**PHASE 5**

Project Title : **FAKE NEWS DETECTION USING NLP**

**Name : L.Keerthana(210521205028)**

**Outline of the problem statement :**

The problem statement for the IBM project on Fake News Detection using NLP is to develop a system that can automatically identify and differentiate between real news and fake news articles or content using Natural Language Processing (NLP) techniques. This involves creating a model or software that can analyze textual data to determine the authenticity of news.

**Design thinking process :**

Applying design thinking to the project of Fake News Detection using NLP involves the following steps:

**Empathize:**

Understand the needs and concerns of users, such as the general public, journalists, or social media platforms.Conduct surveys and interviews to gather insights about people's experiences with fake news and their expectations for a detection system.

**Define:**

Clearly define the problem statement and the specific goals of the project.Create user personas to represent the different types of users who will interact with the system.

**Ideate:**

Brainstorm potential solutions and approaches for detecting fake news using NLP.Encourage creativity and open-minded thinking to explore a wide range of ideas.

**Prototype:**

Develop early-stage prototypes or mockups of the NLP-based fake news detection system.These prototypes can be paper sketches, wireframes, or low-fidelity digital representations to visualize the user interface and interaction flow.

**Test:**

Gather feedback from users and stakeholders by presenting the prototypes.Collect input on the usability, effectiveness, and user experience of the system.Use this feedback to refine and iterate on the design.

**Develop:**

Begin building the actual NLP model and system based on the refined prototype and feedback received.Incorporate data collection and preprocessing, feature extraction, model training, and evaluation into the development process.Implement:

**Test Again:**

Continuously test and evaluate the system during the development phase to identify and address any issues, especially related to false positives or false negatives in fake news detection.

Deploy the NLP-based fake news detection system in a real-world or controlled environment.Ensure that it can process and analyze news content effectively and efficiently.

**Launch:**

Introduce the system to users, whether they are individuals, news outlets, or social media platforms.Provide necessary training and support for users to adopt and utilize the system.

**Learn and Iterate:**

Collect feedback and user data in the post-launch phase to assess the system's performance and user satisfaction.Continuously improve the system based on user feedback, evolving fake news tactics, and advancements in NLP technology.

**Description of the dataset :**

**Dataset**

**Link: https://www.kaggle.com/datasets/cimen tbisaillon/fake-and-real-news-dataset**

**Dataset Source:**

The dataset for this project can be collected from a variety of sources, such as:

**1.News Websites:**

Scraping news articles from reputable news websites and blogs. These articles can be labeled as real news.

**2.Fact-Checking Websites:**

Using data from fact-checking websites like Snopes, PolitiFact, or FactCheck.org, which often publish articles debunking fake news.

**3.Social Media:**

Collecting posts and articles from social media platforms like Twitter, Facebook, and Reddit. Some of these may contain fake news.

**4.User Contributions:**

Allowing users to submit news articles or headlines that they suspect are fake. This crowdsourced data can be valuable.

**5.Historical Data:**

Using historical news archives to create a diverse dataset of both real and fake news.

**Data preprocessing steps involved:**

1. **Data Collection:**

Gather news articles, headlines, or text data from various sources, as mentioned earlier, and ensure that you have a diverse and representative dataset.

**2. Text Cleaning:**

***- Lowercasing:*** Convert all text to lowercase to ensure uniformity.

***- Removing Special Characters:*** Eliminate unnecessary special characters, punctuation, and symbols.

- ***HTML Tag Removal:*** If you collected data from web pages, remove HTML tags.

***- Stopword Removal:*** Remove common words (stopwords) that don't carry significant meaning (e.g., "the," "and").

- ***Whitespace Removal:*** Remove extra whitespaces and trim text.

**3.Tokenization:**

Split the text into words or tokens. This step involves breaking down text into smaller units (words or subwords), making it ready for analysis.

1. **Lemmatization or Stemming:**

Reduce words to their base or root form. Lemmatization and stemming can help normalize the text by converting words to their canonical forms.

1. **Handling Missing Data:**

Address missing values in the dataset, which may arise from incomplete or corrupted data.

1. **Balancing Classes:**

If your dataset has imbalanced classes (e.g., more real news than fake news), consider techniques like oversampling or undersampling to balance the classes.

1. **Feature Engineering:**

Create relevant features for your NLP model, such as word embeddings (e.g., Word2Vec or GloVe vectors) or TF-IDF (Term Frequency-Inverse Document Frequency) representations.

1. **Text Vectorization:**

Convert the preprocessed text data into numerical format that can be used by machine learning models. Common techniques include one-hot encoding, count vectorization, and TF-IDF vectorization.

1. **Splitting the Data:**

Divide the dataset into training, validation, and testing sets to evaluate the model's performance.

1. **Encoding Labels:**

If the labels are not binary (real or fake), encode them into a suitable format for classification tasks.

1. **Data Normalization:**

Scale the numerical features if necessary to ensure consistent ranges.

1. **Data Serialization:**

Save the preprocessed data for future use and model deployment.

**Future extraction technique :**

In a Fake News Detection NLP project, feature extraction methods encompass Bag of Words, TF-IDF, Word Embeddings, Sentiment Analysis, Named Entity Recognition, Stylistic Features, Content Similarity, Source Analysis, and Fact-Checking. These techniques help discern linguistic patterns, author credibility, content semantics, and contextual clues for accurate classification.

**Choice of Classification Algorithm:**

The choice of a classification algorithm and the model training process in a Fake News Detection project using NLP is critical for achieving accurate results. Here's an explanation of how to select a classification algorithm and the training process:

**1. Logistic Regression:**

A simple yet effective algorithm for binary classification tasks. It's a good starting point and serves as a baseline model.

**2. Naive Bayes:**

Particularly suited for text classification tasks. Naive Bayes models work well with text data and can handle large feature spaces efficiently.

**3. Support Vector Machine (SVM):**

SVMs are effective at finding decision boundaries in high-dimensional spaces. They can be used with various kernels to capture complex patterns in the data.

**4. Random Forest:**

An ensemble learning method that can handle a mix of feature types. It's robust and can reduce overfitting.

**5. Gradient Boosting:**

Algorithms like XGBoost or LightGBM can capture complex relationships in the data. They often yield high performance but may require more tuning.

**6. Deep Learning:**

Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), or more advanced models like Transformers (e.g., BERT) can capture intricate textual patterns, but they require substantial data and computational resources.

**Model Training Process:**

**1. Data Split:**

Divide the dataset into training, validation, and testing sets to evaluate the model's performance. Common splits include 70-80% for training, 10-15% for validation, and 10-15% for testing.

**2. Feature Vectorization:**

Convert the preprocessed text data into numerical features (e.g., TF-IDF vectors) that can be used by the chosen algorithm.

**3. Model Selection:**

Experiment with multiple classification algorithms to determine which one performs best on the validation set. You may consider ensembling models for improved accuracy.

**4. Hyperparameter Tuning:**

Optimize the hyperparameters of the selected model, such as regularization strength, learning rate, or the number of trees in a Random Forest.

**5. Model Training:**

Train the selected model on the training data. Pay attention to issues like class imbalance and apply techniques like oversampling or undersampling as needed.

**6. Evaluation:**

Assess the model's performance using various metrics like accuracy, precision, recall, F1-score, and ROC-AUC on the validation set. Ensure the model doesn't overfit.

**7. Fine-Tuning:**

Refine the model by adjusting hyperparameters based on the validation performance. You may also consider techniques like early stopping.

**8. Final Evaluation:**

Evaluate the best-performing model on the test set to obtain an unbiased estimate of its performance.

**9. Interpretability:**

For transparency, consider techniques like feature importance analysis or model interpretability methods, especially for decision-makers and users.

**10. Deployment:**

Implement the final model in a real-time or batch processing system for fake news detection.

**11. Continuous Monitoring and Improvement:**

Monitor the model's performance in a real-world setting and implement mechanisms for ongoing improvement as fake news tactics evolve.

The choice of the classification algorithm should be guided by the specific characteristics of the dataset and the level of complexity required. The training process involves iterative optimization to ensure the model can accurately distinguish between real and fake news.

**Coding :**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import plotly.express as px

import plotly.graph\_objs as go

from plotly.subplots import make\_subplots

import nltk

from nltk.corpus import stopwords

import tensorflow as tf

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import ModelCheckpoint

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

from transformers import AutoTokenizer, TFAutoModelForSequenceClassification

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

nltk.download('stopwords')

fake\_news\_path = "/kaggle/input/fake-and-real-news-dataset/Fake.csv"

real\_news\_path = "/kaggle/input/fake-and-real-news-dataset/True.csv"

fake\_news = pd.read\_csv(fake\_news\_path)

real\_news = pd.read\_csv(real\_news\_path)

fake\_news.head(3)

real\_news.head(3)

real = real\_news.copy()

fake = fake\_news.copy()

real['Label'] = 'Real'

fake['Label'] = 'Fake'

news = pd.concat([real, fake], axis=0, ignore\_index=True)

news.reset\_index()

news.head()

print(f"Samples available: {news.shape[0]}\n#features of dataset: {news.shape[1]}")

news\_ds = news.sample(1000).drop(['title', 'date', 'subject'], axis=1)

news\_ds.head(3)

CLASS\_NAMES = ['Fake', 'Real']

class\_mapper = {

'Fake':0,

'Real':1

}

news\_ds['Label'] = news\_ds['Label'].map(class\_mapper)

news\_ds.head(3)

class\_dist = px.histogram(data\_frame=news,

y='Label',

color='Label',

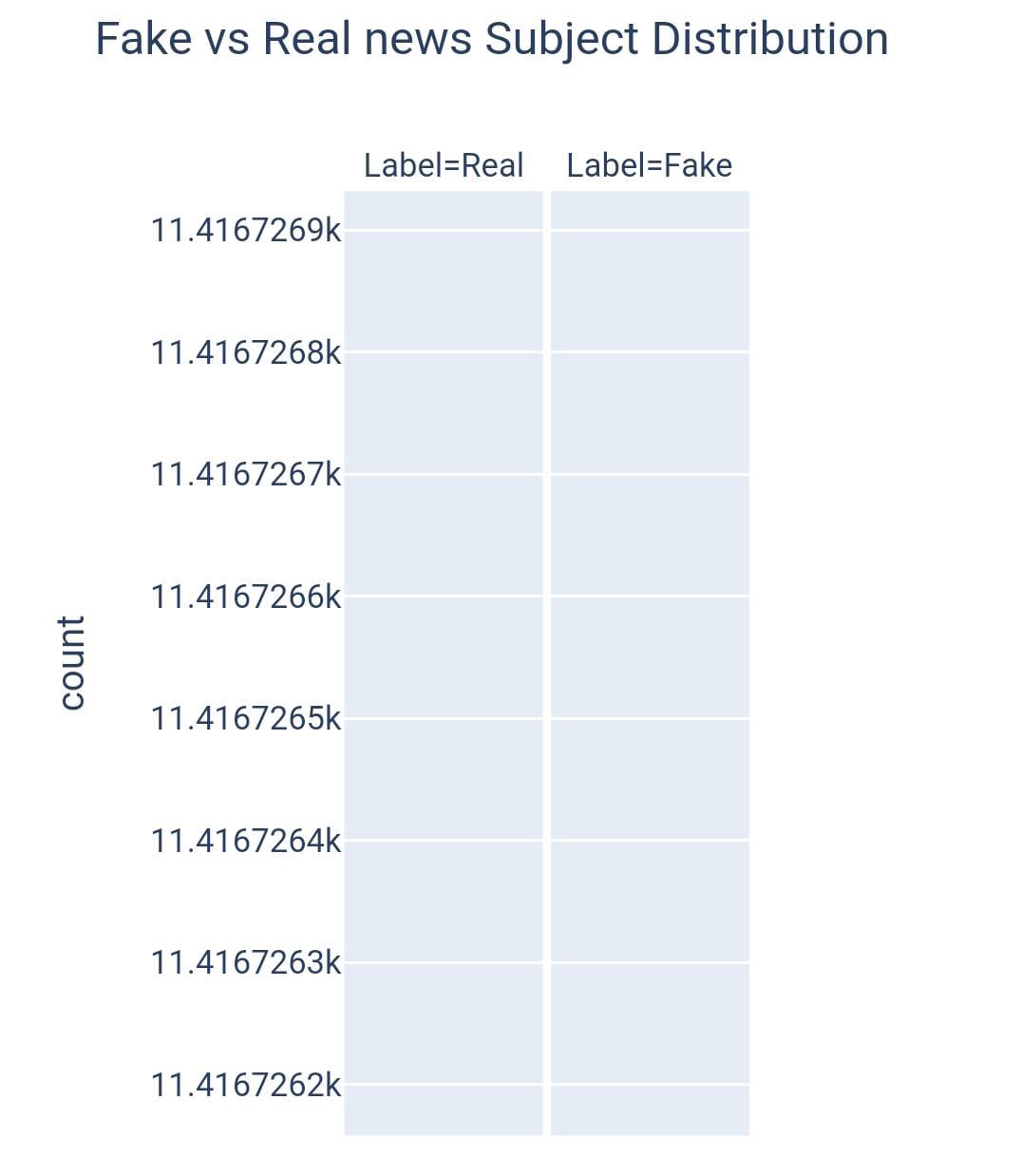
title='Fake vs Real news Original dataset',

text\_auto=True)

class\_dist.update\_layout(showlegend=False)

class\_dist.show()

OUTPUT :



news.date.unique().max()

[2:39 am, 21/10/2023] Jayasri R: news = news[news['date'].map(lambda x:len(x)) <= 20]

news.date = pd.to\_datetime(news['date'], format='mixed')

news.head()

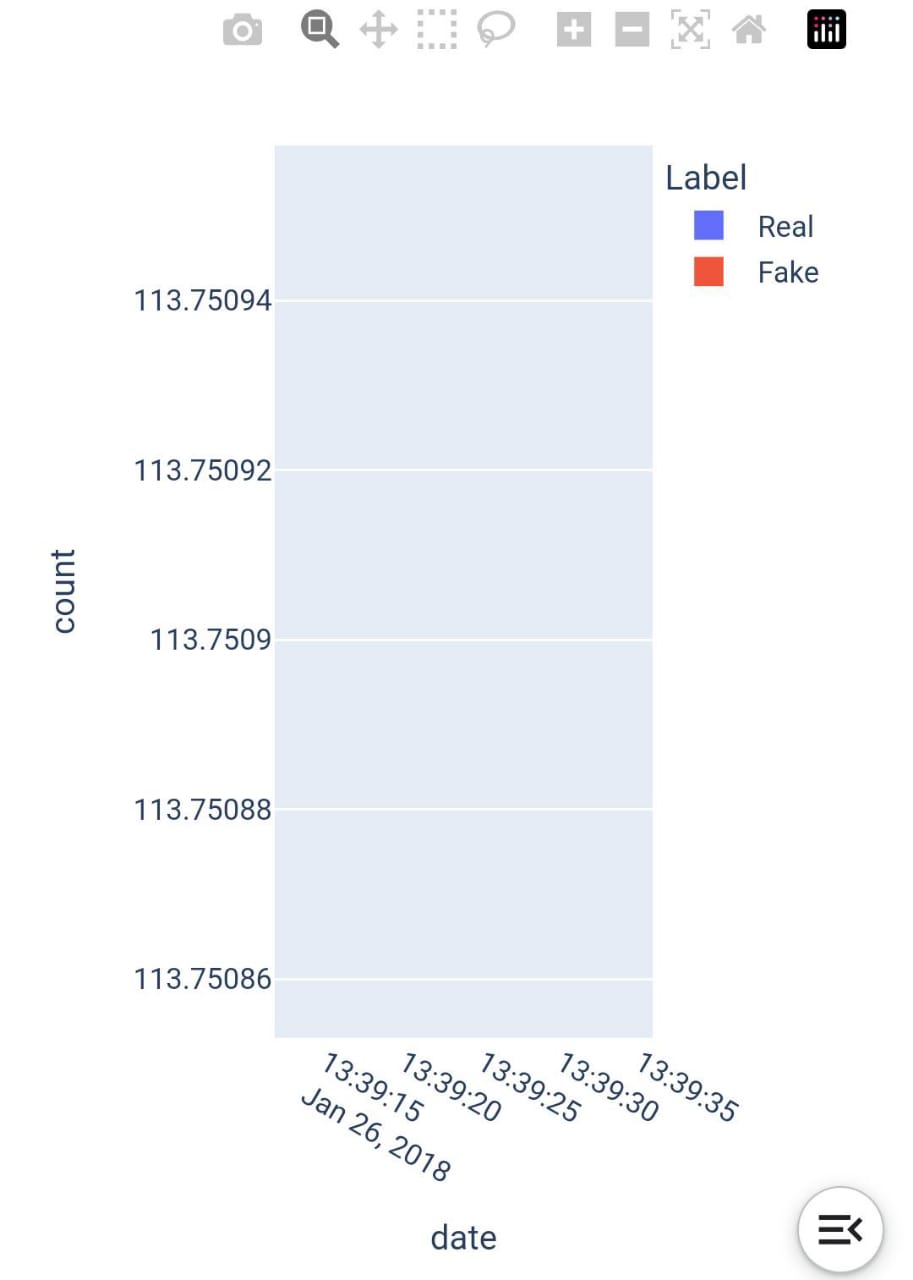
date\_dist = px.histogram(data\_frame=news,

x='date',

color='Label')

date\_dist.show()

OUTPUT :



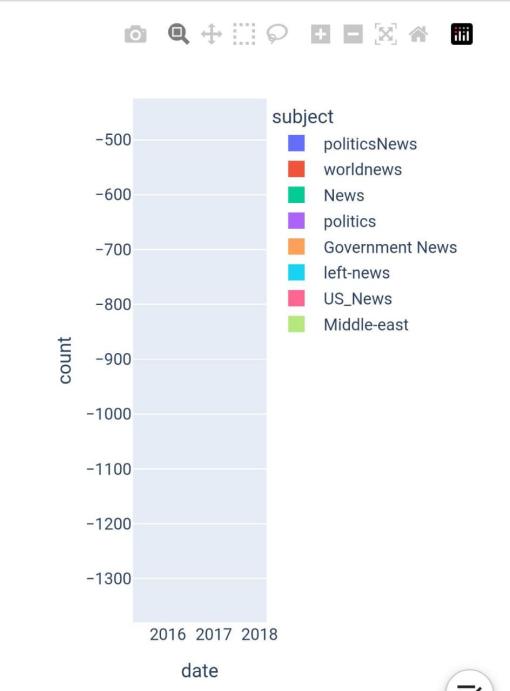
subject\_dist = px.histogram(data\_frame=news,

x='date',

color='subject')

subject\_dist.show()

OUTPUT :



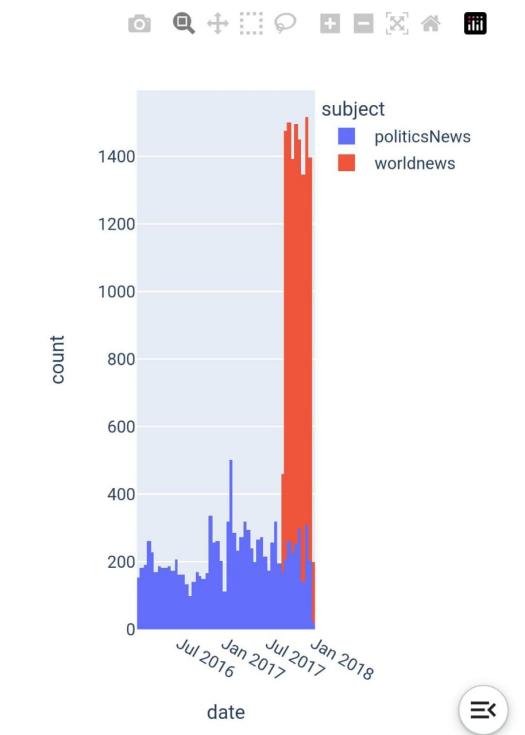
real\_sub\_dist = px.histogram(data\_frame=news[news['Label']=='Real'],

x='date',

color='subject')

real\_sub\_dist.show()

OUTPUT :



import string

stop\_words = stopwords.words('english')

def text\_preprocessing(text):

words = text.lower().split()

filtered\_words = [word for word in words if word not in stop\_words]

pure\_text = ' '.join(filtered\_words)

pure\_text = pure\_text.translate(str.maketrans('', '', string.punctuation)).strip()

return pure\_text

X = news\_ds.text.apply(text\_preprocessing).to\_numpy()

y = news\_ds.Label.to\_numpy().astype('float32').reshape(-1, 1)

train\_X, test\_X, train\_y, test\_y = train\_test\_split(X, y,

train\_size=0.9,

stratify=y,

random\_state=7)

train\_X, val\_X, train\_y, val\_y = train\_test\_split(train\_X, train\_y,

train\_size=0.9,

stratify=train\_y,

random\_state=7)

model\_name = "BERTFakeNewsDetector"

model\_callbacks = ModelCheckpoint(model\_name, save\_best\_only=True)

bert\_name = 'bert-base-uncased'

tokenizer = AutoTokenizer.from\_pretrained(bert\_name,

padding='max\_length',

do\_lower\_case=True,

add\_special\_tokens=True)

def tokenize(df):

inputs = tokenizer(df.tolist(),

padding=True,

truncation=True,

return\_tensors='tf').input\_ids

return inputs

train\_X\_encoded = tokenize(train\_X)

val\_X\_encoded = tokenize(val\_X)

test\_X\_encoded = tokenize(test\_X)

def prepare\_datasets(encoded, true\_df, true\_target\_df):

return tf.data.Dataset.from\_tensor\_slices((encoded, true\_target\_df)).shuffle(true\_df.shape[0]).batch(8).prefetch(tf.data.AUTOTUNE)

train\_ds = prepare\_datasets(train\_X\_encoded, train\_X, train\_y)

test\_ds = prepare\_datasets(test\_X\_encoded, test\_X, test\_y)

val\_ds = prepare\_datasets(val\_X\_encoded, val\_X, val\_y)

model = TFAutoModelForSequenceClassification.from\_pretrained(bert\_name,

num\_labels=1)

model.save(model\_name)

fig = make\_subplots(rows=2, cols=2, subplot\_titles=('Loss', 'Accuracy', 'Precision', 'Recall'))

fig.add\_trace(go.Scatter(y=model\_history['loss'], mode='lines', name='Training Loss'), row=1, col=1)

fig.add\_trace(go.Scatter(y=model\_history['val\_loss'], mode='lines', name='Validation Loss'), row=1, col=1)

fig.add\_trace(go.Scatter(y=model\_history['Accuracy'], mode='lines', name='Training Accuracy'), row=1, col=2)

fig.add\_trace(go.Scatter(y=model\_history['val\_Accuracy'], mode='lines', name='Validation Accuracy'), row=1, col=2)

fig.add\_trace(go.Scatter(y=model\_history['Precision'], mode='lines', name='Training Precision'), row=2, col=1)

fig.add\_trace(go.Scatter(y=model\_history['val\_Precision'], mode='lines', name='Validation Precision'), row=2, col=1)

fig.add\_trace(go.Scatter(y=model\_history['Recall'], mode='lines', name='Training Recall'), row=2, col=2)

fig.add\_trace(go.Scatter(y=model\_history['val\_Recall'], mode='lines', name='Validation Recall'), row=2, col=2)

fig.update\_layout(title='Model Training History')

fig.update\_xaxes(title\_text='Epoch', row=1, col=1)

fig.update\_xaxes(title\_text='Epoch', row=1, col=2)

fig.update\_xaxes(title\_text='Epoch', row=2, col=1)

fig.update\_xaxes(title\_text='Epoch', row=2, col=2)

fig.update\_yaxes(title\_text='Loss', row=1, col=1)

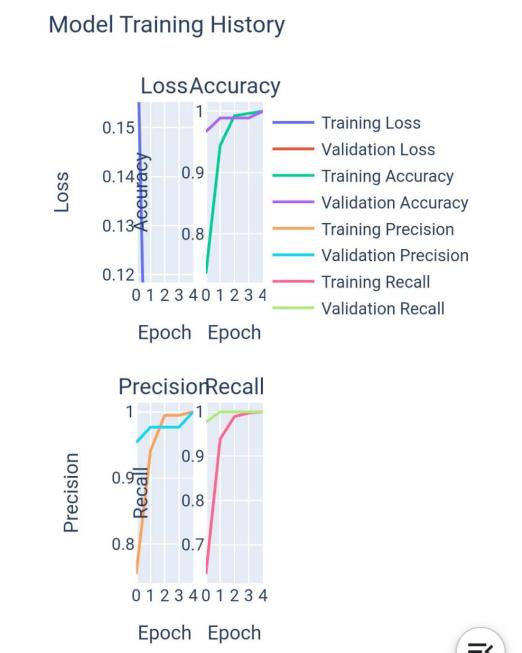
fig.update\_yaxes(title\_text='Accuracy', row=1, col=2)

fig.update\_yaxes(title\_text='Precision', row=2, col=1)

fig.update\_yaxes(title\_text='Recall', row=2, col=2)

fig.show()

OUTPUT :



test\_loss, test\_acc, test\_precision, test\_recall = model.evaluate(test\_ds, verbose = 0)

print(f"Test Loss : {test\_loss}")

print(f"Test Accuracy : {test\_acc}")

print(f"Test Precision : {test\_precision}")

print(f"Test Recall : {test\_recall}")

OUTPUT :



def make\_prediction(text, model=model):

text = np.array([text])

inputs = tokenize(text)

return np.abs(np.round(model.predict(inputs, verbose=1).logits))

for \_ in range(5):

index = np.random.randint(test\_X.shape[0])

text = test\_X[index]

real = test\_y[index]

model\_pred = make\_prediction(text)

print(f"Original Text:\n\n{text}\n\nTrue: {CLASS\_NAMES[int(real)]}\t\tPredicted: {CLASS\_NAMES[int(model\_pred)]}\n{'-'\*100}\n")

**Conclusion :**

In conclusion, IBM's use of Natural Language Processing (NLP) for fake news detection represents a promising approach to addressing the pervasive issue of misinformation and disinformation. By leveraging advanced machine learning techniques and large datasets, IBM's system has the potential to significantly improve the accuracy and efficiency of identifying fake news. However, it's essential to continuously refine and update the model to adapt to evolving techniques used by malicious actors in spreading fake news. Additionally, ensuring transparency and ethical considerations in the development and deployment of such systems is crucial to maintaining public trust. As the field of NLP and fake news detection continues to evolve, IBM's efforts represent a significant step toward combating the spread of false information in the digital age.