# Predicting Used Car Selling Prices using Linear Regression and K-Nearest Neighbors Regression

## Table of Contents

________________________________________________

## 1. Introduction

Predicting the selling price of a used car is one of the big challenges in the automotive industry. Accurate predictions can benefit both the seller and the buyer, which can give fair pricing depending on several factors such as car age, mileage, brand, and other features. In this project, we will try to predict the selling prices of used cars using two regression models, namely:

- **Linear Regression**
- **K-Nearest Neighbors (KNN) Regression**

---

**2. Dataset Description**

Overview

- DataSet Name: Vehicle Dataset from Cardekho
- Sourced from Kaggle Dataset: Vehicle Dataset from Cardekho
- Total Number of Samples: 4340 used car listings
- Number of Features: 8 columns including the target variable Feature Description

1. name: Name of the car, e.g., "Maruti 800 AC"
2. year: Year of manufacture, e.g., 2007
3. selling_price: Price at which the owner wants to sell the car, in INR
4. km_driven: Total kilometers driven by the car, e.g., 70000
5. fuel: Type of fuel used by the car, i.e., Petrol/Diesel/CNG/LPG/Electric
6. seller_type: Type of seller, i.e., Individual/Dealer/Trustmark Dealer
7. transmission: Type of transmission (Manual/Automatic)
8. owner: Ownership status (First Owner/Second Owner/Third Owner/Fourth & Above Owner/Test Drive Car)

Target Variable

**Data Split**

- Training Set: 80% of the data (3472 samples)
- Testing Set: 20% of the data (868 samples)
- Validation Set: Not explicitly used; cross-validation techniques are applied during model evaluation.

---

## 3. Data Preprocessing

The following steps were performed:

3.1 Handling Missing Values

• Check for Missing Values : No missing values were present in the data.

3.2 Feature Engineering

• Log Transformation of selling_price :

• The distribution of the selling_price variable was highly skewed and required transformation into a more normal form.

• Code Snippet:

```
df['log_price'] = np.log10(df['selling_price'])
```

Creation of feature car_age

• Feature car_age has been created by doing a simple subtraction between current_year (Assumed as 2024) and year of manufacturing.

• Code Snippet:

```
current_year = 2024

df['car_age'] = current_year - df['year']
```

Extract brand from name :

- Extracted the brand of the car from the name column to capture the brand effect on the price.
- Code Snippet:

```
df['brand'] = df['name'].str.split(' ').str[0]
```

3.3 Encoding Categorical Variables

- Used Label Encoding for the following categorical features:
- owner
- transmission
- seller_type
- fuel
- brand
- Note: Label Encoding is used here for simplicity. However, One-Hot Encoding could be used to avoid giving a categorical variable ordinal relationships.
- Code Snippet:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

categorical_features = ['fuel', 'seller_type', 'transmission', 'owner', 'brand']

for feature in categorical_features:

   df[feature] = le.fit_transform(df[feature])
```

---

## 4. Exploratory Data Analysis

Understand the distribution of data and the relationship between variables.

4.1 Distribution of Selling Price

- Original selling_price was right-skewed-a few cars had a very high price
- After Log Transformation:
- log_price after transformation is close to a normal distribution.

4.2 Correlation Analysis

- Finding the correlation matrix between features and the target variable.

**5. Model Implementation**

Data Split

- Split the data into an 80-20 split for training and testing sets, respectively.
- Code Snippet:

```
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)
```

5.1 Linear Regression

- Selected Model: Linear Regression is a basic algorithm for the prediction of continuous outcomes.
- Implementation:
- LinearRegression from sklearn was used.
- Code Snippet:

```
from sklearn.linear_model import LinearRegression

lr_model = LinearRegression()

lr_model.fit(X_train, y_train)
```

5.2 K-Nearest Neighbors Regression

- Selected Model: KNN Regression is a non-parametric method that predicts the target by averaging the values of the nearest neighbors.
- Implementation:
- KNeighborsRegressor with n_neighbors=5 was used.
- Code Snippet:

```
from sklearn.neighbors import KNeighborsRegressor

knn_model = KNeighborsRegressor(n_neighbors=5)

knn_model.fit(X_train, y_train)
```

## 6. Model Evaluation

Some metrics used to evaluate regression models:

- Mean Absolute Error, MAE
- Mean Squared Error, MSE
- Root Mean Square Error, RMSE
- R-squared R2

6.1 Evaluate Linear Regression

- Metrics

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
y_pred_lr = lr_model.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mse_lr)
r2_lr = r2_score(y_test, y_pred_lr)
```

- MSE: 0.04075
- MAE: 0.15637
- RMSE: 0.20187
- R-squared ($R^2$): 0.67196

---

## Conclusion

In this project, we implemented and compared Linear Regression and K-Nearest Neighbors Regression models to predict the selling prices of used cars.

- Linear Regression slightly outperformed KNN Regression in predicting the log-transformed selling prices.
- Brand, car age, and fuel type are significant predictors of selling price.