

# Final Project\_Team 7\_Gabi

Gabi Rivera

## Sensor-Fusion Smoke Detection Classification

The goal is to devise a Machine Learning model that will detect smoke through the use of IoT data to trigger a fire alarm.

### Information about the Features:

*Air Temperature*

*Air Humidity*

*TVOC*: Total Volatile Organic Compounds; measured in parts per billion (Source)

*eCO2*: co2 equivalent concentration; calculated from different values like TVCO

*Raw H2*: raw molecular hydrogen; not compensated (Bias, temperature, etc.)

*Raw Ethanol*: raw ethanol gas (Source)

*Air Pressure*

*PM 1.0 and PM 2.5*: particulate matter size < 1.0  $\mu\text{m}$  (PM1.0). 1.0  $\mu\text{m}$  < 2.5  $\mu\text{m}$  (PM2.5)

*Fire Alarm*: ground truth is “1” if a fire is there

*CNT*: Sample counter

*UTC*: Timestamp UTC seconds

*NC0.5/NC1.0 and NC2.5*: Number concentration of particulate matter. This differs from PM because NC gives the actual number of particles in the air. The raw NC is also classified by the particle size: < 0.5  $\mu\text{m}$  (NC0.5); 0.5  $\mu\text{m}$  < 1.0  $\mu\text{m}$  (NC1.0); 1.0  $\mu\text{m}$  < 2.5  $\mu\text{m}$  (NC2.5);

### Pre-Processing:

```
# List of libraries
library(caret)
library(tidyverse)
library(naniar)
library(gt)
library(ggplot2)
```

```

library(dplyr)
library(tidyr)
library(GGally)
library(corrplot)
library(e1071)
library(tibble)
library(MASS)
library(mice)
library(reshape2)
library(ROSE)
library(pROC)

```

Note:

- Train dataset has 5000 observations, 14 predictors, and an outcome variable.
- Test dataset has 12437 observations and 14 predictors.

```

# Upload datasets
train <- read.csv("train_dataset.csv")
test <- read.csv("test_dataset.csv")

```

```

# Use train_dataset moving forward
smokedf <- train
str(smokedf)

```

```

'data.frame':   5000 obs. of  15 variables:
 $ UTC          : int  1655127646 1654734418 1654714047 1654715196 1655125243 ...
 $ Temperature.C.: num  15.1 27.1 26.4 26 -1.2 ...
 $ Humidity...   : num  43 54.8 45.8 48.4 41.4 ...
 $ TVOC.ppb.    : int  199 0 144 180 76 58 60000 81 656 223 ...
 $ eCO2.ppm.   : int  426 400 409 431 400 420 9147 447 400 467 ...
 $ Raw.H2       : int  12775 13058 12784 12771 12791 12834 11410 12785 13732 12747 ...
 $ Raw.Ethanol  : int  20524 19961 20580 20537 20673 20724 16840 20690 20533 20502 ...
 $ Pressure.hPa.: num  937 940 937 937 938 ...
 $ PM1.0        : num  1.55 0.21 1.97 1.93 1.9 ...
 $ PM2.5        : num  1.61 0.22 2.05 2.01 1.97 ...
 $ NC0.5        : num  10.66 1.46 13.59 13.31 13.08 ...
 $ NC1.0        : num  1.663 0.228 2.118 2.075 2.04 ...
 $ NC2.5        : num  0.038 0.005 0.048 0.047 0.046 ...
 $ CNT          : int  3338 1087 1860 3009 935 244 5222 677 5693 3353 ...
 $ Fire.Alarm   : int  0 0 0 0 0 0 0 0 0 0 ...

```

```

class(smokedf)

[1] "data.frame"

#summary(smokedf)

# Missing Values
smokedf[smokedf == ""] <- NA
na_value <- sapply(smokedf, function(x) sum(is.na(x)))
predictors_with_missing <- names(na_value[na_value > 0])

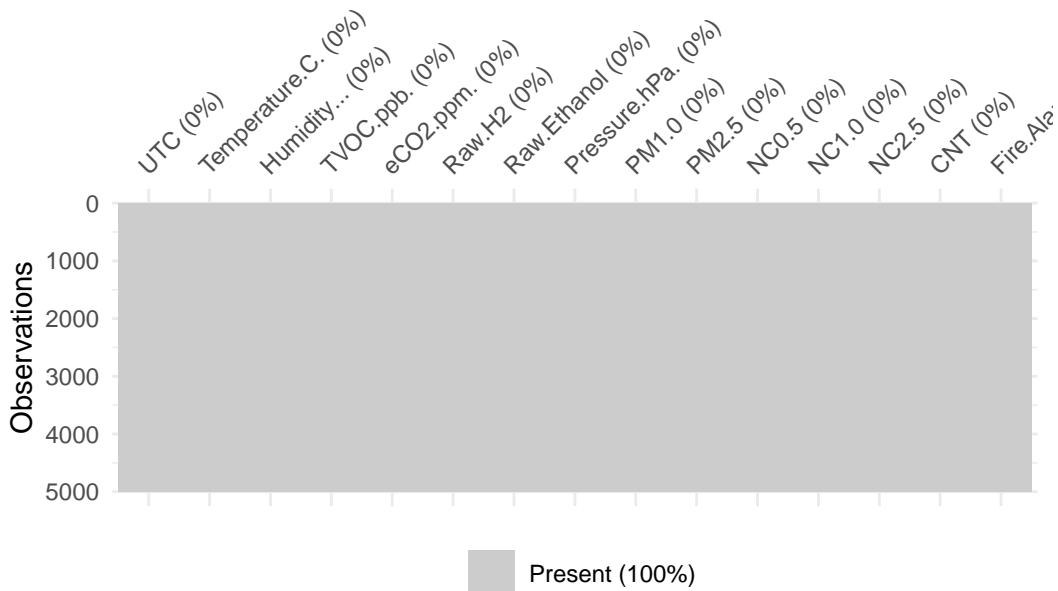
missing_values_table <- data.frame(Predictor = predictors_with_missing,
Missing_Values = na_value[predictors_with_missing])
missing_values_table |> head() |> gt() |>
  tab_header(title = "Predictors with Missing Values")

```

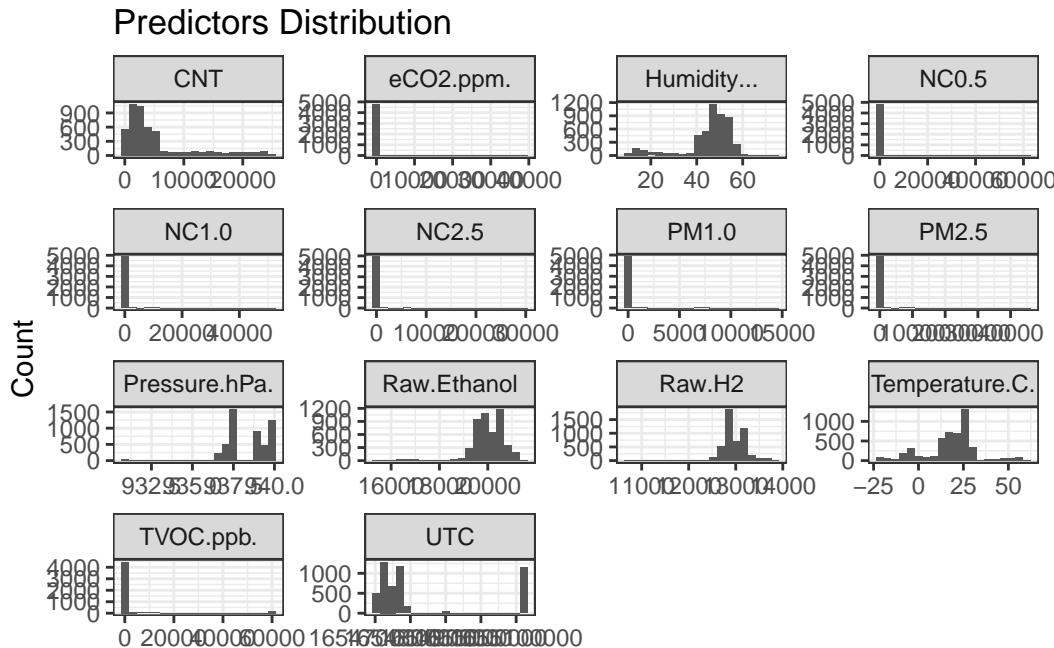
Predictors with Missing Values

Predictor	Missing_Values
UTC (0%)	0

```
vis_miss(smokedf)
```

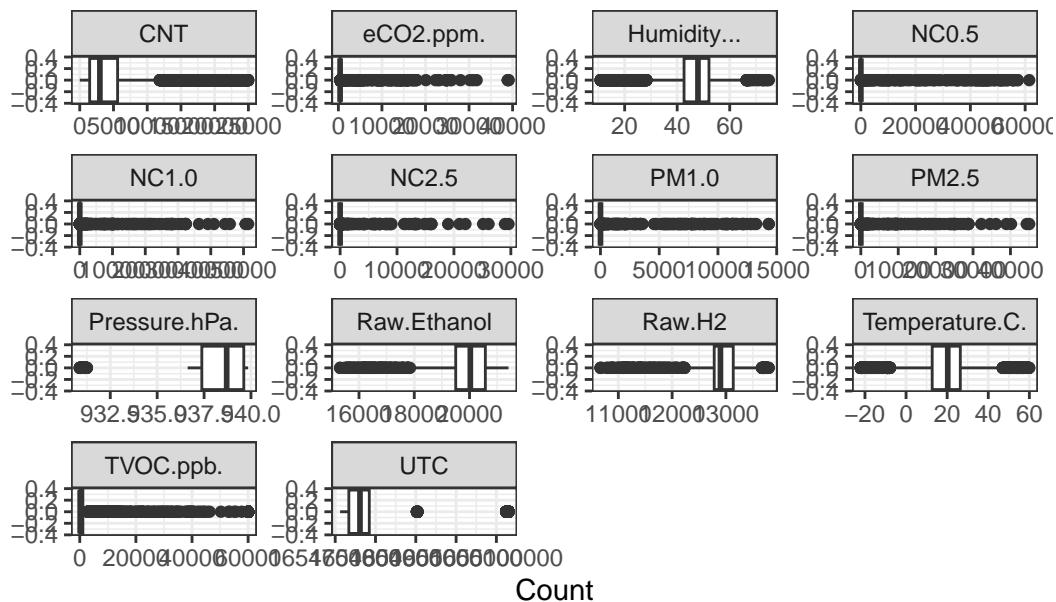


```
# Histogram of each predictors
smokedf |>
  pivot_longer(~Fire.Alarm, names_to = 'Element', values_to = 'value') |>
  ggplot(aes(x = value)) +
  geom_histogram(bins = 20) +
  facet_wrap(~Element, scales = 'free', ncol = 4) +
  theme_bw() +
  labs(title = 'Predictors Distribution', x = NULL, y = "Count")
```



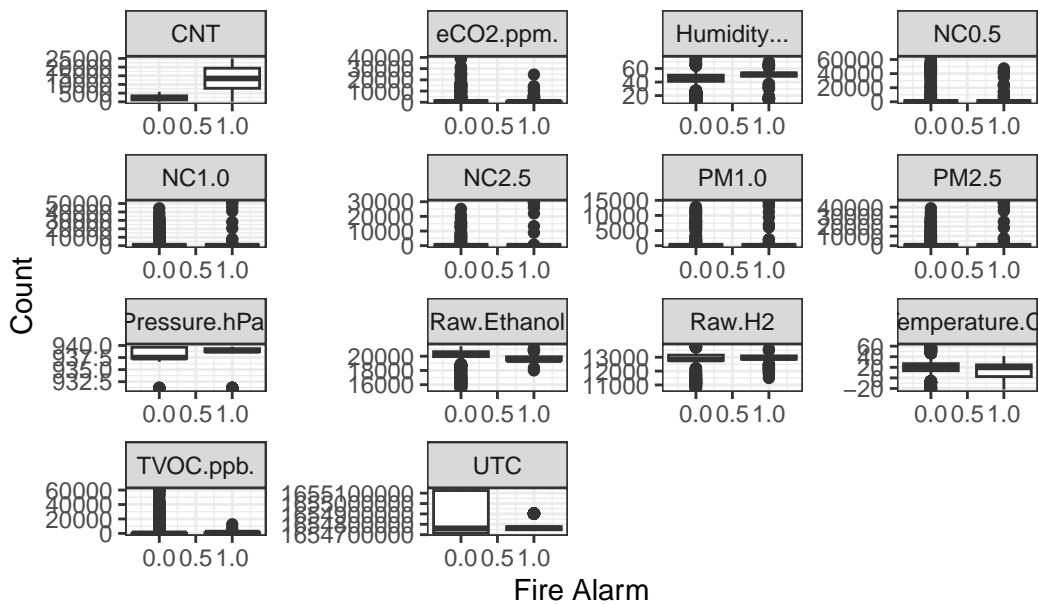
```
# Create box plots for each predictors:
smokedf |>
  pivot_longer(~Fire.Alarm, names_to = 'Element', values_to = 'value') |>
  ggplot(aes(value)) +
  geom_boxplot() +
  facet_wrap(~Element, scales = "free", ncol = 4) +
  theme_bw() +
  labs(title = 'Box Plots of Predictors', x = "Count")
```

## Box Plots of Predictors



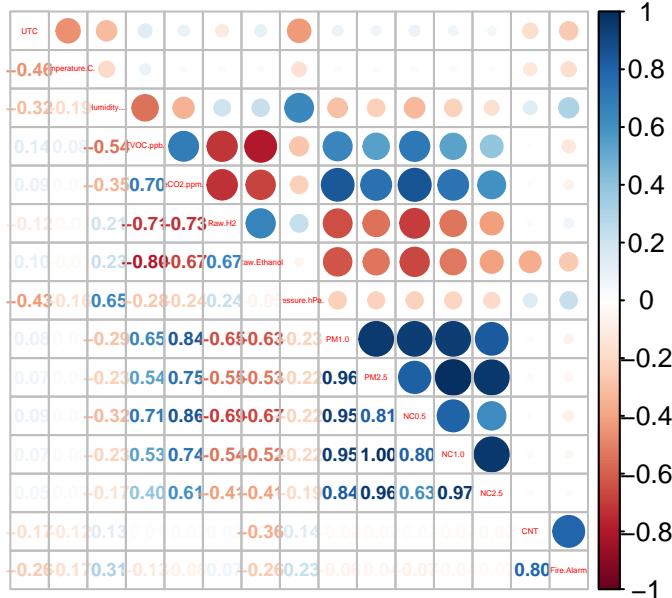
```
# Create box plots of response for each predictors:
smokedf |>
  pivot_longer(~Fire.Alarm, names_to = 'Element', values_to = 'value') |>
  ggplot(aes(Fire.Alarm, value, group = Fire.Alarm)) +
  geom_boxplot() +
  facet_wrap(~Element, scales = "free", ncol = 4) +
  theme_bw() +
  labs(title = 'Box Plots of Predictors over Response', x = "Fire Alarm", y = "Count")
```

## Box Plots of Predictors over Response



```
# Correlation plot including outcome variable
smokedf |>
  mutate(Fire.Alarm = as.numeric(Fire.Alarm)) |>
  cor() |>
  corrplot.mixed(title = "Correlation Plot Between Outcome and Predictors",
    tl.cex = .25, number.cex = 0.6, mar = c(1, 1, 1, 1))
```

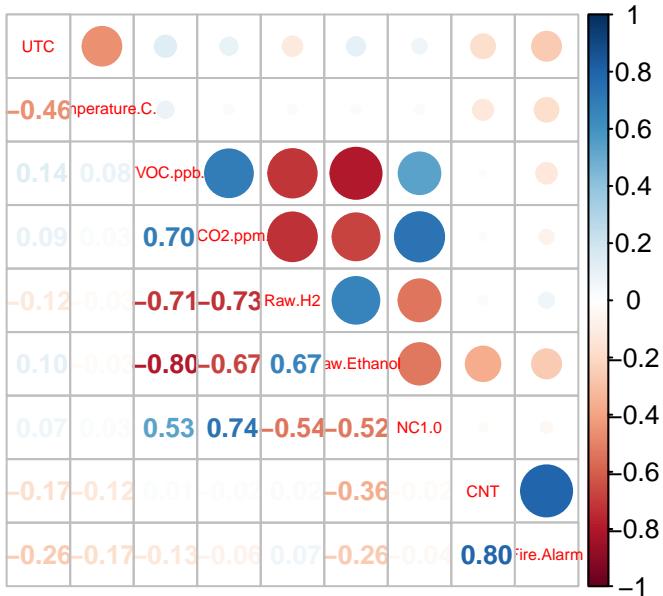
## Correlation Plot Between Outcome and Predictors



```
# Remove highly correlated predictors
correlation_matrix <- cor(smokedf[, -1])
highly_correlated <- findCorrelation(correlation_matrix, cutoff = 0.75)
highly_correlated_names <- colnames(smokedf[, -1])[highly_correlated]
smokedf_reduced <- smokedf[, -highly_correlated]

# Correlation plot including outcome variable
smokedf_reduced |>
  mutate(Fire.Alarm = as.numeric(Fire.Alarm)) |>
  cor() |>
  corrplot.mixed(title = "Reduced Correlation Plot",
                 tl.cex = .5, number.cex = 0.8, mar = c(1, 1, 1, 1))
```

## Reduced Correlation Plot



```
# Calculate skewness of each predictors
# Skew values less than ±0.5 should be considered 'normal enough'
skew_fa <- smokedf_reduced |>
  dplyr::select(-Fire.Alarm) |>
  map_dbl(skewness) |> round(3)

skew_fa_tibble <- tibble(Variable = names(skew_fa), Skewness = skew_fa)
skew_fa_tibble |> gt()
```

Variable	Skewness
UTC	1.211
Temperature.C.	-0.303
TVOCP.ppb.	4.248
eCO2.ppm.	8.395
Raw.H2	-2.335
Raw.Ethanol	-2.229
NC1.0	10.232
CNT	1.669

```
# Determine zero values in the datafram
zero_counts <- sapply(smokedf, function(x) sum(x == 0, na.rm = TRUE))
```

```

# Perform Box Cox Transformation Analysis and determine predictors with >50% improvement
bct_test <- function(x, property = 'skew') {
  stopifnot(property %in% c('skew', 'lambda'))

  x <- x[which(! is.na(x))]
  x2 <- x + ifelse(any(x == 0), 0.0001, 0)

  bct <- BoxCoxTrans(x2)
  x_trans <- predict(bct, x2)
  if (property == 'skew') return(e1071::skewness(x_trans))
  return(bct$lambda)

  skew_smfa_bct <- smokedf_reduced |>
    dplyr::select(-Fire.Alarm) |>
    map_dbl(bct_test)

  bct_analysis <- tibble(
    Property = names(skew_fa),
    `Original Skew` = skew_fa,
    `Skew after BoxCox` = round(skew_smfa_bct, 4),
    Lambda = smokedf_reduced |> dplyr::select(-Fire.Alarm) |>
      map_dbl(~ bct_test(.x, 'lambda')))

  bct_keep <- bct_analysis |>
    filter(abs(`Original Skew`) > 0.5,
          abs(`Skew after BoxCox`) < 0.5)

  bct_keep |>
  gt::gt()
}

```

Property	Original Skew	Skew after BoxCox	Lambda
CNT	1.669	0.0728	0.2

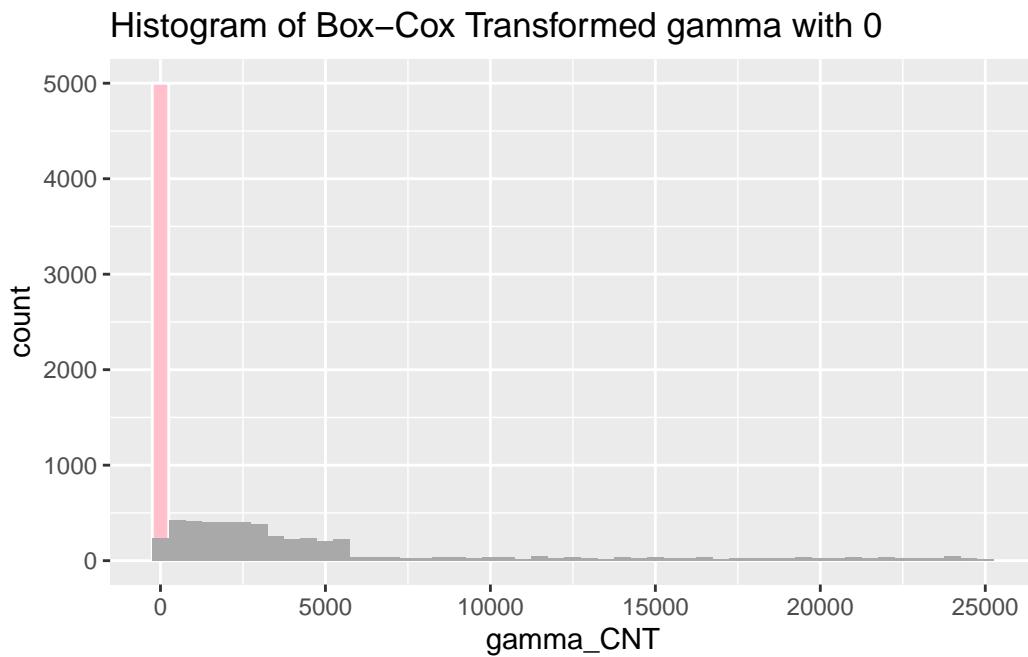
```

# Apply BCT Analysis result in reduced dataframe
bct <- BoxCoxTrans(smokedf_reduced$CNT +
  ifelse(any(smokedf_reduced$CNT == 0), 0.0001, 0))
smokedf_reduced$gamma_CNT <- predict(bct, smokedf_reduced$CNT)

# Plot Changes
smokedf_reduced |> ggplot(aes(x = gamma_CNT)) +

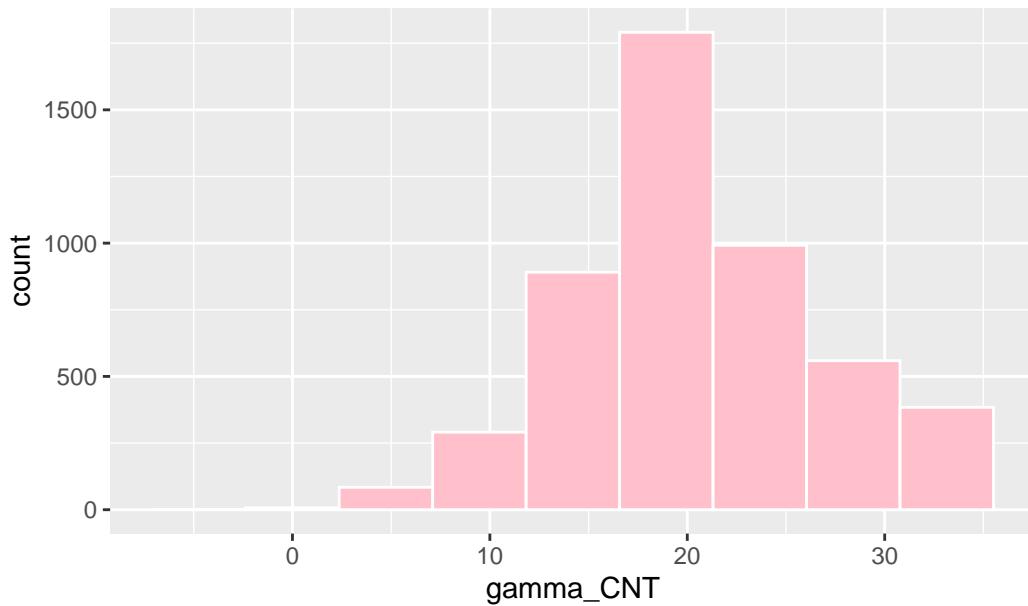
```

```
geom_histogram(binwidth = 500, fill = "pink", color = "white") +  
  geom_histogram(aes(x = CNT), binwidth = 500, fill = "darkgray", alpha = 10) +  
  labs(title = "Histogram of Box-Cox Transformed gamma with 0")
```



```
smokedf_reduced |>  
  ggplot(aes(x = gamma_CNT)) +  
  geom_histogram(bins= 9, color = "white", fill = "pink") +  
  labs(title ="Histogram of Box-Cox Transformed gamma_CNT")
```

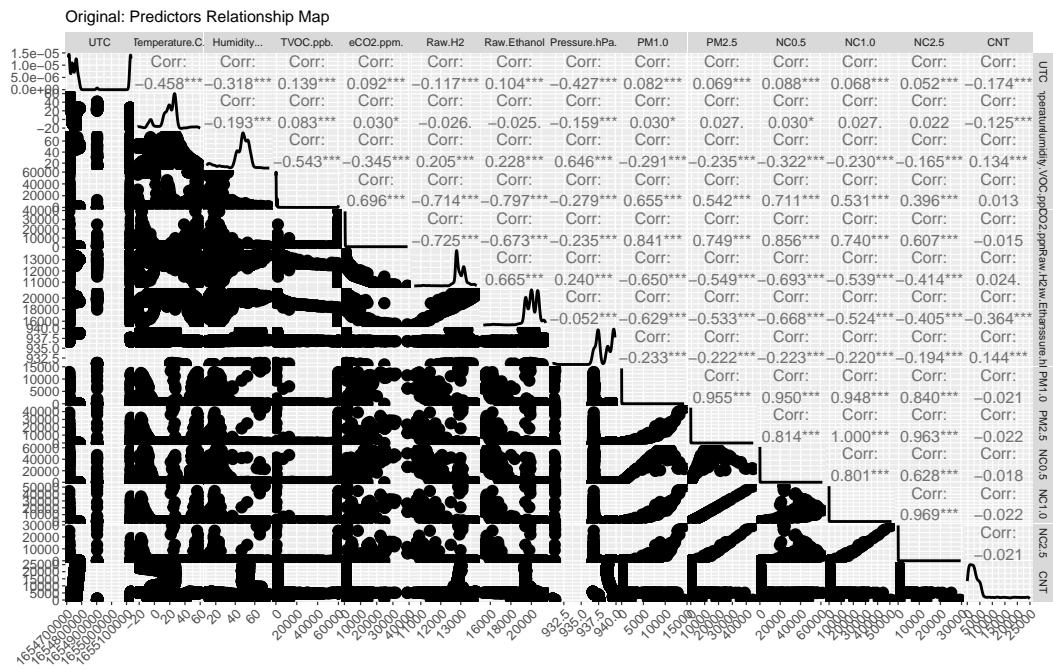
Histogram of Box–Cox Transformed gamma\_CNT



```
# Remove Original CNT in reduced dataframe
smokedf_reduced <- smokedf_reduced |> dplyr::select(-CNT) |>
  relocate("Fire.Alarm", .after = last_col())
```

## Exploratory Data Analysis

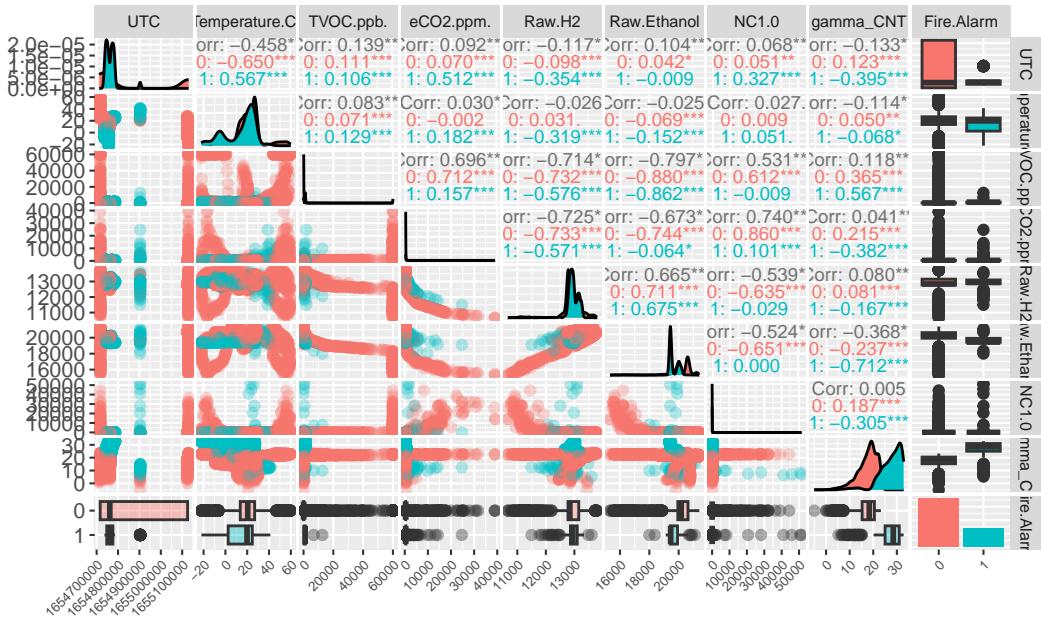
```
# General relationship plot: Multivariate Analysis (Original Data)
smokedf |>
  dplyr::select(-Fire.Alarm) |>
  ggpairs(title = 'Original: Predictors Relationship Map', progress = TRUE,
           upper = list(continuous = wrap("cor", size = 2))) +
  theme_grey(base_size = 5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 5),
        axis.text.y = element_text(size = 5),
        panel.spacing = unit(0.01, "lines"))
```



```
# General relationship plot: Multivariate Analysis (Reduced Data)
smokedf_reduced$Fire.Alarm <- as.factor(smokedf_reduced$Fire.Alarm)

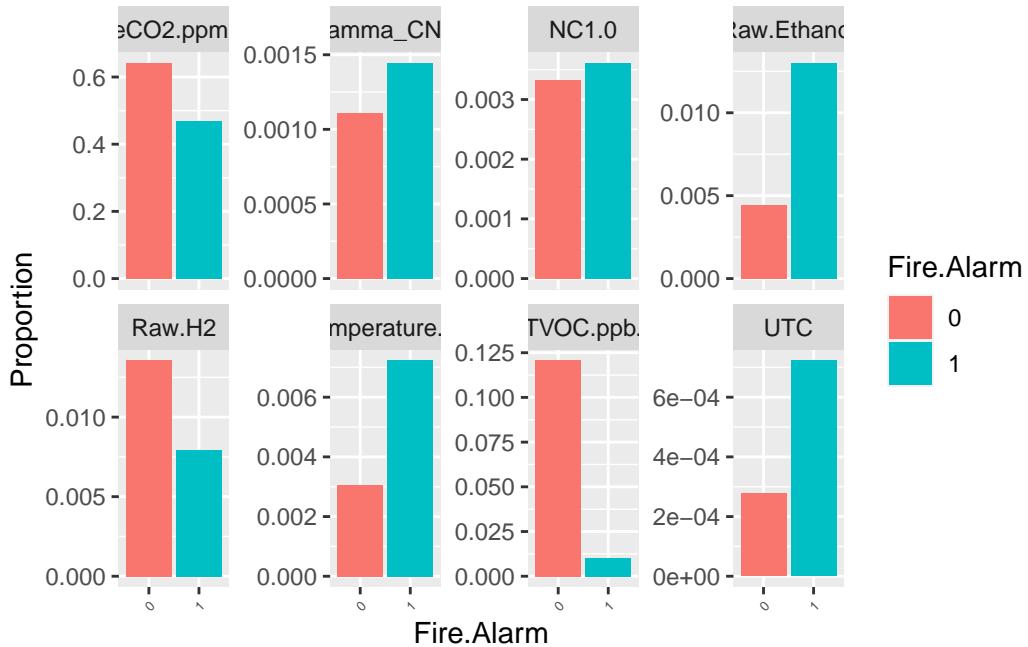
smokedf_reduced |>
  ggpairs(title = 'Reduced: Predictors Relationship Map', progress = TRUE,
          upper = list(continuous = wrap("cor", size = 2.5)),
          lower = list(continuous = wrap("points", alpha = 0.3),
                      combo = wrap("box_no_facet", alpha = 0.4)),
          columns = 1:9, ggplot2::aes(colour = Fire.Alarm)) +
  theme_grey(base_size = 8) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 5),
        axis.text.y = element_text(size = 8),
        panel.spacing = unit(0.1, "lines"))
```

### Reduced: Predictors Relationship Map

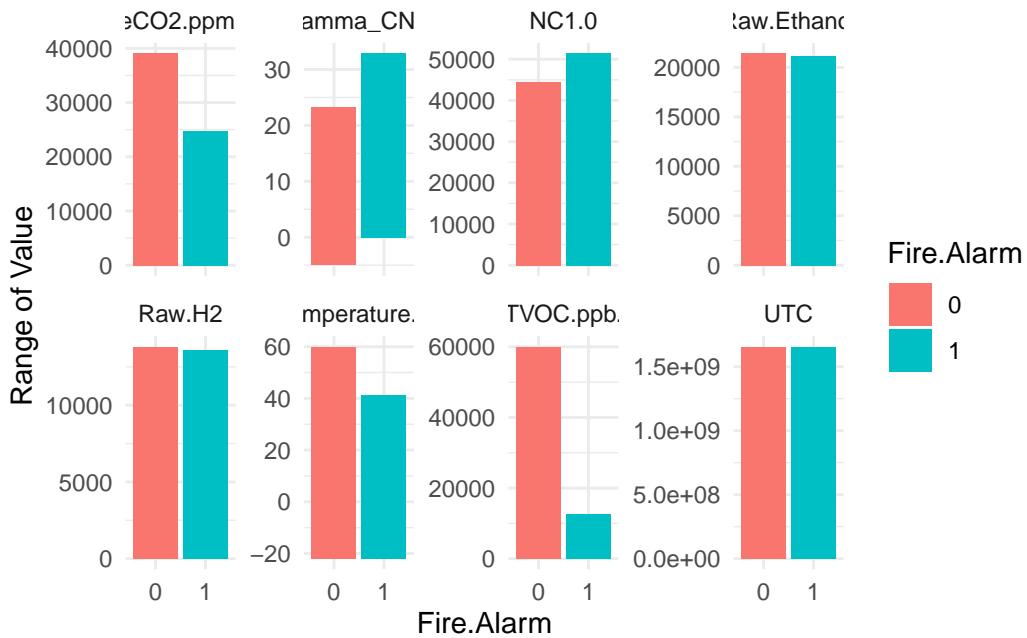


```
# Explore proportion of Fire.Alarm classes against each predictors
proportions <- smokedf_reduced |>
  pivot_longer(-Fire.Alarm, names_to = 'variable', values_to = 'value') |>
  group_by(Fire.Alarm, variable, value) |>
  summarize(Count = n(), .groups = 'drop') |>
  group_by(variable, Fire.Alarm) |>
  mutate(Proportion = Count / sum(Count))

# Create the bar plot
proportions |>
  ggplot(aes(x = Fire.Alarm, y = Proportion, fill = Fire.Alarm)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ variable, scales = "free_y", ncol = 4) +
  labs(x = "Fire.Alarm", y = "Proportion") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 5))
```



```
# Explore the the value range of Fire.Alarm classes against each predictors
smokedf_reduced |>
  pivot_longer(-Fire.Alarm, names_to = 'variable', values_to = 'value') |>
  ggplot(aes(x = Fire.Alarm, y = value, fill = Fire.Alarm)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ variable, scales = "free_y", ncol = 4) +
  labs(x = "Fire.Alarm", y = "Range of Value") +
  theme_minimal()
```



## Training and Test Sets

```

set.seed(503)
smokedf_reduced$Fire.Alarm <- ifelse(smokedf_reduced$Fire.Alarm == 0, "No", "Yes")
smokedf_predictors <- smokedf_reduced[, -1]
smokedf_yield <- smokedf_reduced[, 1]

# Split the data into training and test sets
smokedf_index <- createDataPartition(smokedf_yield, p = 0.8, list = FALSE)
smokedf_train <- smokedf_predictors[smokedf_index, ]
smokedf_test <- smokedf_predictors[-smokedf_index, ]

# Center and scaling using preProcess
smokedf_prep <- preProcess(smokedf_train, method = c("center", "scale"))

# Apply the transformation to the training data
smokedf_train_transformed <- predict(smokedf_prep, smokedf_train)
smokedf_test_transformed <- predict(smokedf_prep, smokedf_test)

# Rebalance train dataset
cat("Number of non-triggered fire alarm (No) = ",
    sum(smokedf_train_transformed$Fire.Alarm == "No"), "\n")

```

```
Number of non-triggered fire alarm (No) = 2914
```

```
cat("Number of triggered fire alarm (Yes) =",  
    sum(smokedf_train_transformed$Fire.Alarm == "Yes"))
```

```
Number of triggered fire alarm (Yes) = 1086
```

```
train_balanced <- ROSE(Fire.Alarm ~ ., data = smokedf_train_transformed)$data  
  
rebalanced <- as.data.frame(table(train_balanced$Fire.Alarm))  
rebalanced |> gt() |>  
  tab_header(title = "Rebalanced Fire Alarm Counts")
```

Rebalanced Fire Alarm Counts

Var1	Freq
No	2009
Yes	1991

## Model Strategies

```
set.seed(503)  
ctrl <- trainControl(method = "cv", summaryFunction = twoClassSummary,  
                      classProbs = TRUE, savePredictions = TRUE)  
  
# Create Logistic Regression  
lrFit <- train(Fire.Alarm ~ .,  
                data = train_balanced,  
                method = "glm",  
                metric = "ROC",  
                trControl = ctrl)  
# Logistic regression final model using train data set  
lrFit$finalModel
```

Call: NULL

Coefficients:

```

(Intercept) Temperature.C.      TVOC.ppb.      eCO2.ppm.      Raw.H2
-1.25969     -0.43238        -1.85039       0.02911       0.06451
Raw.Ethanol    NC1.0          gamma_CNT
-1.13717     0.34951         2.56565

Degrees of Freedom: 3999 Total (i.e. Null); 3992 Residual
Null Deviance: 5545
Residual Deviance: 2218 AIC: 2234

```

### lrFit

Generalized Linear Model

4000 samples  
 7 predictor  
 2 classes: 'No', 'Yes'

No pre-processing  
 Resampling: Cross-Validated (10 fold)  
 Summary of sample sizes: 3600, 3601, 3600, 3600, 3600, 3600, ...  
 Resampling results:

ROC	Sens	Spec
0.9519011	0.9128905	0.8834573

```
# Confusion matrix
lrCM <- confusionMatrix(lrFit, norm = "none")
lrCM
```

Cross-Validated (10 fold) Confusion Matrix

(entries are un-normalized aggregated counts)

	Reference	
Prediction	No	Yes
No	1834	232
Yes	175	1759

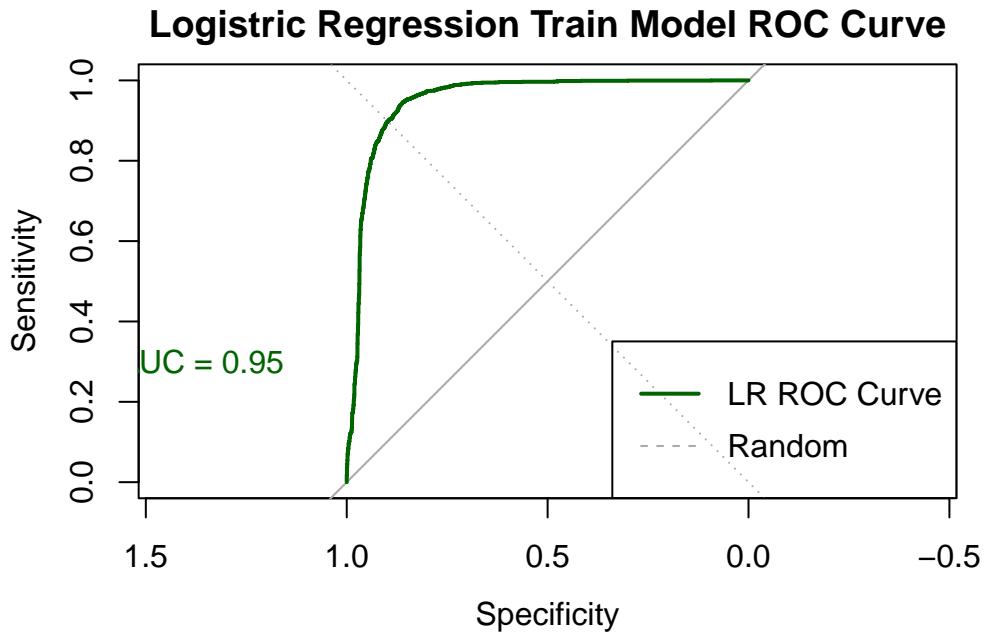
Accuracy (average) : 0.8982

```
# Logistic Regression ROC curve and AUC score
lrRoc <- roc(response = lrFit$pred$obs,
               predictor = lrFit$pred$Yes,
               levels = rev(levels(lrFit$pred$obs)))
```

Setting direction: controls > cases

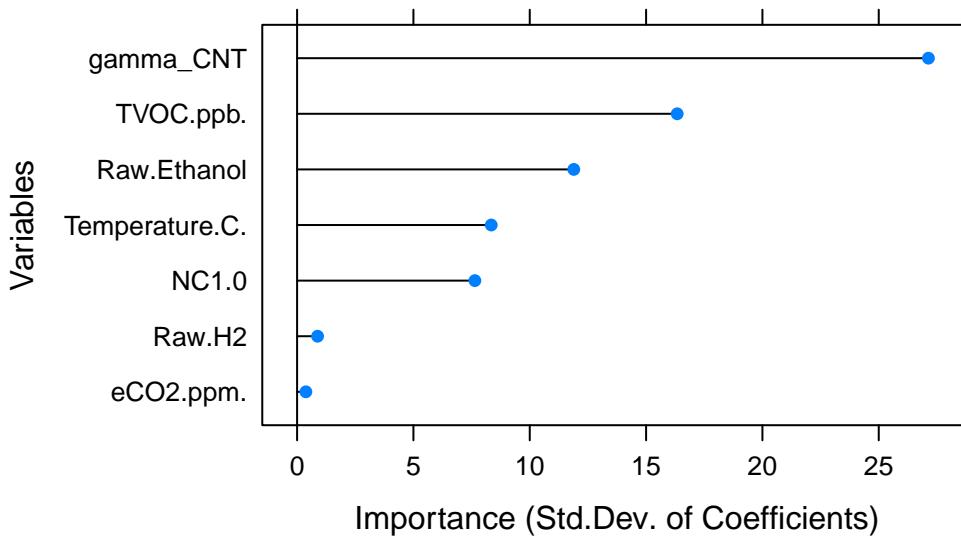
```
auc_score_lr <- auc(lrRoc)

plot(lrRoc, main = "Logistic Regression Train Model ROC Curve",
      col = "darkgreen", lwd = 2)
abline(a = 0, b = 1, lty = 3, col = "darkgray")
text(0.5, 0.3, paste0("AUC = ", round(auc_score_lr, 2)), adj = 2.6, col = "darkgreen")
legend("bottomright", legend = c("LR ROC Curve", "Random"),
       col = c("darkgreen", "darkgray"), lty = c(1, 2), lwd = c(2, 1))
```



```
# Variable importance
lrImp <- varImp(lrFit, scale = FALSE)
plot(lrImp, main = "Variable Importance (Logistic Regression)",
      xlab = "Importance (Std.Dev. of Coefficients)", ylab = "Variables")
```

## Variable Importance (Logistic Regression)



```
# Create Penalized Logistic Regression
glmGrid <- expand.grid(alpha = seq(0, 1, by = 0.1),
lambda = seq(.001, .3, length = 10))
glmFit <- train(Fire.Alarm ~ .,
                  data = train_balanced,
                  method = "glmnet",
                  tuneGrid = glmGrid,
                  preProc = c("center", "scale"),
                  metric = "ROC",
                  trControl = ctrl)
optimal_a <- glmFit$bestTune$alpha
optimal_l <- glmFit$bestTune$lambda
glmmodel <- train(Fire.Alarm ~ .,
                  data = train_balanced,
                  method = "glmnet",
                  preProc = c("center", "scale"),
                  metric = "ROC",
                  trControl = ctrl,
                  tuneGrid = expand.grid(alpha = optimal_a,
                                         lambda = optimal_l))

glmmodel
```

glmnet

```

4000 samples
  7 predictor
  2 classes: 'No', 'Yes'

Pre-processing: centered (7), scaled (7)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3600, 3601, 3600, 3600, 3600, 3600, ...
Resampling results:

ROC      Sens      Spec
0.9520178  0.913908  0.8829724

```

Tuning parameter 'alpha' was held constant at a value of 0.2  
Tuning  
parameter 'lambda' was held constant at a value of 0.001

```

# Confusion matrix
glnmCM <- confusionMatrix(glnmmodel, norm = "none")
glnmCM

```

Cross-Validated (10 fold) Confusion Matrix

(entries are un-normalized aggregated counts)

		Reference
Prediction	No	Yes
No	1836	233
Yes	173	1758

Accuracy (average) : 0.8985

```

# Penalized Logistic Regression ROC curve and AUC score
glnmRoc <- roc(response = glnmmodel$pred$obs,
                 predictor = glnmmodel$pred$Yes,
                 levels = rev(levels(glnmmodel$pred$obs)))

```

Setting direction: controls > cases

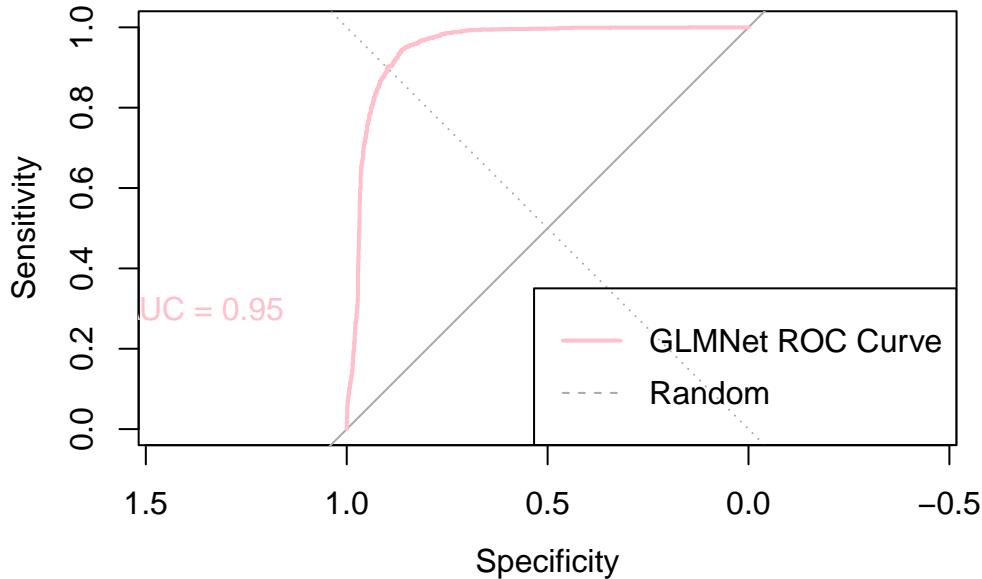
```

auc_score_glmn <- auc(glmnRoc )

plot(glmnRoc, main = "Penalized Logistic Regression Train Model ROC Curve",
     col = "pink", lwd = 2)
abline(a = 0, b = 1, lty = 3, col = "darkgray")
text(0.5, 0.3, paste0("AUC = ", round(auc_score_glmn, 2)), adj = 2.6, col = "pink")
legend("bottomright", legend = c("GLMNet ROC Curve", "Random"),
       col = c("pink", "darkgray"), lty = c(1, 2), lwd = c(2, 1))

```

## Penalized Logistic Regression Train Model ROC Curve

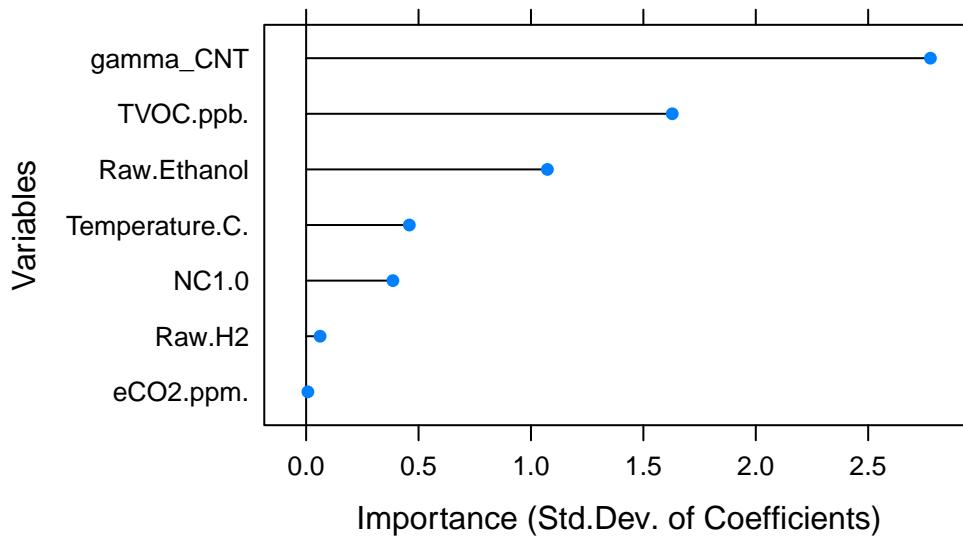


```

# Variable importance
glmnlmp <- varImp(glmnmodel, scale = FALSE)
plot(glmnlmp, main = "Variable Importance (Penalized Logistic Regression)",
      xlab = "Importance (Std.Dev. of Coefficients)", ylab = "Variables")

```

## Variable Importance (Penalized Logistic Regression)



```
# Create Nearest Shrunken Centroids
nscGrid <- expand.grid(threshold = seq(0, 25, length = 30))
nscFit <- train(Fire.Alarm ~ .,
                 data = train_balanced,
                 method = "pam",
                 tuneGrid = nscGrid,
                 metric = "ROC",
                 trControl = ctrl)
```

111111111111

```
nscFit
```

```
Nearest Shrunken Centroids

4000 samples
 7 predictor
 2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3600, 3600, 3599, 3600, 3601, 3600, ...
Resampling results across tuning parameters:
```

threshold	ROC	Sens	Spec
0.000000	0.9440955	0.9263507	0.8724146
0.862069	0.9411334	0.9223706	0.8663869
1.724138	0.9371145	0.9129104	0.8583518
2.586207	0.9313686	0.9029552	0.8498166
3.448276	0.9283388	0.8934950	0.8463015
4.310345	0.9251443	0.8875274	0.8422864
5.172414	0.9230491	0.8850348	0.8407789
6.034483	0.9241579	0.8855299	0.8407789
6.896552	0.9259816	0.8850323	0.8412814
7.758621	0.9277952	0.8855299	0.8432889
8.620690	0.9293390	0.8875174	0.8447965
9.482759	0.9302454	0.8959801	0.8452965
10.344828	0.9302976	0.8959801	0.8447940
11.206897	0.9302976	0.8964776	0.8447940
12.068966	0.9302976	0.8979701	0.8442915
12.931034	0.9302976	0.9014527	0.8437915
13.793103	0.9302976	0.9019502	0.8432889
14.655172	0.9302976	0.9034428	0.8432889
15.517241	0.9302976	0.9049353	0.8432889
16.379310	0.9302976	0.9069279	0.8432889
17.241379	0.9302976	0.9089204	0.8432889
18.103448	0.9302976	0.9094179	0.8397714
18.965517	0.9302976	0.9104129	0.8377613
19.827586	0.9302976	0.9109104	0.8367563
20.689655	0.9302976	0.9129005	0.8337412
21.551724	0.9302976	0.9183781	0.8322337
22.413793	0.9302976	0.9253458	0.8287236
23.275862	0.9302976	0.9313209	0.8232035
24.137931	0.9302976	0.9422662	0.8116558
25.000000	0.9302976	0.9581965	0.7930678

ROC was used to select the optimal model using the largest value.  
The final value used for the model was threshold = 0.

```
# Confusion matrix
nscCM <- confusionMatrix(nscFit, norm = "none")
nscCM
```

Cross-Validated (10 fold) Confusion Matrix

(entries are un-normalized aggregated counts)

Reference		
Prediction	No	Yes
No	1861	254
Yes	148	1737

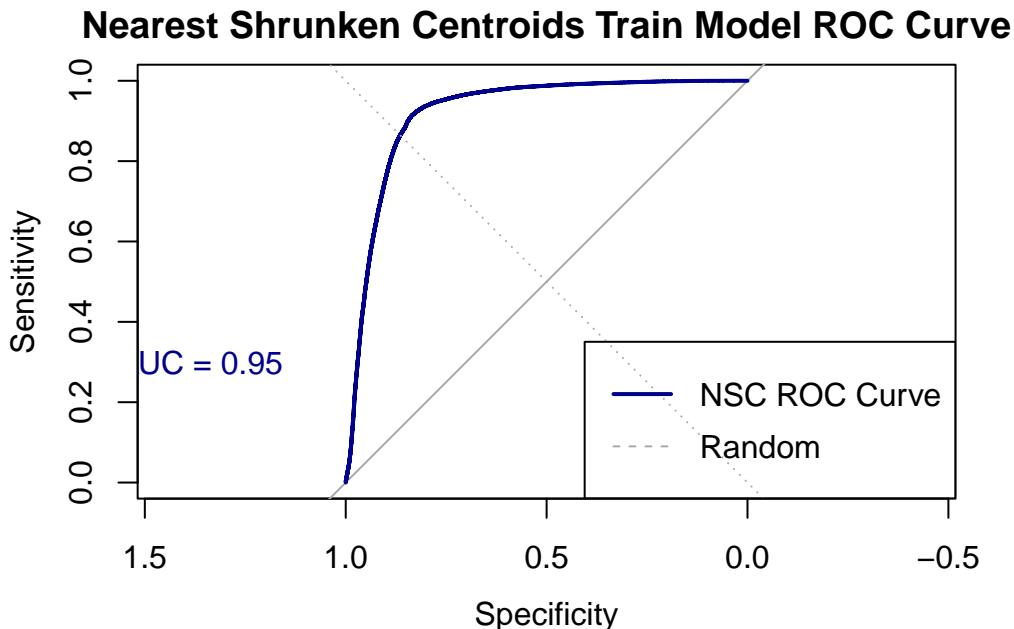
Accuracy (average) : 0.8995

```
# Nearest Shrunken Centroids ROC curve and AUC score
nscRoc <- roc(response = nscFit$pred$obs,
                 predictor = nscFit$pred$Yes,
                 levels = rev(levels(nscFit$pred$obs)))
```

Setting direction: controls > cases

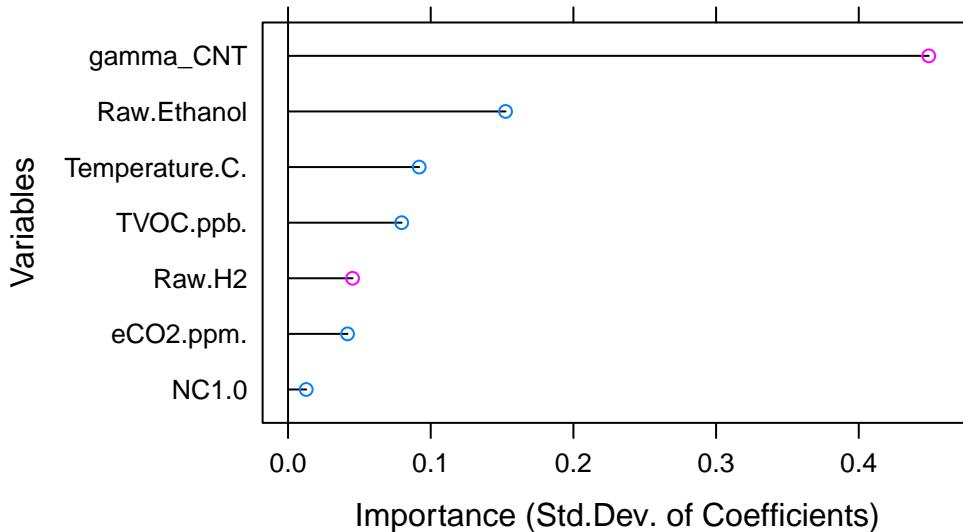
```
auc_score_nsc <- auc(nscRoc)

plot(nscRoc, main = "Nearest Shrunken Centroids Train Model ROC Curve",
      col = "darkblue", lwd = 2)
abline(a = 0, b = 1, lty = 3, col = "darkgray")
text(0.5, 0.3, paste0("AUC = ", round(auc_score_glmn, 2)), adj = 2.6, col = "darkblue")
legend("bottomright", legend = c("NSC ROC Curve", "Random"),
       col = c("darkblue", "darkgray"), lty = c(1, 2), lwd = c(2, 1))
```



```
# Variable importance
nscImp <- varImp(nscFit, scale = FALSE)
plot(nscImp, main = "Variable Importance (Nearest Shrunken Centroids Regression)",
     xlab = "Importance (Std.Dev. of Coefficients)", ylab = "Variables")
```

## Variable Importance (Nearest Shrunken Centroids Regression)



## Validation and Testing

```
# Predict test data using logistic regression
test_predictors <- smokedf_test_transformed[, -8]
test_org <- data.frame(Fire.Alarm = smokedf_test_transformed$Fire.Alarm)
test_org$Fire.Alarm <- factor(test_org$Fire.Alarm)

lr_predictions <- predict(lrFit, newdata = test_predictors, type = "raw")
lr_predictions <- data.frame(Fire.Alarm = lr_predictions)

# Confusion Matrix
lrCM_test <- confusionMatrix(data = as.factor(lr_predictions$Fire.Alarm),
                               reference = test_org$Fire.Alarm, positive = "Yes")
lrCM_test
```

Confusion Matrix and Statistics

```

    Reference
Prediction No Yes
    No 679 25
    Yes 23 273

    Accuracy : 0.952
    95% CI : (0.9369, 0.9644)
No Information Rate : 0.702
P-Value [Acc > NIR] : <2e-16

    Kappa : 0.8851

McNemar's Test P-Value : 0.8852

    Sensitivity : 0.9161
    Specificity : 0.9672
Pos Pred Value : 0.9223
Neg Pred Value : 0.9645
    Prevalence : 0.2980
    Detection Rate : 0.2730
Detection Prevalence : 0.2960
Balanced Accuracy : 0.9417

'Positive' Class : Yes

```

```

# Predict test data using penalized logistic regression
glmn_predictions <- predict(glmnmodel, newdata = test_predictors, type = "raw")
glmn_predictions <- data.frame(Fire.Alarm = glmn_predictions)

# Confusion Matrix
glmnCM_test <- confusionMatrix(data = as.factor(glmn_predictions$Fire.Alarm),
                                 reference = test_org$Fire.Alarm, positive = "Yes")
glmnCM_test

```

#### Confusion Matrix and Statistics

```

    Reference
Prediction No Yes
    No 679 25
    Yes 23 273

```

```

Accuracy : 0.952
95% CI : (0.9369, 0.9644)
No Information Rate : 0.702
P-Value [Acc > NIR] : <2e-16

Kappa : 0.8851

McNemar's Test P-Value : 0.8852

Sensitivity : 0.9161
Specificity : 0.9672
Pos Pred Value : 0.9223
Neg Pred Value : 0.9645
Prevalence : 0.2980
Detection Rate : 0.2730
Detection Prevalence : 0.2960
Balanced Accuracy : 0.9417

'Positive' Class : Yes

# Predict test data using nearest shrunken centroid
nsc_predictions <- predict(nscFit, newdata = test_predictors, type = "raw")
nsc_predictions <- data.frame(Fire.Alarm = nsc_predictions)

# Confusion Matrix
nscCM_test <- confusionMatrix(data = as.factor(nsc_predictions$Fire.Alarm),
                                 reference = test_org$Fire.Alarm, positive = "Yes")
nscCM_test

Confusion Matrix and Statistics

Reference
Prediction  No Yes
      No   677  26
      Yes   25 272

Accuracy : 0.949
95% CI : (0.9335, 0.9618)
No Information Rate : 0.702
P-Value [Acc > NIR] : <2e-16

```

```

Kappa : 0.878

McNemar's Test P-Value : 1

Sensitivity : 0.9128
Specificity : 0.9644
Pos Pred Value : 0.9158
Neg Pred Value : 0.9630
Prevalence : 0.2980
Detection Rate : 0.2720
Detection Prevalence : 0.2970
Balanced Accuracy : 0.9386

'Positive' Class : Yes

```

## Performance Evaluation

```
roc_test_lr <- roc(response = as.factor(test_org$Fire.Alarm),
                     predictor = as.numeric(lr_predictions$Fire.Alarm))
```

Setting levels: control = No, case = Yes

Setting direction: controls < cases

```
roc_test_glmn <- roc(response = as.factor(test_org$Fire.Alarm),
                       predictor = as.numeric(glmn_predictions$Fire.Alarm))
```

Setting levels: control = No, case = Yes

Setting direction: controls < cases

```
roc_test_nsc <- roc(response = as.factor(test_org$Fire.Alarm),
                      predictor = as.numeric(nsc_predictions$Fire.Alarm))
```

Setting levels: control = No, case = Yes

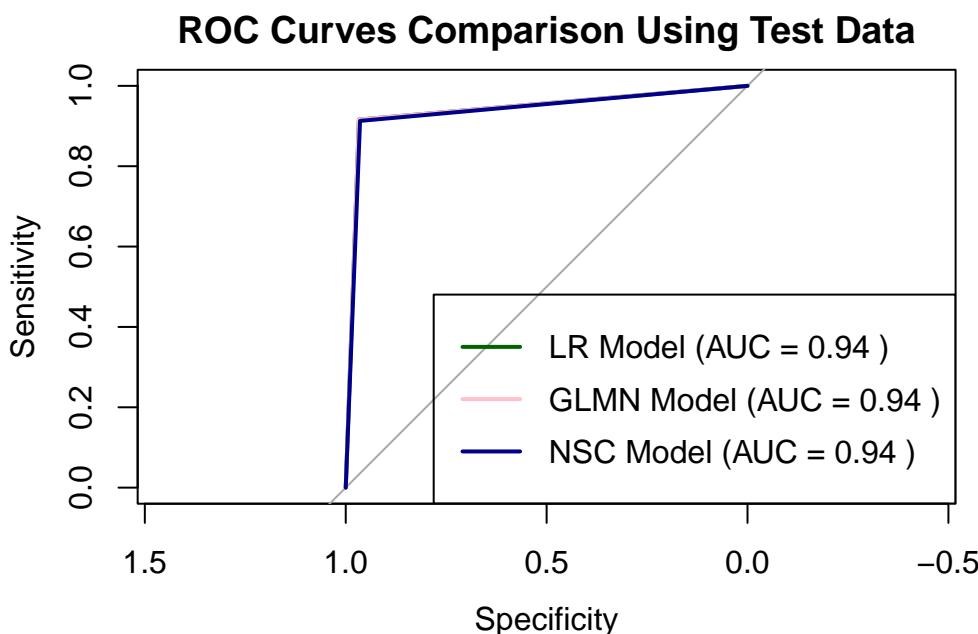
Setting direction: controls < cases

```

auc_test_lr <- auc(roc_test_lr)
auc_test_glmn <- auc(roc_test_glmn)
auc_test_nsc <- auc(roc_test_nsc)

# Plot ROC curve
plot(roc_test_lr, col = "darkgreen", main = "ROC Curves Comparison Using Test Data", lwd = 2)
plot(roc_test_glmn, col = "pink", add = TRUE, lwd = 2)
plot(roc_test_nsc, col = "darkblue", add = TRUE, lwd = 2)
legend("bottomright", legend = c(paste("LR Model (AUC =", round(auc_test_lr, 2), ")"),
                                 paste("GLMN Model (AUC =", round(auc_test_glmn, 2), ")"),
                                 paste("NSC Model (AUC =", round(auc_test_nsc, 2), ")")),
       col = c("darkgreen", "pink", "darkblue"), lwd = 2)

```



```

# Model's ROC score
lrFit_ROC <- round(lrFit$results$ROC,3)
glmnmodel_ROC <- round(glmnmodel$results$ROC,3)
nscFit_ROC <- round(nscFit$results$ROC,3)

# Confusion matrix train accuracy score
lrCM_train_accuracy <- 0.898
glmnCM_train_accuracy <- 0.899
nscCM_train_accuracy <- 0.899

```

```

# Confusion matrix test AUC score
lrCM_AUC <- round(auc(roc_test_lr),3)
glmnmCM_AUC <- round(auc(roc_test_glmn),3)
nscCM_AUC <- round(auc(roc_test_nsc),3)

# Confusion matrix test accuracy score
lrCM_test_accuracy <- lrCM_test$overall["Accuracy"]
glmnmCM_test_accuracy <- glmnmCM_test$overall["Accuracy"]
nscCM_test_accuracy <- nscCM_test$overall["Accuracy"]

# Confusion matrix test class balanced accuracy score
lrCM_test_bal_accuracy <- round(lrCM_test$byClass["Balanced Accuracy"],3)
glmnmCM_test_bal_accuracy <- round(glmnmCM_test$byClass["Balanced Accuracy"],3)
nscCM_test_bal_accuracy <- round(nscCM_test$byClass["Balanced Accuracy"],3)

# Adjust nscFit_ROC
nscFit_ROC_adjusted <- nscFit_ROC[1]

# Now, create the data frame with adjusted nscFit_ROC
results_table <- data.frame(
  Model = c("LR", "Penalized LR",
            "NSC"),
  ROC_Score = c(lrFit_ROC, glmnmmodel_ROC, nscFit_ROC_adjusted),
  Train_Accuracy = c(lrCM_train_accuracy, glmnmCM_train_accuracy,
                      nscCM_train_accuracy),
  Test_AUC = c(lrCM_AUC, glmnmCM_AUC, nscCM_AUC),
  Test_Accuracy = c(lrCM_test_accuracy, glmnmCM_test_accuracy,
                     nscCM_test_accuracy),
  Test_Balanced_Accuracy = c(lrCM_test_bal_accuracy,
                              glmnmCM_test_bal_accuracy,
                              nscCM_test_bal_accuracy))

# Create a gt table
results_table |> gt() |>
  tab_header(title = "Model Comparison",
             subtitle = "Comparison of performance metrics for different models") |>
  cols_label(Model = "Model",
             ROC_Score = "Model ROC",
             Train_Accuracy = "Train Accuracy",
             Test_AUC = "Test AUC",
             Test_Accuracy = "Test Accuracy",
             Test_Balanced_Accuracy = "Test Balanced Accuracy")

```

Model Comparison  
Comparison of performance metrics for different models

Model	Model ROC	Train Accuracy	Test AUC	Test Accuracy	Test Balanced Accuracy
LR	0.952	0.898	0.942	0.952	0.942
Penalized LR	0.952	0.899	0.942	0.952	0.942
NSC	0.944	0.899	0.939	0.949	0.939