THINKFUL

Machine Learning Best Practices

Course

Curriculum

Students understand the problem of data leakage, how it occurs, and how it can be prevented.

Students are able to properly perform k-fold cross validation.

Students are able to construct pipelines combining transformers and estimators.

Students are able to grid search for optimal hyperparameters and serialize the best pipelines.

Module:

➔ Checkpoint 1 Title [linked]
➔ Checkpoint 2 Title [linked]

Agenda

- Warm-up
- Machine Learning Best Practices
- Data Leakage
- K-Fold Cross Validation
- Pipelines
- Grid Search
- Final Evaluation
- Model Persistence

Warm Up

Question: Should we perform dimensionality reduction before or after we split our data into training and test sets?

A. Before
B. After
C. Doesn’t Matter

What is the logic behind the answer you chose?

Machine Learning Best Practices

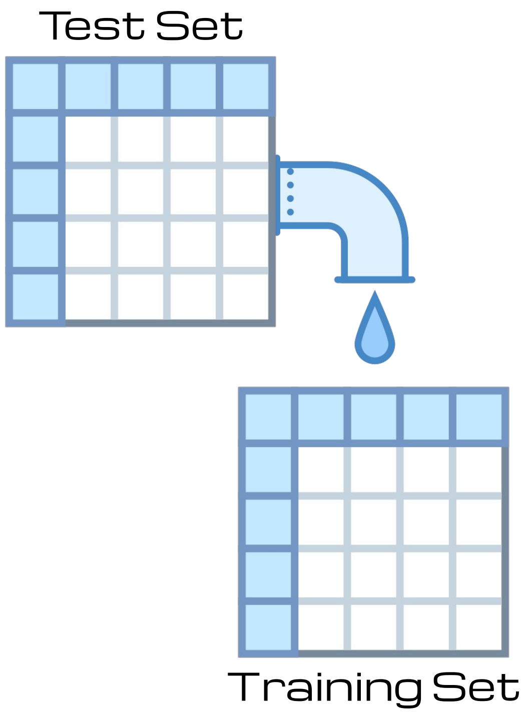
- Thus far in the program, we have experienced how Scikit-Learn makes doing machine learning relatively easy.

- The purpose of this lesson is to ensure that we train our machine learning models correctly by following some established best practices.

- These best practices will help ensure that we can achieve reliable results with a process that is consistent with what machine learning practitioners in the real world do.

Data Leakage

- Occurs when information from the validation or test set is included in the training set.
- Can result in misleading model evaluation scores, as the model is exposed to information it shouldn't have.
- Can result in poorer performance when model is applied to previously unseen data.



What Causes Data Leakage?

- Data leakage usually occurs as a result of applying transformations to the data prior to splitting it into training and testing sets.

- Types of transformations can include:
  - Feature Scaling
  - Normalization
  - Encoding (One-Hot, Label, Ordinal, etc.)
  - Discretization (Binning)
  - Dimensionality Reduction

How to Prevent Data Leakage

- The most straightforward way to prevent data leakage is to split your data into training and test sets ***before*** applying any transformations to it.
- You should fit the transformer to the training set and then call the transform method on both the training and test sets separately.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

pca = PCA(n_components=3)
pca.fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
```

Data Leakage: Difference in Results

Transform Before Split

	0	1	2
0	-0.661072	-0.578745	0.47489
1	0.304665	0.577155	-1.28999
2	-0.661072	-0.578745	0.47489
3	-0.661072	-0.578745	0.47489
4	-0.661072	-0.578745	0.47489

	precision	recall	f1-score	support
0	0.83	0.89	0.86	166
1	0.80	0.69	0.74	101
accuracy			0.82	267
macro avg	0.81	0.79	0.80	267
weighted avg	0.81	0.82	0.81	267

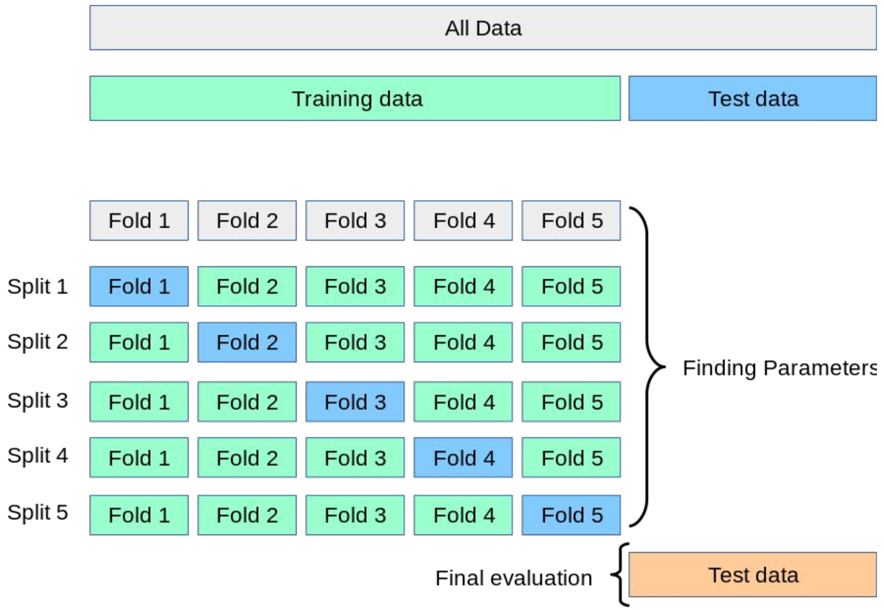
Transform After Split

	0	1	2
0	-0.669513	-0.480482	0.550273
1	0.344759	0.364040	-1.370782
2	-0.669513	-0.480482	0.550273
3	-0.669513	-0.480482	0.550273
4	-0.669513	-0.480482	0.550273

	precision	recall	f1-score	support
0	0.82	0.89	0.86	166
1	0.79	0.68	0.73	101
accuracy			0.81	267
macro avg	0.81	0.79	0.79	267
weighted avg	0.81	0.81	0.81	267

K-Fold Cross Validation

- Models should be cross-validated using the training set to evaluate how well the model performs.
- Test set should be held out until the end to ensure model generalizes well on data it hasn't seen during training.



K-Fold Cross Validation

- We can perform k-fold cross validation and evaluate our models using Scikit-Learn's **`cross_val_score`** function.
- We need to pass it our model, our X's and Y's, and the number of folds.

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X_train, y_train, cv=10)
print('Cross-Validation Score (10-folds):', scores.mean())
```

```
Cross-Validation Score (10-folds): 0.7683819764464926
```

Pipelines

- Scikit-Learn pipelines allow us to package together multiple transformers and an estimator into a single object that can be fit to a feature set.

```
pipeline = Pipeline([('pca', PCA(n_components=3)),
                     ('rf', RandomForestClassifier(random_state=1))])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

pipeline.fit(X_train, y_train)
y_preds = pipeline.predict(X_test)

print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       166
           1       0.78      0.68      0.73       101

    accuracy                           0.81       267
   macro avg       0.80      0.78      0.79       267
weighted avg       0.81      0.81      0.81       267
```

Pipelines

Cross Validation without Pipeline

```
X_pca = pca.fit_transform(X_train)

scores = cross_val_score(model, X_pca, y_train, cv=10)
print('Cross-Validation Score (10-folds):', scores.mean())
```

```
Cross-Validation Score (10-folds): 0.7683819764464926
```

Cross Validation with Pipeline

```
scores = cross_val_score(pipeline, X_train, y_train, cv=10)
print('Cross-Validation Score (10-folds):', scores.mean())
```

```
Cross-Validation Score (10-folds): 0.7603430619559652
```

GridSearch Parameter Tuning

- Machine learning models have hyperparameters that affect various aspects of how they learn from the training data.
- Scikit-Learn's **`GridSearchCV`** function allows us to try different parameters values for each stage in a pipeline and find which ones produce the best cross validation score.
- The best score can be obtained by calling the **`best_score_`** method, the best parameter values can be obtained by calling the **`best_params_`** method, and the best estimator can be obtained by calling the **`best_estimator_`** method.

GridSearch Parameter Tuning

```
pipeline = Pipeline([('pca', PCA()),
                     ('rf', RandomForestClassifier(random_state=1))])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

parameters = {'pca__n_components': [2, 3, 4, 5, 6, 7, 8],
              'rf__n_estimators': [10, 20, 50, 100, 200],
             }

search = GridSearchCV(pipeline, parameters, cv=10)
search.fit(X_train, y_train)

print("Best parameter CV score: %0.3f" % search.best_score_)
print("Best parameters:", search.best_params_)
```

```
Best parameter CV score: 0.773
Best parameters: {'pca__n_components': 8, 'rf__n_estimators': 200}
```

Final Evaluation

- Once we finish cross validating and obtain the best estimator possible, we can perform a final evaluation on the test set.

```
pipeline = search.best_estimator_

pipeline.fit(X_train, y_train)
y_preds = pipeline.predict(X_test)

print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.86       166
           1       0.79      0.68      0.73       101

    accuracy                           0.81       267
   macro avg       0.81      0.79      0.79       267
weighted avg       0.81      0.81      0.81       267
```

Model Persistence

- The final step is to train our best estimator on the full data set and then save the model so it can be applied to new data in the future.
- We can use the pickle library to save the model in a serialized fashion by calling pickle’s `dump` method and passing it our pipeline and an open file where we want it to store the estimator.

```
import pickle

pipeline.fit(X, y)

with open('model.pkl', 'wb') as f:
    pickle.dump(pipeline, f)
```

Model Persistence

- When we receive new data that we want to apply the estimator to, we can simply call picke's **`load`** method and pass it the open file from which we want it to load the estimator.
- We can then call predict on the loaded pipeline to generate predictions on the new data.

```
with open('model.pkl', 'rb') as f:
    loaded_pipe = pickle.load(f)

preds = loaded_pipe.predict(new_data)
```

Summary

In this session, we covered:

- What data leakage is, how it occurs, and how it can be prevented.
- What k-fold cross validation is and how to properly perform it.
- How to construct pipelines that package together transformers and estimators.
- How to grid search hyperparameters to optimize our pipelines.
- How to save our models in a serialized fashion so that they can be loaded at a later date and applied to new data.

Assignment

- Practice combining train-test split, cross validation, grid search CV, pipelines, and pickle to create one solid modeling workflow.
- Use the Pima Indians diabetes dataset.

Notebook

THINKFUL

Thank You