Documentation for Use Car Price Prediction Web App

Introduction

This web application, referred to as AutoValue Insight, predicts the selling price of a car based on various input features such as the car's year of manufacture, kilometres driven, fuel type, transmission type, and additional information. The model employed for prediction is a Random Forest Regressor, which was initially trained on a dataset containing data on various cars and their selling prices.

Dataset Overview

There are 8,128 records in all and 13 columns in the data set. Every entry includes information about an automobile, including its name, manufacturing year, and retail price, and other important details. The below table illustrates 13 columns and their respective datatype.

Column Name	Data Type	Description
name	object	Name of the car
year	int64	Year of manufacture
selling_price	int64	Selling price of the car
km_driven	int64	Kilometers driven
fuel	object	Fuel type (Petrol, Diesel, etc.)
seller_type	object	Type of seller (Individual, Dealer, etc.)
transmission	object	Transmission type (Manual/Automatic)
owner	object	Number of previous owners
mileage	object	Mileage in kmpl
engine	object	Engine capacity in CC
max_power	object	Maximum power of the car
torque	object	Torque (Nm) and RPM
seats	float64	Number of seats

Data Processing

In order to guarantee that the dataset is clear and organized for machine learning, data preparation is an essential step. Several crucial actions were made in this project to deal with missing values, clean the data, and format it so the model could use it.

First, the dataset's missing values were fixed. Since the null values represented less than 3% of the total data, they were dropped without significant loss of information or introducing bias into the model. By performing this step, the dataset's overall standard was raised and its comprehensiveness was retained.

In a similar vein, values with "CC" units were eliminated from the engine column, leaving only numerical values. These were transformed into a format that could be used, and entries that were missing or incorrectly formatted were dealt with properly. The maximum power column

was similarly cleaned, with "bhp" units eliminated and the remaining values transformed into floating-point values for simpler model integration.

Data was displayed in the torque column in a variety of formats, including Nm and kg/m values along with corresponding RPM values. The torque measurements in kg/m were converted to Nm to standardize the data. The machine learning model was able to estimate torque and engine speed by extracting and storing the RPM values in different columns.

Feature Engineering

The year feature was utilized as a numerical value to indicate the year of manufacture of the vehicle. This variable gives important details regarding the age and possible depreciation of the car, which are important when figuring out how much to sell it for. Similarly, kilometres driven, which indicates the car's utilization and wear, was directly incorporated as a numerical feature. A vehicle's value may be affected by high kilometres values, which frequently indicate a well-used car.

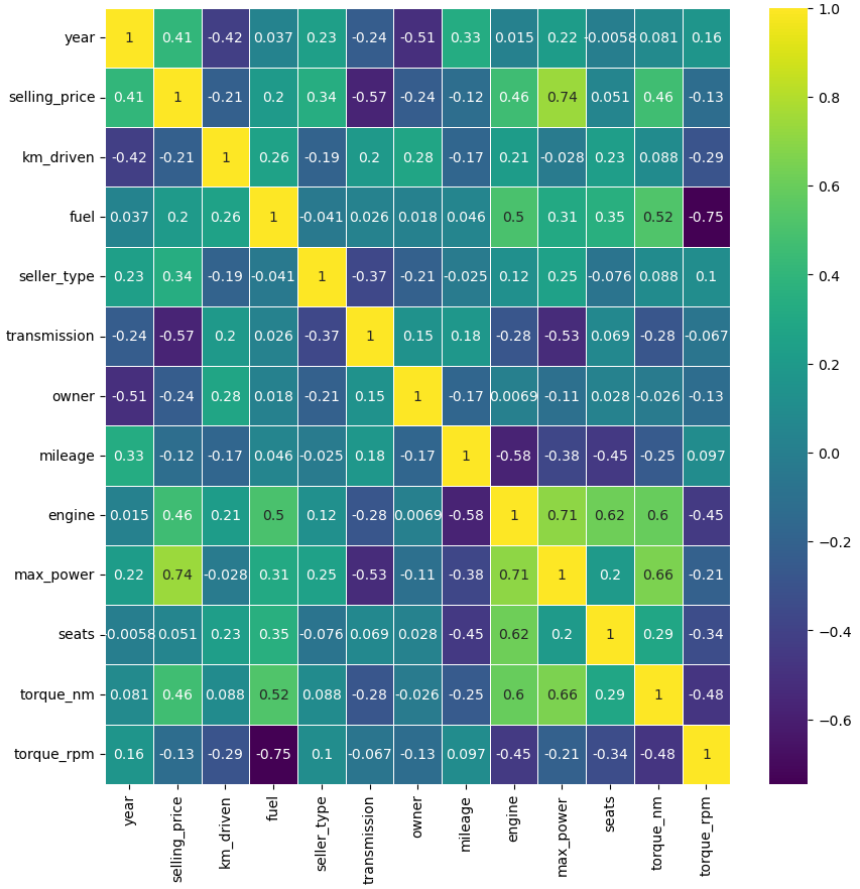
For categorical features like fuel type, seller type, transmission, and ownership history, tailored mappings were created to convert these text-based features into numerical values. To differentiate between gasoline and diesel, for example, the fuel type was mapped into numerical codes, and unsupported categories were assigned a default value. With the use of this mapping, the machine learning model can handle categorical input as numerical, improving its ability to distinguish between different kinds of fuel.

Additionally, the seller type was mapped, classifying the seller as either an individual, a dealer, or a Trustmark dealer. The cost of the car varies depending on each category. Transmission type was also encoded, allowing manual and automatic transmissions to be distinguished. Label encoding was used for this function, giving automatic transmissions one value and manual transmissions another so the model could easily distinguish between the two.

Finally, the owner characteristic, which indicates the number of prior owners, was also encoded. The number of previous owners of the vehicle is indicated by this attribute, and larger values are typically associated with lower selling prices because of increased wear or less appeal. The machine learning model might better comprehend and take advantage of the links between the attributes of the vehicle and its pricing by turning these categorical factors into numerical values, which would result in more precise forecasts.

Exploratory Data analysis
Correlation Analysis
A correlation matrix was created in order to comprehend the connections between the attributes and the target variable (selling_price). The correlation matrix offers a clear image of the relationships between the independent variables and, more crucially, the target variable. These correlations can be seen in corelation plot below.
Among the correlation analysis's main conclusions are:

- The maximum horsepower (bhp) and selling price have the strongest positive association (0.74). This implies that cars with more horsepower are usually worth more on the market.
- Additionally, there is a positive link between engine (CC) and torque (Nm) and the selling price (0.46 for each), suggesting that higher torque and more powerful engines translate into better resale values.
- The number of owners (-0.24) and transmission type (-0.57) exhibit inverse relationships with selling price. This suggests that vehicles with more previous owners and automatic transmissions typically have lower resale prices.
- As would be expected, there is a moderately negative connection (-0.21) between kilometres driven and the selling price, since cars with higher mileage typically lose value.



It is crucial to examine more closely at the relationships between the independent variables themselves after performing correlation analysis to comprehend the relationships between certain features and the target variable. Multicollinearity analysis becomes important at this point. Correlation shows the relationships between specific features and the target, but it misses any overlaps between predictor variables. When two or more predictors have a strong correlation with one another, this phenomenon is known as multicollinearity, and it can have a detrimental effect on the effectiveness of machine learning models. After the correlation study, multicollinearity must be evaluated to make sure the model understands each feature separately and prevents distortions in prediction.

Multicollinearity Analysis

Multicollinearity occurs when two or more independent variables in a regression model are highly correlated, meaning they provide redundant data about the model's behaviour. To detect multicollinearity, we calculated the Variance Inflation Factor (VIF) for each feature. Significant multicollinearity is often thought to be indicated by a VIF value more than 5, which can skew the model's predictions and produce erroneous coefficient values. We chose to leave the features in the model without making any additional changes because none of them showed significant multicollinearity. By doing this, it is ensured that substantial distortions won't be introduced and the model can still accurately represent the relationships between these features and the target variable. These multicollinearity values can be seen below.

	Feature	VIF
1	year	2.179468
2	selling_price	2.991716
3	km_driven	1.436369
4	fuel	3.272005
5	seller_type	1.242397
6	transmission	1.749648
7	owner	1.372876
8	mileage	2.656740
9	engine	5.261089
10	max_power	4.866703
11	seats	2.233023
12	torque_nm	2.318521
13	torque_rpm	2.677334

The dataset was prepared for model training by splitting the data into features (X) and target variable (y). In this instance, the car's selling price is the target variable, and the remaining columns are features for forecasting. By normalizing the data and guaranteeing that every feature has the same scale, standard scaling was used to enhance the performance of machine learning models. This improves the performance of many machine learning algorithms, particularly those that depend on computations at a distance. Several machine learning models were tested, including:

- Linear Regression
- Random Forest Regressor
- XGBoost Regressor
- Support Vector Regressor (SVR)
- Gradient Boosting Regressor
- Lasso Regression
- Ridge Regression
- K-Nearest Neighbors (KNN) Regressor

Each model was trained on the scaled training data, and predictions were made on the test set. Three important measures were used for evaluating each model's performance:

- Mean Squared Error (MSE): Measures the average squared difference among actual and predicted values.
- R-Squared (R^2): Represents the percentage of the variance in the target variable that the model explains.
- Mean Absolute Error (MAE): Provides the average amount of the prediction errors.

We were able to assess the predictive capabilities of the models through this evaluation procedure and determine which model would work best when used in the web application.

Results and analysis

Following the training of multiple machine learning models, Mean Absolute Error (MAE), R-Squared (R^2), and Mean Squared Error (MSE) were used to evaluate each model's performance. Based on test data, these figures show how successfully each model anticipated the selling price of cars.

The Linear Regression model produced an MSE of 216 billion, a R^2 score of 0.659, and an MAE of 277,478, indicating although the model could explain around 66% of the variance in the data, its prediction error was relatively high. Conversely, the Random Forest model yielded far superior results, with an MAE of 71,572, a R^2 of 0.964, and an MSE of 22.68 billion. This indicates that the Random Forest model was the best-performing model overall, explaining more than 96% of the variance and producing accurate predictions with minimal variance.

Additionally showing impressive results were XGBoost and Gradient Boosting, with R2 values of 0.951 and 0.953, respectively. Their lower rankings were a result of their MAE values (94,315 and 91,427) being somewhat higher than Random Forest. With an R2 of 0.915 and an MAE of 102,592, the K-Nearest Neighbours (KNN) model did not perform as well as the ensemble-based models. Support Vector Regressor (SVR) on the other hand, did not perform well for this task; its MSE was greater than 673 billion, its R2 was negative, and its MAE was 377,515. With MSEs of roughly 216 billion and R2 values of 0.659, the Lasso and Ridge regression models both performed comparably to linear regression.

The Random Forest Regressor was the final one to be deployed out of all the models that were examined because of its exceptional performance, which included both high accuracy and low error rates. This model will be incorporated into the web service to provide real-time automobile price predictions, as it is well-suited to forecast car prices based on the provided dataset.

Apart from assessing the models on the test dataset, the R^2 measure was employed to analyse the Random Forest Regressor for accuracy in both training and testing. This makes that there is no overfitting and evaluates the model's capacity for generalization.

During training, the Random Forest model was able to account for 99.53% of the variation in the target variable, selling price, with an R2 score of 0.9953 on the training set. With an R2 score of 0.9642 for the test data, the model was able to account for 96.42% of the variation in the unseen data. The Random Forest model was chosen as the best model for deployment because of its strong R2 scores, which show how well it captures the relationship between the input variables and the car's selling price.

Web Development
To make the car price prediction model obtainable and user-friendly, a web application was developed using Flask, a lightweight web framework in Python. Allowing consumers to enter automobile details and receive an estimated selling price based on the taught machine learning model was the main objective.

Front-end
The frontend of the web application was developed with HTML, CSS, and Bootstrap to create a clean and responsive user interface. The purpose of the form was to gather input features pertaining to cars, such as:

- **Year of Manufacture:** The year the car was manufactured.
- **Kilometres Driven:** The number of kilometres the car has been driven.
- **Fuel Type:** There are drop-down menus for electric, diesel, CNG, LPG, and gasoline.
- **Seller Type:** Select from the Individual, Dealer, or Trustmark Dealer drop-down menus.
- **Transmission Type:** Select between an Automatic or Manual gearbox.
- **Number of past owners**: Dropdown for 0 to 4 owners indicates the number of prior owners.
- **Mileage (kmpl):** The vehicle's mileage.
- **Engine Size (in CC):** The engine's capacity.
- **Max Power:** The highest power the vehicle can produce, measured in horsepower.
- **Torque:** Nm and RPM of the vehicle's torque.

Back-end

Flask was used in the backend development process to manage the following tasks:

- Routing: The routing was done using Flask. The input values are taken, pre-processed, and sent to the trained model via a POST request that the form makes to the /predict route.
- Prediction: Based on the input features, predictions are made using the pre-trained Random Forest Regressor model.
- Display of the result: Following prediction, the outcome is transmitted back to the frontend for user viewing.

Web app screenshots