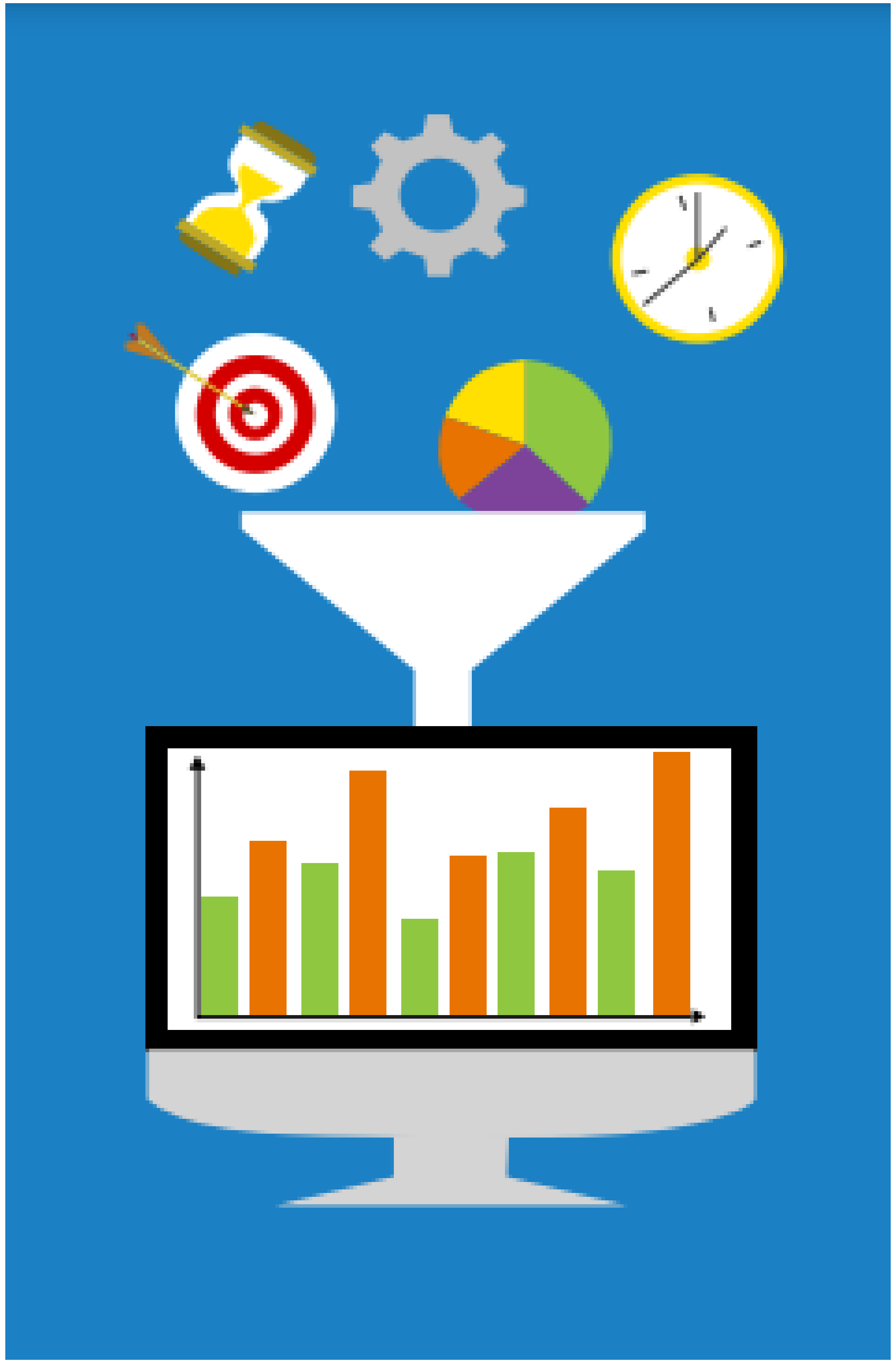
# VISTORA ASSIGNMENT

PRESENTED BY ASHUTOSH SAHOO



# Introduction to Feature Engineering

Feature engineering is the activity of developing, choosing, or manipulating input variables (features) to enhance the performance of a machine learning model. It increases accuracy, decreases complexity, and enables models to learn patterns more effectively, thus being important in constructing efficient and effective ML systems.

# Types of Feature Engineering

Normalization/Scaling – Adjusts numerical features to a common scale (e.g., Min-Max, Z-score) to prevent bias from varying magnitudes.

Binning – Groups continuous data into discrete intervals (e.g., age groups) to reduce noise and overfitting.

Encoding – Converts categorical data into numerical form (e.g., One-Hot, Label Encoding) for model compatibility.

Feature Interaction – Combines features (e.g., multiplying or summing) to reveal relationships that improve predictive power.

Time-Based Aggregations – Extracts trends from timestamps (e.g., daily averages, rolling stats) to capture temporal patterns

# Using Snowflake for Data Storage & Processing

How Snowflake is used for storing structured and semi-structured data.

Snowflake keeps structured data (tables, CSVs) in standard relational formats and semi-structured data (JSON, XML, Parquet) in VARIANT columns, which automatically parse nested fields. Schema-on-read enables querying semi-structured data through SQL without pre-transformations. Inbuilt functions such as FLATTEN extract nested elements, which facilitates effortless analytics over mixed data types without compromising performance and scalability.


Example of SQL queries that extract and preprocess data in Snowflake

```
-- 1. Extract & clean structured data (e.g., sales table)
SELECT
   ORDER_ID,
   CUSTOMER_ID,
   TO_DATE(ORDER_DATE, 'YYYY-MM-DD') AS
FORMATTED_DATE, -- Date normalization
   NULLIF(TOTAL_AMOUNT, 0) AS ADJUSTED_AMOUNT --
Handle zeros
FROM SALES
WHERE STATUS = 'COMPLETED';

-- 2. Parse semi-structured JSON logs (e.g., event data)
SELECT
   LOG_ID,
   EVENT_DATA:user_id::STRING AS USER_ID, -- Extract
JSON field
   EVENT_DATA:timestamp::TIMESTAMP AS EVENT_TIME,
   LOWER(EVENT_DATA:action::STRING) AS ACTION --
Normalize text
FROM EVENT_LOGS
WHERE EVENT_TIME > '2024-01-01';
```

## How Snowflake integrates with ML pipelines.

Snowflake naturally fits into ML pipelines by facilitating direct access to data through Snowpark (for Python/Scala) or through connectors such as Spark. In-database model training is supported through Snowflake ML, and external tools (e.g., TensorFlow, SageMaker) can query processed data through SQL. Efficient feature engineering is ensured through Snowflake's scalability, and its secure data sharing makes collaboration between teams easier, all the way from data preparation to deployment.

# Feature Store Concepts

A Feature Store is a centralized system used to store, manage, and serve machine learning features reliably throughout training and production. It promotes feature consistency, prevents repeated preprocessing, and provides real-time access for model inference. Maintaining a single source of truth for features makes it speed up ML development, lower errors, and enhance collaboration among data scientists and engineers, enabling model deployment to be faster and more dependable

## Compare different feature stores

AWS SageMaker Feature Store excels in real-time and batch feature serving, tightly integrated with AWS ML services. Snowflake Feature Store leverages Snowflake’s SQL engine for seamless feature management within its data cloud, ideal for structured/semi-structured data. Databricks Feature Store integrates with MLflow and Delta Lake, optimizing feature sharing across Spark workflows. While AWS focuses on end-to-end ML, Snowflake emphasizes SQL-native features, and Databricks unifies feature engineering with big data processing, catering to distinct ecosystem needs.

# Implementation And Practical Task

Connect to Snowflake, extract some sample data, and perform feature engineering.

```
USE DATABASE SNOWFLAKE_SAMPLE_DATA;
SHOW SCHEMAS;
USE DATABASE SNOWFLAKE_SAMPLE_DATA;
USE SCHEMA TPCH_SF1;
SHOW TABLES;
```

```
SELECT * FROM CUSTOMER LIMIT 10;
SELECT * FROM ORDERS LIMIT 1000;
SELECT * FROM LINEITEM LIMIT 10;
```

| | O_ORDERKEY | O_CUSTKEY | O_ORDERSTATUS | O_TOTALPRICE | O_ORDERDATE | O_ORDERPRIORITY | O_CLERK |
|---|---|---|---|---|---|---|---|
| 1 | 3000001 | 145618 | F | 30175.88 | 1992-12-17 | 4-NOT SPECIFIED | Clerk#0000 |
| 2 | 3000002 | 1481 | O | 297999.63 | 1995-07-28 | 1-URGENT | Clerk#0000 |
| 3 | 3000003 | 127432 | O | 345438.38 | 1997-11-04 | 5-LOW | Clerk#0000 |
| 4 | 3000004 | 47423 | O | 135965.53 | 1996-06-13 | 4-NOT SPECIFIED | Clerk#0000 |
| 5 | 3000005 | 84973 | F | 209937.09 | 1992-09-12 | 5-LOW | Clerk#0000 |
| 6 | 3000006 | 135136 | O | 140186.32 | 1996-09-26 | 1-URGENT | Clerk#0000 |
| 7 | 3000007 | 78841 | F | 298655.07 | 1992-04-13 | 5-LOW | Clerk#0000 |
| 8 | 3000032 | 124576 | F | 175973.90 | 1992-03-02 | 1-URGENT | Clerk#0000 |
| 9 | 3000033 | 30247 | F | 4635.38 | 1993-11-10 | 1-URGENT | Clerk#0000 |
| 10 | 3000034 | 5498 | F | 348308.79 | 1992-04-21 | 1-URGENT | Clerk#0000 |

## The Normalizing step of Feature Engineering

```
-- Normalize O_TOTALPRICE Column

WITH stats AS (
  SELECT
    MIN(O_TOTALPRICE) AS min_price,
    MAX(O_TOTALPRICE) AS max_price
  FROM ORDERS
)
SELECT
  O_ORDERKEY,
  O_TOTALPRICE,
  (O_TOTALPRICE - stats.min_price) / (stats.max_price - stats.min_price) AS totalprice_normalized
FROM ORDERS, stats
LIMIT 10;
```

| | O_ORDERKEY | O_TOTALPRICE | TOTALPRICE_NORMALIZED |
|---|---|---|---|
| 1 | 5400001 | 270576.60 | 0.48648 |
| 2 | 5400002 | 216696.22 | 0.38929 |
| 3 | 5400003 | 191044.99 | 0.34303 |
| 4 | 5400004 | 263505.65 | 0.47372 |
| 5 | 5400005 | 117459.27 | 0.21030 |
| 6 | 5400006 | 84588.56 | 0.15102 |
| 7 | 5400007 | 50890.64 | 0.09024 |
| 8 | 5400032 | 24285.54 | 0.04225 |
| 9 | 5400033 | 181139.61 | 0.32516 |
| 10 | 5400034 | 54996.61 | 0.09764 |

## Store the features in a Feature Store

```
CREATE DATABASE IF NOT EXISTS FEATURE_DB;
USE DATABASE FEATURE_DB;
CREATE SCHEMA IF NOT EXISTS FEATURE_SCHEMA;
USE SCHEMA FEATURE_SCHEMA;

CREATE OR REPLACE TABLE ORDER_FEATURES AS
WITH stats AS (
  SELECT MIN(O_TOTALPRICE) AS min_price, MAX(O_TOTALPRICE) AS max_price
  FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS
),
cust_first_order AS (
  SELECT O_CUSTKEY, MIN(O_ORDERDATE) AS first_order_date
  FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS
  GROUP BY O_CUSTKEY
)
SELECT
  O.O_ORDERKEY,
  O.O_CUSTKEY,
  O.O_ORDERDATE,
```

_

```
-- Date Features
  EXTRACT(YEAR FROM O.O_ORDERDATE) AS order_year,
  EXTRACT(MONTH FROM O.O_ORDERDATE) AS order_month,
  EXTRACT(DAY FROM O.O_ORDERDATE) AS order_day,
  DAYOFWEEK(O.O_ORDERDATE) AS order_dayofweek,
  WEEKOFYEAR(O.O_ORDERDATE) AS order_weekofyear,

  -- Encoded ORDERSTATUS
  CASE WHEN O.O_ORDERSTATUS = 'F' THEN 1 ELSE 0 END AS is_filled,
  CASE WHEN O.O_ORDERSTATUS = 'O' THEN 1 ELSE 0 END AS is_open,
  CASE WHEN O.O_ORDERSTATUS = 'P' THEN 1 ELSE 0 END AS is_pending,

  -- Encoded ORDERPRIORITY
  CASE WHEN O.O_ORDERPRIORITY = '1-URGENT' THEN 1 ELSE 0 END AS is_urgent,
  CASE WHEN O.O_ORDERPRIORITY = '2-HIGH' THEN 1 ELSE 0 END AS is_high,
  CASE WHEN O.O_ORDERPRIORITY = '3-MEDIUM' THEN 1 ELSE 0 END AS is_medium,
  CASE WHEN O.O_ORDERPRIORITY = '4-NOT SPECIFIED' THEN 1 ELSE 0 END AS is_not_specified,

  -- Normalized Price
  O.O_TOTALPRICE,
  (O.O_TOTALPRICE - stats.min_price) / NULLIF((stats.max_price - stats.min_price), 0) AS totalprice_normalized,

  -- Days since first order
  DATEDIFF('day', cust_first_order.first_order_date, O.O_ORDERDATE) AS days_since_first_order

FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS O
JOIN cust_first_order ON O.O_CUSTKEY = cust_first_order.O_CUSTKEY,
stats;
```

```sql
SELECT * FROM FEATURE_DB.FEATURE_SCHEMA.ORDER_FEATURES LIMIT 10;

DESC TABLE FEATURE_DB.FEATURE_SCHEMA.ORDER_FEATURES;
```

| | O_ORDERKEY | O_CUSTKEY | O_ORDERDATE | ORDER_YEAR | ORDER_MONTH | ORDER_DAY | ORDER_DAYOFWEEI |
|---|---|---|---|---|---|---|---|
| 1 | 2400001 | 21445 | 1995-04-03 | 1995 | 4 | 3 | |
| 2 | 2400002 | 104548 | 1994-08-25 | 1994 | 8 | 25 | |
| 3 | 2400003 | 8512 | 1996-07-09 | 1996 | 7 | 9 | |
| 4 | 2400004 | 85286 | 1996-09-30 | 1996 | 9 | 30 | |
| 5 | 2400005 | 147364 | 1994-05-03 | 1994 | 5 | 3 | |
| 6 | 2400006 | 95339 | 1995-11-14 | 1995 | 11 | 14 | |
| 7 | 2400007 | 46249 | 1993-07-10 | 1993 | 7 | 10 | |
| 8 | 2400032 | 130357 | 1994-08-16 | 1994 | 8 | 16 | |
| 9 | 2400033 | 138706 | 1995-02-01 | 1995 | 2 | 1 | |
| 10 | 2400034 | 79036 | 1994-11-09 | 1994 | 11 | 9 | |

## Retrieve features and link to the Python Notebook to use Features

```
import snowflake.connector
import pandas as pd

# Establish connection
conn = snowflake.connector.connect(
    user='AshutoshSahoo1234',
    password='AshutoshSahoo2025',
    account='IZDSRLY-GR41231',
    warehouse='COMPUTE_WH',
    database='FEATURE_DB',
    schema='FEATURE_SCHEMA'
)

try:
    # Query the feature table
    query = "SELECT * FROM ORDER_FEATURES"
    cursor = conn.cursor()
    cursor.execute(query)

    # Load into a DataFrame
    df = pd.DataFrame(cursor.fetchall(), columns=[col[0] for col in cursor.description])

finally:
    # Always close the cursor and connection
    cursor.close()
    conn.close()

# Now df contains your features and can be used for training
print(df.head())
```

```
   O_ORDERKEY  O_CUSTKEY O_ORDERDATE  ORDER_YEAR  ORDER_MONTH  ORDER_DAY  \
0           1      36901  1996-01-02        1996            1          2
1           2      78002  1996-12-01        1996           12          1
2           3     123314  1993-10-14        1993           10         14
3           4     136777  1995-10-11        1995           10         11
4           5      44485  1994-07-30        1994            7         30

   ORDER_DAYOFWEEK  ORDER_WEEKOFYEAR  IS_FILLED  IS_OPEN  IS_PENDING  \
0                2                 1          0        1           0
1                0                48          0        1           0
2                4                41          1        0           0
3                3                41          0        1           0
4                6                30          1        0           0

   IS_URGENT  IS_HIGH  IS_MEDIUM  IS_NOT_SPECIFIED O_TOTALPRICE  \
0          0        0          0                 0    173665.47
1          1        0          0                 0     46929.18
2          0        0          0                 0    193846.25
3          0        0          0                 0     32151.78
4          0        0          0                 0    144659.20

  TOTALPRICE_NORMALIZED  DAYS_SINCE_FIRST_ORDER
0            0.31168688                    1380
```

# Thank you