David Eccles School of Business
THE UNIVERSITY OF UTAH

KNN Tutorial

Data Mining for Business Intelligence

Matt Pecsok

OUTLINE

- KNN in R with IBk
- IBk package defaults and parameters.
- Exploring Train vs Test set results
- Cross-Validation Performance of various argument settings.
- Grid-Search and IBk best models
- Ablation Analysis

IBk – Package Introduction

See IBk documentation and parameter settings at
http://weka.sourceforge.net/doc.dev/weka/classifiers/lazy/IBk.html
Parameters:

-I
Weight neighbours by the inverse of their distance (use when k > 1)

-F
Weight neighbours by 1 - their distance (use when k > 1)

-K <number of neighbors>
Number of nearest neighbours (k) used in classification. (Default = 1)

-X
Select the number of nearest neighbours between 1 and the k value specified using hold-one-out evaluation on the training data (use when k > 1)

-E
Minimize mean squared error rather than mean absolute error when using -X option with numeric prediction.

Data Import and Categorical Factorization

- Here we import the needed libraries
- Import the required dataset
- Create variables for use throughout the tutorial such as the metrics list and the seed value.
- Explore the structure and summarize the dataset.

```r
library(caret)
library(RWeka)
library(rminer)
library(matrixStats)
library(knitr)
library(tictoc)
library(tidyverse)
tic()

# import data
cloud_wd <- getwd()
setwd(cloud_wd)
insurance <- read.csv(file = "insurance.csv", stringsAsFactors = TRUE)

###  Set up cv parameters

df <- insurance
target <- 7
seedVal <- 500
metrics_list <- c("MAE","RMSE","MAPE","RMSPE")
```

```{r}
str(insurance)
```

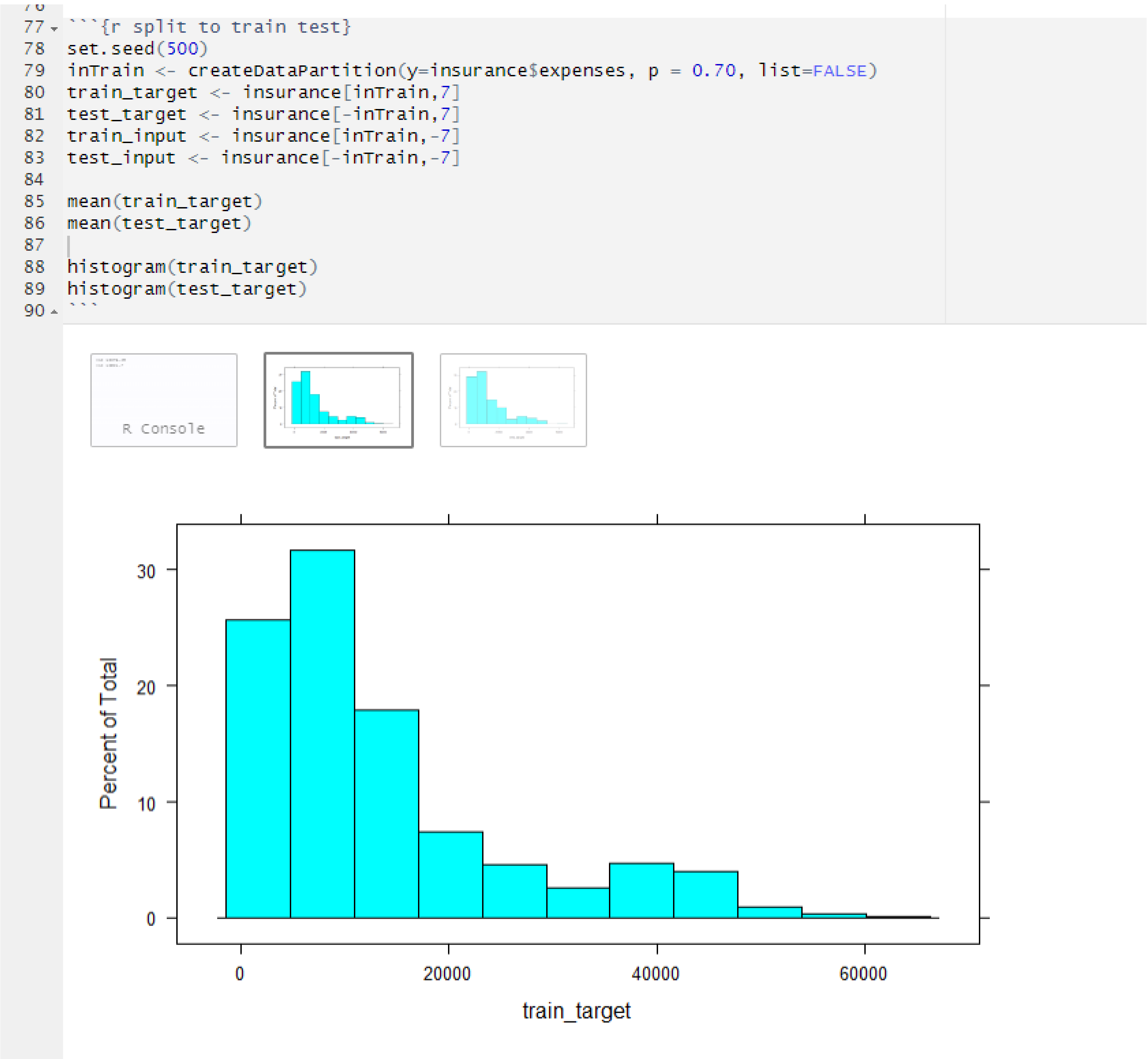
```
'data.frame':   1338 obs. of  7 variables:
 $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
 $ bmi     : num  27.9 33.8 33 22.7 28.9 25.7 33.4 27.7 29.8 25.8 ...
 $ children: int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
 $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...
 $ expenses: num  16885 1726 4449 21984 3867 ...
```

```{r}
summary(insurance)
```

```
      age            sex           bmi           children       smoker          region       expenses    
 Min.   :18.00   female:662   Min.   :16.00   Min.   :0.000   no :1064   northeast:324   Min.   : 1122  
 1st Qu.:27.00   male  :676   1st Qu.:26.30   1st Qu.:0.000   yes: 274   northwest:325   1st Qu.: 4740  
 Median :39.00                Median :30.40   Median :1.000              southeast:364   Median : 9382  
 Mean   :39.21                Mean   :30.67   Mean   :1.095              southwest:325   Mean   :13270  
 3rd Qu.:51.00                3rd Qu.:34.70   3rd Qu.:2.000                              3rd Qu.:16640  
 Max.   :64.00                Max.   :53.10   Max.   :5.000                              Max.   :63770  
```

Data Partition – Simple Holdout

- Split the data into test and train sets, extract the target variable into a new variable for both sets.
- Show the distribution of the target in both train and test to prove both sets are similar when comparing via the target.



K = 1 on Train and Test

- Compare model performance from Train to Test using a K = 1 (1 nearest neighbor setting).
- Train set errors are 0 because each instance can be predicted using itself resulting in no errors at all.
- Test set errors are much higher. The nearest neighbor in Test is not the exact same observation.

```
When using K = 1 on the training set we will then achieve an MAE = 0 (or if classification 100% accuracy)

## MAE for K = 1

```{r Simple IBk examples}

# K = 1

knn_model_k_1 <- IBk(train_target ~ .,data = train_input,control = Weka_control(K=1))
knn_model_k_1
```

 IB1 instance-based classifier
 using 1 nearest neighbour(s) for classification


```{r k1 train}
insample_pred <- predict(knn_model_k_1, train_input)
mmetric(train_target, insample_pred, metrics_list)
```

   MAE RMSE MAPE RMSPE
     0   0   0   0


```{r k1 test}
test_pred <- predict(knn_model_k_1, test_input)
mmetric(test_target, test_pred, metrics_list)
```

         MAE   RMSE   MAPE   RMSPE
 3803.667750 7217.775045   37.651870   9.425661
```

K = 5 on Train and Test

- Higher errors on both Train (expected) and Test.
- It appears K = 5 using unweighted voting is not as good as K = 1.

```
## MAE for K = 5

```{r}
knn_model_k_5 <- IBk(train_target ~ .,data = train_input,control = Weka_control(K=5))
knn_model_k_5
```
```

```
IB1 instance-based classifier
using 5 nearest neighbour(s) for classification
```

```
```{r k5 train}
insample_pred <- predict(knn_model_k_5, train_input)
mmetric(train_target, insample_pred, metrics_list)
```
```

```
        MAE        RMSE        MAPE       RMSPE
2827.042013 4698.609321   26.490765    5.015548
```

```
```{r k5 test}
test_pred <- predict(knn_model_k_5, test_input)
mmetric(test_target, test_pred, metrics_list)
```
```

```
       MAE      RMSE      MAPE     RMSPE
3994.90551 6388.61020  41.18298   7.17385
```

K = 1 and I = TRUE on Train and Test

- The errors shouldn't change from using only k=1 since we are using only 1 neighbor. Weighting is useful when k > 1.

```
## weighted distance

## MAE for K = 1 and I = TRUE

```{r Simple IBk examples I=TRUE}

# K = 1

knn_model_k1_itrue <- IBk(train_target ~ .,data = train_input,control = Weka_control(K=1,I=TRUE))
knn_model_k1_itrue
```
```

```
IB1 instance-based classifier
using 1 inverse-distance-weighted nearest neighbour(s) for classification
```

```
```{r k1 train I=TRUE}
insample_pred <- predict(knn_model_k1_itrue, train_input)
mmetric(train_target, insample_pred, metrics_list)
```
```

```
  MAE  RMSE  MAPE RMSPE
    0     0     0     0
```

```
```{r k1 test I=TRUE}
test_pred <- predict(knn_model_k1_itrue, test_input)
mmetric(test_target, test_pred, metrics_list)
```
```

```
        MAE        RMSE        MAPE       RMSPE
3803.667750 7217.775045   37.651870    9.425661
```

K = 5 and I = TRUE on Train and Test

- Weighted voting has improved Train performance due to more weight on the exact same instance being the most heavily weighted neighbor.
- In test we've also improved our performance as compared to K = 5 without weighted voting (I=FALSE)

```
## MAE for K = 5 and I = TRUE

```{r}
knn_model_k5_itrue <- IBk(train_target ~ .,data = train_input,control = Weka_control(K=5,I=TRUE))
knn_model_k5_itrue
```

IB1 instance-based classifier
using 5 inverse-distance-weighted nearest neighbour(s) for classification


```{r k5 train I=TRUE}
insample_pred <- predict(knn_model_k5_itrue, train_input)
mmetric(train_target, insample_pred, metrics_list)
```

        MAE   RMSE   MAPE   RMSPE
239.6097554 475.9304220   2.6135382   0.7020244


```{r k5 test I=TRUE}
test_pred <- predict(knn_model_k5_itrue, test_input)
mmetric(test_target, test_pred, metrics_list)
```

        MAE   RMSE   MAPE   RMSPE
3821.653457 6260.476947   39.918157   7.330998
```

K = from 1 to 40 and I = TRUE and X=TRUE on Train and Test

- We now ask the model to eval k values from 1 to 40 using leave one out cross validation. The best K will be selected.
- The model has chosen a K = 3 based on its ability to reduce the MAE.

````
## MAE for K = 40, I = TRUE, X = TRUE.

Try k from 1 to 40, using inverse of distance weights.

Demonstrating how to use the Leave one out cross-validation built into the IBk package to determine the best k.

```{r X TRUE}
knn_model_itrue_xtrue <- IBk(train_target ~ .,data = train_input,control = Weka_control(K=40,I=TRUE,X=TRUE))
knn_model_itrue_xtrue
```

 IB1 instance-based classifier
 using 3 inverse-distance-weighted nearest neighbour(s) for classification

The model has chosen k = 3 to minimize the MAE.

```{r}
insample_pred <- predict(knn_model_itrue_xtrue, train_input)
mmetric(train_target, insample_pred, metrics_list)
```

        MAE        RMSE        MAPE       RMSPE
159.9757828 372.7041571   1.8370163   0.6670268

```{r}
test_pred <- predict(knn_model_itrue_xtrue, test_input)
mmetric(test_target, test_pred, metrics_list)
```

        MAE        RMSE        MAPE       RMSPE
3813.619044 6464.275066   40.250375    8.281032
````

IBk custom cross-validation function

- Create a custom cross validation function which allows us to explore k and I values.
- In addition, we change the end of this function to include columns for the k and i parameters as well as return a dataframe instead of a kable.

````
```{r Define cv_IBk}

cv_IBk <- function(df, target, nFolds, seedVal, metrics_list, k, i)
{
# create folds using the assigned values

set.seed(seedVal)
folds = createFolds(df[,target],nFolds)

# The lapply loop

cv_results <- lapply(folds, function(x)
{
# data preparation:

  test_target <- df[x,target]
  test_input <- df[x,-target]

  train_target <- df[-x,target]
  train_input <- df[-x,-target]
  pred_model <- IBk(train_target ~ .,data = train_input,control = Weka_control(K=k,I=i))
  pred <- predict(pred_model, test_input)
  train_pred <- predict(pred_model, train_input)

  return(mmetric(test_target,pred,metrics_list))
})

cv_results_m <- as.matrix(as.data.frame(cv_results))
cv_mean<- as.matrix(rowMeans(cv_results_m))
cv_sd <- as.matrix(rowSds(cv_results_m))
colnames(cv_mean) <- "Mean"
colnames(cv_sd) <- "Sd"
cv_df <- data.frame(t(cbind(cv_mean,cv_sd))) %>% round(2)
cv_df$param_K <- k
cv_df$param_I <- as.logical(i)
cv_df <- cv_df %>% rownames_to_column(var = "measure")
cv_df
}
```
````

IBk custom cross-validation function

- We run the new cv_function with various values of k and I = FALSE

```
```{r call cv_IBk_train and cv_IBk}
cv_IBk(df, target, 3, seedval, metrics_list, 1, FALSE)
```
```

Description: df [2 × 7]

measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <dbl>	param_I <lgl>
Mean	3728.35	7169.88	39.04	11.91	1	FALSE
Sd	218.95	302.90	3.17	1.54	1	FALSE

2 rows

```
## call cross-validation with k=2 and I=FALSE
```{r}
cv_IBk(df, target, 3, seedval, metrics_list, 2, FALSE)
```
```

Description: df [2 × 7]

measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <dbl>	param_I <lgl>
Mean	3497.67	6212.26	38.02	9.81	2	FALSE
Sd	68.79	88.36	3.54	1.55	2	FALSE

2 rows

```
## call cross-validation with k=3 and I=FALSE
```{r}
cv_IBk(df, target, 3, seedval, metrics_list, 3, FALSE)
```
```

Description: df [2 × 7]

measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <dbl>	param_I <lgl>
Mean	3575.87	5986.29	38.02	8.01	3	FALSE
Sd	115.05	202.05	3.05	0.96	3	FALSE

2 rows

Parameter Grid Setups

- Build a grid of k values from 1 to 40 and l options (TRUE,FALSE)
- Notice our dataframe is 80 rows long.

```
# Part 3.

Use a custom grid search to find our best hyperparameter combination.

## Build the grid

```{r}
# Create multiple vectors
param_k <- c(seq(1,40))
param_i <- c(FALSE, TRUE)

# Generate a grid of all combinations
grid <- expand.grid(param_k, param_i, stringsAsFactors = FALSE)

colnames(grid) <- c("k","i")
# Print the grid

grid
```
```

Description: df [80 x 2]

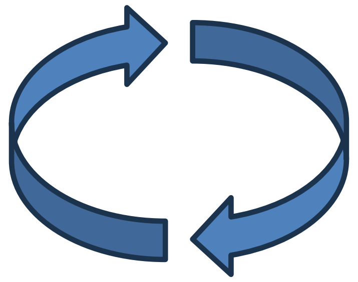
k <int>	i <lgl>
1	FALSE
2	FALSE
3	FALSE
4	FALSE
5	FALSE
6	FALSE
7	FALSE
8	FALSE
9	FALSE
10	FALSE

1-10 of 80 rows

Parameter Grid Looping function

- Define a new named function called “run_grid_cv” which loops over each row in the dataframe and calls the cross validation function for the rows setups for I and K.

- This function will loop 80 times based on the previous grid setups.



```
## define a named function to loop through the grid

This function will take the grid as input and call the cross validation function for each row in the grid. Finally, each cross vali
deviation will be returned as a dataframe.

```{r}

run_grid_cv <- function(grid) {
  results <- data.frame()
  for (i in 1:nrow(grid)) {
    row <- grid[i,]  # Get the i-th row
    cv_result <- cv_IBk(df = df,target = target,nFolds = 5,seedval =  seedval,metrics_list =  metrics_list,k = row$k,i = row$i)
    results <- rbind(results, cv_result)
  }
  return(results)
}
```
```

Calling the grid search function

- We simply pass the grid as a parameter into the new named function.
- The function takes the grid and loops over each row calling the cross validation function each time.
- We store the resulting dataframe in a variable named cv_grid_results.

```
## run the grid search, return a dataframe with the results.

```{r}
cv_grid_results <- run_grid_cv(grid)
```

```

# Filter the result dataframe. Find the best 10 MAE.

- We now demonstrate how to filter the dataframe to find only the Mean results, and arrange the dataframe in ascending order to show the lowest MAE. We use the head function to show only the best 10 MAE results. k = 2 is the winner.

```
```{r}
# pull the top 3 based on MAE

cv_grid_results %>%
  filter(measure == 'Mean') %>%
  arrange(MAE) %>%
  head(10)
```
```

Description: df [10 × 7]

	measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <int>	param_I <lgl>
1	Mean	3261.69	6012.29	32.22	8.18	2	TRUE
2	Mean	3325.64	5850.02	33.11	7.40	3	TRUE
3	Mean	3357.56	6021.81	33.29	8.00	2	FALSE
4	Mean	3366.06	5684.82	33.17	6.46	5	TRUE
5	Mean	3371.33	5772.18	33.48	6.92	4	TRUE
6	Mean	3388.12	5681.46	33.24	6.21	6	TRUE
7	Mean	3416.70	5697.26	33.10	6.00	7	TRUE
8	Mean	3456.47	5719.96	33.15	5.85	8	TRUE
9	Mean	3474.73	5743.77	33.10	5.72	9	TRUE
10	Mean	3483.40	5925.34	34.63	7.18	3	FALSE

1-10 of 10 rows

Filter the result dataframe. Find the best 10 MAPE.

- We now demonstrate how to filter the dataframe to find only the Mean results, and arrange the dataframe in ascending order to show the lowest MAPE. We use the head function to show only the best 10 MAPE results. k = 2 is the winner.
- It's interesting to note that some of the top models are not the same based on MAE vs MAPE.

```r
# pull the top 10 based on MAPE

cv_grid_results %>%
  filter(measure == 'Mean') %>%
  arrange(MAPE) %>%
  head(10)
```

Description: df [10 × 7]

	measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <int>	param_l <lgl>
1	Mean	3261.69	6012.29	32.22	8.18	2	TRUE
2	Mean	3484.70	5757.88	32.80	5.57	10	TRUE
3	Mean	3498.20	5764.89	32.82	5.49	11	TRUE
4	Mean	3516.06	5795.59	33.04	5.45	12	TRUE
5	Mean	3416.70	5697.26	33.10	6.00	7	TRUE
6	Mean	3474.73	5743.77	33.10	5.72	9	TRUE
7	Mean	3325.64	5850.02	33.11	7.40	3	TRUE
8	Mean	3456.47	5719.96	33.15	5.85	8	TRUE
9	Mean	3366.06	5684.82	33.17	6.46	5	TRUE
10	Mean	3388.12	5681.46	33.24	6.21	6	TRUE

1-10 of 10 rows

Ablation Analysis

- Do we see performance drop or improve when we remove a single column from the predictors?
- Here we remove each column one at a time and use cross validation to examine the results.
- We run a baseline first, then remove age, then sex, then bmi, then children then smoker and finally region.
- Based on MAE removing Region appears to lower the MAE, however it increases the MAPE.

````
# Part 4


## Ablation Analysis (test feature removal)

Here we systematically remove each feature from the model to see the impact.
This allows us to get a sense of feature importance.
Some features cause the errors to rise substantially. Removing region actually improves the model.

```{r}
cv_IBk(df, 7, 5, seedval, metrics_list, 2, TRUE) # baseline
cv_IBk(df[,-c(1)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(2)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(3)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(4)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(5)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(6)], 6, 5, seedval, metrics_list, 2, TRUE) # removing region appears to improve the model. slightly.
```
````

data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7

Description: df [2 × 7]

measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <dbl>	param_I <lgl>
Mean	3261.69	6012.29	32.22	8.18	2	TRUE
Sd	290.10	478.84	4.63	1.55	2	TRUE

2 rows

````
```{r}
cv_IBk(df, 7, 5, seedval, metrics_list, 2, TRUE) # baseline
cv_IBk(df[,-c(1)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(2)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(3)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(4)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(5)], 6, 5, seedval, metrics_list, 2, TRUE)
cv_IBk(df[,-c(6)], 6, 5, seedval, metrics_list, 2, TRUE) # removing region appears to improve the model. slightly.
```
````

data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7 | data.frame 2 x 7

Description: df [2 × 7]

measure <chr>	MAE <dbl>	RMSE <dbl>	MAPE <dbl>	RMSPE <dbl>	param_K <dbl>	param_I <lgl>
Mean	3192.73	6050.70	35.21	8.90	2	TRUE
Sd	241.30	377.82	5.22	1.34	2	TRUE

2 rows

David Eccles School of Business
THE UNIVERSITY OF UTAH