# Training Day 16 Report

**Date:** 14 July 2025
**Topic:** Linear Regression with Real Dataset (Car Selling Price Prediction)

### Overview

Today's session extended the concept of **Linear Regression** by applying it to a real dataset (cars.csv). The dataset contains details of used cars, including kilometers driven, fuel type, owner type, and selling price. The goal is to analyze the dataset and build a **regression model** to predict the selling price of cars.

### Dataset Description

The dataset consists of the following columns:

- **brand**: Car manufacturer (e.g., Maruti, Skoda, Honda)
- **km_driven**: Number of kilometers the car has been driven
- **fuel**: Type of fuel (Petrol, Diesel, etc.)
- **owner**: Ownership status (First Owner, Second Owner, etc.)
- **selling_price**: Price at which the car is being sold

Example data:

| brand | km_driven | fuel | owner | selling_price |
|---|---|---|---|---|
| Maruti | 145500 | Diesel | First Owner | 450000 |
| Skoda | 120000 | Diesel | Second Owner | 370000 |
| Honda | 140000 | Petrol | Third Owner | 158000 |
| Hyundai | 127000 | Diesel | First Owner | 225000 |
| Maruti | 120000 | Petrol | First Owner | 130000 |

**Steps Performed**

**1. Importing Libraries and Dataset**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('cars.csv')

# Display first few rows
print(df.head())
```

**2. Data Exploration**

Checking dataset shape and summary:

```
print(df.shape)       # Number of rows and columns
print(df.info())      # Data types and null values
print(df.describe())  # Summary statistics
```

**3. Data Preprocessing**

- Converted **categorical columns** (fuel, owner, brand) into numeric values using one-hot encoding.
- Selected **km_driven** and encoded features as independent variables.
- Selling price was used as the target variable.

```
df_encoded = pd.get_dummies(df, drop_first=True)

X = df_encoded.drop('selling_price', axis=1)
y = df_encoded['selling_price']
```

**4. Train-Test Split**

```
from sklearn.model_selection import train_test_split

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

)

**5. Model Training and Prediction**

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

**6. Model Evaluation**

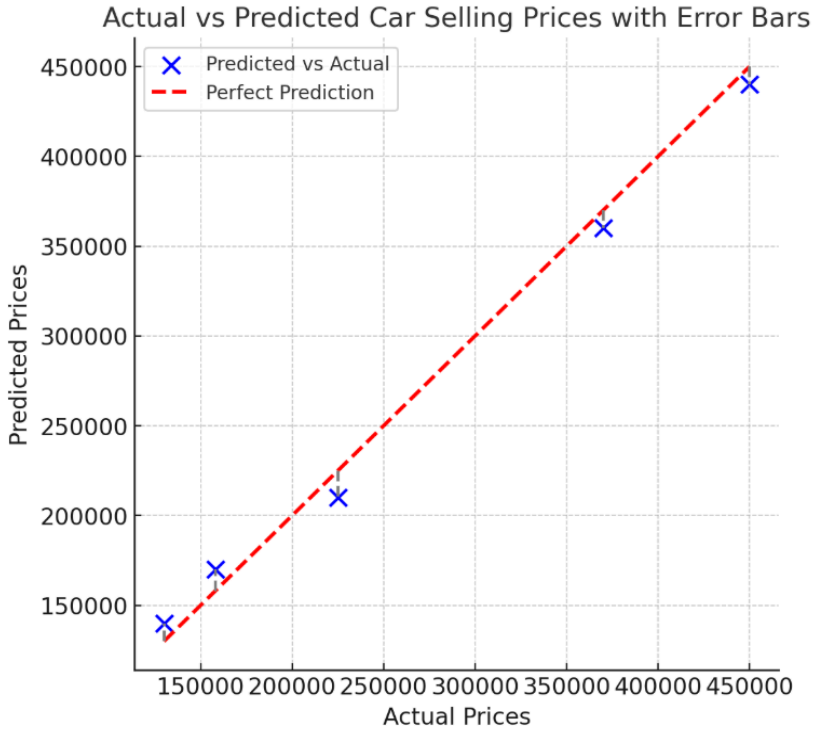
```
from sklearn.metrics import r2_score, mean_squared_error

print("R2 Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

**Visualization**

**Actual vs Predicted Prices**

```
plt.scatter(y_test, y_pred, color="blue")
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Car Selling Prices")
plt.show()
```



- Points close to the diagonal line indicate good predictions.
- Large deviations represent prediction errors.

**Learning Outcome**

- Learned how to apply **Linear Regression** to a real dataset.
- Understood **data preprocessing steps** like handling categorical variables.
- Analyzed the performance of the model using **R2 Score** and **MSE**.
- Built a visualization comparing actual vs predicted prices.