Phase 2

Machine Learning Prediction

Learning Objectives

LO1. Become familiar with machine learning capabilities and tasks

LO2. Tune hyperparameters to improve model performance

LO3. Perform model selection under constraints for an applied problem

Scenario

After analyzing the telemarketing campaign data, you have been asked to use the data to help the bank prioritize the customers to call next week. The `train.csv` data is the historical data so far in the campaign, and `test_X.csv` are data on potential customers to call. You will build a machine learning tool to decide in which order the bank should call those potential customers to maximize the number of term subscriptions gathered by the end of the week. The bank has only allocated 120 call-hours for telemarketing next week.

Introduction and Deliverables

Now that you have prepared the data, you are going to use it to provide predictive analytics for the bank telemarketing campaign. There are many questions we could try to answer with the dataset, but we are going to focus on predicting whether or not someone is going to subscribe to a term deposit. Specifically, the bank has a list of approximately 8,000 clients and wants to know in what order they should be called given that there are a limited number of labor-hours available for the telemarketing campaign.

This phase is scheduled to take 6 weeks with tasks and deliverables as shown in Table 2.1.

There are three main deliverables for this portion of the capstone.

1. An organized Jupyter notebook with all specified tasks. This notebook should contain all the steps needed to train your models and provide inferences on the test set for submission.
2. Inferences on the test set. The inferences should be a single column in the same order as the test data set. You will submit your results to your group mentor in CSV format.
3. A well-structured write-up of the phase. This document will include the questions you answer throughout the Phase as well as intermediate results such as plots.

Week	Task	Deliverable
3	Build model 0	Jupyter notebook
4	Fine-tune hyperparameters of model 0 and submit inference results	Inference results
5	Review with AI Professional mentor	Document
6	Build candidate models with hyperparameter tuning	Jupyter notebook
7	Model selection and final inference results	Inference Results
8	Write-up of Phase II and review with AI Professional mentor	Document

Table 2.1: Deliverables in Phase II. Progress review weeks are highlighted.

Week 3 - Building a Simple Model

As a first step towards answering the bank's question, you are going to build a simple predictive model. This is meant to familiarize you with the scikit-learn library and making predictions with data.

Load the Data

The first step is to load data to train the model and load the test data features to provide predictions on unseen data.

1. `[Ind]` Use Pandas to read the training data into a data frame. This training data should be the output of your work in Phase I.
2. `[Ind]` Split the training data into a data frame without the column "y". These are the features you will use to predict the target variable "y".
3. `[Ind]` Make a one-column data frame of just the column "y" and encode it so that "yes" has the value 1 and "no" has the value 0. You will build a model to predict whether an observation is 1 or 0.
4. `[Ind]` Randomly split the training data into a training and validation set so that 25% of the data is held-out for validation.
5. `[Ind]` Use Pandas to read the test data into a data frame. Note that this data does not have the "y" feature value. You will provide predictions on these features to evaluate how well your model works on data that was not seen during training. Make sure that you apply the preprocessing that you did in Phase I on this data (e.g. one-hot encoding).
6. `[Ind]` Rebalance the training data so that there are equal observations in each class (equal "yes" and "no" y values). There are a few ways to do this, but we recommend using SMOTE. You should have seen in Phase I that there is a large class imbalance with only about 10% of observations falling into the "yes" class. This can be problematic when training ML models on the data because a naïve model that always predicts the majority class can score very well — 90% accuracy in this case!

Fit a Logistic Regression

The first model you will use to classify the data is an unregularized logistic regression. This is a popular model for binary classification tasks like this one, and it serves as a good baseline for a comparison against more sophisticated models. This model makes strong parametric assumptions that the log-odds of the dependent variable is a linear combination of the independent variables.

1. `[Ind]` Create a logistic regression model object with no penalty. We will discuss and use penalties next week.
2. `[Ind]` Fit the model using the training data.
3. `[Ind]` Make predictions on the test data.
4. `[Grp]` Report the accuracy of the model on the training data. Do you think that this accuracy will be similar to the accuracy on the test data? Explain your reasoning.
5. `[Grp]` Now report the accuracy of the model on the validation data. Do you think that this accuracy will be similar to the accuracy on the test data? Explain your reasoning.
6. `[Grp]` Write a function that takes a model, training data, and validation data and prints out a summary of the model performance on each set. At a minimum you should display accuracy and AUC-ROC, but you might want to consider including the built-in classification report. Evaluate your model with the function and add the result to your document.