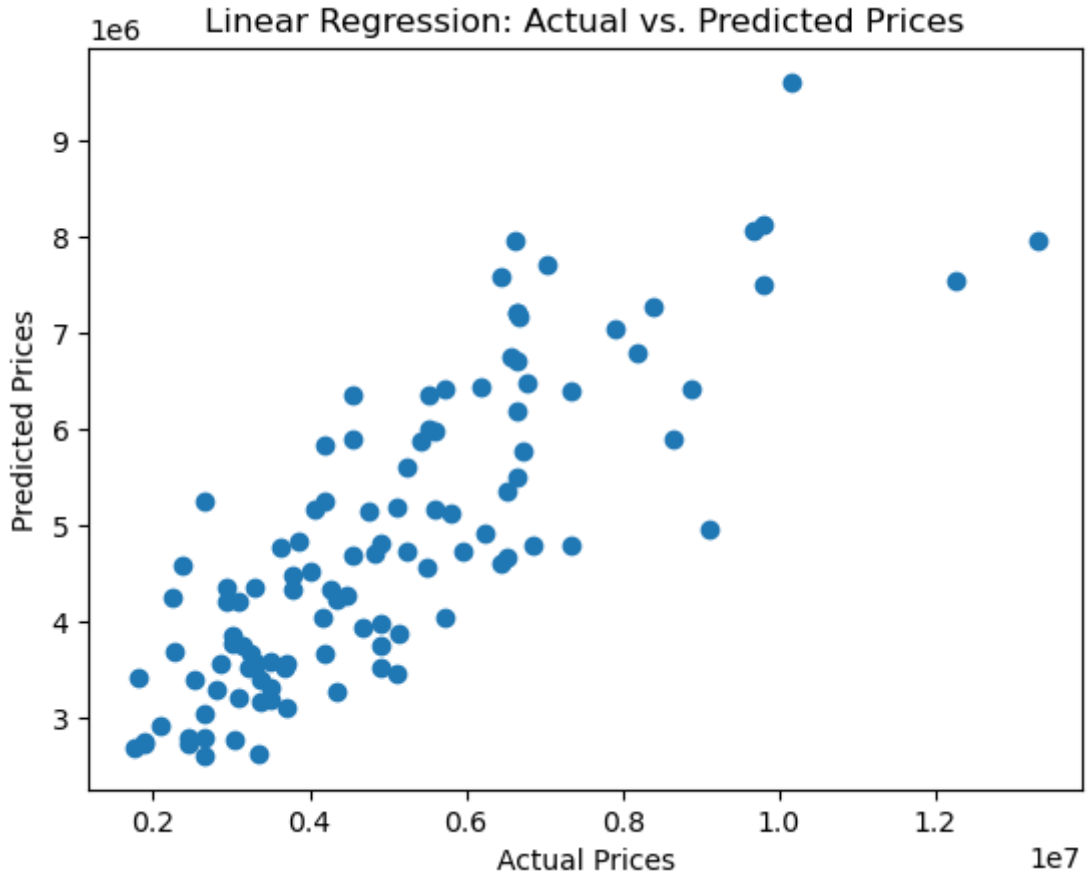
In [8]:

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LassoCV
from sklearn.preprocessing import StandardScaler

# Load the dataset
housing = pd.read_csv("C:\\Users\\91947\\Downloads\\Housing.csv")

# Encode categorical variables
housing = pd.get_dummies(housing, drop_first=True)

# Split the dataset into features (X) and target (y)
X = housing.drop('price', axis=1)
y = housing['price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train Linear Regression model
linear_reg = LinearRegression()
linear_reg.fit(X_train_scaled, y_train)

# Predict and plot
y_pred = linear_reg.predict(X_test_scaled)

# Plot predictions vs. actual prices
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Linear Regression: Actual vs. Predicted Prices")
plt.show()

# Lasso regression with cross-validation to find the best alpha
alphas = np.logspace(-4, 4, 100)  # Define a range of alpha values to test

lasso_cv = LassoCV(alphas=alphas, cv=5)  # Use cross-validation to find the best alpha
lasso_cv.fit(X_train_scaled, y_train)

best_alpha = lasso_cv.alpha_
print(f"Best alpha: {best_alpha}")
```




```
Best alpha: 890.2150854450392
```

In [10]:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features (optional but recommended for logistic regression)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 1: Logistic Regression for binary classification (example: class 0 vs. all others)
binary_classifier = LogisticRegression(random_state=42)
binary_classifier.fit(X_train, (y_train == 0).astype(int))
y_pred_binary = binary_classifier.predict(X_test)

# Calculate accuracy for binary classification
accuracy_binary = accuracy_score((y_test == 0).astype(int), y_pred_binary)
print("Binary Classification Accuracy:", accuracy_binary)

# Step 2: Multiple Logistic Regression for multiclass classification (all 3 classes)
multi_classifier = LogisticRegression(random_state=42, multi_class='multinomial', solver=
multi_classifier.fit(X_train, y_train)
y_pred_multi = multi_classifier.predict(X_test)

# Calculate accuracy for multiclass classification
accuracy_multi = accuracy_score(y_test, y_pred_multi)
print("Multiclass Classification Accuracy:", accuracy_multi)

# Print classification report for detailed evaluation
print("\nClassification Report:")
print(classification_report(y_test, y_pred_multi, target_names=iris.target_names))
```

```
Binary Classification Accuracy: 1.0
Multiclass Classification Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [ ]: