BIG DATA ANALYTICS - ARCHITECTURE AND LIFE CYCLE

Data Analytics Lifecycle Phase

Phase 1 : Discovery - The data science team explores issues & investigatesit. It builds context and understanding. learns about required & available data sources. Builds initial hypothesis that can be tested later with data.

Phase 2 : Data Preparation - Methods to discover, preprocess & condition.. data before modeling & analysis. An analytical sandbox is required. Team performs ETL to get data into sandbox. Tools used are : Alpine Miner, Hadoop, etc.
ie Ingestion - ETL (structured data) & ELT (unstructured data).
Cleaning - Handle missing values, duplicates & outliers.

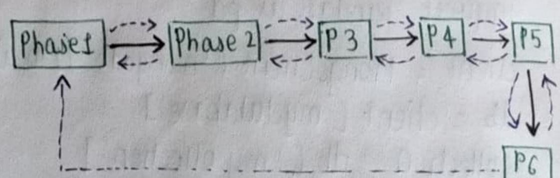
Phase 3 : Model Planning - Explore & select variables, choose algorithms. The data science team studies the data to identify connections. Then it selects variables, models, etc.

Phase 4 : Model Building - Team builds dataset for training, testing & production. It assesses whether existing tools are adequate for running the models. Eg. WEKA, Octave, R and PL/R, etc.

Phase 5 : Communication Results - Team must evaluate model's outcomes to establish success or failure criteria. It should identify key findings, measure business value and build a narrative to summarize & communicate the findings

Phase 6 : Operationalization - Team conveys project's benefits more broadly. Eables developers to gain insights. Team provides final reports, codes & briefings

Eg : Manufacturing company wanting to save costs by improving vendor contracts.



Types of Analysis

1) Descriptive analytics - answers to the questions about events that have already occured. ie Summarizes historical trends
2) Diggnostic analytics - derive reasoning behind events ie identify root causes.
3) Predictive analytics - Forecast and predict future events (eg. demand prediction)
4) Prescriptive analytics - adds human judgement to advise further actions ie.. Recommended actions (eg. strategies)

Eg : Managing vendor costs in Manufacturing Company.
1) Spend 50 Lakhs on vendors last year. 2) Costs were high because some vendors charged more for same service in different locations. 4) Switch to another Vendor to save 20% of transportation costs. 3) If we keep using same vendor, cost may rise by 10% next year.

Analytical Approach - Statistical Analysis, Data Mining, ML, Text Analysis Graph Analysis, etc.

DATA INGESTION FROM DIFFERENT SOURCES

Data ingestion - process of collecting, importing, and loading data from various sources into a centralized system.

1) Read from CSV file

```
import pandas as pd
data = pd.read_csv ("file.csv")
data.
```

2) Read from Excel file

```
import pandas as pd
data = pd.read_excel ("file.xlsx")
data
```

3) Read from JSON file

```
import pandas as pd
data = pd.read_json ("file.json")
```

4) Read from HTML file

```
import pandas as pd
tables = pd.read_html (" link of web")
df = tables [0]  # use first table
```

5) Read from SQlite db.

```
import pandas as pd
import sqlite 3
conn = sqlite3.connect ('dbname.db')
data = pd.read_sql (query, conn)
print(data.head ())
conn.close ()
```

6) Read from MongoDB

```
from pymongo import MongoClient
import pandas as pd
client = MongoClient ('mongodb://localhost:27017/')
db = client ['mydatabase']
collection = db ['mycollection']
# Converting MongoDB docs to DataFrame
df = pd.DataFrame (list (collection.find ()))
print (df.head ())
```

~~conn. close ()~~

7) Read from MySQL db.

```
import pandas as pd
import mysql.connector
conn = mysql.connector.connect (
host = 'localhost',
user = 'username',
password = 'password',
database = 'mydatabase'
)
query = "select * from mytable"
df = pd.read_sql (query, conn)
print (df.head ())
conn. close ()
```

DATA CLEANING

Identifying & correcting errors, inconsistencies and inaccuracies in datasets to ensure they are ready for analysis.

Deals with → Empty cells, Data in wrong format, Wrong data, Duplicates, Missing values

Empty cells/Missing Values

```
import pandas as pd
import numpy as np

df.isnull()   // Checks all missing values

df.isnull().sum()  // returns count per column

df[df.isnull().any(axis=1)]   // find rows with any missing values
df[df.isnull().all(axis=1)]   // find rows with all missing values
```

Duplicates

```
df.duplicated()   // checks for duplicates
df[df.duplicated()]   // shows only duplicated rows
df.drop_duplicates(inplace = True)  // remove duplicate rows.
```

Standardize data formats (eg: round floats)

```
df = df.round(2)
```

HANDLING MISSING VALUES

Techiques to handle missing values:

- Deleting a row
- Deleting a column
- fill with median value
- Fill with mean value
- fill with majority value ie mode
- fill with 0
- Fill with particular value

```
import pandas as pd
df = pd.DataFrame({
    'A': [1, None, 3],
    'B': [4, 5, None]
    })
print(df.isnull())
```

```
df_drop-rows = df.dropna()

df-drop_cols = df.dropna(axis=1)

df["A"] = df["A"].fillna(df["A"].mean())

df["B"] = df["B"].fillna(df["B"].median())

df["A"] = df["A"].fillna(df["A"].mode())

df_fill-zero = df.fillna(0)

df_fill-value = df.fillna(29)
```

Interpolation (smart estimation)

```
df_interpolation = df.interpolate(method='linear')
```

DATA IMPUTATION

Process of replacing missing values in a dataset with substituted values

1) Mean imputation → fillna (df. mean())
2) Median imputation → fillna (df. median())
3) Majority / mode imputation → fillna (df. mode())
4) Constant value imputation, etc.

DATA TRANSFORMATION

- Refers modifying, cleaning or reconstructing data to prepare it for analysis or modeling. It turns raw data → clean, usable data.
- Why needed: improve performance, to make all values same, reduce skewness, handle outliers, etc.

- Types of Data Transformation

1) Normalization (Min-Max Scaling) –
scales data to a range (0 to 1)

formula :
$$\bar{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

min(x), max(x) are the maximum & minimum values over entire dataset.

eg.
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled-data = scaler.fit-transform ([[1], [2], [3]])
```

for 1: $\frac{(1-1)}{(4-1)} = 0$ for 2: $\frac{(2-1)}{(4-1)} = 0.333..$ for 3: $\frac{(3-1)}{(4-1)} = 0.666..$

2) Standardization (z-score scaling) –
Converts data to have mean = 0 and standard deviation = 1

Formula:
$$z = \frac{x - \mu}{\sigma}$$
$$\sigma = \sqrt{\frac{\Sigma(x-\mu)^2}{n-1}}$$

where μ → mean
σ → standard deviation
x → value to be normalised.

eg.
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled-data = scaler.fit_transform ([[1], [2], [3]])
```

3) Log Transformation –
Used to reduce skewness in data. Only works with +ve values.

eg.
```
import numpy as np
log-transformed = np.log ([1, 10, 100])
```

4) Binning (Discretization) - Groups continuous values into categories / Bins.

eg Age → Bin
0-18 → child
19-35 → young
36 + → Adult

5) Encoding Categorical Data - Converts text categories into numbers

Categorical Data are of 2 types

i) Nominal - Eg. Color: Red, Green - Encoding Type: One Hot Encoding
ii) Ordinal - Eg. Size: Small, Medium, Large - Encoding Type: Label encoding with order.

Types of Encoding Techniques:

1) Label Encoding - Assigns a unique integer to each category.

eg.

Color	Encoded
Red →	0
Blue →	1
Green →	2

eg.
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Color-encoded'] = le.fit_transform(df['Color'])
```

Used when categories have ordinal relationship (like low < medium < high)

2) One-hot Encoding - Creates new binary column for each category (1 if present, 0 if not)

eg.

Color	Red	Blue	Green
Red	1	0	0
Blue	0	1	0

Used when categories are nominal (no order)

eg.
```
import pandas as pd
df = pd.DataFrame({'Color': ['Red', 'Blue', 'Green']})
df_encoded = pd.get_dummies(df['Color'])
```
converts categorical value into binary

3) Ordinal Encoding - You manually assign ordered numbers to categories

eg.

Size	Encoded
Small →	0
Medium →	1
Large →	2

eg.
```
df['Size'] = df['Size'].map({'Small': 1,
    'Medium': 2, 'Large': 3})
```

Used when there is a logical order.

Categorical Data - variables that represent categories / groups rather than numeric values.

Eg. Gender (male, female), Eye color (blue, brown, black), etc.

Need of Categorical Data in Encoding -

1) Model compatibility - ML models like SVM, Linear Regression; only understands numbers. So we need conversion of Categories into numbers

2) Pattern Recognition - Allows models to recognize patterns & relationships by encoding the categories.

3) Bias Prevention - Proper encoding ensures all categories are treated fairly, avoiding unintended bias.
4) Feature Engineering - Encoding is crucial step for effective features in modeling
5) Handling High Cardinality - Advanced encoding techniques help manage features with many unique categories.

DATA STANDARDIZATION

done earlier

Converts into standard format : mean = 0 & standard deviation = 1

HANDLING CATEGORICAL DATA WITH 2 AND MORE CATEGORIES

1) One Hot encoding (done) more than 2
2) Label encoding (done) only 2 - to avoid ranking like 0,1,2,3 creates confusion
3) Dummy encoding - similar to one hot but it drops one column to avoid confusion

eg.
```
dummy = pd.get_dummies(df, drop_first = True)
print(dummy)
```

Output : Red → [0, 0]
Blue → [1, 0]
Green → [0, 1]

We can calculate Red, if we know Blue & Green

∴ If feature has 'n' categories, dummy encoding creates (n-1) columns

---

What is dataset ? (PYQ) → collection of data - like a big table - used for analyzing, training models or learning patterns.
Types : structured, unstructured & semi-structured.

STATISTICAL AND GRAPHICAL ANALYSIS METHODS

1) Mode - value of maximum frequency.

$$\text{Mode} = L + \frac{\Delta_1}{\Delta_1 + \Delta_2} h$$

L → lower limit.
Δ_1 → excess of modal frequency over frequency of preceding class.
Δ_2 → excess of modal frequency over following class
h → size of modal class.

eg. x = 1, 2, 3, 2, 4

mode = 2

2) Mean - average of data

$$\text{Mean} = \frac{\Sigma x}{n}$$

eg. x = 1, 2, 3

$$\text{Mean} = \frac{1+2+3}{3} = 2$$

3) median - measure of central tendency that identifies middle value in sorted data

Eg. x = 10, 15, 25, 20

$$\text{median} = \frac{n}{2} \text{ and } \frac{n}{2}+1 \text{ th term}$$

= 15 and 25

x = 10, 15, 25

$$\text{Median} = \frac{n+1}{2} \text{ th term}$$

= 15

4) Variance (σ^2) - how variable is dataset.

Standard deviation $= \sqrt{\sigma^2} = \sqrt{\frac{\Sigma (x-\mu)^2}{n-1}}$

5) Correlation Coefficient

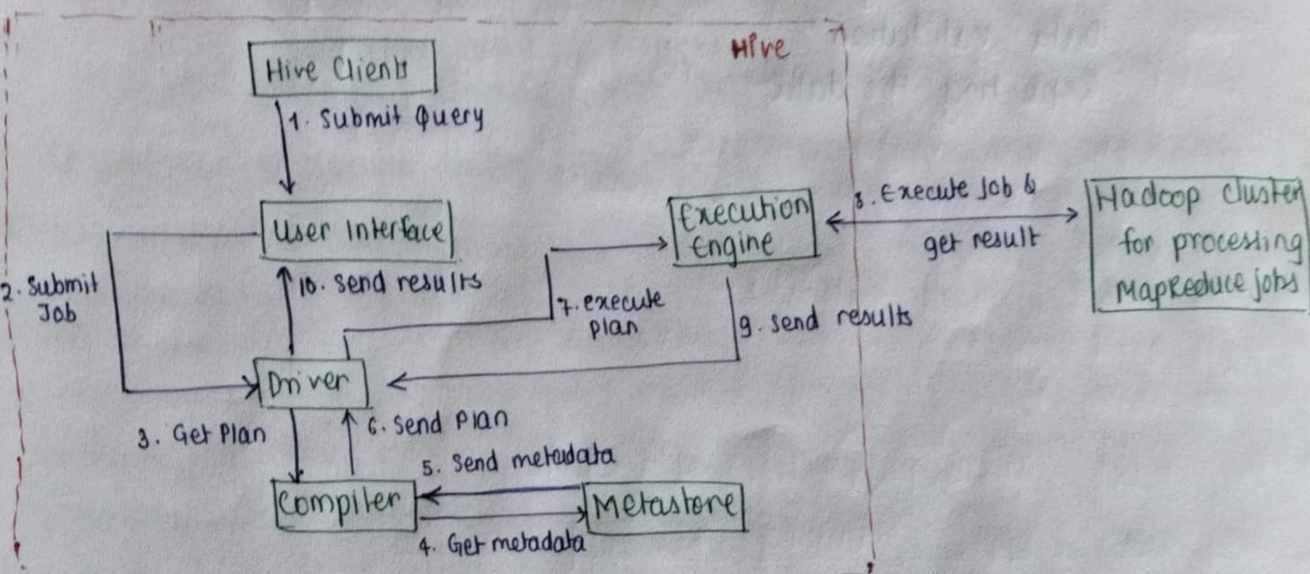
$$r = \frac{\Sigma (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma (x_i - \bar{x})^2 \Sigma (y_i - \bar{y})^2}}$$

-1 → negative relation
0 → no relation.
1 → positive relation

HIVE DATA ANALYTICS

Hive is a data warehouse software built on top of Hadoop.
It helps analyze large datasets stored in Hadoop HDFS)
Hive provides a SQL like query language called HiveQL.

Architecture of Hive



User Interface - write queries in HiveQL using JDBC, ODBC, CLI, Web UI, etc.
Compiler - converts HIVEQL to execution plans.
Execution engine - runs plan on Hadoop cluster
Metastore - Stores metadata info about all.
Storage / Driver - Actual data stored in Hadoop HDFS

HBase vs Hive

Feature	Hive	HBase
Type	Data Warehouse	NoSQL Database.
Data model	Table based (like SQL).	Key-value stored.
Query language	HiveQL	Java API, shell commands
Processing	Batch	Real-time
Schema	fixed	flexible
Speed	Slow	fast

Feature	Hive	HBase
Data format	structured	Un or semi structured
Use case	Analytics Reports	fast lookups, real time data
Support for Joins	Yes	No
Best for	Big Data Analytics & Reporting	Real time Read / write operation

DATA WRAGLING

Process of cleaning, structuring, transforming raw data into a desired format that is more suitable for analysis, reporting or machine learning.

Need: Improves data quality
Enables Analysis
Reduce risk of misleading results
Integrates multiple sources
Supports Decision making

Methods: Data Collection
Data Cleaning
Data Transformation
Data Normalisation
Data Encoding
Data validation
Exporting the data