STAT444 Final Project
Winter 2018

STATISTICAL ANALYSIS ON FOREST FIRE DATA

April 10, 2018

Team Excalibur
Yuke Wu 20566786
Chen Yuan 20561973
Miaojia Pu 20503021
University of Waterloo

Motivation & Introduction

Environment has become a major concern recent years, and forest fires are one of the serious environment issues. In this project, data about forest fires in Portugal are analyzed. Numerical predictor variables such as tempurature, rain, wind speed and categorical variables such as month and day are included in this dataset, and are used to predict the burned area of the forest in a wildfire.

To achieve our goal, after preprocessing the data, we fit four models including smoothing spline, random forest, gradient boosting and generalized linear model based on the given data. In model selection part, we try to provide evidence why or why not the model is a good fit and select the best candidate in conclusion.

Data

Our data is from UCI Machine Learning Repository. Below are all variables included in the dataset and their value range.

1. Input variable

- `X` - x-axis spatial coordinate within the Montesinho park map: 1-9
- `Y` - y-axis spatial coordinate within the Montesinho park map: 2-9
- `month` - month of the year: "jan"-"dec"
- `day` - day of the week: "mon"-"sun"
- `FFMC` - Fine Fuel Moisture Code (FFMC) index from Fire Weather Index (FWI) system: 18.70-96.20
- `DMC` - Duff Moisture Code (DMC) index from FWI system: 1.1-291.3
- `DC` - Drought Code (DC) index from FWI system: 7.9-860.6
- `ISI` - Initial Spread Index (ISI) index from FWI system: 0.00-56.10
- `temp` - temperature in Celsius degrees: 2.20-33.30
- `RH` - relative humidity in %: 15.0-100.0
- `wind` - wind speed in km/h: 0.40-9.40
- `rain` - outside rain in mm/m2: 0.0-6.4

2. Output variable

- `area` - the burned area of the forest in ha: 0.00-1090.84

The following information may provide a general idea about the data.

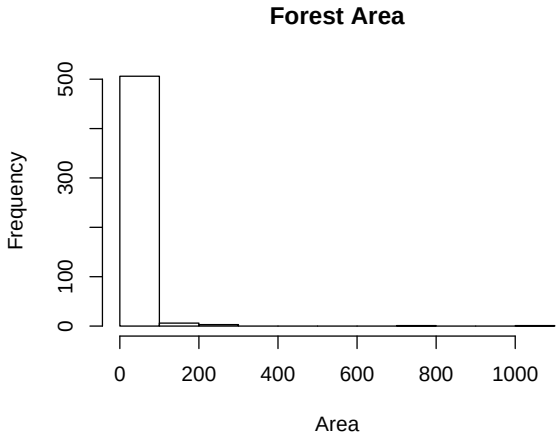
```
##        X               Y           month      day           FFMC
##  Min.   :1.000   Min.   :2.0   aug    :184   fri:85   Min.   :18.70
##  1st Qu.:3.000   1st Qu.:4.0   sep    :172   mon:74   1st Qu.:90.20
##  Median :4.000   Median :4.0   mar    : 54   sat:84   Median :91.60
##  Mean   :4.669   Mean   :4.3   jul    : 32   sun:95   Mean   :90.64
##  3rd Qu.:7.000   3rd Qu.:5.0   feb    : 20   thu:61   3rd Qu.:92.90
##  Max.   :9.000   Max.   :9.0   jun    : 17   tue:64   Max.   :96.20
##                                (Other): 38   wed:54
##       DMC              DC             ISI              temp
##  Min.   :  1.1   Min.   :  7.9   Min.   : 0.000   Min.   : 2.20
##  1st Qu.: 68.6   1st Qu.:437.7   1st Qu.: 6.500   1st Qu.:15.50
##  Median :108.3   Median :664.2   Median : 8.400   Median :19.30
##  Mean   :110.9   Mean   :547.9   Mean   : 9.022   Mean   :18.89
##  3rd Qu.:142.4   3rd Qu.:713.9   3rd Qu.:10.800   3rd Qu.:22.80
##  Max.   :291.3   Max.   :860.6   Max.   :56.100   Max.   :33.30
```

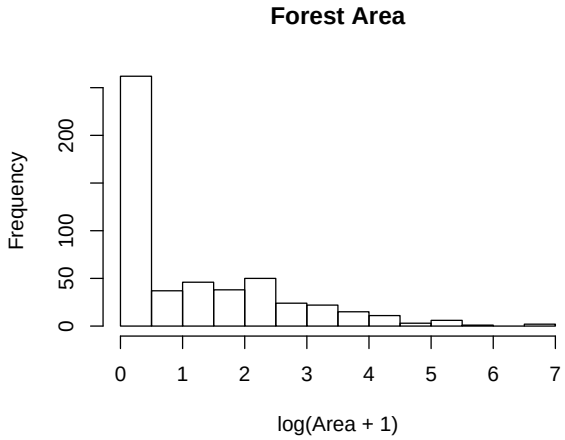
```
##
##        RH              wind            rain              area
##  Min.   : 15.00   Min.   :0.400   Min.   :0.00000   Min.   :   0.00
##  1st Qu.: 33.00   1st Qu.:2.700   1st Qu.:0.00000   1st Qu.:   0.00
##  Median : 42.00   Median :4.000   Median :0.00000   Median :   0.52
##  Mean   : 44.29   Mean   :4.018   Mean   :0.02166   Mean   :  12.85
##  3rd Qu.: 53.00   3rd Qu.:4.900   3rd Qu.:0.00000   3rd Qu.:   6.57
##  Max.   :100.00   Max.   :9.400   Max.   :6.40000   Max.   :1090.84
##
```

Preprocessing

In preprocessing part, we create dummies for categorical vairables(month and day), and perform mean std normalization to the dataset.

Since most of our responsable variables are 0, our data is very skewed, we apply a log transformation to our response variable.





Model Fitting and Model Selection

To fully compare the performances of each model, we put process time into consideration, and also employ the Root-Mean-Squared Error(RMSE), which can be calculated as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

We use a 5-fold corss-validation to obtain the average RMSE.

smoothing spline

When trying to fit the dataset into smoothing splines, we choose three different models. The main difference among the models is the variables that are considered to have an interaction effect on the dataset.

Model 1: No interaction effects

```
## gam(formula = log(area + 1) ~ s(X, k = 9) + s(Y, k = 6) + monthaug +
##     monthdec + monthfeb + monthjan + monthjul + monthjun + monthmar +
##     monthmay + monthnov + monthoct + monthsep + daymon + daysat +
##     daysun + daythu + daytue + daywed + s(FFMC) + s(DMC) + s(DC) +
##     s(ISI) + s(temp) + s(RH) + s(wind) + s(rain, k = 5), data = forest[-flds[[i]],
##     ])
```

Model 2: Interaction effect considered between temp, RH, wind and rain

```
## gam(formula = log(area + 1) ~ s(X, k = 9) + s(Y, k = 6) + monthaug +
##     monthdec + monthfeb + monthjan + monthjul + monthjun + monthmar +
##     monthmay + monthnov + monthoct + monthsep + daymon + daysat +
##     daysun + daythu + daytue + daywed + s(FFMC) + s(DMC) + s(DC) +
##     s(ISI) + s(temp) + s(RH) + s(wind) + s(rain, k = 5) + ti(temp,
##     RH, wind, rain), data = forest[-flds[[i]], ])
```

Model 3: Interaction effect considered between FFMC and DMC

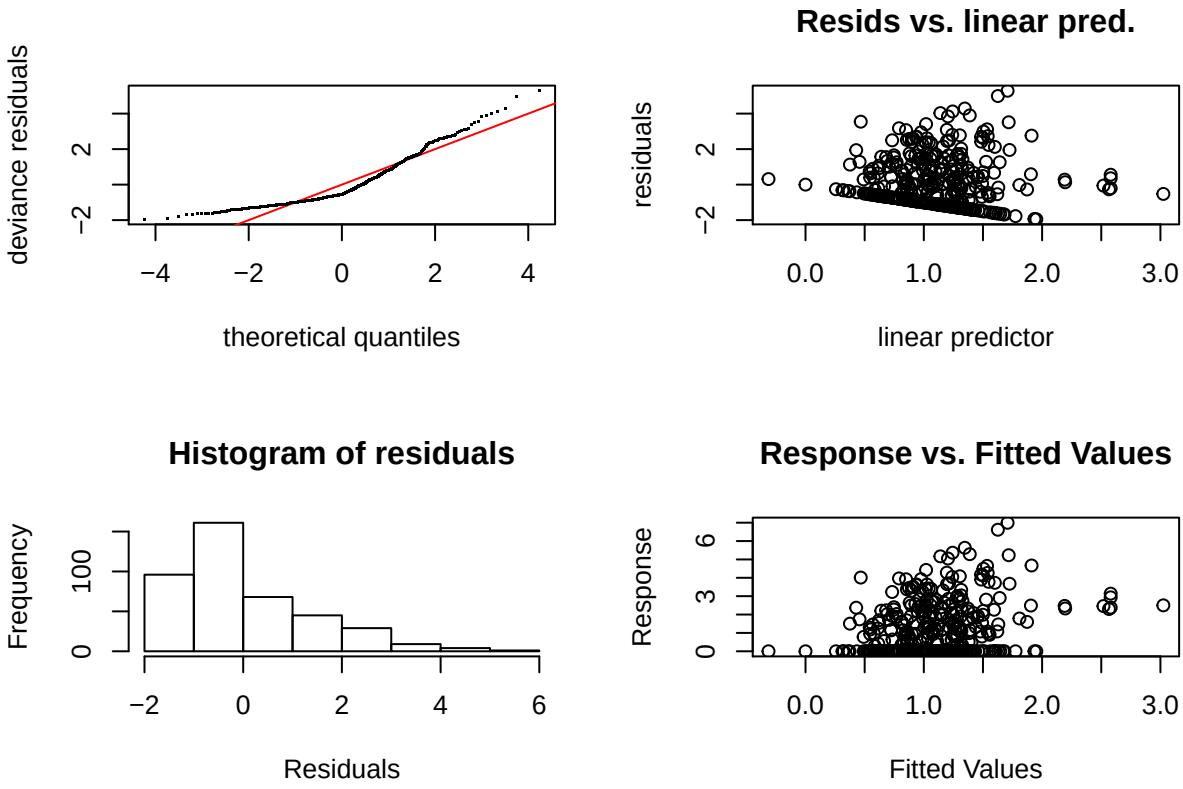
```
## gam(formula = log(area + 1) ~ s(X, k = 9) + s(Y, k = 6) + monthaug +
##     monthdec + monthfeb + monthjan + monthjul + monthjun + monthmar +
##     monthmay + monthnov + monthoct + monthsep + daymon + daysat +
##     daysun + daythu + daytue + daywed + s(FFMC) + s(DMC) + s(DC) +
##     s(ISI) + s(temp) + s(RH) + s(wind) + s(rain, k = 5) + ti(FFMC,
##     DMC), data = forest[-flds[[i]], ])
```

To compare the three models and find the best one, we take RMSE,AIC and Run Time into consideration. The three scores for the three models respectively are calculated as below:

	RMSE	AIC	ProcessTime
model1	47.749(+/-)67.654	1463.674(+/-)13.065	2.63
model2	44.072(+/-)59.507	1461.227(+/-)15.291	35.74
model3	118.022(+/-)224.324	1464.741(+/-)13.374	4.00

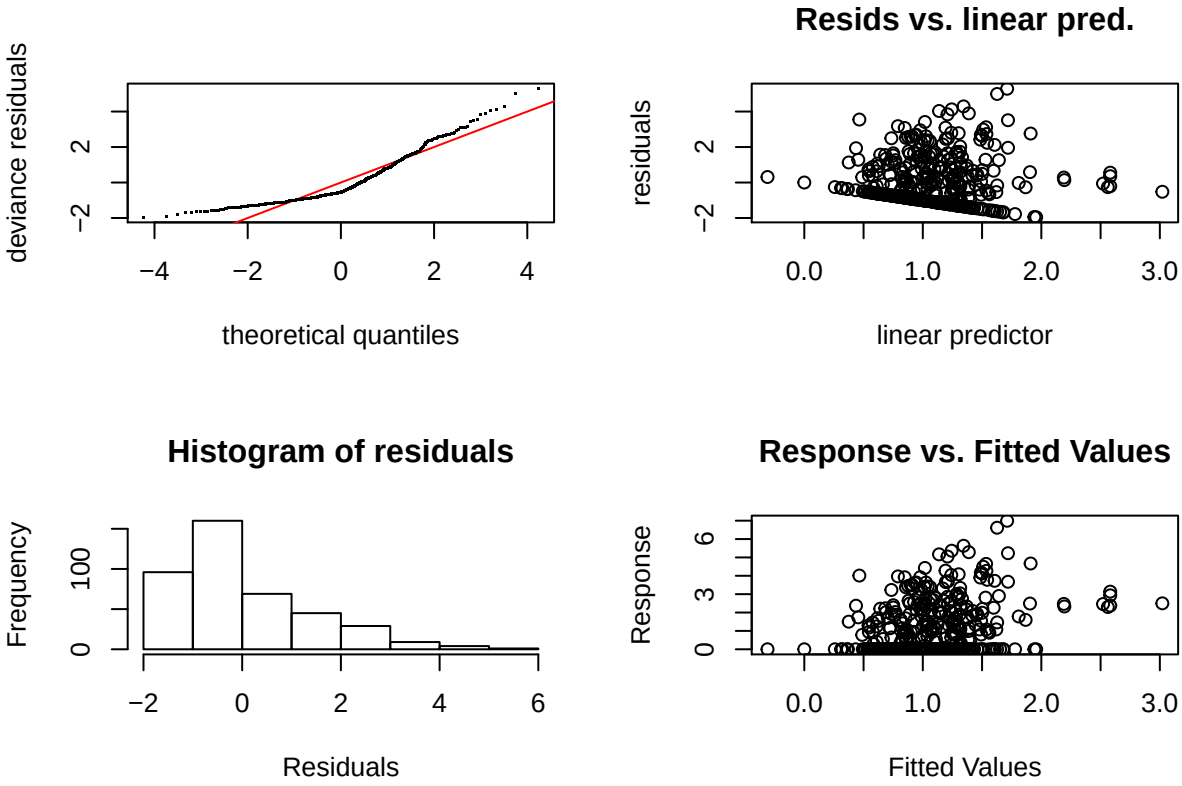
We can also check if the model is a good fit by looking at the residuals of it.

Model 1: No interaction effects



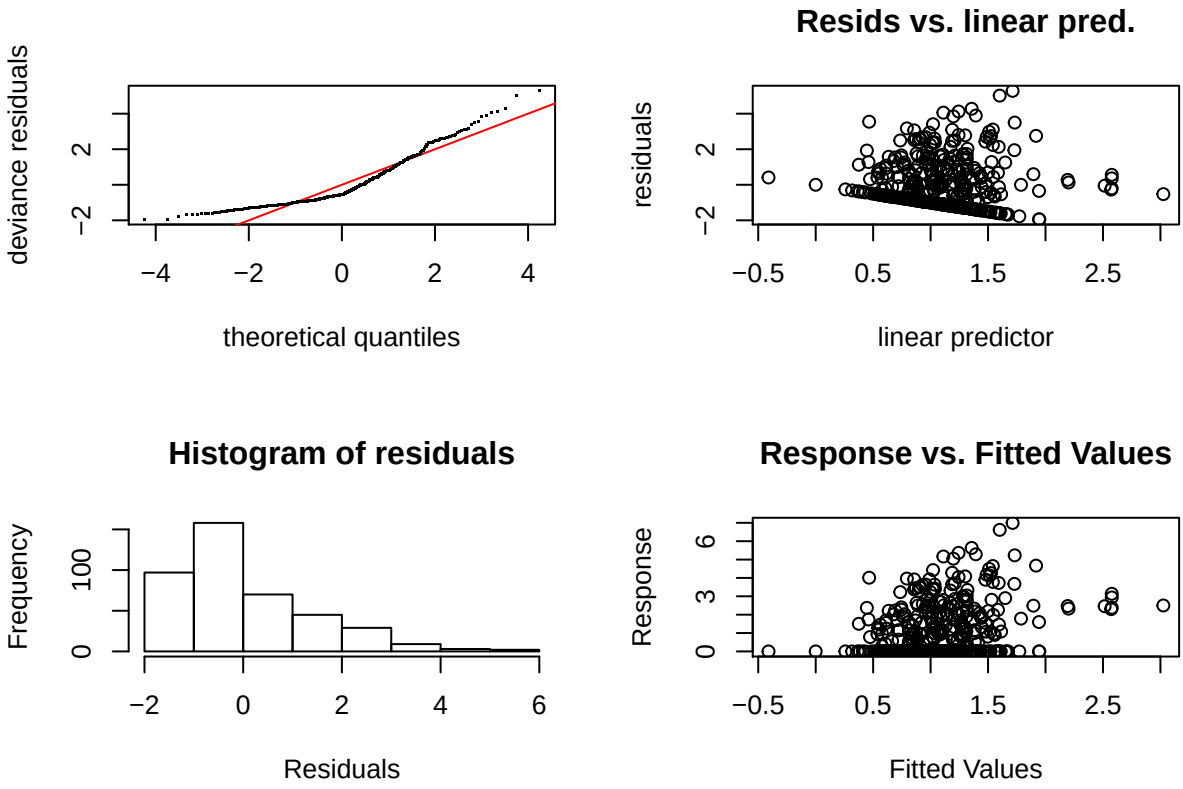

```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 15 iterations.
## The RMS GCV score gradient at convergence was 6.335681e-08 .
## The Hessian was positive definite.
## Model rank =  98 / 98
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'  edf k-index p-value
## s(X)    8.00 1.59    0.57  <2e-16 ***
## s(Y)    5.00 1.00    0.58  <2e-16 ***
## s(FFMC) 9.00 1.00    0.78  <2e-16 ***
## s(DMC)  9.00 1.51    0.92   0.020 *
## s(DC)   9.00 1.00    0.93   0.045 *
## s(ISI)  9.00 1.00    0.78  <2e-16 ***
## s(temp) 9.00 1.46    0.97   0.230
## s(RH)   9.00 1.42    0.90   0.015 *
## s(wind) 9.00 1.00    0.66  <2e-16 ***
## s(rain) 4.00 1.00    0.48  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model 2: Interaction effect considered between temp, RH, wind and rain




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 30 iterations.
## The RMS GCV score gradient at convergence was 3.28905e-05 .
## The Hessian was not positive definite.
## Model rank =  143 / 143
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                              k'      edf k-index p-value
## s(X)                   8.00e+00 1.59e+00    0.57  <2e-16 ***
## s(Y)                   5.00e+00 1.00e+00    0.58  <2e-16 ***
## s(FFMC)                9.00e+00 1.00e+00    0.78  <2e-16 ***
## s(DMC)                 9.00e+00 1.54e+00    0.92   0.035 *
## s(DC)                  9.00e+00 1.00e+00    0.93   0.055 .
## s(ISI)                 9.00e+00 1.00e+00    0.78  <2e-16 ***
## s(temp)                9.00e+00 1.49e+00    0.97   0.300
## s(RH)                  9.00e+00 1.39e+00    0.90   0.025 *
## s(wind)                9.00e+00 1.00e+00    0.66  <2e-16 ***
## s(rain)                4.00e+00 1.00e+00    0.48  <2e-16 ***
## ti(temp,RH,wind,rain) 4.50e+01 3.82e-07    0.97   0.290
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model 3: Interaction effect considered between FFMC and DMC

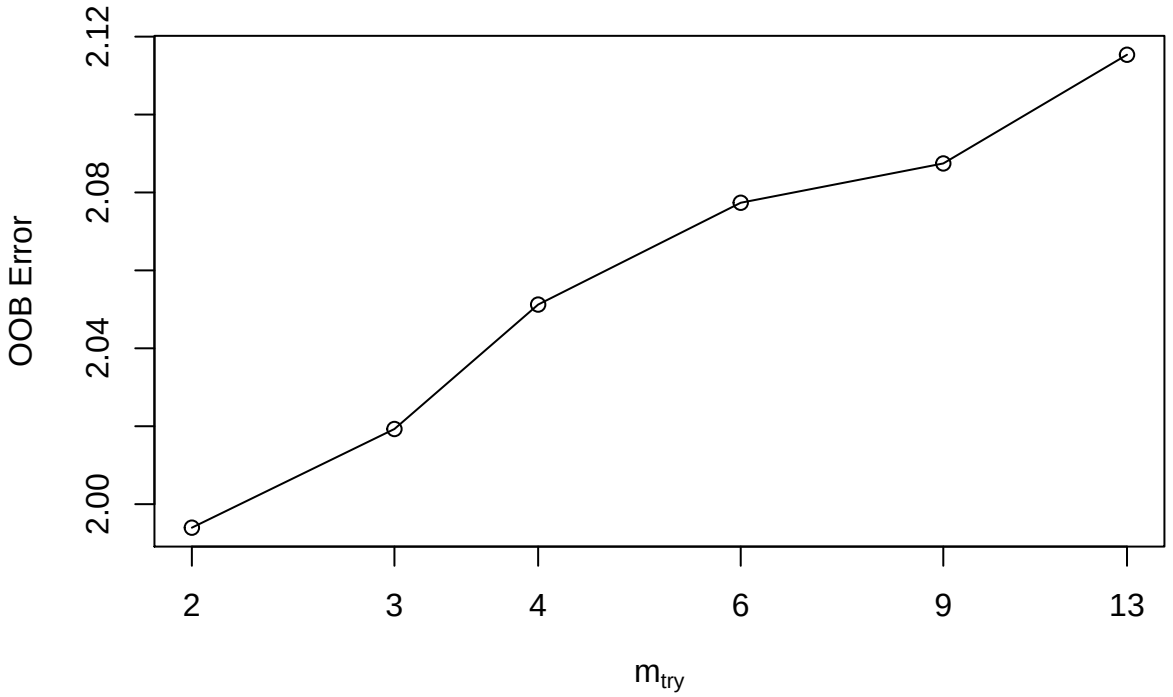



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 18 iterations.
## The RMS GCV score gradient at convergence was 6.042499e-08 .
## The Hessian was positive definite.
## Model rank =  114 / 114
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                  k'   edf k-index p-value
## s(X)           8.00  1.56    0.57  <2e-16 ***
## s(Y)           5.00  1.00    0.58  <2e-16 ***
## s(FFMC)        9.00  1.00    0.78  <2e-16 ***
## s(DMC)         9.00  1.54    0.91    0.04 *
## s(DC)          9.00  1.00    0.93    0.05 *
## s(ISI)         9.00  1.00    0.78  <2e-16 ***
## s(temp)        9.00  1.49    0.97    0.25
## s(RH)          9.00  1.43    0.90    0.03 *
## s(wind)        9.00  1.00    0.66  <2e-16 ***
## s(rain)        4.00  1.00    0.47  <2e-16 ***
## ti(FFMC,DMC) 16.00  1.00    1.02    0.68
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In general, model 3 has the largest value for both RMSE and AIC values, therefore model 3 is no longer considered. The two scores for model 1 and model 2 are quite close. Model 2 has smaller RMSE and AIC score compared with model 1, which means it is better in performance. But the process time increases as the number of interaction term increases, which is a lot of computationally cost when the size of dataset grows large. Although model 2 takes longer to process, but it does improve the level of fitness to the data. Overall, we consider model 2 as the best model among the three smoothing spline models.

Random Forest

Second, we use random forests on our data. The hyperparameter m_{try} is optimized using the `tuneRF` function from the *randomForest* package in **R**. Below is a graph for OOB error is plotted to illustrate our optimizing result.



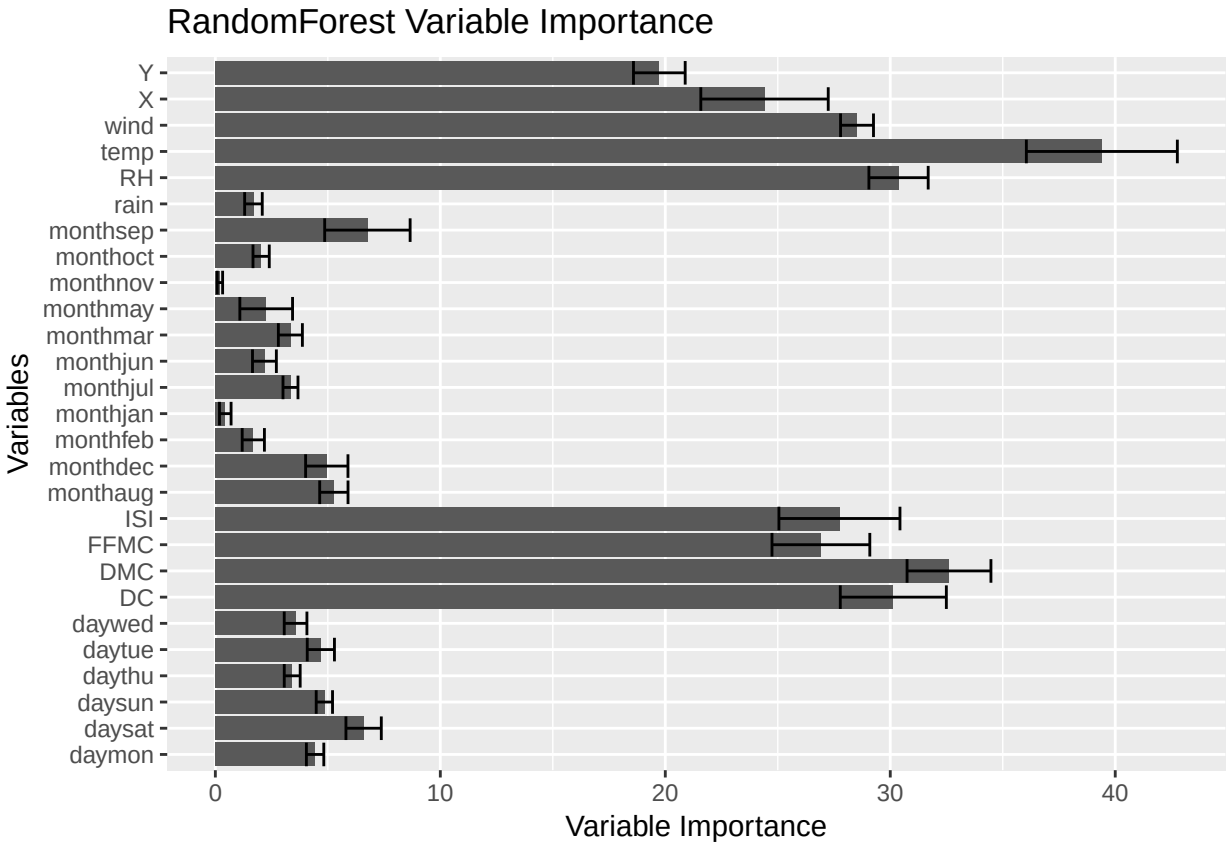
From the graph above, we can see that the smallest OOB error is achieved when m_{try}=2. Therefore, we take this value into our Random Forest model for the cross-validation step.

To evaluate this model, we also calculate its RMSE:

```
## [1] "Random Forest RMSE:  24.2993820387277 (+/-) 17.3502832309186"
```

Compared to that of the smoothing spline model, the RMSE of this random forests model is quite small and less spread out.

For random forests model, we are also interested in its importance of variables.



To show this in a ascending order of variable importance:

	rowMeans.rf_cv_importance.
monthnov	0.2044063
monthjan	0.4417969
monthfeb	1.6876441
rain	1.6944155
monthoct	2.0354462
monthjun	2.1822326
monthmay	2.2608855
monthmar	3.3356815
monthjul	3.3406917
daythu	3.4170370
daywed	3.5666660
daymon	4.4337151
daytue	4.6892772
daysun	4.8458394
monthdec	4.9538772
monthaug	5.2683715
daysat	6.5910255
monthsep	6.7592566
Y	19.7352070
X	24.4110217
FFMC	26.9168613
ISI	27.7415724
wind	28.5223945

	rowMeans.rf_cv_importance.
DC	30.1354805
RH	30.3683399
DMC	32.6109273
temp	39.4051630

We can clearly see that temperature leads the importance of all variables, which is quite reasonable, followed by weather factors like relative humidity and wind, and the four factors from the FWI system. X and Y coordinates have relatively high importance, which implies that the surroundings or the geography of the specific loaction has some effect on the burned area. Month and day have smaller impacts, and so does rain, which is quite surprising.

Gradient Boosting

Next, we use gradient boosting method on the data.

In order to tune hyperparameters, we used a gridsearach for the following hyperparameters, with parameter settings specified as follows:

- `n.trees`: [100, 200, 300]
- `interaction.depth`: [2, 4, 6]
- `shrinkage`: [0.01, 0.1, 1.0]

These are our best hyperparameters:

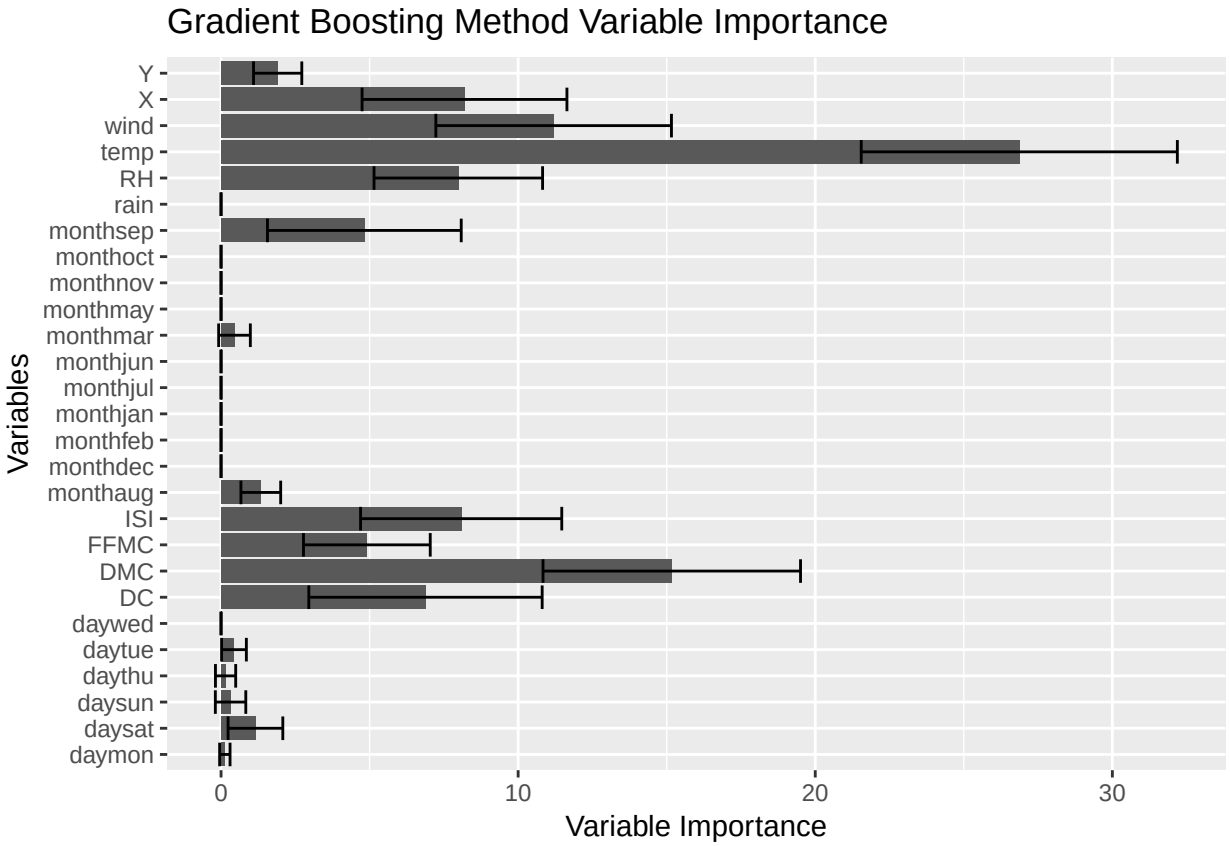
- `n.trees`: 100
- `interaction.depth`: 2
- `shrinkage`: 0.01

To further investigate this model, we first calculate the RMSE value as well:

```
## [1] "Gradient Boosting RMSE:  24.3225450546371 (+/-) 17.3196457462894"
```

We can see that RMSE value for gradient boosting method is also relatively small, which means it might be a good fit for this dataset.

We are also interested in the important variables produced by this model:



Re-order this graph and put it in numerical style:

	rowMeans.gbm_cv_importance.
monthdec	0.0000000
monthfeb	0.0000000
monthjan	0.0000000
monthjul	0.0000000
monthjun	0.0000000
monthmay	0.0000000
monthnov	0.0000000
monthoct	0.0000000
daywed	0.0000000
rain	0.0000000
daymon	0.1276605
daythu	0.1524868
daysun	0.3204646
daytue	0.4350671
monthmar	0.4527913
daysat	1.1580471
monthaug	1.3383527
Y	1.9055723
monthsep	4.8233594
FFMC	4.9080981
DC	6.8815849
RH	7.9859323
ISI	8.0822049

	rowMeans.gbm_cv_importance.
X	8.1949977
wind	11.1948382
DMC	15.1708759
temp	26.8676664

From the information above, we can tell that the results are similar to those of random forests model. Month, day and rain still have the smallest influence on the output. Temperature still has the highest influence, followed by location, wind, humidity and factors from the FWI system.

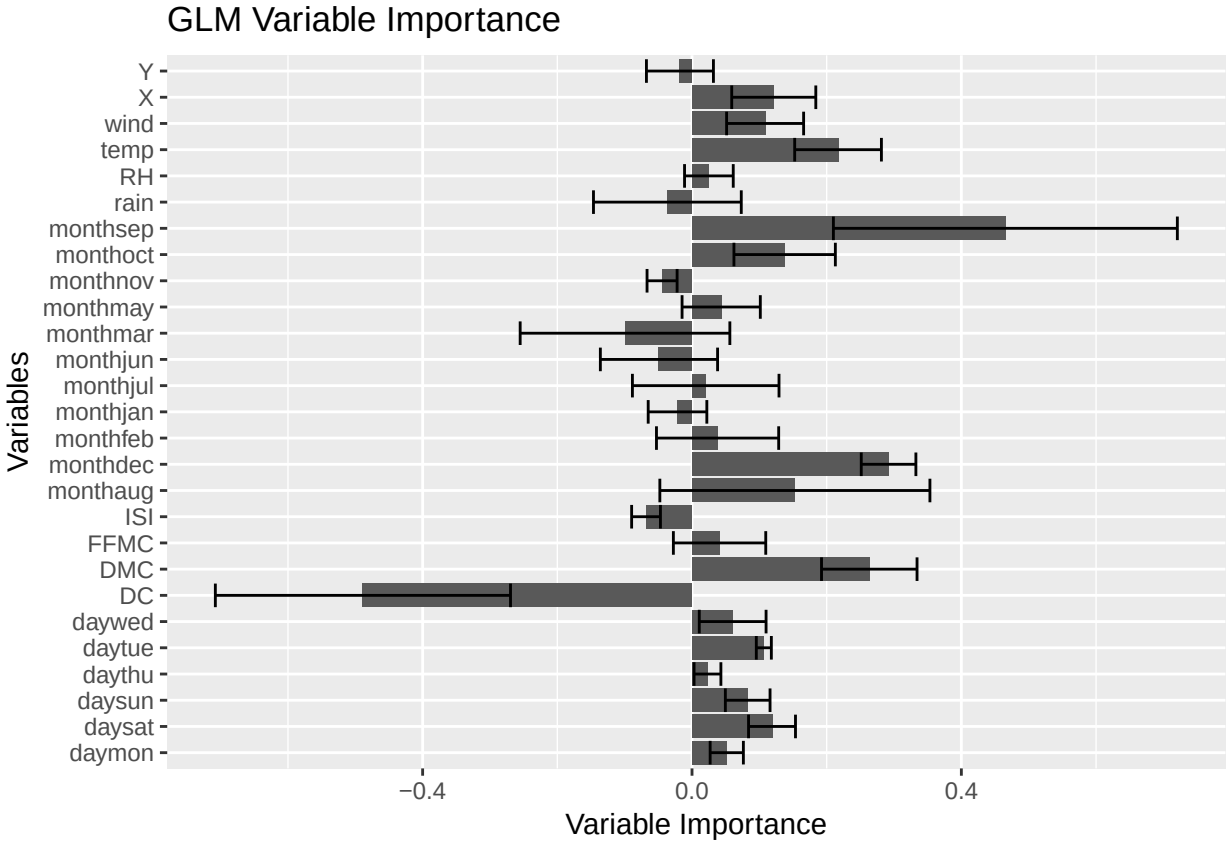
Generalized Linear Model

Finally, we use generalized linear model on our data. The default parameter settings from **glm** function are used.

To further analyze the fitness of the model, we use its RMSE as the prediction error:

```
## [1] "Logistic Regression RMSE:  24.3033625459695 (+/-) 17.2854413919133"
```

We are also interested in the important variables produced by this model:



From the graph above, we can tell that the variable importance of GLM is quite different from that of random forests model and gradient boosting method. Temperature no longer has the highest influence. Instead, month September leads the variable importance. Wind, temperature, humidity and factors from the FWI system have less importance than they do in the previous two models. Some of the months have

great influence on the model, but other months do not. Days of the week still have small importance in the model fitting.

Statisitcal Conclusions

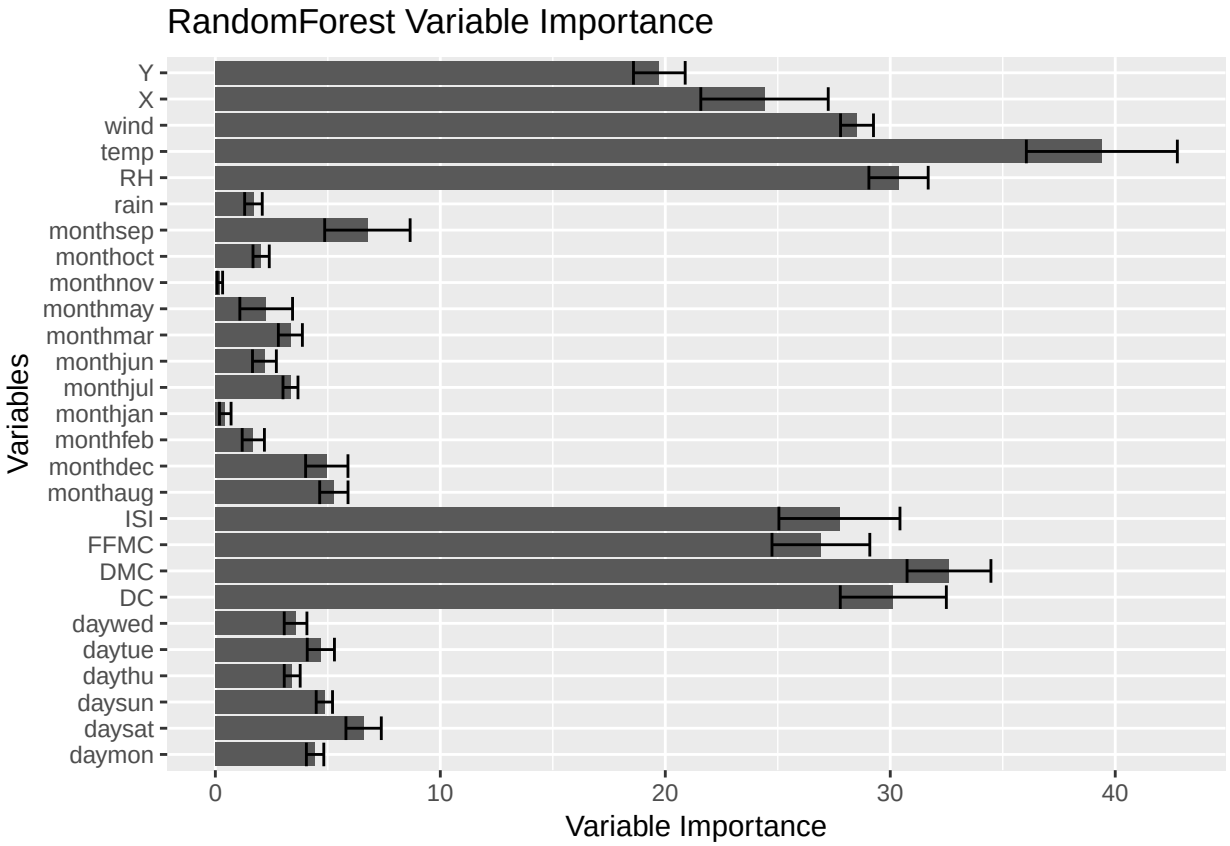
To compare the models above and select the best candidate, factors that we consider are the RMSE value and process time of each model. Note that for smoothing spline, we use model 2 out of the 3 smoothing splines in comparison as we discussed this selection in the smoothing spline part above.

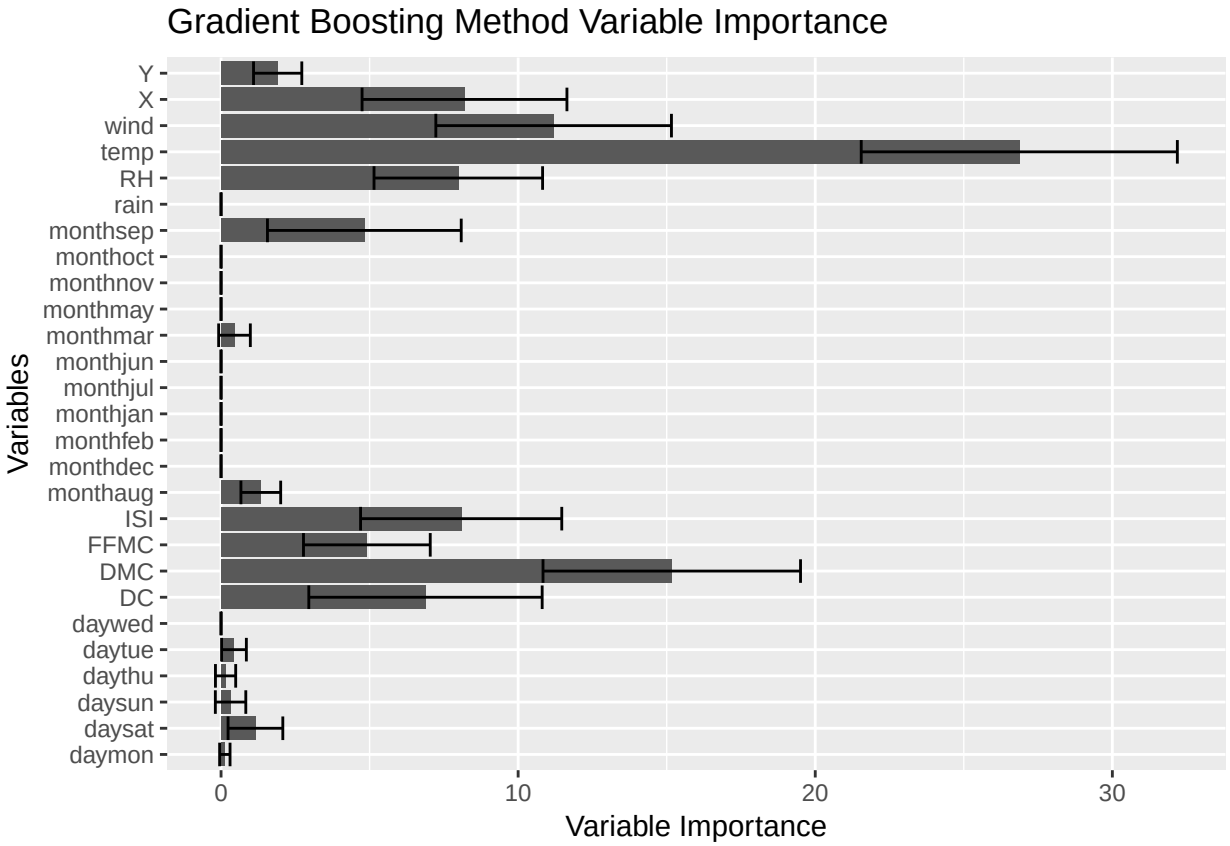
	RMSE	ProcessTime
Random Forest	24.299(+/-)17.350	1.98
Gradient Boosting	24.323(+/-)17.320	0.37
Smoothing Spline	44.072(+/-)59.507	35.74
Generalized Linear Model	24.303(+/-)17.285	0.06

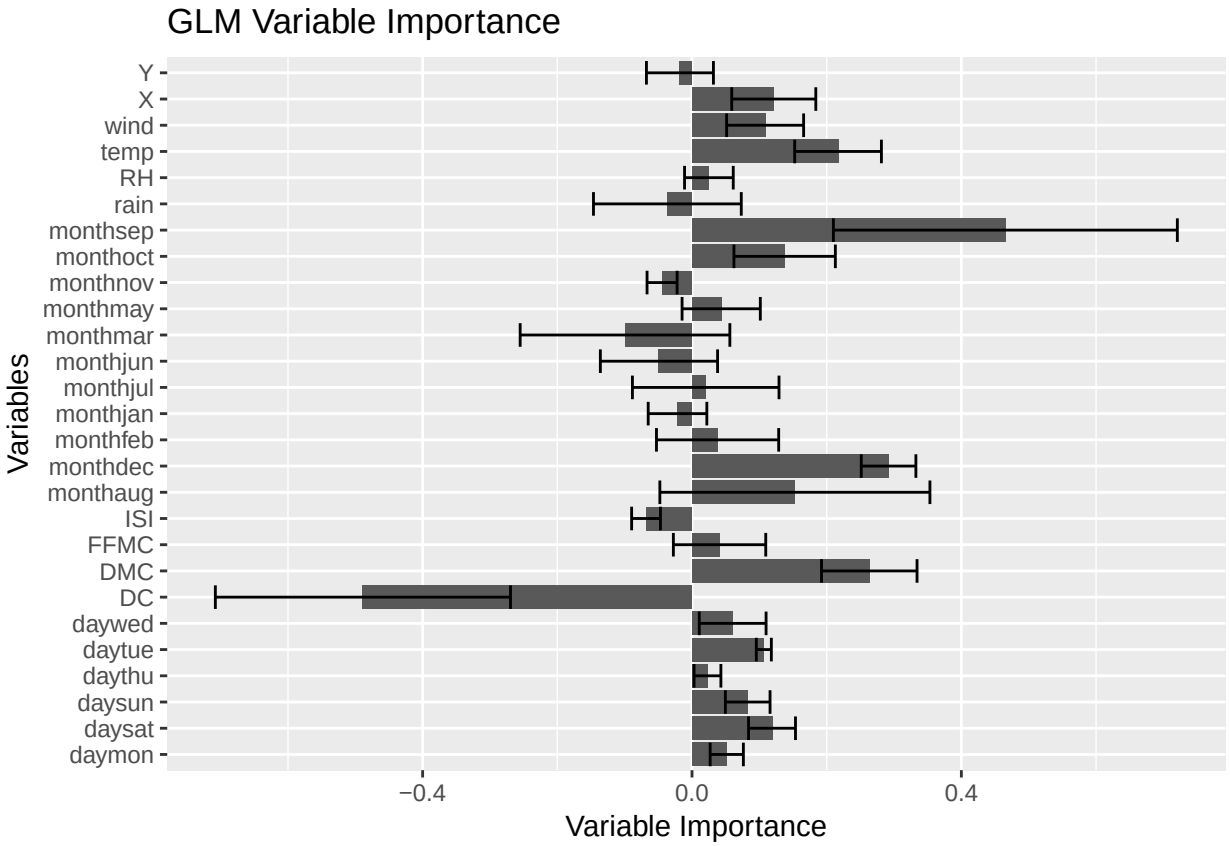
From the table above, smoothing spline model has the largest RMSE value and the longest process time, and the RMSE values for the rest three models are quite close. We decide to pick the best candidate with repect to RMSE value. Therefore, random forests model, which has the smallest RMSE value, becomes our best candidate.

Conclusions in the Context

Although random forests model is our best candidate among these four models, it does not mean that random forests model is the true model for this dataset. What we might be able to conclude is which input variables have larger influence on model fitting and which have smaller influence.


RandomForest Variable Importance
Y
X
wind
temp
RH
rain
monthsep
monthoct
monthnov
monthmay
monthmar
monthjun
monthjul
monthjan
monthfeb
monthdec
monthaug
ISI
FFMC
DMC
DC
daywed
daytue
daythu
daysun
daysat
daymon
Variables
0
10
20
30
40
Variable Importance


Gradient Boosting Method Variable Importance
Y
X
wind
temp
RH
rain
monthsep
monthoct
monthnov
monthmay
monthmar
monthjun
monthjul
monthjan
monthfeb
monthdec
monthaug
ISI
FFMC
DMC
DC
daywed
daytue
daythu
daysun
daysat
daymon
Variables
0
10
20
30
Variable Importance



From the graph above, we consider that temperature has the greatest influence on model fitting. Other factors that have high influence are relative humidity, wind, four factors from FWI system(ISI, FFMC, DMC and DC) and location of the forest fire. All of them are considered to be positively correlated to the burned area. Factors that have very small or even no influence are rain, month, and day (of the week).

Future Work

After conducting the above modeling and analysis, our final model (random forests model) is still not a true model for this dataset. However, based on the information above, we can find a direction to improve the model. One way may be to reduce the weight of less important factors when modeling. Moreover, there are other factors affecting the burned area of forest that have not been considered in this dataset, such as the amount of combustible and the oxygen content in the forest. With data collected for these factors, the prediction of models may be more precise. Some other machine-learning models can be tested, such as Ridge Regression, Neural Network, etc.

Contribution

Data collection: Chen Yuan

Data Cleaning: Chen Yuan

Modeling: Chen Yuan

Model selection: Miaojia Pu

Analysis: Yuke Wu; Chen Yuan

Report Writing: Yuke Wu

Appendix

Data

link: https://archive.ics.uci.edu/ml/datasets/Forest+Fires

Literature

In the paper listed in reference, the output variable was first applied a log transformation to, and the inverse of the transformation after several Data Mining methods were conducted. 10-fold cross-validation and 30 runs were used during the experiments. The best candidate was picked with respect to MAD and RMSE. The Support Vector Machines (SVM) model was found to predict small fires better, which are the majority forest fires. SVM model:

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i \phi_i(x)$$

where

$$\phi_i(x)$$

represents a nonlinear transformation, according to the kernal function

$$K(x, x') = \sum_{i=1}^{m} \phi_i(x)\phi_i(x')$$

.

Code

```
forest <- read.csv("forestfires.csv", head=TRUE)

summary(forest)

forest <- data.frame(scale(model.matrix(area ~ ., forest)[,-1]), forest$area)

colnames(forest)[28] <- "area"

X = forest[1:27]
y = forest[28]

par(mfrow=c(1,2))
hist(forest$area, main="Forest Area", xlab="Area")
hist(log(forest$area + 1), main="Forest Area", xlab="log(Area + 1)")

library(caret)
  # 5 folds
seed = 100
set.seed(seed)
flds <- createFolds(seq(1,nrow(y),1), k = 5, list = TRUE, returnTrain = FALSE)

rf_cv_importance = matrix(rep(0, 5*27), ncol=5)
row.names(rf_cv_importance) <- colnames(forest)[1:27]
gbm_cv_importance = matrix(rep(0, 5*27), ncol=5)
row.names(gbm_cv_importance) <- colnames(forest)[1:27]
```

```
glm_cv_importance = matrix(rep(0, 5*27), ncol=5)
row.names(glm_cv_importance) <- colnames(forest)[1:27]

rf_cv_rmse = rep(0,5)
gbm_cv_rmse = rep(0,5)
glm_cv_rmse = rep(0,5)
sp_cv_rmse = rep(0,5)
sp_cv_AIC = rep(0, 5)
sp2_cv_rmse = rep(0,5)
sp2_cv_AIC = rep(0, 5)
sp3_cv_rmse = rep(0,5)
sp3_cv_AIC = rep(0, 5)

library(mgcv)
  # Start the clock
  start.time <- proc.time()
for (i in 1:5) {
  sp.model <- gam(log(area + 1) ~ s(X, k=9) + s(Y, k=6) +
                  monthaug +
                  monthdec + monthfeb +
                  monthjan + monthjul +
                  monthjun + monthmar +
                  monthmay + monthnov +
                  monthoct + monthsep +
                  daymon +   daysat +
                  daysun + daythu +
                  daytue + daywed +
                  s(FFMC) + s(DMC) + s(DC) + s(ISI) +
                  s(temp) + s(RH) + s(wind) +
                  s(rain, k=5), data=forest[-flds[[i]],])
  # prediction & RMSE
  sp_pred = exp(predict(sp.model, newdata=forest[flds[[i]],])) - 1
  sp_cv_rmse[i] = sqrt(sum((forest[flds[[i]],]$area - sp_pred)^2) / nrow(forest))
  sp_cv_AIC[i] = AIC(sp.model)
}
  # Stop the clock
  sp_time = proc.time() - start.time
sp.model$call

  # interaction
  # Start the clock
  start.time <- proc.time()
for (i in 1:5) {
  sp.model2 <- gam(log(area + 1) ~ s(X, k=9) + s(Y, k=6) +
                  monthaug +
                  monthdec + monthfeb +
                  monthjan + monthjul +
                  monthjun + monthmar +
                  monthmay + monthnov +
                  monthoct + monthsep +
                  daymon +   daysat +
                  daysun + daythu +
                  daytue + daywed +
```

```
                  s(FFMC) + s(DMC) + s(DC) + s(ISI) +
                  s(temp) + s(RH) + s(wind) +
                  s(rain, k=5) + ti(temp, RH, wind, rain), data=forest[-flds[[i]],])
  # prediction & RMSE
  sp_pred2 = exp(predict(sp.model2, newdata=forest[flds[[i]],])) - 1
  sp2_cv_rmse[i] = sqrt(sum((forest[flds[[i]],]$area - sp_pred2)^2) / nrow(forest))
  sp2_cv_AIC[i] = AIC(sp.model2)
}
  # Stop the clock
  sp2_time = proc.time() - start.time
sp.model2$call

  # interaction
  # Start the clock
  start.time <- proc.time()
for (i in 1:5) {
  sp.model3 <- gam(log(area + 1) ~ s(X, k=9) + s(Y, k=6) +
                   monthaug +
                   monthdec + monthfeb +
                   monthjan + monthjul +
                   monthjun + monthmar +
                   monthmay + monthnov +
                   monthoct + monthsep +
                   daymon +   daysat +
                   daysun + daythu +
                   daytue + daywed +
                   s(FFMC) + s(DMC) + s(DC) + s(ISI) +
                   s(temp) + s(RH) + s(wind) +
                   s(rain, k=5) + ti(FFMC, DMC), data=forest[-flds[[i]],])
  # prediction & RMSE
  sp_pred3 = exp(predict(sp.model3, newdata=forest[flds[[i]],])) - 1
  sp3_cv_rmse[i] = sqrt(sum((forest[flds[[i]],]$area- sp_pred3)^2) / nrow(forest))
  sp3_cv_AIC[i] = AIC(sp.model3)
}
  # Stop the clock
  sp3_time = proc.time() - start.time
sp.model3$call

sp_cv_overall_rmse = sqrt((sp_cv_rmse[1]^2 + sp_cv_rmse[2]^2 + sp_cv_rmse[3]^2 +
                           sp_cv_rmse[4]^2 + sp_cv_rmse[5]^2)/5)
sp2_cv_overall_rmse = sqrt((sp2_cv_rmse[1]^2 + sp2_cv_rmse[2]^2 + sp2_cv_rmse[3]^2 +
                            sp2_cv_rmse[4]^2 + sp2_cv_rmse[5]^2)/5)
sp3_cv_overall_rmse = sqrt((sp3_cv_rmse[1]^2 + sp3_cv_rmse[2]^2 + sp3_cv_rmse[3]^2 +
                           sp3_cv_rmse[4]^2 + sp3_cv_rmse[5]^2)/5)

sp.eval <- data.frame(RMSE = integer(3), AIC = integer(3), ProcessTime = integer(3))

sp.eval$RMSE <- c(sprintf("%.3f(+/-)%.3f", mean(sp_cv_rmse), sd(sp_cv_rmse)),
                  sprintf("%.3f(+/-)%.3f", mean(sp2_cv_rmse), sd(sp2_cv_rmse)),
                  sprintf("%.3f(+/-)%.3f", mean(sp3_cv_rmse), sd(sp3_cv_rmse)))
sp.eval$AIC <- c(sprintf("%.3f(+/-)%.3f", mean(sp_cv_AIC), sd(sp_cv_AIC)),
                 sprintf("%.3f(+/-)%.3f", mean(sp2_cv_AIC), sd(sp2_cv_AIC)),
                 sprintf("%.3f(+/-)%.3f", mean(sp3_cv_AIC), sd(sp3_cv_AIC)))
```

```r
sp.eval$ProcessTime <- c(sp_time[3], sp2_time[3], sp3_time[3])
rownames(sp.eval) <- c("model1", "model2", "model3")
library(knitr)
kable(sp.eval)

gam.check(sp.model)

gam.check(sp.model2)

gam.check(sp.model3)

library(randomForest)

  # tuning hypterparameter
seed = 100
set.seed(seed)
bestmtry <- tuneRF(data.matrix(X), log(data.matrix(y)+1), stepFactor=1.5, improve=1e-5, ntree=500, trac

  # Start the clock
  start.time <- proc.time()
for (i in 1:5) {
  # rf model
  rf.model = randomForest(log(area+1) ~ ., data=forest[-flds[[i]],], mtry=2, ntree=500)

  # rf variable importance
  rf.importance <- importance(rf.model)
  rf_cv_importance[,i] <- rf.importance

  # prediction & RMSE
  rf_pred = exp(predict(rf.model, newdata = forest[flds[[i]],])) - 1
  rf_cv_rmse[i] = sqrt(sum((forest[flds[[i]],]$area - rf_pred)^2) / nrow(forest))
}
  # Stop the clock
  rf_time = proc.time() - start.time

  # overall RMSE
rf_cv_overall_rmse = sqrt((rf_cv_rmse[1]^2 + rf_cv_rmse[2]^2 + rf_cv_rmse[3]^2 +
                           rf_cv_rmse[4]^2 + rf_cv_rmse[5]^2)/5)
print(paste("Random Forest RMSE: ", mean(rf_cv_rmse), "(+/-)", sd(rf_cv_rmse)))

  # Importance plot
library(ggplot2)
rf_mean_importance <- data.frame(rowMeans(rf_cv_importance))
vi1 <- ggplot(rf_mean_importance,
              aes(x=row.names(rf_mean_importance),
                  y=rowMeans.rf_cv_importance.)) +
         geom_bar(stat="identity") +
         labs(title="RandomForest Variable Importance",
              x = "Variables", y="Variable Importance") +
         geom_errorbar(aes(ymin = rowMeans.rf_cv_importance. - apply(rf_cv_importance, 1, sd),
                           ymax = rowMeans.rf_cv_importance. + apply(rf_cv_importance, 1, sd))) +
         coord_flip()
vi1
```

```
kable(rf_mean_importance[order(rf_mean_importance), , drop=FALSE])

library(gbm)

  # tuning hyperparameter
control <- trainControl(method="repeatedcv", number=5, repeats=1)
metric <- "RMSE"
tunegrid <- expand.grid(n.trees=c(100, 200, 300), interaction.depth = c(2, 4, 6),
                        shrinkage=c(0.01, 0.1, 1.0), n.minobsinnode = 10)

gbm_grid_seach = train(log(area+1) ~ ., data=forest, method="gbm", metric=metric,
                       tuneGrid=tunegrid, trControl=control)

  # Start the clock
  start.time <- proc.time()
for (i in 1:5) {
  # gbm model
  gbm.model <- gbm(log(area+1) ~., distribution = "gaussian", data=forest[-flds[[i]],], n.trees = 100,
                interaction.depth = 2, shrinkage = 0.01, n.minobsinnode = 10)

  # gbm prediction & RMSE
  gbm_pred = exp(predict(gbm.model, n.trees=100, newdata = forest[flds[[i]],])) -1
  gbm_cv_rmse[i] = sqrt(sum((forest[flds[[i]],]$area - gbm_pred)^2) / nrow(forest))

  # gbm variable importance
  gbm.importance <- summary(gbm.model)
  gbm_cv_importance[,i] = gbm.importance[row.names(gbm_cv_importance),]$rel.inf
}
  # Stop the clock
  gbm_time = proc.time() - start.time

gbm_cv_overall_rmse = sqrt((gbm_cv_rmse[1]^2 + gbm_cv_rmse[2]^2 + gbm_cv_rmse[3]^2 +
                            gbm_cv_rmse[4]^2 + gbm_cv_rmse[5]^2)/5)
print(paste("Gradient Boosting RMSE: ", mean(gbm_cv_rmse), "(+/-)", sd(gbm_cv_rmse)))

  # Importance plot
gbm_mean_importance <- data.frame(rowMeans(gbm_cv_importance))
vi2 <- ggplot(gbm_mean_importance,
              aes(x=row.names(gbm_mean_importance),
                  y=rowMeans.gbm_cv_importance.)) +
         geom_bar(stat="identity") +
         labs(title="Gradient Boosting Method Variable Importance",
              x = "Variables", y="Variable Importance") +
         geom_errorbar(aes(ymin = rowMeans.gbm_cv_importance. - apply(gbm_cv_importance, 1, sd),
                           ymax = rowMeans.gbm_cv_importance. + apply(gbm_cv_importance, 1, sd))) +
         coord_flip()
vi2

kable(gbm_mean_importance[order(gbm_mean_importance), , drop=FALSE])

  # Start the clock
  start.time <- proc.time()
for (i in 1:5) {
```

```r
  glm.model <- glm(log(area+1) ~ ., data = forest[-flds[[i]],], family = gaussian(link = "identity"))

  # prediction & RMSE
  glm_pred = exp(predict(glm.model, newdata=forest[flds[[i]],])) - 1
  glm_cv_rmse[i] = sqrt(sum((forest[flds[[i]],]$area - glm_pred)^2) / nrow(forest))

  # Importance
  glm.importace = coef(glm.model)[-1]
  glm_cv_importance[,i] <- glm.importace
}
  # Stop the clock
  glm_time = proc.time() - start.time

  # overall RMSE
glm_cv_overall_rmse = sqrt((glm_cv_rmse[1]^2 + glm_cv_rmse[2]^2 + glm_cv_rmse[3]^2 +
                              glm_cv_rmse[4]^2 + glm_cv_rmse[5]^2)/5)
print(paste("Logistic Regression RMSE: ", mean(glm_cv_rmse), "(+/-)", sd(glm_cv_rmse)))

  # Importance plot
glm_mean_importance <- data.frame(rowMeans(glm_cv_importance, na.rm=TRUE))
vi3 <- ggplot(glm_mean_importance,
              aes(x=row.names(glm_mean_importance),
                  y=rowMeans(glm_cv_importance, na.rm=TRUE))) +
         geom_bar(stat="identity") +
         labs(title="GLM Variable Importance",
              x = "Variables", y="Variable Importance") +
         geom_errorbar(aes(ymin = glm_mean_importance -
                             apply(glm_cv_importance, 1, function(x) {sd(x, na.rm=TRUE)}),
                           ymax = glm_mean_importance +
                             apply(glm_cv_importance, 1, function(x) {sd(x, na.rm=TRUE)}))) +
          coord_flip()
vi3

model_eval <- data.frame(RMSE = integer(4), ProcessTime = integer(4))
model_eval$RMSE <- c(sprintf("%.3f(+/-)%.3f", mean(rf_cv_rmse), sd(rf_cv_rmse)),
                     sprintf("%.3f(+/-)%.3f", mean(gbm_cv_rmse), sd(gbm_cv_rmse)),
                     sprintf("%.3f(+/-)%.3f", mean(sp2_cv_rmse), sd(sp2_cv_rmse)),
                     sprintf("%.3f(+/-)%.3f", mean(glm_cv_rmse), sd(glm_cv_rmse)))

model_eval$ProcessTime <- c(rf_time[3], gbm_time[3], sp2_time[3], glm_time[3])
rownames(model_eval) <- c("Random Forest", "Gradient Boosting",
                          "Smoothing Spline", "Generalized Linear Model")

kable(model_eval)

vi1
vi2
vi3
```

References

P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9. Available at: http://www.dsi.uminho.pt/~pcortez/fires.pdf