Optimization

Regularization and Cross-Validation

Applied Machine Learning with R

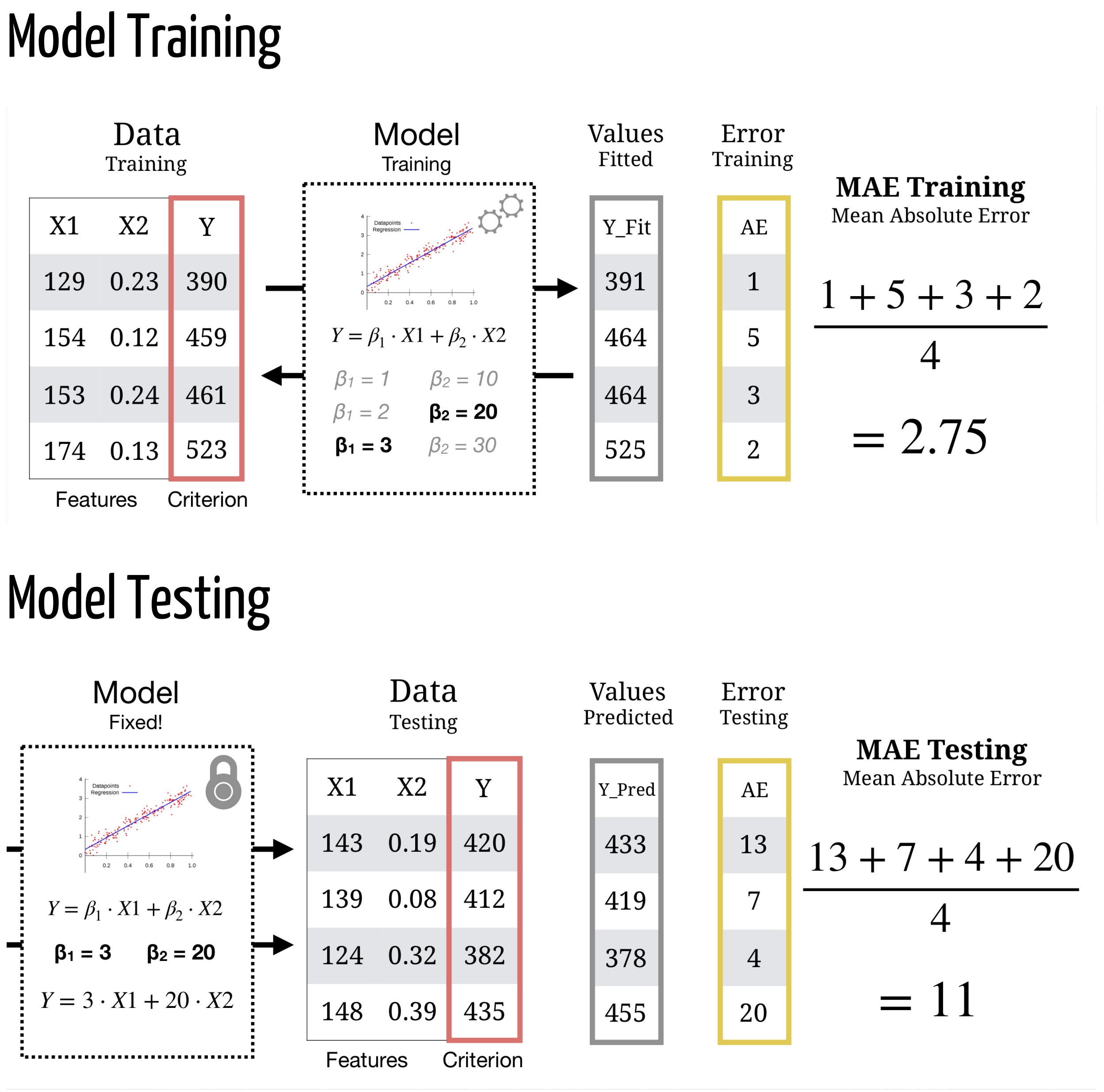
www.therbootcamp.com

@therbootcamp

January 2019

Where we are

- **Train** one of several models (**regression**, **decision trees**, and **random forests**) on training data.
- Explore models - show regression coefficients, plot decision trees (etc)
- Assess model **prediction** performance on **test** data
    - Mean Absolute Error (MAE)

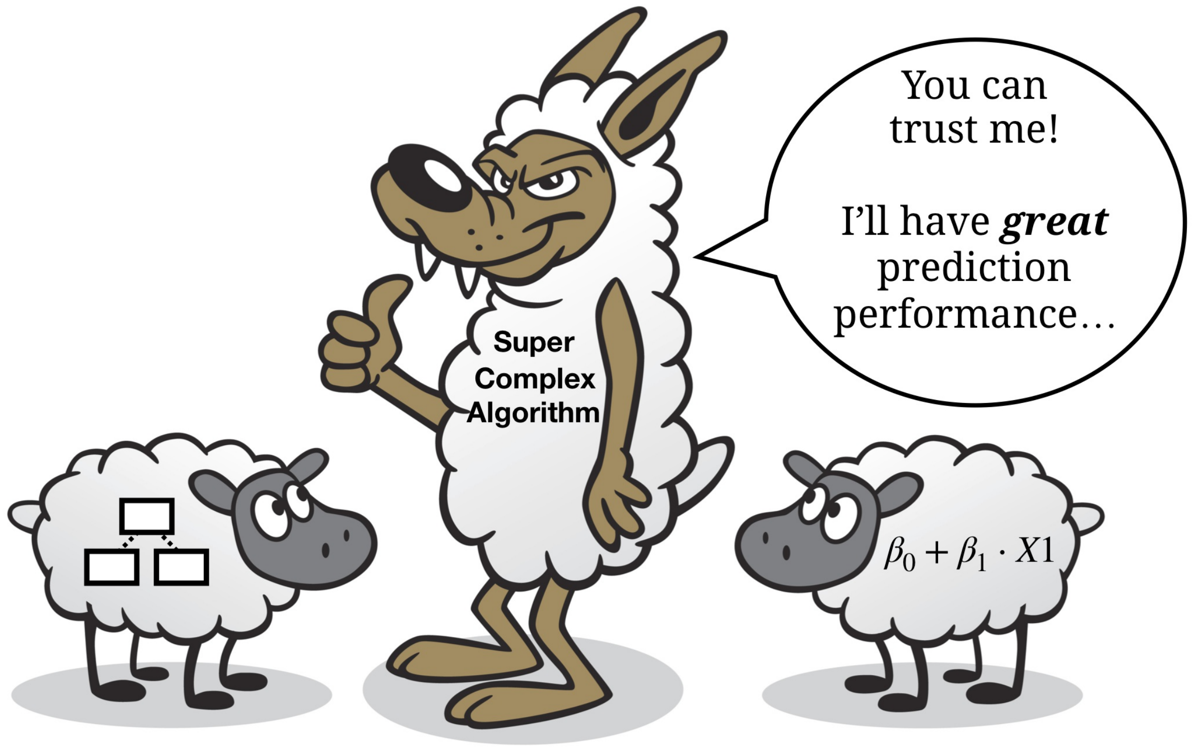


Overfitting

When a model is consistently **less accurate in predicting future data** than in **fitting training data, this is called overfitting**

Just because model A is better than model B in training, does not mean it will be better in testing!

Extremely flexible models that tend to overfit are like 'wolves in sheep's clothing'



victoriarollison.com (adapted)

Overfitting

How will we try to avoid overfitting?

Use regression models with **regularization** terms, such as **ridge** and **lasso** which explicitly **punish model complexity**.

Use **cross-validation** to find **optimal tuning parameters**, including regularization.

Regularized Regression

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

L1 Lasso Penalty

Cross Validation

Tuning parameters (Recap)

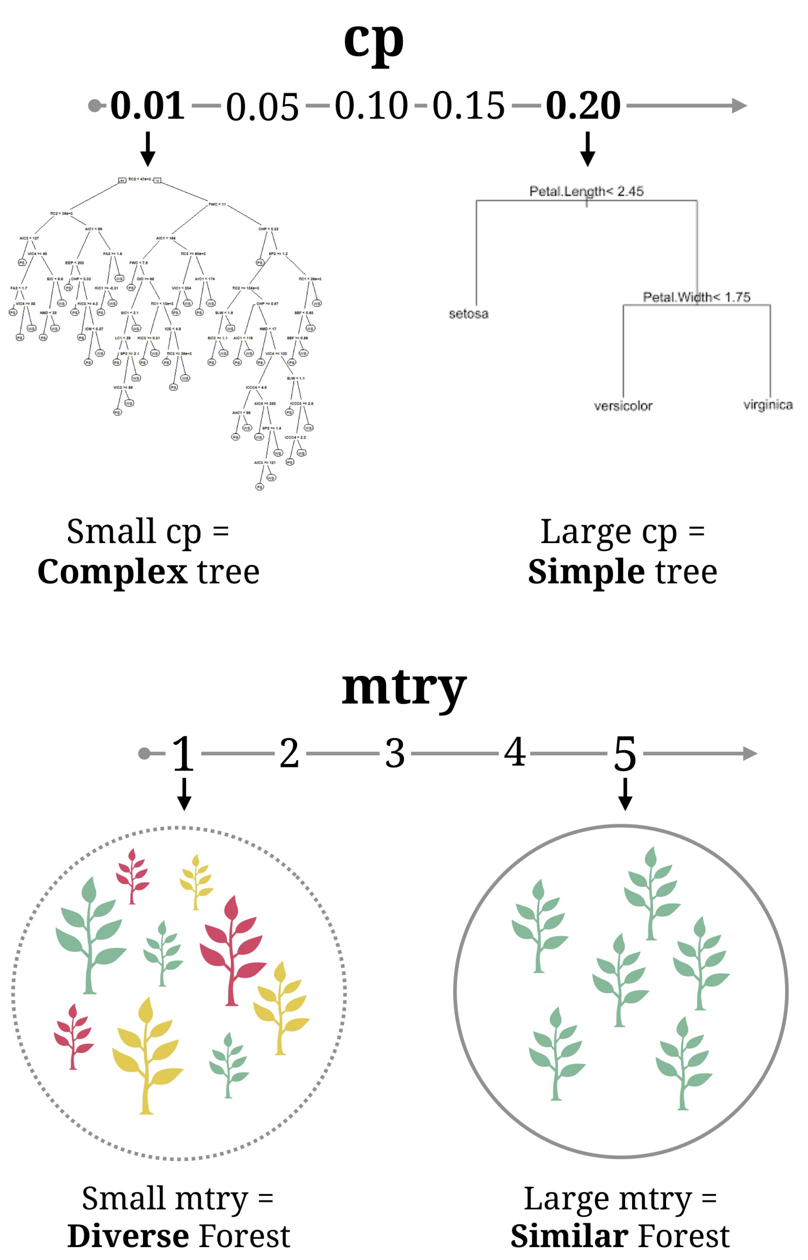
What are tuning parameters?

Tuning parameters are parameters that **guide** (aka. 'tune') a model during fitting. - Decision trees: complexity tuning parameter **cp**

- Random forests diversity tuning parameter **mtry**

Tuning parameters do not show up in the final model (you never see a complexity parameter in a final decision tree)! They are only used to guide fitting.

There is not one 'best' tuning parameter, it always depends on your specific dataset.



Regularized Regression

There are two common methods to fit penalized (aka regularized) regression models: Ridge and Lasso. Each penalizes regression models for having large \(\beta\) values using the **Lambda tuning parameter**

Ridge

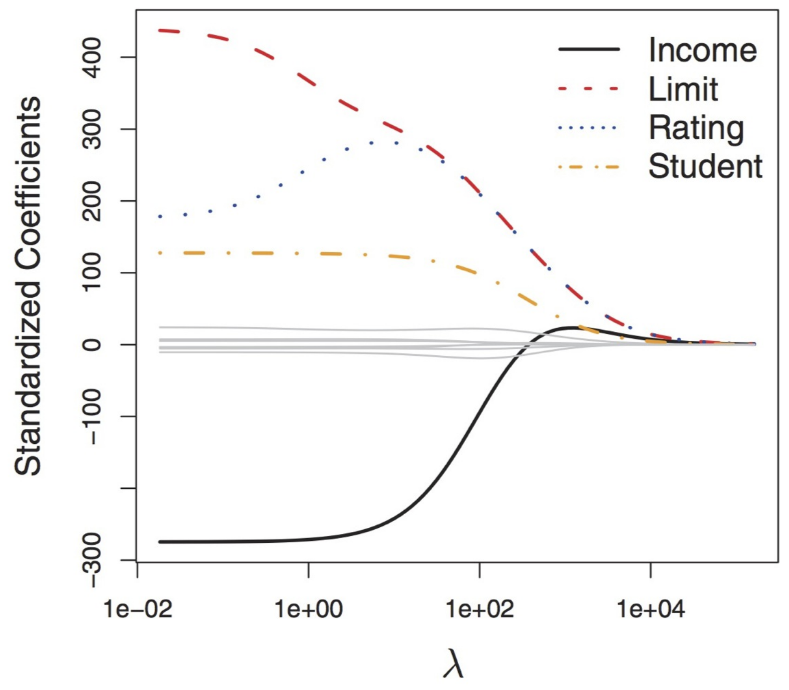
The Ridge penalty is known as the \(\ell2\) norm, where Beta weights are selected by **minimizing** the following equation:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \mathrm{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2,$$

As \(\lambda\) "Lambda" increases, coefficients are pushed towards (but not necessarily exactly to) 0.

See **Wikipedia's Ridge article** to learn more.

Ridge “L2” Norm



James et al., ISLR

Regularized Regression

There are two common methods to fit penalized (aka regularized) regression models: Ridge and Lasso. Each penalizes regression models for having large `\(\beta\)` values using the **Lambda tuning parameter**

Ridge

To fit Ridge penalized regression in R, use `method = "glmnet"`.

In the `tuneGrid` argument:

- `alpha = 0` indicates the `\(\ell2\)` Ridge penalty.
- `lambda` = Vector of lambda tuning parameters values to try.

```
# Train ridge penalised regression model in R

train(form = criterion ~ .,
      data = data_train,
      method = "glmnet",
      trControl = ctrl,
      preProcess = c("center", "scale"),  # Standardise
      tuneGrid = expand.grid(alpha = 0, # Ridge penalty
                             lambda = 1:100)) # Lambda
```

Regularized Regression

There are two common methods to fit penalized (aka regularized) regression models: Ridge and Lasso. Each penalizes regression models for having large \(\beta\) values using the **Lambda tuning parameter**

Lasso

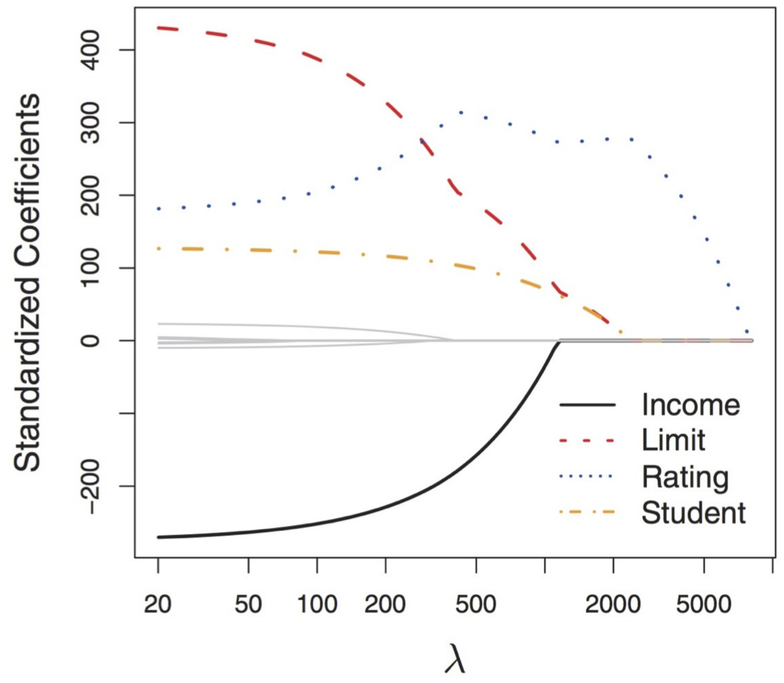
The Lasso penalty is known as the \(\ell1\) norm, where Beta weights are selected by minimizing the following equation:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \;= \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

As \(\lambda\) increases, coefficients are pushed towards 0, with some being forced to **exactly 0**.

See **Wikipedia's Lasso article** to learn more.

Lasso "L1" Norm



James et al., ISLR

Regularized Regression

There are two common methods to fit penalized (aka regularized) regression models: Ridge and Lasso. Each penalizes regression models for having large `\(\beta\)` values using the **Lambda tuning parameter**

Lasso

To fit Lasso penalized regression in R, use `method = "glmnet"`.

In the `tuneGrid` argument:

- `alpha = 1` indicates the `\(\ell1\)` Lasso penalty
- `lambda` = Vector of lambda tuning parameters values to try.

```
# Train Lasso penalised regression model in R

train(form = criterion ~ .,
      data = data_train,
      method = "glmnet",
      trControl = ctrl,
      preProcess = c("center", "scale"),  # Standardise
      tuneGrid = expand.grid(alpha = 1, # Lasso penalty
                             lambda = 1:100)) # Lambda
```

K-Fold Cross-Validation

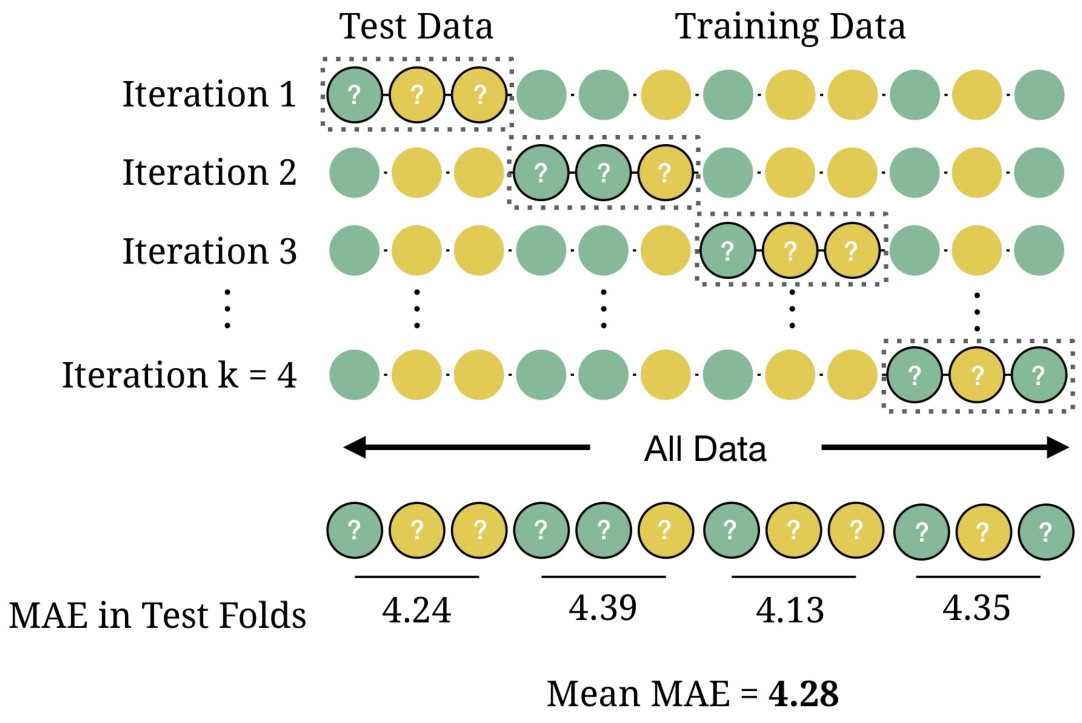
What is it?

Cross-validation is a sampling procedure performed on training data used to **estimate a model's prediction performance** in future test data, and to determine **optimal tuning parameters** selected to minimize prediction error.

Cross-validation is not "cheating": because it is only performed on the training data (never on the true test dataset)

After cross-validation is complete, the model is trained on the entire dataset, using optimal tuning parameters, resulting in a **final model** which can be used for future model testing.

Cross Validation



K-Fold Cross-Validation

Steps

1) Split the original training data into K 'folds' (mutually exclusive groups of cases)

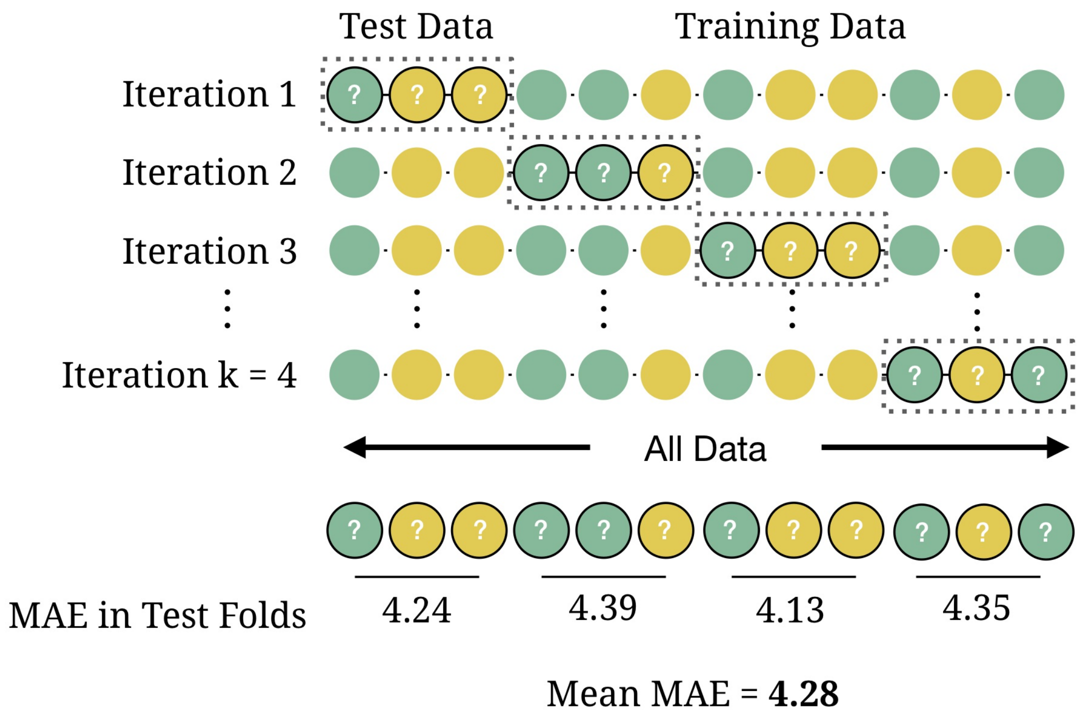
2) Select K - 1 folds for training, and 1 fold for testing.

3) Fit the model to the K - 1 training folds, and evaluate its testing accuracy on the test fold.

4) Repeat the process K times, so each fold is used once for testing.

5) Average the model's prediction error across all K folds

Cross Validation



K-Fold Cross-Validation

Determining optimal Tuning parameters

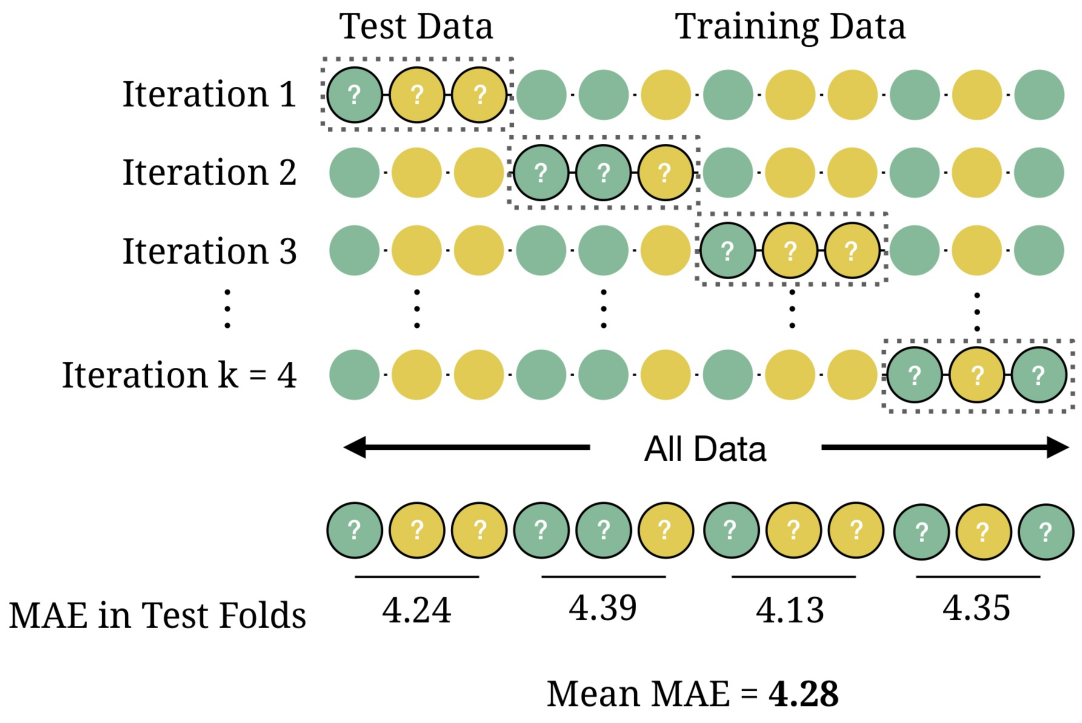
By trying different tuning parameters in each iteration, you can determine which value minimizes prediction error

Ex) Testing MAE values for values of cp

Fold	cp = .05	cp = .10	cp = .15	cp = .20
1	5.13	4.76	4.24	5.38
2	4.96	4.54	4.39	5.72
3	5.34	4.96	4.13	6.17
4	4.76	5.13	4.35	5.20
Mean	5.05	4.85	4.28	5.62

Conclusion: cp = .15 leads to the lowest test MAE

Cross Validation



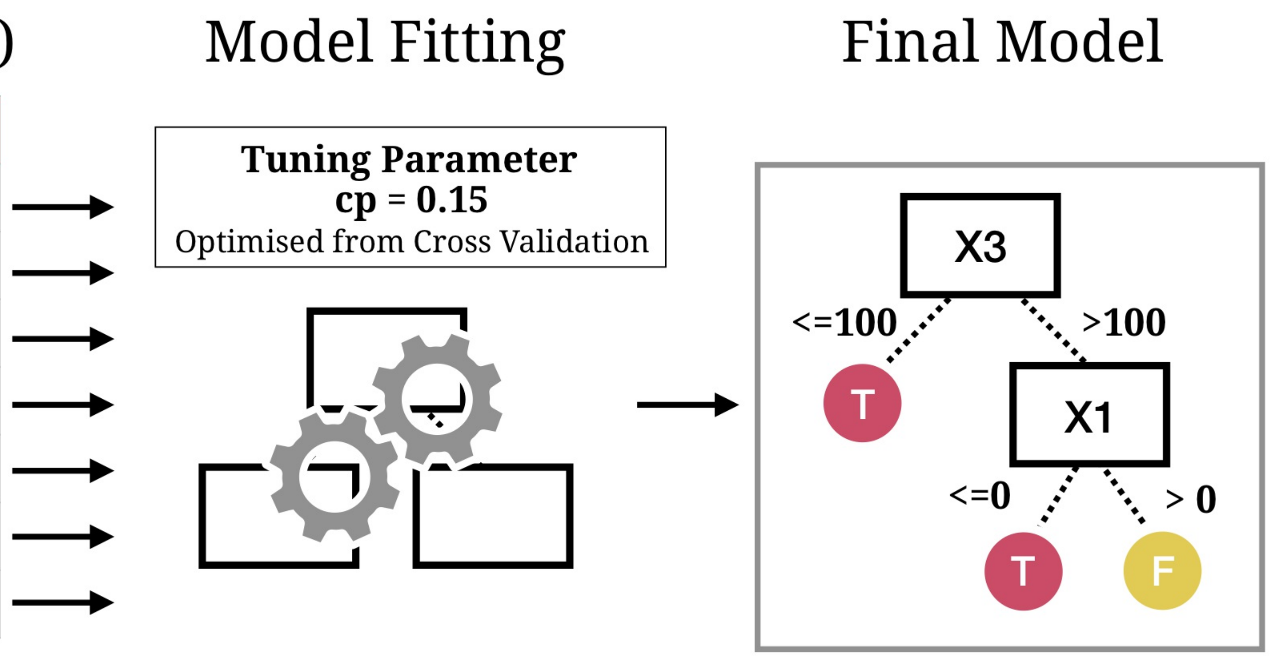
K-Fold Cross-Validation

Determining optimal Tuning parameters

Once the optimal value of a tuning parameter is determined through cross-validation, the algorithm is fit to the **entire training dataset** using the **optimal tuning parameter** resulting in the **Final Model**

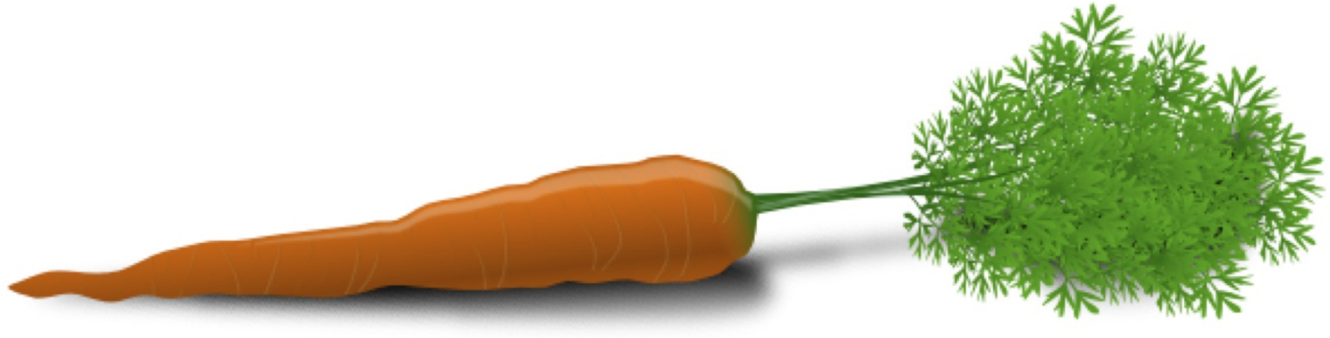
Training Data (100%)

	X1	X2	X3	Y
1	0.1	“A”	142	F
2	0.43	“B”	90	T
3	-0.39	“A”	83	F
5	-0.45	“B”	68	T
6	0.21	“B”	68	F
…	…	…	…	…
1001	-0.30	“A”	101	T



caret

Cross-validation, and tuning parameter optimization



K-Fold Cross validation

Specify the use of k-fold cross-validation using the `trainControl()` function

- `method`: The resampling method, use `"cv"` for cross validation
- `number`: The number of folds

When you pass this object to `train()` (for any model), caret will find best parameters using cross-validation.

```
# Specify 10 fold cross-validation
ctrl_cv <- trainControl(method = "cv",
                        number = 10)

# Predict baselers income using lasso regression

lasso_mod <- train(form = income ~ .,
                   data = baselers,
                   method = "glmnet",  # Penalised regression
                   trControl = ctrl_cv,
                   preProcess = c("center", "scale"),  # Standardis
                   tuneGrid = expand.grid(alpha = 1, # Lasso
                                          lambda = 1:100))
```

K-Fold Cross validation

If you print model object `XX_mod` you will see summary statistics showing how the model performed on averge in test folds for different values of the tuning parameters

At the bottom of the output, you'll see a summary message telling you how the best tuning parameter was found.

`Tuning parameter 'alpha' was held constant at a value of 1.`

`RMSE was used to select the optimal model using the smallest value.`

`The final values used for the model were alpha = 1 and lambda = 28.`

```
# Print summary information
lasso_mod
```

```
glmnet

1000 samples
  19 predictor

Pre-processing: centered (24), scaled (24)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 900, 901, 900, 901, 900, 899, ...
Resampling results across tuning parameters:

  lambda  RMSE  Rsquared  MAE
    1     1029  0.8631    811.1
    2     1029  0.8631    811.1
    3     1029  0.8631    811.0
    4     1028  0.8633    810.5
    5     1028  0.8634    810.0
    6     1027  0.8636    809.6
    7     1026  0.8637    809.2
    8     1026  0.8638    808.8
```

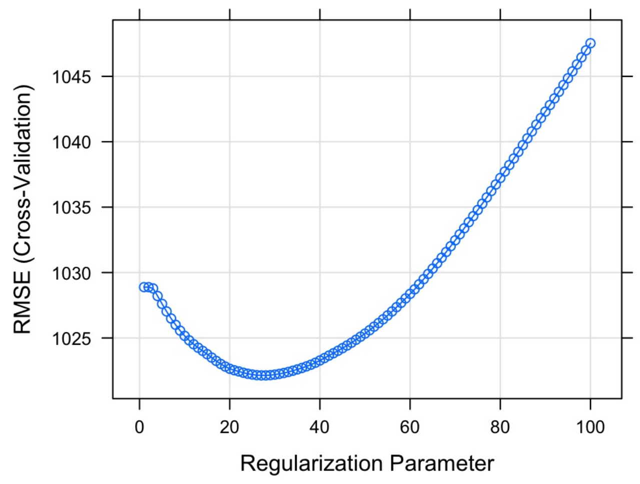
K-Fold Cross validation

If you plot your model object with `plot(XX_mod)` you will see a plot showing the relationship between the tuning parameter and error.

The bottom of the curve shows the best tuning parameter (minimises error)

Print the best tuning parameter value with `XX_mod$bestTune$NAME`

```
# Visualise tuning parameter error curve
plot(lasso_mod)
```




```
# Print best tuning parameter values
lasso_mod$bestTune$lambda
```

```
[1] 28
```

Regularized Regression

Your **final model** is (as always) stored in `XX_mod$finalModel`. This is the model fit to the entire data using the **optimal tuning parameter(s)**.

To get the coefficients from a Ridge or Lasso regression model, use the `coef()` function, with `XX_mod$finalModel` as the first argument, and the best tuning value as the second argument

In the output, values with . are coefficients that have been **removed with the lasso**!

```
# Print final model coefficients using best lambda
coef(lasso_mod$finalModel,      # Final Lasso model
     lasso_mod$bestTune$lambda) # Best lambda value
```

```
25 x 1 sparse Matrix of class "dgCMatrix"
                                      1
(Intercept)                   7569.6832
id                                .
sexmale                           .
age                           1925.9098
height                          21.6221
weight                            .
educationobligatory_school        .
educationSEK_II                   .
educationSEK_III                  .
confessionconfessionless         0.2419
confessionevangelical-reformed    .
confessionmuslim                  .
confessionother                   .
children                       -18.0667
happiness                     -132.5714
```

Comparing models after cross-validation

If you have fit many models with cross-validation, you can compare their estimated prediction performance with `resamples()`

The main argument to `resamples()` should be a list of all of your models created with `train()`

If you print the `summary()` of this object, it will print 'prediction' error statistics from cross-validation during training. This is your estimate of future prediction performance!

```
# Get accuracy statistics across folds for three
#  models that I have fit with cross validation

resamples_mod <- resamples(list(ridge = ridge_mod,
                                lasso = lasso_mod,
                                glm = glm_mod))

# Print summary of accuracies
summary(resamples_mod)

# MAE
#         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
# ridge 1057.9  1078.9 1083.2 1090.1  1112.0 1118.5    0
# lasso  865.7   892.7  897.2  902.9   914.2  944.8    0
# glm    839.7   910.0  920.5  905.9   922.7  936.7    0
```

Questions?

Practical