Laptop Price & Catagory Prediction

July 22, 2023

[1]:
```
#data source: https://www.kaggle.com/datasets/kuchhbhi/latest-laptop-price-list
```

[3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[19]:
```
df=pd.read_csv("Cleaned_Laptop_data.csv")
display(df.head(5))
```

	brand	model	processor_brand	processor_name	processor_gnrtn	\
0	Lenovo	A6-9225	AMD	A6-9225 Processor	10th	
1	Lenovo	Ideapad	AMD	APU Dual	10th	
2	Avita	PURA	AMD	APU Dual	10th	
3	Avita	PURA	AMD	APU Dual	10th	
4	Avita	PURA	AMD	APU Dual	10th	

	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_card_gb	\
0	4 GB GB	DDR4	0 GB	1024 GB	Windows	64-bit	0	
1	4 GB GB	DDR4	0 GB	512 GB	Windows	64-bit	0	
2	4 GB GB	DDR4	128 GB	0 GB	Windows	64-bit	0	
3	4 GB GB	DDR4	128 GB	0 GB	Windows	64-bit	0	
4	4 GB GB	DDR4	256 GB	0 GB	Windows	64-bit	0	

	warranty	Touchscreen	latest_price	old_price	star_rating	ratings	\
0	0	No	24990	32790	3.7	63	
1	0	No	19590	21325	3.6	1894	
2	0	No	19990	27990	3.7	1153	
3	0	No	21490	27990	3.7	1153	
4	0	No	24990	33490	3.7	1657	

	reviews
0	12
1	256
2	159
3	159
4	234

About Data

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 896 entries, 0 to 895
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   brand            896 non-null    object
 1   model            896 non-null    object
 2   processor_brand  896 non-null    object
 3   processor_name   896 non-null    object
 4   processor_gnrtn  896 non-null    object
 5   ram_gb           896 non-null    object
 6   ram_type         896 non-null    object
 7   ssd              896 non-null    object
 8   hdd              896 non-null    object
 9   os               896 non-null    object
 10  os_bit           896 non-null    object
 11  graphic_card_gb  896 non-null    int64
 12  warranty         896 non-null    int64
 13  Touchscreen      896 non-null    object
 14  latest_price     896 non-null    int64
 15  old_price        896 non-null    int64
 16  star_rating      896 non-null    float64
 17  ratings          896 non-null    int64
 18  reviews          896 non-null    int64
dtypes: float64(1), int64(6), object(12)
memory usage: 133.1+ KB
```

[6]:
```
df.dtypes
```

[6]:
```
brand              object
model              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     int64
warranty            int64
Touchscreen        object
latest_price        int64
old_price           int64
star_rating       float64
```

```
ratings              int64
reviews              int64
dtype: object
```

[7]:
```
df.nunique()
```

[7]:
```
brand               21
model              117
processor_brand      5
processor_name      28
processor_gnrtn      8
ram_gb               4
ram_type             6
ssd                  8
hdd                  4
os                   3
os_bit               2
graphic_card_gb      5
warranty             4
Touchscreen          2
latest_price       429
old_price          564
star_rating         30
ratings            310
reviews            152
dtype: int64
```

[8]:
```
df.isnull()
```

[8]:
```
     brand  model  processor_brand  processor_name  processor_gnrtn  ram_gb  \
0    False  False            False           False            False   False
1    False  False            False           False            False   False
2    False  False            False           False            False   False
3    False  False            False           False            False   False
4    False  False            False           False            False   False
..     …      …                …               …                …       …
891  False  False            False           False            False   False
892  False  False            False           False            False   False
893  False  False            False           False            False   False
894  False  False            False           False            False   False
895  False  False            False           False            False   False

     ram_type    ssd    hdd     os  os_bit  graphic_card_gb  warranty  \
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
..        …    …    …    …     …              …       …
891     False  False  False  False   False            False     False
892     False  False  False  False   False            False     False
893     False  False  False  False   False            False     False
894     False  False  False  False   False            False     False
895     False  False  False  False   False            False     False

     Touchscreen  latest_price  old_price  star_rating  ratings  reviews
0          False         False      False        False    False    False
1          False         False      False        False    False    False
2          False         False      False        False    False    False
3          False         False      False        False    False    False
4          False         False      False        False    False    False
..           …             …        …          …      …      …
891        False         False      False        False    False    False
892        False         False      False        False    False    False
893        False         False      False        False    False    False
894        False         False      False        False    False    False
895        False         False      False        False    False    False

[896 rows x 19 columns]
```

[ ]:

Data processing

[9]:
```
df.columns
```

[9]:
```
Index(['brand', 'model', 'processor_brand', 'processor_name',
       'processor_gnrtn', 'ram_gb', 'ram_type', 'ssd', 'hdd', 'os', 'os_bit',
       'graphic_card_gb', 'warranty', 'Touchscreen', 'latest_price',
       'old_price', 'star_rating', 'ratings', 'reviews'],
      dtype='object')
```

[10]:
```
#delet some columns
df = df.drop(['old_price', 'star_rating', 'ratings', 'reviews'], axis=1)
df.head()
```

[10]:
```
    brand    model processor_brand     processor_name processor_gnrtn  \
0  Lenovo  A6-9225             AMD  A6-9225 Processor            10th
1  Lenovo  Ideapad             AMD           APU Dual            10th
2   Avita     PURA             AMD           APU Dual            10th
3   Avita     PURA             AMD           APU Dual            10th
4   Avita     PURA             AMD           APU Dual            10th

    ram_gb ram_type     ssd      hdd       os  os_bit  graphic_card_gb  \
0  4 GB GB     DDR4    0 GB  1024 GB  Windows  64-bit                0
```

```
1  4 GB GB     DDR4    0 GB   512 GB  Windows  64-bit               0
2  4 GB GB     DDR4  128 GB     0 GB  Windows  64-bit               0
3  4 GB GB     DDR4  128 GB     0 GB  Windows  64-bit               0
4  4 GB GB     DDR4  256 GB     0 GB  Windows  64-bit               0

   warranty Touchscreen  latest_price
0         0          No         24990
1         0          No         19590
2         0          No         19990
3         0          No         21490
4         0          No         24990
```

[11]:
```python
import pandas as pd

# Read the CSV file
df = pd.read_csv("Cleaned_Laptop_data.csv")

# Clean and convert "ram_gb" column to integers
df["ram_gb"] = df["ram_gb"].str.extract("(\d+)").astype(int)
df["ssd"] = df["ssd"].str.extract("(\d+)").astype(int)
df["graphic_card_gb"].astype(int)


# Define a function to map the laptop category based on specific criteria
def map_laptop_category(row):
    if row["ram_gb"] >= 8 and row["ssd"] >= 256:
        if "Core i5" in row["processor_name"] or "Ryzen 5" in␣
 ↪row["processor_name"]:
            return "Business users/Business professionals"

    if row["ram_gb"] >= 16 and row["ssd"] >= 512:
        if "Core i7" in row["processor_name"] or "Ryzen 7" in␣
 ↪row["processor_name"]:
            if row["graphic_card_gb"] >= 4:
                return "Creatives/Creative professionals"
            elif row["graphic_card_gb"] >= 6:
                return "Gamers"
            elif row["graphic_card_gb"] >= 2:
                return "Data Science/Analytics Students"


    if row["ram_gb"] >= 4 or row["ssd"] >= 256:
        if "Core i3" in row["processor_name"] or "Ryzen 3" in␣
 ↪row["processor_name"]:
            return "General Student"
```

```
    if row["ram_gb"] >= 8 and row["ssd"] >= 256:
         return "Computer Science/Engineering Students"

    return "Other"


# Apply the function to create the "Laptop Category" column
df["Laptop_Category"] = df.apply(map_laptop_category, axis=1)

# Save the modified DataFrame back to a new CSV file
df.to_csv("Cleaned_Laptop_data_with_category.csv", index=False)
```

[20]:
```
display(df.head(5))
```

```
    brand    model processor_brand     processor_name processor_gnrtn  \
0  Lenovo  A6-9225             AMD  A6-9225 Processor            10th
1  Lenovo  Ideapad             AMD           APU Dual            10th
2   Avita     PURA             AMD           APU Dual            10th
3   Avita     PURA             AMD           APU Dual            10th
4   Avita     PURA             AMD           APU Dual            10th

    ram_gb ram_type     ssd      hdd       os  os_bit  graphic_card_gb  \
0  4 GB GB     DDR4    0 GB  1024 GB  Windows  64-bit                0
1  4 GB GB     DDR4    0 GB   512 GB  Windows  64-bit                0
2  4 GB GB     DDR4  128 GB     0 GB  Windows  64-bit                0
3  4 GB GB     DDR4  128 GB     0 GB  Windows  64-bit                0
4  4 GB GB     DDR4  256 GB     0 GB  Windows  64-bit                0

   warranty Touchscreen  latest_price  old_price  star_rating  ratings  \
0         0          No         24990      32790          3.7       63
1         0          No         19590      21325          3.6     1894
2         0          No         19990      27990          3.7     1153
3         0          No         21490      27990          3.7     1153
4         0          No         24990      33490          3.7     1657

   reviews
0       12
1      256
2      159
3      159
4      234
```

Data Analysis

0.0.1 Which Is own Highest Brand Value?

[13]:
```
avg_star_rating_by_brand = df.groupby('brand')['star_rating'].mean()


print(f' The Highest Brand Value own:  {avg_star_rating_by_brand.idxmax()} ,␣
  ↪Star Rating: {avg_star_rating_by_brand.sort_values(ascending=False)[0]}')
print("")
print(avg_star_rating_by_brand.sort_values(ascending=False))
```

```
 The Highest Brand Value own:  APPLE , Star Rating: 4.7178571428571425

brand
APPLE        4.717857
ALIENWARE    4.400000
realme       4.400000
Nokia        4.300000
RedmiBook    4.266667
Infinix      4.250000
Mi           4.250000
MICROSOFT    4.233333
Vaio         3.960000
iball        3.800000
Smartron     3.633333
HP           3.212676
MSI          3.138462
acer         3.055172
lenovo       2.933333
Lenovo       2.914865
DELL         2.863636
ASUS         2.681496
LG           1.820000
Avita        1.805556
SAMSUNG      0.000000
Name: star_rating, dtype: float64
```

0.0.2 Which Laptop Model is Best?

[14]:
```
avg_ratings_by_brand = df.groupby(['brand','model'])['ratings'].mean()


print(f' Best Laptop Model Name:  {avg_ratings_by_brand.idxmax()} , Ratings:␣
  ↪{avg_ratings_by_brand.sort_values(ascending=False)[0]}')
print("")
print(avg_ratings_by_brand.sort_values(ascending=False))
```

```
 Best Laptop Model Name:  ('ASUS', 'Celeron') , Ratings: 15279.0
```

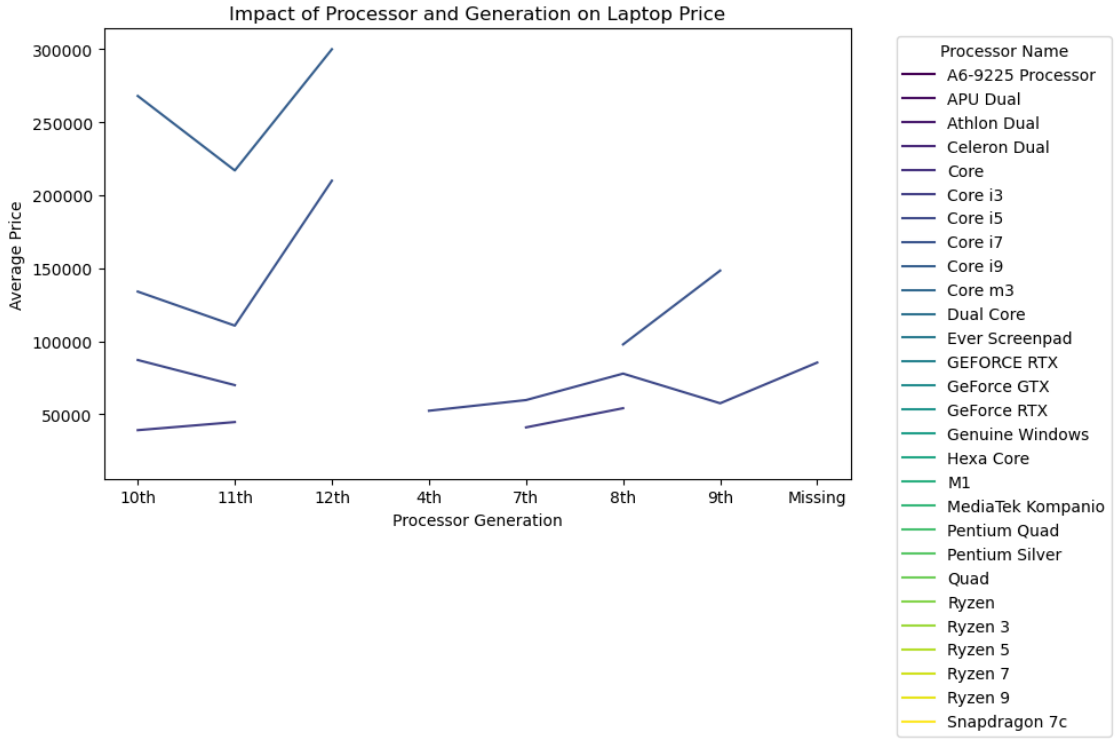
```
brand   model
ASUS    Celeron       15279.0
realme  Book           6352.0
        Book(Slim)     3470.0
HP      Chromebook     2897.6
ASUS    APU            2887.0
                       …
MSI     GS66              0.0
        Creator           0.0
        Delta             0.0
DELL    DELL              0.0
MSI     Sword             0.0
Name: ratings, Length: 140, dtype: float64
```

0.0.3 Laptop processor and generation impact on price?

[15]:
```
# Grouping by 'processor_name' and 'processor_gnrtn' and calculating the␣
 ↪average of 'price' for each group
avg_price_by_processor_generation = df.groupby(['processor_name',␣
 ↪'processor_gnrtn'])['latest_price'].mean().reset_index()

# Pivot the data to create a table with processor_name as columns,␣
 ↪processor_gnrtn as index, and average price as values
pivot_table = avg_price_by_processor_generation.pivot(index='processor_gnrtn',␣
 ↪columns='processor_name', values='latest_price')

# Create a bar chart to visualize the impact of processor_name and␣
 ↪processor_gnrtn on price
ax = pivot_table.plot(kind='line', figsize=(10, 6), colormap='viridis')
ax.set_xlabel('Processor Generation')
ax.set_ylabel('Average Price')
ax.set_title('Impact of Processor and Generation on Laptop Price')
plt.xticks(rotation=0)
plt.legend(title='Processor Name', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

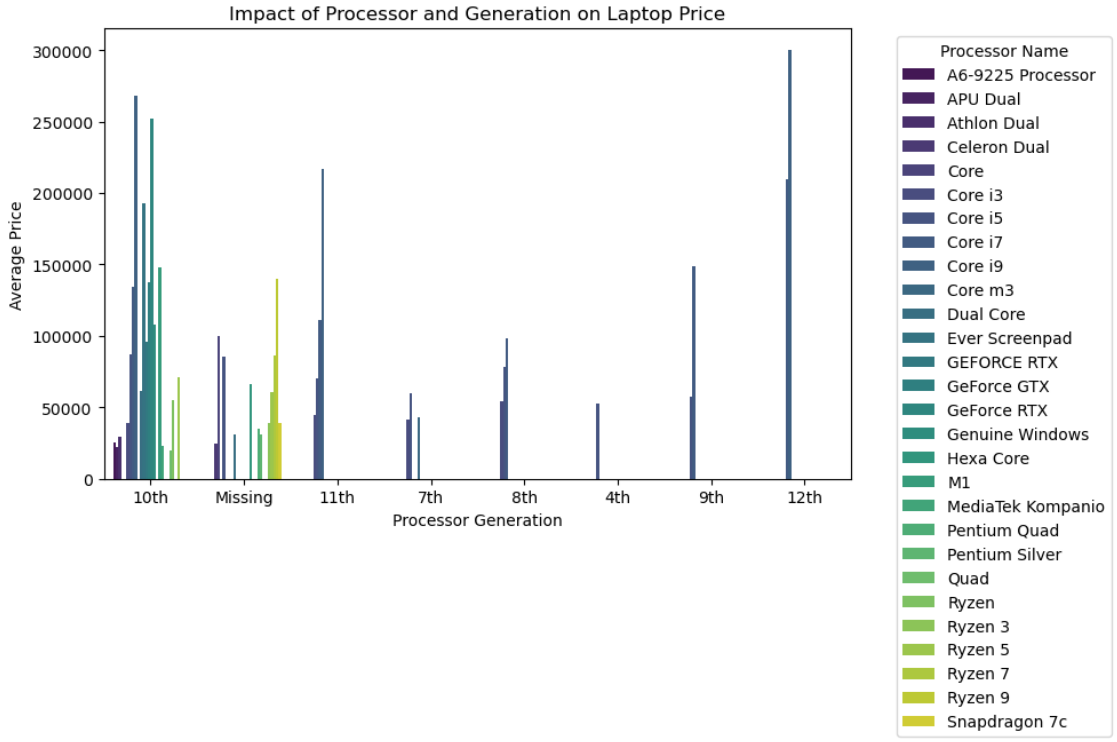


[16]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named 'df' with columns 'processor_name',␣
 ↪'processor_gnrtn', and 'price'

# Grouping by 'processor_name' and 'processor_gnrtn' and calculating the␣
 ↪average of 'price' for each group
avg_price_by_processor_generation = df.groupby(['processor_name',␣
 ↪'processor_gnrtn'])['latest_price'].mean().reset_index()

# Create a bar plot using Seaborn
plt.figure(figsize=(10, 6))
sns.barplot(data=avg_price_by_processor_generation, x='processor_gnrtn',␣
 ↪y='latest_price', hue='processor_name', palette='viridis')
plt.xlabel('Processor Generation')
plt.ylabel('Average Price')
plt.title('Impact of Processor and Generation on Laptop Price')
plt.legend(title='Processor Name', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



0.0.4 Predict Laptop Price & Catagory Using RandomForest

[17]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, mean_squared_error
from flask import Flask, render_template, request
import streamlit as st

# Read the CSV file
df = pd.read_csv("Cleaned_Laptop_data_with_category.csv")



# Convert "processor_name" column to string data type
df["processor_name"] = df["processor_name"].astype(str)

# Feature Selection: Select relevant features for prediction
X = df[['ram_gb', 'ssd', 'processor_name', 'graphic_card_gb']]
y_category = df['Laptop_Category']
y_price = df['latest_price']

# Convert categorical features to numerical using one-hot encoding
```

```
X = pd.get_dummies(X, columns=['processor_name'])

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_category_train, y_category_test, y_price_train, y_price_test␣
 ↪= train_test_split(
    X, y_category, y_price, test_size=0.2, random_state=42)

# Train the Laptop Category Prediction Model (Random Forest Classifier)
category_model = RandomForestClassifier(n_estimators=100, random_state=42)
category_model.fit(X_train, y_category_train)

# Train the Price Prediction Model (Linear Regression)
price_model = LinearRegression()
price_model.fit(X_train, y_price_train)

# Predict the Laptop Category and Price for the test data
y_category_pred = category_model.predict(X_test)
y_price_pred = price_model.predict(X_test)

# Evaluate the Laptop Category Prediction Model
category_accuracy = accuracy_score(y_category_test, y_category_pred)
print("Laptop Category Prediction Accuracy:", category_accuracy)

# Evaluate the Price Prediction Model
price_rmse = mean_squared_error(y_price_test, y_price_pred, squared=False)
print("Price Prediction Root Mean Squared Error (RMSE):", price_rmse)

# Function to input new data and get predictions
def predict_laptop_category_and_price(new_data):
    # Create a DataFrame with all the columns used during training
    new_data = pd.get_dummies(new_data, columns=['processor_name'])
    new_data = new_data.reindex(columns=X.columns, fill_value=0)  # Fill␣
 ↪missing columns with 0

    # Predict Laptop Category
    laptop_category_prediction = category_model.predict(new_data)

    # Predict Price
    laptop_price_prediction = price_model.predict(new_data)

    return laptop_category_prediction[0], laptop_price_prediction[0]

# Input new data as a dictionary
new_data = {
    'ram_gb': [16],
    'ssd': [1024],
    'processor_name': ['Core i7'],
```

```
    'graphic_card_gb': [4]
}

# Create a DataFrame from the new data
new_data_df = pd.DataFrame(new_data)

# Get predictions for the new data
predicted_category, predicted_price =␣
 ↪predict_laptop_category_and_price(new_data_df)

# Print the predicted Laptop Category and Price
print("Predicted Laptop Category:", predicted_category)
print("Predicted Price:", predicted_price)
```

```
Laptop Category Prediction Accuracy: 0.9944444444444445
Price Prediction Root Mean Squared Error (RMSE): 26521.790658047805
Predicted Laptop Category: Creatives/Creative professionals
Predicted Price: 137882.11586507128
```

[18]:
```
### This prediction deploy on streamlit.Please chek it.

### Link: https://laptoppriceandcatagoryprediction-927z42fle45.streamlit.app/
```

[ ]:
```

```