USED CAR PRICE PREDICTION

Introduction

In today's market where prices seem to skyrocket for everything, getting the right price for your vehicle becomes crucial. If you overprice it, you might scare off potential buyers. But if you underprice it, you're losing your hard-earned money. Finding that sweet spot is tough and requires a lot of research to nail down the perfect price. That's where our project comes in. This project aims to take the guesswork out of pricing your vehicle accurately. Whether you're selling or buying, having this tool at your disposal can make the process smoother and stress-free.

Research Question

What if we could analyze all the used car sales data from the past and create a model that would give you an estimated price at which your used car will sell?

Methods

To achieve the objective of creating a robust model capable of predicting the selling price of a vehicle, we followed a systematic approach encompassing data collection, data preprocessing, feature engineering, model selection, and model evaluation.

Data collection

We obtained historical vehicle sales data from Craigslist, sourced from the publicly available dataset on Kaggle. The dataset contains 26 columns and 426,880 rows. Each column represents a specific attribute of the vehicle listings.

Data preprocessing

In our data preprocessing phase, we focused on ensuring the quality and consistency of our dataset to enhance model performance. We employed various data-cleaning techniques, including handling missing values, removing outliers, and standardizing data formats. To address missing values, we dropped columns with a high number of null values and removed

rows containing null values for other columns. We addressed data imbalance in categorical features (Extended data fig 1) by aggregating less frequent categories into a single group, resulting in a more balanced distribution(Extended data fig 2). Numerical features underwent similar data cleaning, with outliers removed and data distribution enhanced by focusing on values within the interquartile range(Extended data fig 3). To prepare the dataset for modelling, categorical data was converted to numerical using the model.matrix() function, and numerical values were standardized using min-max scaling. These preprocessing steps ensure the dataset is optimized for accurate model training and interpretation.

Exploratory Data Analysis

We uncovered several key factors that significantly influence the used car market. Factors such as the model, manufacturer, age, fuel type, and condition were found to have a notable impact on car prices.

Model selection

For training and testing our model, we split our model into training and testing data. The training data now contains 80 percentage of the original dataset and the rest is test data. The subsequent analysis was conducted using three primary regression models:

Linear Regression: As our goal is to predict the price and it is a regression task decided to go with Linear regression because of its simplicity. An added benefit to choosing this model we could extend it to get better results by regularization techniques.

Ridge Regression: We tried this model to see if there is much difference in prediction with this model compared to the earlier one. We gave parameters as alpha 0 and lambda 0.1 for the ridge regression model.

Decision Tree Regression: This model offers a non-linear approach capable of capturing complex patterns through a hierarchical, tree-structured model. We tried to fit our data to this model with the method “anova” and recorded the calculated MSE for model comparisons.

Results

Linear Regression Model: The Linear Regression model demonstrated a mean squared error value of 89618214.65, indicating a relatively good model fit. Additionally, the plot depicting the predicted values against the actual prices for the test data illustrates a close alignment of the predicted points with the actual points along the diagonal line, as depicted in Extended Figure 4.

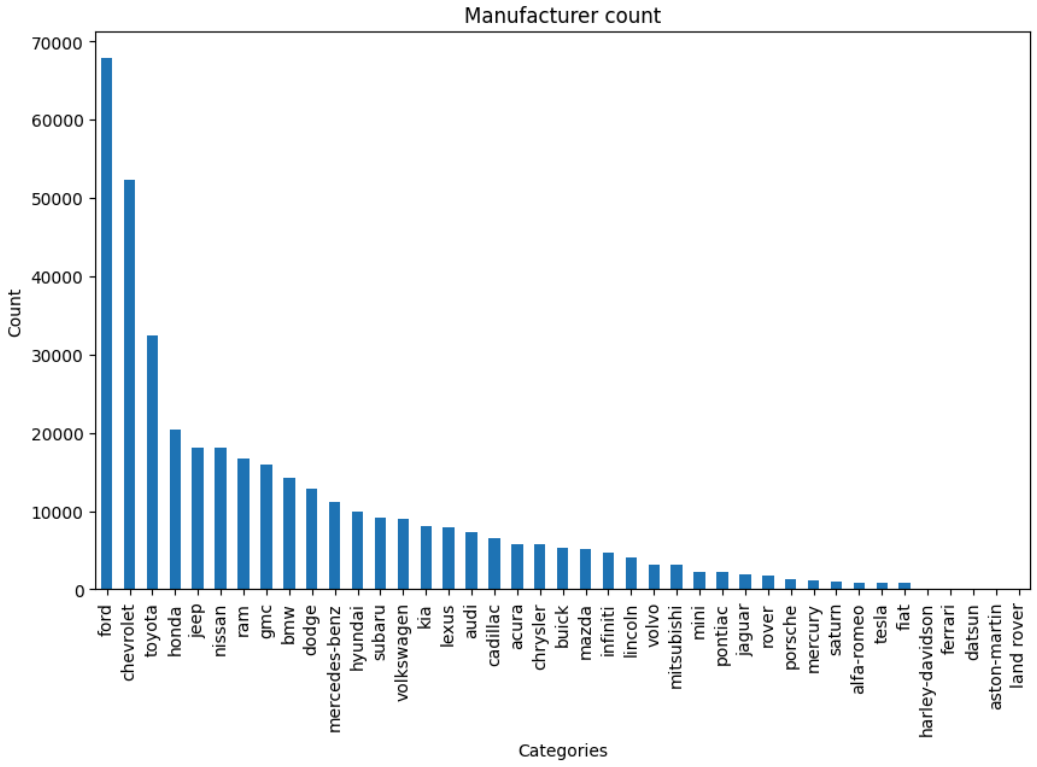
Ridge Regression Model: In contrast, the Ridge Regression model produced a higher mean squared error of 89846928.49. This comparatively higher error suggests a less optimal model performance. Furthermore, analysis of the prediction with the test dataset revealed numerous mispredictions, indicating that the model did not meet expectations. These findings are elaborated upon in Extended Figure 5.

Regression Tree Model: An additional analysis using a Regression Tree model yielded a mean squared error of approximately 114171075.91. This result suggests a comparable level of accuracy to the Linear Regression model, further highlighting the effectiveness of the approach in predicting vehicle prices based on the provided features. These findings are elaborated upon in Extended Figure 6.

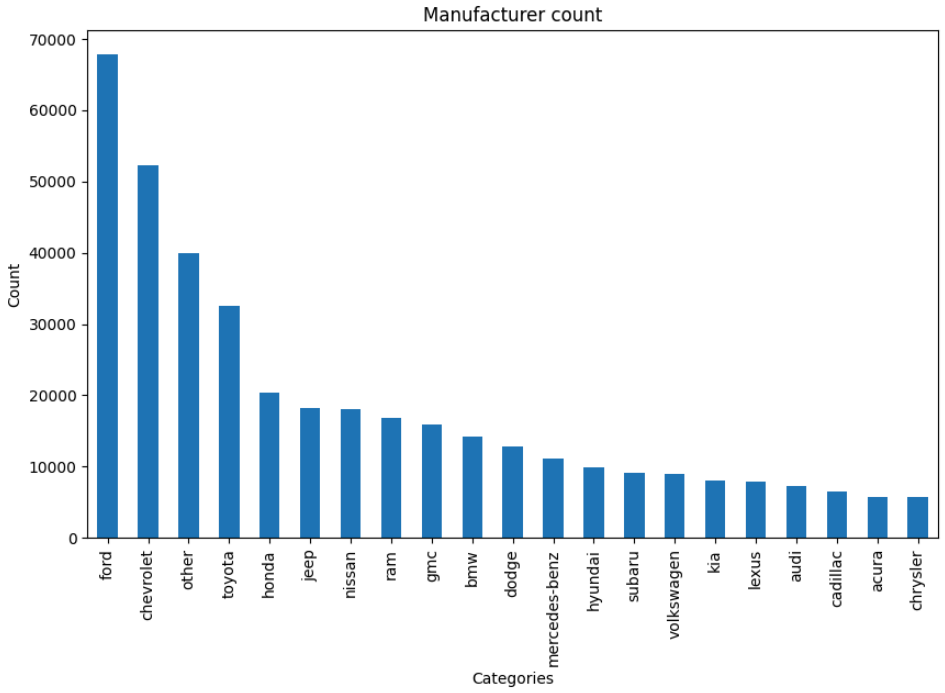
Conclusion

This project highlights the importance of predictive analytics in the used car market, aiding both sellers and buyers. While our models offer valuable insights, they reveal the complexities of price prediction in a dynamic market. Future improvements may involve advanced models like Random Forests and Gradient Boosting Machines, along with integrating detailed data such as maintenance history and regional economic indicators for better accuracy. Despite challenges, leveraging machine learning for used car price prediction shows promise in addressing real-world economic scenarios.

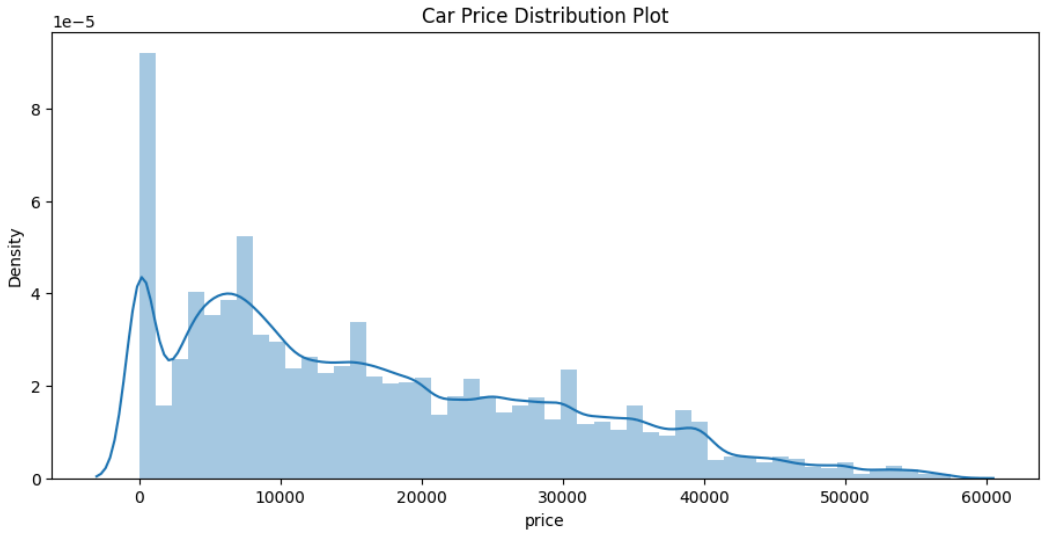
Appendix 1: Figures



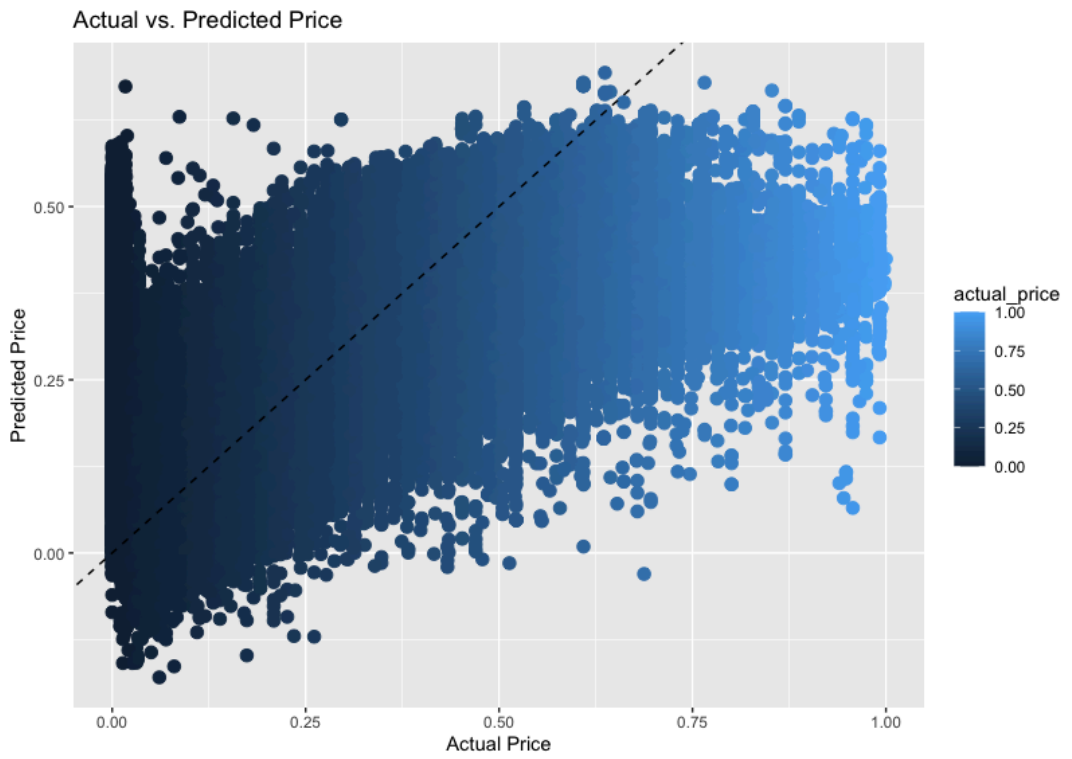
Extended data fig 1 | Bar plot showing count manufacturers of the vehicles



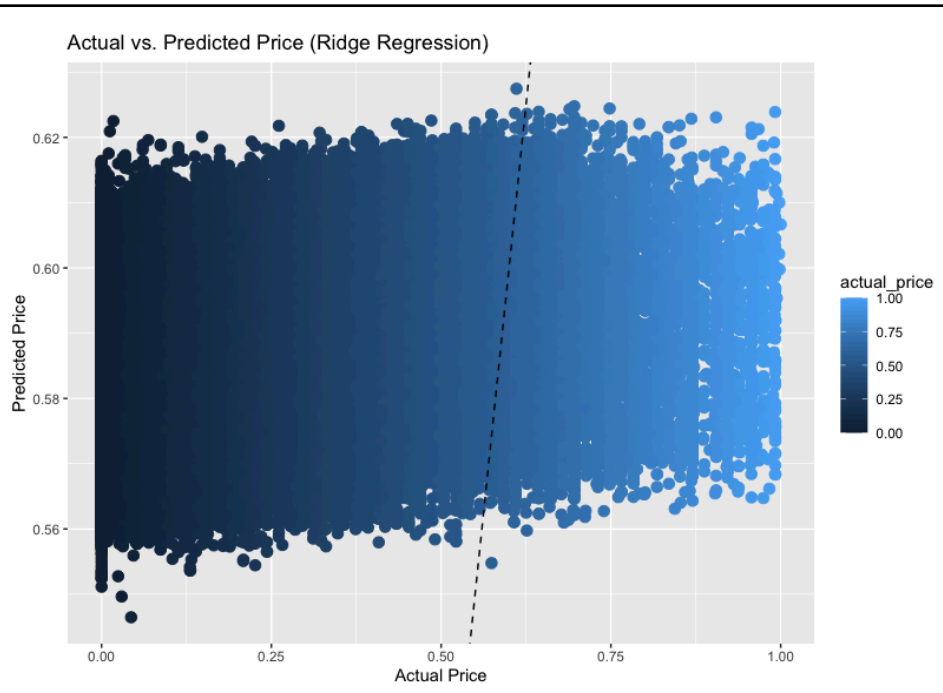
Extended data fig 2 | Bar plot showing count manufacturers of the vehicles after cleaning



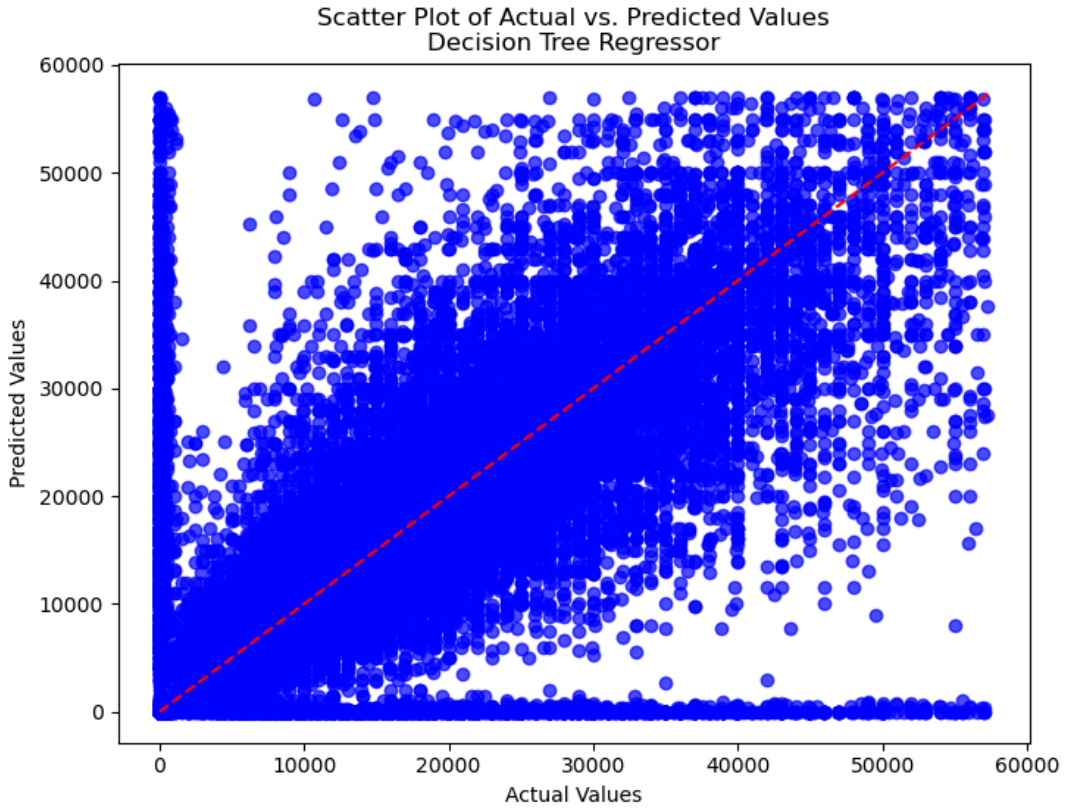
Extended data fig 3 | Price distribution of the vehicles before cleaning.



Extended data fig 4 |Linear regression Predicted price vs actual price.



Extended data fig 5 |Ridge regression Predicted price vs actual price.



Extended data fig 6 |Decision Tree Regression Predicted price vs actual price.

Appendix 1: R Code

```
library(ggplot2)

data <- read.csv("~/Desktop/CourseWorks/Predictive\ analysis\ /vehicles.csv")
summary(data)

#removing county since it does not contain any data
data <- data[, !(names(data) == "county")]
summary(data)

# We can safely remove Id as well since the values are non usable
hist(data$id, main = "Histogram of ID", xlab = "Values")
data <- data[, !(names(data) == "id")]
summary(data)

# posting_date, VIN, description, image_url, URL and region IRL are another features we
can't make use of. Dropping them
data <- data[, !(names(data) %in% c("url", "region_url", "image_url", "VIN", "description",
"posting_date"))]

na_count <- colSums(is.na(data))
print(na_count)

#checking for NA in data
if (any(is.na(data))) {
  df_clean <- na.omit(data)
}
```

```
summary(df_clean)


# Let's plot the DF based on lat and long
df_local_lat_long <- df_clean[df_clean$lat > 24 & df_clean$lat < 50 & df_clean$long > -125
& df_clean$long < -65, ]
ggplot(data = df_local_lat_long, aes(x = long, y = lat)) +
  geom_point() +
  labs(title = "Region Compare") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))


# Cleaning Manufacturer Column
# Get the counts of unique values in the 'manufacturer' column
manufacturer_counts <- table(df_clean$manufacturer)
barplot(manufacturer_counts,
        main = "Manufacturer Count",
        xlab = "Manufacturer",
        ylab = "Count",
        col = "skyblue",
        las = 2)
# Get the counts of unique values in the 'manufacturer' column
manufacturer_counts <- table(df_clean$manufacturer)
# Get the top 20 manufacturers
top_manufacturers <- names(sort(manufacturer_counts, decreasing = TRUE))[1:20]
# Recode 'manufacturer' column
df_clean$manufacturer <- ifelse(df_clean$manufacturer %in% top_manufacturers,
df_clean$manufacturer, "other")
manufacturer_counts <- table(df_clean$manufacturer)
```

```
barplot(manufacturer_counts,
        main = "Manufacturer Count",
        xlab = "Manufacturer",
        ylab = "Count",
        col = "skyblue",
        las = 2)

# Cleaning Region column
# Get the counts of unique values in the 'region' column
region_counts <- table(df_clean$region)

# Get the top 50 regions
top_regions <- names(sort(region_counts, decreasing = TRUE))[1:50]

# Recode 'region' column
df_clean$region <- ifelse(df_clean$region %in% top_regions, df_clean$region, "other")
# Get the counts of unique values in the 'region' column after recoding
region_counts_after_recode <- table(df_clean$region)
# Create a bar plot
barplot(region_counts_after_recode,
        main = "Top 50 Regions Count",
        xlab = "Region",
        ylab = "Count",
        col = "skyblue",
        las = 2)  # Rotate x-axis labels vertically

# Cleaning Model column
# Get the counts of unique values in the 'model' column
```

```
model_counts <- table(df_clean$model)

# Get the top 50 models
top_models <- names(sort(model_counts, decreasing = TRUE))[1:50]

# Recode 'model' column
df_clean$model <- ifelse(df_clean$model %in% top_models, df_clean$model, "other")

# Get the counts of unique values in the 'model' column after recoding
model_counts_after_recode <- table(df_clean$model)

# Create a bar plot
barplot(model_counts_after_recode,
        main = "Top 50 Models Count",
        xlab = "Model",
        ylab = "Count",
        col = "skyblue",
        las = 2)  # Rotate x-axis labels vertically

# Plot of price distribution
# Set the figure size
options(repr.plot.width=11, repr.plot.height=5)

# Create a histogram with kernel density estimate
hist(df_clean$price,
     main = "Car Price Distribution Plot",
     xlab = "Price",
     ylab = "Density",
```

```
    col = "skyblue",
    freq = FALSE)  # Set freq = FALSE to plot density instead of frequency

# Add a kernel density estimate
lines(density(df_clean$price), col = "red", lwd = 2)

# Add a legend
legend("topright", legend = c("Density", "Kernel Density Estimate"), fill = c("skyblue", "red"))
# removing outliers
col <- "price"
q1 <- quantile(df_clean[[col]], 0.25)
q3 <- quantile(df_clean[[col]], 0.75)
iqr <- q3 - q1
lower_limit <- q1 - 1.5 * iqr
upper_limit <- q3 + 1.5 * iqr
filtered_df <- df_clean[df_clean[[col]] > lower_limit & df_clean[[col]] < upper_limit, ]
# Create a histogram with kernel density estimate
hist(filtered_df$price,
    main = "Car Price Distribution Plot",
    xlab = "Price",
    ylab = "Density",
    col = "skyblue",
    freq = FALSE)  # Set freq = FALSE to plot density instead of frequency

# Add a kernel density estimate
lines(density(filtered_df$price), col = "red", lwd = 2)

# Add a legend
```

```
legend("topright", legend = c("Density", "Kernel Density Estimate"), fill = c("skyblue", "red"))


# Same way Removing ouliers for year and odomoter columns
cols <- c("odometer", "year")


# Loop through each column
for (col in cols) {
  # Calculate quantiles
  q1 <- quantile(filtered_df[[col]], 0.25)
  q3 <- quantile(filtered_df[[col]], 0.75)


  # Calculate IQR
  iqr <- q3 - q1


  # Calculate lower and upper limits
  lower_limit <- q1 - 1.5 * iqr
  upper_limit <- q3 + 1.5 * iqr


  # Filter dataframe to remove outliers
  filtered_df <- filtered_df[filtered_df[[col]] > lower_limit & filtered_df[[col]] < upper_limit, ]
}
# Plotting them
# Plot the distribution of 'odometer'
hist(filtered_df$odometer,
     main = "Odometer Distribution Plot",
     xlab = "Odometer",
     ylab = "Frequency",
     col = "skyblue",
```

```
    freq = FALSE)  # Set freq = FALSE to plot density instead of frequency

# Add a density plot
lines(density(filtered_df$odometer), col = "red", lwd = 2)

# Add a legend
legend("topright", legend = c("Density", "Kernel Density Estimate"), fill = c("skyblue", "red"))

# Plot the distribution of 'year'
hist(filtered_df$year,
    main = "Year Distribution Plot",
    xlab = "Year",
    ylab = "Frequency",
    col = "skyblue",
    freq = FALSE)  # Set freq = FALSE to plot density instead of frequency

# Add a density plot
lines(density(filtered_df$year), col = "red", lwd = 2)

# Add a legend
legend("topright", legend = c("Density", "Kernel Density Estimate"), fill = c("skyblue", "red"))

summary(filtered_df)

write.csv(filtered_df,"~/Downloads/filtered_df.csv")
#Model Fitting
filtered_df <- read.csv("~/Desktop/CourseWorks/Predictive\ analysis\ /filtered_df.csv")
summary(filtered_df)
```

```
#Making everything neumerical
filtered_df[] <- data.matrix(filtered_df)
filtered_df

filtered_df = subset(filtered_df, select = -c(X, lat, long) ) #Removed unwanted column
colnames(filtered_df)

#df_normalized <- scale(filtered_df)
#print(df_normalized)
normalize_column <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
df_normalized <- apply(filtered_df, 2, normalize_column)
df_normalized <- as.data.frame(df_normalized)
summary(df_normalized)

library(caret)

set.seed(123)  # For reproducibility
training_index <- sample(1:nrow(df_normalized), size = 0.8 * nrow(df_normalized))
training_data <- df_normalized[training_index, ]
testing_data <- df_normalized[-training_index, ]

#Linear regression Model
model <- lm(price ~ ., data = training_data)
predictions <- predict(model, newdata = testing_data)
mse <- mean((testing_data$price - predictions)^2)
```

```
print(paste("Mean Squared Error:", mse))
summary(model)

library(ggplot2)
data_for_plot <- data.frame(
  actual_price = testing_data$price,  # Replace with your actual price column name
  predicted_price = predictions
)
ggplot(data_for_plot, aes(x = actual_price, y = predicted_price)) +
  geom_point(aes(color = actual_price), size = 3) +  # Color points by actual price
  labs(title = "Actual vs. Predicted Price",
       x = "Actual Price",
       y = "Predicted Price") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed")



# Fit ridge regression model
any(is.na(training_data))
training_matrix <- as.matrix(training_data)
ridge_model <- glmnet(x = training_matrix[, -1], y = training_matrix[, 1], alpha = 0, lambda =
0.1)
ridge_predictions <- predict(ridge_model, newx = as.matrix(testing_data[, -1]))
ridge_mse <- mean((testing_data$price - ridge_predictions)^2)
print(paste("Ridge Regression Mean Squared Error:", ridge_mse))
ridge_data_for_plot <- data.frame(
  actual_price = testing_data$price,
  predicted_price = ridge_predictions
)
```

```
ridge_data_for_plot$predicted_price <- ridge_predictions



# Check the structure of ridge_data_for_plot
str(ridge_data_for_plot)

ggplot(ridge_data_for_plot, aes(x = actual_price, y = predicted_price)) +
  geom_point(aes(color = actual_price), size = 3) +
  labs(title = "Actual vs. Predicted Price (Ridge Regression)",
       x = "Actual Price",
       y = "Predicted Price") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed")

# decision Tree Regressor
library(dplyr)  # for data manipulation
library(rpart)  # for decision tree regression
library(rattle)  # for plotting the decision tree (optional)
library(rpart.plot)

model <- rpart(price ~ ., data = training_data, method = "anova")  # Regression method
predictions <- predict(model, newdata = testing_data)
mse <- mean((testing_data$price - predictions)^2)
print(paste("Mean Squared Error:", mse))

rpart.plot(model, main = "Decision Tree for Price Prediction")

data_for_plot <- data.frame(
  actual_price = testing_data$price,  # Replace with your actual price column name
```

```
  predicted_price = predictions
)

# 3. Create the scatter plot (same code as before)
ggplot(data_for_plot, aes(x = actual_price, y = predicted_price)) +
  geom_point(aes(color = actual_price), size = 3, alpha = 0.5) +  # Adjust alpha for
transparency
  labs(title = "Actual vs. Predicted Price (Decision Tree)",
       x = "Actual Price",
       y = "Predicted Price") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed")
```