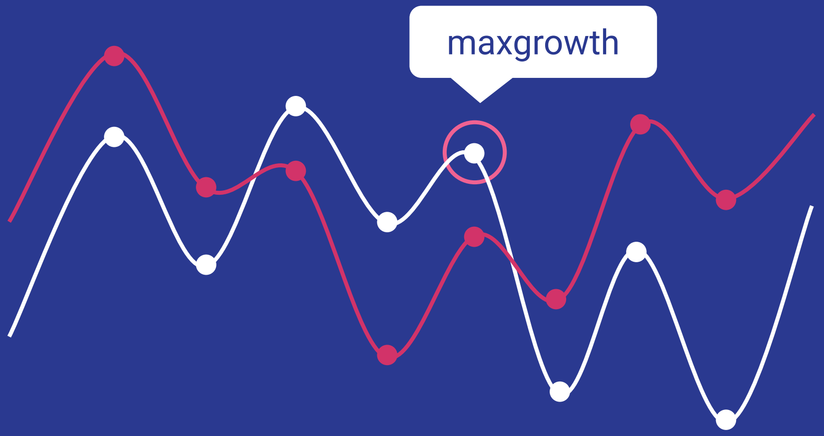
Product Analytics

By Prof.Vishal Chugh



What is Product Analytics ?

Product analytics is the process of collecting and analyzing quantitative data on how users interact with a product, such as feature usage, user journeys, and engagement patterns, to understand user behavior and make data-driven decisions to improve the product's user experience, features, and overall business outcomes.

Importance of Product Analytics

NPD

From idea generation to post-launch iteration, by providing quantitative insights into user needs, product usage, and the market

Product Improvement

User behavioral data to drive continuous product improvement by revealing how users interact with a product, identifying pain points, and enabling informed decision-making

Market Expansion

Providing data-driven insights into new market needs and user behavior

Steps in Product Analytics

Survey / Feedback

Perform regular surveys or feedbacks from time to time to understand what customer wants. This will help business to stay ahead in the competition

A/B Testing

Test various scenarios by segmenting various group of customers with shared characteristics and see where business can get maximum revenue

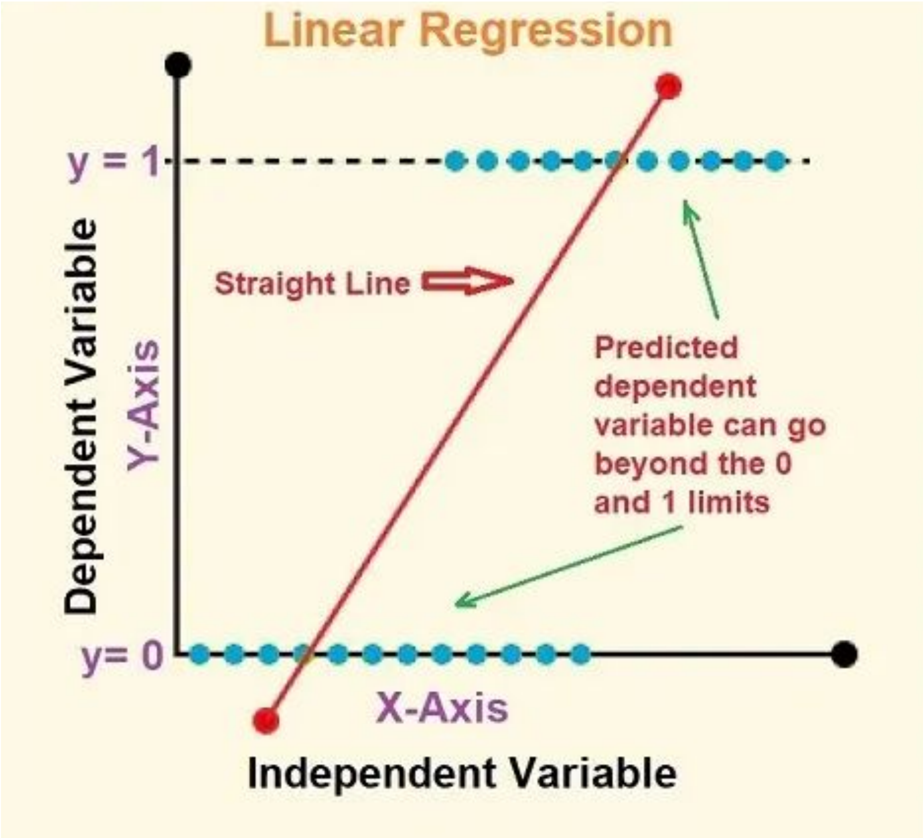
Competition Analysis

Stay updated with the changes competitors are bringing in the market and how it is impacting consumer behaviour

Predictive Analytics

Linear Regression

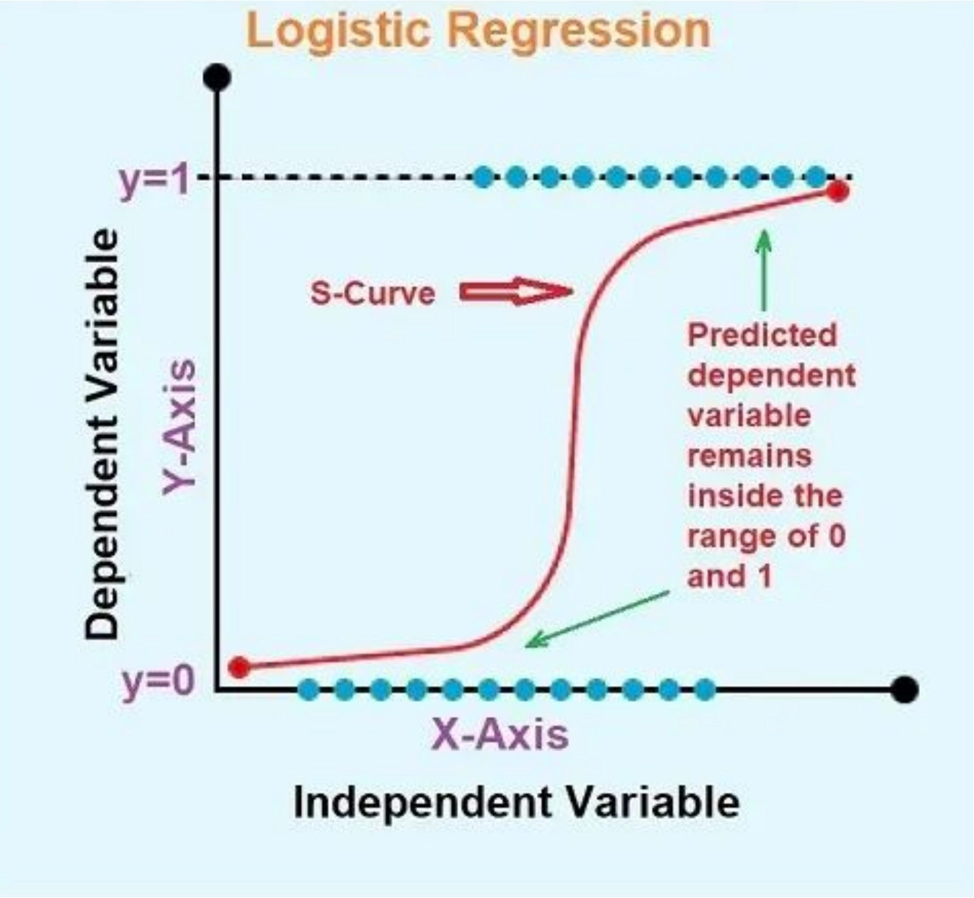
A linear regression model is a statistical and machine learning technique that uses a straight-line equation to describe the relationship between a dependent variable and one or more independent variables.



Predictive Analytics

Logistic Regression

A logistic regression model is a statistical method used in machine learning and data analysis to predict the probability of a binary outcome (e.g., yes/no, pass/fail) based on one or more independent variables.



Key Difference Between Linear and Logistic Regression

Linear Regression

Used for regression problems, where the goal is to predict a continuous numerical outcome (e.g., house prices, temperature, sales figures)

Logistic Regression

Used for classification problems, where the goal is to predict a categorical outcome, most commonly binary outcomes (e.g., yes/no, spam/not spam, true/false).

Linear Regression in Marketing Analytics

Continuous Outcomes

It can be predicted when the dependent variable is a continuous numerical value, such as sales revenue, customer lifetime value, or marketing spend.

Linear Models

Forms a linear relationship between the independent variables (e.g., advertising budget, website traffic) and the dependent variable.

Applications

Forecasting sales, optimizing marketing budgets, predicting customer lifetime value, analyzing the impact of pricing strategies.

Logistic Regression in Marketing Analytics

Categorical Outcomes

Used when the dependent variable is a categorical outcome, often binary (e.g., customer churn/no churn, click/no click on an ad, purchase/no purchase).

Logistic Models

Predicts the probability of a specific category occurring, using a sigmoid function to transform the linear combination of independent variables into a probability between 0 and 1.

Applications

Predicting customer churn, identifying potential leads, assessing the likelihood of ad clicks, segmenting customers based on purchase probability.

“ In essence, if the marketing question involves predicting a numerical value, linear regression is appropriate. If the question involves predicting the likelihood of a specific event or category, logistic regression is the suitable choice.”

New Product Development Case Study

NutriBoost Pvt. Ltd., a mid-sized beverage company, is considering launching a new product: NutriBoost+, a plant-based, sugar-free energy drink aimed at health-conscious millennials and Gen Z (ages 18–35). The product is designed to provide a natural energy boost without artificial sugar, catering to the growing demand for healthy, functional beverages. The management team has commissioned a survey of 500 potential customers to test the idea. The data collected includes demographics, purchase intention, willingness to pay, product feature preferences, and open-ended feedback. Your role as a Marketing Analyst is to evaluate the product idea through the 8 stages of New Product Development (NPD) and determine whether the product should be launched.

Steps in New Product Development

1. Idea Generation
2. Idea Screening
3. Concept Development & Testing
4. Marketing Strategy
5. Business Analysis
6. Product Development
7. Test Marketing
8. Commercialization

If else & elif in Python

By Prof.Vishal Chugh

if else Conditions

`if condition:`:

The `if` keyword initiates the conditional statement. `condition` is an expression that evaluates to either `True` or `False`. If the `condition` is `True`, the indented block of code immediately following the `if` statement is executed.

`else:`:

The `else` keyword provides an alternative block of code to be executed if the `condition` in the `if` statement evaluates to `False`. The code within the `else` block is also indented.

if else Conditions

Example:

Python

```python
age = 20

if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

elif condition

Python

```python
score = 75

if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: F")
```

Explore the Data in Python

By Prof.Vishal Chugh

Import Libraries

```
# import libraries
from cryptography.fernet import Fernet
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import copy
import warnings
```

After Decrypting We Load the Dataset

```
# Load the dataset
df = pd.read_csv('nutriboost.csv')
df.head()
```

	CustomerID	Age	Gender	Income_Level	Health_Conscious_Score	Current_Beverage_Preference	Purchase_Intention	Willingness_To_Pay
0	1	46	Male	Medium	8	Energy Drinks	0	83
1	2	32	Male	High	6	NaN	1	129
2	3	25	Female	Low	5	Water	1	43
3	4	38	Other	Low	6	Water	0	129
4	5	36	Other	Low	2	Juices	0	143

See the column names

```
# Check the column names
df.columns
```

```
Index(['CustomerID', 'Age', 'Gender', 'Income_Level', 'Health_Conscious_Score',
       'Current_Beverage_Preference', 'Purchase_Intention',
       'Willingness_To_Pay', 'Feature_Preference_Taste',
       'Feature_Preference_EnergyBoost', 'Feature_Preference_Packaging',
       'Feature_Preference_Price', 'Feature_Preference_Availability',
       'Expected_Repeat_Purchase_Rate', 'Open_Ended_Feedback',
       'Preferred_Channel', 'Proximity_to_Store', 'Channel_Satisfaction',
       'Awareness_Source', 'Promo_Response', 'Ad_Recall_Score',
       'Coupon_Usage(%)', 'Brand_Engagement'],
      dtype='object')
```

See the number of rows and columns in data

```
# Check the shape of the data
df.shape
```

```
(500, 23)
```

Check for the missing values

```
# Check for the missing values
df.isna().sum()
```

	0
CustomerID	0
Age	0
Gender	0
Income_Level	0
Health_Conscious_Score	0
Current_Beverage_Preference	92

Create a Data Copy

```
# Create the data copy
df_copy = df.copy(deep = True)
```

```
# See the head of the data
df_copy.head()
```

	CustomerID	Age	Gender	Income_Level	Health_Conscious_Score	Current_Beverage_Preference	Purchase_Intention	Willingness_To_Pay
0	1	46	Male	Medium	8	Energy Drinks	0	83
1	2	32	Male	High	6	NaN	1	129
2	3	25	Female	Low	5	Water	1	43
3	4	38	Other	Low	6	Water	0	129
4	5	36	Other	Low	2	Juices	0	143

Unique names in Current_Beverage_Preference

```
# Unique names in Current_Beverage_Preference
df_copy['Current_Beverage_Preference'].unique()
```

```
array(['Energy Drinks', nan, 'Water', 'Juices', 'Soft Drinks'],
      dtype=object)
```

Replace Missing Values with mode

```
# Replace the missing values with mode
df_copy['Current_Beverage_Preference'] = df_copy['Current_Beverage_Preference'].fillna(df_copy['Current_Beverage_Preference'].mode()[0])
```

Idea Screening

Step 1: Idea Screening (1 indicates Customer is ready to purchase while 0 indicates customer is not ready to purchase)

```
[ ] # check the unique values purrchase_intention
    df_copy['Purchase_Intention'].unique()
```

```
array([0, 1])
```

Average Purchase Intention Rate

```
# average_purchase_intent_rate
average_purchase_intent_rate = df_copy['Purchase_Intention'].mean() * 100
round(average_purchase_intent_rate, 2)
```

```
np.float64(62.4)
```

Concept Testing Through Feature Preferences

```python
# Which feature in the product customer prioritize more
feature_means = df_copy[['Feature_Preference_Taste',
       'Feature_Preference_EnergyBoost', 'Feature_Preference_Packaging',
       'Feature_Preference_Price', 'Feature_Preference_Availability']].mean().rename("Feature Means")
round(feature_means,2)
```

	Feature Means
Feature_Preference_Taste	2.97
Feature_Preference_EnergyBoost	3.01
Feature_Preference_Packaging	3.12
Feature_Preference_Price	3.00
Feature_Preference_Availability	2.92

dtype: float64

Marketing Strategy -> Income Levels

```
# Group the customers based on Income Levels
income_levels = df_copy.groupby('Income_Level')['Purchase_Intention'].mean() * 100
round(income_levels,2)
```

	Purchase_Intention
Income_Level	
High	63.89
Low	61.15
Medium	62.55

dtype: float64

Business Analysis -> Willingness to Pay

```
# Willingness to pay
average_wtp = df_copy['Willingness_To_Pay'].mean()
round(average_wtp,2)
```

```
np.float64(96.96)
```

Business Analysis -> Logistic Regression

```
# x are independent and y is dependent variable
x = df_copy[['Age', 'Health_Conscious_Score', 'Feature_Preference_Taste',
        'Feature_Preference_EnergyBoost', 'Feature_Preference_Packaging',
        'Feature_Preference_Price', 'Feature_Preference_Availability']]
y = df_copy['Purchase_Intention']
```

Train and Test Data

```
# Train and test data
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 42)
```

Model Creation

```
# model
model = LogisticRegression(max_iter = 500)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

Accuracy Score

```
# accuracy_score
accuracy = accuracy_score(y_test, y_pred) * 100
print('Accuracy Score is:', accuracy, '%')
```

```
Accuracy Score is: 60.0 %
```

Classification Report

```
# classification report
classification = classification_report(y_test, y_pred)
print(classification)
```

```
              precision    recall  f1-score   support

           0       0.36      0.07      0.12        57
           1       0.62      0.92      0.74        93

    accuracy                           0.60       150
   macro avg       0.49      0.50      0.43       150
weighted avg       0.52      0.60      0.50       150
```

Classification Report - Precision

1. Precision

- Out of all the times the model **predicted positive (1)**, how many were actually correct?
- Formula:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

👉 High precision = few false positives.

In your case for class `1`, precision = **0.62** → 62% of the predicted positives were actually correct.

Classification Report - Recall

2. Recall (Sensitivity / True Positive Rate)

- Out of all the actual positives, how many did the model correctly identify?
- Formula:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

👉 High recall = few false negatives.

In your case for class `1`, recall = **0.92** → the model found 92% of all the actual positives.

Classification Report - F1 Score

3. F1-Score

- A balance between **Precision** and **Recall**.
- Harmonic mean:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

👉 Useful when you want a trade-off between precision and recall.

In your case for class `1`, F1 = **0.74**, which is decent since both precision and recall are fairly good.

Classification Report - Interpretation

🔎 **Interpreting your report:**

- **Class 0 (negative class):** Precision = 0.36, Recall = 0.07 → the model is very bad at detecting class `0`.
- **Class 1 (positive class):** Precision = 0.62, Recall = 0.92 → the model is good at catching positives, but allows some false positives.
- **Accuracy = 0.60 (60%)** → overall correct predictions.
- **Macro avg** → average across classes (treats each class equally).
- **Weighted avg** → average weighted by number of samples (gives more importance to class `1`, since it has more support).

⚖️ **In simple words:**

- Your model is much better at predicting **1s** than **0s**.

Test Marketing

```
# Market Testing -> repeat purchase
repeat_rate = df_copy['Expected_Repeat_Purchase_Rate'].mean() * 100
round(repeat_rate,2)
```

```
np.float64(54.3)
```

Commercialization

```
# Commercilization
if average_purchase_intent_rate > 50 and repeat_rate > 50 and average_wtp > 50:
    decision = 'Go Ahead and Launch'
else:
    decision = 'Do not launch'
print(decision)
```

```
Go Ahead and Launch
```