CSI4142 - A3: Part 1

Group: 9

Members:

- Jay Ghosh (300243766)
- Alexander Azizi-Martin (300236257)

Introduction

This notebook illustrates a high-level workflow for preparing and modeling a dataset using linear regression. The process begins with basic data validation and duplicate removal. Categorical features are then one-hot encoded, and LOF is employed to numeric outliers. New features are engineered to capture aspects like depreciation and usage patterns. Several variants of the dataset were tested using linear regression, with cross-validation guiding the choice of final model.

Dataset Description

Dataset Name: CAR DETAILS FROM CAR DEKHO [1]

Dataset Author: Nehal Birla, Nishant Verma, Nikhil Kushwaha [1]

Purpose: The dataset was built for a pedagogical purpose: to exemplify the use of linear regression in machine learning. [1]

In [1]:
```
import kagglehub
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
# Loading dataset
df_path = kagglehub.dataset_download("nehalbirla/vehicle-dataset-from-cardekho")
df = pd.read_csv(f"{df_path}/CAR DETAILS FROM CAR DEKHO.csv")
df.head()
```

```
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.9), please consider upgrading to th
e latest version (0.3.10).
```

Out[2]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

Dataset Shape

In [3]:
```
df.shape
```

Out[3]:
```
(4340, 8)
```

The dataset has 4340 rows and 8 columns.

Features of the dataset (and what they mean)

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   name           4340 non-null   object
 1   year           4340 non-null   int64 
 2   selling_price  4340 non-null   int64 
 3   km_driven      4340 non-null   int64 
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

Features:

name:

- Type: Categorical
- Purpose: Represents the model or make of a car.

year:

- Type: Numerical
- Purpose: Year of manufacture of the car.

selling_price:

- Type: Numerical
- Purpose: Price at which the car is sold. **The target variable for regression.**

km_driven

- Type: Numerical
- Purpose: Kilometers driven by the vehicle, impacting its depreciation.

fuel

- Type: Categorical
- Purpose: Type of fuel used, e.g. petrol, diesel.

seller_type

- Type: Categorical
- Purpose: Indicates the type of seller, e.g. individual or dealer.

transmission

- Type: Categorical
- Purpose: Indicates vehicle transmission type, e.g. automatic or manual.

owner

- Type: Categorical
- Purpose: Number of previous owners, indicating vehicle usage and condition history.

Section A: Validating and Cleaning

Check 1: Data Type

In [5]:
```python
# Define columns with explicitly expected datatypes
expected_dtypes = {
    "year": "numeric",
    "selling_price": "numeric",
    "km_driven": "numeric",
    "name": "string",
    "fuel": "string",
    "seller_type": "string",
    "transmission": "string",
    "owner": "string"
}

# Checker Code
errors = {}
for col, expected_type in expected_dtypes.items():
  if expected_type == "numeric":
    parsed_col = pd.to_numeric(df[col], errors="coerce")
    failed_mask = parsed_col.isna() & df[col].notna()
  elif expected_type == "string":
```