ECON 490ML Final Project Report

Abdi Lawrence, Henry Chen, Matthew Hong

4/26/2022

Introduction

For our Final Project, we will be attempting to predict the yield of wild blueberries. These are known as "lowbush berries", which grow on low-level bushes and are typically pea-sized. Our dataset can be found at **https://www.kaggle.com/datasets/saurabhshahane/wild-blueberry-yield-prediction**. This data was generated by a simulation model called the Wild Blueberry Pollination Model, which has been validated by experimental data collected in the State of Maine over the last 30 years.

Generally, in the field of agriculture, an increasing quantity of research has gone underway to understand the determinants of crop yield. Machine learning has enabled scientists and farmers to investigate which factors have the greatest impact on crop yield. In this specific case, the crop of interest is the wild blueberry. The paper for which this data was created is called "Simulation-based modeling of wild blueberry pollination". It was published in the January 2018 version of *Computers and Electronics in Agriculture*. The paper can be found at: **https://www.sciencedirect.com/science/article/pii/S0168169916310274?via%3Dihub**.

Handling Data

Below the dataset will be loaded from a .csv file. The `tidyverse` library has been loaded so that the data will automatically be loaded as a tibble, which provides some advantages over a standard data frame. One advantage being that the `read_csv` function will automatically assign types to each column in the tibble.

```
library(tidyverse)
data = read_csv("blueberryData.csv")
colnames(data)
```

```
##  [1] "Row#"                 "clonesize"            "honeybee"
##  [4] "bumbles"              "andrena"              "osmia"
##  [7] "MaxOfUpperTRange"     "MinOfUpperTRange"     "AverageOfUpperTRange"
## [10] "MaxOfLowerTRange"     "MinOfLowerTRange"     "AverageOfLowerTRange"
## [13] "RainingDays"          "AverageRainingDays"   "fruitset"
## [16] "fruitmass"            "seeds"                "yield"
```

```
dim(data)
```

```
## [1] 777  18
```

The dataset contains 18 columns and 777 observations. Along with downloaded the dataset from Kaggle, we downloaded the descriptions of the predictor variables in the dataset.

Table 1. Features and their description

Features	Unit	Description
Clonesize	m^2	The average blueberry clone size in the field
Honeybee	bees/m^2/min	Honeybee density in the field
Bumbles	bees/m^2/min	Bumblebee density in the field
Andrena	bees/m^2/min	Andrena bee density in the field
Osmia	bees/m^2/min	Osmia bee density in the field
MaxOfUpperTRange	℃	The highest record of the upper band daily air temperature during the bloom season
MinOfUpperTRange	℃	The lowest record of the upper band daily air temperature
AverageOfUpperTRange	℃	The average of the upper band daily air temperature
MaxOfLowerTRange	℃	The highest record of the lower band daily air temperature
MinOfLowerTRange	℃	The lowest record of the lower band daily air temperature
AverageOfLowerTRange	℃	The average of the lower band daily air temperature
RainingDays	Day	The total number of days during the bloom season, each of which has precipitation larger than zero
AverageRainingDays	Day	The average of raining days of the entire bloom season

There are 13 predictor variables in the dataset. Their information includes clone size (size of bush), bee density by species, temperature, and precipitation. `fruitset, fruitmass, seeds`, and `yield` are all response variables. All predictor and response variables are quantitative. There are no qualitative variables. `Row#` can be deleted because it serves no purpose.

```r
data = select(data, -'Row#')
dim(data)
```

```
## [1] 777  17
```

The dataset does not have any `NA` values. We also will not need to convert any columns into factors, since there is no categorical data.

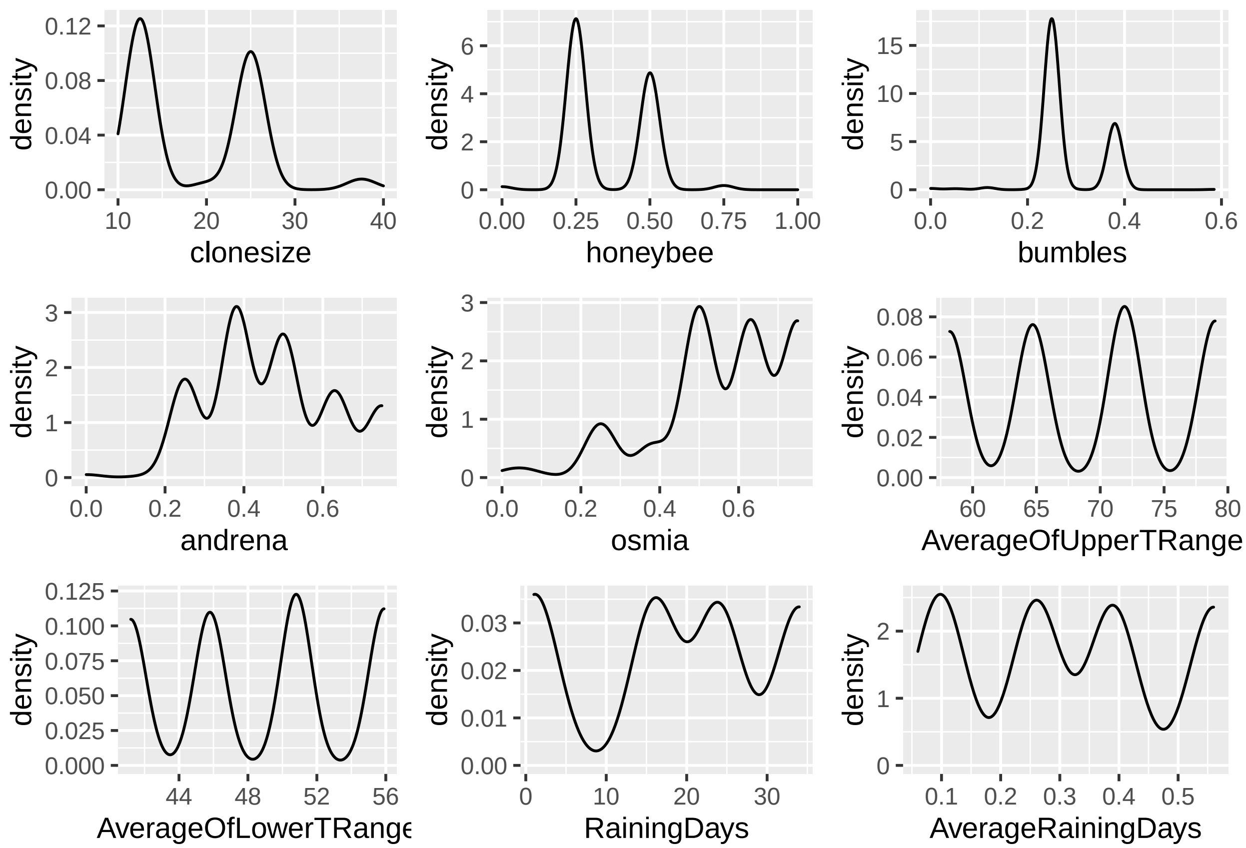
```r
sum(is.na(data))
```

```
## [1] 0
```

Exploration

A look at the density plots of 9 of our predictor variables shows that none of the distributions are approximately normal. Although this will not be detrimental to our algorithms, this was a surprise considering this is a simulation of natural factors (temperature, bees, precipitation).

```r
library(gridExtra)


plot1 = ggplot(data, aes(clonesize)) + geom_density()
plot2 = ggplot(data, aes(honeybee)) + geom_density() + xlim(0, 1)
plot3 = ggplot(data, aes(bumbles)) + geom_density()
plot4 = ggplot(data, aes(andrena)) + geom_density()
plot5 = ggplot(data, aes(osmia)) + geom_density()
plot6 = ggplot(data, aes(AverageOfUpperTRange)) + geom_density()
plot7 = ggplot(data, aes(AverageOfLowerTRange)) + geom_density()
plot8 = ggplot(data, aes(RainingDays)) + geom_density()
plot9 = ggplot(data, aes(AverageRainingDays)) + geom_density()


grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, plot7, plot8, plot9, ncol = 3)
```



Prediction

We will attempt to predict `yield` with the given predictor variables in the dataset. We chose `yield` as opposed to the other response variables because it is measured in quantity of blueberries, which we felt to the most important measure of a harvest.

Before beginning any machine learning algorithms, we decided to only retain `AverageOfLowerTRange` and `AverageOfUpperTRange` among out six temperature variables. The researchers used historical weather data to simulate the ranges of temperatures. Because of this we are comfortable using these two variables as our high and low temperatures. Our decision to remove these variables will simplify the model, and minimize multicolinearity. This will result in 9 predictor variables for our modeling.

```
data = select(data, -c(MaxOfUpperTRange, MinOfUpperTRange, MaxOfLowerTRange,fruitset, fruitmass, seeds,
dim(data)
```

```
## [1] 777  10
```

We will also create training and test sets from our data before starting our modeling. Out split will be 70% training, 30% testing.

```
set.seed(42)
train = sample(c(TRUE, FALSE), size = nrow(data), prob = c(0.7, 0.3), replace = TRUE)
traindata = data[train, ]
testdata = data[!train, ]
```

We will compare the performance of our models based on their RMSE values, which is Root Mean Squared Error.

Multiple Linear Regression

Our first method will be to model the data using Multiple Linear Regression, with all 9 predictor variables. The result of the regression is the formula

$$yield = 7928.955 - 98.207clonesize + 118.492honeybee + 5980.501bumbles + 520.207adrena + 2448.218osmia$$

$$+ 236.815AverageOfUpperTRange - 369.804AverageOfLowerTRange + 51.718RainingDays$$

$$- 8375.642AverageRainingDays$$

Surprisingly, `AverageOfUpperTRange` and `AverageOfLowerTRange` are not statistically significant in this model. Perhaps temperature does not have much an effect yield. It's too early to know for sure, but more evidence may be presented as we continue. This model produced an RMSE value of 590.3259. Another insight this model has provided is that bumblebees may have the largest impact on blueberry yield of the four species. Lastly, it was not expected that the coefficient of `clonesize` would be negative. We assumed that larger bushes produce more berries on average.

```
set.seed(42)
MLR = lm(yield ~ ., data = traindata)
summary(MLR)
```

```
##
## Call:
```

```
## lm(formula = yield ~ ., data = traindata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1596.85  -367.76    28.41   411.26  1399.72
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           7928.955    292.619  27.096  < 2e-16 ***
## clonesize              -98.207      3.778 -25.998  < 2e-16 ***
## honeybee               118.492     23.828   4.973 8.91e-07 ***
## bumbles               5980.501    409.568  14.602  < 2e-16 ***
## andrena                520.207    173.790   2.993  0.00289 **
## osmia                 2448.218    180.627  13.554  < 2e-16 ***
## AverageOfUpperTRange   236.815    306.793   0.772  0.44051
## AverageOfLowerTRange  -369.804    434.823  -0.850  0.39544
## RainingDays             51.718     16.323   3.168  0.00162 **
## AverageRainingDays   -8375.642   1159.401  -7.224 1.75e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 591.5 on 535 degrees of freedom
## Multiple R-squared:  0.8115, Adjusted R-squared:  0.8084
## F-statistic:   256 on 9 and 535 DF,  p-value: < 2.2e-16
```

```r
MLRtest = predict(MLR, newdata = testdata)
sqrt(mean((MLRtest - testdata$yield) ^ 2)) #RMSE
```

```
## [1] 590.3259
```

Best Subset Selection

Using best subset selection, we created a model Multiple Linear Regression model with 8 predictor variables. We chose to use 8 variables because this quantity produced the best score for 3 out the 4 measures. This meant that `AverageOfUpperTRange` was omitted from the model. This method produced the formula

$$yield = 7908.489 - 98.294 clonesize + 118.673 honeybee + 5962.802 bumbles + 524.529 adrena + 2430.845 osmia$$
$$- 34.181 AverageOfLowerTRange + 52.857 RainingDays - 8461.237 AverageRainingDays$$

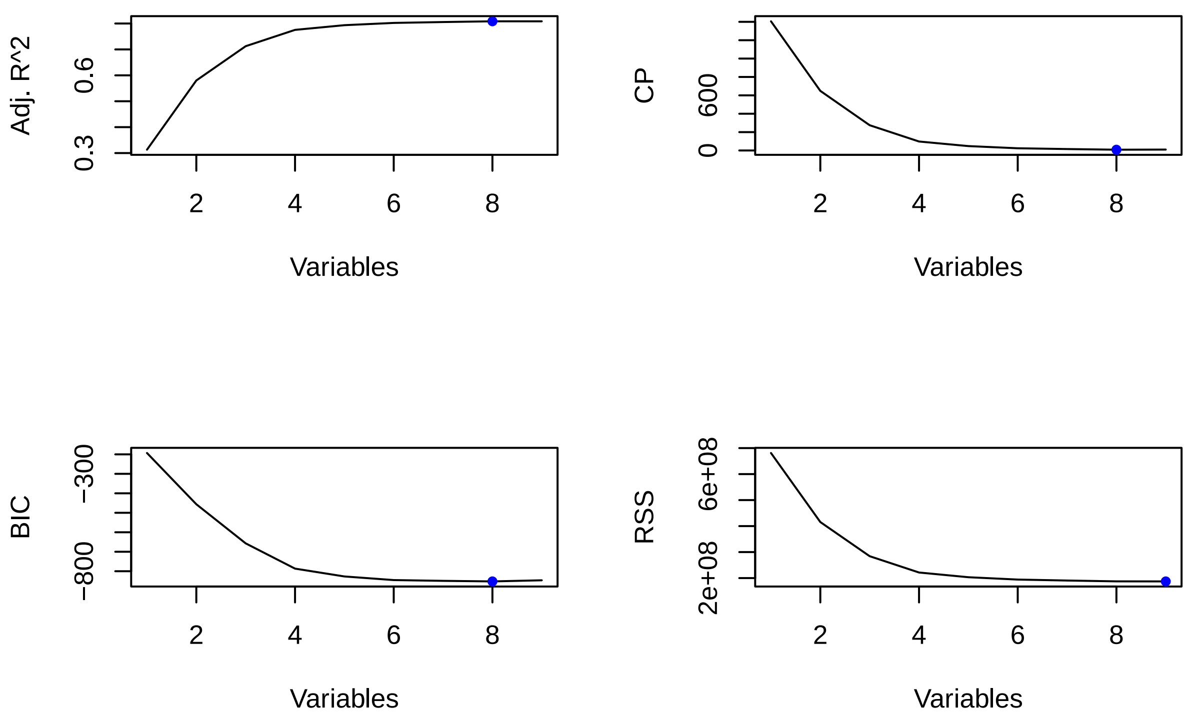
This model produced an RMSE of 593.2259 on the testing data, which is slightly worse in performance than our Multiple Linear Regression Model with all predictor variables.

```
library(leaps)
set.seed(42)

regfit.full = regsubsets(yield ~ ., data = traindata, nvmax = ncol(data) - 1)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(yield ~ ., data = traindata, nvmax = ncol(data) -
##     1)
## 9 Variables  (and intercept)
##                        Forced in Forced out
## clonesize                  FALSE      FALSE
## honeybee                   FALSE      FALSE
## bumbles                    FALSE      FALSE
## andrena                    FALSE      FALSE
## osmia                      FALSE      FALSE
## AverageOfUpperTRange       FALSE      FALSE
## AverageOfLowerTRange       FALSE      FALSE
## RainingDays                FALSE      FALSE
## AverageRainingDays         FALSE      FALSE
## 1 subsets of each size up to 9
## Selection Algorithm: exhaustive
##          clonesize honeybee bumbles andrena osmia AverageOfUpperTRange
## 1  ( 1 ) " "       " "      " "     " "     " "   " "
```

```
## 2  ( 1 ) "*"       " "      " "     " "     " "   " "
## 3  ( 1 ) "*"       " "      " "     " "     "*"   " "
## 4  ( 1 ) "*"       " "      "*"     " "     "*"   " "
## 5  ( 1 ) "*"       " "      "*"     " "     "*"   " "
## 6  ( 1 ) "*"       "*"      "*"     " "     "*"   " "
## 7  ( 1 ) "*"       "*"      "*"     " "     "*"   " "
## 8  ( 1 ) "*"       "*"      "*"     "*"     "*"   " "
## 9  ( 1 ) "*"       "*"      "*"     "*"     "*"   "*"
##          AverageOfLowerTRange RainingDays AverageRainingDays
## 1  ( 1 ) " "                  "*"         " "
## 2  ( 1 ) " "                  " "         "*"
## 3  ( 1 ) " "                  " "         "*"
## 4  ( 1 ) " "                  " "         "*"
## 5  ( 1 ) "*"                  " "         "*"
## 6  ( 1 ) "*"                  " "         "*"
## 7  ( 1 ) "*"                  "*"         "*"
## 8  ( 1 ) "*"                  "*"         "*"
## 9  ( 1 ) "*"                  "*"         "*"
```

```r
reg.summary = summary(regfit.full)

par(mfrow = c(2, 2))
plot(reg.summary$adjr2, xlab = "Variables", ylab = "Adj. R^2", type = "l")
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)], col = "blue", pch

plot(reg.summary$cp, xlab = "Variables", ylab = "CP", type = "l")
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)], col = "blue", pch = 20)

plot(reg.summary$bic, xlab = "Variables", ylab = "BIC", type = "l")
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)], col = "blue", pch = 20)

plot(reg.summary$rss, xlab = "Variables", ylab = "RSS", type = "l")
points(which.min(reg.summary$rss), reg.summary$rss[which.min(reg.summary$rss)], col = "blue", pch = 20)
```




```
coefficients(regfit.full, 8)
```

```
##          (Intercept)            clonesize             honeybee
##           7908.48943            -98.29390            118.67290
##              bumbles              andrena                osmia
##           5962.80210            524.52885           2430.84531
## AverageOfLowerTRange          RainingDays   AverageRainingDays
##            -34.18089             52.85697          -8461.23701
```

```
subsetEQ = lm(yield ~ . - AverageOfUpperTRange, data = traindata) #Run regression with only 8 variables

subsetTest = predict(subsetEQ, newdata = testdata)
sqrt(mean((subsetTest - testdata$yield) ^ 2)) #RMSE
```

```
## [1] 593.2259
```

Ridge Regression

Using the Ridge Regression technique, we estimated the `yield` value based on all 9 predictor variables. The cross-validation method returned a lambda value of 75.6745. With this lambda value, we received the formula

$$yield = 7656.7 - 92.8 clonesize + 113.78 honeybee + 5423.51 bumbles + 500.52 adrena + 2244.37 osmia$$

$$- 11.47 AverageOfUpperTRange - 16.975 AverageOfLowerTRange - 21.945 RainingDays$$

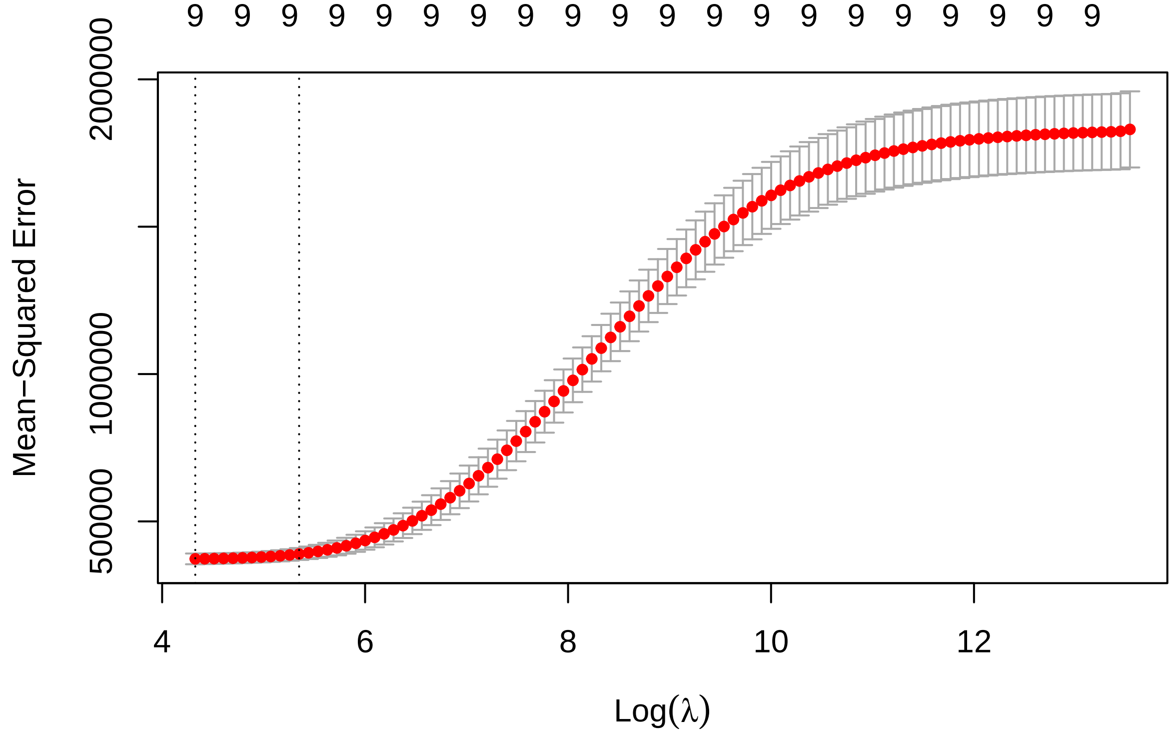
$$- 3012.2 AverageRainingDays$$

Unsurprisingly, the coefficients for both our temperature related variables are much smaller with ridge regression. Because of the penalty involved, it was expected that these variables would have their effects diminished, as they were not statistically significant in our first method. This method produced an RMSE of 599.4205, which is slightly higher than both models we've done so far.

```
library(glmnet)
train.test = model.matrix(yield~., data = traindata)
x.test = model.matrix(yield~., data = testdata)



# Finding lambda chosen by cross-validation
set.seed(42)
ridge.cv = cv.glmnet(train.test, traindata$yield, alpha = 0)
lambda.ridge = ridge.cv$lambda.min
lambda.ridge
```

```
## [1] 75.6745
```

```
plot(ridge.cv)
```




```r
#Fitting to ridge regression
ridge.fit = glmnet(train.test, traindata$yield, alpha = 0, lambda = lambda.ridge)
coef(ridge.fit)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                 s0
## (Intercept)            7656.70319
## (Intercept)                  .
## clonesize               -92.80157
## honeybee                113.78034
## bumbles                5423.50681
## andrena                 500.51521
## osmia                  2244.36986
## AverageOfUpperTRange    -11.47164
## AverageOfLowerTRange    -16.97510
## RainingDays             -21.94500
```

```
## AverageRainingDays   -3012.19532
```

```r
#Mean square error
ridge.pred = predict(ridge.fit, newx=x.test, s = lambda.ridge)
sqrt(mean((testdata$yield - ridge.pred)^2))
```

```
## [1] 599.4205
```

Lasso Regression

With the Lasso Regression technique, we got the best lambda value of 0.586. Using this lambda value, we received the estimated formula

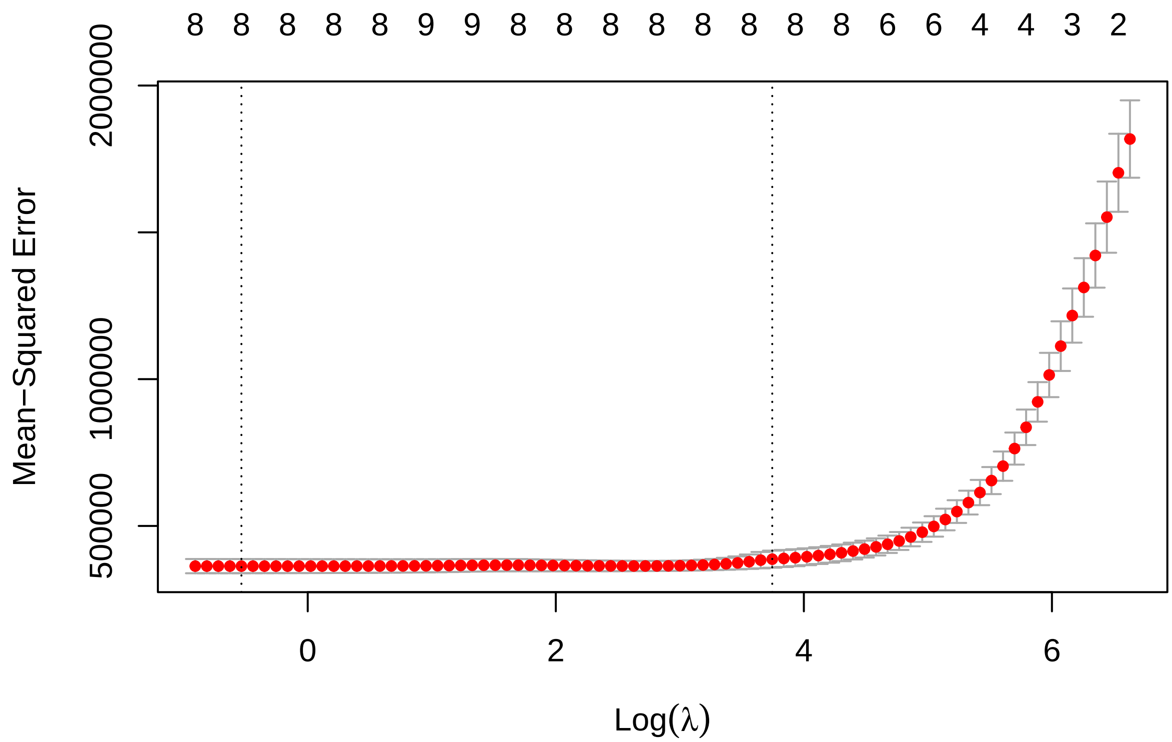
$$
\begin{gathered}
yield = 7880.01\ -\ 98.15 clonesize\ +\ 119.04 honeybee\ +\ 5930.24 bumbles\ +\ 519.9 adrena\ +\ 2419.37 osmia \\
-\ 15.3 AverageOfUpperTRange\ -\ 12.37 AverageOfLowerTRange\ +\ 44.93 RainingDays \\
-\ 7899.73 AverageRainingDays
\end{gathered}
$$

This estimation has an RMSE of 592.97. Similar to the Ridge Regression, the coefficients of the temperature variables are quite small.

```r
# Finding lambda chosen by cross-validation
set.seed(42)
lasso.cv = cv.glmnet(train.test, traindata$yield, alpha = 1)
lambda.lasso = lasso.cv$lambda.min
lambda.lasso
```

```
## [1] 0.5859202
```

```r
plot(lasso.cv)
```




```r
#Fitting to lasso regression
lasso.fit = glmnet(train.test, traindata$yield, alpha = 1, lambda = lambda.lasso)
coef(lasso.fit)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)           7880.01439
## (Intercept)                .
## clonesize              -98.15390
## honeybee               119.03992
## bumbles               5930.24157
## andrena                519.89848
## osmia                 2419.36850
## AverageOfUpperTRange   -15.29857
## AverageOfLowerTRange   -12.36606
## RainingDays             44.92750
```

```
## AverageRainingDays   -7899.73466
```

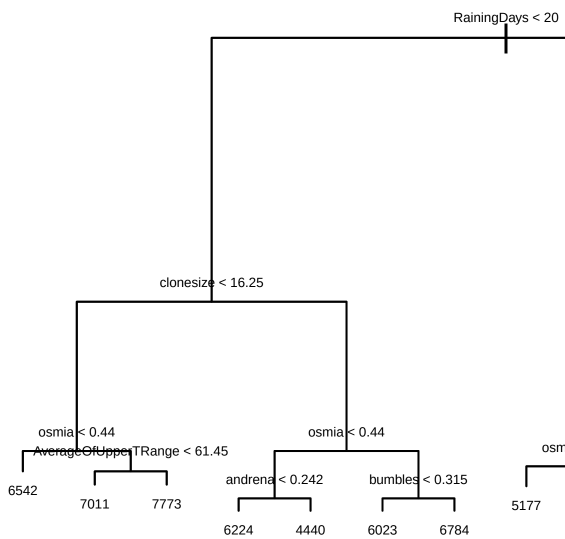
```r
#Mean square error
lasso.pred = predict(lasso.fit, newx=x.test, s = lambda.lasso)
sqrt(mean((testdata$yield - lasso.pred)^2))
```

```
## [1] 592.9726
```

Regression Tree

Below we ran a single regression tree. The tree initially returned 14 nodes. According to the k-fold cross validation technique, pruning the tree does not improve performance. We left our tree unchanged with 14 nodes. This tree produced an RMSE of 650.23. This RMSE is significantly lower than the methods that have been attempted so far.

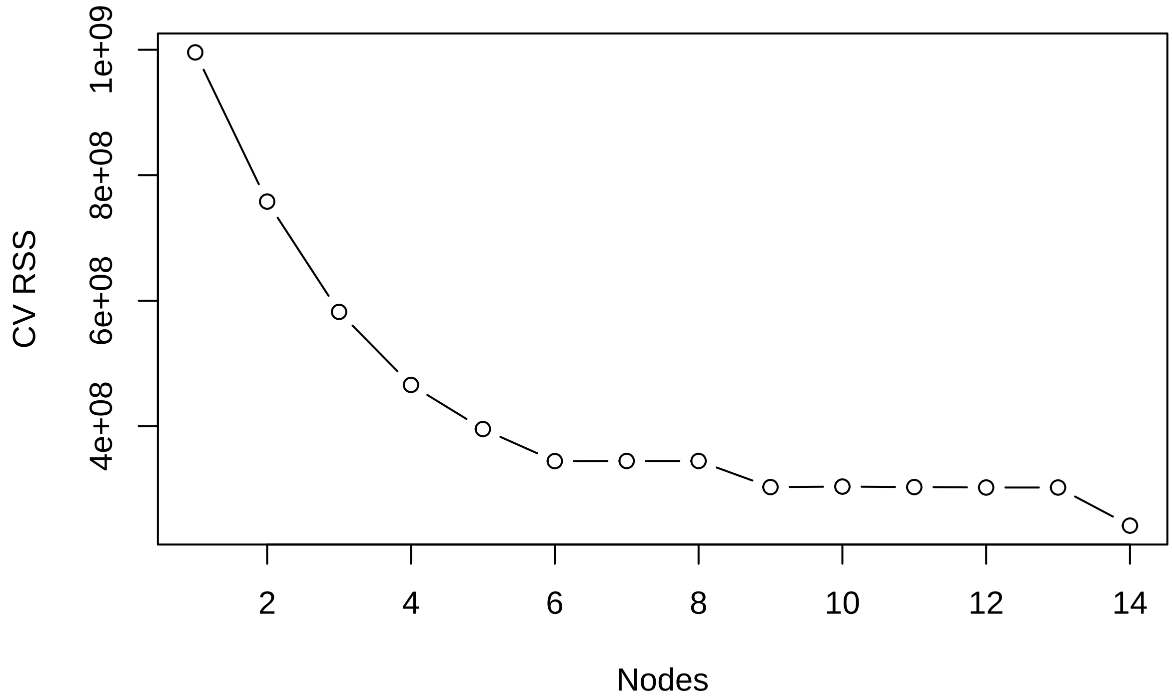
```r
## creating regression tree with training data
library(tree)
set.seed(42)
tree.train =  tree(yield ~ ., data = traindata)
plot(tree.train)
text(tree.train, pretty = 0, cex = 0.45)
```




```r
##test RMSE prior to CV + pruning
predict.yield1 = predict(tree.train, newdata = testdata)
sqrt(mean((predict.yield1 - testdata$yield)^2))
```

```
## [1] 650.2313
```

```r
## apply K-fold cross validation, K = 10 is standard size to use
## graph reveals that the training RSS is at its minimum at 14 nodes
cv.train = cv.tree(tree.train, K = 10)
plot(cv.train$size, cv.train$dev, xlab = "Nodes", ylab = "CV RSS", type = "b")
```



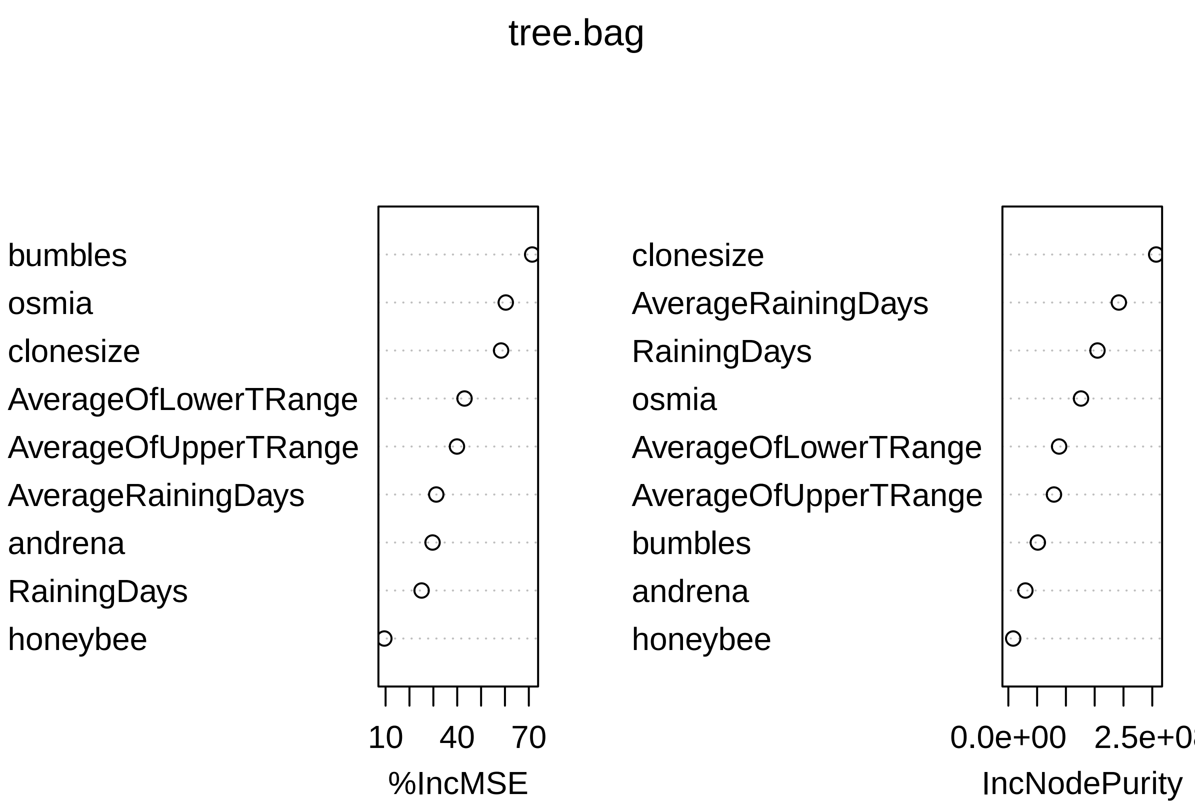

```
## prune tree
tree.prune = prune.tree(tree.train, best = 14)

##conclude pruning does not affect RMSE
predict.yield2 = predict(tree.prune, newdata = testdata)
sqrt(mean((predict.yield2 - testdata$yield)^2))
```

```
## [1] 650.2313
```

Bagging

Below we ran a bagging regression with trees. According to the `importance` function, the most important variables are `bumbles`, `osmia`, and `clonesize` in that order. This methods returned the lowest RMSE so far, with a value of 381.66. As expected, bagging with 500 trees performed better than a single tree.

```r
##Bagging tries to reduce variance which will in turn reduce RMSE
library(randomForest)
set.seed(42)
tree.bag = randomForest(yield ~ ., data = traindata, mtry = 9, importance = T)
##Significantly smaller RMSE relative to what we have done so far
predict.yield3 = predict(tree.bag, newdata = testdata)
varImpPlot(tree.bag)
```



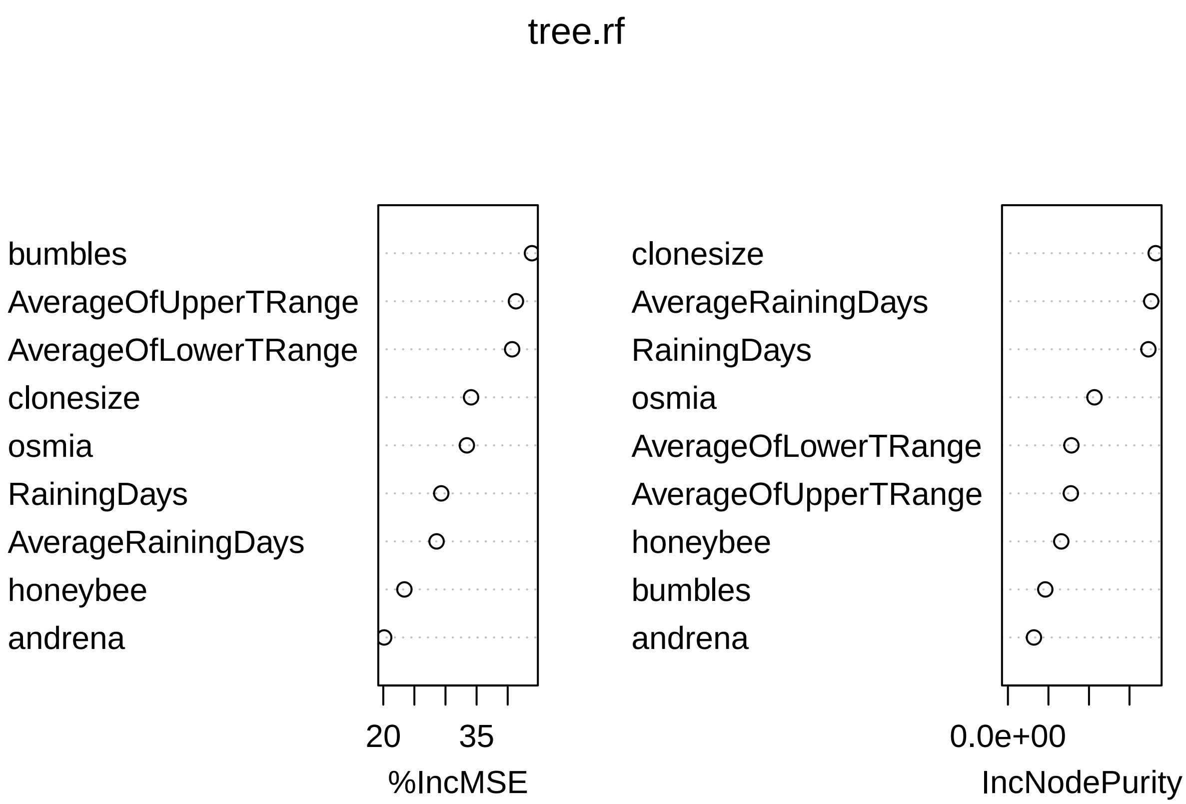

```r
sqrt(mean((predict.yield3 - testdata$yield)^2))
```

```
## [1] 381.6557
```

Random Forests

Lastly, we modeled our data with the Random Forests technique. For each tree, we set the variables selected to 3. According to the `importance` function, the most important variables are `bumbles`, `AverageOfUpperTRange`, and `AverageOfLowerTRange`. The significance of the temperature variables is likely overvalued as a result of the Random Forests method. Because each tree only used 3 variables, the temperature variables had a higher relative impact. `bumbles` on the other hand, is the most impactful variable, which is consistent with our findings so far. This model performed the best out of all our methods, with an RMSE of 367.44.

```r
##Random Forests to see if it'll produce a even smaller RMSE
set.seed(42)
tree.rf = randomForest(yield ~ ., data = traindata, mtry = 3, importance = T)
varImpPlot(tree.rf)
```




```r
predict.yield4 = predict(tree.rf, newdata = testdata)
#RMSE produced by Random Forests is smallest out of all tree methods used
sqrt(mean((predict.yield4 - testdata$yield)^2))
```

```
## [1] 367.4367
```

Conclusion

After running several models, we can conclude that a Random Forest method models our data very well. Unfortunately, a formula cannot be derived from the Random Forest method. Despite this fact, we can still extract valuable information from our models with formulas. Our best performing model was Multiple Linear Regression, with all 9 of our variables. This method resulted in the following formula

$$yield = 7928.955 - 98.207 clonesize + 118.492 honeybee + 5980.501 bumbles + 520.207 adrena + 2448.218 osmia$$

$$+ 236.815 AverageOfUpperTRange \; - \; 369.804 AverageOfLowerTRange \; + \; 51.718 RainingDays$$

$$- 8375.642 AverageRainingDays$$

We discovered that bumblebees have a substantial impact on blueberry yield. According to our MLR model, for each additional bumblebee per m^2 per second, blueberry yield is expected to increase by approximately 5980.5. According to our Random Forest model, bumbleebee density has the greatest impact on blueberry yield.

On the other hand, honeybees do not have much of an impact on blueberry yield. According to our Random Forest models, they are either the least or second least important variable in affecting blueberry yield. This is also the case in our MLR model, in which `bumblebee` only has a coefficient of 118.492, which is the smallest of any bee species.

Lastly, we discovered that `clonesize` has a negative effect on blueberry yield. We must keep in mind ,however, that the range of `clonesize` values is [10, 40]. This likely means that the optimal size is around 10 squared meters, and as size increases yield is expected to decrease. We cannot expect a bush if size 0.5 squared meters to produce a higher yield than one of size 10 squared meters. Our inference is that a larger size may lead to an inefficient distribution of resources (soil, nutrients, water). Perhaps, due to natural circumstances, a smaller bush can sustain itself better.

References https://www.kaggle.com/datasets/saurabhshahane/wild-blueberry-yield-prediction

https://www.sciencedirect.com/science/article/pii/S016816992031156X?via%3Dihub