Week 7 - Model Selection

At this stage, your group has multiple candidate models, but how should you decide which one is the best? This is something that you will encounter any time you are trying to solve a problem with machine learning. You need to be able to quantify the performance of models for the particular task. In this case, you have been given a list of ~8,000 potential customers and you need to rank them in the order that they should be called to get the most subscriptions with the constraint that the bank only has 120 call-hours. For this problem, assume that these 120 hours only apply to time on the phone with the potential customer and not on time between calls or waiting for the customer to pick-up.

Exploratory Analysis

In order to measure predictive performance with constrained call-hours, you will need information about the call durations. As a first step, you will plot the call durations in the training data set.

1. `[Grp]` Make a kernel density plot of call duration for customers that subscribed to the product.
2. `[Grp]` Make a kernel density plot of call duration for customers that did not subsribe to the product.
3. `[Grp]` Overlay the two figures and add it to your document.
4. `[Grp]` What can you conclude from the figure about the call durations? Why do you think the distributions are different?
5. `[Grp]` How do the call distributions affect the performance measures that you care about in a model? How harmful are false positives? False negatives?

Custom Scoring

Now you are going to write a custom scoring function that calculates the number of subscriptions a model would have produced given a specified call-hour constraint.

1. The function will have the following parameters.
   - Model
   - Feature matrix X
   - Target vector y
   - Call duration vector
   - Call-hour limit
2. Have the model predict the probability that each observation will subscribe to the product.
3. Sort the target and call duration vectors by the predicted class probabilities from largest to smallest.
4. Iterate through the sorted predictions and call durations to determine how many subscriptions you would have generated in the given call-hour limit.

Model Selection

Now you are going to use your scoring function on the validation set to perform model selection.

1. `[Grp]` Score the baseline model.
2. `[Grp]` Score all candidate models.
3. `[Grp]` Which model did best? How does it compare to the baseline model?
4. `[Grp]` In addition to a simple baseline model like a logistic regression, we often want to compare against a model that randomly guesses. Make a class called "RandomClassifier" that produces random guesses for binary classification (0 or 1) based on the training data class proportions. Make sure that the class implements the "predict_proba" method to be compatible with the scoring function. How do your models compare to the random classifier?

Final Model

You should now have a model that you selected as performing the best on the validation set. If you wanted, you could safely put this model into production and get reasonable predictions, but remember that you have a set of validation data that was never used to train the model. If you want to give your model the best chance to succeed on unseen data, you should refit on *all* the data.

1. `[Grp]` Rerun your winning model's entire training pipeline, including hyperparameter selection, on the entire data set.
2. `[Grp]` Use the model to predict class probabilities on the test data.
3. `[Grp]` Use the predicted class probabilities to assign the rank order (starting at 0) to the test data. These are what you will submit to evaluate your model on the test data.