1. Data Processing for Machine Learning

The primary stage of the ML model-building procedure is data preprocessing. It helps us to clean and eliminate unwanted data from our dataset.

1.1 Dataset Overview

For our project, we employed a car features related dataset that contains variables like mileage, registration details, vehicle condition, body style, fuel type, and cost. The dataset consists of 12 attributes and 402,005 rows.

1.2 Data Cleaning

Data cleaning process starts with finding and fixing duplicate or missing data. The following steps were executed:

- The 'public_reference' field has been removed since it is not considered necessary for predicting prices.
- For consistency, the boolean column "crossover_car_and_van" is being converted to an integer type.
- We transformed the 'reg_code' column to a numeric data type in order to preserve numerical consistency.

1.3 Handling Missing Values

We found a few columns in our dataset with a good percentage of null values. The following approaches were used to solve the problem:

- Missing values for the numerical column "mileage" were filled with the corresponding column's mean value.
- To maintain data integrity, values for the categorical columns "standard_color," "body_type," and "fuel_type" were imputed using the most frequent value.
- The dropna() method was used to remove rows from the DataFrame with missing values in the ('reg_code,' 'year_of_registration') columns.

1.4 Outlier Detection and Removal

Outlier values can considerably affect the performance of the ML model. Therefore, it was necessary to identify and remove outlier numbers.

- Firstly, we identify the columns that contain outlier values by employing the z-scores method.
- The columns 'mileage,' 'year_of_registration,' 'price,' and 'crossover_car_and_van' are found with outlier values. Then, we applied the Interquartile Range (IQR) technique to remove outliers from these columns.

The objective of conducting these preprocessing steps is to make the dataset compatible with our ML model.

2. Feature Engineering

Feature extraction (FE) is a technique used to create new features from the original input feature. We have selected a couple of characteristics that could potentially influence car costs and have conducted the FE process to get independent variables for predicting future prices. We used algebraic calculations to conduct the FE transformation process. Compared to feature selection methods, feature extraction is more resistant to overfitting and achieves higher accuracy in classification or regression.

2.1 Feature Engineering Techniques

Creation of New Features:

1. Car Age: To calculate the car's age, we took the current year and subtracted the year it was registered. It is a valuable characteristic for predicting car prices.
2. Average Annual Mileage: A car's average annual mileage is calculated by dividing its total mileage by its age.
3. Squared Attributes: To find possible non-linear connections we squared the "mileage" and "year_of_registration" variables.
4. Collaboration Feature: We multiplied the "year_of_registration" and "mileage" features. This approach can uncover new relationships that may impact car costs.

```
#Feature Engineering
current_year = 2024
df['car_age'] = current_year - df['year_of_registration']

df['avg_annual_mileage'] = df['mileage'] / df['car_age']

df['mileage_squared'] = df['mileage'] ** 2
df['year_of_registration_squared'] = df['year_of_registration'] ** 2

df['mileage_year_interaction'] = df['mileage'] * df['year_of_registration']
```

Figure 1: Feature Engineering

The introduction of FE in our car price prediction dataset has resulted in an increase in the number of attributes from 12 to 16.

2.2 Encoding Categorical Variables:

1. Label Encoding: The attributes "vehicle_condition," "standard_color," "fuel_type," "standard_make," "standard_model," and "body_type" contain categorical values in our dataset. We perform label encoding, which assigns a unique integer label to each category to convert object data type into an integer. This process helps the ML model to understand categorical data.

We added more useful information to our dataset by adding new attributes and encoding categorical factors.

3. Feature Selection and Dimensionality Reduction

Feature selection, often referred to as attribute selection or feature subset selection, is the procedure of choosing a subset of features with substantial or comparable effects on the assessment aim of utilizing all features. It presupposes that the data includes extraneous or superfluous characteristics that could diminish the model's efficacy. FS was initially suggested as a means to enhance the precision of the induced classifier in supervised learning algorithms. Feature selection identifies and chooses the most optimal feature from a given dataset.

3.1 Feature Selection

1. Method: A random forest model with 50 estimators was used to automatically determine the most important attributes based on their contribution to the model's prediction.
2. Rationale: Variable elimination is a helpful technique that enhances our understanding of data, minimizes computational requirements, mitigates the negative impact of high-dimensional data, and improves overall performance.
3. Implementation: We utilized the 'SelectFromModel' technique on the training dataset to automatically choose suitable features according to their importance scores, reducing the dataset dimensionality.

```
# feature selection
from sklearn.feature_selection import SelectFromModel

# SelectFromModel with RandomForestRegressor as base estimator
selector = SelectFromModel(RandomForestRegressor(n_estimators=50, random_state=42))
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)
```

Figure 2: Feature Selection

3.2 Dimensionality Reduction

The process of dimensionality reduction involves converting the high-dimensional presentation of data into lower-dimensional representations. Although not currently employed in our project, approaches like Principal Component Analysis (PCA) could be examined in subsequent phases to improve the model's performance. However, after the feature selection procedure, we found six important characteristics.

Table 1: Feature Importance Score

Features Name	Importance Score	Features Name	Importance Score
mileage	1.89%	year_of_registration	16.03%
reg_code	0.97%	body_type	0.00%
standard_colour	1.03%	crossover_car_and_van	4.61%
standard_make	19.54%	fuel_type	9.58%
standard_model	14.14%	car_age	1.47%
vehicle_condition	0.00%	avg_annual_mileage	1.90%
mileage_squared	13.86%	year_of_registration_squared	1.98%

The predictive impact of each attribute on the target variable is determined by these scores.

4. Model Building

In machine learning, building a model means training several different algorithms to obtain accurate estimates of car prices. Below, we describe the models we used to predict car prices.

4.1 A Linear Model

We used Logistic Regression (LR) as a baseline model to predict car prices.

- **Model:** Linear Regression
- **Hyperparameter Tuning:** None.
- **Training and Evaluation:** After training on the selected features, we evaluated the LR model using cross-validation (CV). The model's prediction accuracy was assessed using performance measures such as R^2 score, mean squared error (MSE), and mean absolute error (MAE).

Model Description

Linear regression (LR) is a statistical method used to establish the association between two continuous variables. The target column represents the predictor or independent variable, whereas the remaining properties are considered as dependent variables. We employ this algorithm to identify the linear correlation. The linear regression model's linear equation is as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_1 x_2 + \ldots + \beta_n x_n$$

In this equation, β_0, β_1, and x are the augmented vectors, while y is the predicted value and β represents the coefficient. The forecast of automobile prices, presented as output values (y), relies on a collection of input data (x). Thus, in this project, the input values (x) consistently represent diverse input attributes, while the output values consistently denote price forecasts, both consistently provided in numeric form. This basic and efficient LR approach makes it easy for us to project car prices.

4.2 Random Forest

This project also used Random Forest (RF) regression to predict car prices. The model's specifications are as follows:

- **Model:** RandomForestRegressor
- **Hyperparameter Tuning:** GridSearchCV was used to optimize hyperparameters like the number of estimators and maximum depth of trees.
- **Training and Evaluation:** CV, MSE, MAS, R^2 score. Feature importance was also examined to determine which features influenced price forecasts most.

Model Description

Random forest regression (RFR) is a technique for ensemble learning that is nonparametric and involves creating multiple standard decision trees during training, resulting in an average prediction based on these individual trees. A decision tree is a hierarchical diagram used for analyzing data, where each internal node represents a test function on an independent variable,

each branch signifies the outcome of the test, and each terminal node indicates a decision. The algorithm evaluates the values of the incoming dataset at each internal node. It identifies a threshold for one predictor variable, which is then used to divide the dataset into two branches in a manner that maximizes the similarity of dependent variable values within each branch. In the RF algorithm, individual decision trees are trained using a randomly selected subset of data with replacement from the original training dataset. This sampling method helps improve the model's resistance to overfitting.

4.3 A Boosted Tree

Another technique for ensemble learning is gradient boosting, which creates a series of weak learners (decision trees) by correcting the errors of the trees that come before it. Additionally, Gradient Boosting (GB) regression was used to provide predictions:

- **Model:** GradientBoostingRegressor
- **Hyperparameter Tuning:** GridSearchCV was used to optimize hyperparameters like the learning rate, number of estimators, and maximum depth of trees.
- **Training and Evaluation:** The Gradient Boosting model was trained on the specified features and assessed using CV. Like RF, feature significance analysis was performed to understand the critical predictors of car prices.

Model Description

Gradient boosted (GB) is an ensemble approach that combines multiple weak prediction models, specifically regression trees, to improve accuracy. GB trees similarly utilize multiple decision trees, but unlike random forests, where trees are trained simultaneously on bootstrap samples of the original data set, GB trees are trained sequentially. Predictive outcomes are produced by enhancing estimations by utilizing a differentiable loss function. It aligns successive trees by considering the cumulative loss from prior trees. In this scenario, the trees are constructed utilizing subsets of automobile characteristics and pricing factors. Every tree contributes partially to the final answer. Tree boosting uses a nonlinear regression approach to enhance the precision of trees. Applying boosting to decision trees results in a reduction in the algorithm's speed. Additionally, boosting diminishes the transparency of the model. However, it improves accuracy when the model needs to learn nonlinear decision bounds. One notable advantage of GB is its ability to optimize a loss function.

4.4 An Average/Voter/Stacker Ensemble

Ensemble methods merge multiple base models to construct a single, strong predictor. We incorporated the best two models for our car price prediction system: RF and GB. We developed the ensemble model employing a VotingRegressor:

- **Model:** VotingRegressor
- **Base Models:** RF and GB
- **Training and Evaluation:** The ensemble model incorporated predictions from RF and GB models. The model's performance was assessed employing metrics such as MSE, MAE, and R^2 score.

Ensemble Model Description

In Scikit-learn, an ensemble learning technique called the VotingRegressor combines many regression models to generate predictions. For the final prediction, it calculates the averages of the estimates made by each base regression model.

Voting Regressor:

The voting regression is a powerful ensemble learning technique that combines the predictions of multiple individual ML models to get a definitive outcome. It utilizes collective intelligence to enhance the overall accuracy of predictions and is wildly successful when multiple models offer varying viewpoints on the topic at hand. There are two types of voting repressors, namely hard voting and soft voting. In hard voting, each classifier in the ensemble provides a vote for a specific class, and the ultimate prediction is determined by the class that obtains the highest number of votes. It is mainly used for classification tasks where the classes are distinct. This project used this approach in order to get the best possible outcomes. Soft voting is a technique where probability estimates are obtained from the base models for each class. The final prediction is then established by calculating the average of these probabilities. The effectiveness of this technique can be improved by using probability projections generated by models, as this considers the level of confidence that each model has in its predictions.

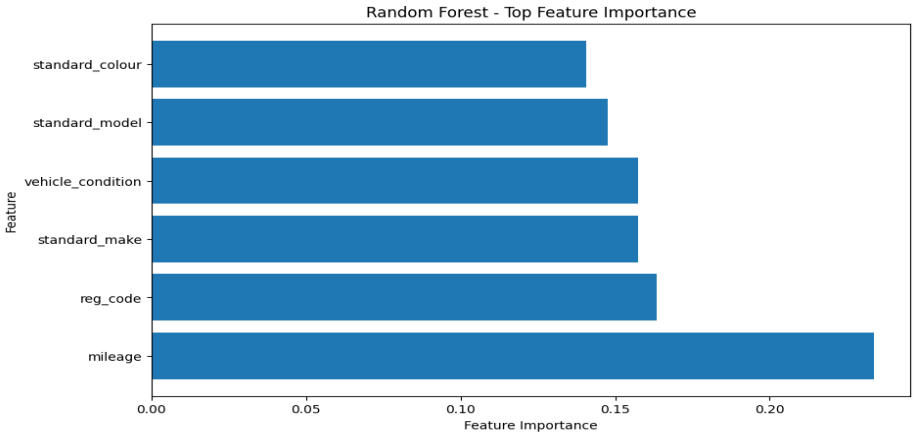

Figure 3: Important Features of RF model

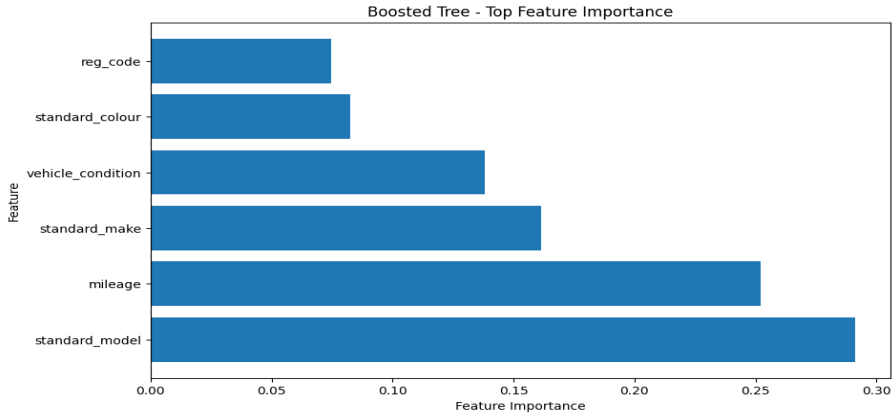

Figure 4: Important Features of GB model

5. Model Evaluation and Analysis

Model evaluation is the procedure of analyzing the performance of any ML model using specific metrics. For our project, we perform a comparative performance analysis for all four models.

5.1 Overall Performance with Cross-Validation

Cross-validation (CV) is an ML method for testing a model on unseen data. It is a process of separating the dataset into folds or subsets, using one as a validation set and the remaining folds for training the model. Table 2 shows the performance of our models using multiple evaluation metrics.

Table 2: Models Performance Evaluation

Model	MSE	MAE	R^2 Score
Linear Regression	28,499,408.41	4,031.07	0.479
Random Forest	5,237,726.07	1,600.07	0.903
Gradient Boosting	12,994,308.97	2,559.83	0.765
Voting Ensemble	7,137,438.94	1,913.39	0.870

Insights:

- Among all models, the RF model performs the best with the lowest CV MSE.
- For the LR model we got higher error results than RF and tree-based models.
- The GB model exhibits competitive performance, although barely lower than RF.
- The voting ensemble model incorporates predictions from RF and GB. Its higher R^2 and smaller MSE indicate a performance improvement over specific models.

5.2 True vs Predicted Plot Analysis

This plot helps us visualize the actual and predicted car prices. We analyzed the actual vs. predicted values using a scatter plot.

Scatter Plot Analysis: Model evaluation is crucial to machine learning because it provides essential information about how well-trained models perform and behave with unseen data. We employed scatter plots to visualize the real versus predicted car prices produced by four models. Various trends appear among the four models when investigating the scatter plots comparing valid and predicted car prices. LR exhibits a linear relationship with a noticeable spread, demonstrating restrictions in capturing complicated relationships. RF and GB models show closer clustering around the diagonal line, indicating enhanced predictive accuracy and consistency compared to LR. Even while both models have tight clustering, GB might have somewhat more significant deviations, suggesting possible sensitivity in certain areas. The voting ensemble model (VEM), incorporating the strengths of RF and GB, shows the tightest clustering, meaning superior predictive accuracy and robustness. The comparative analysis demonstrates how ensemble approaches, particularly the VEM, can improve predictive accuracy in car price prediction tasks.

5.3 Global and Local Explanations with SHAP

We've combined SHAP (Shapley Additive exPlanations) into our project for global and local interpretation objectives. In the context of ML, SHAP values permit us to get the individual contributions of every feature to a model's prediction for a distinctive data point. This becomes particularly useful when the interpretability of the model is crucial. In SHAP calculations, all possible feature arrangements are iterated through, and each feature combination and its impact are evaluated. We shall describe the Random Forest (RF) model's SHAP values below.

SHAP summary plot: Global explanations provide information about the importance of features throughout the whole dataset. We developed a summary plot to illustrate the influence of features on model car price predictions. Figure 4 shows the calculated average magnitude of SHAP values for every feature.

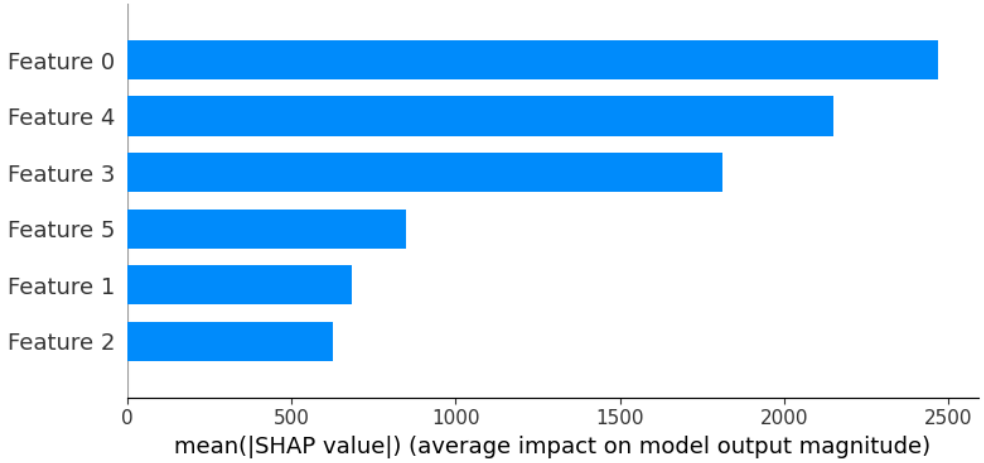

Figure 5: SHAP Global explanations

SHAP force plot: Local explanations SHAP's focus graphs visually depict the influence of characteristics on estimated automobile costs in specific scenarios. According to SHAP values, the horizontal axis represents the impact of the feature on the model's output. The SHAP values are visualized as colored dots, where blue represents a negative influence and red represents a positive influence. The dot sizes reflect an estimate of the impact's magnitude. The magnitude of the features exerts a substantial impact on the output of the model. SHAP values with larger magnitudes have a greater impact than those with smaller magnitudes.

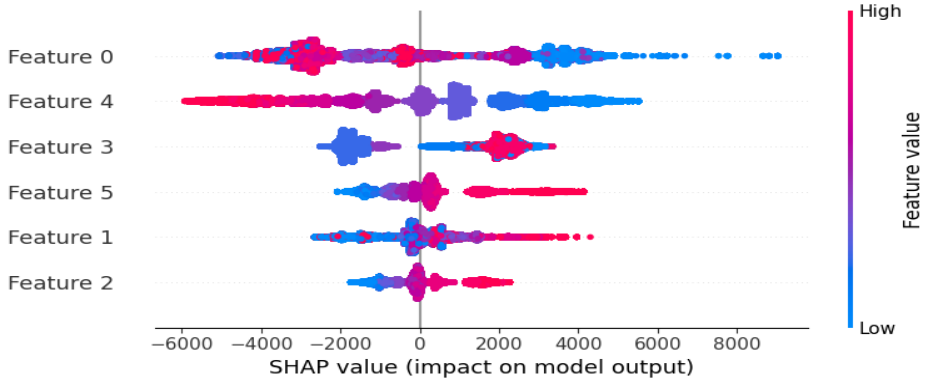

Figure 6: SHAP Local explanations

5.4 Partial Dependency Plots

Partial dependency plots (PDP) illustrate the relationship between the desired response and a particular set of input features while taking into account the values of all other input features (known as the 'complement' features). Finding potential non-linear interactions and learning how different factors affect model predictions depend on these presentations.

Random Forest Partial Dependency Plots:

We employed partial dependency plots to gain insights into the impact of variables on the predicted car pricing. We focused on the following attributes:

1. Mileage: Based on the mileage partial dependency plot, there is an inverse relationship between mileage and car pricing. This implies that as mileage grows, the anticipated price tends to fall. This is consistent with the conventional ideology in the automotive industry, which claims that a lesser number of miles typically correlates with a higher resale value.

2. Mileage Squared: The quadratic relationship between the mileage squared feature and price reduction exhibited a concave downward slope, indicating that the impact of reducing price diminishes as mileage increases. This discovery suggests the potential for a relationship that is not linear.

3. Year of Registration: The partial dependency plot for the year of registration indicated a positive correlation with car pricing, indicating that newer cars typically have higher prices. This outcome aligns with the prevailing understanding of how vehicles lose value over time and people's inclination to buy more recent models.

4. Interaction between Mileage and Year of Registration: Exploring the relationship between mileage and year of registration revealed fascinating findings. The plot demonstrated differing effects of mileage on price, contingent upon the year of enrollment. Notably, the correlation between mileage and price deduction for more recent vehicles appeared less noticeable than for more aged vehicles. This observation suggests that the association between mileage and car pricing is moderated by the year of registration.

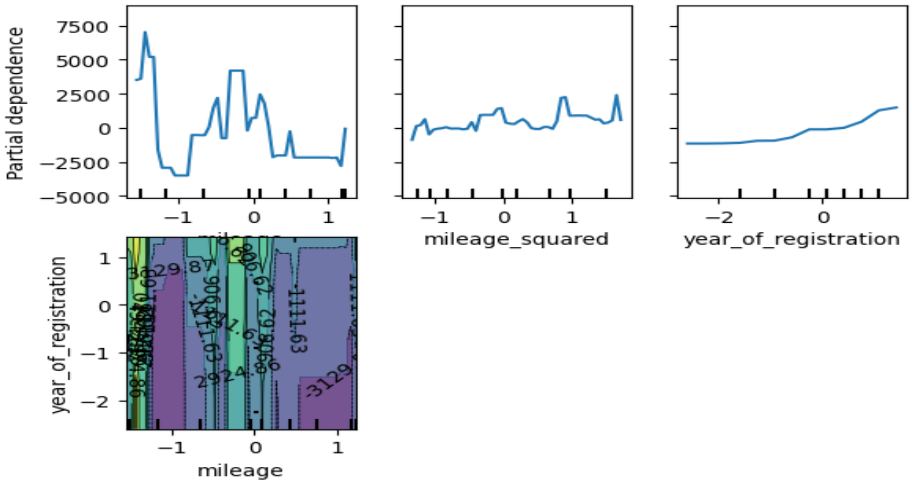

Figure 7: Partial Dependency Plot of RF