CS4013/5013 Assignment 4

Fall 2024

Due Date: Oct 20, 2024.

In this assignment, we will implement some simple machine learning processes in Python and report observations. Below is a concise description of each programming task. We will provide one template for each task. See detailed instructions in the templates.

Part I. Regression

Task 1. Implement the learning process of a regression model and report the impact of training data size on the model's prediction performance in Figure 1. Specifically, the x-axis of this figure is percent of data used for training and the y-axis is prediction error (mean-squared-error). Your figure should contain two curves: one is for training error and the other is for testing error.

Goal: you should aim to observe overfitting in Figure 1.

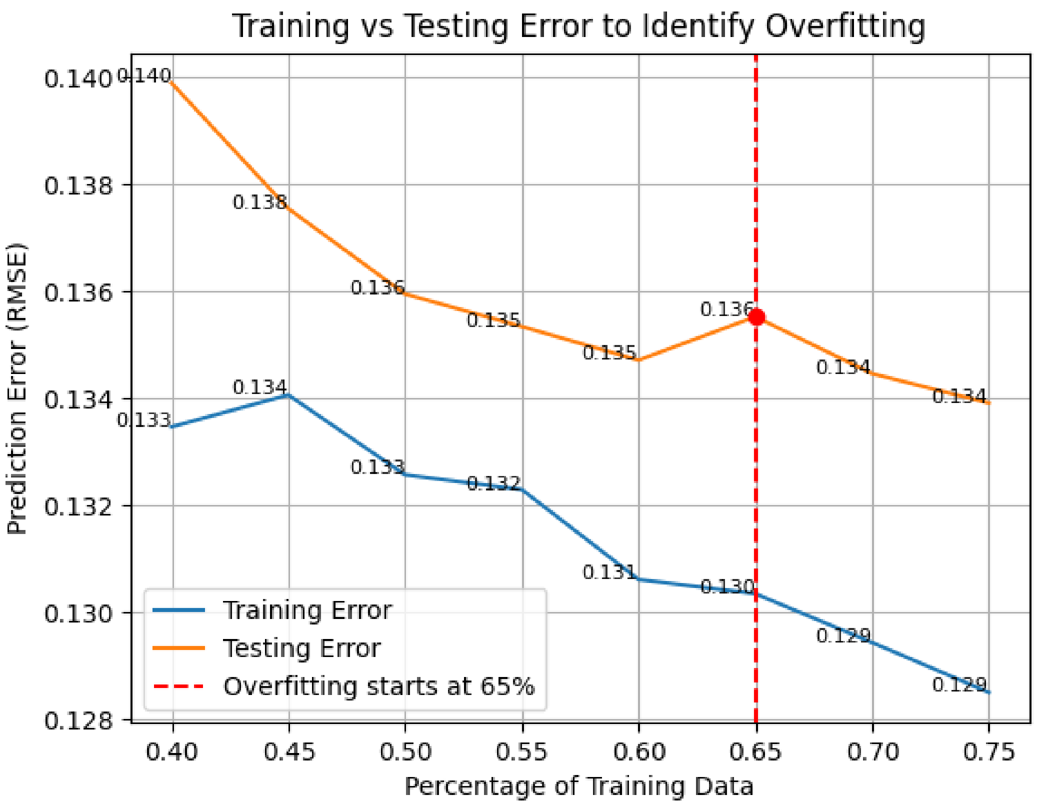

Figure 1: Training and Testing error for linear regression

Task 2. Implement the learning process of a regression model and report the impact of hyper-parameter on the model's prediction performance in Figure 2. Specifically, the x-axis of this figure is hyper-parameter value and the y-axis is prediction error. Your figure should contain two curves: one is for training error and the other is for testing error.

Goal: you should aim to observe both overfitting and underfitting in Figure 2.

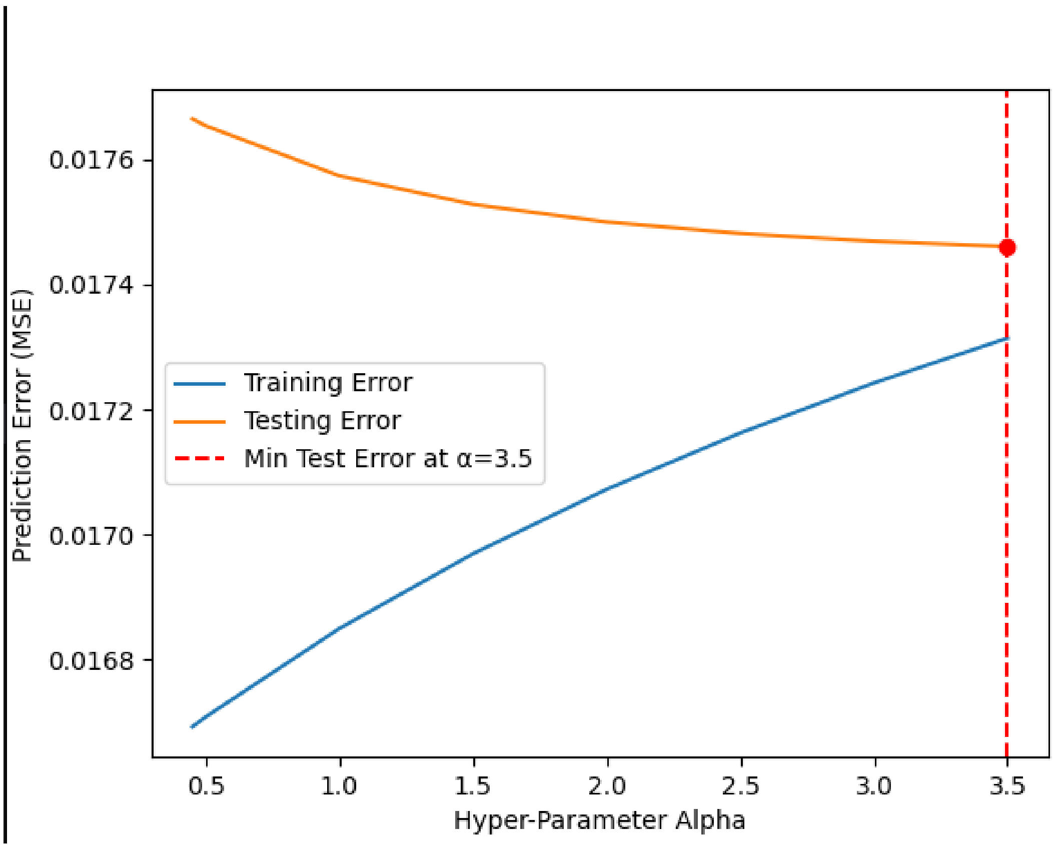

Figure 2: Training and testing error of different ranges of alpha

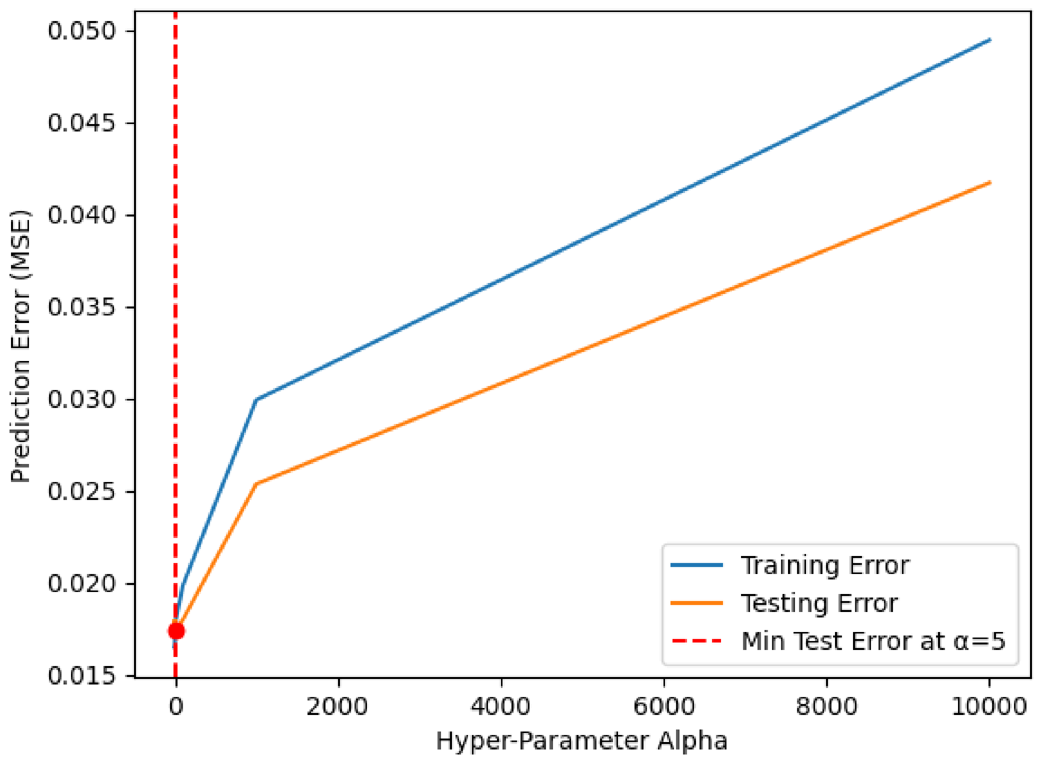

Figure 3: Training and testing error of different ranges of alpha

Task 3. Implement the k-fold cross-validation technique and apply it to select an optimal hyper-parameter for a regression model. (You need to the data splitting, training and evaluation process by yourself, not to call a cross-validation function.) Pick 5 candidate values for the hyper-parameter and report the k-fold cross validation error of each value in Table 1.

Hyper-Parameter	0.5	1	1.5	2	2.5
Validation Error	0.018972	0.01891	0.01890	0.01891	0.0189

Table 1:

Goal: your optimal error should occur when the hyper-parameter is neither too small or large.

This code implements a Ridge regression model to analyze the impact of varying the hyper-parameter alpha on validation and training errors in predicting community crime rates. Using 75% of the dataset for training and applying 5-fold cross-validation, the model evaluates how different alpha values affect error rates, identifying both overfitting and underfitting scenarios. The results are visualized in a plot, highlighting the relationship between the hyper-parameter and validation error. Finally, the optimal alpha value is selected for retraining the model on the entire training dataset and assessing its performance on the test set.

Part II. Classification

Task 4. Implement the learning process of a classification model and report the impact of training data size on the model's prediction performance in Figure 3. Specifically, the x-axis of this figure is percentage of data used for training and the y-axis is prediction error (not mean-square-error but classification error). Your figure should contain two curves: one is for training error and the other is for testing error.

Goal: you should aim to observe overfitting in Figure 3.

This code demonstrates the impact of varying training data sizes on the performance of a logistic regression model for diabetes classification. By reserving the last 25% of the dataset for testing, it evaluates how training error and testing error change with different percentages of training data (from 40% to 75%). The model identifies an overfitting point where testing error increases despite a decrease in training error, highlighting the balance between training size and generalization. The results are visualized through a plot that clearly delineates training and testing errors, along with the identified overfitting point.

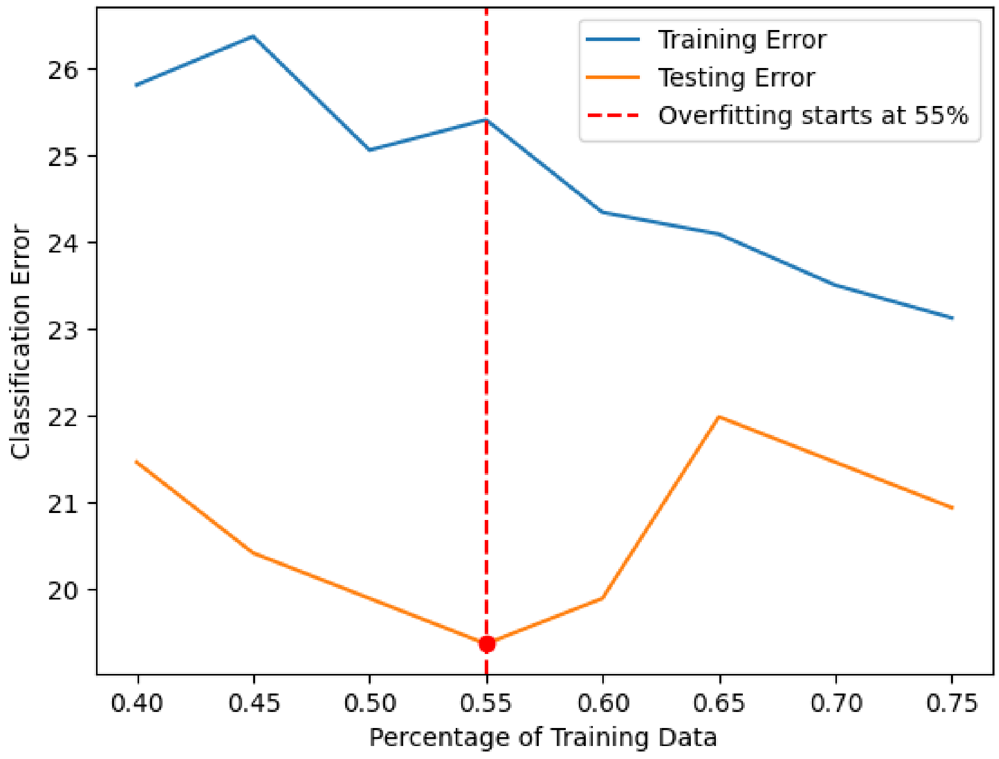

Figure 4: Trianing and testing for logistic regression

Task 5. Implement a learning process for classification model on an unbalanced data set and evaluate model performance (both classification error and AUC score).

In addition, you need to develop your own method to improve the AUC score while maintaining classification error as much as possible. A useful reference is "Haibo He and Edwardo A. Garcia, Learning from Imbalanced Data, IEEE Transactions on Knowledge and Data Engineering, 2009". (pdf is in the assignment folder)

You should report testing performance of both the baseline method and your method in Figure 4 and Figure 5. Figure 4 shows model accuracy versus training data size, while Figure 5 shows model AUC score versus training size.

Goal: your method's [**?**] AUC curve should be higher than the baseline's curve, and your method's accuracy curve should be as close to the baseline's curve as possible.

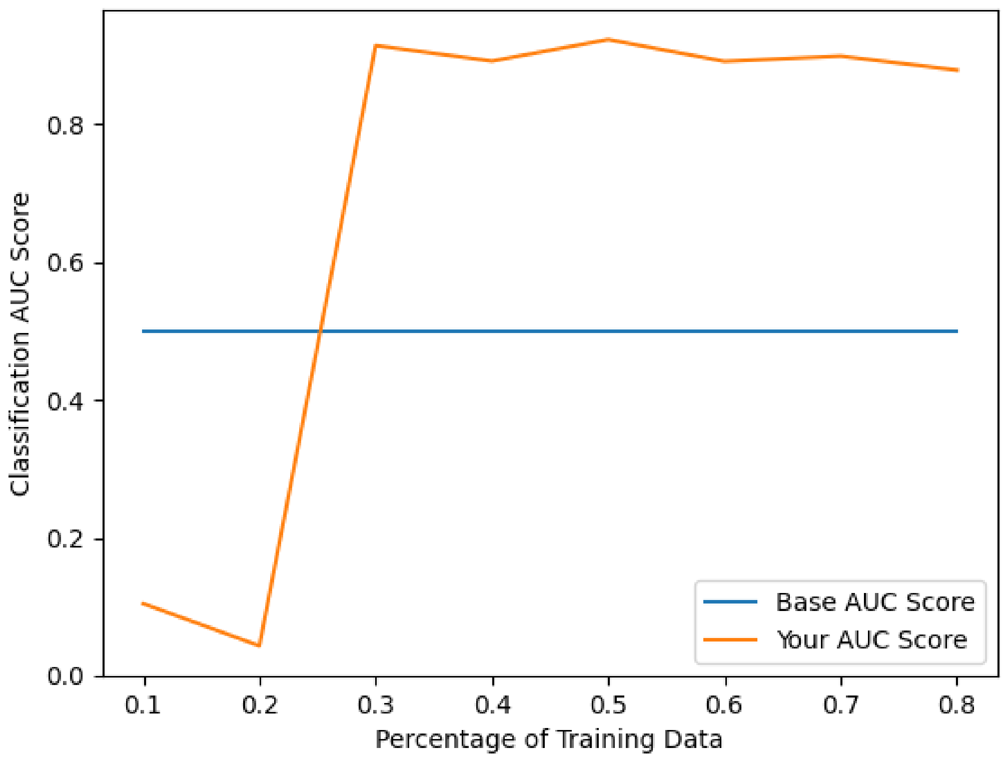

Figure 5: Training and testing AUC graph

In this code, I explore the performance of two classifiers—Logistic Regression and Support Vector Machine (SVM)—on an imbalanced dataset containing diabetes cases. We vary the percentage of training data used (from 10% to 80%) and track both accuracy and AUC scores for each classifier. The Logistic Regression model serves as a baseline for comparison, while the SVM model aims to improve upon these metrics. The results are visualized through two plots, one showing the classification accuracy and the other displaying the AUC scores as the training data percentage changes. This analysis helps illustrate the impact of different training sizes and classifier choices on predictive performance, especially in the context of imbalanced data.

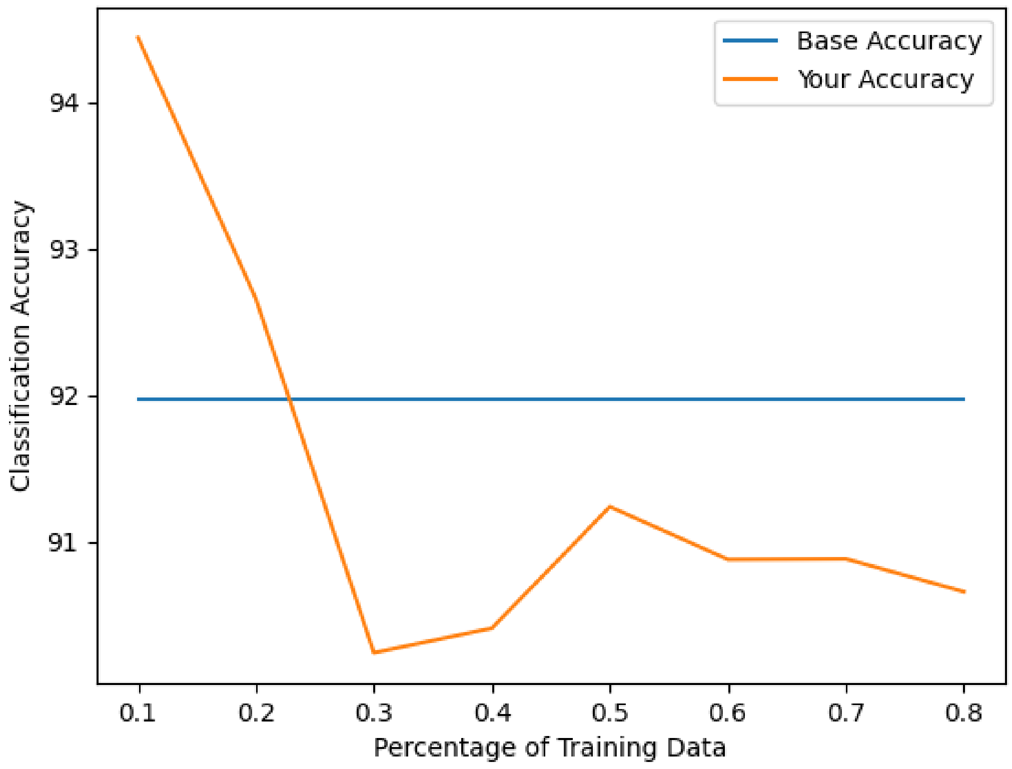

Figure 6: Training and testing accuracy graph

Bonus Point: if your method has some hyper-parameter and you can show a figure demonstrating the (reasonable) impact of that hyper-parameter on your model AUC score in Figure 6, you get a bonus point. This code evaluates the performance of Support Vector Machine (SVM) classifiers with varying hyperparameters on an imbalanced dataset consisting of diabetes cases. By setting aside 75% of the data for training, it trains the SVM with different configurations of the regularization parameter C and kernel types (linear and radial basis function). The accuracies for each configuration are computed and visualized using a bar plot to assess their effectiveness in classifying the positive and negative instances. This approach facilitates the selection of optimal SVM parameters based on training accuracy.

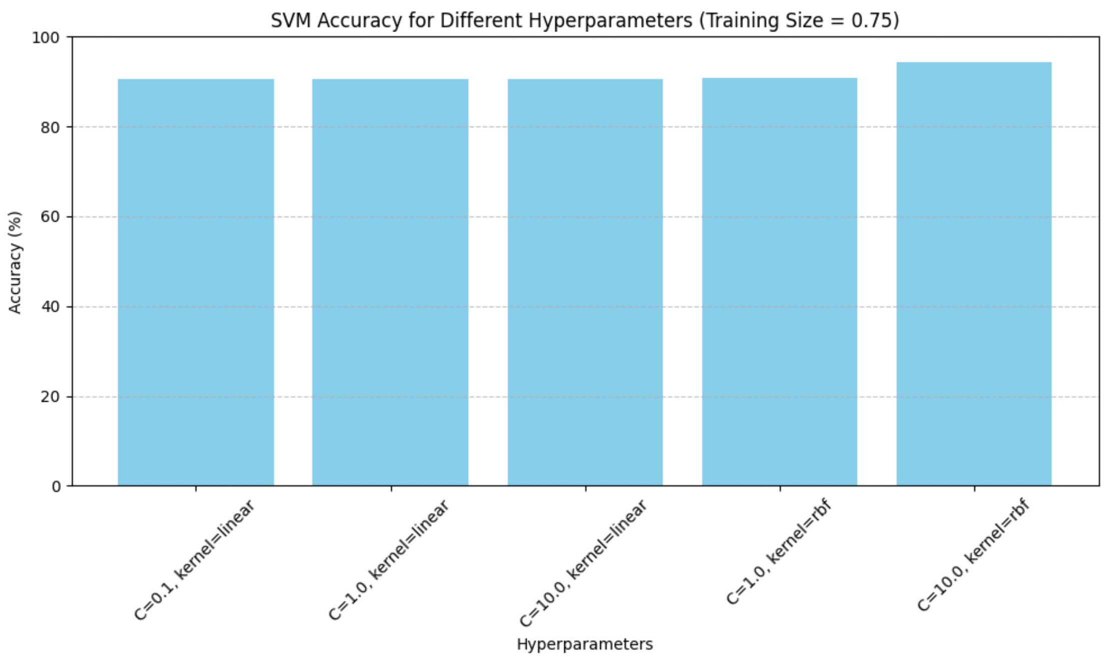

Figure 7: Impact of hyperparameters of SVM on accuracy with training size of 0.75 of whole data