SDS322ResearchProject2

Erik Mercado emm4376

```
library(tidyverse)
library(tidymodels)
library(dplyr)
library(ggplot2)
library(ranger)
library(usmap)
```

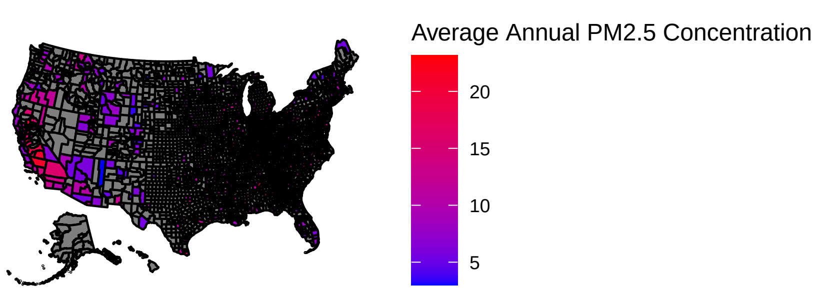
Introduction

In this report, I will be building several prediction models in order to predict ambient air pollution concentrations across the continental United States. The modeling approaches that I will use are Linear Regression, K-Nearest Neighbors, and Random Forest. The predictors were chosen based on which predictors I believed to be important to determining the outcome variable. I expect my RMSE to be less than 5, but firsts let's load the data.

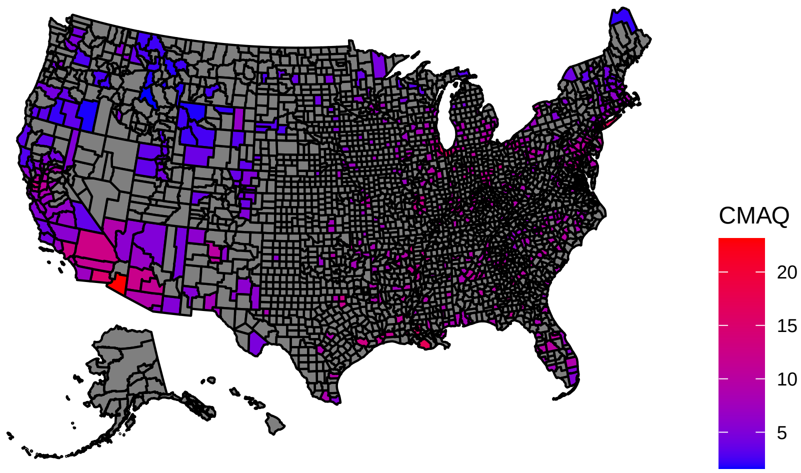
```
dat <- read_csv("https://github.com/rdpeng/stat322E_public/raw/main/data/pm25_data.csv.gz")
```

After loading in the data, I create a few heat maps of the US to see how some other variables stacked up against the **value** value in different regions.

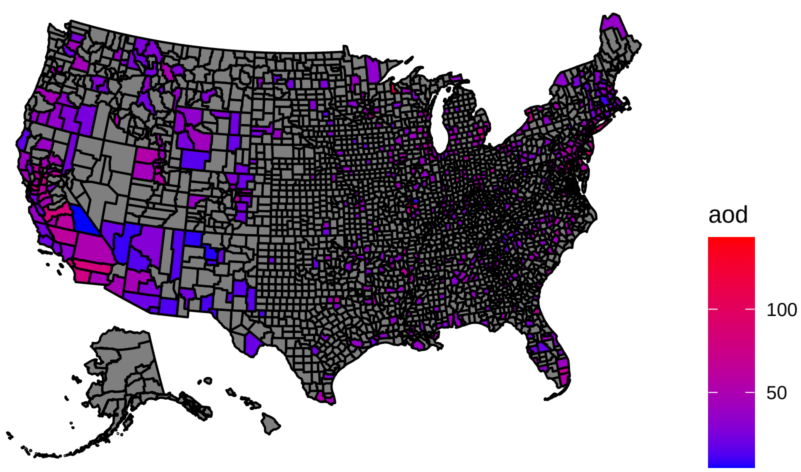
```
plot_usmap(data = dat, values = "value") +
  scale_fill_continuous(
    low = "blue", high = "red", name = "Average Annual PM2.5 Concentration",
    label = scales::comma) + theme(legend.position = "right")
```



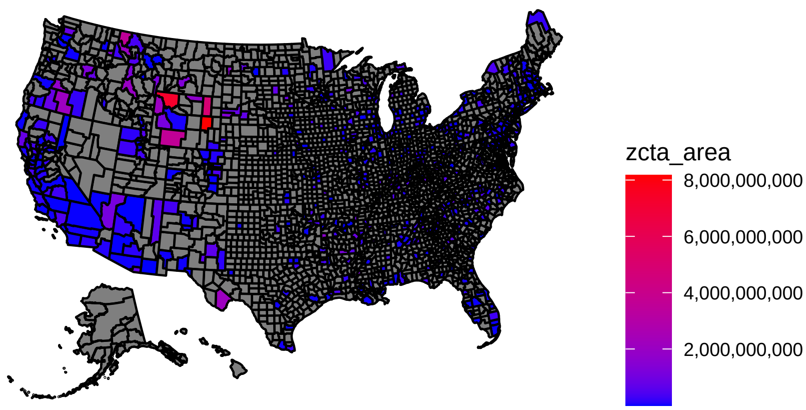
```r
plot_usmap(data = dat, values = "CMAQ") +
  scale_fill_continuous(
    low = "blue", high = "red", name = "CMAQ", label = scales::comma
  ) + theme(legend.position = "right")
```



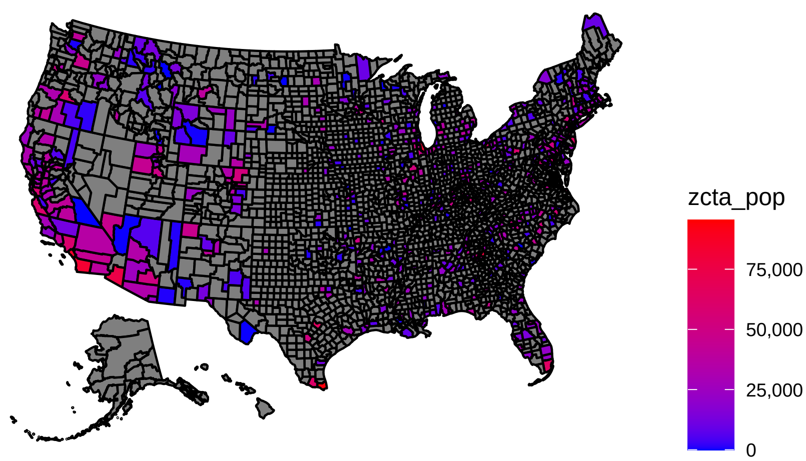

```r
plot_usmap(data = dat, values = "aod") +
  scale_fill_continuous(
    low = "blue", high = "red", name = "aod", label = scales::comma
  ) + theme(legend.position = "right")
```



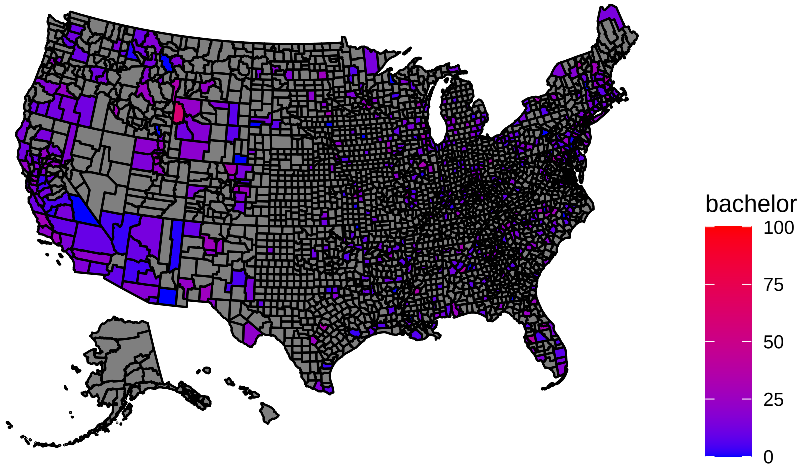

```r
plot_usmap(data = dat, values = "zcta_area") +
  scale_fill_continuous(
    low = "blue", high = "red", name = "zcta_area", label = scales::comma
  ) + theme(legend.position = "right")
```




```r
plot_usmap(data = dat, values = "zcta_pop") +
  scale_fill_continuous(
    low = "blue", high = "red", name = "zcta_pop", label = scales::comma
  ) + theme(legend.position = "right")
```




```r
plot_usmap(data = dat, values = "bachelor") +
  scale_fill_continuous(
    low = "blue", high = "red", name = "bachelor", label = scales::comma
  ) + theme(legend.position = "right")
```



Some of the relationships seen from this exploration is that in areas where **value** is high, the values of **CMAQ**, **aod**, and **zcta_pop** are also high. Other relationships for areas where **value** is high is that the **zcta_area** and **bachelor** values are lower. This is just some of the few exploratory analysis that could have been done for this data set. One thought that could be drawn from these relationships is that Annual Average PM2.5 Concentration is higher in areas that have a higher population, lower land area in square meters, and a lower percentage of people who have at least completed a bachelor's degree.

Wrangling

I then got rid of any NAs that were present in the dataset so that they wouldn't cause any trouble down the road.

```
dat <- dat |>
  filter(!is.na(value))
```

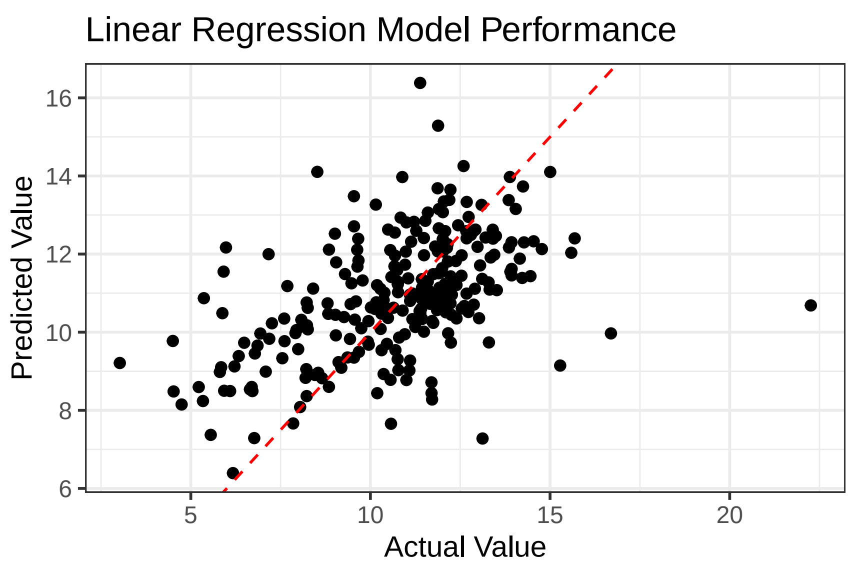
Results

I then split the data set into a training set and a testing set. I created the training and testing sets by randomly indexing a percentage of the original data set into one data frame and put all the observations that weren't indexed into another data set.

```
set.seed(123)
train_index <- sample(1:nrow(dat), size = 0.7 * nrow(dat), replace = FALSE)
# Create the training and testing data sets
train_data <- dat[train_index, ]
test_data <- dat[-train_index, ]
```

I then proceeded to build my models. The first one I went with was a Linear Regression Model. I regressed **value** across all of the numeric variables present in data set with one minor difference. In this first linear regression model, I excluded the predictor of **aod**.

```r
# set up model
lnreg_rec <- train_data |>
  select(-aod, -city, -county, -state) |>
    recipe(value ~ .)
lnreg_model <- linear_reg() %>%
    set_engine("lm") %>%
    set_mode("regression")
lnreg_wf <- workflow() %>%
    add_recipe(lnreg_rec) %>%
    add_model(lnreg_model)
lnreg_res <- fit(lnreg_wf, data = train_data)
# make predictions on the test set
test_preds <- predict(lnreg_res, new_data = test_data)
# combine predicted values and actual values into a data frame
pred_df <- data.frame(actual = test_data$value, predicted = test_preds)
# create a scatter plot
ggplot(pred_df, aes(x = actual, y = pred_df[,".pred"])) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(x = "Actual Value", y = "Predicted Value", title = "Linear Regression Model Performance") +
  theme_bw()
```




```r
# calculate prediction errors
test_data$prediction <- predict(lnreg_res, new_data = test_data)
test_data$error <- test_data$value - test_data$prediction
# find the locations with closest and furthest predictions
closest <- test_data[order(abs(test_data$error))[1:10], ]
furthest <- test_data[order(abs(test_data$error), decreasing = TRUE)[1:10], ]
closest |>
  select(state, city, county)
```

```
## # A tibble: 10 x 3
##    state     city       county
```

```
##    <chr>      <chr>      <chr>
##  1 Illinois   Zion       Lake
##  2 California Woodland   Yolo
##  3 Ohio       Toledo     Lucas
##  4 Montana    Hamilton   Ravalli
##  5 California Sacramento Sacramento
##  6 Missouri   Arnold     Jefferson
##  7 New Jersey Carlstadt  Bergen
##  8 Indiana    Gary       Lake
##  9 Nebraska   Bellevue   Sarpy
## 10 Illinois   Wood River Madison
```

```r
furthest |>
  select(state, city, county)
```

```
## # A tibble: 10 x 3
##    state      city           county
##    <chr>      <chr>          <chr>
##  1 California Clovis         Fresno
##  2 California Merced         Merced
##  3 New Mexico Albuquerque    Bernalillo
##  4 Arizona    Fort Defiance  Apache
##  5 California Quincy         Plumas
##  6 Oregon     Oakridge       Lane
##  7 New Mexico Albuquerque    Bernalillo
##  8 Louisiana  Not in a city  Terrebonne
##  9 Arizona    Tucson         Pima
## 10 Wyoming    Cheyenne       Laramie
```

```r
# cross validation
lnreg_res %>%
    extract_fit_engine() %>%
    summary()
```

```
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5316 -1.2327 -0.0066  1.0999 10.4564
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.125e+02  1.211e+02   2.579 0.010153 *
## id                         -2.748e-01  6.230e-01  -0.441 0.659364
## fips                        2.748e-01  6.230e-01   0.441 0.659363
## lat                        -1.149e-03  2.465e-02  -0.047 0.962858
## lon                        -8.441e-02  2.074e-02  -4.070 5.38e-05 ***
## CMAQ                        3.413e-01  4.423e-02   7.717 5.40e-14 ***
## zcta                       -3.697e-05  1.038e-05  -3.563 0.000398 ***
## zcta_area                  -2.849e-10  1.735e-10  -1.642 0.101131
## zcta_pop                    1.292e-05  5.590e-06   2.310 0.021221 *
```

```
## imp_a500                     -1.355e-02  2.189e-02  -0.619 0.536270
## imp_a1000                     1.875e-02  2.776e-02   0.675 0.499666
## imp_a5000                     1.130e-02  2.992e-02   0.378 0.705813
## imp_a10000                    6.885e-02  4.598e-02   1.498 0.134800
## imp_a15000                   -5.073e-02  3.380e-02  -1.501 0.133886
## county_area                  -1.209e-11  1.873e-11  -0.646 0.518838
## county_pop                   -3.507e-08  8.963e-08  -0.391 0.695764
## log_dist_to_prisec            6.215e-02  1.258e-01   0.494 0.621463
## log_pri_length_5000          -2.946e-02  1.873e-01  -0.157 0.875091
## log_pri_length_10000         -4.348e-02  4.321e-01  -0.101 0.919887
## log_pri_length_15000         -3.837e-01  4.663e-01  -0.823 0.410895
##  [ reached getOption("max.print") -- omitted 26 rows ]
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.155 on 568 degrees of freedom
## Multiple R-squared:  0.3756, Adjusted R-squared:  0.3272
## F-statistic: 7.764 on 44 and 568 DF,  p-value: < 2.2e-16
```

```r
lnreg_folds <- vfold_cv(test_data, v = 5)
lnreg_res <- fit_resamples(lnreg_wf, resamples = lnreg_folds)
# get prediction metrics
lnreg_res %>%
    collect_metrics()
```

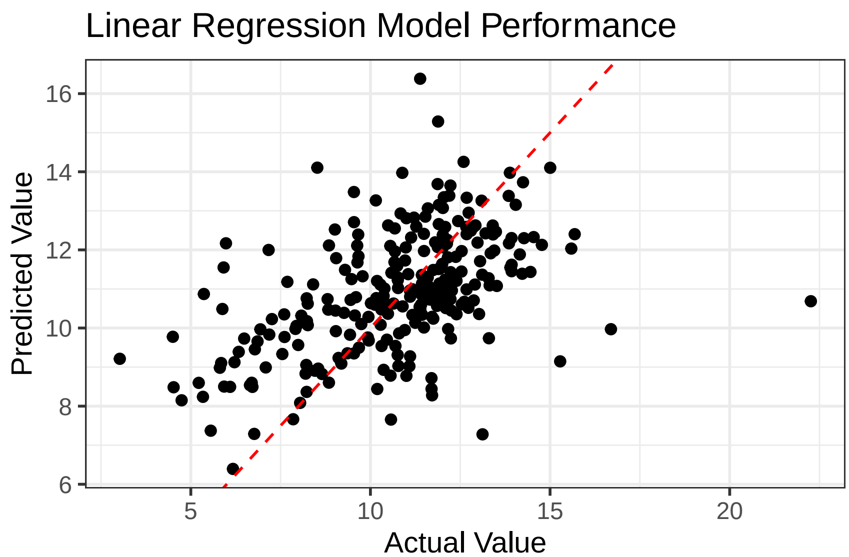
```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   2.31      5  0.143  Preprocessor1_Model1
## 2 rsq     standard   0.206     5  0.0373 Preprocessor1_Model1
```

The output from the first regression is a scatter plot showing the relationship between the actual **value** and the predicted **value**, two tables showing the locations who's predictions were the closet and the furthest, and the prediction metrics for the model. The RMSE for this model was 2.3521513.

With this next model, the set up is basically identical to the first one, but this time I am excluding **aod** and including **CMAQ**.

```r
# set up model
lnreg_rec2 <- train_data |>
  select(-CMAQ, -city, -county, -state)|>
    recipe(value ~ .)
lnreg_wf2 <- workflow() %>%
    add_recipe(lnreg_rec2) %>%
    add_model(lnreg_model)
lnreg_res2 <- fit(lnreg_wf2, data = train_data)
# Make predictions on the test set
test_preds2 <- predict(lnreg_res2, new_data = test_data)
# Combine predicted values and actual values into a data frame
pred_df2 <- data.frame(actual = test_data$value, predicted = test_preds)
# create a scatter plot
ggplot(pred_df2, aes(x = actual, y = pred_df2[,".pred"])) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
```

```
  labs(x = "Actual Value", y = "Predicted Value", title = "Linear Regression Model Performance") +
  theme_bw()
```




```
# calculate prediction errors
test_data$prediction2 <- predict(lnreg_res2, new_data = test_data)
test_data$error2 <- test_data$value - test_data$prediction2
# Find the locations with closest and furthest predictions
closest2 <- test_data[order(abs(test_data$error2))[1:10], ]
furthest2 <- test_data[order(abs(test_data$error2), decreasing = TRUE)[1:10], ]
closest2 |>
  select(state, city, county)
```

```
## # A tibble: 10 x 3
##    state          city                   county
##    <chr>          <chr>                  <chr>
##  1 California     San Andreas            Calaveras
##  2 Montana        Hamilton               Ravalli
##  3 Michigan       Flint                  Genesee
##  4 Texas          Texarkana              Bowie
##  5 New Jersey     Hopewell (Township of) Mercer
##  6 California     Calexico               Imperial
##  7 Missouri       Ladue                  Saint Louis
##  8 Iowa           Clinton                Clinton
##  9 Ohio           Toledo                 Lucas
## 10 South Carolina Not in a city          Chesterfield
```

```
furthest2 |>
  select(state, city, county)
```

```
## # A tibble: 10 x 3
##    state      city             county
##    <chr>      <chr>            <chr>
```

```
##   1 California Clovis          Fresno
##   2 Arizona    Fort Defiance   Apache
##   3 California Merced          Merced
##   4 Wyoming    Cheyenne        Laramie
##   5 Oregon     Oakridge        Lane
##   6 Louisiana  Baton Rouge     East Baton Rouge
##   7 California Salinas         Monterey
##   8 Arizona    Tucson          Pima
##   9 California Quincy          Plumas
##  10 Colorado   Roxborough Park Douglas
```

```
# cross validation
lnreg_res2 %>%
    extract_fit_engine() %>%
    summary()
```

```
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9792 -1.3080  0.0335  1.1573  9.9487
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.413e+02  1.236e+02   1.953  0.05134 .
## id                         -6.552e-01  6.313e-01  -1.038  0.29978
## fips                        6.552e-01  6.313e-01   1.038  0.29979
## lat                        -5.935e-02  2.478e-02  -2.396  0.01692 *
## lon                        -3.239e-02  1.975e-02  -1.640  0.10149
## zcta                       -1.416e-05  1.009e-05  -1.403  0.16118
## zcta_area                  -2.989e-10  1.764e-10  -1.694  0.09077 .
## zcta_pop                    1.280e-05  5.696e-06   2.247  0.02504 *
## imp_a500                   -1.009e-02  2.226e-02  -0.453  0.65062
## imp_a1000                   1.463e-02  2.824e-02   0.518  0.60454
## imp_a5000                   1.422e-02  3.043e-02   0.467  0.64036
## imp_a10000                  6.062e-02  4.677e-02   1.296  0.19543
## imp_a15000                 -7.229e-02  3.451e-02  -2.095  0.03663 *
## county_area                 7.591e-12  1.914e-11   0.397  0.69186
## county_pop                 -3.161e-08  9.117e-08  -0.347  0.72891
## log_dist_to_prisec          3.698e-02  1.278e-01   0.289  0.77244
## log_pri_length_5000        -6.759e-02  1.905e-01  -0.355  0.72288
## log_pri_length_10000       -1.678e-01  4.390e-01  -0.382  0.70243
## log_pri_length_15000       -2.376e-01  4.732e-01  -0.502  0.61576
## log_pri_length_25000        2.443e-01  2.937e-01   0.832  0.40588
##  [ reached getOption("max.print") -- omitted 26 rows ]
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.192 on 568 degrees of freedom
## Multiple R-squared:  0.3541, Adjusted R-squared:  0.3041
## F-statistic: 7.077 on 44 and 568 DF,  p-value: < 2.2e-16
```

```
lnreg_res2 <- fit_resamples(lnreg_wf2, resamples = lnreg_folds)
# get prediction metrics
lnreg_res2 %>%
    collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   2.30      5  0.117  Preprocessor1_Model1
## 2 rsq     standard   0.208     5  0.0493 Preprocessor1_Model1
```

Just like with the first regression, the output from the second regression includes a scatter plot showing the relationship between the actual **value** and the predicted **value**, two tables showing the locations who's predictions were the closet and the furthest, and the prediction metrics for the model. The RMSE for this model was 2.3678149.

Next, I developed my k-NN models. The first one includes **CMAQ** and excludes **aod**. The way I did this was by setting the model to tune to the most optimal number of neighbors.

```
# set up model
knn_rec <- train_data %>%
  select(-aod, -city, -county, -state) |>
    recipe(value ~ .)
knn_model <- nearest_neighbor(neighbors = tune("k")) %>%
    set_engine("kknn") %>%
    set_mode("regression")
knn_wf <- workflow() %>%
    add_model(knn_model) %>%
    add_recipe(knn_rec)
# cross validation
knn_folds <- vfold_cv(test_data, v = 5)
knn_res <- tune_grid(knn_wf, resamples = knn_folds)
# get prediction metrics
knn_res %>%
    show_best(metric = "rmse")
```

```
## # A tibble: 5 x 7
##       k .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    14 rmse    standard    2.19     5   0.178 Preprocessor1_Model9
## 2    12 rmse    standard    2.21     5   0.177 Preprocessor1_Model8
## 3    10 rmse    standard    2.24     5   0.176 Preprocessor1_Model7
## 4     9 rmse    standard    2.25     5   0.176 Preprocessor1_Model6
## 5     7 rmse    standard    2.30     5   0.181 Preprocessor1_Model5
```

```
knn_res %>%
    show_best(metric = "rsq")
```

```
## # A tibble: 5 x 7
##       k .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    14 rsq     standard   0.243     5  0.0432 Preprocessor1_Model9
```

```
## 2    12 rsq     standard   0.231     5  0.0420 Preprocessor1_Model8
## 3    10 rsq     standard   0.217     5  0.0419 Preprocessor1_Model7
## 4     9 rsq     standard   0.211     5  0.0422 Preprocessor1_Model6
## 5     7 rsq     standard   0.193     5  0.0422 Preprocessor1_Model5
```

After running the model, the most optimal number of neighbors was 15 with an RMSE of 2.170212.

I then ran another k-NN model that included **aod** and excluded **CMAQ**. The overall structure is the same as the the first k-NN model.

```
# set up model
knn_rec2 <- train_data %>%
  select(-CMAQ, -city, -county, -state) |>
    recipe(value ~ .)
knn_wf2 <- workflow() %>%
    add_model(knn_model) %>%
    add_recipe(knn_rec2)
# cross validation
knn_res2 <- tune_grid(knn_wf, resamples = knn_folds)
# get prediction metrics
knn_res2 %>%
    show_best(metric = "rmse")
```

```
## # A tibble: 5 x 7
##       k .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    14 rmse    standard    2.19     5   0.178 Preprocessor1_Model10
## 2    13 rmse    standard    2.20     5   0.178 Preprocessor1_Model09
## 3    12 rmse    standard    2.21     5   0.177 Preprocessor1_Model08
## 4    11 rmse    standard    2.22     5   0.176 Preprocessor1_Model07
## 5     8 rmse    standard    2.27     5   0.177 Preprocessor1_Model06
```

```
knn_res2 %>%
    show_best(metric = "rsq")
```

```
## # A tibble: 5 x 7
##       k .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    14 rsq     standard   0.243     5  0.0432 Preprocessor1_Model10
## 2    13 rsq     standard   0.237     5  0.0427 Preprocessor1_Model09
## 3    12 rsq     standard   0.231     5  0.0420 Preprocessor1_Model08
## 4    11 rsq     standard   0.224     5  0.0415 Preprocessor1_Model07
## 5     8 rsq     standard   0.203     5  0.0423 Preprocessor1_Model06
```

The optimal number of neighbors in this model is 14 with a RMSE of 2.174172.

The final type of model that I ran was a random forest model. The first random forest includes **CMAQ** and excludes **aod**. For the model, I set it to tune to the lowest mtry and the minimum n.

```
# set up model
rf_rec <- train_data %>%
  select(-aod, -city, -county, -state) |>
    recipe(value ~ .)
```

```r
rf_model <- rand_forest(mtry = tune("mtry"),
                     min_n = tune("min_n")) %>%
    set_engine("ranger") %>%
    set_mode("regression")
rf_wf <- workflow() %>%
    add_recipe(rf_rec) %>%
    add_model(rf_model)
# cross validation
rf_folds <- vfold_cv(test_data, v = 5)
rf_res <- tune_grid(rf_wf, resamples = rf_folds,
                 grid = expand.grid(mtry = c(1, 2, 5),
                                    min_n = c(3, 5)))
# get prediction metrics
rf_res %>%
    show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     5 rmse    standard    1.93     5   0.170 Preprocessor1_Model6
## 2     5     3 rmse    standard    1.95     5   0.170 Preprocessor1_Model3
## 3     2     5 rmse    standard    2.00     5   0.173 Preprocessor1_Model5
## 4     2     3 rmse    standard    2.01     5   0.176 Preprocessor1_Model2
## 5     1     5 rmse    standard    2.07     5   0.174 Preprocessor1_Model4
```

```r
rf_res %>%
    show_best(metric = "rsq")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     5 rsq     standard   0.396     5  0.0622 Preprocessor1_Model6
## 2     5     3 rsq     standard   0.387     5  0.0620 Preprocessor1_Model3
## 3     2     5 rsq     standard   0.367     5  0.0713 Preprocessor1_Model5
## 4     2     3 rsq     standard   0.356     5  0.0679 Preprocessor1_Model2
## 5     1     5 rsq     standard   0.329     5  0.0757 Preprocessor1_Model4
```

After running the model, I find that the best tree has a mtry of 5, a minimum n of 5, and a RMSE of 1.915338.

For the second random forest, I structured it the same as the first random forest, but this time I included **aod** and excluded **CMAQ** in the model.

```r
# set up model
rf_rec2 <- train_data %>%
  select(-CMAQ, -city, -county, -state) |>
    recipe(value ~ .)
rf_wf2 <- workflow() %>%
    add_recipe(rf_rec2) %>%
    add_model(rf_model)
rf_res2 <- tune_grid(rf_wf2, resamples = rf_folds,
                 grid = expand.grid(mtry = c(1, 2, 5),
```

```r
                                    min_n = c(3, 5)))
# get prediction metrics
rf_res2 %>%
    show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     5 rmse    standard    1.98     5   0.157 Preprocessor1_Model6
## 2     5     3 rmse    standard    1.99     5   0.145 Preprocessor1_Model3
## 3     2     5 rmse    standard    2.03     5   0.165 Preprocessor1_Model5
## 4     2     3 rmse    standard    2.04     5   0.164 Preprocessor1_Model2
## 5     1     5 rmse    standard    2.10     5   0.165 Preprocessor1_Model4
```

```r
rf_res2 %>%
    show_best(metric = "rsq")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     5 rsq     standard   0.372     5  0.0619 Preprocessor1_Model6
## 2     5     3 rsq     standard   0.363     5  0.0629 Preprocessor1_Model3
## 3     2     5 rsq     standard   0.343     5  0.0690 Preprocessor1_Model5
## 4     2     3 rsq     standard   0.333     5  0.0767 Preprocessor1_Model2
## 5     1     5 rsq     standard   0.301     5  0.0684 Preprocessor1_Model4
```

The most optimal tree for this model has a mtry of 5, a minimum n of 3, and a RMSE of 1.937094.

After running all of my models, I collected all the prediction metrics and combined them into a table.

```r
# combine all prediction metrics into a table
all_metrics <- bind_rows(
  lnreg_res %>%
    collect_metrics() |>
    mutate(model_type = "Linear Regression w/o aod"),
  lnreg_res2 %>%
    collect_metrics() |>
    mutate(model_type = "Linear Regression w/o CMAQ"),
  knn_res %>%
    show_best(metric = "rmse") |>
    mutate(model_type = "k-NN w/o aod"),
knn_res %>%
    show_best(metric = "rsq") |>
    mutate(model_type = "k-NN w/o aod"),
knn_res2 %>%
    show_best(metric = "rmse") |>
    mutate(model_type = "k-NN w/o CMAQ"),
knn_res2 %>%
    show_best(metric = "rsq") |>
    mutate(model_type = "k-NN w/o CMAQ"),
rf_res %>%
    show_best(metric = "rmse") |>
```

```
  mutate(model_type = "Random Forest w/o aod"),
rf_res %>%
    show_best(metric = "rsq") |>
  mutate(model_type = "Random Forest w/o aod"),
rf_res2 %>%
    show_best(metric = "rmse") |>
  mutate(model_type = "Random Forest w/o CMAQ"),
rf_res2 %>%
    show_best(metric = "rsq") |>
  mutate(model_type = "Random Forest w/o CMAQ")
)
all_metrics
```

```
## # A tibble: 44 x 10
##    .metric .estimator  mean     n std_err .config      model~1     k  mtry min_n
##    <chr>   <chr>      <dbl> <int>   <dbl> <chr>        <chr>   <int> <dbl> <dbl>
##  1 rmse    standard   2.31      5  0.143  Preprocesso~ Linear~    NA    NA    NA
##  2 rsq     standard   0.206     5  0.0373 Preprocesso~ Linear~    NA    NA    NA
##  3 rmse    standard   2.30      5  0.117  Preprocesso~ Linear~    NA    NA    NA
##  4 rsq     standard   0.208     5  0.0493 Preprocesso~ Linear~    NA    NA    NA
##  5 rmse    standard   2.19      5  0.178  Preprocesso~ k-NN w~    14    NA    NA
##  6 rmse    standard   2.21      5  0.177  Preprocesso~ k-NN w~    12    NA    NA
##  7 rmse    standard   2.24      5  0.176  Preprocesso~ k-NN w~    10    NA    NA
##  8 rmse    standard   2.25      5  0.176  Preprocesso~ k-NN w~     9    NA    NA
##  9 rmse    standard   2.30      5  0.181  Preprocesso~ k-NN w~     7    NA    NA
## 10 rsq     standard   0.243     5  0.0432 Preprocesso~ k-NN w~    14    NA    NA
## # ... with 34 more rows, and abbreviated variable name 1: model_type
```

Discussion

Primary Questions

For my *best model*, I will be using the Linear Regression model that includes **CMAQ** and excludes **aod** because it was the model I was able to get the most information from and had the lowest RMSE of the ones that I could get info from.

1. Based on the the test set performance, the three locations that were the closest to their observed values were Lake County in Zion, Illinois, Yolo County in Woodland, California, and Lucas County in Toledo, Ohio. I believe these locations did so well because their variables were probably the closest to the true effect of what regression assumed the effects of the variables were. The three locations that were furthest from their predicted values were Fresno County in Colvis, California, Merced County in in Merced, California, and Bernalillo County in Albuquerque, New Mexico. These locations probably did the worst because they are most likely outliers.

2. The variables that have the strongest predictive power in this particular model are **lon**, **CMAQ**, and **zcta**. This means that areas that have a higher CMAQ score likely have a higher PM2.5 concentration. Also, locations at a lower longitude tend to have a higher PM2.5 concentration according to the **lon** coefficient. Another variable that might be able to help with my predictions could be the number of cars within each county, because they are a big cause of air pollution.

3. To be more cost efficient, air pollution monitoring approaches should focus on using numerical models like CMAQ because for every model pair, the model that included **CMAQ** did better than the models

that included **aod**. The models that included **CMAQ** had lower RMSE scores and higher R-squared scores.

4. I do not think that my model will perform very well on those two because the infrastructure and just overall climate of those two states is so different from the rest of the United States, so the beta coefficients for the continental United States won't be very accurate on their parameters.

Overall, I did find this project to be challenging but I also gained a lot of knowledge while doing it. The challenging part was trying to get the predictions for the k-NN and random forest models. I would've given myself more time to work on this project if I were to do it again as I wasn't able to figure everything out in the time that I allotted myself. I learned much more about building predictions models and also how to build a heat map for the United States.

I believe that my model preformed pretty well and it met my expectation that I set at the beginning of the report.

The resources that I used to help me were the lecture code (more specifically the ones about the tidymodels as those help me get through a majority of the project) and the following link https://cran.r-project.org/web/packages/usmap/vignettes/mapping.html to help me with the US mapping.