HW3

Areeya Aksornpan, Zayd Abdalla

4/9/2021

```
read_data <- function(df) {
  full_path <- paste("https://raw.githubusercontent.com/jgscott/ECO395M/master/data/",
                     df, sep = "")
  df <- read_csv(full_path)
  return(df)
}
```

What causes what?

Q1. Why can't I just get data from a few different cities and run the regression of "Crime" on "Police" to understand how more cops in the streets affect crime?

This problem poses a causal question, for which establishing a causal relationship is rather complicated. For instance, cities with high levels of crime may have an incentive to hire more police wheras Cities with low crime may have fewer police. Nonetheless, there is potential for this relationship to be confounding due to variation in cities, which makes establishing causality more complicated without controlling for other factors.

Q2. How were the researchers from UPenn able to isolate this effect? Briefly describe their approach and discuss their result in the "Table 2" below, from the researchers' paper.

The researchers isolated this effect by selecting an example where there had been a lot of police for reasons unrelated to crime. This led the researchers to discover the terrorist alert system in DC. When the terror alert level goes to orange, extra police are put on the Mall and other parts of Washington. This means that the causal effect of police on crime can be better observed since the presence of the police is independent of street crime. As a result, the researchers found that more police is associated with less murder, robbery, and assault. As for the table, model 1 indicates that when the system is on high alert, crime falls by 7.316 units (units are unclear from model and problem). Model 2 is similar to model 1 and controls for metro ridership, but the conclusion is similar to model 1 in that more policing reduces crime by 6.046 units.

Q3. Why did they have to control for Metro ridership? What was that trying to capture?

The researchers wanted to address the concern of citizens and tourists avoiding the capital when terrorist alerts are issued, which could mean there are less potential victims on the streets. So they controlled for metro ridership and found that levels of metro ridership were not diminished on days of high terror days. Specifically, while the effect of the police presence on decreasing crime was slightly reduced after controlling for metro ridership, the effect was still significant.

Q4. Below I am showing you "Table 4" from the researchers' paper. Just focus on the first column of the table. Can you describe the model being estimated here? What is the conclusion?

Column 1 models crime on the interaction of high alert and district 1, the interaction of high alert and other districts, and controls for metro ridership levels with the log of midday ridership. When there is a high alert, crime in District 1 decreases by 2.6 units (units are not clear from table or problem), controlling for ridership. This effect is statistically significant. Next, high alerts in other districts are associated with a small and not statistically significant decrease in crime. Since the police presence during a high alert is centralized around District 1, there is evidence that increasing police presence in DC decreases crime.

Predictive model building: Green Certification

We explore data on commercial rental properties from across the United States to build a predictive model for revenue/ sq. ft. per calendar year, and to use this model to quantify the average change in rental income per square foot (whether in absolute or percentage terms) associated with green certification, holding other features of the building constant.

```
green_buildings <-
  read_data("greenbuildings.csv") %>%
  janitor::clean_names() %>%
  # Revenue per sq ft
  mutate(rev_per_sqft = rent * leasing_rate)

# Split into train/test split
set.seed(395)
green_split <- initial_split(green_buildings, strata = rev_per_sqft)
green_train <- training(green_split)
green_test <- testing(green_split)

# v-fold
set.seed(3951)
green_folds <- vfold_cv(green_train, v = 3, strata = rev_per_sqft)
green_folds
```

```
## #  3-fold cross-validation using stratification
## # A tibble: 3 x 2
##   splits               id
##   <list>               <chr>
## 1 <split [3947/1974]> Fold1
## 2 <split [3947/1974]> Fold2
## 3 <split [3948/1973]> Fold3
```

```
green_recipe <-
  recipe(rev_per_sqft ~ ., green_train) %>%
  update_role(contains("id"), new_role = "ID") %>% # Declaring ID vars
  step_mutate(
    green_certf = case_when(
      leed == 1 ~ "leed",
      energystar == 1 ~ "energystar",
      leed == 0 & energystar == 0 ~ "none"
    )
  ) %>%
  step_rm(c(rent, leasing_rate, leed, energystar, green_rating, total_dd_07,
city_market_rent)) %>% # Remove Confounders
  step_nzv(all_predictors(), freq_cut = 0, unique_cut = 0) %>% # Remove zero
variance vars
  step_novel(all_nominal()) %>% # Assigns previously unseen factor level to a
new value
  step_unknown(all_nominal()) %>% # NA's are categorized as unknowns
  step_medianimpute(all_numeric(), -all_outcomes(), -has_role("ID"))  %>% # R
eplace missing numeric obs with median values
  step_dummy(all_nominal(), -has_role("ID")) # Code categorical as dummy vari
ables
```

Decision Tree

```
tree_spec <- decision_tree(
  cost_complexity = tune(),
  tree_depth = tune(),
  min_n = tune()
) %>%
  set_engine("rpart") %>%
  set_mode("regression")

tree_spec
```

```
## Decision Tree Model Specification (regression)
##
## Main Arguments:
##   cost_complexity = tune()
##   tree_depth = tune()
##   min_n = tune()
##
## Computational engine: rpart
```

```r
# Tuning grid
tree_grid <-
  grid_regular(
    cost_complexity(),
    tree_depth(),
    min_n(),
    levels = 4
    )

tree_grid
```

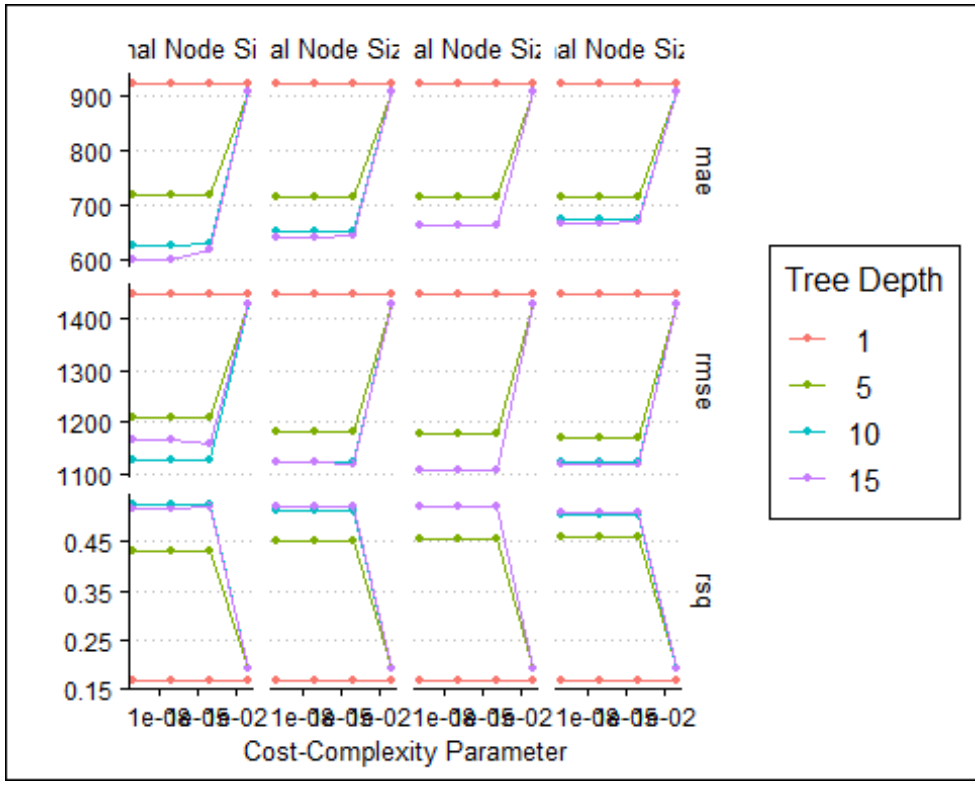
```
## # A tibble: 64 x 3
##    cost_complexity tree_depth min_n
##              <dbl>      <int> <int>
##  1    0.0000000001          1     2
##  2    0.0000001             1     2
##  3    0.0001                1     2
##  4    0.1                   1     2
##  5    0.0000000001          5     2
##  6    0.0000001             5     2
##  7    0.0001                5     2
##  8    0.1                   5     2
##  9    0.0000000001         10     2
## 10    0.0000001            10     2
## # ... with 54 more rows
```

```r
# WF
workflow_tree <-
  workflow() %>%
  add_recipe(green_recipe) %>%
  add_model(tree_spec)

# Check all parameter values on the re-sampled data
doParallel::registerDoParallel()

set.seed(3452)

tree_resample <-
  tune_grid(
    workflow_tree,
    resamples = green_folds,
    grid = tree_grid,
    metrics = metric_set(rmse, rsq, mae)
  )

tree_resample
```

```
## # Tuning results
## # 3-fold cross-validation using stratification
## # A tibble: 3 x 4
```

```
##   splits               id    .metrics           .notes
##   <list>               <chr> <list>             <list>
## 1 <split [3947/1974]> Fold1 <tibble [192 x 7]> <tibble [0 x 1]>
## 2 <split [3947/1974]> Fold2 <tibble [192 x 7]> <tibble [0 x 1]>
## 3 <split [3948/1973]> Fold3 <tibble [192 x 7]> <tibble [0 x 1]>
```

```
# evaluate model
# collect_metrics(tree_rs)

autoplot(tree_resample) + theme_clean()
```



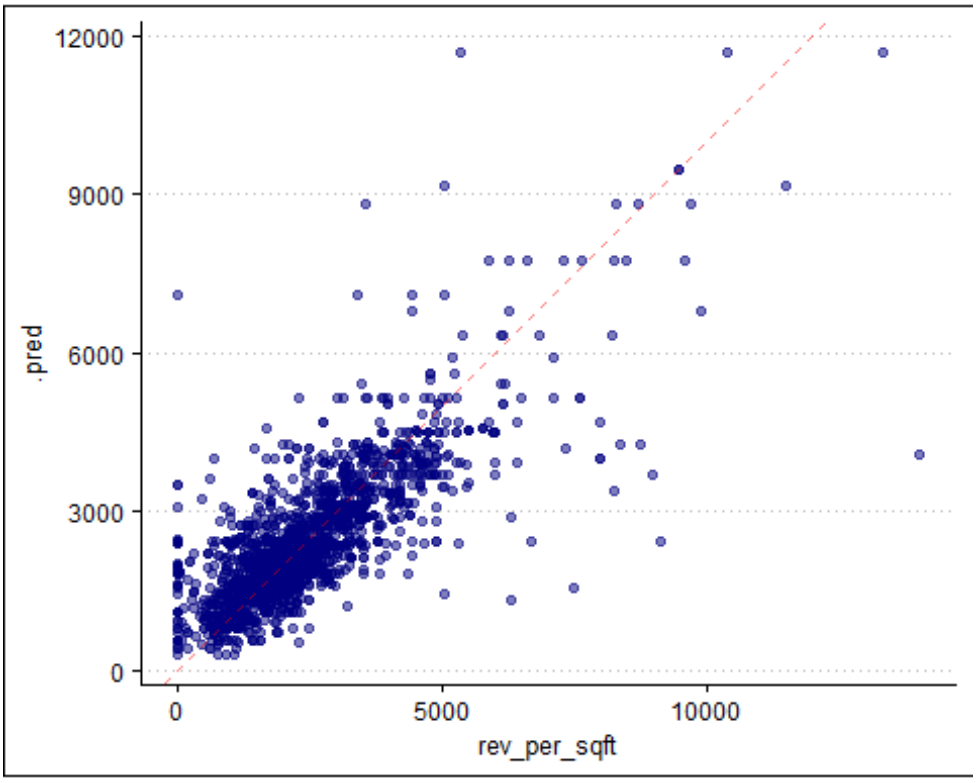

```
lowest_tree_rmse <- select_best(tree_resample, "rmse")

## Out of sample performance

# Finalize WF
final_workflow <-
  workflow_tree %>%
  finalize_workflow(lowest_tree_rmse)

final_workflow
```

```
## == Workflow ====================================================================
======
## Preprocessor: Recipe
## Model: decision_tree()
##
```

```
## -- Preprocessor ----------------------------------------------------------
------
## 7 Recipe Steps
##
## * step_mutate()
## * step_rm()
## * step_nzv()
## * step_novel()
## * step_unknown()
## * step_medianimpute()
## * step_dummy()
##
## -- Model ----------------------------------------------------------------
------
## Decision Tree Model Specification (regression)
##
## Main Arguments:
##   cost_complexity = 1e-04
##   tree_depth = 15
##   min_n = 27
##
## Computational engine: rpart


final_resample <- last_fit(final_workflow, green_split)

# look at test data
collect_metrics(final_resample)[,1:3] %>% kbl(digits = 3, format = "pipe")
```

.metric	.estimator	.estimate
rmse	standard	919.116
rsq	standard	0.609

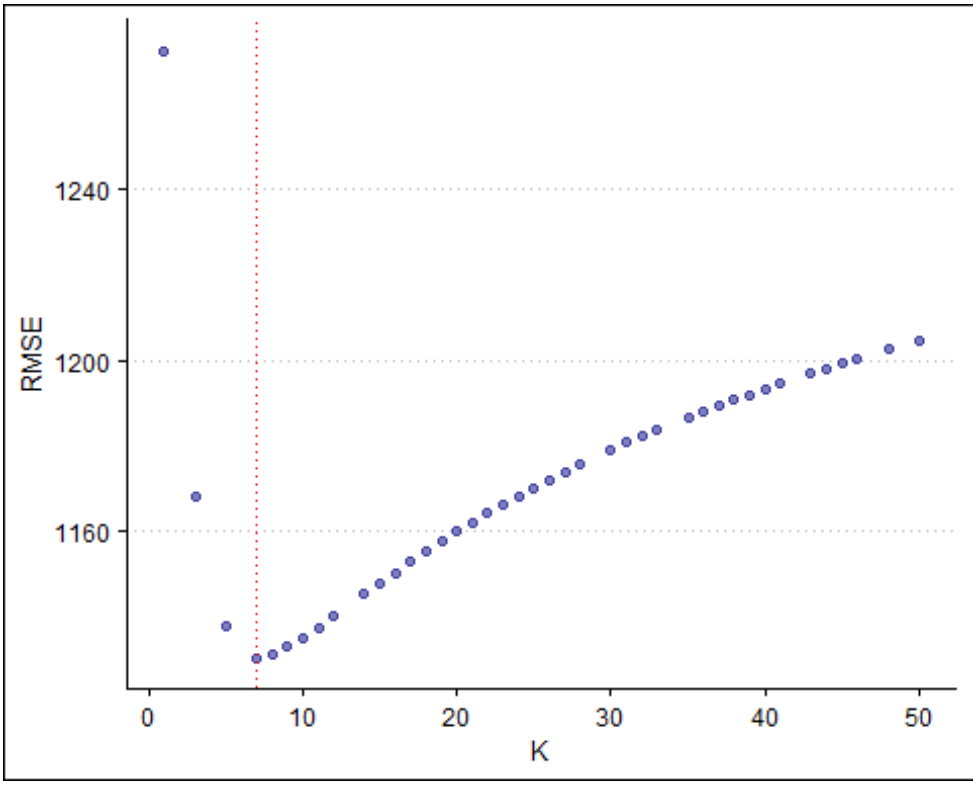
```
# look at predictions
final_resample %>%
  collect_predictions() %>%
  ggplot(aes(rev_per_sqft, .pred)) +
  geom_point(alpha = 0.5, color = "navy") +
  geom_abline(slope = 1, lty = 2, color = "red", alpha = 0.5) +
  theme_clean() +
  coord_fixed()
```



KNN-regression

```
knn_spec <-
  nearest_neighbor(
    mode = "regression",
    neighbors = tune("K")
  ) %>%
  set_engine("kknn")

# construct workflow
workflow_knn <-
  workflow() %>%
  add_recipe(green_recipe) %>%
  add_model(knn_spec)

knn_set <-
  parameters(workflow_knn) %>%
  # try k in 1:50
  update(K = neighbors(c(1, 50)))

set.seed(3952)
knn_grid <-
  knn_set %>%
  grid_max_entropy(size = 50)

knn_grid_search <-
  tune_grid(
```

```
    workflow_knn,
    resamples = green_folds,
    grid = knn_grid
  )


lowest_rmse_knn <- select_best(knn_grid_search, "rmse")

best_k <- as.numeric(lowest_rmse_knn$K)

# plot rmse
collect_metrics(knn_grid_search) %>%
  filter(.metric == "rmse") %>%
  ggplot() +
  geom_point(aes(x = K, y = mean), color = "navy", alpha = 0.5) +
  geom_vline(aes(xintercept = best_k), linetype = 3, color = "red") +
  labs(y = "RMSE") +
  theme_clean()
```




```
## Out of sample performance

# Finalize WF
final_workflow_knn <-
  workflow_knn %>%
  finalize_workflow(lowest_rmse_knn)

final_workflow_knn
```

```
## == Workflow ====================================================================
======
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
------
## 7 Recipe Steps
##
## * step_mutate()
## * step_rm()
## * step_nzv()
## * step_novel()
## * step_unknown()
## * step_medianimpute()
## * step_dummy()
##
## -- Model ----------------------------------------------------------------------
------
## K-Nearest Neighbor Model Specification (regression)
##
## Main Arguments:
##   neighbors = 7
##
## Computational engine: kknn
```

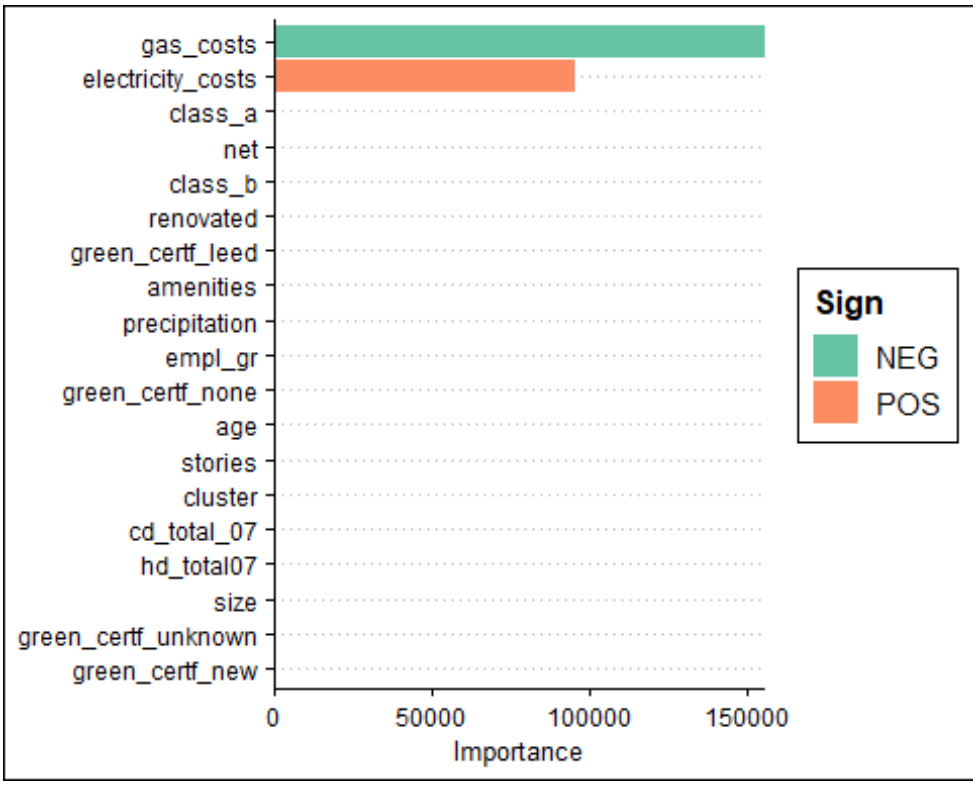
```
# fit the model
last_fit(
  final_workflow_knn,
  green_split
  ) %>%
  collect_metrics() %>%
  select(1:3) %>%
  kbl(digits = 3, format = "pipe")
```

.metric	.estimator	.estimate
rmse	standard	973.463
rsq	standard	0.565

LASSO

```
lasso_spec <-
  linear_reg(
    penalty = tune(),
    mixture = 1
    ) %>%
  set_engine("glmnet")

# WF
workflow_lasso <-
```

```
  workflow() %>%
  add_recipe(green_recipe) %>%
  add_model(lasso_spec)

lambda_grid <- grid_regular(penalty(), levels = 50)

doParallel::registerDoParallel()
set.seed(3955)
lasso_grid <-
  tune_grid(
    workflow_lasso,
    resamples = green_folds,
    grid = lambda_grid
  )

lowest_rmse <-
  lasso_grid %>%
  select_best("rmse")

final_lasso <-
  finalize_workflow(
    workflow_lasso,
    lowest_rmse
  )

final_lasso %>%
  fit(green_train) %>%
  pull_workflow_fit() %>%
  vi(lambda = lowest_rmse$penalty) %>%
  mutate(
    Importance = abs(Importance),
    Variable = fct_reorder(Variable, Importance)
  ) %>%
  ggplot(aes(x = Importance, y = Variable, fill = Sign)) +
  geom_col() +
  scale_fill_brewer(palette = "Set2") +
  scale_x_continuous(expand = c(0, 0)) +
  labs(y = NULL) +
  theme_clean()
```




```
last_fit(
  final_lasso,
  green_split
  ) %>%
  collect_metrics() %>%
  select(1:3) %>%
  kbl(digits = 3, format = "pipe")
```

.metric	.estimator	.estimate
rmse	standard	1218.393
rsq	standard	0.305

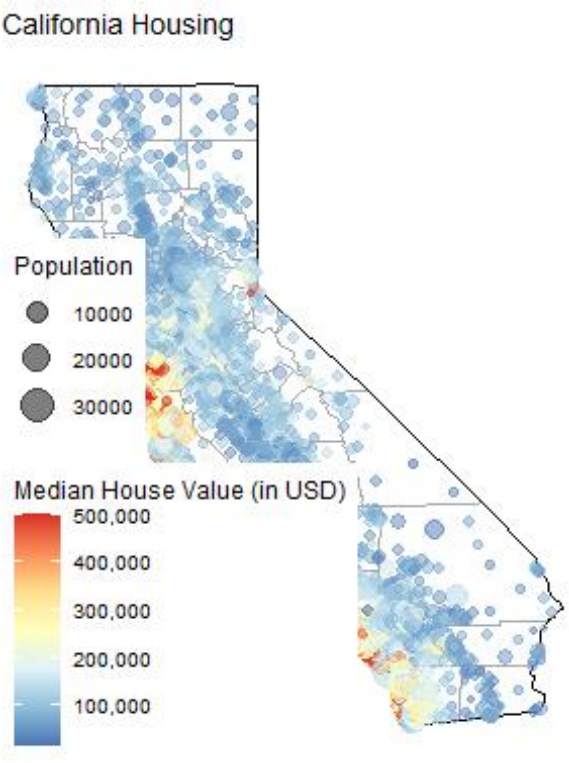
Though we took 3 different approaches, all models are fairly similar in accuracy of predictions. By consulting the vip plots, we find that Green Certification does not appear to be an important feature in determining rent price per sq. ft. Rather, gas costs and electricity costs appear to be the most important in predicting the price of rent.

Predictive model building: California housing

We explore census-tract level on residential housing in the state of California to build a predictive model for median house value.

```
CAhousing <-
  read_data("CAhousing.csv") %>%
  janitor::clean_names()
```

```r
states <- map_data("state")
county <- map_data("county")
ca_df <- subset(states, region == "california")
ca_county <- subset(county, region == "california")

ca_base <-
  ggplot() +
  coord_fixed(1.3) +
  geom_polygon(data = ca_df,
               aes(x = long, y = lat, group = group),
               color = "black", fill = "white")  +
  geom_polygon(data = ca_county,
               aes(x = long, y = lat, group = group),
               fill = NA, color = "dark grey") +
  geom_polygon(data = ca_df,
               aes(x = long, y = lat, group = group),
               color = "black", fill = NA)

ca_base +
  geom_point(data = CAhousing,
    aes(x = longitude, y = latitude,
        color = median_house_value, size = population),
    alpha = 0.5) +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_map() +
  scale_color_distiller(palette = "RdYlBu", labels = comma) +
  labs(title = "California Housing",
       x = "Longitude", y = "Latitude",
       color = "Median House Value (in USD)",
       size = "Population")
```



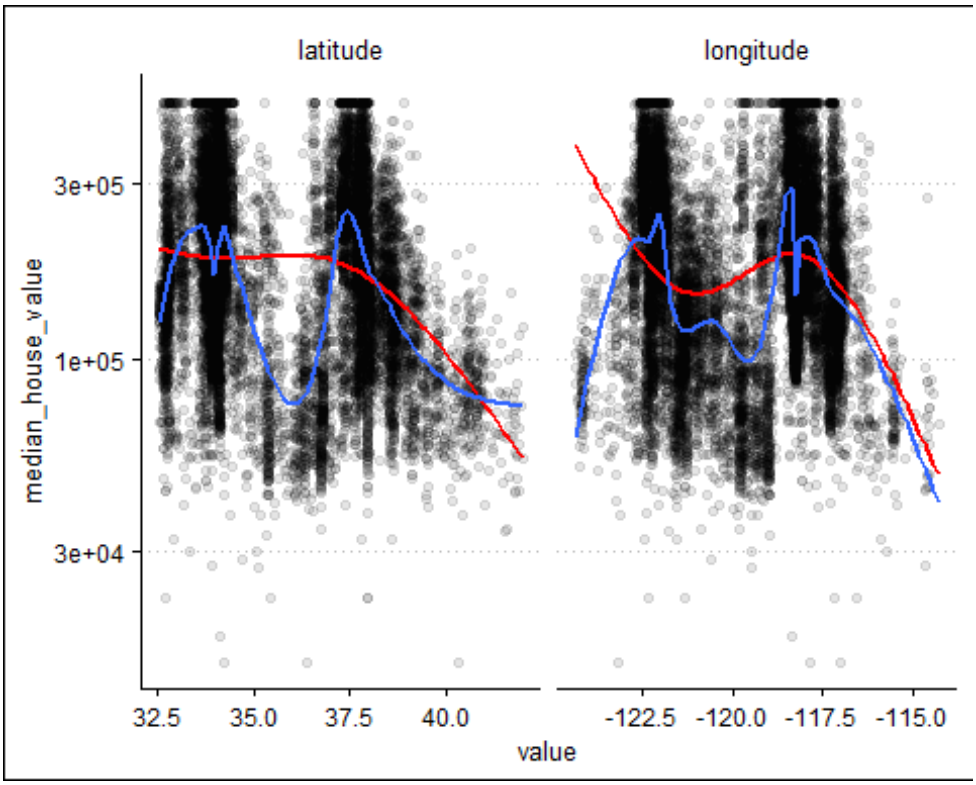

```r
set.seed(395)

# splits
housing_split <- initial_split(CAhousing, prop = 0.75, strata = median_house_
value)
housing_train <- housing_split %>% training()
housing_test <- housing_split %>% testing()

# vfold
housing_vfold <- vfold_cv(housing_train, v = 10, strata = median_house_value)

set.seed(395)

# LM as baseline
lm_model <-
  linear_reg() %>%
  set_engine('lm') %>%
  set_mode('regression')

# Recipe
lm_recipe <-
  # fit on all variables
  recipe(median_house_value ~ ., data = housing_train) %>%
  # log price
  step_log(median_house_value) %>%
  # standardize
```

```
  step_range(total_bedrooms, total_rooms, population, housing_median_age, med
ian_income) %>%
  step_ns(longitude, deg_free = tune("long df")) %>%
  step_ns(latitude,  deg_free = tune("lat df"))

grid_vals <- seq(2, 22, by = 2)

spline_grid <- expand.grid(`long df` = grid_vals, `lat df` = grid_vals)

housing_parameters <-
  lm_recipe %>%
  parameters() %>%
  update(
    `long df` = spline_degree(),
    `lat df` = spline_degree()
  )

housing_parameters

## Collection of 2 parameters for tuning
##
##  identifier     type    object
##     long df deg_free nparam[+]
##      lat df deg_free nparam[+]

# WF
lm_workflow <-
  workflow() %>%
  add_model(lm_model) %>%
  add_recipe(lm_recipe)

tic()
lm_res <-
  lm_workflow %>%
  tune_grid(resamples = housing_vfold, grid = spline_grid)
toc()

## 538.1 sec elapsed

lm_est <- collect_metrics(lm_res)

lm_rmse_vals <-
  lm_est %>%
  dplyr::filter(.metric == "rmse") %>%
  arrange(mean)

lm_final <-
  lm_rmse_vals %>%
  filter(.metric == "rmse") %>%
  filter(mean == min(mean))
```

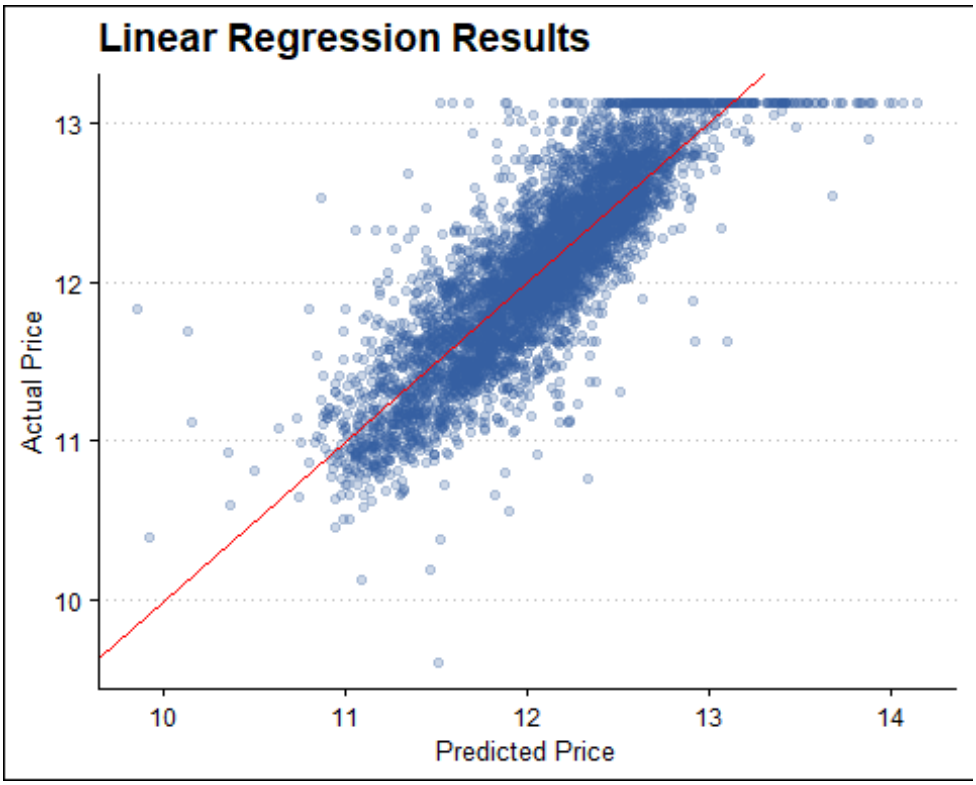
```r
lm_final_workflow <-
  lm_workflow %>%
  finalize_workflow(lm_final)

# fit the model using workflow to test set
lm_fit <-
  lm_final_workflow %>%
  last_fit(split = housing_split)

# Model Performance
lm_fit %>% collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0.296 Preprocessor1_Model1
## 2 rsq     standard       0.728 Preprocessor1_Model1
```

```r
housing_train %>%
  dplyr::select(median_house_value, longitude, latitude) %>%
  tidyr::pivot_longer(cols = c(longitude, latitude),
                      names_to = "predictor", values_to = "value") %>%
  ggplot(aes(x = value, median_house_value)) +
  geom_point(alpha = .1) +
  geom_smooth(se = FALSE, method = lm, formula = y ~ splines::ns(x, df = 3),
col = "red") +
  geom_smooth(se = FALSE, method = lm, formula = y ~ splines::ns(x, df = 16))
+
  scale_y_log10() +
  theme_clean() +
  facet_wrap(~ predictor, scales = "free_x")
```

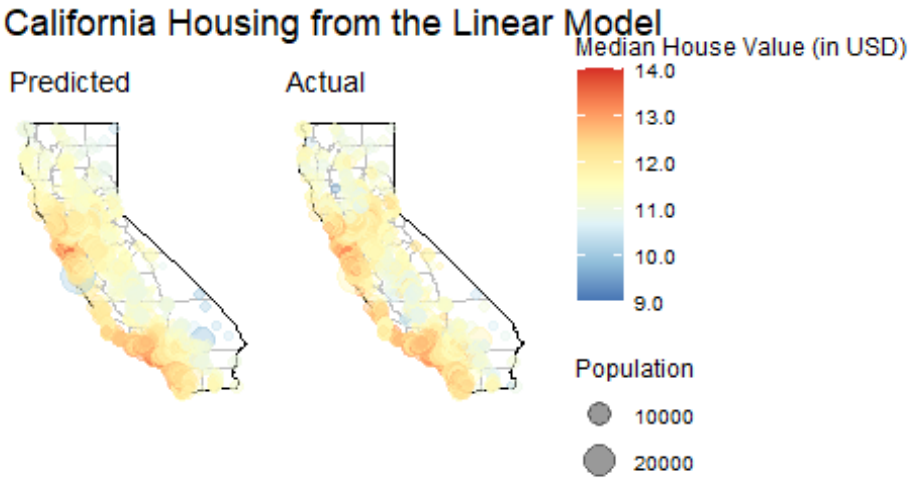



```
# Obtain test set predictions data frame
lm_results <-
  lm_fit %>%
  # save pred results
  collect_predictions()

lm_results <-
  lm_results %>%
  bind_cols(housing_test) %>%
  rename(median_house_value_log = `median_house_value...4`,
         median_house_value = `median_house_value...14`)

# plot pred v actual
lm_results %>%
  ggplot(aes(x = .pred, y = median_house_value_log)) +
  geom_point(color = '#345EA1', alpha = 0.25)  +
  geom_abline(intercept = 0, slope = 1, color = 'red') +
  labs(title = 'Linear Regression Results',
       x = 'Predicted Price',
       y = 'Actual Price') +
  theme_clean()
```

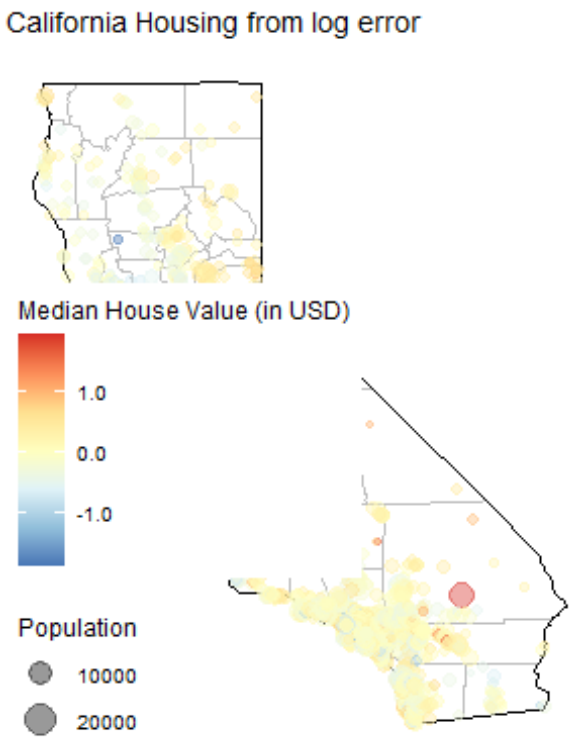



```
p1 <-
  ca_base +
  geom_point(data = lm_results,
    aes(x = longitude, y = latitude,
        color = .pred, size = population),
    alpha = 0.4) +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_map() +
  scale_color_distiller(palette = "RdYlBu", labels = comma,
                        limits = c(9, 14)) +
  labs(title = "Predicted ",
       x = "Longitude", y = "Latitude",
       color = "Median House Value (in USD)",
       size = "Population")

p2 <-
  ca_base +
  geom_point(data = lm_results,
    aes(x = longitude, y = latitude,
        color = median_house_value_log, size = population),
    alpha = 0.4) +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_map() +
  scale_color_distiller(palette = "RdYlBu", labels = comma,
                        limits = c(9, 14)) +
  labs(title = "Actual",
```

```
       x = "Longitude", y = "Latitude",
       color = "Median House Value (in USD)",
       size = "Population")

p1 + p2 +
  plot_layout(guides = 'collect') +
  plot_annotation(title = "California Housing from the Linear Model")
```




```
ca_base +
  geom_point(data = lm_results,
    aes(x = longitude, y = latitude,
        color = median_house_value_log - .pred, size = population),
    alpha = 0.4) +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_map() +
  scale_color_distiller(palette = "RdYlBu", labels = comma) +
  labs(title = "California Housing from log error",
       x = "Longitude", y = "Latitude",
       color = "Median House Value (in USD)",
       size = "Population")
```




```
set.seed(395)

# specify knn model
knn_model <-
  # specify hyperparameters
  nearest_neighbor(neighbors = tune(), weight_func = tune()) %>%
  set_engine('kknn') %>%
  set_mode('regression') %>%
  translate()

# define a recipe
knn_recipe <-
  # fit on all variables
  recipe(median_house_value ~ ., data = housing_train) %>%
  # log price
  step_log(median_house_value) %>%
  # standardize
  step_range(total_bedrooms, total_rooms, population, housing_median_age, med
ian_income) %>%
  # specify tuning hyperparameters
  step_ns(longitude, deg_free = tune("long df")) %>%
  step_ns(latitude,  deg_free = tune("lat df"))

# create a workflow
knn_workflow <-
  workflow() %>%
```

```
  # specify engine
  add_model(knn_model) %>%
  # specify recipe
  add_recipe(knn_recipe)

knn_parameters <-
  knn_workflow %>%
  # how to tune hyperparams
  parameters() %>%
    update(
    `long df` = spline_degree(c(2, 18)),
    `lat df` = spline_degree(c(2, 18)),
    neighbors = neighbors(c(3, 50)),
    weight_func = weight_func(values = c("rectangular", "inv", "triangular"))
  )

ctrl <- control_bayes(verbose = TRUE)

tic()
knn_search <-
  tune_bayes(knn_workflow, resamples = housing_vfold, initial = 5, iter = 10,
                           paramet_info = knn_parameters, control = ctrl)
toc()

## 197.93 sec elapsed

knn_final <-
  knn_search %>%
  collect_metrics() %>%
  dplyr::filter(.metric == "rmse") %>%
  filter(mean == min(mean))


knn_final_workflow <-
  knn_workflow %>%
  finalize_workflow(knn_final)

# fit the model using WF on test set
knn_fit <-
  knn_final_workflow %>%
  last_fit(split = housing_split)

# Model Performance
knn_fit %>%
  collect_metrics() %>%
  select(1:3) %>%
  kbl(digits = 3, format = "pipe")
```

.metric	.estimator	.estimate

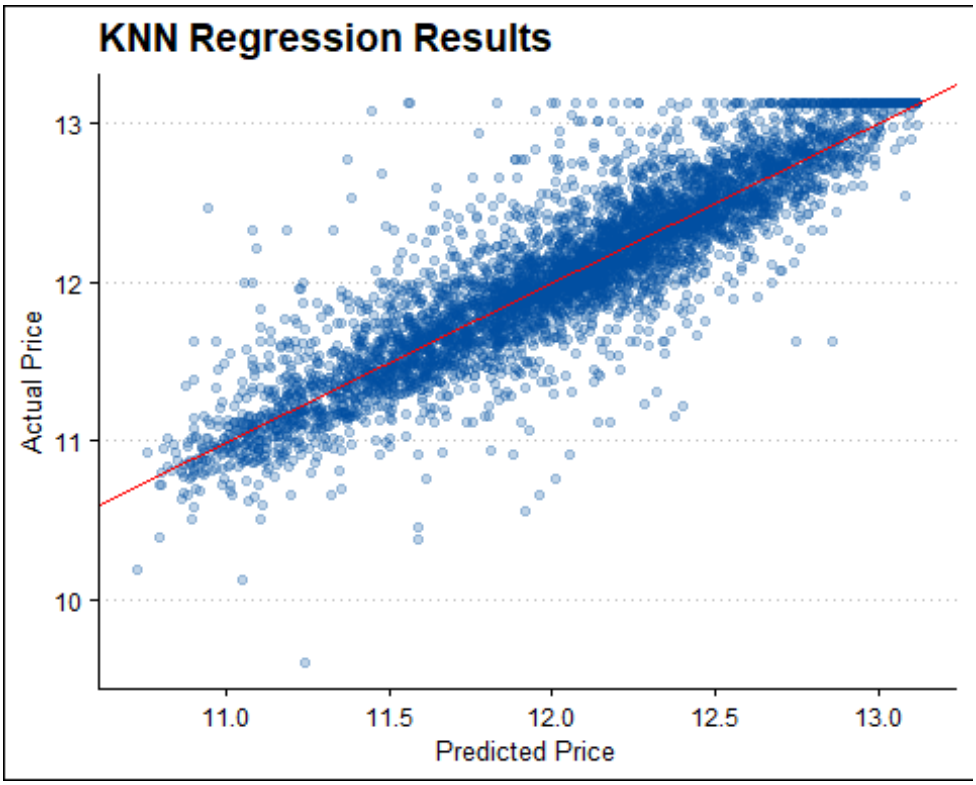
rmse	standard	0.242
rsq	standard	0.819

Bit better performance than linear model.

```
knn_final_workflow
```

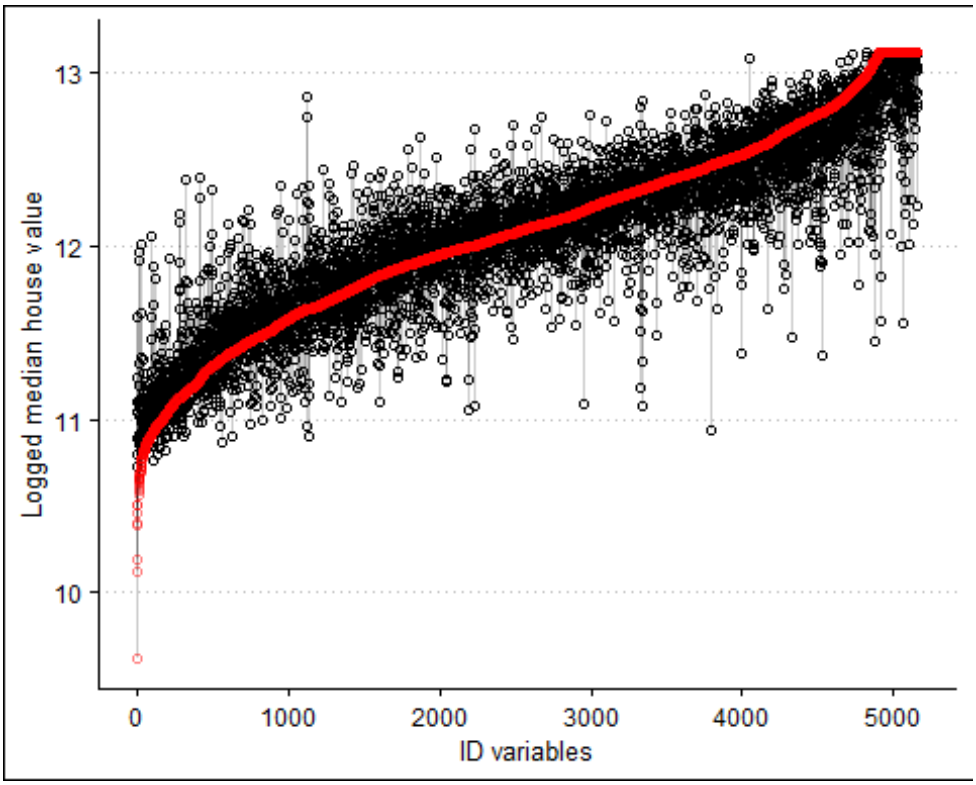
```
## == Workflow ====================================================================
======
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
------
## 4 Recipe Steps
##
## * step_log()
## * step_range()
## * step_ns()
## * step_ns()
##
## -- Model -----------------------------------------------------------------------
------
## K-Nearest Neighbor Model Specification (regression)
##
## Main Arguments:
##   neighbors = 10
##   weight_func = epanechnikov
##
## Computational engine: kknn.
```

```
# Test set predictions
knn_results <-
  knn_fit %>%
  # save pred results
  collect_predictions()

# plot pred v actual
knn_results %>%
  ggplot(aes(x = .pred, y = median_house_value)) +
  geom_point(color = '#004EA1', alpha = 0.25)  +
  geom_abline(intercept = 0, slope = 1, color = 'red') +
  labs(title = 'KNN Regression Results',
       x = 'Predicted Price',
       y = 'Actual Price') +
  theme_clean()
```



Judging by the RMSE, it would indicate that the KNN model is an improvement. Further feature engineering or different modeling may be needed to yield an even better model. Visually, we can confirm that the knn model seems to fit the data better compared to the linear model (pg 23).

```
knn_results <-
  knn_results %>%
    bind_cols(housing_test) %>%
    rename(median_house_value_log = `median_house_value...4`,
           median_house_value = `median_house_value...14`)

knn_results %>%
  arrange(median_house_value_log) %>%
  mutate(id = row_number()) %>%
  ggplot(aes(x = id, y = median_house_value_log)) +
  geom_segment(aes(xend = id, yend = .pred), alpha = .2) +
  geom_point(aes(y = .pred), shape = 1) +
  geom_point(color = "red", shape = 1, alpha = 0.5) +
  labs(x = "ID variables", y = "Logged median house value") +
  theme_clean()
```


13
12
11
10
Logged median house value
0
1000
2000
3000
4000
5000
ID variables