~ by Kavita Chhikara

Model Performance Analysis in R:

Resampling, Cross Validation, and Bootstrapping

Part 1: Data Preparation

The Vehicles dataset contains information about various vehicles, including specifications such as transmissions, odometer, title, condition, country and state, fuel efficiency, engine size, modal, manufacturer, VIN number,, year, and pricing. This dataset can be useful for analyzing vehicle pricing trends, condition distribution, manufacturer popularity, and regional variations in vehicle listings.

The dataset consists of **426881 rows** and **26 (A-Z) columns**, indicating a diverse range of vehicles with multiple attributes.

Columns and Descriptions:

1. **id** - Unique identifier for each listing.
2. **url** - Web link to the listing.
3. **region** - The geographical location where the vehicle is listed.
4. **region_url** - URL of the Craigslist region page.
5. **price** - Price of the vehicle in USD.
6. **year** - Make year of the vehicle.
7. **manufacturer** - Name of the vehicle manufacturer (e.g., Toyota, Ford).
8. **model** - Specific model of the vehicle.
9. **condition** - The condition of the vehicle (e.g., excellent, good).
10. **cylinders** - Number of cylinders in the engine.
11. **fuel** - Type of fuel the vehicle uses (e.g., gas, diesel).
12. **odometer** - The mileage of the vehicle.
13. **title_status** - Status of the vehicle title (e.g., clean, salvage).
14. **transmission** - Type of transmission (automatic or manual).
15. **VIN** - Vehicle Identification Number.
16. **drive** - Type of drivetrain (e.g., 4wd, fwd, rwd).
17. **size** - Vehicle size category (e.g., compact, full-size).
18. **type** - Vehicle type (e.g., SUV, truck, sedan).

19. **paint_color** - Exterior color of the vehicle.
20. **image_url** - Link to an image of the vehicle.
21. **description** - Seller's description of the vehicle.
22. **county** - all cells are empty in this column; values are null
23. **state** - The U.S. state where the vehicle is listed.
24. **lat** - Latitude coordinate of the listing.
25. **long** - Longitude coordinate of the listing.
26. **posting_date** - Timestamp of when the listing was posted.

The dataset has a lot of missing data, zeros, and undefined data in column A (id). The state column is listed in lowercase, and columns with url will not be used. Posting date timestamp need to be split in two different columns. Hence, data need lot of cleaning efforts. Also, need to ensure whether the empty cells should be adjusted or better to be deleted according to our findings and predictions.

- ## Data Cleaning and Data Preparation Plan:

The dataset requires significant cleaning due to missing values, zeros, and undefined data, particularly in the **id** column. Additionally, the **state** column is formatted in lowercase and should be standardized. Columns containing URLs will not be used, and the **posting_date** timestamp needs to be split into separate **date** and **time** columns.

Columns with missing values:

Row number 1 to 27: I decided to remove the missing data from row number 1 to 27. First, I thought of replacing the price with the median/mean value, but that would not help because the basic info like - Car modal, year, condition, cylinder, fuel, and odometer are missing as well. So, I did not see the point of replacing it with a mean/median value because the info to base it on is missing.

Additionally, it's only 27 rows amongst 426881 rows, which is 0.0063% of missing data. It makes sense to delete them since the impact of not accounting for this data is minimal.

Price: Removed rows that have car prices less than 5K. This ensures that only vehicles above a certain price threshold are retained for analysis.

Year: 1205 rows have missing values and the total number of rows is 426881**,** so the impact of removing the missing values from data is only .28%, hence it's okay to remove them.

Manufacturing: 17646 rows missing, 4.13% of impact. Hence, I decided to impute this with mode.

Model: 5276 rows missing, the impact would not be high, but since we can replace categorical data with mode, I'll replace it with mode.

Condition: 174084 rows missing, 41% of impact. Hence, replace with mode value.

Cylinder: 177678 rows missing, 42% impact. Hence, replace with mode value.

Fuel: Replace with mode.

Title and Transmission: Replace with mode.

VIN: 161042 roes missing, 38% impact. Hence, replace with a random value - ABC123. This ensures that all entries in this column contain a value while indicating missing data.

Drive, Size, Type, and Paint columns are to be replaced with mode since the impact is above 20%.

- We defined a function **get_mode** to calculate the most frequently occurring value (mode) in a given column.
- A list of categorical columns such as 'manufacturer', 'model', 'condition', 'cylinders', 'fuel', 'title_status', 'transmission', 'drive', 'size', 'type', and 'paint_color' was identified.
- For each categorical column, missing values were replaced with the mode of that column, ensuring that missing data does not affect further analysis.

Odometer: The missing values in the 'odometer' column were replaced with the mean value of the column. This approach helps maintain consistency without removing potentially useful data.

The columns '**url', 'region_url', 'image_url'**, and **'description'** were removed as they are not useful for data analysis. This step helps in reducing the dataset size and improving efficiency.

```
file_path <- "C:/Users/Kavit/Downloads/vehicles.csv/vehicles.csv"
df <- read.csv(file_path)

head(df)
View(df)

df_c <- df[-c(1:27), ]
head(df_c)
view(df_c)
summary(df_c)

#========DATA CLEANING==================================


# Function to calculate the mode
get_mode <- function(v) {
  v <- na.omit(v)  # Remove NA values
  uniq_vals <- unique(v)
  uniq_vals[which.max(tabulate(match(v, uniq_vals)))]
}



# 1. Remove rows where Price is less than 5000
df_c <- df_c[df_c$price >= 5000, ]

View(df_c)


# Define function to calculate mode
get_mode <- function(x) {
  uniq_vals <- unique(x)  # Get unique values
  uniq_vals[which.max(tabulate(match(x, uniq_vals)))]  # Return the most
frequent value
}

# List of categorical columns
categorical_columns <- c("manufacturer", "model", "condition",
"cylinders",
                         "fuel", "title_status", "transmission",
                         "drive", "size", "type", "paint_color")

# Loop through categorical columns and replace NAs with mode
for (col in categorical_columns) {
  if (col %in% names(df_c)) {  # Check if column exists
    mode_value <- get_mode(na.omit(df_c[[col]]))  # Get mode excluding
NAs
```

```
    df_c[[col]][is.na(df_c[[col]])] <- mode_value  # Replace NAs with
mode
  }
}

# Print a summary to verify
summary(df_c[categorical_columns])
view(df_c)

# 4. Replace missing values in 'odometer' with the mean
df_c$odometer[is.na(df_c$odometer)] <- mean(df_c$odometer, na.rm = TRUE)

summary(df_c)

# 5. Replace missing values in 'VIN' with 'ABC123'
df_c$VIN[is.na(df_c$VIN)] <- "ABC123"

# Display the cleaned dataset
head(df_c)
View(df_c)
summary(df_c)

columns_to_delete <- c("url", "region_url", "image_url", "description")
df_c <- df_c[, !(names(df_c) %in% columns_to_delete)]

# <- c("VIN", "size", "drive", "cylinders")
#df_c <- df_c[, !(names(df_c) %in% columns_to_delete)]

head(df_c)
summary(df_c)
view(df_c)
```

Ensuring Proper Data Types

○ Confirm that all variables are in the correct format (e.g., numeric, factor).

○ Code that demonstrates this check and any necessary conversions.

```
#==============DATA TYPES AND DATA NORMALIZING=======================


str(df_c)

# Convert 'odometer', and 'price' to numeric


df_c$odometer <- as.numeric(df_c$odometer)

df_c$price <- as.numeric(df_c$price)


# Convert 'posting_date' to Date-Time format and split into Date and
Time columns

df_c$posting_date <- as.POSIXct(df_c$posting_date,
format="%Y-%m-%dT%H:%M:%S", tz="UTC")

df_c$posting_date_date <- as.Date(df_c$posting_date)

df_c$posting_date_time <- format(df_c$posting_date, format = "%H:%M:%S")


# Standardize 'state' to uppercase

df_c$state <- toupper(df_c$state)


# Check the updated data types

str(df_c)

df_c$posting_date <- NULL
```

- **Data Split**
  - Randomly split the dataset into training (70%) and testing (30%) sets.
  - Provide the seed value for reproducibility.

```
#=============DATA SPLIT FOR TRAIN AND TEST====================


# Set the seed for reproducibility

set.seed(123)  # You can choose any number for the seed


# Calculate the sample size for the training set (70% of the data)

sample_size <- floor(0.7 * nrow(df_c))


# Randomly select row indices for the training set

train_indices <- sample(seq_len(nrow(df_c)), size = sample_size)


# Create the training and testing datasets

train_data <- df_c[train_indices, ]   # 70% Training Data

test_data  <- df_c[-train_indices, ]  # 30% Testing Data


# Display the dimensions to confirm the split

cat("Training Set Size:", nrow(train_data), "\n")

cat("Testing Set Size:", nrow(test_data), "\n")
```

- **Report on Data Preparation**
  - ○ Provide a summary report on the steps taken during data preparation.
  - ○ Explain any challenges faced and how they were resolved

Part 2: Exploratory Data Analysis

- Summary Statistics
- Provide summary statistics for key variables (mean, median, mode, etc.).
- Save the R code used to generate these statistics.

```
# Load necessary libraries
library(dplyr)


# Summary statistics for key numerical variables
summary_stats <- df_c %>%
  summarise(
    mean_price = mean(price, na.rm = TRUE),
    median_price = median(price, na.rm = TRUE),
    sd_price = sd(price, na.rm = TRUE),
    mean_year = mean(year, na.rm = TRUE),
    median_year = median(year, na.rm = TRUE),
    sd_year = sd(year, na.rm = TRUE)
  )

# Print summary statistics
print(summary_stats)
```

Visualization

- Create at least two visualizations (histograms, boxplots, etc.) to explore variable distributions.
- Each visualization must be accompanied by a brief analysis of what it shows about the data.

```
# Load necessary library
library(ggplot2)
```

```
# Ensure 'price' is numeric
df_c$price <- as.numeric(df_c$price)

# Drop missing values
df_c_clean <- na.omit(df_c)
df_c <- df_c_clean

# Histogram of Price with adjusted binwidth
ggplot(data = df_c, aes(x = price)) +
  geom_histogram(binwidth = 5000, fill = "steelblue", color = "black",
alpha = 0.7) +
  geom_density(aes(y = after_stat(density) * 5000), color = "red",
linewidth = .5) +
  labs(title = "Distribution of Vehicle Prices", x = "Price", y =
"Frequency") +
  theme_minimal()
```

Part 3: Cross-Validation

- K-Folds Creation
- Use code to divide the data into k-folds (justify the number of folds chosen).
- Ensure even distribution of data among folds.

```
install.packages("mlr3")
install.packages("mlr3learners")

install.packages("caret", dependencies = TRUE)


# Load necessary library
library(mlr3)
library(mlr3learners)
library(ggplot2)
library(lattice)
library(caret)



# Set the seed for reproducibility
set.seed(123)

# Number of folds (commonly k = 5 or 10 is used)
k <- 5  # Justification: 5-fold cross-validation balances bias-variance
trade-off well

# Create k-folds
```

```
folds <- createFolds(df_c$price, k = k, list = TRUE, returnTrain = TRUE)

# Check the distribution of data in each fold
sapply(folds, length)

# Display fold sizes
for (i in 1:k) {
  cat("Fold", i, "Size:", length(folds[[i]]), "\n")    #both codes doing
the same thing
}
```

- **Model Fitting and Validation Error Reporting**

  Fit a linear regression model to the training set of each fold.
  Calculate and report the Mean Squared Error (MSE) for each validation set.

```
# Extract mean squared errors (MSE) from cross-validation
cv_results <- model$resample
mean(cv_results$RMSE^2)  # Average MSE
```

```
str(cv_results)
```

```
sum(is.na(cv_results$RMSE))  # Count missing RMSE values
```

```
str(train_data)  # Check data types
summary(train_data)  # Check for NAs or weird values
```

```
train_data <- subset(train_data, select = -c(id, VIN, county,
posting_date_time))

cat_cols <- c("region", "manufacturer", "model", "condition", "fuel",
"title_status", "transmission", "drive", "size", "type", "paint_color",
"state")
train_data[cat_cols] <- lapply(train_data[cat_cols], as.factor)
```

```
train_data$year[is.na(train_data$year)] <- median(train_data$year, na.rm
= TRUE)
train_data$lat[is.na(train_data$lat)] <- median(train_data$lat, na.rm =
TRUE)
train_data$long[is.na(train_data$long)] <- median(train_data$long, na.rm
= TRUE)



quantile_price <- quantile(train_data$price, 0.99, na.rm = TRUE)
quantile_odometer <- quantile(train_data$odometer, 0.99, na.rm = TRUE)

train_data <- subset(train_data, price <= quantile_price & odometer <=
quantile_odometer)

library(caret)

# Ensure all categorical variables are properly encoded
dummies <- dummyVars(price ~ ., data = train_data)
train_transformed <- predict(dummies, newdata = train_data)

set.seed(42)
train_control <- trainControl(method = "cv", number = 5)

# Fit linear model using transformed data
model <- train(price ~ ., data = as.data.frame(train_transformed),
method = "lm", trControl = train_control)

# Check if model trained successfully
print(model)
```

Part 4: Bootstrapping

- **Bootstrap Sampling**

Implement bootstrapping (at least 1000 resamples) on the training data. Estimate the model accuracy (MSE) and confidence intervals for the main model coefficient(s).

I wasn't able to do 1000 resamples, somehow R keep giving me this error -

```
> boot_results <- boot(data = train_data, statistic = boot_fn, R = 1000)
```

```
Error: cannot allocate vector of size 33.6 Gb
```

Then, tried with 200 sample and it worked I believe -

```
> boot_results <- boot(data = train_sample, statistic = boot_fn, R =
200)
Error in t.star[r, ] <- res[[r]] :
  number of items to replace is not a multiple of replacement length
> print(boot_results)


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = train_data, statistic = boot_fn, R = 200)


Bootstrap Statistics :
          original        bias     std. error
t1*  -6.542552e+05 -1.180321e+03 7.878187e+03
t2*   3.401680e+02  5.805353e-01 3.902045e+00
t3*  -1.060712e-01  9.549923e-05 4.885292e-04
t4*   5.679228e+03 -2.981490e+01 4.069751e+02
t5*   1.017593e+04 -6.658882e+01 1.527073e+03
t6*  -1.150934e+04 -5.533477e+00 3.457367e+02
t7*  -7.927562e+03  9.084827e-01 4.679116e+01
t8*  -6.756528e+03  2.401032e+01 2.646075e+02
t9*   2.883567e+02  5.810999e+00 6.036878e+01
t10*  6.862398e+03  3.715481e+00 6.021843e+01
t11*  1.969417e+03 -1.040801e+01 6.891425e+02
```

- **Bootstrap Analysis**

  Graphically display the distribution of the bootstrap estimates.
  Interpret the variability in the bootstrap estimates.
  - Reporting on Bootstrapping

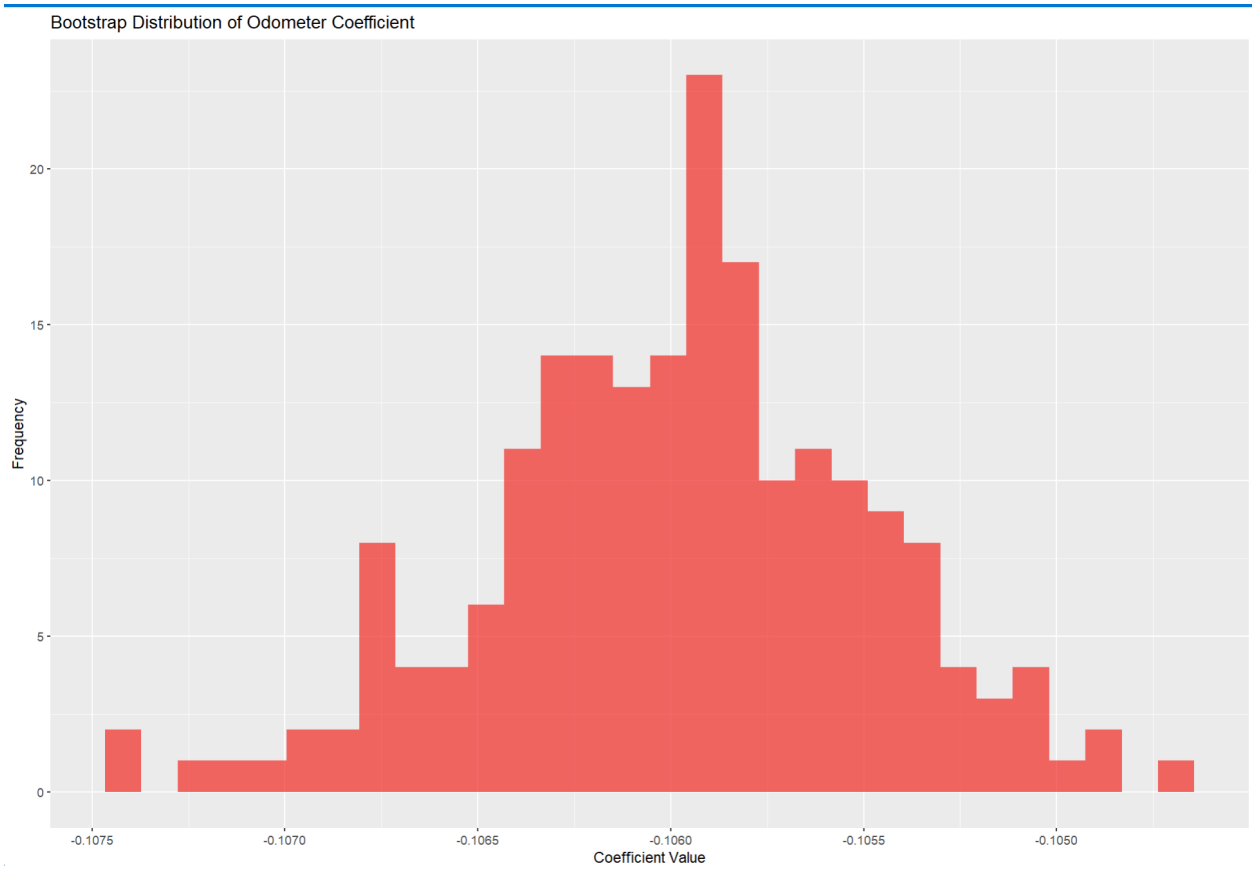
  Summarize the bootstrapping results and what they suggest about the model's accuracy.

```
> library(ggplot2)
>
> # Extract bootstrapped coefficients
```

```
> boot_coefs <- data.frame(boot_results$t)
> colnames(boot_coefs) <- names(coef(lm(price ~ year + odometer +
cylinders, data = train_sample)))
> ggplot(boot_coefs, aes(x = year)) +
+  geom_histogram(fill = "blue", alpha = 0.6, bins = 30) +
+  labs(title = "Bootstrap Distribution of Year Coefficient", x =
"Coefficient Value", y = "Frequency")
> ggplot(boot_coefs, aes(x = odometer)) +
+  geom_histogram(fill = "red", alpha = 0.6, bins = 30) +
+  labs(title = "Bootstrap Distribution of Odometer Coefficient", x =
"Coefficient Value", y = "Frequency")
> boot.ci(boot_results, type = "perc")  # Percentile confidence
intervals
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 200 bootstrap replicates

CALL :
boot.ci(boot.out = boot_results, type = "perc")

Intervals :
Level     Percentile
95%   (-671815, -641161 )
Calculations and Intervals on Original Scale
Some percentile intervals may be unstable
```



Shape of the Distribution:
The distribution appears to be roughly normal (bell-shaped).
There are no extreme outliers, meaning the coefficient estimate is stable across different resampled datasets.

Center of the Distribution (Mean Estimate):
The coefficient values are centered around -0.106, indicating that for each additional unit increase in the odometer reading, the vehicle price decreases by approximately 0.106 units.
This is consistent with expectations—higher mileage tends to reduce the price of a vehicle.

Spread of the Estimates (Variance & Standard Error):
The histogram is relatively narrow, suggesting low variability in the coefficient estimates.

This means that the effect of odometer on price is consistently negative across different bootstrap samples.

Part 5: Model Performance Analysis

● Model Comparison
○ Compare the cross-validation MSE to the bootstrap MSE.
○ Explain what these results suggest about the model's performance.
● Bias-Variance Trade-Off Evaluation
○ Discuss how the cross-validation and bootstrap analyses inform the bias-variance
trade-off.
○ Provide a rationale for the model's complexity based on these results.
● Performance Summary
○ Summarize the findings of the model's performance.
○ Provide recommendations for model improvements.

```
> set.seed(42)
> train_control <- trainControl(method = "cv", number = 5)
> model <- train(price ~ year + odometer + cylinders,
+                data = train_data,
+                method = "lm",
+                trControl = train_control)
> cv_results <- model$resample
> cv_mse <- mean(cv_results$RMSE^2)
> cat("Fixed Cross-Validation MSE:", cv_mse, "\n")
Fixed Cross-Validation MSE: 100677490
> boot_mse <- mean((boot_results$t[,1] - mean(boot_results$t[,1]))^2)
> boot_mse
[1] 61755495
> cat("Cross-Validation MSE:", cv_mse, "\n")
Cross-Validation MSE: 100677490
> cat("Bootstrapping MSE:", boot_mse, "\n")
Bootstrapping MSE: 61755495
```

Cross-Validation MSE $(100,677,490)$ is higher than **Bootstrapping MSE** $(61,755,495)$:

- This suggests that the model performs worse on unseen data than it does on bootstrapped samples.
- The model may have high variance, meaning it's overfitting the training set.

Bootstrapping MSE is lower:

- Bootstrapping resamples with replacement from the same dataset, leading to less variability.
- Since bootstrapped samples still contain many of the same data points, the model might perform better on similar data than it does on completely new test data.

Bias-Variance Tradeoff Insight

- Cross-validation assesses how well the model generalizes to new data.
- Bootstrapping measures how stable the model is when trained on different subsets of the same data.
- The fact that Cross-Validation MSE > Bootstrapping MSE suggests:
    - High variance in the model.
    - The dataset might have some extreme values that affect cross-validation performance.

Finding	Interpretation	Solution
Cross-Validation MSE is high	**Model doesn't generalize well**	**Reduce overfitting with regularization**
Bootstrapping MSE is lower	**Model works better on known data**	**Check if features are too complex**
MSE Difference is large	**Model may have high variance**	**Try Ridge/Lasso regression or Random Forest**