WiDS 5.0 Fake News Detection using NLP

Week 1

FakeNewsDetection (https://github.com/anvaykurve/FakeNewsDetection)

Set up Python in VSCode. Built a pipeline and Git Integration

Setup Git repository

Learned the following things in Python:

Variables & Data Types: Storing data using Strings, Integers, Floats, and Booleans.

String functions

Collections:

Lists: Ordered, mutable collections.

Tuples: Ordered, immutable collections.

Dictionaries: used to store data objects with a key.

Control Flow:

If/Else Statements: Elif is used within a block

Loops: for and while loops

Functions

The def keyword: Defines a reusable block of code.

Modules: Importing external code libraries

File I/O: How to open, read, write, and append text files.

"r": Read only

"w": Write (Overwrite the file completely)

"a": Append (Add to the end of the file)

"r+": Read and Write

Classes, Objects and Inheritance

I can create multiple objects from the same class, and they will each hold their own unique data.

The __init__ function: It runs automatically when you create a new object to set up its initial data.

Inheritance: The ability to create a new class that "inherits" all functions from an existing class.

Brief Introduction to Colab

I can write and execute code in Python directly. It shows the output of blocks below them, which is good to test certain parts.

Week 2

Exploring Columns: Title, Text, and Label

Loading Data: Use pd.read_csv('filename.csv') to import the dataset into a DataFrame.

Inspecting Structure:

Use df.head() to view the first 5 rows and understand the column structure (e.g., verifying title, text, and label exist).

Use df.columns to list all column names and df.dtypes to verify data types (ensuring text columns are objects/strings).

Use df.shape to see the total number of rows (articles) and columns.

Checking Missing Values and Basic Statistics

Missing Values:

Run df.isna().sum() to identify null values in the text or title columns.

Handling Nulls: If a news article has no text, it cannot be classified. You can drop these rows using df.dropna() or fill them if appropriate.

Basic Statistics:

Use df.describe() to get summary statistics. While typically for numbers, on text columns, it shows unique counts and the most frequent entries (useful for spotting duplicate articles).

Duplicate Detection: Use df.duplicated().sum() to check for repeated articles, which can bias the model.

Removing Unwanted Columns

Dropping Columns:

Identify columns irrelevant to classification (e.g., id, author, index).

Use df.drop(['column_name'], axis=1) to remove them.

Subsetting: Alternatively, select only the columns you need: df = df[['title', 'text', 'label']].copy().

Combining Fields: Creating a Content Column

Feature Engineering:

Often, the title of a news article contains vital information not found in the body text.

Action: Create a new column (e.g., content) by concatenating the title and text. This ensures the model treats them as a single rich feature.

Note: While the videos focus on numeric manipulation, text concatenation in Pandas follows standard Python string logic.

Text Cleaning

This is the most critical step to convert human language into a machine readable format.

Lowercasing:

Lowercasing normalises the treatment of two same words. Eg. Apple and apple

Removing URLs, Punctuation, and Numbers:

Regex (Regular Expressions): Use Python's re library.

URLs: Remove patterns starting with http:// or www as they rarely carry sentiment or topic information.

Punctuation: Using string. punctuation or regex to strip characters like '!', '.', ',' which add noise to tokenisation.

Numbers: Digits are often removed unless specific dates/quantities are relevant to the fake news context.

Removing Stopwords: Stopwords are high-frequency words (e.g., the, is) that carry little special meaning.

Library: nltk.corpus.stopwords

Process: Iterate through tokens and filter out any word present in the standard English stopword list. This reduces the dataset size and focuses the model on keywords.

Lemmatisation vs. Stemming: The tutorials highlight that Stemming simply chops off ends of words (changing "giving" to "giv"), which can lead to non-words.

Technique: Lemmatisation (using WordNetLemmatizer) is preferred. It uses a dictionary to resolve words to their meaningful root form (e.g., "better"->"good", "running"->"run").

Prerequisite: Effective lemmatisation often requires Part-of-Speech (POS) tags to know if a word is a noun, verb, or adjective.

Saving Cleaned Text

Final Output:

Store the processed data in a new column (e.g., clean_content) alongside the original text for comparison.

Export: Save the cleaned dataset using df.to_csv('cleaned_news_data.csv', index=False) to be used in next week's model building phase.

Week 3

Introduction to Word Embeddings

Definition: Word2Vec is a "Word Embedding" technique that represents words as real-valued vectors in a high-dimensional space.

Semantic Similarity: Unlike Bag of Words or TF-IDF, Word2Vec ensures that words with similar meanings are placed close to each other in the vector space.

Types of Word Embeddings

Frequency-based: Such as Bag of Words, TF-IDF, and GloVe (Global Vectors), which rely on matrix factorization.

Prediction-based: This is where Word2Vec belongs, as it uses neural networks to predict words and learn their vector representations.

Core Architectures

CBOW (Continuous Bag of Words): Predicts a "target word" based on its surrounding "context words."

Skip-gram: The inverse of CBOW; it uses a single "target word" to predict the surrounding "context words."

Advantages over Traditional Methods

Dimensionality: Traditional methods like One-Hot Encoding create very sparse and high-dimensional vectors (equal to the vocabulary size). Word2Vec creates dense, low-dimensional vectors (e.g., 100 or 300 dimensions).

Contextual Meaning: Word2Vec captures the relationship between words (e.g., "King" and "Queen" will have similar vector patterns).