Laptops Clustering

4th June 2024

1 Zbiór danych

1.1 Podział zbioru na 2 części

Zbiór danych, z którym będziemy pracowały, pochodzi ze źródła [1]. Na samym początku ten zbiór zawierał 991 rekordów, jednak 20% z nich, czyli 199 rekordów, przed rozpoczęciem EDA zostały odseparowane i nie były przez nas używane podczas projektu - ta część danych została oddana zespołowi walidacyjnemu. Korzystałyśmy zatem tylko z pozostałych 80% danych, czyli 792 rekordów.

1.2 Zmienne w naszych danych

Table 1: Opis zmiennych

Nazwa zmiennej (kolumny)	**Opis**	**Typ danych**
index	Numer rekordu w ramce danych	int64
brand	Nazwa marki laptopa	object
Model	Pełna nazwa modelu laptopa	object
Price	Cena laptopa w rupiach indyjskich	int64
Rating	Ranking każdego laptopa według jego specyfikacji	int64
processor_brand	Marka procesora w laptopie	object
processor_tier	Poziom lub kategoria wydajności procesora	object
num_cores	Liczba rdzeni procesora	int64
num_threads	Liczba wątków obsługiwanych przez procesor	int64
ram_memory	Ilość pamięci RAM używanej w laptopie	int64
primary_storage_type	Typ pamięci podstawowej (np. HDD, SSD)	object
primary_storage_capacity	Pojemność pamięci podstawowej w laptopie	int64
secondary_storage_type	Typ pamięci dodatkowej, jeśli dostępny	object
secondary_storage_capacity	Pojemność pamięci dodatkowej w laptopie	int64
gpu_brand	Marka procesora graficznego (GPU)	object
gpu_type	Typ procesora graficznego	object
is_touch_screen	Wskazuje, czy laptop ma funkcję ekranu dotykowego	bool
display_size	Rozmiar ekranu laptopa w calach	float64
resolution_width	Rozdzielczość szerokości ekranu	int64
resolution_height	Rozdzielczość wysokości ekranu	int64
OS	System operacyjny zainstalowany na laptopie	object
year_of_warranty	Okres gwarancji, którym objęty jest laptop	object

2 Eksploracyjna analiza danych (EDA)

2.1 Typy zmiennych

Zgodnie z Tabelą 1 nasza ramka danych zawierała 11 zmiennych numerycznych (int64 i float64), 10 zmiennych kategorycznych (object) oraz 1 typu prawda/fałsz (bool). Jednak zauważyłyśmy, że chociaż zmienna year_of_warranty ma typ object, teoretycznie powinna być typu numerycznego. Zajmiemy się każdą zmienną dokładniej podczas transformacji zmiennych.

2.2 Braki danych i duplikaty

Ramka danych nie zawierała ani duplikatów, ani braków danych. Jednak w kolumnie year_of_warranty istniała wartość 'No information', którą później potraktowałyśmy jako 0 lat gwarancji, bo year_of_warranty przyjmowała wartości '1', '2', '3' i nie było w niej żadnego '0'.

2.3 Wartości odstające (Outliers)

Dla każdej zmiennej w naszej ramce danych zrobiłyśmy boxploty oraz histploty, żeby zobaczyć ich rozkłady i przeanalizować istnienie wartości odstających. Według boxplotów, czyli metodami statystycznymi wychodziło, że w ramce danych jest dość duża ilość outlierów (przykład Figure 1). Jednak ręcznie w internecie sprawdziłyśmy niektóre outliery (na przykład w zmiennej Price) i okazało się, że te skrajne wartości są prawdziwe. Zdecydowałyśmy więc, że nie będziemy ani usuwać, ani zamieniać outlierów, bo nie wynikają one z błędnych zapisów, tylko faktycznie są zgodne z rzeczywistością. Zmieniając lub usuwając te outliery, straciłybyśmy bardzo dużą część prawidłowej informacji.

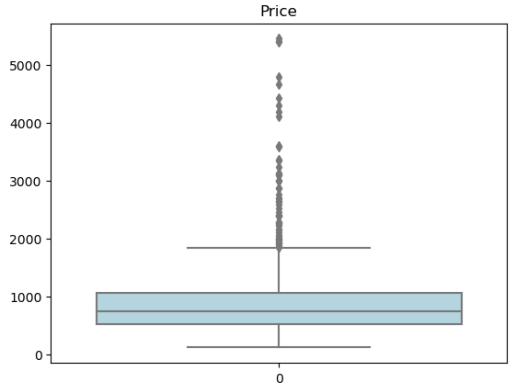

Figure 1: Boxplot for Price

2.4 Zmienne skorelowane

Zrobiłyśmy macierz korelacji między zmiennymi w postaci heatmapy i zauważyłyśmy, że możemy pozbyć się kolumny 'secondary_storage_type' ['No secondary storage' lub 'SSD'], ponieważ ta informacja jest już zawarta w kolumnie 'secondary_storage_capacity'.

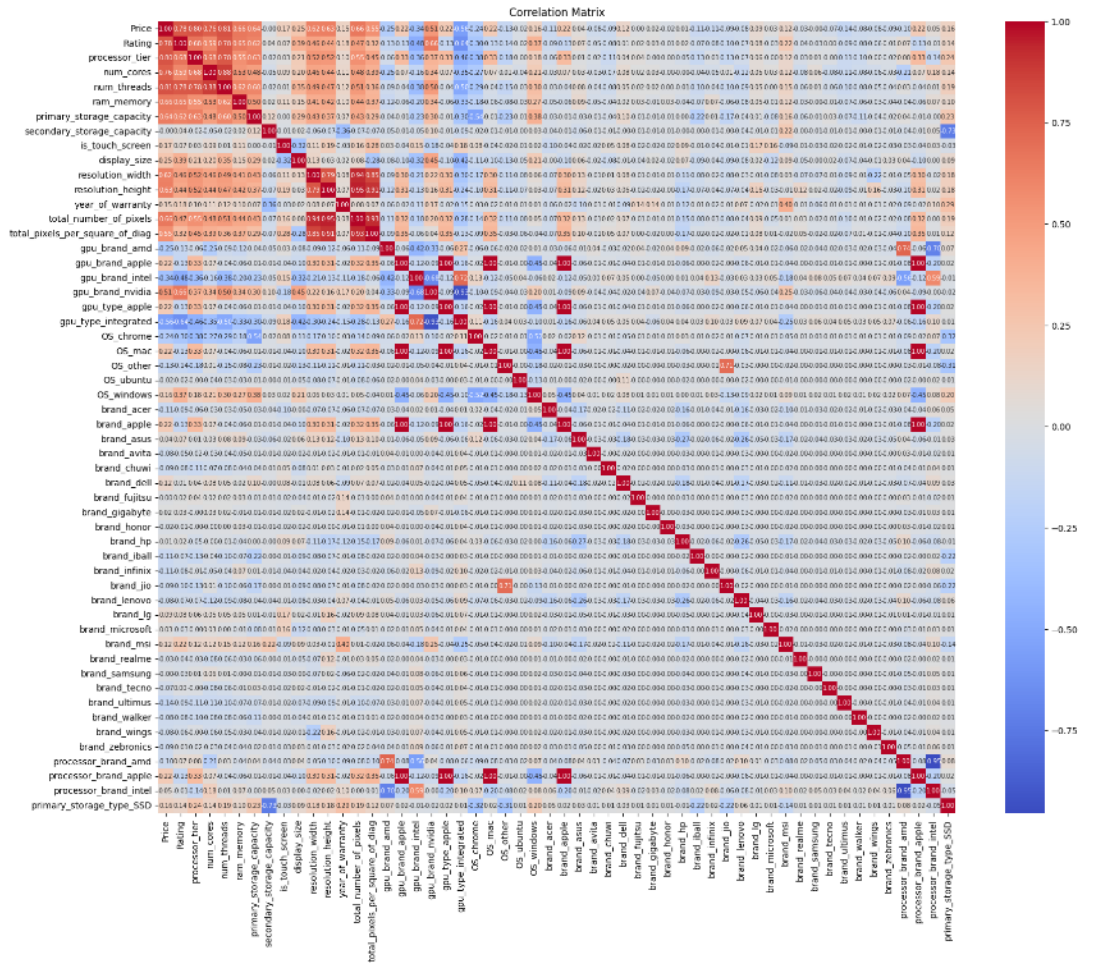

Figure 2: Correlation matrix of features (after transformations)

2.5 Inne wykresy dla lepszego zrozumienia ramki danych

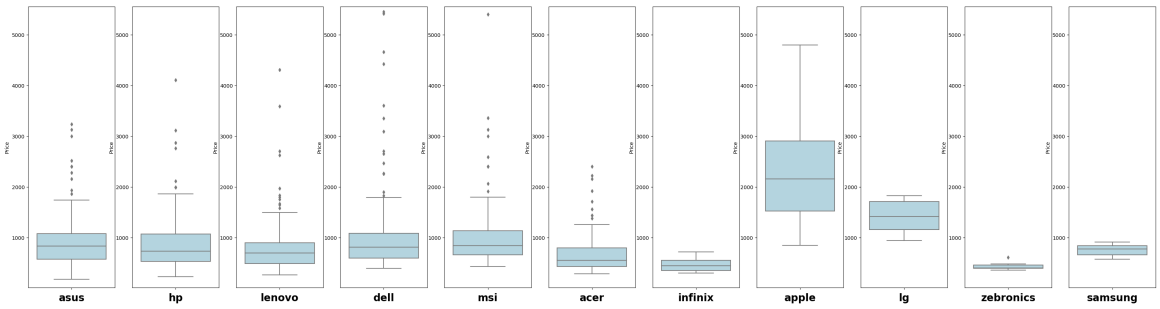

Figure 3: Boxplots of Price for the most popular Brands

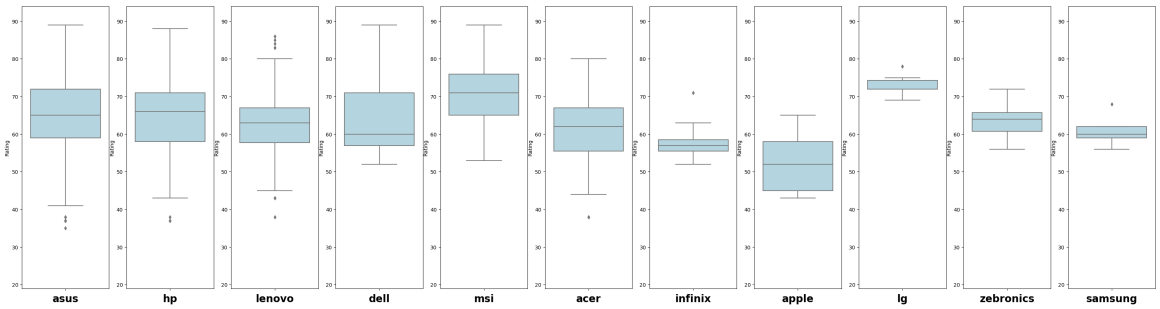

Figure 4: Boxplots of Rating for the most popular Brands

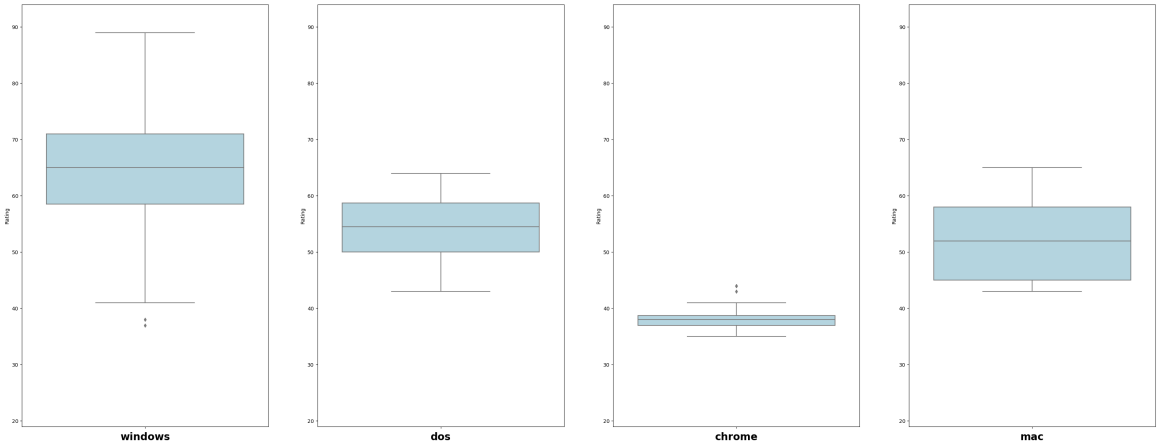

Figure 5: Boxplots of Rating for the most popular Operating systems

3 Transformacje zmiennych

W zależności od typu zmiennej wykonałyśmy następujące transformacje:

1. Jeżeli zmienna była numeryczna, to za pomocą różnych funkcji robiłyśmy transformację, żeby maksymalnie upodobnić ją do rozkładu normalnego (w celu sprawdzenia, czy to się udało, używałyśmy testów Shapiro–Wilka oraz Q-Q (quantile-quantile)), ponieważ duża część algrytmów klastrowania zakłada, że zmienne pochodzą z rozkładów normalnych. Dodatkowo za pomocą MinMaxScaler (bo proporcjalnie zachowuje odległości) wszystkie zmienne numeryczne zostały przeskalowane do przedziału [0, 1]. (Próbowałyśmy również używać StandardScaler, ale na naszych danych działał on znacznie gorzej).

2. Jeżeli zmienna była kategoryczna, to stosowałyśmy zarówno OneHotEncoding (gdy nie można było ustalić żadnego porządku), jak i OrdinalEncoding (gdy można było ustalić porządek).

Tabela 2 zawiera informacje o transformacjach związanych z każdą kolumną.

Table 2: Transformacje kolumn

Kolumna	Transformacja
index	Drop (nie ma żadnej nowej informacji)
brand	OneHotEncoding
model	Drop (nie ma żadnej nowej informacji)
price	np.log() + MinMaxScaler()
rating	np.power(x, 2) + MinMaxScaler()
processor_brand	OneHotEncoding
processor_tier	OrdinalEncoding według mocy procesora (od 0 do 7)
num_cores	np.log() + MinMaxScaler()
num_threads	np.power(x, 0.75) + MinMaxScaler()
ram_memory	MinMaxScaler()
primary_storage_type	OneHotEncoding
primary_storage_capacity	Przekształcenie: 0 -> 0, 32 -> 1, 64 -> 2, 128 -> 3, 256 -> 4, 512 -> 5, 1024 -> 6, 2048 -> 7, + MinMaxScaler()
secondary_storage_type	Drop (z EDA)
secondary_storage_capacity	Przekształcenie: 0 -> 0, 32 -> 1, 64 -> 2, 128 -> 3, 256 -> 4, 512 -> 5, 1024 -> 6, 2048 -> 7, + MinMaxScaler()
gpu_brand	OneHotEncoding
gpu_type	OneHotEncoding
is_touch_screen	Konwersja bool na int 0, 1
display_size	np.power(x, 5) + MinMaxScaler()
resolution_width	np.log() + MinMaxScaler()
resolution_height	np.log() + MinMaxScaler()
OS	OneHotEncoding
year_of_warranty	OrdinalEncoding
total_number_of_pixels	np.log() + MinMaxScaler()
total_pixels_per_square_of_diag	MinMaxScaler()

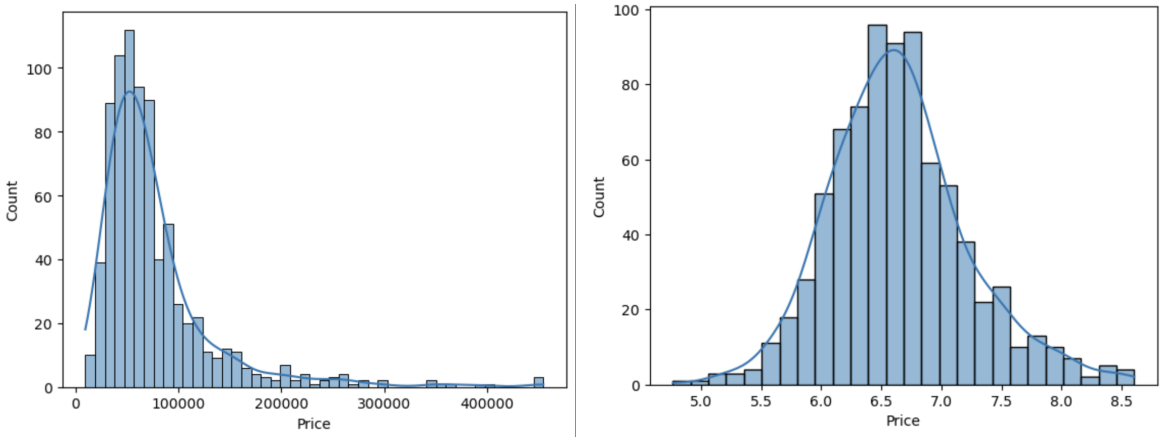

Figure 6: Price before using log scale and after

3.1 Zmienne skorelowane

Po transformacji danych znowu zrobiłyśmy heatmapę i na jej podstawie usunęłyśmy jeszcze kolumny: 'gpu_type_apple', 'OS_mac', 'processor_brand_apple', 'brand_apple' (Fig. 2). Ostatecznie nasza ramka danych do preprocessingu zawierała 49 zmiennych.

3.2 Tworzenie nowych zmiennych

Podczas transformacji danych stworzyłyśmy jeszcze 2 nowe zmienne:

```
data['total_number_of_pixels'] = data['resolution_height'] * data['resolution_width']

data['total_pixels_per_square_of_diag'] = data['total_number_of_pixels'] / (data['display_size'] ** 2)
```

Okazało się jednak, że druga z nich nie wnosi żadnych nowych informacji i jest bardzo mocno skorelowana z pierwszą, finalnie została więc usunięta.

4 Klastrowanie + analiza klastrów

4.1 Użycie UMAP do redukcji danych

4.1.1 Użycie wszystkich zmiennych w danych

Dla wszystkich naszych danych zastosowałyśmy metodę UMAP do redukcji wymiarów. Najlepsze hiperparametry były znalezione za pomocą Optuna, która minimalizowała średnie odchylenie pairwise_distances. W tym momencie wybrałyśmy tę metrykę, a nie silhouette_score, calinski_harabasz_score, davies_bouldin_score, bo nie wiedziałyśmy jeszcze, ile klastrów będzie najlepiej pasować dla zredukowanych danych oraz nie wiedziałyśmy, jakiego algorytmu klasteryzacji będziemy używać. Chciałyśmy najpierw zredukować dane, a potem dobrać dobrą ilość klastrów i użyć różnych metod.
Już po obraniu hiperparametrów i redukcji danych (do 10 wymiarów), używając metryk silhouette_score, calinski_harabasz_score, davies_bouldin_score i metody elbow (Figure 3), zdecydowałyśmy, że najlepszą liczbą klastrów (w metodach gdzie wybieramy ilość klastrów samodzielnie) będzie 3.

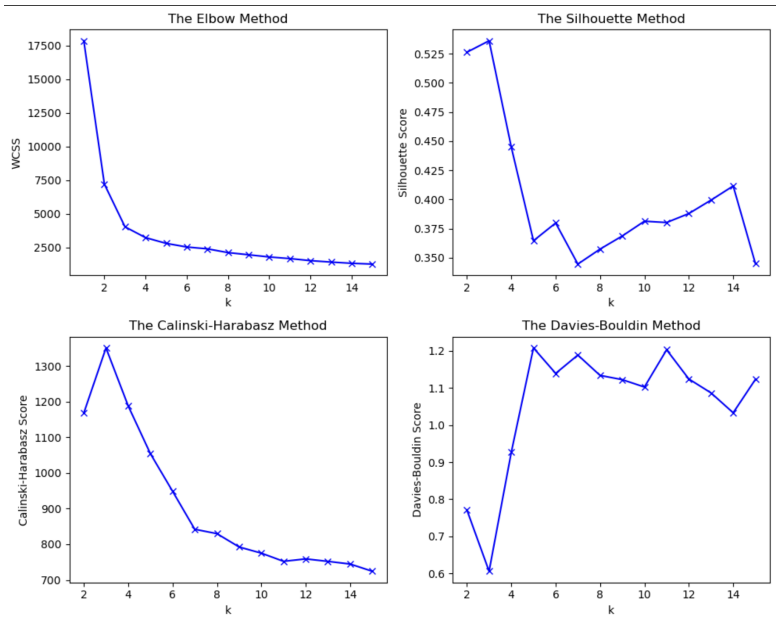

Figure 7: Metrics for UMAP dimension reduction

1. `KMeans:`

Table 3: Metryki Kmeans

Metryka	**Wartość**
Silhouette Score	0.5360302
Calinski-Harabasz Score	1350.5186805091505
Davies-Bouldin Score	0.6061432453306356

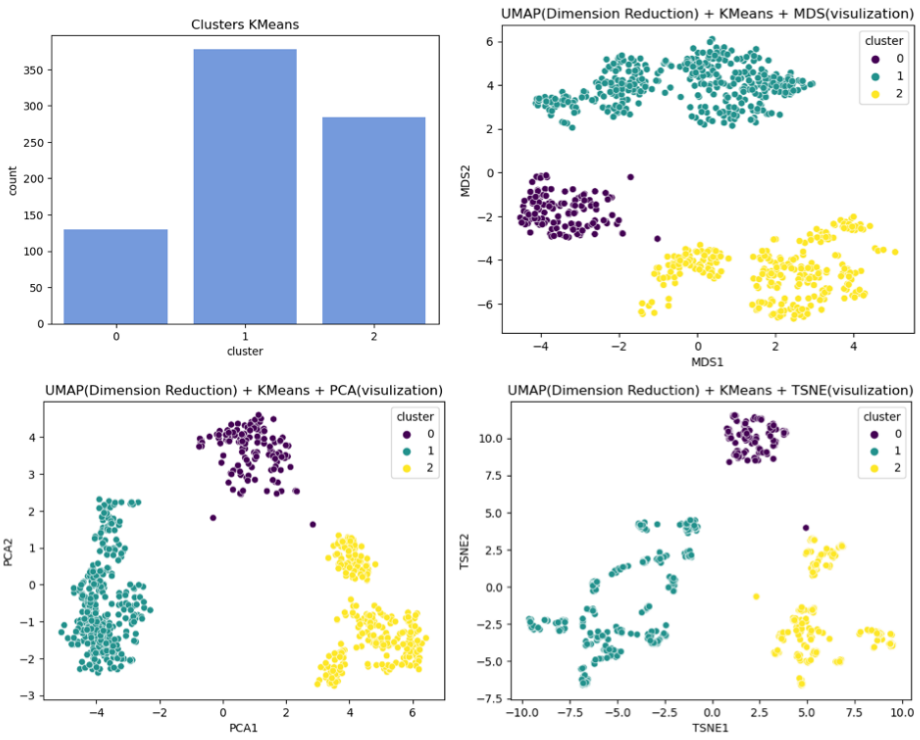

Figure 8: KMeans Clustering Visualization

2. `Agglomerative Clustering:`

Table 4: Metryki Agglomerative Clustering

Metryka	**Wartość**
Silhouette Score	0.536028
Calinski-Harabasz Score	1349.4105589710377
Davies-Bouldin Score	0.6039325241474248

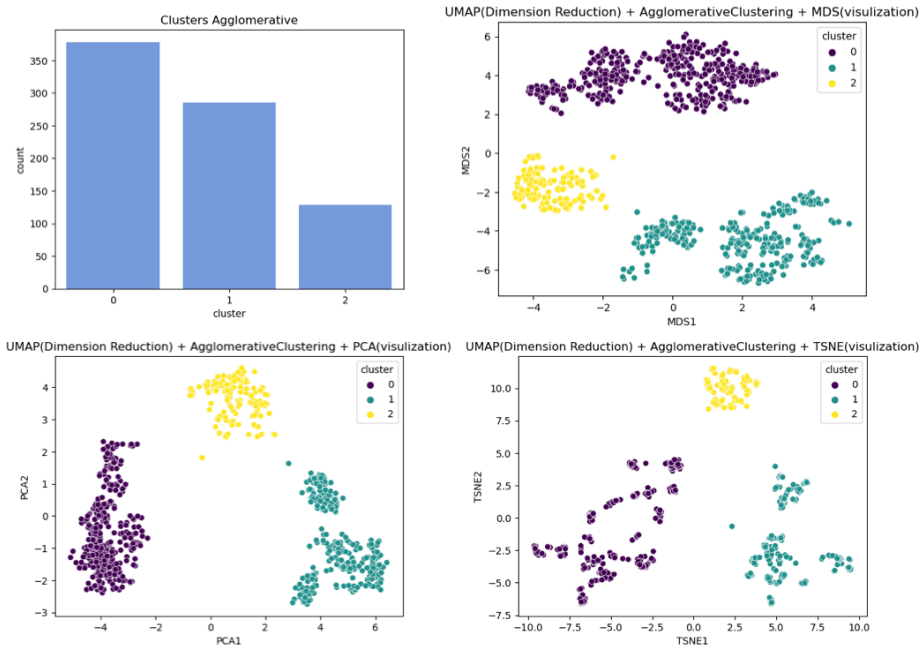

Figure 9: Agglomerative Clustering Visualization

3. `DBSCAN:` Dla wyznaczenia najlepszych hiperparametrów korzystałyśmy z metody Optuna, która maksymalizowała [score = 0.5 * silhouette + 0.5 * calinski].

Table 5: DBSCAN

Metryka	**Wartość**
Silhouette Score	0.49322468
Calinski-Harabasz Score	960.0203006726656
Davies-Bouldin Score	0.6150387611468902

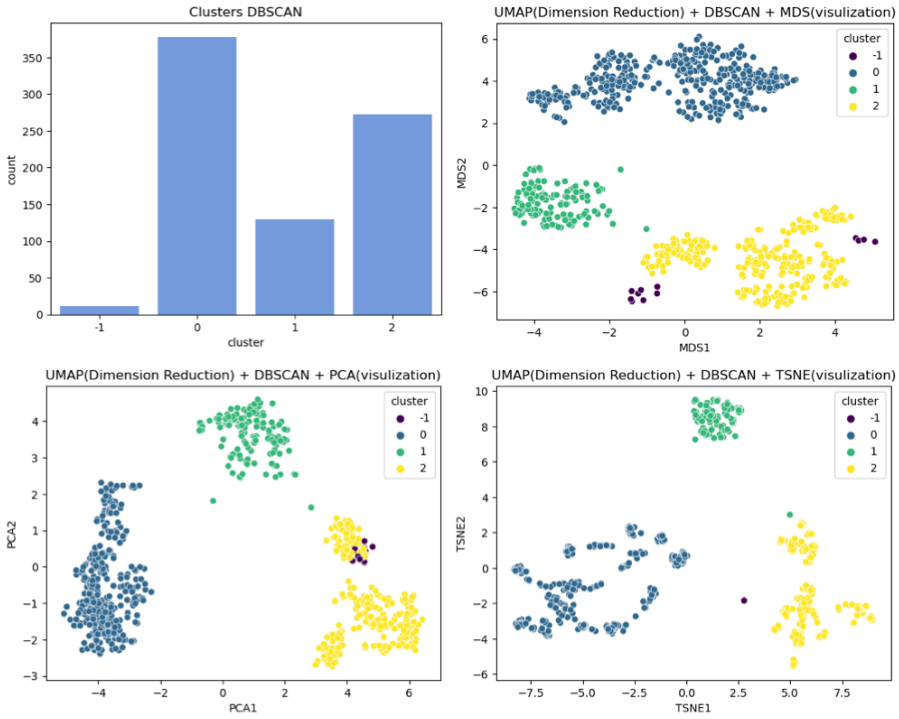

Figure 10: DBSCAN Visualization

4. `Gaussian Mixture Models:`

Table 6: Gaussian Mixture Models

Metryka	**Wartość**
Silhouette Score	0.536028
Calinski-Harabasz Score	1349.4105589710377
Davies-Bouldin Score	0.6039325241474248

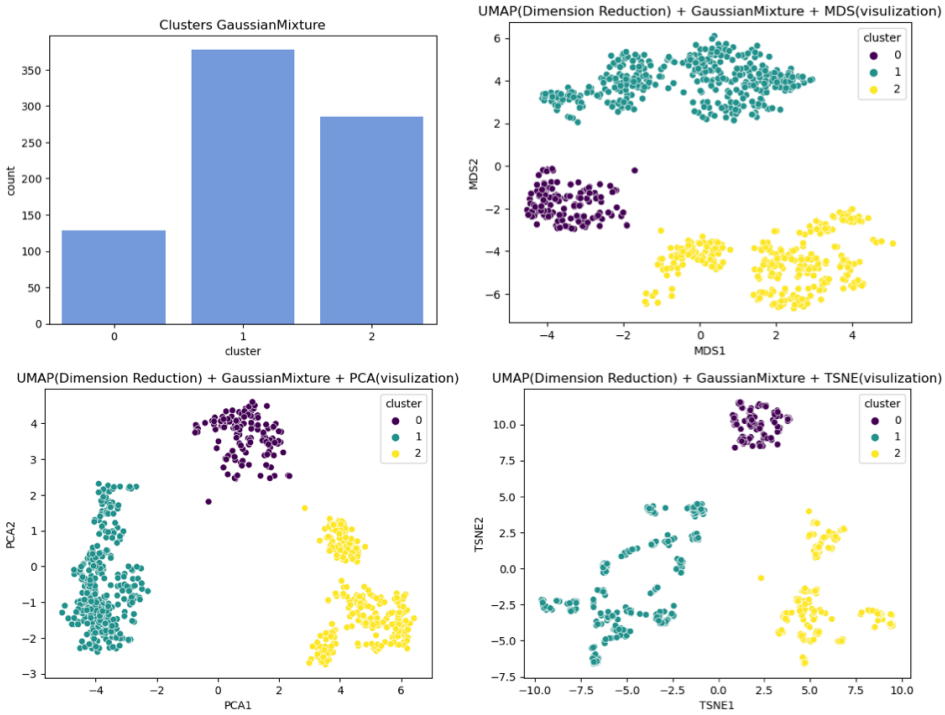

Figure 11: Gaussian Mixture Models Visualization

5. `Spectral Clustering:`

Table 7: Spectral Clustering

Metryka	**Wartość**
Silhouette Score	0.52342284
Calinski-Harabasz Score	1301.8912352028942
Davies-Bouldin Score	0.6611471859074306

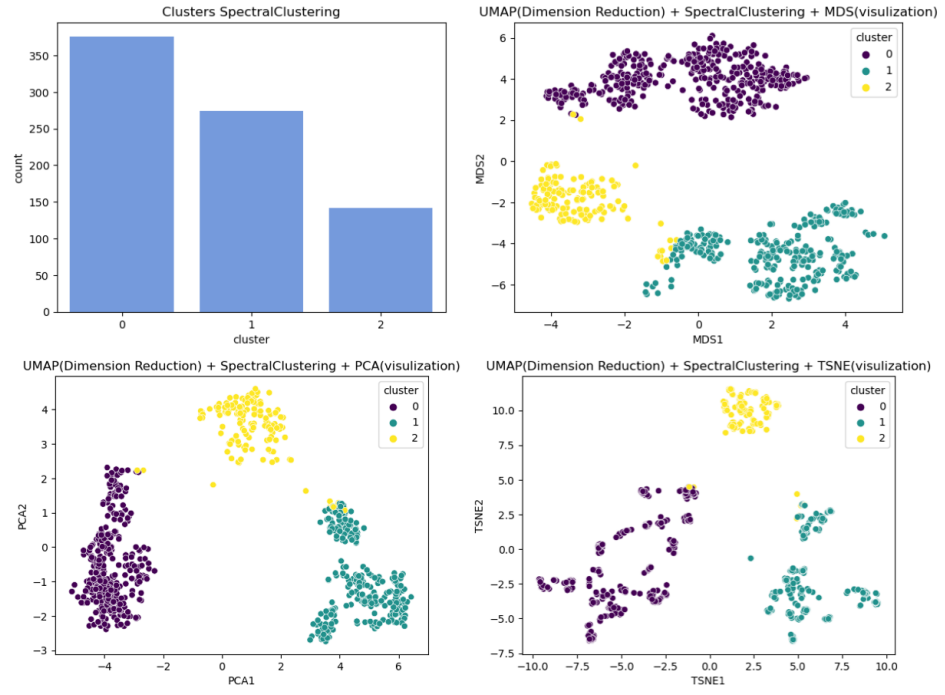

Figure 12: Spectral Clustering Visualization

6. `Analiza klastrów:` Wszystko na razie wyglądało dobrze, jednak zrobiłyśmy boxploty i histploty naszych początkowych danych z podziałem na klastry dla wszystkich używanych przez nas metod Clustering. Z tego zauważyłyśmy, że zmienna gpu_brand prawie na 100% decyduje, do jakiego klastra należy rekord (laptop) - Figure 9. Spróbowałyśmy zatem całkowicie usunąć kolumnę gpu_brand, ale wtedy miałyśmy taki sam problem tylko z kolumną processor_brand.

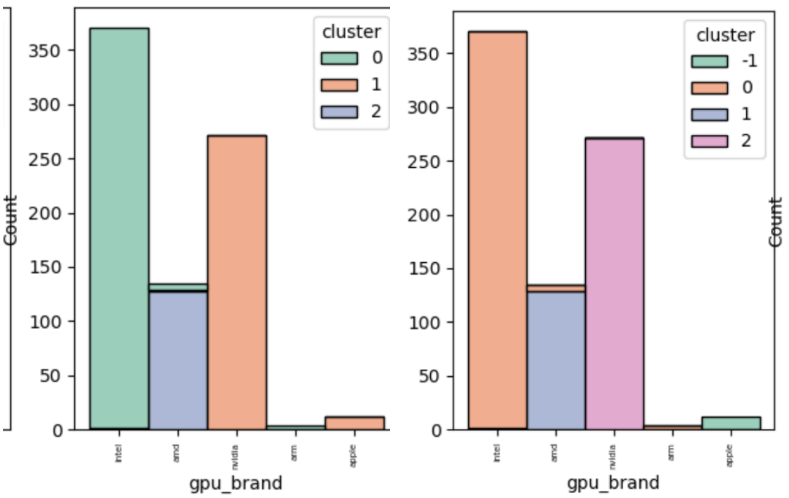

Figure 13: DBSCAN (Right), AgglomerativeClustering (Left) - gpu_brand w dużym stopniu decyduje, do jakiego klastru należy laptop.

4.1.2 Użycie wszystkich zmiennych oprócz gpu_brand, processor_brand

Przeprowadziłyśmy taką samą analizę jak w 4.1.1 tylko dla nowych danych. Według metryk najlepszą liczbą klastrów było zdecydowanie 11, a następnie 2 klastry.

1. Dla 11 klastrów: niestety wszystkie algorytmy klasteryzacji zadziałały jeszcze gorzej. Silhouette Score i Calinski-Harabasz Score znacznie spadły prawie dla wszystkich metod, a Davies-Bouldin Score wzrósł.

2. DBSCAN w ogóle jako najlepsze rozwiązanie zaproponował 45 klastrów (hiperparametry zostały wybrane za pomocą metody Optuna, która maksymalizowała [score = 0.5 * silhouette + 0.5 * calinski]).

3. Dla 2 klastrów: niestety wszystkie algorytmy klasteryzacji zadziałały jeszcze gorzej niż dla 11 klastrów. Silhouette Score i Calinski-Harabasz Score znacznie spadły prawie dla wszystkich metod, a Davies-Bouldin Score wzrósł.

4.2 Użycie SVD (Singular Value Decomposition) do redukcji danych oraz KMeans do klasteryzacji

Podobnie jak w 4.1 na początku za pomocą metody SVD (Singular Value Decomposition) zmniejszyłyśmy wymiar naszych danych do 11 wymiarów w celu zachowania 90% całej informacji znajdującej się w początkowej ramce danych. Wykres zależności procentowej skumulowanej wartości wyjaśnionej wariancji od liczby komponentów SVD przedstawiony jest poniżej.

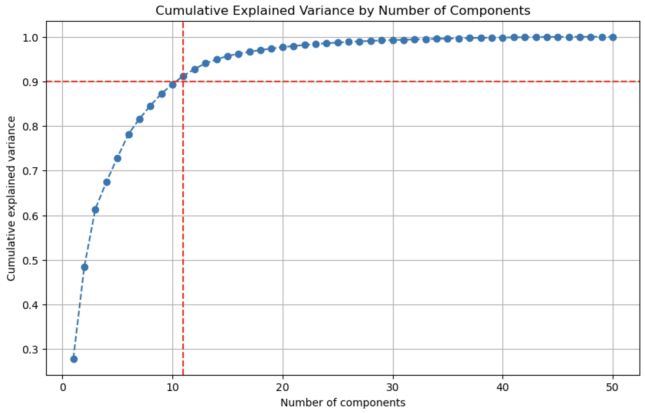

Figure 14: Explained Variance by Component (SVD)

Następnie dla metody KMeans narysowałyśmy wykresy metryk opisujących podział na klastry w zależności od wybranej liczby klastrów. Wykresy są przedstawione poniżej. (Warto dodać, że również spróbowałyśmy użyć metod Spectral Clustering, Agglomerative Clustering i Gaussian Mizture Models, jednak ze wszystkimi tymi metodami miałyśmy taki sam problem, jak w UMAP, Fig.10).

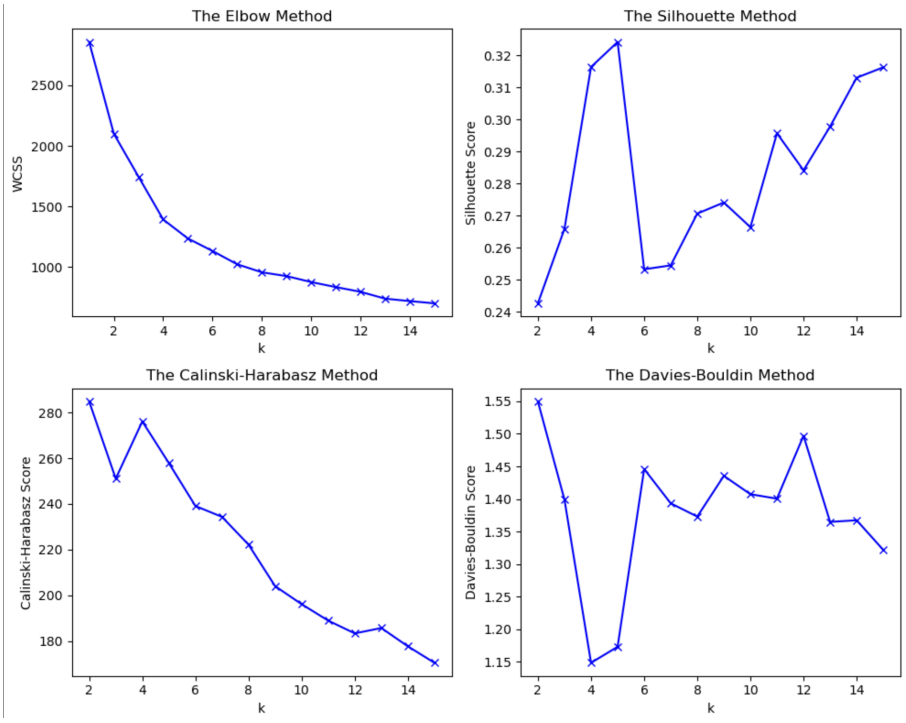

Figure 15: Metrics for SVD+Kmeans

Na podstawie powyższych wykresów jako najlepszą liczbę klastrów wybrałyśmy $k = 5$.

Table 8: SVD+KMeans (k=5) metrics

Metryka	**Wartość**
Silhouette Score	0.3242104496864801
Calinski-Harabasz Score	257.84248448945567
Davies-Bouldin Score	1.1731547842773122

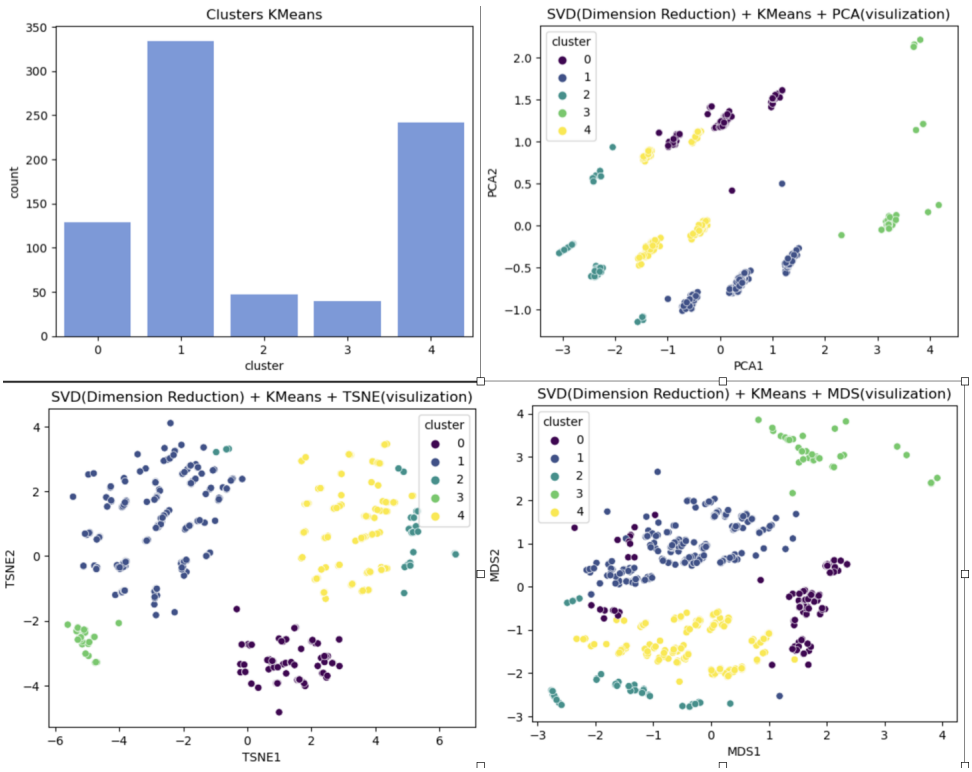

Figure 16: SVD+KMeans(k=5) Visualization

Znowu zrobiłyśmy barploty oraz boxploty z naszych początkowych danych z podziałem na klastry, żeby zobaczyć, czym się różnią klastry, laptopy z jakimi cechami gdzie trafiły i czy nie występuje problem podobny jak na Fig.10 (rzeczywiście nie występuje):

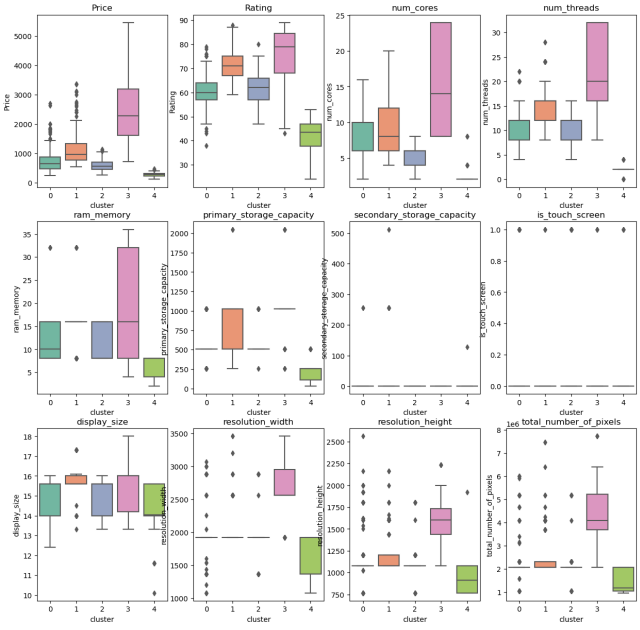

Figure 17: Boxplots of features by clusters

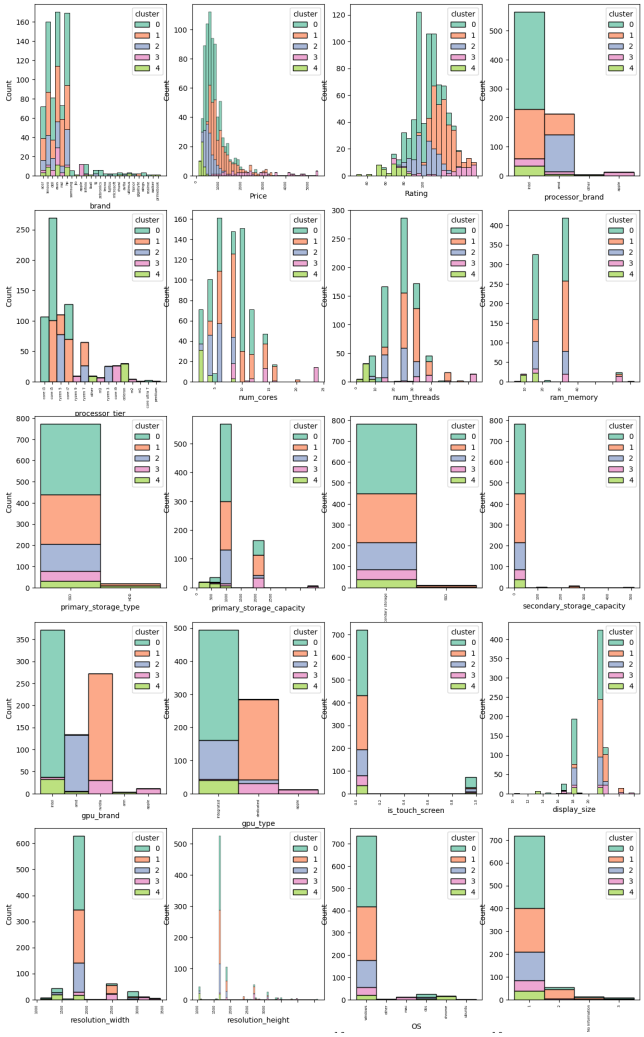

Figure 18: Barplots of features by clusters

Na podstawie powyższych wykresów możemy wyciągnąć następujące wnioski:

- Klaster 0 - pewien kompromis między ceną a jakością dla osób, które chcą procesor Intel i więcej rdzeni niż laptopy w klastrze 2.
- Klaster 1 - laptopy z dużą przekątną ekranu (16 cali) z dedykowanym GPU, o wysokiej ocenie.
- Klaster 2 - pewien kompromis między ceną a jakością dla osób, które chcą procesor AMD.
- Klaster 3 - laptopy z najwyższą ceną i najwyższą oceną oraz świetnym GPU (dedykowany GPU lub Apple GPU), z doskonałą rozdzielczością.
- Klaster 4 - najtańsze laptopy z ekstremalnie niską ceną i najgorszymi cechami.

Chociaż w danej klasteryzacji wartości metryk się pogorszyły w porównaniu z modelami w 4.1.1, ten model ma znacznie większy sens niż modele w 4.1.1.
Warto też jako wadę modelu SVD+KMeans zaznaczyć brak balansu w liczbie rekordów w każdym klastrze.

4.3 Użycie PCA do redukcji danych oraz KMeans do klasteryzacji

Na początku zastosowałyśmy metodę PCA do redukcji wymiarów. Aby wyznaczyć liczbę komponentów do PCA, narysowałyśmy wykres zależności procentowej skumulowanej wartości wyjaśnionej wariancji od liczby komponentów.

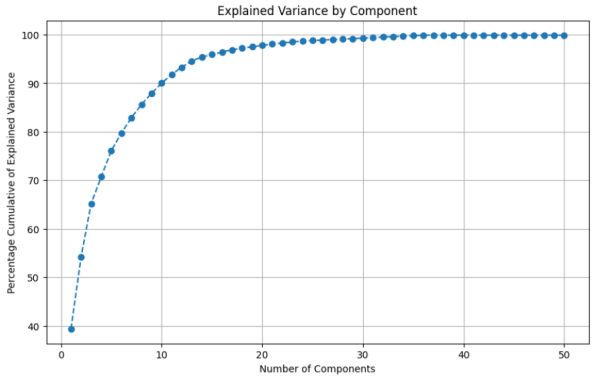

Figure 19: Explained Variance by Component (PCA)

Na podstawie wykresu, chcąc zachować przynajmniej 80% danych, wybrałyśmy jako minimalną liczbę komponentów n = 7. Następnie sprawdziłyśmy, dla jakiej liczby komponentów i jakiej liczby klastrów wartości score dla metryk silhouette, calinski-harabasz i davies-bouldin są najlepsze. Wyszło nam, że najlepsza liczba komponentów to 8 i najlepsza liczba klastrów to 4. Poniżej wykresy dla poszczególnych metryk.

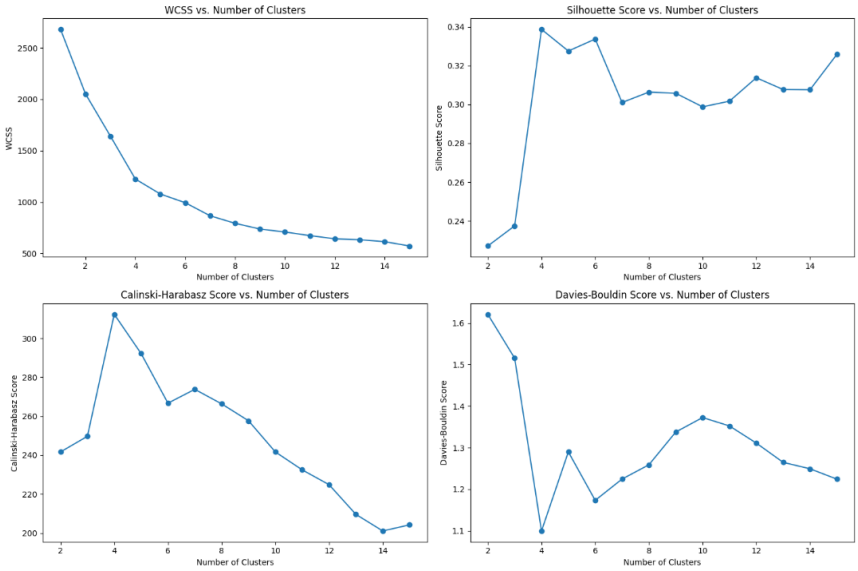

Figure 20: Metrics for PCA+KMeans dimension reduction

Po klasteryzacji liczba obserwacji należących do danego klastra rozkładała się następująco:

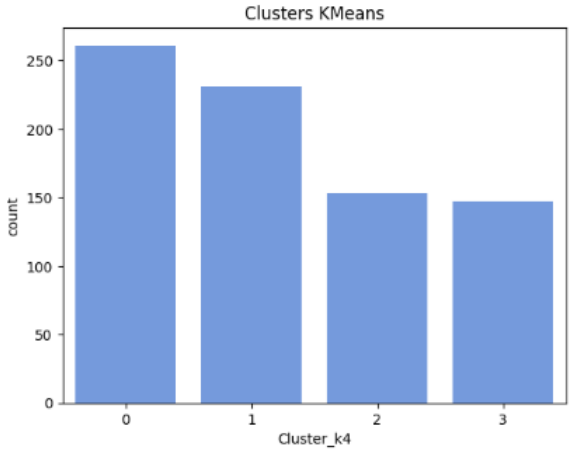

Figure 21: Clusters barplot

Table 9: PCA+KMeans (k=4) metrics

Metryka	Wartość
Silhouette Score	0.3362499888715211
Calinski-Harabasz Score	372.5242116666582
Davies-Bouldin Score	1.2051595975573441

Wyniki klasteryzacji przedstawione zostały na wykresach, gdzie dane zredukowane zostały do 2 wymiarów przy użyciu PCA, tSNE oraz MDS.

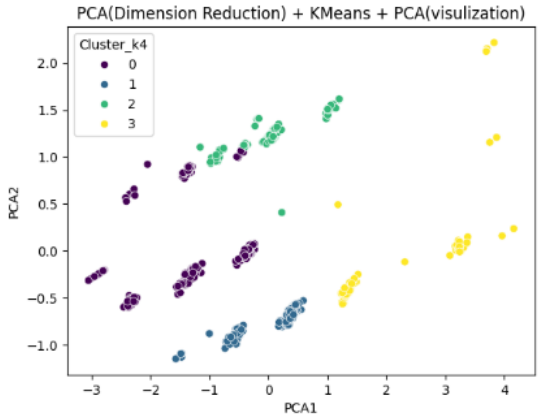

Figure 22: Clustering visualization using PCA

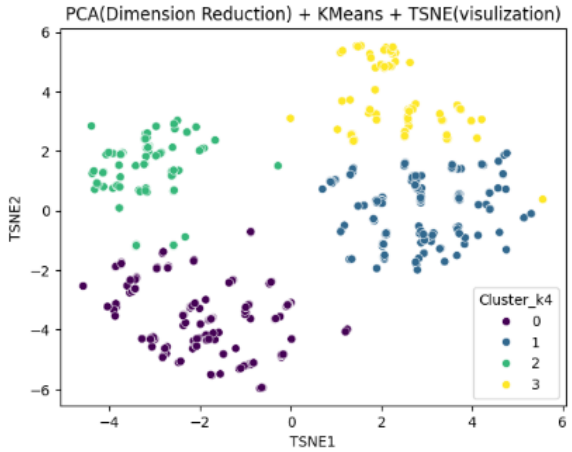

Figure 23: Clustering visualization using tSNE

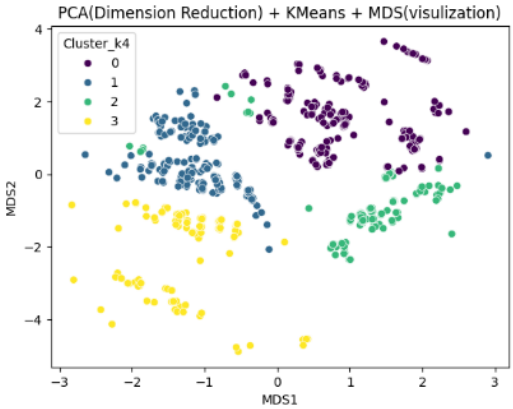

Figure 24: Clustering visualization using MDS

W celu zidentyfikowania, co znajduje się w poszczególnych klastrach, narysowałyśmy barploty oraz boxploty z podziałem na klastry.

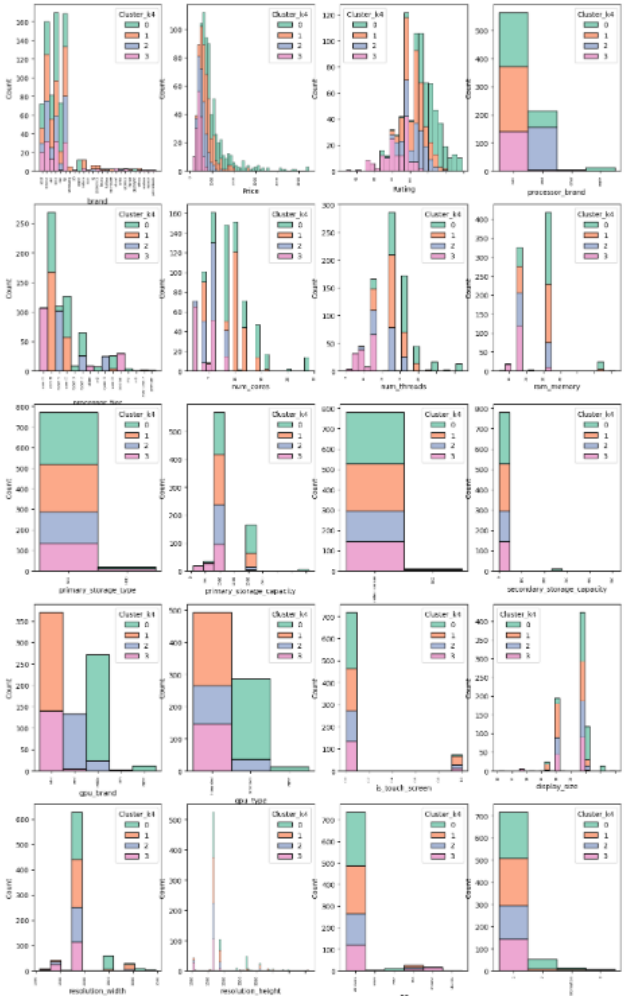

Figure 25: Barplots of features by clusters

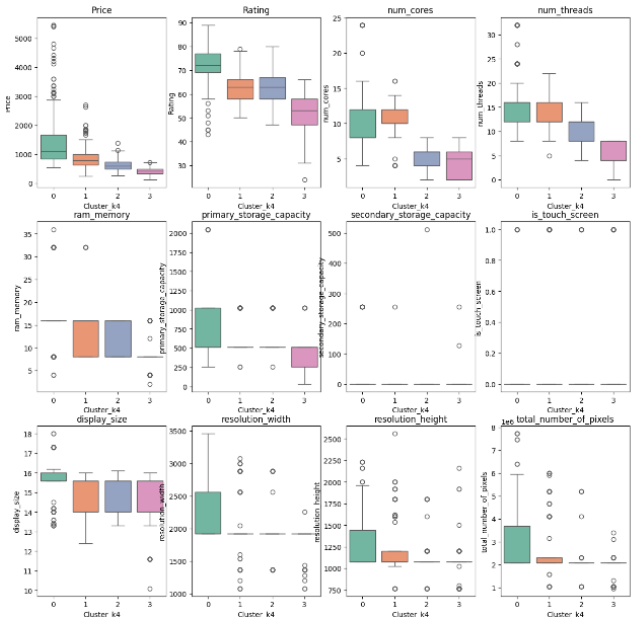

Figure 26: Boxplots of features by clusters

Na podstawie powyższych wykresów wyciągnęłyśmy następujące wnioski:

- **Klaster 0:** Laptopy od najlepszych marek, takich jak Apple, Lenovo, Asus. Laptopy z wysoką ceną i oceną. Wyposażone w karty graficzne NVIDIA oraz najwyższą rozdzielczość wyświetlacza.
- **Klaster 1:** Nieco tańsze i niżej oceniane laptopy. Przeważnie z procesorami Intel. Większość laptopów z ekranem dotykowym znajduje się w tym klastrze.
- **Klaster 2:** Wiele laptopów marki HP. Procesor i GPU to AMD. Cena poniżej 1000 USD.
- **Klaster 3:** Najtańsze i najniżej oceniane laptopy. Procesory głównie Intel. Wszystkie laptopy w tym klastrze posiadają zintegrowaną kartę graficzną.

5 Analiza biznesowa

Do analizy biznesowej wybrałyśmy klastrowanie z użyciem PCA do redukcji wymiarów, a następnie metody KMeans. Można z niego wyciągnąć następujące wnioski przydatne dla biznesu:

1. **Segmentacja rynku w odniesieniu do klastrów:**
   - **Klaster 0:** Laptopy od renomowanych marek, z wysoką ceną i oceną, wyposażone w karty graficzne NVIDIA i najwyższą rozdzielczość wyświetlacza. Ten segment przyciąga klientów premium, takich jak profesjonaliści i entuzjaści technologii. Strategia marketingowa powinna podkreślać najwyższą jakość, wydajność i markę.
   - **Klaster 1:** Laptopy w średnim przedziale cenowym, głównie z procesorami Intel i ekranami dotykowymi. Idealne dla użytkowników, którzy cenią funkcjonalność dotykowego ekranu i dobrą relację jakości do ceny. Marketing powinien koncentrować się na korzyściach płynących z ekranu dotykowego.
   - **Klaster 2:** Ekonomiczne laptopy z procesorami i kartami graficznymi AMD, poniżej 1000 USD. Atrakcyjne dla studentów, małych firm i klientów budżetowych. Warto podkreślać wartość i wydajność przy niskiej cenie.
   - **Klaster 3:** Najtańsze i najniżej oceniane laptopy, głównie z procesorami Intel. Przyciągają klientów szukających najtańszych rozwiązań. Strategia powinna skupić się na niskiej cenie i podstawowej funkcjonalności.
2. **Strategie marketingowe** Wyniki klastrowania umożliwiają precyzyjne targetowanie kampanii marketingowych, co może zwiększyć ich efektywność. Reklamy mogą być dostosowane do specyficznych potrzeb i preferencji każdego segmentu rynku.
3. **Dopasowanie oferty cenowej** Analiza cen w poszczególnych klastrach może pomóc firmom w ustaleniu konkurencyjnych cen i strategii rabatowych.
4. **Planowanie asortymentu w sklepach** Sklepy mogą lepiej planować asortyment, uwzględniając preferencje klientów w różnych segmentach. Może to pomóc w optymalizacji stanów magazynowych i lepszym zarządzaniu zapasami.

6 Podsumowanie

Jako dwa najlepsze modele wybieramy: PCA+KMeans(k=4) - top 1, SVD+KMeans(k=5) - top 2. Chociaż modele te nie mają najlepszych wartości metryk opisujących klastrowanie, wybrałyśmy je, ponieważ mają sens i nie robią podziału na podstawie jednej zmiennej.

References

1. Brand Laptops Dataset https://www.kaggle.com/datasets/bhavikjikadara/brand-laptops-dataset/