A Statistical Approach to Used Car Price Prediction

12/09/2024

Steven Qie
Statistics and Computer Science
University of Illinois Urbana-Champaign
qie2@illinois.edu

Brian Gong
Statistics and Computer Science
University of Illinois Urbana-Champaign
brianhg2@illinois.edu

William Yeh
Statistics and Computer Science
University of Illinois Urbana-Champaign
wy16@illinois.edu

Introduction

With the used car market being significantly larger than the new car market, many consumers are realizing that used cars provide a more affordable option. It plays a significant role in the growth and stability of the U.S. economy, driven by changing consumer preferences, economic factors, and the availability of certain cars. Accurately predicting the price of a used car is a challenging but essential task for buyers, sellers, and market analysts/economists alike.

This report aims to develop various predictive models for used car prices using the Used Car Price Prediction Dataset from Kaggle. This dataset comprises of 4,009 data points, representing unique vehicle listings, as well as nine distinct features that serve as key indicators influencing the value of a used car. We follow a very structured and standard approach, including data exploration, preprocessing, model training, and evaluation using relevant performance metrics. By leveraging these methods, we aim to uncover valuable insights into the world of automobiles and the various factors that are driving used car prices.

Need a section on key findings. . . .

Abstract—

For unsupervised learning, we simply utilized 3 clustering methods we learned from this course, namely kmeans clustering. hierarchical clustering, and spectral clustering, in order to find any hidden patterns in the dataset. Some parts of the methods that we used, such as the Silhouette Method was inspired by the research papers in the literature review, as we saw it being used there.

white space

white space

We utilized AI tools in this report to enhance and assist in our writing. These tools helped play a big role in ensuring clarity, conciseness, and professionalism. We also utilized AI tools to help us with syntax help when writing code in R, as well as discovering potential bugs in our code.

Literature Review

This literature review aims to summarize key findings and approaches from a few noteworthy research papers focused on used car price prediction.

“Price Prediction of Used Cars Using Machine Learning”, written by Chuyang Jin of the University of Sydney, presents a model that can predict a used vehicle’s price given their year of production, mileage, tax, miles per gallon, He hopes that his model can benefit and save time for both sellers and buyers who are looking to sell or serach for second-hand vehicles. Jin used a CSV dataset containing 100,000 records of used cars in the UK, focusing specifically on the Mercedes brand. The nine factors that he considered were the following: model, year, selling price, transmission, mileage, fuel type, tax, miles per gallon (mpg), and engine size. While doing exploratory data analysis and preprocessing, Jin noted that many many predictors had skewed distributions. For example, the overwhelming majority of prices fell in the 0-75,000 range, limiting the model's potential effectiveness for higher price ranges. Jin deemed these data points as outliers and excluded them to ensure that the model would be more accurate and usable. After testing various forms of regression, namely linear, polynomial, SVR, Decision Trees, and Random Forests, Jin found Random Forest Regression yielded the best R squared value of 0.90416.

"Used Car Price Prediction using Machine Learning: A Case Study", written by Mustapha Hankar, Marouane Birjali, and Abderrahim Beni-Hssane, applies several supervised machine learning algorithms to predict used car price prices based on features from a dataset collected from an online eCommerce website called Avito. During preprocessing, the authors of this paper performed recursive feature elimination to maintain only the most relevant features to car prices: year of manufacture, mileage, mark, fuel type, fiscal power, and model. Along with a baseline multiple linear regression model, the study also looked at K-nearest neighbors, Random Forest, Gradient Boosting, and Artificial Neural Networks. The study utilized 2 different performance metrics, Rˆ2 and RMSE, and concluded that the Gradient Boosting Regression Model achieved the best results, with a Rˆ2 of 0.8 and RMSE of 44516.20.

"Car Price Prediction using Supervised and Unsupervised Learning Models and Deep Learning" by Thomas Nsiah approached the problem of car price prediction from a supervised and unsupervised lenses. While supervised models allow a consumer to understand the key factors and predictors that influence pricing of used cars, unsupervised learning oftentimes uncovers hidden connections and patterns within the data. In his paper, Nsiah used a mock dataset of 50,000 UK second hand car sales with features similar to the previous 2 studies, such as model, engine size, fuel type, year, and mileage. Supervised learning models that Nsiah tried included simple linear regression, polynomial regression, and random forest, evaluated using mean absolute error (MAE) and R-squared metrics. He concluded that out of the supervised models, random forest performed best with an R-squared of 0.99849 and a MAE of 289.0691. For unsupervised learning techniques, Nsiah applied K-Means and DBSCAN clustering to identify price patterns, evaluated using the Davis Boudlin Index and the Silhouette Coefficient. He concluded that K-Means clustering for the year of manufacture vs price produced the best clustering results.

Overall, these three studies demonstrate the effectiveness that machinie learning can have on accurately predicting used car prices. The next section will outline our own approach and findings.

Citations:

- C. Jin, "Price Prediction of Used Cars Using Machine Learning," in 2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT), Chongqing, China, 2021, pp. 223-230, doi: 10.1109/ICESIT53460.2021.9696839.
- M. Hankar, M. Birjali, and A. Beni-Hssane, "Used Car Price Prediction using Machine Learning: A Case Study," in 2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC), El Jadida, Morocco, 2022, pp. 1-4, doi: 10.1109/ISIVC54825.2022.9800719.
- T. Nsiah, "Car Price Prediction using Supervised and Unsupervised Learning Models and Deep Learning," unpublished, 2024.

Data Processing and Summary Statistics

Preliminary Data Cleaning/Modifications

Basic Preprocessing: First we removed the dollar sign and comma in price to enable numeric operations. Then we removed mi. and , to enable numeric operation for milage as well. We also corrected the spelling of mileage from milage to mileage. The Engine column contains very useful information such as the horsepower, displacement, cylinders, engine type, and fuel type all in one column so we made each one into its own separate column.

Analyzing categorical variables

Categorical variables with various unique values include brand, model, transmission, ext_col, int_col. Let's examine all of them

First, we look at the "brand" and the "model" columns. Through analysis shown below, we have decided to omit both of these columns. Our reasoning and visualizations are shown below.

There are 57 unique brands with the frequency histogram not showing much dominance in a certain brand. To reduce the dimensionality, we will just omit this column

```
## [1] 57
```

```
## # A tibble: 57 x 4
##    brand         medianprice averageprice count
##    <chr>               <dbl>        <dbl> <int>
##  1 Ford               32378.       36241.   386
##  2 BMW                32999        41072.   375
##  3 Mercedes-Benz      38598        52076.   315
##  4 Chevrolet          31992.       36723.   292
##  5 Porsche            59900        88751.   201
##  6 Audi               34498.       39907.   200
##  7 Toyota             27999        30026    199
##  8 Lexus              30000        35669.   163
##  9 Jeep               30000        31100.   143
## 10 Land               44924        55764.   130
## # i 47 more rows
```

A similar problem is seen in the model column. We also omit this column from the dataset

Now, let's examine colors. There are both intcol and extcol variables. Having too many unique color names can introduce noise into your classification model and make it harder for the model to generalize effectively. Grouping the colors into broader, more general categories can help improve model performance by reducing the dimensionality of the feature and making patterns more apparent.

We narrowed down the colors to 6 generalized colors

```
## [1] "Black" "Other" "White" "Gray"  "Gold"  "Brown"
```

The same thing happens to int_col, but looking at the dataset we decided to have 4 categories.

```
## [1] "Black"       "Gray"        "Other"       "Beige/Ivory"
```

Analyzing Null/Empty Values

We will first look at the problem with NA and Empty values, something that this dataset has a lot of. We will first handle both NA and Empty " " values by replacing them to "NA" to make it easier to preprocess and analyze.

There are several features with empty strings/NA values. Let's examine all them to discover if we can find any patterns.

horsepower

Since there are 348 unique values in horsepower, we can consider horsepower as a continuous variable rather than categorical. However, there are 810 null values in a dataset with 4009 entries which is over 20% null values. This is too many to simply drop, so we want to perform some form of imputation. Looking at the distribution of horsepowers, we can see that the median is a good representative approximation for the distribution so we will use **median imputation**.

displacement (engine size)

There are 61 unique values in displacement (engine size). Although these appear to be discretized measurements (ex: size = 0.8 or size = 3.71 may not make sense), we can treat it as a more continuous predictor for now. There are 396 null values in displacement which is just under 10% null values, so we could consider dropping these. However since the median already exists in the dataset (median = 3.5) we can also proceed with median imputation which is what we did.

The NA values for fuel_type have a higher median price and average price than other types, and makes up a significant count of observations so we are going to treat it as a separate category.

accident

The NA/Empty values for accident exhibit very similar properties to the None reported category, with median price and average price being pretty similar, not to mention a very small percentage of data is represented by this value. Therefore, we replace and combine these observations with the None reported category. Because accident only has 2 unique values now, no accidents and 1 or more accidents, we changed it to 1,0 to be useful for models.

clean_title

The NA values for clean_title clearly have a significantly higher median price and will be treated as a separate category. We apply similar reasoning from accident to clean_title. Since there is only "Yes" and NA, we treat all the yes's to 1 and all the NA values to 0.

fuel type/engine type/cylinder

We decided to make NA it's own category for these categorical variables by factoring the features. This is because after analyzing the relationship between price and every level of each categorical variable we found that NA had its own median price that is distinct from the other levels, so we couldn't set the NA values to a default level.

Removing Outliers

We remove outliers with 1.5*IQR value.

```
## [1] "Number of outliers:  244 and average price of these cars:  214826.76"
```

one hot encoding

We will now one-hot encode the categorical variables: After looking at histograms for both Brand and Transmission, it seems Brand is more uniformly distributed while Transmission has a few salient categories. After exploring the categories of transmissions we found that the top 7 most frequent transmissions account for approximately 67-70% of the data points. Therefore we will one hot encode these 7 categories + an "Other" category for Transmission for a total of 8 transmission categories. We will also one hot encode "fuel type" and "cylinders" since those are categorical variables as well.

Final Summary Statistics

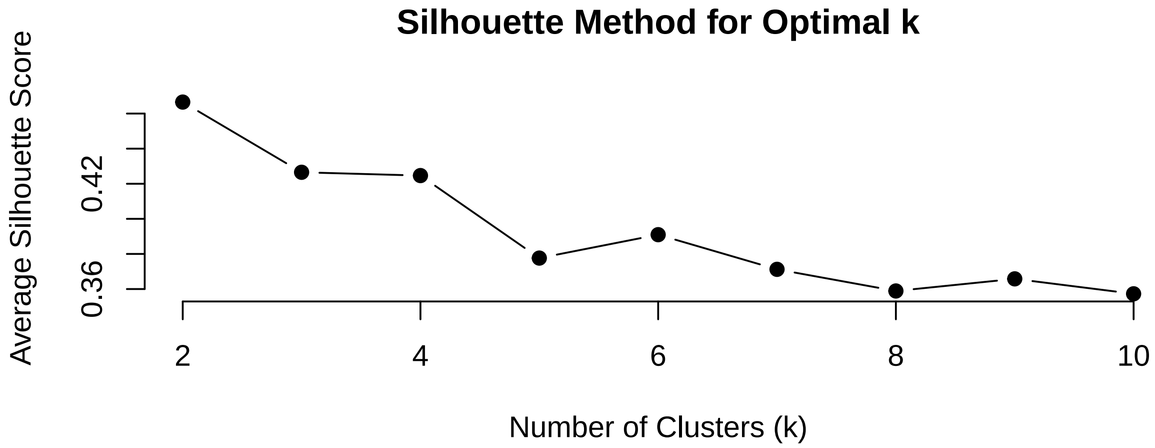
```
## [1] 3765   46
```

```
##    model_year      mileage       fuel_type.Diesel  fuel_type.Electric
##  Min.   :1992   Min.   :   100   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:2012   1st Qu.: 26600   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :2017   Median : 57237   Median :0.00000   Median :0.00000
##  Mean   :2015   Mean   : 68075   Mean   :0.02895   Mean   :0.06135
##  3rd Qu.:2020   3rd Qu.: 97000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :2024   Max.   :405000   Max.   :1.00000   Max.   :1.00000
##  fuel_type.Flex Fuel fuel_type.Gasoline fuel_type.Hybrid   fuel_type.NA
##  Min.   :0.000       Min.   :0.0000     Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.000       1st Qu.:0.0000     1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.000       Median :1.0000     Median :0.00000   Median :0.0000
##  Mean   :0.034       Mean   :0.6887     Mean   :0.00425   Mean   :0.1737
##  3rd Qu.:0.000       3rd Qu.:1.0000     3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.000       Max.   :1.0000     Max.   :1.00000   Max.   :1.0000
##  fuel_type.Plug-In Electric/Gas transmission.6-Speed A/T
##  Min.   :0.00000                Min.   :0.00000
##  1st Qu.:0.00000                1st Qu.:0.00000
##  Median :0.00000                Median :0.00000
##  Mean   :0.00903                Mean   :0.09535
##  3rd Qu.:0.00000                3rd Qu.:0.00000
##  Max.   :1.00000                Max.   :1.00000
##  transmission.6-Speed M/T transmission.7-Speed A/T transmission.8-Speed A/T
##  Min.   :0.00000          Min.   :0.00000          Min.   :0.00000
##  1st Qu.:0.00000          1st Qu.:0.00000          1st Qu.:0.00000
##  Median :0.00000          Median :0.00000          Median :0.00000
##  Mean   :0.06348          Mean   :0.05206          Mean   :0.09854
##  3rd Qu.:0.00000          3rd Qu.:0.00000          3rd Qu.:0.00000
##  Max.   :1.00000          Max.   :1.00000          Max.   :1.00000
##  transmission.A/T transmission.Automatic transmission.Other
##  Min.   :0.0000   Min.   :0.00000        Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000        1st Qu.:0.0000
##  Median :0.0000   Median :0.00000        Median :0.0000
##  Mean   :0.2667   Mean   :0.05657        Mean   :0.2709
##  3rd Qu.:1.0000   3rd Qu.:0.00000        3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000        Max.   :1.0000
##  transmission.Transmission w/Dual Shift Mode ext_col.Black   ext_col.Brown
##  Min.   :0.00000                             Min.   :0.000   Min.   :0.00000
##  1st Qu.:0.00000                             1st Qu.:0.000   1st Qu.:0.00000
##  Median :0.00000                             Median :0.000   Median :0.00000
##  Mean   :0.09641                             Mean   :0.255   Mean   :0.02125
##  3rd Qu.:0.00000                             3rd Qu.:1.000   3rd Qu.:0.00000
##  Max.   :1.00000                             Max.   :1.000   Max.   :1.00000
##   ext_col.Gold      ext_col.Gray    ext_col.Other    ext_col.White
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.00000   Median :0.0000   Median :0.0000   Median :0.000
##  Mean   :0.01116   Mean   :0.2653   Mean   :0.2133   Mean   :0.234
##  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##  int_col.Beige/Ivory int_col.Black     int_col.Gray    int_col.Other
##  Min.   :0.0000      Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000      1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000      Median :1.0000   Median :0.0000   Median :0.0000
##  Mean   :0.1392      Mean   :0.5214   Mean   :0.1246   Mean   :0.2149
##  3rd Qu.:0.0000      3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000      Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##     accident       clean_title         price         horsepower
##  Min.   :0.0000   Min.   :0.0000   Min.   : 2000   Min.   :  70.0
##  1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:16500   1st Qu.: 263.0
##  Median :0.0000   Median :1.0000   Median :29600   Median : 310.0
##  Mean   :0.2595   Mean   :0.8608   Mean   :33518   Mean   : 320.9
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:45500   3rd Qu.: 375.0
##  Max.   :1.0000   Max.   :1.0000   Max.   :99000   Max.   :1020.0
##   displacement   cylinders.10 Cylinder cylinders.12 Cylinder
##  Min.   :0.650   Min.   :0.000000      Min.   :0.000000
##  1st Qu.:2.500   1st Qu.:0.000000      1st Qu.:0.000000
##  Median :3.500   Median :0.000000      Median :0.000000
##  Mean   :3.648   Mean   :0.002922      Mean   :0.005578
##  3rd Qu.:4.400   3rd Qu.:0.000000      3rd Qu.:0.000000
##  Max.   :8.300   Max.   :1.000000      Max.   :1.000000
##  cylinders.3 Cylinder cylinders.4 Cylinder cylinders.5 Cylinder
##  Min.   :0.000000     Min.   :0.0000       Min.   :0.000000
##  1st Qu.:0.000000     1st Qu.:0.0000       1st Qu.:0.000000
##  Median :0.000000     Median :0.0000       Median :0.000000
##  Mean   :0.003453     Mean   :0.1958       Mean   :0.005312
##  3rd Qu.:0.000000     3rd Qu.:0.0000       3rd Qu.:0.000000
##  Max.   :1.000000     Max.   :1.0000       Max.   :1.000000
##  cylinders.6 Cylinder cylinders.8 Cylinder  cylinders.NA    engine_type.DOHC
##  Min.   :0.0000       Min.   :0.000        Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000       1st Qu.:0.000        1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000       Median :0.000        Median :0.0000   Median :0.0000
##  Mean   :0.3118       Mean   :0.251        Mean   :0.2242   Mean   :0.1039
##  3rd Qu.:1.0000       3rd Qu.:1.000        3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000       Max.   :1.000        Max.   :1.0000   Max.   :1.0000
##  engine_type.Electric Motor engine_type.NA   engine_type.SOHC
##  Min.   :0.00000            Min.   :0.0000   Min.   :0.000000
##  1st Qu.:0.00000            1st Qu.:1.0000   1st Qu.:0.000000
##  Median :0.00000            Median :1.0000   Median :0.000000
##  Mean   :0.03825            Mean   :0.8369   Mean   :0.005578
##  3rd Qu.:0.00000            3rd Qu.:1.0000   3rd Qu.:0.000000
##  Max.   :1.00000            Max.   :1.0000   Max.   :1.000000
##  engine_type.Turbo engine_type.Twin Turbo
##  Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.000000
##  Median :0.00000   Median :0.000000
##  Mean   :0.01275   Mean   :0.002656
##  3rd Qu.:0.00000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.000000
```

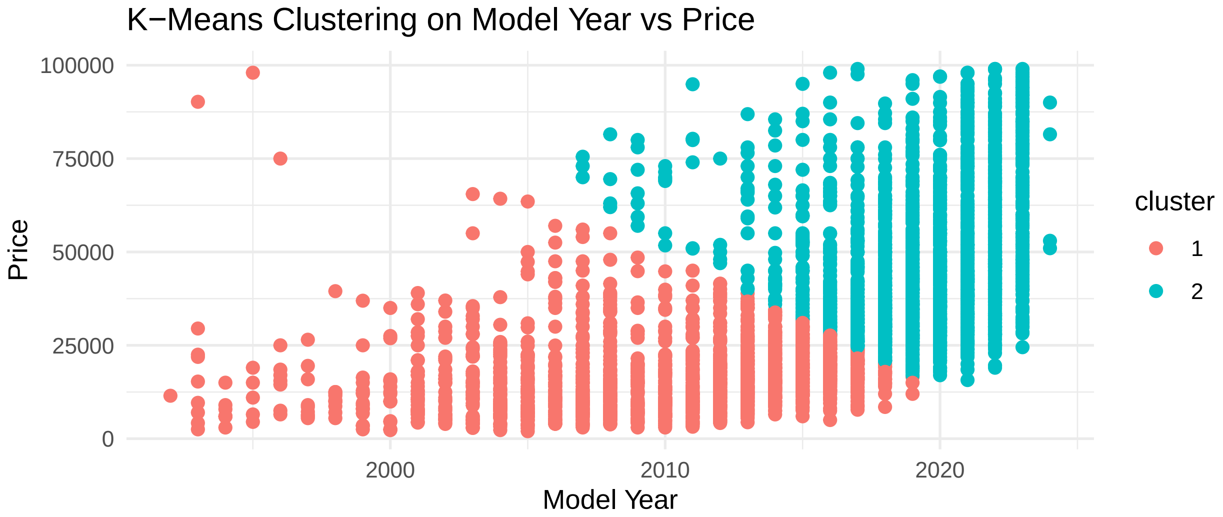
Unsupervised Learning

1. KMeans Clustering

We decided to use kmeans to examine the relation between model_year and price, as we noticed a similar examination in one of the papers while doing the literature review. Because K-means utilizes distance metrics, we scale the data before clustering.



We decided to use the Silhouette Method to determine the optimal number of clusters. This method essentially uses distance measures calculating how close clusters are to themselves and how far away they are to other clusters to judge the optimal number of clusters. In this case, 2 has the highest average silhouette score so we will use k=2.



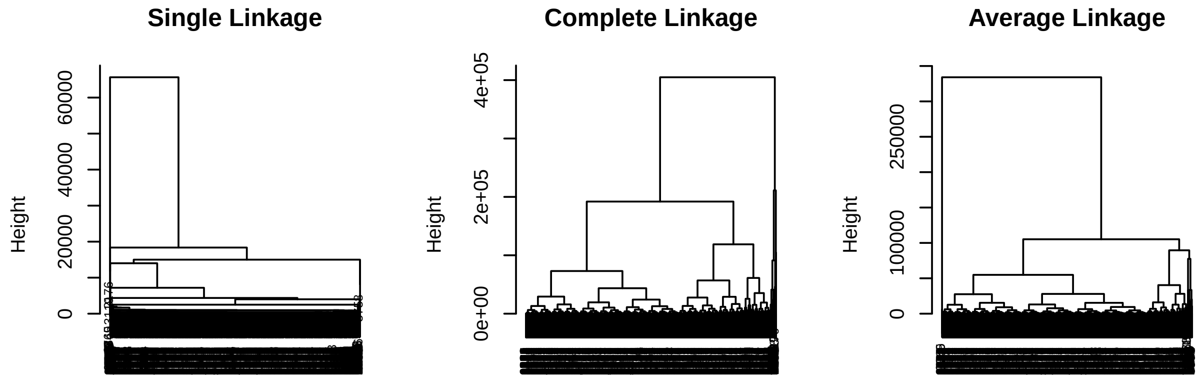
There seems to be a pretty solid relationship between a more recent model_year and higher price. Although the 2 clusters seem to be mostly dominated by model year, it's clear that the average price of cluster 2 is higher than cluster 1.

2. Hierarchical Clustering

Next, we will try hierarchical clustering with three different linkage methods(single, complete, and average) using euclidean distance. Hierarchical Clustering begins with each data point starting as its own cluster. The goal is to progressively group them together until there is only one group. The process involves choosing the closest two groups, calculated through a specific distance metric. The features we will cluster is

```
## [1] "model_year"   "mileage"      "accident"     "clean_title"  "price"
## [6] "horsepower"   "displacement"
```

These are all numeric features as clustering requires numeric features


Single Linkage

Complete Linkage

Average Linkage

We will use complete linkage as it looks like it generates pretty distinct clusters

```
##
##    1    2
## 3674   91
```

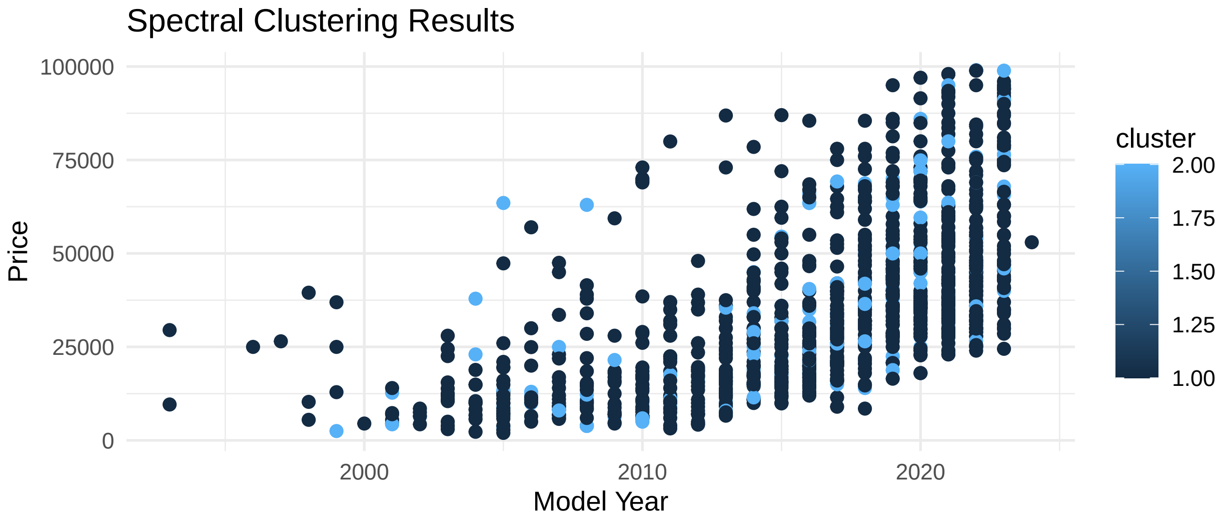
```
## # A tibble: 2 x 7
##   cluster avg_price avg_model_year avg_accident avg_mileage avg_horsepower count
##   <fct>       <dbl>          <dbl>        <dbl>       <dbl>          <dbl> <int>
## 1 1          34086.          2015.        0.254      64111.           322.  3674
## 2 2          10588.          2007.        0.484     228100.           267.    91
```

There are a lot of correlations here that make sense between the 2 clusters. Cluster 1, with a more recent avg_model_year, also has a lower avg_mileage and a lower avg_accident rate, probably because the car has been driven for less time, this cluster also has a much higher avg_price in comparison to cluster 2. The data isn't distributed very well however as a vast majority of the points sit in cluster 1, perhaps suggesting that hierarchical clustering isn't suitable for this dataset.

3. Spectral Clustering

Finally, we will try spectral clustering, which aims to group observations based on their proximity information. This method involves 2 main steps, the first being using the eigenvalues of a similarity matrix to perform dimension reduction, followed by applying a clustering algorithm like K-means. The same features as hierarchical clustering will be used

```
## # A tibble: 2 x 6
##   cluster avg_model_year avg_mileage avg_accident avg_horsepower count
##     <int>          <dbl>       <dbl>        <dbl>          <dbl> <int>
## 1       1          2016.      60899.        0.246           325.   859
## 2       2          2016.      60429.        0.220           329.   141
```



We ran Spectral Clustering on a randomly selected 1000 row subset for the data as the computation time was taking too long for the full dataset.

Similar to hierarchical clustering, cluster 1, with a more recent avg_model_year, also has a lower avg_mileage and a lower avg_accident rate, this cluster also has a much higher avg_price in comparison to cluster 2. The distribution of data points between the 2 clusters seem to be more even in comparison to hierarchically clustering, meaning that perhaps spectral clustering is more suitable for this dataset. Looking at the graph. as we clustered using multiple different feature, there is no clear relationship between model year and price anymore like there was in the kmeans clustering.

Prediction Models

For our supervised models, we will divide the data into training and testing sets using an 80/20 split. The training set (80% of the data) will be used to train the models, while the testing set (20% of the data) will be reserved to evaluate their performance and accuracy.

1. Linear Model(Lasso, Ridge, Elastic Net)

Lasso, Ridge, and Elastic net are techniques used in linear regression to improve generalization and prevent overfitting. These regularization techniques work by penalizing the size of coefficients of the model and work well when dealing with high-dimensional data. All 3 models also involve tuning parameters, which control the strength of the penalty, so k-fold cross validation will be used to find the optimal parameters. The cv.glmnet function trains these models and also automatically scales and centers the data.

```
#Training the models
ridgemodel = cv.glmnet(x = as.matrix(xtrain), y = ytrain, nfolds = 10, alpha = 0)
lassomodel = cv.glmnet(x = as.matrix(xtrain), y = ytrain, nfolds = 10, alpha = 1)
elastic_net_model <- cv.glmnet(x = as.matrix(xtrain), y = ytrain, nfolds = 10, alpha = 0.5)
```

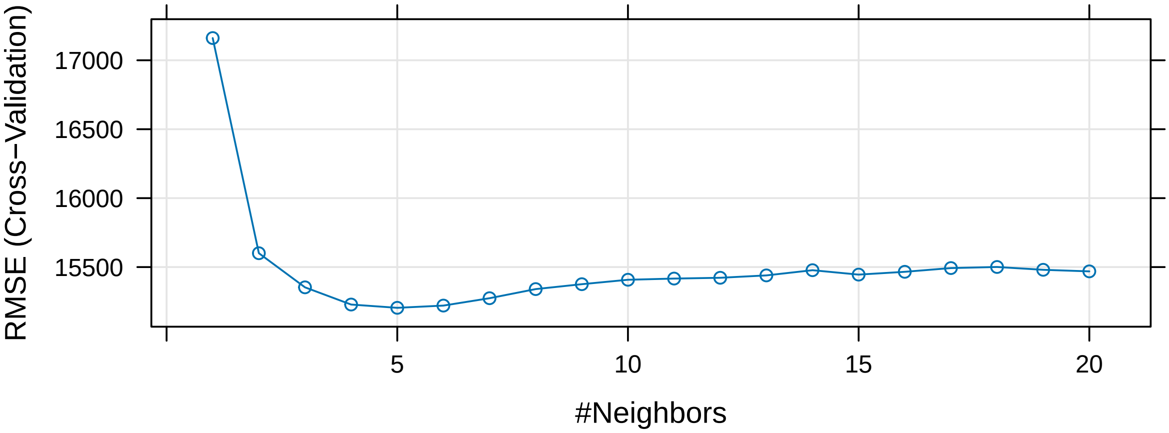
To assess and compare performance of these 3 models, we utilize RMSE, or Root Mean Squared Error. RMSE measures the average difference between the values predicted by a model and the actual values. A lower RMSE indicates better model performance, as it means there are smaller differences between the predicted and actual values.

```r
ridgepred = predict(ridgemodel, newx = as.matrix(xtest), s = "lambda.min")
lassopred = predict(lassomodel, newx = as.matrix(xtest), s = "lambda.min")
elasticpred = predict(elastic_net_model, newx = as.matrix(xtest), s = "lambda.min")

#RMSE Calculations
ridge_rmse = sqrt(mean((ridgepred - ytest)^2)) #outputs 12017.06

lasso_rmse = sqrt(mean((lassopred - ytest)^2)) #outputs 11992.87

elasticnet_rmse = sqrt(mean((elasticpred - ytest)^2)) #outpus 11993.98
```

Out of the three models, the Lasso model performs the best, with a RMSE of 11992.87. Lasso is different from Ridge in that it can shrink coefficients to exactly 0, making it useful for cases when the data has useless features. Examining the coefficients of Lasso, it seems that the most predictive variables include model_year, fuel_type.Diesel, displacement, and engine_type.TwinTurbo. This seems to suggest that variables related to a car's power and performance are key in determining a price of a used car.

2. K Nearest Neighbors(KNN)

KNN regression works by calculating the k nearest training set data points to the test point and predicting the target value by taking the average of their target values. Because KNN is a distance based algorithm, it is sensitive to feature scaling. For example, if one feature has ranges from 1-10 and another one has 1-10000, distance calculations will be biased and results will suffer as a result. In addition to standardizing and normalizing our data, we will also perform Principal Component Analysis(PCA) to further reduce the dimensionality of our data and reduce noise, while retaining most of the overall variance. We will keep components that make up 90% of the explained variance. Finally, KNN is also sensitive to the choice of k. To find the optimal value of k, we will again perform k-fold cross validation. We will again, use RMSE to evaluate the performance of our model.



This graph not only illustrates the cross-validation results for KNN, but also highlights a fundamental concept in machine learning: The Bias-Varaince Trade Off. At low values of K, the RMSE is high due to overfitting, which corresponds to high variance and low bias. The model performs well on the training data, as it relies on very few neighbors, but becomes sensitive when exposed to new testing data. In contrast, high k values correspond to low variance and high bias. The model is more generalized, but it may start to underfit, reflected in the slow increase of RMSE as the k becomes larger and larger. The optimal K achieved through cross validation, K = 5, balances bias and variance to achieve the lowest RMSE.

Finally, we train a knn model using this optimal k and calculate the RMSE, which is found to be 39728.95

```
best_k <- knn.cvfit$bestTune$k #outputs 5
knn_predictions <- knn(xtrain_processed, xtest_processed, ytrain, best_k)

#RMSE Calculation
knn_rmse = sqrt(mean((as.numeric(knn_predictions) - ytest)^2)) #outputs 39728.95
```

3. SVM

Support Vector Machines(SVM) are another supervised machine learning model that excels in high-dimensional spaces, as well as in memory due to the use of support vectors. Their ability to utilize various different kernel functions, both linear and non-linear, allow them to handle complex decision boundaries and capture nonlinear relationships in the data effectively. The radial kernel is a popular kernel choice, and involves 2 key tuning parameters: C and Sigma. Multiple SVM models were fit using different values of C and Sigma, and K-Fold Cross Validation was used to find the combination with the lowest RMSE.

```
train_control <- trainControl(method = "cv", number = 5)
tune_grid <- expand.grid(
  C = c(0.1, 1, 5, 7, 10),         #different values of C
  sigma = c(0.01, 0.1, 0.5, 0.7, 1)    # different values of sigma
)
```

The Best Parameters for C and Sigma were found to be C=7 and Sigma=0.01. The RMSE for this combination was found to be 11054.76

```
svm_model <- train(
  x = as.matrix(xtrain),
  y = as.vector(ytrain),
  method = "svmRadial",       # using the radial kernel
  trControl = train_control,
  tuneGrid = tune_grid
)
optimal_parameters <- svm_model$bestTune #outputs C = 7 and sigma = 0.01
svm_predictions <- predict(svm_model, newdata = as.matrix(xtest))

#RMSE Calculation
svm_rmse <- sqrt(mean((svm_predictions - ytest)^2)) #outputs 11054.76
```

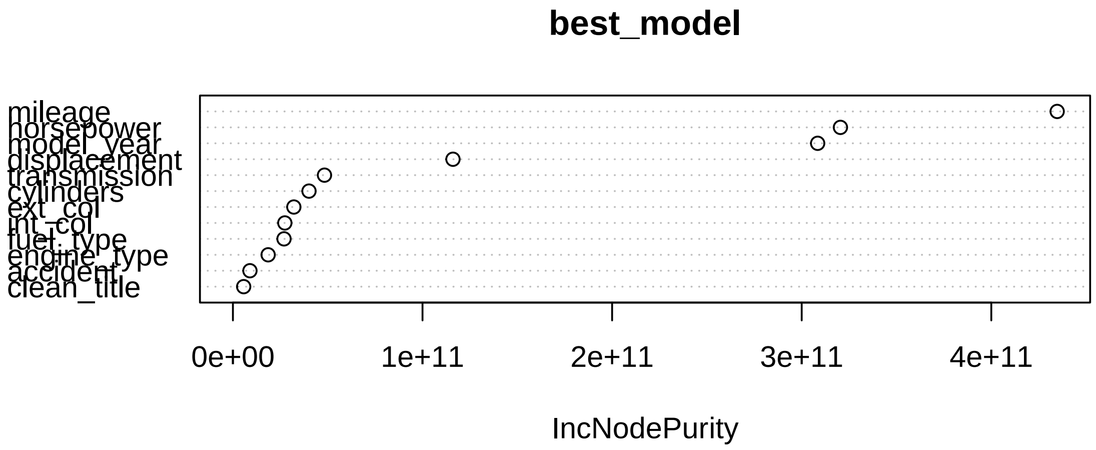
4. Random Forest

Random Forest is another powerful supervised machine learning algorithm that involves building numerous decision trees using different subsets of the data, acquired through a technique called bootstrapping.

```
tune_results <- expand.grid(
  ntree = c(500),
  mtry = c(5, ncol(X_train)),
  nodesize = c(1, 5, 30),
  sampsize = nrow(X_train) # maximum
)
```

```
# tune_results
best_model <- NULL
best_params <- NULL #ends up evaluating to 500, 5, 1
lowest_mse <- Inf
for (i in 1:nrow(tune_results)) {
  params <- tune_results[i, ]
  model <- randomForest(X_train,
                        y_train,
                        ntree = params$ntree,
                        mtry = params$mtry,
                        nodesize = params$nodesize,
                        sampsize = params$sampsize)
  y_pred <- predict(model, X_test)
  mse <- mean((y_pred - y_test)^2)
  if (mse < lowest_mse) {
    lowest_mse <- mse
    best_model <- model
    best_params <- c(params$ntree, params$mtry, params$nodesize)
  }
}
randomforest_rmse = sqrt(lowest_mse) #evalulates to 9497.301
```

```
varImpPlot(best_model)
```



We use MSE to evaluate random forest since that metric is directly comparable with other models. Base random forest gets MSE (not OOB error) is 9855.106 with ntree = 500 and all other parameters being default. Tuned random forest gets MSE of 9497.301 with ntree = 500, mtry = 5, nodesize = 1, and sampsize = nrow(X_train). Tuning was done with standard grid search over the 3 parameters of ntree, mtry, and nodesize. Descending order of variable importance is mileage, horsepower, model year, and displacement.

5. Gradient Boosting Regressor

Open-Ended Question/Conclusion

A researcher is interested in estimating the original price of the cars in our dataset as if they were brand new. To solve this problem, we built a machine learning model using features that we believed are most relevant to understanding depreciation. We utilized the following key features to model depreciation:

1. Brand: Different brands might exhibit varying depreciation patterns. For example, economy brands like Toyota may have a more linear depreciation curve, while luxury brands like BMW may see a steeper initial depreciation. Whether a brand is considered luxury or economy is useful as well, because luxury brands typically have higher starting prices than economy brands. Because of these reasons, we decided to include this feature even though it was omitted in our earlier prediction models.
2. Age: Cars will generally lose value over time as newer models with newer designs and updated features are released. Since we are only given the model_year in the dataset, which represents when the car was made, we created a new column called "Age" that is simply current_year - model_year + 1, which represents the number of years since the car was new.
3. Mileage: Generally, the higher the mileage, the lower the car's value. Increased miles often indicates more wear and tear on car components. High mileage on a car can also signal that more maintenance will be required to keep the car running smoothly, making it less attractive to potential buyers.
4. Accident History: This feature helps capture the impact of damage on a car's value because accidents are usually correlated with the integrity, safety, and reliability of the car. Many buyers will perceive cars with accident histories as less favorable compared to cars without accident histories.
5. Clean Title: Cars without a clean title may depreciate faster due to the higher perceived risks and uncertaintites assocaited with its condition.

For practicality reasons, and to manage potential model complexity issues that arises from having to one-hot encoding every single unique brand, we will train our model on a subset of the data, specifically cars belonging to the seven most common brands in our dataset: Ford, BMW, Mercedes-Benz, Chevrolet, Porsche, Audi, and Toyota. This subset also provides a good balance between luxary and economy brands.

Estimating brand-new car prices using only a dataset of used cars presents a couple of challenges. Because the model has never actually seen cars with zero age and zero mileage, it must infer/extrapoalte what the original prices might have been, which can lead to potential inaccuracies. Additionally, external factors that the data set doesn't capture, such as inflation, shortages, and market competition, also influence the car's original price. For example, the first six weeks of 2022 saw a car shortage caused by factors such as the pandemic and a global semiconductor shortage, none of which are accounted for in the model.

Nevertheless, we aim to leverage these features to develop a machine learning model that can reasonably estimate the original prices of cars.

```
##               price          brand.Audi           brand.BMW     brand.Chevrolet
##           "numeric"           "numeric"           "numeric"           "numeric"
##          brand.Ford brand.Mercedes-Benz       brand.Porsche        brand.Toyota
##           "numeric"           "numeric"           "numeric"           "numeric"
##             mileage            accident         clean_title                 Age
##           "numeric"           "numeric"           "numeric"           "numeric"
```

lasso model

Original Price Estimation and Comparison

We selected one car each from Toyota, Ford, and Porsche and using the trained model, predicted the "brand-new" price for each car. Then, we searched online for the actual original release prices of these cars and compared them with the model's predictions. R Code is hidden to make the report more easy to read.

```
## Randomly picked Toyota Car: 4Runner TRD Off Road. Year of Model: 2019
## Predicted Price: 66952.068
```

The original MSRP was approximately 40,395. Our model predicted the car to be around 26,000 higher than the actual price.

```
## Randomly picked Ford Car: Ford Maverick XLT. Year of Model: 2022
## Predicted Price: 67774.82
```

The original MSRP was approximately 29,541. Our model predicted the car to be around 37,000 higher than the actual price.

```
## Randomly picked Porsche Car: 718 Spyder Base. Year of Model: 2022
## Predicted Price: 111831.508
```

The original MSRP was approximately 98300. Our model predicted the car to be around 13500 higher than the actual price.