In theory, it is possible to find the optimal decision tree for a problem by *exhaustively* evaluating the objective value of every possible tree. However, this is computationally intractable for all but the simplest datasets and shallowest trees. In this problem, you will prove a bound for optimal decision trees with regularization, and use this to reduce the search space for finding the optimal decision tree.

It is convenient for this problem to describe a tree as a set of leaves, where leaf k is a tuple containing the logical preposition satisfied by the path to leaf k, denoted p_k, and the class label predicted by the leaf, denoted $\hat{y}_k$. For a dataset with d binary features, $p_k : \{0,1\}^d \to \{0,1\}$ is a function that returns 1 if a sample x_i satisfies the preposition, and 0 otherwise. That is, leaf k is $(p_k, \hat{y}_k)$, and a tree f with K leaves is described as a set $f = \{(p_1, \hat{y}_1), \ldots, (p_K, \hat{y}_K)\}$. Assume that the label predicted by $\hat{y}_k$ is always the label for the majority of samples satisfying p_k. Finally, let $m_k = \sum_{i=1}^{n} p_k(x_i)$ denote the number of training samples "captured" by leaf k.

x_1	x_2	x_3	x_4	y
0	0	1	1	0
0	1	1	1	1
0	0	0	1	0
0	1	0	1	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	1
1	1	0	1	1

Table 1: A hypothetical dataset for question 4.

For a dataset ($\mathbf{X} \in \{0,1\}^{n \times d}, \mathbf{y} \in \{0,1\}^n$) and a model f, consider the following objective function:

$$R(f, \mathbf{X}, \mathbf{y}) = \ell(f, \mathbf{X}, \mathbf{y}) + \lambda s(f),$$

where $\ell(f, \mathbf{X}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{K} p_j(x_i) \mathbf{1}[\hat{y}_j \neq y_i]$ denotes 0-1 misclassification loss for tree f, $s(f) = |\{(p_1, \hat{y}_1), \ldots, (p_K, \hat{y}_K)\}|$ denotes the number of leaves in f, and $\lambda \in \mathbb{R}^+$ is a non-negative real value. For the tree $f_0 = \{(p_1, \hat{y}_1)\}$, which contains zero splits and thereby a single leaf, p_1 is a function that always returns 1 and $\hat{y}_1$ is the majority label for the dataset.

(a) Given a dataset with d binary features, how many binary trees with depth *at most* $h \in \mathbb{Z}^+$ (i.e., splitting on up to $h-1$ features to reach a leaf) are there? These trees may not split on the same variable multiple times within the path to a given leaf, e.g., splitting on x_1 at the root then splitting on x_1 again at level below is not valid. Compute the number for $d = 3, h = 4$ and $d = 4, h = 5$. Hint: It may be helpful to define this quantity recursively.

(b) Rather than naïvely searching through every possible tree, we can reduce the number of trees we must check by ignoring areas of the search space that could not possibly contain optimal trees. Assume there is one feature that, in our dataset, is always 1, and that $\lambda > 0$. How can we use this fact to reduce the number of "potentially optimal" trees we need to explore? Starting with a dataset of $d = 4$ features, how many trees with depth at most 5 are there to search after this reduction?

(c) Prove that, given a (potentially optimal) tree

$$f = \{(p_1, \hat{y}_1), \ldots, (p_\kappa, \hat{y}_\kappa), \ldots, (p_K, \hat{y}_K)\},$$

the tree $f' = \{(p_1, \hat{y}_1), \ldots, (p_{\kappa_1}, \hat{y}_{\kappa_1}), (p_{\kappa_2}, \hat{y}_{\kappa_2}), \ldots, (p_K, \hat{y}_K)\}$ produced by splitting leaf $(p_\kappa, \hat{y}_\kappa)$ into two leaves $(p_{\kappa_1}, \hat{y}_{\kappa_1})$ and $(p_{\kappa_2}, \hat{y}_{\kappa_2})$ and any tree produced by further splitting $(p_{\kappa_1}, \hat{y}_{\kappa_1})$ or $(p_{\kappa_2}, \hat{y}_{\kappa_2})$ cannot be optimal if $m_\kappa < 2n\lambda$.

(d) Consider the dataset shown in Table 1. Using this dataset, and setting $\lambda = 0.3$, evaluate the loss of every tree *that could be optimal* using a depth first search over trees. That is, starting with the tree with 0 splits and only 1 leaf, recursively consider each possible split of each leaf until you have exhausted every tree. You should skip evaluating trees that can be shown not to be optimal using part (**b**) or (**c**) of this question. What is the objective value for the optimal tree? What proportion of the total number of possible trees of depth at most 5 for this dataset did you need to evaluate to determine this? For the purposes of this question, you did *not* "evaluate" a tree if it is excluded using the bound from part (**b**) or (**c**), and the trivial tree containing 0 splits (1 leaf) is considered a tree.

5 Decision Trees Implementations 15pts - Coding (Yiyang)

In this problem, you are going to play with decision trees and sparse decision trees and compare their performance on the Kidney Stone Prediction Dataset in Kaggle. The goal is to predict the presence of kidney stones

based on urine analysis. It contains 6 features (`gravity`, `ph`, `osmo`, `cond`, `urea`, `calc`) and 1 target (`target`). You can find the meaning of each feature at Kaggle Kidney Stone Prediction. It is a binary classification task.

(a) Simple Decision Tree Construction

In this section, we will use `sklearn.tree.DecisionTreeClassifier` to construct a simple decision tree classifier for kidney stone prediction. Sweep the maximum depth of the decision tree (parameter `max_depth`) from 1 to 10 for the tree and see how the accuracy score, F1 score, and AUC vary over the 5 folds during 5-fold cross-validation. Plot the relationship between `max_depth` and average model performance among test folds and discuss which depth you would choose for the "final" model.

For the question below: Split the data so that 80% of the data is the training dataset and 20% of the data is the test set based on label proportion of the training dataset (see `stratify` in `sklearn.model_selection.train_test_split`) and set the random state as 42.

(b) Traditional Decision Tree Plotting

Apply GridSearch 5-fold Cross Validation on `max_depth` and at least two other parameters of `sklearn.tree.DecisionTreeClassifier` on the training dataset with at least three parameter values for each of the parameters. Find the best parameters for the decision tree, train on the whole training dataset with the best parameters, and plot the tree.

Hint: You can find more information on decision tree plotting in `sklearn.tree.plot_tree`.

(c) Modern Decision Tree Construction

Please use one of the following decision tree packages: `GOSDT`, `chefboost`, `pydl8.5`. For each package, report their F1 scores and AUC Scores on the test dataset. For the constructed tree, please experiment with tuning at least 1 parameter with 5-fold cross-validation and give their average F1 score on the training data and the accuracy and F1 score for the test dataset. (See hints below.)

Hints: Available decision tree implementations include: GOSDT+guesses ("pip install gosdt"), pydl8.5 ("pip install pydl8.5"), chefboost("pip install chefboost") (other implementations are also acceptable, except scikit-learn). The most powerful tools are `GOSDT` and `DL8.5` so we suggest trying one of these if you are able to install it (Sklearn and GOSDT will allow you to calculate tree sparsity). You are able to install GOSDT in colab, if any error occurs, disconnect and reconnect to the colab server. You are encouraged to try out GridSearchCV or other cross-validation parameter search functions, you can directly use `sklearn.model_selection.GridSearchCV`. You will have to make adjustments to encoding methods depending on package requirements. For instance, `GOSDT` needs to first transform each continuous feature into a set of binary features by splitting on the mid-point of every two consecutive values realized in the dataset (i.e. `gravity`$\leq$`1.01`). You can also use `gosdt.model.threshold_guess.compute_thresholds` to generate binary features.

(d) Model Comparison

For this dataset, which implementation performed the best?