Car Price Data Cleaning & Machine Learning Project

Overview

Goal: Clean raw car dataset and prepare it for data analysis and machine learning.
Focus Areas: Data Cleaning, Feature Engineering, Encoding, Scaling, Outlier Detection, and Model Training.
Languages/Tools: Python, Pandas, NumPy, Scikit-Learn, Matplotlib, Seaborn, Jupyter Notebook.

Recommended Folder Structure

```
car-price-ml-project/
├─ data/
│   ├─ raw.csv
│   └─ cleaned.csv
├─ notebooks/
│   └─ data_cleaning.ipynb
├─ src/
│   ├─ cleaning.py
│   └─ ml_pipeline.py
├─ requirements.txt
└─ README.md
```

Workflow Summary

1. Load and explore the dataset (`head()`, `info()`, `describe()`).
2. Clean text-based numeric fields (`Lakh`, `Crore`, `kms`, `cc`, `Seats`).
3. Handle missing and inconsistent data.
4. Detect and treat outliers (IQR method / Boxplot).
5. Feature engineering (e.g., `car_age`, `price_per_km`).

6. Encode categorical columns (LabelEncoder / OneHotEncoder).
7. Scale numeric features (Standard / MinMax / Robust).
8. Train and evaluate machine learning models (Linear Regression, Random Forest).
9. Export cleaned data and model performance reports.

Requirements

requirements.txt

```
pandas
numpy
scikit-learn
matplotlib
seaborn
jupyter
```

Install:

```
pip install -r requirements.txt
```

Step-by-Step Data Cleaning

1 Load Dataset

```
import pandas as pd
df = pd.read_csv("data/raw.csv")
print(df.head())
print(df.info())
```

2 Convert Car Prices (Handle Lakh / Crore / Numbers)

```
def convert_price(price):
    if pd.isna(price): return None
    p = str(price).replace(',', '').strip().lower()
    try:
        if 'lakh' in p:
```

```
            return int(float(p.replace('lakh','').strip()) * 100000)
        if 'crore' in p:
            return int(float(p.replace('crore','').strip()) *
10000000)
        return int(float(p))
    except:
        return None

df['car_prices_in_rupee'] =
df['car_prices_in_rupee'].apply(convert_price)
```

3 Clean Kms Driven

```
def clean_kms(x):
    if pd.isna(x): return None
    try:
        return
int(str(x).lower().replace('kms','').replace(',','').strip())
    except:
        return None

df['kms_driven'] = df['kms_driven'].apply(clean_kms)
```

4 Clean Engine Column

```
def clean_engine(x):
    if pd.isna(x): return None
    try:
        return
int(str(x).lower().replace('cc','').replace(',','').strip())
    except:
        return None

df['engine'] = df['engine'].apply(clean_engine)
```

5 Clean Seats Column

```
df['Seats'] = pd.to_numeric(
    df['Seats'].astype(str).str.replace('seats','', case=False,
regex=False).str.strip(),
    errors='coerce'
).astype('Int64')
```

6 Clean Ownership Column

```
def clean_ownership(x):
    if pd.isna(x): return None
    s = str(x).lower()
    if '1' in s: return 1
    if '2' in s: return 2
    if '3' in s: return 3
    return None

df['ownership'] = df['ownership'].apply(clean_ownership)
```

7 Convert Manufacture to Year (or random realistic dates)

Option A – Year only (recommended):

```
df['manufacture'] = pd.to_datetime(df['manufacture'],
errors='coerce').dt.year.astype('Int64')
```

Option B – Random realistic dates (optional):

```
import numpy as np
rand_ts =
pd.to_datetime(np.random.randint(pd.Timestamp('2010-01-01').value//1
0**9,

pd.Timestamp('2024-12-31').value//10**9,
                                          size=len(df)), unit='s')
df['manufacture'] = pd.Series(rand_ts).dt.date
df['manufacture'] = pd.to_datetime(df['manufacture']).dt.normalize()
```

Outlier Detection (IQR Method)

```
def detect_outliers_iqr(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
```

```
    return series[(series < lower) | (series > upper)], lower, upper

outliers_price, low, up =
detect_outliers_iqr(df['car_prices_in_rupee'])
print("Price outliers:", outliers_price.shape[0])
print(outliers_price.head())
```

Feature Engineering Examples

```
df['car_age'] = 2025 - df['manufacture'].astype(int)
df['price_per_km'] = df['car_prices_in_rupee'] /
df['kms_driven'].replace(0, pd.NA)
```

Encoding Categorical Columns

```
from sklearn.preprocessing import LabelEncoder

cat_cols = df.select_dtypes(include='object').columns.tolist()
le = LabelEncoder()
for c in cat_cols:
    df[c] = le.fit_transform(df[c].astype(str))
```

(Use `pd.get_dummies()` for OneHotEncoding if needed.)

Feature Scaling

```
from sklearn.preprocessing import StandardScaler

features = ['kms_driven','engine','Seats','car_age','company_name',
            'fuel_type','transmission','ownership']

scaler = StandardScaler()
df_scaled = df.copy()
df_scaled[features] = scaler.fit_transform(df[features])
```

Summary

Step	Goal	Key Methods
Data Cleaning	Fix messy formats, missing values	Pandas string ops
Outlier Detection	Identify extremes	IQR, boxplots
Feature Engineering	Add new useful fields	Car age, price/km
Encoding	Convert categories → numbers	LabelEncoder / OneHot
Scaling	Normalize numeric columns	Standard / MinMax / Robust
Model Training	Predict car prices	Linear Regression, RandomForest
Evaluation	Measure accuracy	R^2, MAE, RMSE