

Féidearthachtaí as Cuimse  
Infinite Possibilities

# Programming for Analytics

## Lecture 8: Data Cleaning and Preprocessing

Bojan Božić

School of Computer Science  
TU Dublin, Grangegorman

[bojan.bozic@tudublin.ie](mailto:bojan.bozic@tudublin.ie)



# Overview

- Why Clean Data?
- Missing Data
- Duplicates
- Standardisation and Conversion
- Outliers and Inconsistency



# Why Clean Data?

- Real-world data is messy
- Missing values, typos, inconsistent formats
- Clean data -> better models and insights

# Types of Data Issues

- Missing values
- Incorrect data types
- Duplicates
- Inconsistent formatting (dates, case, etc.)
- Outliers and noise

# Activity: Spot the Problems

Recruit_ID	Name	Age	Faction	Height_cm	Weight_kg	Training_Score	Has_Magic_Potential
1	quick ben	34	Bridgeburners	180	75	92	Yes
2	Kalam Mekhar	Thirty	Bridgeburners	178	81	88	y
3	Dujek Onearm	55	malazan army	182	eighty	77	No
4	Coltaine	48	7th army	184	83	n/a	yes
5	tavore paran	41	bone hunters	165	61	59	no
6	tayschrenn		High mage corps	176	72	99	TRUE
7	Ganoes Paran	28	Host of the Crippled god	179cm	78.5	87	False
8	Adjunct Lorn	37	Claw	173	sixty eight		Y
9	Quick Ben	34	Bridgeburners	180	75	92	Yes

# Checking for Missing Data

- `df.isna().sum()` to count missing values per column
- `df.isnull()` also works the same
- Can also use `.info()` to spot missing types

# Activity: Count Missing Values

The Malazan army's supply officers have been tracking provisions for several legions during a campaign in Seven Cities. Unfortunately, some entries were lost during a chain of demon-related mishaps. You can find the data in `malazan_supply_log.csv`

Use `isna().sum()` to **identify which columns have missing data** and later decide how to handle it.

# Handling Missing Data

- `dropna()` to remove rows
- `fillna(value)` to replace with mean/median/constant
- Decide based on context

# Activity: Handle Nulls

- Fill all missing numerical values with the mean:
  - Rations\_Issued
  - Healing\_Potions
  - Days\_Supplied
  - Moral\_Score

# Standardising Text Data

- Use `.str.lower()`, `.str.strip()`,  
`.str.replace()`
- Fix casing, whitespace, common typos
- Important for categories like ‘Yes’/‘yes’/‘YES’  
or preparation for datatype conversion

# Activity: Clean Text Columns

- Standardise all strings so that everything is used in the same way.

# Data Types and Conversion

- Use `df.dtypes` or `df.info()` to check types
- Use `.astype()` to convert (e.g. `object -> int`)
- Important for calculations and modeling

# Activity: Convert to Int

- Convert the morale score from string to int

# Checking for Duplicates

- `df.duplicated()` marks duplicates
- `df.drop_duplicates()` removes them
- Duplicates can be rows or based on specific columns

# Activity: Drop Duplicates

- Remove duplicated rows to make sure none of the legions appears twice in the dataset.

# Parsing Dates

- `pd.to_datetime()` to convert strings to dates
- Allows sorting, filtering, extracting year/month
- Check for format mismatches

# Activity: Add Dates

Add a column to your dataset showing when the armies were established, based on the following data (all dates 600 years before Burn's Sleep) which needs to be converted to datetime:

1756-04-12 -> Bridgeburners, 1762-09-01 -> 7th Army,  
1749-06-23 -> Claw, 1767-03-10 -> Bonehunters,  
1735-11-15 -> Marines, 1760-07-30 -> Wickans,  
unknown -> T'lan Imass (ancient), 1750-01-02 -> 2nd Army

# Creating New Features

- From dates: extract year, month, day
- From strings: length, word count, etc.
- From numeric: bins or ratios

# Outlier Detection

- Use `.describe()` to spot extreme values
- Visualise with box plots
- Use IQR or Z-score for formal checks

# Activity: Find Outliers

- Implement a function to check for values over or under  $1.5 * \text{IQR}$  from Q1 and Q3.

# Scaling and Normalisation

- StandardScaler – mean=0, std=1
- MinMaxScaler – scale to 0-1 range
- Use `sklearn.preprocessing` or custom code

# Example

```
from sklearn.preprocessing import  
MinMaxScaler  
  
scaler =  
MinMaxScaler(feature_range=(0, 1))  
scaler.fit_transform(x)
```

# Activity: Min-Max Scale a Column

- Scale Morale\_Score to 0-1

# Encoding Categorical Variables

- Label Encoding – convert categories to numbers
- One-Hot Encoding – create binary columns
- Use pandas or sklearn for implementation

# Example

```
pd.get_dummies(df, columns=[...])
```

# Activity: Encode Categories

- One Hot encode the Mage\_Support variable (you can set it to “high” for both claw and t’lan imass).

# Detecting Inconsistent Data

- Check for typos or mismatches using  
  `.value_counts()`
- Use regex or `.apply()` for advanced fixes

# Creating a Cleaning Pipeline

1. Drop/Fill missing
  2. Convert types
  3. Clean text
  4. Scale/encode
  5. Save clean data
- 
- Use function chains or scripts

# Saving Cleaned Data

- `df.to_csv('clean.csv', index=False)`
- Preserve original vs processed version
- Useful for reproducibility and reuse

# Recap and Next Week

- You now know how to clean and preprocess real-world datasets
- Next week NumPy

# Questions?