**Pipelines And Grid Search**

As mentioned earlier in the lesson, a powerful benefit to using pipeline is the ability to perform a grid search on your entire workflow.

Most machine learning algorithms have a set of parameters that need tuning. Grid search is a tool that allows you to define a “grid” of parameters, or a set of values to check. Your computer automates the process of trying out all possible combinations of values. Grid search scores each combination with cross validation, and uses the cross validation score to determine the parameters that produce the most optimal model.

Running grid search on your pipeline allows you to try many parameter values thoroughly and conveniently, for both your data transformations and estimators.

And again, although you can also run grid search on just a single classifier, running it on your whole pipeline helps you test multiple parameter combinations across your entire pipeline. This accounts for interactions among parameters not just in your model, but data preparation steps as well.

# Using Pipeline with GridSearchCV

As you may have seen before, grid search can be used to optimize hyper parameters of a model. Here is a simple example that uses grid search to find parameters for a support vector classifier. All you need to do is create a dictionary of parameters to search, using keys for the names of the parameters and values for the list of parameter values to check. Then, pass the model and parameter grid to the grid search object. Now when you call fit on this grid search object, it will run cross validation on all different combinations of these parameters to find the best combination of parameters for the model.

parameters = {

'kernel': ['linear', 'rbf'],

'C':[1, 10]

}

svc = SVC()

clf = GridSearchCV(svc, parameters)

clf.fit(X\_train, y\_train)

Awesome. Now consider if we had a data preprocessing step, where we standardized the data using StandardScaler like this.

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(X\_train)

parameters = {

'kernel': ['linear', 'rbf'],

'C':[1, 10]

}

svc = SVC()

clf = GridSearchCV(svc, parameters)

clf.fit(scaled\_data, y\_train)

This may seem okay at first, but if you standardize your whole training dataset, and then use cross validation in grid search to evaluate your model, you've got data leakage. Let me explain. Grid search uses cross validation to score your model, meaning it splits your training data into folds of train and validation sets, trains your model on the train set, and scores it on the validation set, and does this multiple times.

However, each time, or fold, that this happens, the model already has knowledge of the validation set because all the data was rescaled based on the distribution of the whole training dataset. Important factors like the mean and standard deviation are influenced by the whole dataset. This means the model perform better than it really should on unseen data, since information about the validation set is always baked into the rescaled values of your train dataset.

The way to fix this, would be to make sure you run standard scaler only on the training set, and not the validation set within each fold of cross validation. Pipelines allow you to do just this.

pipeline = Pipeline([

('scaler', StandardScaler()),

('clf', SVC())

])

parameters = {

'scaler\_\_with\_mean': [**True**, **False**]

'clf\_\_kernel': ['linear', 'rbf'],

'clf\_\_C':[1, 10]

}

cv = GridSearchCV(pipeline, param\_grid=parameters)

cv.fit(X\_train, y\_train)

y\_pred = cv.predict(X\_test)

Now, since the rescaling is included as part of the pipeline, the standardization doesn't happen until we run grid search. Meaning in each fold of cross validation, the rescaling is done only on the data that the model is trained on, preventing leakage from the validation set. As you can see, pipelines are very valuable to removing the risk of data leakage during the data preparation process.

### Note on Run Time

Running grid search can take a while, especially if you are searching over a lot of parameters! If you want to reduce it to a few minutes, try commenting out some of your parameters to grid search over just 1 or 2 parameters with a small number of values each. Once you know that works, feel free to add more parameters and see how well your final model can perform! You can try this out in the next page.