CS5344 Big-Data Analytics Technology
Final Report – Team 09

1. Motivation

In today's automobile market, determining a vehicle's true value remains a complex challenge. While experienced dealers might rely on intuition and market knowledge, there's growing demand for a more systematic, data-driven approach to vehicle valuation. This project explores how machine learning can help predict vehicle pricing by analyzing patterns across millions of car listings. By simultaneously considering various factors such as a vehicle's make, model, year, condition, and additional attributes, this project's approach delivers accurate price predictions, surpassing traditional methods that rely on intuition or limited information.

This project's impact goes beyond individual sales, offering a reliable pricing model that:

1. Help buyers make informed decisions with confidence.
2. Enable sellers to set competitive and fair prices.
3. Provide greater transparency to the automotive market.

2. Target task

The goal of this project is to predict the *resale price of a vehicle* by analyzing various features such as mileage, age, brand, colour, engine specifications, and other vehicle attributes. Using these factors, regression-based machine learning models aim to capture the key determinants of vehicle resale value, providing accurate price predictions for buyers and sellers alike:

Regression Models Used:

1. Linear Regression
2. Decision Trees
3. Random Forest
4. Gradient Boosted Trees (GBT)
5. Extreme Gradient Boosting (XGBoost)

These models were chosen to capture both linear and non-linear patterns in vehicle pricing. *Linear Regression* helps in understanding simple relationships between features and price. *Decision Trees* and *Random Forests* handle complex interactions and non-linearities effectively, while *Boosting methods* like GBT and XGBoost iteratively refine predictions, providing higher accuracy and robustness. This combination of Regression models ensures diverse approaches to modeling vehicle prices.

Key Performance Indicators (KPIs):

To evaluate the performance of the regression models, the following metrics were used:

1. *R^2 Score:*
   Represents the proportion of the variance in the target variable that is captured by the model, indicating how well the model fits the data.
2. *MAE (Mean Absolute Error):*
   Measures the average magnitude of errors in predictions, without considering whether the errors are positive or negative.
3. *MAPE (Mean Absolute Percentage Error):*
   Represents the average percentage difference between predicted and actual values, providing a measure of prediction accuracy in relative terms.
4. *RMSE (Root Mean Square Error):*
   Quantifies the average magnitude of prediction errors in dollars, with larger errors given more weight, offering a more sensitive measure of model performance.

These metrics give a comprehensive way to check how well the model works. They help measure how much the model explains the data, how accurate the predictions are, and how big the errors can be. This makes sure the model's predictions are clear, accurate, and useful in real life.

Note: Both MAE and RMSE are expressed in dollars to reflect their direct impact on price predictions.

3. Dataset

This dataset contains over 3 million new and used vehicle listings from across the United States. The dataset can be accessed through Kaggle at USA-used-cars-dataset.

Key Features:

- **Vehicle Information:** Columns like model_name (representing the vehicle model), year (manufacturing year), make_name (name of the car company) provide essential information about the car's identity and age.
- **Physical Attributes:** Technical specifications, such as engine_displacement and fuel_type, describe the vehicle's physical and mechanical characteristics, which play a significant role in determining its value.
- **Market & Seller Details:** Variables like dealer_zip (location of the dealer), seller_rating (ratings given to the seller), and daysonmarket (time the vehicle has been listed for sale) provide insights into market trends and seller reliability, impacting the resale value of the used vehicle.

Numerical and Categorical Data:

The dataset includes both numerical features (e.g., mileage, horsepower, fuel_tank_volume) and categorical features (e.g., model_name, fuel_type, body_type). This mix provides a rich dataset suitable for building machine learning models, enabling them to capture complex patterns in vehicle pricing.

Target Variable:

The price column serves as the target variable, representing the final sale price of the vehicle and allowing the model to predict accurate vehicle values.

3.1) Big Data Relevance

This project involves a massive dataset of 3 million vehicle listings, presenting challenges in data storage, processing, and analysis. The dataset's diverse features range from complex vehicle specifications like torque and transmission_display to simpler attributes like colour. This adds a layer of complexity that requires advanced feature engineering and data preprocessing techniques. To manage the scale and depth of the data, **PySpark** is used throughout the project, enabling large-scale and parallel processing for efficient handling of big data requirements in model training and prediction.

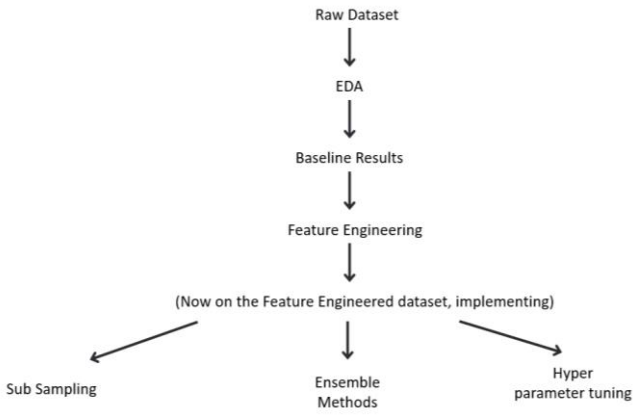
3.2) Challenges with Big Data:

In addition to handling data volume, variety, the project faces challenges related implementation of machine learning algorithms that balance scalability and accuracy. Another critical factor is the time and computing resources required to train models on such a large dataset.

4. Project Workflow

The project workflow begins with the raw dataset, which undergoes Exploratory Data Analysis (EDA) to clean and uncover insights. Following EDA, baseline model results are established to provide a starting point for improvement. Feature engineering is then applied to enhance the dataset with additional meaningful variables, setting the stage for more advanced modelling and prediction techniques.

With the feature-engineered dataset, the process continues through hyperparameter tuning, ensemble methods, and sub-sampling to optimize model performance and increase accuracy in predicting vehicle prices. This structured workflow ensures a systematic and iterative approach to building a reliable pricing model.



- This report also takes a similar route as the flow chart. We first dive into eda, then basline results, then proposal, implementation and results to beat the baseline

The project follows a structured approach to achieve accurate vehicle price predictions. Starting on the raw dataset from Kaggle, I am applying *Exploratory Data Analysis (EDA)* to clean and understand the data and generating baseline results. *Feature engineering* is then performed to enhance the dataset for better model performance. Ensemble methods such as *Bagging* (using XGBoost and Decision Trees) are employed to improve accuracy and robustness.

Several experiments were conducted to optimize the models and enhance performance. Hyperparameter tuning was applied to adjust model parameters for improved accuracy. Sub-sampling techniques, including both row-wise and column-wise sampling, were used to manage and learn about data variability and thus ensure efficient processing. Additionally, BERT embeddings were utilized to transform the description column into meaningful vector features, adding depth to the dataset for better predictions.

5. Approaches

5.1) EDA

Through Exploratory Data Analysis, I carefully examined, cleaned, and improved the raw dataset to address data inconsistency issues. EDA also helped organize the data properly, creating a solid foundation for building predictive models. In this project, EDA was divided into column-wise and row-wise analysis. Column-wise EDA focused on removing irrelevant features, adding meaningful ones, and ensuring feature relevance for prediction. Row-wise EDA dealt with handling missing values in both numerical and categorical columns to maintain data consistency and accuracy.

5.1.1) Columnar EDA

In this project, the dataset originally had 66 columns, which were reduced to 42 columns after removing 27 unnecessary features and adding 5 new columns (major_options_count, log_mileage, combined_fuel_economy, legroom, and RPM).

The following actions were taken during column-wise EDA:

1. I removed columns with more than 40% missing data, as filling in so many missing values could distort the patterns in the data, which I wanted to avoid. By dropping these columns, I ensured that the remaining data would be more consistent and reflective of the true relationships within the dataset.

Bin	count
0-10%	44
10-20%	6
40-50%	7
80-90%	2
90-100%	7

Figure 1: Column distribution across missing value % Bins

Figure 1 shows how columns are grouped based on the percentage of missing values, highlighting how many columns fall into each null percentage category/bin.

2. Eliminated columns with overlapping or similar information, such as ‘engine_type’ and ‘engine_cylinder’, as they conveyed the same values.
3. Dropped irrelevant columns like IDs (‘vin’, ‘listing_id’) and URLs (‘main_picture_url’) that did not contribute to the predictive model.
4. Processed columns with string formats (e.g., '35.1 in') into numerical format (e.g., 35.1) to make the data suitable for predictive modeling.
5. Merged columns, such as ‘city_fuel_economy’ and ‘highway_fuel_economy’ into ‘combined_fuel_economy’, and ‘back_legroom’ and ‘front_legroom’ into ‘legroom’.
6. Separated the torque column into ‘torque’ and ‘RPM’ to better capture the relationship between these features.
7. Introduced new columns like ‘major_options_count’ to represent the number of features the car had and ‘log_mileage’ to address uneven mileage distribution.

5.1.2) Row-wise EDA

In this project, row-wise EDA was conducted to address missing values and ensure data consistency while preserving the quality of the dataset.
I analysed the data distribution for each column by plotting their values before deciding on the appropriate replacement method, whether it was mean, median, or mode. (like in Figure 2)

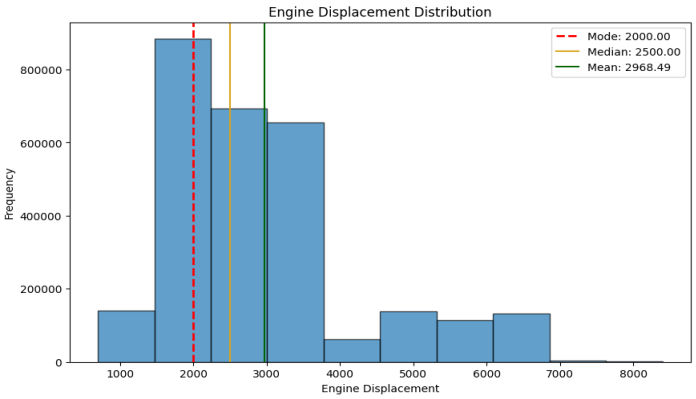


Figure 2: Engine Displacement Distribution

The following actions were taken during row-wise EDA:

1. Missing values in **numerical columns** were replaced based on their distribution:
   a) **Mean:** Used for columns like 'torque', 'fuel_economy', and 'fuel_tank_volume', as their distributions were relatively symmetric.
   b) **Median:** Applied to columns like mileage, length, and engine displacement where the median provided a better central value compared to the mean and mode, particularly in cases of skewed distributions.
   c) **Mode:** This approach was applied to both numerical columns like 'maximum_seating' and 'rating', and categorical columns such as 'transmission' and 'body_type'. Using mode ensured consistency while retaining the most common patterns in the dataset.
2. Addressed missing values in categorical columns by imputing them with the most frequent category (mode) or assigning the value 'Unknown' where applicable. For columns like 'major_options, 'description', 'wheel_system_display', and 'transmission_display', assigning 'Unknown' ensured that no critical data was arbitrarily lost while maintaining uniformity in the dataset. This method helped retain as much information as possible without introducing bias or inconsistencies.
3. Verified the dataset for outliers in numerical columns and applied transformations or scaling where necessary to reduce the impact of extreme values on model performance. The 'mileage' column exhibited significant skewness, with a high concentration near zero and outliers with extremely high values, prompting a *log transformation* to normalize the distribution and reduce the impact of outliers.
4. Processed missing or ambiguous color data by standardizing the 'exterior_color' and 'interior_color' columns. Popular colors were identified using a predefined list, while unmatched or missing values were labeled as "Other." For rows containing multiple detected colors, they were grouped under "Mixed Colors." This approach ensured uniformity and retained meaningful information while handling inconsistencies in the color data effectively.

5.1.3) Summary of EDA

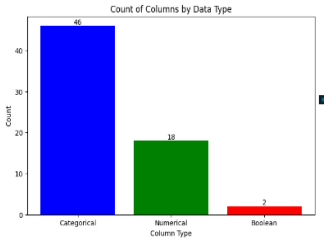


Figure 3: Data Types BEFORE EDA

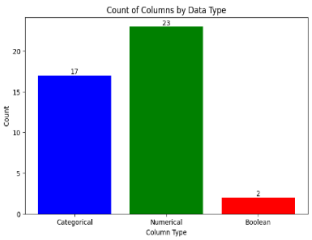


Figure 4: Data Types AFTER EDA

During EDA, the dataset was refined by reducing the number of columns from 66 to 42, with 27 columns deleted and 5 new columns added to improve feature relevance. Additionally, approximately 1/3 of the dataset (1.2 million rows) contained at least one null value. These null values were systematically handled to ensure data completeness and consistency, laying a strong foundation for modeling.

These actions were essential in addressing issues like missing data, irrelevant features, and the need for more meaningful attributes, ultimately setting the stage for building more accurate and effective predictive models.

5.2) Baseline Results

Model	R^2 Score	MAE	MAPE	RMSE	RMSE as % of Mean	Total Hours Trained
Linear Regression	0.775	4284	14.30 %	9170	30.63 %	7 hours
Decision Tree Regressor	0.827	3265	~ 11.00 %	8038	26.85 %	1 hour
Random Forest	0.781	3791	12.66 %	9164	30.61 %	4 hours
XGBoost	0.854	3066	10.24 %	7515	25.10 %	2 hours
GBT Regressor	0.832	3578	11.95 %	7943	26.53 %	~ 4 hours

Table1: Baseline Results

5.2.1) Lessons learnt from the baseline results

1. **Boosting Models' Superiority:** Boosting algorithms like XGBoost and GBT excelled in capturing complex patterns, achieving the best accuracy and lowest errors, making them most suitable for vehicle pricing predictions.
2. **Tree-Based Models' Efficiency:** Tree-based models demonstrated strong performance due to their ability to handle non-linear relationships, with Decision Trees offering a quick but less generalizable solution compared to ensemble methods like Random Forest.
3. **Model Selection Trade-off:** XGBoost emerged as the optimal choice for balancing accuracy and computational efficiency, while simpler models like Linear Regression highlighted their limitations in handling complex datasets.

5.3) Baseline Model Improvement Approaches

Feature Engineering

Feature engineering involved generating new features to capture hidden relationships within the data. Additionally, time-related attributes were extracted to better represent temporal trends and improve prediction accuracy. Domain-specific insights were also included to enrich the dataset with meaningful information that aligns with the problem context.

Hyperparameter Tuning

Hyperparameter tuning was conducted for each of the five models to optimize their performance. For each model, 8-16 different parameters were tested systematically, enabling the identification of the best configurations. This process ensured that the models were fine-tuned to achieve better accuracy and reduced errors.

Bagging

Ensemble techniques, particularly bagging, were implemented to improve prediction performance by combining outputs from multiple models trained on different data samples. Bagging was specifically applied using XGBoost and Decision Tree models, resulting in the best outcomes. This approach successfully reduced the prediction error to less than 4%, demonstrating its effectiveness.

Sub-Sampling

Sub-sampling was used to assess model performance with varying data sizes and feature subsets. For row sampling, models were trained on datasets of 100k, 200k, 300k, and 600k rows to analyse the impact of increased data size. For column sampling, 10 iterations were run for each model, with 70% of the features randomly selected in each iteration. XGBoost and Decision Tree models were primarily used to evaluate the impact of this approach.

5.4) Feature Engineering

Feature engineering played a crucial role in enhancing the dataset by creating new, informative features that improved the model's ability to make accurate predictions.

1. **Interaction Features:**
   Created interaction terms between existing features to capture relationships between variables.
   - Examples: 'hp_x_engine_disp' , 'hp_x_torque'
2. **Date-based Features:**
   Extracted new features from date-related columns to understand time-dependent effects on vehicle price.
   - Examples: 'listed_day', 'listed_month', 'Age'
3. **Domain-Specific Features:**
   Used vehicle-specific knowledge to create useful features that reflect key aspects of a car. These features were designed to capture critical aspects such as a vehicle's resale potential, maintenance cost implications, and luxury positioning.
   • Examples: 'resale_value_score', 'maintenance_cost', 'luxury_score'.

hp_x_engine_disp	hp_x_torque	listed_day	listed_month	listed_year	age	resale_value_score	maintenance_cost	luxury_score
0.06	1.0E-5	5	9	2020	2	29	44	36
0.21	-0.02043	14	7	2020	3	23	38	31
4.74	7.81259	1	7	2020	7	17	42	31
0.06	-0.03274	21	5	2020	0	28	41	30
0.03	0.0	5	8	2020	0	27	42	37

Figure 5: All the feature engineered columns

5.4.1) Detailed Analysis of Engineered Features

1.) Interaction Features:

In this step of the process, I performed a series of actions to identify and create meaningful feature interactions.

1. Calculating Correlations

I calculated the correlation matrix for numerical columns in the dataset to identify highly correlated pairs to find relationships between features, such as 'horsepower' and 'engine_displacement '(0.83 correlation) and 'horsepower' and 'torque' (0.79 correlation).

2. Filtering High Correlation Pairs

I filtered feature pairs with correlation values greater than 0.75, indicating strong relationships. These 2 pairs of columns were identified as potential candidates for interaction terms, because of their unique synergy.

3. Creating Interaction Terms

I created two interaction terms: 'hp_x_engine_disp' (horsepower × engine_displacement) and 'hp_x_torque' (horsepower × torque).

2.) Domain-Specific Feature Engineering:

a) resale_value_score

This gives an overall indication of how well a vehicle is expected to retain its value based on these key factors.

$$resale\_value\_score = (\text{'}mileage\_score\text{'} + \text{'}age\_score\text{'} + \text{'}days\_in\_market\_score\text{'} + \text{'}make\_name\_score\text{'}) / 4$$

b) maintenance_cost

I calculate the total maintenance cost by summing several factors that influence a vehicle's maintenance.

$$maintenance\_cost = ('brand\_cost' + 'age\_cost' + 'horsepower\_cost' + 'transmission\_cost' + 'fuel\_type\_cost' + 'engine\_type\_cost') / 6$$

c) luxury_score

I calculate the overall luxury level of a vehicle by summing various factors that reflect the premium features and exclusivity of the vehicle.

$$luxury\_score = ('brand\_cost' + 'manufacturing\_year\_score' + 'seating\_score' + 'options\_score' + 'transmission\_score') / 5$$

```
# UDF for age score related to resale value
def age_score_udf(age):                 # Very new cars (2 years old or less), highest resale value
    if age <= 2:
        return 10
    elif age <= 5:                      # Relatively new cars (3 to 5 years old), still good resale value
        return 7
    elif age <= 10:                     # Moderately aged cars (6 to 10 years old), average resale value
        return 4
    else:                               # Older cars (more than 10 years old), lower resale value
        return 1

# UDF for days on market score related to resale value (Assumes that vehicles with fewer days on the market have higher demand and higher resale value)
def days_in_market_score_udf(days_in_market):
    if days_in_market < 14:             # Very high demand (quick sale within 2 weeks), highest resale value
        return 10
    elif days_in_market < 35:           # High demand (sold within 1 month), good resale value
        return 7
    elif days_in_market < 82:           # Average demand (sold within 3 months), moderate resale value
        return 5
    elif days_in_market < 215:          # Lower demand (sold within 6-7 months), lower resale value
        return 3
    else:                               # Very low demand (vehicle has been on the market for over 7 months), lowest resale value
        return 1
```

Figure 6: UDF for vehicle age and days on market

As shown in Figure 6, functions like above, assign scores based on predefined ranges, where higher scores indicate attributes associated with better resale value (e.g., newer vehicles or vehicles sold faster).

make_name	log_mileage	age	days_in_market	mileage_score	age_score	days_in_market_score	make_name_score	resale_value_score
Hyundai	11.74	9	25	1	4	7	5	17
Ford	0.69	0	19	10	10	7	7	34
Ford	11.74	14	1	1	1	10	7	19
Cadillac	10.58	1	57	4	10	5	8	27
Chevrolet	0.69	0	192	10	10	3	7	30
Ford	11.26	4	56	1	7	5	7	20
Chevrolet	1.61	0	12	10	10	10	7	37
RAM	8.91	0	248	4	10	1	8	23
Ford	1.61	0	8	10	10	10	7	37
Chevrolet	11.18	4	37	1	7	5	7	20
Nissan	8.91	0	128	4	10	3	5	22
Honda	12.09	8	17	1	4	7	5	17
Chevrolet	8.91	0	6	4	10	10	7	31
Mazda	2.3	0	8	7	10	10	5	32
Ford	7.81	1	58	7	10	5	7	29
Jeep	11.19	3	40	1	7	5	7	20
Dodge	8.91	0	288	4	10	1	5	20
Ford	2.4	0	250	7	10	1	7	25
Toyota	2.2	0	91	10	10	3	5	28
RAM	0.0	0	292	10	10	1	8	29

Figure 7: Feature Scores and Calculations

As shown in Figure 7, The 'resale_value_score' is computed by summing the individua component scores (i.e., age_score) and dividing the total by 4, providing a balanced metric to estimate the vehicle's resale potential. It allows to compare the scores across different rows and understand how each factor contributes to the final score.

5.4.2) Results after Feature Engineering

Model	R^2 Score	MAE	MAPE	RMSE	RMSE as % of Mean	Total Hours Trained
Linear Regression	0.842	4131	13.80 %	7217	24.11 %	17.31 hours
Decision Tree Regressor	0.884	3161	10.50 %	6190	20.68 %	1.25 hours
Random Forest	0.864	3488	11.65 %	6707	22.40 %	~ 8.6 hours
XGBoost	0.918	3018	10.10 %	5268	17.60 %	1.25 hours
GBT Regressor	0.886	3649	12.20 %	6137	20.50 %	11.3 hours

Table 2 : Results after Feature Engineering

Model	Increase in R^2 Score	Decrease in MAE	Decrease in RMSE
Linear Regression	~ 0.07	153	1953
Decision Tree Regressor	~ 0.06	104	1848
Random Forest	~ 0.08	303	2457
XGBoost	~ 0.07	48	2247
GBT Regressor	~ 0.05	-71	1806

Table 3 : Simplified FE result improvement

To better interpret the improvements, the Table 3 summarizes the incremental benefits gained compared to the previous baseline results. It highlights the increase in R^2 Score and the reductions in MAE and RMSE for each model.

Note:

The difference table above reflects the improvements achieved through Feature engineering applied on top of baseline results after EDA.

5.4.3) Observations from the results of Feature Engineering

1. Increase in Model Precision

Adding interaction terms and domain-specific features significantly enhanced model performance across the board. The R^2 scores saw notable increase of 5-8% for most models, while the RMSE values dropped sharply, signifying improved representation of underlying data patterns.

2. RMSE Drop vs. Limited MAE Reduction:

The sharp decline in RMSE (in thousands) compared to a smaller reduction in MAE (in hundreds) highlights the model's improved handling of extreme errors. This can be attributed to feature engineering, which successfully addressed high-impact relationships that previously caused large deviations.

5.4.4) Lessons Learnt from Feature Engineering

1. Role of Domain-Specific Features:

Custom features such as resale_value_score, maintenance_cost, and luxury_score introduced critical vehicle-specific insights, aiding the models in addressing complex patterns and reducing prediction errors, particularly for outliers and edge cases.

2. Capturing Complex Feature Relationships:

Interaction terms like horsepower × engine_displacement allowed the model to uncover hidden relationships in the data, enhancing its ability to predict outcomes that were otherwise overlooked by simpler feature combinations.

5.5) Bagging

I implemented Bagging (Bootstrap Aggregating) as an ensemble method on both XGBoost and Decision Trees to enhance model performance. This technique involved training multiple instances of the models on randomly selected subsets of the data and combining their predictions through averaging.

By introducing randomness in both data sampling and feature selection, Bagging significantly reduced model variance, improved stability, and boosted predictive accuracy. This approach, often referred to as Randomized XGBoost or Bootstrap Aggregating with Decision Trees, took advantage of the individual strengths of these individual models to create a more robust ensemble. The ensemble effectively captured diverse patterns in the dataset, leading to better generalization and improved error metrics compared to standalone models.

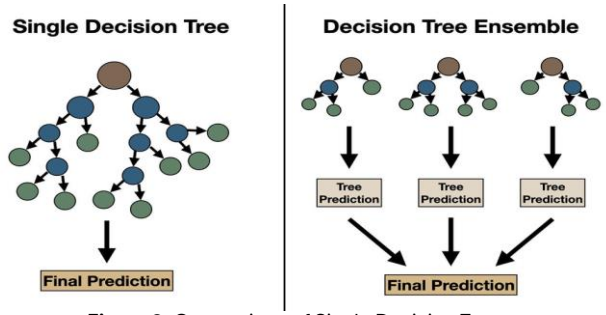


Figure 8: Comparison of Single Decision Tree vs Decision Tree Ensemble

Figure 8 illustrates the core concept of Bagging, comparing a single decision tree with an ensemble of multiple decision trees. While a single tree may be prone to overfitting or capturing noise in the dataset, the ensemble combines predictions from multiple independent trees trained on random subsets of the data. This aggregation reduces variance and provides increased accuracy in the final predictions.

5.5.1) Results of Bagging

Model	R^2 Score	MAE	MAPE	RMSE	RMSE as % of Mean
Baseline (after Feature Engineering)	0.918	3018	10.08 %	5268	17.60 %
Bagging	0.931	**1290**	**4.30 %**	**1820**	**6.08 %**

XGB

Model	R^2 Score	MAE	MAPE	RMSE	RMSE as % of Mean
Baseline (after Feature Engineering)	0.884	3161	10.50 %	6190	20.68 %
Bagging	0.895	**1356**	**4.53 %**	**2293**	**7.66 %**

GBT

Tables 4,5: Results after applying Bagging on XGB, GBT

5.5.2) Observations and Lessons Learnt from Bagging

1. Highest Reduction in MAE and RMSE:

The application of bagging resulted in greatest reductions in both Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), emphasizing its capacity to minimize errors. For XGBoost, the RMSE decreased from 5268 to 1820, and the MAE was reduced from 3018 to 1290, marking the most significant drop in prediction errors. Similarly, for GBT, RMSE dropped from 6190 to 2293, and MAE decreased from 3161 to 1356. These reductions illustrate how bagging mitigates large deviations and outliers, providing more accurate and reliable predictions.

2. Enhanced Model Stability:

Bagging not only improved the metrics but also enhanced the overall stability of the models. By training multiple base models on randomized subsets of data and averaging their predictions, bagging reduced variance and mitigated the risk of overfitting. Each model learned unique patterns from its subset, contributing to a more diverse and generalized ensemble. The averaging mechanism smoothed out anomalies from individual models, creating a balanced output. Additionally, bagging allowed the ensemble to effectively generalize to unseen data, capturing broader trends that a single model might miss. This combination of reduced error metrics and increased reliability underscores the robust performance of bagging in predictive modeling.

5.6) Summary of Approaches

In this section, I implemented a structured workflow starting with EDA, followed by baseline model evaluation, feature engineering, and advanced modeling techniques like bagging. Through EDA, I cleaned and organized the dataset to create a strong foundation. I introduced new features, including interaction terms and domain-specific metrics, which allowed me to capture complex relationships and improve the predictive power of the models. Additionally, by applying bagging, I combined the strengths of multiple models to significantly reduce errors, enhance model stability, and achieve robust predictions.

Transition to Experimentation:

Building on the foundational improvements from EDA, feature engineering, and bagging, I will shift my focus to experimentation, where I explore more advanced techniques to further enhance model performance and gain deeper insights.

6. Experimentation

6.1) Hyper parameter tuning (HPT)

To enhance model performance, I conducted a systematic and detailed hyperparameter tuning process for all five models. Each model's key parameters were carefully examined and adjusted to find the most effective configuration. I explored a range of 8-16 parameters per model, including learning rates, maximum tree depths, regularization terms, and number of iterations. These parameters were varied systematically using techniques like grid search and randomized search to pinpoint the combinations that delivered the best results. This iterative process required running numerous experiments, with each round fine-tuning the parameters based on the model's performance metrics, such as R^2 score, MAE, and RMSE.

Through this rigorous tuning process, I observed significant improvements in both accuracy and error reduction. Hyperparameter tuning not only optimized individual model performance but also provided insights into how different parameters influenced the model's behaviour.

6.1.1) Detailed Analysis on parameters and improvement

1. Decision Trees

- Adjusted Parameters:
    a) minInstancesPerNode: Reduced from 5 to 10.
- Reason for Improvement:
  Increasing minInstancesPerNode ensured that each node contained a minimum amount of data, which prevented small splits and helped reduce overfitting, thereby improving generalization.

2. GBT (Gradient-Boosted Trees)

- Adjusted Parameters: (Introduced these 3 params)
    a) minInstancesperNode: 10
    b) stepSize: 0.1
    c) maxBins: 50
- Reason for Improvement:
  The addition of minInstancesPerNode (set to 10) ensured each node had sufficient data for robust splits, reducing overfitting. Adjusting stepSize to 0.1 enabled controlled, incremental boosting for better convergence. Increasing maxBins to 50 improved split granularities, capturing finer feature details effectively.

3. XGBoost

- Adjusted Parameters:
    a) maxDepth: Increased from 6 to 8
    b) numRound: Increased from 100 to 200
    c) eta: Newly added with a value of 0.1
- Improvement:
  Increasing maxDepth allowed the model to capture more complex feature interactions, improving its ability to handle non-linear relationships. More iterations through numRound refined the boosting process, optimizing predictions over an extended period. The introduction of eta controlled the learning rate, ensuring gradual, stable convergence and reducing the risk of overfitting.

4. Random Forest

- Adjusted Parameters:
    a) numTrees: Increased from 50 to 100.
    b) maxDepth: Increased from 10 to 20.
- Reason for Improvement:
  Increasing numTrees improved ensemble diversity, allowing for better averaging and reducing variance, which helped the model generalize more effectively. Increasing maxDepth enabled each tree to learn more detailed patterns in the dataset, capturing deeper relationships. The higher depth was balanced by the larger forest size to prevent overfitting.

5. Linear Regression

- Adjusted Parameters:
    a) regParam: Introduced with a value of 0.5 (previously no regularization was applied).
    b) elasticNetParam: Introduced with a value of 1.0 (previously no ElasticNet regularization was applied).
    c) maxIter: Increased from 50 to 100.
- Reason for Improvement:
  Adding regParam applied L2 regularization, controlling the magnitude of coefficients and reducing overfitting, especially for features with multicollinearity. Introducing elasticNetParam balanced L1 and L2 regularization, addressing sparsity while effectively maintaining feature weights. Increasing maxIter ensured the model iterated more times, helping it converge on the optimal solution.

6.1.2) Results and observations of HPT

Model	R^2 Score	MAE	MAPE	RMSE	RMSE as % of Mean
Linear Regression	0.844	4097	13.70 %	7204	24.06 %
Decision Tree Regressor	0.889	3232	~ 10.80 %	6577	21.97 %
Random Forest	0.879	3442	11.50 %	6543	21.85 %
XGBoost	0.928	2759	9.40 %	4964	16.60 %
GBT Regressor	0.888	3717	12.40 %	6071	20.30 %

Table 6 : Results after running the models with the BEST PARAMETERS

Model	Increase in R^2 Score	Decrease in MAE	Decrease in RMSE
Linear Regression	0.0021	34	13
Decision Tree Regressor	0.0052	-71	-387
Random Forest	0.0157	46	164
XGBoost	0.0091	259	304
GBT Regressor	0.0025	-68	66

Table 7: Simplified HPT result improvement table

Note:

The difference table in Table 7 reflects the improvements achieved through hyperparameter tuning (HPT) applied on top of the feature-engineered baseline.

6.1.3) Observations from the results of HPT

1. Marginal Improvements in Predictive Accuracy:

Hyperparameter tuning resulted in very less increases in R^2 scores across all models. The changes were minimal, with Random Forest showing the highest gain of 1.5%, and other models like Linear Regression and GBT Regressor barely improving. This shows the most models stagnated.

2. Limited Impact on Error Metrics:

While XGBoost achieved a noticeable reduction in both MAE (259) and RMSE (304), most other models saw only small or even negative changes in error metrics. Decision Tree and GBT Regressor, for example, experienced increases in RMSE and MAE, suggesting that hyperparameter tuning had mixed results in improving predictive precision across the board.

6.1.4) Lesson Learnt from Hyper-parameter tuning

1. Enhanced Stability and Generalization Challenges:

The minimal improvements in performance metrics highlight the limited impact of hyperparameter tuning in enhancing the model's ability to generalize. For some models, such as Decision Tree Regressor, tuning may have introduced overfitting, leading to higher errors. Overall, the results suggest that hyperparameter optimization alone was not sufficient to significantly improve on model's performance.

6.2) Sub Sampling

I implemented both row and column sub sampling.

1.) Row Sub-Sampling:

Row sub-sampling was applied to understand how the size of the dataset impacts model performance. By progressively increasing the number of rows used for training and testing (from 100k to 1 million), the goal was to evaluate the relationship between dataset size and key performance metrics such as R^2, MAE, and RMSE. This approach helped identify the optimal dataset size required for maximizing accuracy while balancing computational efficiency.

1. Small Dataset (100k rows): The models showed relatively low R^2 scores and high error metrics (MAE and RMSE), reflecting underperformance due to insufficient data for learning patterns effectively.
2. Medium Dataset (200k–600k rows): Performance steadily improved as more data was included, with R^2 scores increasing and errors reducing. Increasing to 200k from 100k, saw greatest accuracy improvement. This indicates that larger datasets provide better learning opportunities for the models.
3. Large Dataset (1 million rows): For some models, such as Decision Trees and XGBoost, the R^2 scores slightly decreased when moving from 600k rows to 1 million rows, suggesting that the additional data might not add significant value and could lead to overfitting or diminishing returns.

2.) Column Sub-Sampling:

Column sub-sampling was implemented to test how random subsets of features impact model performance. For each iteration, 70% of the features were randomly selected, and the models were trained and evaluated using these subsets. The goal was to analyse the variability in R^2, MAE, and RMSE between the worst-case and best-case feature combinations.

1. Baseline Comparison: Baseline metrics for XGBoost and Decision Trees were used as reference points to evaluate the impact of feature sub-sampling.
2. Worst Case vs. Best Case: The "Worst Case" scenarios, where sub-sampling led to suboptimal feature selection, resulted in significant increases in MAE and RMSE, demonstrating the importance of meaningful feature selection. On the other hand, the "Best Case" scenarios achieved higher R^2 scores and lower error metrics than the baseline, highlighting the potential for feature sub-sampling to enhance model performance when the right features are chosen.

6.2.1) Results of Sub-Sampling

Size of DF (Training +Testing)	R^2 Score	MAE	RMSE
100k	0.785	3341	8774
200k	0.860	3237	6975
300k	0.874	3746	6674
600k	0.884	3161	6190
1 million	0.865	3144	6794

Decision Trees

Size of DF (Training +Testing)	R^2 Score	MAE	RMSE
100k	0.904	3009	5971
200k	0.907	2938	5509
300k	0.918	3018	5268
600k	0.921	2975	6581

XGB

Size of DF (Training +Testing)	R^2 Score	MAE	RMSE
100k	0.753	3517	9420
200k	0.830	3545	7687
300k	0.846	3594	6981
600k	0.864	3488	6707

Random Forest

Size of DF (Training +Testing)	R^2 Score	MAE	RMSE
100k	0.794	3604	8767
200k	0.856	3638	7073
300k	0.872	3654	6549
600k	0.886	3649	6137

GBT

Size of DF (Training +Testing)	R^2 Score	MAE	RMSE
100k	0.795	4470	8573
200k	0.816	4318	7153
300k	0.836	3916	7457
600k	0.842	4131	7217

Linear Regression

Tables 8-12: Impact of dataset size on Model Performance across Algorithms after Row Sub-Sampling

Baseline Results

Model Name	R^2 Score	MAE	RMSE
XGB	0.918	3018	5268
Decision Trees	0.884	3161	6190

Worst case

Model Name	R^2 Score	MAE	RMSE
XGB	0.894	3300	5990
Decision Trees	0.85	3335	7118

Best case

Model Name	R^2 Score	MAE	RMSE
XGB	0.921	2959	51845
Decision Trees	0.887	3135	6104

Tables 13-15: Comparison of Baseline, Worst-Case, and Best-Case Model performances after Column Sub-Sampling

6.2.2) Lessons Learnt from Sub-Sampling

1.) Row Sub-Sampling:

a) Improved Performance with Larger Data:

All models generally show an improvement in R^2 scores and a reduction in MAE and RMSE as the data size increases, which suggests that larger datasets help capture underlying patterns more effectively, leading to better model accuracy and reduced error rates.

b) Returns at Higher Data Sizes:

While performance improves with increased data size, the gains start to plateau, particularly between 300k and 600k rows. This indicates that after a certain point, additional data yields only marginal improvements, suggesting that the models may have captured most essential patterns already.

2.) Column Sub Sampling:

a) Iteration Results:
Iterative training with 70% of features demonstrated consistently high accuracy (R^2 between 0.894and 0.921 for XGB), indicating that the model performs reliably across different feature subsets.

b) Overall Performance:
The overall performance of the model was strong, with consistent accuracy and error metrics across multiple iterations. This indicates stability in model performance, regardless of feature selection variability. Such stability suggests that the model can generalize well to unseen data.

c) 3. Feature Redundancy:
The 1-2% difference in performance when using 70% of columns compared to 100% suggests that many of my features are likely to contain redundant or correlated information. Some features might have be providing similar information, which means that I can achieve comparable predictive performance without needing all the original columns.

6.3) Transformation of 'Description'

In this method, I used BERT embeddings to transform the text data into meaningful numerical representations, capturing semantic details of the descriptions. I then incorporated these embeddings into the existing DF for model training, aiming to evaluate the added impact of BERT on predictive performance. This allowed me to compare the model's results with and without the enriched description feature.

6.3.1) Implementation Details

To improve the dataset's predictive power, I used BERT embeddings to convert the text in the "description" column into meaningful numerical values that capture the underlying semantic details. First, I loaded a pre-trained BERT model and tokenizer from Hugging Face, which allowed me to tokenize the text and generate embeddings. I created a custom function (get_bert_embeddings) to process the tokenized text through BERT and extract the [CLS] token, which summarizes the entire sentence. This function was then converted into a PySpark UDF, making it easier to apply to the large dataset in the "description" column. The generated embeddings were saved as a new column, bert_embeddings, adding more depth to the dataset. Finally, I trained models like Decision Trees and XGBoost using the updated dataset, comparing the results with and without the original "description" column, to see how the BERT embeddings improved the predictions.

6.3.2) Results of Transforming 'Description'

XGB

Model Name	Dataset Size	R^2 Score	MAE	RMSE
without BERT	100k	0.86	2994	6955
with BERT	100k	0.87	3007	6892

GBT

Model Name	Dataset Size	R^2 Score	MAE	RMSE
without BERT	100k	0.79	3471	8556
with BERT	100k	0.84	3948	7257

Tables 16,17: Performance Comparison of XGB and GBT Models with and without BERT Embeddings

6.3.2) Lessons Learnt from Transforming 'Description'

1. BERT embeddings led to inconsistent model improvements.
2. GBT saw a 4% accuracy boost with BERT, but training/testing time increased to 9 hours (9 X increase).
3. XGBoost showed only a 1% accuracy gain with BERT embeddings.
4. RMSE and MAE did not decrease significantly, suggesting limited performance benefits from BERT embeddings despite the usage of increased resources.

7) Analysis of the Most accurate results

On the results of Bagging with XGB, I identified predictions with an error percentage between **0% and 1%**. I joined them with the original dataset to include all features, and selected relevant columns (price, final prediction, error percentage, and all original features) for further analysis.

The following factors were associated with the least prediction errors and highest model accuracy: (<=1% error)

1. Low mileage, moderate age, and newer model years (2015-2020).
2. Popular body types (SUVs and sedans) and mid-size seating capacities.
3. Mainstream brands (Ford, Chevrolet, Nissan) and
4. common engine types (I4, V6, V8).
5. Gasoline and hybrid fuel types, along with popular colors (white, silver, black).
6. Well-equipped vehicles with advanced features and more premium options. (higher major_options_count)

8) Final comparison

I chose XGBoost as the comparison model because it produced the best results after applying bagging. Therefore, I'm using the XGBoost baseline as the reference point for measuring improvement

Baseline
R2 Score: 0.854
MAE: 3066
MAPE: 10.24 %
RMSE: 7515
RMSE as percentage of Mean: 25.10 %

Final Result
R2 Score: 0.931
MAE: 1290
MAPE: 4.30 %
RMSE: 1820
RMSE as percentage of Mean: 6.08 %

The approaches that gave the best result were

1. Bagging (on XGB and Decision Trees)
2. Feature Engineering
   (after Bagging approach, this showed the most improvement)

Both these approaches provided higher improvements in R^2, MAE (MAPE), and RMSE.

NOTE : In each of the Approaches and Experiments I have listed above, I have included the lessons I learnt from each of them in their respective sections. Apart from that, I will also provide a comprehensive summary of lessons I learnt in this project holistically.

9.) Summary of Lessons Learnt

In this final section, I provide a consolidated overview of the lessons gained from various approaches and experiments undertaken in this project.

1. Tree-based models showed strong performance, Linear Regression struggled with complex patterns, and XGBoost excelled as the top performer with the highest accuracy and lowest errors.
2. Domain-specific features like resale_value_score and interaction terms such as horsepower × engine_displacement were critical in capturing complex relationships and reducing prediction errors, especially for outliers.
3. Hyperparameter Tuning (HPT) had limited impact on improving performance, with minimal gains in metrics. For simpler models like Decision Tree Regressor, tuning occasionally led to overfitting.
4. In the Row Sub-Sampling method, larger datasets improved R^2 scores and reduced error metrics, but performance gains plateaued beyond a certain data size, suggesting diminishing returns after key patterns were captured.
5. Column sub sampling demonstrated consistent model performance with fewer features, highlighting feature redundancy and the potential to reduce dimensionality without significant loss of predictive accuracy.

10.) Appendix

1. **GBT (Gradient Boosted Trees)**:
   A machine learning technique that builds an ensemble of decision trees sequentially. Each tree focuses on correcting errors made by the previous one, optimizing predictive accuracy and minimizing loss functions like RMSE.

2. **XGB (XGBoost)**:
   An advanced implementation of gradient boosting optimized for speed and performance. It supports regularization to prevent overfitting and is widely used for structured data.

3. **HPT (Hyperparameter Tuning)**:
   The process of systematically optimizing a model's hyperparameters to enhance its performance. HPT involves testing different parameter combinations like depth, learning rate, and number of iterations.

4. **FE (Feature Engineering)**:
   The creation, transformation, and selection of input features to improve model learning. Examples include domain-specific features like resale_value_score and interaction terms like hp_x_engine_disp.

5. **KPI (Key Performance Indicator)**:
   Metrics used to evaluate model performance, such as R^2, MAE, RMSE, and MAPE, which collectively indicate the model's success in prediction tasks.

6. **R^2 (R-Squared)**:
   A statistical measure of how well the model explains the variance in the target variable. Higher R^2 values indicate better predictive accuracy.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where:

- y_i: Actual value
- $\hat{y}_i$: Predicted value
- $\bar{y}$: Mean of actual values

7. **MAE (Mean Absolute Error)**:
   A metric that quantifies the average absolute difference between predicted and actual values. It provides an easy-to-interpret measure of prediction accuracy.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- y_i: Actual value
- $\hat{y}_i$: Predicted value
- n: Total number of data points

8. **MAPE (Mean Absolute Percentage Error)**:
   A percentage-based metric that captures the average relative difference between predictions and actual values, useful for comparing across scales.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Where:

- y_i: Actual value
- $\hat{y}_i$: Predicted value
- n: Total number of data points

9. **RMSE (Root Mean Squared Error)**:
   A performance metric that penalizes large prediction errors more heavily than MAE, providing insight into the model's handling of outliers and overall prediction reliability.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where:

- y_i: Actual value
- $\hat{y}_i$: Predicted value
- n: Total number of data points

11.) References

Here are the references I used:

1. S. Han, J. Qu, J. Song, and Z. Liu, "Second-hand car price prediction based on a mixed-weighted regression model," in *2022 7th International Conference on Big Data Analytics (ICBDA)*, pp. 90–95, 2022. doi: 10.1109/ICBDA55095.2022.9760371.
2. R. M. G. Gonçalves, *Text mining techniques for car price prediction*, Master's project, NOVA Information Management School, Universidade Nova de Lisboa, 2021.
3. S. Guo and B. Zhang, "Revolutionizing the used car market: Predicting prices with XGBoost," in *Proceedings of the 4th International Conference on Signal Processing and Machine Learning*, 2024. doi: 10.54254/2755-2721/48/20241349.
4. C. Longania, S. P. Potharaju, and S. Deore, "Price prediction for pre-owned cars using ensemble machine learning techniques," *Recent Trends in Intensive Computing*, vol. 1, no. 1, 2021. doi: 10.3233/APC210194.
5. M. Listiani, *Support vector regression analysis for price prediction in a car leasing application*, Master's thesis, TU Hamburg-Harburg, 2009.
6. L. Li and Z. Ye, "Research on used car price prediction based on stacking model fusion," in *2022 International Conference on Informatics, Networking and Computing (ICINC)*, pp. 86–90, 2022. doi: 10.1109/ICINC58035.2022.00025.