A Machine Learning Approach to Predict Forest Fires using Meteorological Data

BIOSTAT 273: Final Project

Chad Pickering

12/14/2018

Spatial, temporal, and weather-related data and indexes were collected from the Montesinho Natural Park in northeastern Portugal between 2000 and 2003. 517 forest fires were recorded. The variables associated with each fire are:

- X - x-axis spatial coordinate within the Montesinho park map: 1 to 9
- Y - y-axis spatial coordinate within the Montesinho park map: 2 to 9
- month - month of the year: "jan" to "dec"
- day - day of the week: "mon" to "sun"
- FFMC - FFMC index from the FWI system: 18.7 to 96.20
- DMC - DMC index from the FWI system: 1.1 to 291.3
- DC - DC index from the FWI system: 7.9 to 860.6
- ISI - ISI index from the FWI system: 0.0 to 56.10
- temp - temperature in Celsius degrees: 2.2 to 33.30
- RH - relative humidity in %: 15.0 to 100
- wind - wind speed in km/h: 0.40 to 9.40
- rain - outside rain in mm/m2 : 0.0 to 6.4
- area - the burned area of the forest (in ha): 0.00 to 1090.84

I will attempt to predict a fire's area by these predictors using various ML techniques. The continuous response this output variable is very skewed towards 0, so I will make appropriate transformations. I will also categorize the area variable into two bins, small and large, later on to see how binary classification using the same ML techniques performs.

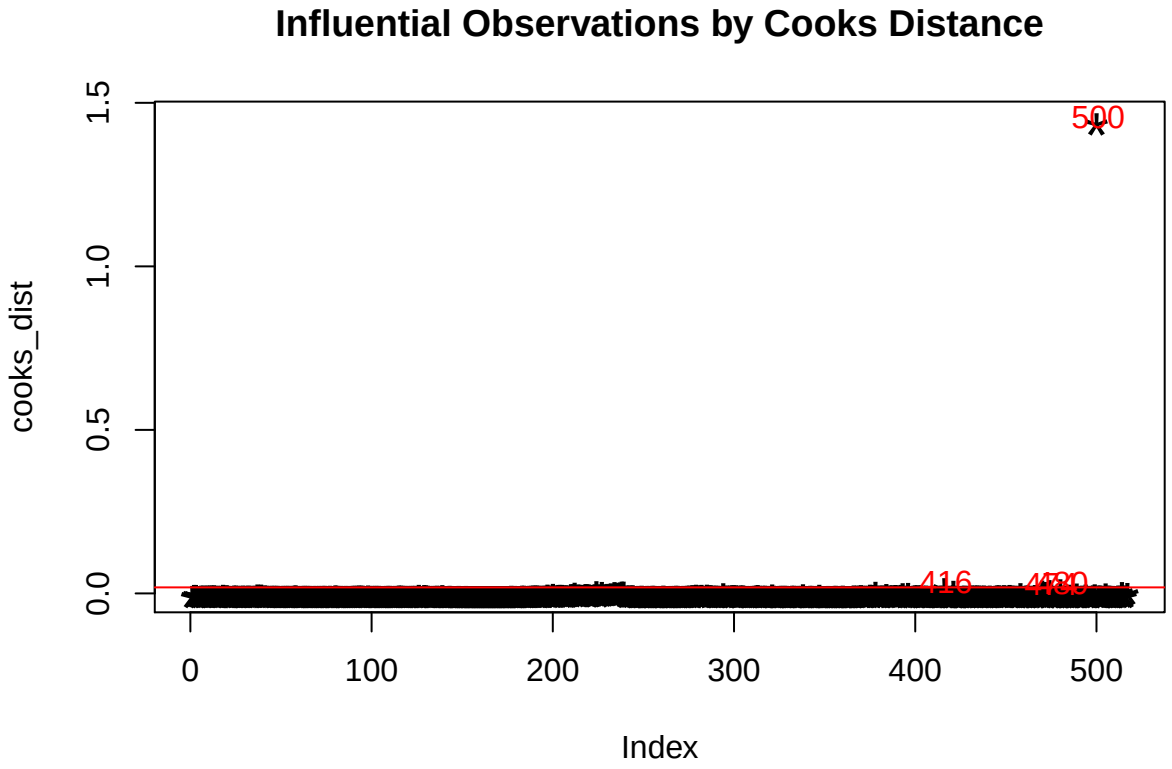
Multiple regression with log transformed response.

First, data exploration is necessary. Because of the uneven distribution of fires in the months of the year, I make a new season variable. I then use a logarithmic transformation on the highly right skewed response variable and run an initial regression to see what variables seem significant and predictive of burned area. No transformations on the covariates largely assisted in improving violation of regression assumptions, so none were applied.

```
##
## Call:
## lm(formula = logarea ~ X + Y + season + day + FFMC + DMC + DC +
##     ISI + temp + RH + wind + rain, data = fires_lr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6788 -1.0826 -0.5859  0.8658  5.5271
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4212961  1.5364990   0.274   0.7841
## X             0.0396106  0.0323586   1.224   0.2215
```

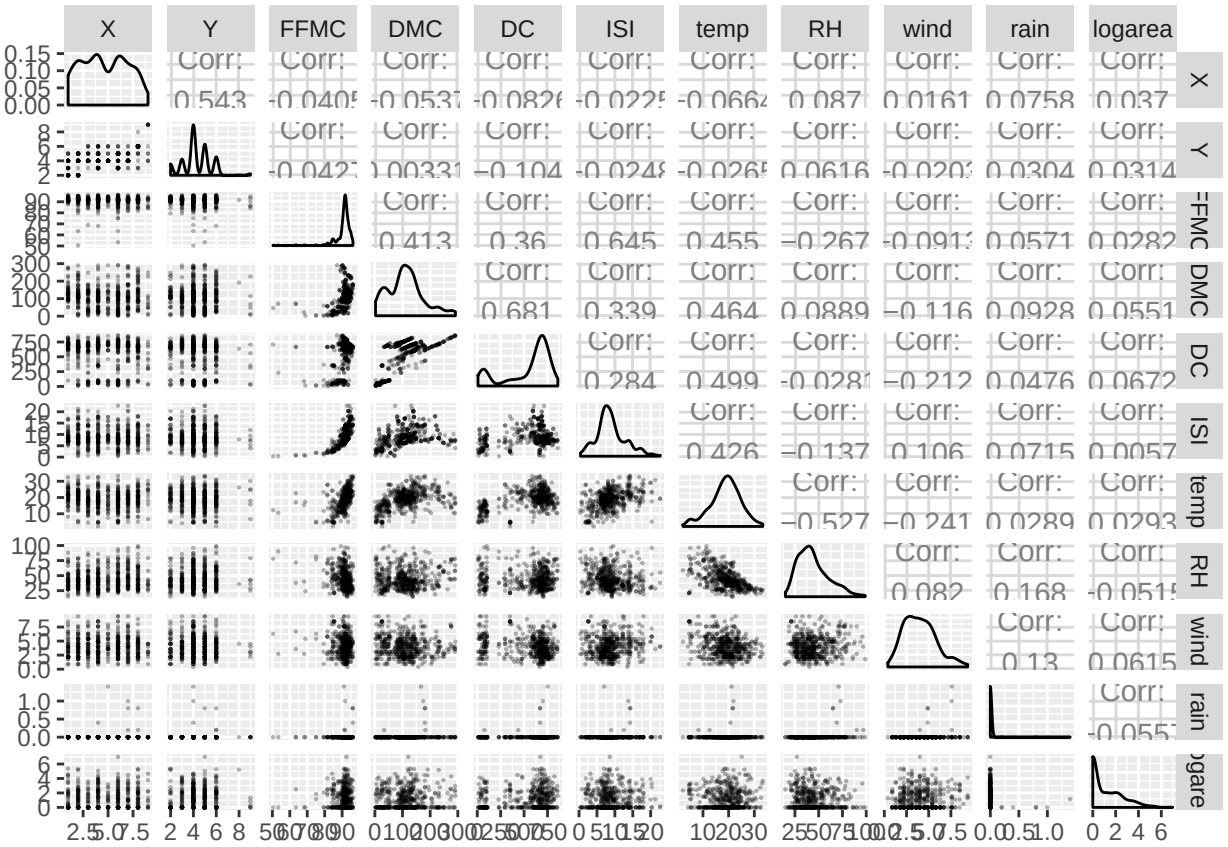
```
## Y             0.0116870  0.0608048   0.192   0.8477
## seasonfall    0.1926127  0.3594103   0.536   0.5923
## seasonwinter -0.4232877  0.4256814  -0.994   0.3205
## seasonspring -0.2139915  0.3997712  -0.535   0.5927
## daytue        0.1115573  0.2463215   0.453   0.6508
## daywed        0.0064840  0.2565130   0.025   0.9798
## daythu       -0.1257025  0.2469961  -0.509   0.6110
## dayfri       -0.1027026  0.2281305  -0.450   0.6528
## daysat        0.1438318  0.2263286   0.635   0.5254
## daysun        0.0584462  0.2237162   0.261   0.7940
## FFMC          0.0082109  0.0146608   0.560   0.5757
## DMC           0.0017685  0.0015390   1.149   0.2510
## DC           -0.0002783  0.0006626  -0.420   0.6747
## ISI          -0.0252158  0.0172994  -1.458   0.1456
## temp         -0.0025071  0.0196695  -0.127   0.8986
## RH           -0.0066355  0.0057460  -1.155   0.2487
## wind          0.0700542  0.0376450   1.861   0.0633 .
## rain          0.0712751  0.2162614   0.330   0.7419
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.4 on 497 degrees of freedom
## Multiple R-squared:  0.03436,    Adjusted R-squared:  -0.002561
## F-statistic: 0.9306 on 19 and 497 DF,  p-value: 0.5446
```

We see a rather dismal result, with global insignificance and no variables besides wind indicating any kind of suggestive assocation with fire size. I then use Cook's distance (with a threshold of 4 times the mean distance) to remove 6 of the 517 observations that had anomalous values in the domain space of the predictors.



Now with outlier observations removed and appropriate transformations made, we can take a look at the scatterplot matrix, distributions, and correlations for every variable pair. Let's take note of every pair with

an absolute correlation of over 0.5 (X, Y; DC, DMC; ISI, FFMC; RH, temp) and adjust for interactions in the next regression.




```
##
## Call:
## lm(formula = logarea ~ X + Y + season + day + FFMC + DMC + DC +
##     ISI + temp + RH + wind + rain + X * Y + DC * DMC + ISI *
##     FFMC + RH * temp, data = fires_lr2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7267 -1.0506 -0.5517  0.8318  5.5095
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.260e-01  2.060e+00  -0.304   0.7614
## X             8.112e-02  8.594e-02   0.944   0.3457
## Y             9.254e-02  1.110e-01   0.834   0.4047
## seasonfall    1.117e-01  3.546e-01   0.315   0.7529
## seasonwinter -4.925e-01  4.572e-01  -1.077   0.2819
## seasonspring -3.001e-01  4.114e-01  -0.729   0.4661
## daytue        2.099e-01  2.428e-01   0.865   0.3877
## daywed        7.673e-02  2.533e-01   0.303   0.7621
## daythu       -1.676e-01  2.457e-01  -0.682   0.4955
## dayfri       -1.001e-02  2.270e-01  -0.044   0.9648
## daysat        1.820e-01  2.240e-01   0.812   0.4170
## daysun        1.618e-01  2.216e-01   0.730   0.4655
```

```
## FFMC          1.536e-02  1.987e-02   0.773   0.4399
## DMC          -5.773e-05  5.372e-03  -0.011   0.9914
## DC           -2.694e-04  8.391e-04  -0.321   0.7483
## ISI          -2.752e-01  4.923e-01  -0.559   0.5764
## temp          1.398e-02  3.248e-02   0.430   0.6672
## RH            3.504e-03  1.129e-02   0.310   0.7563
## wind          7.499e-02  3.812e-02   1.967   0.0497 *
## rain         -8.639e-01  6.964e-01  -1.240   0.2154
## X:Y          -1.315e-02  1.845e-02  -0.713   0.4761
## DMC:DC        2.190e-06  6.984e-06   0.314   0.7540
## FFMC:ISI      2.590e-03  5.176e-03   0.500   0.6171
## temp:RH      -7.001e-04  6.661e-04  -1.051   0.2938
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.364 on 487 degrees of freedom
## Multiple R-squared:  0.03932,    Adjusted R-squared:  -0.006052
## F-statistic: 0.8666 on 23 and 487 DF,  p-value: 0.6447
```

Adjusting for all other covariates and interactions we can see that wind is positively associated with larger fire size at the 0.05 level, but nothing else is significant. We continue with stepwise regression to narrow down the predictors such that we have a model with the lowest AIC value.

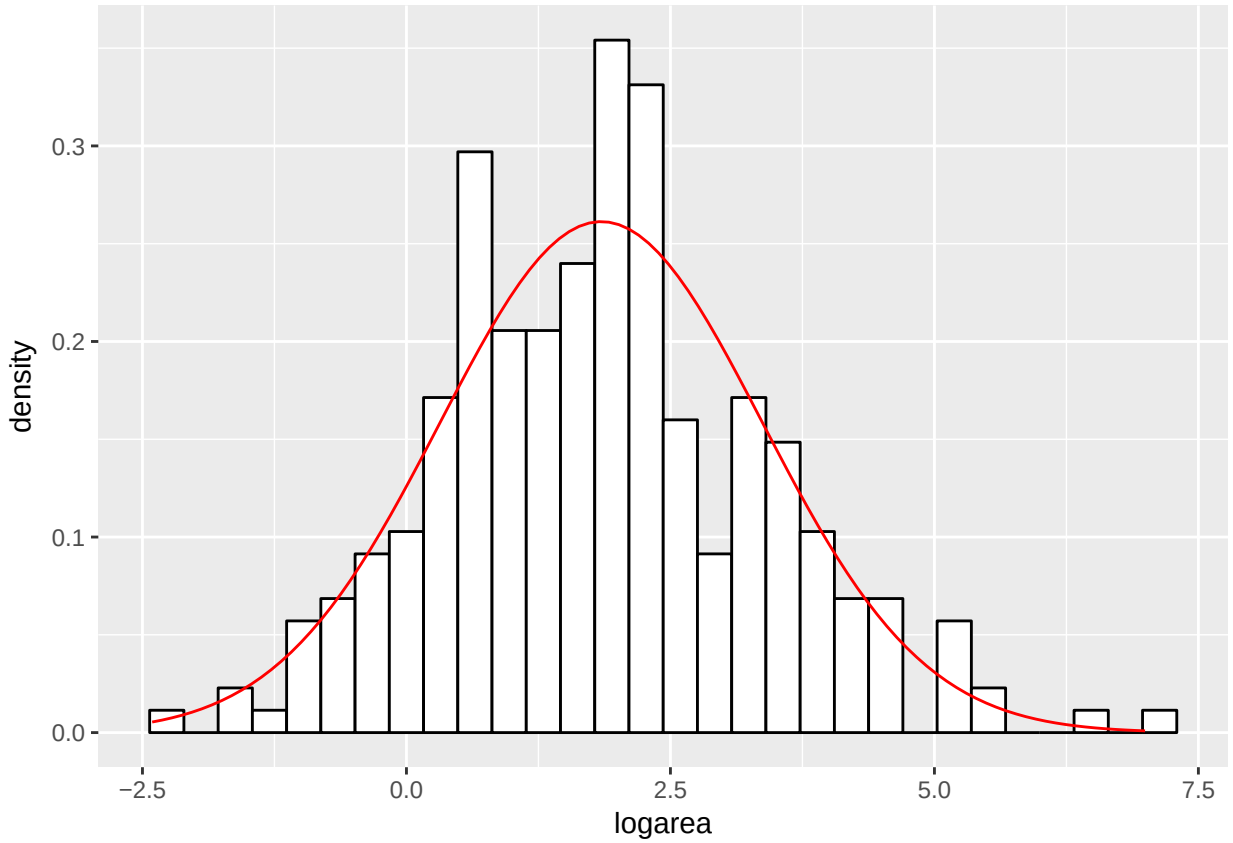
The model with the lowest AIC after variable selection has the DC index, wind, and rain as the sole predictors (AIC reduced from 341.01 to 313.51). This model is marginally significant, with all covariates suggesting association with fire size at the 0.05 level. 5000 RMSEs generated on this model averages out to around 40.7. The model itself is insufficient for the structure of the response variable becuase it allows for negative area and does not capture the right skewed structure still seen even with the log transform. We need to model the zeroes, which make up nearly half of the response, separately to improve prediction power.

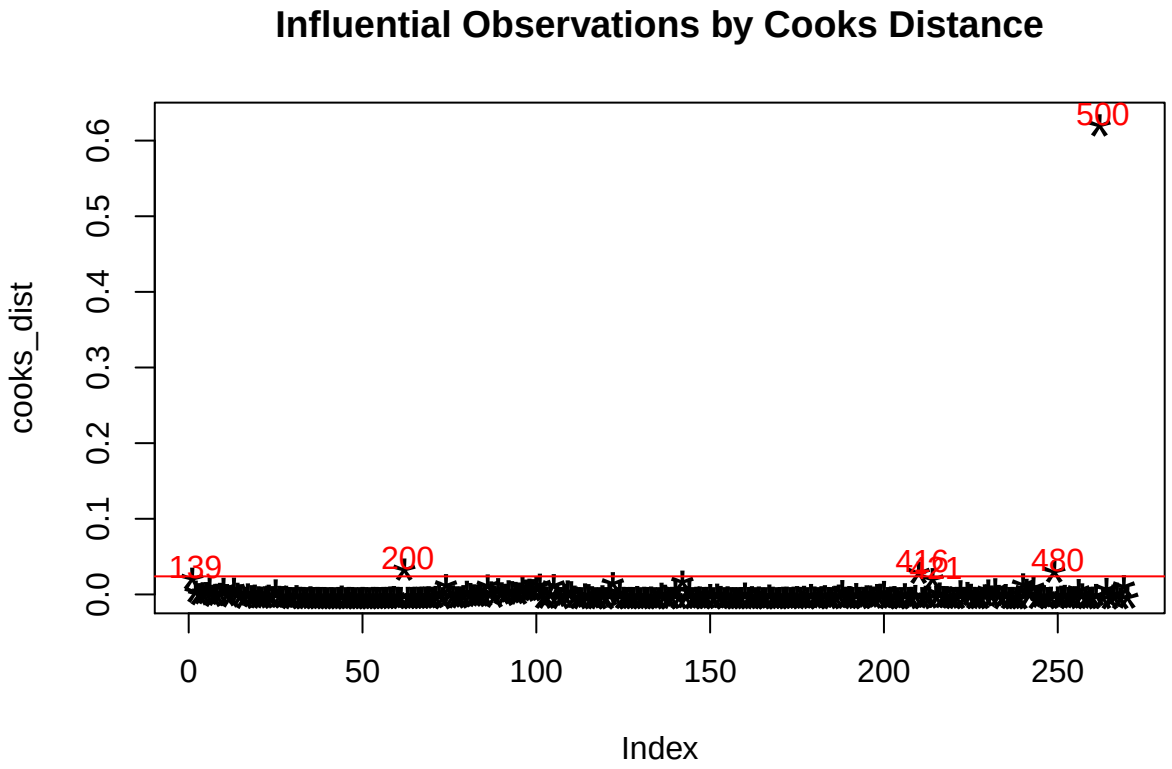
```
##
## Call:
## lm(formula = logarea ~ DC + wind + rain, data = fires_lr2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5011 -1.0782 -0.6488  0.8967  5.8389
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5532754  0.2223553   2.488   0.0132 *
## DC           0.0004916  0.0002481   1.981   0.0481 *
## wind         0.0679869  0.0345261   1.969   0.0495 *
## rain        -1.0440871  0.6502476  -1.606   0.1090
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.354 on 507 degrees of freedom
## Multiple R-squared:  0.01552,    Adjusted R-squared:  0.009691
## F-statistic: 2.664 on 3 and 507 DF,  p-value: 0.04734
```

Multiple regression with two-part models.

Now, we will consider three models, each with two parts. The zeroes will be modeled separately as a logistic regression problem, and the non-zeroes will be modeled with normal, log-normal, or gamma distributional

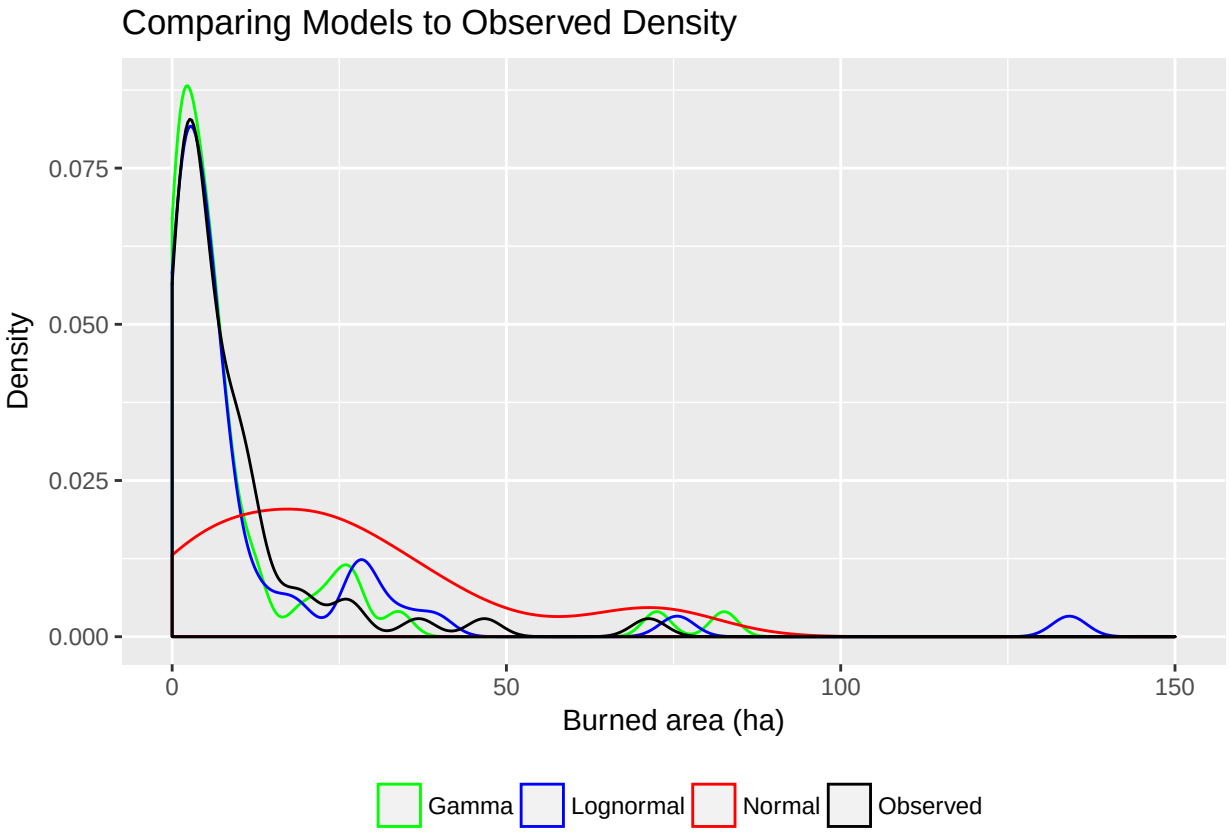
assumptions. The logistic/log-normal model performs the best in terms of average RMSE value after 5000 randomly generated divisions of 80% training and 20% test sets.





```
##      OLS  Log OLS    Gamma
## 41.94905 41.03687 42.15004
```

Finally, I wanted to simulate values from each of the distributions fit to the data and graphically see which model(s) predict the real values the best.



As expected, the logistic/normal model performs the worst of the three because burned area is highly right skewed and allows for negative area. The logistic/gamma performs better than logistic/normal but not as well as logistic/lognormal upon many repetitions; logistic/lognormal seems to predict the distribution of burned area the best, particularly when it comes to smaller fires. Average RMSE values do not improve from the original model because large fires are still not predicted well at all, and these deviations between observed and predicted burned areas still add large squared residuals that inflate overall prediction error.

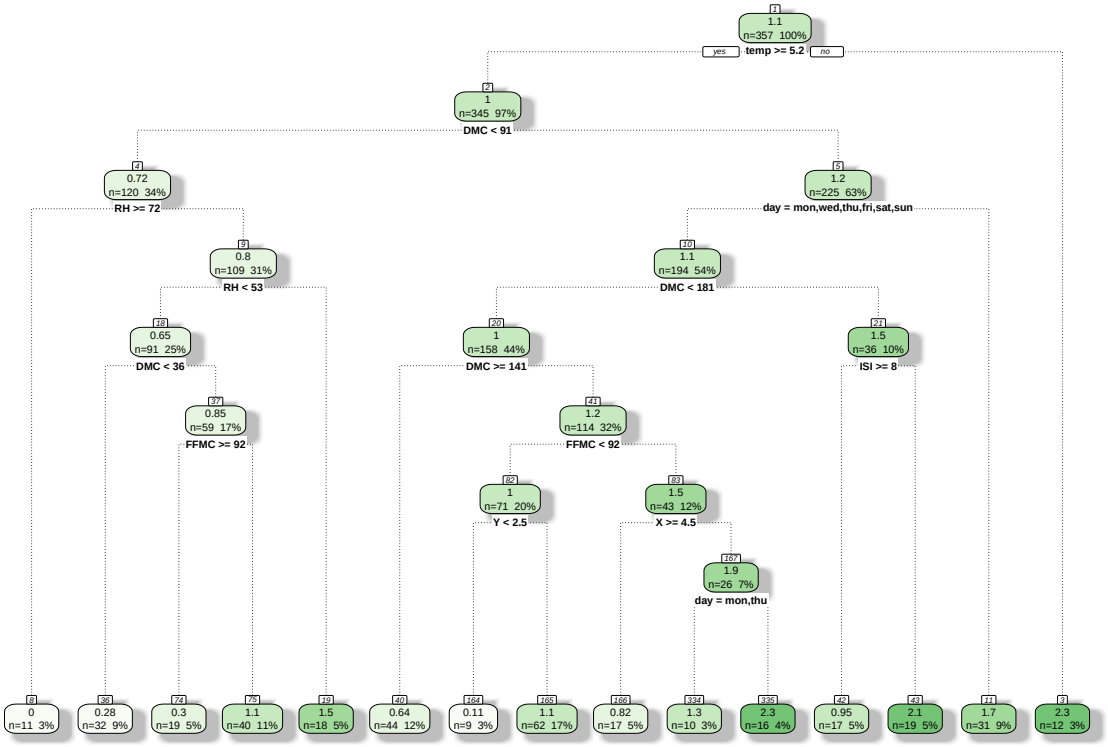
Aside: Principal Component Analysis.

I wanted to confirm my suspicions that this data cannot be described by much less principal components than there are individual predictors. I was correct - 8 out of the 10 PCs are needed to explain 95% of the data's variability. There is no simple extractible structure in this data, or rather, PCA cannot find one.

```
## Importance of components:
##                           PC1    PC2   PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7232 1.2533 1.162 1.1047 0.92768 0.83854 0.68309
## Proportion of Variance 0.2969 0.1571 0.135 0.1220 0.08606 0.07031 0.04666
## Cumulative Proportion  0.2969 0.4540 0.589 0.7111 0.79713 0.86745 0.91411
##                           PC8     PC9    PC10
## Standard deviation     0.6058 0.54363 0.44314
## Proportion of Variance 0.0367 0.02955 0.01964
## Cumulative Proportion  0.9508 0.98036 1.00000
```

Decision Trees with Continuous Response.

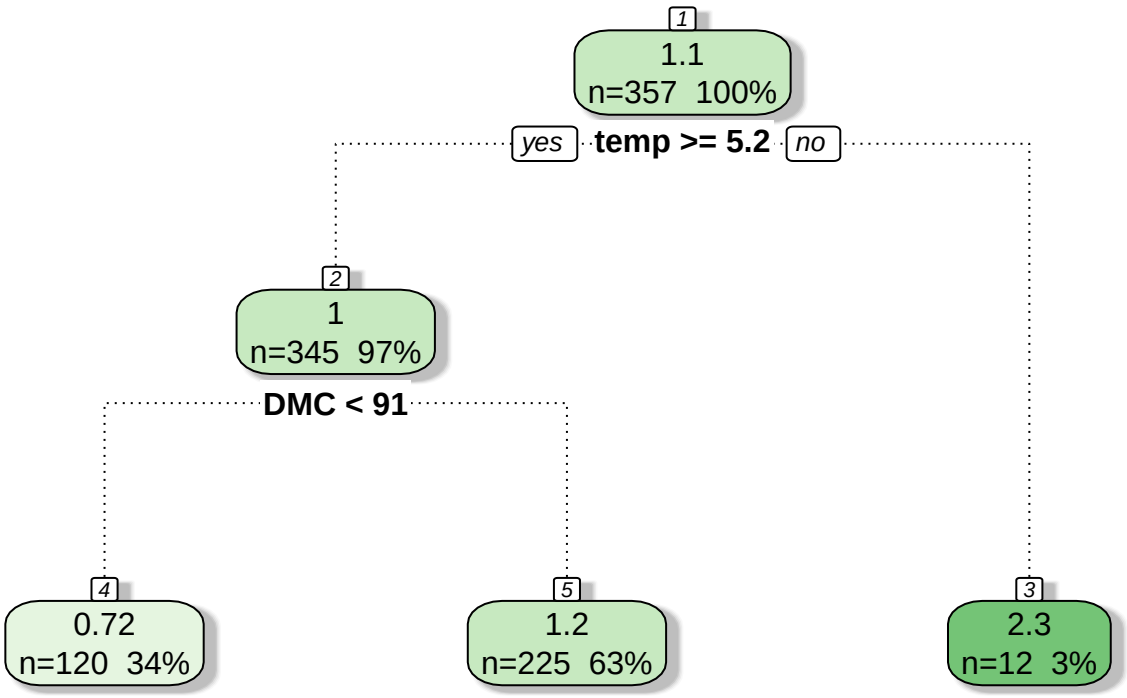
Now, let's use some machine learning techniques to predict burned area, the first being one of the most straightforward, the decision tree, which makes no assumptions about the relationship between the outcome and the covariates. With a continuous response, the tree itself becomes a binary regression tool of sorts, where it finds the optimal places to split on within each of the most predictive explanatory variables to decrease overall error. In the unpruned tree (RMSE = 1.49), the variables day, DC, ISI, RH, temp, and Y are used. If the tree is pruned by looking at the complexity parameter at which the minimum cross validation error is achieved, the tree has only one node, so I take the next highest error rate, which yields five (two splits; see below) - only temp and the DMC index are used (RMSE = 1.43).



Rattle 2018−Dec−14 12:10:14 cepickering

```
##
## Regression tree:
## rpart(formula = form.build("logarea", xvars), data = train.set,
##     method = "anova")
##
## Variables actually used in tree construction:
## [1] day  DMC  FFMC ISI  RH   temp X    Y
##
## Root node error: 640.84/357 = 1.7951
##
## n= 357
##
##         CP nsplit rel error  xerror     xstd
## 1 0.028332      0   1.00000 1.00215 0.081009
## 2 0.015463      2   0.94334 0.99098 0.085713
## 3 0.014576      3   0.92787 1.08572 0.094787
## 4 0.014216      8   0.85462 1.11994 0.096770
## 5 0.013213     10   0.82618 1.11812 0.097014
```

```
## 6 0.012138     11   0.81297 1.14697 0.098488
## 7 0.010416     13   0.78870 1.15834 0.097463
## 8 0.010000     14   0.77828 1.17744 0.100611
```



Interestingly, in the pruned tree, cooler temperatures tend to predict larger fires and a lower DMC index value splits the smaller fires from the rest.

Random Forests with Continuous Response.

Next, we see how the performance of a random forest regressor compares. In regression problems such as this, the predicted value of the response is a weighted mean of the values predicted by each unpruned decision tree generated. Each tree overfits quite a lot, but since they are all different, the errors they make are in all directions, so averaging creates a generally strong classifier when they are put together as an ensemble.

```
##         If absent, % increase MSE
## temp                      11.468
## DMC                       11.203
## DC                         9.895
## season                     5.343
## rain                       4.949
## ISI                        3.266
## FFMC                       2.897
## Y                          2.171
## RH                        -0.839
## day                       -0.952
## wind                      -1.978
## X                         -5.498
```

We see that if RH, day, wind, and X are removed, RMSE may decrease. In the final random forest with 8 covariates, RMSE remains very stable at about 1.46.

```
##        If absent, % increase MSE
## DMC                       17.134
## temp                      15.319
## DC                        14.867
## season                    13.314
## rain                       7.838
## FFMC                       6.404
## ISI                        3.840
## Y                          1.429
```

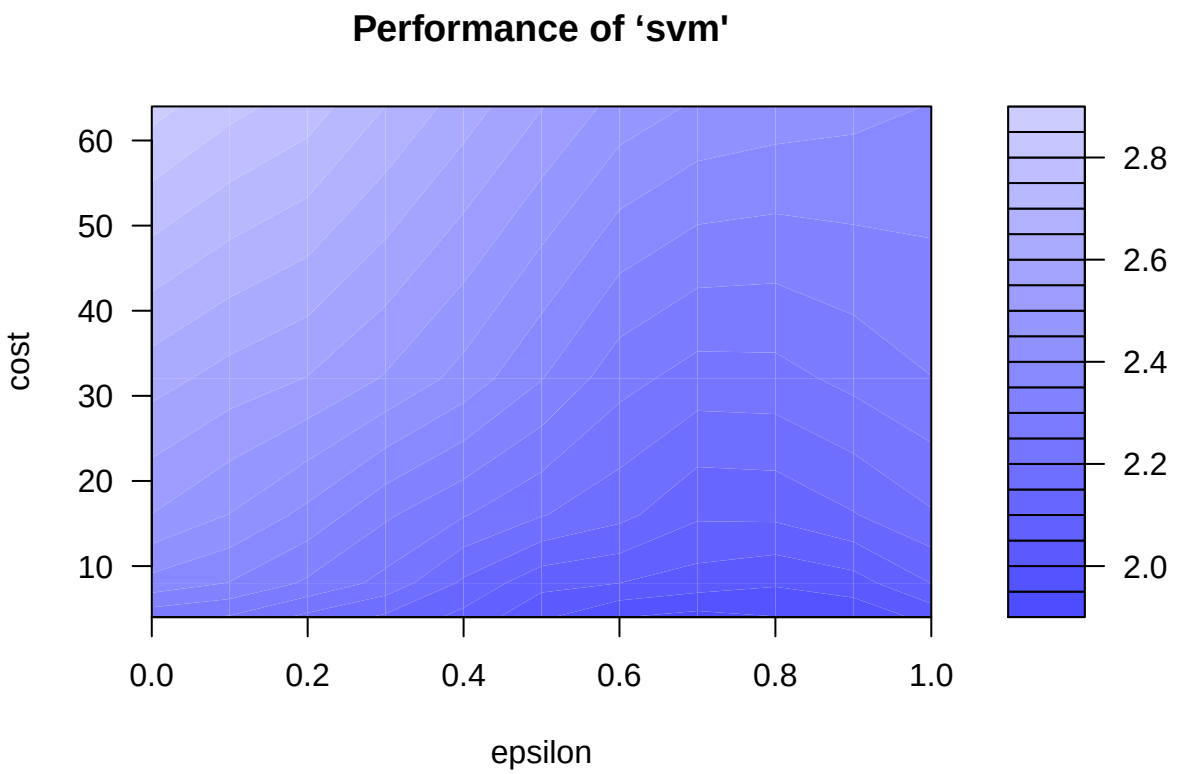
So, we can see that the RMSE of the random forest is actually very close to the pruned decision tree, regardless of if the 4 variables that would increase MSE are in the model or not.

Support Vector Regressor with Continuous Response.

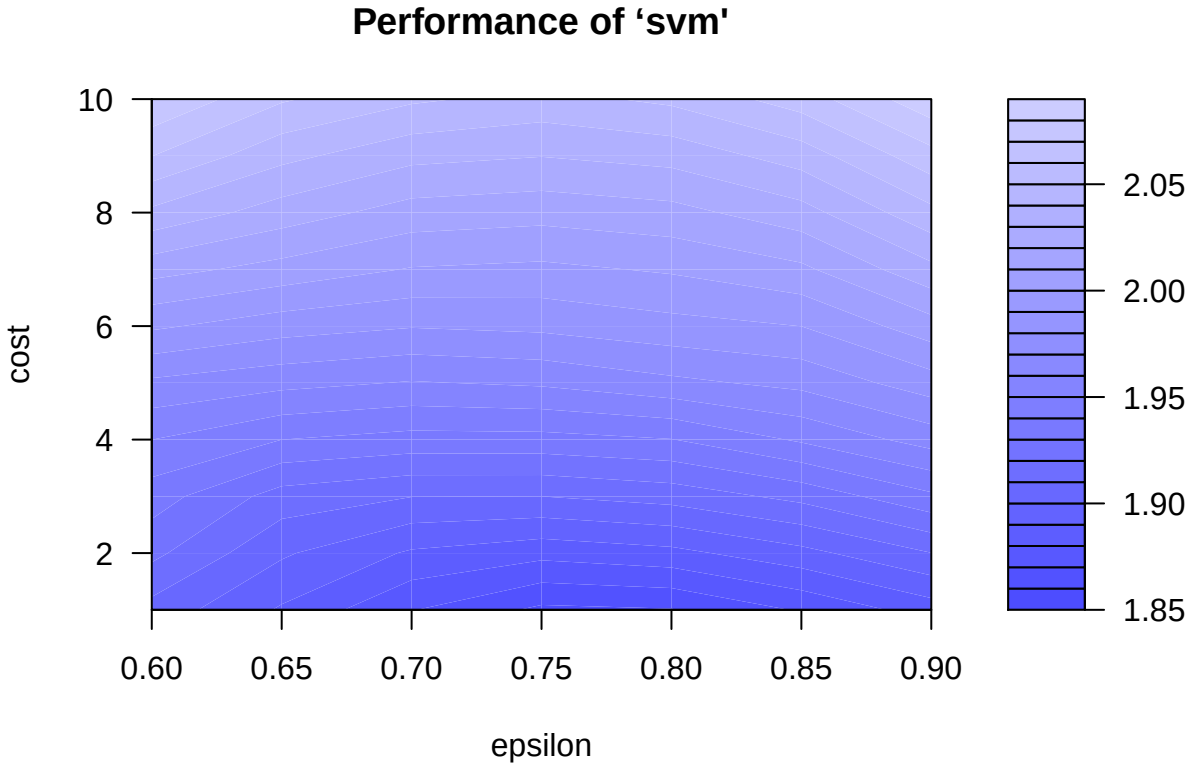
Finally, we will use a support vector regressor - choosing optimal non-linear hyperplanes to divide the multi-dimensional predictor space into sections to better predict the continuous outcome. We start with a default epsilon parameter of 0.1 and cost parameter of 1 - this gives an RMSE of about 1.32.

When tuning the SVR to select the best model, we vary the values of epsilon and the cost parameter which helps avoid overfitting. We see that the best pair after a comprehensive grid search is an epsilon value of 0.7 and a cost parameter of 4 (see plot as well). The RMSE is about 1.39 with this model, a bit higher than the original.

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##      0.7    4
##
## - best performance: 1.93297
```



Now, zoom in on the darker area of the plot where MSE is the lowest and refine the best pair to reduce RMSE. It looks like it is an epsilon value of 0.75 and a cost parameter of 1, yielding a final RMSE of 1.30 with 280 support vectors generated, a hair better than the original SVR.



##

```
## Call:
## best.tune(method = svm, train.x = form.build("logarea", xvars),
##     data = fires_cont, ranges = list(epsilon = seq(0.6, 0.9,
##         0.05), cost = seq(1, 10, 1)))
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05
##     epsilon:  0.75
##
##
## Number of Support Vectors:  280
```

Conclusion.

We conclude that the support vector regressor with an epsilon value of 0.75 and cost parameter of 1 does the best job at predicting the area of fire burn area. See supplementary code and analysis in the Jupyter Notebook PDF, where I discuss which types fires are best predicted. It is probable that more variables are needed to analyze this outcome properly, such as if the land had burned recently and what kind of vegetation is in the vicinity - accuracy of large burn area prediction may improve drastically with this information.

Code Appendix.

```
library(ggplot2)
library(MASS)
library(data.table)
library(GGally)
library(stats)
library(rpart)
library(dplyr)
library(randomForest)
library(caret)
library(e1071)
library(rattle)
fires <- read.csv("~/Documents/Syncplicity/ChadSync/BIOSTATISTICS/Fall 2018/273/forestfires.csv")
colnames(fires) <- c("X", "Y", "month", "day",
                     "FFMC", "DMC", "DC", "ISI", "temp",
                     "RH", "wind", "rain", "area")

# Combine months into seasons since frequencies are very unbalanced. Relevel day.
fires <- fires %>% mutate(season = case_when(
  month %in% c("jan", "feb", "mar") ~ "winter",
  month %in% c("apr", "may", "jun") ~ "spring",
  month %in% c("jul", "aug", "sep") ~ "summer",
  month %in% c("oct", "nov", "dec") ~ "fall",
  TRUE ~ NA_character_
))

fires$season <- factor(fires$season,
```

```
                      levels=c("summer", "fall", "winter", "spring"))

fires$day <- factor(fires$day,
                    levels=c("mon", "tue", "wed", "thu", "fri", "sat", "sun"))
# Duplicate dataset for the basic LR model.
fires_lr <- fires
fires_lr$logarea <- log(fires_lr$area+1) # this transform is clear at the start

fires_basiclr <- lm(logarea ~ X + Y + season + day +
                      FFMC + DMC + DC + ISI + temp +
                      RH + wind + rain, data = fires_lr)
summary(fires_basiclr)
# Decide on outliers and remove them.
cooks_dist <- cooks.distance(fires_basiclr)

plot(cooks_dist, pch="*", cex=2, main="Influential Observations by Cooks Distance")
abline(h = 4*mean(cooks_dist, na.rm=TRUE), col="red")
text(x = 1:length(cooks_dist)+1, y = cooks_dist+0.01,
     labels = ifelse(cooks_dist > 4*mean(cooks_dist, na.rm=TRUE),
                     names(cooks_dist), ""), col="red")
influential <- which(cooks_dist > 4*mean(cooks_dist, na.rm=TRUE))
# fires_lr[influential,] # Obs. 416, 474, 480, 500
fires_lr2 <- fires_lr[-influential,]

# Look at extreme valued members of variables and decide if reasonable to remove.
# fires_lr2[abs(fires_lr2$rain-mean(fires_lr2$rain, na.rm=TRUE))/sd(fires_lr2$rain, na.rm=TRUE)>2,]

# FFMC - obs 380
# ISI - obs 23
# rain - decide all are reasonable, just do log transform
fires_lr2 <- fires_lr2[-c(23, 380),]
# See scatterplot/correlation matrix
ggpairs(fires_lr2[,c("X", "Y", "FFMC", "DMC", "DC", "ISI",
                 "temp", "RH", "wind", "rain", "logarea")],
        lower = list(continuous = wrap("points", alpha = 0.3, size=0.1)))
# With 6 extreme obs removed, no significant difference of conclusions.
fires_basiclr2 <- lm(logarea ~ X + Y + season + day +
                      FFMC + DMC + DC + ISI + temp +
                      RH + wind + rain + X*Y + DC*DMC +
                      ISI*FFMC + RH*temp, data = fires_lr2)
summary(fires_basiclr2)
# Do stepwise regression.
step(fires_basiclr2, direction = "both")
# Full model AIC: 341.01
# Final model AIC (DC + wind + rain): 313.51
fires_lrfinal <- lm(formula = logarea ~ DC + wind + rain, data = fires_lr2)
summary(fires_lrfinal)
# Generate average RMSE for final model with 3 covariates.

# RMSE function.
RMSE <- function(x, y){ sqrt(mean((y - x)^2, na.rm = TRUE)) }

nreps <- 5000
```

```r
df <- matrix(nrow=nreps, ncol=1)
for(i in 1:nreps){

  tryCatch({
    # Randomly shuffle rows and make training (80%) and test (20%) sets.
    shuffled <- fires_lr2[sample(nrow(fires_lr2)),]
    train_data <- shuffled[1:round(0.8 * nrow(fires_lr2)),]
    test_data <- shuffled[(round(0.8 * nrow(fires_lr2)) + 1):nrow(fires_lr2),]

    ## For log OLS fit of final stepwise reg model:
    first.fit <- lm(logarea ~ DC + wind + rain, train_data)

    pred <- as.data.frame(test_data['area'], colname = 'area') # actual
    colnames(pred)[1] <- "area"
    pred$logols <- exp(predict(first.fit, test_data) + summary(first.fit)$sigma^2/2)

    ## Prediction accuracy via RMSE:
    df[i,] <- RMSE(pred$area, pred$logols)
  }, error = function(e){})
}

df <- as.data.frame(df)
na.rows <- sum(is.na(df[,1]))
rmse_avgs <- colSums(df, na.rm=TRUE)/(nreps-na.rows)
names(rmse_avgs) <- "Log OLS RMSE"
rmse_avgs
# Area has a mode at 0; extremely right skewed. Make log(area) transform. For now, area = 0 -> NA
fires$logarea <- ifelse(fires$area > 0, log(fires$area), NA)
ggplot(fires, aes(x = logarea)) +
  geom_histogram(aes(y = ..density..), colour = "black", fill = "white") +
  stat_function(fun = dnorm, args = list(mean = mean(fires$logarea, na.rm=TRUE),
                                         sd = sd(fires$logarea, na.rm=TRUE)), col = 'red')
# Create dataset with only area>0 observations
pos_fires <- fires[which(fires$area > 0),]

# Examine and plot Cook's Distance for full model. Remove outliers.
# These outliers are different because we set logarea = NA if area = 0.
model <- lm(logarea ~ X + Y + season + day +
              FFMC + DMC + DC + ISI + temp +
              RH + wind + rain, data = pos_fires)
cooks_dist <- cooks.distance(model)

influential <- which(cooks_dist > 4*mean(cooks_dist, na.rm=TRUE))
# pos_fires[influential,] # Obs. 139, 200, 416, 421, 480, 500

plot(cooks_dist, pch="*", cex=2, main="Influential Observations by Cooks Distance")
abline(h = 4*mean(cooks_dist, na.rm=TRUE), col="red")
text(x = 1:length(cooks_dist)+1, y = cooks_dist+0.01,
     labels = ifelse(cooks_dist > 4*mean(cooks_dist, na.rm=TRUE), names(cooks_dist), ""), col="red")

pos_fires_noinf <- pos_fires[-influential,]

# Build final dataset for model - all area=0 rows + area>0 rows that remain after outlier removal.
```

```r
final_data <- rbind(fires[which(fires$area == 0),], pos_fires_noinf) # 511x15
final_data$area_bin <- ifelse(final_data$area == 0, 0, 1) # for area=0 model; 511x16
# Build model.
xvars <- c("X", "Y", "day", "season", "FFMC", "DMC", "DC", "ISI", "temp", "RH", "wind", "rain")

form.build <- function(y, xvars){
  return(as.formula(paste(y, "~", paste(xvars, collapse = "+"))))
}

nreps <- 5000
df <- matrix(nrow=nreps, ncol=3)
for(i in 1:nreps){

  tryCatch({
  # Randomly shuffle rows and make training (80%) and test (20%) sets.
  shuffled <- final_data[sample(nrow(final_data)),]
  train_data <- shuffled[1:round(0.8 * nrow(final_data)),]
  test_data <- shuffled[(round(0.8 * nrow(final_data)) + 1):nrow(final_data),]

  ## For area=0:
  logistic.fit <- glm(form.build("area_bin", xvars), train_data, family = binomial)
  ## For area>0:
  ols.fit <- lm(form.build("area", xvars), train_data[train_data$area > 0,])
  logols.fit <- lm(form.build("logarea", xvars), train_data[train_data$area > 0,])
  gamma.fit <- glm(form.build("area", xvars), train_data[train_data$area > 0,],
                   family = Gamma(link = log))

  ## Logistic prediction (part 1):
  phat <- predict(logistic.fit, test_data, type = "response") # predicted
  pred <- as.data.frame(test_data[,'area'], colname = 'area') # actual
  colnames(pred)[1] <- "area"
  ## Ordinary least squares, log OLS, gamma (part 2):
  pred$ols <- phat * predict(ols.fit, test_data)
  pred$logols <- phat * exp(predict(logols.fit, test_data) + summary(logols.fit)$sigma^2/2)
  pred$gamma <- phat * predict(gamma.fit, test_data, type = "response")

  ## Prediction accuracy via RMSE:
  rmse_vect <- c(RMSE(pred$area, pred$ols),
                 RMSE(pred$area, pred$logols),
                 RMSE(pred$area, pred$gamma))
  df[i,] <- rmse_vect
  }, error = function(e){})
}

df <- as.data.frame(df)
na.rows <- sum(is.na(df[,1]))
rmse_avgs <- colSums(df, na.rm=TRUE)/(nreps-na.rows)
names(rmse_avgs) <- c("OLS", "Log OLS", "Gamma")
rmse_avgs
### FOR THE PLOT:

set.seed(2001)
# Randomly shuffle rows and make training (80%) and test (20%) sets.
```

```r
shuffled <- final_data[sample(nrow(final_data)),]
train_data <- shuffled[1:round(0.8 * nrow(final_data)),]
test_data <- shuffled[(round(0.8 * nrow(final_data)) + 1):nrow(final_data),]

## For area=0:
logistic.fit <- glm(form.build("area_bin", xvars), train_data, family = binomial)
## For area>0:
ols.fit <- lm(form.build("area", xvars), train_data[train_data$area > 0,])
logols.fit <- lm(form.build("logarea", xvars), train_data[train_data$area > 0,])
gamma.fit <- glm(form.build("area", xvars), train_data[train_data$area > 0,],
                 family = Gamma(link = log))

## Logistic prediction (part 1):
phat <- predict(logistic.fit, test_data, type = "response") # predicted
pred <- as.data.frame(test_data[,'area'], colname = 'area') # actual
colnames(pred)[1] <- "area"
## Ordinary least squares, log OLS, gamma (part 2):
pred$ols <- phat * predict(ols.fit, test_data)
pred$logols <- phat * exp(predict(logols.fit, test_data) + summary(logols.fit)$sigma^2/2)
pred$gamma <- phat * predict(gamma.fit, test_data, type = "response")

# Predictive simulation.

## Logistic-normal model:
n <- nrow(test_data)
d <- rbinom(n, 1, phat)
y.norm <- d * rnorm(n, pred$ols, summary(ols.fit)$sigma)

## Logistic-lognormal model:
y.lognorm <- d * rlnorm(n, predict(logols.fit, test_data),
                        summary(logols.fit)$sigma)

## Logistic-gamma model:
res <- (gamma.fit$model$area - gamma.fit$fit)/gamma.fit$fit # this is equivalent to gamma.fit$res
# c(sum(res^2)/gamma.fit$df.res, summary(gamma.fit)$dispersion)
a <- gamma.shape(gamma.fit)$alpha # use MASS package gamma.shape to estimate shape parameter (using ML)
b <- a/pred$gamma
y.gamma <- d * rgamma(n, shape = a , rate = b)

## Plot
y <- pred$area
p.dat <- data.table(y = c(y, y.norm, y.lognorm, y.gamma),
                    lab = c(rep("Observed", n), rep("Normal", n),
                            rep("Lognormal", n), rep("Gamma", n)))
ggplot(p.dat[y > 0 & y < 10000], aes(x = y, col = lab)) +
  geom_density(kernel = "gaussian") +
  labs(title = "Comparing Models to Observed Density",
       x = "Burned area (ha)",
       y = "Density") +
  xlim(0, 150) +
  theme(legend.position="bottom") + labs(col = "") +
  scale_color_manual(values=c(Observed = "black", Normal = "red",
                              Lognormal = "blue", Gamma = "green"))
```

```r
#### PCA (keep all continuous predictors for consideration)

fires_preds <- fires_lr2[,c("X", "Y", "FFMC", "DMC", "DC", "ISI",
                           "temp", "RH", "wind", "rain")]
pca <- prcomp(fires_preds,
                 center = TRUE,
                 scale. = TRUE)

summary(pca) # need 8/10 components to explain 95% of the variation in the data
# goes to show that we need most of the variables to explain the variation in the response

# plot(pca, type = "l", main = "Variance for Each Principal Component")
set.seed(2001)
fires_cont <- fires_lr2

train_cont <- sample(nrow(fires_cont), 0.7*nrow(fires_cont), replace = FALSE)
train.set <- fires_lr2[train_cont,]
test.set <- fires_lr2[-train_cont,]

dt.fit <- rpart(form.build("logarea", xvars),
    method="anova", data=train.set)

test.pred.rtree <- predict(dt.fit, test.set)

RMSE.rtree <- sqrt(mean((test.pred.rtree-test.set$logarea)^2))

fancyRpartPlot(dt.fit)
printcp(dt.fit)
# min.xerror <- dt.fit$cptable[which.min(dt.fit$cptable[,"xerror"]),"CP"]

rt.pruned <- prune(dt.fit, cp = 0.023188)

fancyRpartPlot(rt.pruned)
test.pred.rtree.p <- predict(rt.pruned, test.set)
RMSE.rtree.pruned <- sqrt(mean((test.pred.rtree.p-test.set$logarea)^2))
RMSE.rtree.pruned
set.seed(2001)
rf.fit <- randomForest(form.build("logarea", xvars), data=train.set,
                       importance = TRUE, ntree=2000)
imp <- as.data.frame(sort(round(randomForest::importance(rf.fit)[,1], 3), decreasing = TRUE), optional =
names(imp) <- "If absent, % increase MSE"
imp
test.pred.forest <- predict(rf.fit, test.set)
RMSE.forest <- sqrt(mean((test.pred.forest-test.set$logarea)^2))
RMSE.forest
rf.fit2 <- randomForest(logarea ~ temp + DMC + DC + season + rain +
                       ISI + FFMC + Y, data=train.set,
                       importance = TRUE, ntree=2000)
imp <- as.data.frame(sort(round(randomForest::importance(rf.fit2)[,1], 3), decreasing = TRUE), optional
names(imp) <- "If absent, % increase MSE"
imp
test.pred.forest2 <- predict(rf.fit2, test.set)
RMSE.forest2 <- sqrt(mean((test.pred.forest2-test.set$logarea)^2))
```

```
RMSE.forest2
set.seed(2001)
svr.fit <- svm(form.build("logarea", xvars), data=fires_cont, type="eps")
test.pred.svr <- predict(svr.fit, data=fires_cont)
error.svr <- sqrt(mean((test.pred.svr-fires_cont$logarea)^2))
error.svr
# grid search
tuneResult <- tune(svm, form.build("logarea", xvars),  data=fires_cont,
              ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:6))
)
print(tuneResult)
plot(tuneResult)
tuneResult2 <- tune(svm, form.build("logarea", xvars),  data=fires_cont,
                  ranges = list(epsilon = seq(0.6,0.9,0.05), cost = seq(1,10,1))
)
plot(tuneResult2)
tunedModel <- tuneResult2$best.model
tunedModel
tunedModel_predict <- predict(tunedModel, fires_cont)
sqrt(mean((tunedModel_predict-fires_cont$logarea)^2))
```