Research Project 2

Pu Jian

2023-04-09

Ambient Air Pollution Concentrations Analysis

Introduction

This project is named “Ambient Air Pollution Concentrations Analysis”, which aims to predict the impact of different variables on air pollution levels. The original data set contains a large number of variables, but some variables are too similar and duplicated, such as county_pop, county_area, and popdens_county, which can lead to bias. Additionally, not all variable values are suitable for prediction, therefore, we only selected the following variables: lat, lon, CMAQ, popdens_county, popdens_zcta, pov, hs_orless, aod, and two self-created binary variables: pollution_level and urban_level_2013. Lat and lon estimate the effect of geographic position on PM2.5, popdens_county and popdens_zcta estimate the effect of population density on PM2.5, pov estimates the effect of poverty on PM2.5, hs_orless estimates the effect of education level on PM2.5, CMAQ and aod indicate the accuracy and authority of the measuring instruments. These variables can represent the most important variables in the original data set, which are necessary to make the prediction project. In addition, to have a better understanding of the data set, I will clean the data by dropping null and duplicate values and present the PM2.5 values of each city through plotting. Then, I will calculate the maximum, minimum, mean, and median values to have a clearer understanding of the severity of PM2.5 values. After that, I will re-establish the data set and create a binary variable for participation in predicting. This binary variable will be the pollution_level, determined by the size of the PM2.5 value, as 1 or 0. The concept of urbanization will also be presented by the binary variable as 1 and 0. This is because continuous numbers cannot provide us with a good concept of pollution levels and the degree of urbanization. I will establish three predictive models: Categorize Air Pollution Concentrations, Random Forest, and Generalized Additive Models. CART is a machine learning algorithm that uses decision trees for both classification and regression analysis. Random forest is a machine learning algorithm that belongs to the family of ensemble methods; it combines multiple decision trees, each trained on a different subset of the data, to improve the overall predictive accuracy and reduce overfitting. Generalized Additive Model is a type of statistical model used for regression analysis. They extend the linear regression model by allowing for non-linear relationships between the predictor variables and the response variable. Based on the definition of Random Forest model, we can know that Random Forest model should have the most accurate predictive results due to its combination of multiple decision trees. In this case, I expect that Random Forest model will have the lowest RMSE (best prediction model), and the predictive results of CART and GAMs may be comparable.

Wrangling

In the wrangling part, I will cover two aspects: one is data cleaning and establishing preliminary understanding, and the other is preparing data required for building the model. The first aspect includes viewing the data structure, checking null and duplicate values, counting the number of observations, visualizing the data, and checking the significant parameters and cities. The second aspect includes categorizing air pollution concentration levels, creating the binary variable, and selecting the variables for prediction. ##### Print Out and View the Data Structure

```
## Load the packages and the data set we need
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────────────────── tidy
verse 2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────────────────────────────
──── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
me errors
```

```
library(caret)
```

```
## 载入需要的程辑包：lattice
##
## 载入程辑包：'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## 载入程辑包：'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## 载入程辑包：'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(ggthemes)
library(ggfortify)
library(mgcv)
```

```
## 载入需要的程辑包：nlme
##
## 载入程辑包：'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
library(gam)
```

```
## 载入需要的程辑包：splines
## 载入需要的程辑包：foreach
##
## 载入程辑包：'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Loaded gam 1.22-2
##
##
## 载入程辑包：'gam'
##
## The following objects are masked from 'package:mgcv':
##
##     gam, gam.control, gam.fit, s
```

```
library(tibble)
library(sf)
```

```
## Linking to GEOS 3.9.3, GDAL 3.5.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
data<-read_csv("https://github.com/rdpeng/stat322E_public/raw/main/data/pm25_data.csv.gz")
```

```
## Rows: 876 Columns: 50
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): state, county, city
## dbl (47): id, value, fips, lat, lon, CMAQ, zcta, zcta_area, zcta_pop, imp_a5...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Use "head" function to take an observation on the data set
head(data)
```

```
## # A tibble: 6 × 50
##      id value  fips   lat   lon state   county  city        CMAQ  zcta zcta_area
##   <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   <chr>   <chr>      <dbl> <dbl>     <dbl>
## 1 1003.  9.60  1003  30.5 -87.9 Alabama Baldwin Fairhope    8.10 36532 190980522
## 2 1027. 10.8   1027  33.3 -85.8 Alabama Clay    Ashland     9.77 36251 374132430
## 3 1033. 11.2   1033  34.8 -87.7 Alabama Colbert Muscle Sh…  9.40 35660  16716984
## 4 1049. 11.7   1049  34.3 -86.0 Alabama DeKalb  Crossville  8.53 35962 203836235
## 5 1055. 12.4   1055  34.0 -86.0 Alabama Etowah  Gadsden     9.24 35901 154069359
## 6 1069. 10.5   1069  31.2 -85.4 Alabama Houston Dothan      9.12 36303 162685124
## # ℹ 39 more variables: zcta_pop <dbl>, imp_a500 <dbl>, imp_a1000 <dbl>,
## #   imp_a5000 <dbl>, imp_a10000 <dbl>, imp_a15000 <dbl>, county_area <dbl>,
## #   county_pop <dbl>, log_dist_to_prisec <dbl>, log_pri_length_5000 <dbl>,
## #   log_pri_length_10000 <dbl>, log_pri_length_15000 <dbl>,
## #   log_pri_length_25000 <dbl>, log_prisec_length_500 <dbl>,
## #   log_prisec_length_1000 <dbl>, log_prisec_length_5000 <dbl>,
## #   log_prisec_length_10000 <dbl>, log_prisec_length_15000 <dbl>, …
```

Check If There Are Null Values or Duplicate Values

```
## Use "sum" and "is.na" functions to count the total number of null values in the data set.
sum(is.na(data))
```

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

```
## Use "sum" and "duplicated" functions to count the total number of duplicate values in the da
ta set
sum(duplicated(data))
```

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

Since the values are all 0, we know that there are no null values or duplicate values and we don't need to drop any lines.

Count the Total Number of Observations for Each Variable

```
## use "nrow" function to count the total number of observations for each variable.
nrow(data)
```

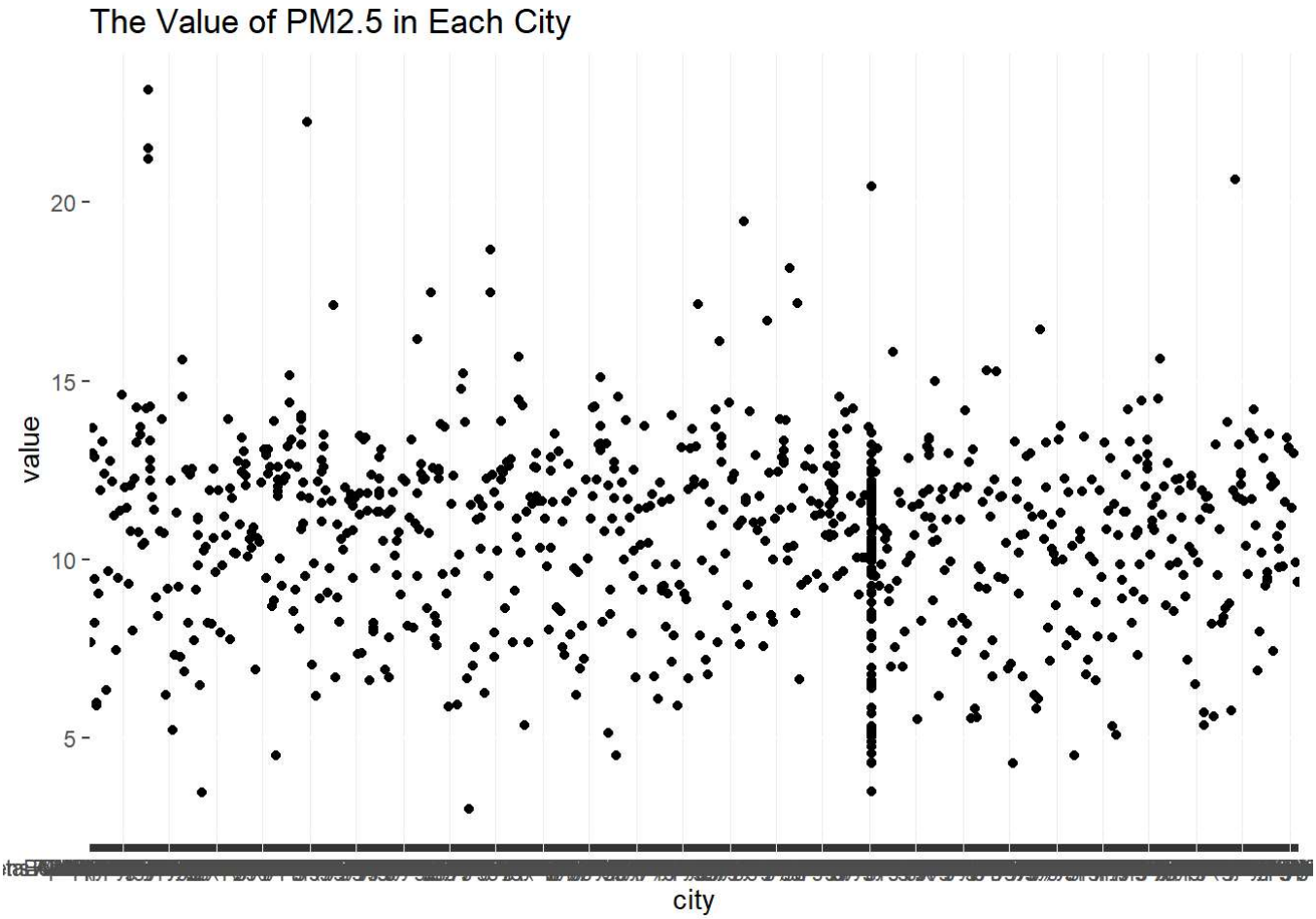
```
## [1] 876
```

Therefore, there are 876 observations for each variable.

Visualize the Value of PM2.5 for Each City

To get a rough idea of the PM2.5 values for each city, I will use the ggplot function to plot each city and its corresponding PM2.5 value.

```
## Use "ggplot" function to create the graph
ggplot(data,aes(x=city,y=value))+geom_point()+ggtitle("The Value of PM2.5 in Each City")
```



By observing this plot, although the names of the cities cannot be clearly displayed due to their dense distribution, we can still see that most of the PM2.5 values are concentrated between 5-15. Some cities have higher PM2.5 values, ranging from 15-20, and a very small proportion of cities have extreme PM2.5 values, greater than 20 or less than 5.

Check the Maximum Value, the Minimum Value, the Mean Value, And the Median Value of the PM2.5, And Then Report the Cities With the Worst Air Pollution And the Least Air Pollution.

```
## Use "max", "min", "mean", and "median" functions to get the values of the maximum, minimum,
mean, and median air pollution
max<-max(data$value)
min<-min(data$value)
mean(data$value)
```

```
## [1] 10.80781
```

```
median(data$value)
```

```
## [1] 11.15267
```

```
print(max)
```

```
## [1] 23.16078
```

```
print(min)
```

```
## [1] 3.02381
```

```
## Use "filter" and "select" functions to get the eligible cities
worst<-data%>%filter(value==max)%>%select(city)
least<-data%>%filter(value==min)%>%select(city)
print(worst)
```

```
## # A tibble: 1 × 1
##   city
##   <chr>
## 1 Bakersfield
```

```
print(least)
```

```
## # A tibble: 1 × 1
##   city
##   <chr>
## 1 Fort Defiance
```

We can see that the maximum value of air pollution is 23.16078, and is in Bakersfield; the minimum value of air pollution is 3.02381, and is in Fort Defiance; the mean value is 10.80781, and the median value is 11.15267.

Categorize Air Pollution Concentrations

By looking at the maximum and minimum values of air pollution levels, we can see that there is a difference of 20.13697. However, the degree of air pollution cannot be clearly realized through data only, therefore, we classify the variable “value”. Since the median is 11.15267 and the mean is 10.80781, we can classify the values larger than 11 as highly polluted, marked as 1, and classify the values smaller than 11 as lightly polluted, marked as 0.

```
## Use "mutate" function to create the binary variable "pollution_level"
value_data<-data%>%mutate(pollution_level=ifelse(value>11,1,0))
value_data
```

```
## # A tibble: 876 × 51
##       id value  fips   lat   lon state   county    city     CMAQ  zcta zcta_area
##    <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   <chr>     <chr>   <dbl> <dbl>     <dbl>
##  1 1003.  9.60  1003  30.5 -87.9 Alabama Baldwin   Fairho…  8.10 36532 190980522
##  2 1027. 10.8   1027  33.3 -85.8 Alabama Clay      Ashland  9.77 36251 374132430
##  3 1033. 11.2   1033  34.8 -87.7 Alabama Colbert   Muscle…  9.40 35660  16716984
##  4 1049. 11.7   1049  34.3 -86.0 Alabama DeKalb    Crossv…  8.53 35962 203836235
##  5 1055. 12.4   1055  34.0 -86.0 Alabama Etowah    Gadsden  9.24 35901 154069359
##  6 1069. 10.5   1069  31.2 -85.4 Alabama Houston   Dothan   9.12 36303 162685124
##  7 1073. 15.6   1073  33.6 -86.8 Alabama Jefferson Birmin… 10.2  35207  26929603
##  8 1073. 12.4   1073  33.3 -87.0 Alabama Jefferson Not in… 10.2  35111 166239542
##  9 1073. 11.1   1073  33.5 -87.3 Alabama Jefferson Not in…  8.16 35444 385566685
## 10 1073. 13.1   1073  33.5 -86.5 Alabama Jefferson Leeds    9.30 35094 148994881
## # ℹ 866 more rows
## # ℹ 40 more variables: zcta_pop <dbl>, imp_a500 <dbl>, imp_a1000 <dbl>,
## #   imp_a5000 <dbl>, imp_a10000 <dbl>, imp_a15000 <dbl>, county_area <dbl>,
## #   county_pop <dbl>, log_dist_to_prisec <dbl>, log_pri_length_5000 <dbl>,
## #   log_pri_length_10000 <dbl>, log_pri_length_15000 <dbl>,
## #   log_pri_length_25000 <dbl>, log_prisec_length_500 <dbl>,
## #   log_prisec_length_1000 <dbl>, log_prisec_length_5000 <dbl>, …
```

Create a Binary Variable About the Degree of Urbanization

In order to know the degree of urbanization, we will classify all values greater than 3 in variable “urc2013” as 1, indicating urbanization, and classify all values smaller and equal to 3 in variable “urc2013” as 0, indicating non-urbanization. We choose urc2013 instead of urc2006 because 2013 is closer to the current year, which is more valuable for reference.

```
## Use "mutate" function to create a new binary variable
binary_data<-value_data%>%mutate(urban_level_2013=ifelse(urc2013>3,1,0))
binary_data
```

```
## # A tibble: 876 × 52
##       id value  fips   lat   lon state   county    city     CMAQ  zcta zcta_area
##    <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   <chr>     <chr>   <dbl> <dbl>     <dbl>
##  1 1003.  9.60  1003  30.5 -87.9 Alabama Baldwin   Fairho…  8.10 36532 190980522
##  2 1027. 10.8   1027  33.3 -85.8 Alabama Clay      Ashland  9.77 36251 374132430
##  3 1033. 11.2   1033  34.8 -87.7 Alabama Colbert   Muscle…  9.40 35660  16716984
##  4 1049. 11.7   1049  34.3 -86.0 Alabama DeKalb    Crossv…  8.53 35962 203836235
##  5 1055. 12.4   1055  34.0 -86.0 Alabama Etowah    Gadsden  9.24 35901 154069359
##  6 1069. 10.5   1069  31.2 -85.4 Alabama Houston   Dothan   9.12 36303 162685124
##  7 1073. 15.6   1073  33.6 -86.8 Alabama Jefferson Birmin… 10.2  35207  26929603
##  8 1073. 12.4   1073  33.3 -87.0 Alabama Jefferson Not in… 10.2  35111 166239542
##  9 1073. 11.1   1073  33.5 -87.3 Alabama Jefferson Not in…  8.16 35444 385566685
## 10 1073. 13.1   1073  33.5 -86.5 Alabama Jefferson Leeds    9.30 35094 148994881
## # ℹ 866 more rows
## # ℹ 41 more variables: zcta_pop <dbl>, imp_a500 <dbl>, imp_a1000 <dbl>,
## #   imp_a5000 <dbl>, imp_a10000 <dbl>, imp_a15000 <dbl>, county_area <dbl>,
## #   county_pop <dbl>, log_dist_to_prisec <dbl>, log_pri_length_5000 <dbl>,
## #   log_pri_length_10000 <dbl>, log_pri_length_15000 <dbl>,
## #   log_pri_length_25000 <dbl>, log_prisec_length_500 <dbl>,
## #   log_prisec_length_1000 <dbl>, log_prisec_length_5000 <dbl>, …
```

Choose the Variables We Need for Predicting, and Save Them As a New Data Set Named “new_data”.

```
## Use "select" function to choose the variables we need
new_data<-binary_data%>%select(pollution_level,lat,lon,CMAQ,popdens_county,popdens_zcta,pov,hs_
orless,urban_level_2013,aod)
## View the new data set by "head" function
head(new_data)
```

```
## # A tibble: 6 × 10
##   pollution_level   lat   lon  CMAQ popdens_county popdens_zcta   pov hs_orless
##             <dbl> <dbl> <dbl> <dbl>          <dbl>        <dbl> <dbl>     <dbl>
## 1               0  30.5 -87.9  8.10          44.3         146.    6.1      33.3
## 2               0  33.3 -85.8  9.77           8.91         13.6  19.5      64.6
## 3               1  34.8 -87.7  9.40          35.5         541.   19        53.7
## 4               1  34.3 -86.0  8.53          35.3          40.7  13.8      66
## 5               1  34.0 -86.0  9.24          75.4         130.    8.8      45.4
## 6               0  31.2 -85.4  9.12          67.6         186.   15.6      47.2
## # ℹ 2 more variables: urban_level_2013 <dbl>, aod <dbl>
```

Results

The result part includes three prediction models (Classification and Regression Tree, Random Forest, and Generalized Additive Models) and their performance. Also, in order to get more accurate predictions, I will perform k-fold cross-validations on these three models to check whether there will be any improvements on the prediction performance. The train and test sets will be split by the ratio of 7:3, the visualization of Generalized Additive Models will be presented, and two tables and visualization graphs that summarize prediction performances' results will be displayed. Finally, we will compare the values of sensitivity, specificity, AUC (these three parameters are used to evaluate the performance of binary classification models), accuracy, R-squared, and MAE to get the best performed model; the main comparison parameter will be RMSE. Since the data is packed randomly, we will set a seed.

Divide the Data Set into Trainning and Testing Sets

```
## Set a seed
set.seed(48)

## Use "createDataPartition" function to create train and test sets
idx_train<-createDataPartition(new_data$pollution_level,p=0.7)[[1]]
data_train<-new_data[idx_train,]
data_test<-new_data[-idx_train,]
```

Splitting the dataset into training and testing sets helps us avoid over fitting, and by using the testing set to evaluate the performance and predictive ability of the model, we can gain a better understanding of its performance and make improvements. A very common splitting ratio is 7:3 (train set is 7 and test set is 3), and I also used this ratio to split my data set into training and testing sets.

Classification and Regression Tree

```
## Set a seed
set.seed(48)

## Use "rpart" function to fit a CART model
train_cart<-rpart(pollution_level~.,data=data_train)
## Use "predict" function to get predictions for the observations
test_cart_prediction<-predict(train_cart,newdata=data_test)
test_pre_class_CART<-ifelse(test_cart_prediction>0.5,"1","0")

## Calculate the accuracy score
mean(test_pre_class_CART==data_test$pollution_level)
```

```
## [1] 0.7709924
```

```
## Use "table" function to build a confusion matrix
conf_M_CART<-table(test_pre_class_CART,data_test$pollution_level)
conf_M_CART
```

```
##
## test_pre_class_CART  0   1
##                   0  91  26
##                   1  34 111
```

```
## Use "sensitivity" and "specificity" functions to get the sensitivity and specificity values
sensitivity(conf_M_CART)
```

```
## [1] 0.728
```

```
specificity(conf_M_CART)
```

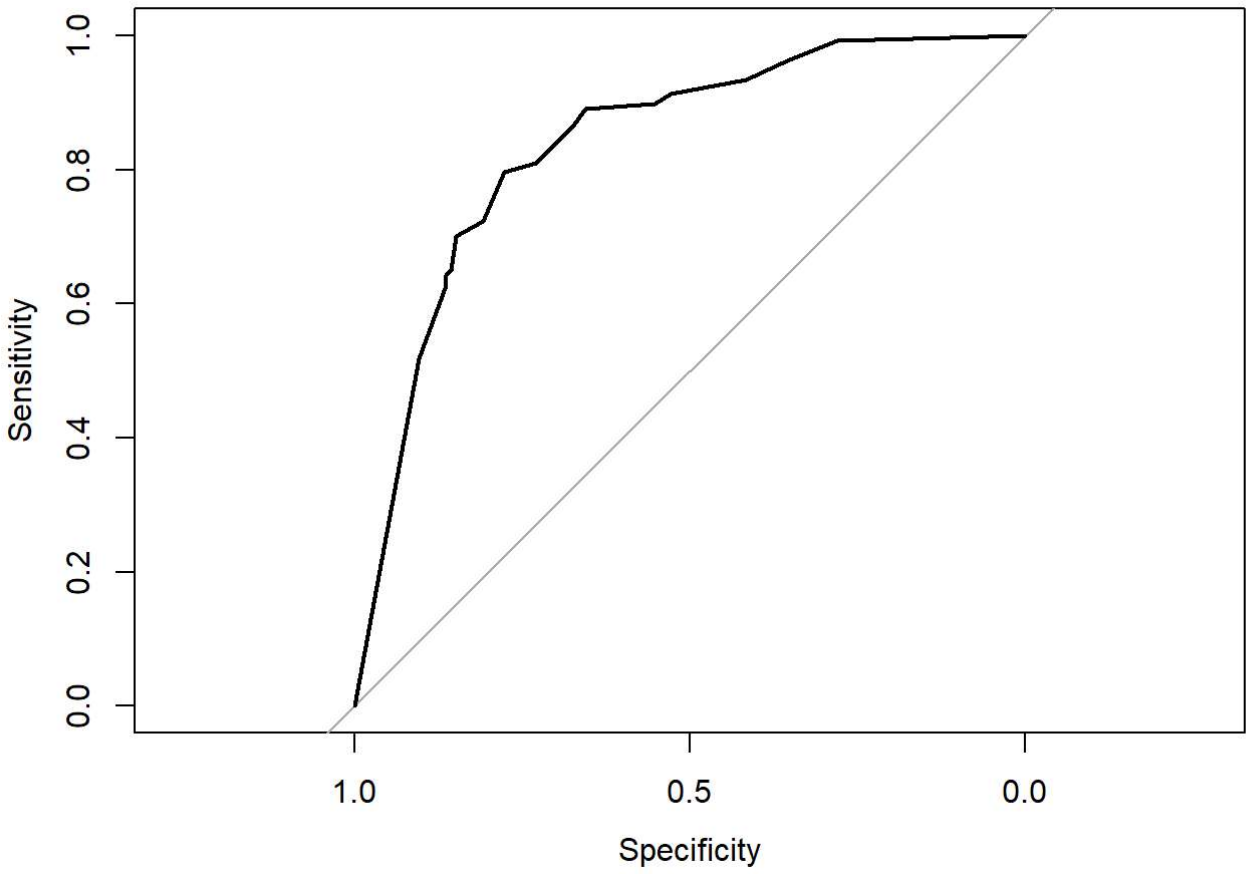
```
## [1] 0.810219
```

```
## Use "roc" function to create ROC curve, and then use "auc" function to calculate the value o
f AUC.
roc_cart<-roc(data_test$pollution_level,test_cart_prediction)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_cart)
```




```
auc(roc_cart)
```

```
## Area under the curve: 0.839
```

```
## Use "RMSE" function to calculate the value of RMSE
RMSE(test_cart_prediction,data_test$pollution_level)
```

```
## [1] 0.4113353
```

The performance of the original CART model is mediocre; its accuracy score is only 0.7709924, the area under the ROC curve is 0.839, the sensitivity is 0.728, even lower than 0.9, and the specificity is 0.810219. The value of RMSE is 0.4113353.

Perform a K-fold Cross-validation on CART Model, and Calculate the Value of RMSE.

```
## Set a seed
set.seed(48)

## Define training control
train_control_CART<-trainControl(method='cv',number=10)

## Use "train" and "method="rpart" function to fit a k-fold cross-validation on CART model
cv_CART<-train(pollution_level~.,
               data=new_data,method='rpart',
               trControl=train_control_CART)
print(cv_CART)
```

```
## CART
##
## 876 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 788, 788, 789, 789, 788, 788, ...
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.05107825  0.4059862  0.3418502  0.3263485
##   0.07960025  0.4200691  0.2966824  0.3460889
##   0.26519826  0.4715569  0.2227397  0.4387681
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.05107825.
```

According to the result, a k-fold cross-validation on CART model doesn't perform much better than the original CART model. We can see that the RMSE is now 0.4059862, just a little bit lower than 0.4113353.

Random Forest

```
## Set a seed
set.seed(48)

## Use "randomForest" function to fit a random forest model
train_rf<-randomForest(pollution_level~.,data=data_train)
## Use "predict" function to get predictions for the observations
test_rf_prediction<-predict(train_rf,newdata=data_test)
test_predict_class_rf<-ifelse(test_rf_prediction>0.5,"1","0")

## Calculate the accuracy score
mean(test_predict_class_rf==data_test$pollution_level)
```

```
## [1] 0.8282443
```

```
## Use "table" function to build a confusion matrix
conf_M_rf<-table(test_predict_class_rf,data_test$pollution_level)
conf_M_rf
```

```
##
## test_predict_class_rf   0   1
##                     0 100  20
##                     1  25 117
```

```
## Use "sensitivity" and "specificity" functions to get the sensitivity and specificity values
sensitivity(conf_M_rf)
```

```
## [1] 0.8
```

```
specificity(conf_M_rf)
```

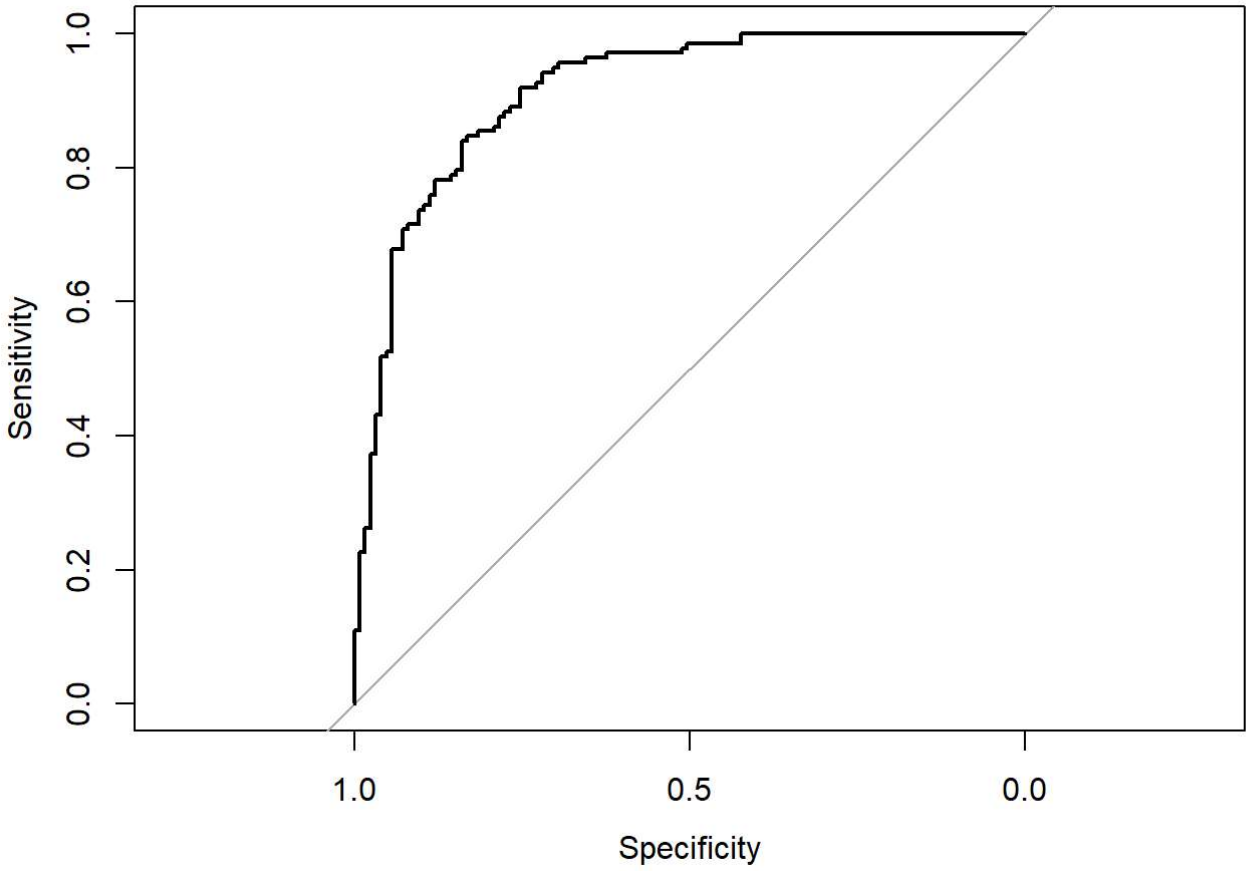
```
## [1] 0.8540146
```

```
## Use "roc" function to create ROC curve, and then use "auc" function to calculate the value o
f AUC.
roc_rf<-roc(data_test$pollution_level,test_rf_prediction)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_rf)
```




```
auc(roc_rf)
```

```
## Area under the curve: 0.9135
```

```
## Use "RMSE" function to calculate the value of RMSE
RMSE(test_rf_prediction,data_test$pollution_level)
```

```
## [1] 0.3463442
```

The performance of the original Random Forest model is great; its accuracy score is 0.8282443, the area under the ROC curve is 0.9135, the sensitivity is 0.8, and the specificity is 0.8540146. The value of RMSE is 0.3463442.

Perform a K-fold Cross-validation on Random Forest Model, and Calculate the Value of RMSE.

```
## Set a seed
set.seed(48)

## Define training control
train_control_rf<-trainControl(method='cv',number=10)

## Use "train" and "method=" function to fit a k-fold cross-validation on random forest model
cv_rf<-train(pollution_level~.,
             data=new_data,method='rf',
             trControl=train_control_rf)
print(cv_rf)
```

```
## Random Forest
##
## 876 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 788, 788, 789, 789, 788, 788, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   2     0.3463817  0.5255529  0.2666002
##   5     0.3402637  0.5365705  0.2426815
##   9     0.3439319  0.5257660  0.2376081
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 5.
```

According to the result, a k-fold cross-validation on Random Forest model doesn't perform much better than the original Random Forest model. We can see that the RMSE is now 0.3402637, just a little bit lower than 0.3463442.

Generalized Additive Models (GAMs)

```
## Set a seed
set.seed(48)

## Use "gam" function to fit a Generalized additive models
train_gam<-gam(pollution_level~.,data=data_train)
## Use "predict" function to get predictions for the observations
test_gam_prediction<-predict(train_gam,newdata=data_test)
test_predict_class_gam<-ifelse(test_gam_prediction>0.5,"1","0")

## Calculate the accuracy score
mean(test_predict_class_gam==data_test$pollution_level)
```

```
## [1] 0.7290076
```

```
## Use "table" function to build a confusion matrix
conf_M_gam<-table(test_predict_class_gam,data_test$pollution_level)
conf_M_gam
```

```
##
## test_predict_class_gam   0   1
##                      0  80  26
##                      1  45 111
```

```
## Use "sensitivity" and "specificity" functions to get the sensitivity and specificity values
sensitivity(conf_M_gam)
```

```
## [1] 0.64
```

```
specificity(conf_M_gam)
```

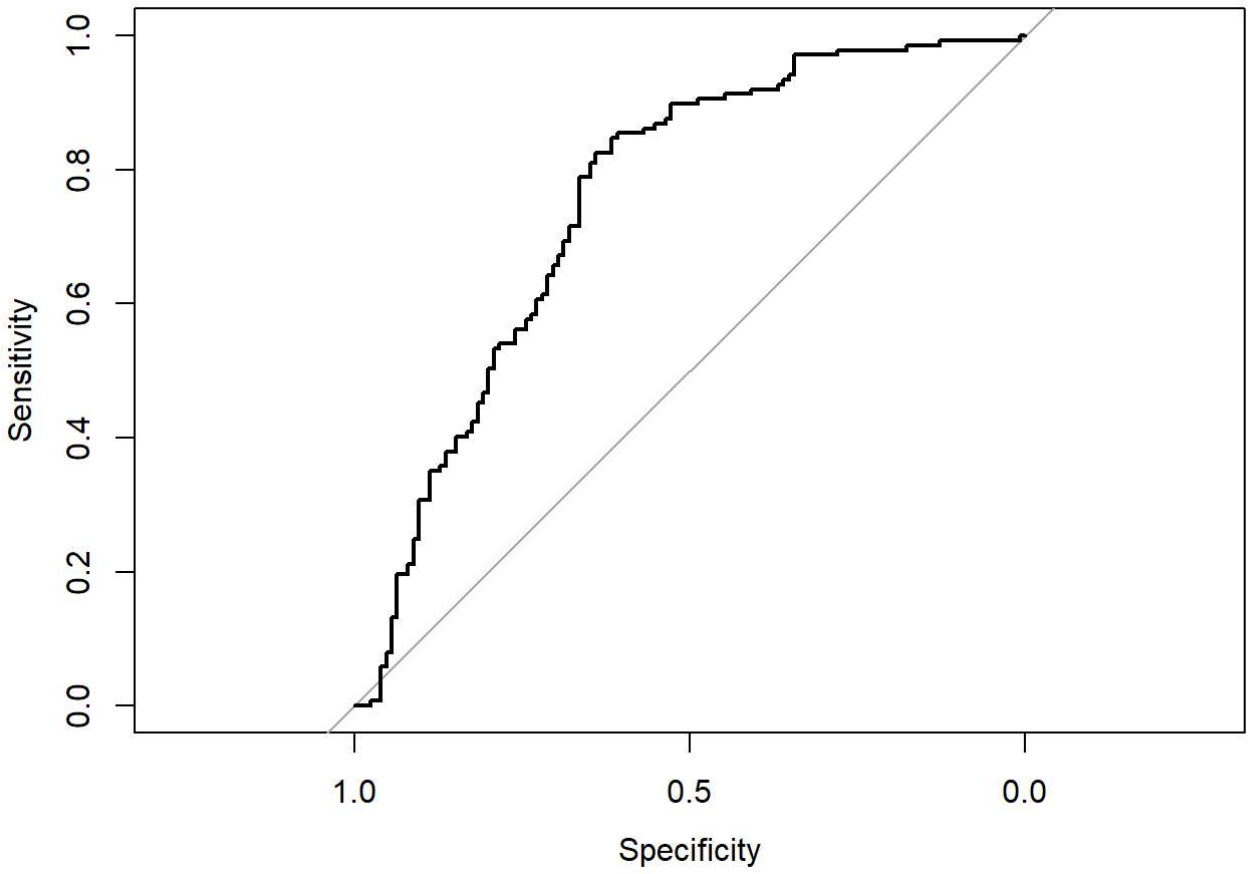
```
## [1] 0.810219
```

```
## Use "roc" function to create ROC curve, and then use "auc" function to calculate the value o
f AUC.
roc_gam<-roc(data_test$pollution_level,test_gam_prediction)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_gam)
```




```
auc(roc_gam)
```

```
## Area under the curve: 0.7516
```

```
## Use "RMSE" function to calculate the value of RMSE
RMSE(test_gam_prediction,data_test$pollution_level)
```

```
## [1] 0.4742228
```

The performance of the original GAMs is mediocre; its accuracy score is only 0.7290076, the area under the ROC curve is 0.7516, the sensitivity is 0.64, even lower than 0.7, and the specificity is 0.810219. The value of RMSE is 0.4742228.

Perform a K-fold Cross-validation on Generalized Additive Models, and Calculate the Value of RMSE.

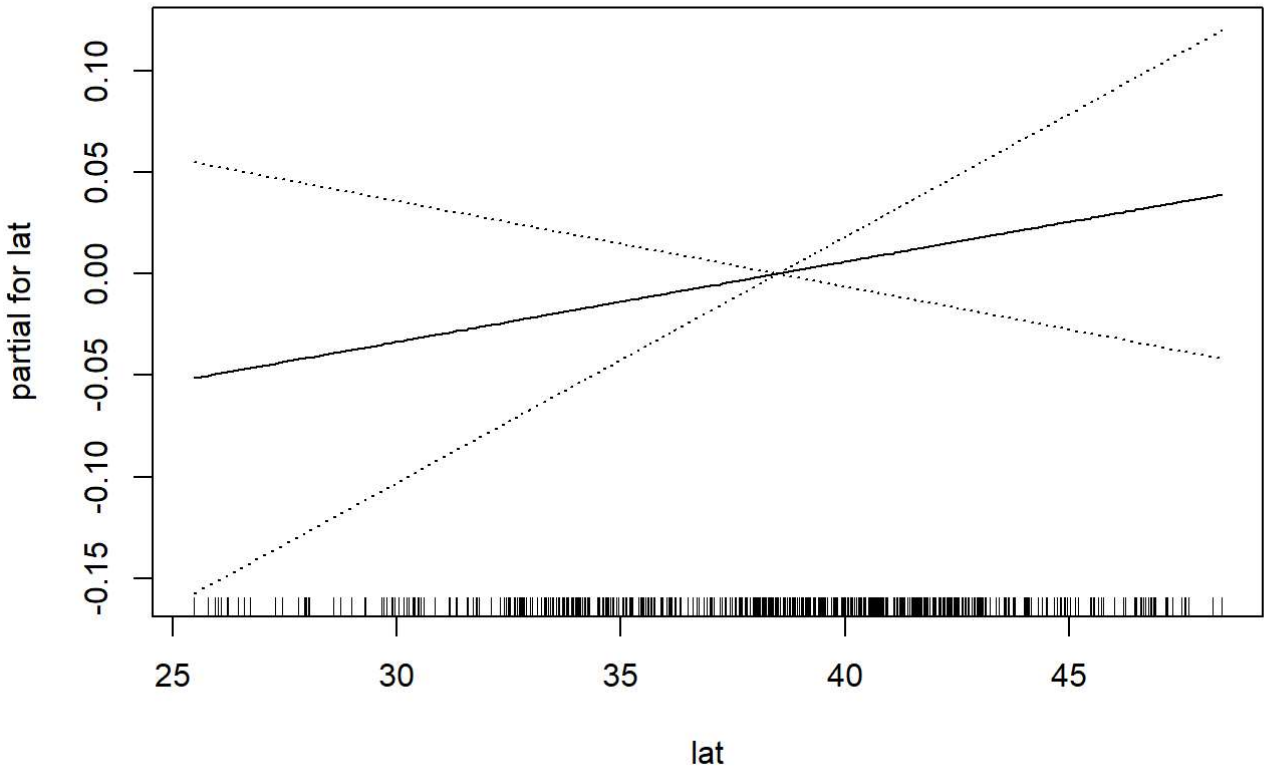
```
## Set a seed
set.seed(48)

## Define training control
train_control_gam<-trainControl(method='cv',number=10)

## Use "train" and "method=" function to fit a k-fold cross-validation on Generalized additive
models
cv_gam<-train(pollution_level~.,
              data=new_data,method='gam',
              trControl=train_control_gam)
print(cv_gam)
```

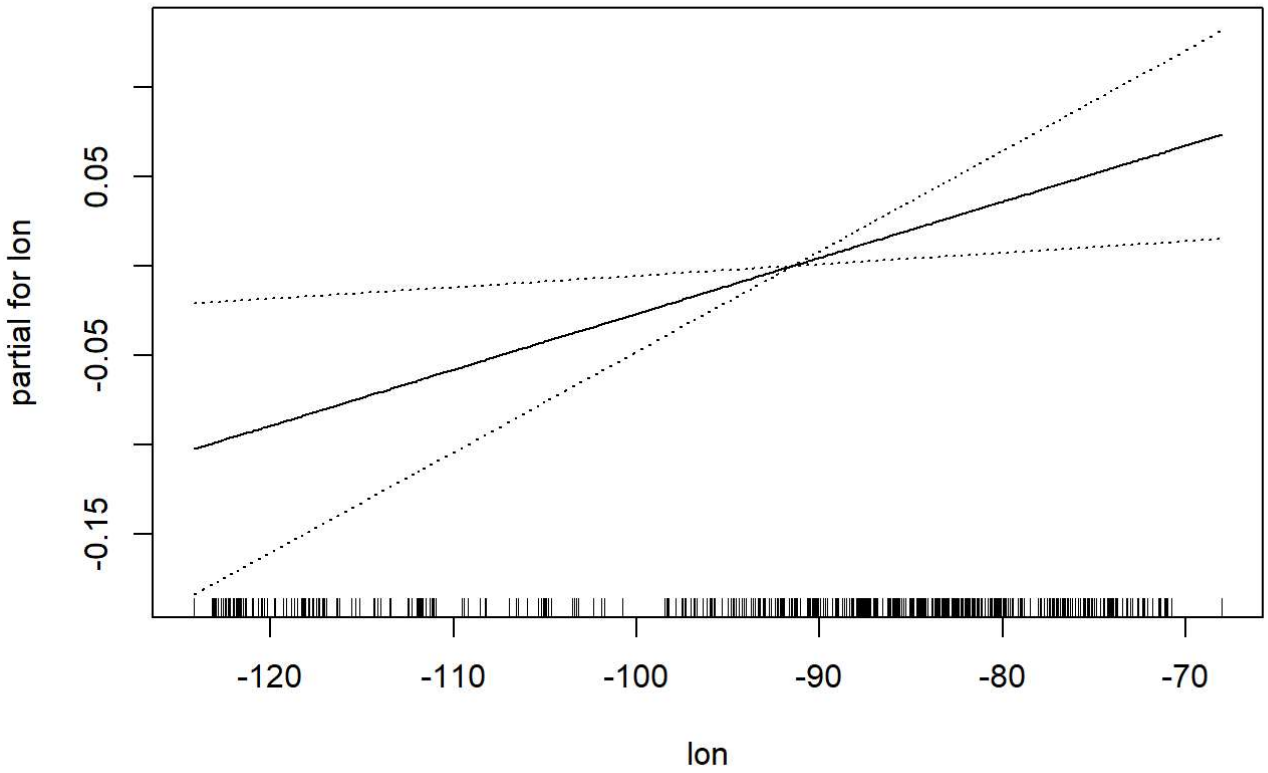
```
## Generalized Additive Model using Splines
##
## 876 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 788, 788, 789, 789, 788, 788, ...
## Resampling results across tuning parameters:
##
##   select  RMSE       Rsquared   MAE
##   FALSE   0.3722462  0.4465237  0.2909819
##    TRUE   0.3729257  0.4451404  0.2909273
##
## Tuning parameter 'method' was held constant at a value of GCV.Cp
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were select = FALSE and method = GCV.Cp.
```

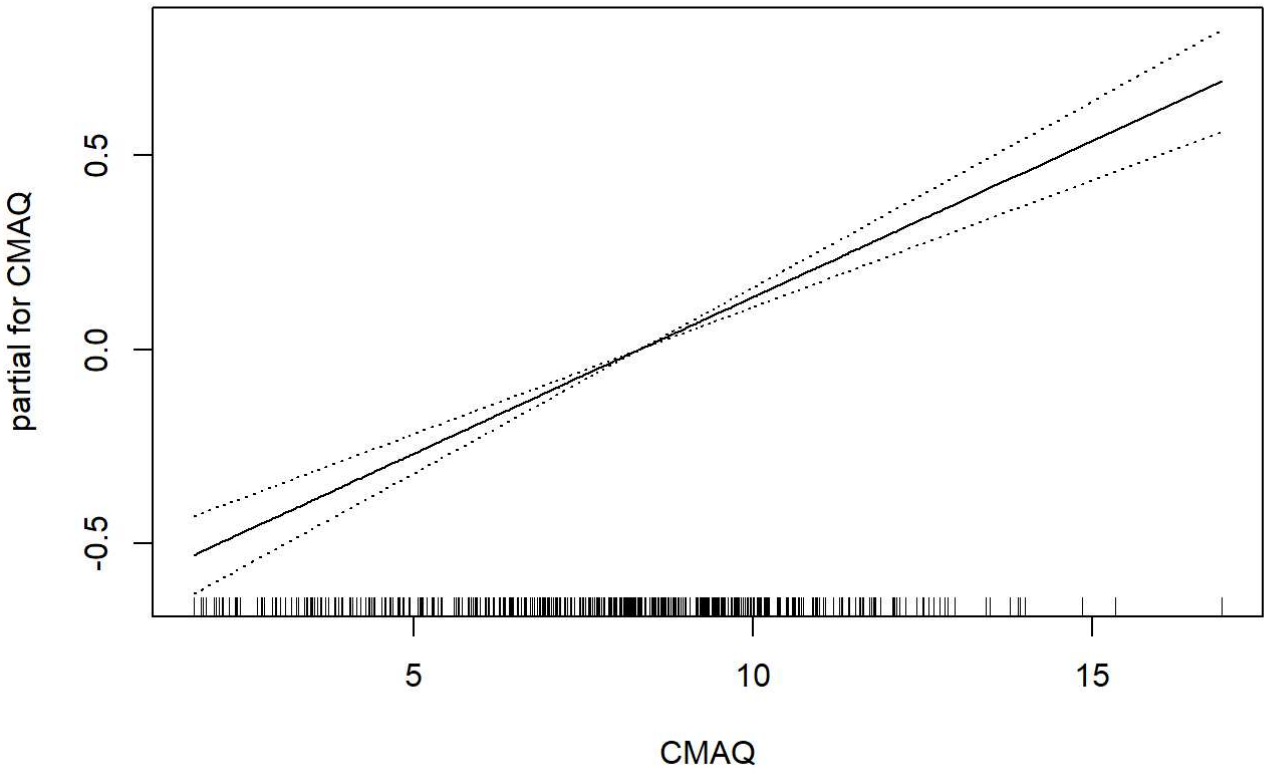
According to the result, a k-fold cross-validation on GAMs model performs better than the original Random GAMs model. We can see that the RMSE is now 0.3722462, much smaller than 0.4742228.

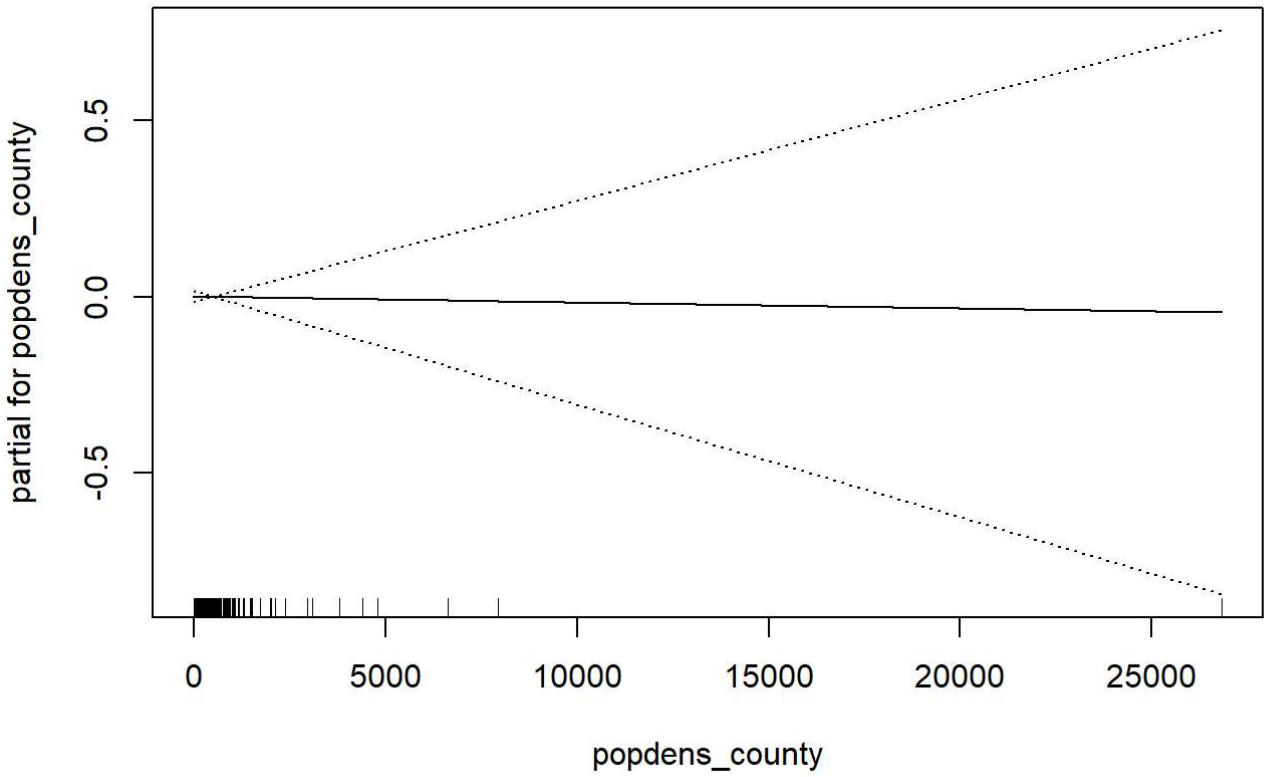
Plot the Result of the Generalized Additive Models

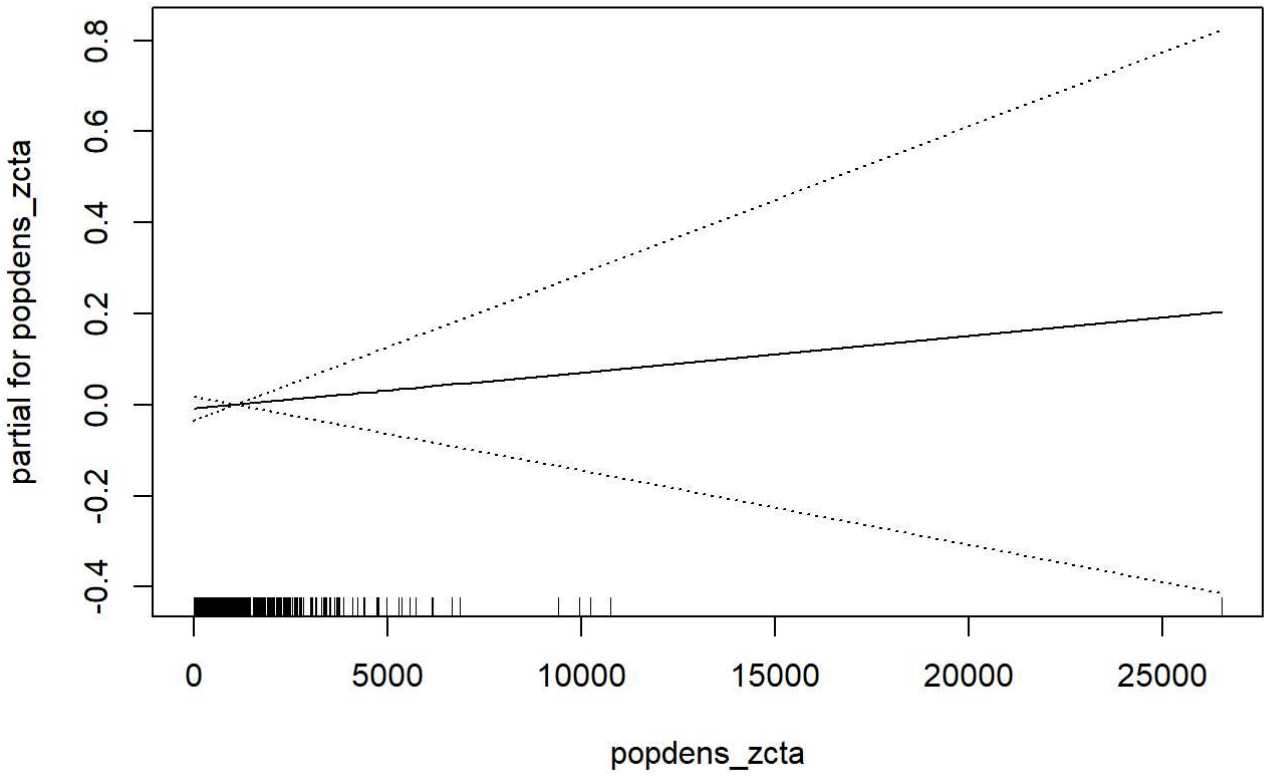
```
## Use "plot" function to plot the visualization graph
plot(train_gam,se=TRUE)
```

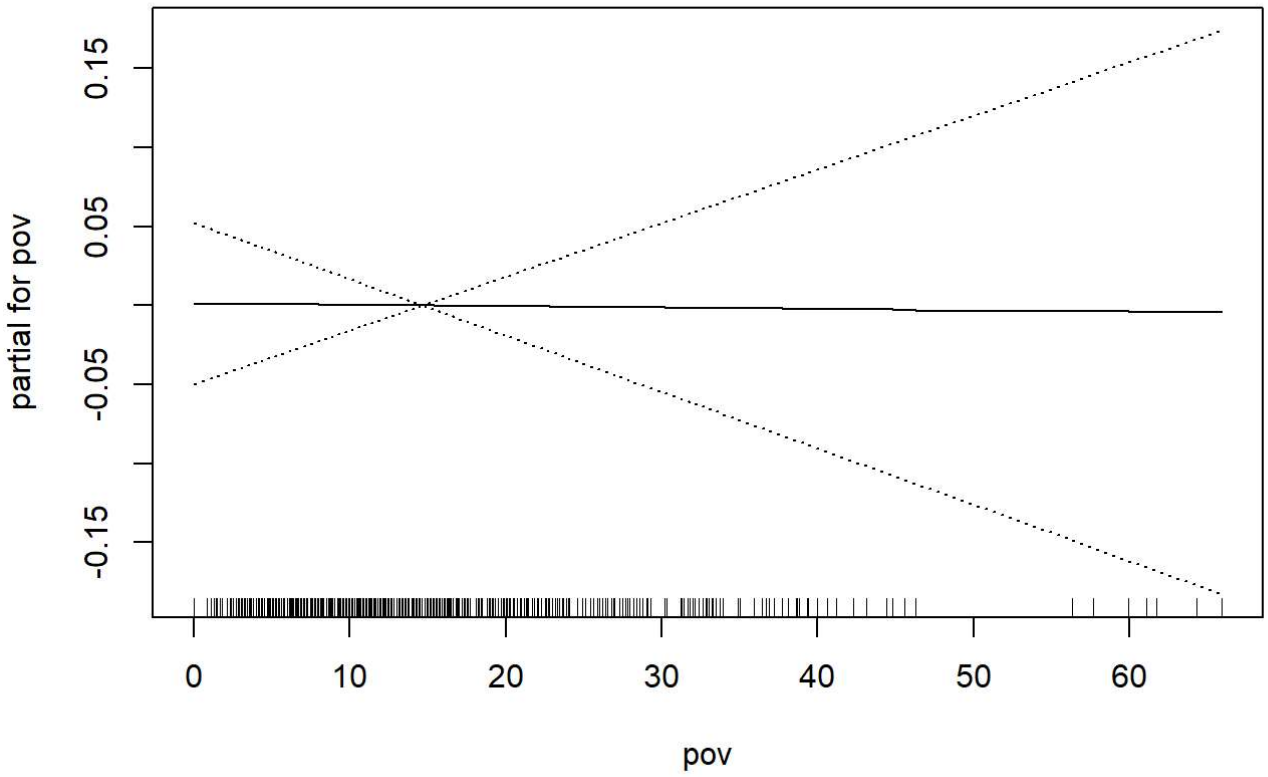

partial for lat
-0.15
-0.10
-0.05
0.00
0.05
0.10
25
30
35
40
45
lat

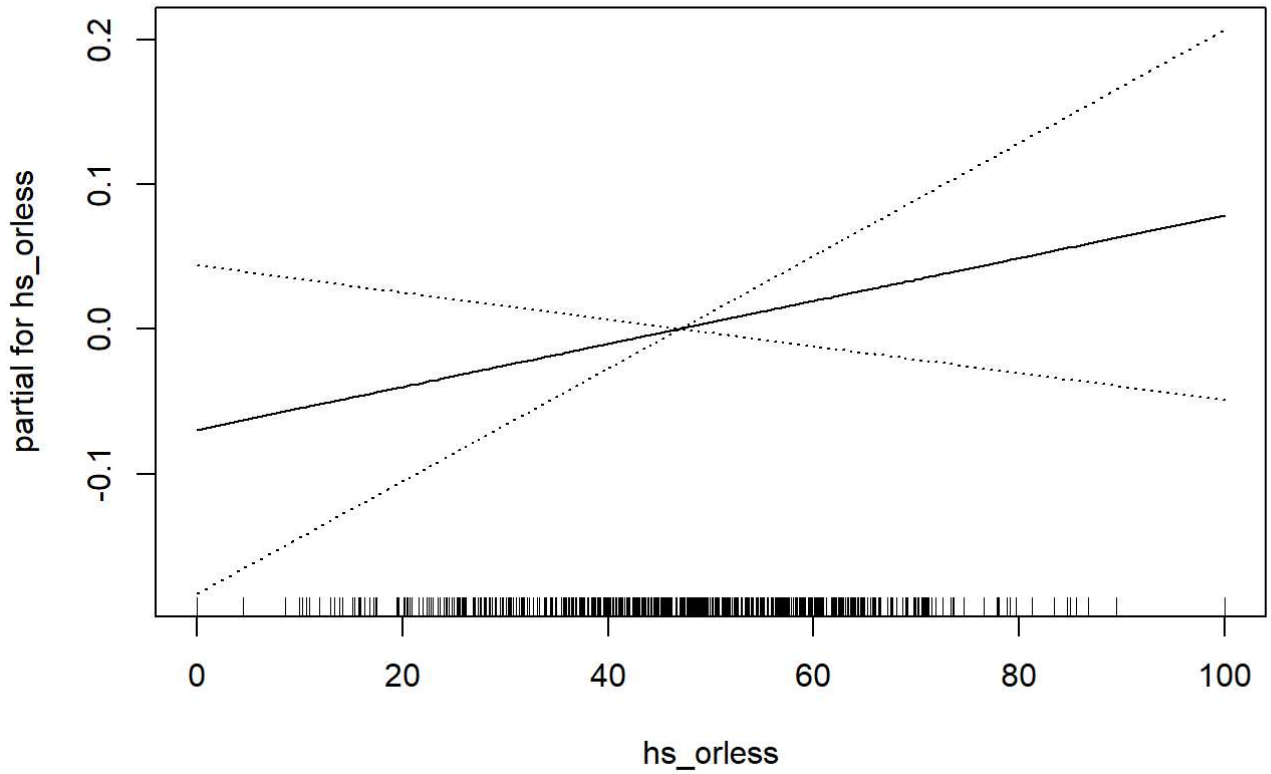

partial for lon
-0.15
-0.05
0.05
-120
-110
-100
-90
-80
-70
lon

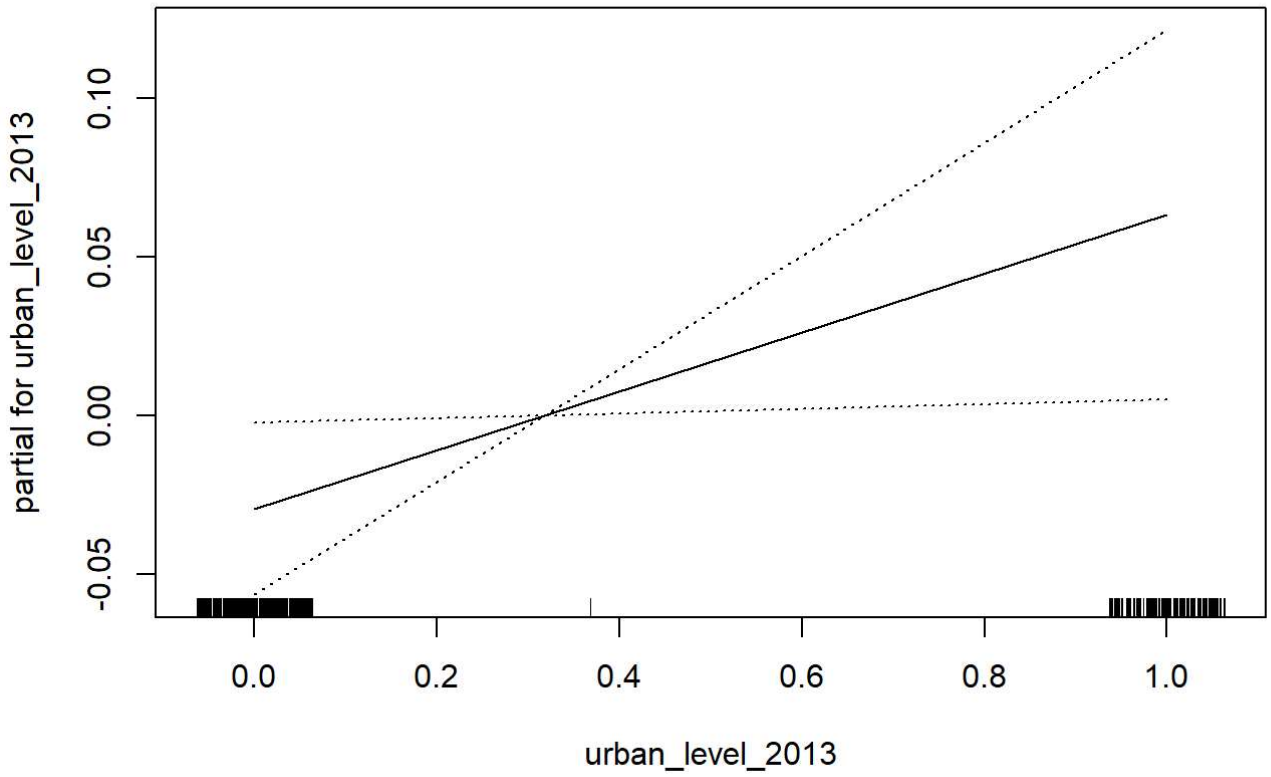

partial for CMAQ
0.5
0.0
-0.5
5
10
15
CMAQ

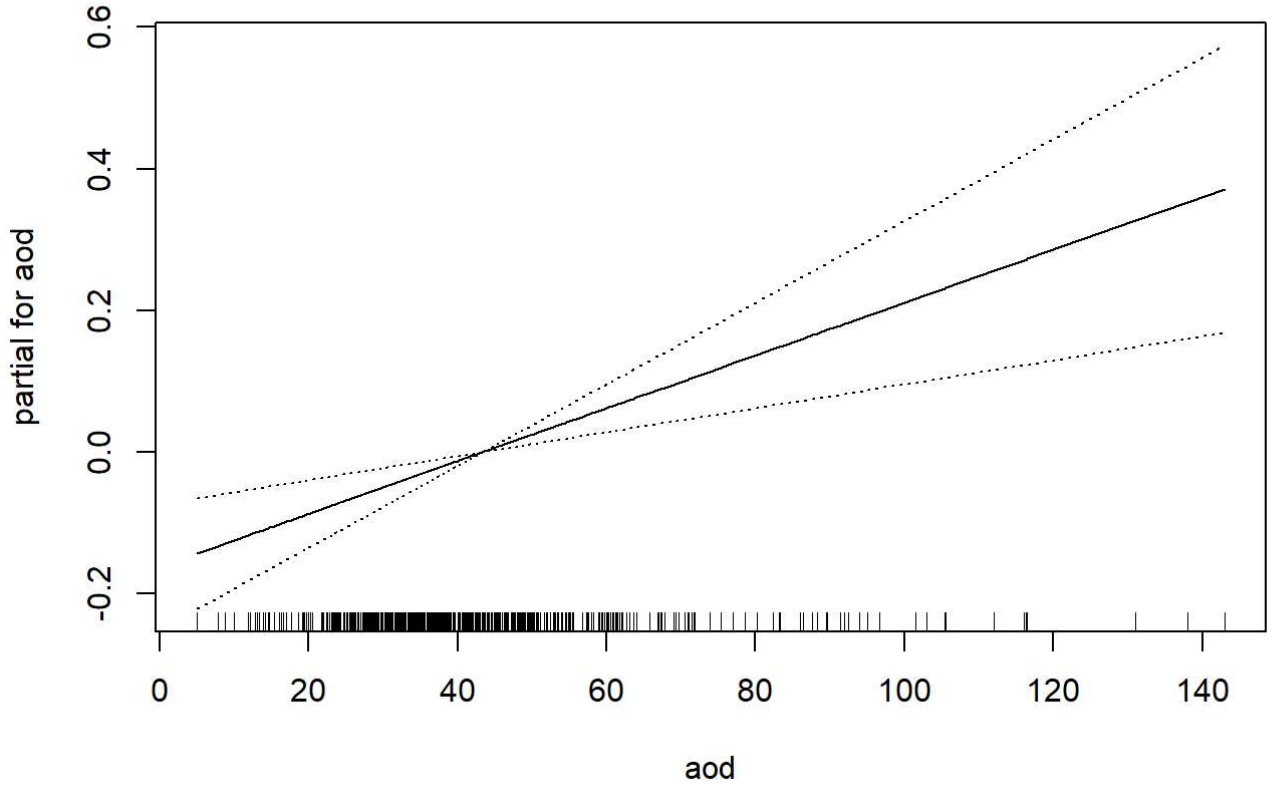

partial for popdens_county
0.5
0.0
-0.5
0
5000
10000
15000
20000
25000
popdens_county


partial for popdens_zcta
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
0
5000
10000
15000
20000
25000
popdens_zcta


partial for pov
-0.15
-0.05
0.05
0.15
0
10
20
30
40
50
60
pov


partial for hs_orless
0.2
0.1
0.0
-0.1
0
20
40
60
80
100
hs_orless


partial for urban_level_2013
0.10
0.05
0.00
-0.05
0.0
0.2
0.4
0.6
0.8
1.0
urban_level_2013



There are totally nine graphs, each of which corresponds to an independent variable. Through these nine graphs, we can see that there is no independent variable that is inversely proportional to the dependent variable. Among all the variables, the trend of popdens_zcta is a horizontal line, which means it has no apparent effect on pollution level; also, although the trend of poverty's curve is sloping to the upper right, the slope is very close to zero. Therefore, poverty also has no apparent effect on pollution level. Furthermore, we can see that the slopes of aod and CMAQ are the largest, indicating that these two variables have the largest effect on pollution level.

Create two Tables Summarizing the Prediction Metrics Across All of the Models

```
## Use "tibble" function to create a table summarizing the prediction metrics across all of the
models
table_1<-tibble(
  Model=c("CART","Random Forest", "GAMs"),
  RMSE=c(0.4113353,0.3463442,0.4742228),
  Sensitivity=c(0.728,0.8,0.64),
  Specificity=c(0.810219,0.8540146,0.810219),
  AUC=c(0.839,0.9135,0.7516),
  Accuracy=c(0.7709924,0.8282443,0.7290076)
)
## Look at the table
table_1
```

```
## # A tibble: 3 × 6
##   Model          RMSE Sensitivity Specificity   AUC Accuracy
##   <chr>         <dbl>       <dbl>       <dbl> <dbl>    <dbl>
## 1 CART          0.411       0.728       0.810 0.839    0.771
## 2 Random Forest 0.346       0.8         0.854 0.914    0.828
## 3 GAMs          0.474       0.64        0.810 0.752    0.729
```

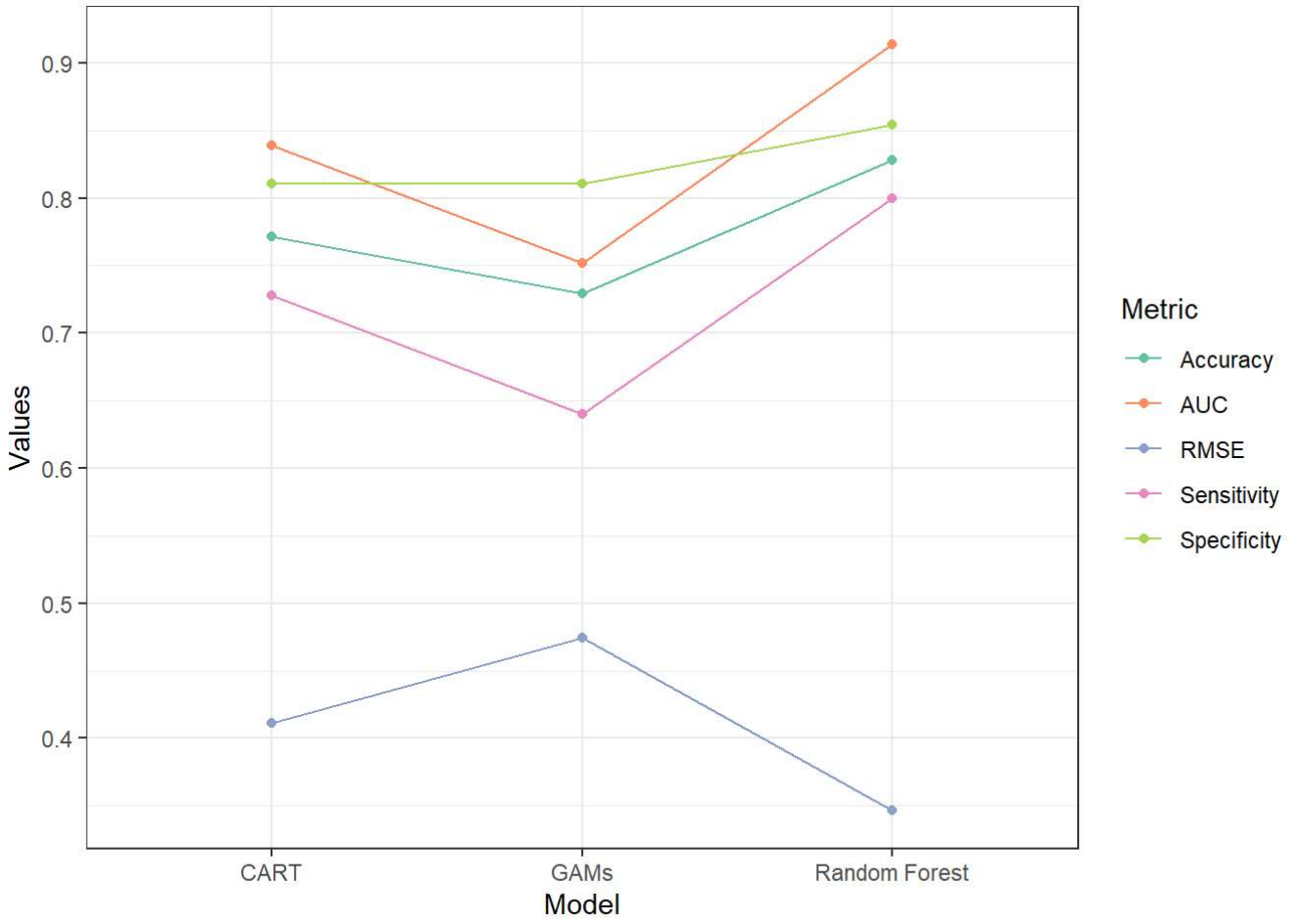
```
## Use "tibble" function to create a table summarizing the prediction metrics across all of the models that are fitted by k-fold cross-validation method
table_2<-tibble(
  Model=c("CART_cv","RF_cv","GAMs_cv"),
  RMSE=c(0.4059862,0.3402637,0.3722462),
  Rsquared=c(0.3418502,0.5365705,0.4465237),
  MAE=c(0.3263485,0.2426815,0.2909819)
)
## Look at the table
table_2
```

```
## # A tibble: 3 × 4
##   Model    RMSE Rsquared   MAE
##   <chr>   <dbl>    <dbl> <dbl>
## 1 CART_cv 0.406    0.342 0.326
## 2 RF_cv   0.340    0.537 0.243
## 3 GAMs_cv 0.372    0.447 0.291
```

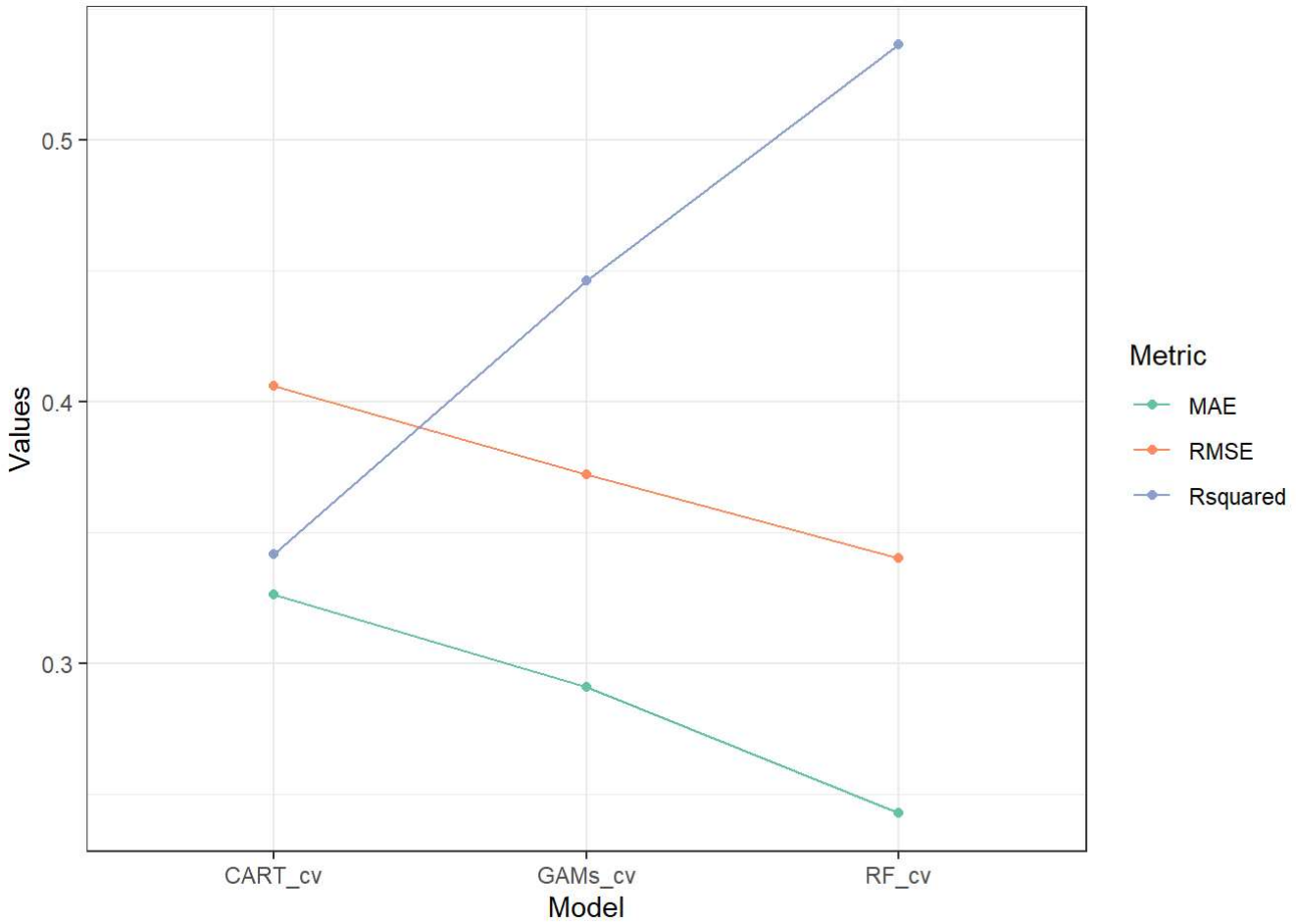
Through comparing the first data frame (the important parameters from the three original models), we can see that Random Forest model has the lowest RSME , the highest sensitivity,the highest specificity, the highest accuracy, and the highest AUC, and the highest accuracy; GAMs model has the highest RSME, the lowest sensitivity, the lowest AUC, and the lowest accuracy; in this case, we can conclude that among these three models, Random Forest model performs best, GAMs model performs worst, and CART model's performance is in the middle level. Then, by comparing the k-fold cross-validation method based on these three models, we can see that RF_cv has the lowest RSME, the highest R-squared, and the lowest MAE, which means it has the smallest root mean-squared error, the strongest explanatory power of an independent variable on a dependent variable, and the smallest degree of the average error between the minimum predicted value and the true value; we can also see that CART_cv has the highest RSME, the highest MAE, and the lowest R-squared; therefore, different from the original model predictions, after applying the k-fold cross-validation method, Random Forest remained the best performing predictive model while CART became the worst performing predictive model. Finally, by comparing the RSME of these two data sets, we can see that for all these three models, after using the k-fold cross-validation method, their RMSEs all become smaller. This means that the k-fold cross-validation method can help improve the performance of models.

Create two Visualization Graphs Summarizing the Prediction Metrics Across All of the Models

```
## Use "pivot_longer" function to transform the "table_1" data set from wide format to long for
mat
table_1_long <- pivot_longer(table_1,
                             cols = c(RMSE, Sensitivity, Specificity, AUC, Accuracy),
                             names_to = "Metric",
                             values_to = "Value")

## Use "ggplot" function to visualize the prediction performances of these three models
ggplot(table_1_long, aes(x = Model, y = Value, color = Metric, group = Metric)) +
  geom_line() +
  geom_point() +
  scale_color_brewer(palette = "Set2") +
  labs(x = "Model", y = "Values", color = "Metric") +
  theme_bw()
```



```
## Use "pivot_longer" function to transform the "table_1" data set from wide format to long for
mate
table_2_long <- pivot_longer(table_2,
                             cols = c(RMSE, Rsquared, MAE),
                             names_to = "Metric",
                             values_to = "Value")

## Use "ggplot" function to visualize the prediction performances of these three models
ggplot(table_2_long, aes(x = Model, y = Value, color = Metric, group = Metric)) +
  geom_line() +
  geom_point() +
  scale_color_brewer(palette = "Set2") +
  labs(x = "Model", y = "Values", color = "Metric") +
  theme_bw()
```



By visualizing the table, we can see the predictive performance of each model more clearly. For the original models, Random Forest model has the highest points of AUC, sensitivity, specificity, and accuracy, and the lowest point of RMSE; GAMs model has the lowest points of AUC, accuracy, and sensitivity, and the highest point of RSME. For the k-fold cross-validation method based models, Random Forest has the lowest points of RMSE and MAE, and the highest point of R-squared; CART model has the the highest points of RMSE and MAE, and the lowest point of R-squared. These results are the same with what we got from the table.

Discussion

In the Discussion part, I will answer the four questions from the assignment requirement, and then discuss the whole process of doing this project, which will include my reflections and gratitude.

The Answer of the Questions

1. Find the Locations that Have the Closest and Furthest Predictions From the Observed Values

In this question, I will use the prediction results from the best model, random forest, to find the closest and the furthest predictions and print out the their corresponding locations (by county). Then, I will give my reasons about why these locations are the good and bad performances.

```
## Calculate the predictions errors by using the equation "prediction values minus true values"
errors<-test_rf_prediction-data_test$pollution_level
## Create the error column
data_test$error<-errors
## Use "order" function to sort the size of the error
test_data_sorted<-data_test[order(as.vector(data_test$error)),]
## List the 10 closest and the furthest predictions
closest_predictions<-test_data_sorted[1:10,]
furthest_predictions<-test_data_sorted[(nrow(test_data_sorted)-9):nrow(test_data_sorted),]
closest_predictions
```

```
## # A tibble: 10 × 11
##    pollution_level   lat    lon  CMAQ popdens_county popdens_zcta   pov
##              <dbl> <dbl>  <dbl> <dbl>          <dbl>        <dbl> <dbl>
##  1               1  29.9  -93.9 11.3           111.        1044.   10.2
##  2               1  26.1  -97.2  5.33          176.         389.   28.7
##  3               1  43.5  -87.8  7.58          143.          36.4   2.8
##  4               1  47.5 -116.   3.76            1.87         5.06 25.8
##  5               1  35.6  -94.8  7.78           24.3         13.7   8.6
##  6               1  43.1  -89.4  8.00          157.         1378.    5.68
##  7               1  39.6  -79.9  7.06          103.         1141.   14.6
##  8               1  40.5 -122.   3.43           18.1         149.   15.3
##  9               1  29.7  -95.3 14.0           928.         529.   17.9
## 10               1  39.5  -80.1  6.51           70.6        802.   11.3
## # ℹ 4 more variables: hs_orless <dbl>, urban_level_2013 <dbl>, aod <dbl>,
## #   error <dbl>
```

```
furthest_predictions
```

```
## # A tibble: 10 × 11
##    pollution_level   lat   lon  CMAQ popdens_county popdens_zcta   pov hs_orless
##              <dbl> <dbl> <dbl> <dbl>          <dbl>        <dbl> <dbl>     <dbl>
##  1               0  38.7 -78.5  8.37           29.9         29.8   4.8      61.8
##  2               0  41.2 -88.2  9.76           64.7         21.7  12.6      50.8
##  3               0  32.8 -80.0  8.53          148.        12577.    0         0
##  4               0  40.5 -74.4  9.13         1012.         3369.   15.8      65.1
##  5               0  35.2 -77.6  8.90           57.3         67.4  30.4      57.7
##  6               0  41.6 -81.6  8.80         1081.         1846.   28.4      59.2
##  7               0  33.0 -80.1  8.53          148.         9271.   23.0      57.3
##  8               0  37.7 -84.3  9.47           73.2         78.9  13.1      42.7
##  9               0  41.6 -87.2 16.9           152.          568.    9.9      53.5
## 10               0  39.1 -77.1 10.4           764.          421.    4.8      31.7
## # ℹ 3 more variables: urban_level_2013 <dbl>, aod <dbl>, error <dbl>
```

```
## Use "filter" and "select" functions to print out the closest and the furthest counties
closest_location<-data%>%filter(pov==4.800,lat>39.1 & lat<39.2)%>%select(county)
furthest_location<-data%>%filter(pov==11.1,lat>30.21 & lat<30.22)%>%select(county)
closest_location
```

```
## # A tibble: 1 × 1
##   county
##   <chr>
## 1 Montgomery
```

```
furthest_location
```

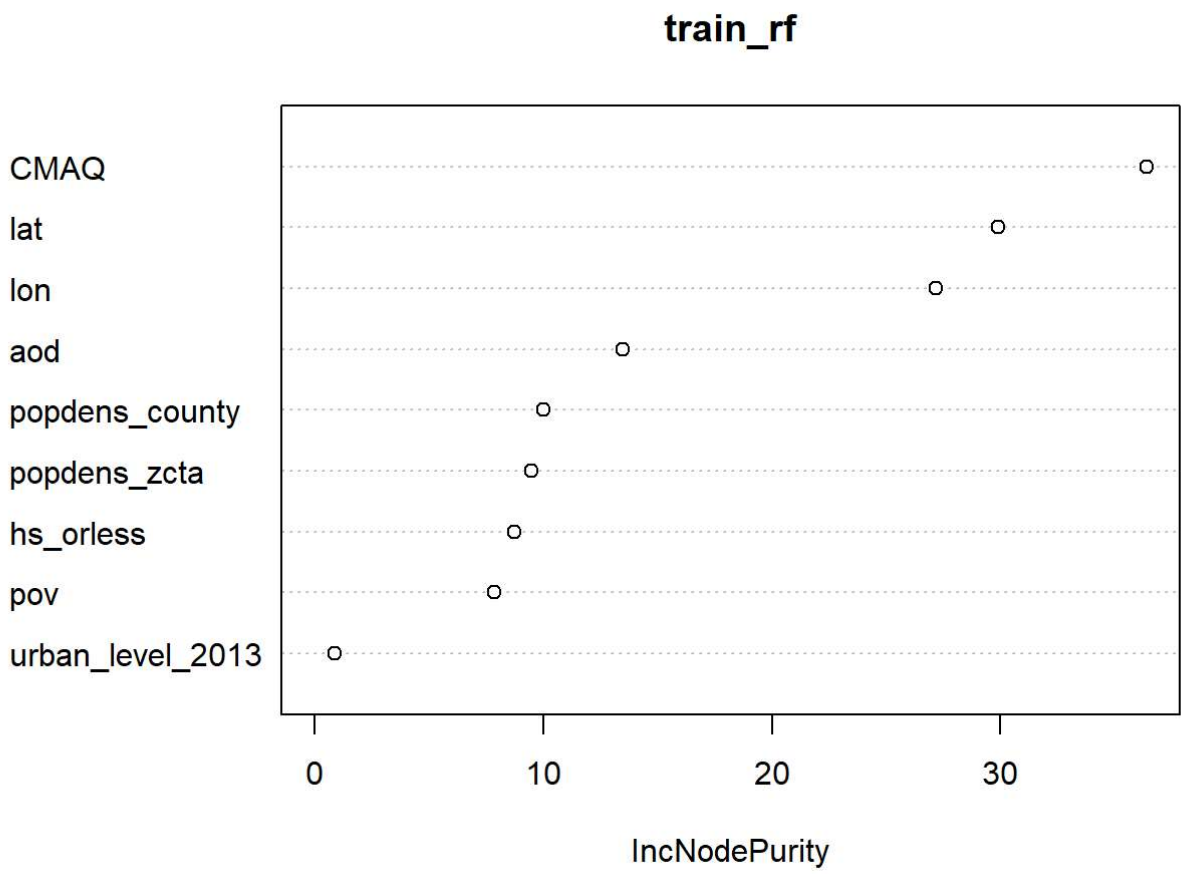
```
## # A tibble: 1 × 1
##   county
##   <chr>
## 1 Iberville
```

Hence, the closest location is Montgomery and the furthest location is Iberville. The reasons for the good or bad performance at these two locations are multiple. Firstly, geographical factor is a big issue: geographical features, such as being located near highways, industrial areas, or construction sites, could have a notable effect on the pollution levels. Secondly, the outcome could be impacted by the time element: there could be a link between higher pollution levels and time periods such as rush hour or the summer season. Lastly, some observations at specific locations might be outliers, indicating that they are significantly different from the other observations at that location, and may cause the model to perform poorly for these locations.

2. The Variables Might Predict where Model Performs Well or Not

In this question, I will plot the significant level of each variables and estimate which variables might predict where the model performs well or not.

```
## Use "varImpPlot" function to make the graph of the feature importance for random forest mode
l
varImpPlot(train_rf)
```



According to the graph, we can see that the three most significant variables are CMAQ, latitude, and longitude, which means the geographic factor and the instrument prediction result might predict where the model performs well or not. Since air quality is affected by traffic condition, number of factories, green area, and so on, I think including these variables can improve the model performance.

3. The Estimation on CMAQ and Aod.

In this question, I will estimate the effectiveness of CMAQ and aod. Since we have already plotted the feature importance of random forest model, we can see how well do CMAQ and aod predict ground-level concentration PM2.5. Then, I will test how does the performance change when CMAQ and aod are not included by running the model again. Firstly, test if aod is not included.

```
## Create the new data set
discussion_data<-binary_data%>%select(pollution_level,lat,lon,CMAQ,popdens_county,popdens_zcta,
pov,hs_orless,urban_level_2013)

## Set a seed
set.seed(48)

## Split the new train and test sets
idx_train_2<-createDataPartition(discussion_data$pollution_level,p=0.7)[[1]]
data_train_2<-discussion_data[idx_train_2,]
data_test_2<-discussion_data[-idx_train_2,]

## Use "randomForest" function to fit a random forest model
train_rf_2<-randomForest(pollution_level~.,data=data_train_2)
## Use "predict" function to get predictions for the observations
test_rf_prediction_2<-predict(train_rf_2,newdata=data_test_2)
test_predict_class_rf_2<-ifelse(test_rf_prediction_2>0.5,"1","0")

## Calculate the accuracy score
mean(test_predict_class_rf_2==data_test_2$pollution_level)
```

```
## [1] 0.8129771
```

```
## Use "table" function to build a confusion matrix
conf_M_rf_2<-table(test_predict_class_rf_2,data_test_2$pollution_level)
conf_M_rf_2
```

```
##
## test_predict_class_rf_2   0   1
##                       0  98  22
##                       1  27 115
```

```
## Use "sensitivity" and "specificity" functions to get the sensitivity and specificity values
sensitivity(conf_M_rf_2)
```

```
## [1] 0.784
```

```
specificity(conf_M_rf_2)
```

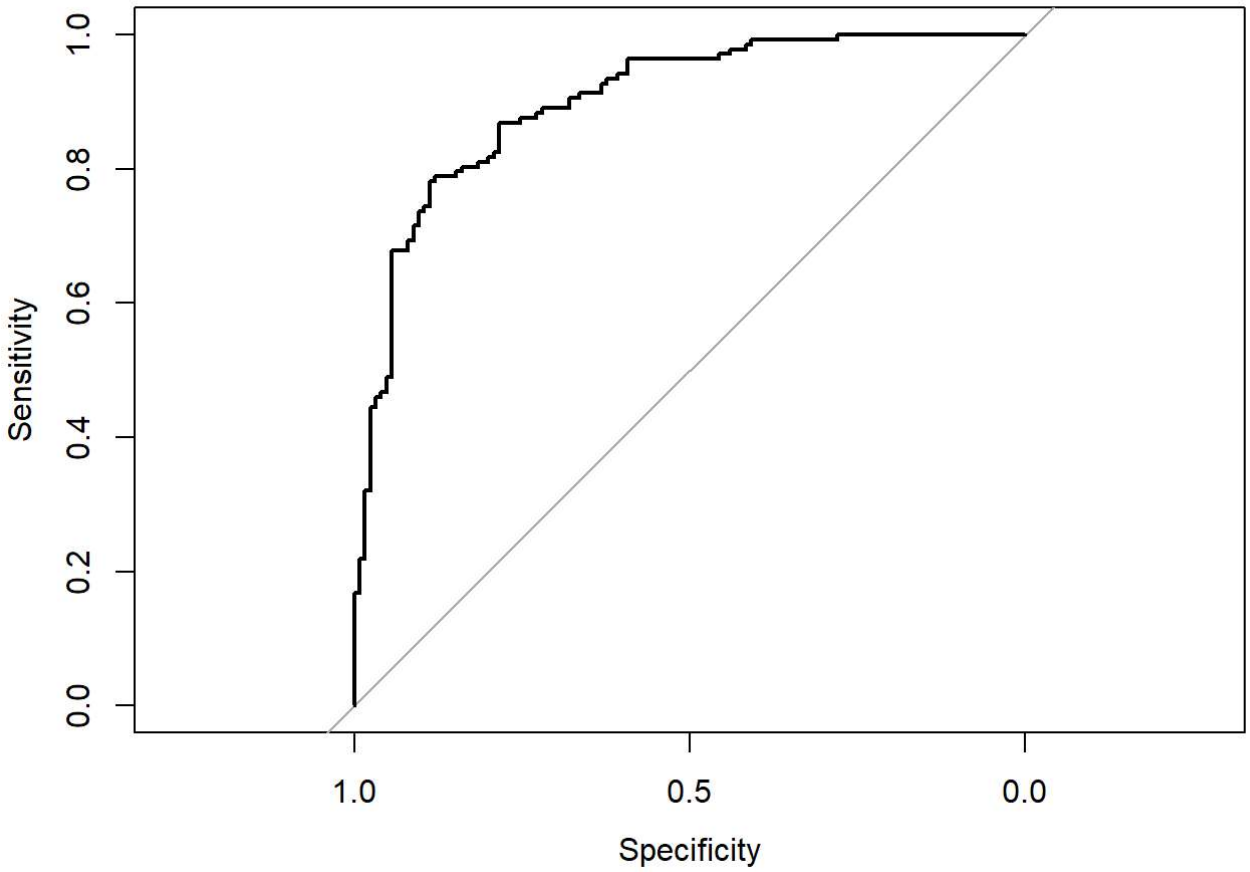
```
## [1] 0.8394161
```

```
## Use "roc" function to create ROC curve, and then use "auc" function to calculate the value o
f AUC.
roc_rf_2<-roc(data_test_2$pollution_level,test_rf_prediction_2)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_rf_2)
```




```
auc(roc_rf_2)
```

```
## Area under the curve: 0.9008
```

```
## Use "RMSE" function to calculate the value of RMSE
RMSE(test_rf_prediction_2,data_test_2$pollution_level)
```

```
## [1] 0.3583722
```

Secondly, test if CMAQ is not included.

```
## Create the new data set
discussion_data_2<-binary_data%>%select(pollution_level,lat,lon,aod,popdens_county,popdens_zct
a,pov,hs_orless,urban_level_2013)

## Set a seed
set.seed(48)

## Split the new train and test sets
idx_train_3<-createDataPartition(discussion_data_2$pollution_level,p=0.7)[[1]]
data_train_3<-discussion_data[idx_train_3,]
data_test_3<-discussion_data[-idx_train_3,]

## Set a seed
set.seed(48)

## Use "randomForest" function to fit a random forest model
train_rf_3<-randomForest(pollution_level~.,data=data_train_3)
## Use "predict" function to get predictions for the observations
test_rf_prediction_3<-predict(train_rf_3,newdata=data_test_3)
test_predict_class_rf_3<-ifelse(test_rf_prediction_3>0.5,"1","0")

## Calculate the accuracy score
mean(test_predict_class_rf_3==data_test_3$pollution_level)
```

```
## [1] 0.8244275
```

```
## Use "table" function to build a confusion matrix
conf_M_rf_3<-table(test_predict_class_rf_3,data_test_3$pollution_level)
conf_M_rf_3
```

```
##
## test_predict_class_rf_3   0   1
##                       0 100  21
##                       1  25 116
```

```
## Use "sensitivity" and "specificity" functions to get the sensitivity and specificity values
sensitivity(conf_M_rf_3)
```

```
## [1] 0.8
```

```
specificity(conf_M_rf_3)
```

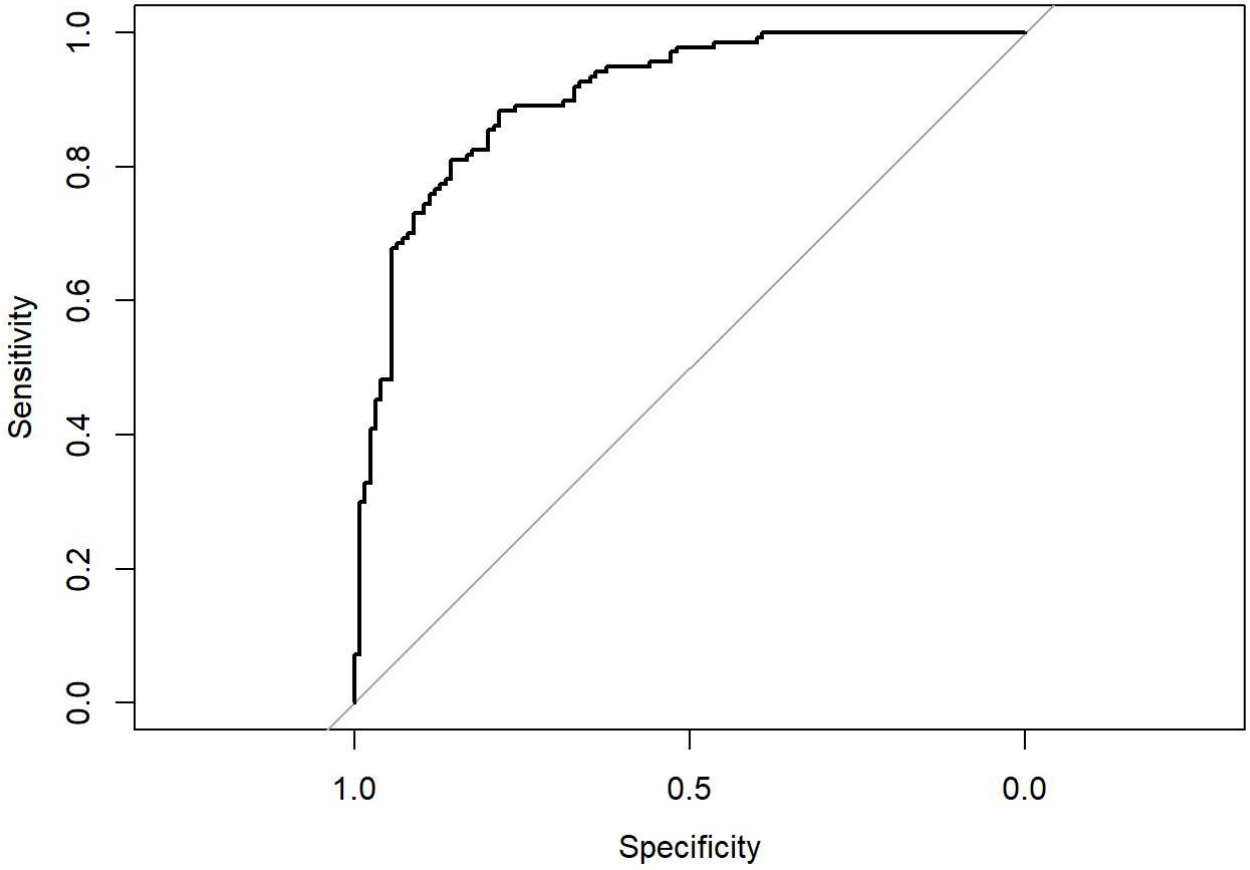
```
## [1] 0.8467153
```

```
## Use "roc" function to create ROC curve, and then use "auc" function to calculate the value o
f AUC.
roc_rf_3<-roc(data_test_3$pollution_level,test_rf_prediction_3)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_rf_3)
```




```
auc(roc_rf_3)
```

```
## Area under the curve: 0.9053
```

```
## Use "RMSE" function to calculate the value of RMSE
RMSE(test_rf_prediction_3,data_test_3$pollution_level)
```

```
## [1] 0.3546066
```

Through the feature importance graph from question 2, we can see that the score of feature importance score for CMAQ is the highest, but for aod is low, which means CMAQ did well on predicting ground_level concentrations of PM2.5, but aod is not as well as CMAQ. After doing the new prediction, for the data set without aod, the accuracy score is 0.8206107, sensitivity is 0.8828829, specificity is 0.7748344, AUC is 0.9091, and RMSE is 0.3542662; for the data set without CMAQ, the accuracy score is 0.8053435, sensitivity is 0.7384615, specificity is 0.8712121, AUC is 0.9162, and RMSE is 0.3549321. By comparing with the original prediction, we can see that no-aod has a higher accuracy score, higher sensitivity, lower specificity, lower AUC, and higher RMSE; no-CMAQ has lower accuracy, lower sensitivity, higher specificity, higher AUC, and higher RMSE. By comparing no-aod with no-CMAQ, we can see that no-aod has higher accuracy, higher sensitivity, lower specificity, lower AUC, but lower RMSE. This indicates that CMAQ predicts much better than aod.

4. The estimation of including Alaska or Hawaii

Since the climatic conditions, air quality regulations, and population densities in Hawaii and Alaska are very different from other states, the model's predictions may be less accurate in these two states, and hence the model may not perform well in those two states.

Discusses

Reflecting on the process of conducting this project, the biggest challenge is selecting the models. This is because not all models fit my project, and I need to have a deeper understanding of the models I wanted to use to check what kind of research are they designed for. Also, the code is very different for each model, which causes me to run bugs very often. The process of resolving errors is difficult and time-consuming. However, through these processes, I learned more model-building methods and I realized the big difference between classification and regression. Reflecting on the performance of my final prediction model, it predicted as well as I originally expected. Just like what I have mentioned in the introduction part, random forest will be the best model, and CART and aod are comparable; also, the estimation on no-CMAQ and no-aod is the same as what I assumed, too. Thanks for all the information about models on the internet which helped me learn to build more models!