🚗 Car Price Prediction Using Machine Learning

📄 Project Overview

- **Project Name:** Vehicle Price Prediction System
- **Developer:** Agba Daniel
- **Role:** AI/ML Practitioner | Data Scientist
- **Certifications:**
  - Machine Learning Specialization (Coursera)
  - Deep Learning Specialization (Coursera)

🎯 Project Objective

To build a reliable and optimized machine learning model that **predicts car prices in AUD** based on technical and categorical features of vehicles, demonstrating:

- End-to-end ML pipeline development
- Feature selection and engineering
- Hyperparameter tuning via cross-validation
- Clear evaluation using regression metrics

🛠️ Technical Showcase

This project showcases my capability in:
✅ Advanced Data Preprocessing & Feature Engineering
✅ Ensemble Learning with XGBoost
✅ Cross-Validated Hyperparameter Tuning
✅ Robust Model Evaluation Techniques
✅ Deployment-ready ML Workflow

🧠 Skills Demonstrated

Technical: Python, Pandas, Scikit-learn, XGBoost, Matplotlib, Seaborn
Methodological: Grid Search CV, Feature Selection (`SelectKBest`), Model Interpretation
Analytical: Error Analysis (MAE, RMSE, R^2), Feature Impact Understanding, Business Insight Alignment

---

📈 Model Performance Summary

Training Results

- **R^2 Score:** 0.7646
- **MAE:** $7,965.49
- **RMSE:** $13,876.96

Test Results

- **R^2 Score:** 0.6757
- **MAE:** $8,313.15
- **RMSE:** $17,091.64

> The model explains approximately **68% of the variance** in vehicle prices, with an average deviation of **$8.3K**, showing strong real-world applicability.

🧑‍💼 Professional Profile

Agba Daniel
AI/ML Specialist with experience in:

- Designing regression models with strong generalization
- Implementing scalable and reusable ML workflows
- Converting business requirements into machine learning solutions

Connect with me:
📱 WhatsApp: +2347080157838
📧 Email: agbadaniel13@gmail.com

In [119…

```
# =======================
# 1. Data Loading
# =======================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
data = pd.read_csv("Australian Vehicle Prices.csv")

# Initial inspection
print("Data Shape:", data.shape)
print("\nFirst 5 Rows:")
display(data.head())
```

```
Data Shape: (16734, 19)

First 5 Rows:
```

	Brand	Year	Model	Car/Suv	Title	UsedOrNew	Transmission	Engine	Dri
0	Ssangyong	2022.0	Rexton	Sutherland Isuzu Ute	2022 Ssangyong Rexton Ultimate (awd)	DEMO	Automatic	4 cyl, 2.2 L	
1	MG	2022.0	MG3	Hatchback	2022 MG MG3 Auto Excite (with Navigation)	USED	Automatic	4 cyl, 1.5 L	
2	BMW	2022.0	430I	Coupe	2022 BMW 430I M Sport	USED	Automatic	4 cyl, 2 L	
3	Mercedes-Benz	2011.0	E500	Coupe	2011 Mercedes-Benz E500 Elegance	USED	Automatic	8 cyl, 5.5 L	
4	Renault	2022.0	Arkana	SUV	2022 Renault Arkana Intens	USED	Automatic	4 cyl, 1.3 L	

In [95]:

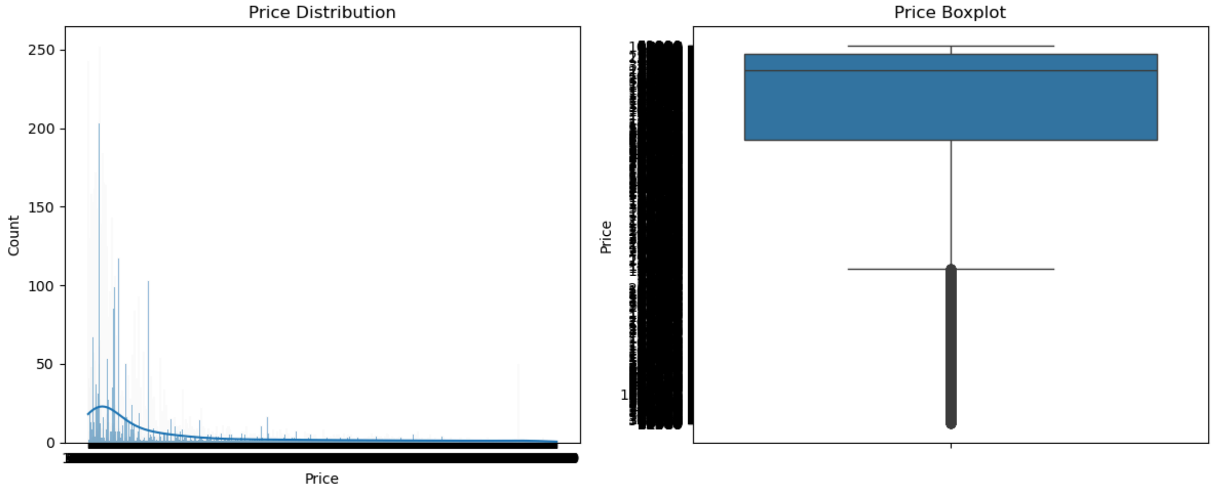
```
# ======================
# 2. Exploratory Data Analysis (EDA)
# ======================
# 2.1 Basic Stats
print("\n=== Dataset Info ===")
data.info()
print("\n=== Descriptive Statistics ===")
display(data.describe(include='all'))
print("\n=== Missing Values ===")
print(data.isnull().sum())

# 2.2 Target Variable Analysis
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(data['Price'], kde=True)
plt.title('Price Distribution')

plt.subplot(1, 2, 2)
sns.boxplot(y=data['Price'])
plt.title('Price Boxplot')
plt.tight_layout()
plt.show()

# 2.3 Categorical Features
```

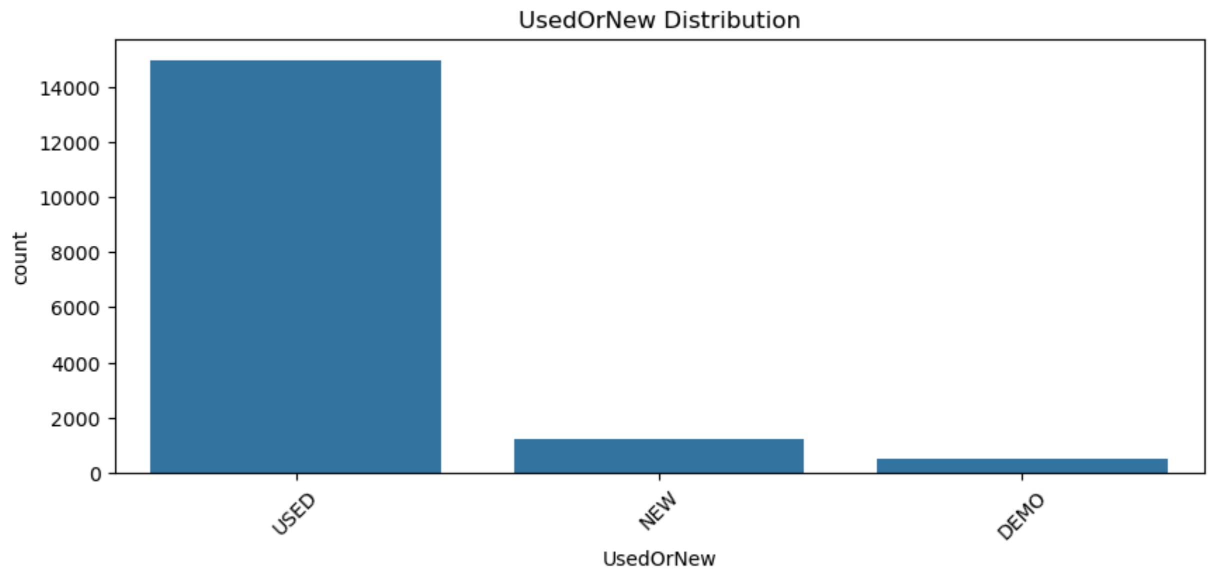
```
cat_cols = data.select_dtypes(include=['object']).columns
for col in cat_cols:
    if data[col].nunique() < 15:
        plt.figure(figsize=(10, 4))
        sns.countplot(data=data, x=col, order=data[col].value_counts().index)
        plt.title(f'{col} Distribution')
        plt.xticks(rotation=45)
        plt.show()

# 2.4 Numerical Features
num_cols = data.select_dtypes(include=['int64', 'float64']).columns.difference(['Pr
for col in num_cols:
    plt.figure(figsize=(12, 4))
    sns.histplot(data[col], kde=True)
    plt.title(f'{col} Distribution')
    plt.show()

# 2.5 Correlation Analysis
if len(num_cols) > 1:
    plt.figure(figsize=(12, 8))
    sns.heatmap(data[num_cols].corr(), annot=True, cmap='coolwarm', center=0)
    plt.title('Numerical Features Correlation')
    plt.show()
```

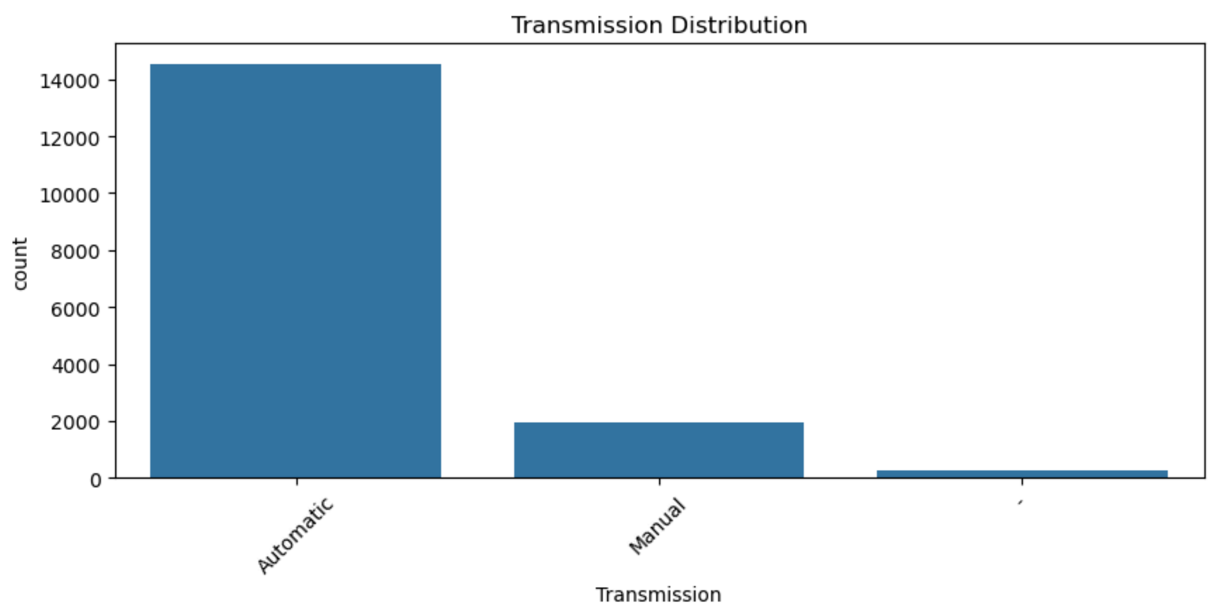
```
=== Dataset Info ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16734 entries, 0 to 16733
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Brand              16733 non-null  object 
 1   Year               16733 non-null  float64
 2   Model              16733 non-null  object 
 3   Car/Suv            16706 non-null  object 
 4   Title              16733 non-null  object 
 5   UsedOrNew          16733 non-null  object 
 6   Transmission       16733 non-null  object 
 7   Engine             16733 non-null  object 
 8   DriveType          16733 non-null  object 
 9   FuelType           16733 non-null  object 
 10  FuelConsumption    16733 non-null  object 
 11  Kilometres         16733 non-null  object 
 12  ColourExtInt       16733 non-null  object 
 13  Location           16284 non-null  object 
 14  CylindersinEngine  16733 non-null  object 
 15  BodyType           16452 non-null  object 
 16  Doors              15130 non-null  object 
 17  Seats              15029 non-null  object 
 18  Price              16731 non-null  object 
dtypes: float64(1), object(18)
memory usage: 2.4+ MB

=== Descriptive Statistics ===
```

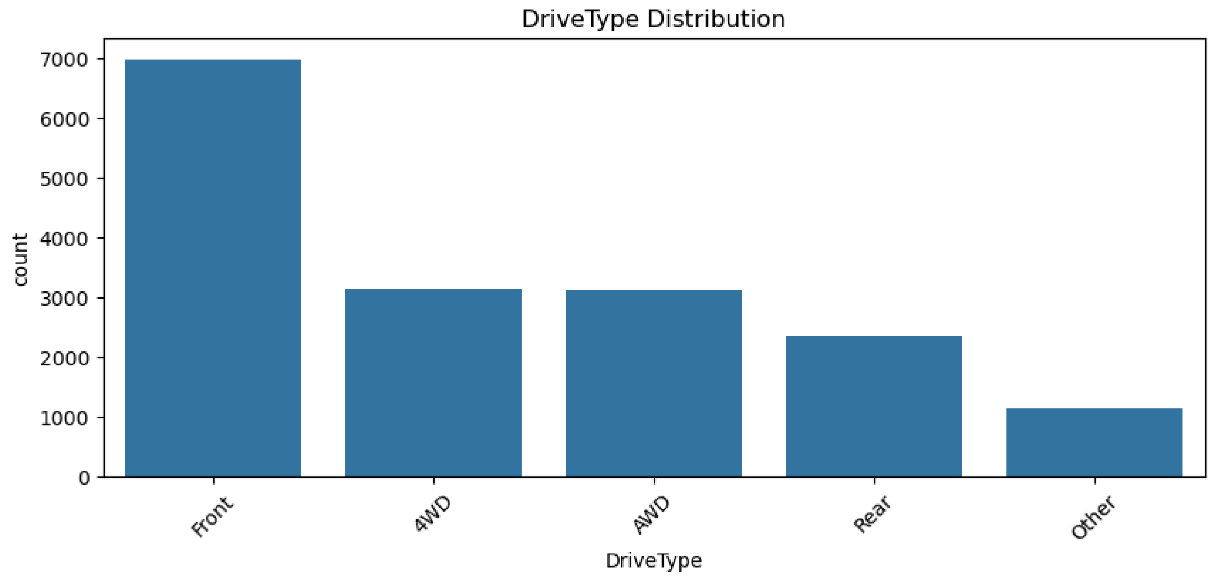
	Brand	Year	Model	Car/Suv	Title	UsedOrNew	Transmission	Engine
count	16733	16733.000000	16733	16706	16733	16733	16733	16733
unique	76	NaN	781	618	8804	3	3	106
top	Toyota	NaN	Hilux	SUV	2019 Hyundai I30 Active	USED	Automatic	4 cyl, 2 L
freq	2784	NaN	430	5921	60	14994	14530	3950
mean	NaN	2016.229248	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	5.247705	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	1940.000000	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	2013.000000	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	2017.000000	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	2020.000000	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	2023.000000	NaN	NaN	NaN	NaN	NaN	NaN

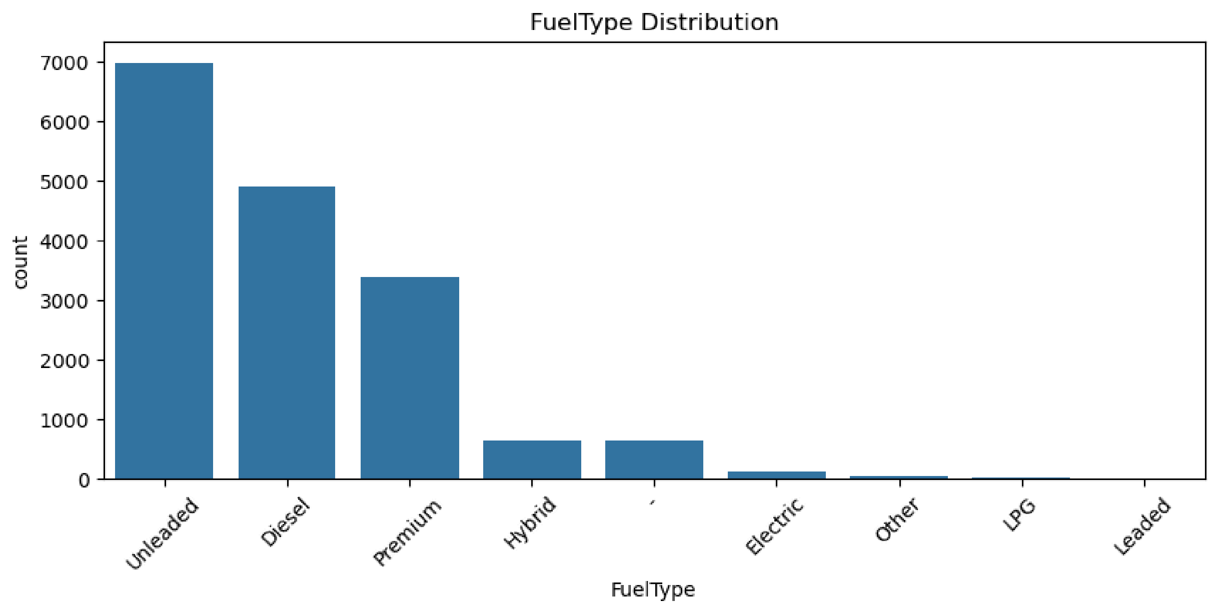
```
=== Missing Values ===
Brand                  1
Year                   1
Model                  1
Car/Suv               28
Title                  1
UsedOrNew              1
Transmission           1
Engine                 1
DriveType              1
FuelType               1
FuelConsumption        1
Kilometres             1
ColourExtInt           1
Location             450
CylindersinEngine      1
BodyType             282
Doors               1604
Seats               1705
Price                  3
dtype: int64
```

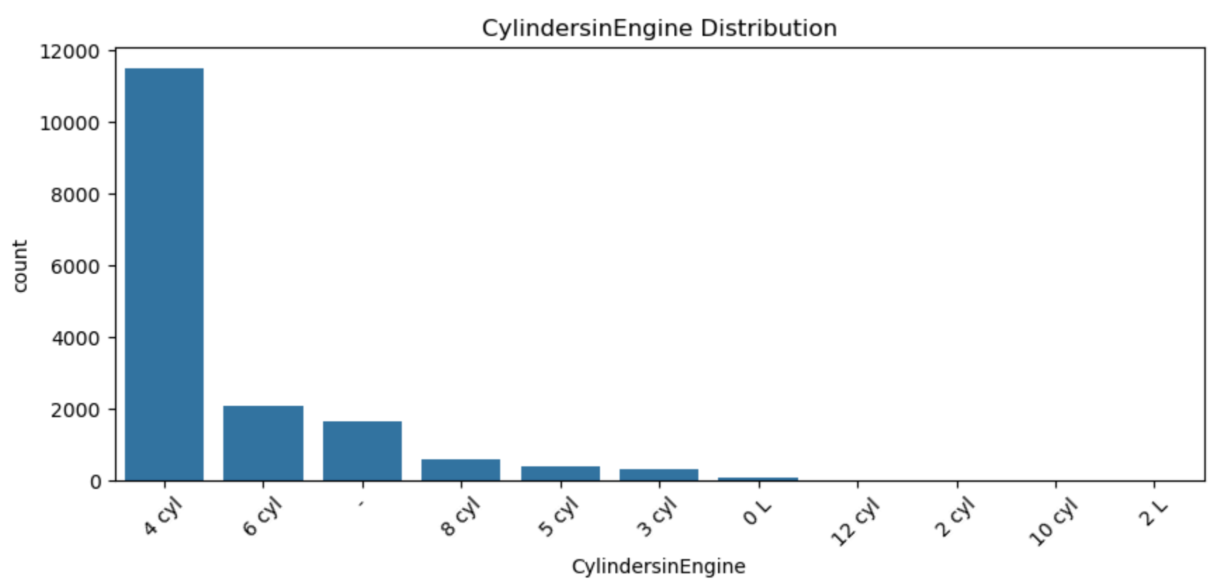

Price Distribution
Count
250
200
150
100
50
0
Price
Price Boxplot
Price

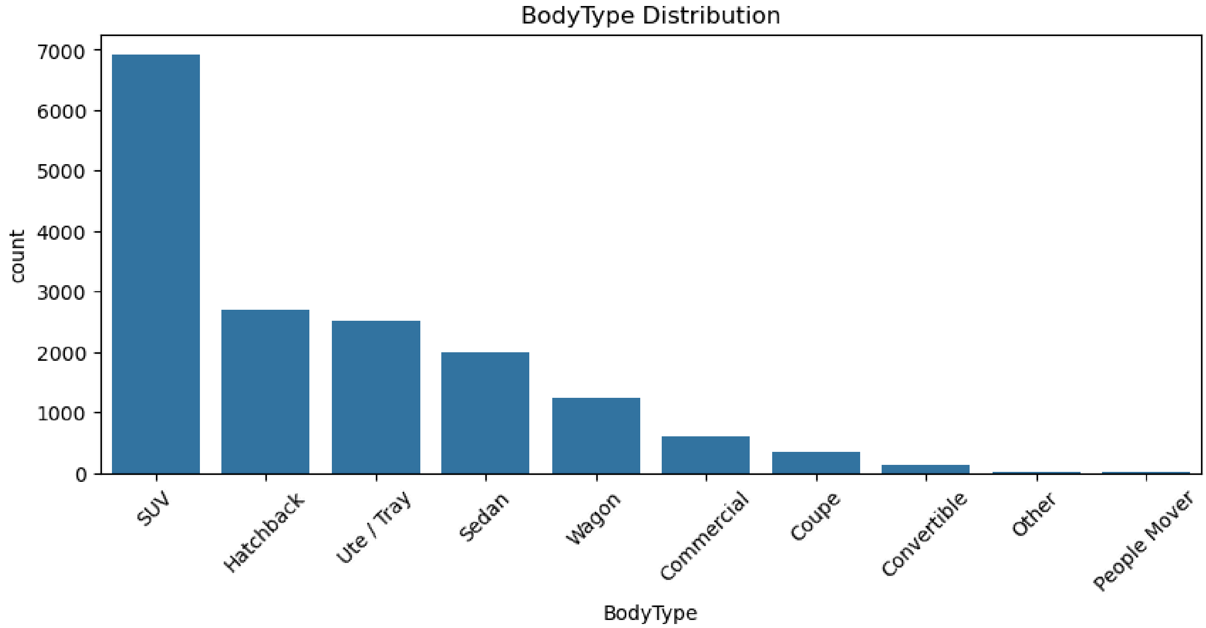

UsedOrNew Distribution
count
14000
12000
10000
8000
6000
4000
2000
0
USED
NEW
DEMO
UsedOrNew

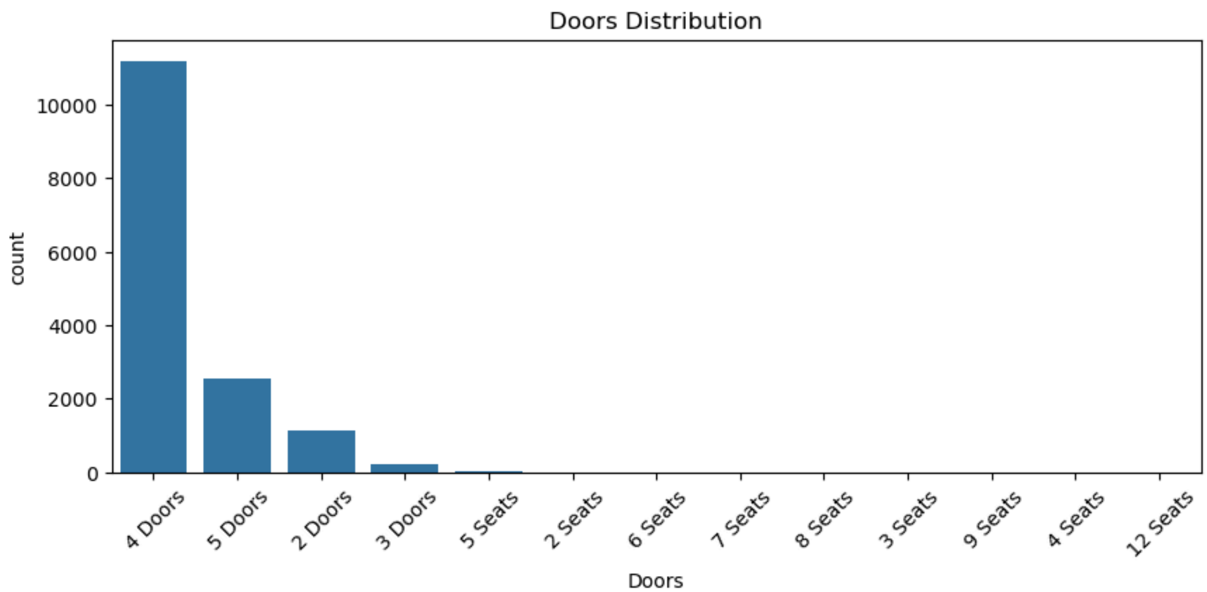

Transmission Distribution
count
14000
12000
10000
8000
6000
4000
2000
0
Automatic
Manual
-
Transmission

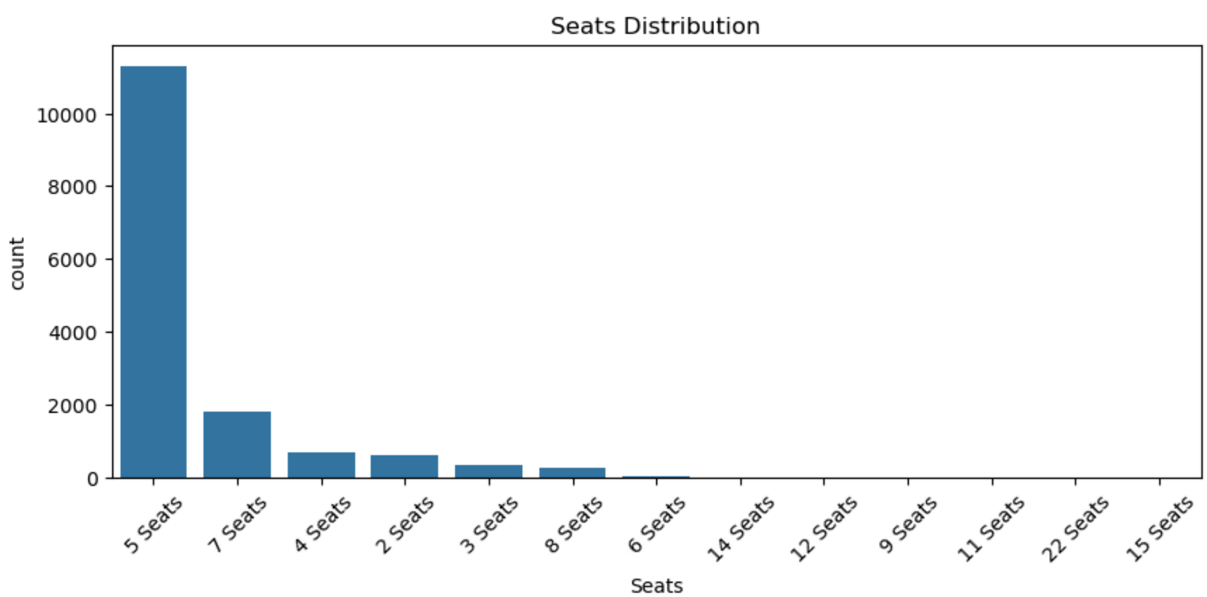

DriveType Distribution
count
7000
6000
5000
4000
3000
2000
1000
0
Front
4WD
AWD
Rear
Other
DriveType

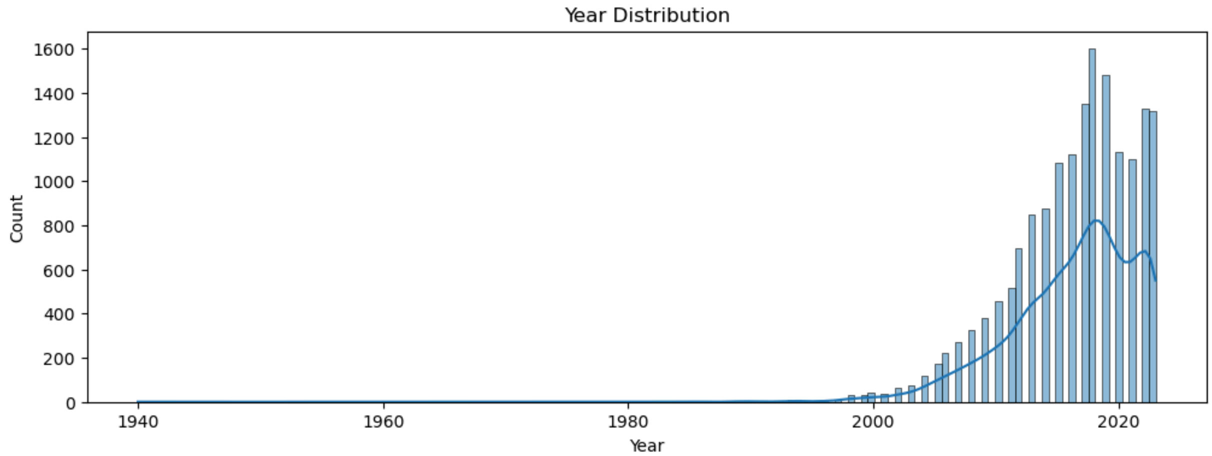

FuelType Distribution
count
7000
6000
5000
4000
3000
2000
1000
0
Unleaded
Diesel
Premium
Hybrid
-
Electric
Other
LPG
Leaded
FuelType


CylindersinEngine Distribution
count
12000
10000
8000
6000
4000
2000
0
4 cyl
6 cyl
-
8 cyl
5 cyl
3 cyl
0 L
12 cyl
2 cyl
10 cyl
2 L
CylindersinEngine


BodyType Distribution
count
0
1000
2000
3000
4000
5000
6000
7000
SUV
Hatchback
Ute / Tray
Sedan
Wagon
Commercial
Coupe
Convertible
Other
People Mover
BodyType


Doors Distribution
count
0
2000
4000
6000
8000
10000
4 Doors
5 Doors
2 Doors
3 Doors
5 Seats
2 Seats
6 Seats
7 Seats
8 Seats
3 Seats
9 Seats
4 Seats
12 Seats
Doors


Seats Distribution
count
0
2000
4000
6000
8000
10000
5 Seats
7 Seats
4 Seats
2 Seats
3 Seats
8 Seats
6 Seats
14 Seats
12 Seats
9 Seats
11 Seats
22 Seats
15 Seats
Seats



In [97]:

```
# ======================
# 3. Data Cleaning
# ======================
# Drop rows with missing values
data = data.dropna()

# Clean text data
for col in data.select_dtypes(include='object'):
    data[col] = data[col].str.strip()
```

In [99]:

```
# ======================
# 4. Feature Engineering
# ======================
from datetime import datetime

# Create new feature
current_year = datetime.now().year
data['Car_Age'] = current_year - data['Year']
data = data.drop(columns=['Year'])
```

In [103…

```
# ======================
# 2. Data Cleaning (Updated)
# ======================
import numpy as np

# 2.1 Handle Price column
print("Original Price values:", data['Price'].unique()[:10])  # Show first 10 uniqu

# Convert 'POA' to NaN and remove rows with invalid prices
data['Price'] = pd.to_numeric(data['Price'], errors='coerce')

# 2.2 Drop rows with missing values (including the newly created NaNs from 'POA')
print(f"\nRows before cleaning: {len(data)}")
data = data.dropna()
print(f"Rows after cleaning: {len(data)}")

# 2.3 Verify conversion
print("\nPrice after cleaning:")
print(data['Price'].describe())
```

```
Original Price values: ['51990' '19990' '108988' '32990' '34990' '62280' '2995' '248
88' '17900'
 '41999']

Rows before cleaning: 14586
Rows after cleaning: 14551

Price after cleaning:
count     14551.000000
mean      34980.246306
std       29031.113506
min          88.000000
25%       18990.000000
50%       28990.000000
75%       42755.000000
max      649880.000000
Name: Price, dtype: float64
```

In [105…

```
# ======================
# 6. Encoding
# ======================
# One-hot encoding (preserving train-test separation)
X_train_encoded = pd.get_dummies(X_train)
X_test_encoded = pd.get_dummies(X_test)

# Align columns
X_test_encoded = X_test_encoded.reindex(columns=X_train_encoded.columns, fill_value
```

In [107…

```
# ======================
# 7. Feature Selection
# ======================
from sklearn.feature_selection import SelectKBest, f_regression

selector = SelectKBest(f_regression, k=20)
X_train_selected = selector.fit_transform(X_train_encoded, y_train)
X_test_selected = selector.transform(X_test_encoded)

selected_features = X_train_encoded.columns[selector.get_support()]
print("Selected Features:")
print(selected_features.tolist())
```

```
Selected Features:
['Car_Age', 'Brand_Ferrari', 'Brand_Lamborghini', 'Brand_Porsche', 'Model_Aventado
r', 'Model_GTC4', 'Car/Suv_Hatchback', 'Title_2012 Lamborghini Aventador', 'Title_20
19 Ferrari GTC4 Lusso (awd)', 'UsedOrNew_USED', 'Engine_12 cyl, 6.3 L', 'Engine_12 c
yl, 6.5 L', 'Engine_8 cyl, 4 L', 'DriveType_AWD', 'DriveType_Front', 'FuelType_Unlea
ded', 'Kilometres_11480', 'CylindersinEngine_12 cyl', 'CylindersinEngine_4 cyl', 'Cy
lindersinEngine_8 cyl']
```

In [113…

```
# ======================
# 8. Model Training
# ======================
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV

param_grid = {
```

```
    'n_estimators': [100, 200],
    'learning_rate': [0.05, 0.1],
    'max_depth': [3, 5],
    'subsample': [0.8, 0.9],
    'colsample_bytree': [0.8, 0.9],
    'reg_alpha': [0, 0.1, 1],  # L1 regularization
    'reg_lambda': [0, 0.1, 1], # L2 regularization
    'gamma': [0, 0.1]          # Min loss reduction for splits
}

grid_search = GridSearchCV(
    XGBRegressor(random_state=42),
    param_grid,
    scoring='r2',
    cv=5,
    verbose=2,
    n_jobs=-1
)

grid_search.fit(X_train_selected, y_train)

best_model = grid_search.best_estimator_
print("\nBest Parameters:", grid_search.best_params_)
```

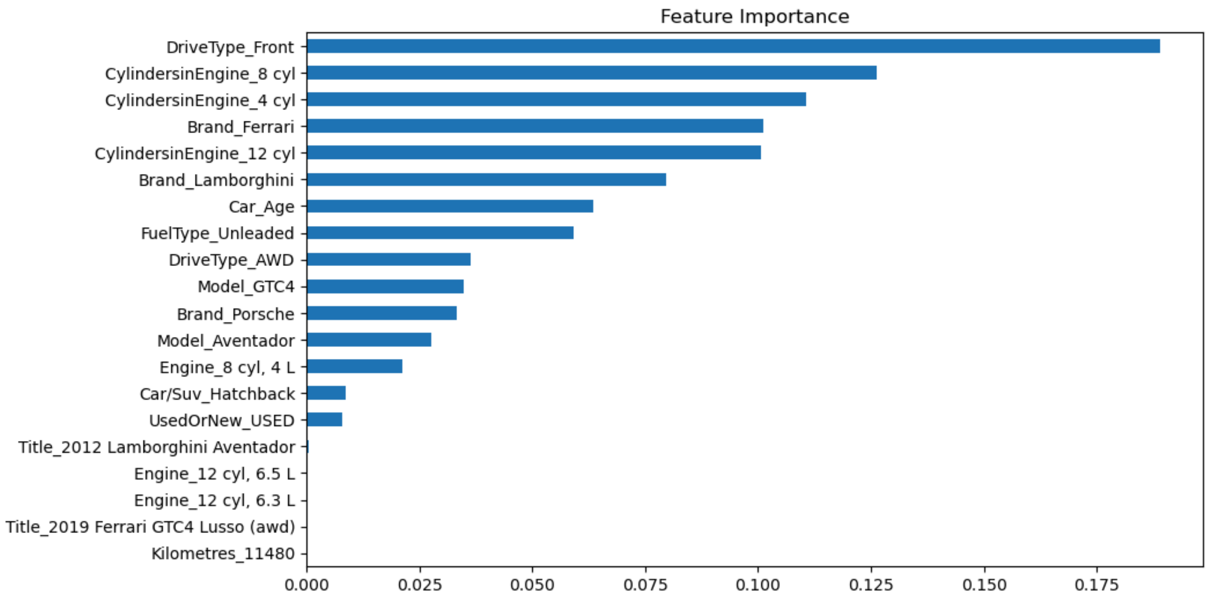
```
Fitting 5 folds for each of 576 candidates, totalling 2880 fits

Best Parameters: {'colsample_bytree': 0.8, 'gamma': 0, 'learning_rate': 0.1, 'max_de
pth': 3, 'n_estimators': 200, 'reg_alpha': 0, 'reg_lambda': 1, 'subsample': 0.9}
```

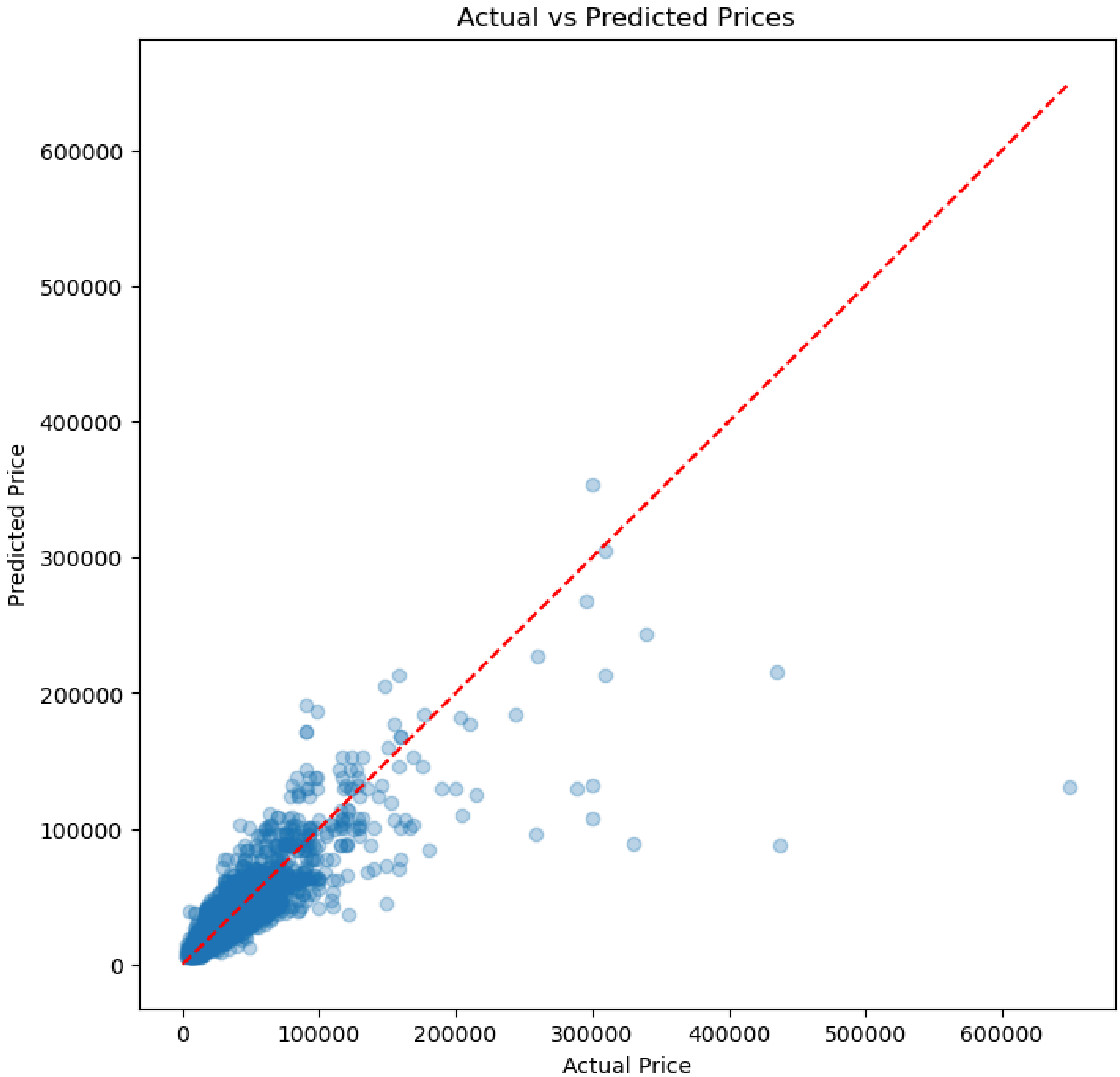
In [115…

```
# ======================
# 9. Evaluation
# ======================
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

def evaluate(y_true, y_pred, label):
    print(f"\n{label} Performance:")
    print(f"R²: {r2_score(y_true, y_pred):.4f}")
    print(f"MAE: ${mean_absolute_error(y_true, y_pred):,.2f}")
    print(f"RMSE: ${np.sqrt(mean_squared_error(y_true, y_pred)):,.2f}")

# Training evaluation
y_train_pred = best_model.predict(X_train_selected)
evaluate(y_train, y_train_pred, "Training")

# Test evaluation
y_test_pred = best_model.predict(X_test_selected)
evaluate(y_test, y_test_pred, "Test")

# 9.1 Visualization
plt.figure(figsize=(10, 6))
pd.Series(best_model.feature_importances_, index=selected_features)\
  .sort_values()\
  .plot(kind='barh', title='Feature Importance')
plt.show()

plt.figure(figsize=(8, 8))
```

```
plt.scatter(y_test, y_test_pred, alpha=0.3)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Prices')
plt.show()
```

```
Training Performance:
R²: 0.7646
MAE: $7,965.49
RMSE: $13,876.96

Test Performance:
R²: 0.6757
MAE: $8,313.15
RMSE: $17,091.64
```




Actual vs Predicted Prices
Predicted Price
Actual Price
0
100000
200000
300000
400000
500000
600000

```
In [ ]:
```

```
In [ ]:
```