STAT 380: Evaluating Prediction Performance of a Supervised Learning Procedure

An Example using Linear Regression on BostonHousing.csv Data

Prediction Using a Statistical/Machine Learning Model

Let $\{y_i, \mathbf{X}_i\}_{i=1}^n$ be the observed data. And there is a statistical/machine learning model that provides a prediction for the responses y_i.

We denoted the predicted value for the responses to be $\hat{y}_i$

Measures Prediction Accuracy

Measuring the Prediction Performance

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Here n denotes the number of predicted values from the model. Smaller values for **RMSE** indicates better fit

$$R^2$$

Larger Values for R^2 indicats better fit.

Over-fitting in Supervised Learning

Types of statistical machine learning algorithms (to remember)

Supervised: The algorithm needs that the data scientist acts as a guide to teach the algorithm to which conclusions it should come. It works with explicit inputs and the desired outputs. The y is known and is split into:

- *training* data contain outcomes to train the machine.
- *validation* data are used for select the best performing approach.
- *test* are used for making predictions, which have no outcomes to predict them.

  $\Longrightarrow$ Classification, **regression models**, discriminant analysis, etc.

Unsupervised: The algorithm is able to learn to identify complex structures and patterns of data sets without a data scientist or without using explicitly-provided labels.

Classification vs. Prediction (to remember)

Supervised learning algorithms can be divided into 2 categories: Classification & Prediction (=>Regression)

Classification: Examine data where the classification is unknown, with the goal of predicting what that classification is. Similar data where the classification is known are used to develop rules. Predicts categorical class labels. Examples:

- Recipient of an offer can respond or not respond.
- Bus can be available for service or unavailable.

Prediction: is similar to classification, except that we are trying to predict the value of a numerical variable (e.g., amount of purchase) rather than a class (e.g., purchaser or non-purchaser).

Evaluating predictive performance

Key question: How well will our prediction or classification model perform when we apply it to new data?

We are particularly interested in comparing the performance of different models so that we can ***choose*** *the one we think will do the best when it is implemented in practice.*

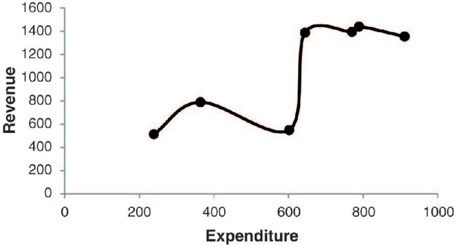
To assure that the chosen model generalizes beyond the current dataset, we

a) use the concept of ***data partitioning*** and
b) try to avoid ***overfitting***.

Overfitting: illustration

The more variables we include in a model, the greater the risk of overfitting the particular data used for modeling. **What is overfitting?**

Example: Advertising expenditures in one period vs sales in a subsequent period

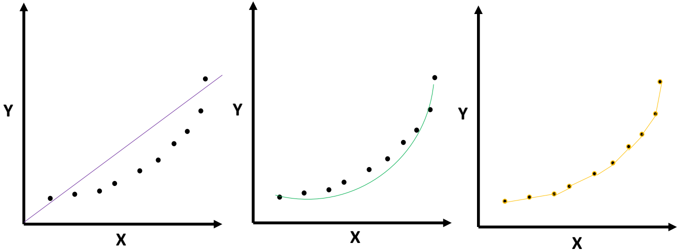


- We could connect points with a smooth **interpolation** - no errors.
- We see that such a curve is **unlikely** to be accurate, or even useful, in predicting future sales.

Overfitting:

- Purpose of building a model is to represent relationships among variables in such a way that this model will do a good job of predicting **future** outcome values based on future predictor values.
- A **simple** straight line might do a better job than the complex function in terms of predicting future sales on the basis of advertising.
- Instead, we devised a **complex** function that fit the data perfectly.
- We **ended up** modeling some variation in the data that is nothing more than chance variation.
- We **mistreated** the noise in the data as if it were a signal.

Overfitting illustration



Linear regression:	Non-linear regression:	Points connection:
- Underfitting	- Desired	- **Overfitting**
- High bias	- Trade-off between variance and bias	- High variance

Overfitting: estimate likely performance

Initial idea:

- Maximizing training accuracy rewards overly complex models.
- We can reach a 100% accuracy but we are not able to generalize well.

Causes:

- The model contains too many predictors (**complexity**).
- The **data set** is too noisy or too small.
- The model has being **refined** over time, but no new data inputs are provided.

Alternative idea:

- We can split the initial data set into different sets so that the model can be trained and tested on different data.
- Testing accuracy is a **preferable** than training accuracy.

Evaluating Prediction Performance of a Supervised-Learning Method

Creation and use of data partitions

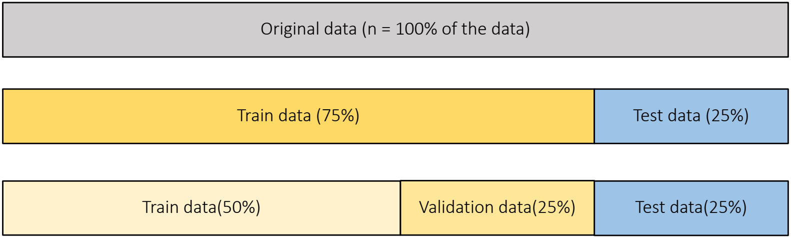
- When we use the same data both to develop the model and to assess its performance, we introduce an ***optimism bias***.
- To address this (overfitting) problem, we simply **partition** our data and develop our model using only one of the partitions.
- After we have selected a model, we try it out on another partition and see how it performs.

Two or three data sets for evaluation

- **Training data**, typically the largest partition, contains the data used to build the various models. The same training partition is generally used to **develop** multiple models.
  Train denotes the set of elements with $|Train| = n_t$.
- **Validation data** is used to **compare** the predictive performance of each model and choose the best one. Sometimes the validation partition may be used in an automated fashion to tune and improve the model.
  Vali denotes the set of elements with $|Vali| = n_v$.
- **Test data** is used to **assess** the performance of the chosen model with new data. *Test* denotes the set of elements with $|Test| = n_{test}$.

Classic partition

In general: $n_t + n_v + n_{test} = n$, where n is the size of the data.



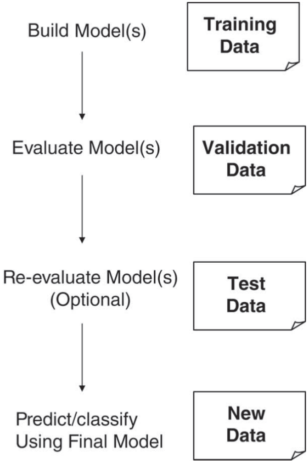
Validation vs. test data sets

Why have both a validation and a test partition?

- We use the validation data to **assess multiple models** and then choose the model that performs best with the validation data.
- The performance of the chosen model on the validation data will be overly **optimistic**.
- Applying the model to the test data, which it has not seen before, will provide **an unbiased estimate** of how well the model will perform with new data.

When we are concerned mainly with **finding** the best model and less with exactly how well it will do, we might use only training and validation partitions.

Data partitions and their role in the data mining process



The Boston Housing Dataset

Boston Housing data set

Boston Housing data set

- The Boston Housing data contain information on census tracts in suburbs of Boston.
- Several measurements are included (e.g., crime rate, pupil-teacher ratio).
- 14 variables for each of the 506 houses.

Possible tasks:

- A supervised predictive task, where the outcome is the median value of a home.
- A supervised classification task, where the outcome is the binary variable *CAT.MEDV* that indicates whether the home value is above or below $\$30,000$.
- An unsupervised task, where the goal is to cluster houses.

Variables in the Boston housing data set

Variable	Name
Crime rate	CRIM
Percentage of residential land zoned for lots over 25,000 ft^2	ZN
Percentage of land occupied by nonretail business	INDUS
Does tract bound Charles River (= 1 if tract bounds river)	CHAS
Nitric oxide concentration (parts per 10 million)	NOX
Average number of rooms per dwelling	RM
Percentage of owner-occupied units built prior to 1940	AGE
Weighted distances to five Boston employment centers	DIS
Index of accessibility to radial highways	RAD
Full-value property tax rate per $10,000	TAX
Pupil-to-teacher ratio by town	PTRATIO
Percentage of lower status of the population	LSTAT
Median value of owner-occupied homes in $1000s	MEDV
Is median value of owner-occupied homes in tract above $30,000 (CAT.MEDV = 1) or not (CAT.MEDV = 0)	CAT.MEDV

Boston housing data set: Overview

The `head()`-command returns the first parts of a vector, matrix, table, data frame or function.

```
> head(Daten)
 CRIM ZN INDUS   RM  AGE   DIS RAD TAX LSTAT MEDV CMEDV
0.006 18  2.31 6.57 65.2 4.090   1 296  4.98 24.0     0
0.027  0  7.07 6.42 78.9 4.967   2 242  9.14 21.6     0
0.027  0  7.07 7.18 61.1 4.967   2 242  4.03 34.7     1
0.032  0  2.18 6.99 45.8 6.062   3 222  2.94 33.4     1
0.069  0  2.18 7.14 54.2 6.062   3 222  5.33 36.2     1
0.029  0  2.18 6.43 58.7 6.062   3 222  5.21 28.7     0
```

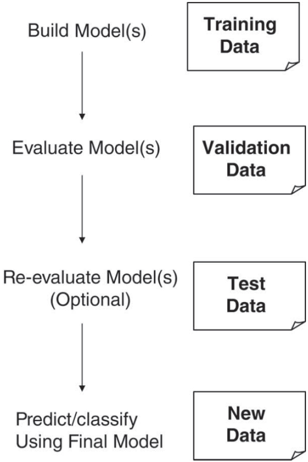
Boston housing data set: Overview

The `str()`-command displays the internal structure of an R object.

```
> str(Daten)
'data.frame': 506 obs. of  14 variables:
 $ CRIM      : num  0.00632 0.02731 0.02729 0.03237 ...
 $ ZN        : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ INDUS     : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 ...
 $ CHAS      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NOX       : num  0.538 0.469 0.469 0.458 0.458 ...
 $ RM        : num  6.58 6.42 7.18 7 7.15 ...
 $ AGE       : num  65.2 78.9 61.1 66.6 96.1 100 85.9 ...
 $ DIS       : num  4.09 4.97 4.97 6.06 6.06 ...
 $ RAD       : int  1 2 2 3 3 3 5 5 5 5 ...
 $ TAX       : int  296 242 242 311 311 311 ...
 $ PTRATIO   : num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 ...
 $ LSTAT     : num  4.98 9.14 4.03 2.94 5.33 ...
 $ MEDV      : num  24 21.6 34.7 33.4 36.2 28.7  ...
 $ CAT..MEDV: int  0 0 1 1 1 0 0 0 0 0 ...
```

We practice to implement the procedure using R. We will consider the Boston Housing Data as an Example.

Data partitions and their role in the data mining process



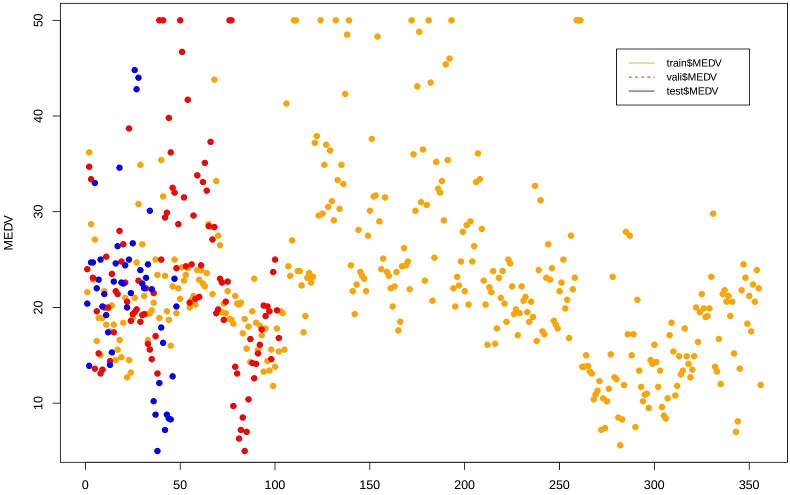
Data partitioning in R

There are several ways to partition the data - one option with the `createDataPartition()`-command:

```
> # Loading libraries and the data
> library(caret)
> Daten<-read.csv("BostonHousing.csv")

> # Split data in 70% Training, 20% Validation, 10% Test
> inTrain <- createDataPartition(Daten$CRIM, p = 0.7,
  list = FALSE)
> train <- Daten[inTrain, ]
> inVali <- createDataPartition(Daten$CRIM[-inTrain], p =
   0.666, list = FALSE)
> vali <- Daten[-inTrain,][inVali, ]
> test <- Daten[-inTrain,][-inVali, ]
```

Data partitioning



Thank You