**Visualization and Analysis of Car Dekho Dataset**

**Introduction**

This report aims to analyse the Car Dekho dataset by performing a series of pre-processing steps, followed by visualizations to understand the relationships between various features and the target variable, Numeric Price. The analysis is divided into the following steps:

1. **Pre-processing** the dataset
2. **Encoding** categorical features and **normalizing** numeric features
3. **Correlation Heatmap** for visualizing relationships between numeric features and Numeric Price
4. **Box plots** to visualize patterns between categorical features and Numeric Price

**Step 1: Data Pre-processing**

**Changing Data Formats**

Some columns might have been in inappropriate formats (e.g., Seats, Price, etc. as strings or numeric columns as objects). These columns were converted to their appropriate data types for better analysis.

**Handling Missing Values**

The dataset was first checked for missing values, which could potentially affect the performance of the machine learning models. The missing values were handled as follows:

* **Categorical features**: Imputed with the most frequent value (mode imputation).
* **Numeric features**: Mostly imputed with the mode but for some of the column imputed with median and mean as well. For example, Car Price column was imputed by mean.

**Step 2: Encoding Categorical Features and Normalizing Numeric Features**

**Encoding Categorical Features**

Categorical features such as Fuel Type, City, Transmission, etc. were one-hot encoded to transform them into a numerical format suitable for machine learning algorithms. I choose **one-hot** encoding because the categorical features were **nominal**, means the categories don’t have any natural order.

# One-hot encoding categorical columns

df\_encoded = pd.get\_dummies(df, columns=['FuelType', 'City', 'Transmission', 'BodyType'], drop\_first=True)

**Visualization of DatasetA graph of a person in a suit

Description automatically generated**

A graph with numbers and a line

Description automatically generated

**Detecting and Removing Outliers**

Detect and remove outliers from data (basically from numeric features), because there was no such concept of an outlier in the categorical data. By the above visualization of only two features, the dataset is not normally distributed. Therefore, I use **the IQR** method for detecting and removing outliers**.**

* Lower bound: Q1−1.5×IQR
* Upper bound: Q3+1.5×IQR. Any data points falling outside this range are considered outliers.

**Normalizing Numeric Features**

Numeric features such as KM, Manufacturing Year, Car Price, etc. were normalized using Min-Max scaling to bring them to a common scale, which improves model performance. I used Min-Max scaling because data was not normally distributed as we saw in above visualization.

scaler = MinMaxScaler()

df[numeric columns] = scaler.fit\_transform(df[numeric columns])

**Step 3: Correlation Heatmap**

**Objective**

To explore the relationship between numeric features and the target variable Numeric Price, a correlation heatmap was created. This helps visualize the strength of linear relationships between different features and the price.

**Visualization**

The heatmap shows the correlation coefficients, where values indicate a stronger linear relationship, and values near 0 indicate weak or no linear relationship.

->Strong Correlation (|correlation| > 0.7):

Features with a correlation greater than 0.7 (positive or negative) have a strong linear relationship with the target variable. These features are usually good candidates for inclusion in your model.

-> Moderate Correlation (0.4 < |correlation| < 0.7):

Features with moderate correlation (between 0.4 and 0.7) may still be useful,

especially if they provide meaningful insights.

->Weak or No Correlation (|correlation| < 0.4):

Features with a correlation lower than 0.4 have a weak linear relationship with the

target variable.

A screenshot of a computer

Description automatically generated

* Features like Manufacturing Year and KM may have a positive correlation with Numeric Price, as newer cars are typically priced higher.

**Step 4: Box Plots for Categorical Features**

**Objective**

Box plots were used to visualize the distribution of the target variable (Numeric Price) across different categories. This helps identify patterns and outliers within the categorical features such as Fuel Type, City, Transmission, Body Type, etc.

**Visualization: Box Plot**

Box plots display the median, quartiles, and potential outliers for Numeric Price based on categorical features. For example-Box plot of Transmission and Fuel Type with Numeric Price (i.e. Car Price) as follows:

A diagram of a diagram

Description automatically generated

A graph of a chart

Description automatically generated with medium confidence

**Interpretation**

* **Fuel Type**: Cars using Petrol may have a wider price range compared to Diesel or CNG cars, potentially indicating a higher variance in petrol car prices.
* **Transmission**: Transmission (manual vs automatic) can have a significant impact on car price. Automatic cars are often more expensive than manual cars, especially in certain markets.

**Conclusion on Visualization**

This analysis provides a detailed look into the Car Dekho dataset, focusing on the relationships between different features and the price of used cars. Key takeaways include:

* Strong correlations were observed between features like KM, and Manufacturing Year with the target variable.
* Categorical features such as Fuel Type, Body Type, City, Transmission and Insurance-Validity exhibited significant patterns in relation to car prices, as visualized through box plots.

**Step 5: Model Development**

**5.1 Model Selection**

To predict car prices, we tested three models: **Linear Regression**, **Decision Tree Regressor**, and **Random Forest Regressor**. Each model was evaluated using key performance metrics such as **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R2)**.

**Linear Regression**

Linear Regression was the simplest model tested. While it provides interpretable results, its assumption of linearity limited its ability to model the non-linear relationships present in the dataset. As a result, it underperformed on key metrics compared to the more complex models.

**Decision Tree Regressor**

The Decision Tree Regressor improved over Linear Regression by capturing non-linear relationships in the data. However, it tended to overfit, which resulted in less generalization to new data (lower performance on the test set).

**Random Forest Regressor**

The **Random Forest Regressor** was selected as the final model because it provided the best balance between capturing non-linear relationships and avoiding overfitting. By averaging the predictions from multiple decision trees, it improved both the generalization and accuracy of predictions. This resulted in the lowest error rates and the highest R-squared value among the tested models.

**Model Comparison**

The table below summarizes the performance of the three models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **MAE** | **MSE** | **RMSE** | **R-squared(R2)** |
| Linear Regression | 0.08 | 0.01 | 0.10 | 0.80 |
| Decision Tree | 0.10 | 0.02 | 0.14 | 0.63 |
| Random Forest | 0.08 | 0.01 | 0.10 | 0.81 |

Since both models (Linear Regression and Random Forest) perform similarly after tuning in terms of error metrics (MAE, MSE, RMSE), the decision can be based on **R-squared** and **model flexibility**:

* **Random Forest (R² = 0.81)** slightly outperforms the **Linear Regression (R² = 0.80)** in terms of explaining variance, it may capture more complexity in the data.
* **Random Forest** is a more flexible, non-linear model, so as our data contains non-linear relationships also, **Random Forest** would likely generalize better.
* **Linear Regression** is simpler and interpretable but may struggle with non-linear patterns.

**Conclusion**:

Based on the comparison of evaluation metrics, the **Random Forest Regressor** was chosen as the final model due to its superior ability to handle the non-linear relationships in the dataset while minimizing overfitting. It demonstrated the best performance in predicting car prices.