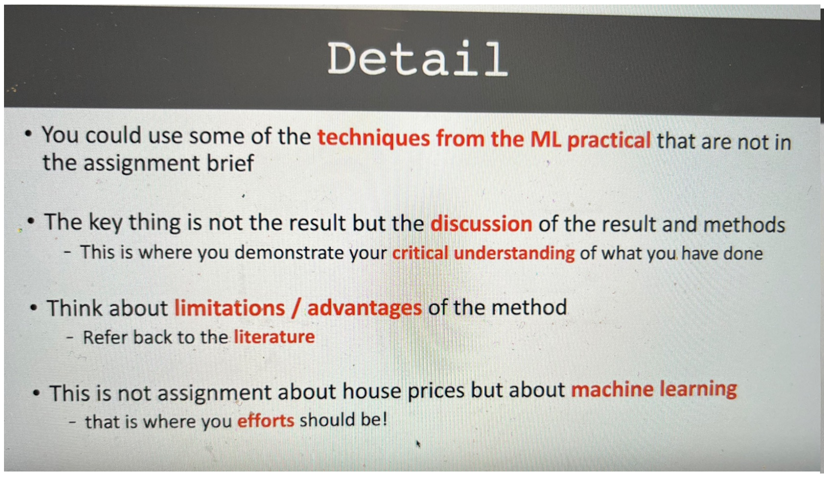
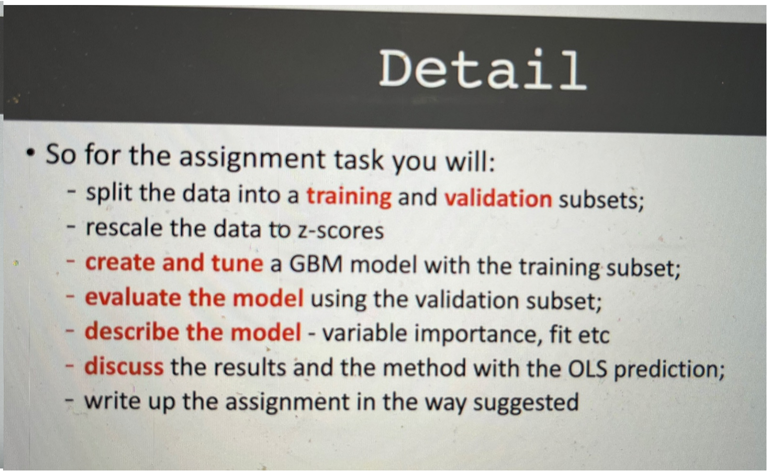


Detail

- You could use some of the techniques from the ML practical that are not in the assignment brief
- The key thing is not the result but the discussion of the result and methods
  - This is where you demonstrate your critical understanding of what you have done
- Think about limitations / advantages of the method
  - Refer back to the literature
- This is not assignment about house prices but about machine learning
  - that is where you efforts should be!



Detail

- So for the assignment task you will:
  - split the data into a training and validation subsets;
  - rescale the data to z-scores
  - create and tune a GBM model with the training subset;
  - evaluate the model using the validation subset;
  - describe the model - variable importance, fit etc
  - discuss the results and the method with the OLS prediction;
  - write up the assignment in the way suggested

Contents

Part 1: Description of the Task

The assignment is a 2500 word project report due before 2pm on Thursday 10th March 2022.

Your task is to construct a predictive Gradient Boosting Machine (GBM) model of house prices using the Boston House dataset (see here).

You will be shown how to tune and construct a GBM model using the `iris` dataset. You then have to construct one for house prices, selecting and transforming any input variables as needed.

You then need to write up what you have done, justifying and discussing any analytical choices you have made, and evaluating the model (normalising data, training validation splits, variable importance etc)

The report should contain the following sections:

1. **Introduction**: describe the problem, the study aims and the potential advantages of the GBM approach to be used (10 marks)
2. **Methods**: describe the data, provide a description any data pre-processing, etc that will need to be undertaken, and your choice of variables for the final model (10 marks)
3. **Results**: describe the GBM model, its tuning, evaluation and its application. This includes the tuning results (is it a good model?), the application to validation / tests data, and other considerations (30 marks)
4. **Discussion**: critically evaluate the model and the results, the method (limitations, assumptions, etc, linking back to the literature and any areas of future / further work (30 marks)

Up to 20 marks will be awarded for presentational clarity, correct and consistent referencing and critical reflection. **NB** The indication of marks in this assignment are a guide to where your effort should be directed and students should remember that the assignments will be marked using the standard School of Geography marking criteria for Masters assignments.

Please read through this assignment carefully before starting:

Part 2: creating a GBM model using the `iris` data

The code below loads some packages and the `iris` data. This has 5 variables and the the GBM is constructed to predict the `Sepal.Length`

```r
## load data and packages
library(caret)
library(gbm)
library(tidyverse)
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

The `createDataPartition` function can be used to create training and testing (validation) subsets

```r
set.seed(1234) # for reproducibility
train.index = createDataPartition(iris$Sepal.Length, p = 0.7, list = F)
data.train = iris[train.index,]
data.test = iris[-train.index,]
```

The distributions of the target variable are similar across the 2 splits:

```r
summary(data.train$Sepal.Length)
summary(data.test$Sepal.Length)
```

The predictor variables should be rescaled in each subset:

```r
data.train.z =
  data.train %>% select(-Sepal.Length) %>%
  mutate_if(is_logical,as.character) %>%
  mutate_if(is_double,scale) %>%  data.frame()
data.test.z =
  data.test %>% select(-Sepal.Length) %>%
  mutate_if(is_logical,as.character) %>%
  mutate_if(is_double,scale) %>%  data.frame()
# add unscaled Y variable back
data.train.z$Sepal.Length = data.train$Sepal.Length
data.test.z$Sepal.Length = data.test$Sepal.Length
```

Inspect the help for GBM and you will see that it takes a number of parameters that can be adjusted or tuned:

```r
modelLookup("gbm")
```

```
##   model         parameter                   label forReg forClass probModel
## 1   gbm           n.trees   # Boosting Iterations   TRUE     TRUE      TRUE
## 2   gbm interaction.depth          Max Tree Depth   TRUE     TRUE      TRUE
## 3   gbm         shrinkage               Shrinkage   TRUE     TRUE      TRUE
## 4   gbm    n.minobsinnode Min. Terminal Node Size   TRUE     TRUE      TRUE
```

A tuning grid can be set for these, and evaluation metric defined:

```r
## Set up tuning grid
caretGrid <- expand.grid(interaction.depth=c(1, 3, 5), n.trees = (0:50)*50,
                   shrinkage=c(0.01, 0.001),
                   n.minobsinnode=10)
```

```r
metric <- "RMSE"
```

And the `trainControl()` function used to define the type of 'in-model' sampling and evaluation undertaken to iteratively refine the model. It generates a list of parameters that are passed to the train function that creates the model. Here a simple 10 fold cross validation will suffice:

```r
trainControl <- trainControl(method="cv", number=10)
```

Then the model can be run over the grid, setting a seed for reproducibility:

```r
## run the model over the grid
set.seed(99)
gbm.caret <- train(Sepal.Length ~ ., data=data.train.z, distribution="gaussian", method="gbm",
              trControl=trainControl, verbose=FALSE,
              tuneGrid=caretGrid, metric=metric, bag.fraction=0.75)
```

And the result examined:

```r
## Examine the results
print(gbm.caret)
ggplot(gbm.caret)
# explore the results
names(gbm.caret)
# see best tune
gbm.caret[6]
```

```r
# see grid results
head(data.frame(gbm.caret[4]))
# check
dim(caretGrid)
dim(data.frame(gbm.caret[4]))
```

The best parameter combination can be determined:

```r
## Find the best parameter combination
# put into a data.frame
grid_df = data.frame(gbm.caret[4])
```

```r
head(grid_df)
```

```r
# confirm best model and assign to params object
grid_df[which.min(grid_df$results.RMSE), ]
```

```
##     results.shrinkage results.interaction.depth results.n.minobsinnode
## 202              0.01                         1                     10
##     results.n.trees results.RMSE results.Rsquared results.MAE results.RMSESD
## 202            2400    0.3468742        0.8389404    0.286335     0.08255408
##     results.RsquaredSD results.MAESD
## 202         0.07432373    0.07196397
```

```r
# assign to params and inspect
params = grid_df[which.min(grid_df$results.RMSE), 1:4 ]
params
```

```
##     results.shrinkage results.interaction.depth results.n.minobsinnode
## 202              0.01                         1                     10
##     results.n.trees
## 202            2400
```

These can be used in the final model:

```
## Create final model
# because parameters are known, model can be fit without parameter tuning
fitControl <- trainControl(method = "none", classProbs = FALSE)
# extract the values from params
gbmFit <- train(Sepal.Length ~ ., data=data.train.z, distribution="gaussian",  method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE,
                 ## only a single model is passed to the
                 tuneGrid = data.frame(interaction.depth = 1,
                                       n.trees = 2400,
                                       shrinkage = 0.01,
                                       n.minobsinnode = 10),
                 metric = metric)
```

The final model can be evaluated, in this case using same data as the model was trained on, and predictions evaluated against observations as in Figure 1:

```
## Prediction and Model evaluation
# generate predictions
pred = predict(gbmFit, newdata = data.test.z)
# plot these against observed
data.frame(Predicted = pred, Observed = data.test.z$Sepal.Length) %>%
    ggplot(aes(x = Observed, y = Predicted))+ geom_point(size = 1, alpha = 0.5)+
    geom_smooth(method = "lm")
```

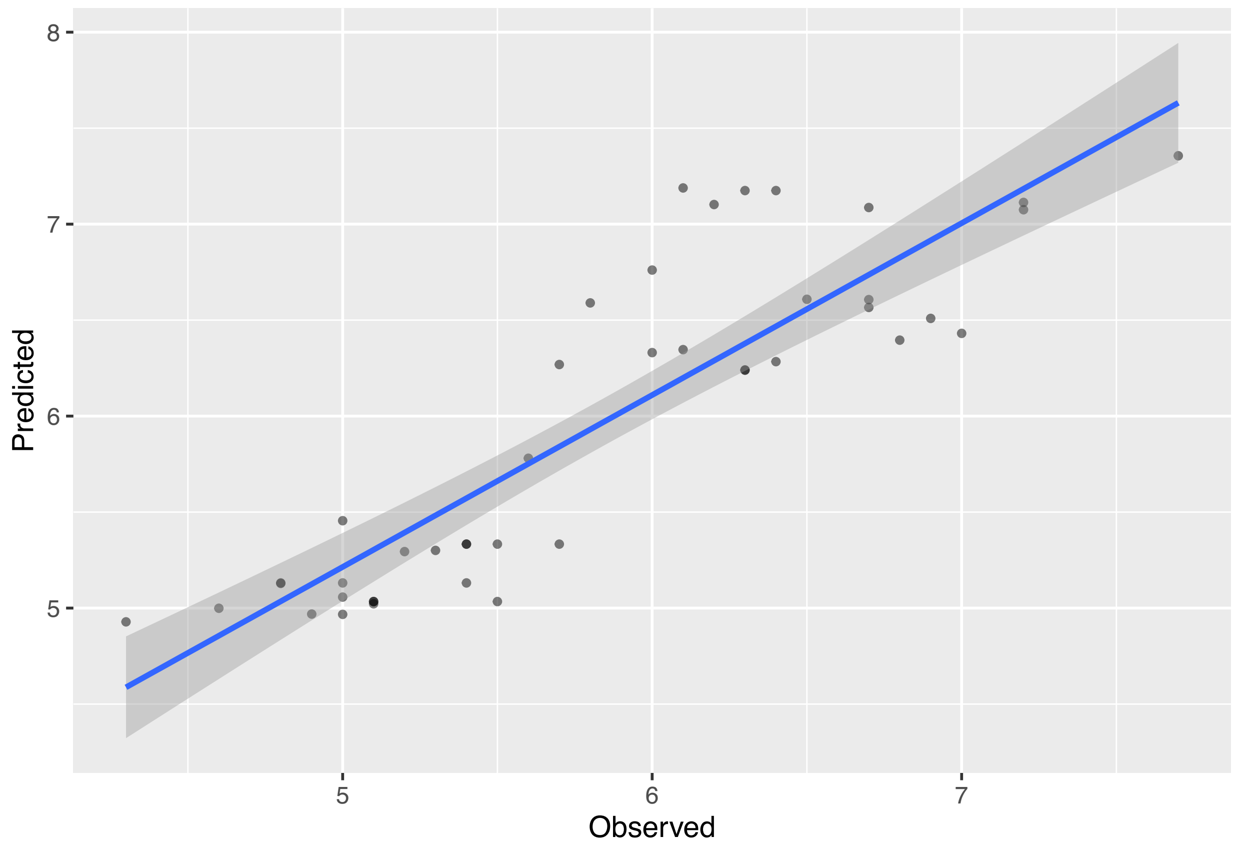


Figure 1: Repdicted against Observed Sepal.Length.

```r
# generate some prediction accuracy measures
postResample(pred = pred, obs = data.test.z$Sepal.Length)
```

```
##      RMSE  Rsquared       MAE
## 0.4196435 0.7724129 0.3147505
```

```r
# examine variable importance
varImp(gbmFit, scale = FALSE)
```

```
## gbm variable importance
##
##                    Overall
## Petal.Length      1752.146
## Petal.Width        267.649
## Sepal.Width        192.888
## Speciesversicolor   49.279
## Speciesvirginica     1.842
```

Part 3: Data for Task

Your task is to develop a GBM model of house price (`medv`) using the `BostonHousing` data from the `mlbench` package:

```r
# load and examine the data
library(mlbench)
data(BostonHousing)
head(BostonHousing)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio      b lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Descriptions of the variables can be found in the help for the data:

```r
?BostonHousing
```

In your Assignment model you should:

1. Split the data into training and testing (validation) subsets using the `createDataPartition` function;
2. Rescale the training and validation subsets, remembering to keep the target variable in its original form.
3. Evaluate the model using the testing subset.