Day 6

📄 Session Summary: Virtual Environment, Git Workflow, Selenium, Pytest & Sentiment Analysis

Session 1: Python Virtual Environment, Git Workflow, Selenium, and Pytest

🔹 1. Setting Up a Python Virtual Environment

Objective:
To highlight the importance of isolated environments for managing project-specific dependencies and avoiding conflicts.

Commands Used:

```
python -m venv myenv

myenv\Scripts\activate  # For Windows
```

Key Takeaways:

- Maintains clean and manageable environments for each project.
- Prevents dependency version conflicts between different projects.
- Dependencies were installed using pip install within the activated environment.

🔹 2. Git & GitHub Workflow

Git Commands Practiced:

- git init – Initializes a new Git repository
- git add – Stages files for the next commit
- git commit – Commits staged changes with a message
- git push – Pushes commits to a remote GitHub repository
- git pull – Pulls and merges changes from a remote repo

GitHub Integration:

- Demonstrated how to create and link a local Git repository to GitHub.
- Explained the end-to-end workflow of pushing code to GitHub for version control and collaboration.

🔹 3. Selenium Automation with Python

Setup:

- Installed Selenium and Chrome WebDriver.

Demonstration Script:

```
from selenium import webdriver
```

```
driver = webdriver.Chrome()
driver.get("https://qxf2.com/selenium-tutorial-main")
name = driver.find_element(by="id", value="name")
name.send_keys("Qxf2")
```

Concepts Covered:

- Locating HTML elements using Selenium methods.
- Automating browser tasks such as form filling.
- Demonstrated efficient automation of user interactions within web pages.

🔹 4. Introduction to Pytest

Overview:

- Introduced Pytest for writing unit tests in Python.
- Discussed how Pytest helps streamline testing workflows with simpler syntax and better test organization.

Session 2: Sentiment Analysis Using Machine Learning

🔹 1. Libraries and Tools

Imported essential libraries for:

- Data processing: pandas, re, nltk
- Text transformation: TfidfVectorizer, PorterStemmer
- Model training and evaluation: LogisticRegression, train_test_split, accuracy_score, confusion_matrix, classification_report
- Visualization: seaborn, matplotlib.pyplot

🔹 2. Data Preparation

Dataset Used: twitter_training.csv

- Cleaned dataset by removing duplicates and irrelevant rows.
- Labeled sentiment values as:
    - *Positive* and *Neutral* → 1
    - *Negative* → -1

🔹 3. Text Preprocessing

Applied a series of transformations to prepare data:

- Converted all text to lowercase.
- Removed URLs, digits, and special characters.
- Removed stopwords using NLTK.
- Applied stemming using PorterStemmer for normalization.

🔹 4. Feature Extraction & Model Training

- Used **TF-IDF Vectorization** to convert text into numerical features.
- Split data into 80% training and 20% testing sets.
- Trained a **Logistic Regression** model for classification.

Evaluation Metrics:

- Accuracy Score
- Confusion Matrix (visualized using Seaborn)
- Classification Report including precision, recall, and F1-score

🔹 5. Sentiment Prediction Testing

Tested the model against a variety of inputs:

- Properly written sentences
- Misspelled or noisy data
- Sarcastic comments
- Emoji-rich inputs
- Very short and long text entries

Output:
Each input was displayed with its corresponding predicted sentiment label (Positive / Negative).

🔹 6. Word Cloud Visualization *(Optional)*

- A word cloud was generated from sample input text to highlight the most frequent terms, improving interpretability and insights.

✅ Conclusion

Across both sessions, participants gained practical experience in:

- Creating and managing Python virtual environments.
- Using Git and GitHub for collaborative development.
- Automating browser actions using Selenium.
- Writing and organizing tests with Pytest.
- Implementing text preprocessing and feature extraction techniques.
- Building, training, and evaluating a sentiment analysis model.
- Handling various input styles, including informal, noisy, and emoji-filled data, for better model robustness.