

Predictive Model kNN

Dr Liew How Hui

June 2022

Review

Week 1:

- CRISP-DM (Data Mining \Rightarrow Original Data (Noisy) \rightarrow Clean Data \leftrightarrow (Data Preprocessing \rightarrow) Pred. Model \rightarrow Validation \rightarrow Deployment in Prediction / Inference)

Relevant Practicals:

- p03_knn1.R
- p04_knn2.R

Outline

1 kNN Models

2 Weighted kNN Models

3 More Performance Evaluation

Theory Behind kNN

The *k-nearest neighbours* (kNN) algorithm is a non-parametric method used for classification and regression.

The assumption of kNN is “similar inputs have similar outputs”. Based on this assumption, a test input \mathbf{x} should be assigned the most common label amongst its k most similar training inputs.

Theory (cont)

Given a positive integer k and an input \mathbf{x} , the kNN algorithm first identifies the k points in the training data (\mathbf{x}_i, y_i) that are “similar” to \mathbf{x} , represented by $N(\mathbf{x})$.

- For kNN classifier, the prediction is

$$h(\mathbf{x}) = \text{mode}(\{y'' : (\mathbf{x}'', y'') \in N(\mathbf{x})\}),$$

$$\mathbb{P}(Y = j | \mathbf{X} = \mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N(\mathbf{x})} I(y_i = j).$$

- For kNN regressor, the prediction is

$$h(\mathbf{x}) = \frac{1}{k} \sum_{(\mathbf{x}'', y'') \in N(\mathbf{x})} y''.$$

Distance

The **measurement of similarity** in the kNN algorithm fundamentally relies on a “dissimilarity” metric or a “distance” function.

What is a “distance”?

A *distance* $d(\mathbf{x}_i, \mathbf{x}_j)$ is a function which satisfies the following conditions: For any $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$,

- (i) $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ and $d(\mathbf{x}_i, \mathbf{x}_j) = 0$ iff $\mathbf{x}_i = \mathbf{x}_j$;
- (ii) $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$; and
- (iii) $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$ (triangle inequality).

A *dissimilarity* is a function with only items 1 and 2 (not restricted by item 3).

Distance (cont)

The most common choice is the *Minkowski distance*:

$$d(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|_r = \left(\sum_{i=1}^p |x_i - z_i|^r \right)^{\frac{1}{r}}, \quad \mathbf{x}, \mathbf{z} \in \mathbb{R}^p. \quad (1)$$

Note that $\|\cdot\|^r$ is called the ℓ^r norm.

When $r = 1$, we have the *Manhattan distance*:

$$\|\mathbf{x} - \mathbf{z}\|_1 = |x_1 - z_1| + |x_2 - z_2| + \cdots + |x_p - z_p|.$$

When $r = 2$, we have the *Euclidean distance*:

$$\|\mathbf{x} - \mathbf{z}\|_2 = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + \cdots + (x_p - z_p)^2}.$$

Distance (cont)

When $r = \infty$, we have the *Chebyshev distance* (or maximum distance):

$$\|\mathbf{x} - \mathbf{z}\|_{\infty} = \max\{|x_1 - z_1|, |x_2 - z_2|, \dots, |x_p - z_p|\}.$$

The Euclidean distance is more sensitive to outliers than the Manhattan distance. When outliers are rare, the Euclidean distance performs very well and is generally preferred. When the outliers are significant, the Manhattan distance is more stable.

More Examples of 'Dissimilarity'

Apart from the Minkowski distance function (which only accepts numeric inputs), the following distance functions can be used in kNN:

A) Mahalanobis (`knnGarden::knnMCN`)

B) Gower (can be used to process mixed numeric and categorical data)

C) Jaccard score (for data that take discrete values, usually 0 and 1, e.g. black and white images) and its extension Tanimoto score (for data that can take on continuous values). They are used in

Chemoinformatics, plagiarism detection, thesaurus extraction, market-basket transactional data, anomalies detection in spatio-temporal data.

Theory: Bayes Optimal Classifier

If we **know** $\mathbb{P}(y|\mathbf{x})$ (which is almost never the case), then the “optimal” prediction is

$$y^* = h_{\text{opt}}(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \mathbb{P}(y|\mathbf{x})$$

For this type of classifier, the **error rate** is

$$\epsilon_{\text{BayesOpt}} = 1 - \mathbb{P}(y^*|\mathbf{x})$$

The Bayes optimal classifier is interesting because it provides a **theoretical lower bound of the error rate**.

Theory (cont)

The **upper bound on the error** is given by the constant classifier.

The kNN Classifier's 'ability to predict' is characterised by the following theory.

Theorem

As $n \rightarrow \infty$, the 1-NN error is no more than twice the error of the Bayes Optimal classifier:

$$\epsilon_{\text{BayesOpt}} \leq \epsilon_{\text{NN}} \leq 2\epsilon_{\text{BayesOpt}}$$

Similar guarantees hold for $k > 1$.

kNN Classifier

Basic R provides a kNN classifier in the class library. It only supports Euclidean distance and has the following form.

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE,  
     use.all = TRUE)
```

Here,

- train: matrix or data frame of training set cases.
- test: matrix or data frame of test set cases.
- cl: factor of true classifications of training set
- k: number of neighbours considered.

Implementation of kNN Classifier

A naive Python + Numpy implementation is listed below.

```
# https://github.com/joelgrus/data-science-from-scratch
from scipy.spatial.distance import euclidean
import logging
logging.basicConfig(level=logging.DEBUG)

def knn_classify(k: int, X, y, x_new, distance=euclidean) -> str:
    import numpy as np, collections
    dist_list = [distance(x, x_new) for x in X]
    logging.debug('distances={}'.format([round(d,4) for d in dist_list]))
    order = np.argsort(dist_list)
    k_nearest_labels = y[order[:k]]
    logging.debug('knn={}'.format(k_nearest_labels))
    vote_counts = collections.Counter(k_nearest_labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    return winner, winner_count/k

if __name__=='__main__':
    from sklearn import datasets
    iris_df = datasets.load_iris()
    X = iris_df['data']; y = iris_df['target']
    knn_classify(1,X,y,[6,3,5,2]); knn_classify(1,X,y,[5,3,1,0])
```

Example of kNN Classifier

Example 1:

A sport school would like to group their new enrolled students into 2 groups, as according to the existing students' weight and height. The weight and height of 7 existing students with group are shown in the table below.

Student	Weight (kg)	Height (cm)	Group
A	29	118	A
B	53	137	B
C	38	127	B
D	49	135	B
E	28	111	A
F	24	111	A
G	30	121	A

Example 1 (cont)

- (a) Perform and use $k = 3$ -NN method (with Euclidean distance) to predict which group the following students will be grouped into, based on **cut-off of 0.7** on group A.

Student	Weight (kg)	Height (cm)
H	35	120
I	47	131
J	22	115
K	38	119
L	31	136

Example 1 (cont)

- (b) The actual groups of the students are {A, B, A, B, B} for students {H, I, J, K, L} respectively. Construct a confusion matrix and calculate the accuracy measurements for a cut-off of 0.5 and a cut-off of 0.7.
- (c) Exercise: Write a Python script to produce the above calculations using the simple implementation in the earlier slide and also try out sklearn's implementation.
- (d) Exercise: Is R script easier to write?

kNN Regressor and Example

The *kNN regressor* applies “mean” instead of “mode” to “predict” the output.

R: `FNN::knn.reg` (kNN regressor with Euclidean distance)

Python: `sklearn.neighbors.KNeighborsRegressor`

Example 2: (Exam SRM Study Manual, p225, Q15.10)

A continuous variable Y is modelled as a function of X using kNN with $k = 3$. With the following data:

X	5	8	15	22	30
Y	4	1	10	16	30

Calculate the fitted value of Y at $X = 12$. Try write a script using R (or Python) to perform the calculation for you.

Classifier Boundary

A “classifier boundary” is the boundary between different classes. We can think of it as the “boundaries” between “countries”, for example, Malaysia has boundaries with Singapore, Thailand and Indonesia.

For binary classification problems, if the label Y is just positive (+) and negative (-), then the mathematical formulation of the classifier boundary of the predictive model (with known conditional probability estimate) is nonrigorously

$$\{\mathbf{x} \mid P(Y = + | X = \mathbf{x}) = P(Y = - | X = \mathbf{x})\}.$$

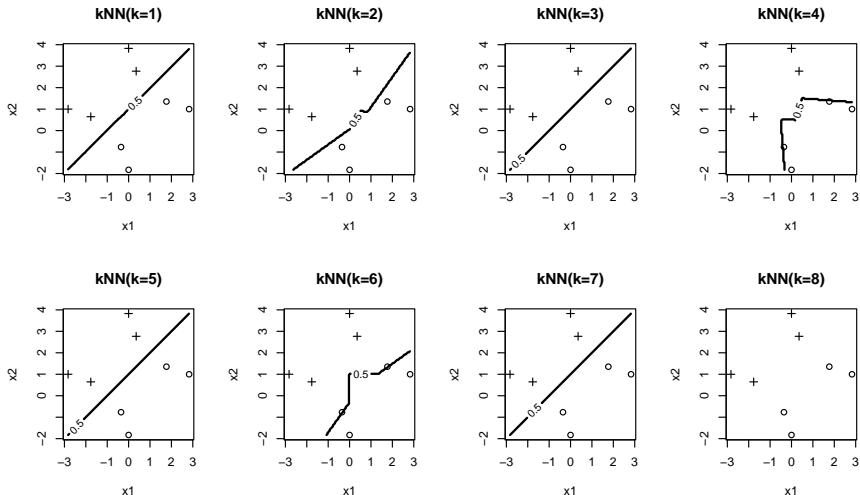
Classifier Boundary (cont)

Most data are not 2D, for example, the iris data. But most analysis will try to reduce to 2D to “visualise” the boundary. We can find a comprehensive analysis of the iris data using R with beautiful diagrams on <https://www.datacamp.com/community/tutorials/machine-learning-in-r>. As for the drawing the kNN boundary between classes, a “contour” plot is required and is illustrated by <https://stats.stackexchange.com/questions/21572/how-to-plot-decision-boundary-of-a-k-nearest-neighbor-c>

If we consider the boundary of the rotated ‘data1’ found in Practical 1 which is symmetrical w.r.t. $y = x$.

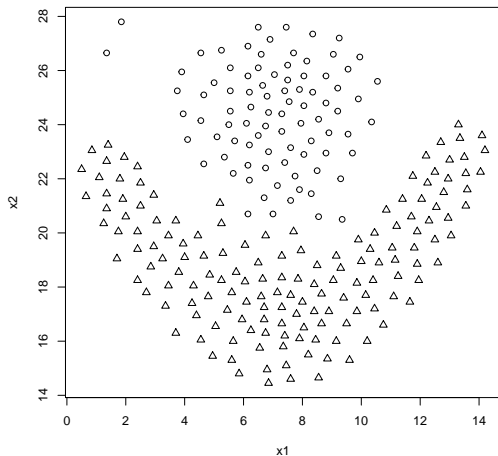
$P(Y = +|X = \mathbf{x}) = P(Y = -|X = \mathbf{x})$ should give $y = x$ but the algorithm may give a slightly different answer due to the algorithm as illustrated in the next slide.

Classifier Boundary (cont)

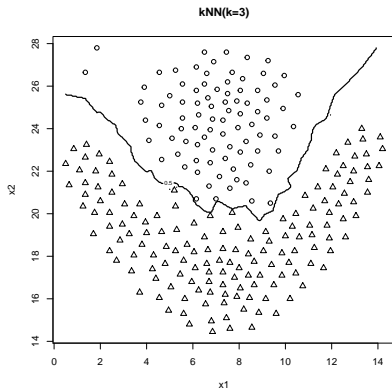
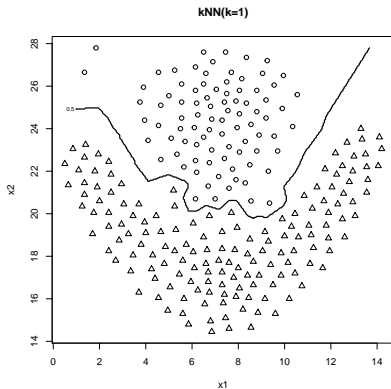


Classifier Boundary (cont)

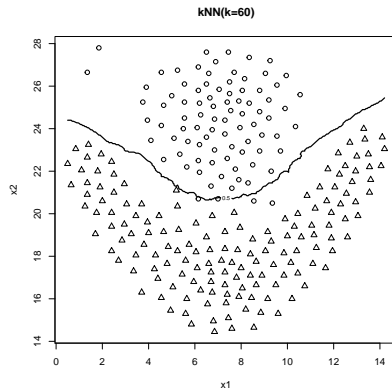
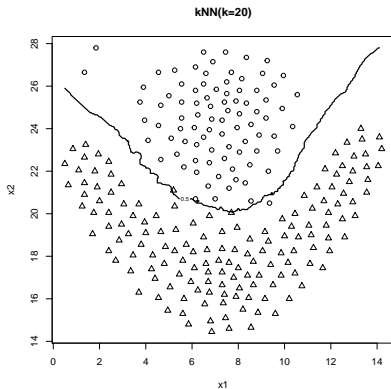
Now, we investigate a data with “nonlinear” classifier boundary.



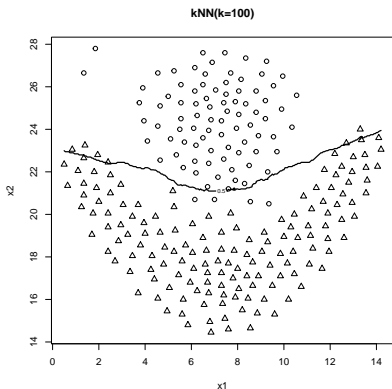
Classifier Boundary (cont)



Classifier Boundary (cont)



Classifier Boundary (cont)



Conclusion: The choice of k depends upon the data.

In general,

- k small, decision boundary is over “flexible” \Rightarrow kNN classifier is low bias, high variance.
- k large, decision boundary is less flexible, “less noise”, “smoother”, more “linear” \Rightarrow kNN classifier is low-variance, high bias.

Feature Scaling / Standardisation

Example: In the above example, if we change student weight to *g* and height to metre, what would happen?

Student	Weight (g)	Height (m)	Group
A	29,000	1.18	A
B	53,000	1.37	B
C	38,000	1.27	B
D	49,000	1.35	B
E	28,000	1.11	A
F	24,000	1.11	A
G	30,000	1.21	A

Some numbers are too large, while others are too small!

Feature Scaling (cont)

For most predictive models, kNN in particular, data with large difference in variances is **bad** for the “predictive model” training.

Feature scaling / Standardisation: Put all variables into a scaled range, the variables are equally weight.

Two popular feature scaling methods:

- Min-Max Normalisation (Rescaling):

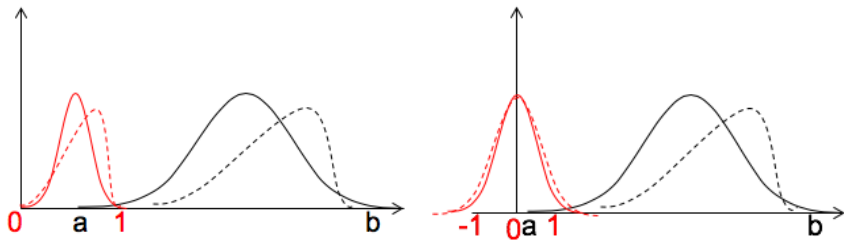
$$M(X_{ij}) = \frac{X_{ij} - X_{\min,j}}{X_{\max,j} - X_{\min,j}}$$

- Standard Scaler / Standardisation:

$$M(X_{ij}) = \frac{X_{ij} - \bar{X}_j}{s_{X,j}}$$

Feature Scaling (cont)

The following diagrams illustrate the effects of min-max scaling and standardisation:



Feature Scaling Example

Example: Given the training data and testing data in Example of kNN Classifier, apply the standardisation to perform and use $k = 3$ -NN method (with Euclidean distance) to predict which group the following students will be grouped into, based on **cut-off of 0.7** on group A.

WARNING

In Final Exam / Assessment, if the question did not mention “feature scaling” you **DON'T NEED TO SCALE** the features.

In real-world situation, you want perform feature scaling a lot of the times using computer.

Feature Scaling Example (cont)

Solution: Step 1: The original data X is standardised to \tilde{X} .

Student	Weight (kg)	Height	Norm_Wgt	Norm_Hgt	Group
A	29	118	-0.6113	-0.4587	A
B	53	137	1.5284	1.3355	B
C	38	127	0.1910	0.3912	B
D	49	135	1.1717	1.1467	B
E	28	111	-0.7005	-1.1197	A
F	24	111	-1.0571	-1.1197	A
G	30	121	-0.5222	-0.1754	A
Mean	35.8571	122.8571			
Std. dev	11.2165	10.5898			

Feature Scaling Example (cont)

Solution (cont): Step 2: Measure distances from new data to training data D .

$k = 3$, cut-off = 0.7:

		H	I	J	K	L
	Group	Distance	Distance	Distance	Distance	Distance
A	A	0.5673	2.0205	0.6854	0.8079	1.7091
B	B	2.2699	0.7792	3.4575	2.1628	1.9637
C	B	0.7131	0.8869	1.8218	0.7554	1.0544
D	B	1.8879	0.4177	3.0596	1.8013	1.6076
E	A	1.0544	2.5370	0.6548	1.1686	2.3759
F	A	1.2977	2.7878	0.4177	1.4590	2.4419
G	A	0.4557	1.7857	0.9109	0.7378	1.4193
$P(Y = A)$		0.6667	0.0000	1.0000	0.6667	0.3333
Cut-off = 0.5	\hat{y}	A	B	A	A	B
Cut-off = 0.7	\hat{y}	B	B	A	B	B

Outline

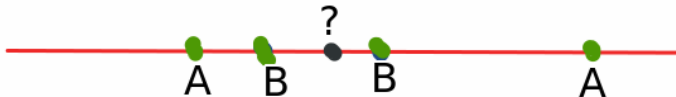
1 kNN Models

2 Weighted kNN Models

3 More Performance Evaluation


Weighted kNN

Some academicians propose an extension to kNN by considering the following scenario:



Here, a new point marked with '?' has 4 similar/closest points. Suppose the kNN with $k = 4$ is used. The conditional probability

$$P(Y = A|X = ?) = P(Y = B|X = ?) = 0.5$$

is obtained. If we use dictionary ordering, A is before B , so $kNN(k = 4)$ will give $Y = A$. This may not be reasonable since B is closer to '?'. 

Weighted kNN (cont)

Weighted kNN (wkNN, available in R in `kknn` library) introduces weights (called kernel) to solve this problem.

- 1 Let $N(\mathbf{x}, k + 1)$ be the $k + 1$ nearest neighbours to \mathbf{x} according to a distance function $d(\mathbf{x}, \mathbf{x}_i)$.
- 2 The $(k + 1)$ th neighbour is used for “standardisation” of the k smallest distances via $D_i = \frac{d(\mathbf{x}, \mathbf{x}_i)}{d(\mathbf{x}, \mathbf{x}_{k+1})}$.
- 3 A weighted majority of the k nearest neighbour

$$\hat{y} = \max_j \left\{ \sum_{i=1}^k K(D_i) I(y_i = j) \right\}.$$

Weighted kNN (cont)

In order for the 'weight' to make sense, we need it to satisfy some requirements. The 'weight' K is called a *kernel (function)* if it satisfies

- (1) $K(x) \geq 0$ for all $x \in \mathbb{R}$;
- (2) $K(x)$ is maximum when $x = 0$;
- (3) $K(x)$ descends monotonously when $x \rightarrow \pm\infty$.

Available kernels in R's `knn` or Python's `sklearn`:

- rectangular/uniform: $K(x) = \frac{1}{2}I(|x| \leq 1)$;
- inv/distance: $K(x) = 1/|x|$
- triangular: $K(x) = (1 - |x|) \cdot I(|x| \leq 1)$.
- optimal: The number of neighbours used for this kernel should be $\frac{2(d+4)}{d+2} k \in [1.2, 2]$, where k is used
- Others: `cos`, `gaussian`, `rank`, `epanechnikov` (or `beta(2,2)`), `biweight` (or `beta(3,3)`), `triweight` (or `beta(4,4)`).

Outline

1 kNN Models

2 Weighted kNN Models

3 **More Performance Evaluation**

Recall: Data Science and CRISP-DM

CRISP-DM (Cross Industry Standard Process for Data Mining)

- Business understanding
- Data understanding
- Data preparation / preprocessing
- Modelling
- Evaluation or Validation
- Deployment

The validation method we learned in Weeks 1 & 2 is the **holdout/validation set** method. For regression problems, the measures are MSE, R^2 ; For classification problem, the measures are accuracy; in particular, for binary classification problem, confusion matrix / contingency table, ROC.

More Performance Evaluation

There are a few things we need to take note regarding the holdout/validation set method:

- It works only for time-independent data. For a time-related data (e.g. the ISLR's Smarket data in the Practical 3 script `p03_knn1.R`), we have to split the data by earlier period and later period based on some cut-off time.
- For classification problem, linear sampling is allowed when the size of the data is large and the classes are 'uniformly' distributed, otherwise, stratified sampling should be used.

More Performance Evaluation (cont)

The holdout/validation set method (cont):

- The result depends on the 'selection' of data 'randomly'. Note that in Practical 3, there is a `set.seed(123)` which fixed the 'selection'. If we change the number 123 to 124 or 125, we will get different results for the performance. If we apply stratified sampling with odd index, we get an accuracy of 0.8011988 and with even index, we get an accuracy of 0.7654691

This point leads to the question: What is the 'real' performance of the (kNN) model?

More Performance Evaluation (cont)

For a data with 1000 rows, if we choose 700 rows for training and 300 rows for testing using linear sampling, there are

$$\frac{1000!}{700!300!}$$

possible combinations! The number is just **too large**.

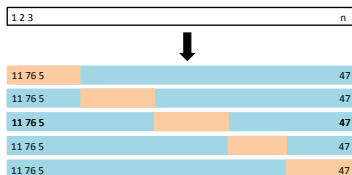
We can try multiple samples to have a better confidence on the performance for the holdout method or we can use the k-fold cross-validation method mentioned next.

Cross-Validation (cont)

k-fold cross validation (CV)

- Randomly divides the set of observations into k “equal” “folds”;
- First fold = validation set & remaining $k - 1$ folds = training set.
- Second fold = validation set & remaining $k - 1$ folds = training set.
- etc.

5-fold example:



Cross-Validation (cont)

Special case $k = n \Rightarrow$ **leave-one-out cross validation (LOOCV)**



Since training sample is close to n , the bias is small.
The variance is usually high because n training sets are similar to each other due to strong correlation.

Cross-Validation (cont)

Software support:

R: `caret::createDataPartition`, `createResamples` (for bootstrapping), `createFolds` (for k -fold cross validation), `createMultiFolds` (for repeated cross-validation), `createTimeSlices` (for dealing with time-series).

Python: `train_test_split` (simple random split), `KFold` or `StratifiedKFold` (k -fold CV), `LeaveOneOut` (LOOCV) from `sklearn.cross_validation`.

Cross-Validation (cont)

If we don't have the right to install additional package like the caret library, we can still use R's basic commands to perform k-fold cross validation. The $k = 10$ example is shown below.

<https://stats.stackexchange.com/questions/61090/how-to-split-a-data-set-to-do-10-fold-cross-validation>

```
#Randomly shuffle the data & create 10 fold
d.f = d.f[sample(nrow(d.f)),]
folds = cut(seq(1,nrow(d.f)),breaks=10,labels=FALSE)
#Perform 10 fold cross validation
for(i in 1:10){
  #Segment your data by fold using which()
  testIndexes = which(folds==i,arr.ind=TRUE)
  testData = d.f[testIndexes, ]
  trainData = d.f[-testIndexes, ]
  ...
}
```

Cross-Validation (cont)

Using caret library (it is a complex R packages with many dependencies):

```
set.seed(123)
train.control <- trainControl(method="cv", number=5)
model <- train(y ~., data=train.data, method="lm",
               trControl=train.control)
print(model)
```

When we split data into two groups — ‘training’ and ‘testing’, we will have **training errors** and **testing errors**. We are usually concerned about the testing errors but for inflexible models (like the logistic regression from next topic), we also need to look at the training errors.

Cross-Validation (cont)

Example: (Final Exam May 2019, Q3)

- (a) Supervised learning includes classification and regression.
 - (i) State the difference between classification and regression in term of response variable. (1 mark)
 - (ii) Explain the sampling methods used in splitting data for classification and regression respectively. (4 marks)
- (b)
 - (i) State an issue that comes along with split validation, which can be overcome by using cross validation. (1 mark)
 - (ii) Describe the process of a 5-fold cross validation. (4 marks)

Cross-Validation (cont)

Example (cont): (Final Exam May 2019, Q3)

- (c) A sample of 500 males and 800 females had been collected to test on a model of gender prediction. The model resulted that 380 males and 510 females were predicted correctly.
- (i) Assume male as positive class and female as negative class, calculate the count of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) for the model's result. (2 marks)
 - (ii) Construct the confusion matrix for the model. State the classification error, specificity and sensitivity of the model. (4 marks)
 - (iii) Compare the recall and precision for both male and female. Interpret your results. (4 marks)