An Analysis and Prediction of Used Car Prices

1 Introduction

In this report, I set out to predict the prices of used cars, a common real-world problem with significant implications for both consumers and businesses. For this purpose, I used a dataset from the public "CarDekho" repository on Kaggle. The dataset contains a comprehensive collection of used car listings, with each data point representing a single car sold. Since the dataset has a continuous numerical value—the car's price—corresponding to each listing, the machine learning task was a supervised regression problem.

I began by formalizing the problem and then proceeded with the data preparation, model selection, and validation methods I used.

2 Problem Formulation

The project's objective was to predict the price of a used car. To do this, I trained a model that could estimate a continuous numerical value—the car's price—by learning from a dataset of historical sales.

- **Data Points:** The dataset contained 4340 instances, with each corresponding to a unique used car listing.
- **Features:** The initial features included name, year, selling_price, km_driven, fuel, seller_type, transmission, and owner.
- **Labels:** The dependent variable was the selling_price of the car, representing the final selling price in Indian Rupees (in Lakhs).

The dataset was sourced from the public CarDekho dataset, available for download on Kaggle.[1]

3 Methods

3.1 Dataset and Preprocessing

The dataset was found to be clean, with no missing values, but required preprocessing to be suitable for a machine learning model.

- **Feature Engineering:** To make the model's predictive power even stronger, I created two new features. I converted the original year column into car_age to directly capture the effect of depreciation. I also simplified the name column by just extracting the car's brand, as I felt the full model name was too specific for the task at hand.
- **Data Transformation:** I noticed that the selling_price had a skewed distribution, which can make things tricky for a model. To handle this, I used a standard logarithmic transformation to normalize the data. This is a best practice for dealing with large value ranges and outliers, and it helped ensure that the model's prediction errors were treated more uniformly across the dataset.
- **Categorical Encoding:** The remaining categorical features (fuel, seller_type, transmission, owner, and brand) were converted to a numerical format using one-hot encoding.

3.2 Exploratory Data Visualization

To better understand the relationships between the features and the target variable and to validate the preprocessing steps, a series of plots was generated. As mentioned in the Data Transformation section, the selling price had a heavily skewed distribution, which was clearly visible in the data.

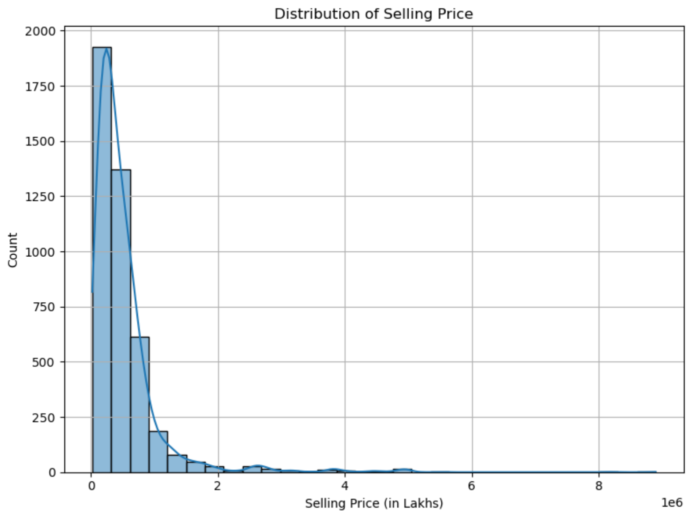
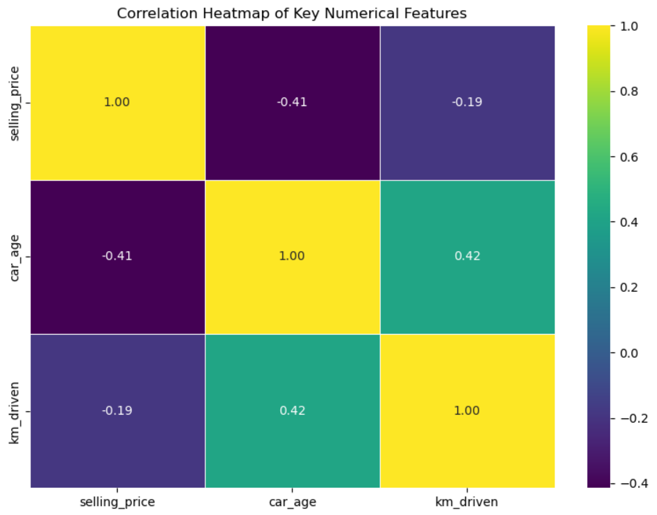


Figure 1: Selling price distribution

To explore the relationships between features, I created a focused correlation heatmap. It showed that car_age and km_driven were inversely correlated with the selling price, which validated using these features in the model. The scatter plots offered a closer look at these specific trends. A bar chart of average prices by brand was also created to confirm a strong relationship between brand and price, supporting my decision to engineer the brand feature.



	selling_price	car_age	km_driven
selling_price	1.00	-0.41	-0.19
car_age	-0.41	1.00	0.42
km_driven	-0.19	0.42	1.00

Figure 2: Numerical correlation heatmap

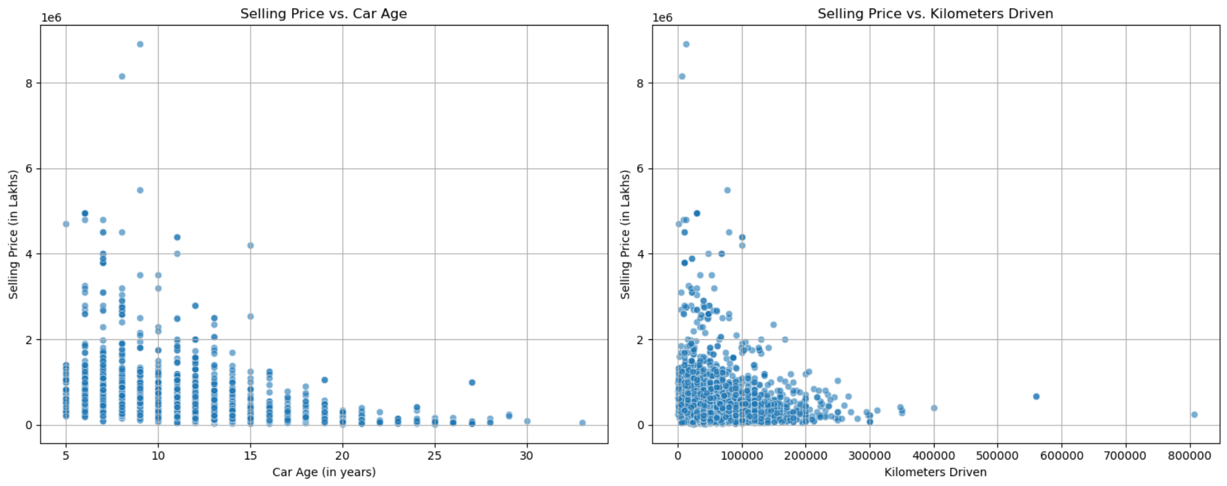


Figure 3: Feature scatter plots

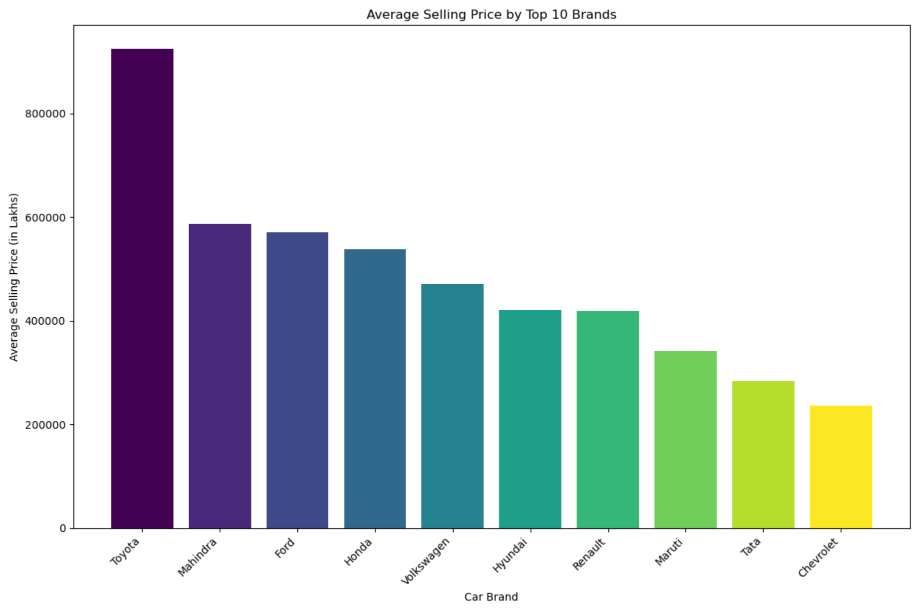


Figure 4: Average price by brand

3.3 Model and Hypothesis Space

A Random Forest Regressor was chosen as the primary machine learning model. As an ensemble method, it builds multiple decision trees and averages their predictions to improve accuracy and control for overfitting. This made it an ideal fit for handling the mix of numerical and one-hot-encoded categorical features in this dataset.

The choice of an ensemble model was motivated by its superior performance compared to simpler models like linear regression when dealing with non-linear relationships, which are common in real-world pricing data.

3.4 Loss Function

For the loss function, I chose to use Mean Squared Error (MSE). It is a standard approach for regression tasks because it calculates the average of the squared differences between the predicted and actual values. This method was particularly useful as it heavily penalizes larger errors, which helped encourage the model to avoid making any significant mistakes in its predictions.

3.5 Model Validation

To ensure the model's performance was not a chance outcome of a single data split, a dual-evaluation strategy was employed. This approach provided a more robust and unbiased assessment of the model's true predictive capability.

- **Single Train-Test Split:** I began by partitioning the dataset into a training set (80% of the data) and a test set (20% of the data). I used this initial randomized split as a baseline to quickly assess the model's performance.
- **5-Fold Cross-Validation:** To get a more rigorous and reliable measure of performance, I performed a robust assessment using 5-fold cross-validation. This method involved dividing the entire dataset into five distinct sections, or folds. I trained and validated the model five separate times. In each iteration, I used one unique fold as the validation set while training on the remaining four. By averaging the performance metrics across all five folds, I obtained a more reliable measure of the model's generalization capability, which helped prevent any optimistic performance estimates that might arise from a single, favorable split.

4 References

[1] Birla, N. (2020). Vehicle Dataset from CarDekho. Kaggle. [https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho]

Appendix: Dataset's Features and Label

Variable name	Role	Type	Description	Units
selling_price	Target, Label	Continuous	The final selling price of the car.	Lakhs
name	Feature	Categorical	The full name and model of the car.	
year	Feature	Integer	The manufacturing year of the car.	Year
km_driven	Feature	Continuous	The total kilometers the car has been driven.	km
fuel	Feature	Categorical	The type of fuel the car uses.	
seller_type	Feature	Categorical	The type of seller (e.g., Individual, Dealer).	
transmission	Feature	Categorical	The type of transmission (e.g., Manual, Automatic).	
owner	Feature	Categorical	How many previous owners the car has had.	

Stage1_Code

September 19, 2025

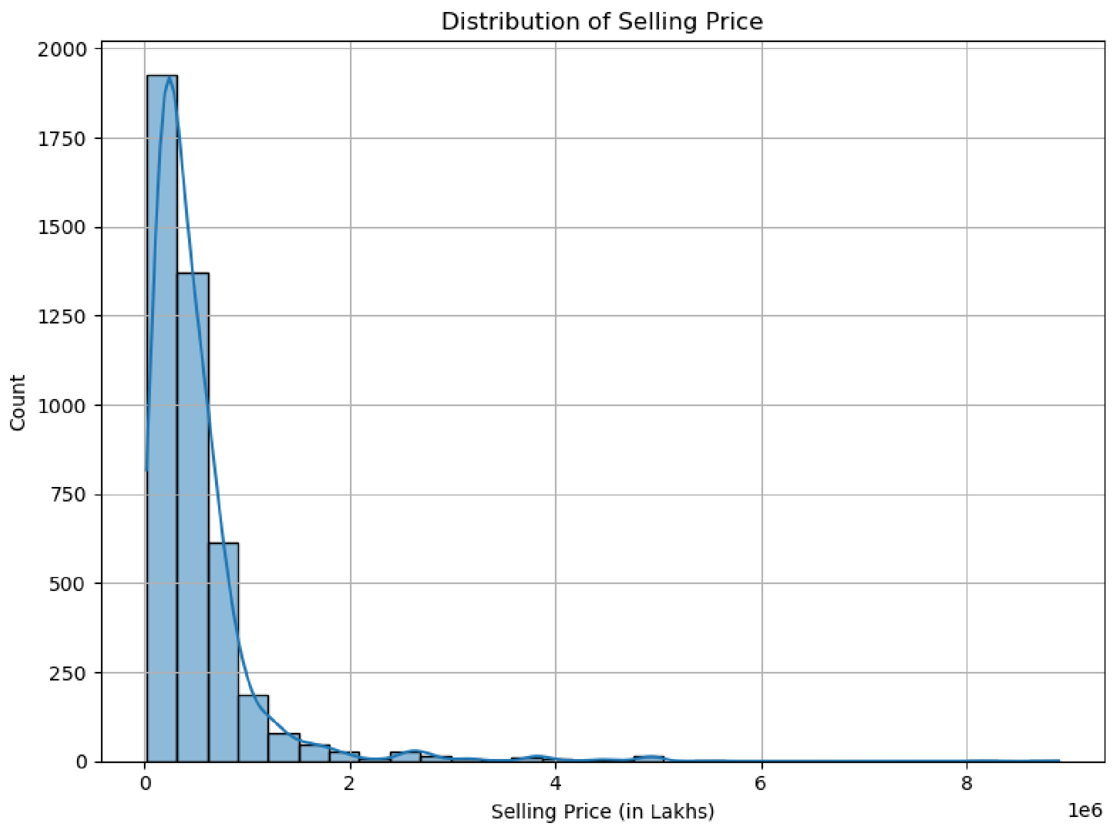
[11]:
```python
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import mean_squared_error
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

# Data Loading
try:
    file_name = 'CAR DETAILS FROM CAR DEKHO.csv'
    df = pd.read_csv(file_name)
except FileNotFoundError:
    print(f"Error: The file '{file_name}' was not found.")
    print("Please make sure the file is in the same directory as this script.")
    exit()

# Dataset Verification
print("Verifying Dataset Against Report Claims")
expected_rows = 4340
expected_cols = 8
if df.shape[0] == expected_rows and df.shape[1] == expected_cols:
    print(f"Verification: Dataset dimensions ({df.shape[0]} rows, {df.shape[1]}␣
 ↪columns) match the expected dimensions.")
else:
    print(f"Warning: Dataset dimensions ({df.shape[0]} rows, {df.shape[1]}␣
 ↪columns) do NOT match the expected dimensions.")

if df.isnull().sum().sum() == 0:
    print("Verification: The dataset has no missing values.")
```

```
else:
    print(f"Warning: The dataset contains missing values. Total found: {df.
 ↪isnull().sum().sum()}")

# Feature Engineering
current_year = datetime.now().year
df['car_age'] = current_year - df['year']
df['brand'] = df['name'].apply(lambda x: x.split(' ')[0])

# Exploratory Data Visualization
# Figure 1: Selling price distribution
plt.figure(figsize=(8, 6))
sns.histplot(df['selling_price'], kde=True, bins=30)
plt.title('Distribution of Selling Price')
plt.xlabel('Selling Price (in Lakhs)')
plt.ylabel('Count')
plt.grid(True)
plt.tight_layout()
plt.savefig('selling_price_distribution.png')
print("Plot saved as selling_price_distribution.png")
plt.show()

# Figure 2: Numerical correlation heatmap
numerical_features_df = df[['selling_price', 'car_age', 'km_driven']]
corr_matrix_numerical = numerical_features_df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix_numerical, annot=True, cmap='viridis', fmt=".2f",␣
 ↪linewidths=.5)
plt.title('Correlation Heatmap of Key Numerical Features')
plt.tight_layout()
plt.savefig('numerical_correlation_heatmap.png')
print("Plot saved as numerical_correlation_heatmap.png")
plt.show()

# Figure 3: Feature scatter plots
fig, axes = plt.subplots(1, 2, figsize=(15, 6))
sns.scatterplot(data=df, x='car_age', y='selling_price', ax=axes[0], alpha=0.6)
axes[0].set_title('Selling Price vs. Car Age')
axes[0].set_xlabel('Car Age (in years)')
axes[0].set_ylabel('Selling Price (in Lakhs)')
axes[0].grid(True)
sns.scatterplot(data=df, x='km_driven', y='selling_price', ax=axes[1], alpha=0.
 ↪6)
axes[1].set_title('Selling Price vs. Kilometers Driven')
axes[1].set_xlabel('Kilometers Driven')
axes[1].set_ylabel('Selling Price (in Lakhs)')
axes[1].grid(True)
```

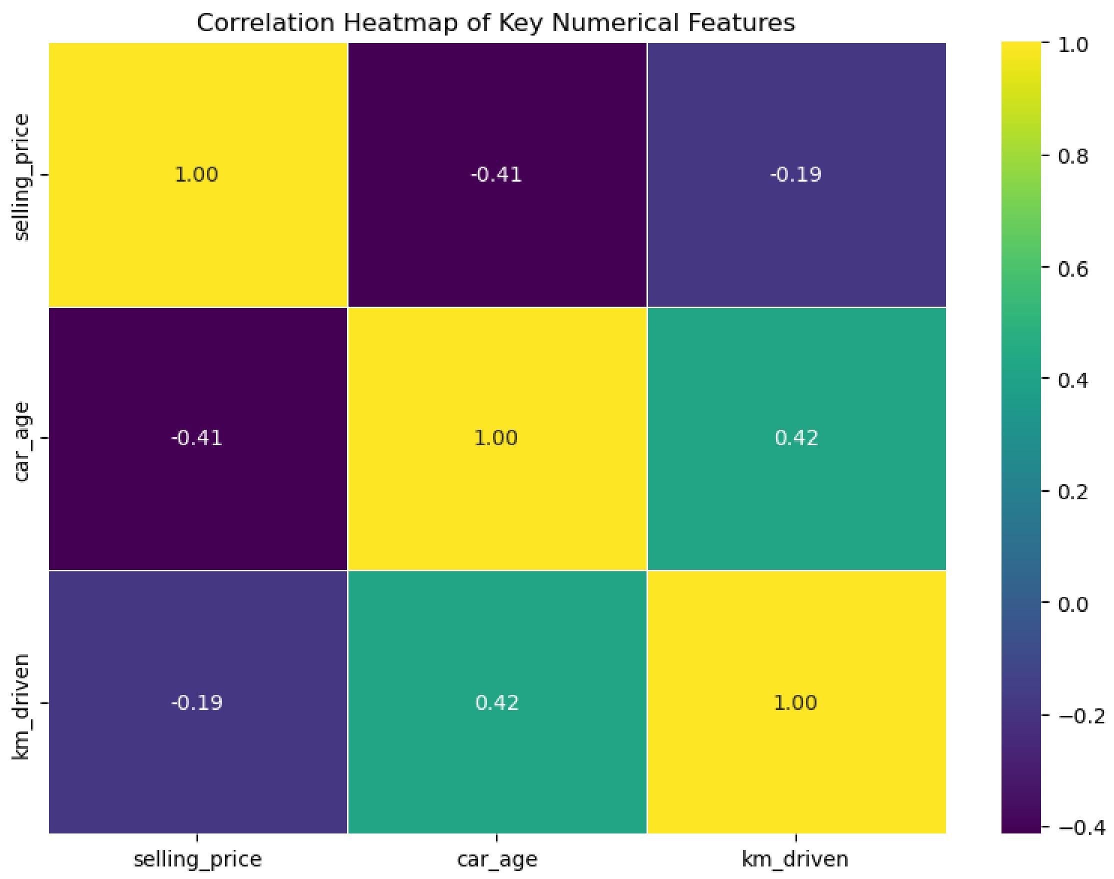
```python
plt.tight_layout()
plt.savefig('feature_scatter_plots.png')
print("Plot saved as feature_scatter_plots.png")
plt.show()

# Figure 4: Average price by brand
brand_counts = df['brand'].value_counts()
top_10_brands = brand_counts.head(10).index
top_brands_df = df[df['brand'].isin(top_10_brands)]
brand_prices = top_brands_df.groupby('brand')['selling_price'].mean().
 ↪sort_values(ascending=False).reset_index()
brand_prices.columns = ['brand', 'selling_price']
plt.figure(figsize=(12, 8))
plt.bar(x=brand_prices['brand'], height=brand_prices['selling_price'],␣
 ↪color=plt.cm.viridis(np.linspace(0, 1, len(brand_prices))))
plt.title('Average Selling Price by Top 10 Brands')
plt.xlabel('Car Brand')
plt.ylabel('Average Selling Price (in Lakhs)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig('average_price_by_brand.png')
print("Plot saved as average_price_by_brand.png")
plt.show()
plt.close('all')

# Data Transformation
df['selling_price_log'] = np.log1p(df['selling_price'])
df.drop(['year', 'name', 'selling_price'], axis=1, inplace=True)

X = df.drop('selling_price_log', axis=1)
y = df['selling_price_log']

# Data Preprocessing
numerical_features = ['km_driven', 'car_age']
categorical_features = ['fuel', 'seller_type', 'transmission', 'owner', 'brand']
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ],
    remainder='passthrough'
)

# Model Selection
model = RandomForestRegressor(random_state=42)

# Single Train-Test Split
```

```
print("\nEvaluating model with a single train-test split")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

pipeline_split = Pipeline(steps=[('preprocessor', preprocessor),
                                 ('regressor', model)])
pipeline_split.fit(X_train, y_train)

y_pred_split = pipeline_split.predict(X_test)
mse_split = mean_squared_error(y_test, y_pred_split)
print(f"Mean Squared Error (MSE) from single split: {mse_split:.4f}")

# 5-Fold Cross-Validation
print("\nEvaluating model with 5-fold cross-validation")
cv = KFold(n_splits=5, shuffle=True, random_state=42)
pipeline_cv = Pipeline(steps=[('preprocessor', preprocessor),
                              ('regressor', model)])

cv_scores = -1 * cross_val_score(pipeline_cv, X, y, cv=cv,␣
 ↪scoring='neg_mean_squared_error')

print("MSE scores for each fold:", np.round(cv_scores, 4))
print(f"Average MSE across 5 folds: {np.mean(cv_scores):.4f}")
print(f"Standard deviation of MSE across 5 folds: {np.std(cv_scores):.4f}")
print("-" * 50)
```

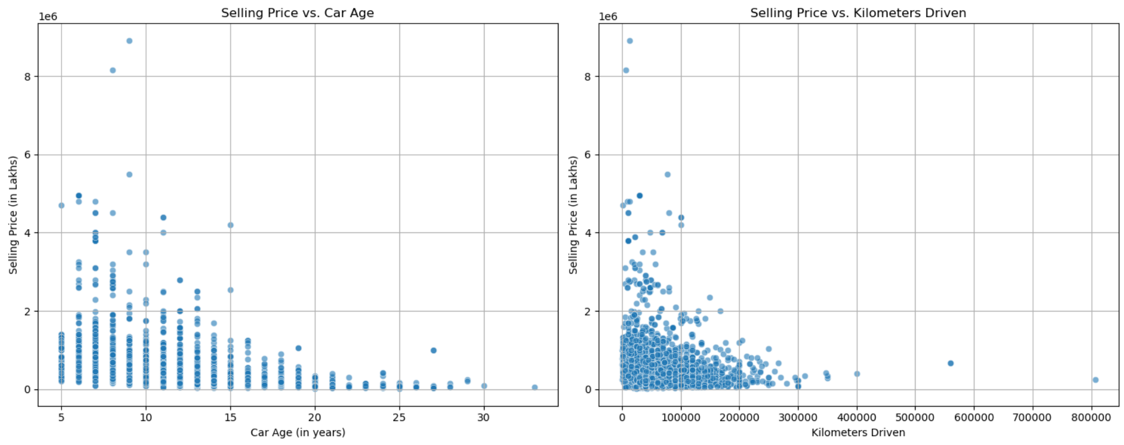
```
Verifying Dataset Against Report Claims
Verification: Dataset dimensions (4340 rows, 8 columns) match the expected
dimensions.
Verification: The dataset has no missing values.
Plot saved as selling_price_distribution.png
```



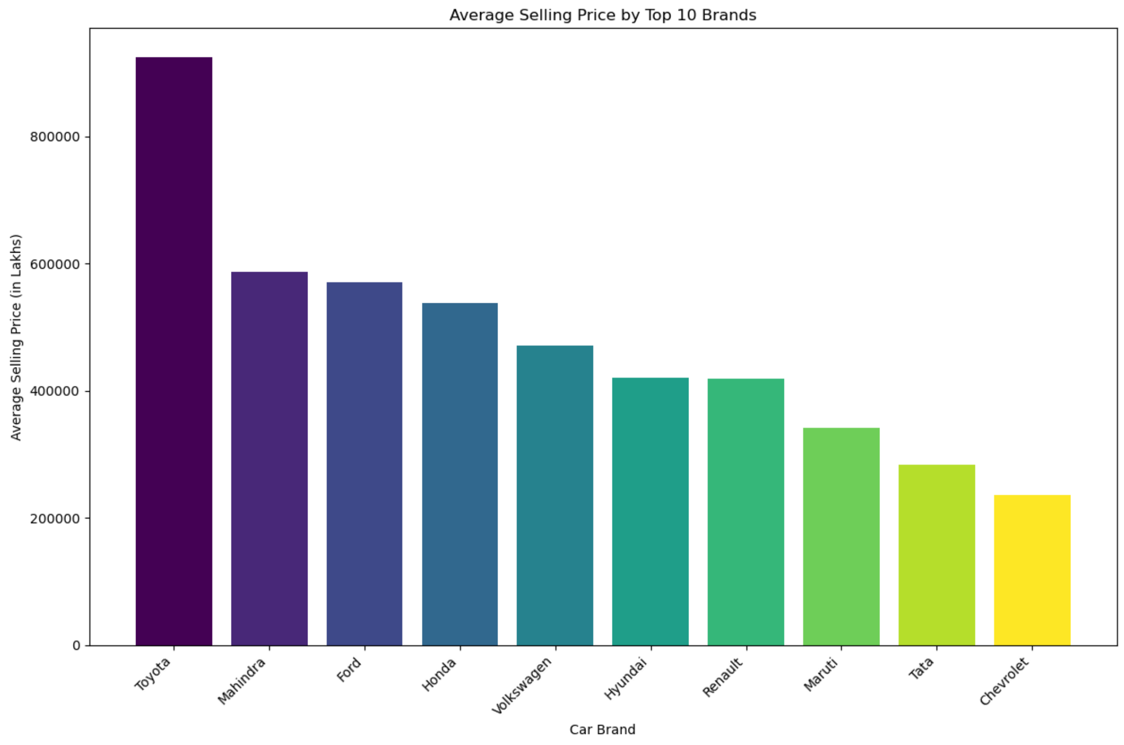
```
Plot saved as numerical_correlation_heatmap.png
```



```
Plot saved as feature_scatter_plots.png
```



```
Plot saved as average_price_by_brand.png
```



```
Evaluating model with a single train-test split
Mean Squared Error (MSE) from single split: 0.1288

Evaluating model with 5-fold cross-validation
MSE scores for each fold: [0.1283 0.1306 0.1394 0.1293 0.1338]
Average MSE across 5 folds: 0.1323
Standard deviation of MSE across 5 folds: 0.0040
--------------------------------------------------
```

```
[ ]:
```