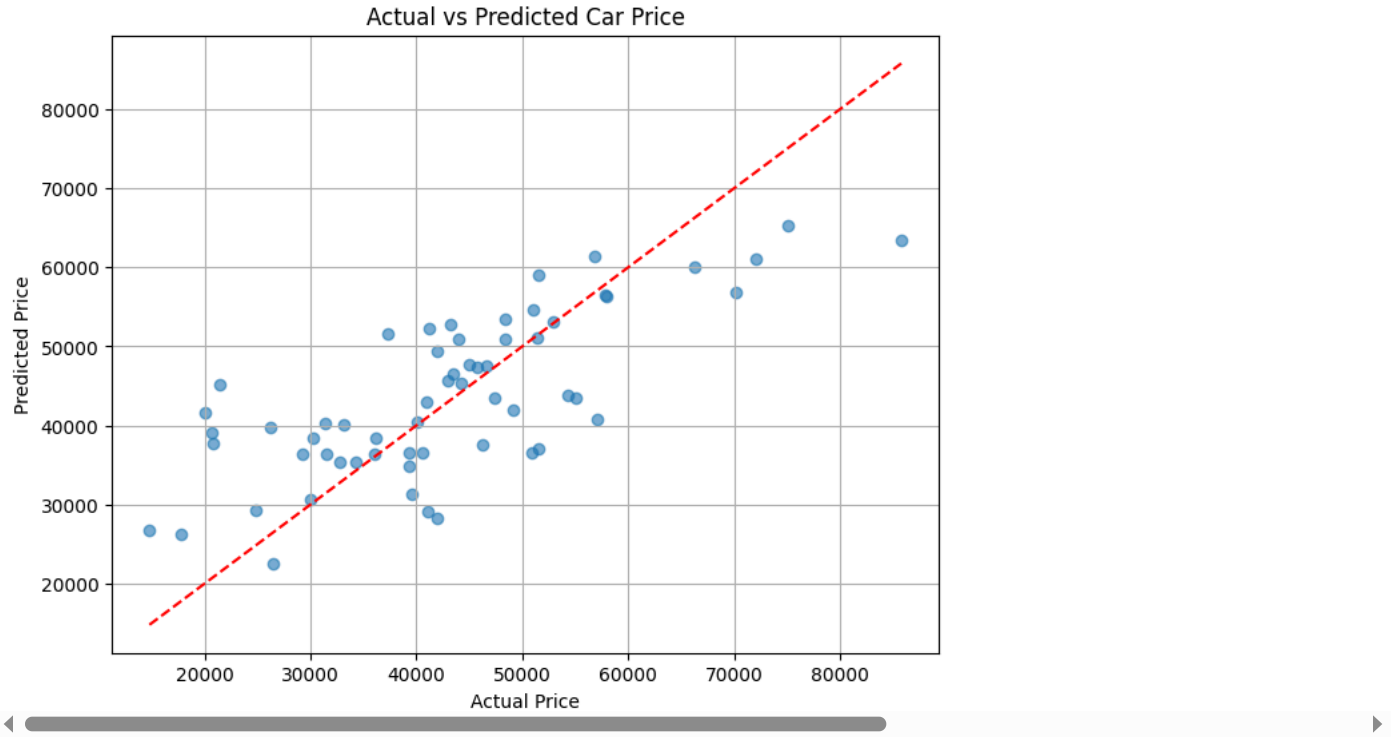
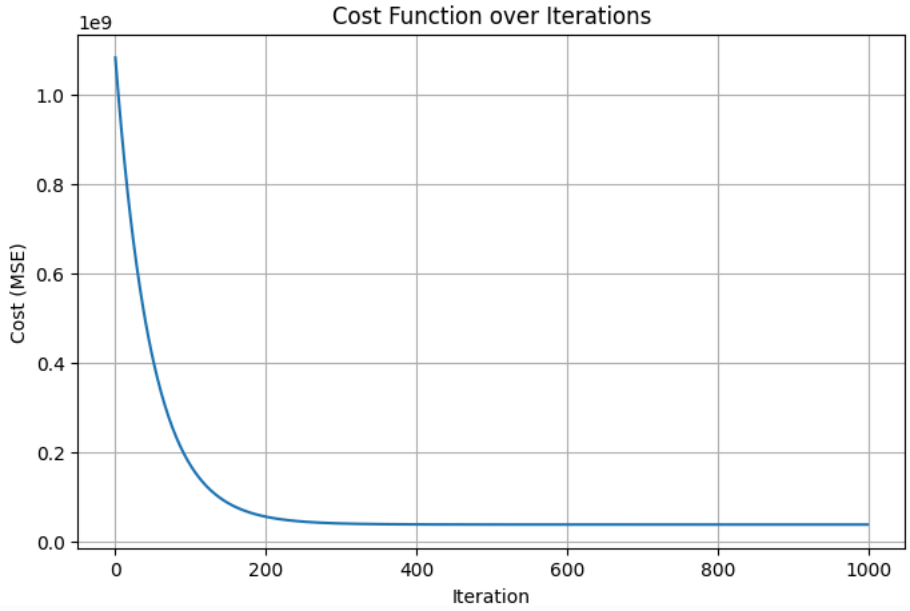
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(42)
n_samples = 200

engine_size = np.random.uniform(80, 250, n_samples)
curb_weight = np.random.uniform(1800, 3500, n_samples)
horsepower = np.random.uniform(70, 250, n_samples)
city_mpg = np.random.uniform(10, 35, n_samples)
highway_mpg = np.random.uniform(15, 45, n_samples)

# Generate price with a known relationship + noise
price = (
    150 * engine_size +
    2.5 * curb_weight +
    120 * horsepower -
    200 * city_mpg -
    150 * highway_mpg +
    np.random.normal(0, 10000, n_samples)
)

# Create DataFrame
df = pd.DataFrame({
    'engine-size': engine_size,
    'curb-weight': curb_weight,
    'horsepower': horsepower,
    'city-mpg': city_mpg,
    'highway-mpg': highway_mpg,
    'price': price
})

# Features and target
X = df[['engine-size', 'curb-weight', 'horsepower', 'city-mpg', 'highway-mpg']]
y = df['price']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Model
model = LinearRegression()
model.fit(X_train, y_train)

# Hypothesis
print("Hypothesis Function:")
hypothesis = f"h(x) = {model.intercept_:.2f} "
for coef, name in zip(model.coef_, X.columns):
    hypothesis += f"+ ({coef:.2f} * {name}) "
print(hypothesis)

# Predict
y_pred = model.predict(X_test)

# Evaluation
print(f"\nTest MSE: {mean_squared_error(y_test, y_pred):.2f}")
print(f"Test MAE: {mean_absolute_error(y_test, y_pred):.2f}")
print(f"R² Score: {r2_score(y_test, y_pred):.4f}")

# Plot
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Car Price")
plt.grid(True)
plt.show()
```

```
Hypothesis Function:
h(x) = 8601.06 + (121.14 * engine-size) + (2.28 * curb-weight) + (134.56 * horsepower) + (-254.63 * city-mpg) + (-251.13 * highway-mpg)

Test MSE: 91370333.69
Test MAE: 7513.05
R² Score: 0.5523
```




```
# Normalize features
X_norm = (X - X.mean()) / X.std()
X_norm['bias'] = 1
X_np = X_norm[['bias', 'engine-size', 'curb-weight', 'horsepower', 'city-mpg', 'highway-mpg']].values
y_np = y.values.reshape(-1, 1)

# Split manually
split_index = int(0.7 * len(X_np))
X_train, X_test = X_np[:split_index], X_np[split_index:]
y_train, y_test = y_np[:split_index], y_np[split_index:]

# Initialize
alpha = 0.01
n_iterations = 1000
m = len(y_train)
n_features = X_train.shape[1]
theta = np.zeros((n_features, 1))
cost_history = []

# Gradient Descent
for i in range(n_iterations):
    predictions = np.dot(X_train, theta)
    error = predictions - y_train
    gradients = (1 / m) * np.dot(X_train.T, error)
    theta -= alpha * gradients
    cost = (1 / (2 * m)) * np.sum(error ** 2)
    cost_history.append(cost)

# Final weights
print("\nFinal learned weights (manual gradient descent):")
for i, val in enumerate(theta.flatten()):
    print(f"Theta {i}: {val:.4f}")

# Cost plot
plt.figure(figsize=(8, 5))
plt.plot(cost_history)
plt.title("Cost Function over Iterations")
plt.xlabel("Iteration")
plt.ylabel("Cost (MSE)")
plt.grid(True)
plt.show()
```

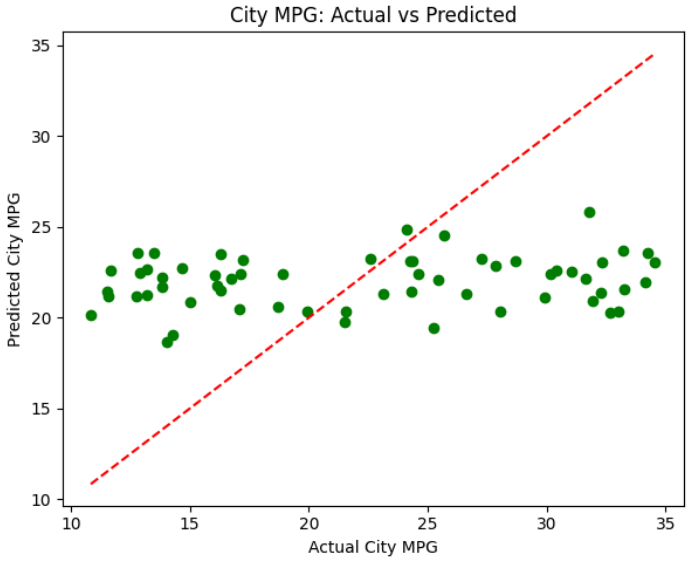
```
Final learned weights (manual gradient descent):
Theta 0: 43942.5113
Theta 1: 5697.6614
Theta 2: 1751.3800
Theta 3: 7676.2560
Theta 4: -1688.5315
Theta 5: -1818.8188
```

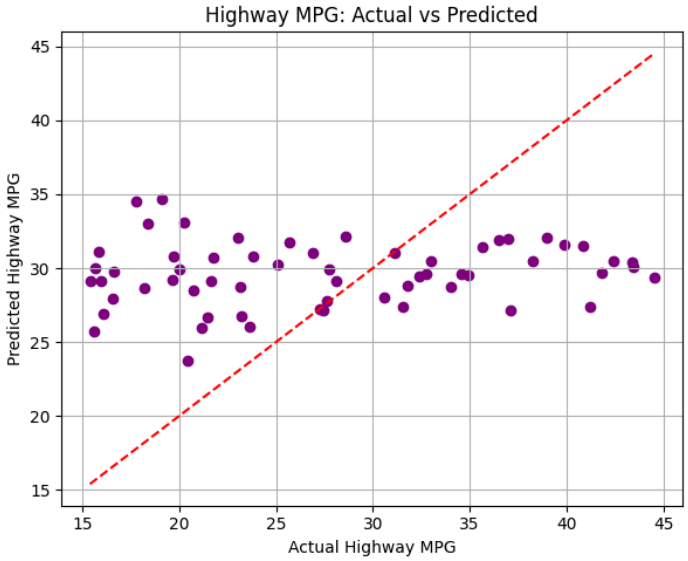



```
# Inputs: engine-size, curb-weight, horsepower, price
X_multi = df[['engine-size', 'curb-weight', 'horsepower', 'price']]
y_multi = df[['city-mpg', 'highway-mpg']]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_multi, y_multi, test_size=0.3, random_state=42)

# Multivariate regression
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# MSE for each target
mse_city = mean_squared_error(y_test['city-mpg'], y_pred[:, 0])
mse_highway = mean_squared_error(y_test['highway-mpg'], y_pred[:, 1])

print(f"\nCity MPG MSE: {mse_city:.2f}")
print(f"Highway MPG MSE: {mse_highway:.2f}")

# Plots
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_test['city-mpg'], y_pred[:, 0], color='green')
plt.plot([y_test['city-mpg'].min(), y_test['city-mpg'].max()],
         [y_test['city-mpg'].min(), y_test['city-mpg'].max()], 'r--')
plt.xlabel("Actual City MPG")
plt.ylabel("Predicted City MPG")
plt.title("City MPG: Actual vs Predicted")

plt.subplot(1, 2, 2)
plt.scatter(y_test['highway-mpg'], y_pred[:, 1], color='purple')
plt.plot([y_test['highway-mpg'].min(), y_test['highway-mpg'].max()],
         [y_test['highway-mpg'].min(), y_test['highway-mpg'].max()], 'r--')
plt.xlabel("Actual Highway MPG")
plt.ylabel("Predicted Highway MPG")
plt.title("Highway MPG: Actual vs Predicted")

plt.tight_layout()
plt.grid(True)
```

```
plt.show()
```

```
City MPG MSE: 56.11
Highway MPG MSE: 80.20
```





+ Code + Text