ACTL4305 Assignment 2

Contents

Executive Summary

This report analyses the results of using 3 tree-based modelling techniques to estimate claims of motor vehicle insurance. These models are regression trees, random forest and bootstrap aggregation, or “bagging”. Firstly, to explore the underlying data trends, several of the most interesting predictor effects were analysed. Then, the data was prepared for modelling by removing any collinear terms and split into training and testing sets. Throughout the report the pure premium was estimated using the “lost-cost” approach and the test RMSE was used for model comparisons. The regression tree was the worst predictor, however, it was very interpretable, even for those without a statistical background. The random forest model was the most accurate model whilst also revealing how younger drivers and those with more claims usually have higher claims. Finally, bagging had an RMSE between the random forest and the regression tree and shows how vehicle size and value can be indicators of individual claim size. Therefore, since the chosen model will form part of the pricing for general insurance products, accuracy is prioritsed over interpretability and so this report recommends using random forest for any tree-based pure premium estimations. But, all models performed relatively poorly compared to GLM’s and so this report advises against taking a tree-based approach.

Data Exploration

One aspect of the data, is that it considerably imbalanced, as seen from table 1. Therefore, the data will largely be explored in two ways - by considering all the data or by considering only the subset of the data where inviduals made a claim. Plots where all data is present can often be largely uninformative due to the high concentration of zero claims.

Table 1: Data Imbalance

% 0 Claims	% At least 1 Claim
0.92	0.08

This imbalance can also be seen from the distribution of the claims incurred from individual policyholders in figure 1. It is clear that even when claimless individuals are removed, most of the claims are small.

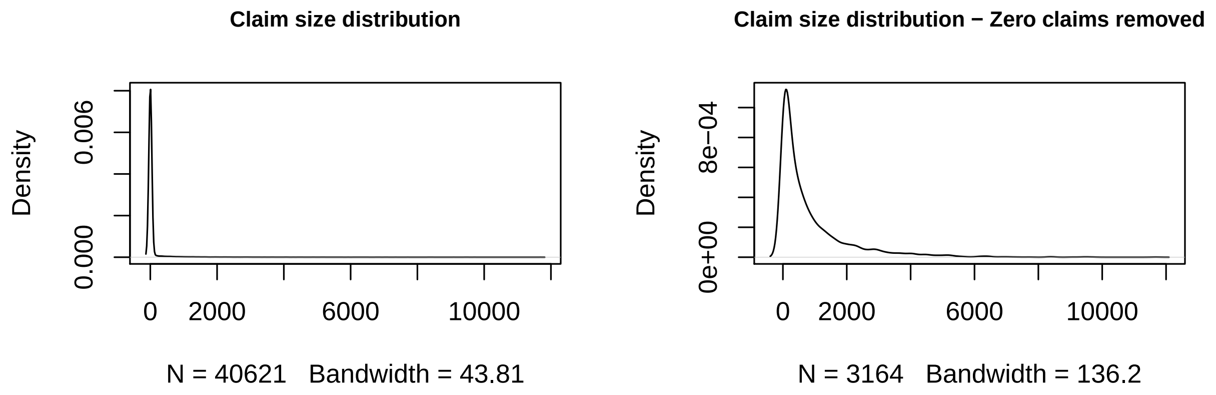

Figure 1: Distribution of Claim severity

This is an expected feature of most insurance data as accidents are rare events amoung individuals however, this can decrease the performance of models and so the effects of imbalanced data will be a consideration during the modelling phase, particularly when selecting an appropriate bias-variance tradeoff. This report will now consider some of the more interesting variable effects on pure premium.

Variable: Exposure

There is a negative correlation between exposure length and the pure premium. As length of exposure increases, generally, the severity of claims increase. However, the pure premium will decrease suggesting that those who buy insurance in larger lengths of time are, on average, safer drivers.

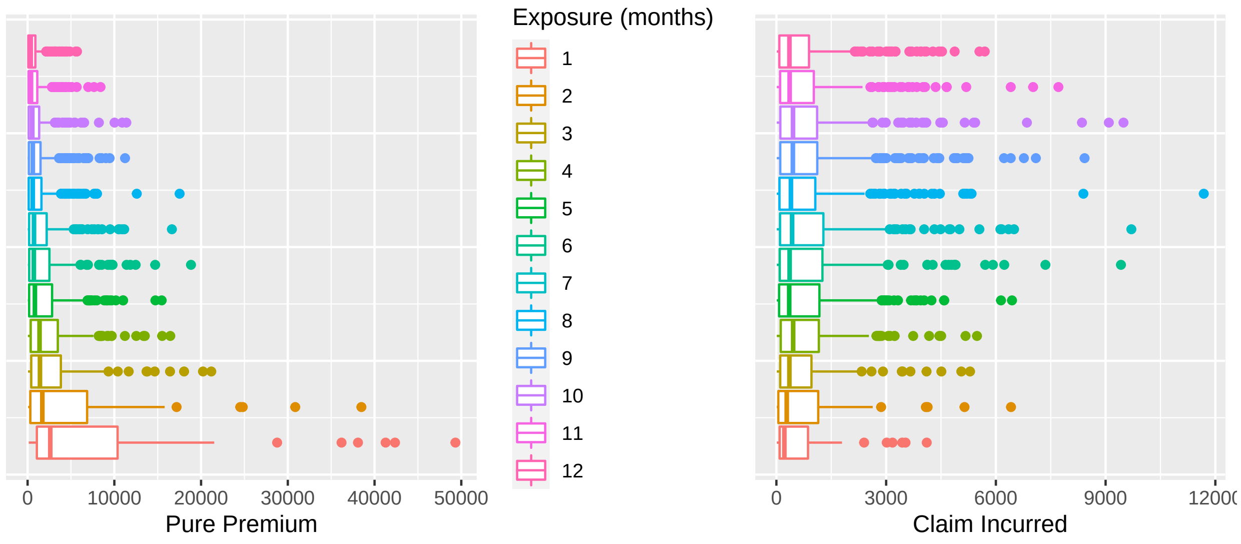

Figure 2: Exposures for Non-Zero Pure Premium and Claim Incurred

Variable: Driver Age

The most common driving age is 40 as seen from Figure 3.

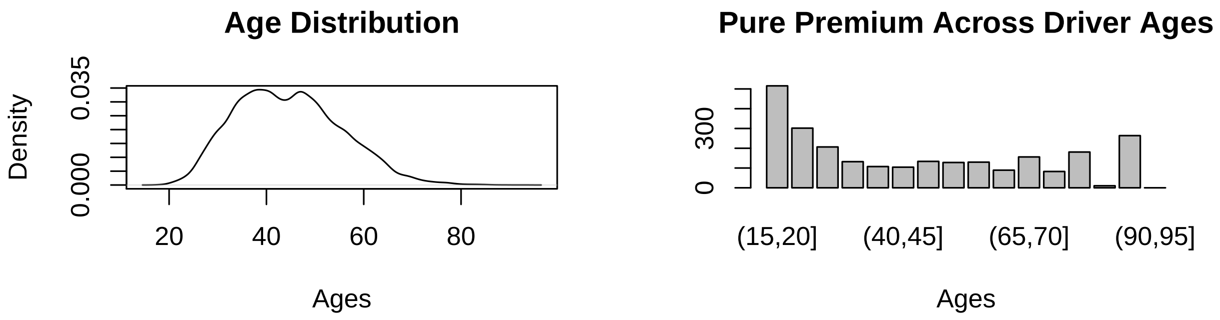

Figure 3: Age Summary

As expected, generally the worst drivers are those that are youngest and least experienced. Drivers again become more risky when they are 80 years old as seen from the spikes in pure premium for elderly drivers. This results in what appears to be a bimodal distribution for the average pure premium across all age groups.

Variable: Prior Claims and NCD level

As expected, there seems to be a positive trend between the number of past claims and the pure premium suggesting that those who have a history of unsafe driving, generally will remain more risky. The more past claims an individual has, the larger the pure premium is expected to be.

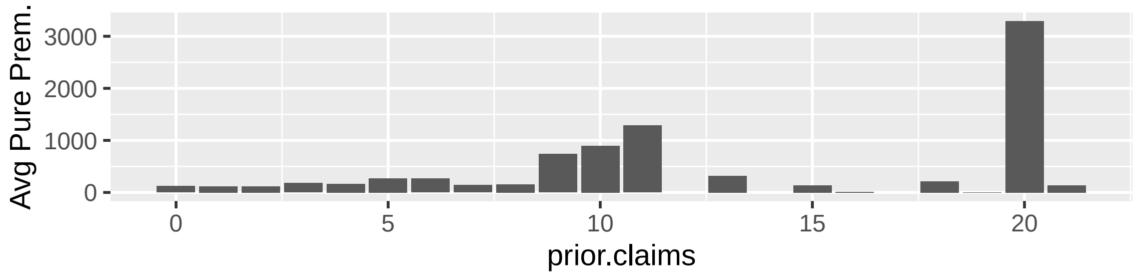

Figure 4: Pure Premium For Number of Prior Claims

Interestingly, there seems to be little correlation between NCD level and past claims as policyholders at level 1 are considered the most risky, but have the least claims on average. However, the prior NCD level still seems to be informative as generally, the higher level of discount indicates a smaller pure premium. The only noteable exception to this trend is level 2 which make the smallest claims despite being the second most expensive tier.

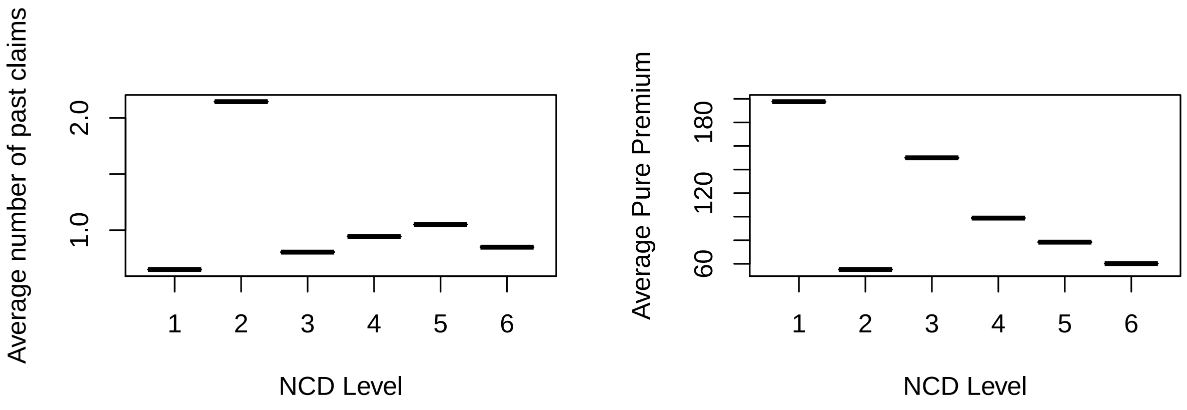

Figure 5: NCD Level vs Past Claims and Pure Premiums

Correlation Among Features

To avoid multicolinearity modelling problems, to reduce run time and increase the interpretation of model outputs, highly correlated terms were removed from the data. From figure 6, there are some correlated terms, including -

- Vehicle Value and Vehicle Age at -0.67
- Weight and length at 0.84
- Horsepower and length at 0.66
- Horsepower and weight at 0.78

Therefore, weight, length and vehicle age were removed from the data.

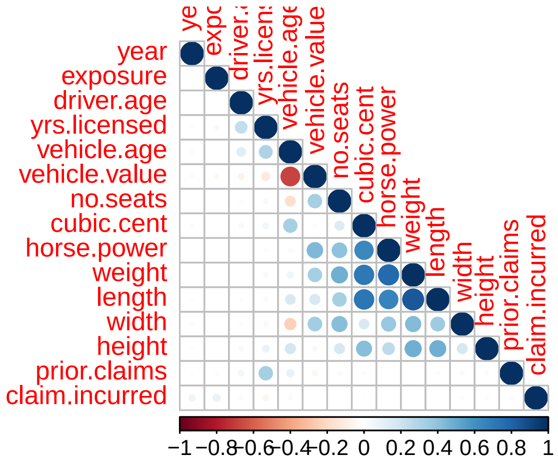

Figure 6: Correlation Plot

Data changes before modelling

Prior to modelling, 70% of the data was split into a training set and the remaining 30% was reserved for testing. Claim count was also removed from the dataset since at the start of a policy an insurer does not have access to this information.

Modelling Methodology

The **lost-cost** approach to estimating the pure premium was used for all tree-based modelling. The claim incurred was used as the response variable to tune hyperparameters and to choose the best performer from each model. For final comparisons amongst different model families, the claim incurred was divided by the exposure to obtain estimates of the pure premium. It was these estimates which were used for the test RMSE.

Feature selection is generally not required for tree-based models since it is done implicitly at each node. Therefore there will be no feature selection in this report. For each model, the hyperparamters were tuned through the "Caret" package and a 5 fold cross validation. 5 fold cross validation was deemed adequate enough to produce results that were significant whilst also having relatively short run times. Even without repeated cross validation or higher folds, each random forest model could take an hour to run.

The bagging and random forest were trained through parrallel computing in Caret.

Regression tree

Test Error: 1084.5

The regression tree was the least accurate model, although it was only marginally worse than either random forest or bagging. The complexity parameter (CP) was tuned through a grid search and the best value found was 0.004.

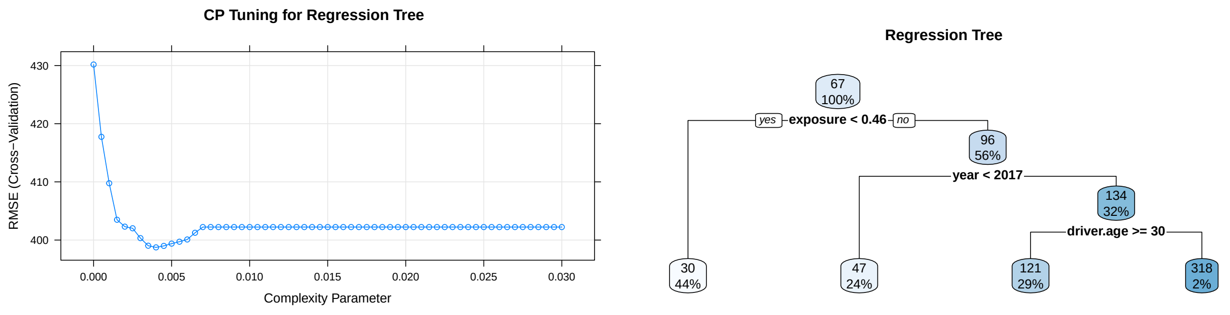

Figure 7: Regression Tree CP Tuning and Tree

Interestingly, any CP value greater than 0.007 produced a zero-split tree, which only had slightly worse predictive power than the standard tree. Therefore, this suggests that the regression tree is not a particularly strong model.

Regression tree: Advantages

The major advantage of the regression tree is its interpretability, it is clear that high exposure, recent policies and younger drivers can all be indicative of larger claims. This information is helpful for business decisions and to generalise the risk an individual poses.

Regression tree: Disadvantages

The main disadvantage of any regression tree is that it will rarely match the accuracy of any other model. From table 2, the best regression tree is only slightly worse than the zero-tree split and so clearly the regression tree is a bad predictor.

Regression tree: Conclusion

Although the basic regression tree has great interpretability and can even be understood by those without a statistical background, it is also the worst predictor, and since our chosen model will be used to price insurance products, accuracy has been prioritised over interpretability.

Table 2: Regression Tree Summary

Model	Complexity Paramter	Test RMSE
Regression Tree	0.004	1084.5
Zero-split Tree	> 0.007	1105.24

Random Forest

Test error: 1082.77

There were 3 hyperparamters tuned for random forest- the number of trees, node size and the number of predictors considered at each tree split, commonly referred to as "m". Since random forest is computationally expensive, each predictor was optimised in isolation from the others. Although this is will not find the best combination of predictors, due to computational limitations it was a necessary tradeoff.

Number of trees:

Although random forest cannot be overfitted with large number of trees, in this data, any number greater than 250 produced a relatively similar CV error and so 250 trees were used for the final model.

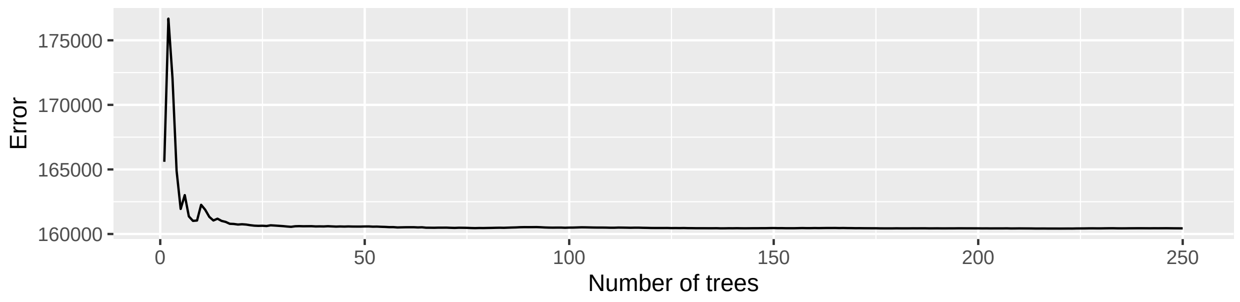

Figure 8: RF MSE changes with number of trees

Note: at 100 trees, the test error remains largely unchanged and so to create more models faster, 100 trees was deemed adequate enough to train the other parameters.

Node Size:

The minimum number of observations at each tree node, or **node size**, can be altered to select model complexity and the optimal level of trade-off in the bias-variance approach. Using Caret's default regression node size value of 5 produced predictions less accurate than even the standard regression tree. Therefore the node size was trained as a hyperparamter however, one of the limitations in this search was the computational cost of training each model. By incrementing the node size by 100 for each model, the optimal value was 2500. Although figure 8 appeared to show a rising MSE after a node size of 2500, larger node sizes should also be tested to avoid potential early stopping. Since it was also necessary to make large 100 node increments, the true optimal value may lie between two tested points. Unfortunately, the time complexity of this training meant that the problem was too expensive for the current equipment, however, a value of 5000 was also used to validate that there was not significant early stopping. Therefore, more research should be done to ensure the most optimal node size was found.

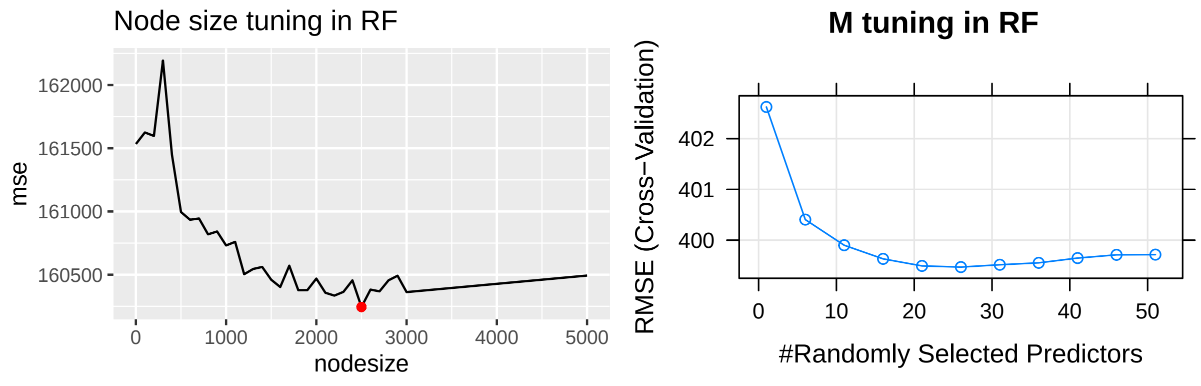

Figure 9: RF Nodesize and M Tuning

Number of predictors considered at each split:

“M” values between 1 and 51 were modelled and the value which minimised the CV error was 26. This grid search was deemed adequate, since there is a clear minimum point in the CV error in figure 9. This means that there are 26 of the 67 total variables which are randomly considered at each split of the tree. Finally these values were combined to create the final random forest model with 250 trees, a minimum of 2500 observations in each node and an m value of 26.

Random Forest: Advantages

The clearest advantage of random forest is its predictive improvements over both bagging and regression trees, however, random forest also has some interpretation from the VIP and PDPs.

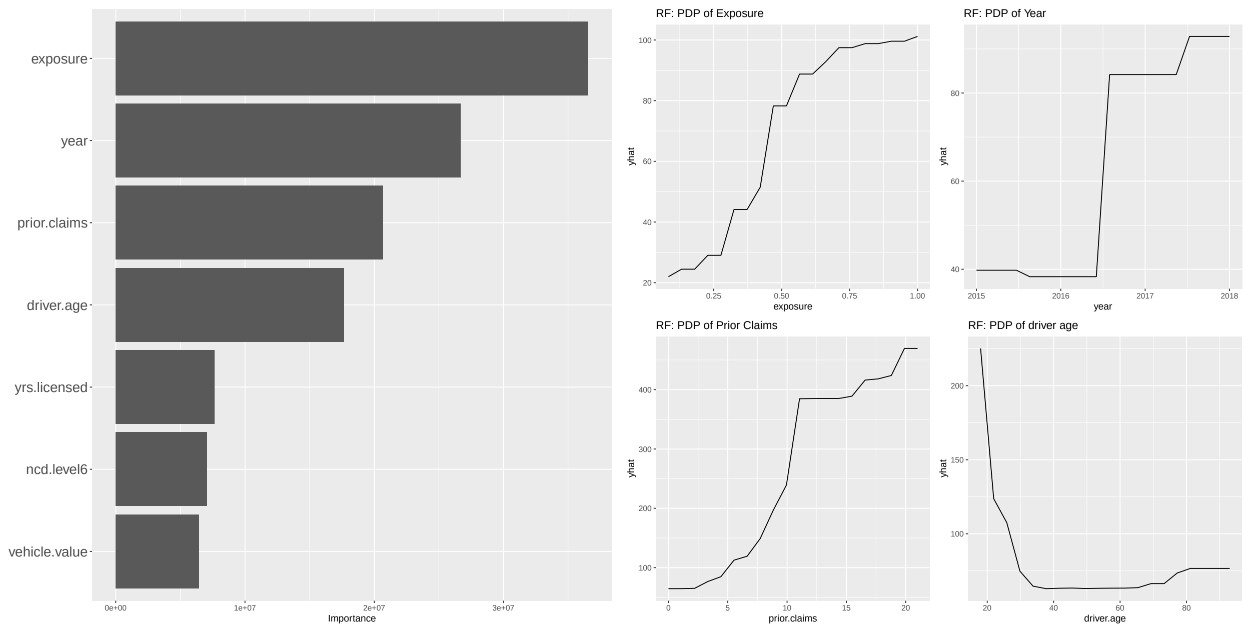

Figure 10: RF Variable Importance and PDP

Figure 10 shows that a policyholders exposure, their past claims, their age and the current year are all important predictors in determining the expected claim incurred. The PDP's also reveal that drivers younger than 30 are much more likely to make more severe claims than their older counterparts. Generally, claim severity decreases with age, except for those older than 80 when a policyholder becomes more risky. The other three variables are more or less expected with all positively correlated with the claim amounts. Although, there does seem to be a significant increase in the claim incurred from 2016 to 2017.

Random Forest: Disadvantages

The biggest obstacle in this report was the computational time and the need to often fit multiple models at once in order to select the best hyperparamter. Whilst this means that creating the initial model is difficult, it also makes the random forest less responsive to new information as it will take a long time to retrain. The second disadvantage is the loss of interpretability that often occurs with many ensemble, or "black box", models. Although random forest is less interpretable than the basic regression tree, the PDP and VIP's demonstrate the effect of variables on the overall estimations.

Random Forest: Conclusion

Overall, the random forest model was a relatively poor model when compared to GLM, yet still outperformed all other tree-based methods. Although it was computationally slow to build and less interpretable than the regression tree, the improvements in the predictive power make up for these disadvantages. To refine this model, more powerful equipment should be used to consider more minimum node sizes and select the best combination of parameters.

Table 3: Regression Tree Summary

Model	Number of Trees	Nodesize	m	Test RMSE
Regression Tree	250	2500	26	1082.77

Bagging

Test Error: 1083.79

Bagging had to tune both the number of trees and the node size. However, it was deemed appropriate to use 250 trees since bagging is a subset of random forest. The main issue with fitting a bagging model was choosing the best node size. As seen from the plot below, there were no clear trends during the parameter tuning, with error values only ranging 1.85 MSE values. Since bagging is growing a series of regression trees based on bootstrapped data, the CP value found from the regression tree was used to generate the model. Although this method produced the lowest test RMSE of all bagged models, in future, more node size values should be tested.

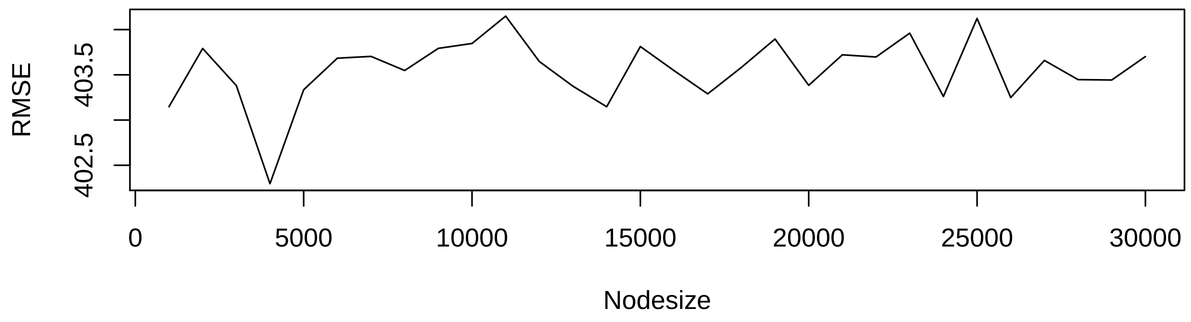

Figure 11: Bagging node size tuning

Bagging: Advantages

The bagging model is somewhat of a middle ground between the regression tree and random forest as it is more accurate than the decision tree, but less accurate than random forest. Since bagging does not have the "m" parameter, it is also more interpretable than random forest. As an ensemble method, much of its interpretation comes from PDP and VIP's. Unlike random forest, there does not seem to be a clear group of best predictors but instead many that are roughly equivalent. The bagging model uses vehicle value, horse power and the vehicle size more frequently than the random forest. But, like random forest, driver age is a significant indicator of individual claim severity, particularly for younger, less experienced drivers. Larger vehicles also can be indicative of more severe claims.

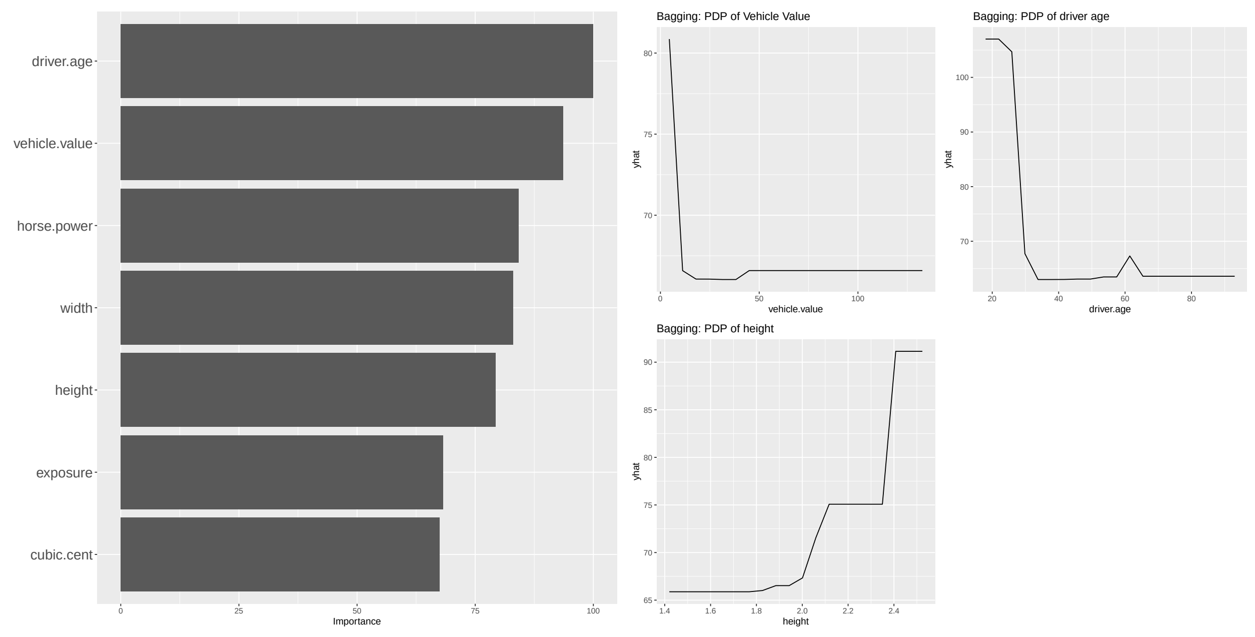

Figure 12: Bagging Variable Importance and PDP

Bagging: Disadvantages

As bagging is a middle ground between random forest and decision trees, it is both not the best predictor and suffers from being not easily understood. It is also computationally expensive, although not as complex as random forest since the "m" value does not have to be trained.

Bagging: Conclusion

The bagging model performs only slightly better than the regression tree but also loses much of its interpretability. Therefore, it is not a good model.

Table 4: Regression Tree Summary

Model	Complexity Paramter	Number of Trees	Test RMSE
Bagging	0.004	250	1083.79

Conclusion

Overall, this report recommends selecting the random forest to model the pure premium. The regression tree is a poor predictor, but does clearly demonstrate how policy year, driver age and exposure can be indicative of claim size. Random forest supports these generalities and adds understanding of how these variables affect the severity of claims through partial dependency plots. The bagging model was better than the regression tree, but ultimately falls short of the predictive power of the random forest. However, seperate team members have shown that the test RMSE for the GLM was 533.53, a significant improvement over even the random forest. Therefore, all tree-based models have a high testing error and are not great predictors, but, the best of these models is random forest.

Table 5: Regression Tree Summary

Model	RMSE
zero-split tree	1105.24
Regression Tree	1084.5
Random Forest	1082.77
Bagging	1083.79
GLM	533.53

Appendix

R code-

```r
library(randomForest)
library(rpart.plot)
library(EnvStats)
library(tidyverse)
library(corrplot)
library(ROSE)
library(caret)
library(vip)
library(pdp)
library(pROC)
library(ROCR)

library(parallel)
library(doParallel)

setwd("~/University/year3/T3/ACTL4305/Assignment/Assignment 2")

# Importing data
data <- read_csv("A2-data.csv")[,-1]

factor_cols <- c("business.type", "driver.gender", "marital.status", "ncd.level", "region", "body.code"
data <- data %>%
  mutate_at(vars(factor_cols), funs(factor)) # makes all these factor variables


# Brief data exploration
str(data)

par(mfrow=c(1,2))
for(i in 1:ncol(data)) {
  if(class(data[,i][[1]]) == "factor") {
    var = paste0(colnames(data)[i])
    plot <- plot(data[,i], main = var)
  }
}

par(mfrow=c(1,2))
plot(density(data$claim.incurred), main = "Claim size distribution")
plot(density(filter(data, claim.incurred > 0)$claim.incurred), main = "Claim size distribution - No zer

## Collinearality check

numeric_cols <- unlist(lapply(data, is.numeric))
numeric_data <- na.omit(data[,numeric_cols])
cor_data <- cor(numeric_data)
corrplot(cor_data, method = "circle")

# remove weight, length and vehicle age
```

```r
data <- data %>% select(-c(weight, length, vehicle.age))

levels(data$claim.count)[6] <- "3"
levels(data$claim.count)[5] <- "3"
levels(data$claim.count)[4] <- "3+"

## Effects of house characteristics on size and price
par(mfrow=c(1,2))
factorVar <- c(3, 5, 6, 8, 9, 10, 17, 19)
for(i in factorVar) {
  var <- colnames(data)[i]
  plot <- ggplot(data) +
    geom_point(aes(driver.age, claim.incurred, colour= factor(get(var)))) +
    labs(color=var) +
    ggtitle(paste("Effects of", var, "on Size vs Price"))
  print(plot)
}


# Box and Whisker

for(i in factorVar) {
  var <- colnames(data)[i]
  plot <- ggplot(filter(data, claim.incurred != 0)) +
    geom_boxplot(aes(claim.incurred, colour= factor(get(var)))) +
    labs(color=var) +
    ggtitle(paste("Spread of non-zero claims across", var))
  print(plot)
}


# Data Splitting
set.seed(654321)

train.index=createDataPartition(data$claim.incurred, p = 0.7, list = FALSE)

train=data[train.index,]
train <- train %>% select(-claim.count)


test=data[-train.index,]

# Loading saved models
rpart0 <- readRDS(file = "Objects/zerosplitFinal.RData")
rpart1 <- readRDS(file = "Objects/rpartFinal.RData")
rf1 <- readRDS(file = "Objects/rfFinal.RData")
treebag0 <- readRDS(file = "Objects/treebagFinal.RData")
rf0 <- readRDS(file = "Objects/mtuneRF.RData")
rf_1 <- readRDS(file = "Objects/rf_1.RData")
for(i in 1:30) {
  j <- i*100
  model_name <- paste0("rf_", j)
  print(model_name)
  filename <- paste0("Objects/", model_name, ".RData")
  assign(model_name, readRDS(file = filename))
}
rf_5000 <- readRDS(file = "Objects/rf_5000.RData")
bagnodeData <- readRDS("Objects/bagnodeData.Rda")
random_oob <- readRDS(file = "Objects/numTreeRF.RData")


# Caret's fit control parameter - held constant for duration of report
fitControl <- trainControl(
    method = "cv",
    number = 5,
    allowParallel = TRUE) # parrallel computing saves run time

################ regression tree ################
rpart.grid <- expand.grid(cp=seq(0,0.03,0.0005))
rpart1 <- train(claim.incurred ~.,
               data = train,
               method = "rpart",
               trControl = fitControl,
               tuneGrid=rpart.grid)

plot(rpart1, main = "regression tree Complexity Parameter Tuning")

rpart.plot(rpart1$finalModel) # requires rpart.plot package
title("Rpart regression tree plot")

bestcp <- rpart1$bestTune

# Zero Split tree
rpart0 <- train(claim.incurred ~.,
               data = train,
               method = "rpart",
               trControl = fitControl,
               tuneGrid=expand.grid(cp=0.5))

################ Random Forest ################
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
# To find optimal number of trees
random_oob <- train(claim.incurred ~ ., data = train,
            method = "rf",
            trControl = fitControl,
            ntree = 250,
            nodesize = 2500)
stopCluster(cluster)
registerDoSEQ()

# number of trees plot
oob <- random_oob$finalModel$mse

# compare error rates
tibble::tibble(`Out of Bag Error`= oob,ntrees = 1:random_oob$finalModel$ntree)%>%
    gather(Metric, Error, -ntrees) %>%
    ggplot(aes(ntrees, Error)) +
    geom_line()+
    xlab("Number of trees") +
  ggtitle("RF MSE changes with number of trees")

# Function to quickly create models with different sizes
sizedRf <- function(size, train) {
  cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
  registerDoParallel(cluster)
  out <- train(claim.incurred ~ ., data = train,
            method = "rf",
            trControl = fitControl,
            ntree = 100,
            nodesize = size)
  stopCluster(cluster)
  registerDoSEQ()
  out
}

maxNodesize <- 3000
assign("rf_1", sizedRf(1, train))
for(i in 1:(3000/100)) {
  j <- i*100
  model_name <- paste0("rf_", j)
  print(model_name)
  assign(model_name, sizedRf(j, train))
}
assign("rf_5000", sizedRf(5000, train))

# Putting different node sizes into a table
bignodesizeMSE <- as.data.frame(c(1, seq(100,3000,100), 5000))
colnames(bignodesizeMSE)[1] <- "nodesize"
bignodesizeMSE$mse <- c(rep(0, nrow(bignodesizeMSE)))
bigbestnodeMSE <- Inf
bestmodel <- NULL
for(i in 1:(nrow(bignodesizeMSE)-1)) {
  j <- c(1,seq(100,3000,100))[i]
  model <- paste0("rf_", j)
  bignodesizeMSE[i, 2] = get(model)$finalModel$mse[100]
  if(bignodesizeMSE[i, 2] < bigbestnodeMSE) {
    bigbestnodesize <- j
    bigbestnodeMSE <- bignodesizeMSE[i, 2]
    bestmodel <- get(model)
  }
}
# some hardcoding
i = i+1
j <- 5000
bignodesizeMSE[i,2] <- rf_5000$finalModel$mse[100]
if(bignodesizeMSE[i, 2] < bigbestnodeMSE) {
    bigbestnodesize <- j
    bigbestnodeMSE <- bignodesizeMSE[i, 2]
```

```
}

plot(bignodesizeMSE$nodesize, bignodesizeMSE$mse, type = "l", xlab = "Nodesize", ylab = "MSE", main = "
points(bigbestnodesize, bigbestnodeMSE, col = "red", pch = 19)

# Tuning m paramter
mtry <- bestmodel$bestTune$mtry
rfGrid <-  expand.grid(.mtry=c(seq(1,51,5)))

cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
rf0 <- train(claim.incurred ~ ., data = train,
            method = "rf",
            trControl = fitControl,
            tuneGrid = rfGrid,
            ntree = 250,
            nodesize = bigbestnodesize)
stopCluster(cluster)
registerDoSEQ()

bestm <- rf0$results$mtry[which.min(rf0$results$RMSE)]
plot(rf0, main = "RF: Tuning Number of predictors used at each split")


# Best Random Forest model
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
rf1 <- train(claim.incurred ~ ., data = train,
            method = "rf",
            trControl = fitControl,
            tuneGrid = expand.grid(.mtry = bestm),
            ntree = 250,
            nodesize = bigbestnodesize)
stopCluster(cluster)
registerDoSEQ()


# VIP and PDP's for random forest
vip(rf1$finalModel) +
  ggtitle("RF Variable Importance")

RfExposure <-partial(rf1, pred.var = "exposure", grid.resolution = 20) %>%
  autoplot() +
  ggtitle("RF: PDP of Exposure")
RfYear <- partial(rf1, pred.var = "year", grid.resolution = 20) %>%
  autoplot() +
  ggtitle("RF: PDP of Year")
RfPriorClaims <- partial(rf1, pred.var = "prior.claims", grid.resolution = 20) %>%
  autoplot() +
  ggtitle("RF: PDP of Prior Claims")
RfAge <- partial(rf1, pred.var = "driver.age", grid.resolution = 20) %>%
  autoplot() +
  ggtitle("RF: PDP of driver age")
```

```r
RfExposure
RfYear
RfPriorClaims
RfAge


############### Bagging ###############
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
treebag0 <- train(claim.incurred ~.,
               data = train,
               method = "treebag",
               trControl = fitControl,
               ntree = 250,
               control = rpart.control(cp = bestcp))
stopCluster(cluster)
registerDoSEQ()

# Function to quickly model different sized bagged trees
sizedBag <- function(size, train) {
  cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
  registerDoParallel(cluster)
  out <- train(claim.incurred ~ ., data = train,
            method = "treebag",
            trControl = fitControl,
            ntree = 100,
            nodesize = size)
  stopCluster(cluster)
  registerDoSEQ()
  out
}

# Table with errors for the different nodesizes
maxNodesize <- 30000
nodeJumps <- 1000
bagnodeData <- as.data.frame(c(seq(nodeJumps,maxNodesize,nodeJumps)))
colnames(bagnodeData)[1] <- "nodesize"
bagnodeData$mse <- c(rep(0, nrow(bagnodeData)))
bagbestnodeMSE <- Inf
bestmodel <- NULL
for(i in 1:(maxNodesize/nodeJumps)) {
  j <- i*nodeJumps
  model_name <- paste0("bag_", j)
  print(model_name)
  model <- sizedBag(j, train)

  bagnodeData[i, 2] = model$results[2]$RMSE
  if(bagnodeData[i, 2] < bagbestnodeMSE) {
    bagbestnodesize <- j
    bagbestnodeMSE <- bagnodeData[i, 2]
    bestmodel <- list(model, model_name)
  }
}
```

```r
# Bagging plots
bagbestnodesize <- bagnodeData$nodesize[which.min(bagnodeData$mse)]
plot(bagnodeData$nodesize, bagnodeData$mse, type = "l", xlab = "Nodesize", ylab = "RMSE", main = "Tuning
points(bagbestnodesize, bagbestnodesize, col = "red", pch = 19)


# VIP and PDP plots for bagging
vip(treebag0) +
  ggtitle("Bagging Variable Importance")

partVehVal <- partial(treebag0, pred.var = "vehicle.value", grid.resolution = 20) %>%
  autoplot() +
  ggtitle("Bagging: PDP of Vehicle Value")
partAge <- partial(treebag0, pred.var = "driver.age", grid.resolution = 20) %>%
  autoplot() +
  ggtitle("Bagging: PDP of driver age")
partHeight <- partial(treebag0, pred.var = "height", grid.resolution = 20) %>%
  autoplot()  +
  ggtitle("Bagging: PDP of height")



# Creating testing error predictions for the tables
predTable <- test %>% select(exposure, claim.incurred)
predTable$zerosplit <- predict(rpart0, newdata = test)
predTable$rpart <- predict(rpart1, newdata = test)
predTable$rf <- predict(rf1, newdata = test)
predTable$treebag <- predict(treebag0, newdata = test)
predTable

predPrem <- predTable
predPrem$claim.incurred <- predPrem$claim.incurred/predPrem$exposure
colnames(predPrem)[2] <- "pure.prem"
predPrem$zerosplit <- predPrem$zerosplit/predPrem$exposure
predPrem$rpart <- predPrem$rpart/predPrem$exposure
predPrem$rf <- predPrem$rf/predPrem$exposure
predPrem$treebag <- predPrem$treebag/predPrem$exposure
predPrem


# Testing RMSE
test.errors <- as.data.frame(c("zero-split tree", "Rpart", "RandomForest", "Treebag"))
test.errors$RMSE <- c(caret::RMSE(predPrem$zerosplit, predPrem$pure.prem),
                      caret::RMSE(predPrem$rpart, predPrem$pure.prem),
                      caret::RMSE(predPrem$rf, predPrem$pure.prem),
                      caret::RMSE(predPrem$treebag, predPrem$pure.prem)
                      )
colnames(test.errors)[1] <- "Model"
test.errors

ggplot(test.errors) +
  geom_point(aes(x = Model, y = RMSE)) +
  ggtitle("Testing errors")


# Saving models so that the modelling process does not have to be rerun every time
# Best models
saveRDS(rpart0,file = "Objects/zerosplitFinal.RData")
saveRDS(rpart1,file = "Objects/rpartFinal.RData")
saveRDS(rf1,file = "Objects/rfFinal.RData")
saveRDS(treebag0,file = "Objects/treebagFinal.RData")

# intermediate models
saveRDS(rf0, file = "Objects/mtuneRF.RData")

saveRDS(rf_1, file = "Objects/rf_1.RData")
for(i in 1:30) {
  j <- i*100
  model_name <- paste0("rf_", j)
  print(model_name)
  filename <- paste0("Objects/", model_name, ".RData")
  saveRDS(get(model_name), file = filename)
}
saveRDS(rf_5000, file = "Objects/rf_5000.RData")

saveRDS(bagnodeData, file="Objects/bagnodeData.Rda")

saveRDS(random_oob, file = "Objects/numTreeRF.RData")
```