--- Problem Description ---

The objective is to predict laptop prices (regression) and categorize them into price segments (classification).

--- Exploratory Data Analysis (EDA) ---

Shape of dataset: (11768, 11)

Data overview:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 11768 entries, 0 to 11767

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Brand 11768 non-null object

1 Processor 11768 non-null object

2 RAM (GB) 11768 non-null int64

3 Storage 11768 non-null object

4 GPU 11768 non-null object

5 Screen Size (inch) 11768 non-null float64

6 Resolution 11768 non-null object

7 Battery Life (hours) 11768 non-null float64

8 Weight (kg) 11768 non-null float64

9 Operating System 11768 non-null object

10 Price ($) 11768 non-null float64

dtypes: float64(4), int64(1), object(6)

memory usage: 1011.4+ KB

None

Summary statistics:

RAM (GB) Screen Size (inch) Battery Life (hours) Weight (kg) \

count 11768.000000 11768.000000 11768.000000 11768.000000

mean 24.852821 15.212305 8.027855 2.341117

std 21.762567 1.436997 2.305400 0.667921

min 4.000000 13.300000 4.000000 1.200000

25% 8.000000 14.000000 6.000000 1.760000

50% 16.000000 15.600000 8.000000 2.340000

75% 32.000000 16.000000 10.000000 2.910000

max 64.000000 17.300000 12.000000 3.500000

Price ($)

count 11768.000000

mean 2183.571608

std 1316.886132

min 279.570000

25% 1272.045000

50% 1840.865000

75% 2698.370000

max 10807.880000

Missing values:

Brand 0

Processor 0

RAM (GB) 0

Storage 0

GPU 0

Screen Size (inch) 0

Resolution 0

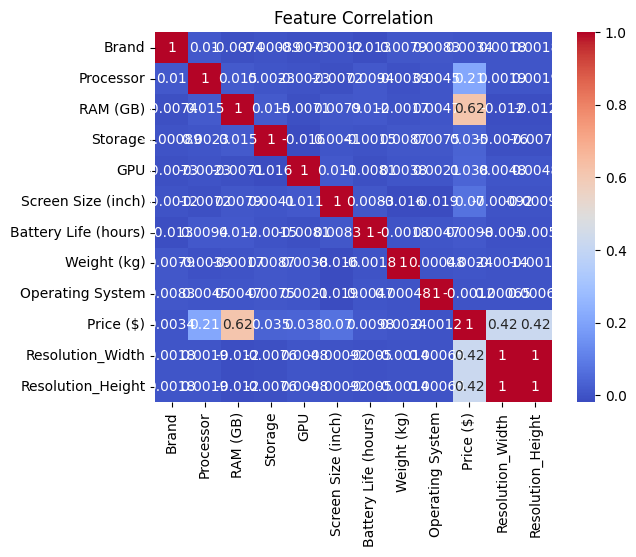
Battery Life (hours) 0

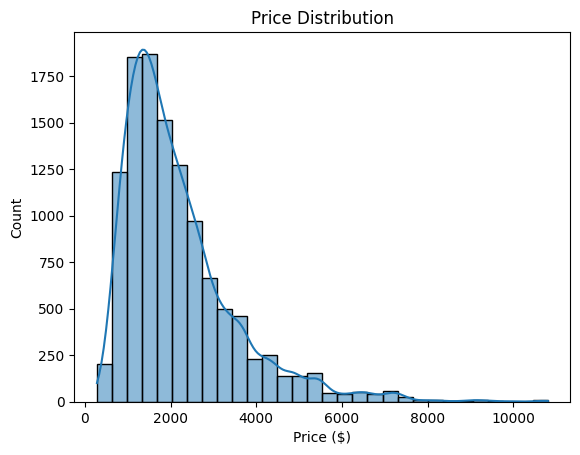
Weight (kg) 0

Operating System 0

Price ($) 0

dtype: int64





--- Model Training and Evaluation ---

Linear Regression RMSE: 832.8247004446964

/usr/local/lib/python3.11/dist-packages/sklearn/linear\_model/\_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

<https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>

n\_iter\_i = \_check\_optimize\_result(

Logistic Regression Classification Report:

precision recall f1-score support

0 0.64 0.30 0.41 312

1 0.59 0.70 0.64 965

2 0.77 0.77 0.77 1077

accuracy 0.68 2354

macro avg 0.67 0.59 0.61 2354

weighted avg 0.68 0.68 0.67 2354

Decision Tree Classification Report:

precision recall f1-score support

0 0.70 0.60 0.64 312

1 0.72 0.77 0.74 965

2 0.86 0.84 0.85 1077

accuracy 0.78 2354

macro avg 0.76 0.74 0.75 2354

weighted avg 0.78 0.78 0.78 2354

SVM Classification Report:

precision recall f1-score support

0 0.00 0.00 0.00 312

1 0.47 0.58 0.52 965

2 0.61 0.65 0.63 1077

accuracy 0.54 2354

macro avg 0.36 0.41 0.38 2354

weighted avg 0.47 0.54 0.50 2354

--- Discussion/Conclusion ---

Linear Regression provides a good baseline for price prediction.

For classification, Decision Tree and SVM show competitive performance, with some trade-offs in accuracy and interpretability.

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))