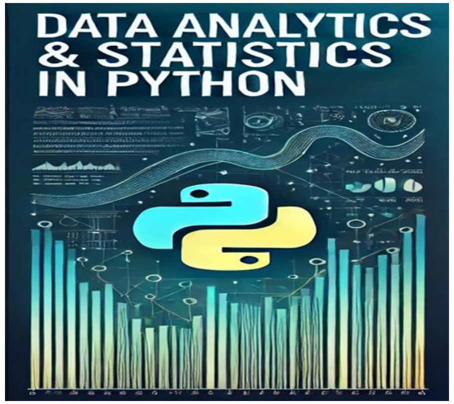
Data Analytics & Statistics in Python

Handling Missing Data: Methods and Best Practices



Learning data-driven decision-making with Python

- **Instructor:** Hamed Ahmadinia, Ph.D.
- **Email:** hamed.ahmadinia@metropolia.fi

Introduction to Missing Data

- **What is Missing Data?**
  - When a dataset has incomplete values in one or more columns.
  - Can be due to system errors, human input mistakes, or sensor failures.
- **Why is it a Problem?**
  - Leads to biased analysis.
  - Reduces model accuracy.
  - Can misrepresent trends.
- **Types of Missing Data:**
  - **MCAR**: Missing completely at random (no pattern).
  - **MAR**: Missing at random (depends on other variables).
  - **MNAR**: Missing not at random (systematic missingness).

Methods to Handle Missing Data

1. Deletion Methods
2. Basic Imputation (Mean/Median/Mode)
3. Advanced Imputation (KNN, MICE)
4. Time-Series Imputation
5. Model-Based & Machine Learning Approaches
6. Deep Learning Approaches

- Each method has its advantages and drawbacks, which we will explore.

Deletion Methods

- **Listwise Deletion**
  - Removes entire rows with missing values.
  - Simple, but leads to data loss.
- **Pairwise Deletion**
  - Uses available data without dropping entire rows.
  - Works for correlation or regression analysis.
- **Column Deletion**
  - Drops columns with excessive missing data (>30%).

```
df.dropna()  # Remove rows with missing values
df.dropna(subset=['col1', 'col2'])  # Keep some data
df.drop(columns=['col_with_too_many_missing'])  # Remove a column
```

Basic Imputation

- **Mean/Median/Mode Imputation**
  - **Mean**: Good for normally distributed data.
  - **Median**: Better for skewed data.
  - **Mode**: Used for categorical data.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="mean")  # Use "median" or "most_frequent" for other cases
df["column1"] = imputer.fit_transform(df[["column1"]])
```

Advanced Imputation

- **K-Nearest Neighbors (KNN) Imputation**
  - Uses similar data points to estimate missing values.
- **Multiple Imputation (MICE)**
  - Generates multiple estimates and averages them.

```
from sklearn.impute import KNNImputer
knn_imputer = KNNImputer(n_neighbors=5)  # Finds 5 similar data points
df_imputed = knn_imputer.fit_transform(df)
```

Time-Series Specific Methods

- **Forward Fill / Backward Fill**
  - Fills missing values using previous or next known value.
- **Interpolation**
  - Uses mathematical methods to estimate missing values.

```
df["column1"] = df["column1"].fillna(method="ffill")  # Forward Fill
df["column1"] = df["column1"].fillna(method="bfill")  # Backward Fill
df["column1"] = df["column1"].interpolate()  # Interpolation
```

Machine Learning for Imputation

- Uses predictive models to fill missing values.
- Works best when there are patterns in the missing data.

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
df_filled = rf.predict(X_missing)
```

Deep Learning & Advanced Methods

- **Neural Networks & Autoencoders**
  - Learns patterns in data to impute missing values.
- **Matrix Factorization & Expectation-Maximization**
  - Used in recommendation systems.

```
# Autoencoder model to fill missing values
from tensorflow import keras
autoencoder = keras.models.Sequential([...])
```

Summary

- **How to Choose the Right Method?**
  - **<5% Missing**: Deletion or Simple Imputation.
  - **5-30% Missing**: KNN, Regression-Based, or MICE.
  - **>30% Missing**: Consider Model-Based or Domain-Specific Approaches.
- **Key Takeaways:**
  - Always analyze missingness before deciding.
  - Compare different methods before selecting the best one.
  - Advanced methods improve accuracy but require more computation.