Used Car Price Prediction

Charleston Wang, clwang5
Sona Krishnan, sonavk2
John Valdivieso, johnav3

Abstract

The primary goal of this project is to predict used car prices based on a dataset of 4,009 car listings. We aimed to develop predictive models that not only achieve high accuracy but also provide insights into the key factors influencing car pricing.

We began with data preprocessing, including imputation, feature scaling, and transformation. Three clustering algorithms: K-means, Hierarchical Clustering, and Spectral Clustering were implemented to segment the dataset and identify patterns. For regression, we developed and tuned five models: Ordinary Least Squares, K-Nearest Neighbors, Random Forest, Support Vector Regression, and XGBoost. Advanced hyperparameter tuning and cross-validation were used to optimize model performance. Feature importance analyses were also conducted to reveal key predictors of car pricing.

Key Findings

- **Best Model**: Random Forest exhibited the best performance with the lowest RMSE (0.669) and highest R_Squared(0.411).
- Feature importance analysis identified mileage, horsepower, and engine displacement as the top predictors of car price.
- Clustering techniques revealed distinct groupings of cars based on age, mileage, and pricing.

Our approach extended beyond the techniques reviewed in the literature review. Unlike the first paper, which focused on lasso regression, multiple regression, and regression trees, we applied advanced ensemble methods like XGBoost and Random Forest. These models leverage tree-based approaches with built-in regularization and feature selection, allowing for improved performance on high-dimensional data.
Additionally, we employed spectral clustering, a graph-based method that effectively identifies complex patterns in non-linear data. This contrasts with the K-means approach in the second paper, which assumes linear separability. Our feature importance analysis provided a deeper understanding of the predictors, which were not explored in the reviewed papers.

How These Methods Differ:

- **XGBoost and Random Forest**: Unlike lasso regression and regression trees, these models optimize using iterative boosting or randomization, enabling higher accuracy and better handling of outliers.
- **Spectral Clustering**: More robust for datasets with non-linear relationships compared to traditional K-means clustering.

AI Usage

AI tools were used in this project to enhance efficiency and accuracy. Specifically:

- Code generation and debugging support were partially provided by tools like ChatGPT, ensuring adherence to best practices.
- Grammar and structure checks for the report were conducted using Grammarly.
  All AI usage adhered to the university's academic integrity policy, ensuring that conceptual and technical contributions were student-driven.

Team Members

- Charleston Wang, clwang5
- Sona Krishnan, sonavk2
- John Valdivieso, johnav3

Literature Review

In the first paper we analyzed [1], the researchers chose to create three separate models to predict used car price. These models were built using lasso regression, multiple regression, and a regression tree with error rates of 3.581%, 3.468%, and 3.512% respectively. Through the use of anova tables they were able to create a test that confirmed that there was a significant difference in these error rates.They then conducted a Tukey Test which allowed them to conclude that there was no significant difference between the error rates of lasso and multiple regression, but there was a significant difference in the error rates between those two models and the one generated through a regression tree. The researchers concluded that the best fitting models were created using lasso regression and multiple regression. The dataset used contained prices from the 2005 Central Edition of the Kelly Blue Book, which shares some predictors with our dataset, such as model and mileage. Through random sampling, the researchers split this data into a training set containing 70% of the observations, and a testing set containing 30% of the observations. Throughout the model building process they checked for outliers with high leverage to make sure that there were no data points that misled the models.

In the second paper we analyzed [2], the main method used was K-nearest neighbor. After processing the data, the researchers tested the accuracy of different models obtained from this method with values of K 2 through 10, and three different training and testing data ratios. They concluded that the most accurate model was achieved through a K value of 4, through a ratio of 85% Training data and 15% testing data. The resulting accuracy was 85%. To confirm that K = 4 was ideal for this dataset, they also conducted cross-validation with 5 and 10 folds. The researchers were able to confirm that their K = 5 model was the optimized model for this method. The dataset used, like the one in the previous paper, shared the predictors model and mileage with our dataset, however this dataset also shares fuel type and transmission.

[1] Venkatasubbu, Pattabiraman, and Mukkesh Ganesh. "Used cars price prediction using supervised learning techniques." Int. J. Eng. Adv. Technol.(IJEAT) 9.1S3 (2019).

[2] Samruddhi, K., and R. Ashok Kumar. "Used car price prediction using K-nearest neighbor based model." Int. J. nov. ReIns. Appl. Sci. Eng.(IJIRASE) 4.3 (2020): 2020-686.

Data Processing

This section outlines the steps taken to clean and prepare the dataset for analysis, ensuring its readiness for accurate insights and model building. The preprocessing process includes handling missing values to maintain data integrity, transforming unstructured variables into usable formats, encoding categorical data for compatibility with analysis techniques, and standardizing numerical features to ensure consistency across all variables. These steps aim to resolve data quality issues and optimize the dataset for effective exploration and predictive modeling.

Step 1: Handle and Check for Missing Values

```
#Convert empty strings to NA
data$fuel_type[data$fuel_type == "" | data$fuel_type == "–"] <- NA
data$accident[data$accident == ""] <- NA
data$clean_title[data$clean_title == ""] <- NA
data$ext_col[data$ext_col == "–"] <- NA
data$int_col[data$int_col == "–"] <- NA

#Replace NA Values
data$fuel_type[is.na(data$fuel_type)] <- "not supported"
data$accident[is.na(data$accident)] <- "None reported"
data$clean_title[is.na(data$clean_title)] <- "No"

# Impute missing values in ext_color and int_color with mode
mode_ext_col <- names(sort(table(data$ext_col), decreasing = TRUE))[1]
mode_int_col <- names(sort(table(data$int_col), decreasing = TRUE))[1]
```

Missing values were represented by empty strings (""), special characters ("–"), or NA. These inconsistencies were unified by treating them as missing data for a consistent approach to cleaning.

- fuel_type: Missing values were replaced with "not supported" to preserve rows as I noticed that most of the empty string values were for electric cars, and the not supported values were also for electric cars.
- accident: Missing values were replaced with "None reported", indicating no known accidents or damages.
- clean_title: missing values were replaced with "No" to denote that the title is not clean as all other values were yes.
- ext_col and int_col: Missing values were imputed with the most frequent value of each column. This approach minimizes bias and preserves consistency.

These replacements ensured that all rows were retained, preserving the integrity of the dataset while minimizing bias introduced by imputation. To confirm, the unique values of the cleaned columns and counts of remaining missing values were inspected.

Step 2: Transform Text Variables

```
#Convert mileage
data$milage <- as.numeric(gsub("[^0-9]", "", data$milage))

#Extract horsepower from engine column
data$horsepower <- as.numeric(gsub(".*?(\\d+\\.\\d+)[Hh][Pp].*", "\\1",
data$engine))

#extract enginge displacement from engine column
data$engine_displacement <- as.numeric(gsub(".*?(\\d+\\.\\d+)[Ll].*", "\\1",
data$engine))

#convert price
data$price <- as.numeric(gsub("[^0-9]", "", data$price))
```

Several columns contained unstructured text data, which were cleaned and converted into usable formats.

- Mileage: Non-numeric characters (e.g., "mi") were removed using regular expressions, and the column was converted to numeric for further analysis.
- Horsepower and Engine Displacement were extracted from unstructured text in the engine column using regular expressions, creating two new numeric columns.

- Price: Similar cleaning was applied to the price column, removing non-numeric characters like commas or currency symbols.

```
# Predict horsepower using linear regression
horsepower_model <- lm(horsepower ~ model_year + fuel_type + milage, data =
data, na.action = na.exclude)

data$horsepower[is.na(data$horsepower)] <- predict(horsepower_model, newdata
= data[is.na(data$horsepower), ])

# Predict engine displacement using linear regression
engine_disp_model <- lm(engine_displacement ~ model_year + fuel_type +
milage, data = data, na.action = na.exclude)

data$engine_displacement[is.na(data$engine_displacement)] <-
predict(engine_disp_model, newdata = data[is.na(data$engine_displacement), ])
```

After extracting the horsepower and enginge displacement from the engine column a large number of NA values were introduced. However, these 2 variables are critical in understanding the car's performance and price. Having missing values for these features could significantly impact clustering and regression results. To address this, we used regression modeling for imputation because:

- Horsepower and engine_displacement are tied to features like model_year, fuel_type and mileage
- Using imputation by regression models can leverage these relationships to generate contextually meaningful estimates
- By using this approach, we retain all the rows and avoid potential biases that could be introduced by dropping rows.

Steps:
We built a linear regression model horsepower as the dependent variable and model_year, fuel_type, and mileage as predictors. We used a similar approach for engine_displacement using the same predictors. The dataset was then rechecked to ensure that there were no longer any missing values.

Step 3: Handling Categorical Variables
Exterior and interior car colors were grouped into broader, standardized categories to simplify analysis and reduce variability in the dataset. The grouping process involved categorizing colors into eight main categories: Black, White, Gray, Blue, Red, Beige, Brown, and Rare. This was achieved using the grepl function, which performs pattern matching to identify colors associated with each category based on descriptive text.

By grouping colors into these broader categories, the dataset becomes more manageable for statistical modeling and visualization. It reduces the dimensionality of the data, avoids overfitting caused by overly specific categories, and enhances interpretability for both clustering and predictive models. Additionally, grouping by broader categories aligns with real-world insights, where subtle variations in shades (e.g., "Carbon Black" vs. "Jet Black") are unlikely to drastically influence customer perceptions or market trends beyond the broader category.

Similarly for transmission, types were categorized into "Automatic", "Dual-Clutch", "Manual", "Single-Speed", and "Other", simplifying the analysis by consolidating similar types.

Categorical variables (fuel_type, transmission_grouped, ext_color_grouped, int_color_grouped) were then one-hot encoded using model.matrix. Binary variables (clean_title, accident) were encoded as 1 (Yes/Reported) and 0 (No/None reported). We decided to use one hot encoding as it transforms categorical data into a format compatible with clustering and regression models. This allowed models like K-means and spectral clustering to process categorical information along with numerical features. For regression models, the binary encoding ensured models like OLS and Random Forest could interpret these variables as numerical values. Grouping color and transmission also reduced multicollinearity issues that might arise.

Step 4: Standardize variables

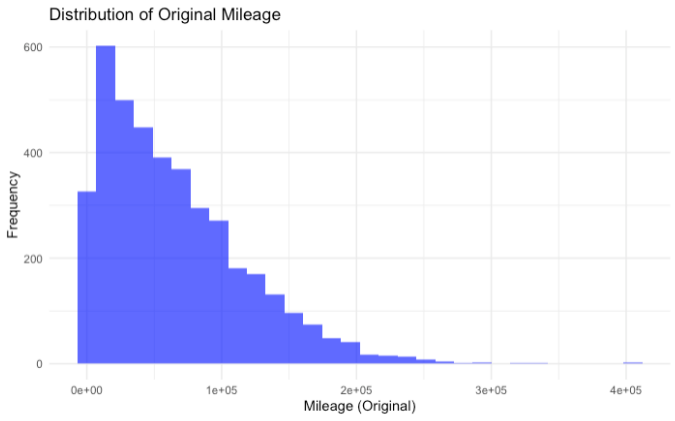
```
# One-hot encoding
```

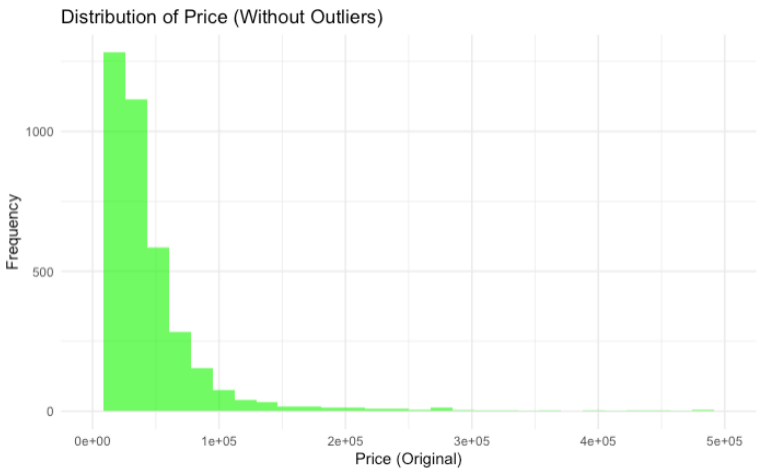
```
data_one_hot <- model.matrix(~ fuel_type + ext_color_grouped - 1, data =
data)
data <- cbind(data, data_one_hot)

# Binary encoding for clean_title and accident
data$clean_title <- ifelse(data$clean_title == "Yes", 1, 0)
data$accident <- ifelse(data$accident == "At least 1 accident or damage
reported", 1, 0)
```

Lastly, we standardized numerical variables (mileage, engine_displacement, horsepower, price) using the scale function. Columns like mileage, engine_displacement, horsepower, and price were standardized using the scale() function to have a mean of 0 and standard deviation of 1. Standardization ensures that features are on the same scale, preventing any single feature from dominating the model due to its magnitude.

Conclusion

We performed a detailed exploration of the dataset by generating summary statistics and visualizations for both numerical and categorical variables. For numerical variables such as mileage, engine displacement, horsepower, and price, we computed summary statistics to understand their distributions, including measures like minimum, maximum, median, and mean. For categorical variables like clean title and accident history, we created frequency tables to examine the prevalence of each category. Additionally, we visualized the distributions of mileage and price using histograms. To address the skewness in price data, we adjusted the x-axis to exclude outliers, focusing on the bulk of the data. These steps provide a comprehensive understanding of the dataset's structure and characteristics, preparing it for further analysis.



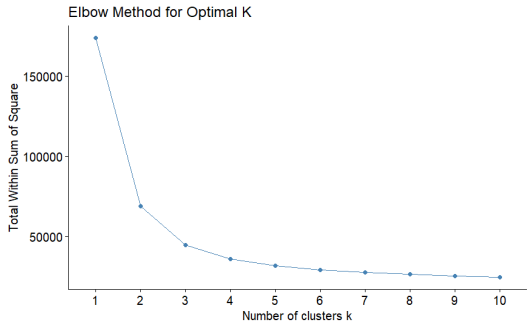


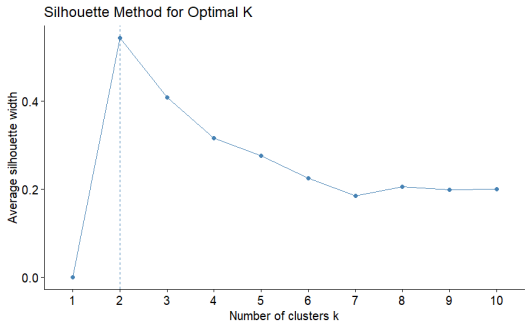
Unsupervised Learning

We used 3 clustering algorithms: K-means, hierarchical, and spectral to group vehicles based on their attributes to uncover patterns and associations that could enhance understanding of the dataset and inform predictive modeling. Each clustering method leverages different algorithms to identify groupings in the data (scaled data because of clustering sensitivity) based on attributes such as model year, mileage, horsepower, engine displacement, and price.

First we had to determine the optimal number of clusters to use. The analysis determined that the optimal number of clusters for the dataset is 2, as indicated by the Elbow and Silhouette methods. The Elbow Method revealed a significant drop in the total within-cluster sum of squares (WSS) at k=2, followed by diminishing returns, while the Silhouette Method showed the highest
average silhouette width at k=2, suggesting well-separated and cohesive clusters.

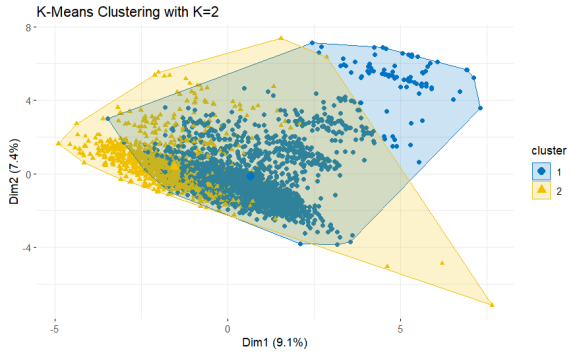
```
# select only numeric data for clustering
clustering_data <- scaled_data %>%
  select(where(is.numeric))
set.seed(123)  # For reproducibility
# Elbow Method
fviz_nbclust(clustering_data, kmeans, method = "wss") +
  labs(title = "Elbow Method for Optimal K")
# Silhouette Method
fviz_nbclust(clustering_data, kmeans, method = "silhouette") +
  labs(title = "Silhouette Method for Optimal K")
```

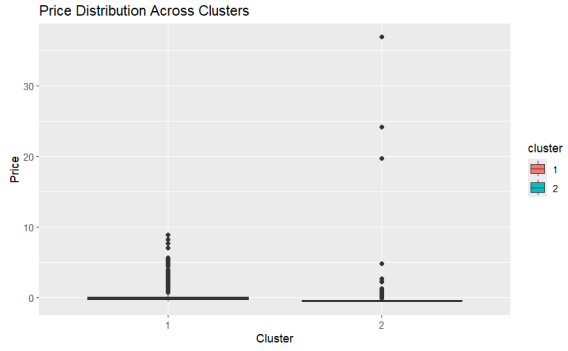




K-means clustering

```
kmeans_model <- kmeans(clustering_data, centers = 2, nstart = 25)
```



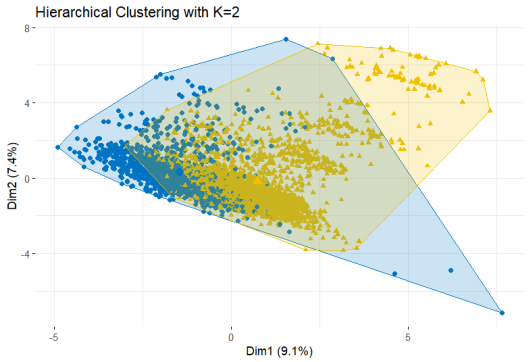


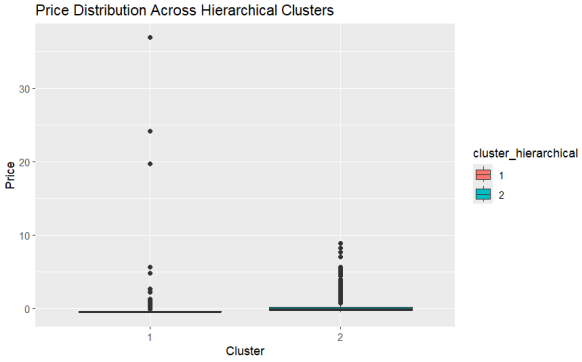
cluster <fctr>	model_year <dbl>	milage <dbl>	accident <dbl>	clean_title <dbl>	price <dbl>	horsepower <dbl>	engine_displacement <dbl>	fuel_typeDiesel <dbl>
1	2018.843	-0.3445133	0.2005700	0.8029925	0.1070501	0.1799472	-0.07588786	0.03170645
2	2007.746	0.8045330	0.3519135	0.9642263	-0.2499914	-0.4202262	0.17721898	0.02246256

We used all numeric data for clustering with 2 centers and nstart=25 for the kmeans model. According to the clustering summary, cluster 1 represents newer, low-mileage, higher-priced cars with better performance and clean titles, while Cluster 2 includes older, higher-mileage cars with lower prices and performance. The price distribution aligns with these attributes, with Cluster 1 having higher average prices and Cluster 2 showing lower ones. Outliers in Cluster 2 may include rare, antique cars with high value despite their age, while lower-priced outliers in Cluster 1 could reflect damaged or less desirable newer cars.

Hierarchical Clustering

```
distance_matrix <- dist(clustering_data, method = "euclidean")

# Perform hierarchical clustering using Ward's method
hclust_model <- hclust(distance_matrix, method = "ward.D2")
```



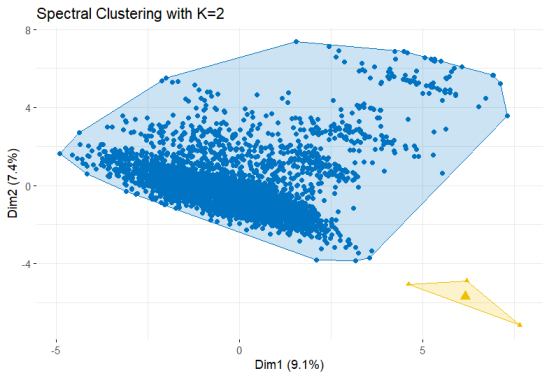


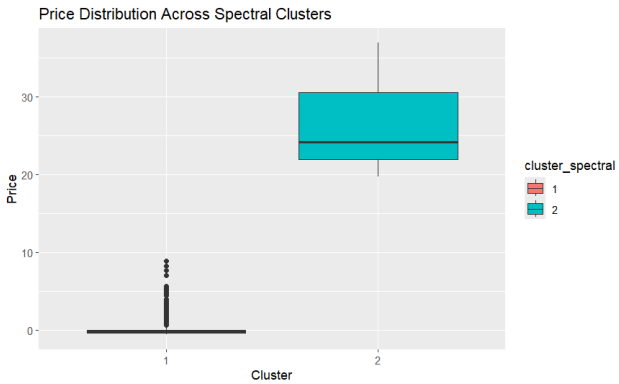
cluster_hierarchical <fctr>	model_year <dbl>	milage <dbl>	accident <dbl>	clean_title <dbl>	price <dbl>	horsepower <dbl>	engine_displacement <dbl>
1	2008.443	0.7722818	0.3508512	0.9600296	-0.2476108	-0.3941738	0.16023965
2	2019.111	-0.3925330	0.1926260	0.7960873	0.1258548	0.2003494	-0.08144611

We used euclidean distance matrix metric and Ward's method for the hierarchical clustering with 2 clusters. Based on the cluster summary, cluster 1 represents older, high-mileage cars with lower prices, and cluster 2 consists of newer, low-mileage cars with higher prices. These results are the same as the K-means clustering results. The clustering results strongly correlate with price, as evidenced by the distinct separation in price distributions between the two clusters. The lower prices in Cluster 1 and higher prices in Cluster 2 align with the attributes of each cluster, such as mileage, model year, and accident history.

Spectral Clustering

```
# Perform spectral clustering with 2 clusters (based on prior analysis)
spectral_model <- specc(as.matrix(clustering_data), centers = 2, kernel =
"rbfdot")

# Add cluster assignments to the original dataset
scaled_data$cluster_spectral <- as.factor(spectral_model@.Data)
```





cluster_spectral <fctr>	model_year <dbl>	milage <dbl>	accident <dbl>	clean_title <dbl>	price <dbl>	horsepower <dbl>	engine_displacement <dbl>
1	2015.522	0.0007227805	0.2458812	0.8512232	-0.02020336	-0.0007567177	-0.001215853
2	2007.000	-0.9651529583	0.3333333	1.0000000	26.97821959	1.0104703186	1.623569077

Cluster 1 represents newer, average-market cars, with balanced attributes across price, mileage, and performance, while cluster 2 includes older, rare, and high-performance vehicles. This clustering was significantly different than the kmeans and hierarchical regression as the vast majority of data was in cluster 1 and cluster 2 contained a few outliers, as opposed to the data being somewhat evenly distributed.

How Clustering Results Can Be Useful for Supervised Learning

- Feature Engineering:
  - ☐ Cluster membership from each method can be added as a categorical feature in a supervised learning model to improve predictions of the price.
  - ☐ These features encapsulate relationships among attributes like model year, mileage, and performance, providing a compact representation of vehicle characteristics.
- Segmented Modeling:
  - ☐ Use clusters to build separate models for each group, tailoring predictions to their unique patterns (e.g., separate pricing models for regular cars vs. antique/collector cars).
- Outlier Detection:
  - ☐ Spectral clustering, in particular, highlights rare, high-value vehicles that could skew predictions. Identifying these outliers allows for better handling in regression models.

Prediction Models

The prediction models aim to estimate car prices based on various features in the dataset. We employed five different regression models: OLS, K-Nearest Neighbor, Random Forest, Support Vector Regression (SVR), and XGBoost. This section details the model-building process, hyperparameter tuning, and evaluation metrics used to assess performance.

Data Splitting
The dataset was divided into training and testing subsets, with 70% of the data allocated for training and the remaining 30% for testing. Non-numeric columns were removed to ensure compatibility with the models.

Model 1: OLS

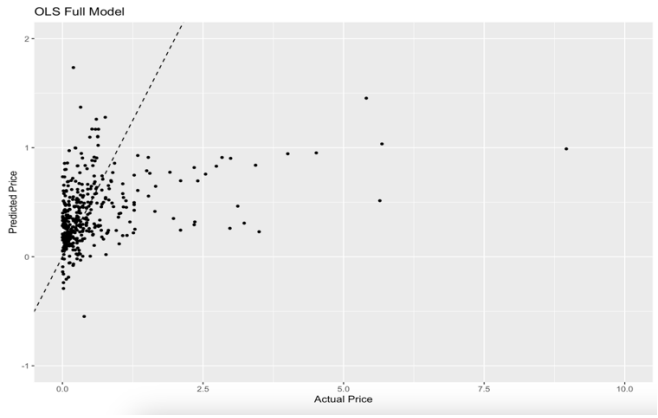
Choosing Parameters

To find the best OLS model, we used forward selection, backwards elimination, and stepwise methods. We concluded that the full model minimized MSE and maximized R-Squared for our dataset.

	MSE	**R-Squared**
Full Model	0.7474024	0.1550012
Forward Selection	0.750827	0.152239
Backward Elimination	0.750827	0.152238
Stepwise	0.8990512	0

Evaluation Metrics:

- **MSE:** 0.7474024
- **R-Squared:** 0.1550012



Model 2: KNN

Distance Metric

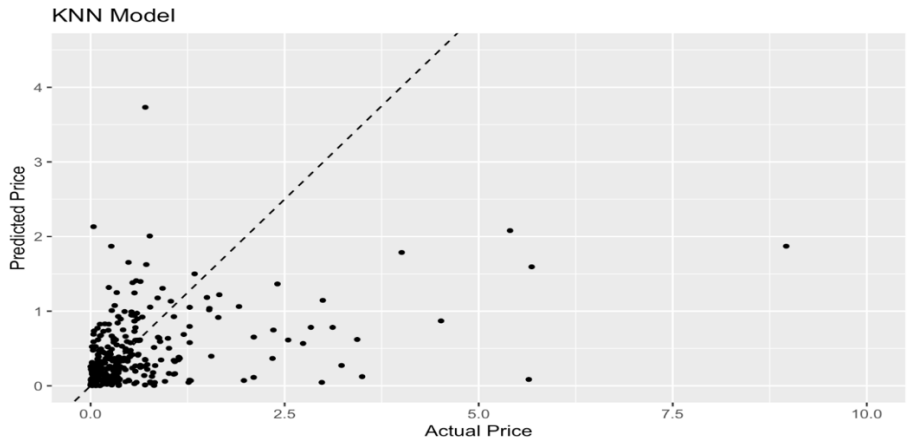
We use the default distance metric for the caret library, Euclidean distance. As for the categorical variables, we used the same approach Samruddhi, K., and R. Ashok Kumar [2] took in their attempt for a KNN model and removed predictors that could not be turned into a numeric form.

Choosing k
Initially we used the default tune length of 3 which gave us an ideal k = 9, however, expanding the tune length we concluded that k = 11 gives us the best KNN model for this dataset.

	RMSE	R-Squared
k = 5	0.7768273	0.3100024
k = 7	0.7684311	0.3222009
k = 9	0.7583563	0.3374811
k = 11	0.7488773	0.3533706
k = 13	0.7524903	0.3468039

Evaluation Metrics:

- **RMSE:** 0.7488773
- **R-Squared:** 0.3533706



Model 3: Random Forest

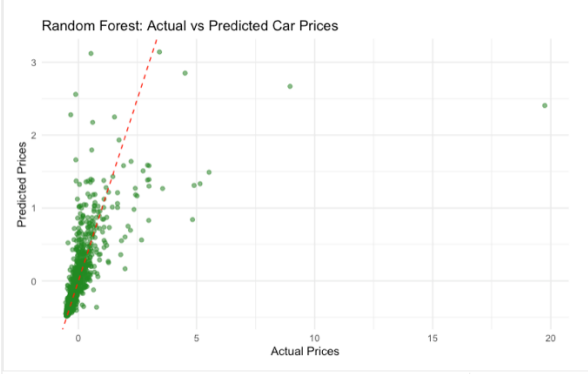
Random Forest was tuned to identify the best values for mtry (number of features considered at each split) and nodesize (minimum samples per leaf node). A grid was constructed with combinations of mtry and node size. A loop iterated over each combination in the grid. For each combination, a random forest model was trained with the specified parameters and predictions were made on the test set. Performance was evaluated with RMSE, MAE, and R_Squared. The combination with the lowest RMSE on the test set was chosen as the optimal parameters. The Random Forest model was retrained on the entire training dataset using the optimal parameters. The best parameters were identified:

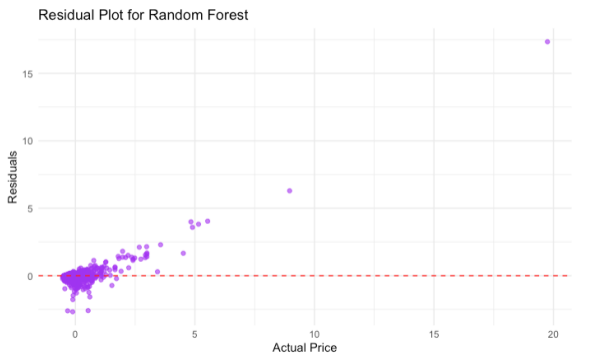
	Mtry	Nodesize
9	9	20

The model was then retrained:

Evaluation Metrics:

- RMSE: 0.6694889
- MAE: 0.2010213
- R_Squared: 0.4105742





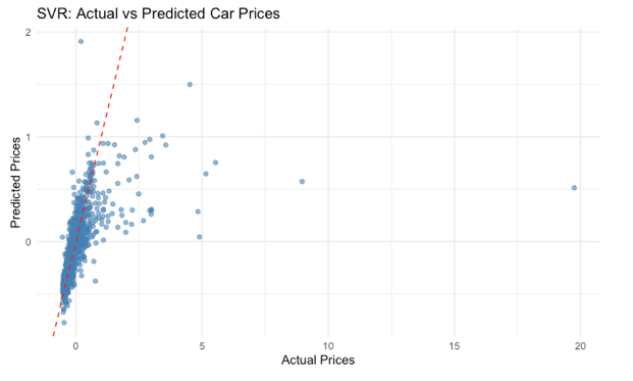
Model 4: Support Vector Regression (SVR)

SVR was tuned using a radial basis function (RBF) kernel, optimizing parameters C (regularization strength) and sigma (kernel width). A grid comprising various combinations of C and sigma values was created to explore different levels of regularization and kernel influence. A 5-fold cross-validation approach was utilized to ensure the model's robustness across different data partitions.The train function evaluated each combination within the grid, training an SVR model with the specified C and sigma values and assessing performance via cross-validated RMSE. The combination with the lowest RMSE was selected as the optimal set of hyperparameters for the SVR model. The best parameters were identified:

	Sigma	C
10	0.01	10

Evaluation Metrics:

- RMSE: 0.7548857
- MAE: 0.2056378
- R_Squared: 0.2785423



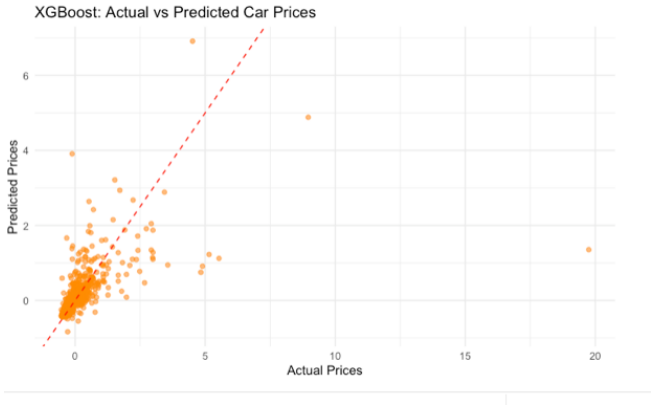
Model 5: XGBoost Regression

XGBoost was fine tuned for several hyperparameters including: nrounds, max_depth, eta, gamma, colsample_bytree, min_child_weight, subsample. A comprehensive grid encompassing a range of values for each hyperparameter was established to explore various model complexities and regularization strengths. Utilized a 5-fold cross-validation approach to ensure robust performance evaluation across different data partitions. The train function systematically evaluated each hyperparameter combination, training an XGBoost model with the specified parameters and assessing performance via cross-validated RMSE. The combination with the lowest RMSE was selected as the optimal set of hyperparameters for the XGBoost model. The best parameters were identified:

	nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight	subsample
567	300	9	0.01	0	1	5	1

Evaluation Metrics:

- RMSE: 0.6881889
- MAE: 0.2064249
- R_Squared: 0.3680489



Summary of results

	RMSE	**MAE**	**R_Squared**
OLS	0.7474024	-	0.1550012
KNN	0.7488773	-	0.3533706
Random Forest	0.6694889	0.2010213	0.4105742
SVR	0.7548857	0.2056378	0.2785423
XGBoost	0.6881889	0.2064249	0.3680489

RMSE

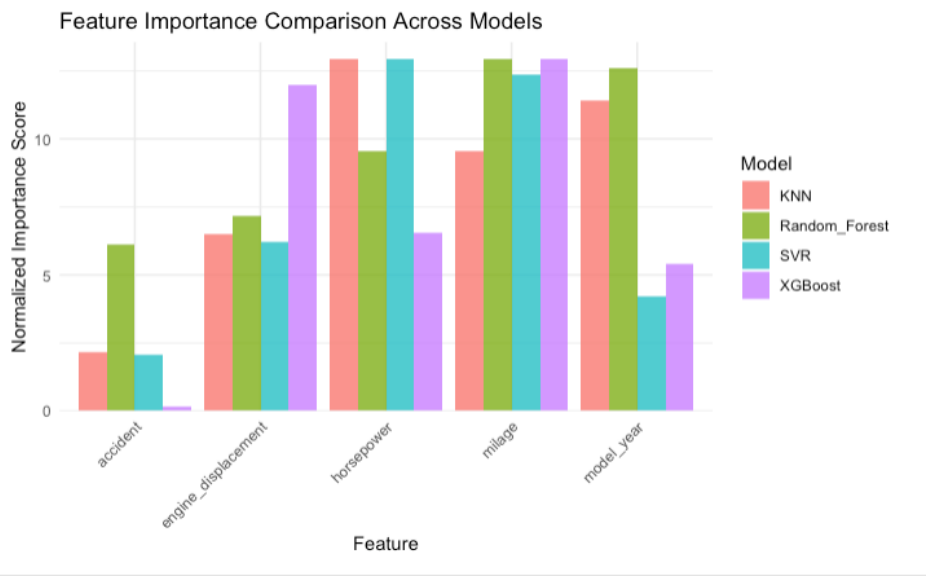
- Advantages:
  - Penalizes large errors more than small ones
  - Interpretable in the target variable's units
- Disadvantages:
  - Sensitive to outliers
  - Does not give the full picture of prediction accuracy

MAE

- Advantages
  - Robust to outliers
  - Straightforward interpretation of average prediction error
- Disadvantages
  - Treats all errors equally

R_Squared

- Advantages
  - Explain the proportion of variance captured by the model
  - Useful for comparing models with different features
- Disadvantages
  - Can give misleading values for non-linear models
  - A high R_Squared does not necessarily imply good predictive performance



This chart compares feature importance scores across four models: KNN**,** Random Forest**,** SVR**,** and XGBoost. The features shown are accident history, engine displacement, horsepower, mileage, and model year.

Observations

- All models agree that mileage is one of the most important features in predicting car prices, suggesting that mileage strongly influences the price of used cars in the dataset. This makes sense as cars with higher mileage are perceive as less reliable and likely to incur higher maintenance costs.

Model Comparison

1. KNN
    a. Focuses heavily on mileage and horsepower but assigns relative low importance to other features
    b. This aligns with KNN's reliance on similarity between data points where dominant features like mileage drive predictions.
2. Random Forest
    a. Distributes importance more evenly across features, reflecting its ability to capture complex interactions.
    b. Places higher weight on model year and accident compared to other models.
3. SVR
    a. Similar to KNN, with emphasis on mileage and horsepower but assigns slightly higher importance to engine displacement
    b. Feature importance scores ae likely influenced by data scaling
4. XGBoost
    a. Consistent with random forest in ranking mileage and horsepower as top features but assigns less importance to accident history and model year
    b. Excels in capturing nonlinear relationships which can explain its focus on mileage and horsepower.

Best Model: Random Forest

- Lower RMSE values indicating the model's predictions are closer to the actual values on average.
- Better distribution of feature importance
- Robust to Non-Linear Relationships
- The scatterplots show points clustering closer to the diagonal line than those for other models, indicating that it generalizes better across test data.

Open Ended Question

To estimate the original prices of the cars in the dataset as if they were brand new, we could use the following approach:

1. Model Selection

Use a regression model such as Random Forest, which performed well in this project. The model will predict new car prices based on extrapolated values for key features like mileage set to 0, accident history set to “None”, and mode year being adjusted to the year of release.

2. Adjust Features

Mileage: Set to zero as a new car would not have any mileage.
Accident History: Set accident history to “None reported” since a new car would not have been in any accidents or simply just exclude this column.
Model Year: Use the release year of the car.

3. Challenges

- The dataset does not include information on new car prices, making it difficult to train the model on the actual relationship between features and original prices.
- The model may not fully capture the effects of depreciation, which can vary significantly across brands and models
- Adjusting features to predict new prices involves extrapolation which increases the risk of overfitting.
- Features that influence used car prices (mileage, accidents) may have different impacts on new car prices.

4. Steps

- Select 3 cars from dataset
- Adjust features
- Predict original prices using trained random forest model

	Predicted Price	Actual Price
Ford Utility Police Interceptor Base 2023	31,724.69	47,165
Hyundai Palisade SEL 2024	51,867.77	39,550
Lexus RX 350 RX 350 2023	47,078.93	47,400

Discussion

- Differences between predicted and actual prices may result from inflation, market dynamics, or unaccounted factors like brand reputation or added features in premium versions.
- **Limitations**:
    - The model was not explicitly trained on new car prices. Instead, it infers relationships based on used car prices, where newer model years indicate less depreciation. This may not accurately capture the full premium value of a brand-new car.
    - The lack of a dataset with original car prices limits model accuracy.
    - Inflation-adjusted prices for older models may introduce additional errors.
    - Market variations are not captured by the model.