UGANDA CHRISTIAN UNIVERSITY
A Centre of Excellence in the Heart of Africa

Data Science Lifecycle DSC8201

Week 2: Lecture 04 (MSDS_1:1)

Topic: *Exploratory Data Analysis and Visualisation*

Dr. Daphne Nyachaki Bitalo
Department of Computing & Technology
Faculty of Engineering, Design & Technology

Breakdown of Course Outline

UGANDA CHRISTIAN UNIVERSITY
A Centre of Excellence in the Heart of Africa

Weeks	Course Goals	Focus Area	Topic Break Down	Practical Skills/ Tools
Week 2	Exploratory Data Analysis & Visualization	Analysis & Storytelling	Hypothesis generation and testing, descriptive statistics, data visualization principles, statistical inference (t-tests, ANOVA), communicating data stories.	Matplotlib, Seaborn, Plotly for visualization, statistical software (e.g., StatsModels, SciPy), Storytelling with data

Lecture Objectives and Learning outcomes

The Objectives of this lecture are to learn:

- How to turn data analyses into actionable insights

By the end of this lecture week, students should be able to:

- **Analyse data and turn it into actionable insights with appropriate visuals and statistical tests**

Lecture Overview

- Data Storytelling Tools
- Further Data Pre-processing
-
- Data Normalisation
- Label Encoding & Discretisation

Data Storytelling: PyNarrative

UGANDA CHRISTIAN UNIVERSITY
A Centre of Excellence in the Heart of Africa

- **PyNarrative** is a new library that tells data stories using charts and narratives that are interractive

PyNarrative: A Python Library for Data Storytelling

Angelica Lo Duca[1] [a] and Roberto Olinto Barsotti[2]
[1]*Institute of Informatics and Telematics of the National Research Council, via G. Moruzzi 1, Pisa, Italy*
[2]*University of Pisa, Pisa, Italy*

Keywords: Data Storytelling, Data Visualization, Python, Data Narrative.

Abstract: Data storytelling is an emerging approach combining data visualization with narrative techniques to enhance data insights' interpretability and emotional impact. Traditional Python libraries for data visualization, such as Matplotlib, Seaborn, and Plotly, offer powerful tools for creating static and interactive graphs. However, they lack specialized features that allow users to effectively structure and convey data-driven narratives. This paper introduces PyNarrative, an innovative Python library designed to fill this gap by integrating storytelling elements—such as annotations, context, and next steps boxes—into data visualizations. PyNarrative enables users to craft data stories that are informative but also engaging and memorable, making complex data accessible to a broader audience. This paper details the design and functionality of PyNarrative and shows a practical use case. Through PyNarrative, we aim to empower developers and data storytellers to transform raw data into meaningful narratives, advancing the field of data storytelling and contributing to more effective data communication.

- Link to Article

PyNarrative Prerequisites

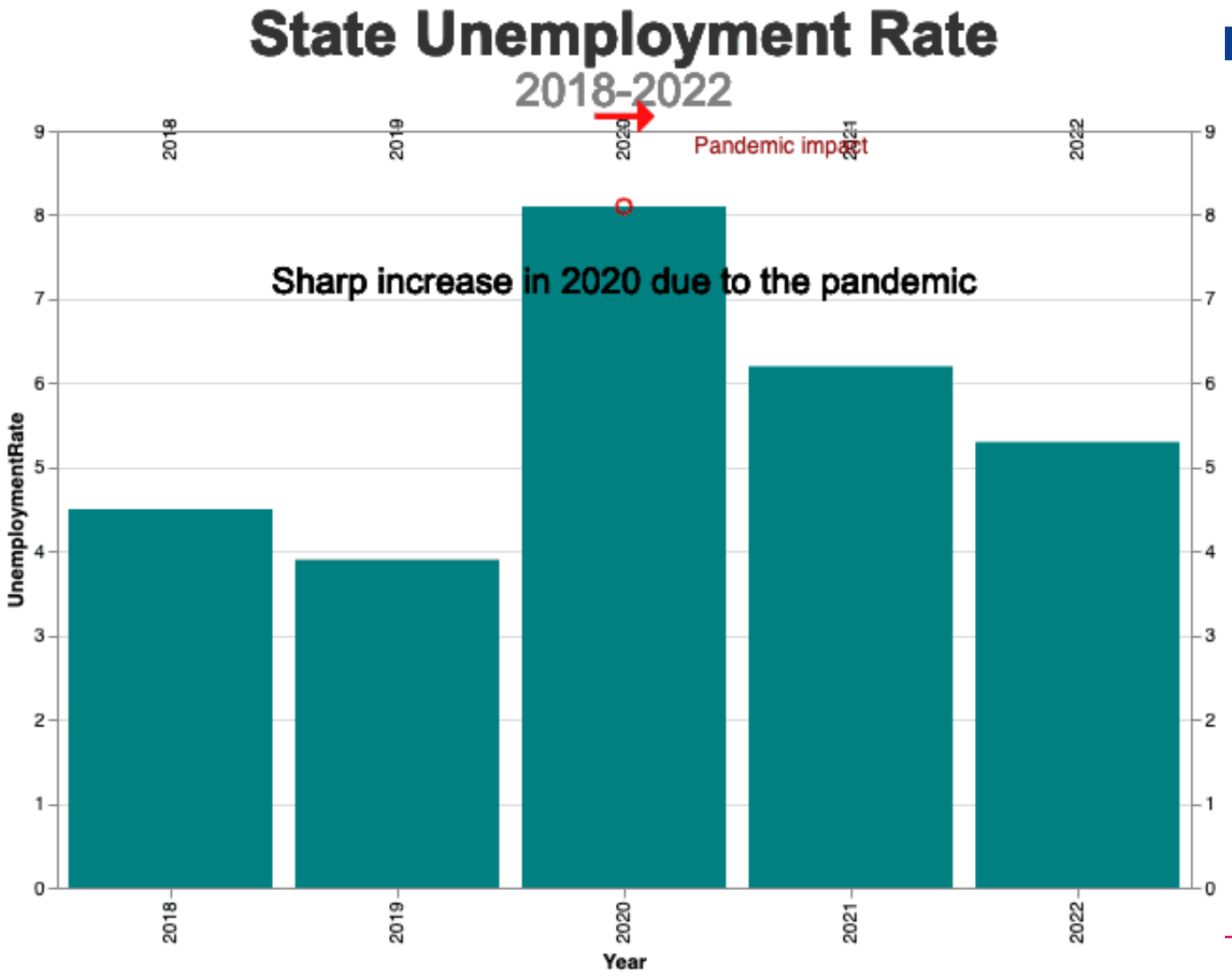
•**Built on Altair:** PyNarrative is an **extension of the Altair** declarative visualization library. It inherits Altair's powerful charting capabilities while adding specialized narrative features.

- ❑ **Declarative Approach:** Like Altair, it uses a declarative approach, allowing users to focus on *what* to tell, not *how* to draw every graphical detail.
- ❑ Requires python version 3.7 and later, altair common libraries like pandas: pip install pynarrative pandas altair

How it Works

Method	Purpose	Narrative Element Added
pn.Story(data)	Initializes the narrative visualization.	Foundation
.add_title(title, subtitle)	Establishes the primary message and subject.	Beginning
.add_context(text, position)	Adds contextual text or background information.	Beginning
.add_annotation(text, position)	Highlights key events, points, or insights directly on the chart.	Middle
.add_next_steps(text, mode)	Provides actionable recommendations or a call to action.	End
.add_source(text)	Includes references and credits for the data.	Supporting Context
.render()	Combines all elements (chart, context, annotations) and displays the final narrative visualization.	Output

Example Chart



Individual Class Practical

- Generate a pandas dataframe of 10 samples, with one categorical variable and one continuous variable
- Generate a suitable graph to convey insights from your dataframe using common libraries like matplotlib or seaborn library
- Generate a chart of the same dataframe using PyNarrative
- Add context and a narrative to your chart

Data Normalisation (Feature Scaling)

- Ensures all numerical input features in a dataset have the same scale. This is vital because machine learning algorithms that rely on distance calculations (like K-Nearest Neighbors, SVMs, and K-Means) or gradient descent (like Neural Networks and Linear Regression)
- **Prevents Feature Domination:** If one feature has a range from 0 to 10,000 (e.g., Annual Income) and another has a range from 0 to 1 (e.g., Customer Rating), the feature with the larger range will **dominate** the distance calculations, regardless of its actual predictive power.
- Ensures efficiency in model's performance (e.g. in gradient-based optimization algorithms like those used for Neural Networks)
- Reduces impact of outliers

Normalisation: Decimal Scaling

- Rescales the data by moving the decimal point of the attribute values. The goal is to transform the original data values, vi, into a normalized value, vi', such that vi' falls within the range of [−1,1].

$$v_i' = \frac{v_i}{10^j}$$

•v: The original data value.

•10^j: The scaling factor.

•j: The number of decimal places to shift the data. It is determined by the number of digits in the maximum absolute value of the feature.

- E.g. A=(20,000,85,000,124,000,30,−90,000)
- Maximum absolute value has 6 digits
- Sensitive to outliers

Original Value (v)	Calculation (v/10^6)	Normalized Value (v')
20,000	20,000/1,000,000	0.02
85,000	85,000/1,000,000	0.085
124,000	124,000/1,000,000	0.124
30	30/1,000,000	0.00003
-90,000	−90,000/1,000,000	-0.09

Normalisation: Min-Max Scaling

- Transforms data to a fixed range, typically **[0, 1]**. It's often referred to simply as "Normalization."
- Rescales the feature by subtracting the minimum value of the feature then dividing by the range
- Suitable when the data is not normally distributed and when the maximum and minimum boundaries are known and fixed.
- It works well for models like_K-Nearest Neighbors (KNN) which compare distance between data points
- Sensitive to outliers
- Use Pandas functions *.min()* and *.max()*.
- Or Scikitlearn function *MinMaxScaler*

Normalisation: Z-score Standardisation

- Transforms the data to have a mean (μ) of 0 and a standard deviation (σ) of 1.
- Suitable when the data is roughly normally distributed or when the presence of outliers makes Min-Max scaling unreliable
- Use sklearn function *StandardScaler*

Normalisation: Robust Scaling

- Robust Scaling uses the median and the interquartile range to scale features
- Suited to data with outliers.
- Use sklearn function *RobustScaler*

Label Encoding

- Label Encoding is best suited for **Ordinal Categorical Data**, where the categories have a **natural, meaningful order or ranking.**

- **Examples:**
  - **Size:** Small < Medium < Large
  - **Education:** High School < Bachelor's < Master's
  - **Rating:** Poor < Fair < Excellent

- **The Limitation and the Alternative**

The major drawback of Label Encoding is that it introduces an **artificial ordinal relationship** (or rank) among categories even when none exists (e.g., the model might wrongly assume that '2' > '1').

- **Scenario:** If you encode a **Nominal** feature like colors (Red=0, Blue=1, Green=2), the model may incorrectly treat Green as "greater" or "more important" than Red

Label Encoding vs One-Hot Encoding

Aspect	Label Encoding	One-Hot Encoding (OHE)
Data Type	**Ordinal** (Has order: Low, Medium, High)	**Nominal** (No order: Red, Blue, Green)
New Features	Creates **1** new integer column.	Creates **N** new binary (0/1) columns (where N is the number of categories).
Dimensionality	Does not increase it.	Greatly increases it (can lead to the Curse of Dimensionality).
Model Impact	Can mislead non-tree models (e.g., Linear, KNN).	Safe for all models, but creates sparse data.

Data Discretisation

- Converting continuous (numerical) data into a set of finite, ordered categories or intervals, also known as **bins**.
- This technique is a crucial step in data preprocessing, especially for algorithms like Decision Trees and Naive Bayes that work better with discrete or categorical features.
- Discretization techniques are broadly classified as **unsupervised** (not using class labels) or **supervised** (using class labels

Unsupervised Discretisation Techniques

- **Binning**: Divides data down to equal width and frequency by creating bins
- **Clustering**: Uses unsupervised ML algorithms like K-means to group data in to clusters (K)
- **Histogram Analysis**: Also creates euqal bins but with visuals

Supervised Discretisation Techniques

- **Decision Tree-Based Discretisation**: Uses a decision tree algorithm (e.g. CART) to partition data. Relies on class labels.
- **Entropy-Based Discretisation**: Uses theory concepts like entropy and information gain (e.g. ChiMerge method which relies on chisquare scores)

UGANDA CHRISTIAN UNIVERSITY
A Centre of Excellence in the Heart of Africa

Uganda Christian University
P.O. Box 4 Mukono, Uganda
Tel: 256-312-350800
https://ucu.ac.ug/ Email: info@ucu.ac.ug.
@ugandachristianuniversity @UCUniversity
@UgandaChristianUniversity

Department of Computing & Technology
FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY
Tel: +256 (0) 312 350 863 | WhatsApp: +256 (0) 708 114 300
@ucucomputeng @ucu_ComputEng
https://cse.ucu.ac.ug/ Email: dct-info@ucu.ac.ug