introduction

DATA SCIENCE

Interdisciplinary field combining stats, maths, computer science and domain expertise.

Goal :- Collect, process, analyse & interpret data and enable decision making & predictions.

Key Stages of DS process

1. Problem definition → understand business goals, convert to analytical problem
2. Data acquisition → Databases, APIs, IOT sensors, web scraping, open databases
3. Data cleaning & preprocessing → Handle missing data, duplicates & outliers
4. Exploratory data analysis → Stats & visualisations
5. Model selection & training → Regression, classification, clustering, neural networks
6. Model evaluation → Accuracy, precision, recall, RMSE
7. Deployment → Integrate models into apps/dashboards
8. Monitoring & maintenance → Retrain/update to avoid degradation

Core skills in DS

- Programming → Python, R, SQL
- Maths & Stats → Probability, hypothesis testing, linear algebra, calculus
- ML techniques → Supervised, unsupervised, deep learning

- Big data tools → Hadoop , Spark
- Visualisation → Tableau , PowerBI , Matplotlib , Seaborn
- Domain knowledge → Industry specific knowledge.

Tools and libraries

- Data handling : Pandas , Numpy
- ML | AI : Scikit-Learn , Tensorflow , Keras , Pytorch
- Visualisation : Matplotlib , Seaborn , Plotly , ggplot2.
- Data enginering : Apache Kafka , Airflow
- Databases : MySQL, PostgreSQL , MongoDB

Challenges in DS

- Data quality (missing values , noisy , biased)
- Model interpretability (black box issue)
- Privacy and security
- Scalability with huge datasets
- Ethical AI usage.

Future trends

- AutoML (Automated ML)
- XAI (Explainable AI)
- Integrated with LLMs
- Edge AI for IoT
- Real time analytics

Artificial Intelligence	Machine Learning	Data Science
Broad concept	Subset of AI	Interdisciplinary field
Focus on mimicking human intelligence	Focus on learning from data	Focus on extracting insights from data
Reasoning, robotics, NLP	Algorithms , Models	Stats , visualisation, ML
Eg:- Self driving cars	Eg:- Recommendation systems	Eg:- Dashboards

PYTHON

High level, interpreted, general purpose programming language.

Created by Guido van Rossum in 1991

Known for simplicity, readability and versatility

Popular among both beginners and professionals

Key features

- Easy to learn & read
- Interpreted language
- Cross platform
- Extensive libraries
- Multi-Paradigm support
- Open source

Applications

- Web dev → Django, flask
- DS & AI → Pandas, Numpy, pytorch
- Automation → File handling
- Game dev → Pygame
- Desktop Apps → Tkinter, PyQt

GOOGLE COLAB

Free cloud based jupyter notebook environment by google

Lets users write & run python code in the browser

Popular for ML, data analytics and education

Provides free GPU & TPU resources for faster computations

Key features

- Free cloud based environment
- Python and jupyter notebook support
- Free GPU & TPU access
- Integration with google drives & github
- Pre installed libraries
- Easy collaborations
- Code + O/P + Visualisations together
- Custom modules & data upload

DATA PREPROCESSING

Process of cleaning, transforming and preparing raw data into a suitable format for analysis or ML.

Steps

1. Data collection

Gather data from sources: Databases, APIs, sensors, web scrapping etc.

2. Data cleaning

- Handling missing values → Remove, fill or predict
- Remove duplicates → Eliminate repeated records
- Handling outliers → Detect / treat unusual values

3. Data transformations

- Scaling / Normalisation → Bring features to same scale
- Standardisation → Data with mean = 0, variance = 1
- Encoding categorical data → convert categories to nums

4. Data integration

Merge data from multiple sources into a single dataset

5. Data reduction

- Feature selection → Keep only most important features
- Dimensionality reduction → PCA, LDA, Autoencoders

6. Data Splitting

Divide dataset into :-

- Training set → used to train model

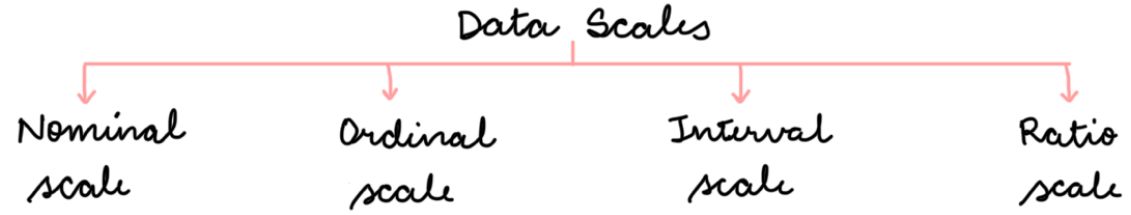
- Validation set → used to tune hyperparameters
- Test set → used to evaluate final performance

Importance

- Improve model accuracy & performance
- Ensures consistency & reliability of results
- Removes noise, bias and irrelevant info

DATA SCALES

In stats & DS, data scales define how data values are categorised, ordered and interpreted



Nominal Scale

Labels or names, no order or ranking

Operations possibles → Counting, mode etc

Eg:- • Gender → Male / Female
• Colors → Red / Blue
• Blood group → A, B, O

Ordinal Scale

Categories with meaningful order, but differences are not measurable.

Operations possible → Ranking, median

Eg:- • Education level → Primary, secondary, graduate
• Customer satisfaction → Poor, Good, Excellent

Interval Scale

Numeric scale with equal intervals, but no absolute

zero

Operations : Addition , substraction , mean , standard deviation

Eg :- Temperature , calendar years

Ratio Scale

Numeric scale with equal intervals + meaningful zero

Operations :- Addition , substraction , multiplication , division

Eg :-
- Height , weight
- Age , salary
- Distance , salary

SIMILARITY MEASURES

Numerical measure of how alike two data objects are

Range → 0 and 1

0 → No similarity

1 → Perfectly similar

Use case : Clustering , recommendation systems , text / document matching

Common Similarity Measures

Cosine Similarity

Measures the cosine of the angle between two vectors

Best for text and high dimensional data.

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Jaccard Similarity

Measure the overlap between two sets

Useful for categorical / binary data.

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|}$$

DISSIMILARITY MEASURE

A numerical measure of how different two data objects are.

Range → 0 and ∞

0 → Perfectly identical

∞ → More dissimilar

Use cases: Clustering, anomaly detection, distance based classification

Common measures

Euclidean Distance

Straight line distance between two points in n-dimensional space.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Manhattan Distance

Distance measured along axes at right angles.

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

SAMPLING AND QUANTIZATION

Sampling

converting a continuous signal into discrete form by taking measurements at fixed variables

Sampling rate = No. of samples per unit time (Hz)

Higher rate = More accurate representation

```
import pandas as pd
import numpy as np

# Create a simple dataset
data = pd.DataFrame({
    'id': range(1, 11),
    'score': np.random.randint(50, 100, 10)
})

print("Original Dataset:")
print(data)

# Random Sampling - select 5 random rows
sampled_data = data.sample(n=5, random_state=42)

print("\nRandom Sample (5 rows):")
print(sampled_data)
```

Quantization

Mapping sampled values to a finite set of discrete levels.

Quantization Error = Actual value - Quantized value

More levels = better accuracy, less error

```
import numpy as np
import pandas as pd
```

```
import pandas as pd

# Create a simple dataset of continuous values
data = pd.DataFrame({
    'value': np.random.uniform(0, 100, 10)  # random float
values between 0 and 100
})

print("Original Data:")
print(data)

# Quantization - divide continuous data into discrete bins
data['quantized'] = pd.cut(data['value'], bins=[0, 25, 50, 75,
100],
                           labels=['Low', 'Medium', 'High', 'Very High'])

print("\nQuantized Data:")
print(data)
```

Trade off :- Higher sampling rate & quantisation levels = better quality but more storage

FILTERING

Process of selecting elements from an iterable (like list, tuple or set) that satisfy a given condition

Syntax:-

filter (function , iterable)

- function → A function that returns true or false
- iterable → Any sequence

Returns a filter object

```
import pandas as pd
```

```
data = pd.DataFrame({
    'name': ['Aman', 'Riya', 'Karan', 'Simran', 'Nikhil'],
    'score': [85, 42, 76, 90, 55]
})

print("Original Data:")
print(data)

filtered_data = data[data['score'] > 70]

print("\nFiltered Data (score > 70):")
print(filtered_data)
```

	name	score
0	Aman	85
2	Karan	76
3	Simran	90

filter() is memory efficient because it returns an iterator (lazy evaluation)

For data processing, filter() is often used with map() and reduce()

DATA TRANSFORMATION AND MERGING

Pandas provide powerful tools for modifying, cleaning and combining datasets.

Data transformation

Refers to modifying data to make it suitable for analysis, reporting or modeling. It may involve restructuring, cleaning or deriving new information

Filtering → Selects rows based on conditions

Aggregation → Summarise data using stats (sum, mean, count)
↳ .groupby()

Creating new columns → Adding new derived features

Handling missing values → Fixing gaps using filling or removal
↳ .fillna()
↳ .dropna()

Data type conversion → Changing col types

Renaming columns → For clarity or consistency
↳ .rename()

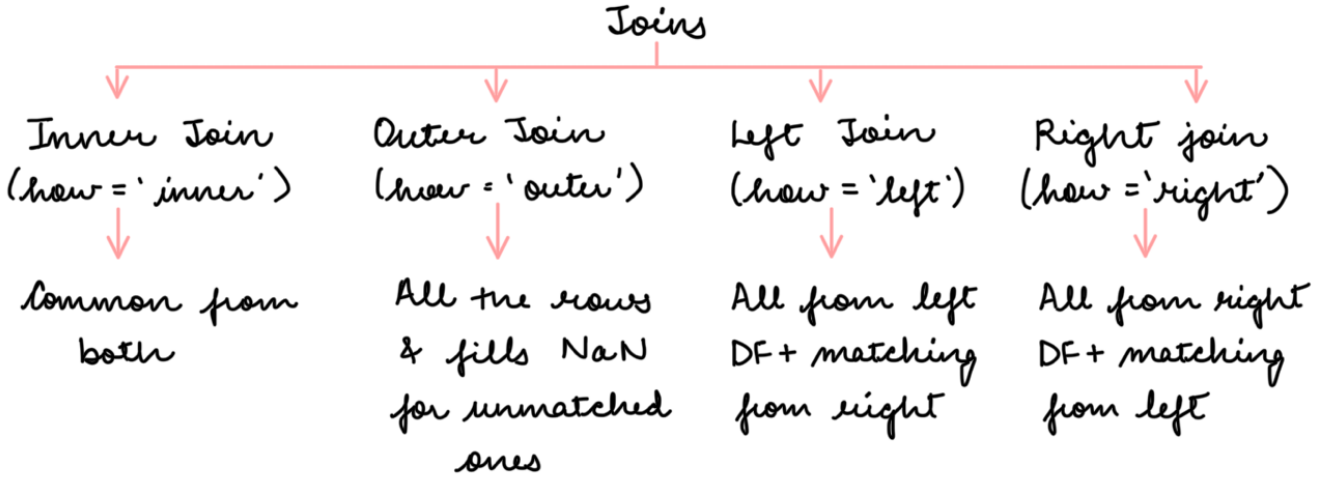
Removing duplicates → Eliminates repeated rows
↳ .drop_duplicates()

Data Merging

Involves combining two or more dataframes using shared columns or indices. It is similar to SQL joins.

Syntax:-

```
pd.merge(df1, df2, on='col_name', how='join_type')
```



Data transformation → clean, format & enhance data
Merging → combines data

DATA VISUALISATION

Process of representing data in visual or graphical formats (charts, graphs, maps, dashboards) to identify patterns, trends and insights effectively

Python libraries → Matplotlib, Seaborn, Plotly, ggplot2

- Matplotlib → Basic plotting, customizable
- Seaborn → Statistical plots, attractive aesthetics
- Plotly → Interactive charts, dashboards

Importance

- Makes complex data easier to understand.
- Helps identify patterns, trends and outliers
- Improves communication of data insights
- Supports decision-making in business & research

Types

Charts & Graphs

Bar chart → compare quantities across categories

Line chart → Show trends over time

Pie chart → Show proportions or percentages.

Statistical Plots

Histogram → Distribution of single variable

Box plot → Spread, median, outliers

Scatter plot → Relationship b/w two variables

Advanced Visualisations

Heatmap → Color coded representation of values

Tree map → Hierarchical data visualisation

Network graph → Relationships between entities

PRINCIPAL COMPONENT ANALYSIS (PCA)

Dimensionality reduction technique used in ML and data analysis. It transforms high dimensional data into fewer dimensions while preserving as much variance possible.

PCA is used when :-
- The data has many correlated features
- Visualisation is needed in 2D or 3D
- To reduce noise and overfitting
- To speed up ML models

Key concepts

Dimensionality reduction

Reducing no. of features while keeping important info

Variance

Measure of how much the data spreads

High variance = imp info

Principal components (PCs)

New transformed features that are linear combinations of original features

Orthogonal components

PCs don't correlate with each other

How PCA works (Steps)

1. Standardise the data
   - Since PCA is affected by scale
   - Eg:- StandardScaler() in sklearn

2. Compute covariance matrix

Shows relation between features

3. Calculate eigenvalues and eigenvectors

- eigenvalues → Level of importance (variance)
- eigenvectors → Directions of new axes (principal components)

4. Sort eigenvalues (descending)
   Keep the top components with highest variance

5. Transform the data
   Convert data into new reduced dimensions

Example

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Sample dataset (3 features)
data = pd.DataFrame({
    'feature1': [2, 4, 5, 6, 8],
    'feature2': [3, 6, 7, 8, 10],
    'feature3': [1, 2, 2, 3, 3]
})

print("Original Data:")
print(data)

# 1. Standardize the data (very important for PCA)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# 2. Apply PCA → reduce to 2 principal components
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_data)
```

```
print("\nPCA Result (2 Components):")
print(pca_result)

# 3. Explained variance
print("\nExplained Variance Ratio:")
print(pca.explained_variance_ratio_)
```

PCA result :-

```
[[ 2.73929374  -0.05927745]
 [ 0.64339049   0.13101   ]
 [ 0.10212423  -0.24766793]
 [-1.20113797   0.46664563]
 [-2.28367049  -0.29071025]]
```

Explained variance ratio :-

```
[0.97239961  0.02561916]
```

Advantages

- Reduces dimensionality
- Improves model speed and efficiency
- Removes multicollinearity
- Helps in visualisation

Limitations

- Loses original feature meaning
- Works only with numeric data
- Assumes linear relationships
- Sensitive to scaling and outliers

CORRELATION

Measures the strength and direction of the relationship between two variables

Positive	Zero	Negative
• Both variance increase or decrease together • Eg:- Height & weight	• No relationship between the variable • Eg:- Shoe size and exam marks	• One variable increases while the other decreases • Eg:- Speed of car & travel time

Correlation coefficient

Range :- -1 to +1

+1 → perfect +ve correlation
-1 → perfect -ve correlation
0 → no correlation

Pearson correlation coefficient formula

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

Other methods to measure correlation

Pearson's correlation : Measures linear relationship (continuous data)

Spearman's rank correlation : Measures monotonic relationship (ranked data)

Kendall's Tau : Measures ordinal association

`df.corr()`
↓
Gives pearson's correlation

CHI-SQUARE TEST

Checks whether there is a significant association between two categorical variables by comparing observed frequencies with expected frequencies under the assumption of no association.

of no association

Chi - Square Test

Test of independence

Determines if two categorical variables are related

Eg:- Gender vs preference of product

Goodness of fit test

Checks if an observed frequency distribution matches an expected distribution

Eg:- Rolling a dice to see if all outcomes are equally likely.

Formula

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where O_i = observed frequency

E_i = Expected frequency

$$\downarrow$$

$$\frac{\text{Row Total} \times \text{Col Total}}{\text{Grand total}}$$