Assignment 2

Deadline: Thursday, February 15, at 11:59pm.

Submission: You need to submit the final PDF file, and any Python scripts, as a compressed folder (.zip or .tgz) on Quercus. If you used Google Colab, including a link is sufficient.

Neatness Point: One point will be given for neatness. You will receive this point as long as we don't have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Assignments are individual work. See the Course Information handout for detailed policies.

1. [**2pts**] **Decision Trees for Heart Failure Prediction.**

   The following table contains training examples that help predict whether a patient is likely to develop a heart failure.

PATIENT ID	CHEST PAIN	MALE	SMOKES	EXERCISES	HEART FAILURE
1	1	1	0	1	1
2	1	1	1	0	1
3	1	0	0	0	1
4	0	1	0	1	0
5	1	1	1	1	1
6	0	1	1	1	0

Table 1: Training Data for Decision Trees

   Which risk factor has the largest information gain at the root level of the decision tree that predicts whether or not a patient is likely to have a heart failure? **SHOW EACH STEP OF THE COMPUTATION.**

2. [**3pts**] **Revisiting patient classification with Random Forests.**

   In Assignment 1 Q4, you applied KNN, decision tree, and logistic regression to classify disease vs. not disease from patient variables. In this question, you will train a random forest model on the same dataset, and compare its results to the previous models.

   (a) [**1pts**] Use `sklearn.ensemble.RandomForestClassifier` to train a random forest classifier. Similar to how you varied the value of `k` in 4b from A1, you should try different values of `n_estimators` (10, 20, 50, 100) and make plots of training and validation accuracy. Then, report the test accuracy for the best Random Forest model. Which one of the models including KNN, Decision Tree, Logistic Regression and Random Forest with different parameters performed the best?

   (b) [**2pts**] One particularly useful attribute of Random Forests is that they provide a simple way to interpret feature importance. By default, the fitted `RandomForestClassifier` provides a method, `feature_importances_`, which gives the impurity-based feature importance. Scikit-learn provides an alternate approach, permutation-based feature importance, through the function `sklearn.inspection.permutation_importance`. Plot the sorted feature importance for your best-performing Random Forest model using each method. For `permutation_importance`, use `scoring="accuracy"` and the training set as the data (to make it consistent with the other method), and make sure to use the `importances_mean` of the returned dictionary. Do the methods agree on which features are important?

You can use this link as a resource to learn more about feature importance methods: https://mljar.com/blog/feature-importance-in-random-forest/

3. **[3pts] Binary Classification with Imbalanced Data**

   In Lecture 3 we talked about different binary classification metrics and their use cases. In this problem we will illustrate some of the strengths and limitations of these metrics.

   (a) **[1.5pts]** Scikit-learn provides tools for generating synthetic "toy" datasets. Use the `sklearn.datasets.make_classification` function to construct a dataset, with `n_samples=1000`, `n_features=20`, and `weights=[0.90,0.1]`. This will produce a dataset where the inputs are 20-dimensional vectors and labels are binary. Note that the dataset is highly imbalanced, with 90% negative examples (0s) and 10% positive examples (1s). Make a train/test split using `sklearn.model_selection.train_test_split`, where 70% of the data is for training and 30% of the data is for testing [1].

   Train a logistic regression using `sklearn.linear_model.LogisticRegression` (with default parameters) on this data, and report the model's accuracy, precision, recall, and F1 score. In addition, plot the ROC and PR curves, using `sklearn.metrics.precision_recall_curve` and `sklearn.metrics.roc_curve`. *Bonus (no credit): try implementing the scoring functions yourself!*

   (b) **[1.5pts]** Propose one way to improve your model's recall without ruining its precision. Feel free to look up common solutions to class imbalance online. Implement this change and train a new logistic regression model. Report the same metrics as in part (a) (including the plots). Were you were able to improve the recall, and how much precision did it cost you?

4. **[2pts] Iterative and Analytic Solutions of Linear Regression**

   Linear regression has an analytic solution, but it can also be solved optimally using gradient descent (with the appropriate learning rate). Using the starter code in this Google Colab, implement the analytic solution for linear regression and the solution that uses (full batch) gradient descent. For simplicity, this model does not have any bias term. You can submit a link to your completed Colab book as the answer to this question.

5. **[2pts] Computation Graph and Backpropagation** Assume we have a simple function $f(w, x, y) = (x - y)^2 * \exp(w)$. Please draw the corresponding computation graph and derive the following four derivatives: $\frac{\partial f}{\partial w}$, $\frac{\partial f}{\partial x}$,$\frac{\partial f}{\partial y}$ using backpropagation.

6. **[2pts] Cell type assignment for single-cell RNA-seq** You are given a dataset of single-cell RNA-seq which consists of more than 800 PMBC cells with 2 cell types. First, you need to implement a function that loads the data stored in the file "hw2_data.csv" and split the data into training data (70%), and test data (30%), similar to question 3a. Then:

[1] for our purposes, we are not playing with any of the model hyperparameters, so we don't need a separate validation set for hyperparameter selection

(a) **[1pts]** Train two models that use XGBoost and Multi-layer Perceptron (MLP) respectively to classify cells to their cell types, and report their classification accuracy on the test data. Note that you should use Scikit-learn to call these two models (`xgboost.XGBClassifier()` and `sklearn.neural_network.MLPClassifier`).

(b) **[1pts]** For both models, use any approach you like to select the top 30 important genes and plot out the heat-maps of these 30 genes on the test data and provide some brief comments on the results. Note that this is an open-ended question without a fixed solution.