FIT5149 S1 2020 Assessment 1: Bushfire Analysis using Meteorological Data

Student information

- Family Name:Rane
- Given Name:Anand
- Student ID:29934176
- Student email:aran0021@student.monash.edu

Programming Language: R 3.5.1 in Jupyter Notebook

R Libraries used:

- psych - discribing the data
- ggplot2 - for plotting
- MASS - applied statistics
- freqdist - for frequency table
- rcompanion - for tukeys and coxbox transformation
- glmnet - For Lasso model
- leaps - for using subset genratation for backward selection
- data.table - for data tranformation
- mltools - for one hot encoding
- caret - for cross validation and step function
- glmnet - for penalised regression
- dplyr - for data mutation

Table of Contents

1. Introduction

- The following ipynb file contains data analysis of data which was collected from northeast region of Portugal between January 2000 and December 2003, which is 4 years worth of data. The dataset given contains 517 fire instances and 12 attributes which includes spatial coordinates of the map, month, day, FFWI indices and other meterological data. Here the target variable will be the area.
- The target variable to be predicted is highly imbalanced, it is easily evident by just eyeballing the data.
- In the below ipynb file, you will find the exploratory data analysis, followed by different models and their comparison.

Loading libraries

In [ ]:

```
# loading required library
library("psych")
library("ggplot2")
library("MASS")
library("freqdist")
library("rcompanion")
library("GGally")
library("glmnet")
library("leaps")
library("data.table")
library("mltools")
library("caret")
library("glmnet")
library("dplyr")
```

2. Data Exploration

Overview of the Training Dataset

In [ ]:

```
#loading the forestfire dataset
dataset=read.csv("forestfires.csv")

cat("The forestfires dataset has ",dim(dataset)[1],"rows and",dim(dataset)[2],"columns"
)

#display the structure
str(dataset)

#describing the statistics of dataset features
describe(dataset)

# #first 10 rows of the dataset
 head(dataset,10)
```

Summary of Attributes

Following list identifies which attributes are numerical and whether they are continuous or discrete, the data types and the comment on what it tells :

- X -> Integer type with mean at 4.66 and median at 4, spatial coordinate within the Montesinho park: 1 to 9
- Y -> Integer type with mean at 4.29 and median at 4, spatial coordinate within the Montesinho park: 2 to 9
- month -> Nominal data type, needs to be encoded using 1 hot encoding technique, wont be used in training the model
- day -> Nominal data type, wont be used in training the model
- FFMC -> numeric type, ranges from 18.7 to 96.20 and the mean and median are 90 and 91.6 respectively which suggests skewness
- DMC -> numeric type, ranges from 1.1 to 291.30 with mean and median lying in the middle
- DC -> numeric type, ranges from 7.9 to 860.60 with mean 547.94, suggests some skewness towards right
- ISI -> numeric type, ranges from 0.0 to 56.10 with mean 9.02, can have some outliers
- temp -> numeric type, temperature in celsius degrees: 2.2 to 33.30, might have strong correlation with target variable
- RH -> relative humidity in % i.e integer type, ranges from 15 to 100 with mean 44.28, can possibly have outliers
- wind -> numeric type, ranging 0.40 to 9.40
- rain -> outside rain in mm/m2, ranging 0.0 to 6.4 with mean 0.02166344, resemble highly skewed data, and outliers
- area -> target variable, contains alot of null values, as some area burned are very small(in ha)

In [ ]:

```
# plotting histogram, boxplot, and scatter plot for all the variables to see the distri
bution
par(mfrow = c(3,3))
dataset1 <- dataset[, -c(3:4)]
for (x in colnames(dataset1[-ncol(dataset)])){

    min_d <- min(dataset1[ , x])
    max_d <- max(dataset1[ , x])
    b <- seq(min_d, max_d, length.out = 20)
    hist(dataset[ , x], col = rgb(0, 1, 0, 0.35), breaks = b, main = paste("Histogram:
 ", x, sep = ""), xlab = x)
    boxplot(dataset[ , x], main = paste("Boxplot: ", x, sep = ""))
    plot(dataset[ , 13],dataset[ , x], main = paste("scatter plot: ", x),
         xlab="area",ylab=x)
}
```

These graphs above depicts:

- Histogram for FFMC, DC show right skewness and can have potential outliers.
- ISI, rain and area (target variable) are left skewed displaying majority of the record on the left side of the graph.
- Out of all the variables, temp variable shows the most uniform spread and has low number of outliers in the boxplot.
- The X and Y are spatial coordinates and do really resemble anything.
- We can see that the area variable is especially skewed as majority of the fire burned areas are close to zero i.e null. We can try and remove the null records so as to judge the distribution of the target variable.

To reduce skewness and improve symmetry, the logarithm function y = ln(x), which is a common transformation that tends to improve regression results for right-skewed targets, is applied to the area attribute

In [ ]:

```
for (i in seq_along(dataset$area))
{
    if (dataset$area[i]>0){
        dataset$logarea[i] <- log(dataset$area[i])
    }
    else{
        dataset$logarea[i] <- NA
    }
}
```

In [ ]:

```
#plotting the graph to check the new distribution using the smooth line
ggplot(dataset, aes(x = logarea)) +
  geom_histogram(aes(y = ..density..), colour = "black", fill = "White") +
  stat_function(fun = dnorm, args = list(mean = mean(dataset$logarea,na.rm=TRUE), sd =
sd(dataset$logarea,na.rm=TRUE)),col = 'red')
```

In [ ]:

```
mean = mean(dataset$logarea,na.rm=TRUE)
sd = sd(dataset$logarea,na.rm=TRUE)

cat("The mean of the distribution is",mean,"and the standard deviation is",sd)
dataset=dataset[,-14]
```

Correlation Plot Function

Now lets separate the non-zero entries in the area variable, and exclude the non-numeric data, for buidling a correlation plot function.

In [ ]:

```
nonzero_data = dataset[which(dataset$area > 0),]
nonzero_data = nonzero_data[, -c(3:4)]
```

In [ ]:

```
colorRange <- c('#69091e', '#e37f65', 'white', '#aed2e6', '#042f60')

myColorRampFunc <- colorRamp(colorRange)

panel.cor <- function(w, z, ...) {
    correlation <- cor(w, z)

    ## because the func needs [0,1] and cor gives [-1,1], we need to shift and scale it
    col <- rgb(myColorRampFunc((1 + correlation) / 2 ) / 255 )

    ## square it to avoid visual bias due to "area vs diameter"
    radius <- sqrt(abs(correlation))
    radians <- seq(0, 2*pi, len = 50) # 50 is arbitrary
    x <- radius * cos(radians)
    y <- radius * sin(radians)
    ## make them full loops
    x <- c(x, tail(x,n=1))
    y <- c(y, tail(y,n=1))

    ## trick: "don't create a new plot" thing by following the
    ## advice here: http://www.r-bloggers.com/multiple-y-axis-in-a-r-plot/
    ## This allows
    par(new=TRUE)
    plot(0, type='n', xlim=c(-1,1), ylim=c(-1,1), axes=FALSE, asp=1)
    polygon(x, y, border=col, col=col)
}

pairs(nonzero_data[sample.int(nrow(nonzero_data),270),], lower.panel=panel.cor, panel=
panel.smooth)
```

- However we can take a closer look at the correlation between FFMC, DMC, DC, ISI and area.

In [ ]:

```
tempdata = nonzero_data[, c(3:6,11)]
# Check correlations (as scatterplots), distribution and print corrleation coefficient
ggpairs(tempdata, title="correlogram with ggpairs()")
```

- From the above correlation matrix we can see that no two features are strongly correlating with each other or to the target variable. We can see DMC and DC correlates with a coefficient of 0.67, while FFMC correlates DC with coefficient of 0.41 and the highest correlation between FFMC and ISI with coeeficient of 0.7.
- We can see no strong correlation with the target variable, we can notice that temp correlates with area with coefficient of 0.11, which is highest as compared to other variables.

Analyzing the month and day variable w.r.t. area

In [ ]:

```
#analysing the monthly burned area over the span of 4 years.
ggplot(data=dataset, aes(x=month, y=area))+geom_bar(stat="identity")
```

The above bar plot gives us a interesting detail about the scenario of forest fires across different months in portugal. Well in our case we have data from 2000 to 2003, so by looking at the above graph, we can see that September and August are the months where severe forest fires had happened, while in the month of January and November no significant forest fires were spectated, we can expect such a behaviour since Portugal has summer season from June - September and is generally very hot. Focusing more on Sept and Aug can help narrow down our exploration.

In [ ]:

```
ggplot(data=dataset, aes(x=day, y=area))+geom_bar(stat="identity")
```

Accordingly I also checked for day, to see if we get any insights, we can see that Saturday had the most area burned, it could be a chance or a bias due to an outlier.

Now, from the first bargraph we can further narrow down our analysis month wise, and it may end up giving us more evidence and adequate data to make some predictions.

In [ ]:

```
month_data = dataset[which(dataset$area > 0),]

month_data = month_data[month_data$month == 'sep' | month_data$month == 'aug',]
```

In [ ]:

```
pairs(month_data[sample.int(nrow(month_data),196),], lower.panel=panel.cor, panel= pane
l.smooth)
```

In [ ]:

```
library(corrplot)
corrplot(cor(month_data[,-c(3,4)]),"number")
```

3. Model Development

First Model : Linear model on all the numeric variables

- Converted the month variable to numeric data by using one-hot encoding method
- excluded day variable as showed no strong significance.
- No tranformation on any variable, no even on the target variable

Converting the categorical variables from the dataset and building the model with just numeric data and no tranformations

In [ ]:

```
#removing day from the dataset
dataset_1=dataset[,c(-4)]
#converting the dataset by encoding
dataset_1<- one_hot(as.data.table(dataset_1))
```

In [ ]:

```
# divide testing and training
# 80% train and 20% test
set.seed(123)
sample_size <- floor(0.8 * nrow(dataset_1))
train_ind <- sample(seq_len(nrow(dataset_1)), size = sample_size)

train.data <- dataset_1[train_ind,]
test.data <- dataset_1[-train_ind,]
```

In [ ]:

```
# splitting the dataset to train and test label and data
train_data <- as.matrix(train.data[,-11])
train_label <- as.matrix(train.data[,11], drop=false)
test_data <- as.matrix(test.data[,-11])
test_label <- as.matrix(test.data[,11], drop=false)
```

In [ ]:

```
#fitting the linear model on the train data
model_1 <- lm(area~. , data = train.data)
model_1.sum <- summary(model_1)
print(model_1.sum)
```

In [ ]:

```
#plotting the model
par(mfrow = c(2,2))
plot(model_1)
```

In [ ]:

```
# lets do prediction on model 1
y_pred_1 = predict.lm(model_1, newdata = test.data)
MAE= mean(abs(y_pred_1-test_label))
cat("The MAE for the second model is:",MAE)
cat("\nThe R2 for the second model is:",summary(model_1)$adj.r.square)
```

Using the step-wise selection function for both directions

In [ ]:

```
model_1=step(lm(area~X+Y+month_apr+month_aug+month_dec+month_feb+month_jan+month_jul+mo
nth_jun+month_mar+month_may+month_nov+month_oct+month_sep+FFMC+DMC+DC+ISI+temp+RH+wind+
rain,data=train.data),direction="both")
```

In [ ]:

```
#summarizing the model
model_1.sum <- summary(model_1)
print(model_1.sum)
```

In [ ]:

```
#predicting and calculating the error
y_pred_1 = predict.lm(model_1, newdata = test.data)
MAE_1=MAE((y_pred_1),(test_label))
cat("The MAE for the second model is:",MAE_1)
cat("\nThe R2 for the second model is:",summary(model_1)$adj.r.square)
```

Training model with interaction variables using step-wise function

In [ ]:

```
#creating a form of all the possible combinations of interaction variable
form = reformulate(apply(combn(names(dataset_1)[-grep("area", names(dataset_1))], 2), 2
, paste, collapse="*"), "area")
form
```

In [ ]:

```
#training the model using step function
# mod = step(lm(form, data=dataset_1),direction="both")
# summary(mod)
```

In [ ]:

```
#saving the significant interaction variables generated by step function in a variable.
form = area ~ X + Y + month_aug + DMC + temp + RH + wind +
    X:temp + Y:wind + DMC:RH

#now organizing the model and printing the summary of the model.
model_1=lm(form, data=dataset_1)
summary(model_1)
```

In [ ]:

```
#predicting and calculating the error
y_pred_1 = predict.lm(model_1, newdata = test.data)
MAE_1=MAE(y_pred_1,test_label)
cat("The MAE for the second model is:",MAE_1)
cat("\nThe R2 for the second model is:",summary(model_1)$adj.r.square)
```

Second model: Removing spacial coordinates and categorical value, and applying transformations

- We remove month and day variable as they dont show any significance.
- Carrying out various tranformations on the imbalanced and skewed variables to have a even spread and to improve the predictions
- Also removing the X and Y which are the spacial coordinates of the area burned.

In [ ]:

```
#Excluding all the spacial variables and categorical variables
dataset_2 = dataset[,-c(4,3,1,2)]
```

Log transforming the target variable

In [ ]:

```
#for loop for tranforming the area
for (i in seq_along(dataset$area))
{
    dataset_2$area[i] <- log(dataset_2$area[i]+1)
}
```

In [ ]:

```
#calculating the frequency table
freq<-freqdist(dataset_2$FFMC)
#setting them in decending order to see the highest frequency
df <-head(freq[order(-freq$frequencies),],5)
df #frequency table
```

Using the tukey's tranformation on FFMC variable

In [ ]:

```
dataset_2$FFMC=transformTukey(dataset_2$FFMC,plotit=FALSE)
#sizing the plot
options(repr.plot.width=5, repr.plot.height=5)
plotNormalHistogram((dataset_2$FFMC))
```

Using Tukey's tranformation on DMC

In [ ]:

```
x<-freqdist(dataset_2$DMC)
df <-head(x[order(-x$frequencies),],10)
dataset_2$DMC=transformTukey(dataset_2$DMC,plotit=FALSE)
plotNormalHistogram((dataset_2$DMC))
```

Using Box and cox tranformation method for DC, as other tranformations did not work quite well

In [ ]:

```
# Transform Turbidity as a single vector and  # Try values -6 to 6 by 0.1
Box = boxcox(dataset_2$DC ~ 1, lambda = seq(-6,6,0.1))
Cox = data.frame(Box$x, Box$y)            # Create a data frame with the results
Cox2 = Cox[with(Cox, order(-Cox$Box.y)),] # Order the new data frame by decreasing y
lambda = Cox2[1, "Box.x"]                 # Extract that lambda
dataset_2$DC = (dataset_2$DC ^ lambda - 1)/lambda   # Transform the original data
plotNormalHistogram(dataset_2$DC)
```

Using the Tukey's tranformation on ISI

In [ ]:

```
dataset_2$ISI=transformTukey(dataset_2$ISI,plotit=FALSE)
plotNormalHistogram((dataset_2$ISI))
```

Using Tukey's tranformation on rain

In [ ]:

```
dataset_2$rain=transformTukey(dataset_2$rain,plotit=FALSE)
plotNormalHistogram((dataset_2$rain))
```

Building the model

In [ ]:

```
# divide testing and training
# 80% train and 20% test
set.seed(123)

sample_size <- floor(0.8 * nrow(dataset_2))

train_ind <- sample(seq_len(nrow(dataset_2)), size = sample_size)

train.data <- dataset_2[train_ind,]
test.data <- dataset_2[-train_ind,]
```

In [ ]:

```
# spliting data and labels seprate.
train_data <- as.matrix(train.data[,-9])
train_label <- as.matrix(train.data[,9], drop=false)
test_data <- as.matrix(test.data[,-9])
test_label <- as.matrix(test.data[,9], drop=false)
```

In [ ]:

```
#fitting the model 2
model_2 <- lm(area~. , data = train.data)
model_2.sum <- summary(model_2)
print(model_2.sum)
```

In [ ]:

```
#predicting and calculating the error
y_pred_2 = predict.lm(model_2, newdata = test.data)
MAE=MAE((exp(y_pred_2)-1),(exp(test_label)-1))
Rsquare = R2(y_pred_2, test_label)
cat("The MAE for the second model is:",MAE)
cat("\nThe R2 for the second model is:",Rsquare)
```

In [ ]:

```
#step function
model_2=step(lm(area~FFMC+DMC+DC+ISI+temp+RH+wind+rain,data=train.data),direction="bot
h")
```

In [ ]:

```
#printing in the summary of the model
model_2.sum <- summary(model_2)
print(model_2.sum)
```

In [ ]:

```
#predicting and calculating the error
y_pred_2 = predict.lm(model_2, newdata = test.data)
MAE=MAE((exp(y_pred_2)-1),(exp(test_label)-1))
Rsquare = R2(y_pred_2, test_label)
cat("The MAE for the second model is:",MAE)
cat("\nThe R2 for the second model is:",Rsquare)
```

Now, lets try and improve the accuracy using interaction variables

In [ ]:

```
#creating a form of all the possible combinations of interaction variable
form = reformulate(apply(combn(names(dataset_2)[-grep("area", names(dataset_2))], 2), 2
, paste, collapse="*"), "area")
form
```

- We use the step-wise selection method to toggle through all the different combinations of independent variables and interaction variables, using both forward and backward propogation.
- The below code takes couple of minutes to run, as it has to try out all the different variables, both forward and backward.

In [ ]:

```
#training the model using step function
# mod = step(lm(form, data=dataset_2),direction="both")
# summary(mod)
```

In [ ]:

```
#saving the significant interaction variables generated by step function in a variable.
form = area ~ DC + temp + RH + wind + rain + DC:RH + temp:wind +
    temp:rain

#now organizing the model and printing the summary of the model.
model_2=lm(form, data=dataset_2)
summary(model_2)
```

In [ ]:

```
#plotting the model and checking its performance
par(mfrow = c(2,2))
options(repr.plot.width=7, repr.plot.height=7)
plot(model_2)
```

In [ ]:

```
#predicting and calculating the error
y_pred_2 = predict.lm(model_2, newdata = test.data)
MAE_2=MAE((exp(y_pred_2)-1),(exp(test_label)-1))
Rsquare_2 = R2(y_pred_2, test_label)
cat("The MAE for the second model is:",MAE_2)
cat("\nThe R2 for the second model is:",Rsquare_2)
```

- Above, we are predicting the areas for the test data which we had excluded before training the model and also checking the mean square error of the model.
- Higher R-square value and lower MAE resembles our model is good and reliable.
- We have managed to get a mean absolute error of 8.81 R2 of 0.02

Third Model : Using Lasso, Ridge and Elastic net regression

In [ ]:

```
#Removing the outliers detected from the above model
dataset_3 = dataset_2[-c(239,416,480,500,4),]
#excluding the days as it seems quite general and doesnt seem to have a major effect.
```

In [ ]:

```
#penalizing factor
lambda <- 10^seq(-3, 3, length = 100)
```

In [ ]:

```
# divide testing and training
# 80% train and 20% test
set.seed(123)

sample_size <- floor(0.8 * nrow(dataset_3))

train_ind <- sample(seq_len(nrow(dataset_3)), size = sample_size)

train.data <- dataset_3[train_ind,]
test.data <- dataset_3[-train_ind,]
```

In [ ]:

```
# Building the ridge regression model
set.seed(123)
ridge <- train(
  area ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 0, lambda = lambda)
  )
# Model coefficients
coef(ridge$finalModel, ridge$bestTune$lambda)
# Make predictions
predictions <- ridge %>% predict(test.data)
# Model prediction performance
data.frame(
    MAE_ridge = MAE((exp(predictions)-1), (exp(test.data$area)-1)),
  Rsquare_ridge = R2(predictions, test.data$area)
)
```

In [ ]:

```
#building the lasso regression model
set.seed(123)
lasso <- train(
  area ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = lambda)
  )
# Model coefficients
coef(lasso$finalModel, lasso$bestTune$lambda)
# Make predictions
predictions <- lasso %>% predict(test.data)
# Model prediction performance
data.frame(
    MAE_lasso = MAE((exp(predictions)-1), (exp(test.data$area)-1)),
  Rsquare_lasso = R2(predictions, test.data$area)
)
```

In [ ]:

```
# Building the elastic net regression model
set.seed(123)
elastic <- train(
  area ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneLength = 10
  )
# Model coefficients
coef(elastic$finalModel, elastic$bestTune$lambda)
# Make predictions
predictions <- elastic %>% predict(test.data)
# Model prediction performance
data.frame(
    MAE_elastic = MAE((exp(predictions)-1), (exp(test.data$area)-1)),
  Rsquare_elastic = R2(predictions, test.data$area)
)
```

4. Model Comparison

From the above trained models and their predictions we now resort to model comparison and see which model is the best and which should be choosen or is more reliable when it comes to predicting forest fires.

- Model 1 : Simple Linear Regression with all the variables
  - In here, we have included on the variables for the predictions and have used one hot encoding for converting the month variable.
  - After running the simple model, then running step-wise selection process and then followed by trying out possible interaction variables, we managed to get a Mean absolute error of 12.001 and R-squared value of 0.021
- Model 2: Linear Regression by removing spacial coordinates and categorical value, and applying transformations
  - In here, we have excluded the spacial coordinates, month and days from the dataset, as it had no correlation with the target.
  - Here we tranformed all the skewed and unbalanced variables including the target variable, then using step function and interaction variables, we achieved a better MAE of 8.81 and R-squared value of 0.022
- Model 3: Lasso, Ridge and Elastic net Regression (Penalized Regression)
  - In this model, there are further more three models which are penalized regression, it involves adding a regularization term or lambda to the cost function of the model.
  - For this we carry forwarded the tranformed dataset for training these models. On doing this we ended up with a MAE of 9.45 and R-squared value 0.005 for ridge, 0.0022 for lasso and 0.0024 for elastic net.

Comparison :

So in model 1, I included all the numeric variables without the tranforming them to check how well it predicts the area and how low the mean absolute error is. I ran step wise function with interaction variables to get to the best MAE possible that that was 12.001, then in the second model I removed the insignificant variables like X,Y,Month and day and ran tranformation like tukley and boxcox to normalize the predictor variable, and as expected it improved and MAE came down to 8.81 which is by far the lowest. For the third model, I went ahead and trained three penalized regression models, by setting a lambda of -3 to +3, after adding the penalty factor and using glmnet I got a MAE of 9.45 which is higher than the previous model. Hence I will go for Model 2 for predicting the area burned for any unknown values.

5. Variable Identification and Explanation

Below is the explaination for choosing the variables that have been used in the training of all the three models:

- 1st Model : For this model, I got some insights from the Exploratory data analysis that I had prior to building the models. I eyeballed that the area which is the target variable for the model is very unbalanced. Then I saw that the day variable correlated very poor with the target variable and so I decided to omit it. For the month variable I used the one-hot encoding method to convert to numeric values. Moving on, I used linear regression, next I used step function and then followed by interaction variables, it finally fetched me a MAE of 12, previously it gave me a MAE of 18.5 which is much higher.
- 2nd Model : For this model, I decided to omit spatial coordinates, month and the day, which by common sense cannot help us predict the area burned. Then I applied tranformations like log, tukley and box and cox for the significant variables like FFMC, DC,DMC, ISI and rain, and also since they were badly unbalanced or skewed. So, number of variables for prediction used were less as 8 and total 9 including the target variable.
- 3rd Model : For this model, I used the same predictors from the second model and removed the 5 outliers that I noticed on plotting the 2nd model. The lambda used had a range -3 to +3. They performed poorly in comparison to the 2nd model.

6. Conclusion

- In conclusion, I think that the second model with tranformations which gave so far the best MAE is the one I would use for predicting the area burned.
- Adding to that, I believe the dataset is very biased to small area burned, if the question had asked to predict the small area burned then, I would have directly omitted the high burned area entries from the dataset, as they are very less in comparison to small area burned entries.
- Right now, all the three models are sitting somewhere in middle of predicting large and small area burned.
- The R2 values are pretty crazy and absurd there is no way in seeing an ideal accuracy anywhere above 60%.

7. References

- http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/#ridge-regression (http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/#ridge-regression)
- https://www.kaggle.com/sazack/forest-fire-burned-area-prediction (https://www.kaggle.com/sazack/forest-fire-burned-area-prediction)
- http://www.columbia.edu/~yh2693/ForestFire.html (http://www.columbia.edu/~yh2693/ForestFire.html)