Session 1: Virtual Environment, Git Workflow, Selenium & Pytest

1. Python Virtual Environment Setup

Objective:
The session began by highlighting the significance of using virtual environments to isolate project dependencies and avoid version conflicts.

Commands Used:

```
python -m venv myenv

myenv\Scripts\activate  # For Windows
```

Key Discussion Points:

- Ensures cleaner and more manageable project environments.
- Prevents package version clashes across different projects.
- Dependencies were installed using pip install within the virtual environment.

---

2. Git & GitHub Workflow

Git Commands Practiced:

- git init – Initialize a Git repository.
- git add – Stage files for commit.
- git commit – Commit staged changes.
- git push – Upload commits to a remote repository.
- git pull – Fetch and merge changes from the remote.

GitHub Integration:

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

---

3. Selenium Automation with Python

Setup:

- Installed the Selenium package 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.
- Performing actions like sending input to form fields.
- Automating browser-based tasks efficiently.

---

4. Introduction to Pytest

Overview:

- A brief introduction to Pytest, its syntax, and usefulness in test automation.
- Discussed how Pytest simplifies writing and organizing test cases in Python.

Session 2: Sentiment Analysis Using Machine Learning

1. Libraries and Tools

Libraries imported for data handling, preprocessing, visualization, and modeling:

```
import pandas as pd, re, nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
```

---

2. Data Preparation

Dataset: twitter_training.csv

- Loaded with appropriate column names.
- Removed irrelevant and duplicate rows.
- Mapped sentiment values for classification:
    - positive and neutral → 1
    - negative → -1

---

3. Text Preprocessing

Applied several preprocessing steps:

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

---

4. Feature Extraction & Model Training

- Used **TF-IDF Vectorization** to transform textual data into numerical format.
- Split dataset into training and testing sets (80/20 ratio).
- Trained a **Logistic Regression** model.

Evaluation Metrics:

- Classification report
- Confusion matrix (visualized using Seaborn)
- Accuracy score

---

5. Sentiment Predictions

Tested model performance on various types of input:

- Grammatically correct and standard sentences

- Misspelled or noisy text
- Sarcastic comments
- Inputs with emojis
- Extremely short or long sentences

Output:

- Each input was printed with its predicted sentiment (Positive / Negative).

---

6. Word Cloud Visualization (Optional)

- A **Word Cloud** was generated to visualize the most frequently used terms in the sample inputs, enhancing interpretability.

Conclusion

Throughout the sessions, participants were exposed to:

- Creating and managing virtual environments for Python projects.
- Using Git and GitHub for version control and collaboration.
- Automating browser interactions with Selenium.
- Implementing text preprocessing techniques.
- Building and evaluating a sentiment analysis model.
- Handling various text types including noisy, sarcastic, and emoji-rich data for more robust prediction.