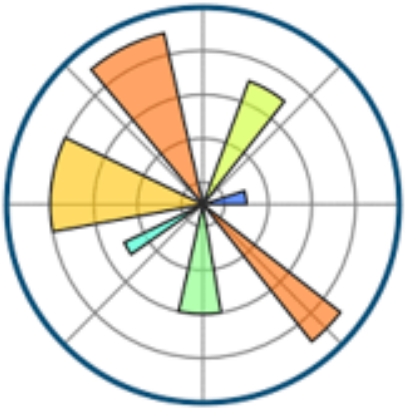
Data Transformation Analytics, Visualization Dashboards and Storytelling with Data in Python & Power BI



Trainer & Career Counsellor

Saiful Hoda, Exeter MBA, B. Tech. in Computer Science

Ex. Citibank, Refinitiv, Virtu Financials and Nomura Holding (Hong Kong)

Note to Learners

A solid foundation in core Python is essential for this program.

This is one of the most comprehensive courses designed to equip you with in-depth knowledge of data acquisition, cleaning, analysis, visualization and storytelling with data using standard Python libraries such as NumPy, Pandas, Matplotlib, and Seaborn. You'll also learn how to build web applications using Flask and create reports and dashboards in Power BI.

By the end of the program, you'll work on real-world projects where you'll apply your skills to solve practical problems. We will guide you on how to showcase your work effectively in your resume, apply for suitable job roles, and support you through the hiring process—right up to acing the interview.

All the best!

Table of Contents

1. Introduction to Python for Data Analysis

1.1. Why Python for Data Analysis?

Discuss Python's popularity and advantages in the data science ecosystem such as Versatility, vast ecosystem of libraries (NumPy, Pandas, Matplotlib, Seaborn), active community, integration with machine learning.

1.2. Setting Up the Environment

Guide through installing Python and setting up a data analysis environment.

1.3. Python Refresher (Data-Centric)

A quick review of Python basics, focusing on concepts relevant to data handling, Variables, data types (lists, tuples, dictionaries, sets), control flow (if/else, for loops), functions, basic file I/O.

2. Statistics Essentials for Data Analysis

Understanding numbers limited to a Data Analytics job profile

- Lightweight introduction to Probability
- Mean
- Median
- Variance
- Standard Deviation
- Percentiles
- Quantiles
- Quartiles
- Interquartile range (IQR)
- Mean Absolute Deviation (MAD)
- Histogram
- Probability Density and Cumulative Probability Density
- Normal Distribution and Bell Curves

3. NumPy for numerical computing

3.1. Introduction to NumPy Arrays (ndarray)

Understand NumPy as the fundamental package for numerical computation in Python, and how other libraries such as Pandas are built on top of Pandas.

- understanding a single dimensional and multi-dimensional array
- Creating a NumPy array
- check the dimensionality/datatype/ # of elements /
- Subsetting the Values of a Numpy array

- Copy vs View of a NumPy array
- Work with multi-dimensional arrays

3.2. Basic Attributes and Built in functions

Leverage built in functions to perform computation on NumPy arrays:

- np.array()
- np.zeros()
- np.ones()
- np.arange()
- np.linspace()
- shape
- dtype
- ndim

3.3. Indexing, Slicing, and Reshaping Arrays

- Integer indexing
- Boolean indexing
- array slicing ([start:end:step])
- reshape()
- flatten ()

3.4. Statistical functions on NumPy array

Leverage built in NumPy Functions for statistical data analysis –

- Sum
- Mean
- Variance
- Minimum
- Maximum
- cumsum
- cumprod.

3.5. Set functions in NumPy array

Apply Numpy built in Set functions

- Unique
- Intersection
- Union,
- Setdiff
- symmetric difference

3.6. Array Operations and Broadcasting

- Enable fast and efficient computations on entire datasets without explicit loops.

3.7. (Optional) Basic Linear Algebra with NumPy

Introduction to fundamental linear algebra operations.

- Dot product (np.dot())
- matrix multiplication (@ operator or np.matmul())
- transpose
- inverse (np.linalg.inv())

4. Data Analysis in Pandas

4.1. Pandas Series

- Introduce Series as a one-dimensional labeled array.
- Creating Series, indexing, slicing, Series operations.

4.2. Pandas Data Frame

Understand a DataFrame as a two-dimensional labeled data structure, like a spreadsheet or SQL table.

Creating DataFrames from dictionaries, lists, NumPy arrays. Further explore a Pandas dataset with built-in functions

- head()
- tail()

- info()
- describe()
- shape

4.3. Reading and Writing Data

Import and export data from various file formats.

- pd.read_csv()
- pd.read_excel()
- pd.read_sql()
- df.to_csv()
- df.to_excel()

4.4. Indexing and Selecting Data in DataFrames

Different ways to access specific rows, columns, or cells in a Data Frame.

- df['col'])
- row selection (df[start:end]),
- loc (label-based indexing)
- iloc (integer-based indexing).

4.5. Handling Missing Data (Data Cleaning & Preprocessing)

Identify and manage missing values (NaN)

- isnull(),
- notnull(),
- dropna() (dropping rows/columns),
- fillna() (imputing values like mean, median, mode, or a constant)

4.6. Dealing with Duplicate Data

Find and remove duplicate rows that can skew statistical measures and model training.

- duplicated()
- drop_duplicates().

4.7. Correcting Data Types

Typecasting

- astype(),
- pd.to_numeric(),
- pd.to_datetime().

4.8. String Operations for Text Data

Clean and manipulate text-based columns, standardization before analysis, common string methods

- lower()
- upper(),
- strip()
- replace()
- contains()).

4.9. Sorting Data (Data Transformation and Aggregation)

Arrange data in ascending or descending order, for quick inspection, identifying trends, or preparing data for visualization.

- sort_values()
- sort_index().

4.10. Filtering Data (Conditional Selection)

Select rows based on specific conditions, essential for focusing on specific subsets of data relevant to a question.

- Boolean indexing (df[df['col'] > value])
- combining conditions (&, |)

4.11. Grouping and Aggregating Data

Leverage powerful summary statistics and comparing groups within your dataset.

- groupby()
- aggregate functions (mean(), sum(), count(), min(), max(), agg()).

4.12. Pivoting Data (Pivot Tables)

Reshape data into a summary table

- pivot_table() (index, columns, values, aggfunc).

4.13. Combining Data Frames

Bring together data from multiple Data Frames, and integrate them for analysis.

- pd.concat() (stacking or appending DataFrames),
- pd.merge() (database-style joins: inner, outer, left, right), df.join().

4.14. Reshaping Data (Stacking/Unstacking, Melt)

Transform data from wide to long format and vice versa.

- stack(), unstack(), melt().

5. Time Series Analysis

5.1. Working with Datetime Objects in Pandas

- Import an external dataset
- convert a date to adatetime object
- Index Datetime columns
- extract individual components of a datetime object
- slicing and dicing based on datetime
- resampling data with datetime indexing
- calculate rolling values with datetime
- rolling correlation
- rolling covariance
- exponentially moving averages
- shift through datetime index

5.2. Time-Based Indexing and Selection

- Analyse trends, seasonality, and patterns over time.
- Slicing by date
- resample () (changing frequency)
- rolling () (moving window calculations)

6. Python Database Integration

- Environment Setup & Essentials
- MySQL set up
- Download key libraries
- Establishing MySQL Connection in Python
- Creating a Cursor Object
- Fetching Results:
- Leveraging Pandas Data Frames
- Fetched Data to Data Frame
- Basic Data Frame Operations:
- Writing Data to MySQL

7. (Optional) Feature Engineering and Data Preprocessing

7.1. Creating New Features

Derive new, more informative features from existing ones.

- Combining columns,
- extracting components from dates (e.g., month, day of week),
- creating interaction terms.

7.2. Encoding Categorical Variables

Convert text categories into numerical formats suitable for analysis.

- One-hot encoding (pd.get_dummies()), Label Encoding (conceptual).

7.3. Feature Scaling (Conceptual)

Standardizing or normalizing numerical features.

- Standardization (Z-score normalization)
- Min-Max scaling.

8. (Optional)Performance and Best Practices

8.1. Optimizing Pandas Operations

Efficient Pandas code to handle larger datasets and improved speed.

- Vectorization
- avoiding explicit loops - apply(), map()),using appropriate data types.

8.2. Memory Management

Techniques to reduce memory footprint.

Using category dtype for categorical data, downcasting numerical types.

8.3. Clean Code and Documentation

Write readable, maintainable, and well-documented analysis code.

- meaningful variable names,
- comments, docstrings,
- Markdown in Jupyter Notebooks.

9. Matplotlib

9.1. Matplotlib Basics: Figure, Axes, and Plots

Understand the core components of a Matplotlib plot.

- plt.figure(),
- plt.subplot(),
- ax.plot(),
- plt.show().

9.2. Basic Plot Types

Create common static visualizations.

- Line plots (plt.plot()),
- Scatter plots (plt.scatter())
- Bar plots (plt.bar()),
- Histograms (plt.hist())
- Box plots (plt.boxplot())

9.3. Customizing Plots

Enhance plots for clarity and aesthetics.

- plt.title()),
- labels (plt.xlabel(), plt.ylabel()),
- legends (plt.legend()), colors, markers, line styles, grid, figsize.

10. Advanced Visualization with Seaborn

10.1. Introduction to Seaborn

- Basic concepts such as enhanced aesthetics, simpler syntax for complex plots, integration with Pandas DataFrames.

10.2. Statistical Plot Types

Create specialized plots for statistical analysis. Visualize relationships, distributions, and comparisons within data more effectively.

Distribution plots

- Histplot
- Kdeplot
- displot

Relational plots

- Scatterplot
- Lineplot
- Relplot

Categorical plots

- Boxplot
- Violinplot
- Countplot
- Catplot
- swarm plots
- scatter plots

- **Correlation Heatmaps**

10.3. Multi-Plot Grids

Create grids of plots to visualize relationships across different subsets of data.

- FacetGrid
- PairGrid
- pairplot()

10.4. Customizing Seaborn Plots and Themes

Adjust the appearance of Seaborn plots, that allows for consistent and professional-looking visualizations.

- set_style()
- set_palette()
- adding titles and labels (like Matplotlib)
- adjusting figure size.

11. Plotly

Learn to make your plots interactive through Plotly, make the

- Import the libraries
- Define the go.scatter details

- Aesthetics - the title, and the axes
- Pass the data and layout to the Figure
- Plot the graph that include

    - Scatter plots
    - OHLCV plots
    - Bubble chart
    - Line Chart
    - Bar Chart
    - Box Plot
    - Distribution Plots

12. Flask

12.1. What is Flask?

- Understanding the "microframework" philosophy and its advantages.

12.2. Introduction to HTML and CSS

HTML and CSS essentials

12.3. Environment Setup:

- Python installation (brief reminder).
- Virtual Environments (venv): Importance for project isolation.
- Flask installation

12.4. Your First Flask Application

- Writing the minimal Flask app.
- Understanding @app.route decorator and basic view functions.
- Running the Flask development server.

12.5. Bringing the elements together

Routing & HTTP Methods

- Mapping URLs to Python functions (/, /about, /contact).
- URL variable rules: Capturing dynamic parts of the URL (/user/<username>).
- Understanding GET (retrieving data) and POST (submitting data).
- Specifying allowed methods for routes: methods=['GET', 'POST'].
- Returning simple strings.
- Returning JSON data using jsonify.
- Redirection: redirect(url_for(''))

Templating with Jinja2

- Separating Logic from Presentation: Why templates are essential.
- Rendering HTML Templates:
- Jinja2 Syntax Basics:
- Static Files:

Forms & User Input

- Accessing form data using request.form.
- Distinguishing GET vs. POST for form handling.

Introduction to WTForms (Flask-WTF):

- Simplifying form creation and validation.
- Defining form classes with fields and validators.
- Rendering forms in templates.
- Basic form validation (form.validate_on_submit()).

Database Integration (Basic with Flask-SQLAlchemy)

- Introduction to ORMs (Object-Relational Mappers

- Flask-SQLAlchemy Setup:
- Connecting to a simple database (e.g., SQLite for local development).
- Basic configuration.

Defining Database Models

- Creating Python classes that map to database tables.

Basic CRUD Operations:

- **C**reate: Adding new records.
- **R**ead: Querying data.
- **U**pdate: Modifying existing records.
- **D**elete: Removing records.

Application and Request Contexts:

- Understanding how Flask manages resources.

Project Structure & Deployment Considerations

- Typical Flask project layout (app.py, templates/, static/, models.py, etc.).
- Brief mention of **Blueprints** for larger applications.
- Using app.config for settings.
- Environment variables.

Debugging Flask Applications:

- Enabling debug mode (app.debug = True)

High-Level Deployment Concepts:

- Why you need a production-ready server (Gunicorn, Waitress).
- Web servers (Nginx, Apache) as reverse proxies.

12.6. Projects in Flask

- Personal Finance Mini-Dashboard:
- Data visualisation Dashboard in Flask that reads an excel file

13. Data Analysis and Visualization using Power BI

13.1. Getting Started

- Overview of Power BI
- Power BI versus Excel
- Power BI platform Services
- Understanding of Data and the Final Report
- Power BI desktop Installation
- Power Bi desktop interface and workflow

13.2. Data Acquisition and Transformation in Power BI

- Connect to and Import Data into Power BI
- Data Preparation
- Calculated columns, measures and Tables using DAX
- Data Modelling
- Summary

13.3. Visualize and Analyse Data using Power BI

- Overview
- Best Practices
- AI Visuals
- Bookmarks and Performance Analyser
- Summary

13.4. Publishing Reports and Scheduling Data Refresh

- Introduction
- Power BI Services and its interface
- Reports in Power BI Service
- Setting up Data Refresh and Row level Security
- Dashboards and Applications
- Summary and Recap

14. Effective Data Story telling

14.1. Overview

- What is data Story telling
- Importance of Data Story telling
- Components of a good story

14.2. Characteristics of a good story

- Narrative
- Importance of Objective and Agenda
- Five patterns of Insight

14.3. Components of a good story with data

- Structure and Flow
- Pyramid Principle
- Importance of Visualization
- Visualization of Qualitative data
- Visualization of Quantitative variables
- Visualization of advanced analytical techniques

14.4. Golden Rules for an effective data story telling

- Visual design principles and storyboarding

- Best Practices
- Case Study

15. Case Studies and Projects

15.1. Exploratory Data Analysis (EDA) of a Public Dataset

Perform a comprehensive EDA using NumPy, Pandas, Matplotlib and Seaborn on a datasheet from Kaggle

15.2. Data Cleaning and Transformation Pipeline

Transform a messy dataset and build a robust data cleaning and transformation script/notebook that handles missing values, duplicates, incorrect data types, and prepares it for analysis.

15.3. Time Series Data Analysis and Visualization

Analyze a time series dataset (e.g., stock prices, weather data) to identify trends, seasonality, and create appropriate time-based visualizations.

15.4. Leverage your programming skills to build a Dashboard

Create a Dashboard using Flask.