Introduction to Machine Learning

M390

Dave Miller

1/23/2019

Machine Learning

- **Machine learning** is a data analytics technique that teaches computers to do what comes naturally to humans and animals: learn from experience.

- Algorithms use computational methods to “learn” information directly from data without relying on a predetermined equation as a model.

Types of Machine Learning

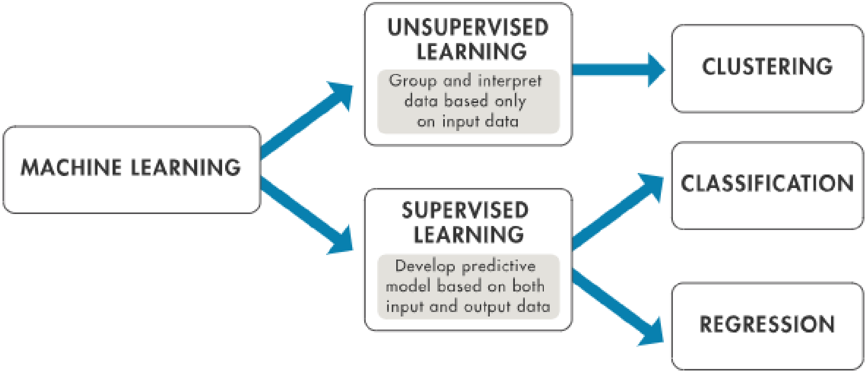

MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Figure 1. Machine learning techniques include both unsupervised and supervised learning.

Figure 1:

Supervised Learning

- **Supervised machine learning** builds a model that makes predictions based on evidence in the presence of uncertainty.
- A supervised learning algorithm takes a known set of indicators (input data) and **known responses** to the data (output) and trains a model to generate reasonable predictions for the response to new data:

$$Y \sim f(X)$$

- Use supervised learning if you have known data for the output you are trying to predict.

Regression

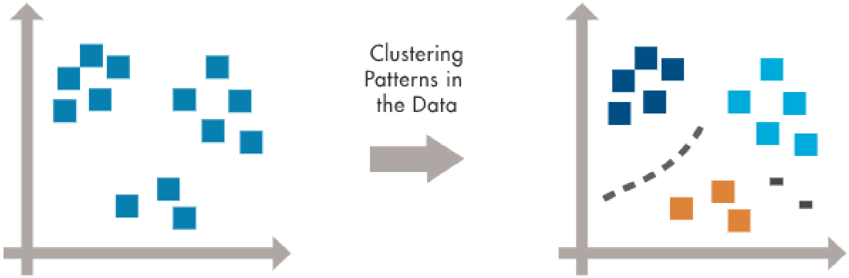
- **Regression** techniques predict continuous responses:
  - House (or anything else) price,
  - Scores of a game,
  - Test score,
  - etc.
- Use regression techniques if you are working with a data range or if the nature of your response is a real number.
- Common regression algorithms include linear regression, nonlinear models, regularization, stepwise regression, boosted and bagged decision trees, neural networks.

Classification

- Classification techniques predict discrete responses:
  - whether a tumor is cancerous or benign,
  - Yes or No,
  - Win or loss,
  - Pass or fail.
  - etc.
- Use classification if your data can be tagged, categorized, or separated into specific groups or classes.
- Common classification algorithms include logistic regression, support vector machine (SVM), boosted and bagged decision trees and neural networks.

Unsupervised Learning: Clustering

- **Unsupervised learning** finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data **without labeled responses.**
- **Clustering** is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data.



Preparing for Regression Problems: R packages

- We will use the tidyverse package extensively throughout the course. More information ca be found at http:https://www.tidyverse.org/.
- To install tidyverse, use `install.packages(tidyverse)`. If you already have tidyverse installed, run `library(tidyverse)`. The first two lines of all of my R Scripts look like the following:

```r
#install.packages(tidyverse)
library(tidyverse)
```

Preparing for Regression Problems: Getting data

- Preloaded data in R:
  - View all by typing `data()`
  - Example: mtcars
- Loading your own data from my GitHub:

```
ex_1 <-
read.csv('https://raw.github.com/davmiller/M390/master/
data/example_data_1.csv')
```

Preparing for Regression Problems: Looking at the data

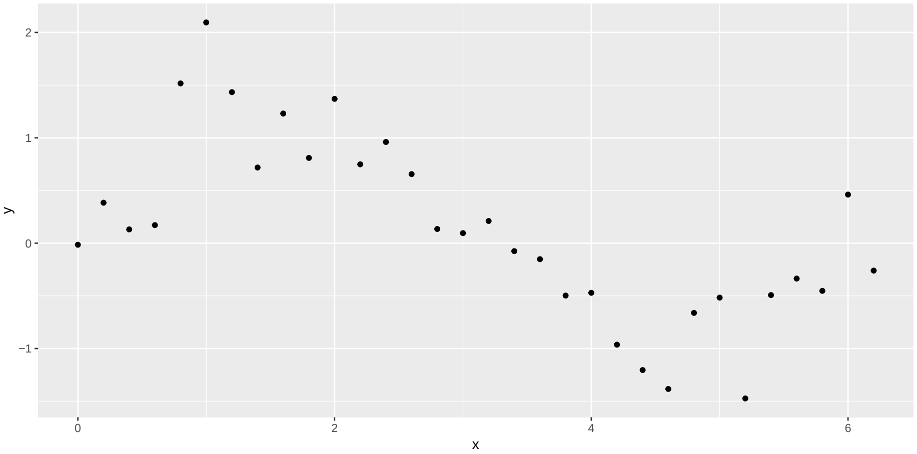
```r
# Look at the first 6 rows.
head(ex_1)
```

```
##     x           y
## 1 0.0 -0.01514382
## 2 0.2  0.38408346
## 3 0.4  0.13086018
## 4 0.6  0.17121048
## 5 0.8  1.51642042
## 6 1.0  2.09502139
```

- `View(ex_1)` to view entire dataset.

Plotting the data

```r
# Plot using ggplot
ggplot(data=ex_1, aes(x=x,y=y))+
  geom_point()
```



Preparing for Regression Problems: Data Splitting

- Main goal of regression is to find an algorithm $f(X)$ that most accurately predicts future values (or responses) Y based on a set of inputs (or indicators) X:

$$Y \sim f(X)$$

- We want an algorithm that not only fits well to our past data, but more importantly, one that predicts a future outcome accurately. This is called the **generalizability** of our algorithm.

Data Splitting

- To provide an accurate understanding of the generalizability of our final optimal model, we split our data into training and test data sets:
  - **Training Set**: these data are used to train our algorithms and tune hyper-parameters.
  - **Test Set**: having chosen a final model, these data are used to estimate its prediction error (generalization error). These data should not be used during model training!

- Given a fixed amount of data, typical recommendations for splitting your data into training-testing splits include 60% (training) - 40% (testing), 70%-30%, or 80%-20%.

Data Splitting

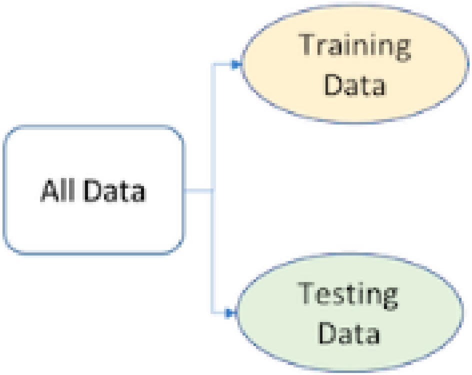


Figure 3: Splitting data into training and test sets

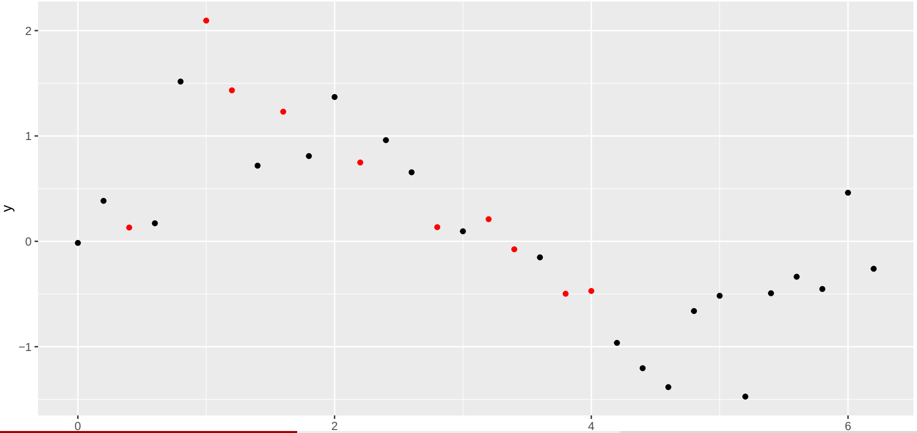
Simple random splitting

- The simplest way to split the data into training and test sets is to take a simple random sample.

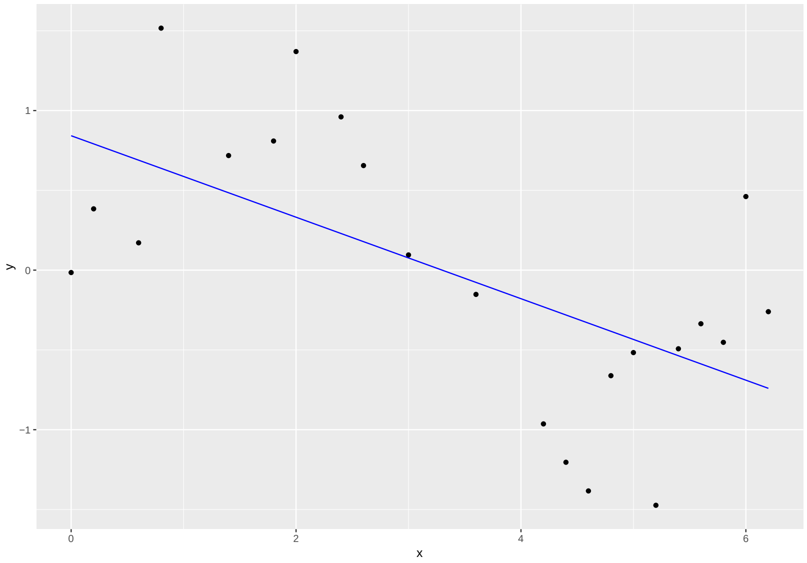
```r
# 70% random train / test split
set.seed(123)
index   <- sample(1:nrow(ex_1), round(nrow(ex_1) * 0.7))
train_1 <- ex_1[index, ]
test_1  <- ex_1[-index, ]
```

Plotting train / test sets

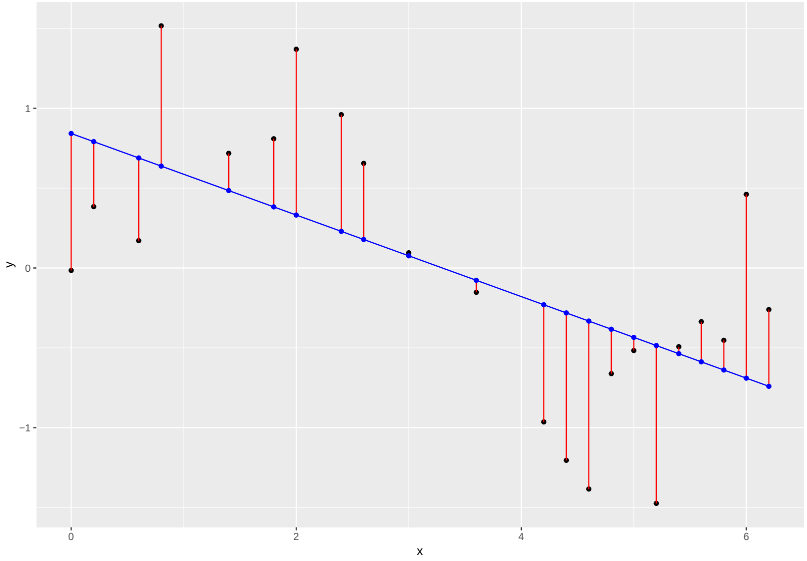
```
# Train in black, test in red.
ggplot()+
  geom_point(data=train_1, aes(x=x,y=y), color='black')+
  geom_point(data=test_1, aes(x=x,y=y), color='red')
```



Models trained on training set



Evaluating a regression model



```
## [1] "Train MSE: 0.4197"
```

Evaluating a regression model

- Mean squared error (**MSE**) is the average of the squared error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2.$$

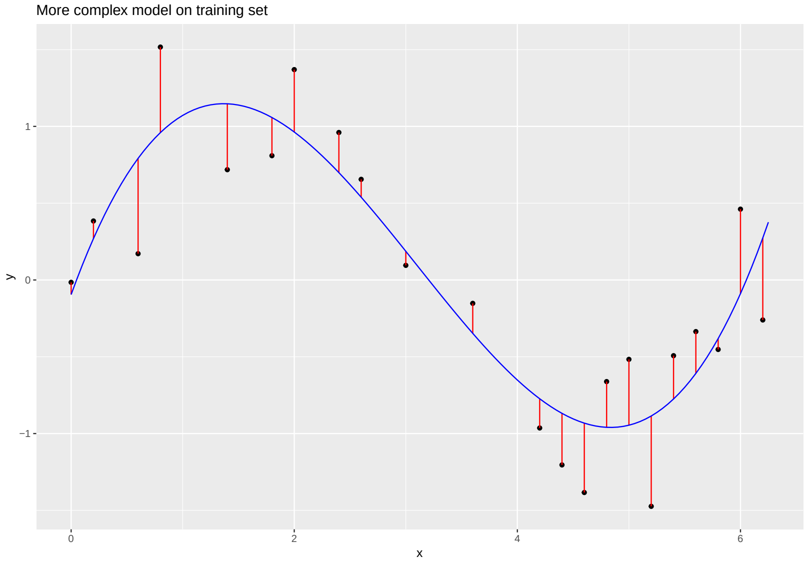
Objective: minimize

- Root mean squared error (**RMSE**) simply takes the square root of the MSE:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2}.$$

Objective: minimize

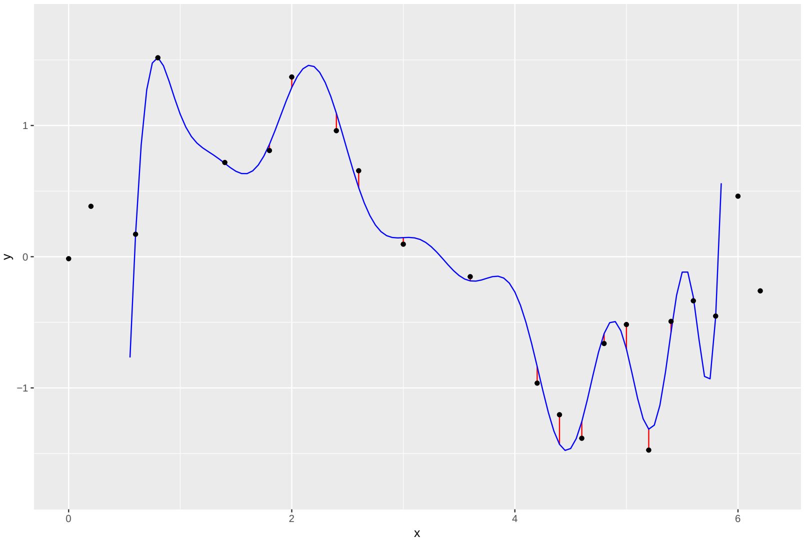
Other models



```
## [1] "Train MSE: 0.1351"
```

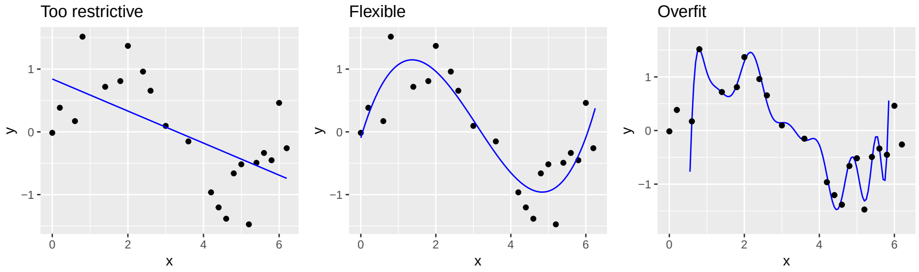
Other models

Even more complex model on training set



```
## [1] "Train MSE: 0.0092"
```

Tuning a model



Overfitting

- **Overfitting** refers to a model that models the training data too well.
- Overfitting occurs when the evaluation metric on the training set is much better than on the test set.
- Underfitting refers to a model that can neither model the training data nor generalize to new data.

Overfitting

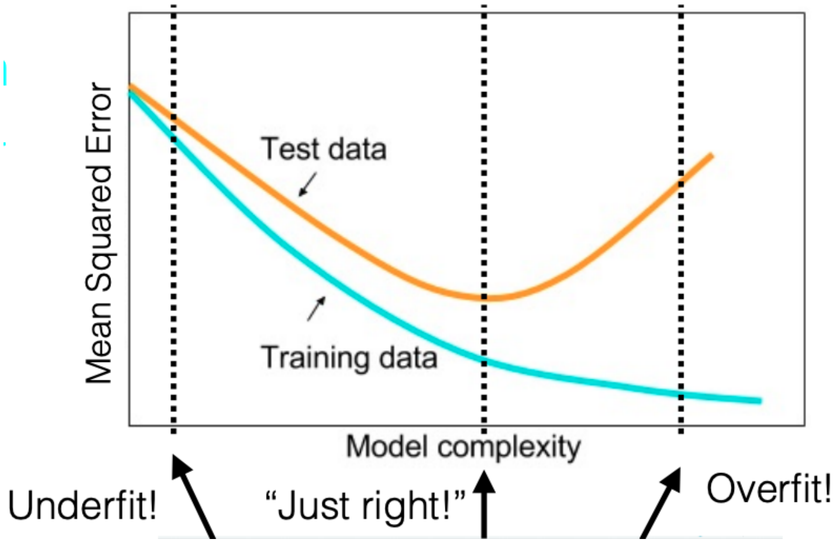
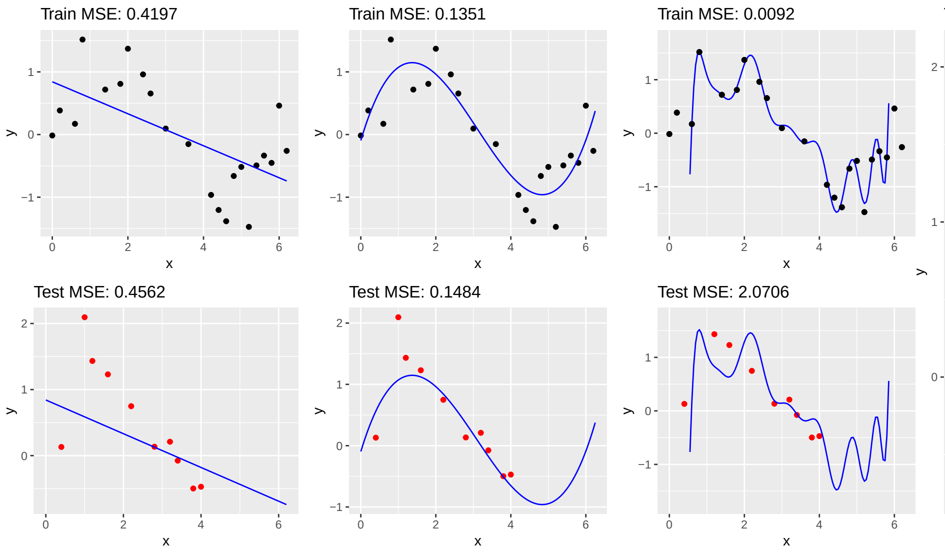


Figure 4: Finding the balance

Overfitting



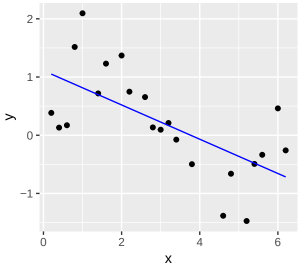
Bias-variance tradeoff

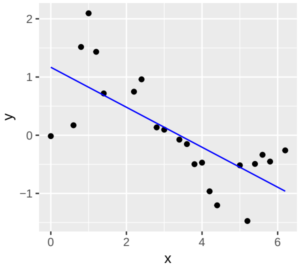
- A model that is rigid and consistent, which doesn't change much at all when provided a new training sample is said to have high **bias**.

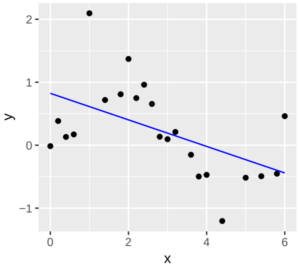
- A model that changes significantly with small changes to the training sample is said to have high **variance**.

Bias

Three different training sets from same base set.

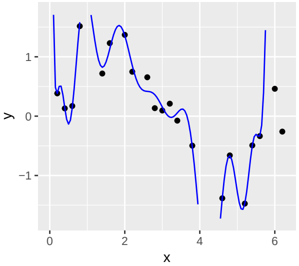


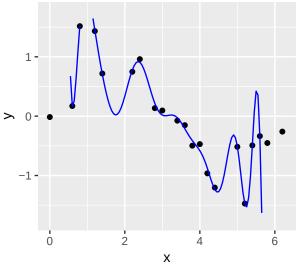


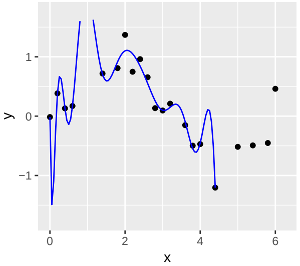


Variance

Three different training sets from same base set.

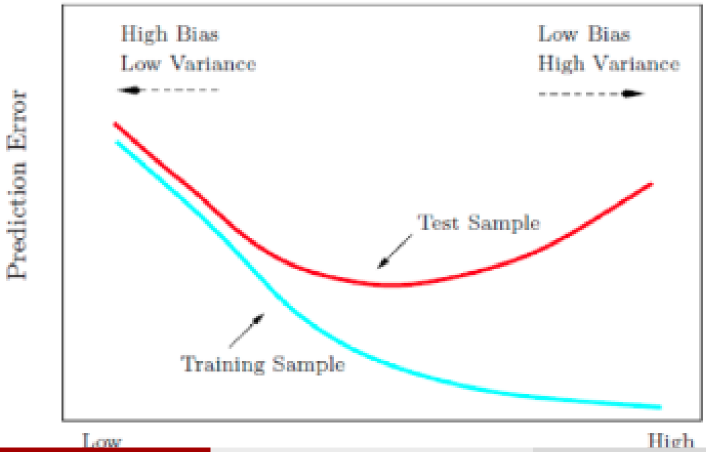






Bias-variance tradeoff

We want a model that balances bias and variance, and likely will minimize the error on future unseen data compared to the high bias or high variance models.



Cross valdiation

- To find the model that balances the bias-variance tradeoff, we search for a model that minimizes a k-**fold cross-validation error metric**.
- k-fold cross-validation is a resampling method that randomly divides the training data into k groups (aka folds) of approximately equal size.
- The model is fit on $k - 1$ folds and then the held-out validation fold is used to compute the error.
- This procedure is repeated k times; each time, a different group of observations is treated as the validation set.
- This process results in k estimates of the test error, which are averaged to compute th k-fold CV estimate, which provides us with an approximation of the error to expect on unseen data.

Cross validation

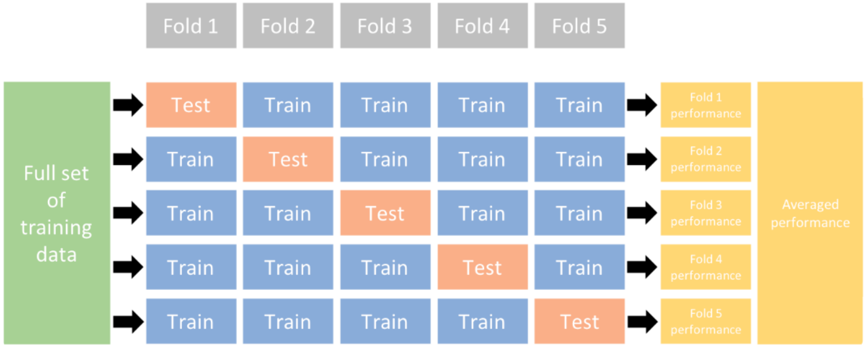


Figure 6: Cross validation