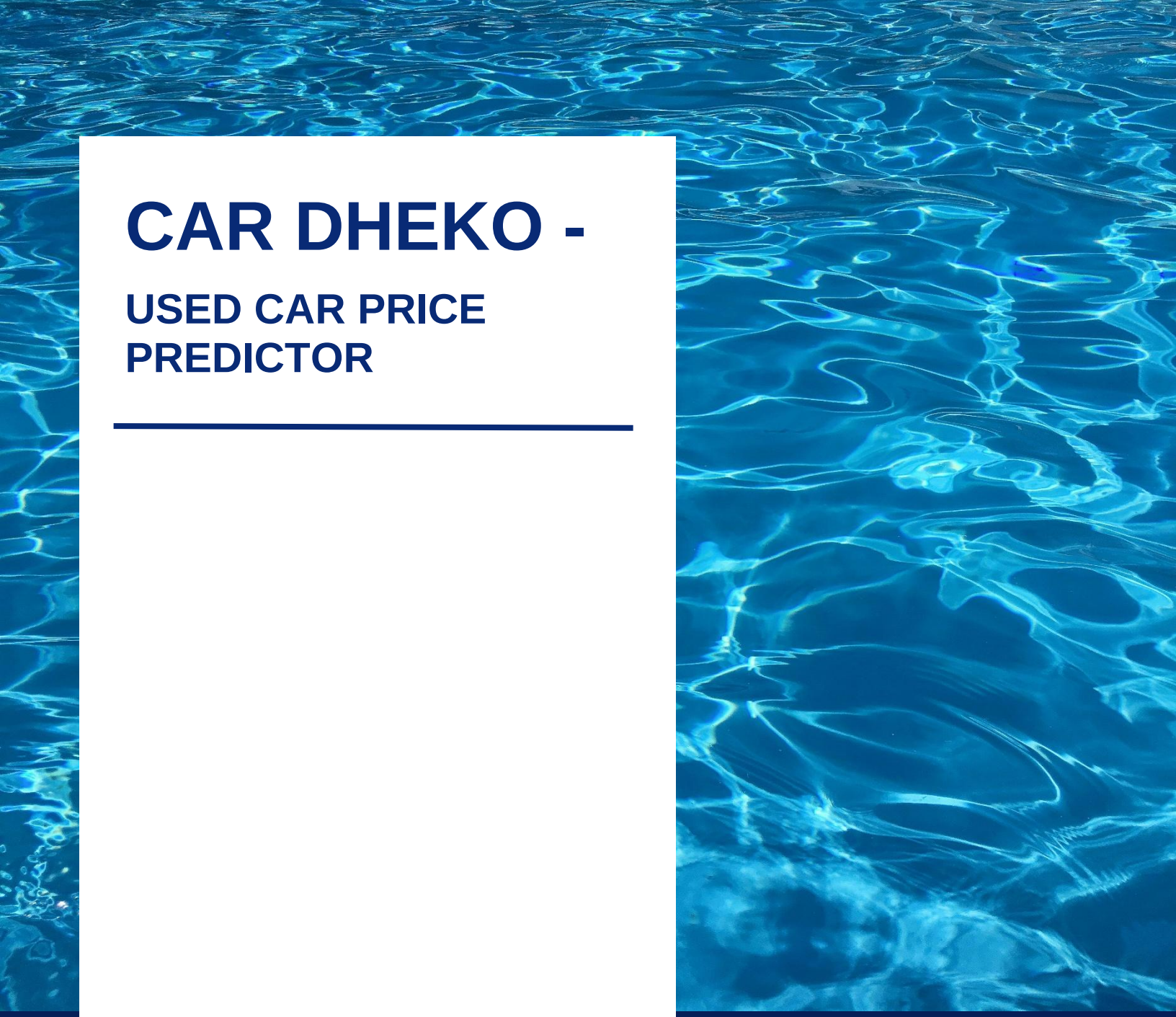

CAR DHEKO -

USED CAR PRICE PREDICTOR

PROJECT 3

POOJA SPANDANA

CAR DHEKO – USED CAR PRICE PREDICTOR

DOCUMENTATION

CONTENTS:

1. Objectives:

Given is the historical data of used car prices of six cities including various features such as make, model, year, fuel type, transmission type, and other relevant attributes from Car Dheko. Your task is to:

- Import the unstructured datasets, convert them into a structured format & concatenate all datasets into a single dataset.

- Preprocess the structured data (Handle missing values, Standardize data formats, Remove outliers & Encode categorical variables).

- Do Exploratory Data Analysis (EDA) on the data (Provide descriptive statistics, data visualization & identify important features)

- Develop a machine learning model that can accurately predict the prices of used cars based on these features.

- Evaluate model performance using MSE, MAE & R-squared metrics & compare models.

- Optimize the model using feature engineering or regularization to improve model performance.

- Choose the best model & integrate it into a Streamlit-based web application to allow users to input car details and receive an estimated price instantly.

2. Domain:

i. Automotive Industry
ii. Machine Learning
iii. Data Science

3. Skill Take Aways from this Project:

i. Data Cleaning and Preprocessing
ii. Exploratory Data Analysis
iii. Machine Learning Model Development
iv. Price Prediction Techniques
v. Model Evaluation and Optimization
vi. Model Deployment
vii. Streamlit Application Development
viii. Documentation and Reporting

4. Unstructured Datasets:

i. Unclean datasets Link:
https://drive.google.com/drive/folders/16U7OH7URsCW0rf91cwyDqEgd9UoeZAJh

ii. Feature Description Link:
https://docs.google.com/document/d/1hxW7IvCX5806H0IsG2Zg9WnVIpr2ZPueB4AElMTokGs/edit?usp=sharing

5. Data Cleaning & Preprocessing

i. Handling Missing Values:

Feature	Missing values	Method
Engine	4	Used Mode of Engine, as there are only a few missing values to fill
Seats	6	Used Mode of Seats, as there are only a few missing values to fill
Max Power	60	Used Mode of Max Power, as there are only a few missing values to fill
Torque	60	Used Mode of Torque, as there are only a few missing values to fill
Mileage	287	Used Mean of Mileage, as using Mode to fill the missing values might skew the data

ii. Standardizing Data Formats

- Extracted only the number part of the string & converted it into integer data type for the required features such as Kms Driven, Transmission, Seats, Mileage, Engine, Max Power, & Torque.

- Used a function convert_to_numeric(value) to remove ₹, commas, and strip spaces, check for 'Lakh' multiply by 100,000 or check for 'Crore' multiply by 10,000,000 to get the Price value & convert it into integer type using astype().

- Removing unwanted rows: Upon descriptive analysis Kms Driven & Engine were found to have values that are 0, since these features cannot have 0 value, the rows that had '0' were removed in these features. Also removed duplicate rows while keeping the 1st occurrence.

iii. Exploratory Data Analysis [EDA]

- **Numerical features of the dataset** - ['Kms Driven', 'Owner Number', 'Price', 'Seats', 'Mileage', 'Engine', 'Max Power', 'Torque', 'Car Age']

- **Categorical features of the dataset** - ['City', 'Fuel Type', 'Body Type', 'Transmission', 'Brand', 'Model', 'Variant Name']

- **Univariate Analysis** of numerical & categorical features using box plot (to identify outliers) & histplot with kde (to identify skewness of the feature)
- **Bivariate Analysis** of numerical & categorical features w.r.t Price (Target variable) to understand which features are related to Price (bar charts)
- **Multicollinearity** of numerical features using Correlation Matrix (Heatmap) to find relationships between features (values close to +1 indicate strong correlation)

iv. Handling Outliers:

- Used a function outlier() that takes a Data Frame (all_cities) and a specific column (column) as inputs. This function detects and handles outliers using the Interquartile Range (IQR) Method.

- It calculates the IQR, which is the difference between Q3 (75th percentile) and Q1 (25th percentile). It sets the upper and lower bounds for detecting outliers:
    - **Upper Bound** = Q3+(1.5×IQR)
    - **Lower Bound** = Q1−(1.5×IQR)

- Any value beyond these bounds is clipped (restricted) to the upper/lower bound instead of being removed. This ensures extreme values don't affect the model while preserving all data points.

v. Identifying important categorical features:

- **Chi-square test:** It is a statistical test that helps in feature selection for machine learning models with categorical data. It identifies which categorical features are important in predicting the target variable.

- **Hypothesis:**
    - **Null Hypothesis (H_0)**: The categorical feature and Price are independent (no association).
    - **Alternative Hypothesis (H_1)**: The categorical feature and Price are dependent (significant association exists).

- Removed few features like Model, Variant Name, & Body Type based on the results of the Chi-Square test, but also retained few features like Fuel Type & Brand based on domain knowledge.

6. Cleaned Dataset Overview:

i. The top 5 rows of All_Cities_Cleaned:

	City	Fuel Type	Kms Driven	Transmission	Owner Number	Brand	Price	Seats	Mileage	Engine	Max Power	Torque	Car Age
0	Bangalore	Petrol	120000	Manual	3	Maruti	400000	5	23	998	67	90	10
1	Bangalore	Petrol	32706	Manual	2	Ford	811000	5	17	1497	121	150	7
2	Bangalore	Petrol	11949	Manual	1	Tata	585000	5	23	1199	84	114	7
3	Bangalore	Petrol	17794	Manual	1	Hyundai	462000	5	19	1197	81	113	11
4	Bangalore	Diesel	60000	Manual	1	Maruti	790000	5	23	1248	88	200	10

ii. Feature Description of All_Cities_Cleaned:

- **City:** Name of the city where the car is being sold
- **Fuel Type:** Fuel type of the car
- **Kms Driven:** No. of kilometers driven by the car
- **Transmission:** Gear transmission of the car
- **Owner Number:** No. of people who owned the car previously
- **Brand:** Brand name of the car
- **Price:** Price at which the car was sold (also, the TARGET variable of the dataset)
- **Seats:** No. of seats in the car
- **Mileage:** kmpl that the car can travel
- **Engine:** CC of the car's engine
- **Max Power:** Maximum power that the car's engine can produce
- **Torque:** Traction that the car's engine can produce
- **Car Age:** Age of the car

7. Model Building:

i. Load Dataset:

- Loaded the clean dataset as data using pd.read_csv()
- data = pd.read_csv(r'file_path\All_Cities_Cleaned.csv')
- Final Dataset used for model building has 8226 rows × 13 columns with 4 categorical_features & 9 numerical_features

ii. Encoding Categorical variables:

- **LabelEncoder():** Used LabelEncoder() to encode categorical variables ['Transmission', 'Brand', 'Fuel Type', 'City'] as categorical variables data has no ordinal relationship & is used with models that can handle numeric categorical features (e.g., Decision Trees, Random Forest).

iii. Defining Feature Matrix & Target variable:

- y = data['Price']
- X = data.drop(columns=['Price'], axis=1)

iv. Train-test split:

- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

- X_train.shape() → (6580, 12)
- X_test.shape() → (1646, 12)
- y_train.shape() → (6580,)
- y_test.shape() → (1646,)

v. Scaling Numerical features:

- **MinMaxScaler():** Used MinMaxScaler() to scale ['Kms Driven', 'Mileage', 'Engine', 'Max Power', 'Torque', 'Car Age'] as they have large ranges & are on different scales

- If one feature has values from 0 to 1,000,000 and another from 0 to 1, the model might focus on the larger values. MinMaxScaler() ensures all features contribute equally by transforming features to a fixed range (0 to 1 or -1 to 1).

vi. Models Used:

- **Linear Regression** is a supervised machine learning algorithm that is used to predict a continuous dependent variable (i.e. a real value) based on one or more independent variables. It assumes a linear relationship between the independent variables and the dependent variable, and uses this relationship to fit a line through the data points.
  Model Evaluation:
    - **Training R^2** = 0.856 → 85.6% of variance in training data.
    - **Testing R^2** = 0.860 → Slightly better fit on test data.
    - **Issue** → High MSE & MAE, indicates that the model doesn't capture nonlinear relationships effectively.
    - ***Inference → Linear Regression assumes a straight-line relationship between features & target. Since the dataset is non-linear, this model struggles to capture the full complexity.***

- **Decision Tree Regressor** is a type of supervised machine learning algorithm that is used for regression tasks. It is a decision tree algorithm where each internal node represents a feature and each leaf node represents a predicted value. The tree is constructed by recursively splitting the feature space into smaller regions, with each split chosen to minimize a certain impurity criterion such as mean squared error.
  Model Evaluation:
    - **Training R^2** = 0.999 → Almost perfectly fits training data.
    - **Testing R^2** = 0.895 → Huge drop, showing overfitting.
    - **Issue →** Overfitting Model memorized training data but fails to generalize.
    - ***Inference → Extreme overfitting on train data & poor generalization to unseen data***

- **Random Forest Regressor** is an ensemble machine learning algorithm that is used for regression tasks. It is based on the decision tree algorithm, but instead of building a single decision tree, it builds multiple decision trees and combines their predictions to produce a final result.
  Model Evaluation:
    - **Training R^2** = 0.991 → High but less overfit than Decision Tree
    - **Testing R^2** = 0.943 → Good but shows gap, fails to generalize
    - **Issue →** Overfitting but much better than Decision Tree.
    - ***Inference → The testing MSE & MAE are lower than Decision Tree, meaning it makes better predictions.***

- **Gradient Boosting Regressor** is an ensemble learning machine learning algorithm that is used for classification & regression tasks. It is a boosting algorithm which combine multiple weak learners to create a strong predictive model. It works by sequentially training models where each new model tries to correct the errors made by its predecessor.
  Model Evaluation:
    - **Training R^2** = 0.941 → Lower than Random Forest (suggests less overfitting).
    - **Testing R^2** = 0.933 → Good generalization, slightly lower than Random Forest.
    - **Issue →** High Train & Test MAE than RF indicating large error range but performance is stable.
    - ***Inference → MSE & MAE are slightly worse than Random Forest, but still better than Decision Tree & Linear Regression.***

8. Optimizing Models:

Optimizing both Random Forest & Gradient Boosting models using the same feature engineering techniques & hyper parameter tuning to get better results

i. Feature Selection:

- The process of feature selection is an important step in ML, used to identify & select the most relevant features from a dataset to improve the performance and efficiency of a ML model.
- It can help to reduce overfitting, increase model interpretability, & improve the accuracy of predictions. Feature selection methods used in this project are:
    - Chi-squared Test
    - Correlation Matrix
    - Feature Importance from models

ii. Feature Engineering:

- df = data.copy()
- **Created New Features**
    - df['Age vs Performance'] = df['Car Age'] * df['Mileage'] (Older cars may have reduced efficiency)
    - df['Fuel Efficiency'] = df['Mileage'] / df['Engine'] (Smaller engines with higher mileage are more efficient)
- **Dropped Fuel Type & Seats as it has low importance & Torque as it has a strong correlation with Max Power**
    - df.drop(columns=['Torque', 'Seats', 'Fuel Type'], inplace=True)

- **Feature Engineered Dataset:**
    - City
    - Kms Driven
    - Transmission
    - Owner Number
    - Brand
    - Price
    - Mileage
    - Engine
    - Max Power
    - Car Age
    - Age vs Performance
    - Fuel Efficiency

iii. Pre-processing steps: (All the same pre-processing steps are applied on this new dataset before optimizing these models)

- **Encoding Categorical variables** → (features are already encoded as we are using a copy of data)
- **Defining Feature Matrix & Target variable** → (X, y)
- **Train-test split** → (X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42))
- **Scaling Numerical features** → MinMaxScaler() on ['Kms Driven', 'Mileage', 'Engine', 'Max Power', 'Car Age', 'Age vs Performance', 'Fuel Efficiency']

iv. Building Models:

- **Models 0 (Random Forest & Gradient Boosting)**
    - Uses the OG dataset & trained on default parameters
- **Models 1 (Random Forest(Base) & Gradient Boosting(Base))**
    - Used the Feature Engineered dataset & trained on default parameters
- **Models 2 (HP Tuned Random Forest & HP Tuned Gradient Boosting)**
    - Used the Feature Engineered dataset & hyper parameter tuned using RandomSearchCV
- **Models 3 (HP Tuned Random Forest 2 & HP Tuned Gradient Boosting 2)**
    - Further Feature Elimination based on feature_importances_ of HP Tuned Random Forest & HP Tuned Gradient Boosting respectively & hyper parameter tuned using RandomSearchCV

9. Evaluation of Optimized Models:

i. Model-Specific Analysis of Random Forest Models:

Model	Model name	Training R^2	Testing R^2
0	Random Forest	0.991674	0.943553
1	Random Forest (Base)	0.991485	0.939585
2	HP Tuned RF	0.991562	0.941124
3	HP Tuned RF 2	0.951445	0.927455

- **Random Forest (Default Parameters) - OG Dataset**

  Performance:

  - ***Training*** → **MSE**: 1.94 B, **MAE**: 27369, **R^2**: 0.991674 (Very high, potential overfitting)
  - ***Testing*** → **MSE**: 1.46 B, **MAE**: 74323 (Higher than training, showing overfitting), **R^2**: 0.943553 (Good, but lower than training)

  Inference:

  - The model has a large gap between training and testing error, confirms overfitting.
  - The feature importance suggests dominance of a few features(Max Power (~0.55)), making the model potentially biased.

- **Random Forest Base (Default Parameters) - Feature Engineering & Feature Elimination were applied**

  Performance:

  - ***Training*** → **MSE**: 1.98 B, **MAE**: 27729, **R^2**: 0.991485
  - ***Testing*** → **MSE**: 1.56 B, **MAE**: 76781, **R^2**: 0.939585 (Slight drop in generalization)

  Inference:

  - Even with engineered features Max Power still dominates (~0.44)
  - Still, the error gap remains significant, suggesting that default hyperparameters are not sufficient.

- **HP Tuned Random Forest - Feature Engineering, Feature Elimination, & Hyperparameter Tuning were applied**

  Performance:

  - *Training* → **MSE**: 1.96 B, **MAE**: 28787, **R^2**: 0.991562
  - *Testing* → **MSE**: 1.52 B, **MAE**: 77385, **R^2**: 0.941124 (Slight improvement over Model 1)

  Inference:

  - Hyperparameter tuning slightly reduced overfitting, as seen in the smaller error gap. However, the model is still overfitting
  - More balanced contribution across features than before as few features gained importance.

- **HP Tuned Random Forest 2 - Further Feature Elimination, & Hyperparameter Tuning were applied**

  Performance:

  - ***Training*** → **MSE**: 11.31 B, **MAE**: 72171, **R^2**: 0.951274
  - ***Testing*** → **MSE**: 18.76 B, **MAE**: 90082, **R^2**: 0.927194 (Lower than Model 2 but more balanced)

  Inference:

  - Training error increased, but testing error did not increase significantly, showing better generalization.
  - The model is less overfit than before, proving that removing weaker features improved generalization.

- ***HP Tuned RF 2 sacrifices a bit of training performance but significantly reduces overfitting & Feature reduction helped balance the importance scores, leading to better generalization, making it the best model.***
- ***Though the HP Tuned RF 2 is the best model it's test MAE is higher than the basic Gradient Boosting model. So, let's see the results of optimized Gradient Boosting models & see which model is performing better***

ii. Model-Specific Analysis of Gradient Boosting Models:

Model	Model name	Training R^2	Testing R^2
0	Gradient Boosting	0.941162	0.933065
1	Gradient Boosting (Base)	0.939063	0. 931166
2	HP Tuned GB	0. 9904	0. 953564
3	HP Tuned GB 2	0. 968806	0. 938032

- **Gradient Boosting (Default Parameters) - OG Dataset**

  Performance:

  - ***Training*** → **MSE**: 13.7 B, **MAE**: 80,942, **R^2**: 0.941
  - ***Testing*** → **MSE**: 17.3 B, **MAE**: 87,804, **R^2**: 0.933

  Inference:

  - The feature importance suggests model is biased towards dominant variables, causing high variance.
  - Though generalization is decent, the error gap between training & testing shows some overfitting & suggests room for improvement.

- **Gradient Boosting Base (Default Parameters) - Feature Engineering & Feature Elimination were applied**

  Performance:

  - ***Training*** → **MSE**: 14.2 B, **MAE**: 82,729, **R^2**: 0.939
  - ***Testing*** → **MSE**: 17.8 B, **MAE**: 88,992, **R^2**: 0.931

  Inference:

  - Feature engineering did not significantly improve generalization. MSE & MAE slightly increased, but R^2 is slightly lower
  - Still, the error gap remains significant, suggesting that default hyperparameters are not sufficient.

- **HP Tuned Gradient Boosting - Feature Engineering, Feature Elimination, & Hyperparameter Tuning were applied**

  Performance:

  - ***Training*** → **MSE**: 2.23 B, **MAE**: 33,696, **R^2**: 0.990
  - ***Testing*** → **MSE**: 12.01 B, **MAE**: 69,605, **R^2**: 0.953

  Inference:

  - Hyperparameter tuning significant improved generalization! Testing R^2 increased to 0.953, while MAE reduced to 69,605
  - More balanced contribution across features than before as few features gained importance. Possible slight overfitting, as Training R^2 is much higher than Testing R^2.

- **HP Tuned Gradient Boosting 2 - Further Feature Elimination, & Hyperparameter Tuning were applied**

  Performance:

  - ***Training*** → **MSE**: 7.27 B, **MAE**: 60,808, **R^2**: 0.968
  - ***Testing*** → **MSE**: 16.03 B, **MAE**: 85,514, **R^2**: 0.938

  Inference:

  - Feature elimination dropped Training R^2 to 0.968, which suggests the model has reduced overfitting, Testing R^2 slightly decreased to 0.938 but is still strong.
  - MSE & MAE worsened on testing data, indicating that some important features may have been removed, performance dropped slightly
  - Reduced complexity & overfitting at the cost of accuracy

10. Conclusion:

HP Tuned Gradient Boosting is the best model overall as it has the best tradeoff between Accuracy & Generalization, though slightly Overfit.