Uppsala University

Introduction to Machine Learning, Big Data, and AI

Assignment 3

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**.
- Report all results in a single, *.pdf -file. *Other formats, such as Word, Rmd, or similar, will automatically be failed.*
- The report should be submitted to the Studium.
- A report that do not contain the general information (see template) will be automatically rejected.
- When working with R, we recommend writing the reports using R markdown and the provided **R markdown template**. The template includes the formatting instructions and how to include code and figures.
- If you have a problem with creating a PDF file directly from R markdown, start by creating an HTML file, and then just print the HTML to a PDF.
- Instead of R markdown, you can use other software to make the pdf report, but the same instructions for formatting should be used. These instructions are also available in **the PDF produced from the R markdown template**.
- The course has its own R package `uuml` with data and functionality to simplify coding. To install the package just run the following:
  1. `install.packages("remotes")`
  2. `remotes::install_github("MansMeg/IntroML",`
     `subdir = "rpackage")`
- We collect common questions regarding installation, and technical problems in a course Frequently Asked Questions (FAQ). This can be found **here**.
- Deadline for all assignments is **Sunday at 23.59**. See the course page for dates.
- If you have any suggestions or improvements to the course material, please post in the course chat feedback channel, create an issue, or submit a pull request to the public repository!

1 Decision Trees

The dataset Hitters contain data on the salary of Baseball players and additional information on years in the baseball league (`Years`) and the number of hits (`Hits`). To access the data, use

```
library(uuml)
data("Hitters")
```

We are now going to build an algorithm to grow decision trees. The algorithm used below is based on the algorithm explained in Chapter 9.2.2 in ESL.

1. Create a test set by setting aside the 30 first observations. Remove those observations from your training set.

```
# Remove NA values
Hitters <- Hitters[complete.cases(Hitters),]

# Create test and training set
X_test <- Hitters[1:30, c("Years", "Hits")]
y_test <- Hitters[1:30, c("Salary")]
X_train <- Hitters[31:nrow(Hitters), c("Years", "Hits")]
y_train <- Hitters[31:nrow(Hitters), c("Salary")]
```

2. Implement an R function to split observations binary greedily (ignore `NA` values). Here we use the algorithm in ESL. The function should take a design matrix `X`, an `y` variable and a minimal leaf size `l`. The function should return a list with the index set of the observations in `R1`, `R2`, the split value `s` and the covariate used for the split `j`. See below for an example.

```
# Lets choose a small data to try out our algorithm with
X_check <- Hitters[31:50, c("Years", "Hits")]
y_check <- Hitters[31:50, c("Salary")]
# These are the names of the players we look at
rownames(Hitters)[31:50]

##  [1] "-Bob Melvin"       "-BillyJo Robidoux" "-Bill Schroeder"
##  [4] "-Chris Bando"      "-Chris Brown"      "-Carmen Castillo"
##  [7] "-Chili Davis"      "-Carlton Fisk"     "-Curt Ford"
## [10] "-Carney Lansford"  "-Chet Lemon"       "-Candy Maldonado"
## [13] "-Carmelo Martinez" "-Craig Reynolds"   "-Cal Ripken"
## [16] "-Cory Snyder"      "-Chris Speier"     "-Curt Wilkerson"
## [19] "-Dave Anderson"    "-Don Baylor"

# This is how it should work
tree_split(X_check, y_check, l = 5)

## $j
## [1] 1
##
```

```
## $s
## [1] 6
##
## $R1
##  [1]  1  2  3  5  6  9 13 16 18 19
##
## $R2
##  [1]  4  7  8 10 11 12 14 15 17 20
```

3. What is the first split based on the whole training data `X_train` and `y_train`?
4. Use the function `tree_split()` to create a function `grow_tree()` that takes the arguments `X`, `y`, and `l` and then build a decision tree. The returned tree can be a `data.frame` that looks as follows. Note that `R1_i` and `R2_i` indicates the row of the `data.frame` where to go next. This is just one example implementation, you are free to implement this however you want.

```
tr <- grow_tree(X_check, y_check, l = 5)
tr


##    j   s R1_i R2_i gamma
## 1  1   6    2    3    NA
## 2  1   4    4    5    NA
## 3  2 102    6    7    NA
## 4 NA  NA   NA   NA   317
## 5 NA  NA   NA   NA   496
## 6 NA  NA   NA   NA   274
## 7 NA  NA   NA   NA   539
```

5. Finally implement a function `predict_with_tree(new_data, tree)` to use the tree to predict new observations X_{new} as follows:

```
X_new <- Hitters[51:52, c("Years", "Hits")]
X_new


##                  Years Hits
## -Daryl Boston        3   53
## -Darnell Coles       4  142


y_new <- Hitters[51:52, c("Salary")]
y_new


## [1]  75 105


predict_with_tree(new_data = X_new, tree = tr)


## [1] 317 496
```

6. What is the mean squared error on the test set for a tree trained on the whole training data `X_train` and `y_train`?

1.1 Bagging

Now we can use our regression tree to do a Bagged Tree regression model. The main difference from the previous tree is that we now will draw B bootstrap samples from our dataset, then train B trees, use them for a combined bagged prediction.

See Section 8.7 in ESL, and especially Eq. 8.51 for details.

1. Implement a bagged tree regression model using `grow_tree()` and `predict_with_tree()`.
2. Train your bagged tree regression model on the training data and predict on the test data. What is the RMSE of your predictions on the test set?

2 Running standard tools for boosting and random forests

The last part of the assignment consists of using Random Forest and XGBoost (Boosted regression trees).

1. Use the `randomForest` R package to fit a random forest regression to the training data using the `randomForest` function. Note you would need to handle `NA`s and reformulate the data slightly.
2. How many variables are used at each split. Why do you get this number?
3. Use the trained random forest to predict the test set and compute the RMSE of your predictions. Compare your `randomForest` to your bagged tree algorithm.
4. Next, use the `xgboost` R package to fit a boosted regression tree model to the training data using the `xgboost` function. Note. Again, you would need to handle `NA`s and reformulate the data slightly.
5. What is the RMSE of the predictions using the boosted regression tree model? How does that compare to the random forest regression model?