Python Data Products

Course 2: Design thinking and predictive pipelines

Lecture: Introduction to Training and Testing

Learning objectives

In this lecture we will...

- Introduce the concepts of **training versus testing**
- Discuss the importance of evaluating models on **unseen** data
- Show how to adjust our Python code to make use of these ideas

Training and testing?

In the previous lecture we saw that we can obtain good performance with a simple classifier, but highlight a possible issue:

If we **evaluate** a system on the same data used to **train** the system, we may overestimate its performance

Really, we want to know how well a method is likely to work on **unseen data**

Training and testing?

To estimate how well a system is likely to perform on new data, we can split our dataset in to two components:

- A **training set** to train the machine learning model
- A **test set** used to estimate the performance on new data

We'll investigate both of these ideas more in **Course 3**, but for the moment, let's quickly explore how we can adapt our previous code to incorporate these two components

Code: Training and testing

First we read the dataset, exactly as we did in previous lectures:

In [1]:
```
f = open("/home/jmcauley/datasets/mooc/5year.arff", 'r')
```

In [2]:
```
while not '@data' in f.readline():
    pass
```

In [3]:
```
dataset = []
for l in f:
    if '?' in l:
        continue
    l = l.split(',')
    values = [1] + [float(x) for x in l]
    values[-1] = values[-1] > 0 # Convert to bool
    dataset.append(values)
```

Code: Training and testing

The first thing we do differently is to **shuffle** the data:

```
In [4]: import random

In [5]: random.shuffle(dataset)

In [6]: X = [values[:-1] for values in dataset]

In [7]: y = [values[-1] for values in dataset]
```

We do this because we we want the training and test set to be **random samples** of the data – if we didn't use random samples, different subsets of the data could have distinct characteristics that could cause the model to under- (or over) perform on one of them

Code: Training and testing

Next we split the data into a **train** and a **test** portion

```
In [8]: N = len(X)
        X_train = X[:N//2]
        X_test = X[N//2:]
        y_train = y[:N//2]
        y_test = y[N//2:]
```

```
In [9]: len(X), len(X_train), len(X_test)

Out[9]: (3031, 1515, 1516)
```

Note that we split both the data (*X*) and the labels (*y*) in the same way

Code: Training and testing

Next we train our model as before, but we use **only the training data and labels**

```
In [10]: from sklearn import linear_model

In [11]: model = linear_model.LogisticRegression()

In [12]: model.fit(X_train, y_train)

Out[12]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

Code: Training and testing

Finally we can compute the accuracy of the model, but this time we do so separately for the training and test portions

```
In [13]: predictionsTrain = model.predict(X_train)
         predictionsTest = model.predict(X_test)
```

```
In [14]: correctPredictionsTrain = predictionsTrain == y_train
         correctPredictionsTest = predictionsTest == y_test
```

```
In [15]: sum(correctPredictionsTrain) / len(correctPredictionsTrain) # Training accuracy

Out[15]: 0.9630363036303631
```

```
In [16]: sum(correctPredictionsTest) / len(correctPredictionsTest) # Test accuracy

Out[16]: 0.9637203166226913
```

The latter quantity measures **how well the model is likely to perform on new data**

Summary

This lecture presented a very brief introduction to training and testing. Though we'll cover more in Course 3, the basic concepts are:

- Simply training on a dataset doesn't give us a sense of how a model will **generalize to new data**
- This generazilation ability can be estimated using a test set
- Training and test sets should be **non-overlapping, random** splits of our data

Summary of concepts

- Introduced the concepts of training and testing sets
- Briefly described the difference between training performance versus generalization ability
- Showed how to adapt our classification code to measure performance on the training set

On your own...

- Try repeating this exercise for our **regression** example from the previous lecture, i.e., split the data into training/testing portions and measure its training and testing performance