Laptop Price Prediction

```
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.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler
```

```
df = pd.read_csv("laptopPrice.csv")
df
```

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_card_gb	weight
0	ASUS	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual
1	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual
2	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GB	Casual
3	ASUS	Intel	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	32-bit	2 GB	Casual
4	ASUS	Intel	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	64-bit	0 GB	Casual
...	...	...	...	...	...	...	...	...	...	...	...	...
818	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	64-bit	0 GB	Casual
819	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	64-bit	0 GB	Casual
820	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	64-bit	4 GB	Casual
821	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	64-bit	4 GB	Casual
822	Lenovo	AMD	Ryzen 5	10th	8 GB	DDR4	512 GB	0 GB	DOS	64-bit	0 GB	ThinNlight

823 rows × 19 columns

Next steps: Generate code with df View recommended plots

Data Preprocessing

```
df.shape
```

```
(823, 19)
```

```
df.columns
```

```
Index(['brand', 'processor_brand', 'processor_name', 'processor_gnrtn',
       'ram_gb', 'ram_type', 'ssd', 'hdd', 'os', 'os_bit', 'graphic_card_gb',
       'weight', 'warranty', 'Touchscreen', 'msoffice', 'Price', 'rating',
       'Number of Ratings', 'Number of Reviews'],
      dtype='object')
```

```
df.dropna()
```

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd
0	ASUS	Intel	Core i3	10th	4 GB	DDR4	0 GB
1	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB
2	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB
3	ASUS	Intel	Core i5	10th	8 GB	DDR4	512 GB
4	ASUS	Intel	Celeron Dual	Not Available	4 GB	DDR4	0 GB
...	...	...	...	...	...	...	...
818	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB
819	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB
820	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB
821	ASUS	AMD	Ryzen 9	Not Available	4 GB	DDR4	1024 GB
822	Lenovo	AMD	Ryzen 5	10th	8 GB	DDR4	512 GB

823 rows × 19 columns

```
df.info
```

```
pandas.core.frame.DataFrame.info
def info(verbose: bool | None=None, buf: WriteBuffer[str] | None=None, max_cols:
int | None=None, memory_usage: bool | str | None=None, show_counts: bool |
None=None) -> None
```

```
---  ------    --------------    -----
 0   column_1  1000000 non-null  object
 1   column_2  1000000 non-null  object
 2   column_3  1000000 non-null  object
dtypes: object(3)
memory usage: 165.9 MB
```

```
df.describe
```

```
pandas.core.generic.NDFrame.describe
def describe(percentiles=None, include=None, exclude=None) -> NDFrameT
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py
Generate descriptive statistics.

Descriptive statistics include those that summarize the central
tendency, dispersion and shape of a
dataset's distribution, excluding ``NaN`` values.
```

```
data = df.drop(columns=['Number of Ratings', 'Number of Reviews'])
df = pd.get_dummies(data)


df.isnull()
```

	Price	brand_APPLE	brand_ASUS	brand_Avita	brand_DELL	brand_HP	brand_Lenovo	br
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
818	False	False	False	False	False	False	False	
819	False	False	False	False	False	False	False	
820	False	False	False	False	False	False	False	
821	False	False	False	False	False	False	False	
822	False	False	False	False	False	False	False	

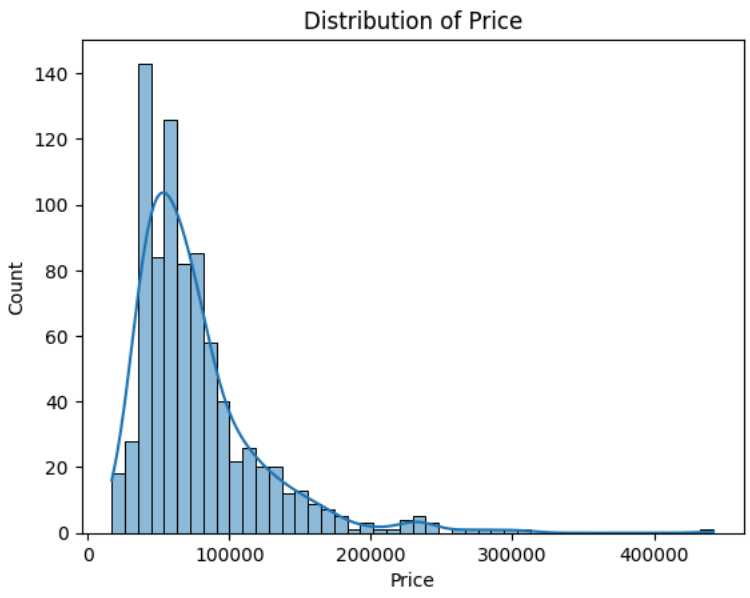
823 rows × 78 columns

```
df.isnull().sum()
```

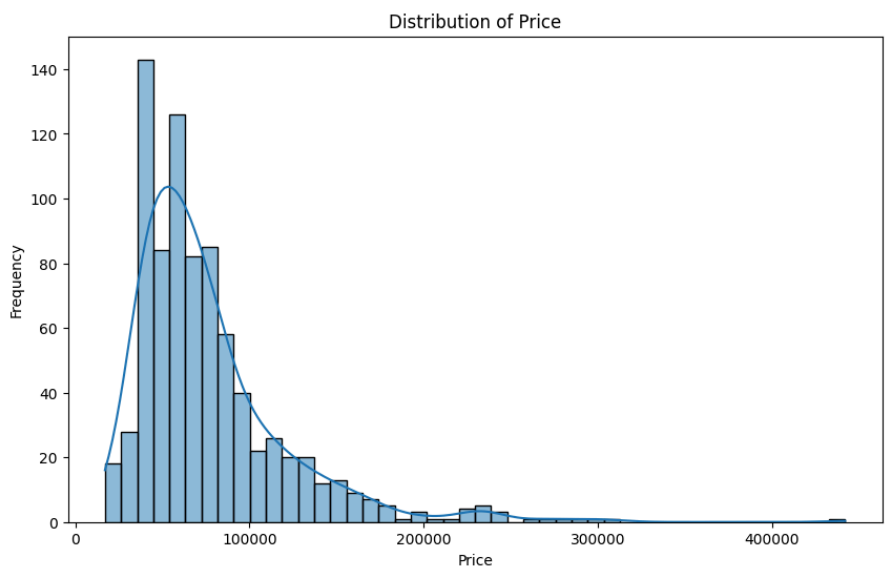
```
Price             0
brand_APPLE       0
brand_ASUS        0
brand_Avita       0
brand_DELL        0
                 ..
rating_1 star     0
rating_2 stars    0
rating_3 stars    0
rating_4 stars    0
rating_5 stars    0
Length: 78, dtype: int64
```

Exploratory Data Analysis

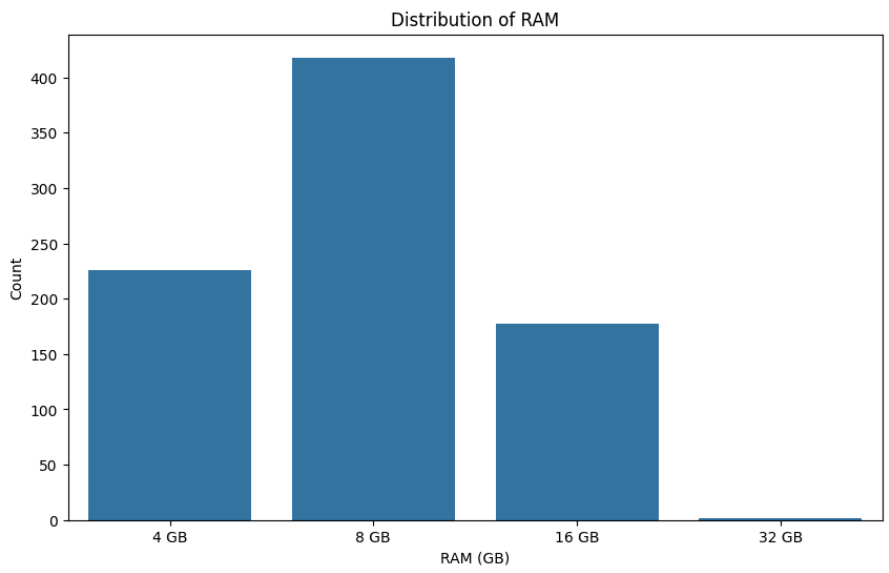
```
sns.histplot(df['Price'], kde=True)
plt.title('Distribution of Price')
plt.show()
```



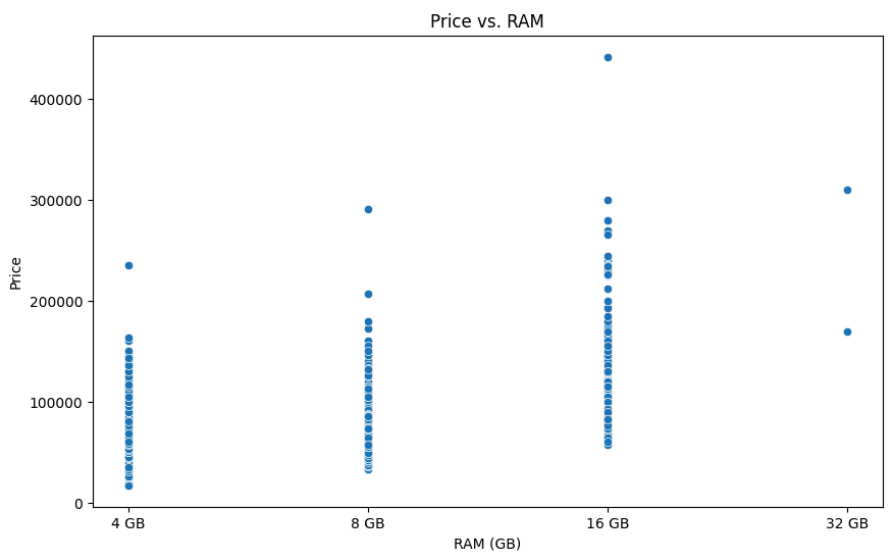
```
plt.figure(figsize=(10, 6))
sns.histplot(data['Price'], kde=True)
plt.title('Distribution of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.countplot(x='ram_gb', data=data)
plt.title('Distribution of RAM')
plt.xlabel('RAM (GB)')
plt.ylabel('Count')
plt.show()
```



```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='ram_gb', y='Price', data=data)
plt.title('Price vs. RAM')
plt.xlabel('RAM (GB)')
plt.ylabel('Price')
plt.show()
```



Model Building

```
X = df.drop(columns=['Price'])
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
gb_regressor = GradientBoostingRegressor()
```

```
gb_regressor.fit(X_train, y_train)
```

```
▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

Model Evaluation

```
y_pred_train = gb_regressor.predict(X_train)
```

```
y_pred_test = gb_regressor.predict(X_test)
```

```
cv_scores = cross_val_score(gb_regressor, X, y, cv=5, scoring='neg_mean_squared_error')
print("Cross-Validation MSE:", -cv_scores.mean())
```

```
Cross-Validation MSE: 1084536480.5838351
```

Applying Random Forest

```
rf_regressor = RandomForestRegressor()
rf_regressor.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
y_pred_train_rf = rf_regressor.predict(X_train)
y_pred_test_rf = rf_regressor.predict(X_test)

mse_train_rf = mean_squared_error(y_train, y_pred_train_rf)
mse_test_rf = mean_squared_error(y_test, y_pred_test_rf)

mae_train_rf = mean_absolute_error(y_train, y_pred_train_rf)
mae_test_rf = mean_absolute_error(y_test, y_pred_test_rf)

r2_train_rf = r2_score(y_train, y_pred_train_rf)
r2_test_rf = r2_score(y_test, y_pred_test_rf)

print("\nRandom Forest Regressor:")
print("Mean Squared Error:", mse_train_rf)
print("Mean Absolute Error:", mae_train_rf)
print("R² Score:", r2_train_rf)
```

```
Random Forest Regressor:
Mean Squared Error (Train): 109004799.99510047
Mean Squared Error (Test): 847109644.6252016
Mean Absolute Error (Train): 5153.513912993922
Mean Absolute Error (Test): 14653.341007011675
R² Score (Train): 0.9468854469048346
R² Score (Test): 0.5653188474553668
```

```
cv_scores_rf = cross_val_score(rf_regressor, X, y, cv=5, scoring='neg_mean_squared_error')
print("Cross-Validation MSE:", -cv_scores_rf.mean())
```

```
Cross-Validation MSE: 1140308074.9998791
```