```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

df = pd.read_csv('encoded_car_data (1).csv')
print(df.head())
```

```
   diesel  gas  std  turbo  convertible  hardtop  hatchback  sedan
wagon  \
0     0.0  1.0  1.0    0.0          1.0      0.0        0.0    0.0
0.0
1     0.0  1.0  1.0    0.0          1.0      0.0        0.0    0.0
0.0
2     0.0  1.0  1.0    0.0          0.0      0.0        1.0    0.0
0.0
3     0.0  1.0  1.0    0.0          0.0      0.0        0.0    1.0
0.0
4     0.0  1.0  1.0    0.0          0.0      0.0        0.0    1.0
0.0

   4wd  ...  wheelbase  curbweight  enginesize  boreratio  horsepower
\
0  0.0  ...       88.6      2548.0       130.0       3.47       111.0

1  0.0  ...       88.6      2548.0       130.0       3.47       111.0

2  0.0  ...       94.5      2823.0       152.0       2.68       154.0

3  0.0  ...       99.8      2337.0       109.0       3.19       102.0

4  1.0  ...       99.4      2824.0       136.0       3.19       115.0


   carlength  carwidth  citympg  highwaympg    price
0      168.8      64.1     21.0        27.0  13495.0
1      168.8      64.1     21.0        27.0  16500.0
2      171.2      65.5     19.0        26.0  16500.0
3      176.6      66.2     24.0        30.0  13950.0
4      176.6      66.4     18.0        22.0  17450.0

[5 rows x 36 columns]
```

```python
X = df[['enginesize', 'horsepower', 'citympg', 'highwaympg']] #
Numerical features only
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 1. Linear Regression with scaling
linear_model = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)

poly_model = Pipeline([
    ('poly', PolynomialFeatures(degree=2)),
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])
poly_model.fit(X_train, y_train)
y_pred_poly = poly_model.predict(X_test)


print("Linear Regression:")

print(f"MSE: {mean_squared_error(y_test, y_pred_linear):.2f}")
print(f"R2 Score: {r2_score(y_test, y_pred_linear):.2f}")

print("\nPolynomial Regression:")

print(f"MSE: {mean_squared_error(y_test, y_pred_poly):.2f}")
print(f"R2 Score: {r2_score(y_test, y_pred_poly):.2f}")
```

```
Linear Regression:
MSE: 16471505.90
R2 Score: 0.79

Polynomial Regression:
MSE: 15247661.89
R2 Score: 0.81
```

```python
plt.figure(figsize=(10, 5))

plt.scatter(y_test, y_pred_linear, label='Linear', alpha=0.6)

plt.scatter(y_test, y_pred_poly, label='Polynomial (degree-2)',
alpha=0.6)

plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', label='Perfect
Prediction')
```

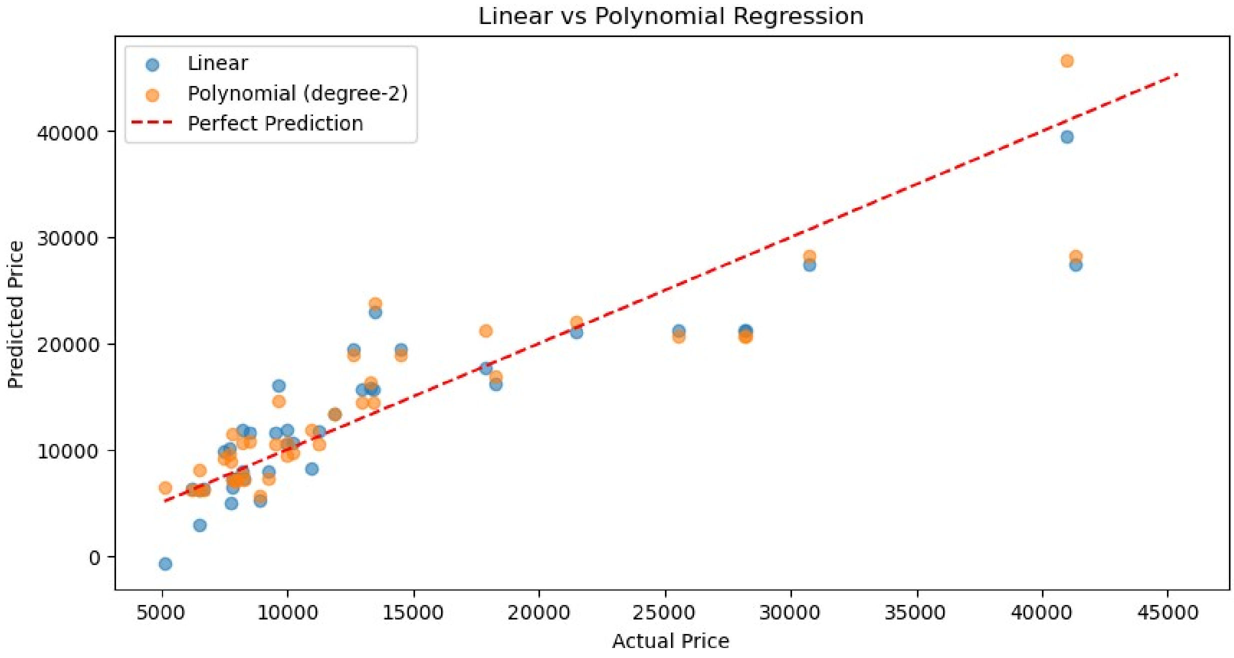
```
plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title("Linear vs Polynomial Regression")

plt.legend()

plt.show()
```


Linear vs Polynomial Regression
Linear
Polynomial (degree-2)
Perfect Prediction
Predicted Price
Actual Price