ECON 490ML Final Project Report

Abdi Lawrence, Henry, Matthew

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 Main 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.

```r
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.

```
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.

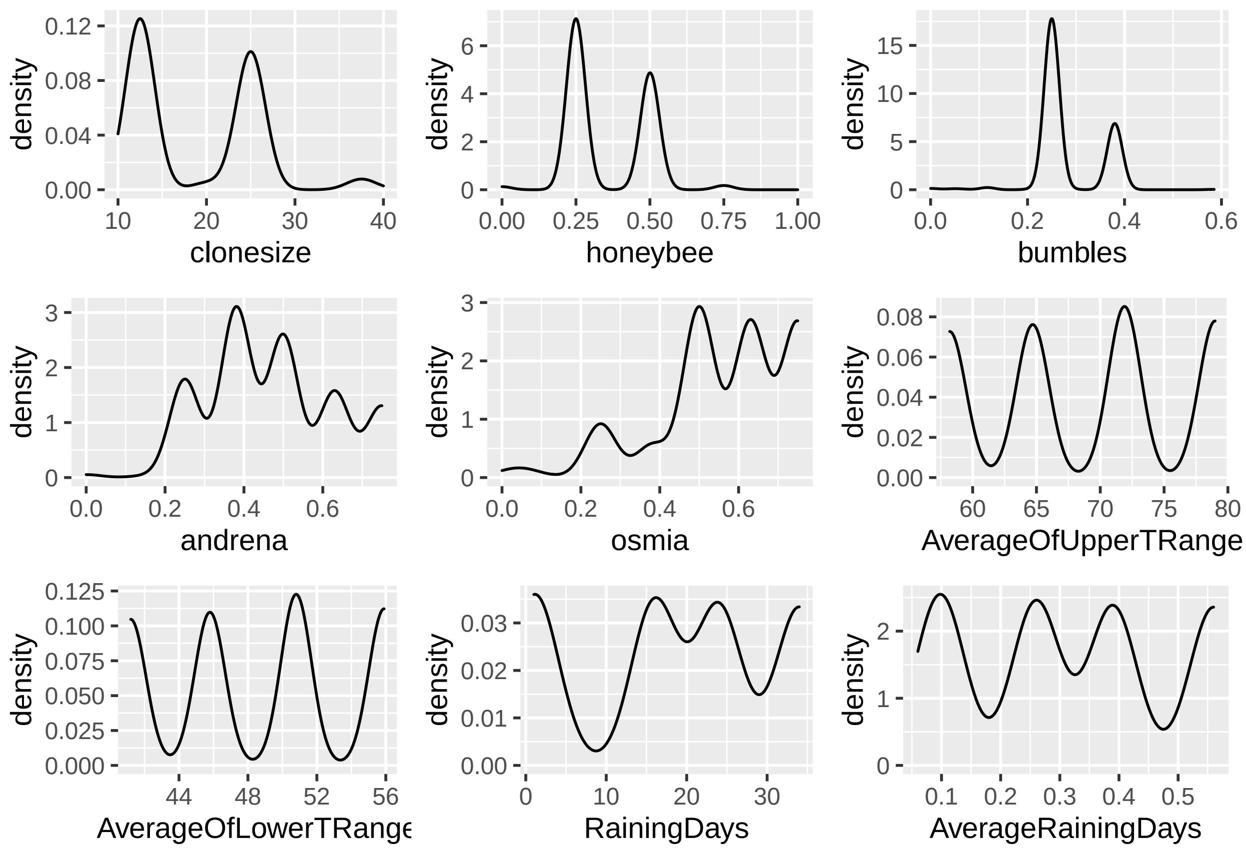
```
sum(is.na(data))
```

```
## [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).

```
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` amongst 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, MinOfLowerTRange, fruitset
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, ]
```

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

$$yield = 79.28.955 - 98.207 clonesize + 118.492 honeybee + 5980.501 bumbles + 520.207 adrena + 2448.218 osmia$$

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

$$- 8375.642 AverageRainingDays$$

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. The MSE produced was quite high, at a value of 348484.6. The model may be overfitting the data, since it performed so poorly on the test data.

```
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
```

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

```
## [1] 348484.6
```

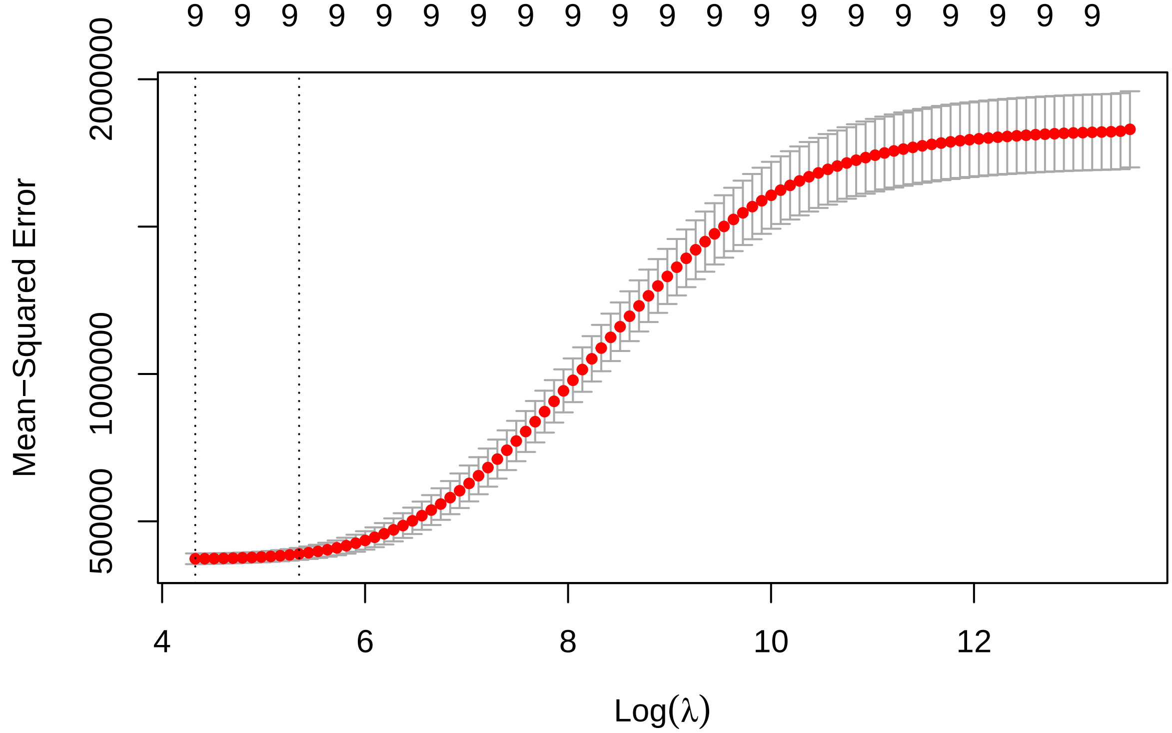
Ridge Regression

```
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)
```




```
#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)
mean((testdata$yield - ridge.pred)^2)
```

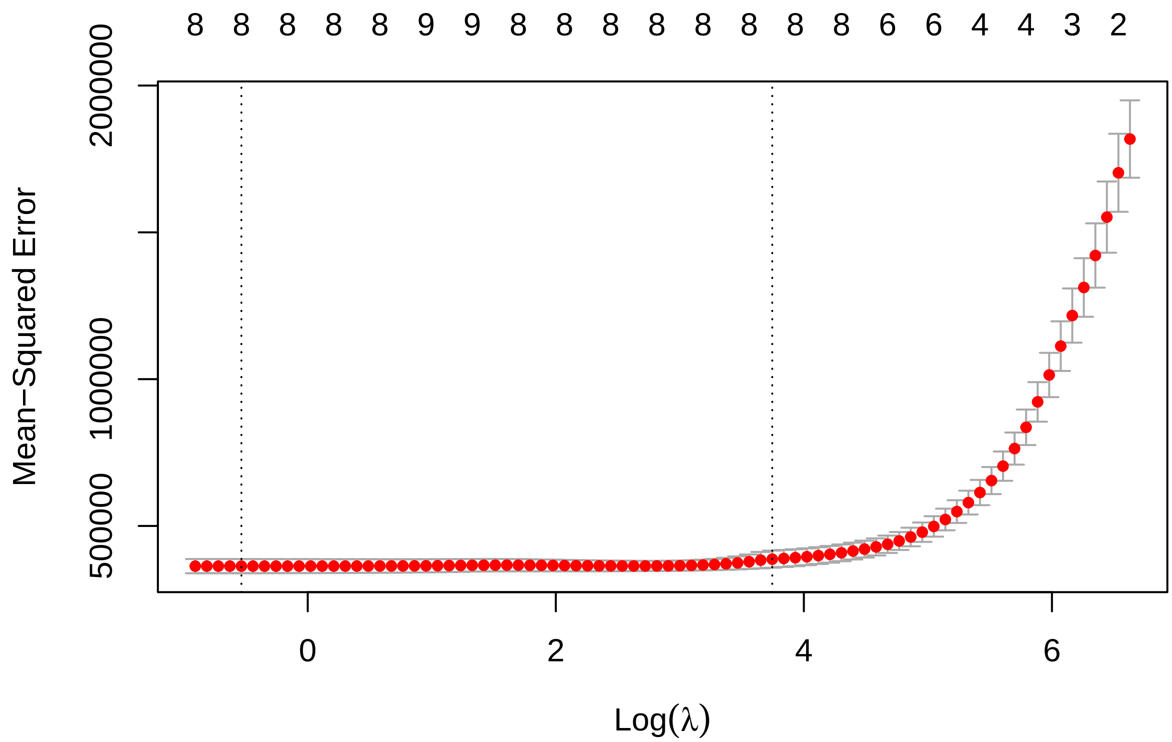
```
## [1] 359304.9
```

Lasso Regression

```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)
mean((testdata$yield - lasso.pred)^2)
```

```
## [1] 351616.5
```

There are 11 non-zero coefficient estimates in the lasso regression