Machine learning project for Classification of biome type

Agnes McFarlin

2023-12-12

Abstract

This report focuses on loading and visualizing data, and testing different types of Machine learning models in R. Their performances were analyzed and discussed. The specific subject in question was data that did not follow a normal distribution in general. And the goal was to determine which model performed best under the circumstances to categorize different biomes based on enviornmenal factors. The models used were: A Random Forest Classifier, Support Vector Machine(SVM), Decision Tree Classifier and Naive Bayes Classifier.

Introduction/Background

The Data being used comes from the U.S. Environmental Protection Agency (EPA Henceforth). There were two files, one for 2007 and one for 2012, each file contains information about the evaporation-to-inflow ratio of and water residence time for over a thousand lakes around the United States.[1] The below table briefly describes the column names of the data that were of interest to us. The data was originally collected as part of a study ''Lake Water Levels and Associated Hydrologic Characteristics in the Conterminous U.S.''[2]''Lake Hydrologic study variables include water-level drawdown and two water stable isotope-derived parameters: evaporation-to-inflow (E:I) and water residence time."[2] Given just the characteristics of an area, can a Machine Learning model classify the type of area.

Name of Variable & Description
ECO_BIO - Type of Enviornment
RT - Retention time of water in each lake
EI - Evaporation Inflow Rate
dD_H2O - Water Type used for comparison
d18_H2O - Water Type used for comparison

references

https://catalog.data.gov/dataset/nars-hydrologic-data%5B1%5D

https://onlinelibrary.wiley.com/doi/10.1111/1752-1688.12817%5B2%5D

Data Loading and initial set up

Data is loaded in and some basic cleaning is shown here. Mostly getting rid of null and NA values or empty rows.

```
library(dplyr)
library(caret)
library(rpart.plot)
library(ggplot2)
library(caTools)
library(e1071)
library(randomForest)
library(caTools)
library(randomForest)

nla2012_isotopes_wide <- read.csv("C:/Users/amcfa/gitfiles/Projects/RStudio/Project_water_table_Isotopes
nla2007_isotopes_wide <- read.csv("C:/Users/amcfa/gitfiles/Projects/RStudio/Project_water_table_Isotopes

abc <-nla2012_isotopes_wide
defg <- nla2007_isotopes_wide



length(which(is.na(abc)))
```

```
## [1] 62
```

```
length(which(is.na(defg)))
```

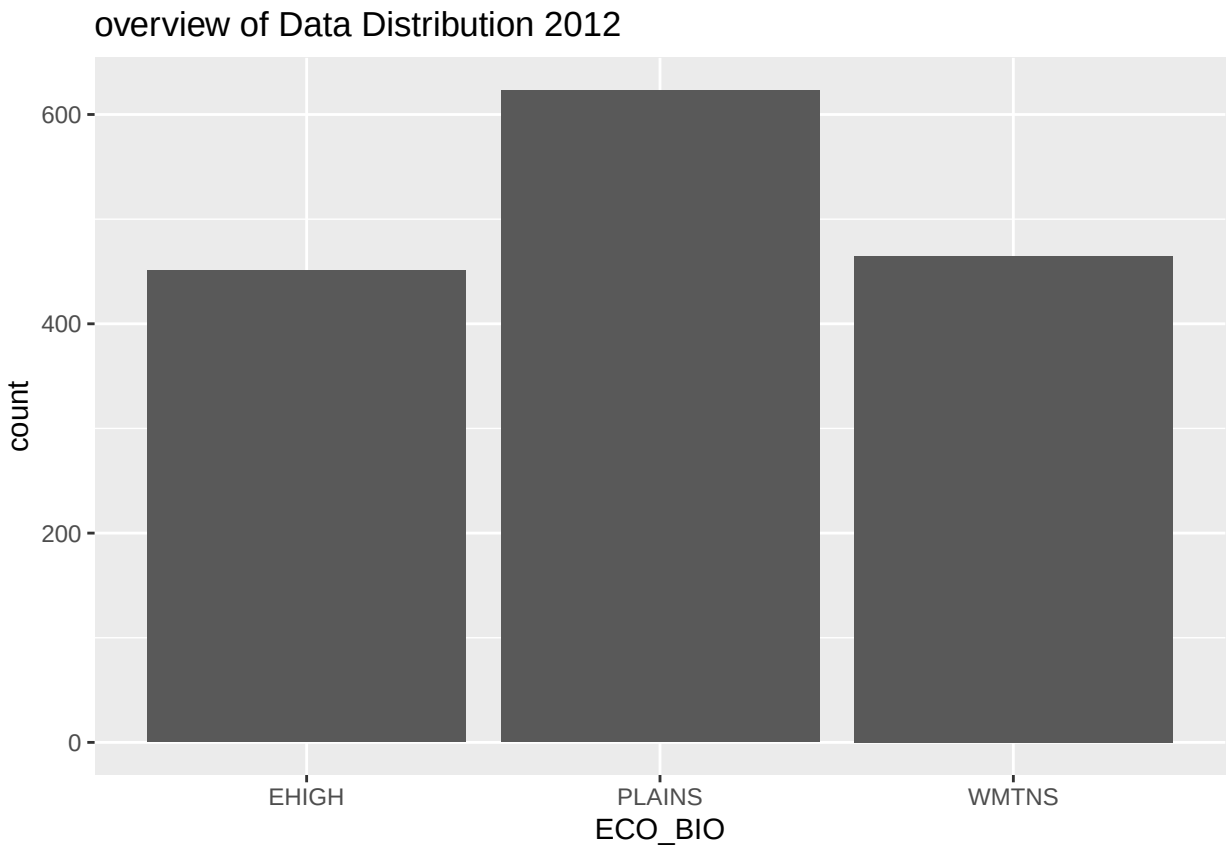
```
## [1] 0
```

```
abc <- na.omit(abc)
defg <- na.omit(defg)
```

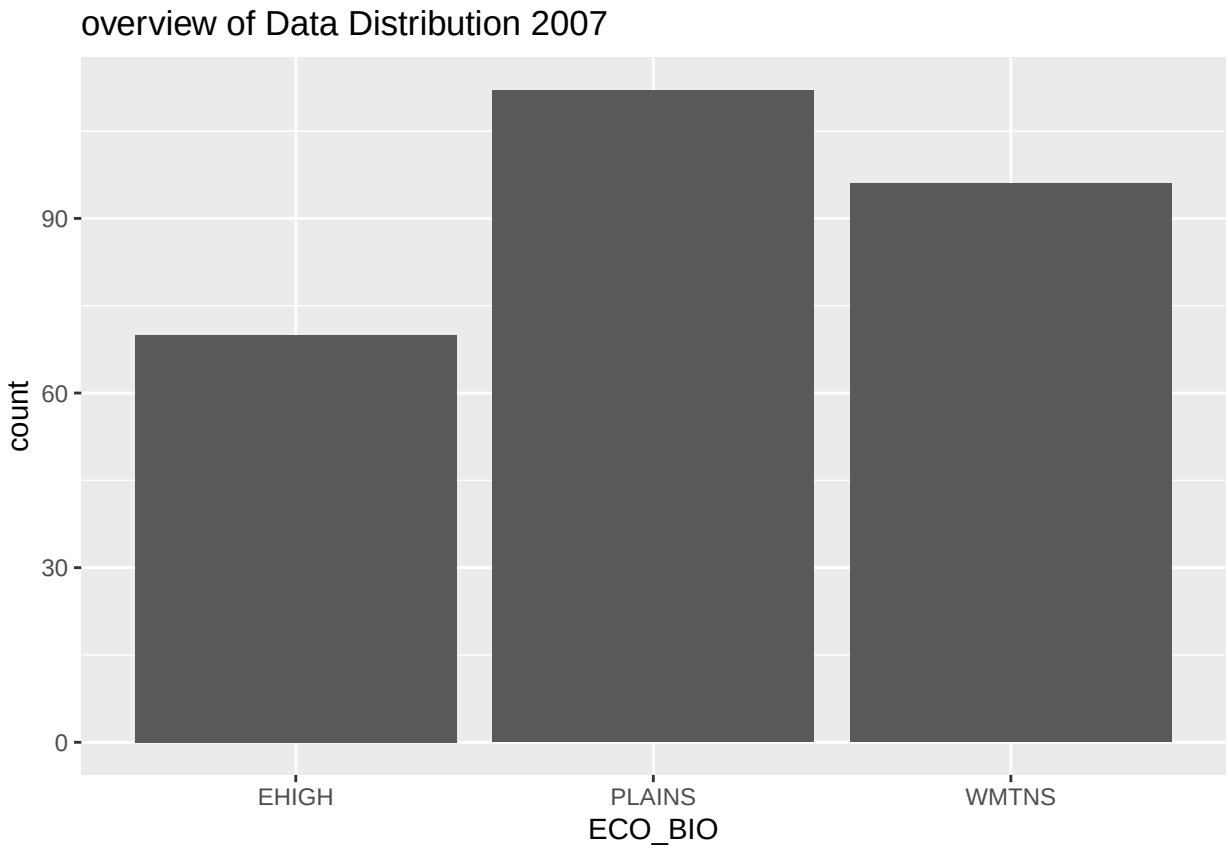
The different biomes were turned into factors. The most common biomes were used for this study, which were the plains, Western Mountains and Eastern Highlands. To provide more data for the models to work with, the two dataframes were combined and randomly sampled from to create training and test sets.

```
abc$ECO_BIO <- factor(abc$ECO_BIO)
defg$ECO_BIO <- factor(defg$ECO_BIO)
abc1 <-select(abc, c('ECO_BIO','RT','E_I','dD_H2O','d18O_H2O'))
defg1 <-select(defg, c('ECO_BIO','RT','E_I','dD_H2O','d18O_H2O'))



set.seed(42)

trying_shorter<-bind_rows(abc1, defg1)

smp_size <- floor(0.85 * nrow(trying_shorter))

train_ind <- sample(seq_len(nrow(trying_shorter)), size = smp_size)

abc1 <- trying_shorter[train_ind, ]
defg1 <- trying_shorter[-train_ind, ]
```

```r
target <- c("PLAINS","WMTNS",'EHIGH')
abc1<-filter(abc1,ECO_BIO %in% target)
defg1<-filter(defg1,ECO_BIO %in% target)
abc1$ECO_BIO <- factor(abc1$ECO_BIO)
defg1$ECO_BIO <- factor(defg1$ECO_BIO)

ggplot(abc1,aes(ECO_BIO))+
     geom_bar() +
     ggtitle("overview of Data Distribution 2012")
```




```r
ggplot(defg1,aes(ECO_BIO))+
     geom_bar() +
     ggtitle("overview of Data Distribution 2007")
```



A first model

A model was run without performing any outlier removal or scaling. The accuracy was fair, at 68.7%. The decision was made to scale the data and remove outliers to improve model metrics.

```
set.seed(50)
trControl <- trainControl(method = "cv",
    number = 5,
    search = "grid")


rf_default <- train(ECO_BIO~.,
    data = abc1,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl)



print(rf_default)
```

```
## Random Forest
##
## 1539 samples
```

```
##    4 predictor
##    3 classes: 'EHIGH', 'PLAINS', 'WMTNS'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1231, 1232, 1231, 1232, 1230
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7426847  0.6096491
##   3     0.7420311  0.6082605
##   4     0.7407387  0.6064747
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
prediction <-predict(rf_default, defg1)
confusionMatrix(prediction, defg1$ECO_BIO)
```

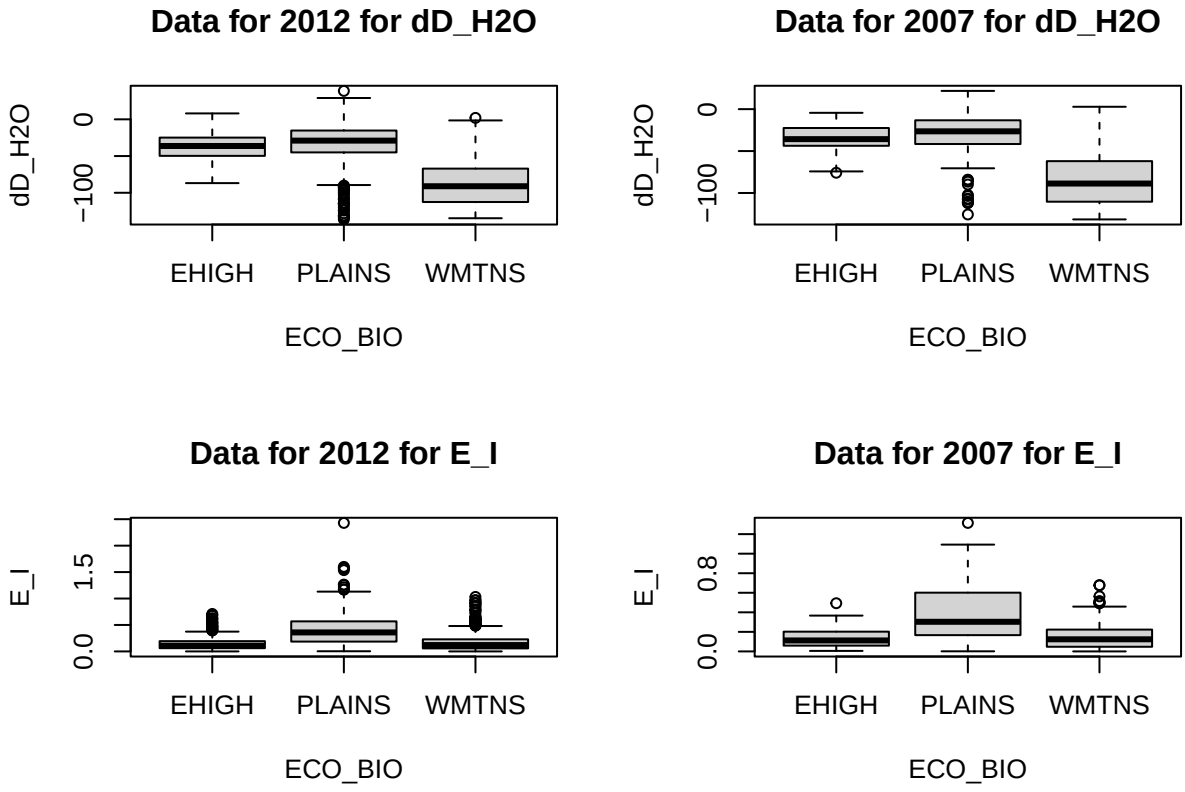
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction EHIGH PLAINS WMTNS
##     EHIGH     52     24    14
##     PLAINS    15     75    18
##     WMTNS      3     13    64
##
## Overall Statistics
##
##                Accuracy : 0.6871
##                  95% CI : (0.629, 0.7411)
##     No Information Rate : 0.4029
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5277
##
##  Mcnemar's Test P-Value : 0.01856
##
## Statistics by Class:
##
##                      Class: EHIGH Class: PLAINS Class: WMTNS
## Sensitivity                0.7429        0.6696       0.6667
## Specificity                0.8173        0.8012       0.9121
## Pos Pred Value             0.5778        0.6944       0.8000
## Neg Pred Value             0.9043        0.7824       0.8384
## Prevalence                 0.2518        0.4029       0.3453
## Detection Rate             0.1871        0.2698       0.2302
## Detection Prevalence       0.3237        0.3885       0.2878
## Balanced Accuracy          0.7801        0.7354       0.7894
```

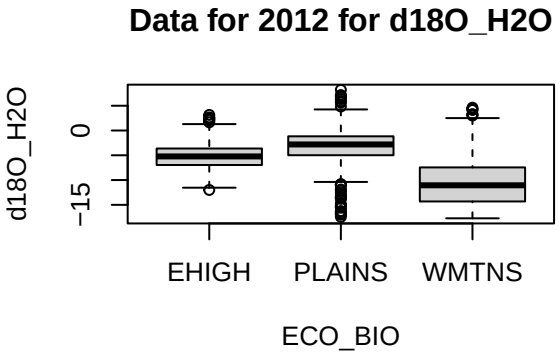
Visualizations,outlier detection and scaling

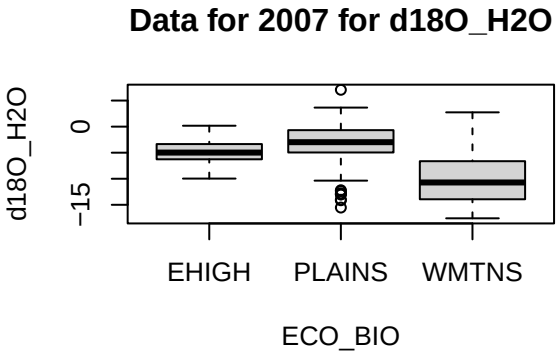
Basic EDA was performed using bar graphs, and box plots (above). The purpose of the box plots was for outlier detection. And there were no factors that were outlier free. In order to remove outliers the `IQR` method was used. (Interquartile range method)
Where a 'fence' was set up outside of Q1 and Q3. Anything outside of this fence was considered an outlier. The formula for calculating the 'fence' :
$((1.5 * IQR) - Q1)$
OR
$((1.5 * IQR) + Q3)$

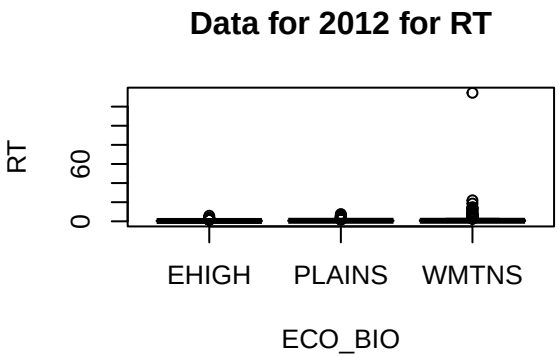
- IQR is the interquartile range. Which is obtained by subtracting Q1-Q3.

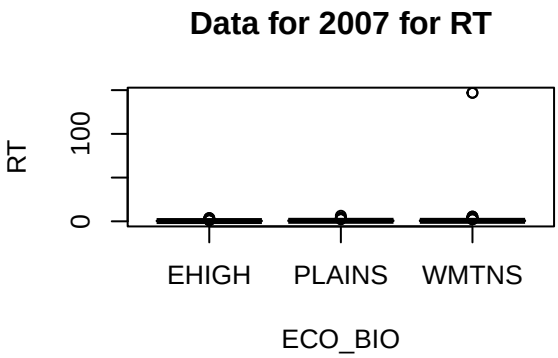
The data was also scaled which takes a value, subtracts it from the overall mean and divides by the standard deviation.
$\frac{X - \bar{(}X)}{sd}$



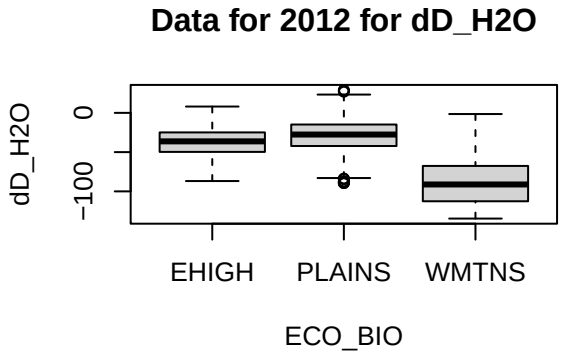

Data for 2012 for d18O_H2O
d18O_H2O
−15
0
EHIGH
PLAINS
WMTNS
ECO_BIO

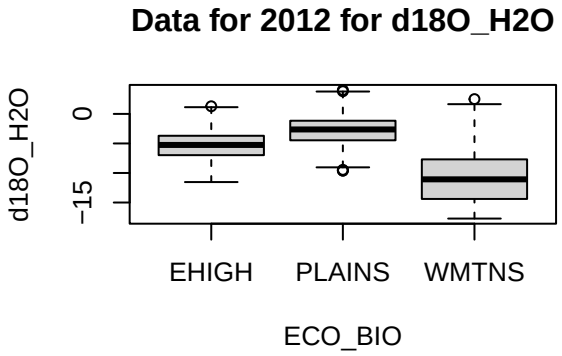

Data for 2007 for d18O_H2O
d18O_H2O
−15
0
EHIGH
PLAINS
WMTNS
ECO_BIO

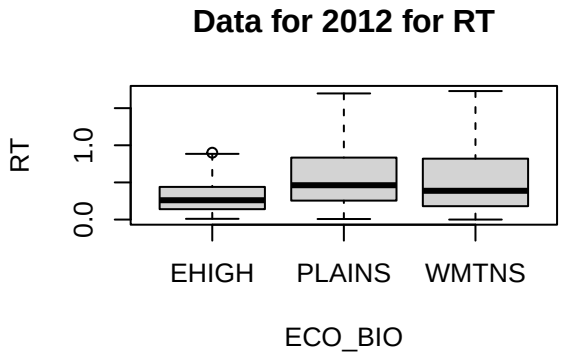

Data for 2012 for RT
RT
0
60
EHIGH
PLAINS
WMTNS
ECO_BIO

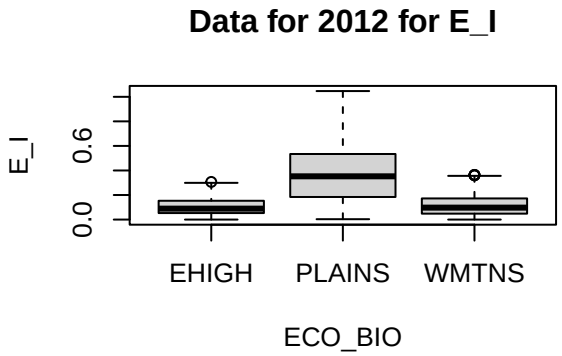

Data for 2007 for RT
RT
0
100
EHIGH
PLAINS
WMTNS
ECO_BIO

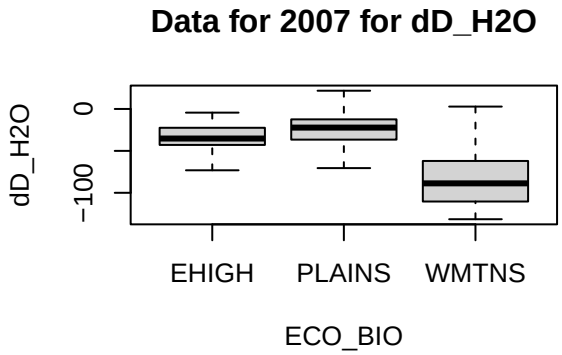
After Cleaning

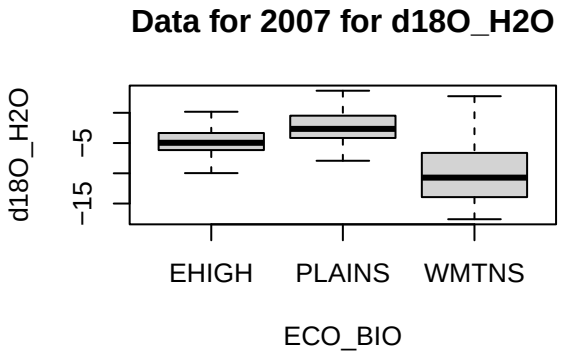


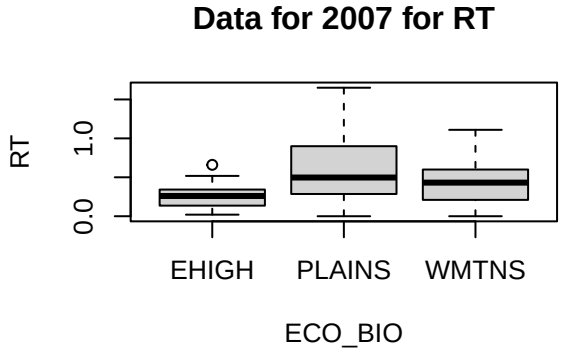


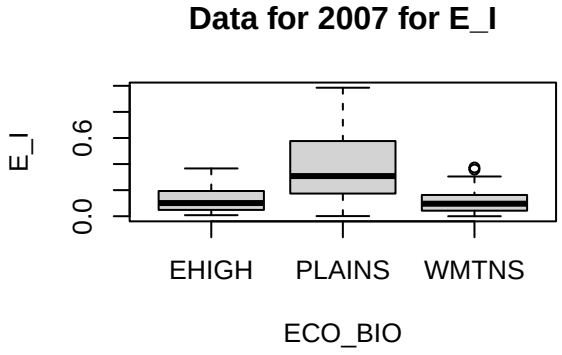


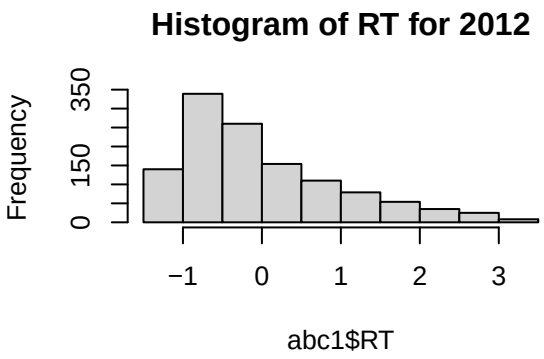


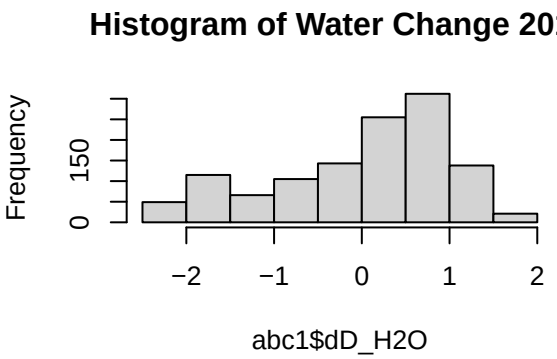


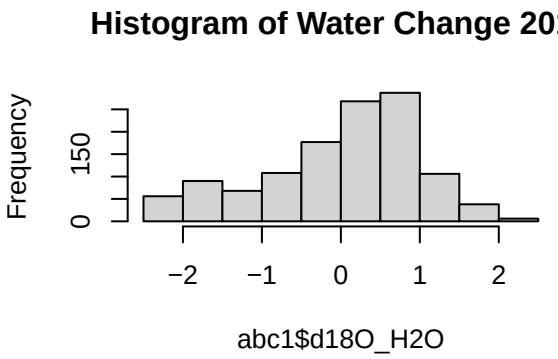


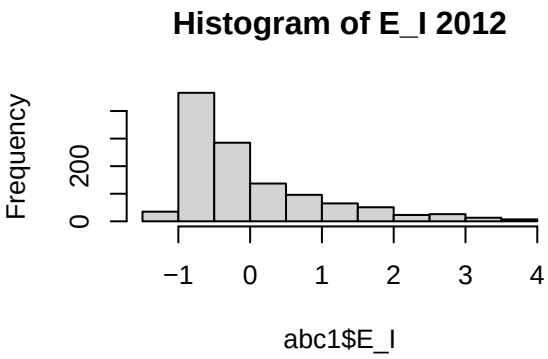


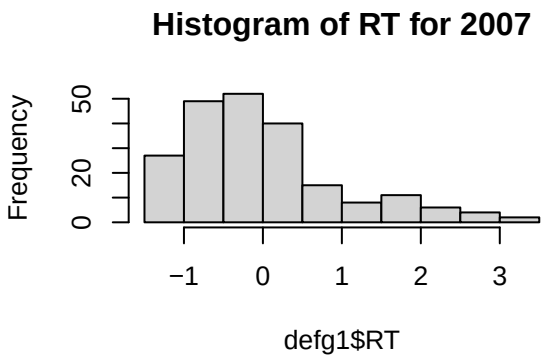


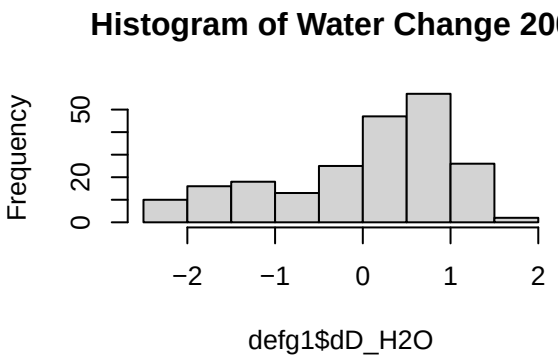

Histogram of RT for 2012
Frequency
0
150
350
−1
0
1
2
3
abc1$RT

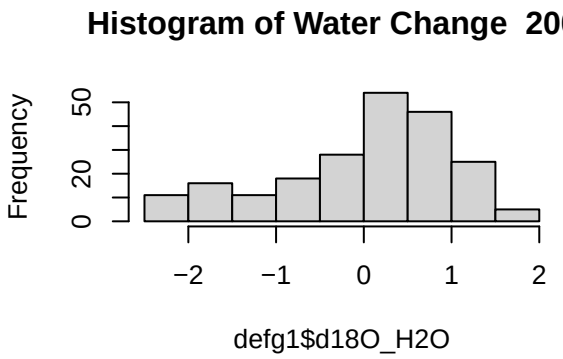

Histogram of Water Change 2012
Frequency
0
150
−2
−1
0
1
2
abc1$dD_H2O

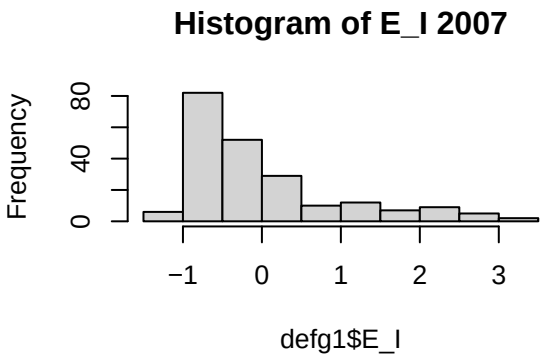

Histogram of Water Change 2012
Frequency
0
150
−2
−1
0
1
2
abc1$d18O_H2O


Histogram of E_I 2012
Frequency
0
200
−1
0
1
2
3
4
abc1$E_I









A random Forest Classifier.

None of the data follows a traditional normal distribution. Random forest classifiers are said to do well with data that fit into that category. They can be used for classification and regression. Are resistant to overfitting and can handle data with many features.

After outlier removal and scaling the same model performs 8% better than the original with a 73.8% accuracy score.

```
set.seed(50)
trControl <- trainControl(method = "cv",
    number = 5,
    search = "grid")


rf_default <- train(ECO_BIO~.,
    data = abc1,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl)



print(rf_default)
```

```
## Random Forest
##
## 1204 samples
##    4 predictor
##    3 classes: 'EHIGH', 'PLAINS', 'WMTNS'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 963, 964, 963, 964, 962
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7840890  0.6698408
##   3     0.7766165  0.6579887
##   4     0.7774568  0.6596456
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
prediction <-predict(rf_default, defg1)
confusionMatrix(prediction, defg1$ECO_BIO)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction EHIGH PLAINS WMTNS
##     EHIGH     39     21     5
##     PLAINS    10     64    10
##     WMTNS      3      7    55
##
## Overall Statistics
##
##                Accuracy : 0.7383
##                  95% CI : (0.674, 0.7959)
##     No Information Rate : 0.4299
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6024
##
##  Mcnemar's Test P-Value : 0.1768
##
## Statistics by Class:
##
##                      Class: EHIGH Class: PLAINS Class: WMTNS
## Sensitivity                0.7500        0.6957       0.7857
## Specificity                0.8395        0.8361       0.9306
## Pos Pred Value             0.6000        0.7619       0.8462
## Neg Pred Value             0.9128        0.7846       0.8993
## Prevalence                 0.2430        0.4299       0.3271
## Detection Rate             0.1822        0.2991       0.2570
## Detection Prevalence       0.3037        0.3925       0.3037
## Balanced Accuracy          0.7948        0.7659       0.8581
```

```r
set.seed(72)
trControl <- trainControl(method = "cv",
    number = 10,
    search = "grid")


rf_default <- train(ECO_BIO~.,
    data = abc1,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl)



print(rf_default)
```

```
## Random Forest
##
## 1204 samples
##    4 predictor
##    3 classes: 'EHIGH', 'PLAINS', 'WMTNS'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1084, 1084, 1084, 1083, 1085, 1083, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7932759  0.6835721
##   3     0.7949495  0.6860114
##   4     0.7882757  0.6759086
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```r
prediction <-predict(rf_default, defg1)
confusionMatrix(prediction, defg1$ECO_BIO)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction EHIGH PLAINS WMTNS
##     EHIGH     38     22     7
##     PLAINS    11     61     7
##     WMTNS      3      9    56
##
## Overall Statistics
##
##                Accuracy : 0.7243
##                  95% CI : (0.6592, 0.783)
##     No Information Rate : 0.4299
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.5831
##
##  Mcnemar's Test P-Value : 0.1376
##
## Statistics by Class:
##
##                      Class: EHIGH Class: PLAINS Class: WMTNS
## Sensitivity                0.7308        0.6630       0.8000
## Specificity                0.8210        0.8525       0.9167
## Pos Pred Value             0.5672        0.7722       0.8235
## Neg Pred Value             0.9048        0.7704       0.9041
## Prevalence                 0.2430        0.4299       0.3271
## Detection Rate             0.1776        0.2850       0.2617
## Detection Prevalence       0.3131        0.3692       0.3178
## Balanced Accuracy          0.7759        0.7578       0.8583
```

Tuning the tree

```
set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(1: 5))
rf_try <- train(ECO_BIO~.,
    data = abc1,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = trControl,
    importance = TRUE,
    nodesize = 14,
    ntree = 300)

print(rf_try)


max(rf_try$results$Accuracy)

best_mtry <- rf_try$bestTune$mtry
best_mtry
```

```
set.seed(45)
results_tree$values['60~Accuracy']
```

```
##    60~Accuracy
## 1    0.8083333
## 2    0.8016529
## 3    0.8595041
## 4    0.8235294
## 5    0.8166667
## 6    0.7394958
## 7    0.7520661
## 8    0.8016529
```

```
## 9    0.8016529
## 10   0.8264463
```

```
best_fiit1 <- train(ECO_BIO~.,
    abc1,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = trControl,
    importance = TRUE,
    nodesize = 14,
    ntree = 60,
    maxnodes = 40,
    )



prediction <-predict(best_fiit1, defg1)
confusionMatrix(prediction, defg1$ECO_BIO)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction EHIGH PLAINS WMTNS
##     EHIGH     47     26    12
##     PLAINS     5     64     6
##     WMTNS      0      2    52
##
## Overall Statistics
##
##                Accuracy : 0.7617
##                  95% CI : (0.6989, 0.8171)
##     No Information Rate : 0.4299
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6444
##
##  Mcnemar's Test P-Value : 3.256e-06
##
## Statistics by Class:
##
##                      Class: EHIGH Class: PLAINS Class: WMTNS
## Sensitivity                0.9038        0.6957       0.7429
## Specificity                0.7654        0.9098       0.9861
## Pos Pred Value             0.5529        0.8533       0.9630
## Neg Pred Value             0.9612        0.7986       0.8875
## Prevalence                 0.2430        0.4299       0.3271
## Detection Rate             0.2196        0.2991       0.2430
## Detection Prevalence       0.3972        0.3505       0.2523
## Balanced Accuracy          0.8346        0.8027       0.8645
```

Classification tree using rpart

Decision trees are supervised learning algorithms often used for classification or regression. Data is split based on Decision nodes with the output being leaf nodes.

```
library(rpart)
classifier <- rpart(formula = ECO_BIO~E_I+RT+dD_H2O, data = abc1,method='class')

y_pred <- predict(classifier, newdata=defg1,type='class')

cm = table(defg1$ECO_BIO, y_pred)

accuracy_Test <- sum(diag(cm)) / sum(cm)
accuracy_Test
```

```
## [1] 0.7476636
```

Support Vector Machines

A base model is below

Support vector Machines are great for multidimensional data. It partitions the data into n hyperplanes and places the data points in those planes. SVMs are great for multidimensional data.

```
svmfit = svm(ECO_BIO~., data=abc1, kernel = "linear", cost = 10, scale = FALSE)

print(svmfit)
```

```
##
## Call:
## svm(formula = ECO_BIO ~ ., data = abc1, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  539
```

```
y_pred = predict(svmfit, newdata = defg1[-1])
cm = table(defg1$ECO_BIO, y_pred)

accuracy_Test <- sum(diag(cm)) / sum(cm)
accuracy_Test
```

```
## [1] 0.7663551
```

Here is some model tuning

```
set.seed(42)
obj <- tune(svm, ECO_BIO~., data = abc1,
            ranges = list(gamma = c(0.01,0.1,1), cost = c(1:10),kernel=c("linear","radial"),epsilon=c(0
          tunecontrol = tune.control(sampling = "fix",nrepeat =3)
            )


summary(obj$best.parameters)
```

```
##      gamma          cost      kernel     epsilon
##  Min.   :0.1   Min.   :2   linear:0   Min.   :0.001
##  1st Qu.:0.1   1st Qu.:2   radial:1   1st Qu.:0.001
##  Median :0.1   Median :2              Median :0.001
##  Mean   :0.1   Mean   :2              Mean   :0.001
##  3rd Qu.:0.1   3rd Qu.:2              3rd Qu.:0.001
##  Max.   :0.1   Max.   :2              Max.   :0.001
```

Tuned model

```
set.seed(42)
svmfit = svm(ECO_BIO~., data=abc1, kernel = "radial", cost = 2, gamma=.1,epsilon = 0.001, scale = FALSE

print(svmfit)
```

```
##
## Call:
## svm(formula = ECO_BIO ~ ., data = abc1, kernel = "radial", cost = 2,
##     gamma = 0.1, epsilon = 0.001, method = "C-classification", shrinking = TRUE,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  2
##
## Number of Support Vectors:  559
```

```
y_pred = predict(svmfit, newdata = defg1[-1])
cm = table(defg1$ECO_BIO, y_pred)

accuracy_Test <- sum(diag(cm)) / sum(cm)
accuracy_Test
```

```
## [1] 0.7429907
```

Last but not least Naive Bayes

```r
NBclassfier=naiveBayes(ECO_BIO~., data=abc1)
print(NBclassfier)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     EHIGH    PLAINS     WMTNS
## 0.2707641 0.4318937 0.2973422
##
## Conditional probabilities:
##         RT
## Y              [,1]      [,2]
##   EHIGH  -0.4561600 0.5453785
##   PLAINS  0.2053951 1.0161710
##   WMTNS   0.1170466 1.1543790
##
##         E_I
## Y              [,1]      [,2]
##   EHIGH  -0.5662889 0.3375382
##   PLAINS  0.7112057 1.1148735
##   WMTNS  -0.5173654 0.4013079
##
##         dD_H2O
## Y              [,1]      [,2]
##   EHIGH   0.3518994 0.4508867
##   PLAINS  0.6097512 0.5759431
##   WMTNS  -1.2061168 0.7674713
##
##         d18O_H2O
## Y              [,1]      [,2]
##   EHIGH   0.1431418 0.4581564
##   PLAINS  0.6999079 0.5505544
##   WMTNS  -1.1469730 0.8285648
```

```r
predicted.classes <- NBclassfier %>% predict(defg1)

mean(predicted.classes == defg1$ECO_BIO)
```

```
## [1] 0.7429907
```

Results and conclusion

All models performed similarly.

Model	Result
Random Forest Classifier	76%
Classification Tree	75%
SVM	76%
Naive Bayes	74%

There is definitely more tuning I can perform, since each tuned model performed only marginally better than a model tested against uncleaned data. For a first test run using these Algorithms against this data set, they performed well.