

CAR DHEKO – USED CAR PRICE PREDICTION.

Importing Libraries and Reading Data:

The code begins by importing the necessary libraries:

- `pandas`: For data manipulation and analysis.
- `numpy`: For numerical operations.
- `ast`: To safely evaluate and process string representations of lists/dictionaries.
- `re`: For regular expression operations.
- `matplotlib.pyplot` and `seaborn`: For visualizations.
- `scipy.stats`: For statistical analysis, especially in identifying outliers.

Reading Excel Files:

The data for cars from various cities such as Bangalore, Chennai, Delhi, Hyderabad, Jaipur, and Kolkata is loaded from Excel files. Each city's data is then assigned a `Location` attribute corresponding to its respective city.

```
bangalore = pd.read_excel("C:/Users/Darkk/Downloads/cardheko/Dataset/bangalore_cars.xlsx")
```

Merging the Datasets:

All individual city datasets are concatenated into a single `DataFrame` using `pd.concat()`:

```
df_concat= pd.concat([bangalore,chennai,delhi,hyderabad,jaipur,kolkata],axis = 0)
```

The merged data is saved to an Excel file for future use and reloaded into the `merged_car` `DataFrame` for further processing.

Data Normalization and Cleaning

Unpacking Nested Dictionaries:

The columns in the dataset (e.g., `new_car_detail`, `new_car_overview`, `new_car_feature`, `new_car_specs`) are stored as dictionaries or lists. These columns are normalized using the `pd.json_normalize()` method and custom functions that extract the relevant key-value pairs.

Processing Specific Columns:

The `new_car_feature` column is processed by extracting top-level features as well as nested data within it.

```
def process_features(row):  
    features = {item['value']: True for item in row['top']]  
    return features
```

Removing Duplicate and Unnecessary Data:

The dataset is cleaned by removing duplicate rows, dropping columns with more than 50% missing values, and removing columns that have only one unique value.

```
threshold=len(nr_cars)*0.5
nr_cars.dropna(axis=1,thresh=threshold,inplace=True)
nr_cars.shape
```

Converting Price, Kilometres, and Mileage Columns:

To make the data uniform, columns like price, km and mileage are processed to remove unwanted symbols or convert data types for better numerical analysis. For example, prices are converted from strings like "₹ 7 Lakh" to numerical values in rupees.

```
price_str = price_str.replace('Lakh', '').strip()
return float(price_str) * 100000
```

Outlier Detection, Analysis, and Final Cleaning

Identifying Outliers:

The z-score method is used to detect outliers in numeric columns such as price, km and mileage. Any value with a z-score greater than 3 is considered an outlier.

Removing Outliers:

Once outliers are identified, they are filtered out from the dataset to prevent skewed analysis.

Correlation Analysis:

The correlation between numerical columns is analysed, particularly focusing on the correlation with the price column. A heatmap is generated to visualize these correlations using seaborn.

Dropping Irrelevant Columns:

After the correlation analysis, columns that are deemed irrelevant or uncorrelated with price are dropped to simplify the dataset.

3.5. Final Adjustments:

Lastly, categorical columns like RTO are mapped to broader categories using a dictionary, and redundant data is dropped. The cleaned data is saved to a final Excel file.

This imports, cleans, and processes a large dataset of car listings from multiple cities. It performs significant data normalization, removes outliers, and prepares the data for further analysis, such as price predictions or exploratory data analysis. The final cleaned dataset is well-structured and ready for more advanced operations such as machine learning models or statistical analysis.

Loading Data:

- The Excel file containing car data is loaded into a Pandas Data Frame(df_cars_final).
- Categorical and numerical features are identified separately.

Data Preparation:

- The target variable price is separated from the features.
- One-hot encoding is applied to categorical features to convert them into numerical format, suitable for machine learning models.
- The data is split into training and testing sets using an 80-20 split.

Model Training & Evaluation:

- Four regression models are defined: Linear Regression, Decision Regression, Random Forest Regression and XGB Regression.
- For each model:

The model is trained on the training data.

Predictions are made on both the training and testing sets.

Performance metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared (R2), and Mean Absolute Percentage Error (MAPE) are calculated for both training and testing sets.

- The results for all models are stored and displayed.

Before Fitting Model:

	MSE_train	MAE_train	R2_train	MAPE_train \
Linear Regression	7.178467e+10	158572.249671	0.865367	28.463700
Decision Tree	1.812904e+08	704.667625	0.999660	0.079229
Random Forest	6.362064e+09	41600.658961	0.988068	6.022444
XGBoost	9.945143e+09	65721.555442	0.981348	10.836095

	MSE_test	MAE_test	R2_test	MAPE_test
Linear Regression	7.988207e+10	161289.733168	0.811879	29.684028
Decision Tree	6.707526e+10	130917.766922	0.842039	18.239362
Random Forest	3.416245e+10	99923.812095	0.919548	14.953844
XGBoost	3.375828e+10	97552.215921	0.920500	14.406064

Hyperparameter Tuning (XG Boost & Random Forest):

- Randomized Search CV is used to fine-tune the hyperparameters of XG Boost using a random search strategy.
- The best hyperparameters are identified and used to retrain the XG Boost model with these parameters

n_estimators=300, max_depth=5, min_samples_split=2, min_samples_leaf=1, random_state=42

```
XGBoost Train MSE: 12361176071.35584
XGBoost Train R^2: 0.9768164570966686
XGBoost Train MAPE: 12.428734735746211%
XGBoost Test MSE: 31687653737.574192
XGBoost Test R^2: 0.9253760379212769
XGBoost Test MAPE: 14.645574731494088%
```

- Similarly, Random Forest is fine-tuned with manually specified best parameters, and both models are evaluated again with the optimal settings.

```
Random Forest Train MSE: 81416820414.9345
Random Forest Train R^2: 0.8473017180366543
Random Forest Train MAPE: 25.834345559759996%
Random Forest Test MSE: 76097144102.37793
Random Forest Test R^2: 0.820792336257405
Random Forest Test MAPE: 25.033820002890728%
```

Saving the Results:

- The tuned models (XG Boost and Random Forest) are saved using joblib for later use.
- The predictions on the test set are saved to both CSV and Excel formats.
- The feature importance of the XG Boost model is extracted, saved to CSV, and Excel formats for analysis.

The final output will be a Streamlit web application where users can input various details about a used car (such as brand, colour, transmission, fuel type, model year, kilometres driven, engine capacity, etc.) via a sidebar form. Once the user clicks the "Predict" button, the app will process the input data, predict the price of the car using the pre-trained XG Boost model, and display the predicted price in an INR (Indian Rupee) format, along with a car image.

Display of streamlit application the user will get:

1. **Sidebar Form:** Users can input car details such as brand, colour, fuel type, transmission, engine size, kilometres driven, etc.
2. **Prediction Button:** Once the details are provided, clicking the "Predict" button triggers the model.
3. **Result Display:** The predicted price is displayed in INR format along with an image of a car.
4. **Background:** The app will have a customized background colour set to #D2B48C (light tan).

Enter car details:

Brand:

Color:

Transmission:

Fuel Type:

Gearbox:


Location:

Model Year:

Mileage/Kilometers:

Engine Capacity:

Car Dheko: Used Car Price Predictor



The predicted price is: ₹ 5,89,020.4