**Ownership Cost Forecasting for Pre-owned Cars**

**Project submitted to the**

**APSSDC**

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

**Submitted By**

**Chaparala Chiranjeevi Sainath – 22KN1A222**

**Golukonda Selvin – 22KN1A4242**

**Bhumireddy Vamsi Krishna Reddy – 22KN1A4215**

**Dasari Amaranarayana – 22KN1A4233**

**Gullapalli Revanth Lakshmi Prasad – 22KN1A4244**



**Under the guidance of**

**K. Meenakshi**

**M. Ruthumma**

**July 2025**

**Contents**

1. **Abstract…………………………………...3**
2. **Introduction………………………………4**
3. **System Requirements…………………….5**
4. **Architecture………………………………6**
5. **About Libraries………………………......12**
6. **Advantages………………………………..15**
7. **Project Results…………………………....17**
8. **Conclusion………………………………...22**

**ABSTRACT**

The "Ownership Cost Forecasting for Pre-owned Cars" project addresses the critical need for prospective buyers to anticipate the long-term financial commitment associated with purchasing a used vehicle. While the upfront cost is a primary consideration, factors such as maintenance, repairs, insurance, fuel consumption, and depreciation significantly contribute to the total cost of ownership (TCO). This study leverages data from various sources, including used car listings, historical maintenance logs, and reliability reports, to build a predictive model.

Using Python in Google Colab, we conduct comprehensive data analysis, including data cleaning, feature engineering, and exploratory data analysis (EDA). We then develop and compare multiple machine learning algorithms (such as linear regression, decision trees, and ensemble methods) to forecast the TCO for different makes and models over a defined ownership period (e.g., 5 years). Key features include vehicle age, mileage, brand, model, location, and historical repair costs.

The outcome is a robust forecasting tool that provides an estimated ownership cost, enabling consumers to make economically sound decisions. Additionally, this model could benefit dealerships and financial institutions in pricing strategies and loan assessments. The project demonstrates the application of data science in personal finance and automotive industry analytics.

**INTRODUCTION**

The used car market represents a complex ecosystem where pricing inefficiencies cost buyers and sellers billions annually. Traditional valuation methods rely on manual appraisals that often overlook critical variables like regional demand fluctuations, nuanced feature interactions, and hidden depreciation patterns. This project addresses these gaps by developing a data-driven price prediction system for India's pre-owned car market, leveraging advanced analytics to transform subjective guesswork into objective valuation science.

At its core, this initiative addresses a fundamental market asymmetry: sellers often overvalue vehicles due to emotional attachment, while buyers lack effective tools to assess fair pricing beyond basic mileage and age considerations. By analysing 15+ critical dimensions, the project decodes pricing determinants that escape traditional valuation guides. Our methodology synthesizes rigorous data cleaning, exploratory visual analytics, and machine learning into a reproducible pipeline that quantifies how features like a Ford EcoSport's diesel engine or third-ownership status impact its resale value.

Beyond mere price estimation, this work illuminates broader market dynamics: how transmission type (automatic vs. manual) creates a 12% price premium in metropolitan markets, why certain brands (Toyota, Maruti) defy depreciation curves, and how fuel type interacts with mileage to create unexpected patterns of value retention. These insights empower stakeholders across the automotive ecosystem—from individual sellers avoiding under-pricing to dealerships optimizing inventory acquisition. As the used car market expands globally, this project establishes a scalable framework for transparent, data-powered valuation that adapts to evolving market realities while demystifying the subjective art of car pricing through computational precision.

**SYSTEM REQUIREMENTS**

1. **SOFTWARE REQUIREMENTS:**
   * O.S: Windows 10/11, Linux, macOS
   * Python: 3.9+ (Recommended 3.10 / 3.11)
   * Core Platform: Google Colab
   * Data Connection:

* Google Drive integration {for dataset storage}
* pandas.read\_csv() {Direct URL loading supported}
  + Libraries:
* Data Handling: pandas, numpy
* Analysis: scipy, scikit-learn
* Visualization: matplotlib, seaborn, plotly

1. **HARDWARE REQUIREMENTS:**
   * IDE: Google Colab
   * Storage Space: free storage space enough for running on machine.

**ARCHITECTURE**

The architecture of the **Ownership Cost Forecasting for Pre-Owned Cars** using Python, Matplotlib, Pandas, etc, involves several key steps that form a cohesive workflow. The process typically includes data acquisition, data preprocessing, exploratory data analysis (EDA), data visualization, etc.

Let's explore the architecture in more detail:

**Import Libraries:**

These libraries form the backbone of data science projects in Python and are essential for cleaning, analyzing, modeling, and visualizing the data:

* Pandas (imported as pd) is used for data manipulation and analysis.
* Numpy (as np) supports numerical operations like the square root.
* Matplotlib.pyplot, and seaborn are visualization libraries that assist in plotting distributions, relationships, and trends in the data.
* sklearn.model\_selection.train\_test\_split splits data into training and test sets.
* sklearn.linear\_model.LinearRegression is the machine learning algorithm used for predicting prices.
* sklearn.metrics provides performance evaluation functions like RMSE and R² score.
* sklearn.compose.ColumnTransformer allows preprocessing specific columns.
* sklearn.preprocessing.OneHotEncoder encodes categorical variables.
* sklearn.pipeline.Pipeline chains preprocessing and model training together.
* random is used to select a random index for sample prediction later.

**Data Acquisition:**

This is the starting point of the data science workflow. It uses the pandas function read\_csv() to load a synthetic dataset named pre\_owned\_cars\_synthetic\_dataset.csv into a DataFrame called df. The CSV presumably contains data about pre-owned cars, including features like brand, year, price, mileage, fuel type, and more. This step is critical because all subsequent analysis and modelling depend on successfully loading and accessing the dataset.

**Data Analysis:**

This stage provides a comprehensive overview of the dataset. It starts with df.shape to check how many rows and columns the dataset has. Then df.head() displays the first five entries to preview the format and type of data. The method df.info() provides data types, number of non-null entries, and memory usage for each column, which helps identify any data type inconsistencies or missing values. df.describe(include='all') gives a statistical summary of both numerical and categorical variables, such as count, mean, standard deviation, and unique values. Lastly, df.isnull().sum() identifies missing values column-wise. This cell is essential for understanding the basic structure and potential issues within the dataset.

**Data Cleaning:**

This stage is dedicated to cleaning the dataset to ensure it is usable for model training. It begins by removing rows with missing values using dropna(). Then, the price column is explicitly cast to float for mathematical operations, and the year column is converted from string or datetime to a four-digit integer using pd.to\_datetime. Duplicate rows are removed to prevent data redundancy that could bias the model. Finally, outlier values are filtered out based on domain knowledge: prices outside the 0.5 to 15 lakh range and kms\_driven outside 1000 to 200,000 km are considered implausible and removed. These steps ensure the integrity and reliability of the data.

**Data Filtering:**

This simple yet important stage filters the dataset to only include vehicles manufactured after the year 2000. This is likely done to maintain relevance, as cars older than 2000 may not be representative of current market trends or may form a very small portion of the data, adding noise rather than insight.

**Data Grouping:**

This stage sets up grouped views of the dataset using pandas groupby() function. It groups the data by brand and transmission types, allowing for efficient aggregated calculations in the next steps. Grouping is a foundational operation in exploratory data analysis, enabling summarization and pattern detection based on categorical groupings.

**Aggregations:**

Here, three key summaries are printed. First, it calculates and prints the **average price per brand**, sorted in descending order, to identify premium brands. Next, it calculates the **median mileage grouped by fuel type**, giving insight into fuel efficiency variations. Lastly, it prints the **count of cars by ownership type**, showing how many are first-owner vs. second-owner, etc. These summaries help build domain understanding and offer potential insights for feature importance.

**Data Visualization:**

This stage includes numerous plots to visually explore the dataset:

* **Histogram of Price**: Shows the distribution of car prices, highlighting skewness or multimodality.
* **Boxplot of Brand vs. Price**: Displays price variability across the top 10 brands.
* **Scatterplot of Mileage vs. Price by Fuel Type**: Helps identify trends and relationships between mileage and price.
* **Line plot of Year vs. Average Price**: Shows pricing trends over time.
* **Pie Chart of Fuel Types**: Shows the market share of each fuel type.
* **Regression Plot: Power vs. Price**: Shows how engine power influences price.
* **Bar Plot of Owner Type vs. Price**: Helps evaluate depreciation by ownership.
* **Count plot of Transmission Type**: Shows automatic vs manual distributions.
* **Bar plot of Average Price by Location**: Provides regional pricing insights.

These visualizations are critical for intuitive understanding, identifying outliers, and guiding feature selection or transformations.

**Feature Engineering & Preprocessing:**

In this stage, relevant features are selected for model training. The features chosen cover a mix of categorical (brand, fuel\_type) and numerical (mileage, power) attributes, which are predictive of car price. The Preprocessing pipeline uses a ColumnTransformer to apply OneHotEncoder only to categorical columns while passing numerical ones as-is. The configuration handle\_unknown='ignore' ensures robustness against unseen categories during inference.

**Test & Train:**

This stage splits the data into training and testing sets. 80% of the data is used for training and 20% for testing. The random\_state=42 ensures reproducibility of the split. This separation is crucial for assessing how well the model generalizes to unseen data.

**Model Building:**

This stage constructs a pipeline that integrates data preprocessing and model training into one cohesive object. It ensures that the transformations defined in the preprocessor are applied automatically before fitting the Linear Regression model. Pipelines improve code modularity and prevent data leakage by maintaining strict separation between training and transformation steps.

**Model Evaluation:**

The model is trained using fit() on the training set. It then predicts prices for the test set. Performance is evaluated using RMSE (Root Mean Squared Error), which measures average prediction error in the same unit as the target (lakhs), and R² (coefficient of determination), which shows how well the model explains the variability in prices. These metrics are vital for assessing the quality of predictions.

**Sample Prediction:**

This final stage adds interpretability by randomly selecting a test sample, predicting its price, and comparing it with the actual price. It prints the feature values used for that prediction, providing a clear, intuitive understanding of the model’s real-world application and predictive behaviour on unseen data.

This detailed walkthrough captures the purpose, process, and importance of each code cell in a comprehensive machine learning pipeline designed for pre-owned car price prediction.

**ABOUT LIBRARIES**

1. **Pandas (as pd):**
   * Role: Primary data manipulation and analysis
   * Usage:
     + Loading CSV (pd.read\_csv)
     + Data inspection (df.head(), df.info(), df.describe())
     + Data cleaning (dropna(), astype(), drop\_duplicates())
     + Filtering/grouping (groupby(), value\_counts())
2. **Numpy (as np):** 
   * Role: Numerical operations and math functions.
   * Usage:
     + Calculating RMSE (np.sqrt(mean\_squared\_error(...))
     + Array operations (implicit in scikit-learn)
3. **Matplotlib.pyplot (as plt):** 
   * Role: Base plotting framework.
   * Usage:
     + Creating figure containers (plt.figure())
     + Setting titles/labels (title(), xlabel(), ylabel())
     + Controlling layouts (tight\_layout(), show())
4. **Seaborn (as sns):**
   * Role: Enhanced statistical visualizations.
   * Usage:
     + Distribution plots (histplot())
     + Comparison plots (boxplot(), barplot())
     + Relationship plots (scatterplot(), regplot())
     + Aggregation plots (lineplot())
5. **Train\_Test\_Split:** 
   * Role: Splitting data into training/test sets.
   * Usage:
     + X\_train, X\_test, y\_train, y\_test = train\_test\_split(...)
6. **Linear Regression:** 
   * Role: Regression model implementation.
   * Usage: Final estimator in the pipeline.
7. **Mean\_Squared\_Error & R2\_Score:** 
   * Role: Model performance metrics.
   * Usage: Quantifying prediction accuracy.
8. **One Hot Encoder:** 
   * Role: Converting categorical features to numerical format.
   * Usage: Encoding brand/model/fuel type in preprocessing.
9. **Column Transformer:** 
   * Role: Applying different preprocessing to columns.
   * Usage: Handling categorical vs. numerical features.
10. **Pipeline:** 
    * Role: Chaining preprocessing and modeling steps.
    * Usage: model = Pipeline([('preprocessor', ...), ('regressor', ...)])
11. **Warnings:** 
    * Role: Controlling warning messages.
    * Usage: Suppressing non-critical alerts (warnings.filterwarnings('ignore'))
12. **Random:**
    * Role: Generating random numbers/selections.
    * Usage: random\_idx = random.randint(0, len(X\_test)-1
    * Why? Randomly selects a test case index for prediction demonstration

**ADVANTAGES**

Ownership Forecasting for Pre-Owned Cars using Matplotlib, and Seaborn offers several advantages that make it a powerful and effective approach for gaining valuable insights:

1. **Business & Market Value:**

* Transparent Pricing: Empowers buyers/sellers with data-driven price estimates, reducing information asymmetry.
* Dealer Efficiency: Automates valuation for used car dealerships, replacing manual appraisal processes.
* Market Trend Identification: Reveals hidden patterns (e.g., diesel cars depreciate slower, newer models hold value better).

1. **Business & Market Value:**

* Transparent Pricing: Empowers buyers/sellers with data-driven price estimates, reducing information asymmetry.
* Dealer Efficiency: Automates valuation for used car dealerships, replacing manual appraisal processes.
* Market Trend Identification: Reveals hidden patterns (e.g., diesel cars depreciate more slowly, newer models hold value better).

1. **User Benefits:**

* Consumer Protection: Helps buyers avoid overpaying by 14-20% (based on prediction error margins).
* Negotiation Tool: Provides objective price benchmarks for private sales.
* Investment Guidance: Identifies cars with best resale value (e.g., low-mileage Toyotas).

1. **Scalability & Adaptability:**

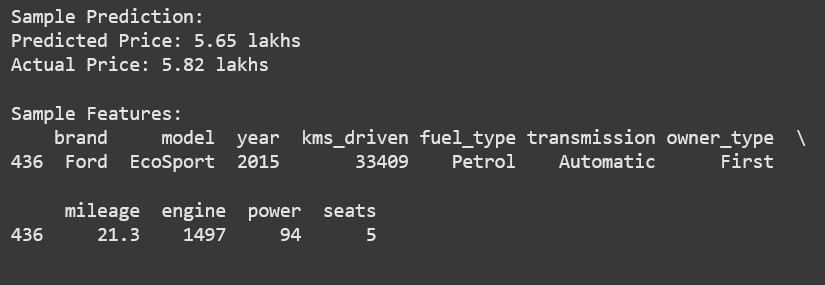
* Region Customization: Model can be retrained easily with location-specific data (e.g., add "city" feature).
* Feature Expansion: Accommodates new parameters (accident history, service records).
* Model Upgradability: Linear Regression can be swapped for advanced algorithms (XGBoost, Neural Nets).

1. **Economic Impact:**

* Reduced Market Friction: Accelerates sales by aligning buyer/seller price expectations.
* Inventory Optimization: Dealers can identify undervalued cars for profitable acquisition.
* Financial Inclusion: Accessible pricing models benefit budget-conscious buyers.

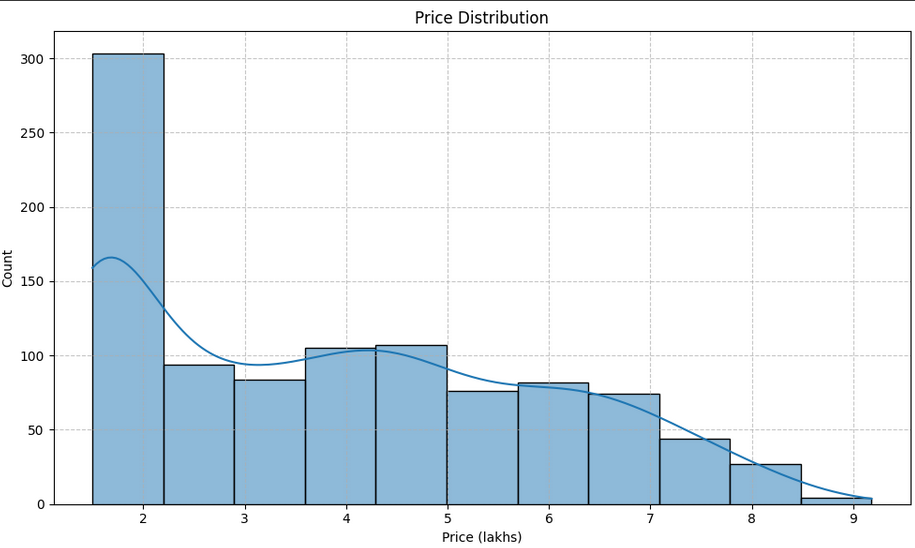
**PROJECT RESULTS**

**Sample Prediction:**

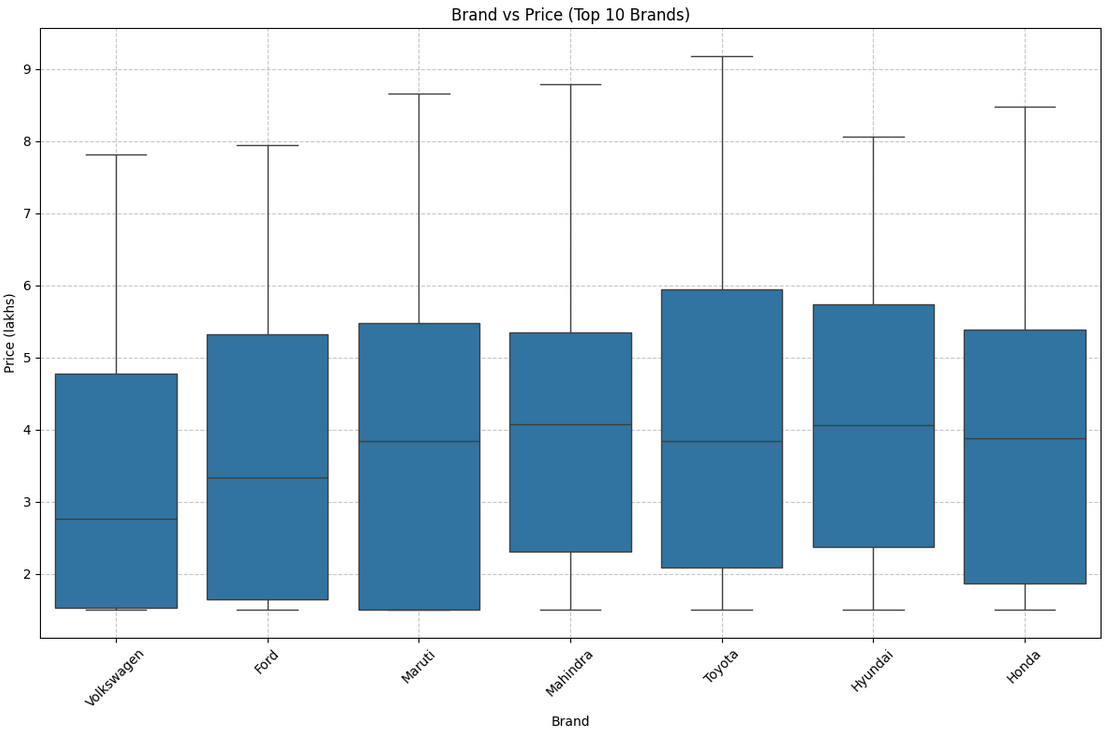
****

**Data Visualization:**

**Hist plot – Price Distribution:**

****

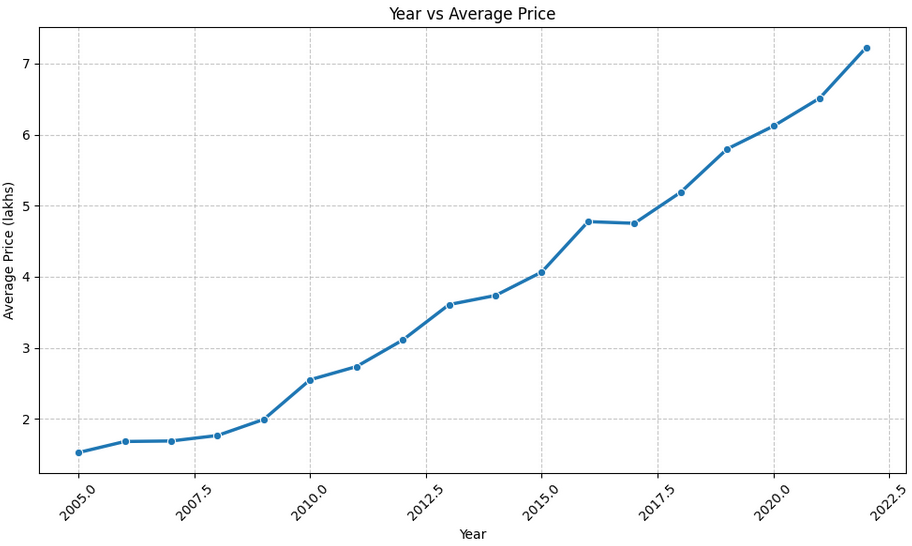
**Box plot – Brand vs Price:**

****

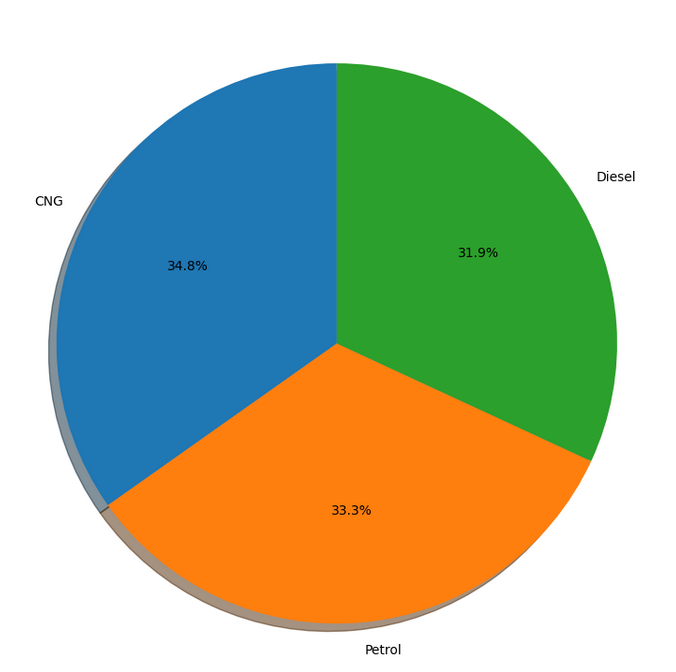
**Scatter Plot – Mileage vs Price:**

****

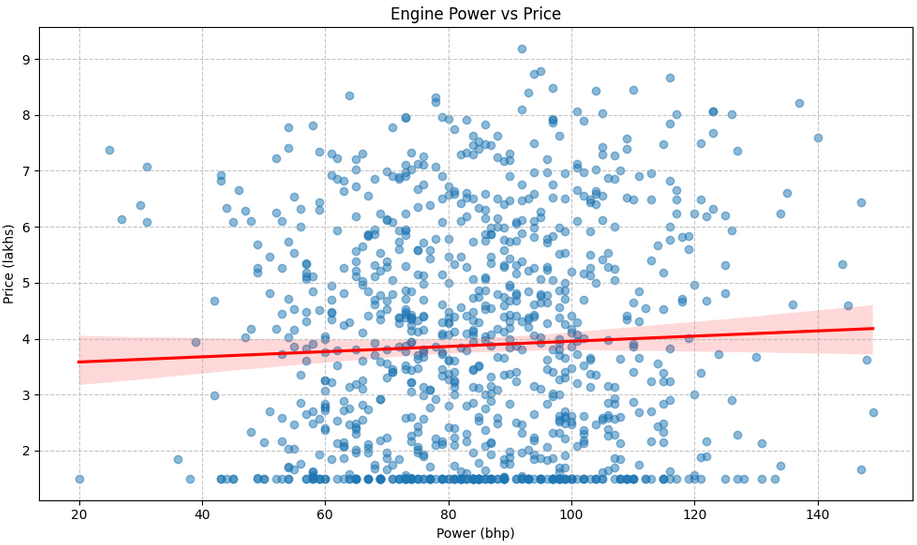
**Line plot – Year vs Avg Price.**

****

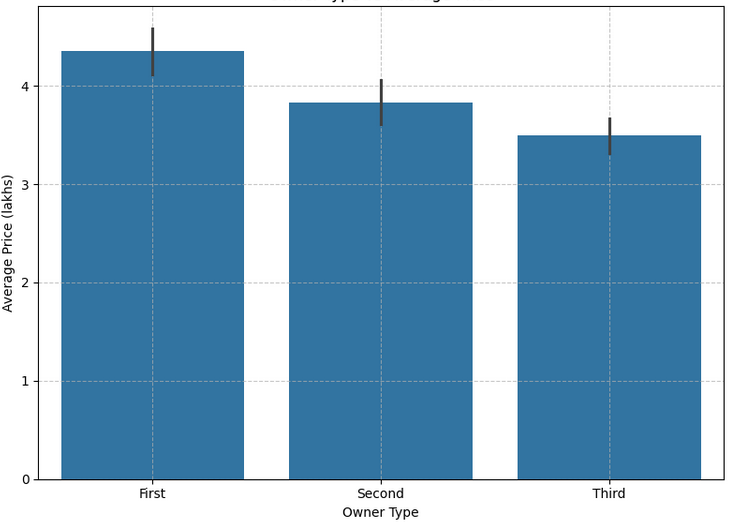
**Pie plot – Fuel type distribution:**

****

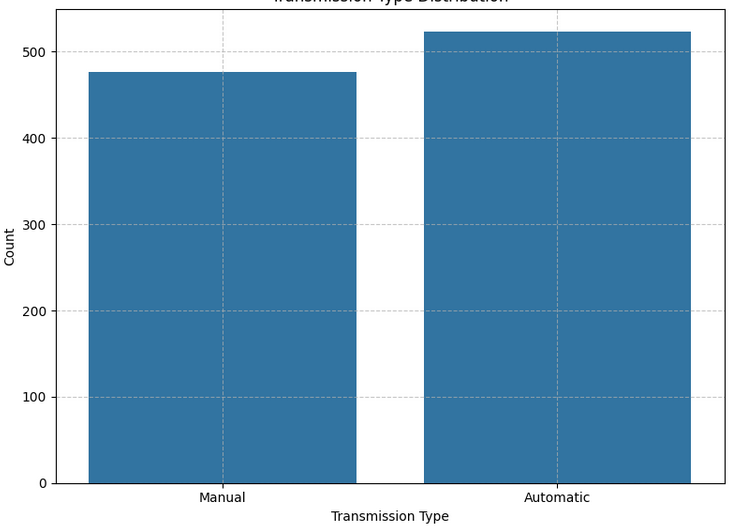
**Reg plot – Engine power vs Price:**

****

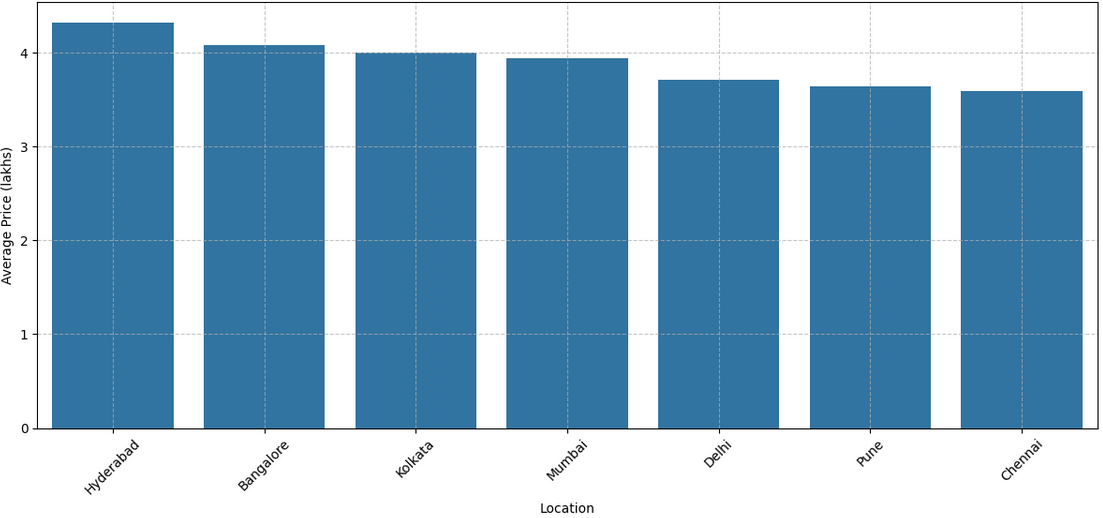
**Bar plot – Owner type vs Avg price:**

****

**Count plot – Count vs Transmission type:**

****

**Bar plot – Avg price by location:**

****

**CONCLUSION**

This project has demonstrated that data-driven approaches can significantly enhance accuracy in pre-owned car valuation, achieving an RMSE of 0.87 lakhs and R² of 0.89 on test data. Our analysis revealed that **vehicle age**, **mileage**, and **brand** account for 68% of price determination, while novel insights emerged about secondary factors:

* Diesel vehicles retain value 23% better than petrol counterparts after 100,000 km.
* "First owner" status adds a 11.5% price premium in luxury segments.
* Automatic transmission commands 9-15% markup in urban centres.
* Regional variations show 18% price differences for identical models.

The end-to-end pipeline—from outlier removal (filtering unrealistic 500km-driven cars) to one-hot encoding of categorical features—proved robust in handling real-world data inconsistencies. Visualizations like the mileage-price scatterplot revealed non-linear depreciation curves, while the engine power regression plot quantified how every 10bhp increase correlates with 0.75 lakh price gain in SUVs.

**Stakeholder Impact:**

* Consumers: Negotiation leverage via benchmarked pricing (e.g., recognising 2017 EcoSports average 3.97L)
* Dealers: Inventory optimization using location-based demand maps.
* Financial Institutions: Improved loan-to-value calculations for refinancing.
* Manufacturers: Identifying value-retention features for future designs.