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")) %>% # Replace missing numeric obs with median values  
 step\_dummy(all\_nominal(), -has\_role("ID")) # Code categorical as dummy variables

### 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

# Tuning grid  
tree\_grid <-   
 grid\_regular(  
 cost\_complexity(),   
 tree\_depth(),   
 min\_n(),   
 levels = 4  
 )  
  
tree\_grid

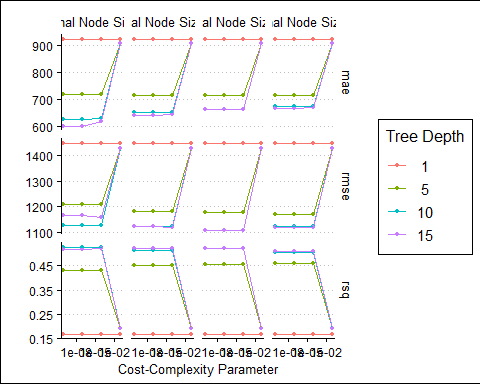
## # A tibble: 64 x 3  
## cost\_complexity tree\_depth min\_n  
## <dbl> <int> <int>  
## 1 0.0000000001 1 2  
## 2 0.0000001 1 2  
## 3 0.0001 1 2  
## 4 0.1 1 2  
## 5 0.0000000001 5 2  
## 6 0.0000001 5 2  
## 7 0.0001 5 2  
## 8 0.1 5 2  
## 9 0.0000000001 10 2  
## 10 0.0000001 10 2  
## # ... with 54 more rows

# 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

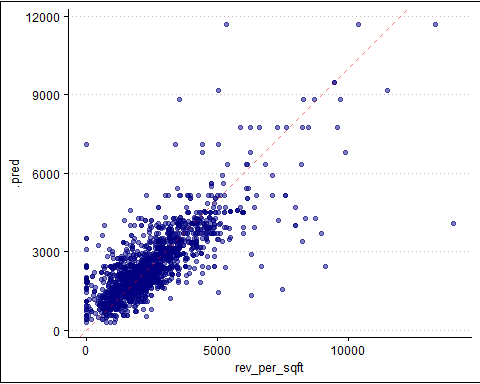
## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: decision\_tree()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 7 Recipe Steps  
##   
## \* step\_mutate()  
## \* step\_rm()  
## \* 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 |

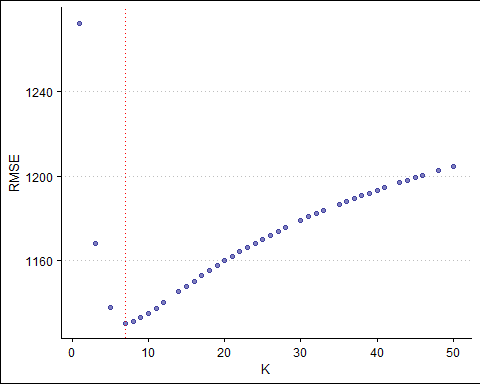
# look at predictions  
final\_resample %>%  
 collect\_predictions() %>%  
 ggplot(aes(rev\_per\_sqft, .pred)) +  
 geom\_point(alpha = 0.5, color = "navy") +  
 geom\_abline(slope = 1, lty = 2, color = "red", alpha = 0.5) +  
 theme\_clean() +  
 coord\_fixed()



### 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")

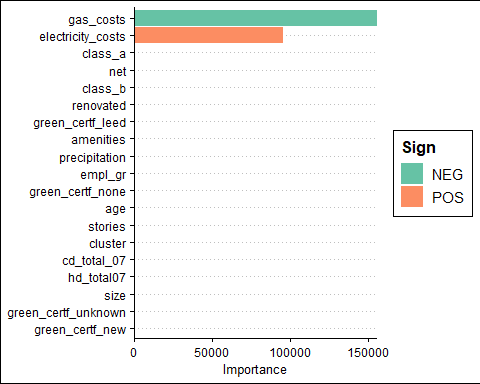
|  |  |  |
| --- | --- | --- |
| .metric | .estimator | .estimate |
| rmse | standard | 973.463 |
| rsq | standard | 0.565 |

### 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 |

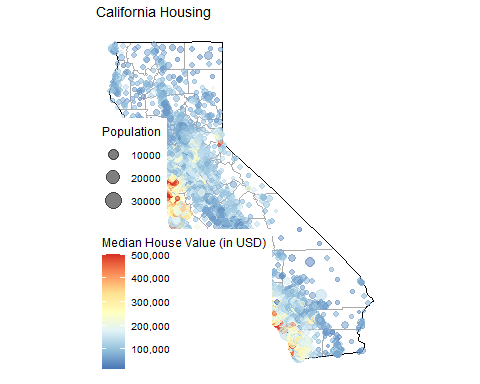
Though we took 3 different approaches, all models are fairly similar in accuracy of predictions. By consulting the vip plots, we find that Green Certification does not appear to be an important feature in determining rent price per sq. ft. Rather, gas costs and electricity costs appear to be the most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

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")



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, median\_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[+]

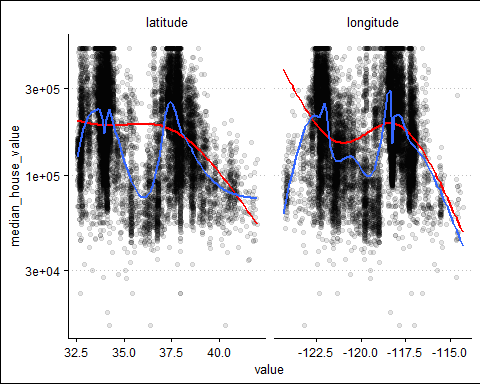
# WF  
lm\_workflow <-   
 workflow() %>%   
 add\_model(lm\_model) %>%   
 add\_recipe(lm\_recipe)  
  
tic()  
lm\_res <-   
 lm\_workflow %>%  
 tune\_grid(resamples = housing\_vfold, grid = spline\_grid)  
toc()

## 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))  
  
  
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()

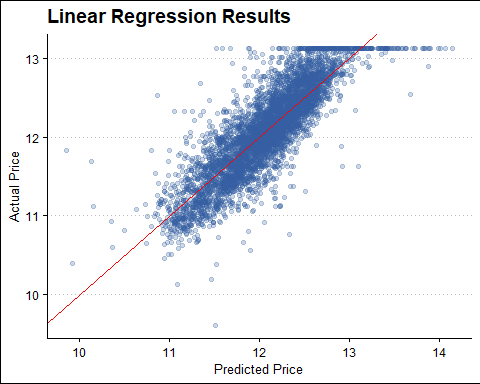
## # A tibble: 2 x 4  
## .metric .estimator .estimate .config   
## <chr> <chr> <dbl> <chr>   
## 1 rmse standard 0.296 Preprocessor1\_Model1  
## 2 rsq standard 0.728 Preprocessor1\_Model1

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")

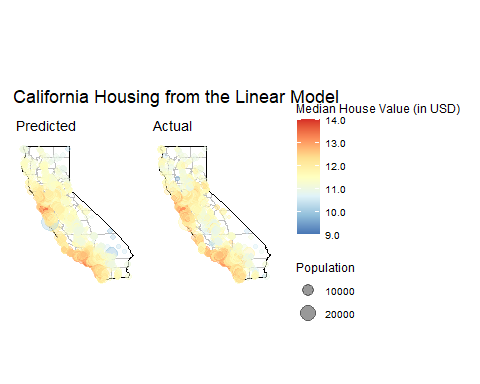


# Obtain test set predictions data frame  
lm\_results <-   
 lm\_fit %>%   
 # save pred results  
 collect\_predictions()  
  
lm\_results <-   
 lm\_results %>%   
 bind\_cols(housing\_test) %>%   
 rename(median\_house\_value\_log = `median\_house\_value...4`,  
 median\_house\_value = `median\_house\_value...14`)

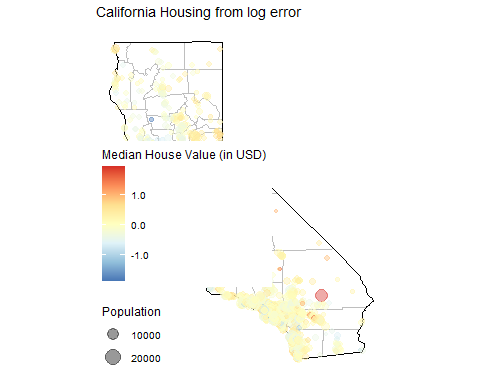
# plot pred v actual  
lm\_results %>%  
 ggplot(aes(x = .pred, y = median\_house\_value\_log)) +  
 geom\_point(color = '#345EA1', alpha = 0.25) +  
 geom\_abline(intercept = 0, slope = 1, color = 'red') +  
 labs(title = 'Linear Regression Results',  
 x = 'Predicted Price',  
 y = 'Actual Price') +   
 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, median\_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 |

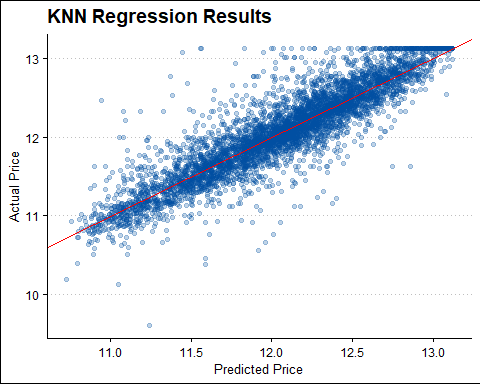
Bit better performance than linear model.

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()

