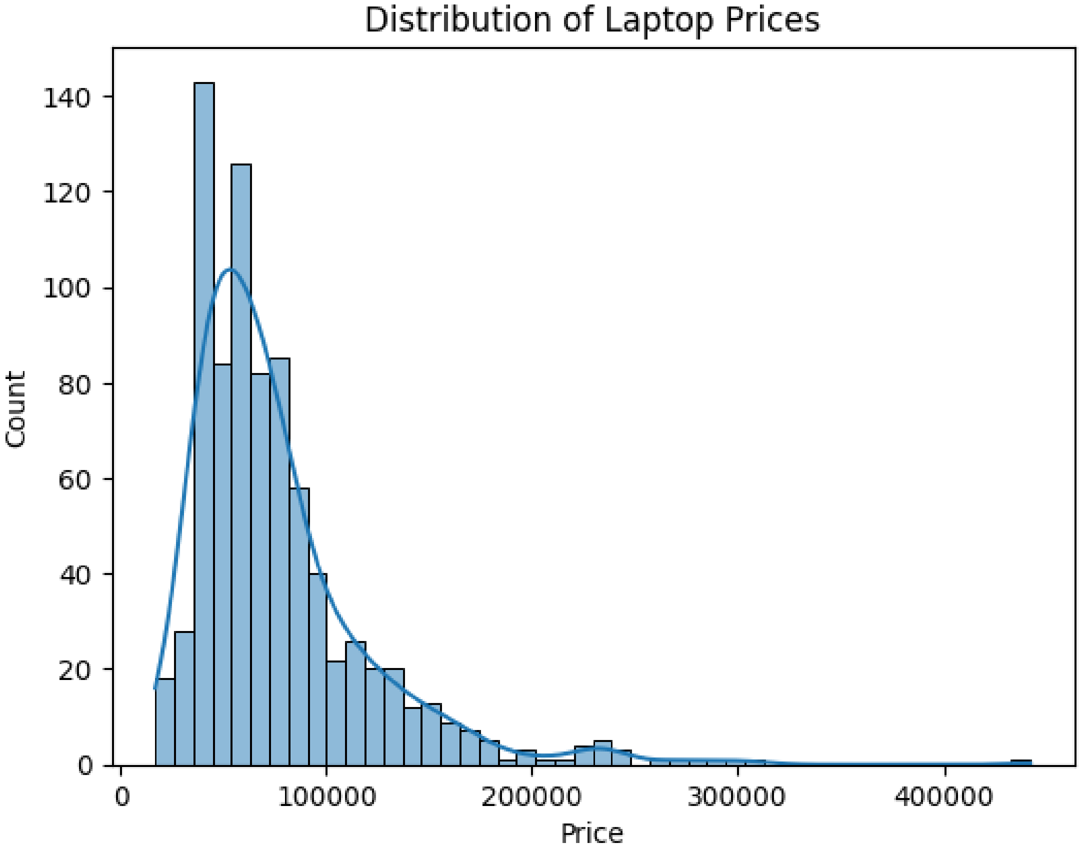
```
# Filepath
file_path = 'laptop prices.csv'

# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Step 1: Load the dataset
df = pd.read_csv(file_path)

# Data Exploration
print("Dataset Head:\n", df.head())
print("\nDataset Info:\n", df.info())
print("\nSummary Statistics:\n", df.describe())
```

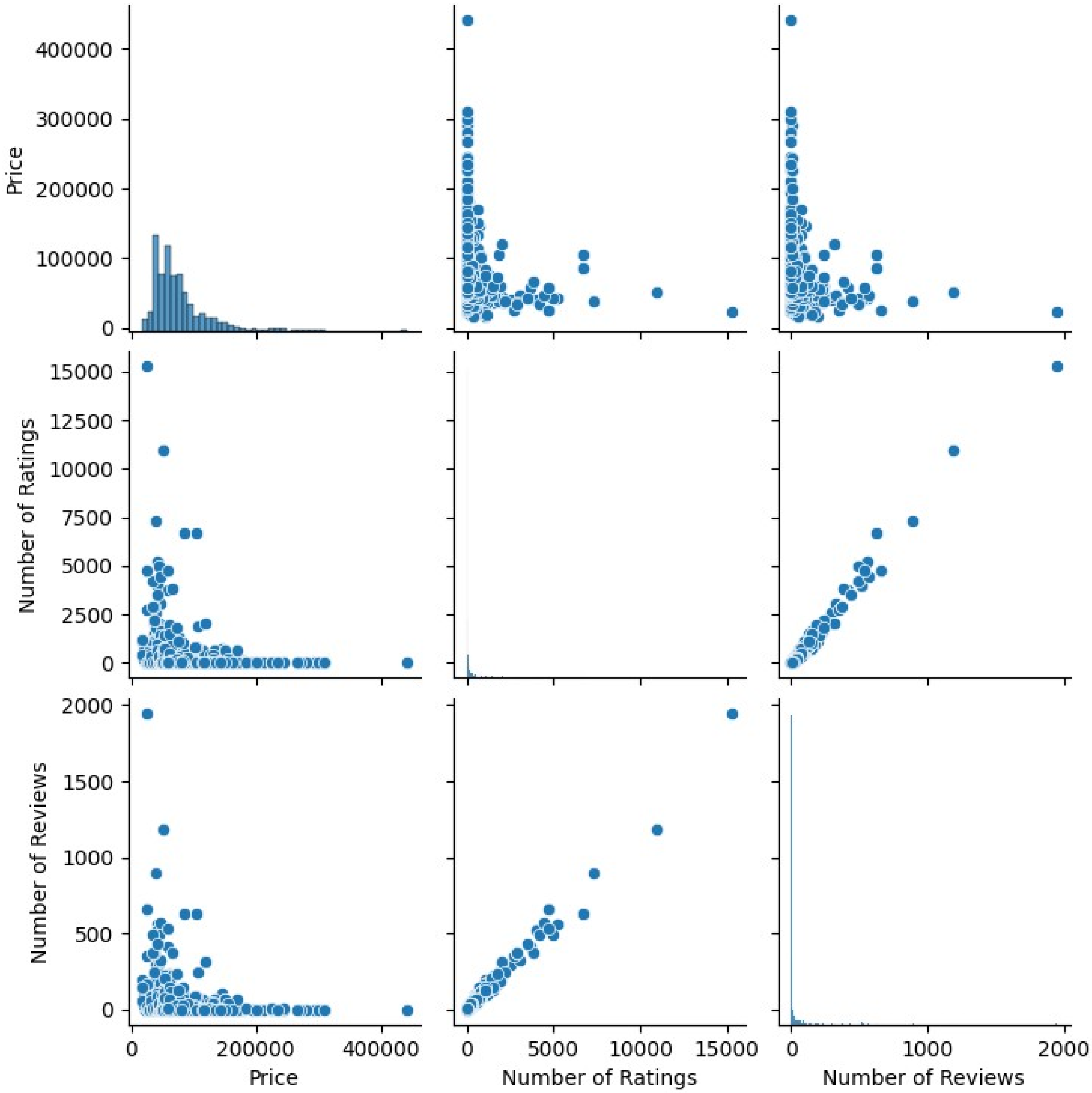
```
Dataset Head:
     brand processor_brand processor_name processor_gnrtn ram_gb
ram_type  \
0    ASUS           Intel        Core i3            10th   4 GB
DDR4
1  Lenovo           Intel        Core i3            10th   4 GB
DDR4
2  Lenovo           Intel        Core i3            10th   4 GB
DDR4
3    ASUS           Intel        Core i5            10th   8 GB
DDR4
4    ASUS           Intel   Celeron Dual   Not Available   4 GB
DDR4

      ssd      hdd       os  os_bit graphic_card_gb  weight
warranty  \
0    0 GB  1024 GB  Windows  64-bit            0 GB  Casual  No
warranty
1    0 GB  1024 GB  Windows  64-bit            0 GB  Casual  No
warranty
2    0 GB  1024 GB  Windows  64-bit            0 GB  Casual  No
warranty
3  512 GB     0 GB  Windows  32-bit            2 GB  Casual  No
warranty
4    0 GB   512 GB  Windows  64-bit            0 GB  Casual  No
warranty
```

```
  Touchscreen msoffice  Price   rating  Number of Ratings  Number of
Reviews
0          No       No  34649  2 stars                  3
0
1          No       No  38999  3 stars                 65
5
2          No       No  39999  3 stars                  8
1
3          No       No  69990  3 stars                  0
0
4          No       No  26990  3 stars                  0
0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 823 entries, 0 to 822
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   brand              823 non-null    object
 1   processor_brand    823 non-null    object
 2   processor_name     823 non-null    object
 3   processor_gnrtn    823 non-null    object
 4   ram_gb             823 non-null    object
 5   ram_type           823 non-null    object
 6   ssd                823 non-null    object
 7   hdd                823 non-null    object
 8   os                 823 non-null    object
 9   os_bit             823 non-null    object
 10  graphic_card_gb    823 non-null    object
 11  weight             823 non-null    object
 12  warranty           823 non-null    object
 13  Touchscreen        823 non-null    object
 14  msoffice           823 non-null    object
 15  Price              823 non-null    int64 
 16  rating             823 non-null    object
 17  Number of Ratings  823 non-null    int64 
 18  Number of Reviews  823 non-null    int64 
dtypes: int64(3), object(16)
memory usage: 122.3+ KB

Dataset Info:
 None

Summary 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

# Visualize the distribution of the target variable (Price)
sns.histplot(df['Price'], kde=True)
plt.title('Distribution of Laptop Prices')
plt.show()
```



```
# Visualize relationships between features and target variable
sns.pairplot(df)
plt.show()
```



```
# Step 2: Data Preprocessing and Feature Engineering
# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())


Missing Values:
 brand                0
processor_brand      0
processor_name       0
processor_gnrtn      0
ram_gb               0
ram_type             0
```

```
ssd                    0
hdd                    0
os                     0
os_bit                 0
graphic_card_gb        0
weight                 0
warranty               0
Touchscreen            0
msoffice               0
Price                  0
rating                 0
Number of Ratings      0
Number of Reviews      0
dtype: int64
```

```
# Handle missing values (if any)
df = df.dropna()  # Dropping rows with missing values for simplicity

# Convert categorical columns to numerical
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

# Feature scaling
scaler = StandardScaler()
df[df.columns] = scaler.fit_transform(df[df.columns])

# Step 3: Split the data into training and testing sets
X = df.drop('Price', axis=1)  # Features
y = df['Price']  # Target variable

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

# Step 4: Model Selection and Training
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(n_estimators=100,
random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100,
random_state=42)
}

trained_models = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    trained_models[model_name] = model
```

```python
# Step 5: Model Evaluation
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    return mae, rmse

for model_name, model in trained_models.items():
    mae, rmse = evaluate_model(model, X_test, y_test)
    print(f"{model_name} - MAE: {mae}, RMSE: {rmse}")
```

```
Linear Regression - MAE: 0.432176348476466, RMSE: 0.6885798734737727
Random Forest - MAE: 0.2860755773765587, RMSE: 0.5846661440561662
Gradient Boosting - MAE: 0.31462902291940364, RMSE: 0.601076982247702
```

```python
# Cross-validation
def cross_val(model, X, y):
    scores = cross_val_score(model, X, y,
scoring='neg_mean_squared_error', cv=5)
    rmse_scores = np.sqrt(-scores)
    return rmse_scores

for model_name, model in models.items():
    scores = cross_val(model, X, y)
    print(f"{model_name} - Cross-Validation RMSE: {scores.mean()}")
```

```
Linear Regression - Cross-Validation RMSE: 0.7828636881582345
Random Forest - Cross-Validation RMSE: 0.7368779115436246
Gradient Boosting - Cross-Validation RMSE: 0.6880702219334448
```

```python
# Sample prediction
sample_data = X_test.iloc[0].values.reshape(1, -1)
predicted_price = trained_models['Linear
Regression'].predict(sample_data)
print(f'Sample Predicted Price: {predicted_price[0]}')
```

```
Sample Predicted Price: 0.6900953951677803
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```python
# Save the trained models, encoders, and scaler
import pickle

for model_name, model in trained_models.items():
    with open(f'/content/{model_name.replace(" ",
"_").lower()}_model.pkl', 'wb') as file:
        pickle.dump(model, file)
```

```
with open('/content/label_encoders.pkl', 'wb') as file:
    pickle.dump(label_encoders, file)

with open('/content/scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)
```