Project Report

Introduction

The goal of this project is to build a prediction model that can predict ambient air pollution concentrations across the continental United States.

The dataset contains annual average concentrations of fine particulate matter (PM2.5) across the U.S. Environmental Protection Agency's monitoring network in the continental U.S. You can read the data in the **simpleExcel.xlsx** file.

Information & Summary of Variables (Predictors and Response)

The outcome is in a column called value and it represents the annual average PM2.5 con- centration at a monitor. It is measured in micrograms per cubic meter. The measurements here were taken using what is known as a gravimetric method and it is considered the gold standard for outdoor air pollution concentrations. The rest of the predictor variables and what each of them represent can be viewed in the **information_Airpollution.pdf**.

```
options(readr.show_col_types = FALSE)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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 1.2.0 --
## v broom        1.0.5      v rsample      1.2.1
## v dials        1.2.1      v tune         1.2.1
## v infer        1.0.7      v workflows    1.1.4
## v modeldata    1.3.0      v workflowsets 1.1.0
## v parsnip      1.2.1      v yardstick    1.3.1
## v recipes      1.0.10
## -- 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 tidymodels_prefer() to resolve common conflicts.
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
dat <- read_csv("pm25_data.csv.gz")
library(writexl)
write_xlsx(dat,"simpleExcel.xlsx",  col_names = TRUE,
  format_headers = TRUE)
```

Predictor Variables First we found the correlation between all the numeric features and the response to find the best predictors. We excluded the zcta variable as it was essentially character values encoded in numeric form.

```
numeric_dat <- dat |>
    select(where(is.numeric))

numeric_dat <- subset(numeric_dat, select = -c(zcta))

output <- numeric_dat |>
    cor()
```

The total number of predictors chosen were 31. The predictors chosen were

- Aod
- pov
- somehs
- nohs
- popdens_county
- popdens_zcta
- All log_nei_2008 (6)
- All log_prisec_length (6)
- All log_pri_length (4)
- county_pop
- All imp_a (4)
- CMAQ
- zcta_pop
- lat
- lon

We chose all the predictors that had a correlation with response variable (value) greater than .10 with the exception of *hs_orless*. We excluded *hs_orless* as it contained data of already chosen predictors such as *somehs* and *nohs* and felt that it would be redundant. We also included latitude even though latitude had a negative correlation as it made no sense to include longitude without latitude.

Models We are using Linear Regression, K nearest neighbor, and the random forrest models.

Expectations Our expectation is that the K nearest neighbor would be the best model of the three. The KNN performs best on problems where the data is continuous, and where the output depends on the similarity of the data points which fits with the data we have. KNN also works best when we do not have any knowledge of the relationships in the data, giving more flexibility.

We expect Linear regression won't work every well because the dataset may not be linearly related among the predictors and response. Also it does not account for real world complexity as it is too simple of a model so outliers may have a large effect on the predictions.

We expect decision tree models to not work well because the data doesn't follow a hierarchical structure. We also know decision trees work poorly when the data is continuous and where the output depends on the complex interactions such as the subtle discrepancy in the data.

Results

Split the dataset into train and test, 75% for the training set and 25% for the test set.

```
dat_split <- initial_split(dat)
dat_train <- training(dat_split)
dat_test <- testing(dat_split)
```

Linear Regression

```
set.seed(123)

## Create the recipe
rec_lr <- dat_train |>
    recipe(value ~ lat + lon + CMAQ + zcta_pop + imp_a500 + imp_a5000 + imp_a10000 + imp_a15000 + county
log_prisec_length_10000 + log_prisec_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_100

## Create the model
model_lr <- linear_reg() |>
    set_engine("lm") |>
    set_mode("regression")

## Create the workflow
wf_lr <- workflow() |>
    add_recipe(rec_lr) |>
    add_model(model_lr)

## Check performance using Cross Validation
folds_lr <- vfold_cv(dat_train, v=10)

res_lr <- fit_resamples(wf_lr, resamples = folds_lr)

res_lr |>
    collect_metrics()
```

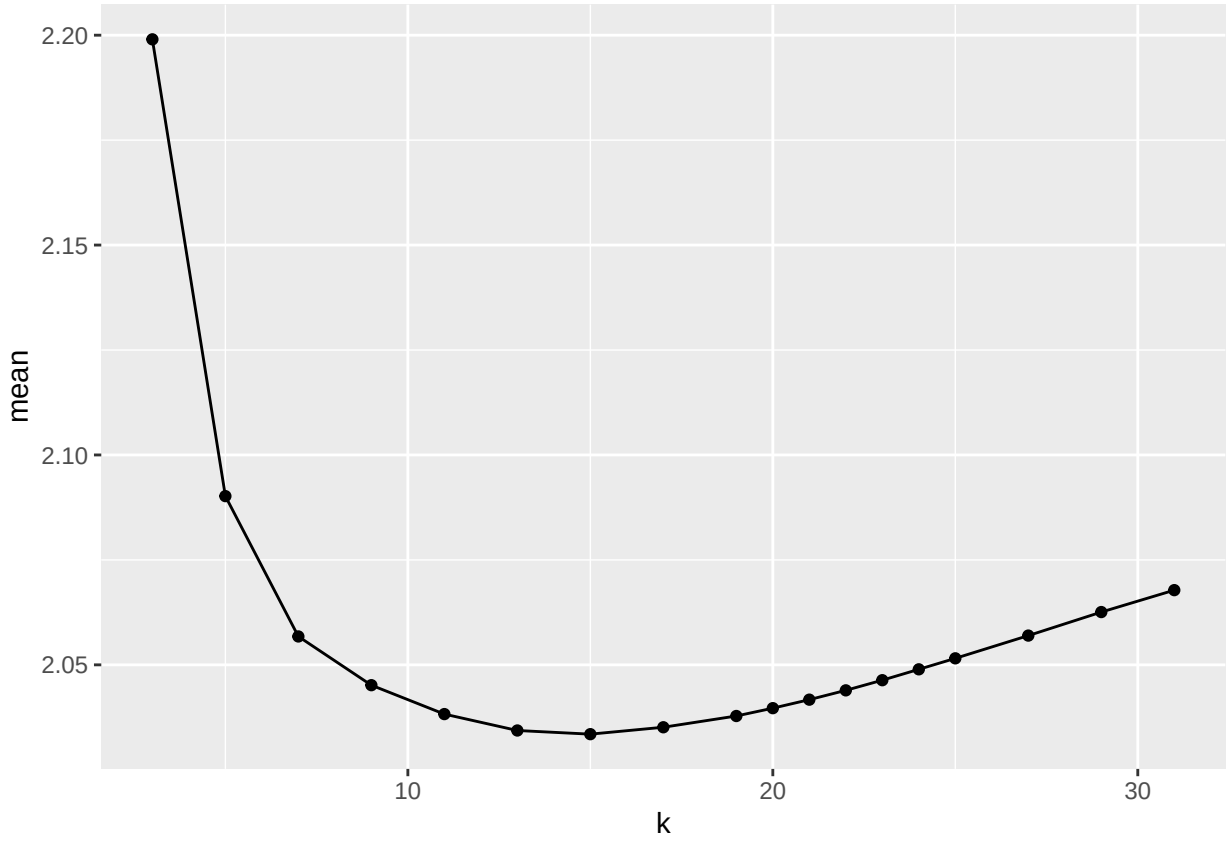
```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   2.22     10  0.111  Preprocessor1_Model1
## 2 rsq     standard   0.281    10  0.0452 Preprocessor1_Model1
```

The RMSE value for Linear Regression Model is 2.1354479.

```
set.seed(123)

## Create the recipe, we normalize all the predictor variables
rec_knn <- dat_train |>
    recipe(value ~ lat + lon + CMAQ + zcta_pop + imp_a500 + imp_a5000 + imp_a10000 + imp_a15000 + county
log_prisec_length_10000 + log_prisec_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_100
    step_normalize(starts_with("imp"), starts_with("log"), lat, lon, CMAQ, zcta_pop, county_pop, popdens

## Create the model
model_knn <- nearest_neighbor(neighbors = tune("k")) |>
    set_engine("kknn") |>
    set_mode("regression")

## Create the workflow
wf_knn <- workflow() |>
    add_recipe(rec_knn) |>
    add_model(model_knn)

## Check performance using Cross Validation
folds_knn <- vfold_cv(dat_train, v=10)

## find the most optimal k neighbor by tuning
res_knn <- tune_grid(wf_knn, resamples = folds_knn, grid = tibble(k = c(3,5,7,9,11,13,15,17,19,20,21,22
```

KNN Regression Visualize which K value produced the best RMSE.

```
res_knn |>
    collect_metrics() |>
    filter(.metric == "rmse") |>
    ggplot(aes(k, mean)) +
    geom_point() +
    geom_line()
```



show the best k values

```
res_knn |>
    show_best(metric="rmse")
```

```
## # A tibble: 5 x 7
##       k .metric .estimator  mean     n std_err .config
##   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    15 rmse    standard    2.03    10  0.0881 Preprocessor1_Model07
## 2    13 rmse    standard    2.03    10  0.0890 Preprocessor1_Model06
## 3    17 rmse    standard    2.04    10  0.0878 Preprocessor1_Model08
## 4    19 rmse    standard    2.04    10  0.0878 Preprocessor1_Model09
## 5    11 rmse    standard    2.04    10  0.0910 Preprocessor1_Model05
```

We can see that k value of 17 produced the most optimal RMSE value for knn model which was 1.966808.

```
set.seed(123)

## Create the recipe
rec_tree <- dat_train |>
    recipe(value ~ lat + lon + CMAQ + zcta_pop + imp_a500 + imp_a5000 + imp_a10000 + imp_a15000 + county
log_prisec_length_10000 + log_prisec_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_100
```

```
    step_normalize(starts_with("imp"), starts_with("log"), lat, lon, CMAQ, zcta_pop, county_pop, popdens


## Create the model
model_tree <- decision_tree(tree_depth = tune("tree_depth"), min_n = tune("min_n")) |>
    set_engine("rpart") |>
    set_mode("regression")

## Create the workflow
wf_tree <- workflow() |>
    add_recipe(rec_tree) |>
    add_model(model_tree)

## Check performance using Cross Validation
folds_tree <- vfold_cv(dat_train, v=10)

res_tree <- tune_grid(wf_tree, resamples = folds_tree, grid = expand.grid(tree_depth=c(5,10,15,20,25,30)
```

Decision Tree Show the best values after tuning tree_depth and min_n.

```
res_tree |>
    show_best(metric="rmse")
```

```
## # A tibble: 5 x 8
##   tree_depth min_n .metric .estimator  mean     n std_err .config
##        <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1         10     9 rmse    standard    1.95    10  0.0978 Preprocessor1_Model14
## 2         15     9 rmse    standard    1.95    10  0.0978 Preprocessor1_Model15
## 3         20     9 rmse    standard    1.95    10  0.0978 Preprocessor1_Model16
## 4         25     9 rmse    standard    1.95    10  0.0978 Preprocessor1_Model17
## 5         30     9 rmse    standard    1.95    10  0.0978 Preprocessor1_Model18
```

We can see that the decision tree regression model at min_n = 9 regardless of tree depth produces the best RMSE value at 2.004067.

Final Result, based on the above result we can conclude that the knn model produced the best results.

```
set.seed(123)

## Create the recipe, we normalize all the predictor variables
rec_knn <- dat_train |>
    recipe(value ~ lat + lon + CMAQ + zcta_pop + imp_a500 + imp_a5000 + imp_a10000 + imp_a15000 + county
log_prisec_length_10000 + log_prisec_length_15000 + log_prisec_length_25000 + log_nei_2008_pm25_sum_100
    step_normalize(starts_with("imp"), starts_with("log"), lat, lon, CMAQ, zcta_pop, county_pop, popdens

## Create the model
model_knn <- nearest_neighbor(neighbors = 17) |>
    set_engine("kknn") |>
    set_mode("regression")

## Create the workflow
wf_knn <- workflow() |>
    add_recipe(rec_knn) |>
```

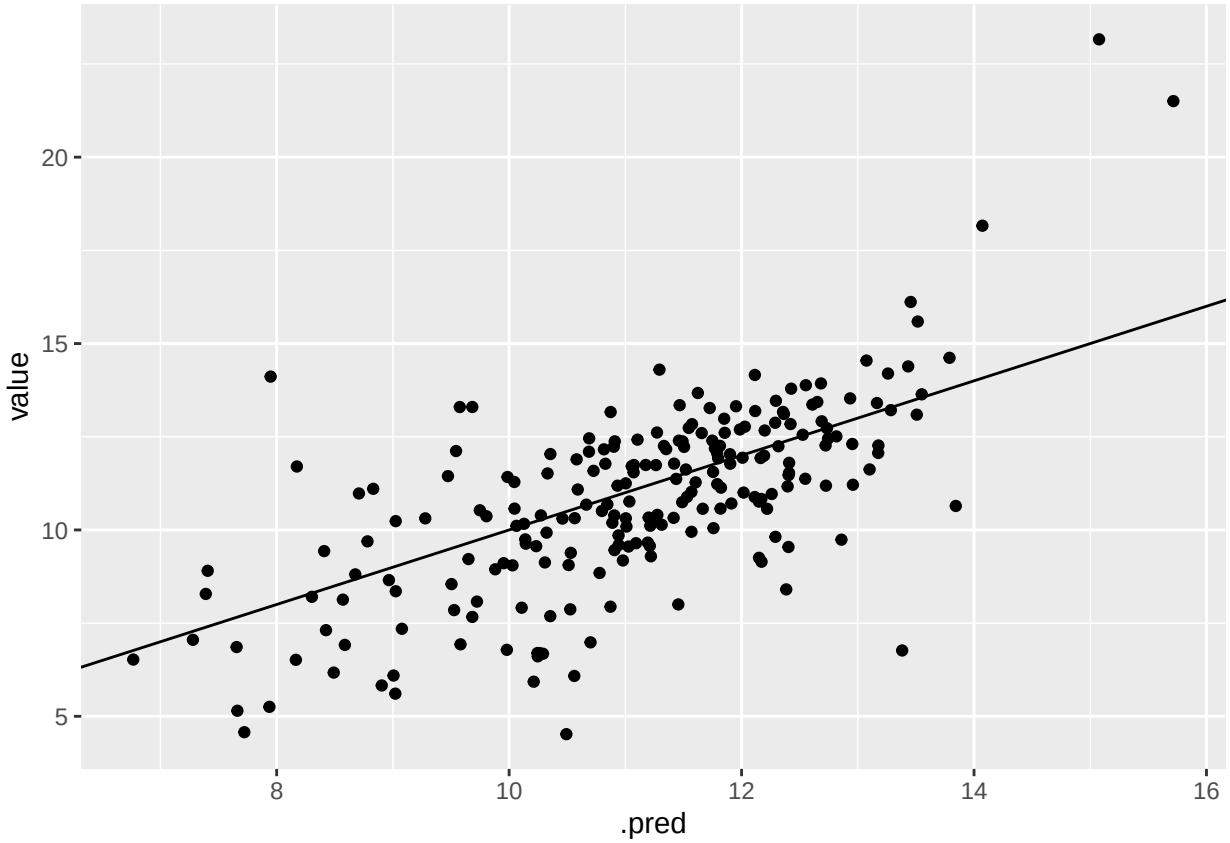
```
    add_model(model_knn)

## Check performance using Cross Validation
folds_knn <- vfold_cv(dat_train, v=10)

## fit the final model to entire training set, evalutae on test set
final <- wf_knn |>
    last_fit(split = dat_split)

final |>
    collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       1.90  Preprocessor1_Model1
## 2 rsq     standard       0.469 Preprocessor1_Model1
```

The RMSE value on the test dataset was 2.0430673 using the KNN models with k neighbors = 17.

Scatterplot of predicted values vs. observed values for the best and final model in the test dataset.

```
final |>
    collect_predictions() |>
    ggplot(aes(.pred, value)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1)
```



Discussion

1. Based on test set performance, at what geographic locations in the test set does your model give predictions that are closest and furthest from the observed values? What do you hypothesize are the reasons for the good or bad performance at these locations?

Wyoming, South Dakota, and Main had the most values that were the furthest from the observed values. The reason may be due to the fact that a lot of the data collected on these states were very few. In addition, these are among the most sparsely populated states in America so attention may not have been given to these states as others. A lot of the South and Eastern states did very well. These were also the states with the most data collected so the accuracy may be due to that factor.

2. What variables might predict where your model performs well or not? For example, are their regions of the country where the model does better or worse? Are there variables that are not included in this dataset that you think might improve the model performance if they were included in your model?

Population Centers in the South and East had the best predictions by the model.

3. There is interest in developing more cost-effect approaches to monitoring air pollution on the ground. Two candidates for replacing the use of ground-based monitors are numerical models like CMAQ and satellite-based observations such as AOD. How does the prediction performance of your best and final model change when CMAQ or aod are included (or not included) in the model?

The model becomes worse without the CMAQ or aod. The RMSE values were higher overall for all models across the board.

4. The dataset here did not include data from Alaska or Hawaii. Do you think your model will perform well or not in those two states? Explain your reasoning.

No, because both states are located so far away from the mainland states, the knn model would definately perform worse.

- Reflect on the process of conducting this project. What was challenging, what have you learned from the process itself?

Researching the models and how to tune each of them were the most challenging part, but also allowed me to learn a lot of R related skills in regards to analyzing data through regression models.

- Reflect on the performance of your final prediction model. Did it perform as well as you originally expected? If not, why do you think it didn't perform as well?

Yes, the knn model performed as expected and gave us the best result of the three used. The final results in the West coast states like California for some reason did not perform as well, but the rest was predicted very closely.