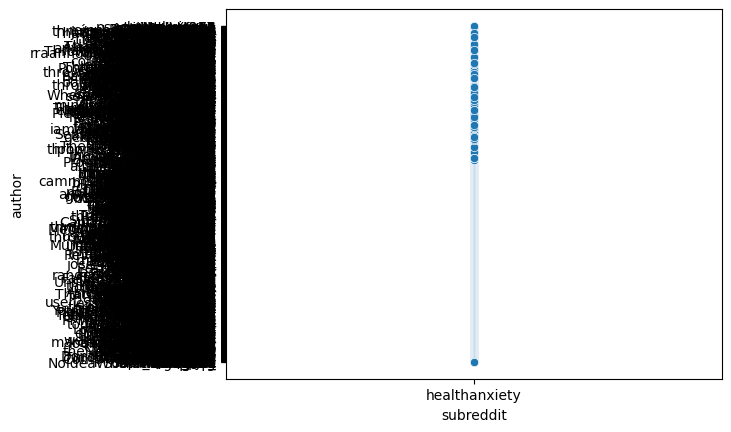
**Week 2: Fundamentals and ML Specific Concepts**

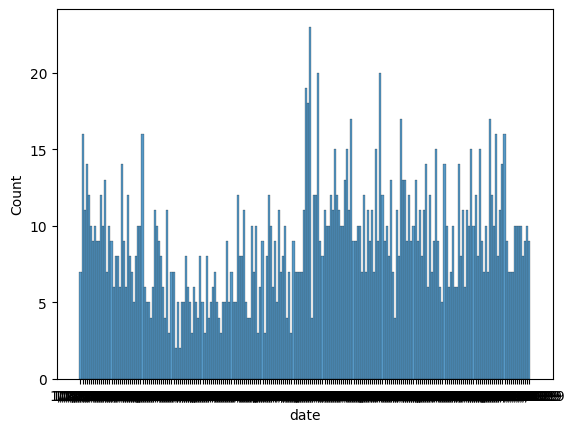
**Assignment 1: Exploring Data with Pandas and Seaborn**

Task:

* Load a dataset using Pandas.
* Explore the dataset (e.g., check for missing values, data types, summary statistics).
* Visualize some important features using Seaborn.
* pip install pandas seaborn matplotlib
* import pandas as pd
* # Replace 'dataset.csv' with the path to your dataset or the URL
* data = pd.read\_csv('healthanxiety\_dataset.csv')
* # Check the first few rows
* print(data.head())
* # Check for missing values
* print(data.isnull().sum())
* # Check data types
* print(data.dtypes)
* # Summary statistics
* print(data.describe())
* import seaborn as sns
* import matplotlib.pyplot as plt
* # Scatter plot
* sns.scatterplot(x='subreddit', y='author', data=data)
* plt.show()
* # Histogram
* sns.histplot(data=data, x='date', bins=20)
* plt.show()
* # Heatmap for correlation
* sns.heatmap(data.corr(), annot=True)
* plt.show()

**Output:**

****

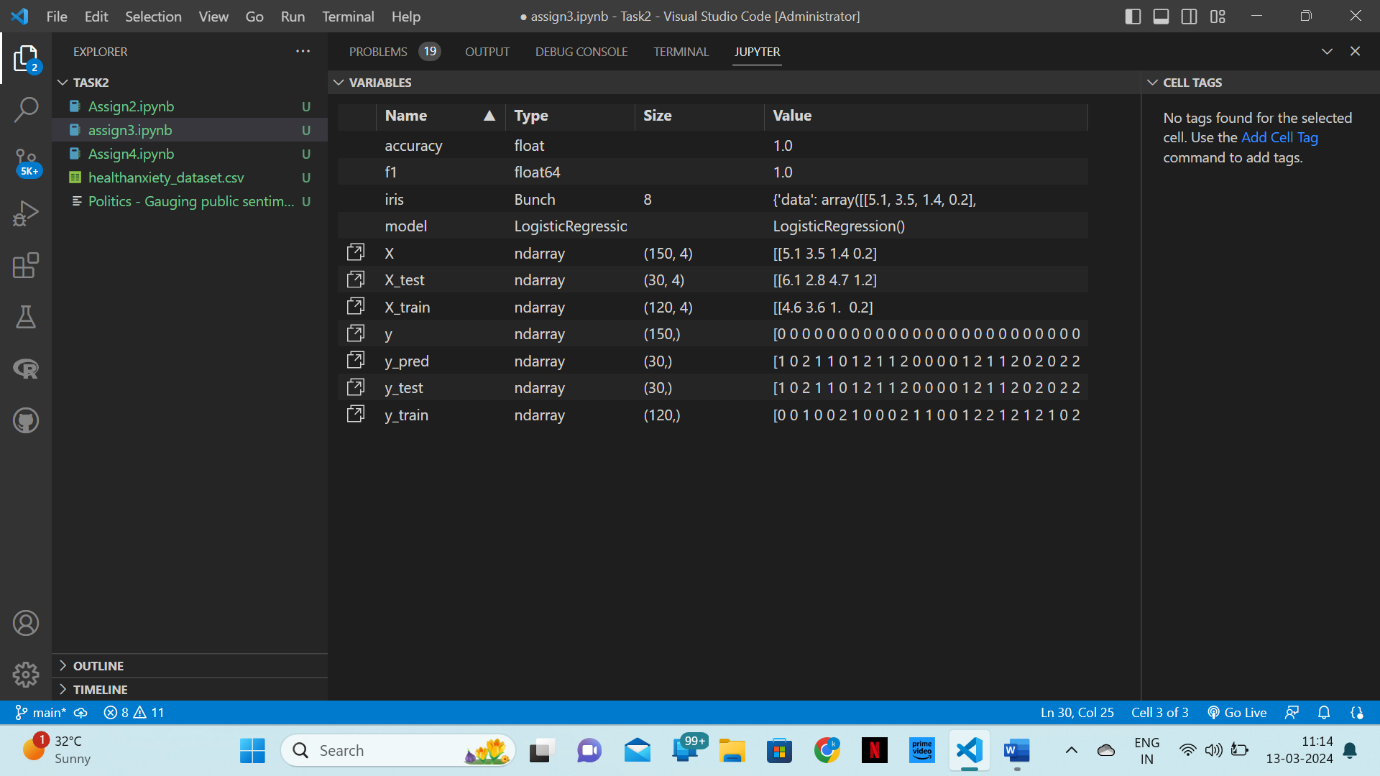
****

**Assignment 2: Implementing Logistic Regression**

Task:

* Load a dataset suitable for classification.
* Split the dataset into training and testing sets using train\_test\_split.
* Train a Logistic Regression model on the training data.
* Evaluate the model using accuracy and F1-score.
* import pandas as pd
* from sklearn.datasets import load\_iris
* from sklearn.linear\_model import LogisticRegression
* from sklearn.model\_selection import train\_test\_split
* from sklearn.metrics import accuracy\_score, f1\_score
* # Load the iris dataset
* iris = load\_iris()
* X = iris.data
* y = iris.target
* # Split the data into training and testing sets
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
* # Create a Logistic Regression model
* model = LogisticRegression()
* # Train the model
* model.fit(X\_train, y\_train)
* # Make predictions on the test set
* y\_pred = model.predict(X\_test)
* # Calculate accuracy
* accuracy = accuracy\_score(y\_test, y\_pred)
* print(f"Accuracy: {accuracy}")
* # Calculate F1-score
* f1 = f1\_score(y\_test, y\_pred, average='macro')
* print(f"F1-score: {f1}")

**Output:**

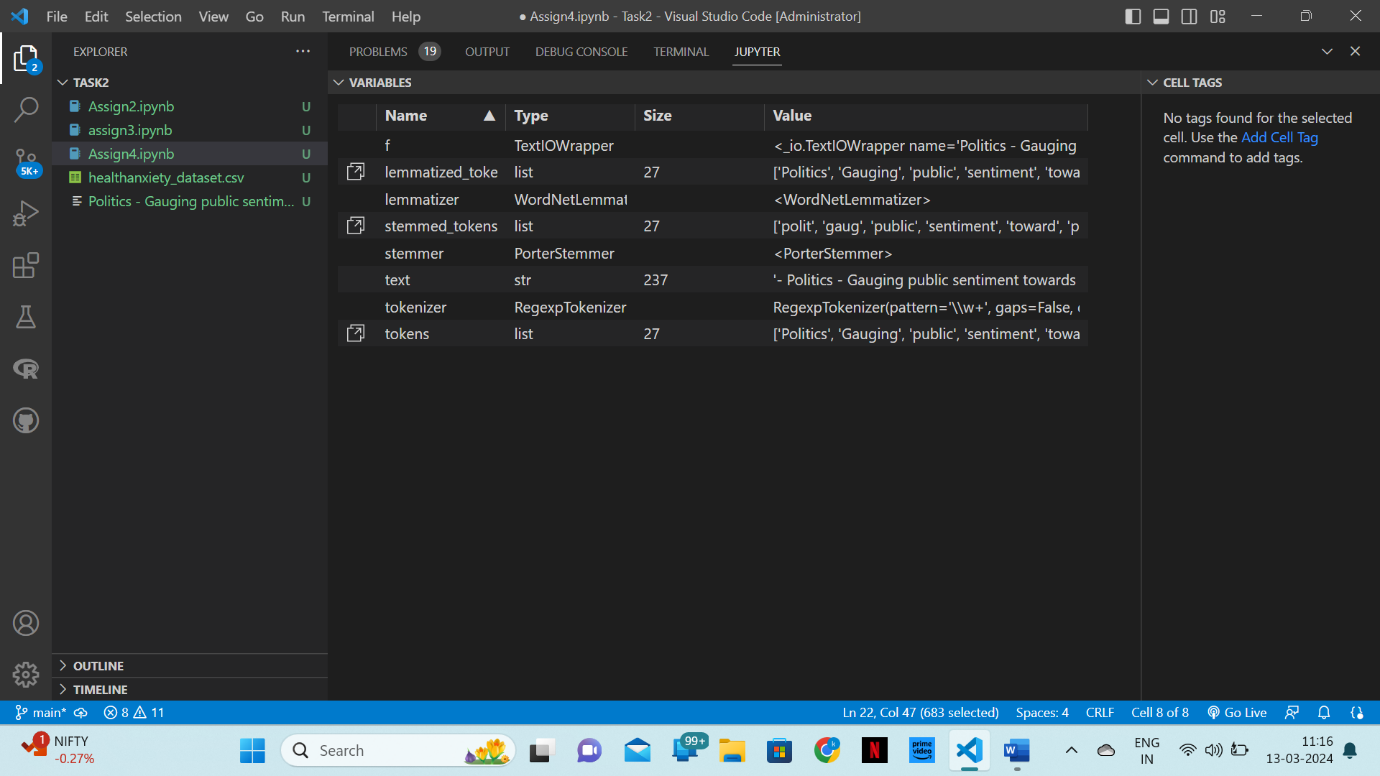
****

**Assignment 3: NLP Task with NLTK**

Task:

* Preprocess a text dataset using NLTK.
* Perform stemming and lemmatization.
* Tokenize the text using regexp tokenizer.
* import nltk
* nltk.download('punkt')
* nltk.download('wordnet')
* with open("Politics - Gauging public sentime.txt", "r", encoding="utf-8") as f:
* text = f.read()
* from nltk.tokenize import RegexpTokenizer
* tokenizer = RegexpTokenizer(r'\w+')
* tokens = tokenizer.tokenize(text)
* from nltk.stem import PorterStemmer
* stemmer = PorterStemmer()
* stemmed\_tokens = [stemmer.stem(token) for token in tokens]
* from nltk.stem import WordNetLemmatizer
* lemmatizer = WordNetLemmatizer()
* lemmatized\_tokens = [lemmatizer.lemmatize(token) for token in tokens]
* print("Original Tokens:", tokens)
* print("Stemmed Tokens:", stemmed\_tokens)
* print("Lemmatized Tokens:", lemmatized\_tokens)

**Output:**

****