Cross-Validation

Use cross-validation to evaluate parameters for XGBoost.

Chapter Goals:

- Learn how to cross-validate parameters in XGBoost

A. Choosing parameters

Since there are many parameters in XGBoost and several possible values for each parameter, it is usually necessary to *tune* the parameters. In other words, we want to try out different parameter settings and see which one gives us the best results.

We can tune the parameters using cross-validation (for a detailed explanation of cross-validation, see the **Data Modeling** section). In XGBoost, the `cv` function performs cross-validation for a set of parameters on a given training dataset.

The code below demonstrates cross-validation in XGBoost.

```
# predefined data and labels
dtrain = xgb.DMatrix(data, label=labels)
params = {
  'max_depth': 2,
  'lambda': 1.5,
  'objective':'binary:logistic'
}
cv_results = xgb.cv(params, dtrain)
print('CV Results:\n{}'.format(cv_results))
```

The output of `cv` is a pandas DataFrame (see the **Data Processing** section for details). It contains the training and testing results (mean and standard deviation) of a *K*-fold cross-validation applied for a given number of boosting iterations. The value of *K* for the *K*-fold cross-validation is set with the `nfold` keyword argument (default is 3).

The keyword argument `num_boost_round` specifies the number of boosting iterations. Each boosting iteration will try to improve the model through gradient boosting. The default number of iterations is 10.

```
# predefined data and labels
dtrain = xgb.DMatrix(data, label=labels)
params = {
  'max_depth': 2,
  'lambda': 1.5,
  'objective':'binary:logistic'
}
cv_results = xgb.cv(params, dtrain, num_boost_round=5)
print('CV Results:\n{}'.format(cv_results))
```