Siddhartha Dev Shrestha

Overview of Today’s EDA Workflow

1. Setup & Imports

What we did: Imported libraries like pandas, numpy, matplotlib, seaborn, and later sweetviz.

Why:

pandas → data handling (loading CSV, cleaning, summarizing).

numpy → numerical operations (used for outlier detection).

matplotlib & seaborn → visualization of distributions, boxplots, correlations.

sweetviz → automated visual EDA report.

👉 This step lays the foundation for data manipulation and visualization.

---

2. Loading the Dataset

Code: Used pd.read_csv() with the Penguins dataset.

Why: Every EDA starts with bringing the dataset into memory.

Output: Checked shape (df.shape) and first few rows (df.head()) to see columns and structure.

👉 This confirmed we had **numeric + categorical features** and some missing values.

---

3. Structural Summary

Code: df.info(), df.describe() for numeric and categorical.

Theory:

info() → shows datatypes, memory usage, and missing values.

describe() → gives statistics (mean, std, quartiles) for numeric features.

For categoricals, it shows unique, top, and frequency.

👉 This helped us **understand data types** and spot potential issues.

---

4. Data Quality Checks

Code:

df.isna().sum() → missing values per column.

df.duplicated().sum() → duplicate rows.

df.dtypes → datatype inspection.

Theory: Data quality is key before analysis.

Missing values can bias results.

Duplicates may cause over-representation.

Wrong data types (e.g., categorical stored as string) must be fixed.

👉 Found and noted **missing values** (penguins dataset has some NaNs in body measurements).

---

5. Data Type Fixes

Code: Converted object-type columns to category.

Why:

Saves memory.

Explicitly marks categorical variables (useful for groupby, one-hot encoding later).

---

6. Univariate Distributions

Code: Histograms + boxplots for numeric features.

Theory:

Histogram → distribution shape (normal, skewed, multimodal).

Boxplot → spread, median, and outliers.

👉 Example: bill length/flip length distributions, with boxplots showing outliers.

---

7. Categorical Counts

Code: Bar plots of counts (value_counts + seaborn countplot).

Theory:

Helps visualize distribution of categories.

Checks class imbalance (important if dataset is for ML).

👉 Example: species count, island distribution.

---

8. Outlier Detection (IQR Method)

Code: Computed Q1, Q3, IQR → flagged values beyond 1.5×IQR as outliers.

Theory: Outliers may represent errors, rare cases, or important extremes.

Can distort mean, correlations, and models.

Decision later: drop, cap, or keep depending on domain.

👉 Example: some extreme body mass values flagged as outliers.

---

9. Bivariate Analysis (Numeric vs Numeric)

Code: Scatter plots of numeric pairs, optional seaborn pairplot.

Theory:

Identifies linear or non-linear relationships.

Shows collinearity (important for modeling).

👉 Example: bill length vs bill depth shows clear separation among species.

---

10. Categorical vs Numeric

Code: Boxplots of numeric variables split by categorical groups.

Theory:

Useful for comparing distributions across categories.

Highlights group differences and variance.

👉 Example: body mass across species/islands.

---

11. Correlation Analysis

Code: df.corr() + heatmap.

Theory:

Correlation (Pearson) measures linear relationship.

High correlation → redundancy or multicollinearity.

Low correlation → independent features.

👉 Example: bill length & bill depth had moderate negative correlation.

---

12. Grouped Insights

Code: df.groupby(cat)[num].agg([...])

Theory:

Aggregates statistics (mean, median, std) per category.

Useful for summarizing patterns across groups.

👉 Example: average body mass per species/island.

---

13. Basic Cleaning

Code:

Dropped duplicates.

Filled numeric NaNs with median, categorical with mode.

Theory:

Median = robust imputation (less affected by outliers).

Mode = most frequent category for categorical features.

👉 Dataset became **clean and consistent**.

---

14. Encoding Categoricals

Code: pd.get_dummies() → one-hot encoding.

Theory:

ML models can't use text directly.

Encoding turns categories into 0/1 binary columns.

drop_first=True avoids dummy variable trap.

---

15. Saving Cleaned Data

Code: df_clean.to_csv(), df_encoded.to_csv().

Why: To preserve cleaned versions for future modeling without repeating preprocessing.

---

16. Automated Profiling (Sweetviz instead of ydata-profiling)

Code:

```
import sweetviz as sv

report = sv.analyze(df)

report.show_html("eda_profile_report.html")
```

Theory:

Sweetviz auto-generates interactive reports with distributions, correlations, and comparisons.

Alternative to pandas-profiling / ydata-profiling.

Key Takeaways

EDA = Understand + Clean + Explore.

You practiced:

Data inspection (shape, info, describe).

Data quality checks (missing, duplicates, outliers).

Visualization (histograms, boxplots, scatter, heatmap).

Relationship analysis (numeric vs numeric, categorical vs numeric).

Cleaning (imputation, encoding).

Saving datasets for ML.

Automated EDA with **Sweetviz**.

Siddhartha Dev Shrestha