Ensemble Learning

4375 Machine Learning with Dr. Mazidi

David Allen

April 4, 2022

For this homework I will use my regression dataset to improve my results from the R Project.

This is a regression dataset for predicting the death rate of the coronavirus based on location and date in 2020.

- Number of Rows: 50k

Source: https://www.kaggle.com/datasets/imdevskp/corona-virus-report

The three algorithms used are Linear Regression, kNN, and Decision Trees.

Load the Data

```
df <- read.csv("R Project/data/covid_19_complete.csv")
str(df)
```

```
## 'data.frame':    49068 obs. of  10 variables:
##  $ Province.State: chr  "" "" "" "" ...
##  $ Country.Region: chr  "Afghanistan" "Albania" "Algeria" "Andorra" ...
##  $ Lat           : num  33.9 41.2 28 42.5 -11.2 ...
##  $ Long          : num  67.71 20.17 1.66 1.52 17.87 ...
##  $ Date          : chr  "2020-01-22" "2020-01-22" "2020-01-22" "2020-01-22" ...
##  $ Confirmed     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Deaths        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Recovered     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Active        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ WHO.Region    : chr  "Eastern Mediterranean" "Europe" "Africa" "Europe" ...
```

Data Exploration and Cleaning

```
summary(df$Confirmed)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       4     168   16885    1518 4290259
```

```
summary(df$Deaths)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
##      0.0      0.0      2.0    884.2     30.0 148011.0
```

```
mean(df$Confirmed == 0)
```

```
## [1] 0.2050012
```

There is massive variance in the data. Confirmed cases range from 0 cases to 4 million! This will have costly effects on the algorithms.

It also appears about 20% of the data is simply 0's from the beginning of the pandemic.

```
sum(df$Province.State == "")
```

```
## [1] 34404
```

Province.State is a feature that is mostly blank for this data set. But ultimately, this doesn't matter for our problem. Latitude, longitude, and WHO region (as a factor) will account for the location data.

Cleaning Steps

- Convert Date to numerical data
- Convert region to a factor

We will need to adjust the date values from categorical to numerical to use in regression:

```
parseDate <- function(date) {
  numeric_date <- as.numeric(as.Date(date, "%Y-%m-%d"))
  return (numeric_date)
}
```

```
df$Date.Numeric <- sapply(df$Date, parseDate)
summary(df$Date.Numeric)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   18283   18330   18377   18377   18423   18470
```

Now let's scale the data:

```
df$Date.Numeric <- df$Date.Numeric - mean(df$Date.Numeric)
summary(df$Date.Numeric)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  -93.50  -46.75    0.00    0.00   46.75   93.50
```

```r
df$WHO.Region <- as.factor(df$WHO.Region)
str(df)
```

```
## 'data.frame':    49068 obs. of  11 variables:
##  $ Province.State: chr  "" "" "" "" ...
##  $ Country.Region: chr  "Afghanistan" "Albania" "Algeria" "Andorra" ...
##  $ Lat           : num  33.9 41.2 28 42.5 -11.2 ...
##  $ Long          : num  67.71 20.17 1.66 1.52 17.87 ...
##  $ Date          : chr  "2020-01-22" "2020-01-22" "2020-01-22" "2020-01-22" ...
##  $ Confirmed     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Deaths        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Recovered     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Active        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ WHO.Region    : Factor w/ 6 levels "Africa","Americas",..: 3 4 1 4 1 2 2 4 6 6 ...
##  $ Date.Numeric  : num  -93.5 -93.5 -93.5 -93.5 -93.5 -93.5 -93.5 -93.5 -93.5 -93.5 ...
```

Now we have 4 reliable features to use as predictors for regression: Lat, Long, Date.Numeric, and WHO.Region. There are 4 categories to predict, which could potentially work as features for each other: Confirmed, Deaths, Recovered, and Active. Let's predict Deaths from our 4 predictors.

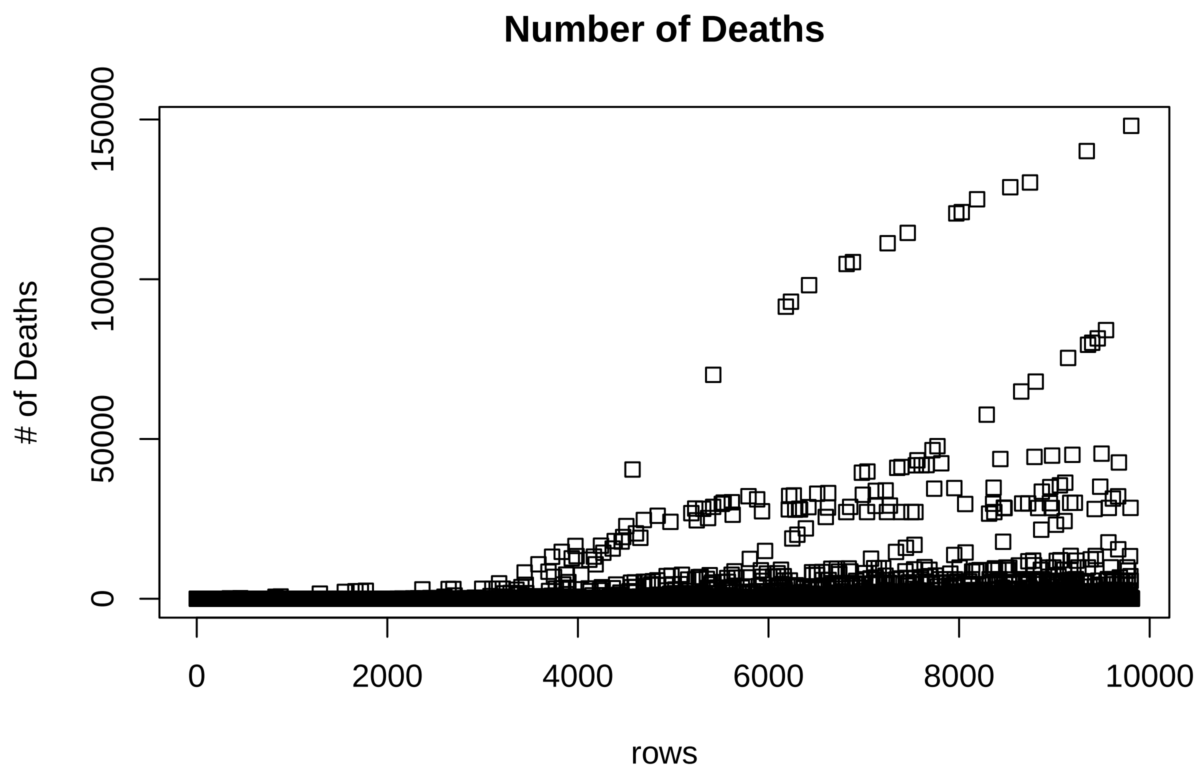
Testing the Model

```r
set.seed(777)
i <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train <- df[i,]
test <- df[-i,]
train_matrix <- data.matrix(train[,c(3, 4, 6, 11)])
test_matrix <- data.matrix(test[,c(3, 4, 6, 11)])
```

Data Summary

A reminder of what our test results look like. Very not straightforward.

```r
rows = 1:length(test$Deaths)
plot(rows, test$Deaths, main="Number of Deaths", ylab="# of Deaths", pch=0)
```



### Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.3
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
start <- proc.time()
rf <- randomForest(Deaths~Lat+Long+Date.Numeric+Confirmed, data=train, importance=TRUE)
proc.time()-start
```

```
##    user  system elapsed
##  337.84    1.50  355.65
```

```
rf
```

```
##
## Call:
##  randomForest(formula = Deaths ~ Lat + Long + Date.Numeric + Confirmed,      data = train, importanc
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 441948.5
##                     % Var explained: 98.92
```

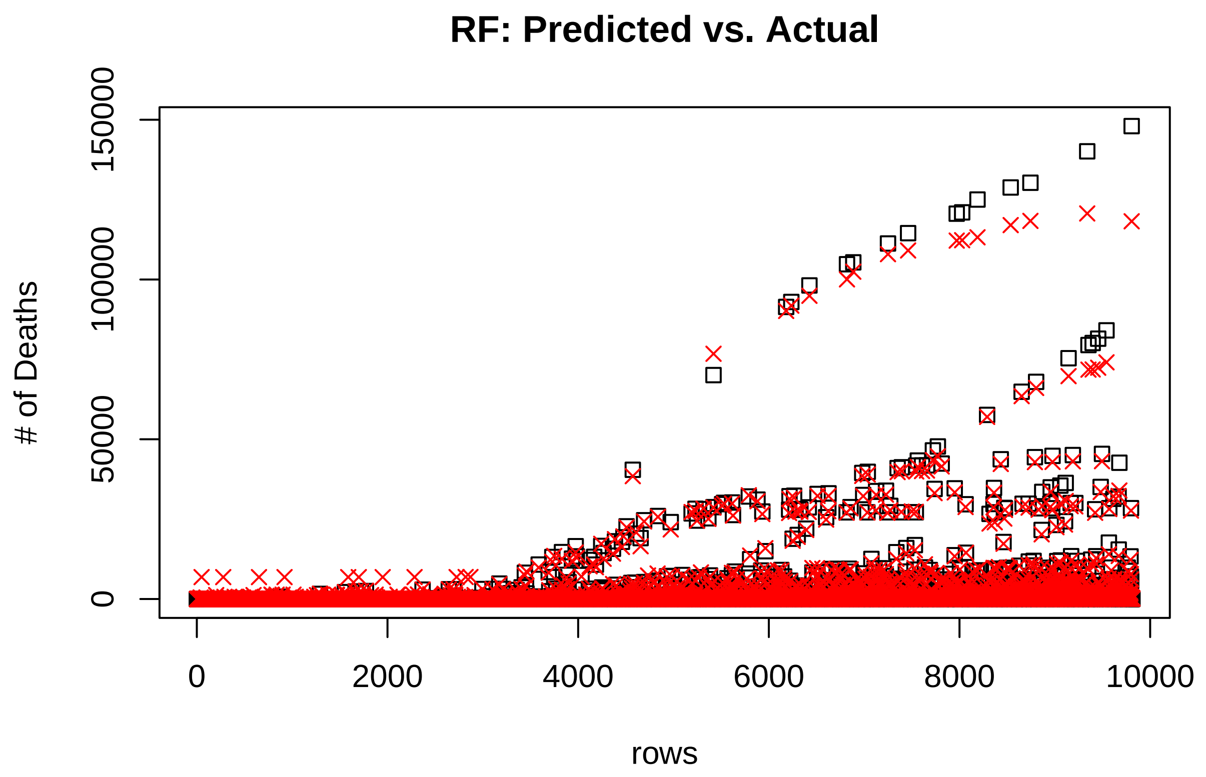
```
pred = predict(rf, newdata=test, type="response")
cor <- cor(pred, test$Deaths)
print(paste("Correlation: ", cor))
```

```
## [1] "Correlation:  0.996898192402959"
```

```
rmse <- sqrt(mean((pred - test$Deaths)^2))
print(paste("RMSE: ", rmse))
```

```
## [1] "RMSE:  593.42686980583"
```

```
plot(rows, test$Deaths, main="RF: Predicted vs. Actual", ylab="# of Deaths", pch=0)
points(rows, pred, pch=4, col="red")
```



### Bagging Bagging is a special case of Random Forests which uses all predictors.

```
start <- proc.time()
bag1 <- randomForest(Deaths~Lat+Long+Date.Numeric+Confirmed, data=train, mtry=4, importance=TRUE)
proc.time()-start
```

```
##    user  system elapsed
##  953.91    3.30  992.04
```

```
bag1
```

```
##
## Call:
##  randomForest(formula = Deaths ~ Lat + Long + Date.Numeric + Confirmed,      data = train, mtry = 4,
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 22344.98
##                     % Var explained: 99.95
```

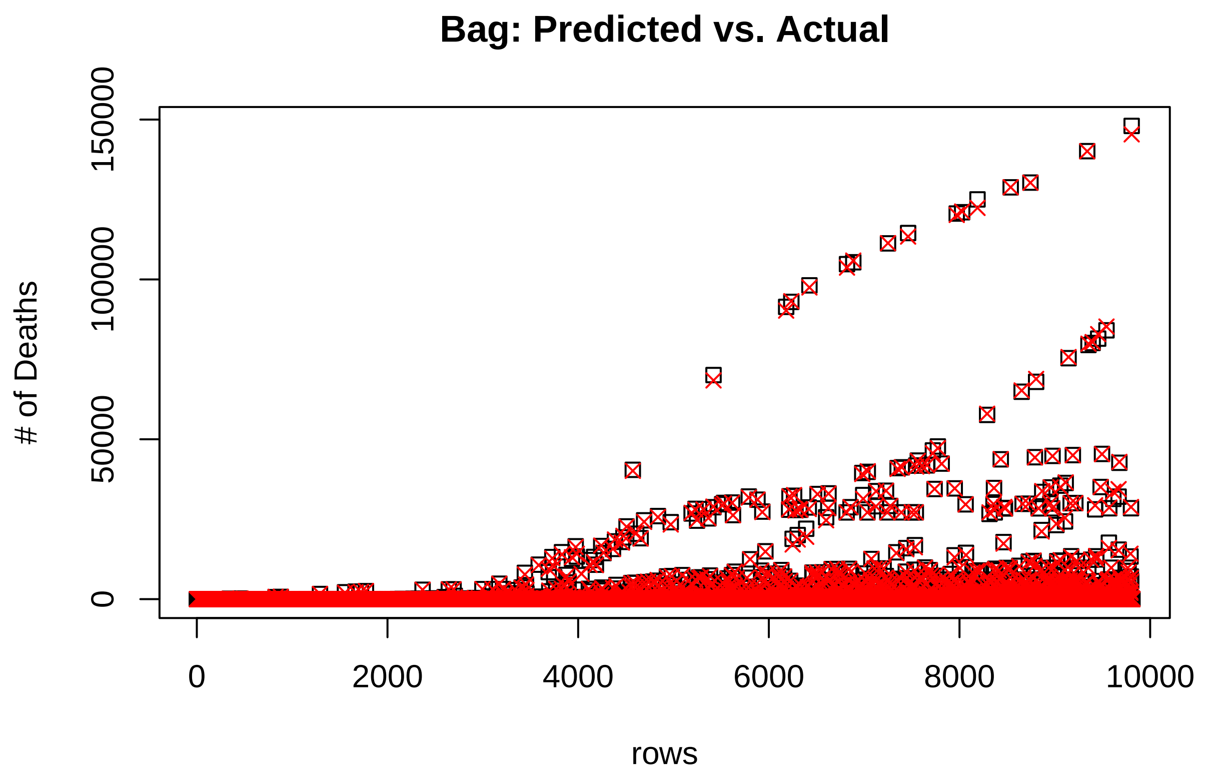
```
pred = predict(bag1, newdata=test, type="response")
cor <- cor(pred, test$Deaths)
print(paste("Correlation: ", cor))
```

```
## [1] "Correlation:  0.99986720921638"
```

```
rmse <- sqrt(mean((pred - test$Deaths)^2))
print(paste("RMSE: ", rmse))
```

```
## [1] "RMSE:  98.5045018605669"
```

```
plot(rows, test$Deaths, main="Bag: Predicted vs. Actual", ylab="# of Deaths", pch=0)
points(rows, pred, pch=4, col="red")
```



Adaboost

Apparently adaboost does not work for regression in R, so for this algorithm in particular we will do classification of WHO Region

```
library(adabag)
```

```
## Warning: package 'adabag' was built under R version 4.1.3
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.1.3

## Loading required package: doParallel

## Warning: package 'doParallel' was built under R version 4.1.3

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.1.3

## Loading required package: parallel
```

```
start <- proc.time()
adab1 <- boosting(WHO.Region~Date.Numeric+Confirmed+Deaths, data=train, boos=TRUE, mfinal=20, coeflearn
proc.time()-start
```

```
##    user  system elapsed
##   20.78    0.37   21.79
```

```
summary(adab1)
```

```
##            Length Class   Mode
## formula         3 formula call
## trees          20 -none-  list
## weights        20 -none-  numeric
## votes      235524 -none-  numeric
## prob       235524 -none-  numeric
## class       39254 -none-  character
## importance      3 -none-  numeric
## terms           3 terms   call
## call            6 -none-  call
```

```
library(mltools)
```

```
## Warning: package 'mltools' was built under R version 4.1.3
```

```
pred = predict(adab1, newdata=test, type="response")
acc <- mean(pred$class==test$WHO.Region)
print(paste("Accuracy: ", acc))
```

```
## [1] "Accuracy:  0.373446097411861"
```

```r
mcc <- mcc(factor(pred$class), test$WHO.Region)
print(paste("MCC: ", mcc))
```

```
## [1] "MCC:  0.170819385523896"
```

This performance is the worst by far.

XGBoost

```r
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.1.3
```

```r
xgb_train = xgb.DMatrix(data = train_matrix, label = train$Deaths)
xgb_test = xgb.DMatrix(data = test_matrix, label = test$Deaths)
```

Create train/test matrices.

```r
start <- proc.time()
model <- xgboost(data=xgb_train, max.depth=3, nrounds=1000, verbose=0)
proc.time()-start
```

```
##    user  system elapsed
##   24.49    1.14    9.03
```

```r
summary(model)
```

```
##                 Length  Class              Mode
## handle                1 xgb.Booster.handle externalptr
## raw             1167958 -none-             raw
## niter                 1 -none-             numeric
## evaluation_log        2 data.table         list
## call                 14 -none-             call
## params                2 -none-             list
## callbacks             1 -none-             list
## feature_names         4 -none-             character
## nfeatures             1 -none-             numeric
```

How did the model perform?

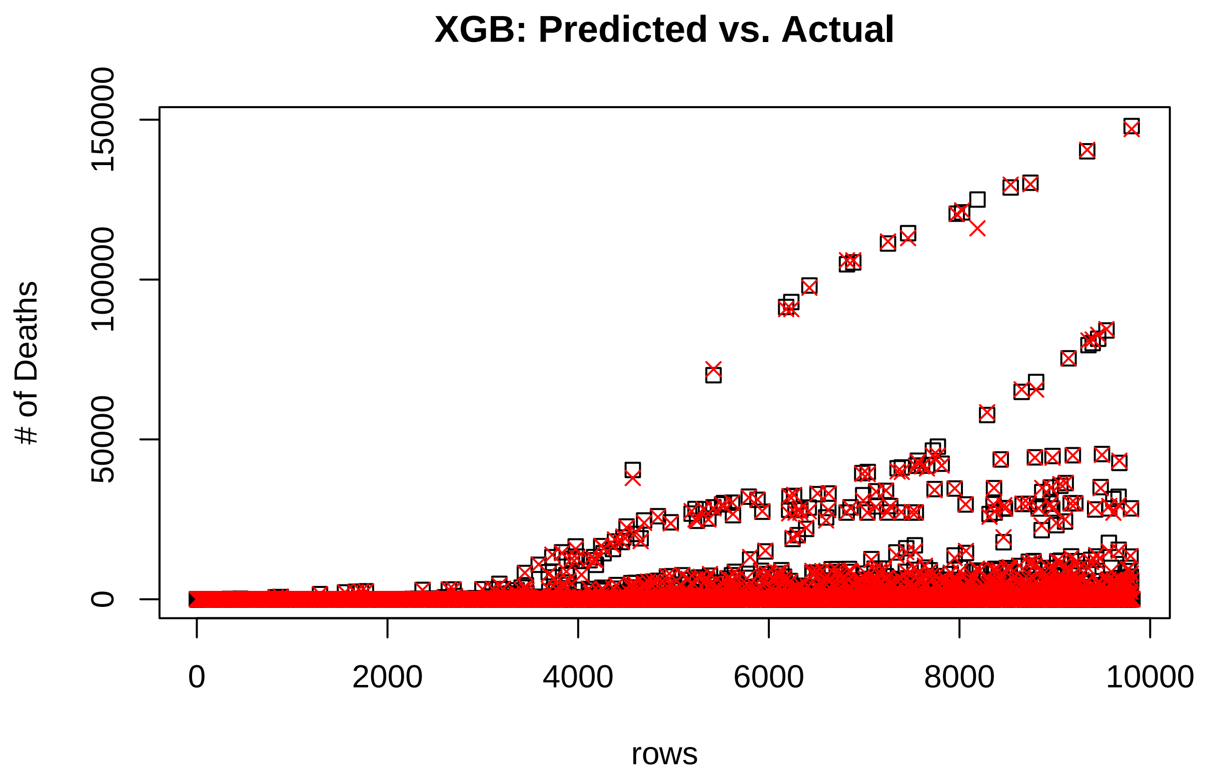
```r
pred = predict(model, xgb_test)
cor <- cor(pred, test$Deaths)
print(paste("Correlation: ", cor))
```

```
## [1] "Correlation:  0.999584810801145"
```

```r
rmse <- sqrt(mean((pred - test$Deaths)^2))
print(paste("RMSE: ", rmse))
```

```
## [1] "RMSE:  174.332434999602"
```

```r
plot(rows, test$Deaths, main="XGB: Predicted vs. Actual", ylab="# of Deaths", pch=0)
points(rows, pred, pch=4, col="red")
```



Summary

All of the applicable algorithms vastly outperformed those used in my R project, narrowing down the RMSE by a whole order of magnitude.

Random Forest

Random Forest performed well, although it only used one predictor (most likely Confirmed cases) on each decision tree, much to its detriment. The correlation was above 99%, but the algorithm took over five minutes to compute.

Bagging

Bagging had the highest correlation of all the algorithms (~99.8%) and the lowest RMSE (~101), making it the best-performing method. However, it took well over 10 minutes to run, also making it the slowest method. Clearly each predictor makes a difference, but comes with a cost.

Adaboost

Ignoring how well the other algorithms handled the regression problem, the adaboost algorithm still performed extremely poorly in classifying region based on date and number of cases. Granted, this data set is not ideal for classification, and accuracy is probably not a good metric for multi-class identification. This algorithm ran the fastest all things considered.

XGBoost

XGBoost performed phenomenally on the test data. Even though it ran 1000 rounds, it still completed in 30 seconds! While not as accurate as Bagging, the XGBoost algorithm had a rmse less than 200 in a dataset with values ranging from 0 to over 100,000, making it one of the best models overall.