Hyperparameter Tuning

Apply grid search cross-validation to XGBoost models.

Chapter Goals:

- Apply grid search cross-validation to an XGBoost model

A. Using scikit-learn's `GridSearchCV`

One of the benefits of using XGBoost's scikit-learn style models is that we can use the models with the actual scikit-learn API. A common scikit-learn object used with XGBoost models is the `GridSearchCV` wrapper. For more on `GridSearchCV` see the **Data Modeling** section.

The code below applies grid search cross-validation to a binary classification XGBoost model.

```
model = xgb.XGBClassifier()
params = {'max_depth': range(2, 5)}

from sklearn.model_selection import GridSearchCV
cv_model = GridSearchCV(model, params, cv=4, iid=False)

# predefined data and labels
cv_model.fit(data, labels)
print('Best max_depth: {}\n'.format(
  cv_model.best_params_['max_depth']))

# new_data contains 2 new data observations
print('Predictions:\n{}'.format(
  repr(cv_model.predict(new_data))))
```

In the code above, we applied grid search cross-validation to a binary classification XGBoost model to find the optimal `'max_depth'` parameter (in the range from 2 to 4, inclusive). The *K*-fold cross-validation (the default for grid search) uses 4 folds. Note that the cross-validation process works the same for an `XGBRegressor` object.

After calling `fit` on `data` and `labels`, `cv_model` represents the cross-validated classification model trained on the dataset. The grid search cross-validation automatically chose the best performing `'max_depth'` parameter, which in this case was 4. The `best_params_` attribute contains the best performing hyperparameters after cross-validation.

The official XGBoost documentation provides a list of the possible parameters we can tune for in a model. A couple commonly tuned parameters are `'max_depth'` and `'eta'` (the learning rate of the boosting algorithm).