Connecticut Real Estate Data Analysis and Modeling with Education and Crime Variables

Austin Wong, ID: 921851121 Chenyu Zhu, ID: 922149382
Xujiayun Deng, ID: 922151477 Suyesh Niraula, ID: 922384244
Matthew Wang, ID 922391444

2025-06-09

Introduction

In this project, we explore how crime rates and educational performance metrics across towns in Connecticut affect real estate sale prices. We start with exploratory data analysis and a series of regression models, including polynomial and tree regression. We hypothesized that lower crime rates and higher-performing school districts are associated with higher property values. To evaluate model reliability and predictive power, we will use cross-validation and bootstrapping techniques on our models. Our goal is to see how criminal statistics and educational factors affect the prices of real estate properties in Connecticut.

Datasets

Connecticut Real Estate Data: download real estate data (click)

- The Connecticut real estate data contains data on real estate sales in Connecticut from 2001-2022. The variables include List Year, Date Recorded, Town, Address, Assessed Value, Sale Amount, etc.

2012 Connecticut Crime Data: download data (click)

- The Connecticut Crime Data contains data on the population and number of various crimes in Connecticut towns during 2012.

2011-2012 Connecticut District Performance: download district performance (click)

- This data was created by our team using the district report data from this website link (click) from the 2011-2012 subset. The variables include average SAT scores for the various subjects, average fitness performance, average graduation rate, and average rate of higher education and employment after graduation.

Cleaning

The first step was to do some initial cleaning on the real estate and crime datasets. After loading in the real estate data from my files, I filter it so that it is 2012 data only because the crime data is limited to 2012. I also cleaned the crime dataset, doing things like deleting columns and modifying the data so that its neat. Finally, we join the crime and real estate datasets by their Town/City.

Additionally we joined a dataset containing education performance metrics for each district in Connecticut. We had to do some further cleaning on the combined crime and real estate data and the new district performance data. First, we cleared some unnecessary columns that took up space in the original combined dataset. We took the districts dataset and removed all rows containing `NA` values. I see that the districts dataset classified a lot of numerical variables as character variables, so I had to fix that by changing its

class across many of the columns so that we can perform numerical analysis. After some more miscellaneous cleaning, I was ready to join the real estate and crime data with the district data by `Town`, leaving us with `final_df`.

All code in the cleaning section is commented out because we have the cleaned versions of the dataset saved locally, so we don't want that to be rewritten over just in case.

```
# load the real estate data
# real_estate <- fread("real_estate.csv")
# crime <- read_excel("connecticut_crime_2012.xls")
```

```
# filter for only 2012 sales
# real_estate <- real_estate |>
  # filter(str_ends(`Date Recorded`, "2012"))

# reorganizing crime data
# crime <- crime |>
  # slice(-(1:3))

# update_colnames <- as.character(crime[1,])
# colnames(crime) <- update_colnames

# crime <- crime[-1,]

# joining the datasets
# df <- left_join(real_estate, crime, by = c("Town" = "City"))

# removing NAs
# df <- df |>
  # filter(if_all(15:ncol(df), function(x) !is.na(x)))

# saving the df (commented out so it doesn't resave)

# write_csv(df, "final_connecticut_real_estate.csv")
```

```
# clearing unnecessary cols
# df <- df |>
  # select(-`Non Use Code`, -`Assessor Remarks`, -`OPM remarks`, -`Location`)

# load schools
# districts <- read_csv("connecticut_district_performance2.csv")

# clearing NA and cleaning numerical data
# districts[districts == "#N/A"] <- NA
# districts <- districts |>
  # filter(if_all(everything(), function(x) !is.na(x)))

# districts <- districts |>
  # mutate(across(.cols= 3:ncol(districts), function(x) as.numeric(x)))

# districts <- districts |>
  # mutate(across(contains("%"), function(x) x/100))

# renaming district variables
# districts <- districts |>
  # rename(district_fitness_avg_prop = `Physical Fitness Standard Met %`,
```

```
         # state_fitness_avg_prop = `Physical Fitness STATE %`,
         # district_graduate_avg_prop = `% Graduates 2011`,
         # state_graduate_avg_prop = `% Graduates STATE`,
         # district_higher_ed_employed_avg_prop = `% Higher Education or Employed`,
         # state_higher_ed_employed_avg_prop = `% STATE Higher Education or Employed`)

# delete rows with no info
# df <- df |>
  # filter(`Property Type` != "" & `Residential Type` != "")

# joining and cleaning
# final_df <- left_join(df, districts, by = "Town")
# final_df <- na.omit(final_df)

# saving the df (commented out so it doesnt resave)
# write_csv(final_df, "final_df.csv")
```

1: How do education performance metrics of surrounding school districts affect real estate price?

This question focuses on how the surrounding school districts and education performance metrics of those districts affects the sale amount of houses. Our general hypothesis was that higher valued houses will tend to have better education for children in the town.

Exploratory Data Analysis for education variables

From our summary statistics on the response variable, the most interesting statistics are the minimum and maximum. Since they are so far from the median, this indicates there will be outliers in the response, which we deal with later.

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        0   123000   207000   369673   371000 22250000
```

We find correlation and variance matrix for education variables, to get a look at how the variables are related to the response. SAT Math scores had the highest correlation, but it was still low, with about 0.34. This was a first sign as to how the education variables were not good predictors to the Sale amount. The high variance of sale amount with itself shows that the response is extremely spread out, again hinting to potential outliers.

```
# correlation and variance matrix for education variables
edu_df <- final_df |>
  select(Sale.Amount, 23:ncol(final_df)) |>
  select(-contains("STATE"))|>
  select(-contains("state"))

cor_edu <- as.data.frame(cor(na.omit(edu_df)))
var_edu <- as.data.frame(var(na.omit(edu_df)))

# summary
cat(strwrap(paste("The highest correlated education variable with Sale Amount is:", colnames(cor_edu)[wh
```

```
## The highest correlated education variable with Sale Amount is: SAT 2011 Math
## Average with 0.339543187288714
```

```
cat(strwrap(paste("The lowest correlated education variable with Sale Amount is:", colnames(cor_edu)[wh
```

```
## The lowest correlated education variable with Sale Amount is:
## district_graduate_avg_prop with 0.155592991110994
```

```
cat(strwrap(paste("The highest covariance education variable with Sale Amount is:", colnames(var_edu)[w
```

```
## The highest covariance education variable with Sale Amount is: Sale.Amount with
## 443624534278.774
```

```
cat(strwrap(paste("The lowest covariance education variable with Sale Amount is:", colnames(var_edu)[wh
```

```
## The lowest covariance education variable with Sale Amount is:
## district_higher_ed_employed_avg_prop with 7111.02068941514
```

Basic Modeling

Now, we wanted to do some simple preliminary modeling to get an idea of how the variables are used in some models, and gauge their significance. I used `rsample` to make training and test splits.

We then trained three polynomial regression models (linear, quadratic, cubic) on the training split. By making multiple regression models, I can see which one performs the best against unseen data using test MSE.

For the linear model, every variable is statistically significant at an 0.01 significance level except for SAT Critical Reading Average. For the quadratic model, every variable is significant at a 0.01 significance level. For the cubic model, every variable is significant at a 0.01 significance level except for `district_fitness_avg_prop`, along with its squared and cubed terms.

```
# linear model for education vars
linear <- lm(`Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
               district_graduate_avg_prop+
               district_higher_ed_employed_avg_prop,
             data= train_df
             )
```

The quadratic and cubic models are modeled similar adding the appropriate quadratic and cubic terms where necessary, so their code is not shown for clarity.

Predictions and MSE for the models

Using the models, I then make predictions using the trained models for the testing split.

Using the predictions and the model information, I can then calculate training and test MSE, which will be useful to finding the performance of those models. When assessing these MSE values, they were extremely high, so we knew that something about the data or the models was wrong.

```
pre_linear <- predict(linear, newdata= test_df)
pre_quad <- predict(quadratic, newdata= test_df)
pre_cub <- predict(cubic, newdata= test_df)

# training MSE
MSE_lin_train <- mean(linear$residuals^2)
MSE_quad_train <- mean(quadratic$residuals^2)
```

```r
MSE_cub_train <- mean(cubic$residuals^2)

cat("Linear Training MSE: ", MSE_lin_train, "\n")
```

```
## Linear Training MSE:  399323676307
```

```r
cat("Quadratic Training MSE: ", MSE_quad_train, "\n")
```

```
## Quadratic Training MSE:  347239113882
```

```r
cat("Cubic Training MSE: ", MSE_cub_train, "\n")
```

```
## Cubic Training MSE:  3.37549e+11
```

```r
# test MSE
actual_response <- test_df$Sale.Amount
MSE_lin_test <- mean((actual_response-pre_linear)^2)
MSE_quad_test <- mean((actual_response-pre_quad)^2)
MSE_cub_test <- mean((actual_response-pre_cub)^2)

cat("Linear Test MSE: ", MSE_lin_test, "\n")
```

```
## Linear Test MSE:  3.54916e+11
```

```r
cat("Quadratic Test MSE: ", MSE_quad_test, "\n")
```

```
## Quadratic Test MSE:  304761599467
```

```r
cat("Cubic Test MSE: ", MSE_cub_test, "\n")
```

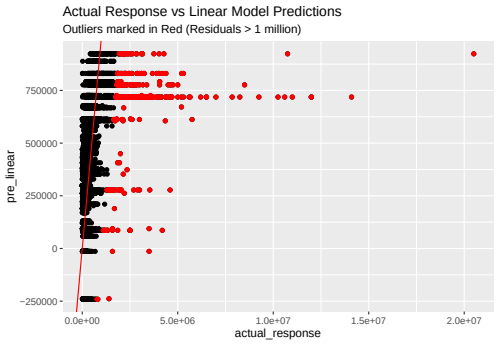
```
## Cubic Test MSE:  295160622099
```

Visual assessment of predictions

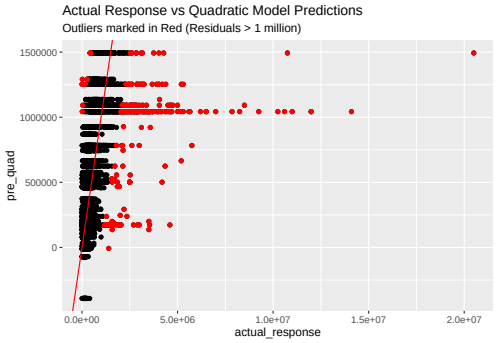
Because of the high MSE, I decided to visually assess the predictions and actual values. I made plots for actual response of the testing split plotted against predictions of each model. Furthermore, in each plot, I made a line representing where predicted would equal actual test values by using a slope 1, intercept 0 line. Finally, I identified the outliers, defined as over 1 million in residual value, by coloring them in red. From this, I discovered a big contributor to MSE was the very large outliers. According to this definition of outliers, there were 325 in the linear model, 250 in the quadratic model, and 254 in the cubic model.

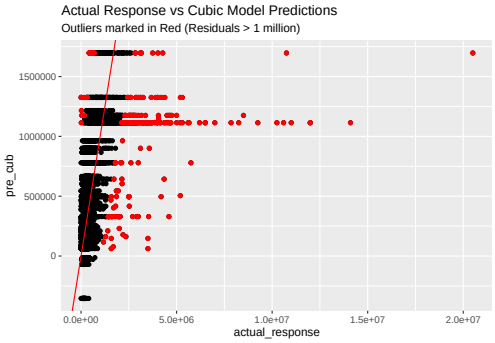
This changed our process somewhat because now we wanted to do our analysis on both the education dataset with outliers and without outliers.

```r
# plotting actuals vs linear model predictions
ggplot(data.frame(actual_response,pre_linear), aes(x=actual_response,y=pre_linear))+
  geom_point()+
  geom_abline(slope=1, intercept = 0, color = "red") +
  geom_point(subset(data.frame(actual_response, pre_linear), abs(actual_response-pre_linear)> 1000000),
             mapping = aes(x=actual_response,y=pre_linear),
             color = "red") +
  labs(
    title= "Actual Response vs Linear Model Predictions",
    subtitle = "Outliers marked in Red (Residuals > 1 million)"
  )
```



The other model's visualizations are plotted similarly, so the code is not shown for clarity.





Bootstrapping evaluation

Since the regression models made are subject to the sample data we had, our next step was to train and test those models but using 100 bootstrapped datasets so that our average MSE would not be subject to so much randomness.

This function was made to take the formula for the model (linear, quadratic, cubic), the data we were going to use (`final_df`), and the number of bootstraps to be done. It then uses the bootstrapping function from `rsample` to create 100 bootstraps. By iterating through the splits in the `boots` variable, we can train models using the `formula` on the bootstrap training split and test predictions using the bootstrap testing split. After gathering all these MSE's into a list, average training and testing MSE are computed.

```
# function for bootstrap evaluations of models

bootstrap_evaluation <- function(n_bootstraps =100, formula, data, response) {
  set.seed(2025)

  boots <- bootstraps(data, times = n_bootstraps)

  mse_train_vec <- numeric(n_bootstraps)
  mse_test_vec <- numeric(n_bootstraps)

  for (i in 1:n_bootstraps) { # training and finding statistics using bootstraps
    split <- boots$splits[[i]] # iterate through splits
    training <- analysis(split)
    testing <- assessment(split)

    model <- lm(formula, data = training) # fit model

    predictions <- predict(model, newdata= testing) # make predictions
```

```
    actual_test_response <- testing[[response]]

    # finding MSEs
    MSE_train <- mean(model$residuals^2)
    MSE_test <- mean((actual_test_response- predictions)^2)



    # putting in a vector
    mse_train_vec[i] <- MSE_train
    mse_test_vec[i] <- MSE_test

  }

  # statistics for MSE for all bootstraps
  avg_mse_train <- mean(mse_train_vec)
  avg_mse_test <- mean(mse_test_vec)


  # 95 % confidence intervals
  CI_mse_train <- quantile(mse_train_vec, c(0.025,0.975))
  CI_mse_test <- quantile(mse_test_vec, c(0.025,0.975))

  return(list(
    avg_MSE_train = avg_mse_train,
    avg_MSE_test = avg_mse_test
  ))
}
```

Bootstrapping the full dataset

In this step, we do the bootstrapping for the dataset with outliers using our formula defined above for each polynomial formula. The results were similar to our previous case, where the MSE's were still extremely high. This was expected, as the results were simply the average over 100 bootstrapped samples of each model.

```
# bootstrap evaluations for linear model

boot_lin_model <- bootstrap_evaluation(formula = `Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
               district_graduate_avg_prop+
               district_higher_ed_employed_avg_prop,
             data = final_df, response ="Sale.Amount")

paste("Average training MSE for with-outliers data - linear:", boot_lin_model[[1]])
```

```
## [1] "Average training MSE for with-outliers data - linear: 379038315538.047"
```

```
paste("Average testing MSE for with-outliers data -  linear:", boot_lin_model[[2]])
```

```
## [1] "Average testing MSE for with-outliers data -  linear: 376568207092.511"
```

The quadratic and cubic models were bootstrap evaluated in a similar way except with the quadratic and cubed terms, so their code is not shown for clarity.

```
## [1] "Average training MSE for with-outliers data -  quadratic: 327697237398.617"
## [1] "Average testing MSE for with-outliers data -  quadratic: 326081618752.411"
## [1] "Average training MSE for with-outliers data -  cubic: 317967680320.155"
## [1] "Average testing MSE for with-outliers data -  cubic: 316686876196.598"
```

Visually assessing model performance with bootstraps Next we wanted to visually see what the bootstrapped models performance looked like. Our first step was to gather the MSE results in the previous step in a dataframe. Adding an RMSE column and pivoting was necessary to get the results we wanted and to make the dataframe in a form easier to plot.

```
# gather MSE into a dataframe
full_boot_performance <- data.frame(
  model_order = c(1,2,3),
  MSE_train = c(boot_lin_model$avg_MSE_train,
                boot_quad_model$avg_MSE_train,
                boot_cub_model$avg_MSE_train),
  MSE_test = c(boot_lin_model$avg_MSE_test,
               boot_quad_model$avg_MSE_test,
               boot_cub_model$avg_MSE_test)


)

# add a RMSE col
full_boot_performance <- full_boot_performance |>
  mutate(RMSE_train = sqrt(MSE_train),
         RMSE_test = sqrt(MSE_test))

# pivot
full_boot_performance <- full_boot_performance |>
  pivot_longer(c(RMSE_train, RMSE_test), names_to = "split", values_to = "RMSE")
```

In the plot, we can see that the "elbow" occurs at the quadratic model. If our priority was to save computational time, then we would choose the quadratic model, as the cubic model offers less benefits for more computation time. However, the computation was relatively quick for all models, so the cubic model works the best for RMSE.

```
full_boot_performance |>
  ggplot(aes(x = model_order, y=RMSE, color = split)) +
  geom_point() +
  geom_line() +
  labs(title = "RMSE for different order polynomial regression",
       subtitle = "Response: Sale Amount, Explanatory: School District Variables",
       x = "Order of the Regression",
       color = "Data Split")
```

Bootstrap evaluations on no-outlier data

In this step, the same bootstrap is done, but using data with no outliers. To do this, we must first make a `clean_df`, which was made with the IQR definition of outliers. By filtering for `Sale.Amount` within the range of acceptable values, we created our new dataset.

```
# removing outliers from final_df using IQR def of outliers
Q1_response <- quantile(final_df$Sale.Amount, 0.25)
Q3_response <- quantile(final_df$Sale.Amount, 0.75)
IQR_response <- Q3_response-Q1_response

upper_outlier <- Q3_response+ 1.5* IQR_response
lower_outlier <- Q1_response- 1.5* IQR_response

clean_df <- final_df |>
  filter(Sale.Amount >= lower_outlier & Sale.Amount <= upper_outlier)
```

Using the same bootstrap function defined earlier, we just did the same thing except the data was `clean_df`. Our results showed a great improvement, with much less MSE, showing that our previous hypothesis of outliers causing problems was part of the problem. However, the MSE is still quite high.

```
# bootstrap for the linear model with clean data
clean_boot_lin <- bootstrap_evaluation(formula = `Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
               district_graduate_avg_prop+
               district_higher_ed_employed_avg_prop,
             data = clean_df,
             response ="Sale.Amount")

paste("Average training MSE for no-outliers data -  linear:", clean_boot_lin[[1]])
```

```
## [1] "Average training MSE for no-outliers data -  linear: 19640174638.2787"
```

```
paste("Average testing MSE for no-outliers data -  linear:", clean_boot_lin[[2]])
```

```
## [1] "Average testing MSE for no-outliers data -  linear: 19636788985.5619"
```

The quadratic and cubic bootstraps are evaluated similar to the linear case, so their code is not included for clarity.

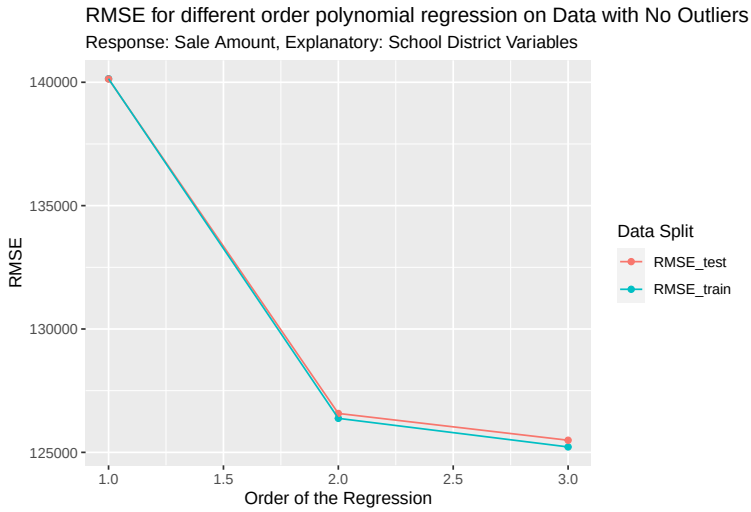
```
## [1] "Average training MSE for no-outliers data -  quadratic: 15971511152.4111"
## [1] "Average testing MSE for no-outliers data -  quadratic: 16021875422.2777"
## [1] "Average training MSE for no-outliers data -  cubic: 15680092945.3891"
## [1] "Average testing MSE for no-outliers data -  cubic: 15748126519.5903"
```

Visually assessing bootstrapped model performance on cleaned data A similar process to the previous was done to gather the MSE in a data frame, only this time it was the MSE from the clean data bootstraps.

```
# gather MSE into a dataframe
clean_boot_performance <- data.frame(
  model_order = c(1,2,3),
  MSE_train = c(clean_boot_lin$avg_MSE_train,
                clean_boot_quad$avg_MSE_train,
                clean_boot_cub$avg_MSE_train),
  MSE_test = c(clean_boot_lin$avg_MSE_test,
               clean_boot_quad$avg_MSE_test,
               clean_boot_cub$avg_MSE_test)


)

# add a RMSE col
clean_boot_performance <- clean_boot_performance |>
  mutate(RMSE_train = sqrt(MSE_train),
         RMSE_test = sqrt(MSE_test))

# pivot
clean_boot_performance <- clean_boot_performance |>
  pivot_longer(c(RMSE_train, RMSE_test), names_to = "split", values_to = "RMSE")
```

Looking at the visualization, we see that the "elbow" once again occurs at the quadratic model, but for the same reasons as the previous bootstrapping, the computation time is only marginally longer, so the best model in this case is also the cubic. One interesting thing to note is that the test RMSE is actually higher than the train, and this disparity is more apparent as the model gets more flexible, suggesting a bit of overfitting.

```
clean_boot_performance |>
  ggplot(aes(x = model_order, y=RMSE, color = split)) +
  geom_point() +
  geom_line() +
  labs(title = "RMSE for different order polynomial regression on Data with No Outliers",
       subtitle = "Response: Sale Amount, Explanatory: School District Variables",
       x = "Order of the Regression",
       color = "Data Split")
```



Evaluating polynomial models with cross validation

The next method we wanted to use was cross validation to evaluate the models further. By using a second method of evaluation on our models, we can compare with the bootstrap to see if there's a difference. If the findings are similar, it means our findings are likely to be true. We used the `caret` library to do the cross validation. A small function was made that took the model formula, data, and number of folds to be done incorporating the `caret` library's functions to perform a cross validation. This way, we can do cross validation quickly by just replacing the model formula and make any small tweaks.

```
# function for k fold CV

k_CV <- function(formula, data, folds= 10) {
  set.seed(2025)
  train_method <- trainControl(method = "cv", number = folds)
  cv_model <- train(formula, data= data,
                    method = "lm",
                    trControl = train_method
                    )
  return(mean(cv_model$resample$RMSE))
}
```

CV for models using with-outlier data

Using the function defined above, we performed a 10 fold cross validation on the linear, quadratic and cubic models. Also, this was with the full data, including outliers. As expected, the MSE values of each model were extremely similar to their corresponding bootstrapped versions, although slightly smaller for the CV MSE.

```
# full data CV for linear
CV_lin <- k_CV(`Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
               district_graduate_avg_prop+
               district_higher_ed_employed_avg_prop,
               data = final_df)

paste("RMSE Linear for with-outliers data:", CV_lin)
```

```
## [1] "RMSE Linear for with-outliers data: 611727.60497066"
```

The quadratic and cubic models were evaluated using k fold CV similar to the linear case, so their code is not included for clarity.

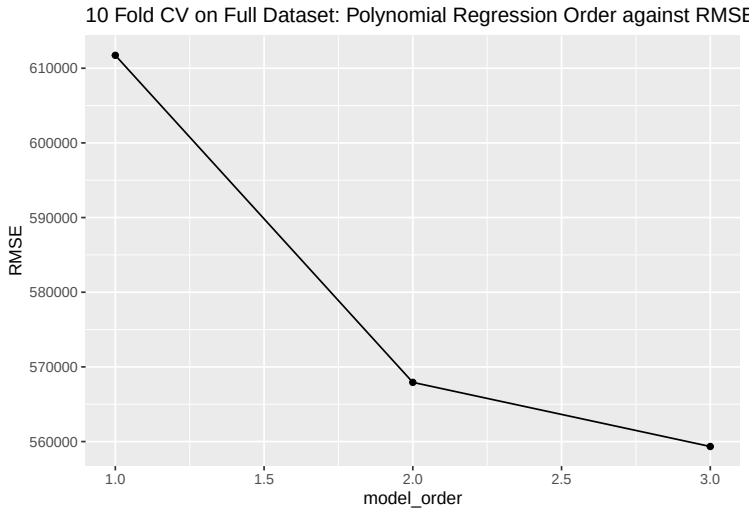
```
## [1] "RMSE Quadratic for with-outliers data: 567928.444565548"
```

```
## [1] "RMSE Cubic for with-outliers data: 559338.092799786"
```

Visual assessment of CV on full data After gathering the RMSE data into a data frame, we made a plot with the model polynomial order against the RMSE. Once again, the shift from linear to quadratic made the biggest decrease in RMSE, but since the cubic model didn't take much longer to compute, it is a better and feasible model according to RMSE.

```
full_CV_performance <- data.frame(
  model_order = c(1,2,3),
  RMSE = c(as.numeric(CV_lin[1]),
           as.numeric(CV_quad[1]),
           as.numeric(CV_cub[1]))
)
```

```
full_CV_performance |>
  ggplot(aes(x= model_order, y=RMSE))+
  geom_point()+
  geom_line() +
  labs(
    title = "10 Fold CV on Full Dataset: Polynomial Regression Order against RMSE"
  )
```



CV evaluation of models using no-outlier data

We did the exact same CV process with the same function for each model, except using `clean_df` to find the RMSE of the no outlier dataset. The RMSE's between the bootstrapped and CV are extremely close.

```
# cleaned data CV for linear
CV_clean_lin <- k_CV(`Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
```

```
                district_graduate_avg_prop+
                district_higher_ed_employed_avg_prop,
                data = clean_df)

paste("RMSE Linear for no-outliers data:", CV_clean_lin)
```

```
## [1] "RMSE Linear for no-outliers data: 140136.749378655"
```

The k fold evaluation is done similar for the quadratic and cubic case using cleaned data, so code is not included for clarity of the report.

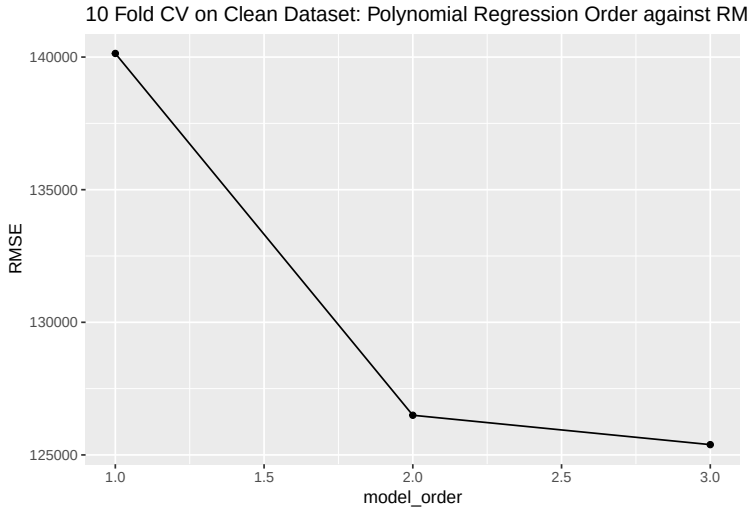
```
## [1] "RMSE Quadratic for no-outliers data: 126495.819394714"
```

```
## [1] "RMSE Cubic for no-outliers data: 125388.125964851"
```

Visualizing model performance for clean data using CV Doing another visualization of the model order plotted against RMSE shows the same results we've been seeing for the previous CV and the bootstraps. Thus, the cubic model is still the best one in terms of RMSE and feasibility of computation.

```
clean_CV_performance <- data.frame(
  model_order = c(1,2,3),
  RMSE = c(as.numeric(CV_clean_lin[1]),
           as.numeric(CV_clean_quad[1]),
           as.numeric(CV_clean_cub[1]))
)
```

```
clean_CV_performance |>
  ggplot(aes(x= model_order, y=RMSE))+
  geom_point()+
  geom_line() +
  labs(
    title = "10 Fold CV on Clean Dataset: Polynomial Regression Order against RMSE"
  )
```



Tree based methods

Our final method was to use a completely different modeling method: the tree regression. We used this to get a completely different model that was not polynomial in order to compare different modeling methods. Using the `rpart` library's tree creation function, a bootstrapping function was created similar to the previous

one, but incorporating rpart's tree creation ability. Otherwise the code remains similar, reporting MSE for test and training.

```
# function for passing many bootstrapped samples into the regression tree

boot_tree <- function(n_bootstraps =100,formula, data, response) {
  set.seed(2025)

  boots <- bootstraps(data, times = n_bootstraps)

  mse_train_vec <- numeric(n_bootstraps)
  mse_test_vec <- numeric(n_bootstraps)

  for (i in 1:n_bootstraps) { # training and finding statistics using bootstraps
    split <- boots$splits[[i]] # iterate through splits
    training <- analysis(split)
    testing <- assessment(split)


    tree <- rpart(formula, data = training, method = "anova")
    predict_train_tree <- predict(tree, newdata = training)
    predict_test_tree <- predict(tree, newdata = testing)

    MSE_train_tree <- mean((training[[response]]- predict_train_tree)^2)
    MSE_test_tree <- mean((testing[[response]]- predict_test_tree)^2)

    mse_train_vec[i] <- MSE_train_tree
    mse_test_vec[i] <- MSE_test_tree
  }
  avg_mse_train_tree <- mean(mse_train_vec)
  avg_mse_test_tree <- mean(mse_test_vec)

  CI_train_tree <- quantile(mse_train_vec, c(0.025, 0.975))
  CI_test_tree <- quantile(mse_test_vec, c(0.025, 0.975))

  return(list(MSE_train_tree = avg_mse_train_tree, MSE_test_tree = avg_mse_test_tree))

}
```

Using our function, we applied the bootstrap tree model onto the dataset with outliers, and also without outliers. Surprisingly, testing MSE was a good chunk lower than our cubic model's test MSE using bootstrap for the full dataset. However, testing MSE was not much lower than our cubic model's test MSE with the no-outliers dataset. Our interpretation of this statistic is that the MSE for the clean data can't get much better anymore, since our bootstrap, k fold CV, and tree seem to place the testing MSE around the region of 15 billion.

```
# bootstrapping tree regression using with-outlier data
boot_tree_full <- boot_tree(
          formula =`Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
               district_graduate_avg_prop+
               district_higher_ed_employed_avg_prop,
```

```
          data= final_df,
          response = "Sale.Amount")

paste("Average training MSE for with-outliers data tree:", boot_tree_full[[1]])
```

```
## [1] "Average training MSE for with-outliers data tree: 288485232840.727"
```

```
paste("Average testing MSE for with-outliers data tree:", boot_tree_full[[2]])
```

```
## [1] "Average testing MSE for with-outliers data tree: 288405371893.576"
```

```
# bootstrapping tree regression using no-outlier data
boot_tree_clean <- boot_tree(n_bootstraps =100,
          formula =`Sale.Amount`~
               district_fitness_avg_prop +
               `SAT 2011 Math Average`+
               `SAT 2011 Critical Reading Average`+
               `SAT 2011 Writing Average`+
               district_graduate_avg_prop+
               district_higher_ed_employed_avg_prop,
          data= clean_df,
          response = "Sale.Amount")

paste("Average training MSE for no-outliers data tree:", boot_tree_clean[[1]])
```

```
## [1] "Average training MSE for no-outliers data tree: 15496660163.8351"
```

```
paste("Average training MSE for no-outliers data tree:", boot_tree_clean[[2]])
```

```
## [1] "Average training MSE for no-outliers data tree: 15607604223.5944"
```

Visually assessing full vs clean data with tree regression

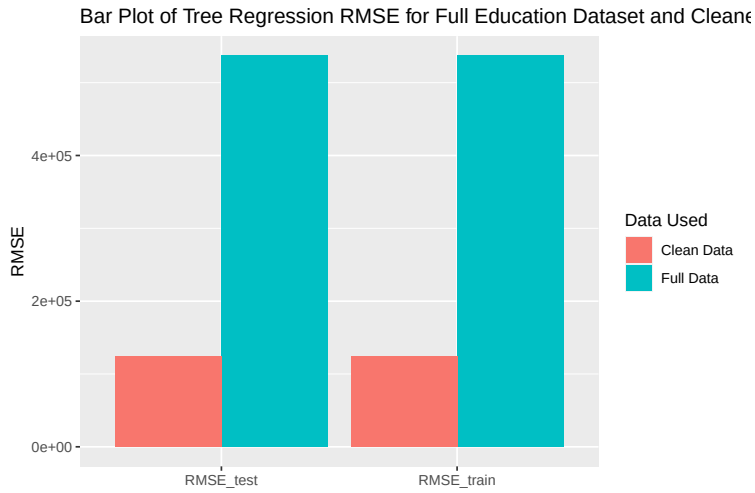
We put the tree MSEs from both the dataset with outliers and the dataset without into a data frame, and modified it to calculate RMSE and be in a good shape to make a plot. We then used this data frame to make a bar plot. The bar plot shows how much the having no outliers reduces the RMSE of the models in both the training and testing setting.

```
# create data frame

tree_df <- data.frame(
  model = c("Full Data", "Clean Data"),
  MSE_train <- c(boot_tree_full$MSE_train_tree,
                 boot_tree_clean$MSE_train_tree),
  MSE_test <- c(boot_tree_full$MSE_test_tree,
                boot_tree_clean$MSE_test_tree)

)

tree_df <- tree_df |>
  mutate(RMSE_train = sqrt(MSE_train),
         RMSE_test = sqrt(MSE_test))

tree_df <- tree_df |>
  pivot_longer(cols = c(RMSE_train, RMSE_test),
               names_to = "split",
               values_to = "RMSE")
```

```
tree_df
```

```
## # A tibble: 4 x 5
##   model      MSE_train....c.boot_tree_full.MSE_train_tree..boot_tree_clean.MSE_train_tree. MSE_test.
##   <chr>                                                                                        <dbl>
## 1 Full Data                                                                          288485232841.
## 2 Full Data                                                                          288485232841.
## 3 Clean Data                                                                          15496660164.
## 4 Clean Data                                                                          15496660164.
```

```
tree_df |>
  ggplot(aes(x= split, y = RMSE, fill = model))+
  geom_bar(stat= "identity", position = "dodge") +
  labs(
    title = "Bar Plot of Tree Regression RMSE for Full Education Dataset and Cleaned Education Data",
    x = "",
    fill = "Data Used"
  )
```



Limitations

We believe the reason our models have such high MSE despite removing outliers and applying different evaluation and modeling techniques is because our education variables just don't explain the data well. With our data, the error just seems to hit a plateau where the MSE is around 14-15 billion no matter what technique is used. One problem is may be due to the fact that education variables are town based, leaving no specificity for individual houses within the same town to be explained. Another problem where this high MSE may arise is that perhaps many people do not consider the education opportunities around them when making a decision to purchase a house.

2: How different crime variables affect the real estate sales from 2012 across all towns in Connecticut, and which are the most important?

We specially aim at Violent crime and the Property crime and how they affect the sell amount and Asset value. Firstly we use two data sets, the first one is Real Estate Sales 2001-2022 GL on the website: (https://catalog.data.gov/dataset/real-estate-sales-2001-2018). And data set is Crime in the United States 2012 on the website: (https://ucr.fbi.gov/crime-in-the-u.s/2012/crime-in-the-u.s.-2012/tables/8tabledatadecpdf/tab

le-8-state-cuts/table_8_offenses_known_to_law_enforcement_by_connecticut_by_city_2012.xls). We then combine them together and do basic Summary Statistics and Correlation matrix and Variance matrix. We highlighted the correlation and variance of selected variables with sale amount.

EDA with correlation and variance matrices and basic modeling

```
df <- read.csv("final_connecticut_real_estate.csv", stringsAsFactors = FALSE)
names(df) <- tolower(names(df))
crime_keywords <- c("crime", "murder", "rape", "robbery", "assault", "burglary", "larceny", "vehicle",
crime_vars <- grep(paste(crime_keywords, collapse = "|"), names(df), value = TRUE)
target_vars <- c("assessed.value", "sale.amount")
vars_needed <- c(crime_vars, target_vars)
df_clean <- na.omit(df[, vars_needed])
df_clean <- as.data.frame(sapply(df_clean, as.numeric))
options(width = 200)
cor_matrix <- round(cor(df_clean), 4)
var_matrix <- round(var(df_clean), 2)
list(paste("Correlation of sale amount and violent crime:", as.data.frame(cor_matrix)$violent.crime[12]]
     paste("Correlation of sale amount and burglary:",
           as.data.frame(cor_matrix)$burglary[12]))
```

```
## [[1]]
## [1] "Correlation of sale amount and violent crime: -0.0787"
##
## [[2]]
## [1] "Correlation of sale amount and burglary: -0.1062"
```

```
list(paste("Covariance of Sale amount and violent crime:", as.data.frame(var_matrix)$violent.crime[12])
     paste("Covariance of sale amount and burglary:",
           as.data.frame(var_matrix)$burglary[12]))
```

```
## [[1]]
## [1] "Covariance of Sale amount and violent crime: -42504064.98"
##
## [[2]]
## [1] "Covariance of sale amount and burglary: -44825196.04"
```

```
formula_sale <- as.formula(paste("sale.amount ~", paste(crime_vars, collapse = " + ")))
model_sale <- lm(formula_sale, data = df_clean)

print(list(coefficients =summary(model_sale)$coefficients, r_squared = summary(model_sale)$r.squared))
```

```
## $coefficients
##                                        Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)                          626654.3484 12197.6452 51.375027 0.000000e+00
## violent.crime                          -612.5362   125.2209 -4.891646 1.005699e-06
## murder.and.nonnegligent.manslaughter  13288.2916  9550.6706  1.391346 1.641318e-01
## forcible.rape                           357.5327   282.9449  1.263612 2.063802e-01
## robbery                                3021.9383   301.1484 10.034715 1.179919e-23
## property.crime                          455.2034   186.8253  2.436519 1.483578e-02
## burglary                              -1768.5532   253.2224 -6.984190 2.931255e-12
## larceny..theft                         -529.4137   202.0754 -2.619881 8.800947e-03
## arson                                 -2250.1995  1521.1026 -1.479321 1.390661e-01
##
## $r_squared
```

```
## [1] 0.02147505
```

```
formula_assess <- as.formula(paste("assessed.value ~", paste(crime_vars, collapse = " + ")))

model_assess <- lm(formula_assess, data = df_clean)

print(list(coefficients = summary(model_assess)$coefficients, r_squared = summary(model_assess)$r.squar
```

```
## $coefficients
##                                          Estimate Std. Error    t value      Pr(>|t|)
## (Intercept)                          598767.11669 23681.1975 25.2844949 1.969431e-139
## violent.crime                            61.91586   243.1109  0.2546816  7.989710e-01
## murder.and.nonnegligent.manslaughter -74169.95713 18542.2114 -4.0000600  6.349470e-05
## forcible.rape                          1900.55001   549.3253  3.4597898  5.414299e-04
## robbery                                2492.95778   584.6665  4.2638973  2.015723e-05
## property.crime                         1278.69828   362.7131  3.5253710  4.235886e-04
## burglary                              -2786.90856   491.6202 -5.6688239  1.452433e-08
## larceny..theft                        -1359.41512   392.3207 -3.4650610  5.309323e-04
## arson                                 11377.12562  2953.1544  3.8525332  1.171716e-04
##
## $r_squared
## [1] 0.004936888
```

According to the Correlation Matrix We find that higher violent crime levels are negatively correlated with both sale price and assessed value, but only weakly. And for the Regression Results we can find that Violent crime has a significant negative impact on sale price (for this model sale.amount ~ violent.crime + other crimes). And for the other model (assessed.value ~ violent.crime + other crimes) we can see that Violent crime is not a significant predictor for assessed value. There is a trend between violent crime and sale price, but the Assessed value lacks an obvious correlation. Based on the Correlation Matrix, we can observe that violent crime is negatively correlated with the sale amount (-0.0787), while its correlation with the assessed value is very low (-0.0209). The analysis based on the Variance Matrix indicates that the variance of violent crime is much smaller than that of the sale amount, suggesting that the fluctuations in crime data are small, but the fluctuations in the sale amount are very large. At the same time we checked the correlation coefficient with the sale.amount and burglary, it is around -0.1062 and the Regression coefficient estimate of burglary is also very high around -1768.6. So we tried to do some more tests on the burglary. Since violent crime has collinearity with several other variables, we will conduct an MSE analysis separately to examine the impact of violent crime and burglary. And we checked the data find that most of the crimes have a very slight influence on the Assessed value so we can say that the confidence of buyers can be affected by the crimes happening in that region but will not have a strong impact on how sellers and second hand selling institutions to assume a price they think is very reasonable so we will aiming at the relation between crimes and sale amount and burglary and sale amount basing on the Polynomial Model.

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        0   125075   217000   426396   395000 57500000
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    11.0    30.0   236.3   272.0  1870.0
```

10 fold Cross validation on violent crime variable

```
#Before we do the cross validation, by the degree of regression, we can get three models in here

model_deg1 <- lm(Sale.Amount ~ Violent.crime, data = data)

model_deg2 <- lm(Sale.Amount ~ poly(Violent.crime, 2, raw = TRUE), data = data)
```

```r
model_deg3 <- lm(Sale.Amount ~ poly(Violent.crime, 3, raw = TRUE), data = data)


#calculate the mean square error here, obtain the training mse
train_MSE <- c(
  deg1 = mean(resid(model_deg1)^2),
  deg2 = mean(resid(model_deg2)^2),
  deg3 = mean(resid(model_deg3)^2)
)
```

Now we focusing on the Sale amount and Violent crime, we do the cross validation, by the degree of regression and calculate the mean square error here, obtain the training MSE basing on the result we get that from deg1 to deg2 both CV and MSE become lower and from deg2 to deg3 MSE remain almost the same so we want to choose deg2 for it is the most stable one.

```r
#use the kfold method here
library(caret)
set.seed(2025)
folds <- createFolds(data$Sale.Amount, k = 10)
cv_MSE_manual <- numeric(length = 3)
names(cv_MSE_manual) <- c("deg1","deg2","deg3")

# In here I used sapply function, to calculate all 3 groups, to make sure the data only contains the tr

#here we need to let lm() functions only knows the training set first, so the data wont get overfitted

cv_MSE <- sapply(1:3, function(deg) {
  mse_per_fold <- sapply(folds, function(test_idx) {
    fit  <- lm(
      Sale.Amount ~ poly(Violent.crime, deg, raw = TRUE),
      data = data[-test_idx, ]
    )
    # now do predictions on the testing set and calculate the testing mse
    pred <- predict(fit, data[test_idx, ])
    mean((data$Sale.Amount[test_idx] - pred)^2)
  })
  mean(mse_per_fold)
})
names(cv_MSE) <- paste0("deg", 1:3)

# also here we will compare the training and testing mse
compare <- data.frame(
  train_MSE = train_MSE,
  cv_MSE    = cv_MSE
)
print(compare)
```

```
##         train_MSE       cv_MSE
## deg1 1.216901e+12 1.217140e+12
## deg2 1.209655e+12 1.210071e+12
## deg3 1.209474e+12 1.210056e+12
```

From all the output listed above, we are able to see data with degrees of 2 have relatively the smallest MSE, which means this is the best model fit.

```r
library(boot)
boot_fn <- function(df, idx) {
  d_boot <- df[idx, ]
  fit    <- lm(Sale.Amount ~ poly(Violent.crime, 2, raw = TRUE),
               data = d_boot)
  pred   <- predict(fit, newdata = df)
  mean((df$Sale.Amount - pred)^2)
}
boot_out <- boot(data = data, statistic = boot_fn, R = 100)

#boot out the original mse
boot_out$t0
```

```
## [1] 1.209655e+12
```

```r
# get the confidence interval and the means

mean(boot_out$t)
```

```
## [1] 1.209783e+12
```

```r
paste0("confidence interval:", " (", boot.ci(boot_out, type = "perc")$percent[4], ", ",boot.ci(boot_out
```

```
## [1] "confidence interval: (1209658607761.48, 1210129317948.82)"
```

```r
#from the output of confidence interval, the mse is pretty stable and reliable
```

```r
boot_coef <- function(df, i) {
  d <- df[i, ]
  fit <- lm(Sale.Amount ~ poly(Violent.crime, 2, raw=TRUE), data=d)
  coef(fit)[2]
}
library(boot)
b <- boot(data, boot_coef, R=100)

paste0("confidence interval:", " (", boot.ci(b, type = "perc")$percent[4], ", ",boot.ci(b, type = "perc
```

```
## [1] "confidence interval: (-1179.84191294349, -902.216576561887)"
```

```r
# look at the confidence interval, they are both smaller than 0, proved that they are negatively relate
```

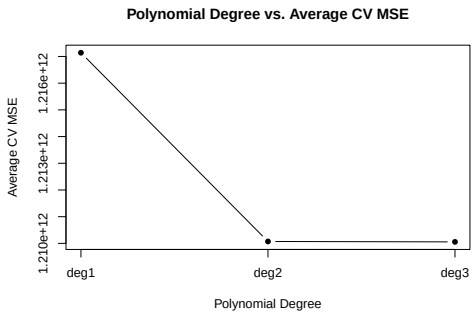
Then we are doing the bootstrap to test the error stability. The 95% confidence interval is [1.209666,1.210187]* 10^12 so we can find that the CI is very narrow and the CV MSE(deg2) is right at the middle of the CI so we can say that deg2 is very stable and we can trust it. For the second bootstrap, I am using it to prove the negative relationship between the violent crime and housing prices. So based on the output of the second confidence interval, which is [-1182,-916], they are both significantly smaller than 0 which means every time there is a increase in violent crimes, there will be a decrease in price.

```r
mse_fold_mat <- sapply(1:3, function(deg) {
  sapply(folds, function(test_idx) {
    # here for the martix, we need to calculate every folds mse
    train_df <- data[-test_idx, ]
    test_df  <- data[test_idx, ]

    fit <- lm(Sale.Amount ~ poly(Violent.crime, deg, raw = TRUE),
              data = train_df)
    pred <- predict(fit, test_df)
    mean((test_df$Sale.Amount - pred)^2)
```

```
  })
})
```

Visually assessing CV methods for violent crime

```
# the graph for Polynomial Degree vs. Average CV MSE (line plot)

avg_mse <- cv_MSE
plot(seq_along(avg_mse), avg_mse, type = "b", pch = 16,
     xaxt = "n",
     xlab = "Polynomial Degree",
     ylab = "Average CV MSE",
     main = "Polynomial Degree vs. Average CV MSE")
axis(1, at = 1:length(avg_mse), labels = names(avg_mse))
```



From this plot we can find the relation between Polynomial Degree and Average CV MSE. From deg1 to deg2, the MSE decreased significantly, indicating that the deg2 polynomial model fits the data better than the deg1 model. From deg2 to deg3, the MSE change is very small and almost flat, indicating that increasing to a third-order polynomial does not bring a significant performance improvement. So we can see that the best model should be deg2 for low MSE and not import more complexity.

10 fold CV evaluation on models trained on burglary variable and visual assessment

```
library(caret)
library(boot)
set.seed(2025)
data <- read.csv("final_connecticut_real_estate.csv", stringsAsFactors = FALSE)
names(data) <- tolower(names(data))

summary(data$burglary)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     4.0    45.0   111.0   265.4   274.0  1451.0
```

```
model_deg1 <- lm(sale.amount ~ burglary, data = data)

model_deg2 <- lm(sale.amount ~ poly(burglary, 2, raw = TRUE), data = data)

model_deg3 <- lm(sale.amount ~ poly(burglary, 3, raw = TRUE), data = data)

train_MSE <- c(
```

```
  deg1 = mean(resid(model_deg1)^2),
  deg2 = mean(resid(model_deg2)^2),
  deg3 = mean(resid(model_deg3)^2)
)


set.seed(2025)
folds <- createFolds(data$sale.amount, k = 10)
cv_MSE <- sapply(1:3, function(deg) {
  mse_per_fold <- sapply(folds, function(test_idx) {
    fit  <- lm(
      sale.amount ~ poly(burglary, deg, raw = TRUE),
      data = data[-test_idx, ]
    )
    pred <- predict(fit, data[test_idx, ])
    mean((data$sale.amount[test_idx] - pred)^2)
  })
  mean(mse_per_fold)
})
names(cv_MSE) <- paste0("deg", 1:3)


compare <- data.frame(
  train_MSE = train_MSE,
  cv_MSE    = cv_MSE
)
print(compare)
```

```
##         train_MSE       cv_MSE
## deg1 1.210662e+12 1.210861e+12
## deg2 1.200632e+12 1.200994e+12
## deg3 1.199890e+12 1.200341e+12
```

```
boot_fn <- function(df, idx) {
  d_boot <- df[idx, ]
  fit    <- lm(sale.amount ~ poly(burglary, 2, raw = TRUE), data = d_boot)
  pred   <- predict(fit, newdata = df)
  mean((df$sale.amount - pred)^2)
}
boot_out <- boot(data = data, statistic = boot_fn, R = 100)

boot_out$t0
```

```
## [1] 1.200632e+12
```

```
mean(boot_out$t)
```

```
## [1] 1.200746e+12
```

```
paste0("confidence interval:", " (", boot.ci(boot_out, type = "perc")$percent[4], ", ",boot.ci(boot_out
```

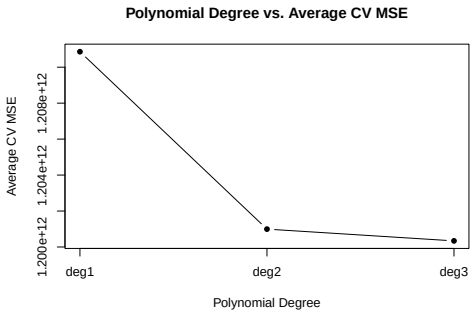
```
## [1] "confidence interval: (1200639335229.19, 1201020519185.12)"
```

```
boot_coef <- function(df, i) {
  d <- df[i, ]
  fit <- lm(sale.amount ~ poly(burglary, 2, raw=TRUE), data=d)
```

```
  coef(fit)[2]
}
b <- boot(data, boot_coef, R=100)
paste0("confidence interval:", " (", boot.ci(b, type = "perc")$percent[4], ", ",boot.ci(b, type = "perc
```

```
## [1] "confidence interval: (-1381.95152707638, -1149.50253838334)"
```

```
mse_fold_mat <- sapply(1:3, function(deg) {
  sapply(folds, function(test_idx) {
    train_df <- data[-test_idx, ]
    test_df  <- data[test_idx, ]
    fit <- lm(sale.amount ~ poly(burglary, deg, raw = TRUE), data = train_df)
    pred <- predict(fit, test_df)
    mean((test_df$sale.amount - pred)^2)
  })
})


avg_mse <- cv_MSE
plot(seq_along(avg_mse), avg_mse, type = "b", pch = 16,
     xaxt = "n",
     xlab = "Polynomial Degree",
     ylab = "Average CV MSE",
     main = "Polynomial Degree vs. Average CV MSE")
axis(1, at = 1:length(avg_mse), labels = names(avg_mse))
```



For this part we are doing the sale amount and burglary and sale amount basing on the Polynomial Model. In the deg1 model we can find that The burglary coefficient is -308.38, significantly negative ($p < 2e\text{-}16$). For each additional burglary, the average house price decreases by approximately \$308 and for the R^2=0.01125, indicating that the model can account for only 1.13% of the house price fluctuations. And for the deg2 model we find out that-1,259 still negative and for all terms are significant ($p < 2e\text{-}16$) and the adj R^2=0.01948, with a slight increase in explanatory power. For the deg3 model we checked that it was still very significant and R^2=0.02009. So we conduct the CV fold test We can get that deg2 will be the most fitted model to choose. For deg2's bootstrap CI The average value is approximately 1.2007×10^{12} and the 95% CI =(1.200641e+12, 1.201089e+12). So we can draw a conclusion that Burglary has a statistically significant negative impact on house prices, reducing prices by an average of \$300 to \$1,300 per additional case.This influence has been fully manifested in the simple linear model and the Bootstrap results confirm that this negative trend is stable and has a basis for confidence. Although R^2 is small, this suggests that burglary is only one of many factors influencing house prices, but it is indeed a negative factor. From the last plot we can find the relation between Polynomial Degree and Average CV MSE. From deg1 to deg2, the MSE decreased significantly, indicating that the deg2 polynomial model fits the data better than the deg1 model. From deg2 to deg3, the MSE change is very small and almost flat, indicating that increasing to a third-order polynomial does not bring a significant performance improvement. So we can see that the best model should

be deg2 for low MSE and not import more complexity.

Conclusion

From our EDA and modeling evaluations, it is clear that the variables in our datasets leave a large amount of unexplained variation. Our EDA process showed us massive amounts of variation in the response variable, and correlation that was not ideal. Using this information, we attempted to circumvent these problems by removing the outliers and retraining the models that way, which significantly improved the MSE. However, the models still had very high MSE. The lowest we could get the MSE using education predictors was using a tree regression and the no-outlier data giving us an average prediction error of about $124,000. Our hypothesis about higher performing schools being correlated with higher real estate prices was also difficult to interpret using our polynomial models, as some variables had a big positive impact, while surprisingly, other variables such as the graduation rate had a massive negative impact that was hard to explain. For the data analysis between the crime and housing prices, we can conclude that both violent crime and burglary will negatively affect the housing price. But since we are using single variable analysis, the MSE will be way bigger. Overall, although we made serious attempts to get better predicting power out of our models, we can say that these variables only partially explain the variation in the data, and our high MSE may be the result of irreducible error due to the limited data. Many factors can explain this, like how most potential buyers aren't thinking about exact crime rates or education quality for their children when buying a house.