Car Price Prediction Using Linear Regression

1. Project Overview

This project focuses on building a **Car Price Prediction system** using **Linear Regression**. The goal is to predict the selling price of a used car based on multiple features such as car name, company, manufacturing year, kilometers driven, and fuel type.

The project follows a complete **Machine Learning pipeline**, including: - Data loading - Data cleaning and preprocessing - Exploratory analysis - Feature engineering - Model building - Model evaluation - Model serialization (saving the trained model)

---

2. Libraries Used

The following Python libraries are used:

- **pandas** – for data manipulation and analysis
- **numpy** – for numerical computations
- **scikit-learn** – for machine learning models, preprocessing, and evaluation
- **pickle** – for saving the trained model

---

3. Dataset Description

The dataset (`car_data.csv`) contains **892 records** and **6 columns**:

Column Name	Description
name	Car model name
company	Manufacturer name
year	Manufacturing year
Price	Selling price of the car
kms_driven	Total kilometers driven
fuel_type	Type of fuel used

Initially, all columns were of type **object**, which required cleaning and conversion.

---

4. Data Loading

The dataset is loaded using pandas:

```
car = pd.read_csv('sample_data/car_data.csv')
```

Initial inspection is done using: - `head()` - `shape` - `info()`

---

5. Data Cleaning and Preprocessing

Real-world datasets are often messy. The following cleaning steps were applied:

5.1 Year Column

- Removed non-numeric values
- Converted to integer

```
car = car[car['year'].str.isnumeric()]
car['year'] = car['year'].astype(int)
```

5.2 Price Column

- Removed entries with "Ask For Price"
- Removed commas
- Converted to integer

```
car = car[car['Price'] != 'Ask For Price']
car['Price'] = car['Price'].str.replace(',', '').astype(int)
```

5.3 Kilometers Driven

- Extracted numeric value
- Removed commas
- Converted to integer

```
car['kms_driven'] = car['kms_driven'].str.split().str.get(0).str.replace(',',
'')
car = car[car['kms_driven'].str.isnumeric()]
car['kms_driven'] = car['kms_driven'].astype(int)
```

5.4 Fuel Type

- Removed rows with missing fuel type

```
car = car[~car['fuel_type'].isna()]
```

5.5 Car Name Simplification

Only the first three words of the car name were kept to reduce dimensionality:

```
car['name'] = car['name'].str.split().str.slice(0, 3).str.join(' ')
```

5.6 Outlier Removal

Cars priced above **6 million** were removed:

```
car = car[car['Price'] < 6e6]
```

---

6. Cleaned Dataset

After cleaning: - Invalid and missing values were removed - Data types were corrected - The dataset became suitable for machine learning

The cleaned dataset was saved as:

```
car.to_csv('Cleaned_Car_data.csv')
```

---

7. Feature Selection

Independent Variables (X):

- name
- company
- year
- kms_driven
- fuel_type

Dependent Variable (y):

- Price

```
x = car.drop(columns='Price')
y = car['Price']
```

---

8. Train-Test Split

The data was split into **80% training** and **20% testing**:

```
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)
```

---

9. Categorical Encoding

Since Linear Regression cannot handle categorical variables, **One-Hot Encoding** was applied to: - name - company - fuel_type

```
ohe = OneHotEncoder()
ohe.fit(x[['name', 'company', 'fuel_type']])
```

A **Column Transformer** was used to combine encoded categorical features with numerical ones:

```
column_trans = make_column_transformer(
    (OneHotEncoder(categories=ohe.categories_), ['name', 'company',
'fuel_type']),
    remainder='passthrough'
)
```

---

10. Model Building

A **Linear Regression** model was used:

```
lr = LinearRegression()
pipe = make_pipeline(column_trans, lr)
```

The pipeline ensures preprocessing and model training happen together.

---

11. Model Training

```
pipe.fit(X_train, y_train)
```

---

12. Model Evaluation

Predictions were made on the test set:

```
y_pred = pipe.predict(X_test)
```

The performance was measured using **R^2 Score**:

```
r2_score(y_test, y_pred)
```

Initial R^2 Score:

> **0.57**

---

13. Model Optimization (Random State Tuning)

The model was trained **1000 times** using different random states to find the best split:

```
scores = []
for i in range(1000):
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=i)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    scores.append(r2_score(y_test, y_pred))
```

Best result: - **Random State:** 433 - **Best R^2 Score: 0.8457**

This significantly improved model performance.

---

14. Final Model Training

The model was retrained using the best random state:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=433)
pipe.fit(X_train, y_train)
```

---

15. Model Saving

The trained model was saved using `pickle`:

```
pickle.dump(pipe, open('LinearRegressionModel.pkl', 'wb'))
```

This allows reuse of the model without retraining.

---

16. Making Predictions

Example prediction:

```
pipe.predict(pd.DataFrame([
    ['Audi A8', 'Audi', 2019, 100, 'Petrol']],
    columns=['name', 'company', 'year', 'kms_driven', 'fuel_type']
))
```

Predicted Price:

₹1,709,228 (approx.)

---

17. Conclusion

- The project demonstrates a **complete ML workflow**
- Proper data cleaning greatly improved accuracy
- One-hot encoding and pipelines ensured clean preprocessing
- Linear Regression achieved a strong **R^2 score of 0.84**

This project is suitable for: - Academic submissions - Beginner-to-intermediate ML portfolios - Understanding regression-based prediction systems

---

18. Future Improvements

- Use advanced models (Random Forest, XGBoost)
- Add more features (ownership, mileage, location)
- Perform cross-validation
- Deploy as a web application

End of Document