Project 2

Alfred Zhu azz82

Predicting Ambient Air Pollution

Title and Introduction

The main goal of this project is to create a model that can predict the average PM2.5 concentration. Monitoring the PM2.5 concentration can be expensive and as a result, monitors tend to be placed in major cities so being able to predict the air quality in areas without those monitors can be beneficial for public health. I will test three different prediction models: linear regression, k-nearest neighbor, and random forest. I will choose the best model using the root mean-squared-error (RMSE) metric.

```
# Sets up needed packages and data
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.4     v rsample      1.1.1
## v dials        1.2.0     v tune         1.1.1
## v infer        1.0.4     v workflows    1.1.3
## v modeldata    1.1.0     v workflowsets 1.0.1
## v parsnip      1.1.0     v yardstick    1.2.0
## v recipes      1.0.6
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(parsnip)

dat0 <- 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.
```

I am going to select these variables as the predictors for PM2.5 concentration and will be using them for all models to make things more simple.

CMAQ: It is a candidate for replacing ground-based monitors using satellite observations

AOD: Same as CMAQ

zcta_pop: The population in the zip-code area could affect the air quality. Conventional thinking says a bigger population would generally result in more pollution.

log_pri_length_10000: The number of primary roads could affect air quality due to it resulting in a bigger density of motor vehicles resulting in more pollution. 10000 is arbitrarily chosen.

log_nei_2008_pm25_sum_25000: Major emissions would affect the air quality. The number is arbitrarily chosen.

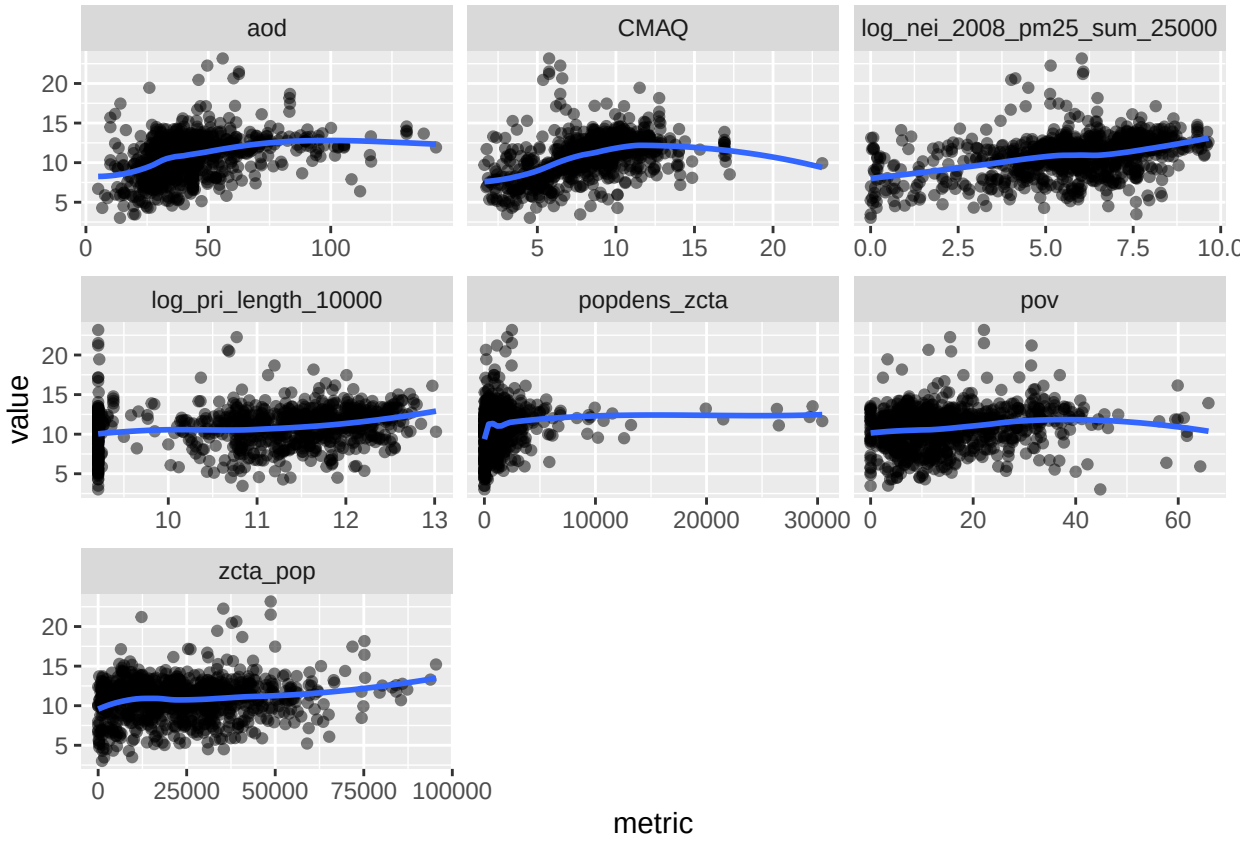
popdens_zcta: The population density could affect pollution as conventianal thinking says a bigger population density would generally result in more pollution.

pov: Places with greater poverty may not have the resources or want to reduce air pollution.

let's first explore the data set's variables and its potential relationship with 'value'.

```
#Plots the relationship between our predictors and PM2.5 concentration
dat0 |>
  select(value, CMAQ, aod, zcta_pop, log_pri_length_10000, log_nei_2008_pm25_sum_25000, popdens_zcta, po
  pivot_longer(-value, names_to = "name", values_to = "metric") |>
  ggplot(aes(metric, value)) +
  geom_point(alpha = 1/2) +
  geom_smooth(se = FALSE) +
  facet_wrap(vars(name), scales = 'free_x')
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Overall, there seems to be a slightly positive relationship between the different variables and air pollution. The population density however is harder to tell as there is funneling in the data.

Let's also check to correlations between our chosen predictors and PM2.5 concentration

```
# Calculates the correlations between PM2.5 and our different predictor variables
cor(dat0$CMAQ, dat0$value)
```

```
## [1] 0.4661511
```

```
cor(dat0$aod, dat0$value)
```

```
## [1] 0.3498192
```

```
cor(dat0$log_nei_2008_pm25_sum_25000, dat0$value)
```

```
## [1] 0.377129
```

```
cor(dat0$log_pri_length_10000, dat0$value)
```

```
## [1] 0.2178476
```

```
cor(dat0$popdens_zcta, dat0$value)
```

```
## [1] 0.133054
```

```
cor(dat0$pov, dat0$value)
```

```
## [1] 0.163784
```

```
cor(dat0$zcta_pop, dat0$value)
```

```
## [1] 0.1563638
```

Overall, the correlations for all of our predictor variables are relatively low with the highest correlation being from CMAQ.

Based on the scatter plots and correlation values, I expect the RMSE value of our models to be relatively high. Maybe around 2 or 3.

Wrangling

The data that is already loaded is already tidy so I will just do some minor changes I want to remove the clutter from the variables that I would not use and remove rows with NA values.

```
# Selects our variables and remove any potential NA values
dat <- dat0|>
  select(value, CMAQ, aod, zcta_pop, log_pri_length_10000, log_nei_2008_pm25_sum_25000, popdens_zcta, p
  na.omit()
```

No rows were removed from the na.omit() function. We are now ready to fit our models for linear regression, k-nearest neighbor, and random forest.

Results

Let's first split our data between training and testing

```
# Sets seed so potential randomized actions can be repeated
set.seed(322)

# Splits data into training and testing data
dat_split <- initial_split(dat)
# Gets the training data
dat_train <- training(dat_split)
```

Let's first fit a linear regression model

```r
# Create recipe for our model and normalizes variables
rec <- dat_train|>
  recipe(value ~ CMAQ + aod + zcta_pop + log_pri_length_10000 +
           log_nei_2008_pm25_sum_25000 + popdens_zcta + pov) |>
  step_normalize(-value)

# Sets linear regression model
model <- linear_reg()|>
  set_engine('lm')|>
  set_mode('regression')

wf <- workflow()|>
  add_recipe(rec)|>
  add_model(model)

# Cross validation using 10 folds
folds <- vfold_cv(dat_train, v = 10)

res <- fit_resamples(wf, resamples = folds,
                     metrics = metric_set(rmse))
res|>
  collect_metrics()
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard    2.06    10  0.0750 Preprocessor1_Model1
```

The root mean squared error is 2.063 for our linear regression model which is close to what I have initially predicted

Let's save this into a table

```r
# Creates empty 3x2 table
table <- matrix(nrow = 3, ncol = 2, byrow = T)
# Adds column and row names
colnames(table) <- c('RMSE', 'Std Error')
rownames(table) <- c('Linear Regression', 'K-nearest neighbors', 'Random Forest')
# Fills in values I collected
table[1,1] <- 2.06
table[1,2] <- 0.075
table
```

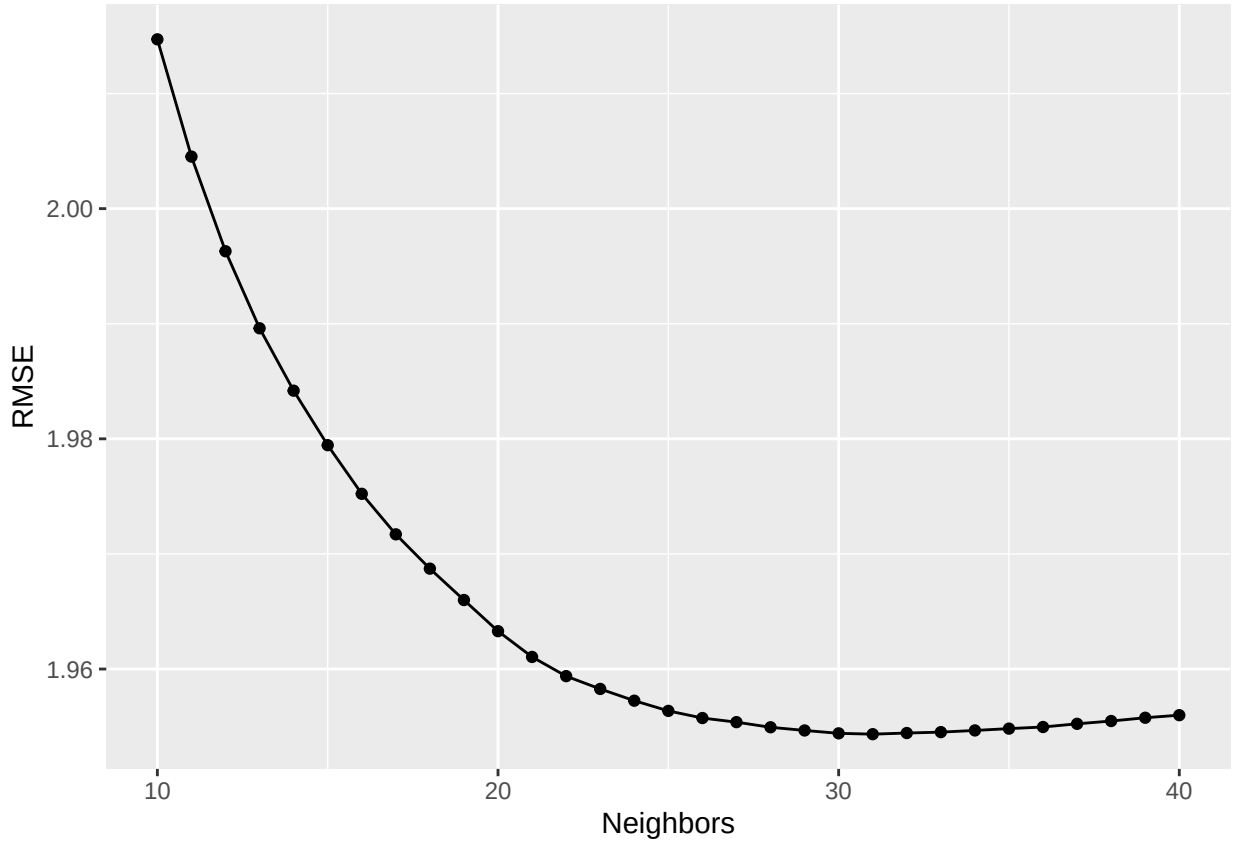
```
##                     RMSE Std Error
## Linear Regression   2.06     0.075
## K-nearest neighbors   NA        NA
## Random Forest         NA        NA
```

Let's move onto K-nearest neighbors. I am going to use the same recipe but the model will now use nearest_neighbor instead. I am also going to tune the neighbors parameter.

```r
# Model uses K-nn and will tune the neighbors parameter
model <- nearest_neighbor(neighbors = tune("k")) |>
  set_engine("kknn") |>
  set_mode("regression")

wf <- workflow()|>
  add_recipe(rec)|>
  add_model(model)

# Cross validation using 10 folds
folds <- vfold_cv(dat_train, v = 10)

res <- tune_grid(wf, resamples = folds,
                 grid = tibble(k = seq(10, 40, 1)),
                 metrics = metric_set(rmse))

# Finds which number of neighbors will result in the lowest RMSE value
res|>
  collect_metrics()|>
  arrange(mean)
```

```
## # A tibble: 31 x 7
##        k .metric .estimator  mean     n std_err .config
##    <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1    31 rmse    standard    1.95    10  0.0609 Preprocessor1_Model22
##  2    30 rmse    standard    1.95    10  0.0613 Preprocessor1_Model21
##  3    32 rmse    standard    1.95    10  0.0606 Preprocessor1_Model23
##  4    33 rmse    standard    1.95    10  0.0604 Preprocessor1_Model24
##  5    29 rmse    standard    1.95    10  0.0617 Preprocessor1_Model20
##  6    34 rmse    standard    1.95    10  0.0601 Preprocessor1_Model25
##  7    35 rmse    standard    1.95    10  0.0599 Preprocessor1_Model26
##  8    28 rmse    standard    1.95    10  0.0621 Preprocessor1_Model19
##  9    36 rmse    standard    1.95    10  0.0597 Preprocessor1_Model27
## 10    37 rmse    standard    1.96    10  0.0595 Preprocessor1_Model28
## # i 21 more rows
```

```r
# Plots RMSE values across k values
res|>
  collect_metrics()|>
  ggplot(aes(x = k, y = mean))+
  labs(x = 'Neighbors', y = 'RMSE')+
  geom_point() +
  geom_line()
```



The K-nearest neighbor model yielded a lower RMSE value than the linear regression model when k is set at 31. While the RMSE value is lower, it is still close to 2.

Let's now add the K-nn results onto our table

```
# Fills in values I collected
table[2,1] <- 1.95
table[2,2] <- 0.061
table
```

```
##                      RMSE Std Error
## Linear Regression    2.06     0.075
## K-nearest neighbors 1.95     0.061
## Random Forest          NA        NA
```

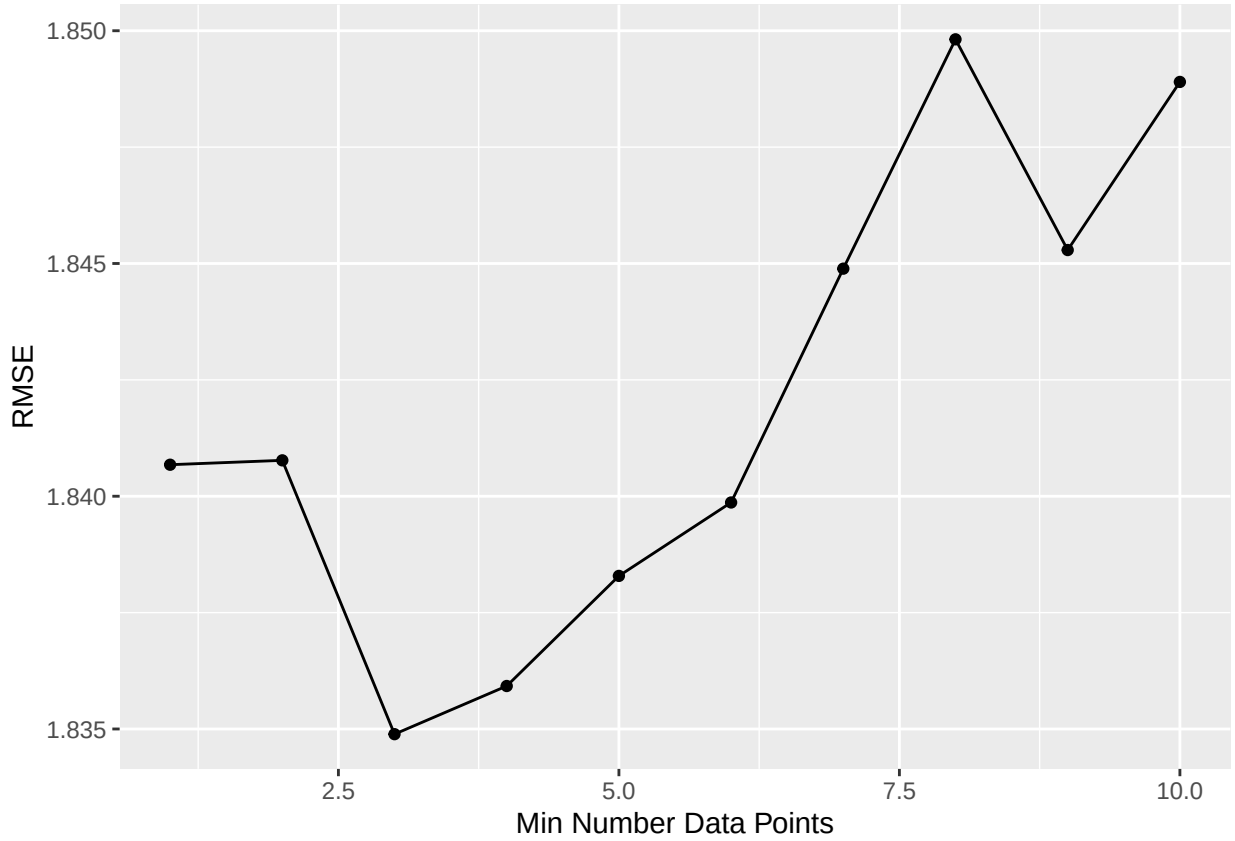
Now let's move onto our final model random forest.

```
# Model uses random forest and will tune the min_n parameter
model <- rand_forest(min_n  = tune("min")) |>
  set_engine("randomForest") |>
  set_mode("regression")

wf <- workflow()|>
  add_recipe(rec)|>
  add_model(model)

# Cross validation using 10 folds
```

```r
folds <- vfold_cv(dat_train, v = 10)

res <- tune_grid(wf, resamples = folds,
                 grid = tibble(min = seq(1, 10, 1)),
                 metrics = metric_set(rmse))

# Finds which number of neighbors will result in the lowest RMSE value
res|>
  collect_metrics()|>
  arrange(mean)
```

```
## # A tibble: 10 x 7
##      min .metric .estimator  mean     n std_err .config
##    <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     3 rmse    standard    1.83    10  0.0572 Preprocessor1_Model03
##  2     4 rmse    standard    1.84    10  0.0580 Preprocessor1_Model04
##  3     5 rmse    standard    1.84    10  0.0601 Preprocessor1_Model05
##  4     6 rmse    standard    1.84    10  0.0560 Preprocessor1_Model06
##  5     1 rmse    standard    1.84    10  0.0584 Preprocessor1_Model01
##  6     2 rmse    standard    1.84    10  0.0586 Preprocessor1_Model02
##  7     7 rmse    standard    1.84    10  0.0570 Preprocessor1_Model07
##  8     9 rmse    standard    1.85    10  0.0568 Preprocessor1_Model09
##  9    10 rmse    standard    1.85    10  0.0592 Preprocessor1_Model10
## 10     8 rmse    standard    1.85    10  0.0601 Preprocessor1_Model08
```

```r
# Plots RMSE values across k values
res|>
  collect_metrics()|>
  ggplot(aes(x = min, y = mean))+
  labs(x = 'Min Number Data Points', y = 'RMSE')+
  geom_point() +
  geom_line()
```



Random forest model yielded the lowest RMSE value so far with the min_n parameter set at 3. It is interesting to see that the plot for the RMSE values over min_n has an irregular shape.

Let's now add this final model onto our table

```
# Fills in values I collected
table[3,1] <- 1.83
table[3,2] <- 0.057
table
```

```
##                     RMSE Std Error
## Linear Regression   2.06     0.075
## K-nearest neighbors 1.95     0.061
## Random Forest       1.83     0.057
```

Random forest has the lowest RMSE value of 1.83 out of all the tested prediction models

I am also curios what happens when I remove CMAQ and AOD from the model

```
# Recipe without CMAQ and AOD
rec <- dat_train|>
  recipe(value ~ zcta_pop + log_pri_length_10000 +
           log_nei_2008_pm25_sum_25000 + popdens_zcta + pov) |>
  step_normalize(-value)

model <- rand_forest(min_n = 3) |>
  set_engine("randomForest") |>
```

```
  set_mode("regression")

wf <- workflow()|>
  add_recipe(rec)|>
  add_model(model)

folds <- vfold_cv(dat_train, v = 10)

res <- fit_resamples(wf, resamples = folds,
                     metrics = metric_set(rmse))
res|>
  collect_metrics()
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard    2.17    10  0.0647 Preprocessor1_Model1
```

Without CMAQ and AOD, the RMSE becomes higher.

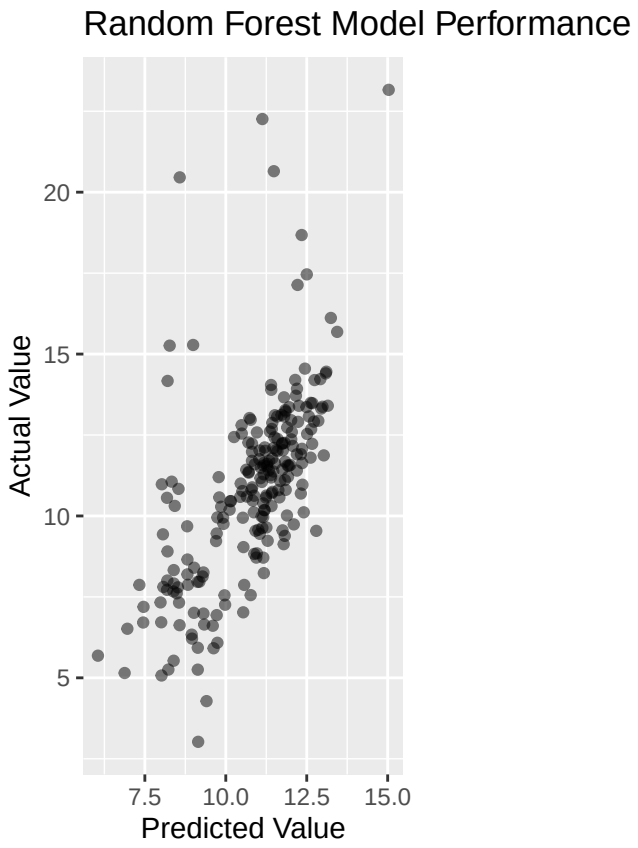
Let's now do a final test with the random forest model with CMAQ and AOD against the testing data.

```
# Original recipe
rec <- dat_train|>
  recipe(value ~ CMAQ + aod + zcta_pop + log_pri_length_10000 +
           log_nei_2008_pm25_sum_25000 + popdens_zcta + pov) |>
  step_normalize(-value)

# Using random forest model with trees set at
model <- rand_forest(min_n = 3) |>
  set_engine("randomForest") |>
  set_mode("regression")

wf <- workflow()|>
  add_recipe(rec)|>
  add_model(model)

## Plot the observed PM2.5 values vs. model predictions
res <- last_fit(wf, split = dat_split)
res |>
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       2.29  Preprocessor1_Model1
## 2 rsq     standard       0.407 Preprocessor1_Model1
```

```
dat_test <- rec |>
  prep() |>
  bake(new_data = testing(dat_split))
```

```
res %>%
  extract_fit_parsnip() %>%
  augment(new_data = dat_test) %>%
  select(value, .pred) %>%
  ggplot(aes(.pred, value)) +
  labs(y = 'Actual Value', x = 'Predicted Value',
       title = 'Random Forest Model Performance') +
  coord_fixed(ratio = 1) +
  geom_point(alpha = 1/2)
```



The random forest model when predicting the testing data has a RMSE value of 2.3 and a R-squared value of 0.4.

Let's also test the model against some select states

```
# Testing model against different states

res <- fit(wf, dat0[dat0$state == 'California',])
res
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
```

```
##
## * step_normalize()
##
## -- Model ---------------------------------------------------------------------
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, nodesize = min_rows(~3,      x))
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 9.723342
##                     % Var explained: 42.99
```

```
res <- fit(wf, dat0[dat0$state == 'Texas',])
res
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ---------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model ---------------------------------------------------------------------
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, nodesize = min_rows(~3,      x))
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 5.412444
##                     % Var explained: -21.45
```

```
res <- fit(wf, dat0[dat0$state == 'New York',])
res
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ---------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model ---------------------------------------------------------------------
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, nodesize = min_rows(~3,      x))
```

```
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1.047657
##                     % Var explained: 78.11
```

```
res <- fit(wf, dat0[dat0$state == 'Florida',])
res
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, nodesize = min_rows(~3,      x))
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1.135529
##                     % Var explained: -15.69
```

It is interesting to see that the state does affect the performance of the model.

Discussion

Primary Questions

1. My model seems to give okay predictions when actual PM2.5 concentration is relatively low however when PM2.5 concentrations are above around 13, the model does not do a particularly well job predicting the actual concentration. This can be seen when the actual value is above 15 and the model predicted those values to be lower than what they actually are. I think the cause for this may be some confounding variables in those areas with high PM2.5 concentrations that were not considered. Maybe those areas have other sources of air pollution that were not recorded in the data like planes. I believe the low PM2.5 concentrations are better predicted as those areas may have similar conditions between each other.

2. My model seems to perform better for states closer to the east coast as the RMSE value of the state of New York is 1 while the RMSE value for California is 10. I think this may be because of some other variables that are not included in this data set. I think there should be some other metric for different industries in the area as for example California has more international trade when compared to New York which means there is more large scale shipping in that area. Having some more metrics for different industries might help improve the model.

3. CMAQ and AOD seems to help the random forest model more accurately predict PM2.5 concentrations as RMSE increases by around 0.5 when those variables are not included.

4. I do not think the data will perform well in Alaska and Hawaii as those states' geography and economy are way different than the contiguous U.S. states. This means that their industries and local climate are different which could lead to having a number of confounding variables to deal with.

Reflection

Overall the project itself was somewhat challenging for me as I am not too familiar with the different functions associated with tidymodels so I had to refer back to the lecture code a number of times. I have also learned to refer back to the documentation and references for tidymodels as a lot of the time the documentation will have what I am looking for. For example I refered to the documentation on tidymodels for random forest where I learned exactly what min_n is.

For the model itself, it did better than I thought it did as when I was looking at the initial relationships and correlation, I did not have a great deal of confidence in the eventual model which is why I predicted the model to have a RMSE value of 2 or 3. With the model fitted to the training data, its RMSE value was a fair bit lower than what I predicted of around a 1.8. While the model RMSE is 2.3 when fitted on the testing data, it was still better than I expected as I was expecting a value of around 3.