Driving the Future: Forecasting Pre-Owned Car Prices in India

ORIE 4741: Learning with Big Messy Data

Sunny Sun (xs275) and Sanjana Vakacherla (sav65)

Spring 2024

Introduction

A large majority of individuals use cars as a form of transportation to jobs, schools, and shopping, making driving one of the most accessible means of transportation. The pre-owned car market is a significant subset of the automotive industry, offering more accessible vehicle options to individuals who are looking for cheaper alternatives. Accurate price prediction for used cars can benefit sellers and buyers by providing fair market value assessments. This project aims to develop a predictive model that determines a used car's market value. We use the relevant features of the dataset such as make, model, year, mileage, condition, and location sold. The main contributions of this project are 1) highlighting the importance of precise valuation in the car industry, 2) using a methodological approach including linear regression, support vector machines (SVMs), and gradient descent to produce actionable results, and 3) determining the most important features that affect the prices of used cars on the market. This analysis will not only help in understanding the depreciating value of cars but also assist in making informed decisions in the used car market.

Problem Importance

The importance of developing a model that can predict the resale price of a car lies both within being able to help people who are looking to buy/sell a used car and also the larger implications of this model. Creating a model that can take in information such as age, location, engine power, brand, and miles would allow people trying to sell to know what range they should list their car for. This could help them get a higher estimate for their car and can help people who maybe aren't able to get to a dealership or can't afford to partner with a dealership to sell their car for a good price. This also works the other way around. For example, if there is someone who can't afford to go to a dealership and pay an extra premium or someone who can't afford to buy a car at full price, this model can help them identify the range they should be willing to pay for the car that they want. This can also prevent them from getting scammed or offered a much higher price, ensuring that they can get a fair deal on their vehicle. In addition to helping people, a successful model that is able to predict these metrics could also extend its applications. We could use this model for motorcycles, scooters, and other vehicles. India is a country where cars are not necessarily the majority automobile in the country - according to a recent survey, 55% of families own two-wheelers whereas only 1 in 12 families own cars. Being able to make an accurate model to predict car prices could also be extended to predicting motorcycle scooter or even auto rickshaw prices. These could help families looking for vehicles other than cars, or could help someone who is looking to buy an auto rickshaw (a popular taxi alternative in India) to start a business or make some money on the side.

The Approach

We adopt a comprehensive machine-learning approach, integrating data analysis and modeling techniques to address the project's objectives. The methods used in the project are two-fold. First, for predicting the price of used cars, we started off by using a simple linear regression model. More complex models such as lasso regression, random forest regression, and ridge regression were then used to obtain higher training and test set accuracy. SVMs are then used to create models that are more robust to outliers, allowing us to create more accurate predictions while minimizing the impact of outliers on our predictions. Next, in order to determine the most important features used for predicting the price of used cars, we employ a blend of data visualization and exploratory analysis techniques. By constructing informative graphs and visualizations, we aim to discern correlations between various features and the resale prices of cars. To this end, we performed feature engineering on the dataset through data cleaning and transformations such as frequency encoding. Additionally, we used k-means clustering to group similar used cars based on their features, such as mileage, engine size, power, and other relevant characteristics. The goal is to identify patterns or segments within the market. Lastly, we decided that gradient descent combined with mean squared error, L1 loss, hinge loss, and regularization would be good models to use for finding the biggest factors in the resale price of a car.

The Dataset

The dataset used for our project is the Used Car Prices Prediction Dataset from Kaggle. The dataset compiles key metrics about second-hand car sales in various parts of India. Before cleaning and preprocessing, this dataset contains around 6019 rows, with 14 columns (features). We summarize the important columns (features and labels) below, along with information such as the initial type of data the information was formatted in.

1. **Name:** Name, brand, and model of the car. For example: Maruti Wagon R LXI CNG
2. **Location:** Where the vehicle is sold or can be purchased. For example: Bangalore, Mumbai, Hyderabad.
3. **Year:** Release year of the model.
4. **Kilometers Driven:** Total mileage on the vehicle by the previous owner(s) given in kilometers.
5. **Fuel Type:** The type of fuel the car uses. For example: Diesel.
6. **Transmission:** The type of gear the car uses, either automatic or manual.
7. **Owner Type:** Indicates how many previous owners the vehicle has had.
8. **Mileage:** Standard mileage offered by the car company in kmpl or km/kg.
9. **Engine:** Engine displacement in cubic centimeters.
10. **Power:** Maximum power of the engine in bhp.
11. **Seats:** Number of seats in the vehicle.
12. **New Price:** The price of a new car of the same model in Indian Rupees (INR).
13. **Price:** The price that the used car eventually sold for in Indian Rupees (INR).

Data Exploration & Feature Engineering

Data Preparation

In order for our data to become a suitable input for our various models, we first had to do some data cleaning and preprocessing. We first began with dropping the column with the name 'Unnamed: 0' and 'New Price'. Since we didn't need the row indices since the dataframe itself already keeps track of that, we removed 'Unnamed: 0'. In addition, since we were predicting the price of used cars, the original selling price of that car when bought for the first time was irrelevant to our investigation, leading us to remove the 'New Price' column as well. Next, we searched for missing values in these columns. We chose the method of replacing missing values with the mean of that column, as we thought that completely removing entries or rows with missing values could cause us to lose a significant portion of our data. In addition to this, we then examined our 'Seats' column and found that there were certain entries where the number of seats was zero. Since logically, a car cannot have 0 seats, we then removed these entries from our dataset. In addition to this, we filter our dataset by only including cars where the 'Kilometers Driven' column contained a value less than 300,000 in order to eliminate egregious outliers.

Feature Engineering

The first piece of feature engineering that we used was transforming our string data into integers or floating values that could be used as inputs for our various models. To accomplish this task, we looked at the 'Mileage', 'Engine', and 'Power' columns. Example values for these columns are 26.6 km/kg, 998 CC, and 58.16 bhp. We only want the numerical values without the units of measurement when we pass these numbers as input into our models. Since all of these are originally stored as strings, we then split these strings using a space as our delimiter. By doing this, we are able to change "26.6 km/kg" into an array containing ["26.6", "km/kg"], and from here we can just choose the first element and convert it into a floating point number. Using this method of feature engineering for these three columns allows us to extract just the numerical values for all entries. We also transformed our 'Name' column into a format that could be used in our models. At first, our 'Name' column contained strings representing the make, model, and version of the car. We needed separate columns containing our brand and model, so that we could make graphs plotting certain features and seeing if they varied between brands. In order to achieve this, we split these strings once again using the space character as our delimiter, so that we can get the company name and the car's model separately. We know that the company is always the first word of an entry in the 'Name' column and that the model is the next word. Using the same string splitting method, we then extract the first and second words in the 'Name' field and create two additional columns; 'Company' and 'Model', that contain only the company and model of the car respectively. From these steps, we were able to clean and transform our data into a more readable format.

Exploratory Data Analysis

After performing all data cleaning and transformation steps, we were able to create visualizations of our data and determine preliminary relationships between variables. We also used these initial graphs to gain a better understanding of what our data was looking like.
The initial scatterplot in Figure 1 depicts the relationship between the size of a engine and its resale price point. Looking at this graph, we can visually estimate that there seems to be a linear relationship between the engine size and the price point, indicating that Engine Size is a strong factor in determining the resale price of a car. This was useful when thinking about our models since we are able to determine one feature that could be useful to look more into. This helps us investigate if there is a strictly linear relationship between the two categories. We also plotted an Engine Size vs Power graph in Figure 2, which showed a strong positive relationship between engine size and the power of a car. Since an increase in engine size seems to reflect an increase in price, we can conclude that an increase in power would increase the price as well, which intuitively seems logical. In addition to this graph, we also create graphs depicting the number of cars per company in Figure 3. This plot showed us that Maruti, Hyundai, and Honda were by far the most popular car brands being resold. This could suggest that these three are the most popular brands amongst people that are looking to resell their cars, giving us an idea at what kind of customer base we are looking at while conducting our research. Creating a plot as depicted in Figure 4 that shows the distribution of car prices (in INR Lakhs) reinforced the previous idea, as the majority of prices were contained within the 0 to 10 Lakh mark, which also happens to be the price range for a lot of Maruti brand cars (shown in the plot below), showing that a lot of the cars in our dataset would share similar characteristics, potentially making it easier to find relationships between variables. We also created a bar plot that showed the distribution of prices for each brand of car in Figure 5, and from the chart below we can see that Maruti's tend to sell at around the 5 to 6 Lakh range. We can compare this to the Porsche, which we can see has a much, much higher resale price (with an average at around 45 to 50 Lakhs). In addition, we also plotted the owner type against price in Figure 6, to see how much the price of the car was affected if it was a first-time, second-time, or third-time owner. After plotting these boxplots, we examined the medians and outliers for each of the first, second, and third-time owners, and we found that the median resale price tended to be around the same for each group. However, first-time owners had a lot more outliers, and there were quite a few cars that ended up selling for much higher than the median as compared to the fourth and above category, which didn't have any outliers or values greater than 10 lakhs. The first-time owner category had outliers up to 160 Lakhs, which was remarkably higher than many other points on the graph. While analyzing these findings and looking at them from a logical perspective, this does seem to make sense. If a car is being sold for the fourth time, it stands to reason that its price is established and cannot be sold for above the price that it sold for the first three times, especially considering that it probably has even more miles on it at this point. For a car that is being sold for the first time, we can see from this graph that there is a lot more variation in the price range it can sell at. There are many more outliers, and depending on the amount this car has been used, it can be sold at a potentially higher price. As the number of previous owners increases, the number of outliers decreases and the data becomes more consistent.

Price Prediction

Regression Methods

Before performing any regression, categorical values such as Company, Model, Location, Owner Type, Fuel Type, and Transmission were encoded and transformed into a numerical format. This is done through a many-hot encoding method which transforms the categorical data into a usable numerical feature for the columns. This method was implemented by using the built-in pandas function *get_dummies* where values are converted into dummy/indicator variables (many 0/1 variables).
Then we defined the train test split to be 80% and 20% respectively. Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), and R-squared score (R^2) were used as performance measures to evaluate the model's predictive ability. Mean Absolute Error (MAE) is used to express the average of the absolute values of the differences between predicted and actual values. A lower MAE value indicates that the model performs better and its predictions are closer to the true values which can be used to evaluate the accuracy of models in regression problems. Mean Square Error (MSE) is the average of the squared differences between predicted values and actual values. It generally emphasizes large error values because the impact of large errors is increased by squaring the differences. Root Mean Square Error (RMSE) is calculated by taking the square root of MSE. RMSE allows errors to be interpreted by returning them to the original unit of measurement and is a derivative of MSE. R-squared score (R^2) is used to measure how well a regression model fits the data and indicates how much of the variance on the dependent variable is explained by the independent variables.
Generally, lower MAE and RMSE values indicate that a better model makes fewer errors. The R-squared score indicates

how well the fit to the data is. The closer it is to 1, the better the model fits the data. As it approaches 0, the model becomes less able to fit the data. This means lower error values (MAE, MSE, RMSE) and higher R-squared score indicate the presence of a better model.

Linear Regression

The linear regression model was chosen as the most basic regression model to predict the revenue of future used car prices. The LinearRegression function was imported from sklearn linear model to perform the computations. The model outputted an MAE score of 2.344, MSE score of 16.764, RMSE score of 4.094, and R^2 score of 0.829. The R^2 score of 0.829 suggests that the model explains approximately 83% of the variance in the used car prices, indicating a strong fit of the model to the data. However, the relatively high MSE and RMSE values hint at significant errors in some predictions, which could be due to outliers or anomalies in the data that the model fails to capture accurately.
Additionally, the histogram of residuals in Figure 7 shows a distribution centered close to zero, which is ideal for linear regression. Most residuals cluster around the mean, but there is noticeable skewness and a few outliers, which likely contribute to the higher MSE. The scatter plot in Figure 8 comparing actual to predicted prices shows a close alignment along the diagonal, which indicates accurate predictions across a range of values. However, there are deviations, particularly for higher values, where the model appears to underestimate prices. The learning curves in Figure 9 reveal that both training and cross-validation scores stabilize with an increase in data points. The training score decreases slightly, and the cross-validation score increases, converging to a point that suggests adding more data might not significantly improve the model's performance. These results indicated that handling the outliers could help with preventing the skewing of the model's performance. More robust methods might help capture additional nuances in the data.

Ridge Regression

The ridge regression model was chosen next to predict the price of used cars due to its effectiveness in handling multi-collinearity among predictors and its ability to minimize overfitting, which is especially beneficial in scenarios where independent variables are highly correlated. Additionally, ridge regression can estimate coefficients of multiple regression models in scenarios where the independent variables are highly correlated. The RidgeRegression function was imported from sklearn linear model to perform the computations. The model outputted an MAE score of 2.351, MSE score of 16.656, RMSE score of 4.081, and R^2 score of 0.831. The R^2 score of 0.831 indicates that the model explains approximately 83.1% of the variability in the used car prices, which is a slight improvement over the linear regression model previously used. This means it explains slightly more of the variance in the data set. Ridge regression can outperform linear regression in some cases because it is used to reduce overfitting problems of the linear regression model by increasing the complexity of the data set.
Additionally, the residuals plot in Figure 10 shows a distribution centered close to zero with a bell-shaped curve, which is ideal as it suggests that the model errors are normally distributed. The scatter plot in Figure 11 closely aligns along the 45-degree line, indicating a strong correlation between predicted and actual prices. However, there are a few outliers that might be the cause of errors leading to a higher MSE. The learning curve for Ridge Regression in Figure 12 illustrates a good convergence between the training and cross-validation scores, which stabilizes as more training examples are used. This suggests that the model is generalizing well from the training data to unseen data, although there's a slight decline in the training score, possibly due to the increased bias from the Ridge penalty. By penalizing the magnitude of the coefficients, Ridge Regression manages to reduce model complexity without significant sacrifices in performance, thus mitigating overfitting effectively.

Lasso Regression

Next, we decided to use Lasso Regression to improve prediction accuracy and model interpretability by selecting variables and simplifying the model by applying a penalty that reduces the coefficients of less important features to zero. The benefit of Lasso regression is applying a penalty to prevent overfitting and enhance the accuracy of statistical models. We also wanted to looked into whether a different balance of bias-variance trade-off from ridge and linear regression could yield better results. The LassoRegression function was imported from sklearn linear model to perform the computations. The model outputted an MAE score of 3.635, MSE score of 32.069, RMSE score of 5.663, and R^2 score of 0.675. These results indicate that Lasso Regression has a higher error value when the predictions are further away from the true values since the model has significantly higher values for MAE and MSE, and is the model with the highest RMSE value. Lasso regression has a lower R^2 score compared to linear and ridge regression. This indicates that the Lasso Regression model is less successful than other models in explaining the variance in the data set. This might result from too much penalization leading to underfitting, or that the features selected through Lasso's regularization did not capture all the necessary predictive signals.

The histogram in Figure 13 shows residuals that are mostly centered around zero but with a distribution that suggests potential biases or an inability to capture all the variability in the data. The scatter plot in Figure 14 shows a greater dispersion of points around the 45-degree line compared to Ridge Regression, indicating less accurate predictions, especially at the higher price range. The learning curves in Figure 15 show a consistent gap between the training and cross-validation scores, which do not converge as closely as those for Ridge Regression. This gap often indicates a model that could be underfitting, as it does not perform equally well on unseen data. The Lasso Regression's lower performance, as indicated by higher error metrics and a lower R^2 score, suggests it might not be the best model for this particular dataset, due to its tendency to overly simplify the model, thus missing out on critical predictive features.

Random Forest Regression

Lastly, we decided to use random forest regression since it's a powerful model that can handle the dataset containing continuous variables for both classification and regression tasks. Additionally, the random forest has extremely high accuracy, scales well, is interpretable, and robust which applies well to our problem. This ensemble learning method, which operates by building multiple decision trees and merging them to get a more accurate and stable prediction, is particularly well-suited for datasets with high dimensionality and complexity. The RandomForestRegressor function was imported from sklearn ensemble to perform computations. The model outputted an MAE score of 1.349, MSE score of 6.133, RMSE score of 2.476, and R^2 score of 0.938. Random forest regression can explain around 93.8% of the variance in the dataset due to the high R^2 score. This is significantly higher compared to the linear, ridge, and lasso regression models previously tested. Random Forest Regression also has lower MAE, MSE, and RMSE values compared to other regression models. This indicates that the estimates are quite close to the actual values and further validate the model's high accuracy and predictive quality. It also has lower error values and higher explanatory power than other models.
The residual distribution in Figure 16 is tightly centered around zero with a sharp peak, suggesting that most predictions are very close to the actual values, and there are few outliers. The scatter plot in Figure 17 closely aligns along the 45-degree line, which clearly shows that the predicted prices are very close to the actual prices across the range, with few deviations. The learning curves in Figure 18 indicate that the model performs consistently well as more training data is provided, with the training score plateauing and the cross-validation score gradually increasing, indicating good generalization ability. Overall, the Random Forest Regression model has outperformed other tested regression models in all key metrics, proving its efficacy in handling the prediction of used car prices with high accuracy.

Regression Training vs Test Accuracy

Based on the 80/20 split between the training and test datasets, we obtain the training and test set accuracy for different regression models as seen in Table 1. Overall, linear regression performs well on both training and test sets, and doesn't indicate over/underfitting. The close performance metrics between the training and test sets indicate excellent model generalization. This proves that linear regression, while straightforward, proves effective when relationships in the data are predominantly linear and the model complexity is not a limiting factor for accuracy. Ridge regression has similar performance as linear regression on the training set, but has higher test set accuracy. This indicates that Ridge's regularization helps mitigate overfitting by penalizing large coefficients. This is beneficial in scenarios with multicollinearity or when the feature space is large. Lasso Regression achieves lower accuracy compared to the other regression models due to underperformance on training and test sets. This indicates that Lasso Regression could be discarding important features in the data due to reducing the coefficient size too much. Some underfitting is occurring since the model is too simple to capture the underlying patterns. Lastly, Random Forest Regression achieves high accuracy on both training and test sets compared to all the other regression models. There is some degree of overfitting since there is a large difference between then training and test set.

Support Vector Regression (SVR)

In Figure 19 and Table 2, we have the results of using Support Vector Margin Regression, or Support Vector Regression (SVR) on our dataset to create an effective model to predict prices given certain features. To fit this model, we first created our features and targets, and then split our data into training and testing sets. We then used StandardScaler() to standardize all of our features in relation to each other. Our features, 'X' consist of every column of our original dataset, but without the 'Price' column. We let our target variable 'y' be the price column. After fitting an SVR model to our data, we then identify the support vectors, or the points that most influenced the resulting line of best fit. In our plot above, these vectors are denoted by a green 'X' symbol, and we can see that they lay along the line of best fit, and are clustered together closer towards the origin, where the density of data points is much higher. Once we find and plot our support vectors as well, we can see the line of best fit does a pretty good job at estimating the correct prices. We can see that our MAE (Mean Absolute Error) is around 2.01, and our MSE (Mean Squared Error) is around 19.2. Our MSE and MAE being relatively low are a good sign, showing that a linear model works quite well in this situation. This means that there seems to be a relatively strong positive linear relationship between the input features and the resulting resale price.

Kernelized Support Vector Regression (SVR)

After plotting a linear SVR model and examining the loss and support vectors, we wanted to see if kernelizing the data differently could make any changes to the result of the SVR. In this next part, we explored the effect of polynomial, linear, and RBF (Radial Basis Function) kernels on our data. To create our features 'X' and target 'y' we followed mostly the same exact steps that we followed for the previous example. However, we used varying kernels, which are described below.

For the linear kernel, we just compute pairwise dot products between all of our feature vectors. The formula is listed below. This is shown in Figure 20.

$$K_{linear}(x_i, x_j) = x_i \cdot x_j$$

For the polynomial kernel, we compute the kernel pairwise for all input feature vectors using the formula below. This is shown in Figure 21.

$$K_{polynomial}(x_i, x_j) = (1 + x_i \cdot x_j)^d$$

Where d is the degree. In our case, $d = 3$. For the RBF kernel, we compute the kernel pairwise for all input features using the formula below. This is shown in Figure 22.

$$K_{rbf}(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$$

In our model, we did not specify a value for σ, meaning that the SVR trains and then determines an optimal σ to achieve minimal losses.

After manipulating our input feature vectors using each of these different transformation functions listed above, we then plot the best fit line for each of the three different kernels. We want to see if transforming the data has unearthed any underlying nonlinear relationships between the input features and the line of best fit. If either the polynomial or RBF kernelized data had a line of best fit with smaller error than the linearly kernelized data, we can then infer that there is a stronger nonlinear than linear relationship between the input features and the price of a used car. However, we can see from our error outputs in Table 3 that the linear kernel resulted in the best line of best fit, followed by the RBF kernel, and then followed by the polynomial kernel. Both the MAE and MSE reflect this fact. After conducting this additional experiment, we can conclude that some sort of linear model is the best way to represent the relationship between input features and the resale price of a car.

Feature Importance Extraction

Clustering

We used k-means clustering combined with PCA to analyze which features are most important or affect the prices of used cars most. Clustering is instrumental in discovering patterns and categorizing vehicles based on features such as mileage, engine size, power, and other relevant attributes. In addition, k-means clustering is robust since it can handle outliers and it also scales well to large datasets. Principal Component Analysis (PCA) is used to reduce the dimensionality of the dataset.
The use of k-means clustering in this context starts by selecting the optimal number of clusters via the Elbow Method as shown in Figure 23, which involves plotting the sum of squared errors (SSE) for different numbers of clusters and identifying the point where the decrease in SSE becomes less pronounced. This "elbow point" represents a balance between model complexity and explanatory power, suggesting an optimal cluster count.

In our analysis, the used cars are grouped into three main clusters as shown in Figure 24:

1. Budget-Friendly Cars (Cluster 0): Cars in this cluster typically have higher mileage, older model years, smaller engines, and manual transmissions, often belonging to less expensive brands. Older models like Maruti Suzuki Alto and Hyundai Santro are common examples and the purpose is to attract buyers seeking economical, fuel-efficient vehicles for daily commuting.

2. Mid-Range Family Cars (Cluster 1): These vehicles generally feature moderate mileage, mid-sized engines, and a mix of manual and automatic transmissions. They tend to be newer than those in the budget-friendly category but not as recent as the premium cars. Common examples are Hyundai i20, and Toyota Corolla. This category is ideal for families requiring reliable and comfortable transportation without a premium price.

3. Premium and Performance Cars (Cluster 2): Vehicles in this cluster are characterized by low mileage, larger engine capacities, recent model years, automatic transmission, higher power, and luxury amenities. This generally includes models such as BMW 3 Series, Audi A4, and Mercedes-Benz C-Class. This category is to appeal to consumers desiring luxury, high performance, and cutting-edge technological features at a slightly discounted price.

Overall, the clustering approach reveals that the most important features tend to be Name (brand/model), Year, Kilometers Driven, Mileage, Engine, and Price. Budget-friendly cars are distinctly separated based on age and mileage. Mid-range cars often overlap slightly with budget and premium cars, indicating a gradual change in features. Premium cars are usually well-separated from the others due to distinct features like low mileage and high power. The results as shown in Figure 14 indicate that there are a lot fewer Cluster 2 (Premium and Performance) cars than Cluster 0 (Budget-Friendly Cars).
Using PCA along with clustering has benefits, however, one limitation is that PCA compresses the data, potentially losing some information that affects cluster interpretation. Additionally, there is some overlap, particularly between Clusters 1 and 2, which could indicate either a transition zone where certain characteristics are shared or potential issues with how well the k-means algorithm has fitted the data. Different clustering algorithms, like DBSCAN or hierarchical clustering, might be explored to assess their efficacy in this context.

Gradient Descent

Another technique that we used to identify the relationships between feature variables and price was gradient descent. In this case, we are using gradient descent with Mean Absolute Error (MAE) and Mean Squared Error (MSE) as our loss functions. Gradient descent aims to find the ideal model that minimizes the loss between our predicted and target prices by adjusting the model parameters based on the gradient of the loss function. For our model, we first initialized our MSE loss function with L2 regularization, to reduce overfitting. We then define a function that calculates the gradient of the MSE loss function with respect to the weights of our model. Afterwards, we define the main gradient descent function which also takes in the learning rate, or step size. We then iteratively update our weight and bias vectors using the gradient of the loss function at each step. This function then returns the optimized weight, bias, and loss vectors, which we then use to make our predictions. The errors of this model are as seen in Table 4.
These errors and R^2 scores from gradient descent using MSE tell us that although the train and test MSE's are higher than we'd like, the train and test R^2 scores are very good. We know that R^2 scores range between 0 and 1, so both of them being in the 0.82-0.86 range indicates a good fit, since an R^2 score of 1 represents a perfect fit. Our train and test MSE's tell us that on average, our predictions stray by about 17 to 19 units from the true value. We then repeat the same process as above using MAE rather than MSE. The next step in evaluating our gradient descent model was to see if we could find the optimal hyperparameters to find a configuration that resulted in an even lower error. We defined a method that took in features, targets, step sizes, regularization parameters, and the number of iterations for gradient descent. This method simply iterations through all the possible learning rates and regularization parameters and finds the error of the gradient descent model for each combination. We then output the best parameters along with the corresponding error values and plot the loss curve over many iterations, the residual plot, and a graph showing the relative importance of each feature. Below, we've included all of these visualizations and the chart depicting the best error values as a result of hyperparameter tuning.
From Table 5, we can see that hyperparameter tuning has not resulted in a good model whatsoever. The MSE values are extremely high, and although the MAE is low, the R^2 scores are very poor for both training and testing. A sub 0 R^2 score indicates very poor fit, and combined with the poor MSE, we can conclude that these hyperparameters were not very good.
From the visualizations in Figures 25, 26, and 27, we can see that after around 200 iterations we are already very close to the loss that is achieved after 1000 iterations. In the feature importance chart we can see that relative to the other features, the year in which the car was being sold was extremely important when calculating price, followed by the engine power. In our residual plot we can see that up until a predicted price of 10 Lakhs, the residuals seem to be following a negative linear trend, which indicates that for cars that would sell for under 10 Lakhs, this model may not be the best fit. For all predicted prices above 10 Lakhs, the residuals seem to be randomly and evenly distributed above and below the 0 line, indicating a good fit. Overall, while looking at our error values and these three plots shown above, it seems that although gradient descent doesn't result in the most perfect model, it does result in a model that is decently accurate and fits relatively well.

Conclusion

After using a variety of models, various loss functions, and examining various plots we can come to the conclusion that although not perfect, a linear model is the best way to go about modeling the relationship between all the input features (in this case, engine power, number of owners, kilometers driven, etc.) and the final predicted price. Surprisingly, we also saw that the year in which the car was sold was a big factor in determining the price, and this could be because of the inflation fluctuation between years as well. While using Kernelized Support Vector Regression (or SVR), we also saw that a polynomial or RBF model had a higher error rate than our linear kernel, again reinforcing that the linear model was the best choice. We even see in our gradient descent model that for cheaper cars, a linear model may not be the best fit but as the price increases, a linear model becomes more and more accurate as opposed to others. From all of our regression models, we can see that the residuals tend to hover around zero, implying that our line of best fit is doing a good job at predicting car prices given input features. Even in our initial data exploration, we had seen that there could be a positive linear relationship between Engine Size and the Price which is again reiterated in our models. We thought that logically it made sense for the price to go up if the car had a more powerful engine, so a lot of our findings were quite intuitive and easier to interpret. Overall, using various machine learning techniques and models helped us solidify our understanding of the relationships between the various features and the predicted price, as well as the most important features.

Future Improvements

There are additional features that we could potentially add to this model. As mentioned in the introduction of this report, one implication of a successful model that can accurately predict car prices is being able to expand this model to vehicles other than cars. Almost half of all Indian households own a motorcycle or a scooter - both are cheaper alternative methods of transportation and easier to store compared to cars, making them more attractive options to people who either aren't able to afford a car or aren't able to keep one at their place of living. Being able to expand this model to take in different vehicle types as input could be one way to generalize this model for use across all automobile types. One potential feature is to select which vehicle (car, motorcycle, scooter, truck, etc.) you are looking at and input corresponding features, which could get a price estimate for that specific vehicle. Our model should be able to correctly identify and give a price estimate using data specific to the inputted vehicle. Other than being able to generalize our model across multiple automobile types, one other improvement we could make could be adding additional features. One feature that could be useful to have as input could be a damage report that details all damage done to the car prior to selling. We could then use feature engineering and encoding to represent the damage report as a number from 0 to 10, reflecting the amount of damage the car has sustained.

Fairness & Weapons of Math Destruction

In the used car market, predictive modeling can significantly impact both buyer and seller behaviors by determining the fair market value of vehicles. Models applied to the used car market can lead to fairness issues and potentially become what Cathy O'Neil describes as "Weapons of Math Destruction" (WMDs). These concerns are particularly relevant when models disproportionately affect certain demographics or perpetuate existing biases.

The goal of our project is to provide accurate and fair predictions of used car prices based on features such as make, model, year, mileage, condition, and location. However, if not properly handled, the model could inadvertently favor certain brands or types of cars, thereby influencing market dynamics in a biased manner. For instance, if the data used to train the model has inherent biases such as higher values consistently assigned to cars from specific countries or manufacturers, the model could replicate these biases, affecting fairness in pricing predictions. Some potential Weapons of Math Destruction can result from scale and impact since the use of these predictive models can be extensively used in the used car market, potentially impacting a large number of individuals. If the model inaccurately predicts higher prices for less reliable cars based on biased historical data, it could mislead buyers, affecting their investment and safety. Additionally, if the criteria and data used for these predictions are not transparent, sellers and buyers may not understand how the prices are determined. This can lead to mistrust and market inefficiencies. Lastly, mispriced cars can lead to financial losses for buyers and may also impact sellers if cars are undervalued. This can alter the economic stability of individuals relying on these vehicles for essential daily activities.

Overall, to ensure fairness in predictive models used in the used car market, we think it's important to incorporate a more diverse dataset that reflects a wide range of cars, conditions, and historical transactions to avoid biases towards certain types or makes of cars. Additionally, carefully selecting and weighting features that contribute to car valuation is crucial. For instance, while the car's age and mileage are significant predictors, their impact should be balanced with other factors like maintenance history and market demand.

In the used car market, ensuring the fairness of predictive models is vital. This approach not only enhances the market's efficiency and trustworthiness but also supports ethical business practices and consumer protection.

References

Information on proportion of India using two-wheelers from:
`https://timesofindia.indiatimes.com/auto/policy-and-industry/india-on-sixth-in-list-of-countries-with-highest-two-wheeler-usage-check-full-list/articleshow/103320483.cms` Information on how many people in India use motorcycles from:
`https://theprint.in/india/only-8-indian-families-own-cars-nfhs-finds-over-50-still-use-bicycles-bikes-scooters/971413/#:~:text=The%20vast%20majority%20of%20Indians,cent%2C%20according%20to%20the%20survey`
Information on fairness from:
`https://people.orie.cornell.edu/mru8/orie4741/lectures/fairness.pdf`
Information on Weapons of Math Destruction from:
`https://people.orie.cornell.edu/mru8/orie4741/lectures/limits.pdf`

Appendix

Contributions

Sunny Sun:
Code - Data Cleaning, Exploratory Data Analysis, Feature Engineering, Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression, Clustering, Gradient Descent.
Report - Introduction, Approach, Regression Methods (Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression), Clustering, Fairness & Weapons of Math Destruction, References, Appendix.

Sanjana Vakacherla:
Code - Data Cleaning, Exploratory Data Analysis, Feature Engineering, Support Vector Regression, Kernelized Support Vector Regression.
Report - Problem Background, Dataset Description, Data Exploration and Feature Engineering, Support Vector Regression, Kernelized Support Vector Regression, Gradient Descent, Future Improvements, Conclusion, References.

Figures/Tables

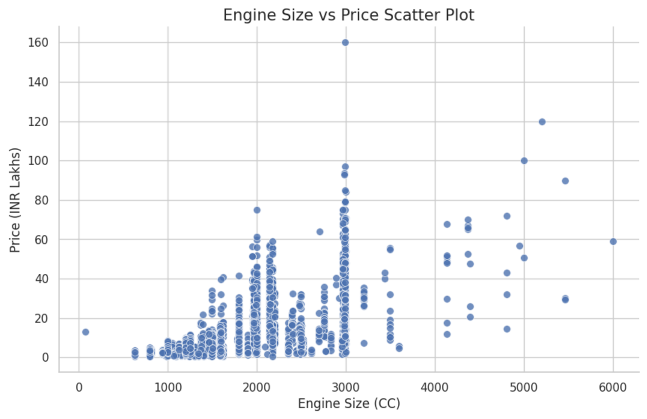

Figure 1: Engine Size (CC) vs Price (INR Lakhs)

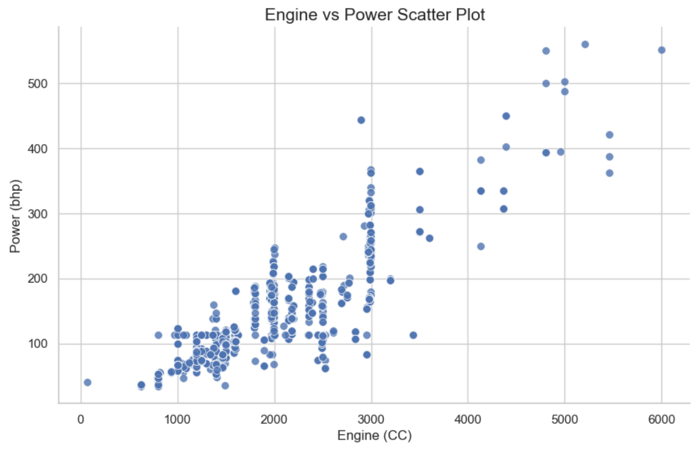

Figure 2: Engine Size (CC) vs Power (bhp)

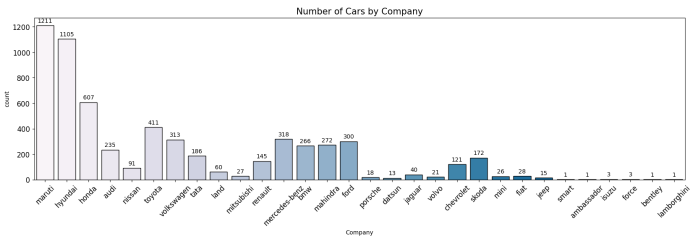

Figure 3: Number of Cars per Company

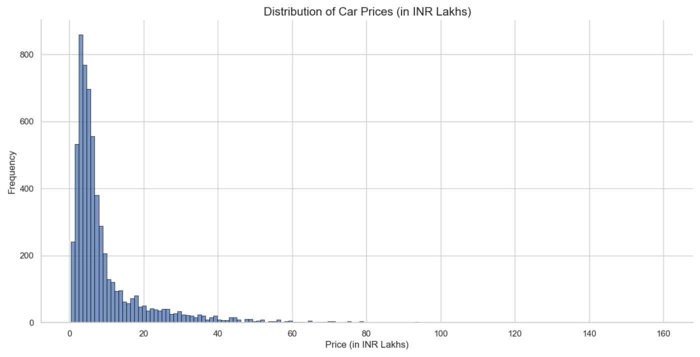

Figure 4: Distribution of Car Prices

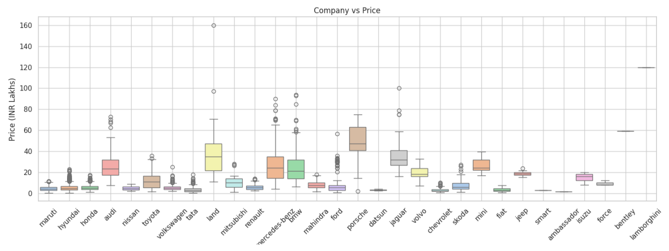

Figure 5: Company (CC) vs Price (INR Lakhs)

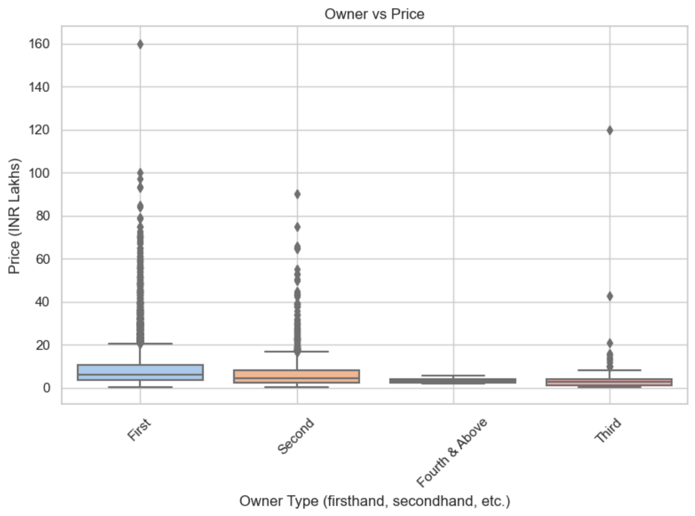

Figure 6: Owner vs. Price

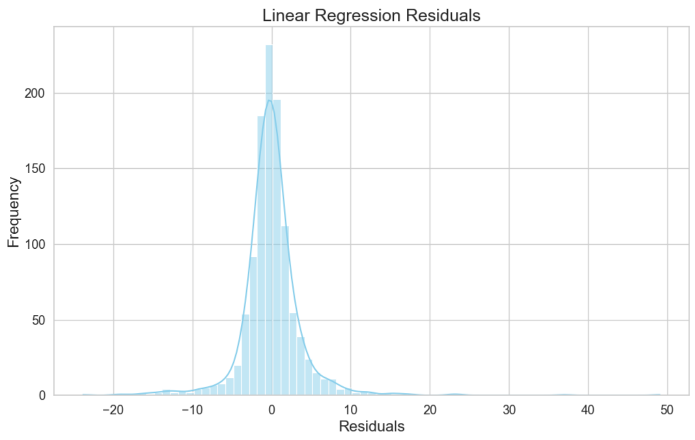

Figure 7: Linear Regression Residuals

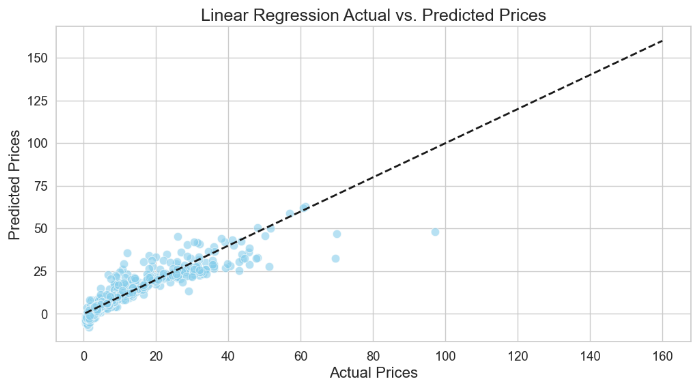

Figure 8: Linear Actual vs. Predicted Prices

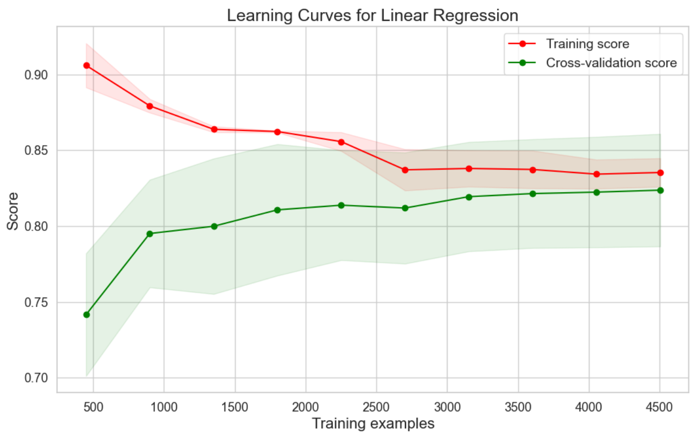

Figure 9: Linear Regression Learning Curves

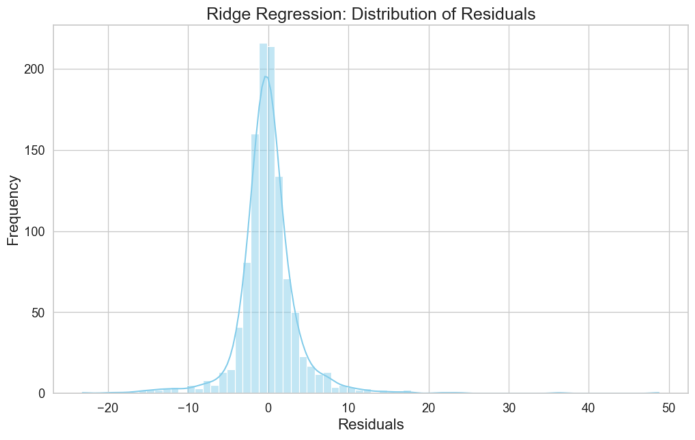

Figure 10: Ridge Regression Residuals

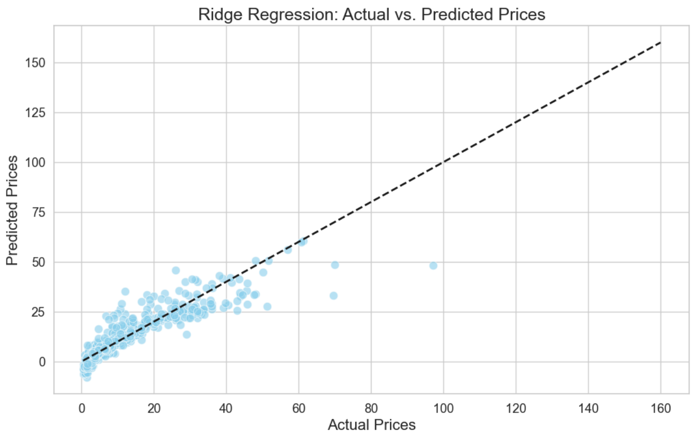

Figure 11: Ridge Actual vs. Predicted Prices

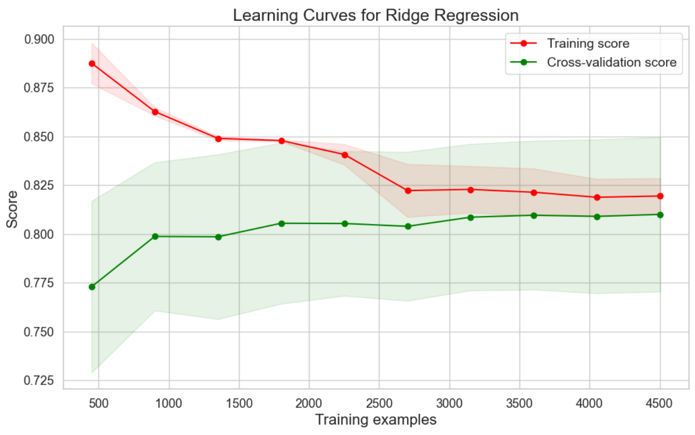

Figure 12: Ridge Regression Learning Curves

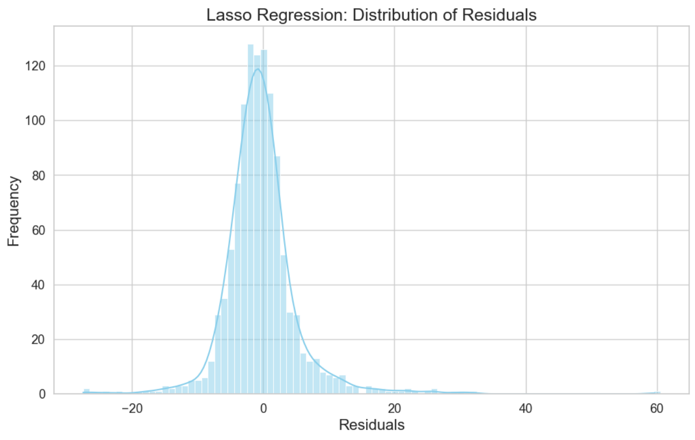

Figure 13: Lasso Regression Residuals

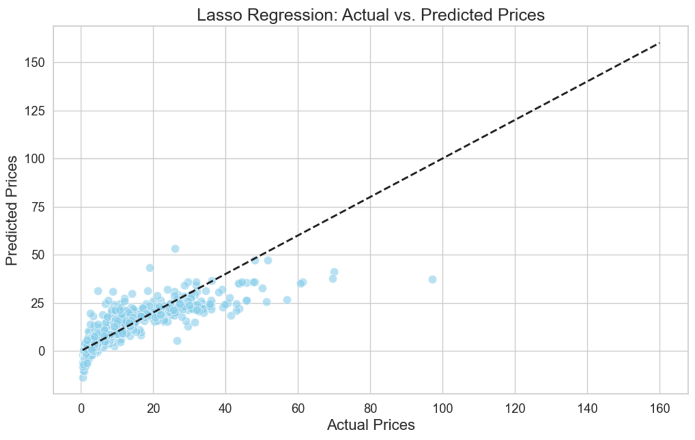

Figure 14: Lasso Actual vs. Predicted Prices

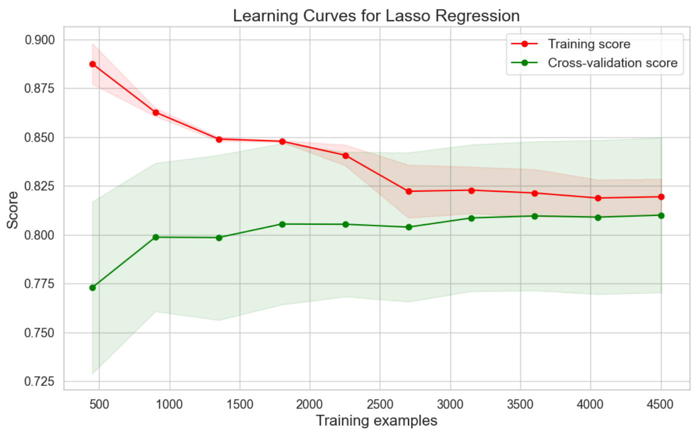

Figure 15: Lasso Regression Learning Curves

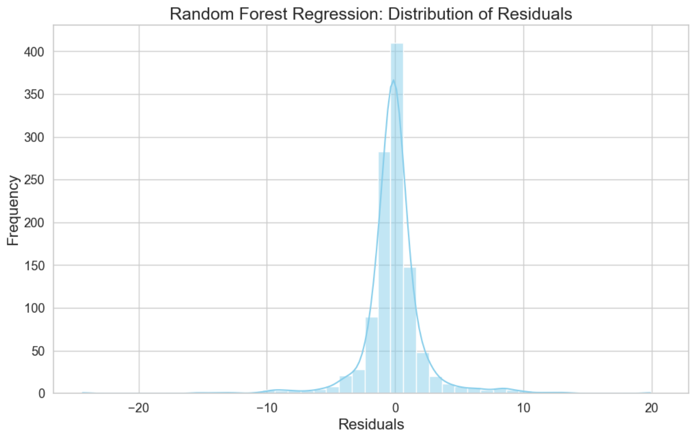

Figure 16: Random Forest Regression Residuals

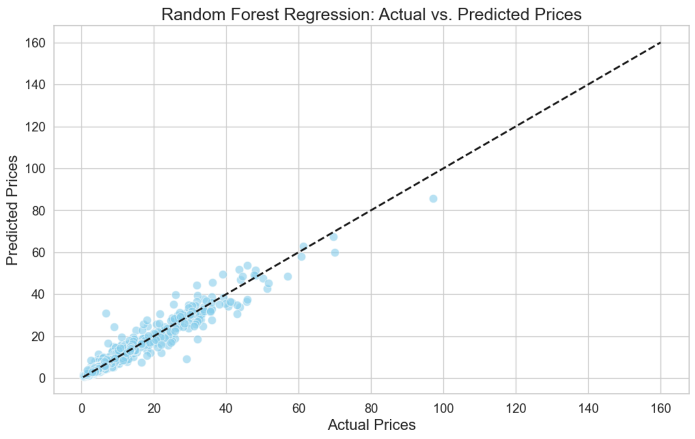

Figure 17: Random Forest Actual vs. Predicted Prices

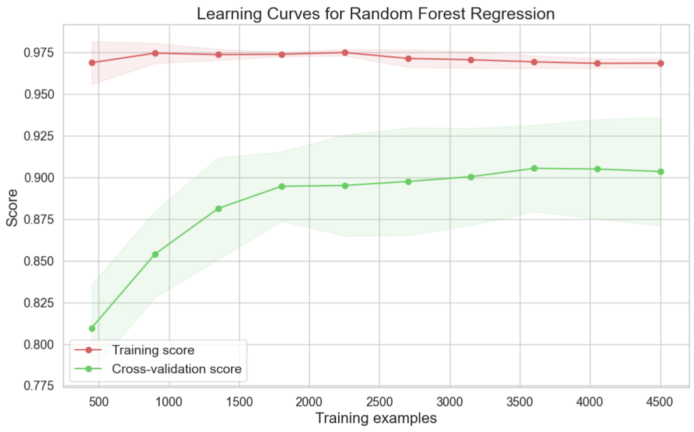

Figure 18: Random Forest Regression Learning Curves

Model	Training Set Accuracy	Test Set Accuracy
Linear Regression	0.833643	0.829888
Ridge Regression	0.833144	0.83099
Lasso Regression	0.667695	0.67459
Random Forest Regression	0.968988	0.937772

Table 1: Regression Training and Test Accuracy

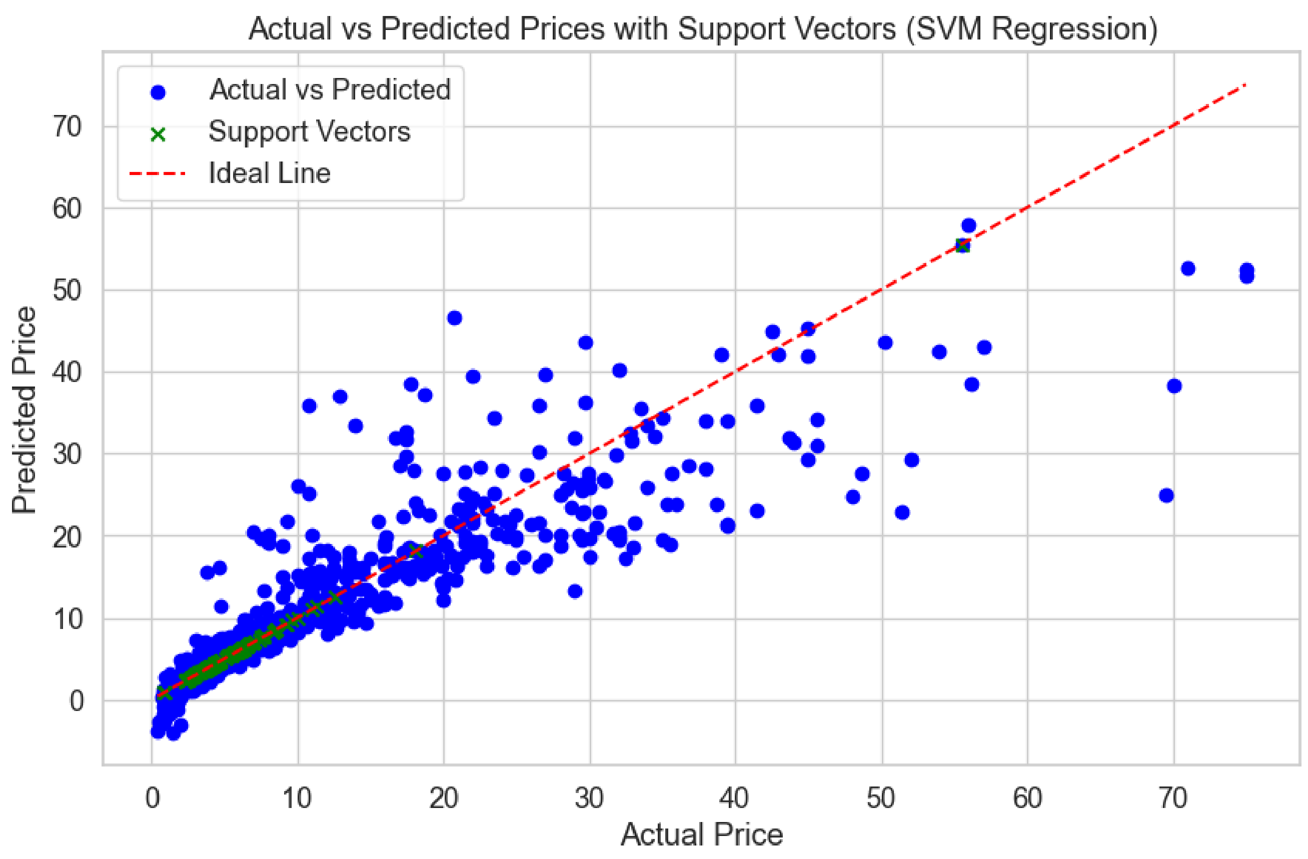

Figure 19: Support Vector Regression

Error Type	Error Value
Mean Absolute Error	2.0153622135568066
Mean Squared Error	19.229524538463696

Table 2: SVM Mean Absolute and Square Error

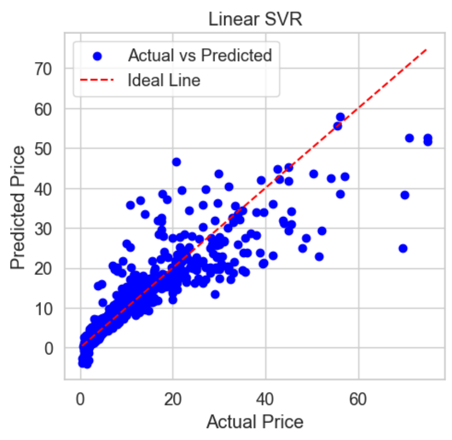

Figure 20: Linear Kernel SVR

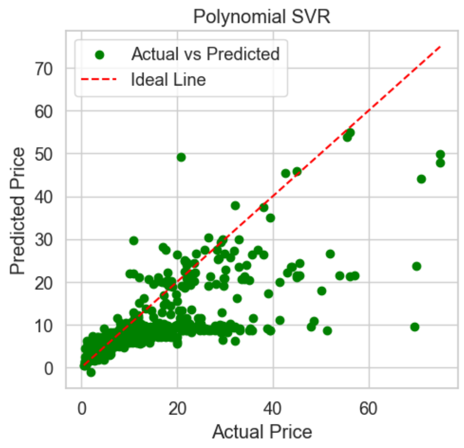

Figure 21: Polynomial Kernel SVR

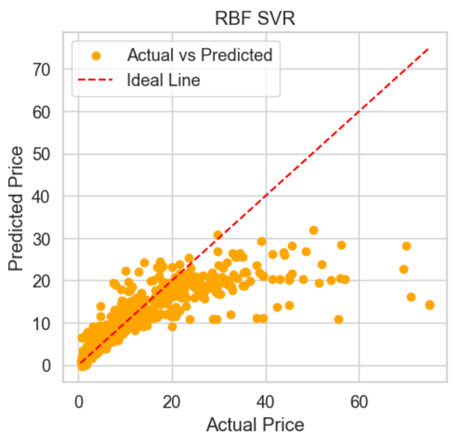

Figure 22: RBF Kernel SVR

Model	Error Type	Error Value
Linear SVR	Mean Absolute Error (MAE)	2.015362
Linear SVR	Mean Squared Error (MSE)	19.229524
Polynomial SVR	MAE	3.385549
Polynomial SVR	MSE	46.933168
RBF SVR	MAE	2.449079
RBF SVR	MSE	37.578966

Table 3: Kernelized Support Vector Regression Error

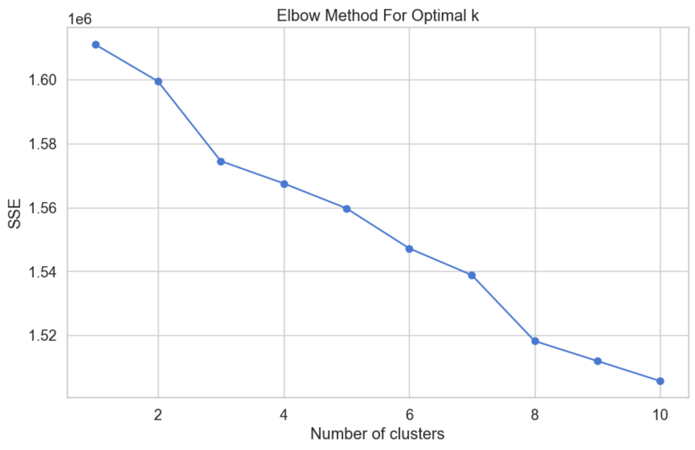

Figure 23: Clustering Elbow Method

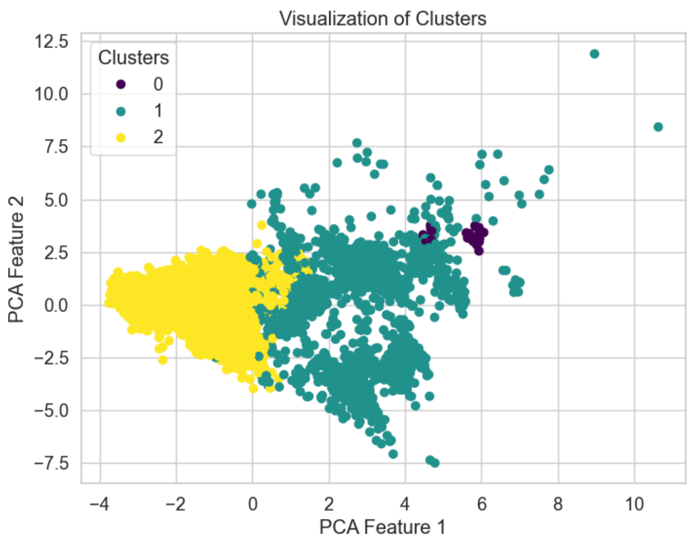

Figure 24: K-means Clustering

Metric	Value
Train MSE	19.122530
Test MSE	17.477262
Train R^2 Score	0.853687
Test R^2 Score	0.826974

Table 4: Gradient Train and Test Error

Training Metrics:

Metric	Value
MSE	134.552769
MAE	5.729577
R^2 Score	-0.029506

Testing Metrics:

Metric	Value
MSE	108.439842
MAE	5.466198
R^2 Score	-0.073560

Table 5: Training and Testing Metrics

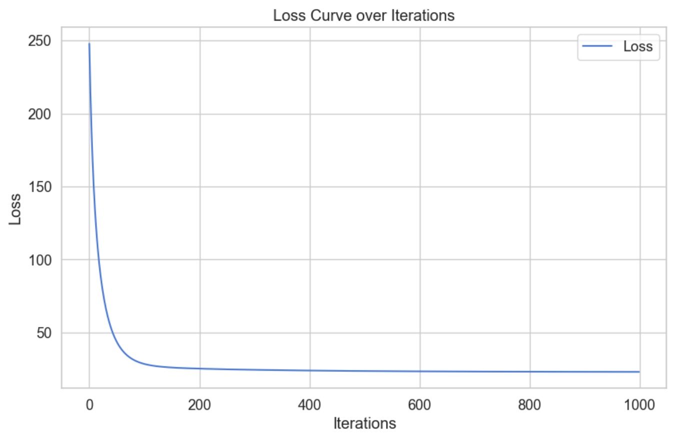

Figure 25: Loss Curve over Many Iterations

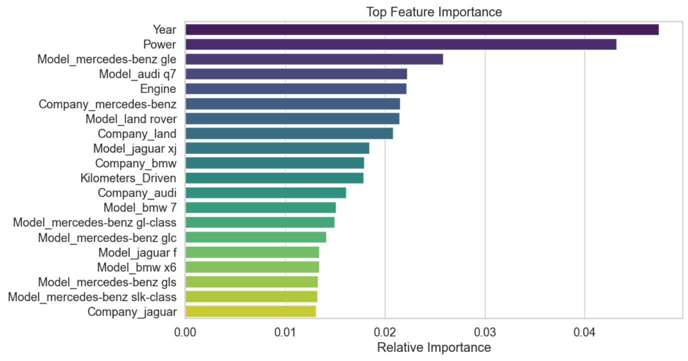

Figure 26: Top Feature Importance

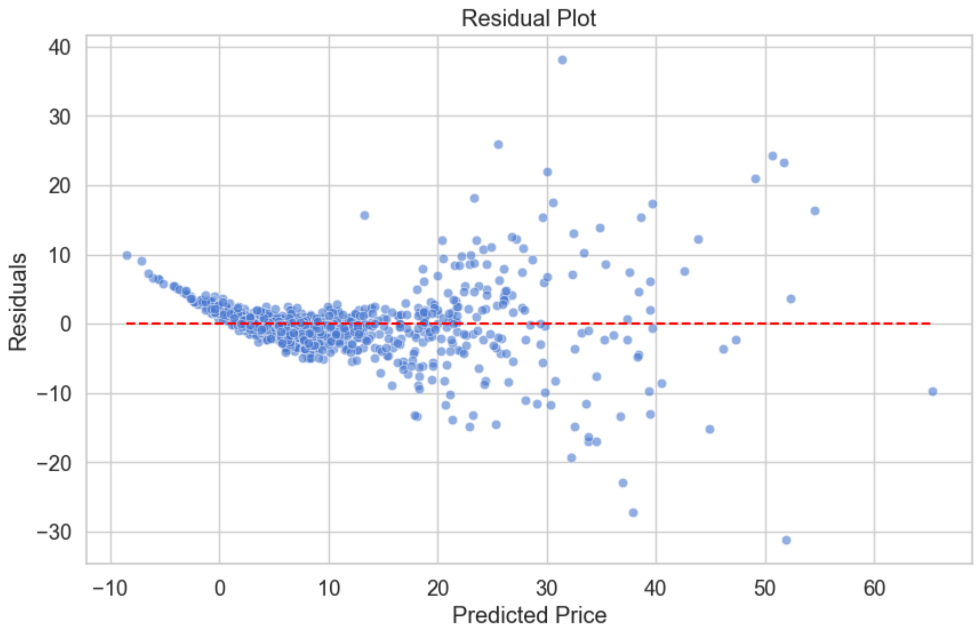

Figure 27: Residual Plot

Link to GitHub repository

https://github.com/sunny525s/ORIE-4741-Final-Project