Car Price Prediction with Linear Regression

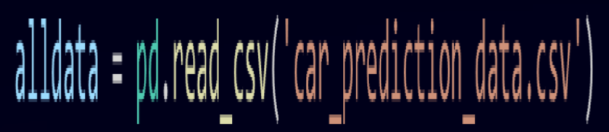
This presentation explores a Python-based linear regression model. It predicts car prices using the scikit-learn library. Key steps include data preprocessing, model training, and performance evaluation.

by Ibrahim Abdel Nasser

Data Loading and Preprocessing

Loading the Dataset

First, import **pandas** to read the **car_prediction_data.csv** file.

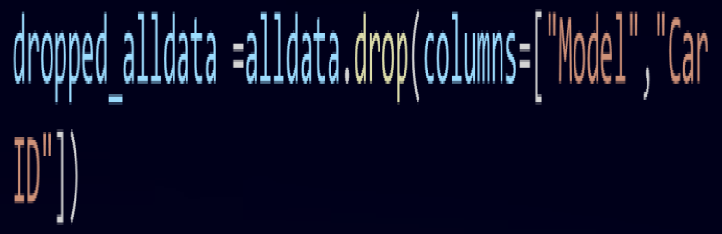


```
alldata = pd.read_csv('car_prediction_data.csv')
```

https://www.kaggle.com/datasets/zafarali27/car-price-prediction

Rename and Drop Data

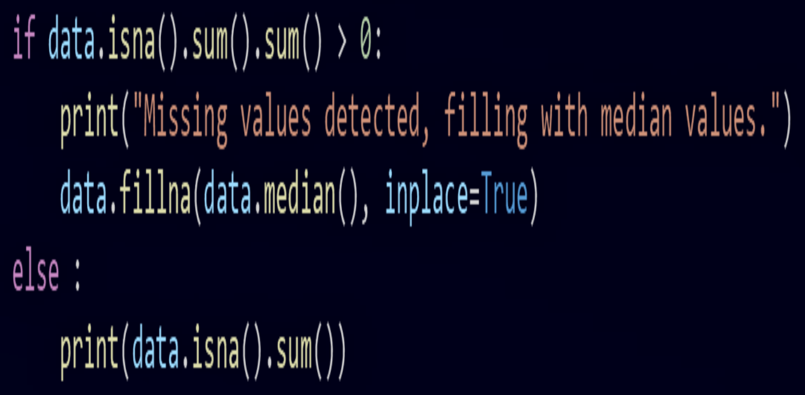
Drop unneeded data (Car ID). Rename some of data (from Mileage to Kilometers).



```
dropped_alldata =alldata.drop(columns=["Model","Car
ID"])
```

Handling Missing Data

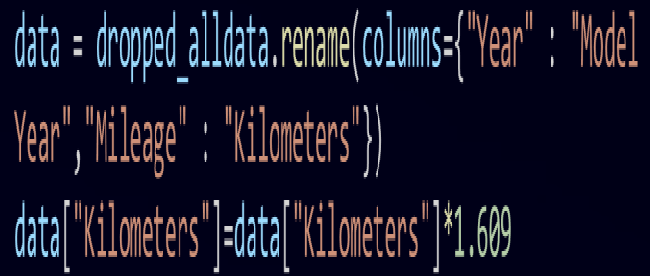
Then, fill missing values with the median. This ensures data completeness.



```
if data.isna().sum().sum() > 0:
    print("Missing values detected, filling with median values.")
    data.fillna(data.median(), inplace=True)
else :
    print(data.isna().sum())
```

Feature Engineering

Convert "Mileage" from miles to kilometers. Rename columns to improve readability.



```
data = dropped_alldata.rename(columns={"Year" : "Model
Year","Mileage" : "Kilometers"})
data["Kilometers"]=data["Kilometers"]*1.609
```

Exploratory Data Analysis

1 Kilometers vs. Price

A scatter plot visualizes the relationship. Price typically decreases as mileage increases.

```
plt.figure(figsize=(10, 6))
plt.scatter(data["Kilometers"],data["Price"],color='blue',
marker='o',alpha=0.5)
plt.xlabel("Kilometers (KM)")
plt.ylabel("Price (EGP)")
plt.title("Mileage Vs Price")
plt.show()
```

2 Average Price per Brand

A bar chart displays median car prices. Each bar represents the median price per brand.

```
plt.bar(data["Brand"], data["Price"].median(),
color='green')
plt.title("Average Car Price per Brand")
plt.xlabel("Model")
plt.ylabel("Price")
plt.show()
```

Data Splitting

1

Train-Test Split

train_test_split divides data.

2

80/20 Ratio

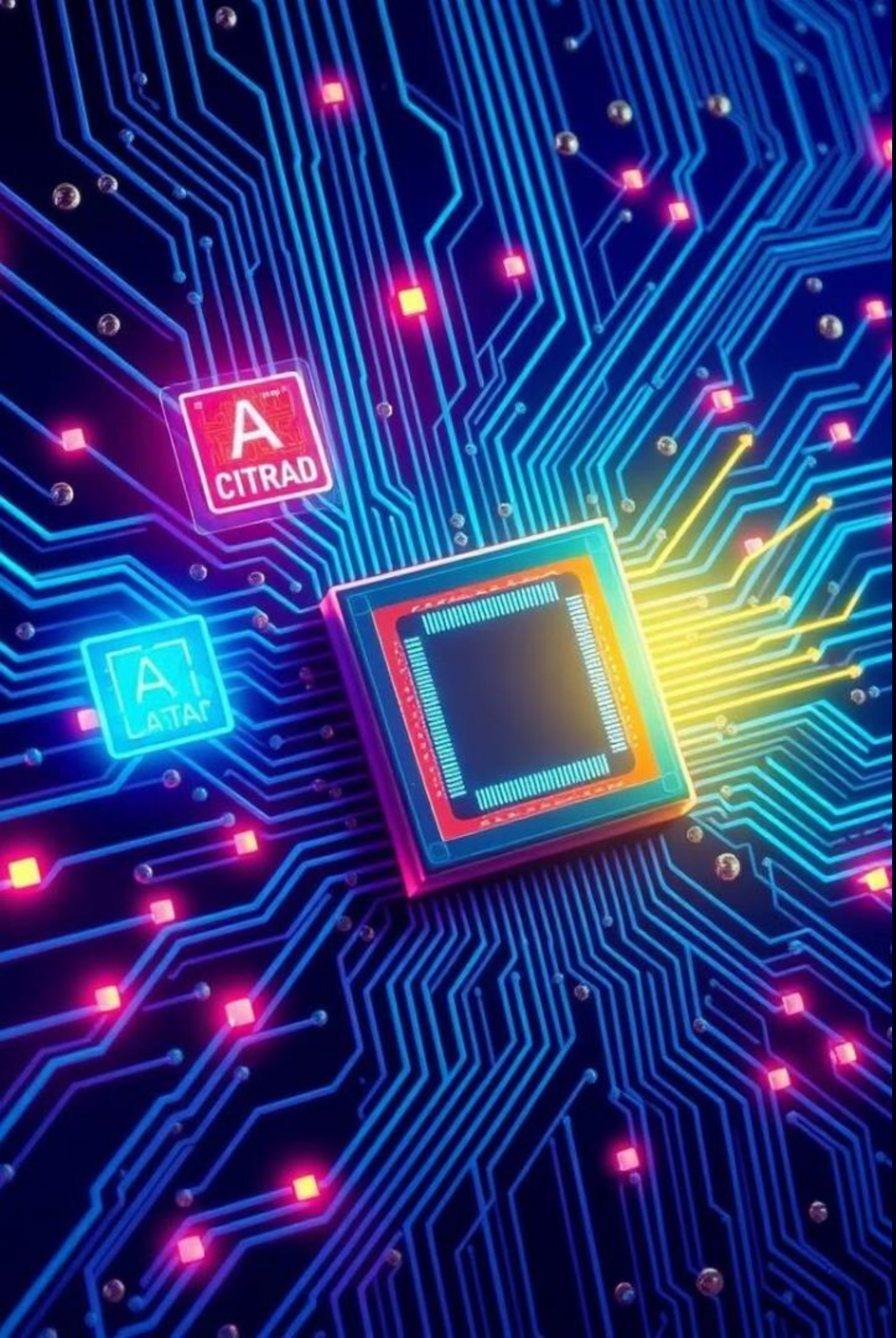
80% of data is for training and 20% is for testing.

3

Random State

random_state=42 ensures reproducibility.

```
X = data.drop(columns=["Brand"])
y = data["Brand"]
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)
```



Feature Selection and Encoding

Categorical columns include Brand, Fuel Type, Transmission, Condition, and Engine Size.

```
categorical_cols = ["Brand", "Fuel Type","Transmission","Condition","Engine Size"]
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le
```

LabelEncoder from scikit-learn converts categories to numerical labels.

Features are selected for model training after encoding is complete.

Model Training

1

Linear Regression

A **LinearRegression** model is initialized from **sklearn.linear_model**.

2

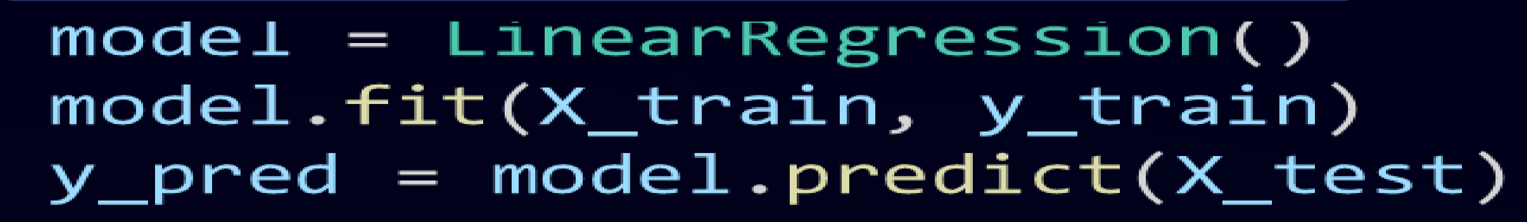
Model Fitting

The model learns from training data. It finds coefficients.

3

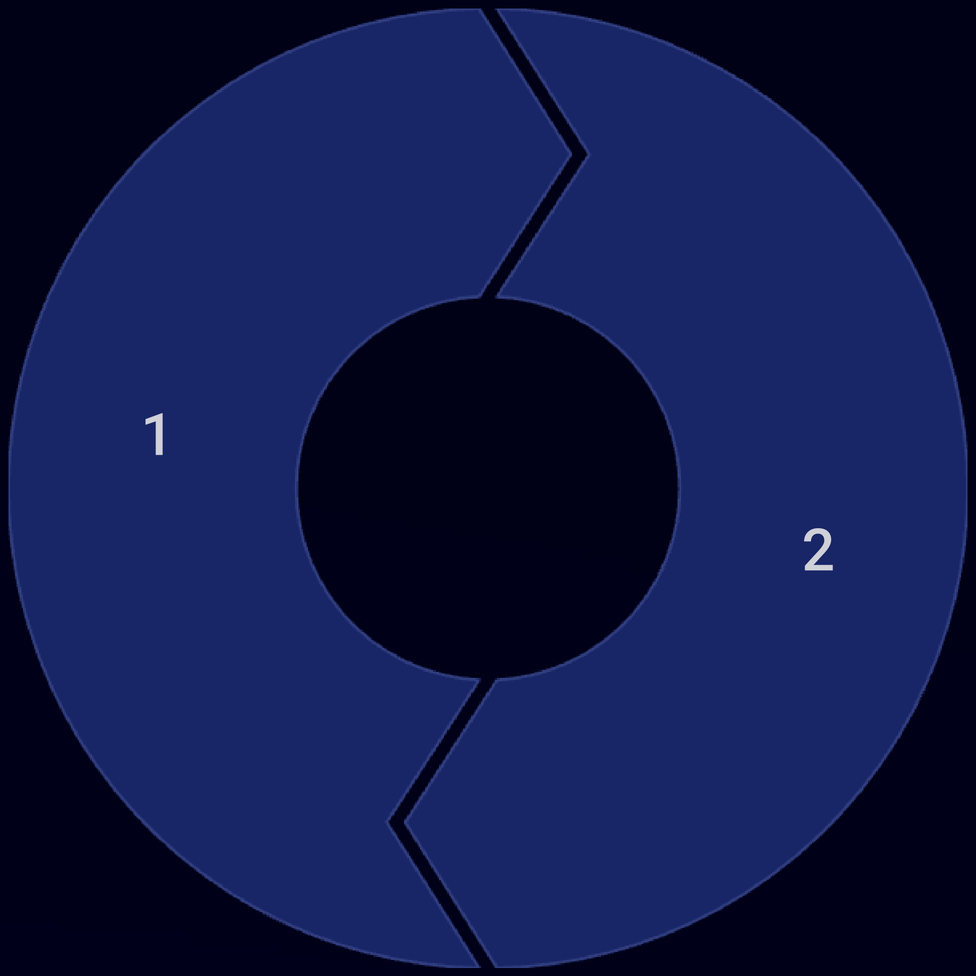
Prediction

The model predicts on the test set.



```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Model Evaluation

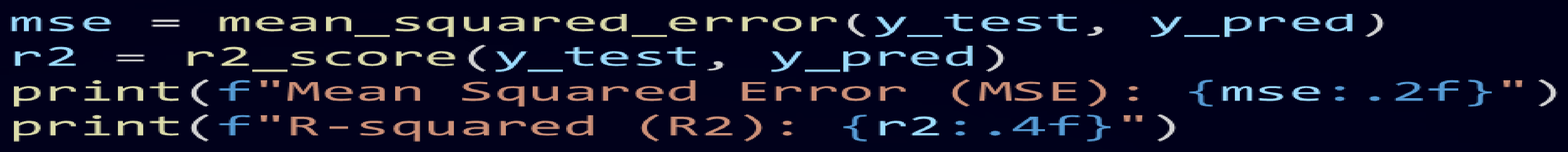


Mean Squared Error

MSE measures prediction accuracy.

R-squared

R2 indicates model fit. Ranges from 0 to 1.



```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.4f}")
```

Summary and Next Steps

Key Takeaways

Linear regression predicts car prices. **Scikit-learn** simplifies the process.

Further Exploration

Improve the model with more features. Consider other regression models.