Course: DS5002
Data Science Tools and Techniques

Data Preprocessing

Dr. Safdar Ali

Explore and discuss the process of data cleaning, with understanding of its importance, common challenges, and effective techniques along with data transformation.

Data Visualization

- It helps in understanding complex data through visual ways.
- It can reveal trends, patterns, and outliers in your data.
- It makes data insights accessible to stakeholders who may not be familiar with technical analysis.

Visualizations types:

- Exploratory Visualizations: Used to explore the dataset and identify trends or patterns.
- Explanatory Visualizations: Designed to communicate the results of the analysis to a broader audience.

Overview of essential libraries Matplotlib, Seaborn, and Plotly

Best Practices

Choosing the Right Plot:

- Line Plots: Time series or continuous data.
- Bar Plots: Categorical data comparisons.
- Scatter Plots: Relationships between two continuous variables.
- Box Plots: Distribution and outliers.
- Heatmaps: Correlation matrices.

Aesthetic Design:

- Keep it simple and clean: Avoid cluttering with too many elements.
- Use appropriate colors: For example, avoid using too many colors, and consider colorblind accessibility.
- Label your axes and provide a clear title.

Color Usage:

- Ensure that colors you choose do not confuse interpretation.
- Use color to highlight important parts of your data, but not to overwhelm viewer.

Visualizing data in Python

Install **Matplotlib**, **Seaborn**, and **Plotly** using pip:

- pip install matplotlib seaborn plotly
- **Imports for Data Visualization:**

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import pandas as pd
import numpy as np
```

Data visualizations types

- **Basic Plots**

Line Plot

A **line plot** is typically used to represent time series data, showing trends over a continuous interval.

 # Line Plot using Matplotlib

```
x = np.linspace(0, 10, 100)

y = np.sin(x)

 plt.plot(x, y, label="Sine Wave", color='blue')

plt.title("Line Plot Example")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.legend()

plt.show()
```

Bar Plot

A **bar plot** is used to compare quantities across different categories.

 # Bar Plot using Seaborn

```
categories = ['A', 'B', 'C', 'D']

values = [10, 15, 7, 12]


sns.barplot(x=categories, y=values, palette='coolwarm')

plt.title("Bar Plot Example")

plt.show()
```

Histogram

A **histogram** is used to show the distribution of a dataset.

 # Histogram using Matplotlib

```
data = np.random.randn(1000)


plt.hist(data, bins=30, color='purple', alpha=0.7)

plt.title("Histogram Example")

plt.xlabel("Value")

plt.ylabel("Frequency")

plt.show()
```

Box Plot

A **box plot** is used to show the distribution of a dataset and detect outliers.

 # Box Plot using Seaborn

```
sns.boxplot(data=data)

plt.title("Box Plot Example")

plt.show()
```

Scatter Plot

A **scatter plot** is used to represent the relationship between two variables.

 # Scatter Plot using Seaborn

```
x = np.random.randn(100)

y = 2 * x + np.random.randn(100)

 sns.scatterplot(x=x, y=y)

plt.title("Scatter Plot Example")

plt.show()
```

Heatmap

A heatmap is useful for visualizing correlation between variables in a matrix format.

```
# Heatmap using Seaborn
data_matrix = np.random.rand(10, 12)
sns.heatmap(data_matrix, cmap="YlGnBu", annot=True)
plt.title("Heatmap Example")
plt.show()
```

Pie Chart

A pie chart is useful for showing proportions of a whole.

```
# Pie Chart using Matplotlib
sizes = [15, 30, 45, 10]
labels = ['A', 'B', 'C', 'D']
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart Example")
plt.show()
```

Pair Plot

A pair plot visualizes pairwise relationships between multiple variables.

```
# Pair Plot using Seaborn

iris = sns.load_dataset("iris")

sns.pairplot(iris, hue="species")

plt.title("Pair Plot Example")

plt.show()
```

Matplotlib

Basic Syntax and Plotting

 # Simple Line Plot using Matplotlib

```
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y, color='green', marker='o')
plt.title("Matplotlib Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid(True)
plt.show()
```

 # Customize Matplotlib Plot

```
plt.plot(x, y, label="Line", color="red", linestyle='--', marker='x')

plt.title("Customized Plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.legend(loc='best')

plt.grid(True)

plt.show()
```

Seaborn for Statistical Plots

Example: Boxplot

```
# Boxplot using Seaborn
sns.boxplot(data=iris,
x='species',
y='sepal_length')
plt.title("Boxplot Example")
plt.show()
```

Example: Violin Plot

A violin plot combines aspects of box plots and density plots.

```
# Violin Plot using Seaborn
sns.violinplot(x='species',
y='sepal_length', data=iris,
palette='muted')
plt.title("Violin Plot Example")
plt.show()
```

Plotly interactive plots can be embedded into web applications.

```
# Interactive Scatter Plot using Plotly
fig = px.scatter(iris, x="sepal_width", y="sepal_length",
color="species", title="Interactive Scatter Plot")
fig.show()
```

Practice

i. Create a line plot comparing two different time series datasets.
ii. Create a box plot and violin plot using Seaborn for the dataset of your choice such Iris dataset and compare the distributions.
iii. Create an interactive heatmap using Plotly to show correlations between variables in a dataset of your choice.

Integrated Development Environment (IDE)

- Designed to simplify software development process
- A comprehensive set of tools into a single interface to help programmers in:
  - **developing, managing, compiling, testing, deploying and debugging**
- Selection of right IDE depends on programming language and specific project requirements

Main Components of an IDE

Typically features of an IDE:

- **Source Code Editor** – Provides syntax highlighting, auto-completion, and code formatting.
- **Compiler/Interpreter** – Converts the code into machine-executable form.
- **Debugger** – Helps find and fix errors in the code.
- **Build Automation Tools** – Automates repetitive tasks like compiling, linking, and packaging.
- **Version Control Integration** – Supports Git, SVN, or other version control systems.
- **Terminal/Command Line Interface** – Allows running scripts and commands inside the IDE.
- **Project Management Tools** – Helps organize files, dependencies, and libraries.

Commonly used IDEs for DS

IDE and Useful for	Language and main features
Spyder (Scientific Computing, Data Cleaning, Statistical Analysis& Small Projects)	• Language: Python • Dataframe viewer for Pandas and Part of the Anaconda • Built-in debugging and profiling tools • Matlab-like interface (variable explorer, console, plots) • Installation: Comes with Anaconda or *pip install spyder*
PyCharm (Python development, Large-scale Data Science, AI, ML projects)	• Language: Python • Advanced code completion and debugging • Virtual environment and package management • Integrated Jupyter Notebook, GitHub, Docker, and database • Best for full-scale machine learning applications
Jupyter Notebook (Machine Learning, Data Analysis, Data Visualization) ***Best for beginners***	• Language: Python, R • Web-based interface and supports Markdows for documentation • Excellent for data visualization (Matplotlib, Seaborn, Plotly) • Integration to Python libraries like NumPy, Pandas, Scikit-learn • Easy to share notebooks via .ipynb format • Installation: Available via Anaconda or pip install jupyter
RStudio (Statistical Analysis, Data Visualization, R Progmming)	• Language: R, Python • Optimized for R-based data science workflows • Integrated support for R Markdown & Shiny Apps • SQL & Python support via Reticulate • Best use for R-based Data Science
Visual Studio Code (VS Code) Machine Learning, Deep Learning, Big Data	• Language: General Purpose , Python, R, SQL, Julia, etc. • Extensions for Python, Jupyter, R, SQL, Supports Git for version control and integrated terminal and debugging tools

Commonly used IDEs

IDE	Best Use for	Supported Languages
Spyder	Scientific computing	Python
PyCharm	Python development	Python
Jupyter Notebook	Data science & machine learning	Python, R
RStudio	Data analysis	R
Visual Studio Code (VS Code)	General Purpose & Extensions	Python, JavaScript, Java, C++, etc.
Eclipse	Java Development	Java, C++, Python
IntelliJ IDEA	Enterprise Software Development	Java, Kotlin, etc.
Android Studio	Mobile App Development	Java, Kotlin
Xcode	iOS/macOS Development	Swift, Objective-C