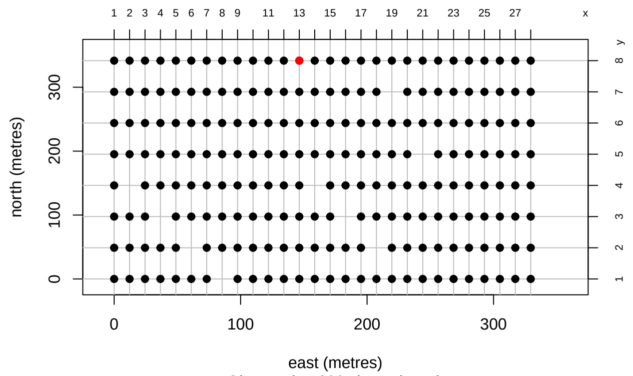
STAT6020_PredictiveAnalytics_Assignment3

Prepared by Tim Virga

17/10/2021

```r
# Use these two lines to read the data and set the column names
gy <- read.table("Yield.dat")
names(gy) <- c("x", "y", "yield", "B", "Ca", "Cu", "Fe",
"K", "Mg", "Mn", "Na", "P", "Zn")
# create basic plot
plot(I((y-1)*48.8)~I((x-1)*12.2), data=gy, xlab="east (metres)", type="n",
ylab="north (metres)", xlim=c(-10,360), ylim=c(-10,360))
# add grid lines
abline(v=(1:28-1)*12.2, h=(1:8-1)*48.8, col="grey")
# add secondary axes
axis(side=3, at=(1:28-1)*12.2, labels=1:28, line=0, cex.axis=0.7, mgp=c(3,1,0))
mtext("x", side=3, line=1, cex=0.7, adj=1)
axis(side=4, at=(1:8-1)*48.8, labels=1:8, line=0, cex.axis=0.7, mgp=c(3,1,0))
mtext("y", side=4, line=1, cex=0.7, adj=1)
# plot the site locations
points(I((y-1)*48.8)~I((x-1)*12.2), gy, pch=19)
# plot observation 200 in red
points(I((y-1)*48.8)~I((x-1)*12.2), gy, pch=19, col="red", subset=200)
title(sub="Observation 200 shown in red.", col="red")
```



Observation 200 shown in red.

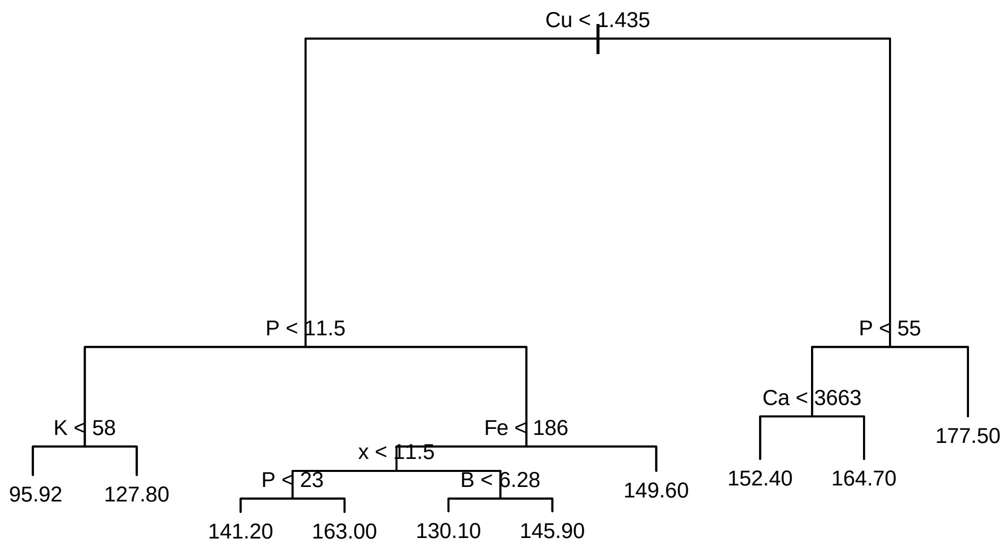
```r
# remove observation 200 from the data
gy <- gy[-200,]
```

Regression Tree

Q1

```
library(tree)

tree.gy = tree(yield ~ ., data=gy)
plot(tree.gy)
text(tree.gy, cex=.7)
```



```
tree.gy
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 214 85910.0 153.10
##    2) Cu < 1.435 81 32040.0 137.90
##      4) P < 11.5 11   3618.0 110.40
##        8) K < 58 6   588.8  95.92 *
##        9) K > 58 5   259.8 127.80 *
##      5) P > 11.5 70 18770.0 142.30
##       10) Fe < 186 43 12340.0 137.70
##         20) x < 11.5 11  2064.0 151.10
##           40) P < 23 6   221.5 141.20 *
##           41) P > 23 5   550.2 163.00 *
##         21) x > 11.5 32  7589.0 133.00
##           42) B < 6.28 26  4742.0 130.10 *
##           43) B > 6.28 6  1618.0 145.90 *
##       11) Fe > 186 27  4067.0 149.60 *
##    3) Cu > 1.435 133 23980.0 162.30
##      6) P < 55 109 15470.0 159.00
##       12) Ca < 3663 51  6715.0 152.40 *
##       13) Ca > 3663 58  4619.0 164.70 *
##      7) P > 55 24  1785.0 177.50 *
```

Here we observe the list of decision rules generated by the tree() model, noting that for each node we have visibility of the split condition, number of samples at that split, deviance and predicted yval at that split condition or terminal node.

Q2

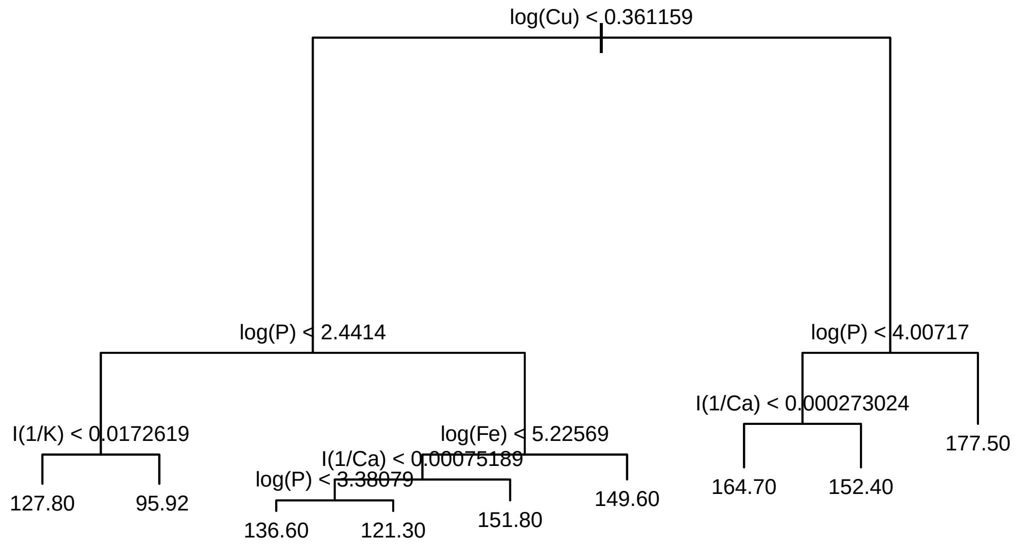
We are applying the decision rules to observation #22, condition statements and outcomes below are specific to observation #22:
1. Beginning from the root node, the first split condition Cu $<$ 1.435 is met, moving our observation prediction to the left child node P $<$ 11.5
2. P $<$ 11.5 is a split node, observation does not meet the split condition, therefore moving to the right child node Fe $<$ 186
3. Fe $<$ 186 is a split node, observation does not meet the split condition, therefore moving to the right terminal node associated with a predicted yield prediction of 149.60 along with 26 other observations.

Q3

The stopping criteria used in this tree is made up of yield prediction values determined by the tree() function algorithm as being the most computationally cost-effective based on modelling the observation data. Importantly the criteria considers an optimal degree of purity in pruning the tree to ensure accuracy not just for training data, but also future test data through means of not splitting the observations to completely pure nodes that result in overfitting.

Q4

```
tree.gy2 = tree(yield~B+I(1/Ca)+log(Cu)+log(Fe)+I(1/K)+I(1/Mg)+log(Mn)+Na+log(P)+log(Zn+1), data=gy)
plot(tree.gy2)
text(tree.gy2, cex=.7)
```



```
gy22 <- gy[22,]
predict(tree.gy2, newdata=gy22)
```

```
##       22
## 149.6063
```

Comparing the transformed soil nutrients with the untransformed soil nutrients, we observe that the split condition values that determine the left or right split are lower in threshold as a result of the transformation, outliers in particular are most affected by this sort of transformation. It is generally unnecessary to transform variables for regression tree models because there is no distance consideration between features in regression, unlike classification. With that being said, regression trees are mostly insensitive to transformed predictors. For example, in the case of observation #22 the prediction remains unchanged at 149.60 yield for transformed soil nutrients and untransformed soil nutrients.

Q5

```
# log linear model
gy.loglm = lm(yield~B+I(1/Ca)+log(Cu)+log(Fe)+I(1/K)+I(1/Mg)+log(Mn)+Na+log(P)+log(Zn+1), data=gy)
#RSS calculation
lm_res = sum(residuals(gy.loglm)^2)
gy_res = sum(residuals(tree.gy)^2)
gy_res2 = sum(residuals(tree.gy2)^2)
lm_res # linear model RSS
```

```
## [1] 37658.03
```

```
gy_res # untransformed predictors RSS
```

```
## [1] 25167.15
```

```
gy_res2 # transformed predictors RSS
```

```
## [1] 27390.91
```

We have calculated the RSS by summing and squaring residuals for the linear model, tree based model and tree based model with transformed variables. We observe the smallest RSS appointed to the tree based model with untransformed variables, noting a negative impact to the RSS for the soil nutrients variables having made no modifications to the model other than transforming the variables.

Q6

```
summary(tree.gy)
```

```
##
## Regression tree:
## tree(formula = yield ~ ., data = gy)
## Variables actually used in tree construction:
## [1] "Cu" "P"  "K"  "Fe" "x"  "B"  "Ca"
## Number of terminal nodes:  10
## Residual mean deviance:  123.4 = 25170 / 204
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -37.9300  -7.5150  -0.4605   0.0000   6.9550  29.8500
```

```
summary(gy.loglm)
```

```
##
## Call:
## lm(formula = yield ~ B + I(1/Ca) + log(Cu) + log(Fe) + I(1/K) +
##     I(1/Mg) + log(Mn) + Na + log(P) + log(Zn + 1), data = gy)
##
## Residuals:
```

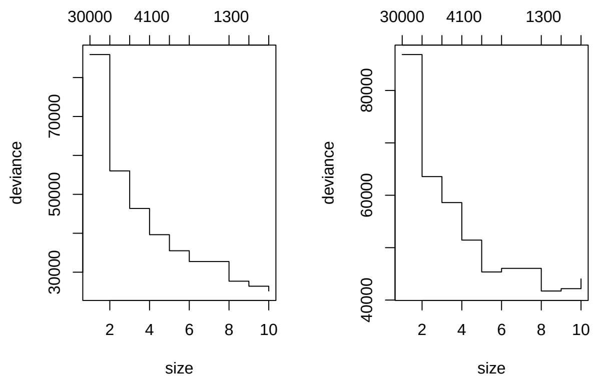
```
##     Min      1Q  Median      3Q     Max
## -50.963  -9.644  -0.029   8.956  32.202
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.285e+02  2.829e+01   4.540 9.63e-06 ***
## B            3.798e-01  7.856e-01   0.483 0.629289
## I(1/Ca)      2.396e+04  1.262e+04   1.899 0.058977 .
## log(Cu)      1.366e+01  3.810e+00   3.586 0.000421 ***
## log(Fe)     -2.082e+00  2.855e+00  -0.729 0.466769
## I(1/K)      -7.911e+02  4.140e+02  -1.911 0.057423 .
## I(1/Mg)     -4.346e+03  1.975e+03  -2.200 0.028938 *
## log(Mn)      3.194e+00  4.597e+00   0.695 0.487902
## Na          -9.069e-02  5.604e-01  -0.162 0.871598
## log(P)       5.652e+00  3.259e+00   1.734 0.084370 .
## log(Zn + 1)  7.425e+00  5.103e+00   1.455 0.147255
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.62 on 203 degrees of freedom
## Multiple R-squared:  0.5616, Adjusted R-squared:  0.5401
## F-statistic: 26.01 on 10 and 203 DF,  p-value: < 2.2e-16
```

As we can see by comparing summaries of our linear model and the untransformed tree model, the tree based model has performed variable selection as part of the function to produce a better model; constructing the tree with Cu, P, K, Fe, x, B and Ca only. Whereas our linear model without manual modification, incorporates all variables for the RSS calculation.

Furthermore, tree based models are advantageous for their easy interpretability, tree based models often more intuitively reflect human decision making processes. Despite the known limitations of tree based models adopting a 'greedy' optimisation approach, there are methods that can be employed to address this such as Bagging, Random Forest and Boosting.

Q7

```r
par(mfrow=c(1,2))
tree.gy.seq = prune.tree(tree.gy) # pruning plot for single decision tree
tree.gycv = cv.tree(tree.gy) # pruning plot for cross-validated trees
plot(tree.gy.seq)
plot(tree.gycv)
```



```r
print(tree.gy.seq$size) # size of tree
```

```
## [1] 10  9  8  6  5  4  3  2  1
```

```r
print(tree.gy.seq$dev) # single decision tree deviance
```

```
## [1] 25167.15 26395.87 27687.86 32737.08 35506.99 39639.30 46371.83 56021.04
## [9] 85908.48
```

```r
print(tree.gycv$dev) # cv trees deviance
```

```
## [1] 44048.55 42174.69 41728.86 46047.73 45363.56 51443.92 58603.11 63563.71
## [9] 86852.85
```

A tree should be pruned if we are interested in determining the optimal number of variables for our tree based model with bias-variance trade-off considered. By pruning the original untransformed predictor variables using prune.tree() we can expect result of an increase in deviance with the reducing size of the tree, converging eventually to a tree size of 1. Introducing cross-validation to a tree pruning process cv.tree() will calculate deviance across different samples of entire training data set over K folds to as a function of the cost complexity parameter k, leaving out 10% of the samples with each iteration. The result of this is that in our cv.tree() plot, we observe a higher rate of error in models with more variables (larger trees) because the variance between the k-folds averages out to a higher deviance.

It's important to run cv.tree() multiple times given the random nature of the data that is selected for modelling as part of the cross-validation method. In this example by averaging the α penalty term of the lowest deviance size tree a few times, we can be confident that the pruned tree will result in a better performing model that addresses the bias-variance trade-off. We observe that the optimal tree size in the cv.tree() output is around 6 to 8 leaf nodes.

Q8

```
library(randomForest)
```

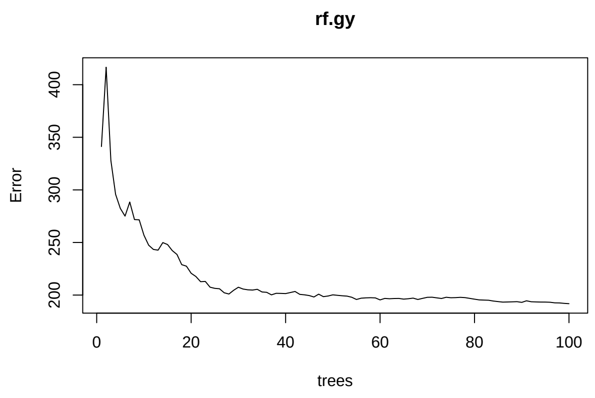
```
## randomForest 4.6-14

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

```
# random forest model with 100 trees
rf.gy = randomForest(yield ~.,data=gy, importance =TRUE, ntree=100)
rf.gy
```

```
##
## Call:
##  randomForest(formula = yield ~ ., data = gy, importance = TRUE,      ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 191.813
##                     % Var explained: 52.22
```

```
plot(rf.gy)
```



```
rf.gy.RSS <- sum((gy$yield - predict(rf.gy))^2) # RSS calculation for rf.gy
rf.gy.RSS
```

```
## [1] 41047.97
```

```
gy_res # RSS from earlier tree
```

```
## [1] 25167.15
```

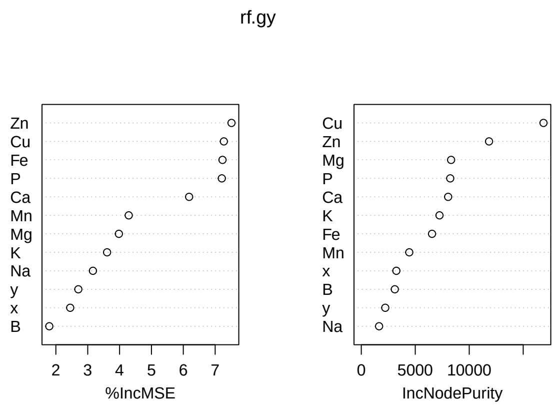
Comparing the RSS of Q5 single decision tree 25,167 to the RSS of our random forest model 39,773 is not a valid comparison. The single decision tree model is built once from the entire dataset that adopts a greedy optimisation method for split conditions that may result in a lower RSS however, at a cost of bias. The rf method in our example is modelling 100 trees with random variable selection at each split to determine the prediction and then averages the results (or 'votes') to determine the best model, addressing the bias-variance trade-off more effectively.

Q9

```r
importance(rf.gy) # importance of variables
```

```
##     %IncMSE IncNodePurity
## x  2.449942      3250.033
## y  2.704500      2221.577
## B  1.793449      3105.868
## Ca 6.182299      8046.160
## Cu 7.272787     16880.779
## Fe 7.230492      6553.727
## K  3.606966      7244.223
## Mg 3.978034      8321.111
## Mn 4.286944      4444.810
## Na 3.162965      1649.453
## P  7.209029      8240.068
## Zn 7.512969     11835.901
```

```r
varImpPlot(rf.gy)
```



By calling importance() we get a list of all predictors' respective %IncMSE and IncNodePurity for the random forest model. %IncMSE refers to the mean decrease in accuracy for the predictor variable when the variable is taken out of the model, a high %IncMSE such as I(1/Ca), log(Zn+1), log(P) and log(Cu) represent the most important variables to reduce error in the model. IncNodePurity measures the difference between RSS before and after the splits for each variable averaged over all trees; we observe log(Cu), log(Zn+1) and log(P) with the greatest RSS differences.

Comparing this information to the LASSO linear model which has plotted the most important variables by evaluation of the size of the penalty term applied for the respective coefficients to converge to 0, we follow the increasing penalty term for each variable to determine importance. We observe similarities in the optimal feature selection between models where LASSO reports: log(Cu), log(Zn+1), I(1/K) and log(P) as the most important predictors.

Interestingly, the LASSO model reports I(1/K) as being a much stronger predictor than the random forest model, this tells us that in the case of the LASSO model, once the λ value reaches a high enough value, that I(1/K) reveals itself as a strong predictor of the response variable.

Additionally, our LASSO model reported log(Zn+1) and log(Cu) as the two most important variables, in our random forest model they remained important variables however, positioned 2nd and 3rd in the scale of %IncMSE, suggesting that although these variables are strong predictors of the response variable, they may also be subject to a higher impact to bias due to their large contribution to node purity.