DS UNIT 1

◴ Created	@May 13, 2025 12:17 PM

1. What is Data Science?

- **Definition:** Data Science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data.
- **Key Components:** Data analysis, Machine Learning, Big Data, Data Visualization, Data Wrangling, and Communication.
- **Applications:** E-commerce, Finance, Healthcare, Marketing, Social Media, etc.

Benefits and Uses of Data Science

1. **High Demand:** Increasing opportunities across industries.
2. **Career Opportunities:** Many job roles such as Data Analyst, Data Scientist, etc.
3. **Attractive Salaries** due to skill shortage.
4. **Versatility:** Applicable in various domains (healthcare, retail, sports, etc.).
5. **Improves Data Quality:** Helps clean and structure data for decisions.
6. **Prestige:** Highly valued specialists.
7. **Automation:** Reduces repetitive/boring tasks.
8. **Smart Products:** Powers AI-driven features (e.g., recommendation engines).
9. **Saving Lives:** Predictive modeling in healthcare, disaster prediction, etc.

---

2. Data Types & Facets of Data

Types of Data:

- **Structured Data:** Organized in rows/columns (databases, spreadsheets).

- **Unstructured Data:** Lacks clear structure (text, media files).
- **Natural Language:** Human languages, challenging to analyze (requires NLP).
- **Machine-generated Data:** Produced automatically (logs, sensor data).
- **Graph-based/Network Data:** Models relationships (social networks, graph databases).
- **Audio, Video, Image Data:** Multimedia data, needs special processing.
- **Streaming Data:** Real-time, continuous flow (IoT, transactions, social media).

3. Big Data vs Data Science

Big Data	Data Science
Focuses on extremely large/complex datasets	Extracts insights and knowledge from data
Covers Volume, Variety, Velocity	Uses analytical methods, statistics, ML
Often unmanageable by traditional tools	Utilizes Big Data as input

- **Big Data**: The raw material (like crude oil).
- **Data Science**: The refinery that processes raw data into useful knowledge.

Characteristics of Big Data

- **Volume:** Huge daily data volumes (terabytes+, billions of rows).
- **Variety:** Diverse formats and sources.
- **Velocity:** Rapid generation and processing.

4. The Data Science Process

Steps in Data Science Process

1. **Setting the Research Goal**
    - Define objective, deliverables, feasibility, timeline.
2. **Retrieving Data**

- Sources: Text files, databases, data lakes, web APIs, web scraping.
- Data lakes for all data types, data warehouses for structured data.

3. **Cleansing, Integration, Transformation**
    - Remove inconsistencies, outliers, errors, missing values.
    - Sanity checks (e.g., 0 <= age <= 120).
    - Data types: join/stack tables, create views.
    - Enrich and transform data (dummy variables, feature engineering).
    - Reduce dimensionality where feasible.
4. **Data Exploration**
    - Visualization (bar charts, histograms, scatter plots, overlays, brushing/linking).
    - Basic descriptive stats (mean, median, stddev, etc.).
5. **Model Building**
    - Select modeling technique (statistics, ML, etc.).
    - Feature selection/engineering, iterative model development.
    - Evaluate if deployable, maintainable, explainable.
6. **Model Deployment and Communication**
    - Deploy to production (not detailed in syllabus, but next logical step).
    - Communicate findings.

---

5. Data Acquisition (Gathering Data)

- **Internal Company Sources:** Databases, data warehouses, lakes, marts.
- **External:** APIs, open datasets, web scraping.
- **Definitions:**
    - *Data Warehouse*: Central repository for structured data, complex ETL.
    - *Data Mart*: Subset, simpler, focused on business department.

- *Data Lake*: Stores all types (structured/unstructured), highly scalable.**Comparisons:**

- DWH: Complex, all departments, large, slow to build.
- Data Mart: Simple, one department, small, faster.
- Data Lake vs DWH: Data lakes better for unstructured data, DWH for structured.

6. Data Cleansing, Integration & Transformation

Data Cleansing

- **Purpose:** Eliminate errors and inconsistencies; produce true/consistent data.
- **Common Issues:** Outliers, typos, impossible/suspicious values, missing data, extra spaces/capitalization errors.
- **Sanity Checks:** Ensure data falls within realistic bounds.

Integration

- **Joining:** Combine records by shared keys (e.g., user_ID).
- **Appending/Stacking:** Add rows from one data source to another.
- **Views:** Virtual tables to avoid duplication.

Data Transformation

- Shape data for modeling (e.g., dummy variables, normalization, scaling).
- Reduce variables via feature selection/dimensionality reduction.

7. Introduction to NumPy

What is NumPy?

- **NumPy (Numerical Python):** High-performance array processing library for Python.

- Provides the ndarray object for multidimensional arrays, and functions for manipulation.
- Supports advanced scientific computing (linear algebra, FFT, etc.).

Creating Arrays

```
import numpy as np

# 1D Array
a = np.array([1, 2, 3])
# Using arange and reshape
b = np.arange(20).reshape(4, 5)
# Zeros/ones
c = np.zeros((2, 4))
d = np.ones((3, 6))
e = np.full((2,2), 3)
# Identity matrix
f = np.eye(3,3)
g = np.identity(4)
```

- **Other Functions:** empty_like, full_like, asarray, diag, frombuffer, fromfile, mat, vander, triu, tril, tri, diagflat, fromfunction, logspace, meshgrid.

Array Attributes and Operations

- **ndim:** Number of dimensions
- **shape:** Shape of the array
- **itemsize:** Size of each element in bytes
- **dtype:** Data type
- **reshape():** Change shape

Indexing and Slicing

```
arr = np.array([1,2,5,6,7])
print(arr[2:5]) # [5 6 7]
```

```
arr2d = np.array([[1,2,3],[4,5,6]])
print(arr2d[1, 1]) # 5
```

Copying Arrays

- **np.copy():** Makes a copy
- **Assignment (=):** Just copies reference
- **np.empty_like():** Empty array, same shape/type

Iterating

```
for x in arr:
    print(x)
for x in np.nditer(arr2d):
    print(x)
```

8. Introduction to Pandas

What is Pandas?

- **Pandas:** Data analysis/manipulation library providing Series (1D) and DataFrame (2D) structures.
- Widely used in data science for cleaning, exploration, transformation.

Series

- **Definition:** 1D labeled array.
- Creatable from lists, arrays, dicts, etc.

```
import pandas as pd
s = pd.Series([1, 2, 3], index=['a','b','c'])
print(s['b'])  # 2
```

- **Vectorized Operations:** `s + s`

- **Automatic Alignment:** When indexes overlap.
- **NaN Handling:** Missing data is represented as NaN.

DataFrame

- **Definition:** 2D tabular data, like a spreadsheet.

```
data = {'Name': ['John', 'Alice', 'Bob'], 'Age': [25, 30, 35]}
df = pd.DataFrame(data)
# Adding column
df['Salary'] = [50000, 60000, 70000]
```

Indexing

- `df['column']` : Select column
- `df.loc[0]` : Select row by label (index)
- `df.iloc[0]` : Select row by integer position
- `df.at[0, 'Name']` : Access single value

Column Operations

- Arithmetic, filtering, sorting (see below)
- Adding/Removing columns
- Handling NaN (missing values):
    - `df.dropna()` : Remove missing
    - `df.fillna(0)` : Replace missing with 0

Grouping and Aggregation

```
grouped = df.groupby('City')['Age'].mean()
```

9. Index Objects and Operations in Pandas

Index

- Unique labels/identifiers for Series/DataFrame rows and columns
- Types: integer, string, datetime, multi-index

Creating Index

```
df = pd.DataFrame(data, index=['A', 'B', 'C'])
```

Reindex

- Creates a new DataFrame with a modified index/columns.

```
new_index = ['A', 'B', 'D']
df_reindexed = df.reindex(new_index)
```

Drop Entry

- Removes specific rows/columns.

```
df.drop("column_name", axis='columns')
```

Selecting Entries

- By position: `df.iloc[1]`
- By condition: `df[df['Age'] > 30]`

Data Alignment

- Aligns objects by index, not position; fills missing with NaN.

```
df1, df2 = df1.align(df2, fill_value=np.nan)
```

10. Rank, Sort & Summary Statistics

Ranking

- Assigns ranks based on column's value.

```
df['rank'] = df['col'].rank()
```

Sorting

- Sorts values or index.

```
df.sort_values(by='column')
df.sort_values(by='Population', ascending=False)
```

Summary Statistics

- Use built-in methods:
    - `df.describe()`
    - `df.mean(), df.median(), df.std(), df.min(), df.max()`

11. Index Hierarchy (MultiIndex)

- Use for grouping by multiple fields/levels.

```
import pandas as pd
index = pd.MultiIndex.from_tuples([('A', 1), ('A', 2), ('B', 1)])
df = pd.DataFrame({'value': [10, 20, 30]}, index=index)
```

12. Data Acquisition Methods

- **APIs:** Access data from web services (REST, SOAP, etc.)
- **Open Data Sources:** Public datasets (government, Kaggle)
- **Web Scraping:** Extracting data from websites (BeautifulSoup, Scrapy)

Summary Table—Key numpy & pandas Functions

Purpose	NumPy Function	Pandas Function
Create array	np.array(), np.arange(), etc.	pd.Series(), pd.DataFrame()
Shape	.shape, .reshape()	.shape, .T
Indexing/slicing	arr[i], arr[i:j]	.loc[], .iloc[]
Copy	np.copy(), =	.copy()
Iteration	for, np.nditer()	.iterrows(), .apply()
Identity matrix	np.identity(), np.eye()	N/A
Sorting	np.sort(), etc.	.sort_values(), .sort_index()
Aggregation/Stats	np.mean(), np.sum(), etc.	.mean(), .sum()
Handle missing	N/A	.dropna(), .fillna()
Group & Aggregate	N/A	.groupby(), .agg()

# End of Unit 1 Notes		
These notes cover all topics mentioned in your syllabus, in a clear and systematic manner. For problem-solving or practical sessions, refer to the examples/commands given in each section.		