CS589: Machine Learning - Fall 2025

Homework 5: Performance Assessment and Experiment Design

Assigned: Friday, October 24th, 2025

Getting Started: This assignment consists of coding problems. Download the assignment archive from Canvas and unzip the file. Starter code for select coding problems is provided in the code directory. For general clarification questions, please submit public posts to Campuswire. For questions related to your specific solutions, please submit private posts on Campuswire. In-person help is available through regularly scheduled office hours.

Due Date and Late Work: This assignment is due at 8:00pm ET on October 30th, 2025. Students can submit up to 11:59pm on the due date with no penalty. Work submitted up to 11:59pm on one day after the assignment is due is subject to a penalty of 10%. Work submitted up to 11:59pm two days after the assignment is due is subject to a penalty of 20%. Gradescope will close at 11:59pm two days after the assignment is due and work can not be submitted for credit after this point.

How to Submit: Your written report must be submitted to Gradescope as a PDF file. You must select the page on which each answer appears. You are encouraged to typeset your PDF solutions using LaTeX. The source of this assignment is provided to help you get started. You may also submit a PDF containing scans of *clear* hand-written solutions. Work that is illegible will not count for credit. For this assignment, code should be submitted to Gradescope as Python 3.10+ Jupyter Notebooks. Autograding will not be used for this assignment. You may submit both the code and the report as many times as you like before Gradescope closes. Only your final submission will be graded. Any late penalties will be based on the timestamp of your final submission as determined by Gradescope.

Academic Honesty Reminder: Homework assignments are individual work. Being in possession of another student's solutions, code, code output, or plots/graphs for any reason is considered cheating. Sharing your solutions, code, code output, or plots/graphs with other students for any reason is considered cheating. Copying solutions from external sources (books, web pages, etc.) is considered cheating. Collaboration indistinguishable from copying is considered cheating. Posting your code to public repositories like GitHub (during or after the course) is not allowed. Manual and algorithmic cheating detection are used in this class. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Generative AI Use Reminder: The use of generative AI tools to help with coding is permitted for programming problems in this course. The use of generative AI for all other work is considered cheating.

1. (*30 points*) **Compute evaluation metrics for classification.** Consider a medical diagnosis problem where a model has been trained to classify the respiratory illness that a patient has into three classes: Flu, COVID and Pneumonia. The model has been applied to a test set containing data for 100 patients and the following confusion matrix was produced. Use this confusion matrix to answer the following questions.

	Pred: Flu	Pred: COVID	Pred: Pneumonia
True: Flu	52	1	3
True: COVID	6	13	7
True: Pneumonia	2	4	12

a. (*2 pts*) Compute the class proportions. Explain your answer.

b. (*3 pts*) Compute the overall error rate. Show your work.

c. (*5 pts*) Compute the balanced error rate, which is defined as the average over classes of the per-class error rate. Show your work.

d. (*5 pts*) Compute the Precision for the *COVID* class. To do this, treat the COVID class as the positive class, and group the other two classes together as the negative class. Show your work.

e. (*5 pts*) Compute Recall for the *COVID* class. To do this, treat the COVID class as the positive class, and group the other two classes together as the negative class. Show your work.

f. (*5 pts*) Compute the F1 score for the *COVID* class. To do this, treat the COVID class as the positive class, and group the other two classes together as the negative class. Show your work.

g. (*5 pts*) Making different errors when diagnosing patients can have different costs. Use the cost matrix below (in dollars) and the confusion matrix given above to compute the misclassification cost. Show your work.

	Pred: Flu	Pred: COVID	Pred: Pneumonia
True: Flu	0	1500	8500
True: COVID	3000	0	10000
True: Pneumonia	10000	8000	0

2. (*25 points*) **Classifier Calibration.** A classifier is well-calibrated if its predicted probabilities reflect true empirical frequencies. In this question, you will compare the calibration properties of two models that achieve similar classification error rates, but exhibit different calibration. Starter code is in `calibration.ipynb`.

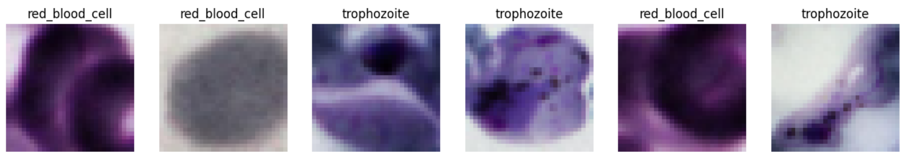
a. (*5 pts*) Using the training data, fit both a scikit-learn `LogisticRegression` model and a scikit-learn `GaussianNB` model. For the logistic regression model, use C=1.0, max_iter=1000 as hyper-parameters. Add your code to `calibration.ipynb`. Report the test error rate achieved by both models.

b. (*5 pts*) Using the scikit-learn function `calibration.calibration_curve` with `n_bins=10`, obtain the positive class frequency and average probability per bin for both models. Produce a single figure showing calibration curves for both models. Make sure to provide a legend and label axes. Add your code to `calibration.ipynb`. Describe the qualitative differences you observe between calibration curves of the two models.

c. (*10 pts*) Implement a function to compute the expected calibration error using the output from the `calibration.calibration_curve` function. Add your code to `calibration.ipynb`. Apply your function to compute and report the expected calibration error of both models.

d. (*5 pts*) What properties of the two model classes might account for the observed difference in calibration levels?

3. (*25 points*) **Imbalanced Classification.** In this problem, you will work with a real disease classification data set that is imbalanced. The problem is to detect blood cells that have been infected with the Malaria parasite. The two classes are regular red blood cells and blood cells infected with Malaria in the trophozoite stage. Each data case is a 32 x 32 color image. In the notebook `malaria.ipynb`, you are provided with starter code to load the data set as well as the implementation of a two-hidden layer MLP model that you will use to experiment with and modify. Example data cases are shown below.



a. (*5 pts*) Load the data set and produce one bar chart that visualizes the class proportions for the training data set and one bar chart that visualizes the class proportions for the test data set. Include the charts in your report and the code to compute the class proportions in `malaria.ipynb`. Based on these charts, are the test set class proportions out of distribution relative to the training set class proportions?

b. (*5 pts*) Use the provided model implementation to train a two-hidden layer MLP model on the provided training data set. Use two hidden layers of size 20, a learning rate of 0.0001, a batch size of 512 and 50 epochs. Add your code to `malaria.ipynb`. Using the test data set, compute and report the overall error rate, the error rate per class, and the balanced error rate of the learned model.

c. (*10 pts*) Update the implementation to train the model using the balanced negative log likelihood as shown below. This objective function is equivalent to computing the average over the classes of the per-class negative average log likelihood. Use weights computed from the full training data set. Train the model using the training data set. Then, using the test data set, compute and report the overall error rate, the error rate per class, and the balanced error rate. Add your code to `malaria.ipynb`.

$$bnll(\theta, \mathcal{D}) = -\sum_{n=1}^{N} w_{y_n} \cdot \log P(Y = y_n | \mathbf{X} = \mathbf{x}_n)$$

$$w_c = \frac{1}{C \cdot \sum_{n=1}^{N} \mathbb{I}[y_n = c]}$$

d. (*5 pts*) Based on your results, does learning the model using the balanced negative log likelihood improve the balanced error rate of the model? What is the impact of training using the balanced negative log likelihood on the test error rate?

4. (*20 points*) **Experiment Design.** In this problem, you will compare hyper-parameter selection using a single train-validation-test split against 5-fold cross-validation. You will use scikit-learn models and data partitioning methods. The provided data set includes a learning set and a test set. You will need to further partition the learning set into training and validation sets. Starter code is provided in `cv.ipynb`.

a. (*5 pts*) To begin, partition the learning set into a training set and a validation set. Use an 80/20 split produced by the scikit-learn `train_test_split` function with the random seed set to 42. Now conduct a train-validation-test experiment to select the regularization strength hyper-parameter α for the scikit-learn ridge regression model. Use the provided values of α. Report the best α and the corresponding train and validation MSEs. Add your code to `cv.ipynb`.

b. (*5 pts*) Next, implement a 5-fold cross validation experiment to select the α parameter of the ridge regression model. To do this, you will need to construct a scikit-learn KFold cross validation iterator using `KFold(n_splits=5, shuffle=True, random_state=42)` and then use it to construct the cross validation splits. Use the provided values of α. Report the best α and the corresponding train and cross validation MSEs. Add your code to `cv.ipynb`.

c. (*5 pts*) In the same figure, visualize the validation MSE versus α curves for the single train-val-test experiment and the 5-fold cross-validation experiment. Be sure to include a legend and label axes. Add the figure to your report.

d. (*5 pts*) Lastly, re-fit the ridge model using the whole learning data set and the optimal hyper-parameter values found using the two experiment designs. Compute and report the test set MSE of the resulting re-trained models. Add your code to `cv.ipynb`. Which experiment design gives a better test set MSE?

e. (*5 pts*) **Bonus**: Design and carry out a meta-experiment to assess the stability of the result from part (d) regarding which hyper-parameter selection experiment design gives better test set MSE with respect to the randomness in the data partitioning. Add your code to `cv.ipynb`. Explain your approach and include at least one supporting result figure in your report.

5. (*0 points*) If you used generative AI tools to help complete any programming questions on this assignment, please briefly describe which tools you used, how you used them, and for which problems you used them. If you did not use generative AI tools, please indicate that as your response to this question.