Data Pre-processing and Exploration

Laptop Price

presented by

Wissarut Kanasub 6510545721

Assoc.Prof.Dr. Kitsana Waiyamai
Data Analytics 01219367
Kasetsart University

Impact of outliers

Outliers affects:

- **Mean:** Susceptible to outliers; can be distorted by extreme values, pulling it towards them and giving a false impression of central tendency.

- **Median:** Less affected by outliers; represents the middle value in a dataset and is not sensitive to extreme values, making it a better measure of the typical value, especially in datasets with outliers.

:

- **Strength of Relationship:** Outliers with extreme values in both variables can exaggerate the perceived strength of their relationship.

- **Direction of Relationship:** Outliers can skew the perceived direction of the relationship, making it appear more positive or negative than it actually is.

- **Regression Models:** Outliers can significantly impact the slope and fit of regression lines, potentially leading to misleading interpretations.

- **Misleading Conclusions:** Outliers, by their nature, do not represent the majority of the data, and conclusions drawn from data with outliers may be misleading as they might not reflect typical behavior or patterns.

Outliers effects on Models:

- **Normal Distribution Assumption:** Outliers can violate the assumption of normal distribution in statistical tests, leading to incorrect results.
- **Model Performance:** Outliers can negatively impact the performance of statistical models like linear regression by causing them to focus excessively on extreme values.
- **Overfitting:** Outliers can contribute to overfitting in machine learning models, where the model becomes too complex and fits the training data too closely.
- **Bias:** Outliers can introduce bias into machine learning algorithms, particularly those relying on distance metrics like k-means clustering, leading to inaccurate results.

Data source

ANUBHAV GOYAL · UPDATED 10 MONTHS AGO

Laptop Prices Dataset

Laptop prices for Regression practice

[Laptop Prices Dataset (kaggle.com)]

Data exploration and Preprocessing

Determine shape and data type of dataset. (data source)

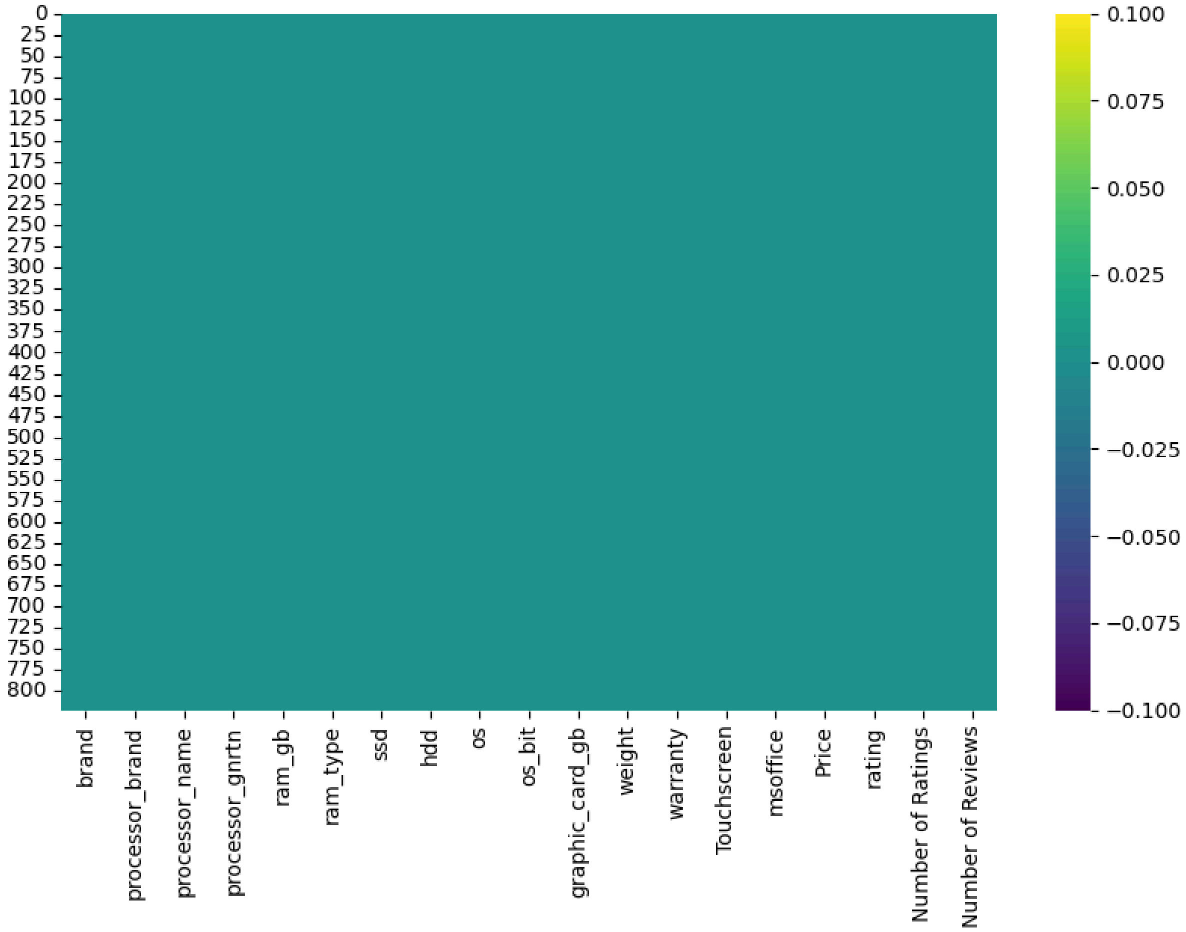
```
Data shape: (823, 19)
-------------------------------
brand                    object
processor_brand          object
processor_name           object
processor_gnrtn          object
ram_gb                   object
ram_type                 object
ssd                      object
hdd                      object
os                       object
os_bit                   object
graphic_card_gb          object
weight                   object
warranty                 object
Touchscreen              object
msoffice                 object
Price                     int64
rating                   object
Number of Ratings         int64
Number of Reviews         int64
dtype: object
-------------------------------
```

- Dataset has 823 rows and 19 columns.
- Dataset has only 3 columns of numerical type.

Determine descriptive statistics

	Price	Number of Ratings	Number of Reviews
count	823.000000	823.000000	823.000000
mean	76745.177400	315.301337	37.609964
std	45101.790525	1047.382654	121.728017
min	16990.000000	0.000000	0.000000
25%	46095.000000	0.000000	0.000000
50%	64990.000000	17.000000	2.000000
75%	89636.000000	139.500000	18.000000
max	441990.000000	15279.000000	1947.000000

Determine null or missing values by using Heatmap.



- There are no null or missing values.

Drop needless columns.

Column name	Reason to drop
processor_name	Redundant if you have another column specifying the processor model.
processor_gnrtn	Generation hard to compare with other generations in other brands.
os	Most devices likely have an OS, so this column adds little value unless OS specifics are crucial.
os_bit	Redundant as most modern systems are 64-bit; can be inferred or assumed.

weight	Might not be relevant unless analyzing portability.
warranty	Relevant for reliability analysis but not crucial price analysis.
msoffice	Redundant if bundled software is covered elsewhere or not a key feature.
Number of Ratings, Number of Reviews,	From original dataset even number of rating and review are 0 it still has a rating value.

	brand	processor_brand	ram_gb	ram_type	ssd	hdd	graphic_card_gb	Touchscreen	Price	rating
0	ASUS	Intel	4 GB	DDR4	0 GB	1024 GB	0 GB	No	34649	2 stars
1	Lenovo	Intel	4 GB	DDR4	0 GB	1024 GB	0 GB	No	38999	3 stars
2	Lenovo	Intel	4 GB	DDR4	0 GB	1024 GB	0 GB	No	39999	3 stars
3	ASUS	Intel	8 GB	DDR4	512 GB	0 GB	2 GB	No	69990	3 stars
4	ASUS	Intel	4 GB	DDR4	0 GB	512 GB	0 GB	No	26990	3 stars
...	...	...	...	...	...	...	...	...	...	...
818	ASUS	AMD	4 GB	DDR4	1024 GB	0 GB	0 GB	No	135990	3 stars
819	ASUS	AMD	4 GB	DDR4	1024 GB	0 GB	0 GB	No	144990	3 stars
820	ASUS	AMD	4 GB	DDR4	1024 GB	0 GB	4 GB	No	149990	3 stars
821	ASUS	AMD	4 GB	DDR4	1024 GB	0 GB	4 GB	No	142990	3 stars
822	Lenovo	AMD	8 GB	DDR4	512 GB	0 GB	0 GB	No	57490	4 stars

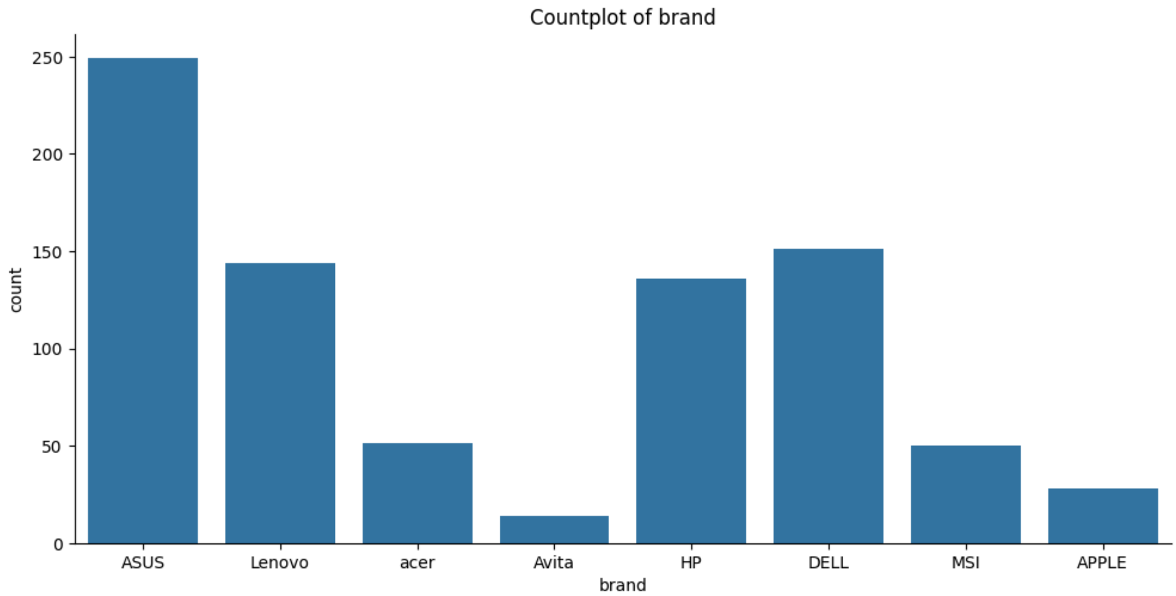
823 rows × 10 columns

(Dataset after drop needless columns)
shape: (823,10)

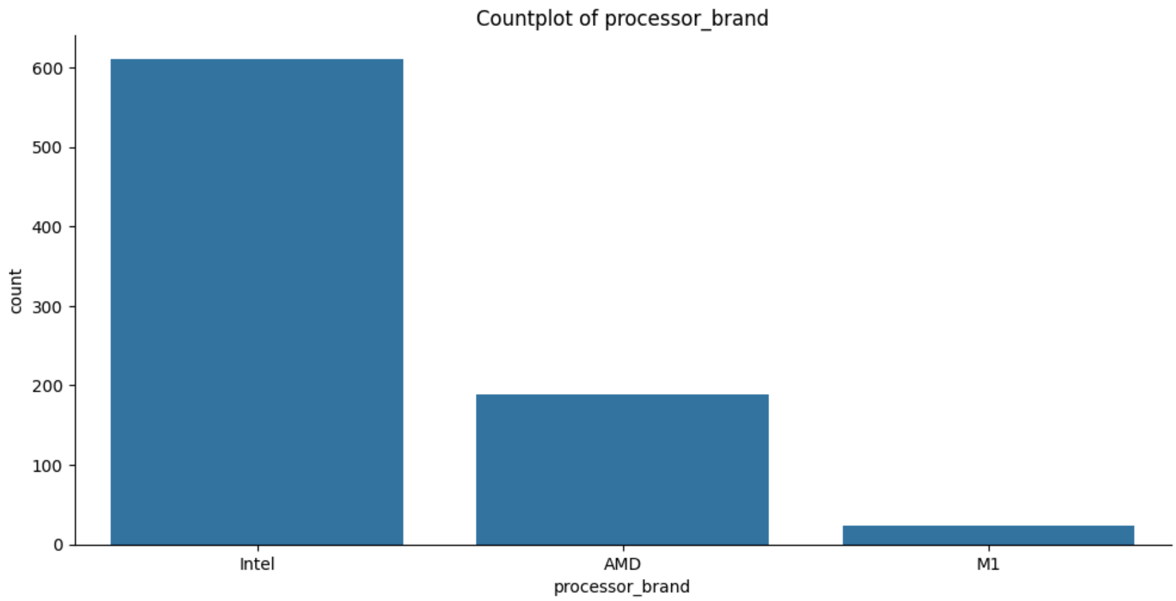
Explore categorical data.

- Explore categorical data by using a count plot.

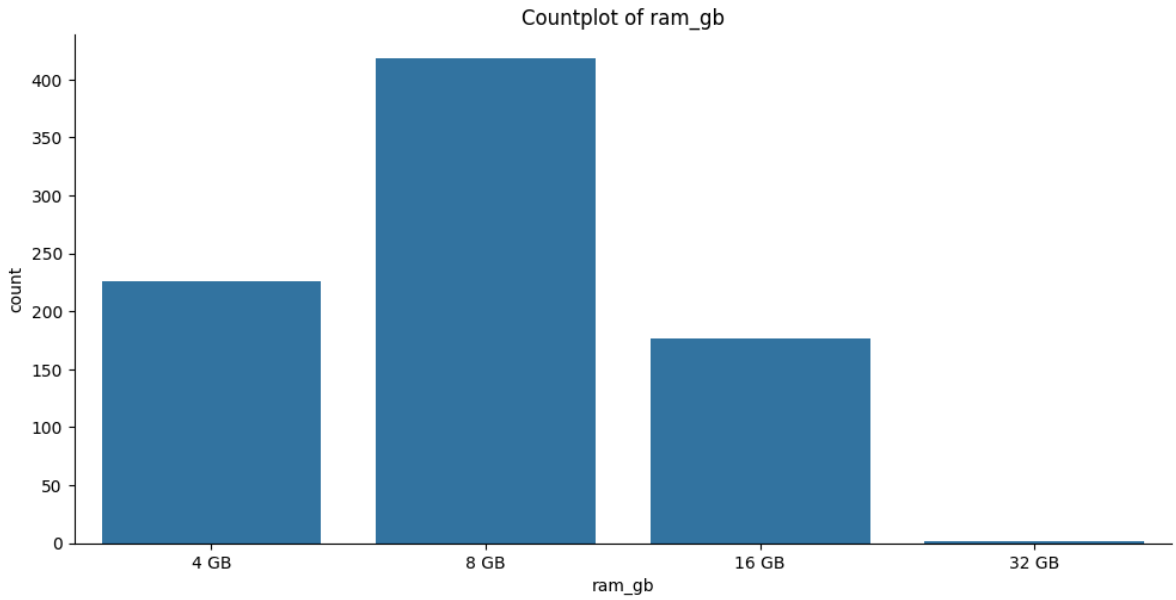
```
# Explore Categorical data by count plot.
categorical_features = original_data.select_dtypes(include="object").columns.tolist()

for col in categorical_features:
    sns.catplot(data=original_data, x=col, kind='count', aspect=2)
    plt.title(f'Countplot of {col}')
    plt.show()
```



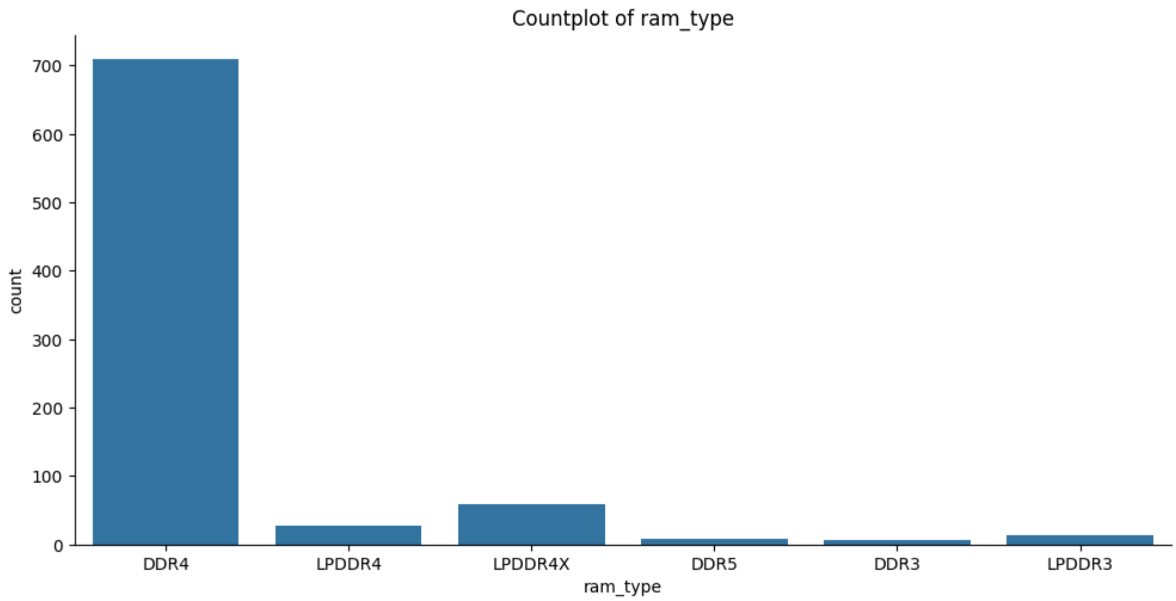
Countplot of brand
count
250
200
150
100
50
0
ASUS
Lenovo
acer
Avita
HP
DELL
MSI
APPLE
brand



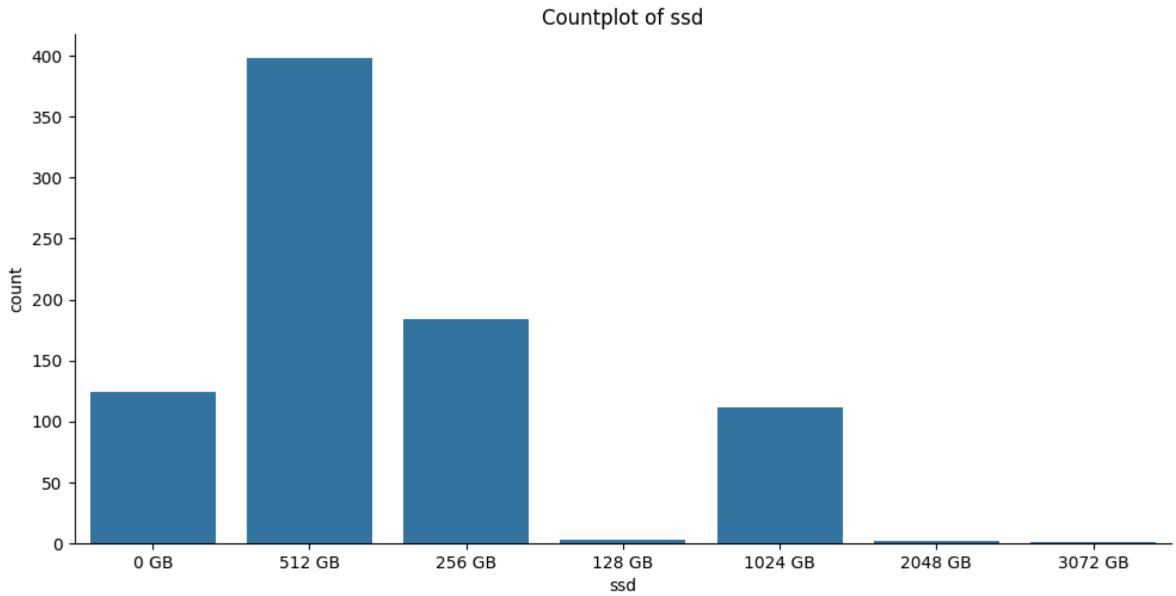
Countplot of processor_brand
count
600
500
400
300
200
100
0
Intel
AMD
M1
processor_brand

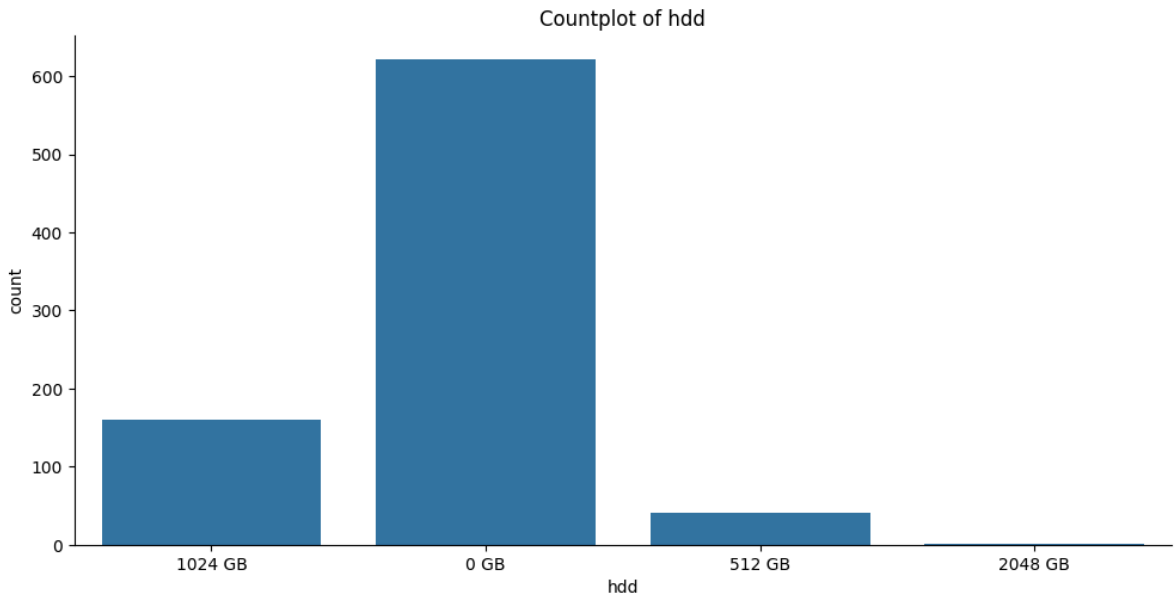


Countplot of ram_gb
count
0
50
100
150
200
250
300
350
400
4 GB
8 GB
16 GB
32 GB
ram_gb

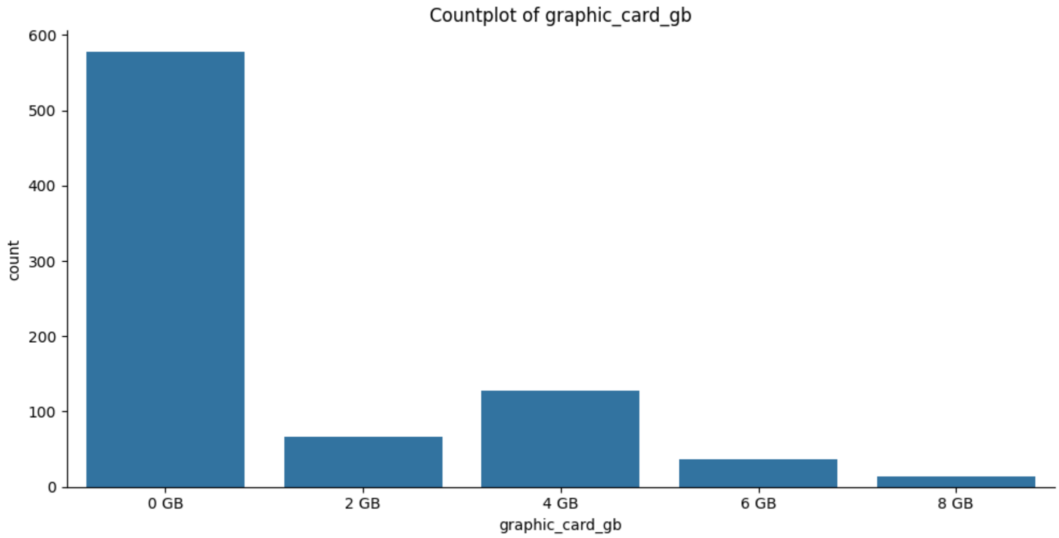


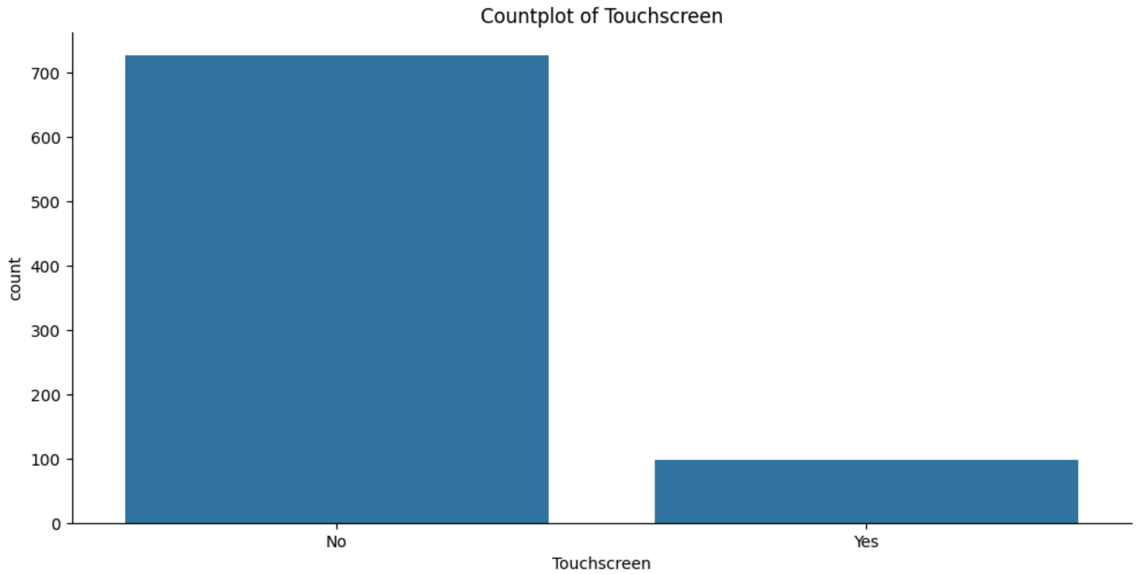
Countplot of ram_type
count
0
100
200
300
400
500
600
700
DDR4
LPDDR4
LPDDR4X
DDR5
DDR3
LPDDR3
ram_type

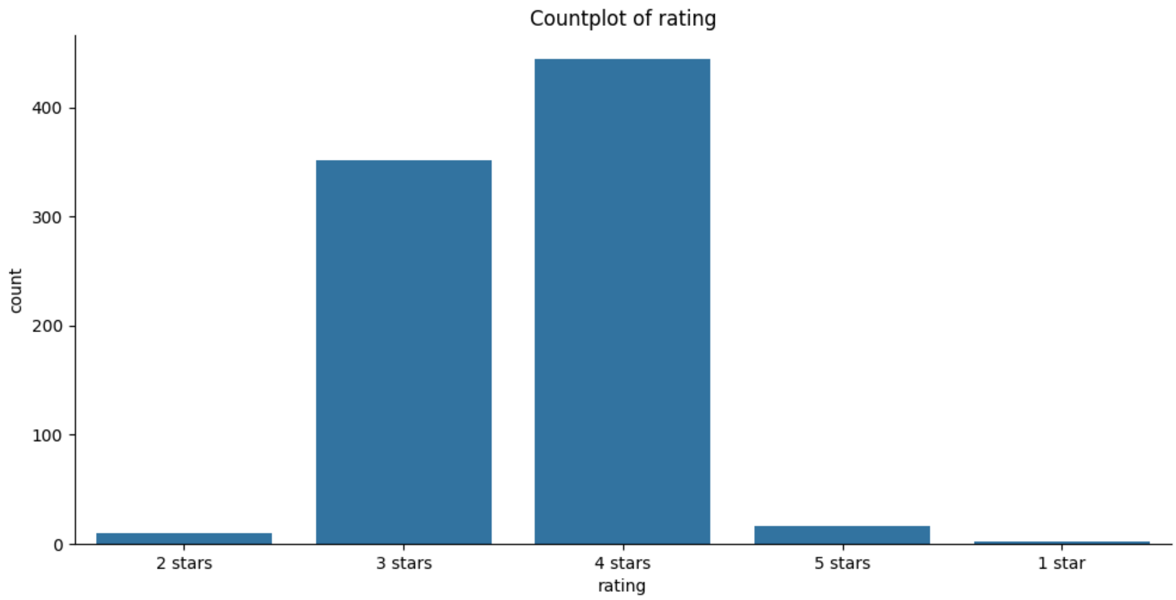




From both the SSD and HDD count plots, we will notice that the highest value for HDD is often 0 GB. This suggests that many laptops in the dataset are using SSDs instead of HDDs.


Countplot of graphic_card_gb
count
0
100
200
300
400
500
600
0 GB
2 GB
4 GB
6 GB
8 GB
graphic_card_gb

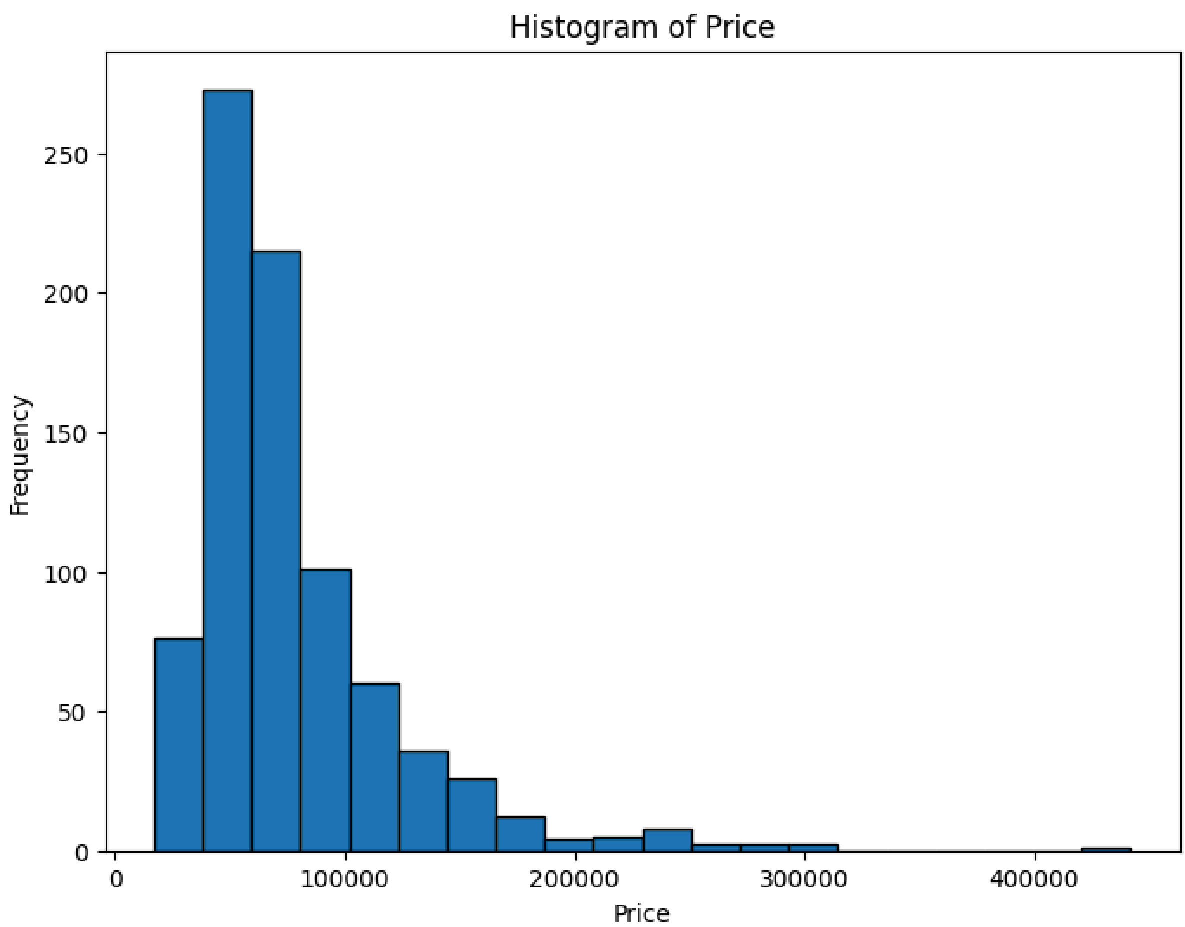

Countplot of Touchscreen
count
0
100
200
300
400
500
600
700
No
Yes
Touchscreen


Countplot of rating
count
0
100
200
300
400
2 stars
3 stars
4 stars
5 stars
1 star
rating

Explore numerical data.

- By using histogram plot.

```python
# Explore numerical data by histogram.
numerical_features = original_data.select_dtypes(exclude="object").columns.tolist()

# Create histograms for each numerical feature
for col in numerical_features:
    plt.figure(figsize=(8, 6))
    plt.hist(original_data[col], bins=20, edgecolor='black')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {col}')
    plt.show()
```



From histogram we noticed that it has right-skewed distributions.

Convert categorical to numerical.

- From exploring categorical data some of the data is numeric but it represents its string or text form.

- So we need to convert it to numeric

```python
# Function to covert categorical to numerical type.
def convert_categorical_to_numerical(data, column):
    # Extract the numerical part from the categorical data
    data[column] = data[column].str.extract('(\d+)').astype(float)
    return data
```

```python
# Converting Code Area.
to_convert_column = ['ram_gb', 'ssd', 'hdd', 'graphic_card_gb', 'rating'] # List of data to converting.
for col in to_convert_column:
    original_data = convert_categorical_to_numerical(original_data, col)
```

- Data after convert

	brand	processor_brand	ram_gb	ram_type	ssd	hdd	graphic_card_gb	Touchscreen	Price	rating
0	ASUS	Intel	4.0	DDR4	0.0	1024.0	0.0	No	34649	2.0
1	Lenovo	Intel	4.0	DDR4	0.0	1024.0	0.0	No	38999	3.0
2	Lenovo	Intel	4.0	DDR4	0.0	1024.0	0.0	No	39999	3.0
3	ASUS	Intel	8.0	DDR4	512.0	0.0	2.0	No	69990	3.0
4	ASUS	Intel	4.0	DDR4	0.0	512.0	0.0	No	26990	3.0
...	...	...	...	...	...	...	...	...	...	...
818	ASUS	AMD	4.0	DDR4	1024.0	0.0	0.0	No	135990	3.0
819	ASUS	AMD	4.0	DDR4	1024.0	0.0	0.0	No	144990	3.0
820	ASUS	AMD	4.0	DDR4	1024.0	0.0	4.0	No	149990	3.0
821	ASUS	AMD	4.0	DDR4	1024.0	0.0	4.0	No	142990	3.0
822	Lenovo	AMD	8.0	DDR4	512.0	0.0	0.0	No	57490	4.0

823 rows × 10 columns

Using LabelEncoder.

- Encode some columns to numeric to improve modeling performance in the future.

```
# Convert categorical to numberical with Label Encoder. This use only Unique data.
to_convert_column = ['brand', 'processor_brand', 'ram_type', 'Touchscreen']
label_encoder = preprocessing.LabelEncoder() # Create object.

for col in to_convert_column:
    original_data[col] = label_encoder.fit_transform(original_data[col])
    display(f"{col}: {dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))}")

display(original_data)
```

- After Encoder.

	brand	processor_brand	ram_gb	ram_type	ssd	hdd	graphic_card_gb	Touchscreen	Price	rating
0	1	1	4.0	1	0.0	1024.0	0.0	0	34649	2.0
1	5	1	4.0	1	0.0	1024.0	0.0	0	38999	3.0
2	5	1	4.0	1	0.0	1024.0	0.0	0	39999	3.0
3	1	1	8.0	1	512.0	0.0	2.0	0	69990	3.0
4	1	1	4.0	1	0.0	512.0	0.0	0	26990	3.0
...	...	...	...	...	...	...	...	...	...	...
818	1	0	4.0	1	1024.0	0.0	0.0	0	135990	3.0
819	1	0	4.0	1	1024.0	0.0	0.0	0	144990	3.0
820	1	0	4.0	1	1024.0	0.0	4.0	0	149990	3.0
821	1	0	4.0	1	1024.0	0.0	4.0	0	142990	3.0
822	5	0	8.0	1	512.0	0.0	0.0	0	57490	4.0

823 rows × 10 columns

Brands	
Apple	0
Asus	1
Avita	2
Dell	3

HP	4
Lenovo	5
MSI	6
acer	7

Processor Brands

AMD	0
Intell	1
M1	2

Ram Type

DDR3	0
DDR4	1
DDR5	2
LPDDR3	3
LPDDR4	4
LPDDR4X	5

Tochscreen

NO	0
YES	1

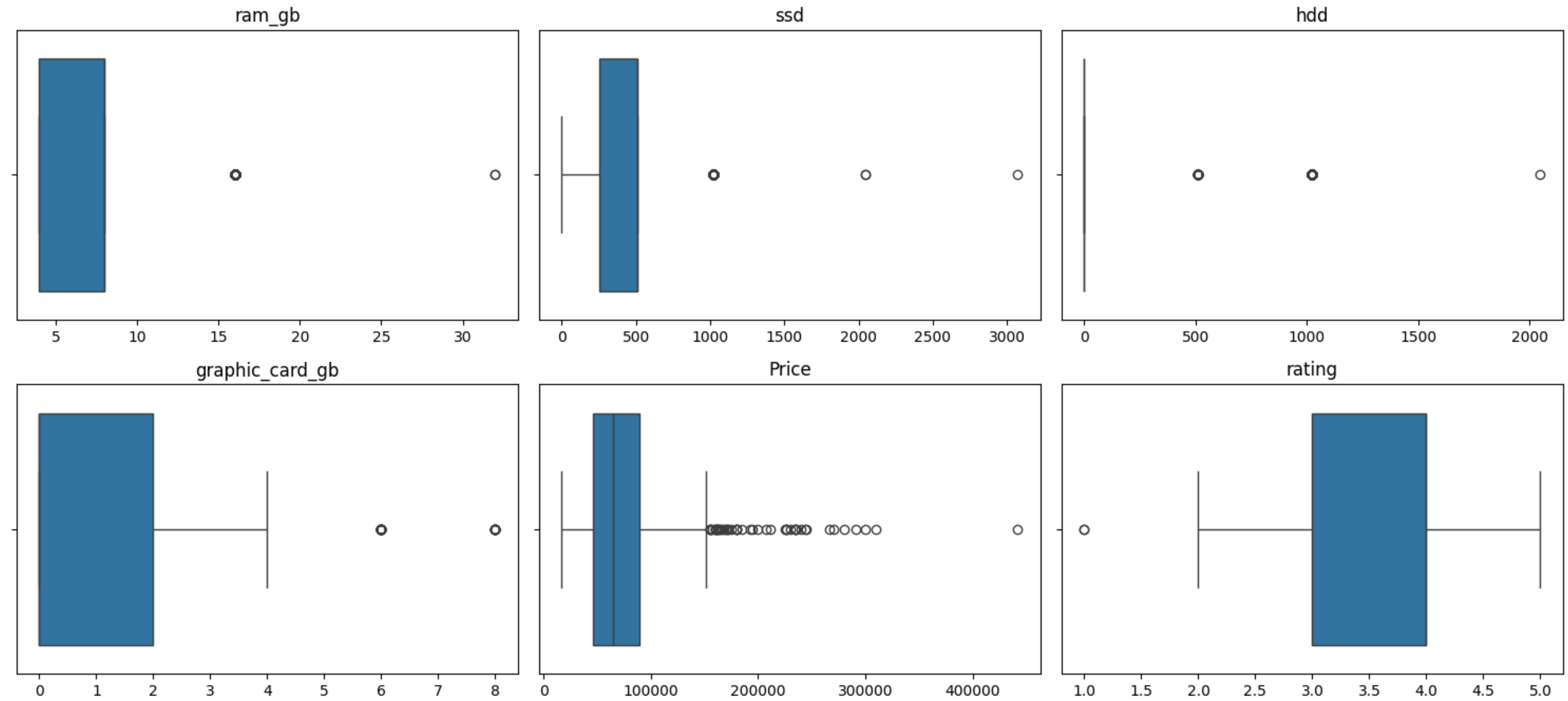
Using Boxplot to finding outliers

```python
# Plot box plots for each numerical feature
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features, start=1):
    plt.subplot(3, 3, i)
    sns.boxplot(x=original_data[feature])
    plt.title(feature)
    plt.xlabel('')
plt.tight_layout()
plt.show()
```

- From the plot below we see that there are columns containing outliers.

- I need to replace outliers with the mode only for the 'Rating' attribute because I believe that other numerical attributes, such as RAM memory, VGA capacity, HDD size, and SSD storage, are crucial factors affecting the price of laptops in real-world scenarios. This decision was made to ensure that the model's accuracy remains high and reflects the significant impact of these hardware specifications on laptop pricing.

- I need to replace outliers with mean (Technique that we interested) on column “Price”


ram_gb
ssd
hdd
graphic_card_gb
Price
rating

Replace rating's outliers with mode.

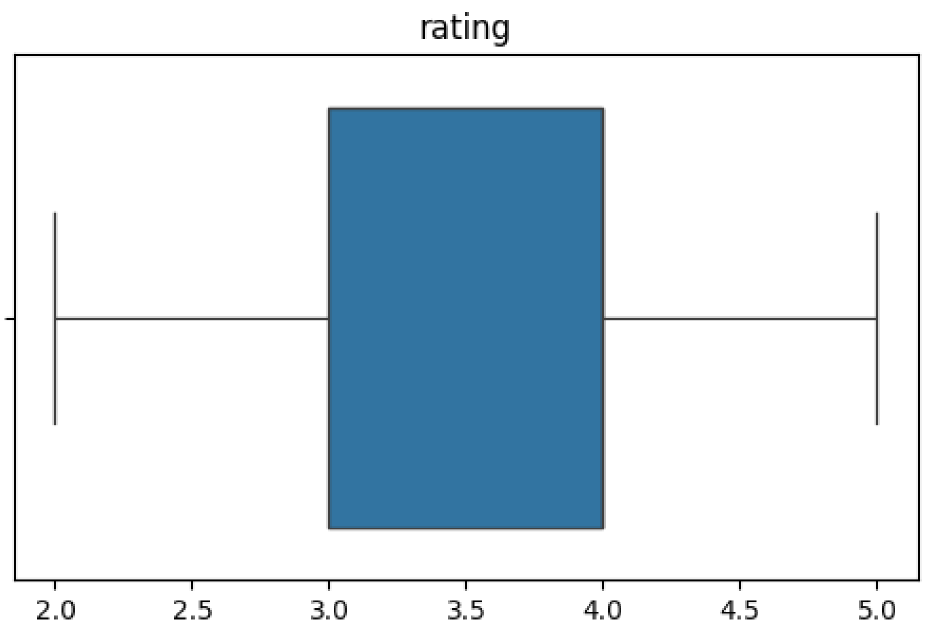
- Function:

```python
def replace_outliers_with_mode(data, columns):
    for col in columns:
        mode_val = data[col].mode()[0]  # Calculate the mode for the column
        Q1 = data[col].quantile(0.25)  # First quartile
        Q3 = data[col].quantile(0.75)  # Third quartile
        IQR = Q3 - Q1  # Interquartile range

        # Define the outlier thresholds
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Replace outliers with mode
        data[col] = data[col].apply(lambda x: mode_val if x < lower_bound or x > upper_bound else x)

    return data
```

- Coding process:

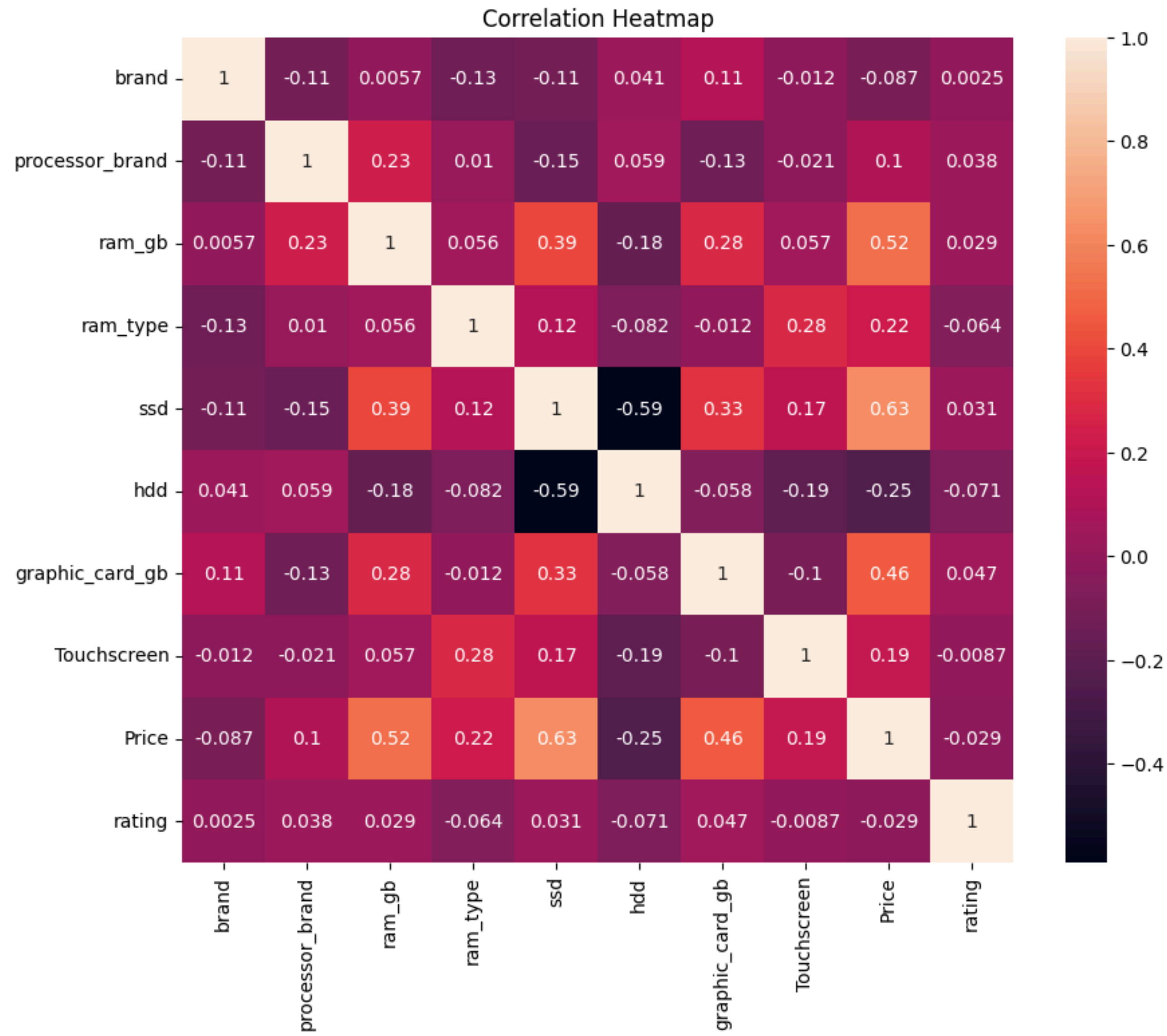
```python
# Replacing outliers with mode beacuse some of data is discrete type such as Size of memory, Rating or Stars
to_replace_with_mode = ['rating']

for col in to_replace_with_mode:
    original_data = replace_outliers_with_mode(original_data, [col])

display(original_data)
```

- After replacing outliers with mode.



Find Correlations.

- Using a heat map to find Correlation.

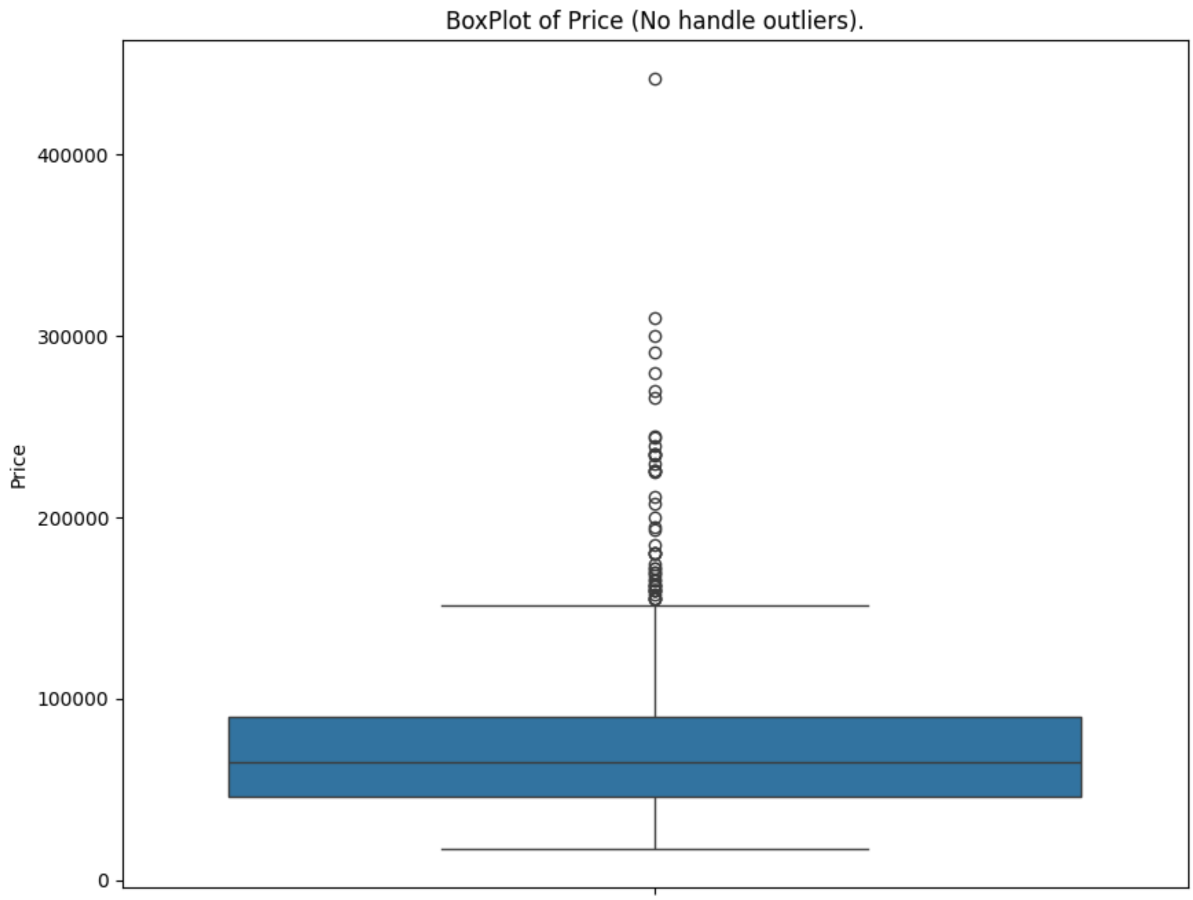



```python
# Find Correlation by heatmap.
plt.figure(figsize=(10,8))
sns.heatmap(data=original_data.corr(), annot=True)
plt.title("Correlation Heatmap")
plt.show()
```

Prepare data for the modeling part.

- data_no_handle_outliers

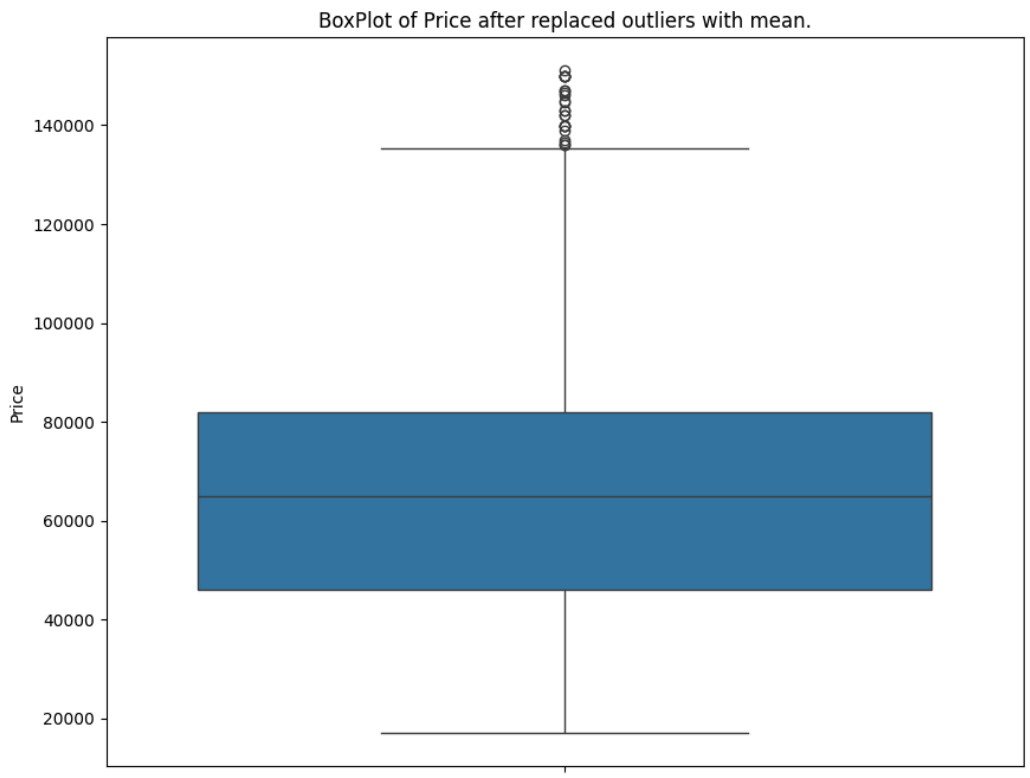
```python
attribute = "Price"
data_no_handle_outliers = original_data.copy()

plt.figure(figsize=(10,8))
sns.boxplot(data=data_no_handle_outliers, y=attribute)
plt.title(f'BoxPlot of Price (No handle outliers).')
plt.show()
```



- data_outliers_replaced_with_mean

```python
def replace_outliers_with_mean(data, column):
    # Calculate mean
    mean_val = data[column].mean()
    # Calculate first and third quartiles
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    # Calculate interquartile range (IQR)
    IQR = Q3 - Q1
    # Define outlier thresholds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with mean
    data[column] = data[column].apply(lambda x: mean_val if x < lower_bound or x > upper_bound else x)
    return data

```

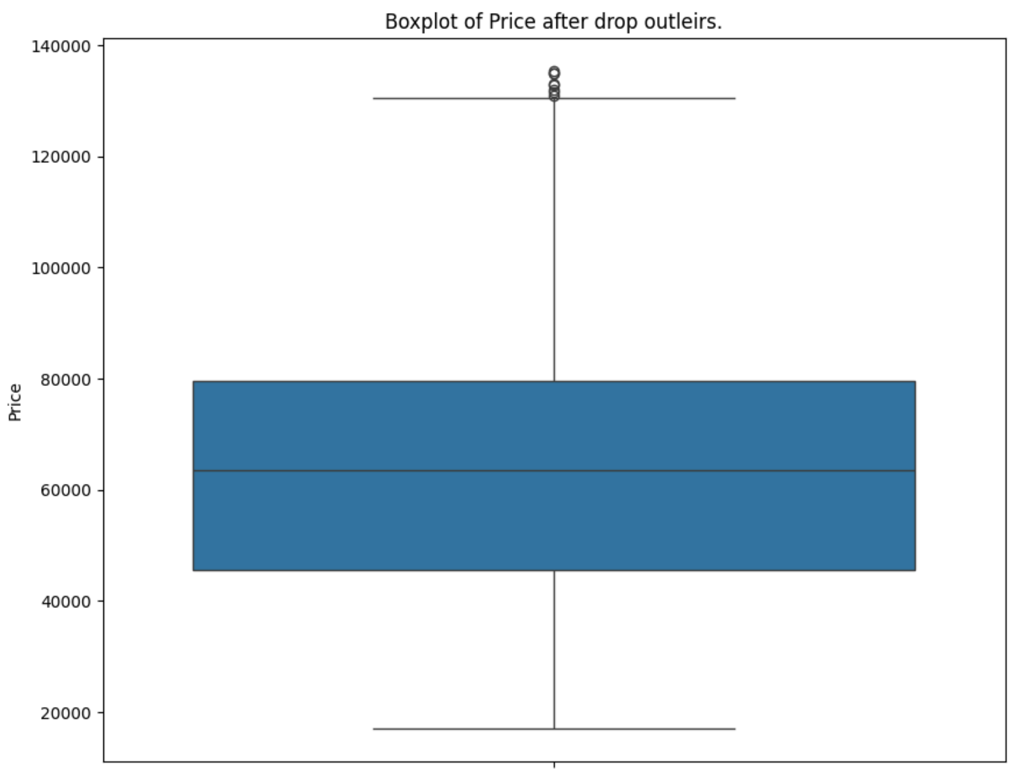
```python
# Replace outliers with mean.
to_replace_with_mean = "Price"
data_outliers_replaced_with_mean = replace_outliers_with_mean(original_data, to_replace_with_mean)

plt.figure(figsize=(10,8))
sns.boxplot(data=data_outliers_replaced_with_mean, y=to_replace_with_mean)
plt.title(f'BoxPlot of Price after replaced outliers with mean.')
plt.show()
```



- data_deleted_outliers

```python
def drop_rows_with_outliers(data, column):
    # Calculate first and third quartiles
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    # Calculate interquartile range (IQR)
    IQR = Q3 - Q1
    # Define outlier thresholds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Drop rows containing outliers
    data = data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]
    return data
```

```python
# Drop outliers
to_drop_outliers = "Price"
data_deleted_outliers = drop_rows_with_outliers(original_data, to_drop_outliers)

plt.figure(figsize=(10,8))
sns.boxplot(data=data_deleted_outliers, y=to_drop_outliers)
plt.title(f'Boxplot of Price after drop outleirs.')
plt.show()
```



We will use these 3 data sets for training models and evaluate and compare the model.

Both techniques I only handle for one time because Replacing outliers multiple times can lead to distortion of the data and potentially skew the distribution.

Training Model Part

Linear regression :

- We will use a linear regression model to predict the "Price" of laptops. The predictor attributes used for modeling are consistent across all datasets.

- The three datasets, each processed differently with respect to handling outliers, will be utilized to assess the impact of outlier handling techniques on the performance of the linear regression model in predicting laptop prices.

Setup modeling constant :

- Predictor is all attributes
- Target is “Price”
- Same Random state : Reference in this report is 423
- Test Size : 0.3

```python
import random

predictors = original_data.columns.tolist()
target = "Price"
state = random.randint(1,1000)
test_size = 0.3
```

Prepare a function to evaluate the model.

- Function :

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, mean_absolute_percentage_error

def evaluate_linear_regression(model, X_test, y_test):
    # Make predictions
    predictions = model.predict(X_test)

    # Print the linear regression equation
    print("Linear Regression Equation:")
    print("y =", model.intercept_, end=" ")
    for i, coef in enumerate(model.coef_):
        print("+", coef, "* X" + str(i+1), end=" ")
    print()

    # Evaluate the model
    mae = mean_absolute_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)
    mape = mean_absolute_percentage_error(y_test, predictions)

    # Print evaluation metrics
    print("\nEvaluation Metrics:")
    print(f"Mean Absolute Error (MAE): {mae}")
    print(f"Mean Absolute Percentage Error (MAPE): {mape}%")
    print(f"R-squared (R2) Score: {r2}")
```

Training model (No handling outliers)

```python
x_no_handle = data_no_handle_outliers[predictors]
x_no_handle = x_no_handle.drop(target, axis=1)
y_no_handle = data_no_handle_outliers[target]

# Split the data into training and testing sets.
x_no_handle_train, x_no_handle_test, y_no_handle_train, y_no_handle_test = model_selection.train_test_split(x_no_handle, y_no_handle, test_size=test_size, random_state=state)
# Scale train data.
scaler = preprocessing.StandardScaler()
x_no_handle_train_scaled = scaler.fit_transform(x_no_handle_train)
x_no_handle_test_scaled = scaler.transform(x_no_handle_test)
# Modeling
model_no_handle = linear_model.LinearRegression()
model_no_handle.fit(x_no_handle_train_scaled, y_no_handle_train)
```

Training model (Replace outliers with mean)

```python
x_replaced = data_outliers_replaced_with_mean[predictors]
x_replaced = x_replaced.drop(target, axis=1)
y_replaced = data_outliers_replaced_with_mean[target]

# Split the data into training and testing sets.
x_replaced_train, x_replaced_test, y_replaced_train, y_replaced_test = model_selection.train_test_split(x_replaced, y_replaced, test_size=test_size, random_state=state)
# Scale train data.
scaler = preprocessing.StandardScaler()
x_replaced_train_scaled = scaler.fit_transform(x_replaced_train)
x_replaced_test_scaled = scaler.transform(x_replaced_test)
# Modeling
model_replaced = linear_model.LinearRegression()
model_replaced.fit(x_replaced_train_scaled, y_replaced_train)
```

Training model (Drop outliers)

```python
x_drop = data_deleted_outliers[predictors]
x_drop = x_drop.drop(target, axis=1)
y_drop = data_deleted_outliers[target]

# Split the data into training and testing sets.
x_drop_train, x_drop_test, y_drop_train, y_drop_test = model_selection.train_test_split(x_drop, y_drop, test_size=test_size, random_state=state)
# Scale train data.
scaler = preprocessing.StandardScaler()
x_drop_train_scaled = scaler.fit_transform(x_drop_train)
x_drop_test_scaled = scaler.transform(x_drop_test)
# Modeling
model_drop = linear_model.LinearRegression()
model_drop.fit(x_drop_train_scaled, y_drop_train)
```

Evaluate the model

Evaluate technique:

- **MAE (Mean Absolute Error):** Average of the absolute differences between predicted and actual values. It measures the average magnitude of errors.

- **MAPE (Mean Absolute Percentage Error):** Average of the absolute percentage differences between predicted and actual values. It measures the average magnitude of percentage errors.

- **R-squared (R2) Score:** Proportion of variance in the dependent variable explained by the independent variables in the model. It ranges from 0 to 1, with higher values indicating better fit.

No handle outliers data set

Equation:

$y = 76285.80 - (1532.06 * X1) + (6602.07 * X2) + (8115.63 * X3) + (6691.74 * X4) + (23009.13 * X5) + (5577.03 * X6) + (14730.81 * X7) + (4409.49 * X8) - (2727.68 * X9)$

Mean Absolute Error **(MAE)**	**21007.95**
Mean Absolute Percentage Error **(MAPE)**	**0.276**
R-squared **(R2)** Score	**0.530**

Replace outliers with mean

Equation:

$y = 68545.19 - (921.19 * X1) + (393.18 * X2) + (5010.77 * X3) + (5020.03 * X4) + (6424.01 * X5) - (1514.39 * X6) + (8690.12 * X7) + (5438.34 * X8) - (2430.38 * X9)$

Mean Absolute Error **(MAE)**	**15668.13**
Mean Absolute Percentage Error **(MAPE)**	**0.250**
R-squared **(R2)** Score	**0.448**

Drop outliers

Equation:

$y = 66372.31 - (1462.28 * X1) - (97.06 * X2) + (6769.80 * X3) + (3908.28 * X4) + (5775.34 * X5) - (1434.02 * X6) + (5669.88 * X7) + (5401.64 * X8) - (1344.15 * X9)$

Mean Absolute Error **(MAE)**	**14043.48**
Mean Absolute Percentage Error **(MAPE)**	**0.232**
R-squared **(R2)** Score	**0.428**

Comparison each models

outliers Replace with mean vs No handling

- Replace outlier with mean can reduce MAE and MAPE
- Replace outlier with mean tend to increase R-squared (R2)score

Conclusion :
Replacing outliers with the mean tends to reduce MAE and MAPE because it brings extreme values closer to the central majority. However, it often decreases the R-squared score because it flattens the data distribution, compromising the model's ability to capture true variability.

outliers Replace with mean vs drop outliers

- Replace outliers with mean tend to have MAPE value larger than MAPE of data which drop rows of outliers.
- But Replace outliers with mean tend to have R2 score larger than R2 score of data which drop rows of outliers.

Conclusion :
Replacing outliers with the mean may inflate the MAPE due to distortion of the data distribution, it can improve the R-squared score by preserving more data points and providing a better representation of the overall trend in the data.

Handle outliers vs no Handle outliers

-

> **Conclusion :**
> No matter what techniques are used in managing outliers It will result in the model's predictions being more accurate.

Conclusion technique.

Benefit consideration:

- **Impact Reduction:** Replacing outliers with the mean can indeed help mitigate the influence of extreme values on statistical measures such as the mean and standard deviation.
- **Maintaining Data Structure:** Imputing with the mean does indeed preserve the overall structure of the dataset.
- **Potential Biases:** However, imputing with the mean can introduce biases, particularly if the outliers represent genuine patterns in the data.
- **Sensitive to Outliers:** This method doesn't eliminate the impact of outliers; instead, it redistributes their influence across the dataset.

Cons consideration:

- **Distortion of Data Distribution:** Replacing outliers with the mean can distort the original distribution of the data. By assigning extreme values to the mean, the representation of the data's spread and variability may become skewed, leading to potential misinterpretation of the dataset's characteristics.

- **Loss of Information:** Outliers can sometimes carry valuable information or signal about unique patterns or events in the data.

Reference source

- Dataset : Laptop Prices Dataset (kaggle.com)

- Definition of Outliers:
  Navigating Outliers for Accurate Data Analysis & Decisions (statusneo.com)

- Sklearn metrics:
  บันทึก training data science EP 4: Scikit-learn & Linear Regression – แนวโน้มของเส้นตรง (bluebirz.net)

- Pandas: https://pandas.pydata.org/