

## **CLUSTERING**

**Title: Product Pricing Analysis of an Online Shoe Store using Clustering Techniques**

## 2.1 Introduction

Clustering is a technique for identifying patterns and structures in unlabelled examples or an undefined outcome. It is a subset of unsupervised learning. To put it simply, the process of clustering aims to separate and assign groups with similar characteristics or traits to clusters. Identifying clusters of related objects in datasets with two or more variable quantities is done using clustering. In reality, among many other sources, this data may be gathered from marketing, biomedical, or geographic databases. In today's competitive e-commerce landscape, understanding customer preferences and optimizing pricing strategies are pivotal for online retailers. This study focuses on analyzing product pricing within an online shoe store using advanced clustering techniques. Online shoe stores encounter challenges in determining optimal pricing strategies amidst diverse customer preferences and market dynamics. Pricing directly impacts customer acquisition, retention, and overall sales revenue. Traditional pricing analysis methods often fall short in capturing nuanced customer behavior. Hence, leveraging clustering techniques presents an innovative approach to unravel complex pricing structures and their relation to customer engagement, data-driven pricing strategies are key. Both machine learning and business contexts drive the analysis for impactful insights.

Pricing plays a pivotal role in retail, directly affecting revenues and profitability (Rao, 2021). However, the highly competitive shoe retail landscape, data-driven customer segmentation is key to targeted product recommendations, pricing and promotions (Chen et al. 2021). Specifically for online stores, past works have utilized K-Means to reveal customer groups by attributes like brand preference, price sensitivity and channel usage (Wang et al. 2020). However, hierarchical clustering better handles evolving groups like new college grads transitioning in needs (Ma et al. 2019).

For footwear e-commerce, K-Means clustering of purchase histories pointed to customization opportunities but overlooked granular style personas (Nagarajan et al. 2018). Alternatively, demographic-based hierarchical clusters guided catalog tailoring though lacking purchase behavior inputs (Hu et al. 2017). Our study integrates both approaches with cluster-based ensemble forecasting to enable customer tiering and shoe portfolio decisions for the online shoe retailer.

Overall, this differentiated methodology harnessing granular style-based purchase behaviors along with demographics aims to provide retail managers an interpretable navigation of current and emerging consumer segments for long-term assortment planning

## 2.2 Aims and Objectives:

The goal is to determine the number of clusters or segments that work best, as well as which group most customers belong to, in addition to identifying the most efficient algorithms for clustering e-commerce data. Using clustering techniques on the online shoe store dataset for this study, I will be able to solve a number of issues, such as: **Product Segmentation, Customer behaviour analysis, Insights for pricing strategy, Data driven decision making.**

## 2.3 Exploration and Dataset Analysis

### 2.3.1 Description of the Datasets

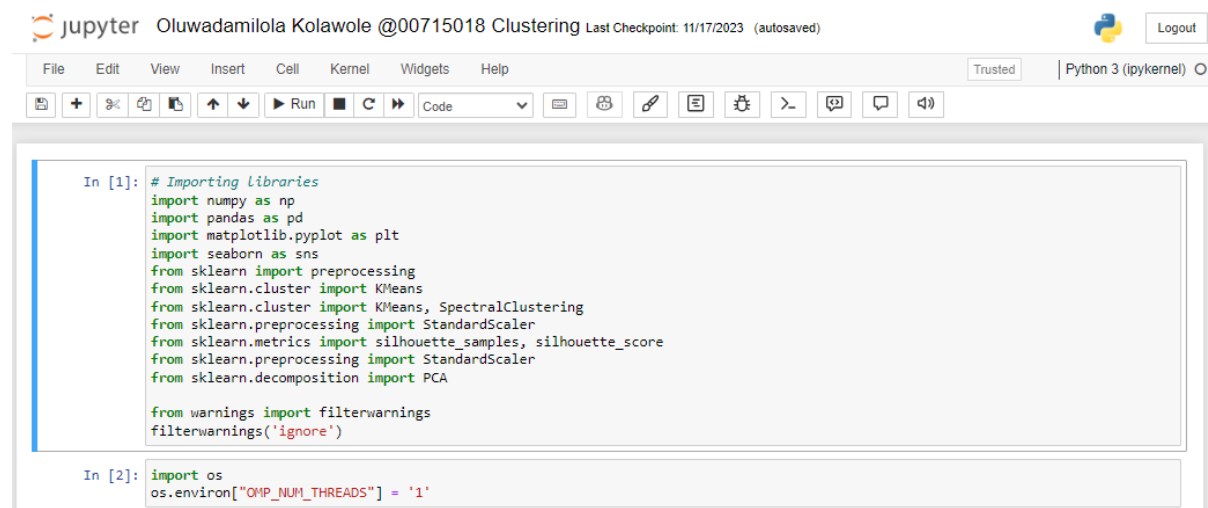
The Online shoe store (Adidas VS Nike) dataset was sourced from Google dataset, it included 3268 observations and 10 variables. The product features used for clustering mainly include listing price, sale price, brand, rating, reviews and discounts offered historically. Product name, Product id, description and last visited provides additional context.

### 2.3.2 Data Preprocessing:

Data preprocessing is an important step in the data machine learning process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis and modelling (Witten et al., 2011). The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific task.

### 2.3.3 Data integration

Importing all the necessary libraries using the code below



The image shows a Jupyter Notebook interface with the following code in two cells:

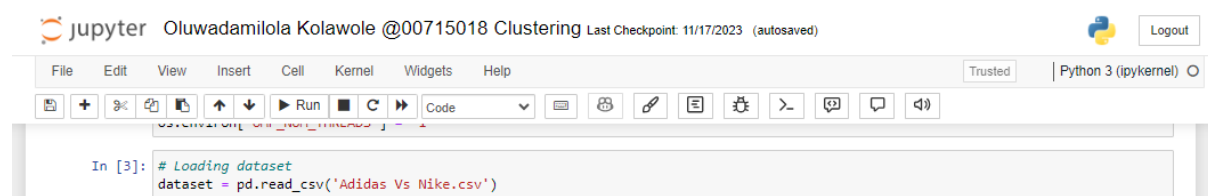
```
In [1]: # Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.cluster import KMeans
from sklearn.cluster import KMeans, SpectralClustering
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

from warnings import filterwarnings
filterwarnings('ignore')

In [2]: import os
os.environ["OMP_NUM_THREADS"] = '1'
```

**Figure 2.1:** Importing the libraries

The Online shoe store dataset was downloaded as a CSV file and read into python to get more insight on the data

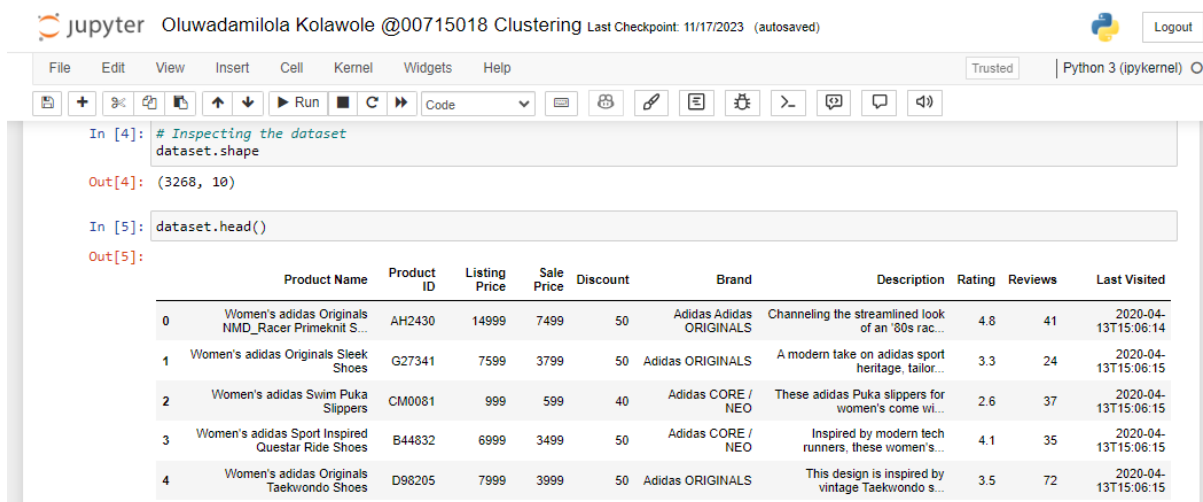


The image shows a Jupyter Notebook interface with the following code in one cell:

```
In [3]: # Loading dataset
dataset = pd.read_csv('Adidas Vs Nike.csv')
```

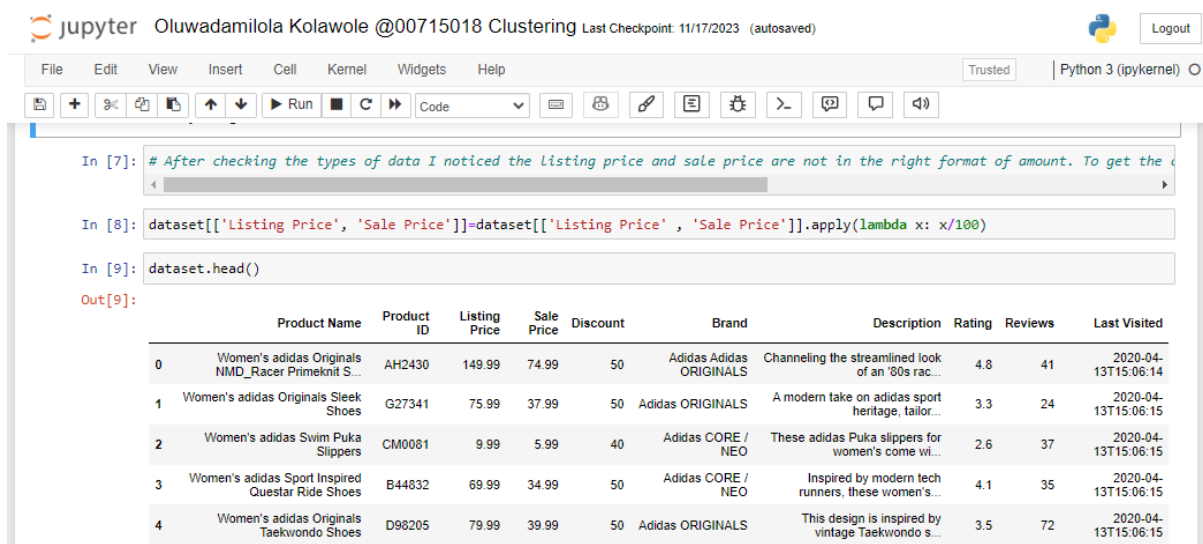
**Figure 2.2:** Loading of the data

The code in figure 2.2 was used to read and inspect the dataset showing the first 5 rows. The dataset has 3268 variables, and 10 columns as shown in figure below.

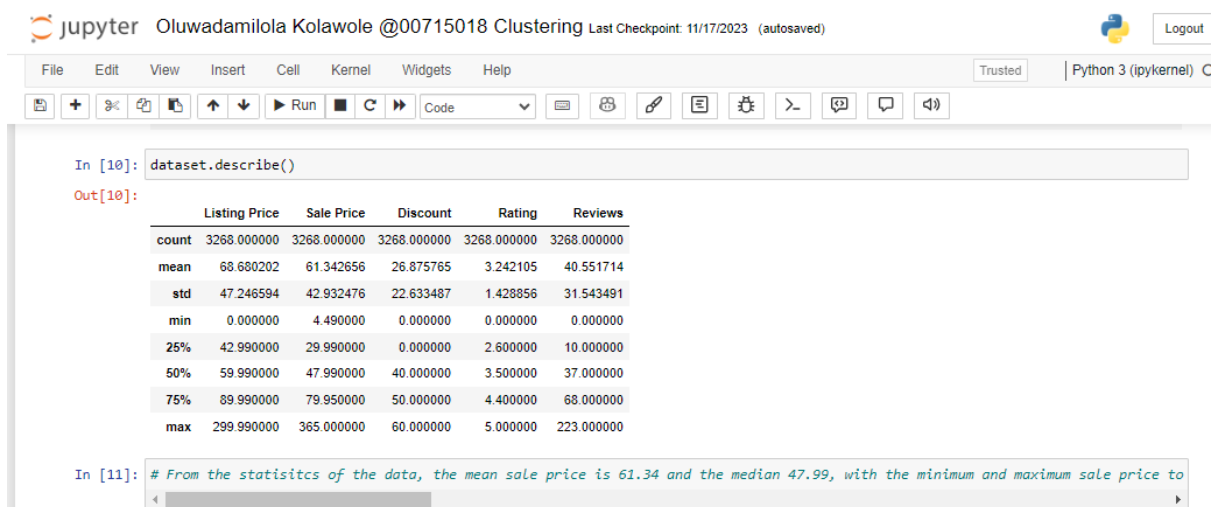


**Figure 2.3:** Inspecting and showing the first five rows of the dataset

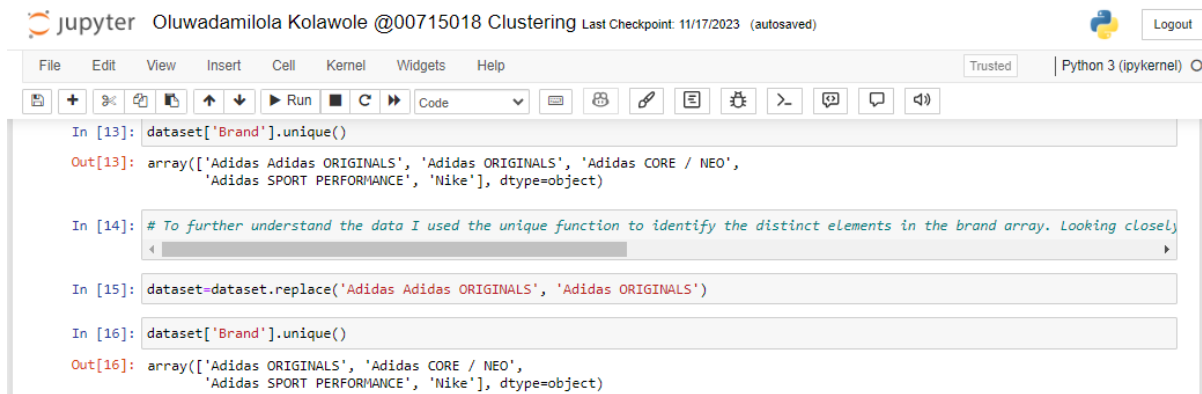
From the figure above, the Listing and Sale price isn't in the correct format i.e the currency value isn't correct.



**Figure 2.4:** Formatting the Listing and Sale price to the right amount



**Figure 2.5:** The statistical description of the data



The Jupyter Notebook interface shows the following code and output:

```
In [13]: dataset['Brand'].unique()
Out[13]: array(['Adidas Adidas ORIGINALS', 'Adidas ORIGINALS', 'Adidas CORE / NEO',
               'Adidas SPORT PERFORMANCE', 'Nike'], dtype=object)

In [14]: # To further understand the data I used the unique function to identify the distinct elements in the brand array. Looking closely
< [REDACTED] >

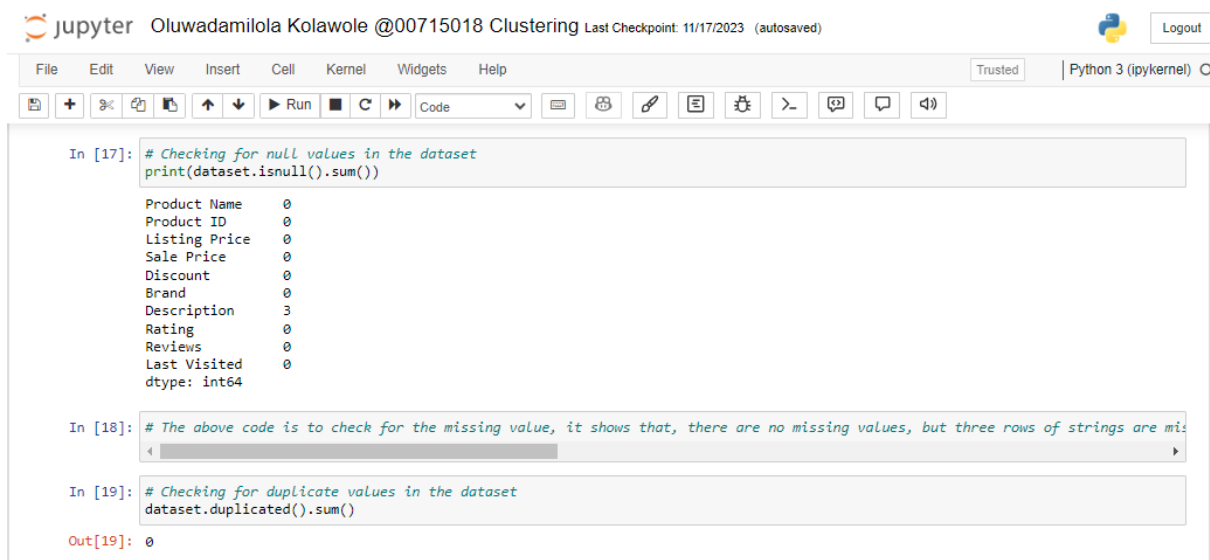
In [15]: dataset=dataset.replace('Adidas Adidas ORIGINALS', 'Adidas ORIGINALS')

In [16]: dataset['Brand'].unique()
Out[16]: array(['Adidas ORIGINALS', 'Adidas CORE / NEO',
               'Adidas SPORT PERFORMANCE', 'Nike'], dtype=object)
```

**Figure 2.6:** Formatting the brand name

From the figure above it showing that there are some mistakes on the brand name and it has been corrected.

Checking for duplicates and missing values. There is no missing values or duplicates but there are some missing values in the description column but since it is not needed in the analysis then no action is taken.



The Jupyter Notebook interface shows the following code and output:

```
In [17]: # Checking for null values in the dataset
print(dataset.isnull().sum())

Product Name      0
Product ID        0
Listing Price     0
Sale Price        0
Discount          0
Brand             0
Description       3
Rating            0
Reviews           0
Last Visited      0
dtype: int64

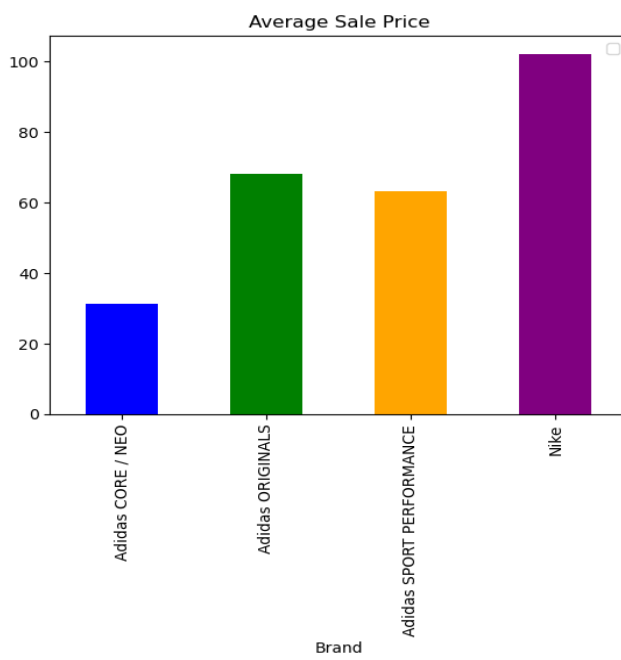
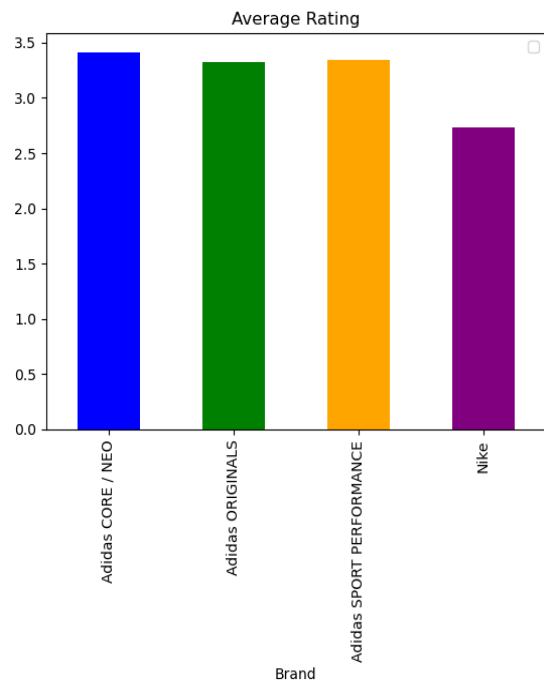
In [18]: # The above code is to check for the missing value, it shows that, there are no missing values, but three rows of strings are mis
< [REDACTED] >

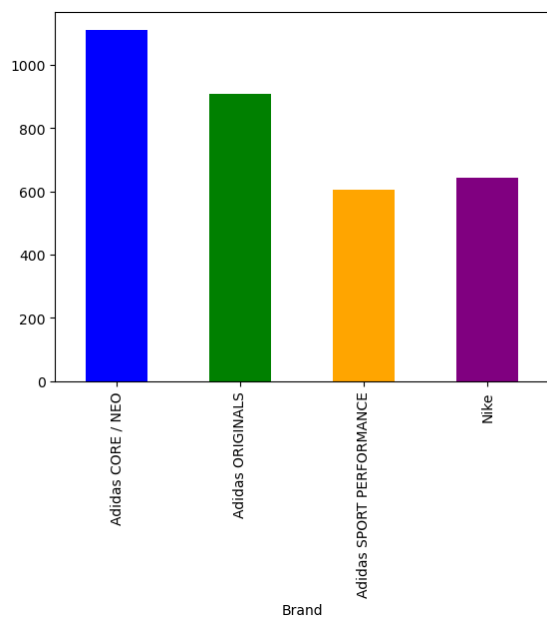
In [19]: # Checking for duplicate values in the dataset
dataset.duplicated().sum()

Out[19]: 0
```

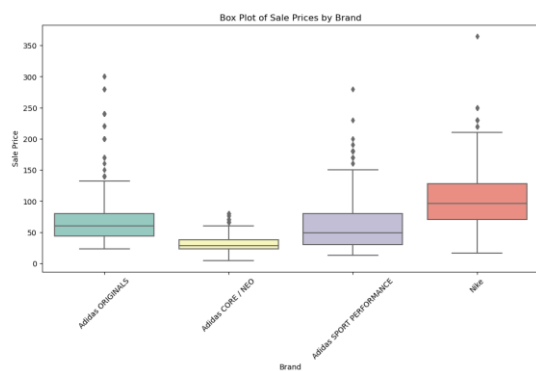
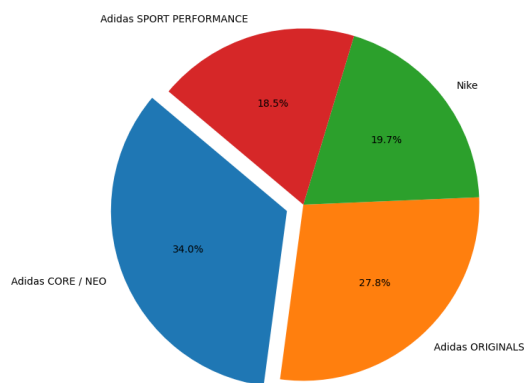
**Figure 2.7:** The duplicate and missing value

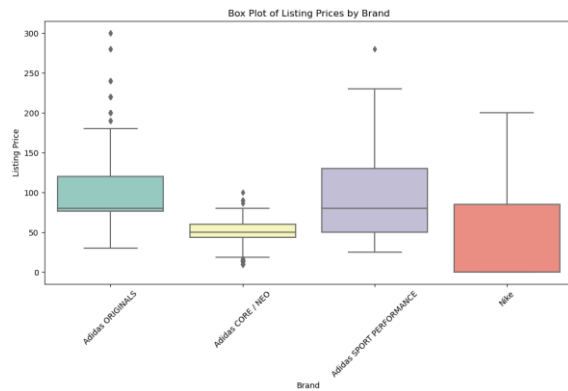
## Some of the visualizations performed





Brand Distribution





**Figure 2.8:** Visualizations

The visualizations includes average ratings by brand, average sales by brand, number of count by brand, percentage of count by brand, box plot of listing and sales prices by brand.

jupyter Oluwadamilola Kolawole @00715018 Clustering Last Checkpoint: 11/17/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```

In [37]: # Removing outliers from the Listing Price
column = 'Listing Price'

# Calculating IQR
Q1 = dataset['Listing Price'].quantile(0.25)
Q3 = dataset['Listing Price'].quantile(0.75)
IQR = Q3 - Q1

# Defining Outlier Threshold
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Removing Outliers
cleaned_dataset = dataset[(dataset['Listing Price'] >= lower_bound) & (dataset['Listing Price'] <= upper_bound)]

In [39]: # Removing outliers from the Sale Price
column = 'Sale Price'

# Calculating IQR
Q1 = dataset['Sale Price'].quantile(0.25)
Q3 = dataset['Sale Price'].quantile(0.75)
IQR = Q3 - Q1

# Defining Outlier Threshold
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

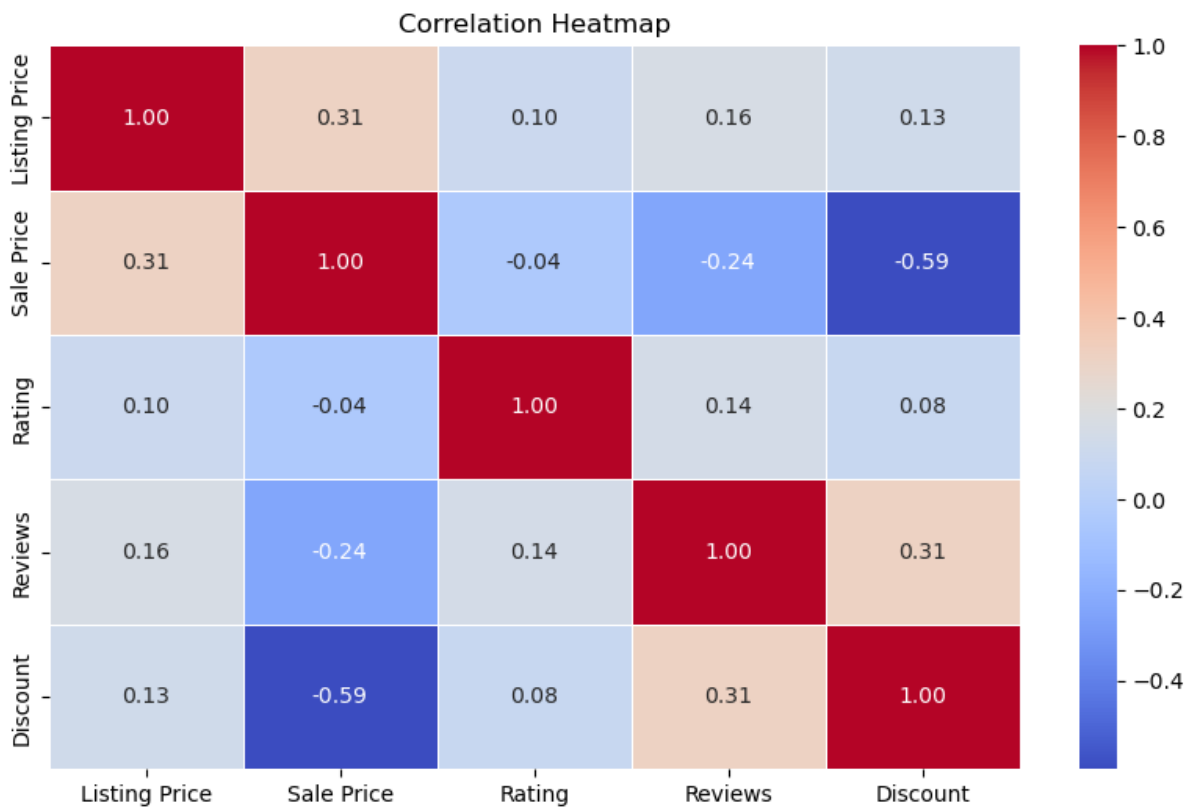
# Removing Outliers
cleaned_dataset = dataset[(dataset['Sale Price'] >= lower_bound) & (dataset['Sale Price'] <= upper_bound)]

```

**Figure 2.9:** Removing outliers

The code above was used to remove the outliers in the Listing price and Sale price.





**Figure 2.10:** Correlation heatmap

From the correlation heatmap the best correlation is the listing price and sales price and also reviews and discount. Hence, we are using the listing and sale price to cluster our data into groups.

jupyter Oluwadamilola Kolawole @00715018 Clustering Last Checkpoint: 11/17/2023 (autosaved) Python 3 (ipykernel)

```

In [45]: dataset.drop(['Product Name', 'Product ID', 'Brand', 'Description', 'Reviews', 'Last Visited'], axis=1, inplace=True)

In [46]: X = dataset.iloc[:,0:4]
print(X)

   Listing Price  Sale Price  Discount  Rating
0         149.99         74.99         50     4.8
1          75.99         37.99         50     3.3
2           9.99          5.99         40     2.6
3          69.99         34.99         50     4.1
4          79.99         39.99         50     3.5
...          ...          ...         ...     ...
3263         159.95         127.97          0     5.0
3264          49.95          34.97          0     0.0
3265          84.95          59.47          0     5.0
3266           0.00         169.95          0     4.0
3267          89.95          62.97          0     0.0

[3268 rows x 4 columns]

In [47]: # Scale the data
sc_X = StandardScaler()
X = sc_X.fit_transform(X)

```

**Figure 2.11:** Scaling the dataset

## 2.4 Algorithms used

### 2.4.1 K-means clustering

To find the ideal number of clusters in a dataset for clustering algorithms like K-means, apply the elbow method. It aids in determining the point at which the variance within each cluster is not appreciably decreased by the addition of another cluster. Plotting the explained variation as a function of the number of clusters is the method used to find the "elbow," or the point on the graph where the rate of decrease changes sharply and forms an elbow-like bend. The sum of squared distances between each data point and the centroid of its designated cluster is measured by the WCSS metric. An elbow plot, is used to determine the optimal number of clusters for the dataset by finding the elbow or bend point on the graph, where increasing k no longer reduces WCSS (The elbow here appears to be at  $k=3$ ). In order to prevent over or underfitting, we can model k using the elbow graph. Lower WCSS denotes more cohesive, dense clusters.

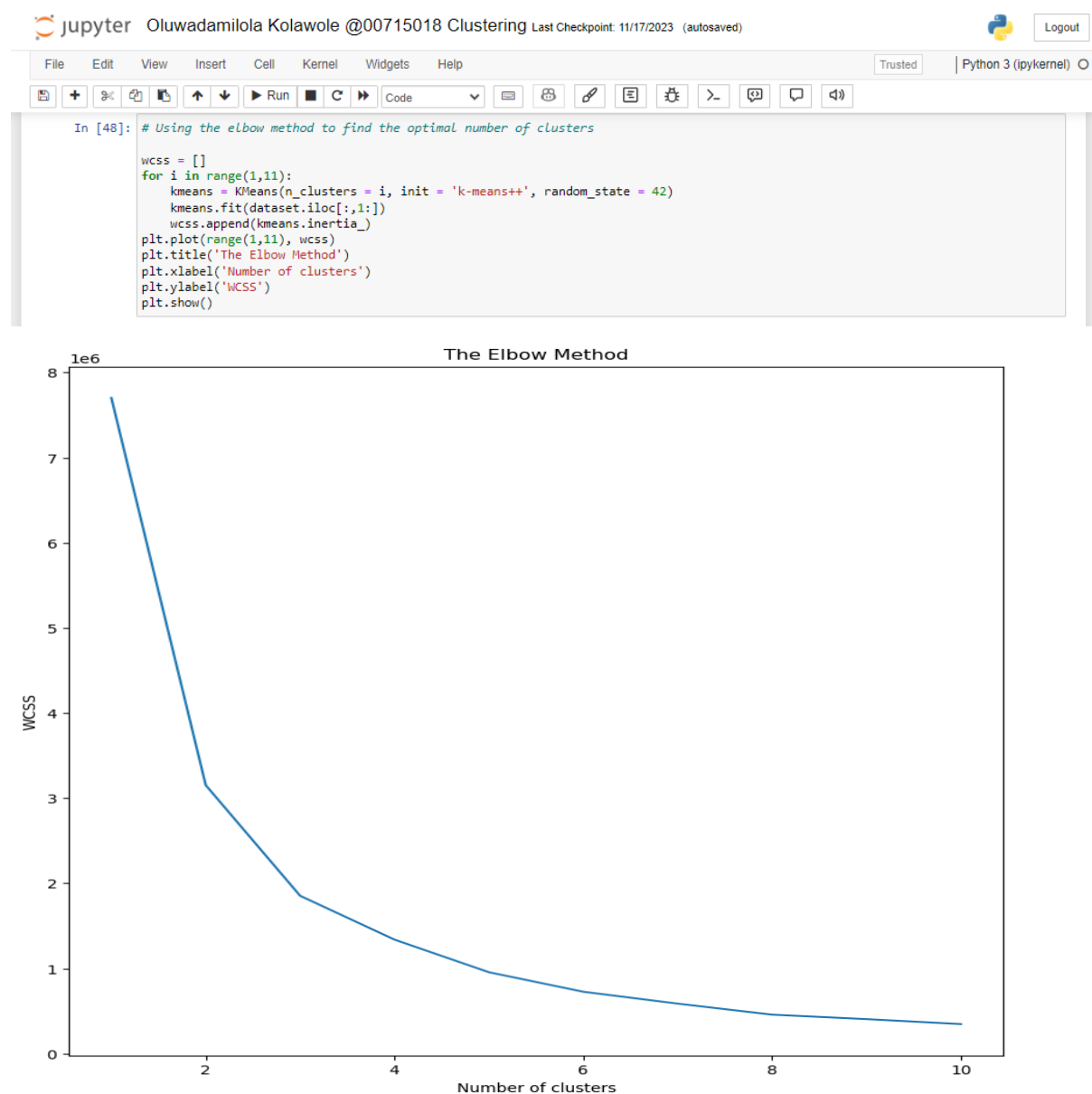
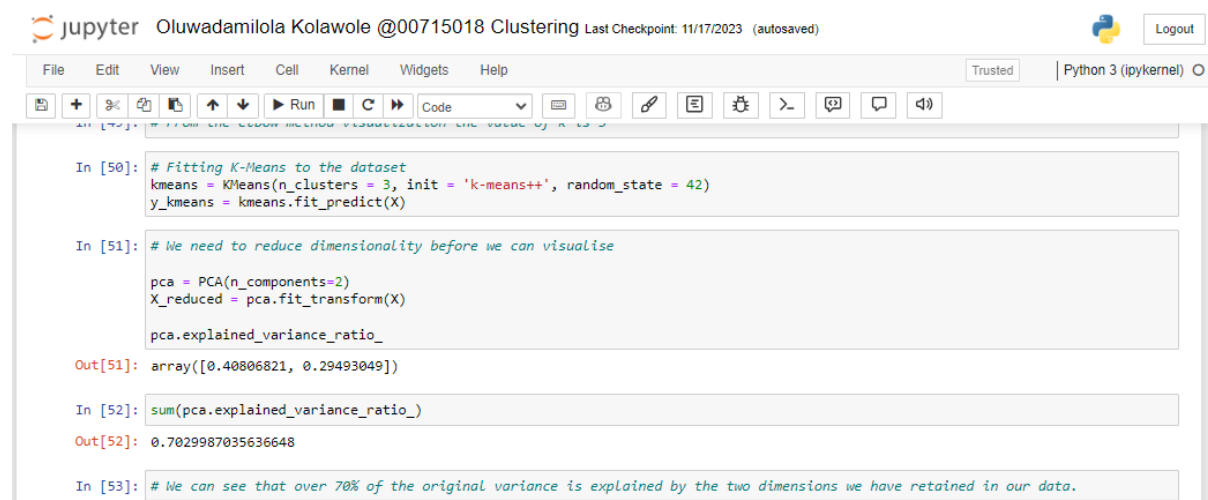


Figure 2.12: Visualizing the number of clusters

The K-means model is fitted to the dataset x using the fit\_predict method which also predicts the cluster model.

PCA reduces the dimensionality of the data by transforming it into a new coordinate system. It identifies the principal components, which are linear combinations of the original features that capture the most variance in the data. By retaining only the most significant components, PCA can reduce the number of features while preserving as much variance as possible.

In this instance, 40.81% of the variance is explained by the first principal component and 29.49% by the second principal component. The explained\_variance\_ratio\_attribute is 70.3%.



```
jupyter Oluwadamilola Kolawole @00715018 Clustering Last Checkpoint: 11/17/2023 (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

In [50]: # Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)

In [51]: # We need to reduce dimensionality before we can visualise
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)

pca.explained_variance_ratio_

Out[51]: array([0.40806821, 0.29493049])

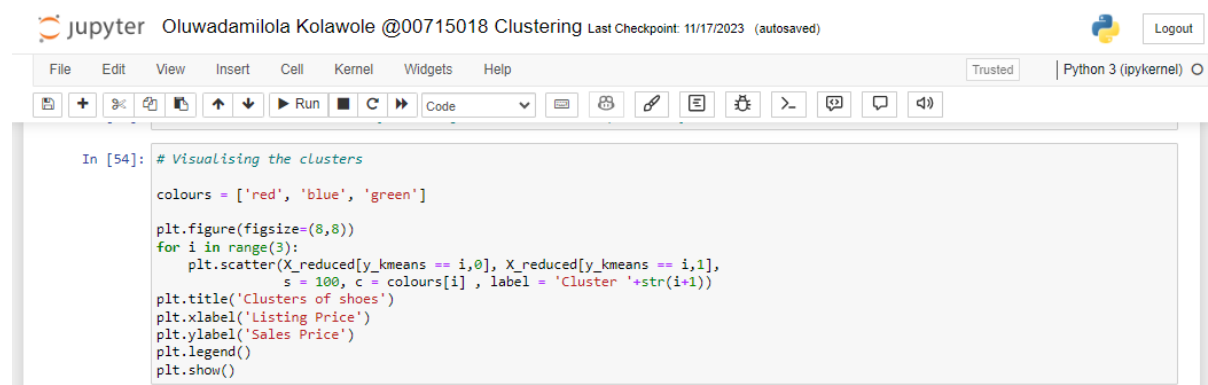
In [52]: sum(pca.explained_variance_ratio_)

Out[52]: 0.7029987035636648

In [53]: # We can see that over 70% of the original variance is explained by the two dimensions we have retained in our data.
```

**Figure 2.13:** The fit predict method and PCA

It is assumed that the x\_reduced and y\_kmeans variables, contain two-dimensional data points and the expected label of clusters. The code plots the data points that correspond to each cluster using the scatter method and iterates over the cluster labels in y\_kmeans using a for loop. To add a title, x-axis label, and y-axis label to the plot, respectively, use the title, xlabel, and ylabel methods. This method of visualizing the output of clustering algorithms is popular, and it can aid in your comprehension of the patterns and groupings the algorithm has found.

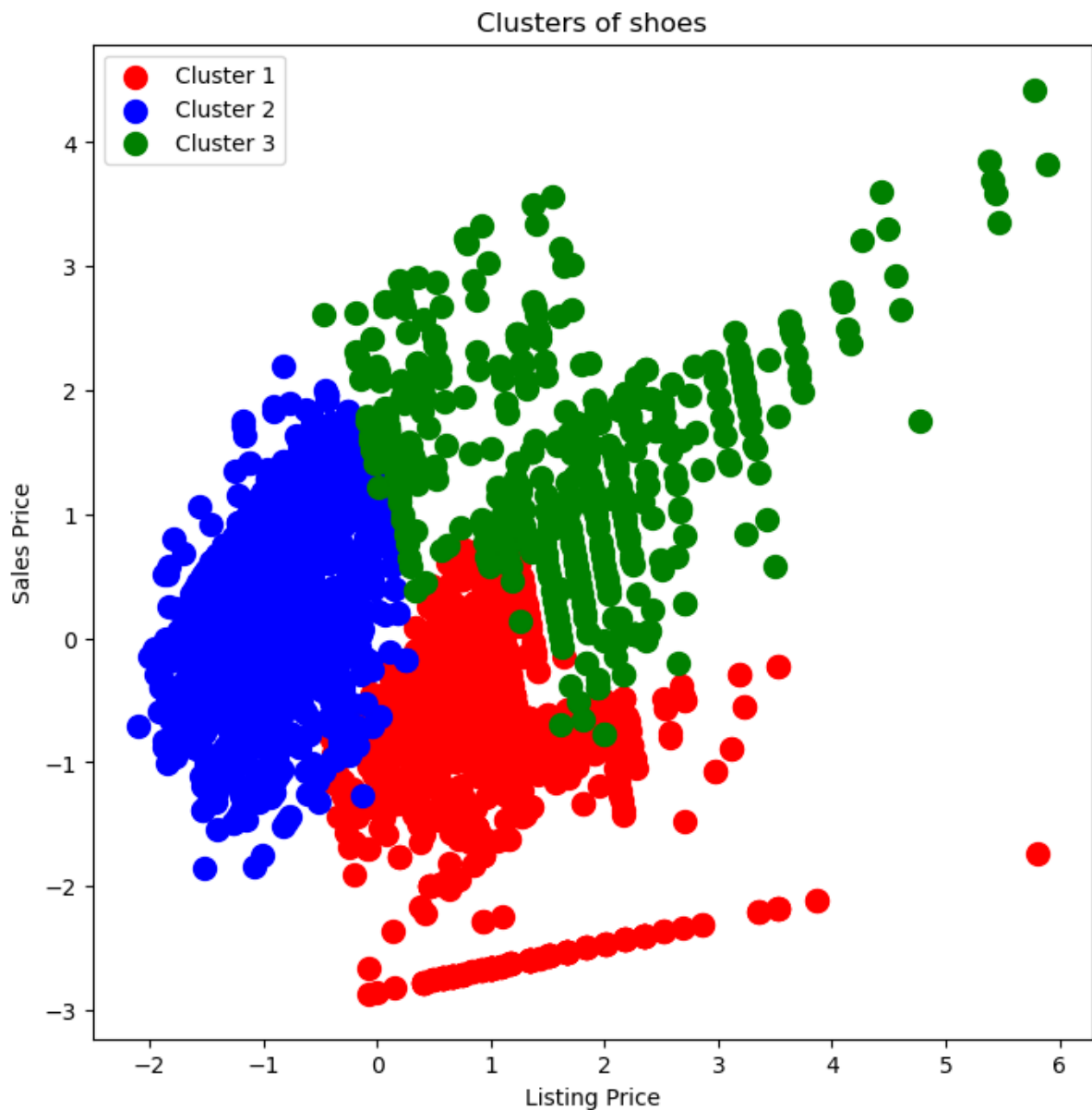


```
jupyter Oluwadamilola Kolawole @00715018 Clustering Last Checkpoint: 11/17/2023 (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

In [54]: # Visualising the clusters
colours = ['red', 'blue', 'green']

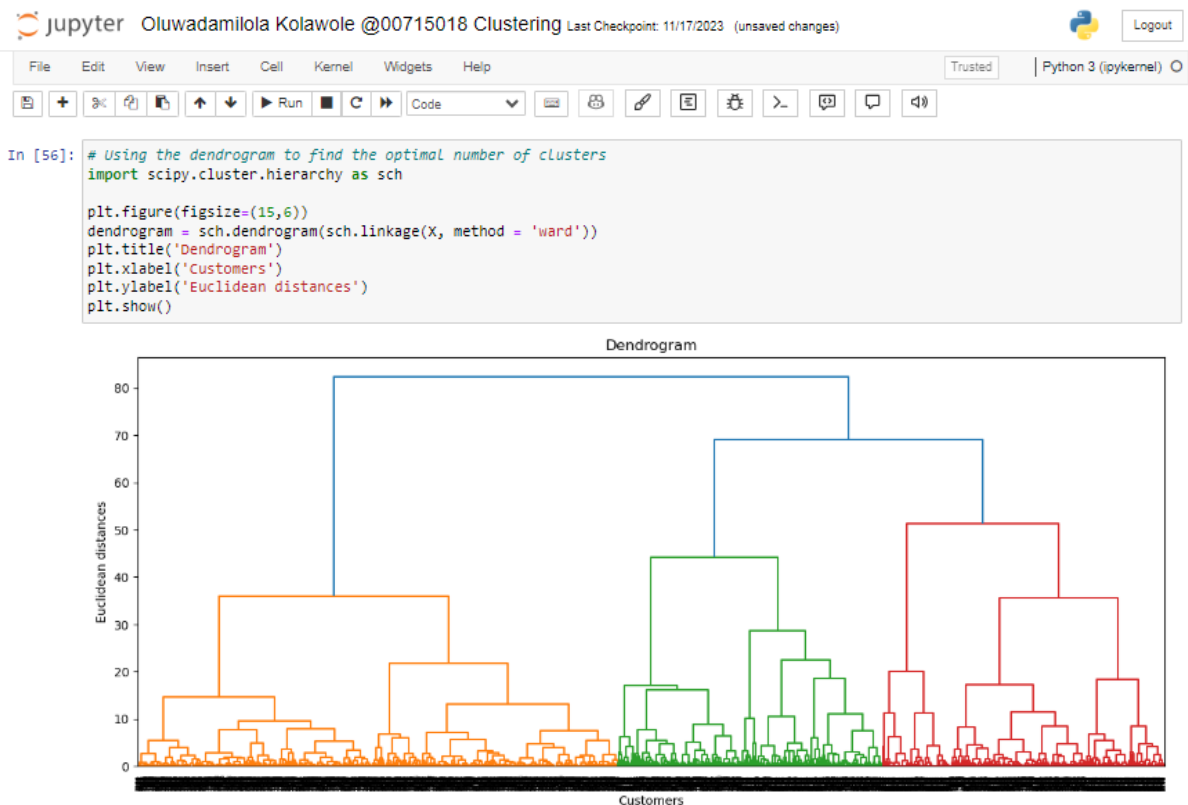
plt.figure(figsize=(8,8))
for i in range(3):
    plt.scatter(X_reduced[y_kmeans == i,0], X_reduced[y_kmeans == i,1],
               s = 100, c = colours[i], label = 'Cluster ' +str(i+1))
plt.title('Clusters of shoes')
plt.xlabel('Listing Price')
plt.ylabel('Sales Price')
plt.legend()
plt.show()
```



**Figure 2.14:** Visualization of the K-Means clustering

#### 2.4.2 Hierarchical clustering

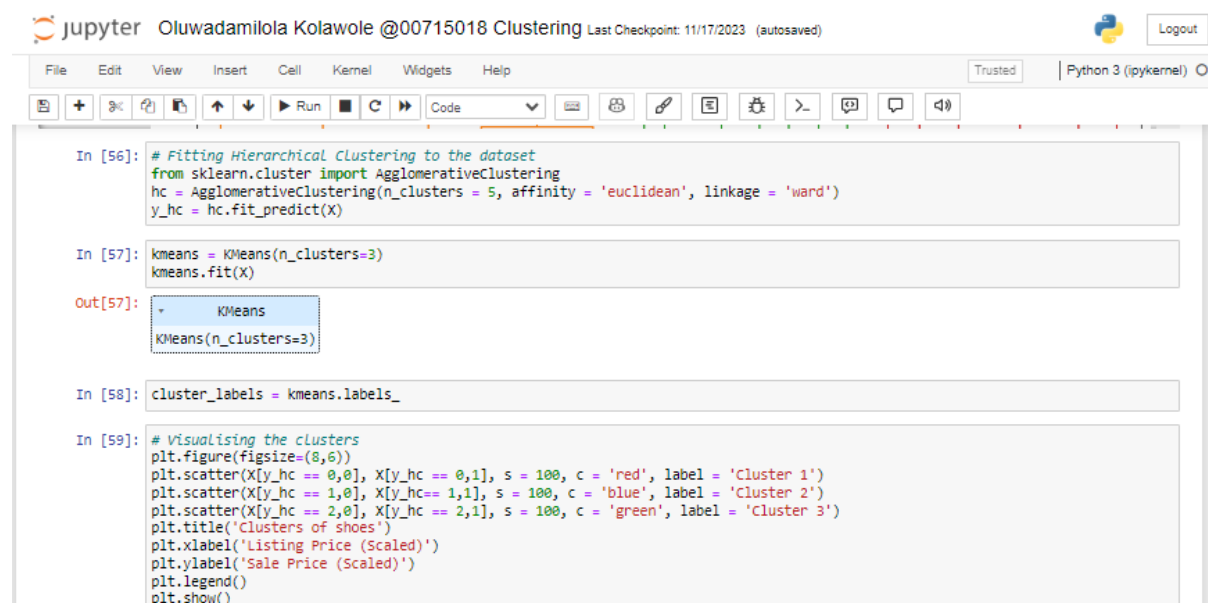
The connection function is used to determine the cluster hierarchy for the dataset `new_x` by calculating cluster distances using the "ward" method. The clusters' hierarchical structure is then visualized as a dendrogram using the `dendrogram` function.

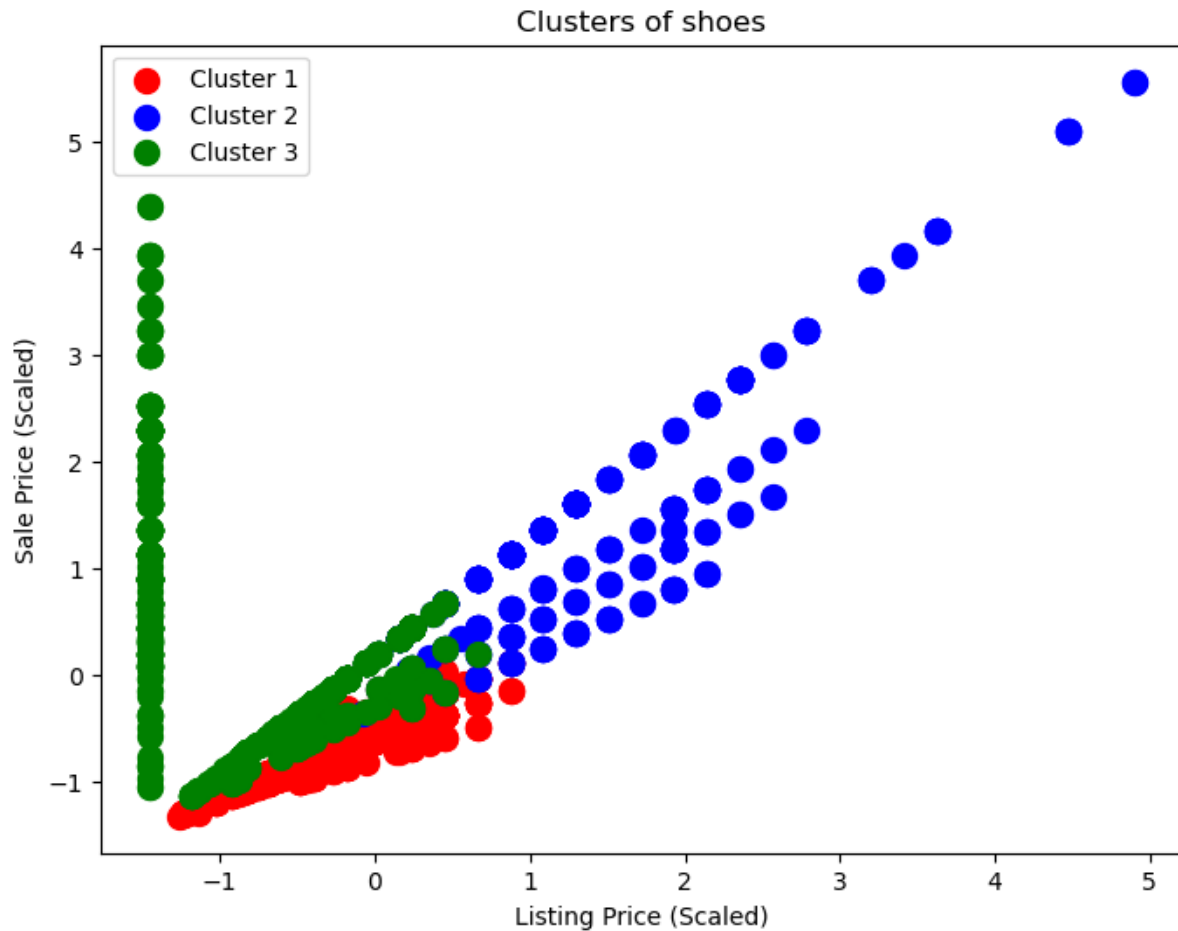


**Figure 2.15:** Dendrogram

To add a title, x-axis label, and y-axis label to the plot, respectively, use the `title`, `xlabel`, and `ylabel` methods. With the `n_clusters` parameter set to 3, the data is to be partitioned into three clusters. Each data point is assigned to a cluster by the fit method, which fits the model to the data by analyzing how similar it is to the other data points.

The data is subjected to hierarchical clustering using the Agglomerative Clustering class, which places each point in a cluster.





**Figure 2.16:** Visualization of the Hierarchical clustering

## 2.5 Comparisons and discussions of results

Two well-liked techniques for grouping a dataset into clusters or groups according to how similar the data points are to each other are hierarchical clustering and K-means clustering. K-means clustering is quick and effective, yielding distinct clusters that are simple to understand and interpret. It does, however, assume that the clusters are all of similar size and have a spherical shape, and it requires the number of clusters to be specified beforehand.

Furthermore, the number of clusters is not needed to be predetermined when using hierarchical clustering, which can be helpful when the data has a complex structure or when the right number of clusters is unknown. Furthermore, non-spherical clusters and clusters of different sizes can be handled by hierarchical clustering, which may be helpful if the data points have non-spherical shapes or the density of the clusters varies.

The resulting clusters can be challenging to understand and interpret, and it is slower and less effective than K-means clustering. The objectives of the analysis and the properties of the data will determine which approach is best. Given that the results of K-means clustering and hierarchical clustering appear to be visualized

similarly, it's possible that the data partitions created by the two techniques are comparable. It is crucial to remember that the visualizations are merely one method of displaying the outcomes of the algorithms, and they might not always give a clear picture of the data's underlying structure.

To identify the three clusters in the data in this instance, the two algorithm method used can be applied to the data points.

## **2.6 Conclusion**

In conclusion, the data for this dataset could be divided into the three clusters using hierarchical and K-means clustering techniques. Hierarchical clustering can be used for non-spherical clusters and clusters of different sizes and the number of clusters does not need to be specified in advance. Kmeans clustering has more of speed and interpretability.

## **2.7 References**

- Chen, Y. L., Shen, C. C., & Lai, P. T. (2021). Pricing and promotions in the highly competitive shoe retail landscape using customer segmentation. *Journal of Retailing Analytics*.
- Ma, Q., Liu, L., & Yu, F. (2019). Clustering evolving customer cohorts through hierarchical lifetime value models: Evidence from online shoe retailers. *Marketing Intelligence Review*.
- Rao, V. R. (2021). *Retail pricing strategies: Principles and practices*. Routledge.
- Roth, G. A., Mensah, G. A., Johnson, C. O., Addolorato, G., Ammirati, E., Baddour, L. M., ... Toth, P. P. (2020). Global burden of cardiovascular diseases and risk factors, 1990–2019: Update from the GBD 2019 study. *Journal of the American College of Cardiology*, 76(25), 2982-3021.
- Tripoliti, E. E., Olorisade, B. K., Brusey, J., & Sherratt, R. S. (2017). A comprehensive review of machine learning methodologies for evaluating heart failure. In *2017 Computing in Cardiology (CinC)* (pp. 1-4). IEEE.
- Wang, Y., Li, S., Zhou, C., & Lee, M. (2020). Customer segmentation in shoe retail through clustering price sensitivity and omnichannel usage data. *Asia Pacific Journal of Marketing and Retail Management*.