Introduction

- 876 rows that are each of a different county in the US.
- Models: linear regression, k-nearest neighbors, random forest, decision trees.

```
#Load libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── 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 becom
e errors
```

```
library(tidymodels)
```

```
## ── Attaching packages ────────────────────────────────────── tidymodels 1.2.0 ──
## ✔ broom        1.0.6     ✔ rsample      1.2.1
## ✔ dials        1.3.0     ✔ tune         1.2.1
## ✔ infer        1.0.7     ✔ workflows    1.1.4
## ✔ modeldata    1.4.0     ✔ workflowsets 1.1.0
## ✔ parsnip      1.2.1     ✔ yardstick    1.3.1
## ✔ recipes      1.1.0
## ── Conflicts ───────────────────────────────────────── tidymodels_conflicts() ──
## ✖ scales::discard() masks purrr::discard()
## ✖ dplyr::filter()   masks stats::filter()
## ✖ recipes::fixed()  masks stringr::fixed()
## ✖ dplyr::lag()      masks stats::lag()
## ✖ yardstick::spec() masks readr::spec()
## ✖ recipes::step()   masks stats::step()
## • Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(kknn)
library(randomForest)
```

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

```
library(rpart)
```

```
##
## Attaching package: 'rpart'
##
## The following object is masked from 'package:dials':
##
##     prune
```

```
#Load data
dat <- read_csv("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.
```

```
# Get columns names with correlation > 0.25

# Calculate the correlation coefficients
cor_vec <- sapply(select(dat, where(is.numeric)), function(x) cor(x, dat$value))

#Get the names of the columns with correlation higher than 0.25.
high_cor_names <- names(cor_vec)[abs(cor_vec) > 0.25]

high_cor_names
```

```
##  [1] "value"                       "CMAQ"
##  [3] "zcta_area"                   "imp_a500"
##  [5] "imp_a1000"                   "imp_a5000"
##  [7] "imp_a10000"                  "imp_a15000"
##  [9] "log_pri_length_25000"        "log_prisec_length_5000"
## [11] "log_prisec_length_10000"     "log_prisec_length_15000"
## [13] "log_prisec_length_25000"     "log_nei_2008_pm25_sum_10000"
## [15] "log_nei_2008_pm25_sum_15000" "log_nei_2008_pm25_sum_25000"
## [17] "log_nei_2008_pm10_sum_10000" "log_nei_2008_pm10_sum_15000"
## [19] "log_nei_2008_pm10_sum_25000" "urc2013"
## [21] "urc2006"                     "aod"
```

Target: to get an RMSE of 2 or less.

Data Wrangling

```
# Clean the dataframe
dat_filtered <- dat |>
  select(value,CMAQ,zcta_area,imp_a500,imp_a1000,imp_a5000,imp_a10000,imp_a15000,log_pri_length_
25000,log_prisec_length_5000,log_prisec_length_10000,log_prisec_length_15000,log_prisec_length_2
5000,log_nei_2008_pm25_sum_10000,log_nei_2008_pm25_sum_15000,log_nei_2008_pm25_sum_25000,log_nei
_2008_pm10_sum_10000,log_nei_2008_pm10_sum_15000,log_nei_2008_pm10_sum_25000,urc2013,urc2006,ao
d,lat,lon,state)

# Split data into training and testing sets
set.seed(322)
dat_split <- initial_split(dat_filtered, prop = 0.8)
train <- training(dat_split)
test <- testing(dat_split)
```

Results - modeling

LINEAR REGRESSION ANALYSIS

```
#Regression
fit <- lm(formula = value ~ CMAQ + zcta_area + imp_a500 + imp_a1000 + imp_a5000 + imp_a10000 + i
mp_a15000 + log_pri_length_25000 + log_prisec_length_5000 + log_prisec_length_10000 + log_prisec
_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_10000 + log_nei_2008_pm25_sum_15
000 + log_nei_2008_pm25_sum_25000 + log_nei_2008_pm10_sum_10000 + log_nei_2008_pm10_sum_15000 +
log_nei_2008_pm10_sum_25000 + urc2013 + urc2006 + aod, data = train)
summary(fit)
```

```
## 
## Call:
## lm(formula = value ~ CMAQ + zcta_area + imp_a500 + imp_a1000 + 
##     imp_a5000 + imp_a10000 + imp_a15000 + log_pri_length_25000 + 
##     log_prisec_length_5000 + log_prisec_length_10000 + log_prisec_length_15000 + 
##     log_prisec_length_25000 + log_nei_2008_pm25_sum_10000 + log_nei_2008_pm25_sum_15000 + 
##     log_nei_2008_pm25_sum_25000 + log_nei_2008_pm10_sum_10000 + 
##     log_nei_2008_pm10_sum_15000 + log_nei_2008_pm10_sum_25000 + 
##     urc2013 + urc2006 + aod, data = train)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -5.6724 -1.1609 -0.0789  1.0590  9.7750 
## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                 -6.148e-02  2.272e+00  -0.027 0.978424    
## CMAQ                         2.746e-01  3.264e-02   8.413 2.37e-16 ***
## zcta_area                   -2.404e-10  1.524e-10  -1.577 0.115202    
## imp_a500                    -9.404e-03  1.764e-02  -0.533 0.594092    
## imp_a1000                    1.887e-02  2.260e-02   0.835 0.404176    
## imp_a5000                   -9.410e-03  2.514e-02  -0.374 0.708249    
## imp_a10000                   6.920e-02  4.047e-02   1.710 0.087783 .  
## imp_a15000                  -5.787e-02  2.874e-02  -2.014 0.044429 *  
## log_pri_length_25000        -2.284e-01  1.203e-01  -1.899 0.057951 .  
## log_prisec_length_5000       1.196e-02  2.528e-01   0.047 0.962271    
## log_prisec_length_10000      1.521e-01  5.995e-01   0.254 0.799833    
## log_prisec_length_15000     -4.345e-01  6.863e-01  -0.633 0.526828    
## log_prisec_length_25000      9.235e-01  4.081e-01   2.263 0.023933 *  
## log_nei_2008_pm25_sum_10000 -1.089e-01  3.170e-01  -0.344 0.731187    
## log_nei_2008_pm25_sum_15000 -4.924e-01  4.128e-01  -1.193 0.233287    
## log_nei_2008_pm25_sum_25000  1.343e+00  3.827e-01   3.510 0.000477 ***
## log_nei_2008_pm10_sum_10000  2.903e-01  3.159e-01   0.919 0.358395    
## log_nei_2008_pm10_sum_15000  4.207e-01  4.246e-01   0.991 0.322226    
## log_nei_2008_pm10_sum_25000 -1.330e+00  3.912e-01  -3.400 0.000715 ***
## urc2013                      1.392e-01  2.509e-01   0.555 0.579193    
## urc2006                      9.897e-02  2.530e-01   0.391 0.695787    
## aod                          2.197e-02  4.623e-03   4.752 2.45e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2 on 678 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:   0.34 
## F-statistic: 18.14 on 21 and 678 DF,  p-value: < 2.2e-16
## predict the values and calculate residsuals
values <- predict(fit, newdata = test)

# Calculate the residuals
residuals <- test$value - values

# Get RMSE
lin_RMSE <- sqrt(mean(residuals^2))
lin_RMSE
```

```
## [1] 2.641683
```

RMSE: 2.641683

K-NEAREST NEIGHBORS ANALYSIS

```
# Define the KNN model
knn_spec <- nearest_neighbor(neighbors = 9, weight_func = "rectangular", dist_power = 2) |>
  set_engine("kknn") |>
  set_mode("regression")

# Create a recipe for preprocessing
recipe <- recipe(value ~ CMAQ + zcta_area + imp_a500 + imp_a1000 + imp_a5000 + imp_a10000 + imp_a15000 + log_pri_length_25000 + log_prisec_length_5000 + log_prisec_length_10000 + log_prisec_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_10000 + log_nei_2008_pm25_sum_15000 + log_nei_2008_pm25_sum_25000 + log_nei_2008_pm10_sum_10000 + log_nei_2008_pm10_sum_15000 + log_nei_2008_pm10_sum_25000 + urc2013 + urc2006 + aod, data = train)

# Define the workflow
workflow <- workflow() |>
  add_model(knn_spec) |>
  add_recipe(recipe)

# Fit the model
fit_knn <- workflow |> fit(data = train)

# Predict on the test set and calculate residuals
predictions <- predict(fit_knn, new_data = test) |>
  bind_cols(test) |>
  mutate(resid = value - .pred)

# Calculate RMSE
k_rmse <- sqrt(mean(predictions$resid^2))
k_rmse
```

```
## [1] 2.430612
```

RMSE: 2.430612

DECISION TREES ANALYSIS

```
# Define the Decision Tree model
dt_spec <- decision_tree(tree_depth = 10, min_n = 5) |>
  set_engine("rpart") |>
  set_mode("regression")

# Create a recipe for preprocessing
recipe <- recipe(value ~ CMAQ + zcta_area + imp_a500 + imp_a1000 + imp_a5000 + imp_a10000 + imp_
a15000 + log_pri_length_25000 + log_prisec_length_5000 + log_prisec_length_10000 + log_prisec_le
ngth_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_10000 + log_nei_2008_pm25_sum_15000
+ log_nei_2008_pm25_sum_25000 + log_nei_2008_pm10_sum_10000 + log_nei_2008_pm10_sum_15000 + log_
nei_2008_pm10_sum_25000 + urc2013 + urc2006 + aod, data = train)

# Define the workflow
workflow <- workflow() |>
  add_model(dt_spec) |>
  add_recipe(recipe)

# Fit the model
fit_dt <- workflow |> fit(data = train)

# Predict on the test set and calculate residuals
predictions <- predict(fit_dt, new_data = test) |>
  bind_cols(test) |>
  mutate(resid = value - .pred)

# Calculate RMSE
dt_rmse <- sqrt(mean(predictions$resid^2))
dt_rmse
```

```
## [1] 2.780553
```

RMSE: 2.780553

RANDOM FOREST ANALYSIS

```
set.seed(322)

# Define the Random Forest model
rf_spec <- rand_forest(trees = 150, mode = "regression") |>
  set_engine("randomForest") |>
  set_mode("regression")

# Recipe for preprocessing
recipe <- recipe(value ~ CMAQ + zcta_area + imp_a500 + imp_a1000 + imp_a5000 + imp_a10000 + imp_a15000 + log_pri_length_25000 + log_prisec_length_5000 + log_prisec_length_10000 + log_prisec_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_10000 + log_nei_2008_pm25_sum_15000 + log_nei_2008_pm25_sum_25000 + log_nei_2008_pm10_sum_10000 + log_nei_2008_pm10_sum_15000 + log_nei_2008_pm10_sum_25000 + urc2013 + urc2006 + aod, data = train)

# Define the workflow
workflow <- workflow() |>
  add_model(rf_spec) |>
  add_recipe(recipe)

# Fit the model
fit_rf <- workflow |> fit(data = train)

# Predict on the test set and calculate residuals
predictions <- predict(fit_rf, new_data = test) |>
  bind_cols(test) |>
  mutate(resid = value - .pred)


# Calculate RMSE
rf_rmse <- sqrt(mean(predictions$resid^2))
rf_rmse
```

```
## [1] 2.375692
```

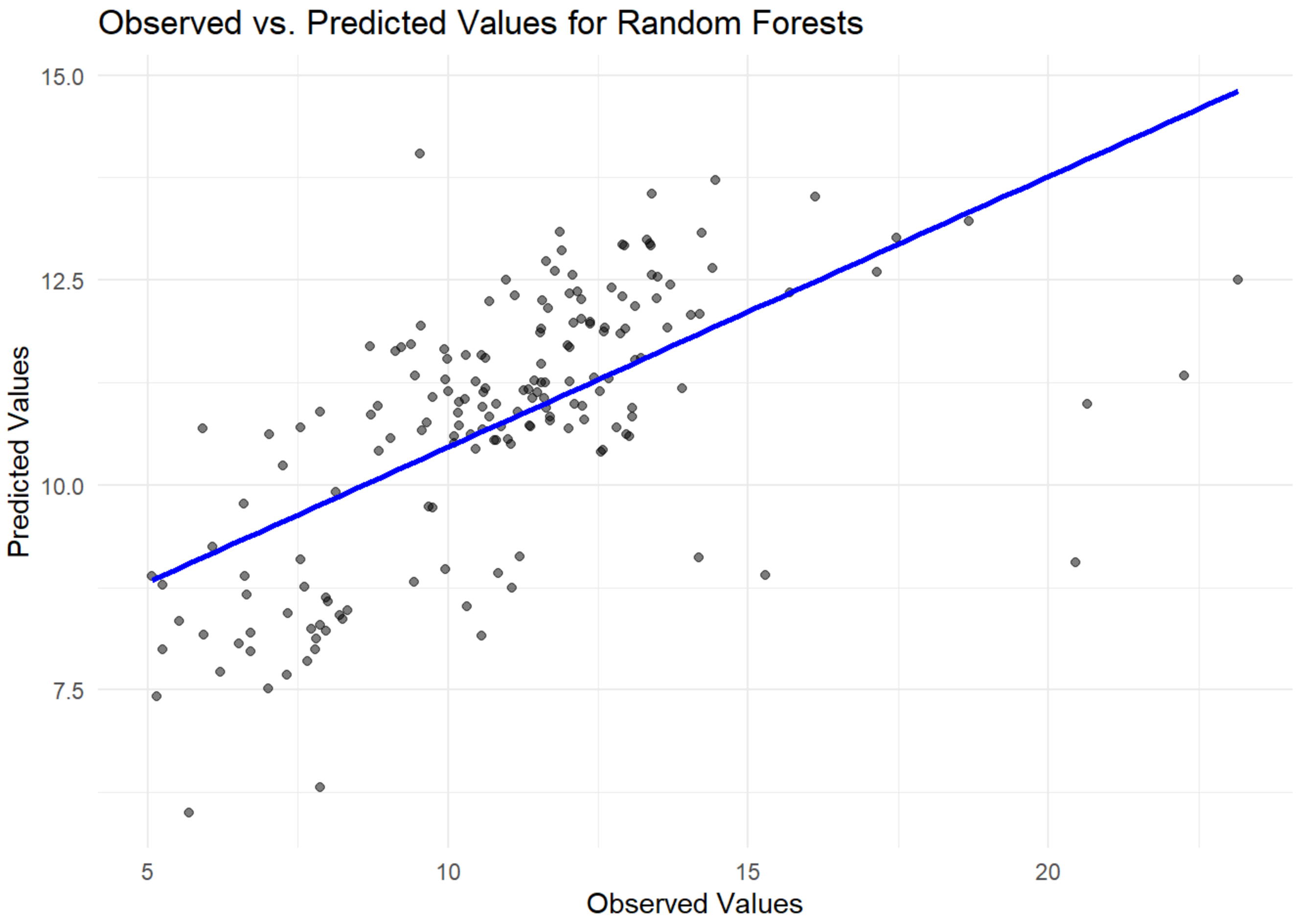
RMSE: 2.375692

Results - Best model

```
## Random Forest Scatterplot (predicted vs observed values)

ggplot(predictions, aes(x = value, y = .pred)) +
  geom_point(alpha = 0.5) +  # Adding some transparency to points
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Adds a regression line without confidence envelope
  labs(x = "Observed Values", y = "Predicted Values", title = "Observed vs. Predicted Values for Random Forests") +
  theme_minimal()
```

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



```
# RMSE VALUES TABLE

# Manually created a table with the RMSE values and the name of the method used.
rmse_values <- c(lin_RMSE, k_rmse, dt_rmse, rf_rmse)
model_names <- c("Linear Regression", "K-Nearest Neighbors", "Decision Trees","Random Forests")

rmse_summary <- data.frame(Model = model_names, RMSE = rmse_values)
rmse_summary
```

```
##                 Model     RMSE
## 1   Linear Regression 2.641683
## 2 K-Nearest Neighbors 2.430612
## 3      Decision Trees 2.780553
## 4      Random Forests 2.375692
```

Based on these results the best model for the data set is Random Forests.

Observations

```
## BEST AND WORST PREDICTIONS

# Sort by the absolute values of residuals to find the best predictions
predictive_values <- predictions |>
  arrange(abs(resid))

# Sort by location and arrange by residual values.
geolocations <- predictive_values |>
  select(lat, lon, value, .pred, resid,state) |>
  arrange(abs(resid))

geolocations
```

```
## # A tibble: 176 × 6
##      lat    lon value .pred   resid state
##    <dbl>  <dbl> <dbl> <dbl>   <dbl> <chr>
##  1  40.0  -75.1 12.9  12.9  -0.0124 Pennsylvania
##  2  38.7  -90.2 12.9  12.9   0.0148 Missouri
##  3  43.4  -89.7 10.5  10.4   0.0206 Wisconsin
##  4  38.1  -94.7  9.75  9.73  0.0276 Kansas
##  5  41.6  -87.3 12.2  12.3  -0.0352 Indiana
##  6  35.1  -98.3  9.68  9.74 -0.0561 Oklahoma
##  7  36.1  -79.8 11.6  11.5   0.0792 North Carolina
##  8  46.5  -84.3 10.6  10.7  -0.0992 Michigan
##  9  33.7 -116.   8.25  8.36 -0.106  California
## 10  35.3  -93.1 11.3  11.2   0.109  Arkansas
## # ℹ 166 more rows
```

The regions with the closest predictions are located near the center of the US, except for North Dakota that is up north. They are all smaller states that don’t have huge metroplexes and are filled with smaller cities. This means there is less variation in the variables, which leads to better model accuracy. The 5 bottom down are all in California, which makes sense as it is overpopulated and there is a lot of traffic, which means more pollution. Since California is an outlier state in terms of its variables, our model is less accurate when predicting its values. Overall, the model performs best on states without major cities, and more rural populations. On the other hand, the model performs worst on states with big metropolitan cities.