**CAPSTONE PROJECT**

**KISHORE VASUDEVAN**

**22/07/2025**

**DATA ANALYTICS & DATA SCIENCE**

**FEB 25’**

**INTRODUCTION:**

This project focuses on analyzing e-commerce shoe data collected through web scraping from online platforms. Using tools like Selenium and BeautifulSoup, the data was extracted and cleaned for further analysis. The aim was to uncover patterns in the data using unsupervised learning (clustering) and predict product categories through classification models. Key features such as price, brand, and customer rating were considered. Based on domain knowledge, products were grouped into segments like low, medium, and premium. The final structured data was stored in a MySQL database to enable organized access and usage for reporting or future analysis**.**

**OBJECTIVES:**

* To collect shoe product data from e-commerce websites using web scraping techniques.
* To clean and prepare the data for machine learning applications.
* To segment products into meaningful categories using clustering.
* To classify products based on brand, price, and rating using supervised learning models.
* To evaluate and compare different models for better performance.
* To store the final dataset in a MySQL database for structured access and usage.

**TOOLS USED:**

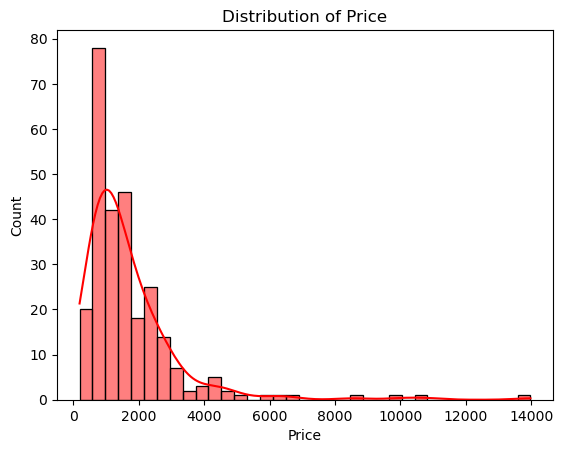
* Python – For data scraping, processing, and machine learning tasks.
* Selenium & BeautifulSoup – For web scraping and extracting structured data from websites.
* Pandas & NumPy – For data cleaning, transformation, and manipulation.
* Matplotlib & Seaborn – For data visualization and exploratory analysis.
* Scikit-learn – For implementing machine learning models and evaluation metrics.
* XGBoost – For building a high-performance classification model.
* MySQL – For storing the cleaned and final dataset in a structured database.

**WEB SCRAPPING:**

* Scraped complete shoe product listings from an e-commerce website without applying any brand or category filters
* Implemented Selenium to handle dynamic content, automate clicks, and scroll through multiple pages
* Used BeautifulSoup to parse HTML and extract structured product information
* Collected essential features such as Product Title, Brand, Price, and User Rating
* Ensured delay handling and page loading with time.sleep() to avoid blocking or bans
* Added logic to navigate through all available pages until the end of the catalog
* Stored the final scraped dataset in Pandas DataFrame for further analysis and cleaning
* Verified data consistency and checked for missing values before moving to preprocessing

**DATA CLEANING & EDA:**

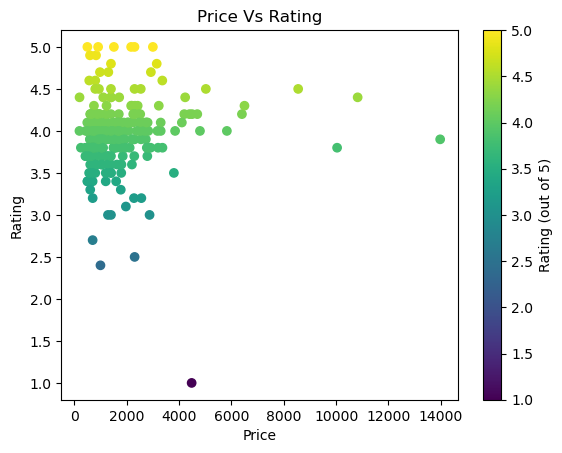
* Removed null values and duplicate records to maintain data quality.
* Converted the Rating column into float format for accurate numerical analysis.
* Verified and corrected data types to ensure compatibility with modeling algorithms.
* Created a new Category column by extracting shoe types from the title (e.g., Sneakers, Running Shoes, Walking Shoes, Others).
* Performed univariate and bivariate analysis to understand distribution and relationships.
* Visualized key metrics like price range, ratings, and brand frequency for insights.



The distribution is **right-skewed** (positively skewed), meaning:

1.Most of the shoe products are priced **between ₹500 and ₹3,000**.

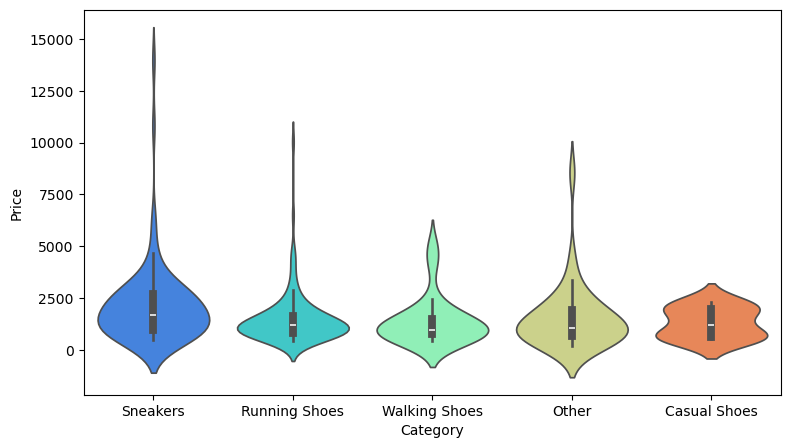
2.Only a few high-end products are priced **above ₹5,000**, extending the tail of the curve.

* There's a **sharp peak** around ₹1,000–₹1,500, suggesting that this is the most common price range.
* Very few items fall in the higher price brackets like ₹10,000 and above.
* The tail shows the dataset has the outliers.
* **Most shoes are priced under ₹3000** and tend to receive **higher ratings (above 4)**.
* There's **no strong correlation** between price and rating — high-rated products are available in both low and high price ranges.
* A **majority of products cluster between ₹500 and ₹2500**, with ratings between **3.5 and 5**.
* A few **expensive shoes** (₹8000 and above) are present but are **not always rated higher** than cheaper ones.
* **Outliers** can be seen — for example, a shoe priced around ₹5000 with a rating close to **1**, which might indicate customer dissatisfaction.



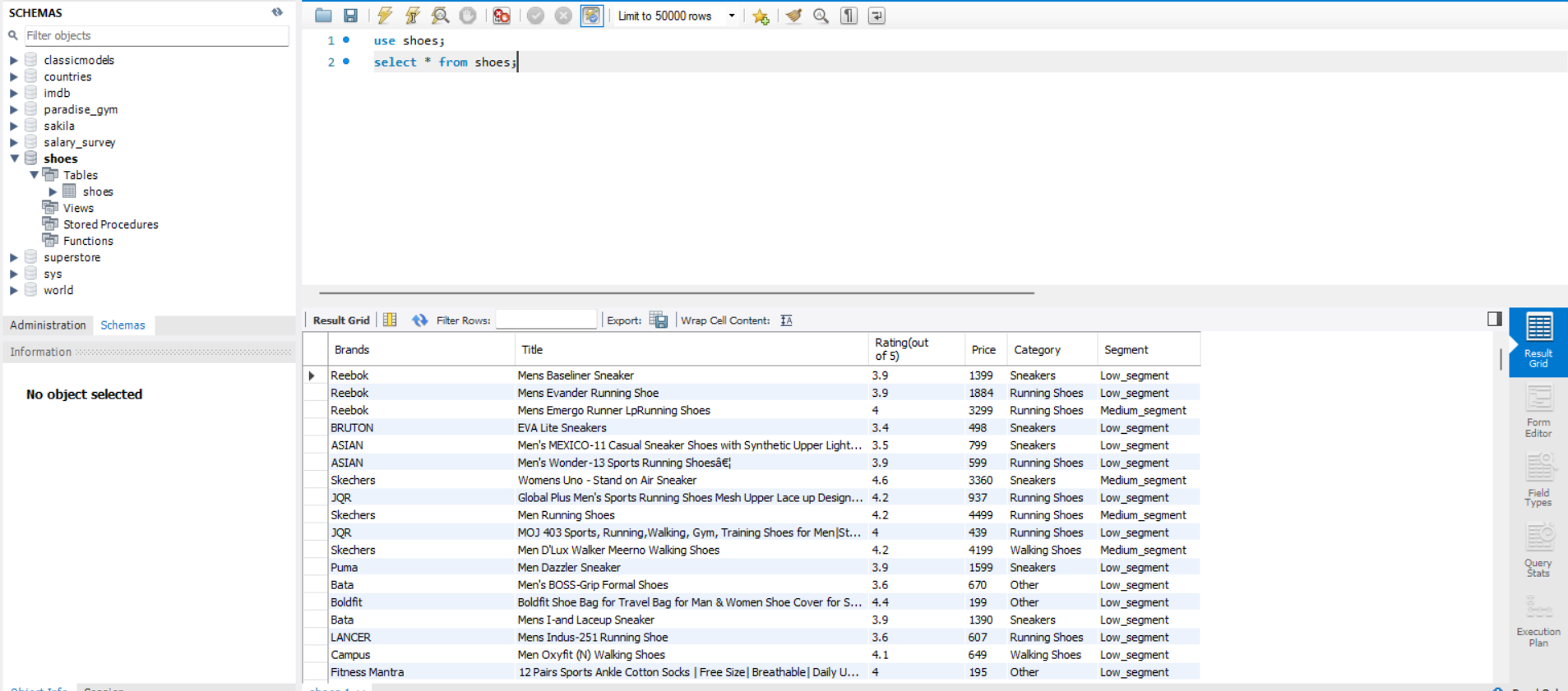
Correlation Coefficient (0.078):

* This is a very weak positive correlation.
* Values close to 0 indicate almost no linear relationship.
* So, as price increases, ratings don't necessarily increase (or decrease) in any consistent way.



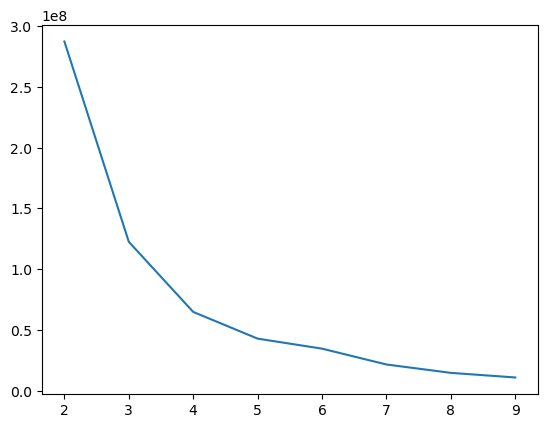
* **Sneakers are premium-priced items**, possibly due to brand value or design features.
* **Casual and Walking Shoes are more budget-friendly**, making them more consistent in pricing.
* **Outliers** in every category (the long tails) show that some products are priced **significantly higher** than average.

**SQL IMPORT(DATA STORAGE):**

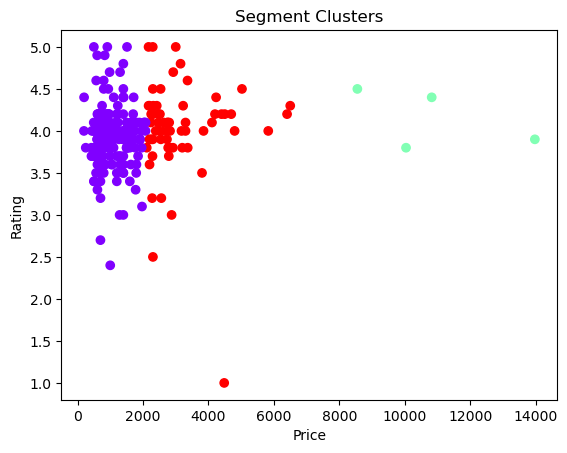
* Created a new schema named **shoes** using the right-click context menu in MySQL Workbench.
* Created a table also named **shoes** under the schema for storing final cleaned and labeled data.
* ****Used the **Table Data Import Wizard** to load the processed .csv file into the MySQL table.

**UNSUPERVISED LEARNING:**

**ELBOW METHOD:**



* The plot shows the number of clusters (K) on the x-axis and the Within-Cluster Sum of Squares (WCSS) on the y-axis.
* As K increases, the WCSS decreases — meaning the clusters fit better.
* However, after a certain point, the decrease becomes marginal. This point is called the "elbow".
* In this chart, the elbow appears around K = 3 or 4, suggesting that 3 or 4 clusters is optimal for grouping the shoe data.
* Choosing the elbow point balances model accuracy and simplicity, avoiding overfitting.

**KMEANS CLUSTERING:**

**kmeans = KMeans(n\_clusters=3,n\_init=10,random\_state=42)**

**df["Segment"] = kmeans.fit\_predict(X)**

**Cluster 0 (Red):**

* Mid-range prices (₹1500–₹6000)
* High ratings (mostly 4 to 5)
* Represents **popular, well-rated shoes**

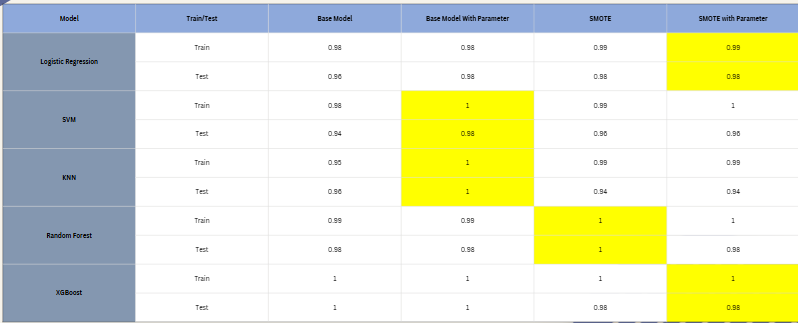
**Cluster 1 (Purple):**

* Low-priced shoes (below ₹3000)
* Very high ratings (mostly above 4)
* Indicates **value-for-money products**

**Cluster 2 (Green):**

* High-priced segment (₹8000–₹14000)
* Moderate to high ratings
* Represents **premium or luxury products**

**SUPERVISED LEARNING:**

****

* **Logistic Regression**:
  + Consistently high accuracy (up to **0.99**) with **SMOTE + Tuning**
  + Very stable between train and test sets
* **SVM**:
  + Reached **100% accuracy** on training after tuning
  + Test performance remains strong (**0.96-0.98**) with SMOTE
* **KNN**:
  + Training reaches **1.0** with tuning
  + Slight drop in test accuracy with SMOTE (**0.94**)
* **Random Forest**:
  + Slight overfitting without SMOTE
  + SMOTE + tuning improves test accuracy (**0.98**)
* **XGBoost**:
  + Perfect accuracy (**1.0**) on both Train and Test with tuning
  + Slight drop with SMOTE, but still excellent (**0.98**)

**OBSERVATION:**

* Hyperparameter tuning significantly boosts performance for all models
* SMOTE helps in handling class imbalance, improving test set generalization
* XGBoost shows the highest overall performance but may risk overfitting

**FINAL MODEL SELECTION:**

* XGBOOST is final selected model.
* Achieved perfect accuracy (1.0) on both train and test sets before tuning.
* Maintained high accuracy (0.98) even after applying SMOTE with parameter tuning.
* Demonstrated strong generalization and robustness across all configurations.
* Best suited for handling imbalanced data and complex patterns in the dataset.

**CONCLUSION:**

* Successfully scraped shoe data from Amazon using Selenium and BeautifulSoup.
* Performed comprehensive data cleaning, normalization, and preprocessing.
* Applied unsupervised learning (KMeans) to cluster products into pricing segments: Low, Medium, and High.
* Labeled clustered data and used it for supervised classification using multiple ML models.
* XGBoost with SMOTE & hyperparameter tuning gave the best performance with 98% accuracy on test data.
* Final cleaned and labeled data was imported into MySQL for persistence and future usage.
* The project demonstrates a complete ML pipeline: from web scraping to database integration.