Analytics Project: Tree-based methods for predicting bond prices

14 September 2014

University of Cape Town

Department of Statistics

Ben Walwyn

WLWBEN001

Predicting the Trade Price of a Bond

The aim of this report is to determine the best method for predicting the price of US corporate bonds based on the information of previously traded bonds using regression trees and ensemble methods for improving fit. First, the bond data is explored. Secondly, regression trees, bootstrap aggregation, random forests and boosting models are motivated, validated and interpreted. Finally, the results are compared with a conventional autoregressive model of trade prices.

1. Exploratory Analysis

The information for 3277 rows of trades is stored in *bond25.csv*. The trade price of the bond in a trade is the response variable and there are 57 variables recording the relevant current and past information for any trade.

Variable	Type	Description
Trade price	Continuous	The price at which the trade occurred on R100 nominal.
Current coupon	Continuous	The coupon of the bond at the time of the trade
Time till maturity	Continuous	The number of years until the bond matures at the time of the trade
Is callable	Binary	A binary value indicating whether or not the bond is callable by the issuer
Reporting delay	Continuous	The number of seconds after the trade occurred that it was reported
Trade size	Continuous	The notional amount of the trade
Trade type	Categorical	2=customer sell, 3=customer buy, 4=trade between dealers. We would expect customers to get worse prices on average than dealers
Curve based price	Continuous	A fair price estimate based on implied hazard and funding curves of the issuer of the bond
Received time difference for last{1:10}	Continuous	The time difference between the trade and that of the previous {1:10}
Trade price last{1:10}	Continuous	The trade price of the last {1:10} trades
Trade size last{1:10}	Continuous	The notional amount of the last {1:10} trades
Trade type last{1:10}	Categorical	The trade type of the last {1:10} trades
Curve based price last{1:10}	Continuous	The curve based price of the last {1:10} trades

Explanatory variables

The variables based on previous trades are obviously dependant on previous values. Most significantly, the trade price is heavily correlated with the previous trade prices.

Table 1: Correlation of trade prices

trade_price	1
trade_price_last1	0.99179
trade_price_last2	0.98807
trade_price_last3	0.98563
trade_price_last4	0.98433
trade_price_last5	0.98386
trade_price_last6	0.98308
trade_price_last7	0.98196
trade_price_last8	0.98069
trade_price_last9	0.97998
trade_price_last10	0.97882

Table 1 lists the correlations between the previous price variables and the current trade price. Their strong dependence can be clearly seen, even up to the 10 last trades. Trade price is also related to curve based price. Intuitively, the actual trade price should at least resemble the fair price even if it trades at a discount or premium.

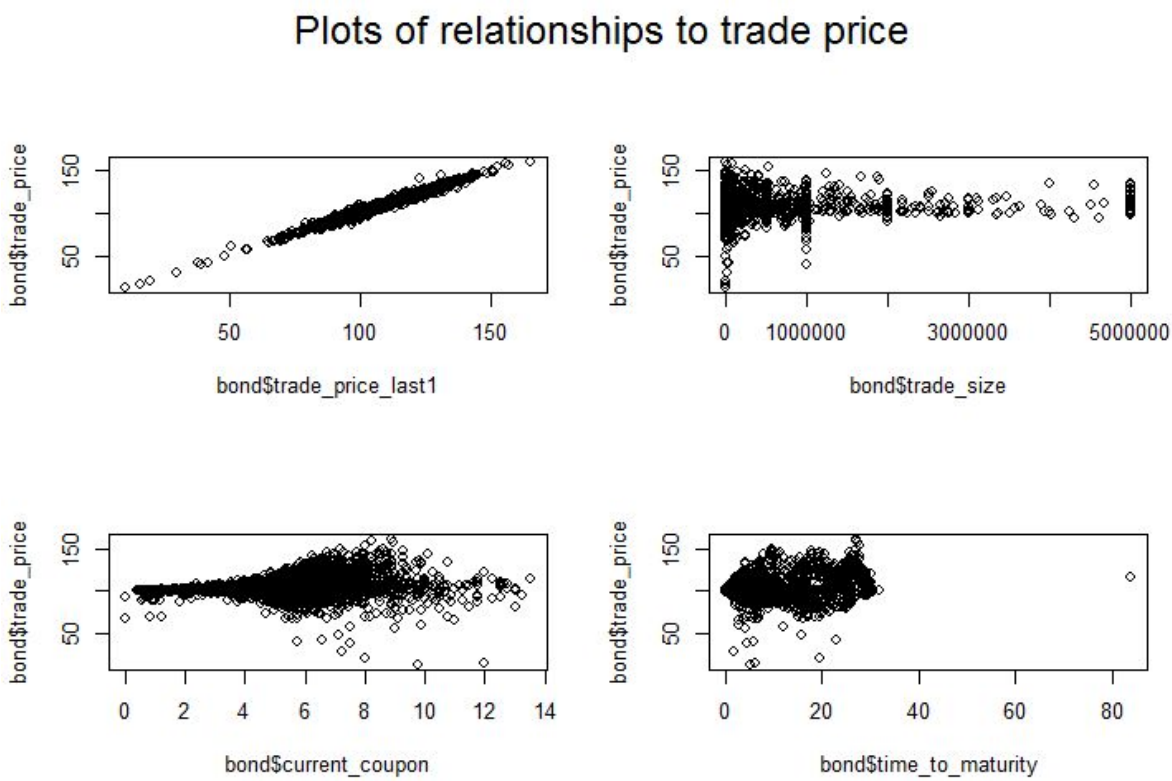


Figure 1: Scatterplots of relationships to response

Figure 1 depicts the relationships between four explanatory and the trade price. The last trade price plot confirms that previous trade prices are good predictors. There is a higher variability of prices at smaller trade sizes, but this is expected due to the higher volume of trades at smaller sizes. Bonds with small current coupons tend to trade around nominal value, with higher price variability at higher coupon rates. There is no apparent relationship of price and time to maturity - 0 to 30 years. There is an outlying bond with a time to maturity of over 80 years. Clearly any model will be heavily influenced by the previous trade price variables.

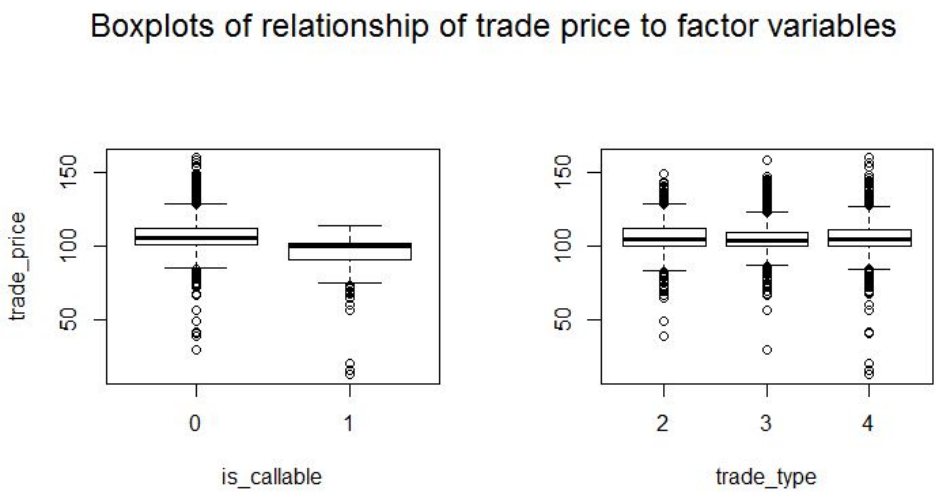


Figure 2: Shows the difference in price between factor levels of the categorical variables

A callable bond appears to have a lower mean price, although the data is also subject to higher variability with only 581 observations. The trade type does not seem to impact majorly on the trade price. Boxplots did not show that the trade type of previous trades has any affect either.

Response

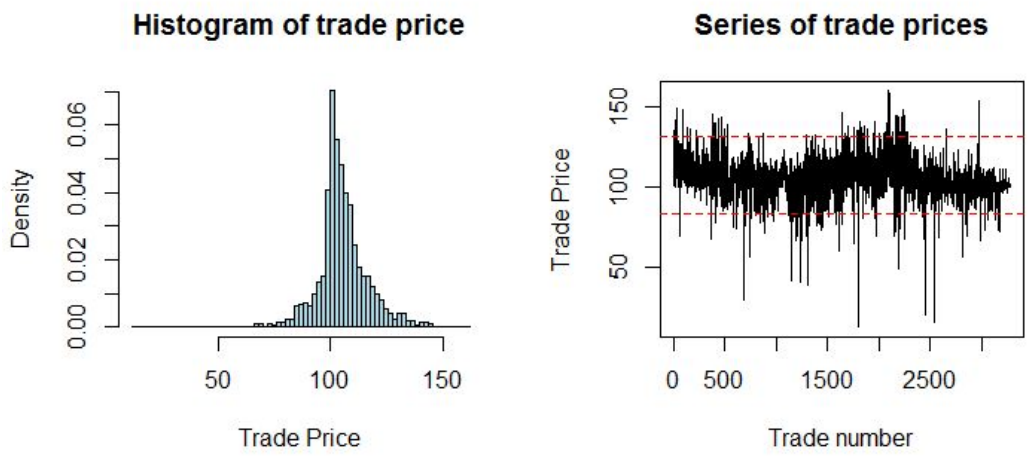


Figure 3: a.) Histogram of trade prices shows slight skewness to the left, with more prices above nominal than less. b.) Trade prices also exhibit some cyclical behaviour but appear to be stationary.

Median trade price of 104.2 is above nominal value meaning the majority of bonds trade above par. There are a number of factors that affect bond prices but one explanation for an above par trend might be that the inherent higher risk of corporate stock drives higher prices.

The series of trades in this data plotted in figure 3.b. exhibits some stationary behaviour. There are periods of extreme volatility, with very low prices, and a cycle period of about 2000 trades. Without further information about the time period and economic environment of these trades, little can be said about the cycle.

The exploratory analysis reveals that the next trade value is likely to be similar to the previous one and the curve based price. The linear relationship between these variables, therefore, needs to be captured by a model predicting the bond price of the next trade.

2. Tree-based methods

Various trees and ensembles are fitted to the data and cross-validation or other equivalent error measures are used to test the predictions. Ultimately, the tree or ensemble with the lowest error is chosen.

2.1. Regression trees

Regression trees use binary splitting algorithms to identify a partition space that minimises the residual sum of square errors. Regions of the features space are separated by a set of splitting rules which define a set of terminal nodes and predict a price for each region. If the relationships, however, cannot be partitioned exactly by binary splitting on explanatory variables then the tree

suffers from under and over-estimation errors. Figure 4 shows the advantage and disadvantage the splitting algorithm can have. This disadvantage may affect the regression in this case, since it is likely that previous trade prices will be highly influential, and there is strong linear relationship.

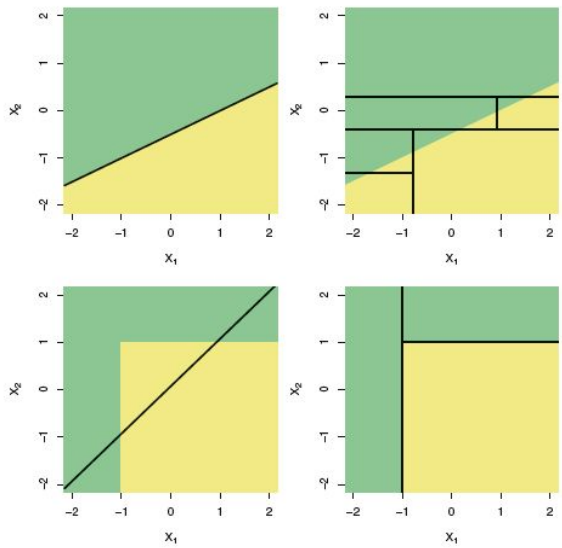


Figure 4: Logistic regression on the left show neat predictions for linear relationships, but inadequate linear fit for other binary splits. Tree models perform poorly for linear data on the right but well in binary splits.

Table 2: Settings for growing regression tree

Argument		Input
nobs	The number of observations in the training set.	3277
minsize	Minimum size of an internal node	10 (default)
mincut	Minimum size of a terminal node	5 (default)
mindev	Minimum deviance reduction for a split to be implemented	0 and 0.004

All variables are considered in sequence with a split made at each possible point. The algorithm chooses the split with the largest reduction in residuals sum of squares (RSS). Splitting is done repetitively in a branch until:

- An internal node has less than 10 observations or,
- A terminal node has less than 5 or,
- The reduction in RSS is less than a 0.004, or 0 for a full tree. I.e. no reduction required.

The tree with a minimum reduction of 0.004 in RSS has 11 terminal nodes. For a continuous response, this might be too few. A more extensive tree can be grown by specifying no minimum reduction in RSS required to implement a node split. Cost-complexity pruning is then used to trim back the tree.

The fully grown tree has 512 terminal nodes. A tree with this many nodes relative to the size of the training set and few observations in each terminal node over fits the training data. This means that the tree can predict any value within the training set with high accuracy. The algorithm has

learned the idiosyncrasies and underlying noise of the data and can overcome it. However, when introducing a different test set, the model is unlikely to perform well.

Although there is no test data partitioned in advance, to use after the model has learned on the training set, cross-validation of the tree can identify over fitting and determine the optimum tree size. The cross-validation errors for the bond regression tree up to 20 terminal nodes can be seen in figure 5. Cross-validation estimates test error by partitioning the data into folds. For each fold, a tree is fitted on the out-of-fold data and squared errors from predictions on observations in the fold are stored.

Cost-complexity pruning penalises the RSS according to the size of the tree, similar to the Akaike penalty on number of covariates in a general linear model. The penalty size depends on a non-negative tuning parameter α.

Equation 1: Cost-complexity pruning: $R_\alpha(T) =$ penalised RSS. R(T) is RSS and T ̃is the number of terminal nodes.

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}|\,,$$

The values for α are plotted on the top axis in figure 5 and the selected value is chosen such that the pruned tree has the lowest error. The final tree is reduced to 9 terminal nodes, displayed in figure 6.

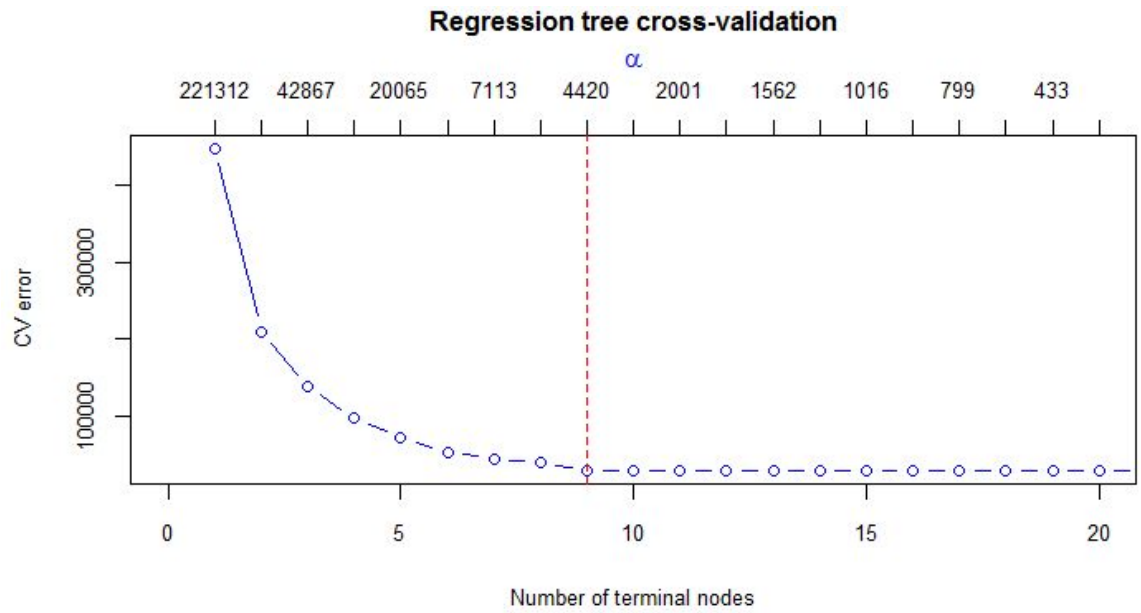


Figure 5: The CV error using 100 folds for a regression tree, plotted against terminal nodes. Although full model has 512 nodes, the CV error bottoms out at 9 nodes (α = 4420.01).

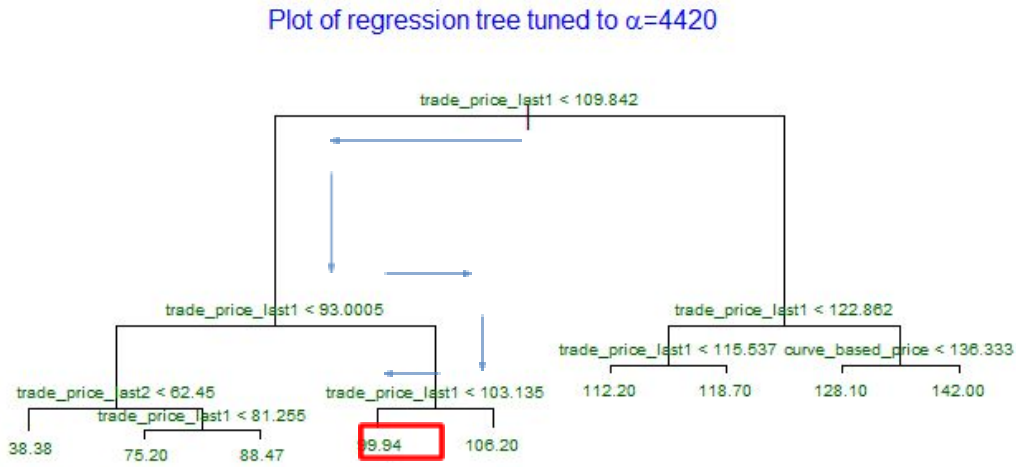


Figure 6: Tree diagram for selected model. Each split produces two nodes. For X<x proceed down left branch, for X>x proceed down right branch.

Figure 6 is a graphical representation of the pruned decision tree. The first split is made where the last trade price is less than or greater than 109.8. The second split is also based on previous price, at the points 93.0 and 122.9. For any set of past trade information, the decision tree can be followed from the top to the terminal node prediction. For example a previous trade price of 100, will have follow the path depicted with blue arrows in figure 6, to a predicted trade price of 99.94 in the 4th terminal node bordered in red.

Single tree predictions, however, have high variability. This means they will produce different decisions for different datasets from the same population. By aggregating many trees, using boosted aggregation, random forests and boosting, predictions can be improved.

2.2. Bootstrap Aggregation

Bootstrap aggregation or bagging fits trees to bootstrapped samples of the data. For each tree, a number of observations are omitted, called the out-of-bag (OOB) sample. Tree are grown deep and not pruned so that each has minimal bias but high variability. A large number of complex trees can aggregate predictions on OOB data that consequently have lower variance. Square errors are calculated on the average OOB predictions. Importantly for validity of error estimates, the predictions are made only on prices that were not used to fit the tree itself.

The plot of bag OOB mean square error (MSE) is shown in figure 8 along with the random forest results. The mean square error for 500 trees is 2.16, although the test error does not improve beyond 100 trees.

When there is a strong predictor present in the data, most or all trees fitted in bagging will have the same top split. Therefore, each tree will look similar, and predictions will be correlated. In this case, last trade price has an overwhelming effect on the decision tree; all trees have a root split based on this variable.

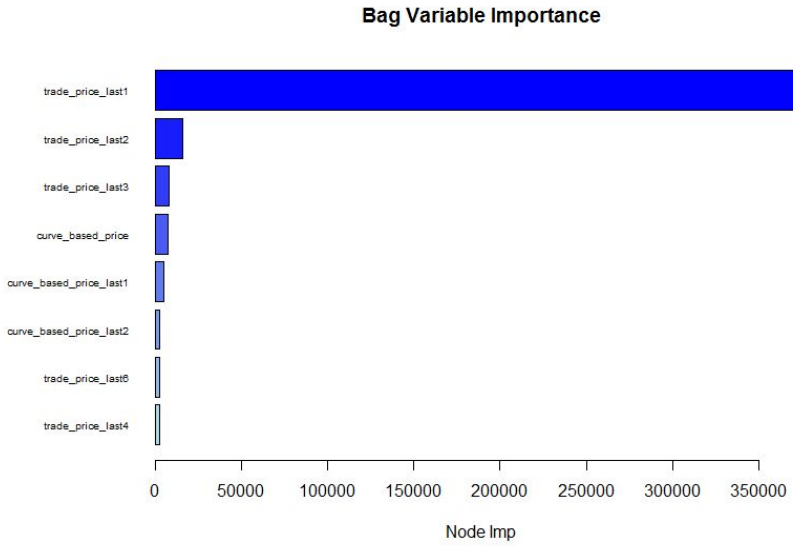


Figure 7: Variable importance plot for bag model. Last trade price dominates, contributing the most to reductions in RSS. The dependency on this variable means most trees will resemble each other. Hence, the predictions are correlated and the bagging improvement is limited.

Since 500 trees were used to make predictions, there is no single tree diagram to interpret for bagging showing the variable influence and splits; the process improves test accuracy at the expense of interpretability. However, the average overall reduction in RSS due to splits for a given variable over all trees can measure that variable's importance. Plotting the results of the top 8 predictors in figure 7 shows their relative importance. Other effects are completely dominated by the last trade price.

The partial plots for last trade price and curve price are shown in figure 1 of appendix A. A partial plot takes the average prediction over all other variables for every value of a specific variable and plots the relationship. This effectively shows how the predictor influences the response. The extent of the influence can be seen on the y-axis in each plot. Last trade price can account for increased from below 80 to above 120, while the scale for curve based price is less than 1. Methods to reduce this dependence on one single predictor should be used to improve the model.

2.3. Random Forests

Random forests forces each split during to consider a subset *m* of the variables only. By sampling variables for split candidates, instead of considering them all, random forests can improve bagging models by reducing the overall dependence on a single predictor. This *decorrelates* the trees. Predictions will therefore be less correlated and their average will have a lower variance. Bagging is essentially a random forest where $m = p$, the number of predictors.

Table 3: Random forest settings

Argument		Input
ntree	The number of samples to bootstrap and trees to grow	500
mtry	The number of predictors to randomise at each split	5 20 (default)

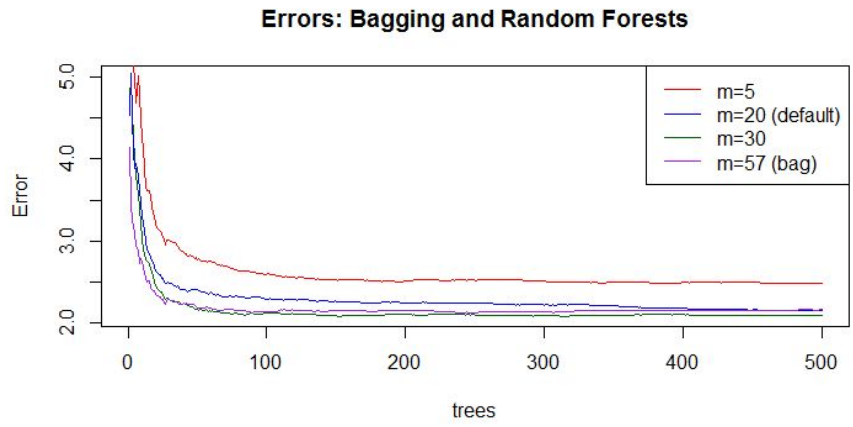


Figure 8: Random forest error for different numbers of sampled predictors.

Bagging minimises error the fastest, utilising the high predictive power of last trade price, but two of the three random forests eventually improve the error.

Table 4: OOB MSE for different tuning parameters of random forests. Sampling 30 predictors, 10 more than the default, has the lowest error. It takes into account the strong predictive power of previous trade price but also decorrelates the trees, allowing for the model to take into account the other variables.

m=5	m=19 (default)	m=30	m=57 (bagging)
2.480032	2.154765	2.100331	2.16344

There is a slight improvement in OOB error for the default *m,* and a larger improvement for *m* = 30 over the bagging model.

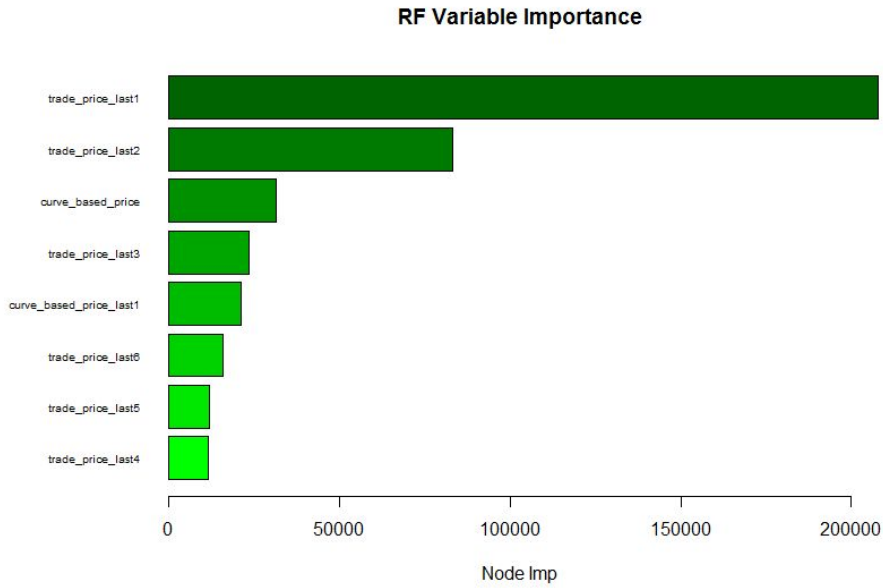
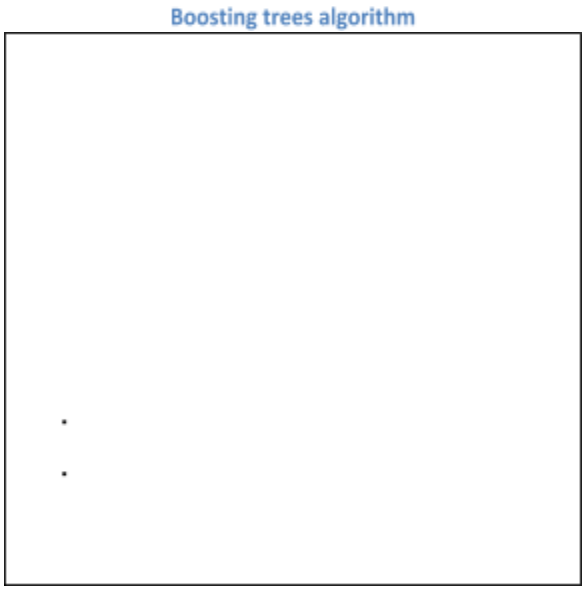


Figure 9: The important variables for the best random forest model include a much higher important of all variables other than last trade price compared to bagging. *m* = 30.

Figure 9 shows there is now a significant importance on second last trade price and the curve based price. All further previous curve prices and trade prices are more important too. This confirms that the random forest reduces reliance on a strong predictor and improves the model with increased reliance on other predictors.

2.4.Boosting

Whereas bagging and random forest fit trees separately on different bootstrap samples, boosting fits tree sequentially. Each tree modifies data to which the following tree is fitted and accounts for variation in *y* not previously accounted for. . Given the current model, the next tree is fitted on the residuals. Boosting, therefore, *learns slowly* over a sequence of trees.



The shrinkage parameter controls how fast the algorithm learns and the interaction depth is just the number of splits, d, per tree. The number of splits in each tree determines the complexity. The cross-validation error for boosting under different combinations of interactions and shrinkages for 1000 trees is plotted in figure 10.

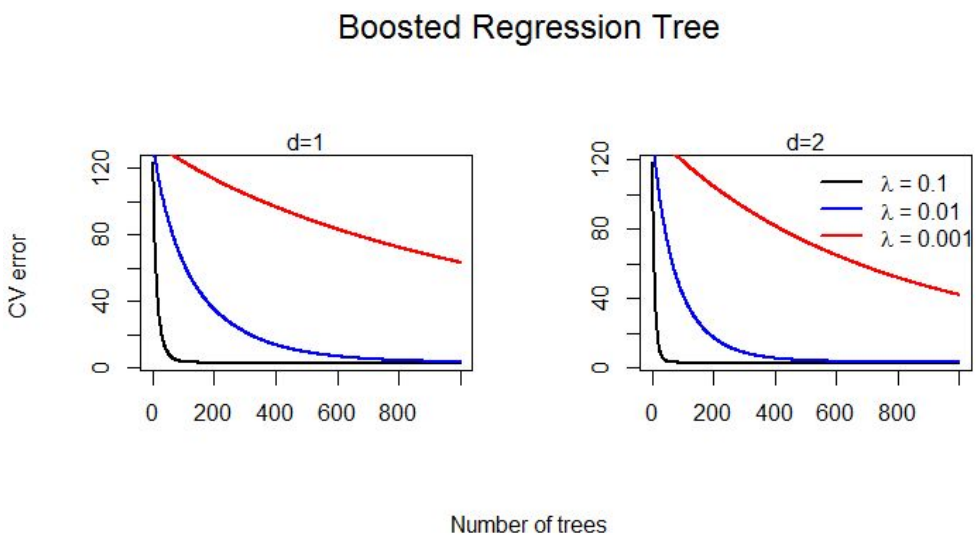


Figure 10: Boosted tree test error for different combinations of tuning parameters *d* and λ over 1000 trees.

For smaller shrinkage parameters, many thousands of trees need to be fitted to determine lowest cross-validation. For the sake of computational time, the optimum combination for learning trade prices is d=2 and λ=0.1, a relatively quick rate. The default number of folds for cross-validation, 10, was used and fitting was done in parallel.

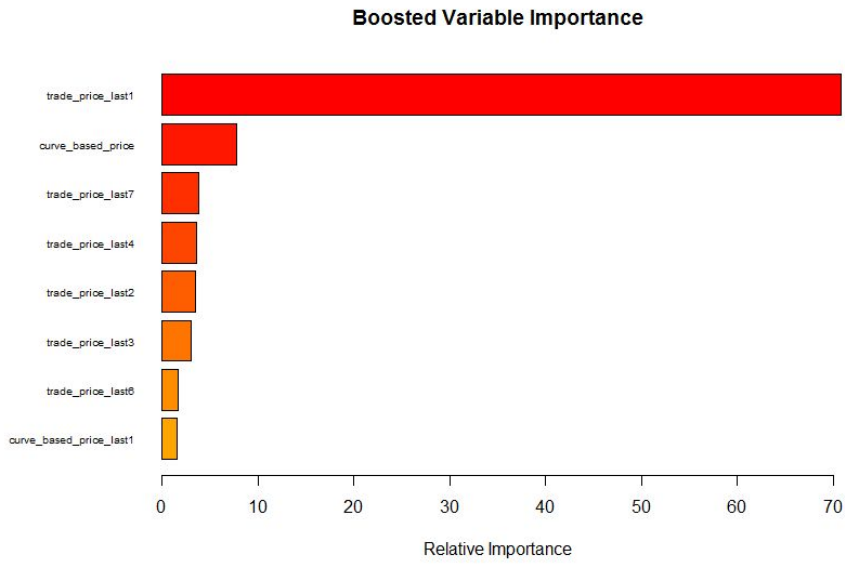


Figure 11: Variable importance of boosted model.

Unlike bagging and random forests, boosting can over fit the data. At some point, the next tree fitted is learning noise. It therefore overemphasises very noisy examples, evident when testing on other data, like in cross-validation. Eventually a boosted model test error will start to increase and there is indeed a slight increasing slope for boosting after 220 trees, shown in figure 12.

2.5.Model comparison and selection

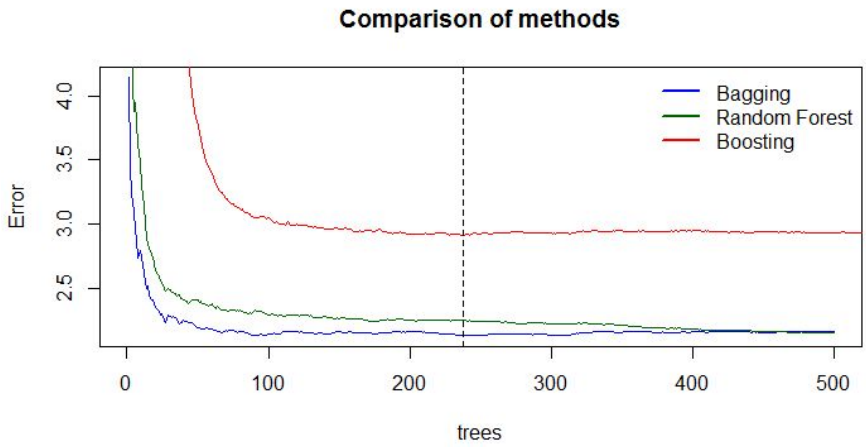


Figure 12: Test error as a function of number of trees for different ensemble methods. Best random forest model is used.

Boosting most likely suffers from the strength of the last trade price predictor. There is likely to be a long sequence of trees initially that select it as the splitting variable. This is shown by its large relative importance. As the first few trees contribute the most to reduction in RSS, the other variables are then more likely to be fitted to noise thus reducing their usefulness.

Random forests do better because it takes into account the predictive power of other predictors besides last trade price. The randomness in variable consideration does mean that it takes more trees until the random forest OOB MSE flattens out. Therefore, bagging performs better initially for smaller numbers of trees, but eventually random forests surpass bagging and improve on the model.

The partial plots in appendix A confirm a less consistent relationship with the response for the predictors under boosting compared with bagging and random forests. The random forest model shows a slightly smoother relationship for curve based price than bagging, as well as a slightly larger influence.

Although all the trees manage to predict trade prices fairly well, as shown in appendix A, the random forest model performs better than the other models when considering the OOB error, variable importance and partial plots. Therefore, the random forest randomising 30 predictors at each node is the best model.

3. Auto-regressive model

The strong correlation with previous trade prices shown in table 1 indicates an autoregressive model would be a suitable model.

The form of the model equation: $y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \ldots + \beta_p y_{t-p} + \varepsilon_t$

The best order *p* for the model is chosen by fitting the trade price to each number of previous trade prices. Cross-validation is performed on each of these models in parallel and the results are plotted in figure 13.

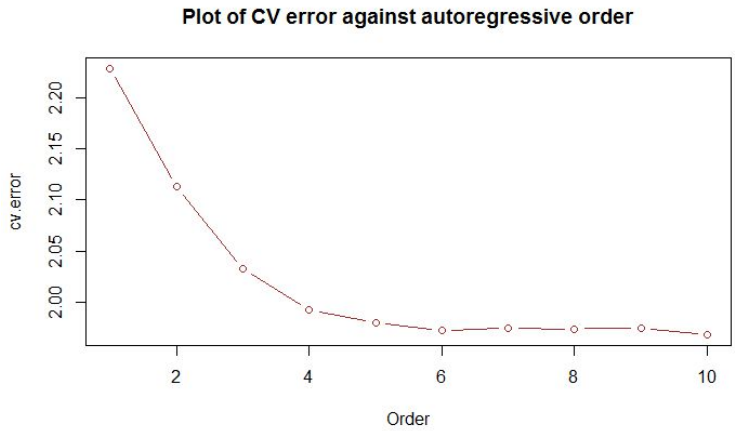


Figure 13: Cross-validation error for each autoregressive model plotted against its corresponding order. CV error is minimised at order 6.

An autoregressive model of order 6 is adequate to minimise cross-validation; further variables only complicates the model without reducing test error. Based on equivalent measures of test error shown in table 5, the autoregressive model outperforms all the tree-based methods but only marginally improves on the best random forest.

Table 5: Comparison of errors for all methods

Regression tree	RF: *m*=5	RF: *m*=20	RF: *m*=30	Bagging	Boosting	AR model: *p*=6
	2.480032	2.154765	2.100331	2.16344	3.225694	1.972446

Note, due to the linear relationships and strong correlations between trade prices, the tree-based methods were not expected to outperform other regression techniques.

The auto-regressive linear model based on history of trade prices should be used to estimate the next trade price.

Appendix A

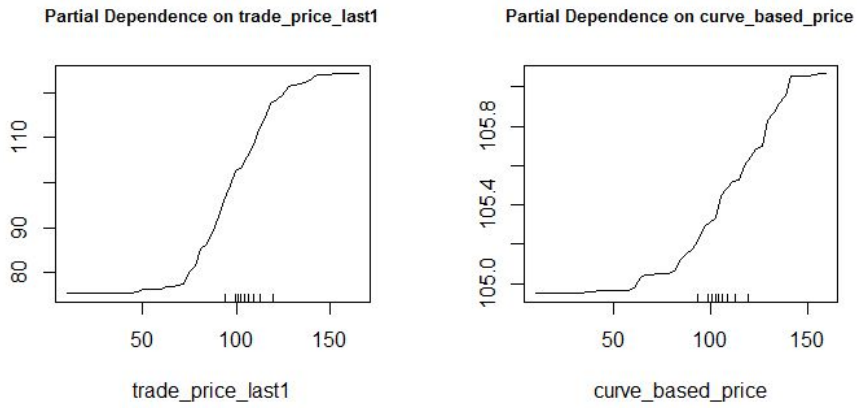


Figure1: Bagging 500 trees. Smooth relationship for last trade price and a slightly more jagged relationship for curve based price.

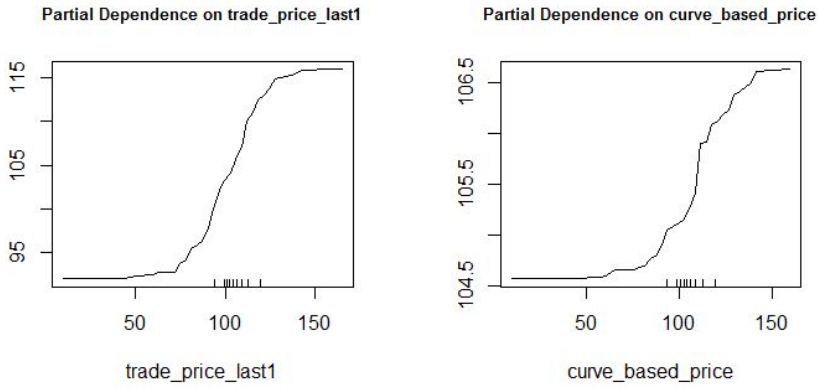


Figure 2: Random forest m=30, 500 trees. Smooth relationship.

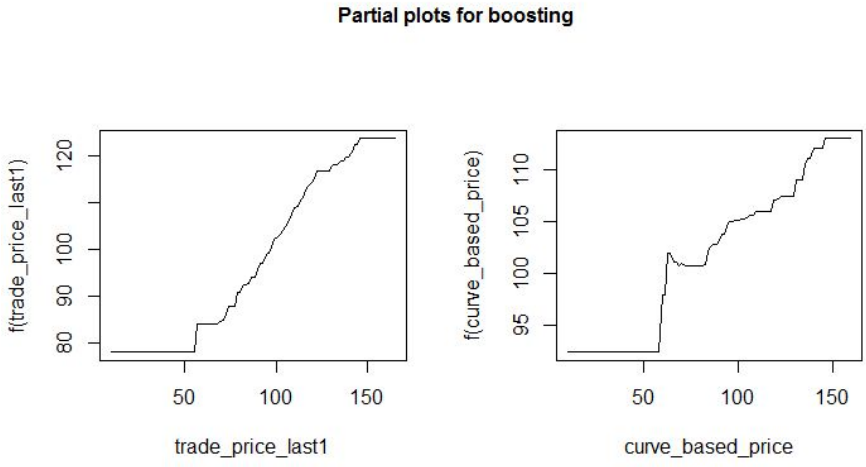


Figure 3: Boosting d=2, λ =0.1. Inconsistent relationship of trade price with ach predictor. Last trade price is slightly smoother than curve based.

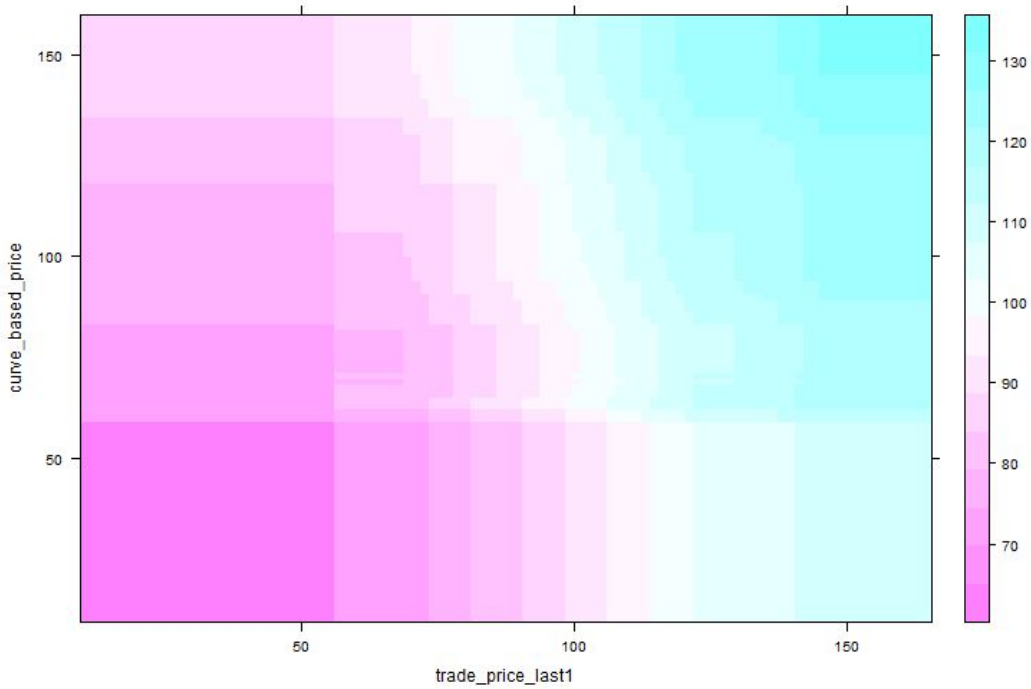


Figure 4: Partial plot for both last and curve price. The response, trade price, is depicted by colour. The linear relationship is apparent from the upward gradient in colour change.

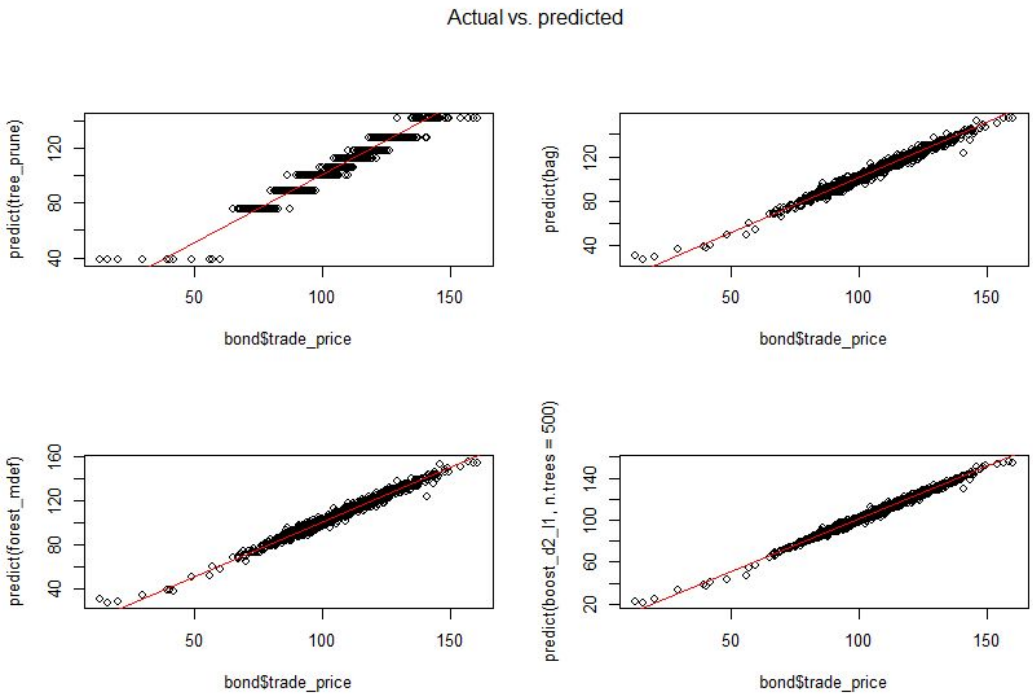


Figure 5: Actual against predicted. Plots are all centred on the 45 degree line, indicating they predict adequately.

Appendix B: R code

Part A

```
#Ben Walwyn
#Analytics project R
#04/09/14

#----------------------Part 1-------------------------#
setwd("F:/Analytics")
install.packages("tree")
install.packages("MASS")
install.packages("randomForest")
install.packages("doParallel")
install.packages("foreach")
install.packages("ipred")
install.packages("gbm")
install.packages("DAAG")
library(DAAG)
library(gbm)
library(tree)
  library(MASS)
library(randomForest)
library(doParallel)
library(foreach)
library(ipred)
library(stats)
options(scipen=999)

bond <- read.csv("bond25.csv",header=T)
```

```
newdata<-data.frame(bond$trade_price,bond$trade_price_last1,bond$trade_price_
last2,

bond$trade_price_last3,bond$trade_price_last4,bond$trade_price_last5,

bond$trade_price_last6,bond$trade_price_last7,bond$trade_price_last8,
                    bond$trade_price_last9,bond$trade_price_last10)


#--- Exploratory analysis ---#

par(mfrow=c(1,2))
#variables
median(bond$trade_price)
length(bond)
#histogram of response
hist(bond$trade_price,breaks=75,col="lightblue",freq=F,xlab='Trade
Price',main='Histogram of trade price')
#trade series
plot(bond$trade_price,type='l',xlab='Trade number',ylab='Trade
Price',main='Series of trade prices',col='black')
abline(a=quantile(bond$trade_price,0.025),b=0,col='red',lty=2)
abline(a=quantile(bond$trade_price,0.975),b=0,col='red',lty=2)
#correlation with prevuious trades
cor(newdata,bond$trade_price)
#plotting explanatory variables
par(mfrow=c(2,2))
par("oma"=c(0,0,2,0))
plot(bond$trade_price_last1,bond$trade_price)#strong relationship
plot(bond$trade_size,bond$trade_price)
plot(bond$current_coupon,bond$trade_price)#spreads out
plot(bond$time_to_maturity,bond$trade_price)#an extremely long term bond
mtext("Plots of relationships to trade price",outer=T,side=3,cex=1.5)
par(mfrow=c(1,1))
par("oma"=c(0,0,0,0))
#boxplots of relationships
par(mfrow=c(1,2))
par("oma"=c(0,0,2,0))
boxplot(bond$trade_price~as.factor(bond$is_callable),xlab='is_callable',ylab=
'trade_price')
boxplot(bond$trade_price~as.factor(bond$trade_type_last10),xlab='trade_type')
mtext('Boxplots of relationship of trade price to factor
variables',outer=T,side=3,cex=1.5)
par(mfrow=c(1,1))
par("oma"=c(0,0,0,0))

### (1) Regression tree

full_tree <- tree(bond$trade_price~.,
data=bond,control=tree.control(nrow(bond),mindev=0))
summary(full_tree)
plot(full_tree)

stop_tree <- tree(bond$trade_price~., data=bond,
control=tree.control(nrow(bond),mindev=0.004)) #defaults
summary(stop_tree)
plot(stop_tree)
text(stop_tree,cex=0.6)
```

```r
#--- Cross-val on full tree
cross_val<-cv.tree(full_tree,K=100)
#plot the cross validation error against no. of terminal nodes and the
tuning parameter
plot(cross_val$size,round(cross_val$dev),type='b',xlab="Number of terminal
nodes",ylab="CV error",xlim=c(0,20),cex.lab=0.8,col="blue",cex.axis=0.8)
alpha<-round(cross_val$k)
axis(3, at=cross_val$size, lab=alpha, cex.axis=0.8)
mtext(expression(alpha), side=3,line=2, col="blue")
mtext(expression(bold("Regression tree cross-validation")),3,3,)

#optimum number of terminal nodes, minimzing overfitting
abline(v=9,col="red",lty=2)
#the selected value for the tuning parameter
best_alpha<-cross_val$k[cross_val$size==9]

#--- Tree Pruning
tree_prune <- prune.tree(full_tree, k=best_alpha+0.1)
#Interpretation
plot(tree_prune)
text(tree_prune, cex=0.6,col="darkgreen")
mtext(expression(paste("Plot of regression tree tuned to ", alpha,
"=4420")),3,line=2,col="blue")
summary(tree_prune)


### (2) Bagging

#default #trees is 500

B <- 500 #bag samples and trees
core <- 5 #cores to be used for parallel

#Perform bagging in parallel
cl <- makeCluster(core)
registerDoParallel(cl)
getDoParWorkers()

t_bagpar<-system.time({ bag_dopar<-foreach(i=1:core, .combine="combine",
.packages="randomForest") %dopar% {
    randomForest(bond$trade_price~., data=bond, mtry=57, ntree=B/core,
importance=TRUE)
  }
})

mse_parallel <- mean((bag_dopar$predicted-bond$trade_price)^2)

stopCluster(cl)

#Bagging without parallel
t_bag<-system.time({ bag<-randomForest(bond$trade_price~., data=bond,
mtry=57, ntree=500, importance=TRUE, na.action=na.exclude)})


### (3) Random forests

#default number of variables randomized in each permutation (deafult=19)
```

```
t_for20<-system.time({ forest_mdef<-randomForest(bond$trade_price~.,
data=bond, ntree=500, importance=TRUE, na.action=na.exclude)})
t_for5<-system.time({ forest_m5<-randomForest(bond$trade_price~., data=bond,
mtry=5, ntree=500, importance=TRUE, na.action=na.exclude)})
t_for30<-system.time({ forest_m30<-randomForest(bond$trade_price~.,
data=bond, mtry=30, ntree=500, importance=TRUE, na.action=na.exclude)})

#parallel
cl2 <- makeCluster(core)
registerDoParallel(cl2)

t_for30par<-system.time({ forest_dopar<-foreach(i=1:core, .combine="combine",
.packages="randomForest") %dopar% {
  randomForest(bond$trade_price~., data=bond,mtry=30, ntree=B/core,
importance=TRUE)
}
})

stopCluster(cl2)

attributes(forest_dopar)

### (4) Boosting

t_boostd1<-system.time({boost_d1_l1=gbm(bond$trade_price~., data=bond,
distribution='gaussian',n.trees=1000, interaction.depth=1, shrinkage=0.1,
cv.folds=10, bag.fraction=1, n.minobsinnode=10, n.core=7)})
system.time({boost_d1_l2=gbm(bond$trade_price~., data=bond,
distribution='gaussian',n.trees=1000, interaction.depth=1, shrinkage=0.01,
cv.folds=10, bag.fraction=1, n.minobsinnode=10, n.core=7)})
system.time({boost_d1_l3=gbm(bond$trade_price~., data=bond,
distribution='gaussian',n.trees=1000, interaction.depth=1, shrinkage=0.001,
cv.folds=10, bag.fraction=1, n.minobsinnode=10, n.core=7)})
t_boostd2<-system.time({boost_d2_l1=gbm(bond$trade_price~., data=bond,
distribution='gaussian',n.trees=1000, interaction.depth=2, shrinkage=0.1,
cv.folds=10, bag.fraction=1, n.minobsinnode=10, n.core=7)})
system.time({boost_d2_l2=gbm(bond$trade_price~., data=bond,
distribution='gaussian',n.trees=1000, interaction.depth=2, shrinkage=0.01,
cv.folds=10, bag.fraction=1, n.minobsinnode=10, n.core=7)})
system.time({boost_d2_l3=gbm(bond$trade_price~., data=bond,
distribution='gaussian',n.trees=1000, interaction.depth=2, shrinkage=0.001,
cv.folds=10, bag.fraction=1, n.minobsinnode=10, n.core=7)})


#--- Tuning parameter selection ---#

#pruning -> alpha selected best using graph

#bag: how many trees to fit

#forests
#comparison plot for m
plot(forest_m30,col="darkgreen",main="Errors: Bagging and Random Forests")
lines(forest_mdef$mse,col="blue")
lines(forest_m5$mse,col='red')
lines(bag$mse,col='purple')
legend("topright",c("m=5","m=20 (default)","m=30",'m=57
(bag)'),lty=c(1,1,1,1),col=c("red","blue","darkgreen",'purple'))
```

```
forest_mdef$mse[500] #fastest reduction in error
forest_m5$mse[500]
forest_m30$mse[500] #best but slower
bag$mse[500]

#boost
min(boost_d2_l1$cv.error)
par(mfrow=c(1,2))
par("oma"=c(1,1,2,0))
plot(boost_d1_l1$cv.error, type="l", ylab="CV error",ann=FALSE,lwd=2)
lines(boost_d1_l2$cv.error,col="blue",lwd=2)
lines(boost_d1_l3$cv.error,col="red",lwd=2)
abline(cve,0,lty=2,col="darkgreen")
mtext("d=1", side=3,line=0)

plot(boost_d2_l1$cv.error, type="l",ann=FALSE,lwd=2)
abline(cve,0,lty=2,col="darkgreen")
text(800, cve+5, "regression tree")
lines(boost_d2_l2$cv.error,col="blue",lwd=2)
lines(boost_d2_l3$cv.error,col="red",lwd=2)
legend('topright',
       legend=c(expression(lambda~"= 0.1"),expression(lambda~"=
0.01"),expression(lambda~"= 0.001")),
       col=c("black","blue", "red"),
       lwd=2,
       bty='n')
mtext("d=2", side=3,line=0)
mtext("Boosted Regression Tree", outer = TRUE,side=3, cex = 1.5)
mtext("Number of trees", outer = TRUE,side=1, cex = 1)
mtext("CV error", outer = TRUE,side=2, cex = 1)
par("oma"=c(0,0,0,0))
par(mfrow=c(1,1))

#at the endpoints there d=2 and lambda =0.1 is the best.
#Possibly more trees need to be fitted

#--- Comparison of methods ---#



#OOB mse plot
#single tree(dashed line on cv error),bagging,random forest,boost

plot(bag,col="blue",main="Comparison of methods")
lines(forest_mdef$mse, col="darkgreen")
lines(boost_d2_l1$cv.error,col="red")
abline(v=which(boost_d2_l1$cv.error==min(boost_d2_l1$cv.error)),col="black",l
ty=2)
legend("topright", legend=c("Bagging", "Random Forest","Boosting"),
col=c("blue", "darkgreen","red"), lwd=2,bty='n')

#Actual vs. predicted
#names
par(mfrow=c(2,2))
par("oma"=c(0,0,2,0))
plot(bond$trade_price,predict(tree_prune))
abline(1,1,col='red')
```

```
plot(bond$trade_price,predict(bag))
abline(1,1,col='red')
plot(bond$trade_price,predict(forest_mdef))
abline(1,1,col='red')
plot(bond$trade_price,predict(boost_d2_l1,n.trees=500))
abline(1,1,col='red')
mtext("Actual vs. predicted",side=3,outer=T)
par("oma"=c(0,0,0,0))
par(mfrow=c(1,1))
#boosted prediction expected to get close on the whole data set as the
following tree is grown on the residuals, thereby reducing them.

#--- Interpretation ---#

#Variable Importance and partial plots

#bag
par("oma"=c(0,4,0,0))
colblue <- colorRampPalette(c("lightblue", "blue"))
barplot(sort(bag$importance[,2])[50:57],offset=200,cex.names=0.6,horiz=T,col=
colblue(8),las=1,xpd=F,main="Bag Variable Importance",xlab="Node Imp")

par("oma"=c(0,0,2,0))
par(mfrow=c(1,2))
partialPlot(bag,bond,trade_price_last1,cex.main=0.8)
partialPlot(bag,bond,curve_based_price,cex.main=0.8)

partialPlot(bag,bond,trade_price_last10)

#random forest
colgreen <- colorRampPalette(c("green", "darkgreen"))
barplot(sort(forest_m30$importance[,2])[50:57],offset=200,cex.names=0.6,horiz
=T,col=colgreen(8),las=1,xpd=F,main="RF Variable Importance",xlab="Node Imp")


partialPlot(forest_m30,bond,trade_price_last1,cex.main=0.8)
partialPlot(forest_m30,bond,curve_based_price,cex.main=0.8)
partialPlot(forest_m30,bond,trade_price_last10)

#boosting
colred <- colorRampPalette(c("orange", "red"))
barplot(sort(summary(boost_d2_l1)[1:8,2]),names.arg=summary(boost_d2_l1)[8:1,
1],cex.names=0.6,las=1,horiz=T,col=colred(8),main="Boosted Variable
Importance",xlab="Relative Importance")


plot(boost_d2_l1,i=c("trade_price_last1",'curve_based_price'))
plot(boost_d2_l1,i=c("trade_price_last1"))
plot(boost_d2_l1,i=c("curve_based_price"))
mtext(expression(bold('Partial plots for boosting')),side=3,outer=T)
par("oma"=c(0,0,0,0))
par(mfrow=c(1,1))
#most important variable are k asat traded prices and the j last curved
prices, indicates an autoregressive model is suitable

#--- Parallel Computing and Time components ---#

#t's
```

```
#--- (b) Linear Model ---#


cl <- makeCluster(7)
registerDoParallel(cl)

ms<-foreach(i=1:10,.combine='c', .packages="DAAG") %dopar% {
  fit <- lm(bond.trade_price~.,data=newdata[,1:(i+1)])
  cv<-cv.lm(fit, df=newdata[,1:(i+1)],printit=T,plotit=F,m=10)
  attributes(cv)$ms
}

stopCluster(cl)

plot(seq(1:10),unlist(ms),type='b',main="Plot of CV error against
autoregressive order",xlab='Order',ylab='cv.error',col='brown')
unlist(ms)
```