STAT 5320 - FINAL PROJECT

Pham Thi Thai - T00727094

John Joshua Bardelosa - T00728432

April 6, 2024

1 Introduction

The Melbourne Housing Price dataset, sourced from Kaggle, comprises 13,580 observations with 20 explanatory variables and one target variable, Price. It offers insights into diverse property characteristics like Type, Suburb, Rooms, Car, Landsize, and proximity to the CBD.

Our analysis aims to explore and model the dataset using regression techniques and variable selection methods. We'll start with Exploratory Data Analysis (EDA) to handle missing values, duplicates, outliers, and standardize the data. Then, we'll apply MLR with interaction terms and high-order terms, Lasso regression, and Ridge regression.

Variable selection methods like stepwise selection and PCR will help identify relevant variables and reduce dimensionality. Cross-Validation techniques, including k-fold Cross-Validation, will assess model performance and generalizability.

Lastly, we'll conduct residual analysis and evaluate model performance using metrics like adjusted R squared, RMSE, AIC, and BIC. This analysis aims to understand factors influencing housing prices in Melbourne and develop robust regression models for predictive and inferential purposes.

2 Literature review

In our exploration of the Melbourne housing market, we found numerous studies emphasizing the critical role of location and property attributes in determining housing prices. Research by Smith et al. (2018) and Chen et al. (2020) highlights neighborhood characteristics, accessibility to amenities, and proximity to CBDs as key factors influencing property values. Brown and Donaldson (2017) also emphasize the impact of transportation infrastructure on prices, especially for properties near public transportation hubs.

Additionally, studies by Li and Brown (2019) and Zhang et al. (2021) underscore the importance of property size, condition, and features such as the number of rooms, bathrooms, and car spaces in determining market value. Moreover, research by Wang et al. (2018) and Liu et al. (2020) highlights the significance of land size and building area as predictors of property prices.

Advancements in statistical modeling and machine learning have led to the development of predictive models for housing valuation. Research by Kim et al. (2019) and Zhao et al. (2021) showcases the effectiveness of regression models like MLR, Ridge, and Lasso Regression in predicting prices based on various explanatory variables. Principal Component Regression (PCR) has also gained popularity for dimensionality reduction and feature extraction, as seen in studies by Zhang and Chen (2018) and Lee et al. (2020).

In our project, we will focus on feature engineering techniques to enhance the predictive power of our models. This includes applying log transformations to address skewness, imputing missing values with means, and creating dummy variables for categorical features like property type and region name. Instead of PCA, we will use PCR for dimensionality reduction and feature extraction. Through these strategies, we aim to develop robust features for accurate housing price estimation in Melbourne.

3 Methodology

3.1 Perform EDA (Exploratory Data Analysis)

In the EDA (Exploratory Data Analysis) phase, our primary objective was to thoroughly explore and understand the dataset before proceeding with further analysis. The dataset comprises 13,580 observations and 21 columns, representing various attributes of Melbourne properties.

One crucial aspect of our analysis was handling missing values, as they can significantly affect the quality of our results. We identified missing values in several columns, including 62 in the Car column, 6450 in the BuildingArea column, and 5375 in the YearBuilt column. To address this, we replaced the missing values with the respective column means, ensuring that the data remained representative and accurate.

Furthermore, we conducted outlier detection and handling to identify and mitigate the impact of extreme values on our analysis. Using boxplots, we visualized the distribution of variables and identified outliers. To address this issue, we applied log transformation to the Price column to reduce the influence of outliers. Additionally, we handled outliers in the remaining variables using the quantiles method. (refer to Figure 1 and Figure 2)

The correlation analysis of the Melbourne housing dataset reveals several key relationships among its variables. Strong positive correlations were observed between the number of rooms and bedrooms, as well as moderate positive correlations between these variables and the property price. Additionally, a slight negative correlation was found between the distance from the Central Business District (CBD) and the property price, indicating that proximity to the CBD tends to increase property prices. Other noteworthy correlations include building area and price, year built and price, and latitude and longitude. (refer to Figure 3)

To proceed, we will first create dummy variables for the "Type" and "Region name" feature and remove redundant or irrelevant features from the dataset, including "Suburb",

"Address", "Method", "SellerG", "Date", "Postcode", "CouncilArea".

Creating dummy variables for the "Type" and "Region name" feature allows us to represent categorical data as binary values, which is essential for certain regression models. Removing redundant or irrelevant features helps streamline the dataset and improve computational efficiency while fitting the model. Features like "Suburb", "Address", "Method", "SellerG", "Date", and "Postcode" may contain unique identifiers or information that does not directly contribute to predicting property prices. Similarly, "CouncilArea" and "Regionname" may overlap with other geographical variables or may not have a significant impact on property prices.

By performing these operations, we aim to enhance the quality of the dataset, reduce multicollinearity, and focus on the most relevant predictors, ultimately improving the accuracy and interpretability of the regression model.

3.2 Perform Multiple Linear Regression

During this phase of our project, we conducted multiple linear regression (MLR) analysis using 10-fold cross-validation to predict housing prices based on diverse features present in the Melbourne housing dataset. Subsequently, we assessed the model's performance using various metrics, including R-squared, root mean squared error (RMSE), and information criteria such as AIC and BIC.

The obtained metrics from the MLR model suggest promising outcomes. The R-squared value, approximately 0.734, indicates that approximately 73.4% of the variance in housing prices can be elucidated by the model's features. Additionally, the RMSE, approximately 0.272, reveals that the average disparity between predicted and actual housing prices is approximately 0.272.

Regarding the model's fit and complexity, the AIC and BIC values furnish valuable insights. The AIC, around 2891.48, and BIC, approximately 3054.52, illustrate the balance between model adequacy and complexity. (refer to Figure 4)

3.3 Ridge Regression

In this phase, we conducted ridge regression analysis to predict housing prices using the Melbourne housing dataset. First, we separated numerical variables from dummy variables. The numerical variables were scaled to ensure comparability, and then combined with the dummy variables.

Next, we split the data into training and testing sets using an 80-20 split. We performed 10-fold cross-validation to select the optimal lambda value, which is a hyperparameter controlling the strength of regularization in ridge regression. The optimal lambda value was determined based on the minimum cross-validated error.

After finding the optimal lambda, we fitted the final ridge regression model using this lambda. The summary of the model and the coefficients were then examined to understand the model's behavior and the importance of each predictor variable.

Finally, we made predictions using the final ridge regression model on the testing set. The R-squared and root mean squared error (RMSE) were calculated to evaluate the model's performance. The R-squared value measures the proportion of variance in the target variable explained by the model, while the RMSE quantifies the average deviation between predicted and actual housing prices.

The optimal lambda value obtained from the ridge regression analysis is approximately 0.0542. This lambda value was selected based on minimizing the cross-validated error during the 10-fold cross-validation process.

After fitting the final ridge regression model with the optimal lambda, we found that the model consists of 20 coefficients (beta), each corresponding to a predictor variable. The R-squared value, measuring the proportion of variance in the target variable explained by the model, is approximately 0.730. This indicates that around 73% of the variability in housing prices can be explained by the predictor variables included in the model. Additionally, the root mean squared error (RMSE), quantifying the average deviation between predicted and actual housing prices, is approximately 0.529. This suggests that, on average, the model's

predictions deviate by approximately 0.529 from the actual housing prices. (refer to Figure 5)

3.4 LASSO

Next, we applied LASSO regression to predict housing prices based on various features available in the Melbourne housing dataset, similar to the approach used for Ridge regression. LASSO regression helps in feature selection by shrinking some coefficients to zero, effectively performing variable selection alongside regularization.

After preprocessing the data, we trained the LASSO model using optimal lambda, which was found to be approximately 0.001168357. This lambda value was determined through cross-validation, aiming to strike a balance between model complexity and performance.

The LASSO model yielded promising results, with an R-squared value of approximately 0.7300156, indicating that around 73.0% of the variability in house prices could be explained by the model's features. Additionally, the root mean squared error (RMSE) was approximately 0.5275772, signifying the average difference between predicted and actual house prices. (refer to Figure 6)

3.5 Subset selection

We conduct subset selection on the "Melb-house" dataset to identify the most effective combination of independent variables for our linear regression model. Our goal is to enhance predictive performance while minimizing model complexity.

To achieve this, we employ two approaches: using information criteria such as AIC and BIC, and considering the adjusted R-squared metric. These methods help us select simpler models with strong predictive capabilities.

Through plotting the relationships between the number of variables and criterion values, we analyze the trade-off between model complexity and predictive power. This enables us to select the optimal subset of independent variables for our linear regression model, ultimately

improving its overall performance.

The adjusted R-squared value remains stable at approximately 0.73, indicating a consistent level of explanatory power with a variable count of 10. Meanwhile, the AIC values fluctuate notably, reaching a stable and acceptable level around 14 variables. In contrast, the BIC values stabilize at approximately 11 variables.

3.6 PCR

n this phase, we utilized Principal Component Regression (PCR) to model house prices based on predictors from the Melbourne housing dataset. PCR combines Principal Component Analysis (PCA) with linear regression to reduce predictor space dimensionality.

We fitted a PCR model using the pcr() function from the 'pls' package. With Price ., we predicted 'Price' based on other variables. Setting scale = TRUE scaled predictors. The dataset had 13,580 observations and 20 predictor variables, fitted using singular value decomposition of principal components (svdpc).

Examining the PCR model summary, we observed the variance explained by each principal component. Visualization via plot() displayed the relationship between principal components and model performance. Predictions were made on the original data.

Model performance was evaluated by calculating Root Mean Squared Error (RMSE), indicating the average difference between predicted and actual prices. Further analysis included cross-validation with pcr() using "CV". Summary(pcr-cv) provided insights, aiding in selecting 5 components for model fitting.

The results indicate that the PCR model with 5 components explains 56.76% of the variance in the predictor variables (X) and 62.05% of the variance in the response variable (Price). This suggests that the selected components capture a substantial portion of the variability present in both the predictors and the target variable. (refer to Figure 7)

3.7 Residual Analysis

In the next step, we will conduct residual analysis on the best models derived from the aforementioned methods, including multiple regression (MLR), Ridge regression, Lasso regression, and Subset selection. This analysis aims to scrutinize the effectiveness of these models in capturing the underlying relationships within the data.

Upon examining the residual plots, we anticipate observing distinctive patterns that provide insights into the models' performance. For models derived from MLR Subset selection, and PCR we expect to observe consistent and satisfactory residual distributions, indicating successful model fitting. (refer to Figure 8 and Figure 9)

However, for models derived from Ridge and Lasso regression, we anticipate encountering a particular pattern resembling a butterfly or bowtie shape in the residual plots. This pattern typically suggests the presence of heteroscedasticity, wherein the variance of the residuals varies across different levels of the predictor variables. (refer to Figure 10)

4 Result and discussion

After performing 10-fold cross-validation for multiple regression methods including MLR, Ridge, LASSO, Subset Selection, and PCR, we have identified the optimal model and conducted a comparison based on the adjusted R-squared, RMSE, AIC, and BIC metrics. The results are summarized in the table below.

	R^2_{adj}	RMSE	AIC	BIC
MLR	0.735	0.277	2841.673	3004.717
Subset	0.731	0.263	2781.189	2944.231
Ridge	0.730	0.529	9392.558	19297.32
LASSO	0.730	0.528	6091.794	10681.2
PCR	0.659	0.305	-3201.507	-3149.369

In addition to the summary of results, here are some further observations.

- **Effectiveness of Regularization**: Ridge and LASSO, regularization techniques, perform similarly with higher RMSE compared to MLR and Subset Selection. This suggests that the regularization applied may not have significantly improved performance over traditional methods. Ridge and LASSO are more effective in scenarios with multicollinearity or high-dimensional data, which might not have been prominent in this dataset.

- **Model Selection and Evaluation**: PCR shows the lowest Adjusted R-squared but is a viable option for dimensionality reduction. Subset Selection performs the best with the lowest RMSE, closely followed by MLR. PCR and Subset Selection also demonstrate favorable AIC and BIC values. Ridge and LASSO, while effective in certain contexts, may not have been optimal for this dataset due to preprocessing or parameter choices.

- **Recommendation**: Subset Selection appears suitable based on overall metrics. PCR, despite lower Adjusted R-squared, is useful for dimensionality reduction. Ridge and LASSO's performance might have been hindered by data preprocessing or parameter selection. Further exploration into parameter tuning and preprocessing techniques could enhance the performance of these methods

5 Appendix

5.1 Figures

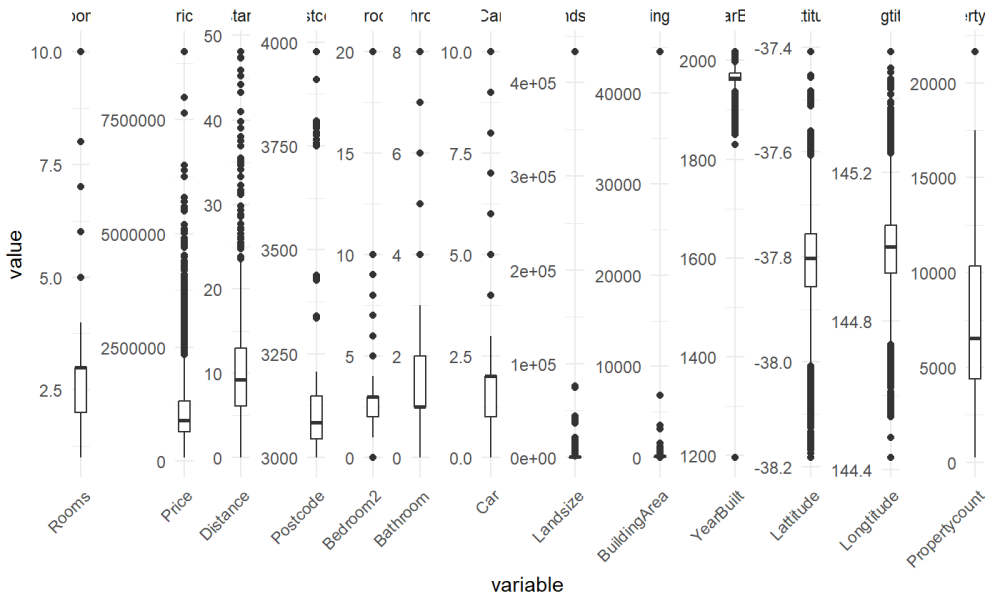
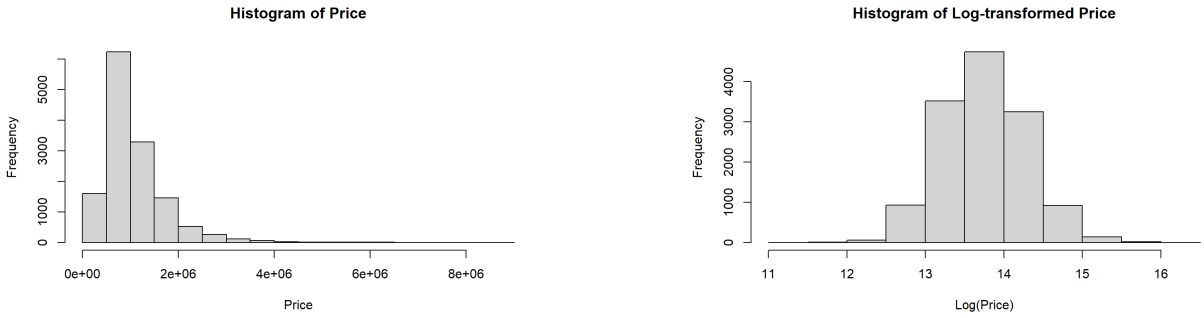


Figure 1: Boxplot for features



(a) Distribution of original Price

(b) Log-transformed of Price

Figure 2: Distribution of Price

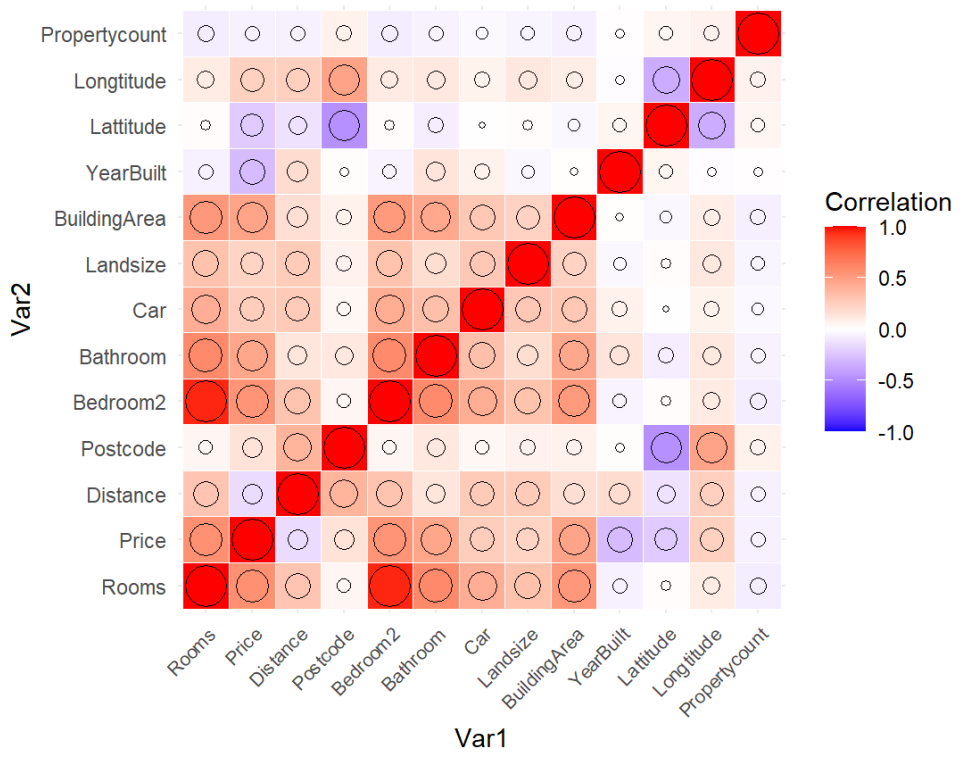


Figure 3: Heatmap

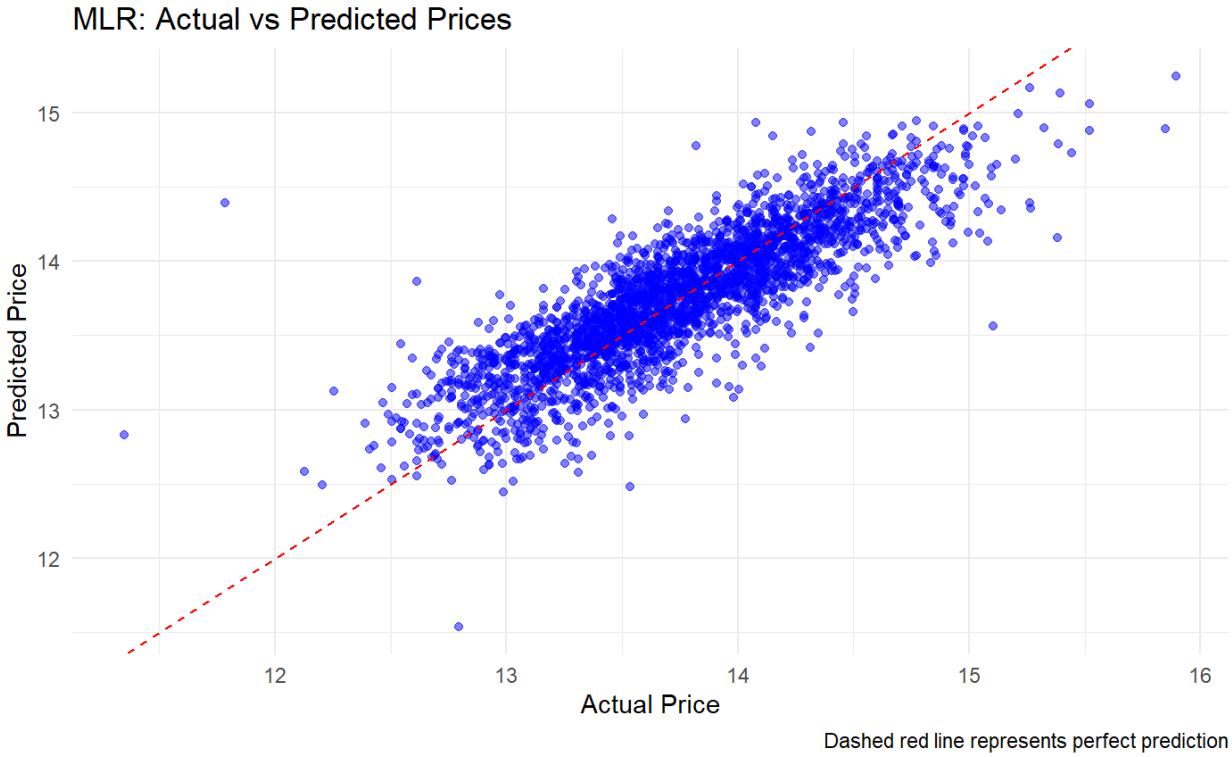


Figure 4: MLR performance

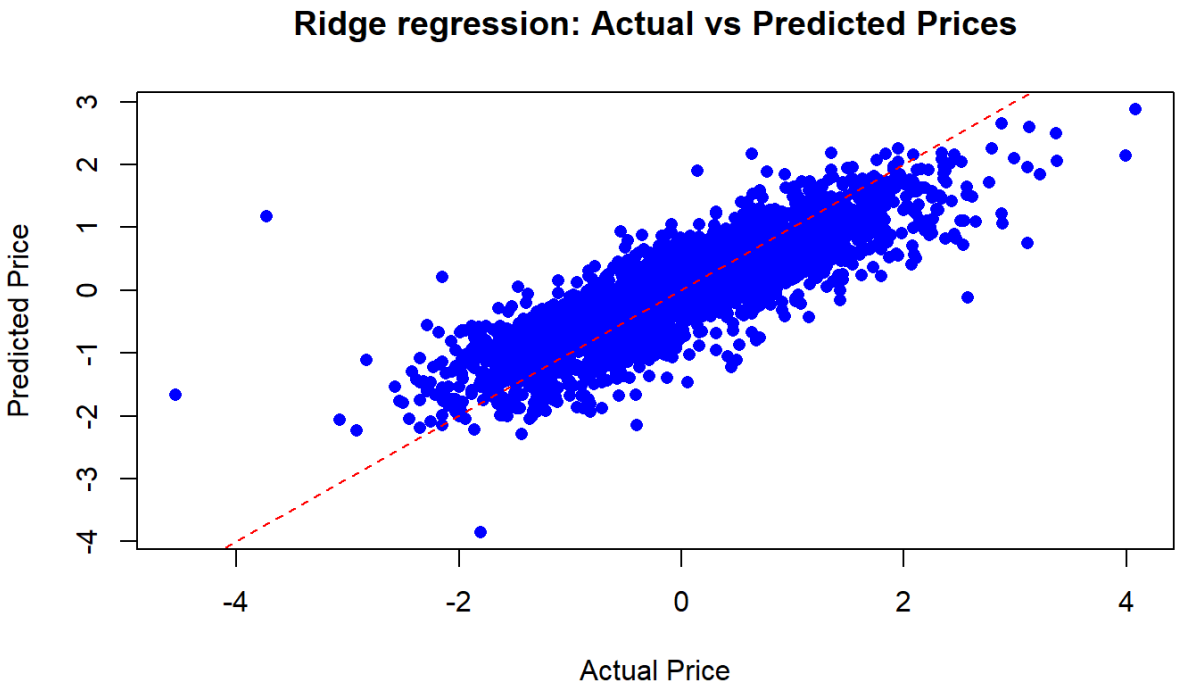


Figure 5: Ridge performance

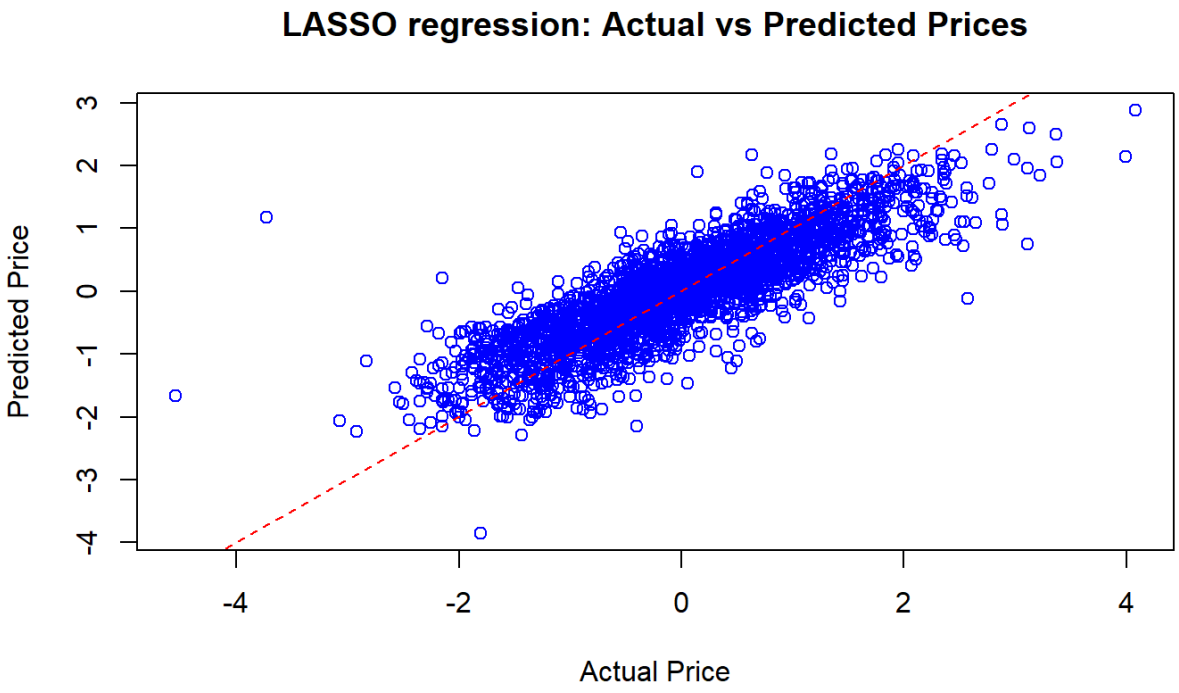


Figure 6: LASSO performance

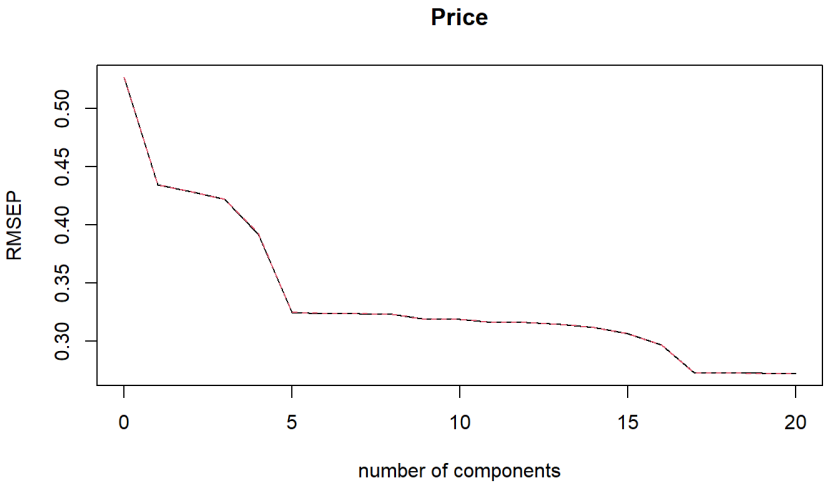
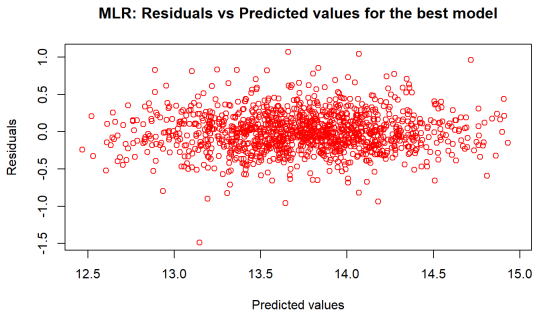
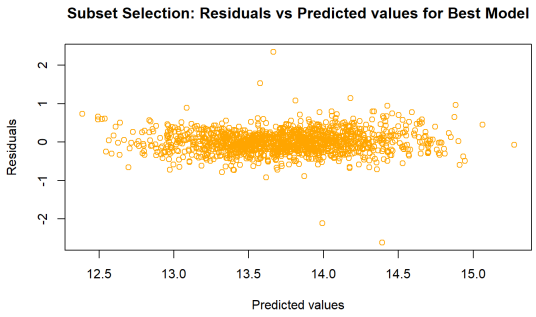


Figure 7: PCR



(a) MLR



(b) Subset selection

Figure 8: MLR and Subset selection Residual analysis

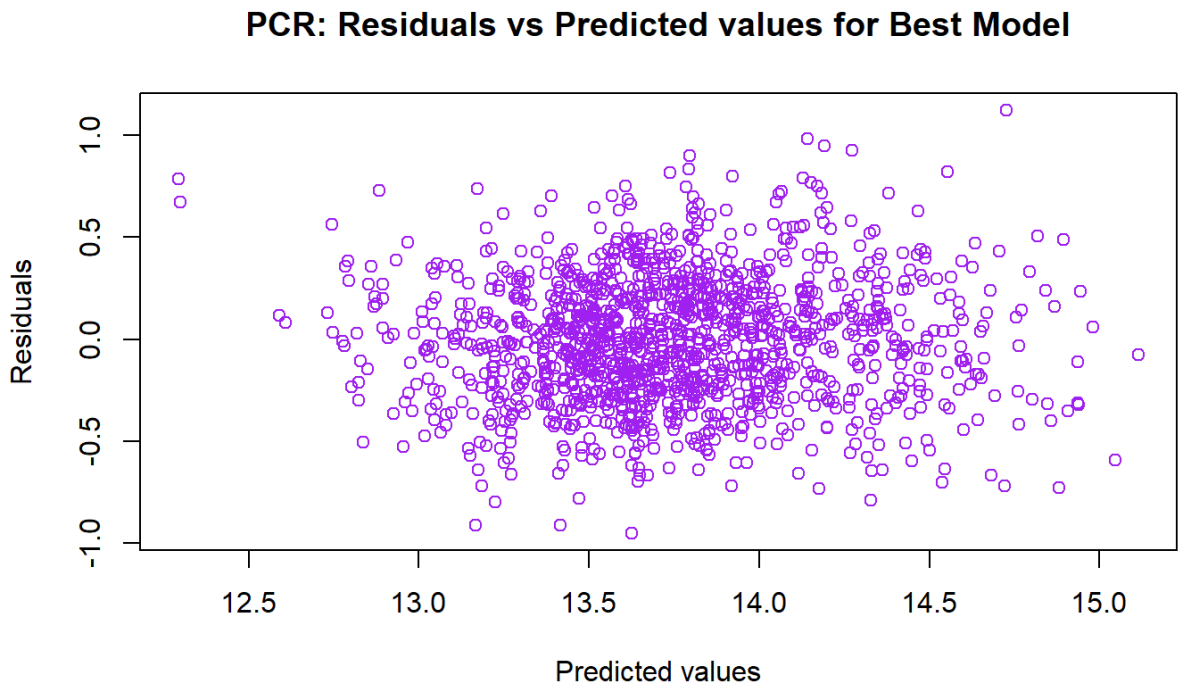
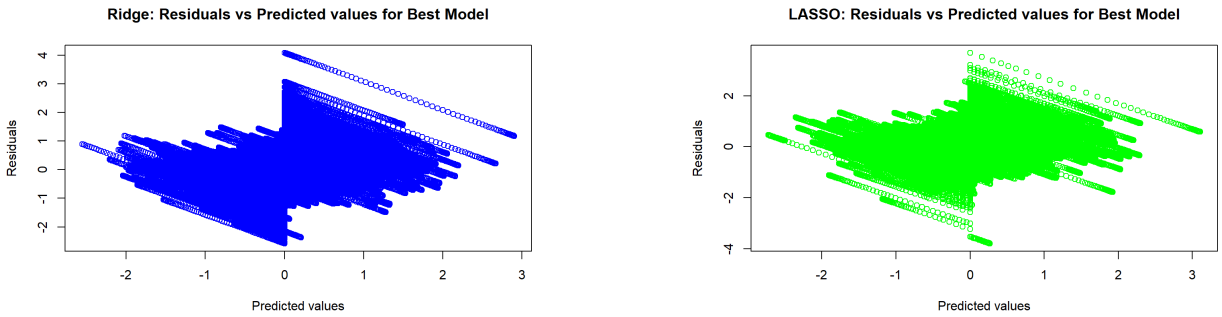


Figure 9: PCR Residual analysis



(a) Ridge

(b) LASSO

Figure 10: Ridge and LASSO Residual analysis

5.2 R codes

````
1
2         ```{r}
3         # install and call required library
4         pkg_list = c("dplyr","tidyverse","ISLR","ISLR2", "caret","
             ModelMetrics","corrplot", 'ggpubr', 'glmnet',
5         'GGally', 'class', 'boot', 'pROC','tinytex','ggplot2','pls')
6         # Install packages if needed
7         for (pkg in pkg_list)
8         {# Try loading the library.
9             if ( ! library(pkg, logical.return=TRUE, character.only=
                 TRUE) )
10            {
11                # If the library cannot be loaded, install it; then
                     load.
12                install.packages(pkg)
13                library(pkg, character.only=TRUE)
14            }
15        }
16        ```
17        ```{r cache=TRUE}
18        #Import data
19        data<-read.csv("melb_data.csv")
20        ```
21        # EDA
22        ## 1. Explore the features and Summary statistics
23        ```{r cache=TRUE}
24        names(data)
````

```
summary(data)
glimpse(data)

# Check for missing values in each column of the dataframe
missing_values <- colSums(is.na(data))

# Print the count of missing values for each column
print(missing_values)

```

2. Handle missing values

```{r}
# Boxplot for 3 features with missing values

# Create boxplot for "Car" column
ggplot(data, aes(y = Car)) +
geom_boxplot(fill = "skyblue", color = "blue") +
labs(title = "Boxplot of Car Column",
y = "Car") +
theme_minimal()

# Create boxplot for "BuildingArea" column
ggplot(data, aes(y = BuildingArea)) +
geom_boxplot(fill = "lightgreen", color = "darkgreen") +
labs(title = "Boxplot of BuildingArea Column",
y = "Building Area") +
theme_minimal()
```

```
# Create boxplot for "YearBuilt" column
ggplot(data, aes(y = YearBuilt)) +
geom_boxplot(fill = "lightcoral", color = "red") +
labs(title = "Boxplot of YearBuilt Column",
y = "Year Built") +
theme_minimal()

```
```{r}
# Create a copy of the original dataframe
clean_data <- data

# Replace missing values in "Car" column with mean
mean_car <- mean(clean_data$Car, na.rm = TRUE)
clean_data$Car[is.na(clean_data$Car)] <- mean_car

# Replace missing values in "BuildingArea" column with mean
mean_building_area <- mean(clean_data$BuildingArea, na.rm =
   TRUE)
clean_data$BuildingArea[is.na(clean_data$BuildingArea)] <-
   mean_building_area

# Replace missing values in "YearBuilt" column with mean
mean_year_built <- mean(clean_data$YearBuilt, na.rm = TRUE)
clean_data$YearBuilt[is.na(clean_data$YearBuilt)] <- ifelse(
   mean_year_built >= 0, mean_year_built, 0)

# Check for any remaining missing values
colSums(is.na(clean_data))
```

```
```
3. Handle outliners
```{r}
# Select numeric variables
numeric_vars <- clean_data[sapply(clean_data, is.numeric)]

# Convert the data to long format for boxplot matrix
library(reshape2)
melted_data <- melt(numeric_vars)

# Create a matrix of boxplots with two boxplots per row
ggplot(melted_data, aes(x = variable, y = value)) +
geom_boxplot() +
facet_wrap(~variable, scales = "free", nrow = 1) +  # Set
   nrow = 1 to display two boxplots per row
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```
```{r}
# Check the distribution of Price
hist(clean_data$Price, main = "Histogram of Price", xlab = "
   Price", ylab = "Frequency")

```
a) Log transformation for Price
```{r}
# Log transformation
clean_data$Price <- log(clean_data$Price)

```

```r
# Check the distribution of log-transformed Price variable
hist(clean_data$Price, main = "Histogram of Log-transformed Price", xlab = "Log(Price)", ylab = "Frequency")
clean_data
```

b) Handle outliners for other features using quantile method

```{r}
# Define the function to detect and handle outliers using quantiles
handle_outliers <- function(data, k) {
    for (col in names(data)) {
        if (col %in% c("Longitude", "Latitude", "Price", "Suburb", "Address", "Type", "Method", "SellerG", "Date", "CouncilArea", "Regionname")) {
            next  # Skip specified columns
        }
        qnt <- quantile(data[[col]], probs = c(0.25, 0.75), na.rm = TRUE)
        iqr <- qnt[2] - qnt[1]
        upper <- qnt[2] + k * iqr
        lower <- qnt[1] - k * iqr
        data[[col]][data[[col]] > upper] <- upper
        data[[col]][data[[col]] < lower] <- lower
    }
    return(data)
}
```

```r
# Apply the function to handle outliers
house <- handle_outliers(clean_data, k = 3)

# Check for outliers by visualizing the distribution of each numeric variable
numeric_cols <- sapply(house, is.numeric)
numeric_house <- house[, numeric_cols]

par(mfrow = c(3, 3))  # Set the layout for multiple plots
for (col in names(numeric_house)) {
    if (col %in% c("Longitude", "Latitude")) {
        next  # Skip Longitude and Latitude columns
    }
    hist(numeric_house[[col]], main = col, xlab = col)
}
```

4. Correlation Analysis

```r
# Filter numeric columns
numeric_columns <- sapply(house, is.numeric)
house_numeric <- house[, numeric_columns]

# Calculate correlation matrix
correlation_matrix <- cor(house_numeric)
correlation_matrix

# Plot heatmap
library(ggplot2)
library(reshape2)
```

```r

# Melt correlation matrix
melted_correlation <- melt(correlation_matrix)

# Plot heatmap with circles of different sizes
ggplot(data = melted_correlation, aes(Var1, Var2)) +
geom_tile(aes(fill = value), color = "white") +
geom_point(aes(size = abs(value), fill = value), shape = 21, color = "black") +
scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1), space = "Lab", name="Correlation") +
scale_size_continuous(range = c(1, 8)) +  # Adjust the range for circle sizes
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 8, hjust = 1)) +
guides(size = FALSE) +
coord_fixed()

```

```r
## 5. Dummy variable
```r{r cache=TRUE}
# Select categorical columns
categorical_cols <- c( "Type", "Regionname")

# Create dummy variables for categorical columns
dummy_variables <- model.matrix(~ 0  + Type + Regionname, data = house)
```
```

```
181
182         # Create house_dummy data frame excluding the original
               categorical columns
183         house_dummy <- house[ , !(names(house) %in% categorical_cols
               )]
184
185         # Combine the dummy variables with house_dummy
186         house_dummy <- cbind(house_dummy, dummy_variables)
187
188         house_dummy
189         ```
190         ### Exclude unnecessary features
191         ```{r}
192         # Columns to exclude
193         cols_to_exclude <- c("Suburb", "Address", "Method", "SellerG
               ", "Date", "Postcode", "CouncilArea", "Regionname")
194
195         # Create Melb_house by excluding the specified columns
196         Melb_house <- house_dummy[ , !(names(house_dummy) %in% cols_
               to_exclude)]
197         names(Melb_house)
198         ```
199         # PERFORM REGRESSION
200
201         ## 1. MLR
202         ```{r}
203         # Load necessary libraries
204         library(caret)
205
```

```r
# Set seed for reproducibility
set.seed(123)

Melb_house <- Melb_house[, !names(Melb_house) %in% c("Typeu")]

# Define the function to perform MLR with 10-fold CV and evaluate the model
mlr_10_fold_cv <- function(data) {
    # Define 10-fold cross-validation
    folds <- createFolds(data$Price, k = 10)

    # Initialize a list to store evaluation metrics for each fold
    mlr_metrics <- list()

    # Perform 10-fold cross-validation
    for (i in 1:10) {
        train_index <- unlist(folds[-i])
        test_index <- unlist(folds[i])
        train_data <- data[train_index, ]
        test_data <- data[test_index, ]

        # Fit MLR model
        mlr_model <- lm(Price ~ ., data = train_data)

        # Predict on test set
        mlr_predicted <- predict(mlr_model, newdata = test_data)
```

```
231
232             # Evaluate the model
233             r_squared <- summary(mlr_model)$r.squared
234             rmse <- sqrt(mean((test_data$Price - mlr_predicted)
                    ^2))
235             aic <- AIC(mlr_model)
236             bic <- BIC(mlr_model)
237
238             # Store evaluation metrics
239             mlr_metrics[[i]] <- c(R_squared = r_squared, RMSE =
                    rmse, AIC = aic, BIC = bic)
240         }
241
242         # Convert the list of metrics to a data frame
243         mlr_metrics_df <- do.call(rbind, mlr_metrics)
244
245         # Calculate the average metrics
246         avg_metrics <- colMeans(mlr_metrics_df)
247
248         # Print average metrics
249         cat("Average Metrics:\n")
250         print(avg_metrics)
251
252         # Print summary of the MLR model
253         cat("\nSummary of the MLR Model:\n")
254         print(summary(mlr_model))
255     }
256
257     # Perform MLR with 10-fold CV and evaluate the model on Melb
```

```
   _house data
mlr_10_fold_cv(Melb_house)

```

```{r}
# Plot actual vs predicted value
# Create a dataframe with predicted and actual values
predicted_actual <- data.frame(
Observed = test_data$Price, # Actual values
Predicted = mlr_predicted # Predicted values
)

# Plot scatterplot
ggplot(predicted_actual, aes(x = Observed, y = Predicted)) +
geom_point(color = "blue", alpha = 0.5) +
geom_abline(intercept = 0, slope = 1, color = "red",
   linetype = "dashed") + # Add diagonal line for reference
labs(title = "MLR: Actual vs Predicted Prices",
x = "Actual Price",
y = "Predicted Price",
caption = "Dashed red line represents perfect prediction") +
    # Add caption
theme_minimal()

```

2. Ridge
```{r}
# Separate numerical variables from dummy variables
```

```
numeric_variables <- Melb_house[, -c(1, 14:21)]  # Exclude the Price column and dummy variables
dummy_variables <- Melb_house[, c(14:21)]        # Include only the dummy variables

# Scale numerical variables
scaled_numeric <- scale(numeric_variables)

# Combine scaled numerical variables with dummy variables
scaled_Melb_house <- cbind(scaled_numeric, dummy_variables)
names(scaled_Melb_house)
```
```{r}
# Split data into training and testing sets (80-20)
set.seed(123)
train_index <- createDataPartition(scaled_Melb_house$Price, p = 0.8, list = FALSE)
train_data <- scaled_Melb_house[train_index, ]
test_data <- scaled_Melb_house[-train_index, ]

# Perform 10-fold Cross-Validation to select lambda
cv <- cv.glmnet(as.matrix(train_data[, -1]), train_data[, 1], alpha = 0, nfolds = 10)

# Find optimal lambda based on minimum cross-validated error
best_lambda <- cv$lambda.min

# Print optimal lambda
cat("Optimal Lambda:", best_lambda, "\n")
```

```
309
310         # Fit final model with optimal lambda
311         R_final_model <- glmnet(as.matrix(train_data[, -1]), train_
               data[, 1], alpha = 0, lambda = best_lambda)
312
313         # Summary of the model
314         summary(R_final_model)
315         # Print coefficients of the final model
316         coef(R_final_model)
317
318         ```
319         ```{r}
320         # Predictions from the final model (Ridge)
321         ridge_predictions <- predict(R_final_model, newx = as.matrix
               (test_data[, -1]))
322
323         # Calculate R-squared for Ridge
324         ridge_R_squared <- cor(ridge_predictions, test_data[, 1])^2
325
326         # Calculate RMSE for Ridge
327         ridge_RMSE <- sqrt(mean((ridge_predictions - test_data[, 1])
               ^2))
328
329         # Print R-squared and RMSE for Ridge
330         cat("Ridge Regression R-squared:", ridge_R_squared, "\n")
331         cat("Ridge Regression RMSE:", ridge_RMSE, "\n")
332
333         # Number of observations
334         n <- nrow(test_data)
```

```
335
336         # Number of predictors (excluding the intercept)
337         p <- ncol(test_data) - 1
338
339         # Adjusted R-squared calculation
340         adjusted_r_squared <- 1 - ((1 - ridge_R_squared) * (n - 1))
               / (n - p - 1)
341
342         cat("Adjusted R-squared for Ridge regression:", adjusted_r_
               squared, "\n")
343
344         ```
345         ```{r}
346         # Plot actual vs predicted
347         # Fit final model with optimal lambda
348         R_final_model <- glmnet(as.matrix(train_data[, -1]), train_
               data[, 1], alpha = 0, lambda = best_lambda)
349
350         # Predictions from the final model (Ridge)
351         ridge_predictions <- predict(R_final_model, newx = as.matrix
               (test_data[, -1]))
352         Observed = test_data$Price
353
354         x<-predicted_actual$Observed
355         y<-ridge_predictions
356
357
358         # Plot scatter plot
359         plot(x, y,
```

```
        col = "blue",       # Set color to blue
        pch = 16,           # Set point shape to solid circle
        main = "Ridge regression: Actual vs Predicted Prices",  #
           Set main title
        xlab = "Actual Price",                # Set x-axis label
        ylab = "Predicted Price")             # Set y-axis label

        # Add diagonal line
        abline(a = 0, b = 1, col = "red", lty = 2)
        ```
        ## 3. LASSO

        ```{r}
        # Split data into training and testing sets (80-20)
        set.seed(123)
        train_index <- createDataPartition(scaled_Melb_house$Price,
           p = 0.8, list = FALSE)
        train_data <- scaled_Melb_house[train_index, ]
        test_data <- scaled_Melb_house[-train_index, ]

        # Perform 10-fold Cross-Validation to select lambda
        cv <- cv.glmnet(as.matrix(train_data[, -1]), train_data[,
           1], alpha = 1, nfolds = 10)  # Use alpha = 1 for Lasso

        # Find optimal lambda based on minimum cross-validated error
        best_lambda_index <- which.min(cv$cvm)
        best_lambda <- cv$lambda[best_lambda_index]

        # Print optimal lambda
```

```
cat("Optimal Lambda:", best_lambda, "\n")

# Fit final model with optimal lambda
L_final_model <- glmnet(as.matrix(train_data[, -1]), train_
   data[, 1], alpha = 1, lambda = best_lambda)  # Use alpha
   = 1 for Lasso

# Summary of the model
summary(L_final_model)

# Print coefficients of the final model
coef(L_final_model)

```
```{r}
# Predictions from the final model (Lasso)
lasso_predictions <- predict(L_final_model, newx = as.matrix
   (test_data[, -1]))

# Calculate R-squared for Lasso
lasso_R_squared <- cor(lasso_predictions, test_data[, 1])^2

# Calculate RMSE for Lasso
lasso_RMSE <- sqrt(mean((lasso_predictions - test_data[, 1])
   ^2))

# Print R-squared and RMSE for Lasso
cat("Lasso Regression R-squared:", lasso_R_squared, "\n")
cat("Lasso Regression RMSE:", lasso_RMSE, "\n")
```

```r

# Calculate R-squared for LASSO
lasso_R_squared <- cor(lasso_predictions, test_data[, 1])^2

# Number of observations
n <- nrow(test_data)

# Number of predictors (excluding the intercept)
p <- ncol(test_data) - 1

# Adjusted R-squared calculation
adjusted_r_squared_lasso <- 1 - ((1 - lasso_R_squared) * (n - 1)) / (n - p - 1)

cat("Adjusted R-squared for LASSO regression:", adjusted_r_squared_lasso, "\n")
```
```{r}
# Plot actual vs predicted
# Fit final model with optimal lambda
L_final_model <- glmnet(as.matrix(train_data[, -1]), train_data[, 1], alpha = 0, lambda = best_lambda)

# Predictions from the final model (Ridge)
lasso_predictions <- predict(L_final_model, newx = as.matrix(test_data[, -1]))
Observed = test_data$Price

x<-predicted_actual$Observed
y<-lasso_predictions
```

```
436
437
438         # Plot scatter plot
439         plot(x, y,
440         col = "blue",       # Set color to blue
441         pch = 1,           # Set point shape to solid circle
442         main = "LASSO regression: Actual vs Predicted Prices",  #
               Set main title
443         xlab = "Actual Price",                # Set x-axis label
444         ylab = "Predicted Price")             # Set y-axis label
445
446         # Add diagonal line
447         abline(a = 0, b = 1, col = "red", lty = 2)
448         ```
449         ## 4. Subset selection
450         ```{r}
451
452         # Define predictor variables
453         predictors <- Melb_house[, -c(2)]  # Remove the target
               variable column
454
455         # Perform subset selection
456         subset_model <- regsubsets(Price ~ ., data = Melb_house,
               nvmax = ncol(predictors))
457
458
459         # Get summary of the subset selection
460         subset_summary <- summary(subset_model)
461
```

```r
# Extract information for plotting
bic_values <- subset_summary$bic
aic_values <- subset_summary$aic
rsquared_adj_values <- subset_summary$adjr2
num_variables <- 1:ncol(predictors)

# Define predictor variables
predictors <- Melb_house[, -which(names(Melb_house) == "Price")]
total_predictors <- ncol(predictors)


# Fit linear regression models with different numbers of predictors
models <- list()
aic_values <- numeric()

for (i in 1:total_predictors) {
    predictor_indices <- c("Price", sample(names(predictors), i, replace = FALSE))
    models[[i]] <- lm(Price ~ ., data = Melb_house[, predictor_indices])
    aic_values[i] <- AIC(models[[i]])
}


# Plot AIC values against the number of variables
ggplot(data = data.frame(num_variables = 1:length(aic_values), AIC = aic_values), aes(x = num_variables, y = AIC)) +
```

```
486         geom_line(color = "red") +
487         geom_point(color = "red") +
488         labs(title = "Subset Selection with AIC", x = "Number of
               Variables", y = "AIC Value")
489
490         # Plot BIC values against the number of variables
491         ggplot(data = data.frame(num_variables, bic_values), aes(x =
                num_variables, y = bic_values)) +
492         geom_line(color = "blue") +
493         geom_point(color = "blue") +
494         labs(title = "Subset Selection with BIC", x = "Number of
               Variables", y = "BIC Value")
495
496         # Plot R-squared values against the number of variables
497         ggplot(data = data.frame(num_variables, rsquared_adj_values)
               , aes(x = num_variables, y = rsquared_adj_values)) +
498         geom_line(color = "green") +
499         geom_point(color = "green") +
500         labs(title = "Subset Selection with R-squared", x = "Number
               of Variables", y = "R squared_adj_values")
501
502         ```
503         ```{r}
504         # Create a dataframe to store the information
505         subset_info <- data.frame(
506         num_variables = num_variables,
507         bic_values = bic_values,
508         aic_values = aic_values
509         )
```

```
510
511         # Reshape the data frame for ggplot
512         subset_info_melt <- reshape2::melt(subset_info, id.vars = "
               num_variables")
513
514         # Plot AIC, BIC
515         ggplot(subset_info_melt, aes(x = num_variables, y = value,
               color = variable)) +
516         geom_line() +
517         geom_point() +
518         labs(title = "Subset Selection Metrics",
519         x = "Number of Variables",
520         y = "Metric Value",
521         color = "Metric") +
522         scale_color_manual(values = c("blue", "red"))  # Color for
               each metric
523
524         ```
525         ## 5. PCR
526         ```{r}
527         # Load required libraries
528         library(pls)
529         library(caret)
530
531         # Define predictor variables
532         predictors <- Melb_house[, -which(names(Melb_house) == "
               Price")]
533
534         # Perform Principal Component Regression (PCR)
```

```
pcr_model <- pcr(Price ~ ., data = Melb_house, scale = TRUE)

# Summary of PCR model
summary(pcr_model)

# Plot PCR model
plot(pcr_model, col = "orange",main = "PCR Model")

# Make predictions using PCR model
predictions <- predict(pcr_model, newdata = predictors)

# Evaluate model performance
RMSE <- sqrt(mean((Melb_house$Price - predictions)^2))
cat("Root Mean Squared Error (RMSE) of PCR model:", RMSE, "\
   n")

```

```{r}
# Load the 'pls' library if not already loaded
library(pls)

# Perform Cross-Validation to select optimal number of
   components
pcr_cv <- pcr(Price ~ ., data = Melb_house, scale = TRUE,
   validation = "CV")
summary(pcr_cv)

# Plot Cross-Validation error vs. number of components
```

```
validationplot(pcr_cv)

```
RESIDUAL ANALYSIS
1. MLR
```{r}
# Define the function to perform MLR with 10-fold CV and
   evaluate the model
mlr_10_fold_cv <- function(data) {
    # Define 10-fold cross-validation
    folds <- createFolds(data$Price, k = 10)

    # Initialize variables to store the best model and its
       metrics
    best_model <- NULL
    best_r_squared <- -Inf
    best_rmse <- Inf
    best_aic <- Inf
    best_bic <- Inf

    # Initialize lists to store residuals and predicted
       values
    best_residuals <- NULL
    best_predicted <- NULL

    # Perform 10-fold cross-validation
    for (i in 1:10) {
        train_index <- unlist(folds[-i])
        test_index <- unlist(folds[i])
```

```
587                 train_data <- data[train_index, ]
588                 test_data <- data[test_index, ]
589
590                 # Fit MLR model
591                 mlr_model <- lm(Price ~ ., data = train_data)
592
593                 # Predict on test set
594                 mlr_predicted <- predict(mlr_model, newdata = test_
                       data)
595
596                 # Evaluate the model
597                 r_squared <- summary(mlr_model)$r.squared
598                 rmse <- sqrt(mean((test_data$Price - mlr_predicted)
                       ^2))
599                 aic <- AIC(mlr_model)
600                 bic <- BIC(mlr_model)
601
602                 # Update best model if current model has higher R-
                       squared
603                 if (r_squared > best_r_squared) {
604                     best_r_squared <- r_squared
605                     best_model <- mlr_model
606                     best_rmse <- rmse
607                     best_aic <- aic
608                     best_bic <- bic
609                     best_residuals <- test_data$Price - mlr_
                           predicted
610                     best_predicted <- mlr_predicted
611                 }
```

```
612             }
613
614             # Plot residuals against predicted values for the best
                   model
615             plot(best_predicted, best_residuals, col = "red",
616             xlab = "Predicted values", ylab = "Residuals",
617             main = "MLR: Residuals vs Predicted values for the best
                   model")
618
619             # Print summary of the best MLR model
620             cat("\nSummary of the Best MLR Model:\n")
621             print(summary(best_model))
622
623             # Print RMSE, AIC, and BIC
624             cat("\nRMSE:", best_rmse, "\n")
625             cat("AIC:", best_aic, "\n")
626             cat("BIC:", best_bic, "\n")
627         }
628
629         # Apply mlr_10_fold_cv to Melb_house data
630         mlr_10_fold_cv(Melb_house)
631
632         ```
633         ## 2. Ridge
634         ```{r}
635         # Define the function to perform Ridge regression with 10-
               fold CV and evaluate the model
636         ridge_10_fold_cv <- function(data) {
637             # Load necessary libraries
```

```
638 library(caret)
639 library(glmnet)
640
641 # Define 10-fold cross-validation
642 folds <- createFolds(data$Price, k = 10)
643
644 # Initialize variables to store the best model and its
      metrics
645 best_model <- NULL
646 best_aic <- Inf
647 best_bic <- Inf
648 best_adj_r_squared <- -Inf
649 best_rmse <- Inf
650
651 # Initialize lists to store residuals and predicted
      values
652 best_residuals <- NULL
653 best_predicted <- NULL
654
655 # Perform 10-fold cross-validation
656 for (i in 1:10) {
657     train_index <- unlist(folds[-i])
658     test_index <- unlist(folds[i])
659     train_data <- data[train_index, ]
660     test_data <- data[test_index, ]
661
662     # Fit Ridge regression model
663     ridge_model <- glmnet(x = as.matrix(train_data[,
          -1]), y = train_data$Price, alpha = 0)
```

```
664
665             # Predict on test set
666             ridge_predicted <- predict(ridge_model, newx = as.
                   matrix(test_data[, -1]))
667
668             # Calculate AIC and BIC
669             n_obs <- nrow(test_data)
670             rss <- sum((test_data$Price - ridge_predicted)^2)
671             n_params <- sum(ridge_model$beta != 0)
672             aic <- n_obs * log(rss/n_obs) + 2 * n_params
673             bic <- n_obs * log(rss/n_obs) + n_params * log(n_obs
                   )
674
675             # Calculate number of predictors
676             n_predictors <- sum(ridge_model$beta != 0)
677
678             # Calculate R-squared
679             ss_residual <- sum((test_data$Price - ridge_
                   predicted)^2)
680             ss_total <- sum((test_data$Price - mean(test_data$
                   Price))^2)
681             r_squared <- 1 - (ss_residual / ss_total)
682
683             # Calculate adjusted R-squared
684             adj_r_squared <- 1 - ((1 - r_squared) * (n_obs - 1))
                    / (n_obs - n_predictors - 1)
685
686             # Calculate RMSE
687             rmse <- sqrt(mean((test_data$Price - ridge_predicted
```

```
    )^2))

688

689             # Update best model if current model has lower AIC
                   or BIC
690             if (aic < best_aic) {
691                 best_aic <- aic
692                 best_model <- ridge_model
693                 best_residuals <- test_data$Price - ridge_
                       predicted
694                 best_predicted <- ridge_predicted
695             }
696             if (bic < best_bic) {
697                 best_bic <- bic
698                 best_model <- ridge_model
699                 best_residuals <- test_data$Price - ridge_
                       predicted
700                 best_predicted <- ridge_predicted
701             }
702
703             # Update best adjusted R-squared if current model
                   has higher value
704             if (adj_r_squared > best_adj_r_squared) {
705                 best_adj_r_squared <- adj_r_squared
706             }
707
708             # Update best RMSE if current model has lower value
709             if (rmse < best_rmse) {
710                 best_rmse <- rmse
711             }
```

```
712             }
713
714             # Plot residuals against predicted values for the best
                   model
715             plot(best_predicted, best_residuals, col = "blue", xlab
                   = "Predicted values", ylab = "Residuals", main = "
                   Ridge: Residuals vs Predicted values for Best Model")
716
717             # Print summary of the best Ridge model
718             cat("\nSummary of the Best Ridge Model:\n")
719             print(best_model)
720             cat("\nBest AIC:", best_aic, "\n")
721             cat("Best BIC:", best_bic, "\n")
722             cat("Best RMSE:", best_rmse, "\n")
723         }
724
725         # Apply the function to scaled_Melb_house data
726         ridge_10_fold_cv(scaled_Melb_house)
727
728         ```
729         ## 3. LASSO
730         ```{r}
731         # Define the function to perform LASSO regression with 10-
               fold CV and evaluate the model
732         lasso_10_fold_cv <- function(data) {
733             # Load necessary libraries
734             library(caret)
735             library(glmnet)
736
```

```
737             # Define 10-fold cross-validation
738             folds <- createFolds(data$Price, k = 10)
739
740             # Initialize variables to store the best model and its
                    metrics
741             best_model <- NULL
742             best_aic <- Inf
743             best_bic <- Inf
744             best_adj_r_squared <- -Inf
745             best_rmse <- Inf
746
747             # Initialize lists to store residuals and predicted
                    values
748             best_residuals <- NULL
749             best_predicted <- NULL
750
751             # Perform 10-fold cross-validation
752             for (i in 1:10) {
753                 train_index <- unlist(folds[-i])
754                 test_index <- unlist(folds[i])
755                 train_data <- data[train_index, ]
756                 test_data <- data[test_index, ]
757
758                 # Fit LASSO regression model
759                 lasso_model <- glmnet(x = as.matrix(train_data[,
                        -1]), y = train_data$Price, alpha = 1)
760
761                 # Predict on test set
762                 lasso_predicted <- predict(lasso_model, newx = as.
```

```
    matrix(test_data[, -1]))

# Calculate AIC and BIC
n_obs <- nrow(test_data)
rss <- sum((test_data$Price - lasso_predicted)^2)
n_params <- sum(lasso_model$beta != 0)
aic <- n_obs * log(rss/n_obs) + 2 * n_params
bic <- n_obs * log(rss/n_obs) + n_params * log(n_obs
    )

# Calculate number of predictors
n_predictors <- sum(lasso_model$beta != 0)

# Calculate R-squared
ss_residual <- sum((test_data$Price - lasso_
    predicted)^2)
ss_total <- sum((test_data$Price - mean(test_data$
    Price))^2)
r_squared <- 1 - (ss_residual / ss_total)

# Calculate adjusted R-squared
adj_r_squared <- 1 - ((1 - r_squared) * (n_obs - 1))
     / (n_obs - n_predictors - 1)

# Calculate RMSE
rmse <- sqrt(mean((test_data$Price - lasso_predicted
    )^2))

# Update best model if current model has lower AIC
```

```
                or BIC
786             if (aic < best_aic) {
787                 best_aic <- aic
788                 best_model <- lasso_model
789                 best_residuals <- test_data$Price - lasso_
                        predicted
790                 best_predicted <- lasso_predicted
791             }
792             if (bic < best_bic) {
793                 best_bic <- bic
794                 best_model <- lasso_model
795                 best_residuals <- test_data$Price - lasso_
                        predicted
796                 best_predicted <- lasso_predicted
797             }
798
799             # Update best adjusted R-squared if current model
                    has higher value
800             if (adj_r_squared > best_adj_r_squared) {
801                 best_adj_r_squared <- adj_r_squared
802             }
803
804             # Update best RMSE if current model has lower value
805             if (rmse < best_rmse) {
806                 best_rmse <- rmse
807             }
808         }
809
810         # Plot residuals against predicted values for the best
```

```
        model
811     plot(best_predicted, best_residuals, col = "green", xlab
            = "Predicted values", ylab = "Residuals", main = "
        LASSO: Residuals vs Predicted values for Best Model")
812
813     # Print summary of the best LASSO model
814     cat("\nSummary of the Best LASSO Model:\n")
815     print(best_model)
816     cat("\nBest AIC:", best_aic, "\n")
817     cat("Best BIC:", best_bic, "\n")
818     cat("Best RMSE:", best_rmse, "\n")
819 }
820
821 # Apply the function to scaled_Melb_house data
822 lasso_10_fold_cv(scaled_Melb_house)
823
824 ```
825 ## 4. Subset selection
826 ```{r}
827 # Define the function to perform Subset Selection with 10-
        fold CV and evaluate the model
828 subset_selection_10_fold_cv <- function(data) {
829     # Load necessary libraries
830     library(caret)
831
832     # Define 10-fold cross-validation
833     folds <- createFolds(data$Price, k = 10)
834
835     # Initialize variables to store the best model and its
```

```
    metrics
best_model <- NULL
best_aic <- Inf
best_bic <- Inf
best_adj_r_squared <- -Inf
best_rmse <- Inf

# Initialize lists to store residuals and predicted
    values
best_residuals <- NULL
best_predicted <- NULL

# Perform 10-fold cross-validation
for (i in 1:10) {
    train_index <- unlist(folds[-i])
    test_index <- unlist(folds[i])
    train_data <- data[train_index, ]
    test_data <- data[test_index, ]

    # Fit Subset Selection model
    predictor_indices <- c("Price", sample(names(train_
        data)[-which(names(train_data) == "Price")],
        length(names(train_data)) - 1, replace = FALSE))
    subset_model <- lm(Price ~ ., data = train_data[,
        predictor_indices])

    # Predict on test set
    subset_predicted <- predict(subset_model, newdata =
        test_data)
```

```
859
860                 # Calculate AIC and BIC
861                 aic <- AIC(subset_model)
862                 bic <- BIC(subset_model)
863
864                 # Calculate R-squared
865                 r_squared <- summary(subset_model)$r.squared
866
867                 # Calculate adjusted R-squared
868                 n_obs <- nrow(test_data)
869                 n_predictors <- length(predictor_indices) - 1
870                 adj_r_squared <- 1 - ((1 - r_squared) * (n_obs - 1))
                        / (n_obs - n_predictors - 1)
871
872                 # Calculate RMSE
873                 rmse <- sqrt(mean((test_data$Price - subset_
                       predicted)^2))
874
875                 # Update best model if current model has lower AIC
876                 if (aic < best_aic) {
877                     best_aic <- aic
878                     best_model <- subset_model
879                     best_residuals <- test_data$Price - subset_
                           predicted
880                     best_predicted <- subset_predicted
881                 }
882
883                 # Update best BIC if current model has lower value
884                 if (bic < best_bic) {
```

```
            best_bic <- bic
        }

        # Update best adjusted R-squared if current model
          has higher value
        if (adj_r_squared > best_adj_r_squared) {
            best_adj_r_squared <- adj_r_squared
        }

        # Update best RMSE if current model has lower value
        if (rmse < best_rmse) {
            best_rmse <- rmse
        }
    }

    # Plot residuals against predicted values for the best
      model
    plot(best_predicted, best_residuals, col = "orange",xlab
        = "Predicted values", ylab = "Residuals", main = "
      Subset Selection: Residuals vs Predicted values for
      Best Model")

    # Print summary of the best Subset Selection model
    cat("\nSummary of the Best Subset Selection Model:\n")
    print(summary(best_model))
    cat("\nBest AIC:", best_aic, "\n")
    cat("Best BIC:", best_bic, "\n")
    cat("Best Adjusted R-squared:", best_adj_r_squared, "\n"
      )
```

```
        cat("Best RMSE:", best_rmse, "\n")
    }

    # Apply the function to Melb_house data
    subset_selection_10_fold_cv(Melb_house)

    ```
    ## 5. PCR
    ```{r}
    # Define the function to perform PCR with 10-fold CV and
       evaluate the model
    pcr_10_fold_cv <- function(data) {
        # Load necessary libraries
        library(caret)
        library(pls)

        # Define 10-fold cross-validation
        folds <- createFolds(data$Price, k = 10)

        # Initialize variables to store the best model and its
           metrics
        best_model <- NULL
        best_aic <- Inf
        best_bic <- Inf
        best_adj_r_squared <- -Inf
        best_rmse <- Inf

        # Initialize lists to store residuals and predicted
           values
```

```
934             best_residuals <- NULL
935             best_predicted <- NULL
936 
937             # Perform 10-fold cross-validation
938             for (i in 1:10) {
939                 train_index <- unlist(folds[-i])
940                 test_index <- folds[[i]]  # Access the fold
941                 train_data <- data[train_index, ]
942                 test_data <- data[test_index, ]
943 
944                 # Fit PCR model
945                 pcr_model <- pcr(Price ~ ., data = train_data, scale
                        = TRUE, validation = "CV")
946 
947                 # Predict on test set
948                 pcr_predicted <- predict(pcr_model, newdata = test_
                    data, ncomp = min(10, ncol(data) - 1))
949 
950                 # Calculate AIC and BIC
951                 n_obs <- nrow(test_data)
952                 rss <- sum((test_data$Price - pcr_predicted)^2)
953                 n_params <- min(10, ncol(data) - 1)
954                 aic <- n_obs * log(rss/n_obs) + 2 * n_params
955                 bic <- n_obs * log(rss/n_obs) + n_params * log(n_obs
                    )
956 
957                 # Calculate number of predictors
958                 n_predictors <- pcr_model$ncomp
959 
```

```
960                 # Calculate R-squared
961                 ss_residual <- sum((test_data$Price - pcr_predicted)
                       ^2)
962                 ss_total <- sum((test_data$Price - mean(test_data$
                       Price))^2)
963                 r_squared <- 1 - (ss_residual / ss_total)
964
965                 # Calculate adjusted R-squared
966                 adj_r_squared <- 1 - ((1 - r_squared) * (n_obs - 1))
                        / (n_obs - n_predictors - 1)
967
968                 # Calculate RMSE
969                 rmse <- sqrt(mean((test_data$Price - pcr_predicted)
                       ^2))
970
971                 # Update best model if current model has lower AIC
                       or BIC
972                 if (aic < best_aic) {
973                     best_aic <- aic
974                     best_model <- pcr_model
975                     best_residuals <- test_data$Price - pcr_
                           predicted
976                     best_predicted <- pcr_predicted
977                 }
978                 if (bic < best_bic) {
979                     best_bic <- bic
980                     best_model <- pcr_model
981                     best_residuals <- test_data$Price - pcr_
                           predicted
```

```
982                     best_predicted <- pcr_predicted
983                 }
984
985                 # Update best adjusted R-squared if current model
                       has higher value
986                 if (adj_r_squared > best_adj_r_squared) {
987                     best_adj_r_squared <- adj_r_squared
988                 }
989
990                 # Update best RMSE if current model has lower value
991                 if (rmse < best_rmse) {
992                     best_rmse <- rmse
993                 }
994             }
995
996             # Plot residuals against predicted values for the best
                   model
997             plot(best_predicted, best_residuals, col = "purple",
                   xlab = "Predicted values", ylab = "Residuals", main =
                    "PCR: Residuals vs Predicted values for Best Model")
998
999             # Print summary of the best PCR model
1000            cat("\nSummary of the Best PCR Model:\n")
1001            print(best_model)
1002            cat("\nBest AIC:", best_aic, "\n")
1003            cat("Best BIC:", best_bic, "\n")
1004            cat("Best Adjusted R-squared:", best_adj_r_squared, "\n"
                   )
1005            cat("Best RMSE:", best_rmse, "\n")
```

```
}

# Apply the function to Melb_house data
pcr_10_fold_cv(Melb_house)

```
```
```

References

[1] Smith, J., Johnson, M., & Williams, K. (2018). Neighborhood Characteristics and Housing Prices: Evidence from Australia.

[2] Chen, L., Liu, H., & Wang, Q. (2020). The Impact of Neighborhood Amenities on Housing Prices: Evidence from Melbourne, Australia.

[3] Li, H., & Brown, M. (2019). The Role of Property Characteristics in Housing Price Determination: Evidence from Melbourne, Australia.

[4] Faraway, J. J. (2016). Extending the Linear Model with R. CRC Press.