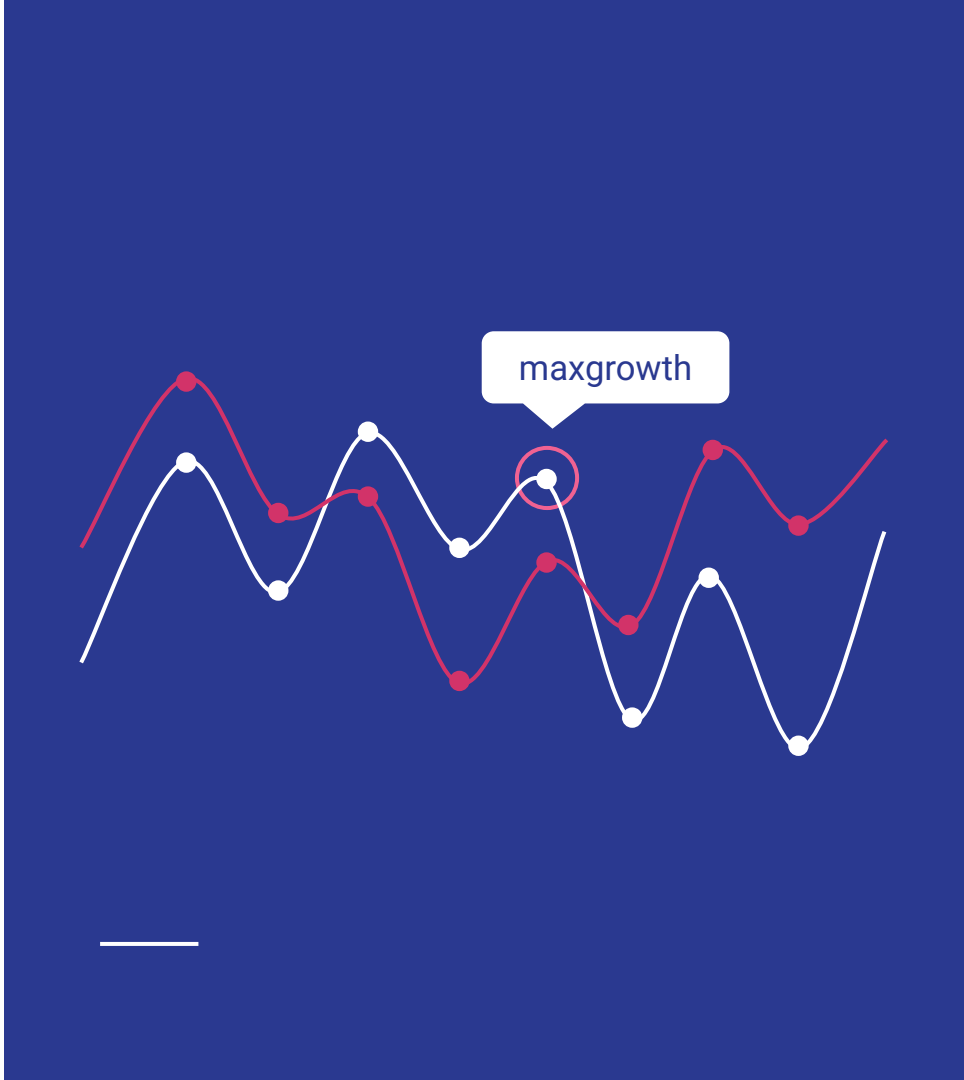
Analytics in Retail

By Prof.Vishal Chugh



What is Analytics in Retail ?

Analytics in Retail refers to the process of collecting, processing, and analyzing data from various retail operations—such as sales, customers, products, supply chain, marketing, and inventory—to generate actionable insights that improve decision-making, efficiency, and profitability.

Key Areas of Retail Analytics

Customer Analytics

Understanding customer behavior, preferences, and lifetime value for better targeting, personalization, and retention.

Sales Analytics

Tracking sales trends, seasonality, promotions, and pricing effectiveness.

Inventory Analytics

Ensuring the right product is available at the right place and time while reducing stockouts and overstocking.

Key Areas of Retail Analytics

Marketing Analytics

Measuring campaign performance, channel effectiveness, and promotion ROI.

Ecom Analytics

Analyzing footfall, dwell time, online browsing, conversion rates, and channel effectiveness.

Product Analytics

Identifying bestsellers, slow movers, and optimizing product assortment and pricing.

Benefits of Retail Analytics

Forecasting	Pricing & Promotion	Inventory
Better Demand Forecasting	Optimized pricing and promotions	Improved inventory management

Benefits of Retail Analytics

Personalization

Enhanced customer experience through personalization

Sales

Increased sales and profitability

Data Driven Decisions

Data-driven strategic planning

Retail Analytics transforms raw retail data into business intelligence to help retailers stay competitive in a dynamic market.”

Supermarket (Offline Retail)

🛒 Example 1: Supermarket (Offline Retail)

- **Data Collected:**
  - What products are being bought
  - Time of day/week when customers shop
  - Basket size (how many items per visit)
  - Which products are often bought together
- **How Analytics is Used:**
  1. **Product Placement:** If data shows people buying *chips and cold drinks together*, the store places them side by side to increase sales.
  2. **Demand Forecasting:** If sales data shows *ice creams sell more in summer evenings*, the supermarket stocks extra ice creams before summer starts.
  3. **Promotion Analytics:** If "Buy 1 Get 1 Free" on shampoo leads to higher sales but no profit, the store adjusts the promotion strategy.
- **Outcome:** Reduced stockouts, better promotions, and higher sales per customer visit.

Amazon (Online Retail)

📦 **Example 2: Amazon (Online Retail)**

- **Data Collected:**
  - Customer browsing history
  - Purchase history
  - Click-through rate on recommended items
  - Delivery speed and returns data
- **How Analytics is Used:**
  1. **Personalized Recommendations:** If you buy a laptop, Amazon suggests *laptop bags, mouse, and antivirus software*.
  2. **Dynamic Pricing:** Prices change based on demand, competitor pricing, and season (e.g., discounts on Black Friday).
  3. **Inventory Optimization:** Analytics predicts which warehouses need to stock which items so delivery is faster.
- **Outcome:** Higher customer satisfaction, increased repeat purchases, and lower logistics cost.

👉 In short:

- Supermarkets use analytics to **decide where to place products, what to stock, and when to promote.**
- Amazon uses analytics to **personalize shopping, set competitive prices, and deliver fast.**

What is Market Basket Analysis

Market Basket Analysis is a data mining technique used in retail to identify patterns of products that are frequently bought together. Example: If customers often buy bread and butter together, MBA helps retailers:

It is widely used in recommendation systems, like Amazon's "Customers who bought this also bought …".

Key Metrics in Market Basket Analysis

To measure these associations, we use **Association Rules**, expressed as:

```
css

{A} → {B}
```

Meaning: "If a customer buys A, they are likely to also buy B."

The strength of such rules is measured using **Support, Confidence, and Lift**:

Support

- Definition: Proportion of transactions in which *both A and B appear together.*
- Formula:

$$\text{Support}(A \rightarrow B) = \frac{\text{Transactions containing both A and B}}{\text{Total Transactions}}$$

- Example: If 100 customers shopped, and 15 bought **bread & butter together**,

$$\text{Support} = \frac{15}{100} = 0.15\ (15\%)$$

- Meaning: 15% of all transactions include bread and butter.

Confidence

- Definition: Probability that a customer buys B **given that they bought A.**
- Formula:

$$\text{Confidence}(A \rightarrow B) = \frac{\text{Transactions with A and B}}{\text{Transactions with A}}$$

- Example: If 20 people bought **bread**, and 15 of them also bought **butter**,

$$\text{Confidence} = \frac{15}{20} = 0.75\ (75\%)$$

- Meaning: 75% of people who buy bread also buy butter.

Lift

- Definition: How much more likely customers are to buy B when they buy A, compared to just buying B anyway.
- Formula:

$$\text{Lift}(A \to B) = \frac{\text{Confidence}(A \to B)}{\text{Support}(B)}$$

- Example:
  - Confidence (Bread → Butter) = 75%
  - Support(Butter) = 30% (30 out of 100 customers buy butter overall)

$$\text{Lift} = \frac{0.75}{0.30} = 2.5$$

- Meaning: Customers who buy bread are **2.5 times more likely** to buy butter than the average shopper.

Summary of Metrics

✅ **Summary of Metrics**

- **Support** → How often A & B occur together.
- **Confidence** → How often B is bought when A is bought.
- **Lift** → How much stronger the A → B relationship is compared to random chance.

User Defined Functions

By Prof.Vishal Chugh

User Defined Functions

```
def fun( x ):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")

fun(2) # function calling
```

Output

```
even
```

```
def fun(name):
    print("Hello,", name)

# function call
fun("shakshi")
```

Output

```
Hello, shakshi
```

Explore the Data in Python (Online - Retail)

By Prof.Vishal Chugh

Import Libraries

```
# import libraries
from cryptography.fernet import Fernet
from mlxtend.frequent_patterns import apriori, association_rules
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

After Decrypting We Load the Dataset

```
# Load the dataset through Pandas
df = pd.read_excel('online_retail.xlsx')
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Shape of the data (rows, columns)

```
# Shape
df.shape
```

```
(541909, 8)
```

Missing Values

```
# Missing Values
df.isna().sum()
```

	0
InvoiceNo	0
StockCode	0
Description	1454
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0

Data Types

```
# datatypes
df.dtypes
```

	0
InvoiceNo	object
StockCode	object
Description	object
Quantity	int64
InvoiceDate	datetime64[ns]
UnitPrice	float64
CustomerID	float64
Country	object

Make the copy of the Data

```
# Make of the copy of the data
df_copy = df.copy(deep = True)
df_copy.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Unique Values In Description

```
# Unique Description
df_copy['Description'].nunique()
```

```
4223
```

Remove White Spaces In Description

```
# We want to get rid of white spaces
df_copy['Description'] = df_copy['Description'].str.strip()
```

Again Check Unique Values in Description

```
# New Unique
df_copy['Description'].nunique()
```

```
4210
```

Price and Description Mapping

```
# Price and Description mapping
df_copy['Description'] = df_copy['Description'].astype(str)
price_desccription_map = df_copy.dropna(subset = ['Description']).drop_duplicates(subset = ['UnitPrice'], keep = 'first').set_index('UnitPrice')['D
df_copy['Description'] = df_copy['Description'].fillna(df_copy['UnitPrice'].map(price_desccription_map))
```

Check for the Missing Values

```
# Check for the Missing Values
df_copy.isna().sum()
```

	0
InvoiceNo	0
StockCode	0
Description	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0

Calculate The Sales

```
# Calculate the Sales
df_copy['Sales'] = df_copy['Quantity'] * df_copy['UnitPrice']
```

Show the cancelled invoices

```
# Show the cancelled Invoices
df_copy[df_copy['InvoiceNo'].astype(str).str.contains('C')].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales
141	C536379	D	Discount	-1	2010-12-01 09:41:00	27.50	14527.0	United Kingdom	-27.50
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	2010-12-01 09:49:00	4.65	15311.0	United Kingdom	-4.65
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	2010-12-01 10:24:00	1.65	17548.0	United Kingdom	-19.80
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	17548.0	United Kingdom	-6.96
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	17548.0	United Kingdom	-6.96

Exclude the cancelled invoices

```
# Exlude the Cancelled invoices
df_copy['InvoiceNo'] = df_copy['InvoiceNo'].astype(str)
df_new = df_copy[~df_copy['InvoiceNo'].str.contains('C')]
df_new.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34

Grouping

```
# Grouping
grouped_df = df_copy[df_copy['Country'] == 'Germany'].groupby(['InvoiceNo', 'Description']).size().unstack().fillna(0).reset_index().set_index('I
grouped_df.head()
```

Custom Encoding

```
# Custom  Encoding
def custom_encoding(x):
  if x <= 0:
    return 0
  if x >= 1:
    return 1
grouped_df = grouped_df.applymap(custom_encoding)
grouped_df.drop('POSTAGE', inplace = True, axis = 1)
```

Apriori

```
# Frequent Items
frequent_items = apriori(grouped_df, min_support = 0.05, use_colnames = True)
```

Association Rules

```
# Rules
rules = association_rules(frequent_items, metric = 'confidence', min_threshold = 0.06)
```

Final Outcomes

```
# Final Outcome
outcome = rules.sort_values(by ='lift', ascending = False)
outcome
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.109453	0.094527	0.051410	0.469697	4.968900
1	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.094527	0.109453	0.051410	0.543860	4.968900
5	(ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.129353	0.197347	0.102819	0.794872	4.027796

Explore the Data in Python (OmniChannel - Retail)

By Prof.Vishal Chugh

After Decrypting We Load the Dataset

```
# Load the data with pandas
retail = pd.read_csv('retail_data.csv')
retail.head()
```

	visit_id	customer_id	visit_date	channel	dwell_seconds	is_purchase	items	num_items	total_amount	total_cogs	return_flag
0	V100029	C0297	2025-06-01 02:35:05	E-Commerce	106	True	Chips (x1)	1	40.0	23.86	True
1	V100255	C0025	2025-06-01 07:26:34	Store	151	True	Floor Cleaner (x1)	1	180.0	115.54	False
2	V100057	C0087	2025-06-01 12:08:39	E-Commerce	710	False	NaN	0	0.0	0.00	False
3	V100472	C0046	2025-06-02 00:36:32	E-Commerce	62	False	NaN	0	0.0	0.00	False
4	V100454	C0169	2025-06-02 07:42:42	App	44	False	NaN	0	0.0	0.00	False

Shape of the Data

```
# Shape of the data
retail.shape
```

```
(500, 17)
```

Missing Values

```
retail.isna().sum()
```

	0
visit_id	0
customer_id	0
visit_date	0
channel	0
dwell_seconds	0
is_purchase	0
items	391
num_items	0
total_amount	0
total_cogs	0
return_flag	0
fulfillment_status	391
satisfaction	391
first_visit_date	0
is_new_customer	0
acquisition_channel	0
Category	391

Check Whether Missing Data is Due to No Purchase

```
# Check whether these are because of no purchase
missing_rows = retail[retail.isna().any(axis = 1)]
missing_no_purchase = (missing_rows['is_purchase'] == 0).all()
print("The missing values shown are due to no purchase?", missing_no_purchase)
```

```
The missing values shown are due to no purchase? True
```

Data Types

```
# Check the datatypes
retail.dtypes
```

	0
visit_id	object
customer_id	object
visit_date	object
channel	object
dwell_seconds	int64
is_purchase	bool
items	object
num_items	int64
total_amount	float64
total_cogs	float64
return_flag	bool
fulfillment_status	object
satisfaction	float64
first_visit_date	object
is_new_customer	bool
acquisition_channel	object
Category	object

Make Copy of the data

```
# copy of the data
retail_copy = retail.copy(deep = True)
retail_copy.head()
```

	visit_id	customer_id	visit_date	channel	dwell_seconds	is_purchase	items	num_items	total_amount	total_cogs	return_flag	fulfillment_status	sa
0	V100029	C0297	2025-06-01 02:35:05	E-Commerce	106	True	Chips (x1)	1	40.0	23.86	True	Fulfilled	
1	V100255	C0025	2025-06-01 07:26:34	Store	151	True	Floor Cleaner (x1)	1	180.0	115.54	False	In-Store Pickup/Immediate	
2	V100057	C0087	2025-06-01 12:08:39	E-Commerce	710	False	NaN	0	0.0	0.00	False	NaN	
3	V100472	C0046	2025-06-02 00:36:32	E-Commerce	62	False	NaN	0	0.0	0.00	False	NaN	
4	V100454	C0169	2025-06-02 07:42:42	App	44	False	NaN	0	0.0	0.00	False	NaN	

Change the data type of visit_date to datetime

```
# Change the datatype for visit date
retail_copy['visit_date'] = pd.to_datetime(retail_copy['visit_date'])
```

Footfalls

```
# Footfalls
customer_footfalls = retail_copy.groupby(retail_copy['visit_date'].dt.date)['customer_id'].nunique().sort_values(ascending = False)
customer_footfalls
```

visit_date	customer_id
2025-08-18	11
2025-07-29	11
2025-07-30	10
2025-08-09	10
2025-08-29	8

Average Dwell Time

```
# Average Dwell Time
avg_dwell_time = retail_copy.groupby('customer_id')['dwell_seconds'].mean()
avg_dwell_time.mean().round(2)
```

```
np.float64(268.11)
```

Store Layout

```
# Store Layout
layout_eff = retail_copy.groupby('Category')['dwell_seconds'].mean().sort_values(ascending = False)
layout_eff
```

Category	dwell_seconds
Grocery	376.187500
Apparel	321.428571
Cosmetics	296.954545
Electronics	274.545455
Stationery	274.380952

Average Spend

```
# Average Spend
customer_spend = retail_copy.groupby('customer_id')['total_amount'].sum().reset_index()
customer_spend.rename(columns = {'total_amount' : 'total_spend'}, inplace = True)
```

Create a column Spend Score

```
retail_copy['spend_score'] = (customer_spend['total_spend'] - customer_spend['total_spend'].min()/
                              customer_spend['total_spend'].max() - customer_spend['total_spend'].min()).round(2)
```

Buying Behaviour Classification

```python
# Buying Behaviour Classification
def classify_behavior(row):
    if row['spend_score'] > 70 and row['dwell_seconds'] > 40:
        return 'Complex Buying'
    elif row['spend_score'] > 70 and row['dwell_seconds'] <= 40:
        return 'Impulsive Buying'
    elif row['spend_score'] < 40 and row['dwell_seconds'] < 20:
        return 'Habitual Buying'
    else:
        return 'Variety Seeking Buying'

retail_copy['Buying_Behavior'] = retail_copy.apply(classify_behavior, axis=1)

print("Buying Behavior Distribution:")
print(retail_copy['Buying_Behavior'].value_counts(normalize=True) * 100)
```

```
Buying Behavior Distribution:
Buying_Behavior
Variety Seeking Buying    83.6
Complex Buying            15.4
Impulsive Buying           1.0
Name: proportion, dtype: float64
```

Channel Footfall

```
# Channel Footfall
channel_footfall = retail_copy.groupby('channel')['customer_id'].nunique().sort_values(ascending = False)
channel_footfall
```

channel	customer_id
Store	153
E-Commerce	130
App	99

Conversion Rate

```
# Conversion rate
conversion_rate = retail_copy.groupby("channel")["is_purchase"].mean().sort_values(ascending = False)
conversion_rate
```

channel	is_purchase
Store	0.240566
E-Commerce	0.206897
App	0.192982

Average Order Value Based on Spend Score

```
# Average Order Value based on spend score
aov = retail_copy.groupby('channel')['spend_score'].mean().sort_values(ascending = False)
aov
```

channel	spend_score
E-Commerce	443.341772
Store	137.634783
App	103.283333

Return Rate

```
# Return Rate
retail_copy["return_flag"] = retail_copy["return_flag"].astype(int)

return_rate = retail_copy.groupby("channel")["return_flag"].mean() * 100
print("Return Rate by Channel (%):")
print(return_rate.round(2))
```

```
Return Rate by Channel (%):
channel
App            1.75
E-Commerce     1.72
Store          0.47
Name: return_flag, dtype: float64
```

Omni-Channel Fulfilment Rate

```
# Omni-channel fulfillment rate (proxy: customers buying in >1 channel)
multi_channel_customers = retail_copy.groupby('customer_id')['channel'].nunique()
omni_rate = (multi_channel_customers  > 1 ).mean()
print('Omni_Channel Fulfillment Rate is', round(omni_rate,2))
```

```
Omni_Channel Fulfillment Rate is 0.43
```

Average Customer Satisfaction

```
# Average Customer Satisfaction
if 'satisfaction' in retail_copy.columns:
    satisfaction = retail_copy.groupby('channel')['satisfaction'].mean().sort_values(ascending = False)
    print(satisfaction)
else:
    satisfaction = "Not available"
    print(satisfaction)
```

```
channel
Store          4.176471
E-Commerce     4.111111
App            4.045455
Name: satisfaction, dtype: float64
```