**GROUP 5 - Fake News Detection Using NLP**

**Phase 4: Building the project by selecting a machine learning algorithm, training the model, and evaluating its performance.**

**Coding for COVID19 Fake News Dataset NLP:**

import pandas as pd

# Load the dataset into a DataFrame

df=pd.read\_csv(r'C:\Users\DELL\Documents\datasets\csvfile\Constraint\_Test.csv')

# Explore the first few rows of the dataset

print(df.head())

# Assuming 'tweet' is the column containing the text and 'label' is the column containing labels

# Removing unnecessary columns

df = df[['tweet', 'label']]

# Checking for and handling missing values

df.dropna(inplace=True)

# Converting labels to binary (1 for fake news, 0 for real news)

df['label'] = df['label'].apply(lambda x: 1 if x > 0 else 0)

# Text cleaning (you might need more advanced text preprocessing steps)

import re

def clean\_text(text):

text = re.sub(r'http\S+', '', text) # Remove URLs

text = re.sub(r'[^a-zA-Z]', ' ', text) # Keep only letters

text = text.lower() # Convert to lowercase

return text

df['cleaned\_tweet'] = df['tweet'].apply(clean\_text)

from sklearn.feature\_extraction.text import TfidfVectorizer

# Tokenization and vectorization

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english', max\_features=5000)

X = tfidf\_vectorizer.fit\_transform(df['cleaned\_tweet'])

y = df['label']

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, classification\_report

# 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)

# Train a Naive Bayes classifier

clf = MultinomialNB()

clf.fit(X\_train, y\_train)

# Predictions on the test set

predictions = clf.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, predictions)

print(f'Accuracy: {accuracy:.2f}')

print(classification\_report(y\_test, predictions))

**The code performs the following tasks related to building a fake news detection model using the Naive Bayes algorithm and evaluating its performance:**

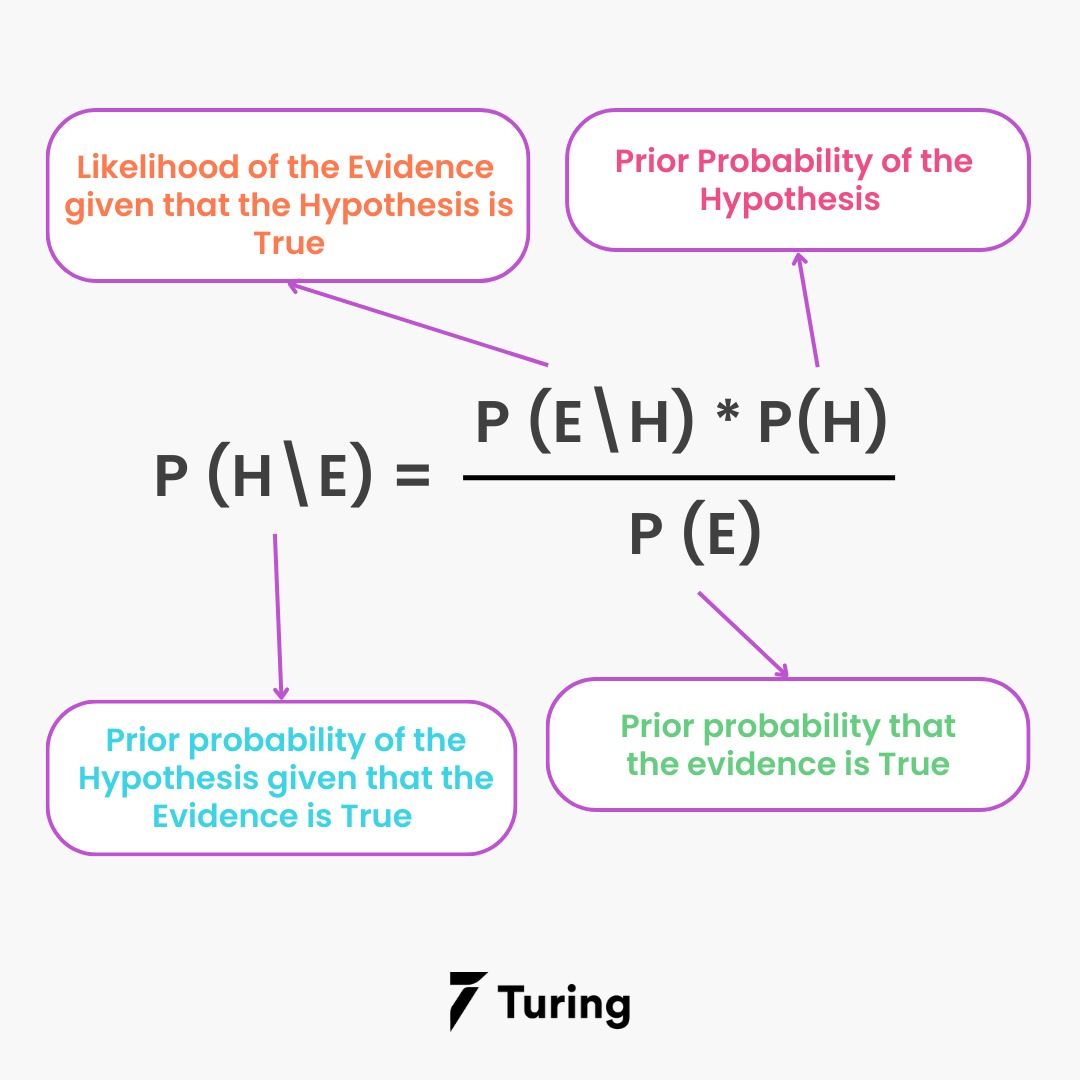
* Data Loading: Loads the dataset from a CSV file into a pandas DataFrame. Displays the first few rows of the dataset to provide an overview.
* Data Preprocessing: Keeps only the 'tweet' and 'label' columns, removing unnecessary columns. Handles missing values by dropping rows with missing data.Converts the 'label' column to binary values (1 for fake news, 0 for real news).Performs basic text cleaning, including removing URLs, non-alphabetic characters, and converting text to lowercase.
* Feature Extraction: Uses TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert text data into numerical vectors .Limits the number of features to 5000 for simplicity.
* Data Splitting: Splits the data into training and testing sets (80% training, 20% testing) using train\_test\_split from scikit-learn.
* Model Training :Chooses the Multinomial Naive Bayes classifier. Fits the classifier on the training data.
* Model Evaluation: Predicts labels for the test set. Computes accuracy and prints a classification report, including precision, recall, and F1-score.

# Naïve Bayes classifier:

Up to this point we already got feature vectors for all samples stored in X variable. To make things more intuitive, I will also define y variable, which I will use it to store all ground truths (a.k.a. labels).

y = df['label'].values

Now we can use y as the replacement of df[‘label’].values



Before training a classifier, we are going to split the data into train and test, where 20% of the entire samples in the dataset are going to be used to test the overall performance of the model. This splitting can easily be done using train\_test\_split function taken from Sklearn module:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, shuffle=True, test\_size=0.2, random\_state=11)

After running the code above, we got 4 new variables .

Now as the data already split it’s time to define a model which in this project I will be using Naïve Bayes. Mathematically speaking, this algorithm works by calculating the class (label) prediction probability based on given features (text) of each sample using Bayes’ theorem.

# Model evaluation:

After training a model, I usually also create a confusion matrix in order to find out the number of misclassified samples in more detail. In order to do so, I need to predict the class of test data first:

predictions = clf.predict(X\_test)

Next, we can just compare the values of predictions variable with its ground truth y\_test using confusion\_matrix() function coming from Sklearn module. accuracy = accuracy\_score(y\_test, predictions).

**Output:**

label tweet

0 1 Our daily update is published. States reported...

1 2 Alfalfa is the only cure for COVID-19.

2 3 President Trump Asked What He Would Do If He W...

3 4 States reported 630 deaths. We are still seein...

4 5 This is the sixth time a global health emergen...

Accuracy: 1.00 precision recall f1-score support

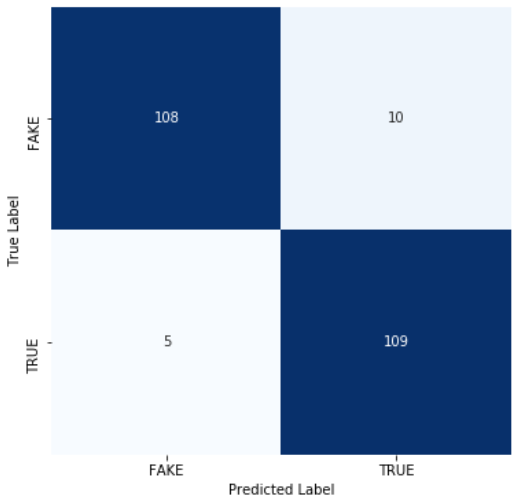
1 1.00 1.00 1.00 428

accuracy 1.00 428

macro avg 1.00 1.00 1.00 428

weighted avg 1.00 1.00 1.00 428

Confusion matrix of Naïve Bayes model on COVID-19 news dataset.



----------------------------------------------------------------------------------------------------------------