MACHINE LEARNING PROJECT 1

by Taibah Shahbaz
2023-BSAI-024 4th A

Project 1: Predicting Selling Price of Used Cars Introduction

- **Goal:** Use regression techniques to estimate car prices
- **Algorithm:** Linear Regression
- **Dataset:**

Source: car_data.csv (Kaggle)

Total Records: 301

Features: 9(Year, Price, Kms Driven, Fuel Type, etc.)

Target Variable: Selling_Price

Preprocessing Steps

- **Library Imports:** pandas, numpy, seaborn, sklearn
- **Initial Data Exploration:** .head(), .info(), .describe()
- **Data Cleaning:** Remove Car_Name, handle duplicates
- **Outlier Removal:** Boxplots + IQR method
- **Encoding & Scaling:** LabelEncoder for categorical, StandardScaler for numeric
- **Dimensionality Reductio**: PCA to retain 95% variance

Preprocessing Steps (project 1)

Reading data

```
[9]: data = pd.read_csv('car data.csv')
     data.head()
```

[9]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

```
[10]: data.tail()
```

[10]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
296	city	2016	9.50	11.6	33988	Diesel	Dealer	Manual	0
297	brio	2015	4.00	5.9	60000	Petrol	Dealer	Manual	0
298	city	2009	3.35	11.0	87934	Petrol	Dealer	Manual	0
299	city	2017	11.50	12.5	9000	Diesel	Dealer	Manual	0
300	brio	2016	5.30	5.9	5464	Petrol	Dealer	Manual	0

Preprocessing Steps (project 1)

```
[11]: data.shape

[11]: (301, 9)

[12]: data.sample()
```

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
113	Mahindra Mojo XT300	2016	1.15	1.4	35000	Petrol	Individual	Manual	0

```
[13]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Selling_type   301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

```
[14]: data.describe()
```

[14]:

	Year	Selling_Price	Present_Price	Driven_kms	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.642584	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

Checking missing values

```
[16]: data.isnull().sum()

[16]: Car_Name         0
      Year             0
      Selling_Price    0
      Present_Price    0
      Driven_kms       0
      Fuel_Type        0
      Selling_type     0
      Transmission     0
      Owner            0
      dtype: int64
```

Checking unique values

```
[18]: data.nunique()

[18]: Car_Name         98
      Year             16
      Selling_Price    156
      Present_Price    148
      Driven_kms       206
      Fuel_Type          3
      Selling_type       2
      Transmission       2
      Owner              3
      dtype: int64
```

Modeling (project 1)

- **Model Used:** Linear Regression
- **Library:** sklearn.linear_model.LinearRegression
- **Train-Test Split:** 80:20 ratio
- **Training:** Model trained on scaled and PCA-reduced features
- **Tools:** .fit(), .predict()

Modeling (project 1)

Data Splitting

[41]:
```
X_selected = X
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)

print("Training Set Shape:", X_train.shape)
print("Test Set Shape:", X_test.shape)
```

```
Training Set Shape: (208, 8)
Test Set Shape: (53, 8)
```

Train the Linear Regression Model

[43]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

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

[43]:
```
LinearRegression
LinearRegression()
```

Evaluation (project 1)

Metrics Used:

- **MAE** (Mean Absolute Error)
- **MSE** (Mean Squared Error)
- **RMSE** (Root Mean Squared Error)
- **R^2 Score**

Visual Evaluation:

- Actual vs. Predicted plot
- Residual Plot

Evaluation (project 1):

Make Predictions

```
[76]: y_pred = model.predict(X_test)
```

Compare Actual vs Predicted Values

```
[79]: df_preds = pd.DataFrame({
          'Actual': y_test.squeeze(),
          'Predicted': y_pred.squeeze()
      })

      print("Actual vs Predicted:")
      print(df_preds.head(10))
```

```
Actual vs Predicted:
      Actual  Predicted
0   5.037904   4.498754
1   8.865488   9.152736
2   7.396518   8.466439
3   7.065746   7.852142
4   6.343712   5.591731
5   6.942462   6.607172
6   5.757981   5.778528
7  11.044395   8.975125
8   5.038909   4.258159
9   6.334288   6.239831
```

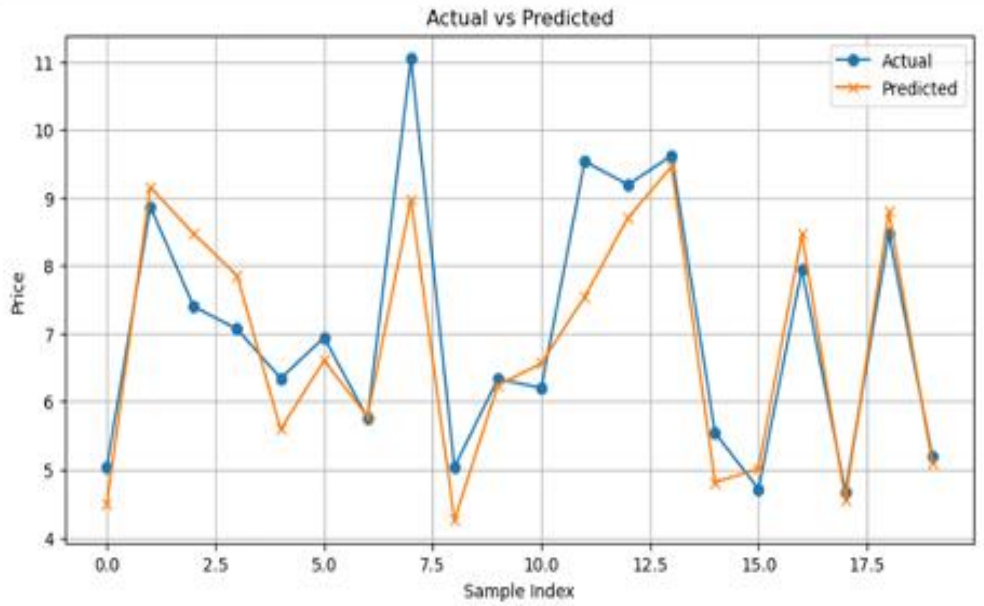
Evaluate the Model

```
[86]: print("\nModel Evaluation:")
      print("MAE:", mean_absolute_error(y_test, y_pred))
      print("MSE:", mean_squared_error(y_test, y_pred))
      print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
      print("R² Score:", r2_score(y_test, y_pred))
```

```
Model Evaluation:
MAE: 0.5913425779189777
MSE: 0.6536995137170021
RMSE: 0.8085168605026132
R² Score: 0.8072059636181392
```

Plot Actual vs Predicted

```
[93]: plt.figure(figsize=(10, 5))
      plt.plot(y_test, label='Actual', marker='o')
      plt.plot(y_pred, label='Predicted', marker='x')
      plt.title('Actual vs Predicted')
      plt.xlabel('Sample Index')
      plt.ylabel('Price')
      plt.legend()
      plt.grid(True)
      plt.show()
```



Key Findings

- **Linear Regression** successfully modeled price predictions with moderate accuracy
- **Data cleaning, feature scaling, and PCA** significantly enhanced model performance
- **Visual comparisons** of predicted vs. actual prices validated the model's effectiveness
- **Practical Insight:** Helps buyers/sellers estimate car value in the used vehicle market

Conclusion

- **Linear Regression** effectively predicts used car prices with moderate accuracy
- **Data Cleaning, Scaling, and PCA** significantly improved model performance
- **Business Insight:** Helps sellers and buyers estimate fair market value

Thank you