Git:

1. Cloning a Repository

- Used git clone <repo_url> to copy a remote repository to my local machine.

2. Setting Up a Virtual Environment

- Created a virtual environment using python -m venv <env_name>.
- Activated it using:
    - Windows: <env_name>\Scripts\activate
    - macOS/Linux: source <env_name>/bin/activate

3. Installing and Using Selenium

- Installed Selenium inside the virtual environment with pip install selenium.
- Used Selenium for basic web automation, like opening a browser and interacting with elements.

4. Git Workflow

- Created a new branch using git checkout -b <branch_name>.
- Made changes, added them with git add ., and committed with git commit -m "message".
- Merged the branch into main using git checkout main followed by git merge <branch_name>.
- Pushed the updated main branch to the remote repository with git push origin main.

Software testing:

Today, I learned a lot about working with text data and training a machine-learning model for sentiment analysis.

1. **Cleaning the text**
   - I loaded a dataset and removed unwanted stuff like URLs, special characters, and extra spaces.
   - Converted all text to lowercase to keep things consistent.
   - Removed common words like "the", "is", and "and" (stopwords) since they don’t add much meaning.
   - Used stemming to reduce words to their base form (like "running" → "run").
2. **Handling labels**
   - Standardized labels by converting them to lowercase.
   - Removed tweets marked as "irrelevant".
   - Changed labels so that "positive" and "neutral" = 1 and "negative" = -1.
   - Got rid of any missing data.
3. **Turning text into numbers**
   - Used TF-IDF to convert words into a format that a machine-learning model can understand.
4. **Training and testing the model**
   - Split the data into training and test sets.
   - Used a logistic regression model to predict sentiment.
   - Checked how well the model performed using accuracy and classification reports.
   - Plotted a confusion matrix to see where the model made mistakes.
5. **Testing with different types of messages**
   - Tried messages with emojis, misspellings, sarcasm, and very short texts.
   - Tested on noisy messages with repeated characters, random symbols, and numbers replacing letters.

The model might not handle sarcasm or misspellings well, so adding spell-checking or an emoji interpreter could help.

- Right now, "neutral" is treated the same as "positive," which might not be ideal.
- Trying other models like Random Forest or SVM might improve accuracy.

Overall, I built a complete sentiment analysis model and tested it with real-world challenges.