|  |
| --- |
|  |
| FAKE NEWS DETECTION USING NLP |
| phase 3-development 1 |

|  |
| --- |
|  |

**FAKE NEWS DETECTION USING NLP**



**Phase 2: Innovation**

Prepare seismic data, utilize python with scikit-learn or tensorflow to build regression model in fake news detection using nlp.

Our approach encompasses the following key steps:

* **Exploratory Data Analysis (EDA):** Gain insights into dataset characteristics and identify potential features for classification.
* **Text Preprocessing:** Prepare textual data for modeling through tokenization and removal of stopwords.
* **Model Development:** Utilize BERT-based classification models and fine-tune them on our dataset.
* **Model Evaluation:** Assess model performance using key metrics such as accuracy, precision, recall, and F1-score.
* **Interpretability:** Explore techniques to understand the model's decision-making process and identify important features.
* **Deployment:** Implement the trained model for automatic classification of news articles into fake or real categories.

**Set Up**

In this section, we'll guide you through the initial steps to prepare your **environment** for working with the **dataset** and building the **fake news detection model**. We will make the **required imports** and will also set some **constants & hyperparameters** which will be later used.

[In1]

%pip install transformers datasets --quiet

Note: you may need to restart the kernel to use updated packages

[In2]

*# Imports for Dataset*

import time

import numpy as np

import pandas as pd

import nltk

import string

import tensorflow as tf

from nltk.corpus import stopwords

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

nltk.download('stopwords')

*# Data Visualization*

import plotly.express as px

*# Classification Model*

from transformers import AutoTokenizer, TFAutoModelForSequenceClassification

*# Model Training*

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import ModelCheckpoint

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

[nltk\_data] Downloading package stopwords to /usr/share/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

In [3]:

*# Data set management*

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

MAPPING\_DICT = {

"Fake":0,

"Real":1

}

*# Model Callbacks*

model\_name = "BERTFakeNewsDetector"

MODEL\_CALLBACKS = [ModelCheckpoint(model\_name, save\_best\_only=True)]

**Data Loading & Pre-Processing**

Now that we have **completed our setup**, it's time to **load our dataset**. In this section, we'll take a **closer look** at **the data**, perform **essential preprocessing steps**, and gain a **better understanding** of its **structure**. Let's delve into the **data loading and preprocessing** to set the stage for our **fake news detection journey.**

In [4]:

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

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

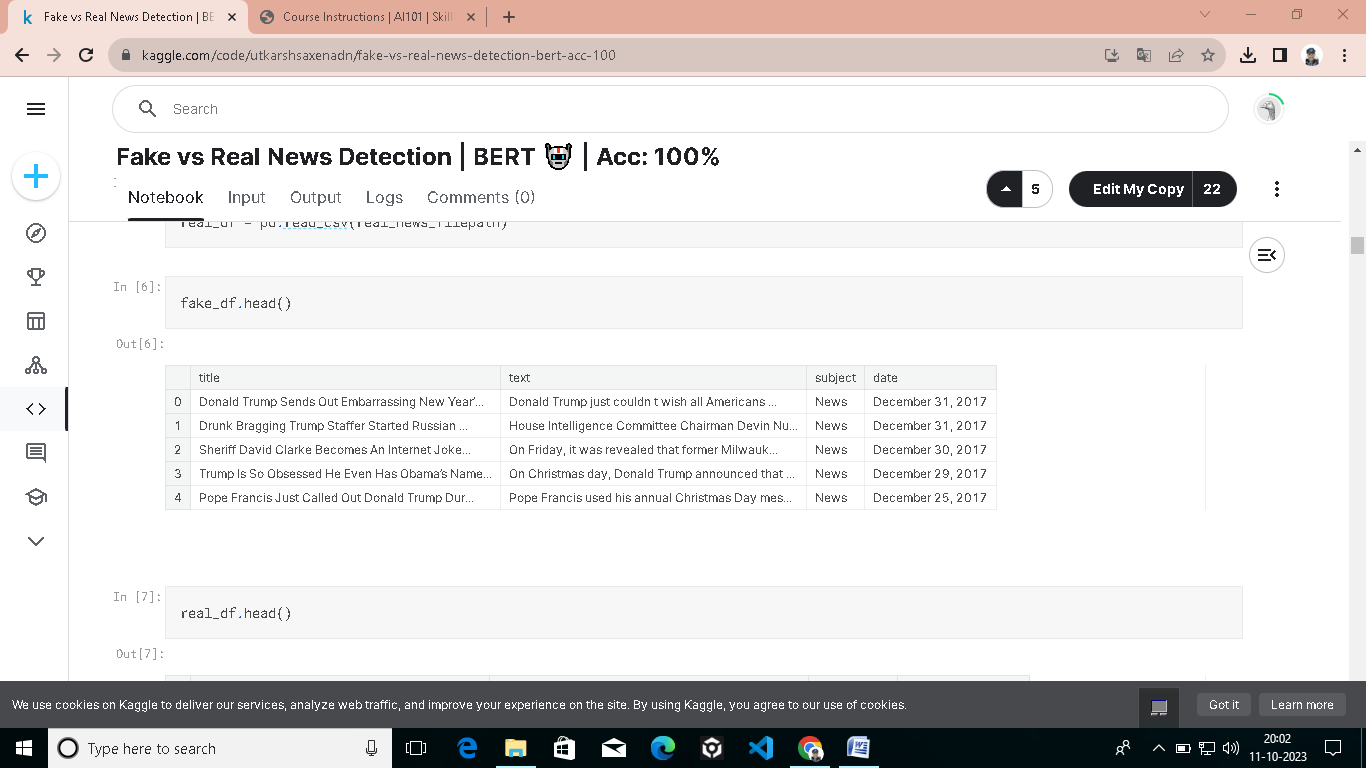
In [5]:

fake\_df = pd.read\_csv(fake\_news\_filepath)

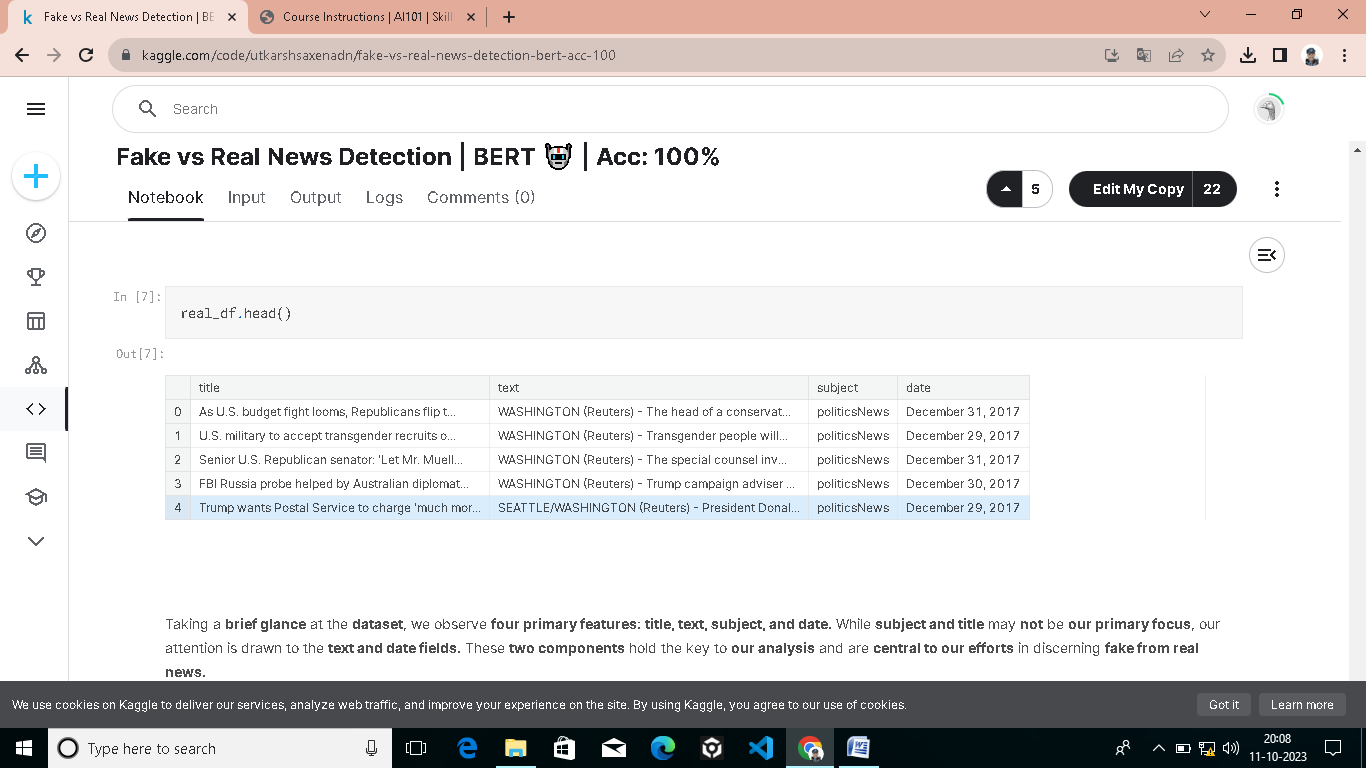
real\_df = pd.read\_csv(real\_news\_filepath)

In [6]:

fake\_df.head()

  
IN[7]:

real\_df.head()



Taking a **brief glance** at the **dataset**, we observe **four primary features**: **title, text, subject, and date**. While **subject and title** may **not** be **our primary focus**, our attention is drawn to the **text and date fields**. These **two components** hold the key to **our analysis** and are **central to our efforts** in discerning **fake from real news**

Currently, we have **two seperate data frames** for the **real data and the fake data**. Let's combine them in a **single data frame**, which will make it easier to process the information.

.In [8]:

*# Classification Labels*

real\_df["Label"] = "Real"

fake\_df["Label"] = "Fake"

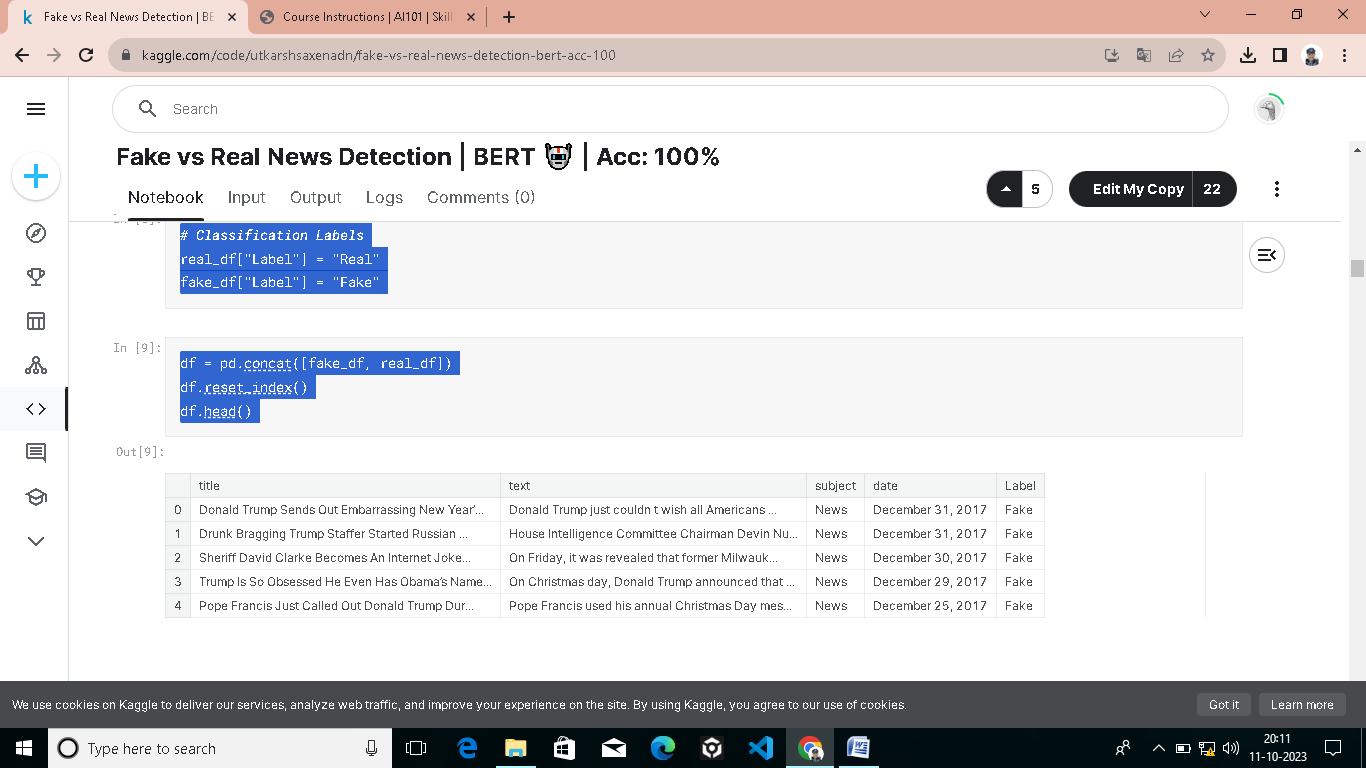
In [9]:

linkcode

df = pd.concat([fake\_df, real\_df])

df.reset\_index()

df.head()



In[10]:

print(f"Dataset Size: **{**len(df)**}**")

Dataset Size: 44898

Given the substantial size of this dataset, and considering the **limitations of memory resources**, we have opted to downsize the **dataset significantly**, capping it at **1,000 samples**.

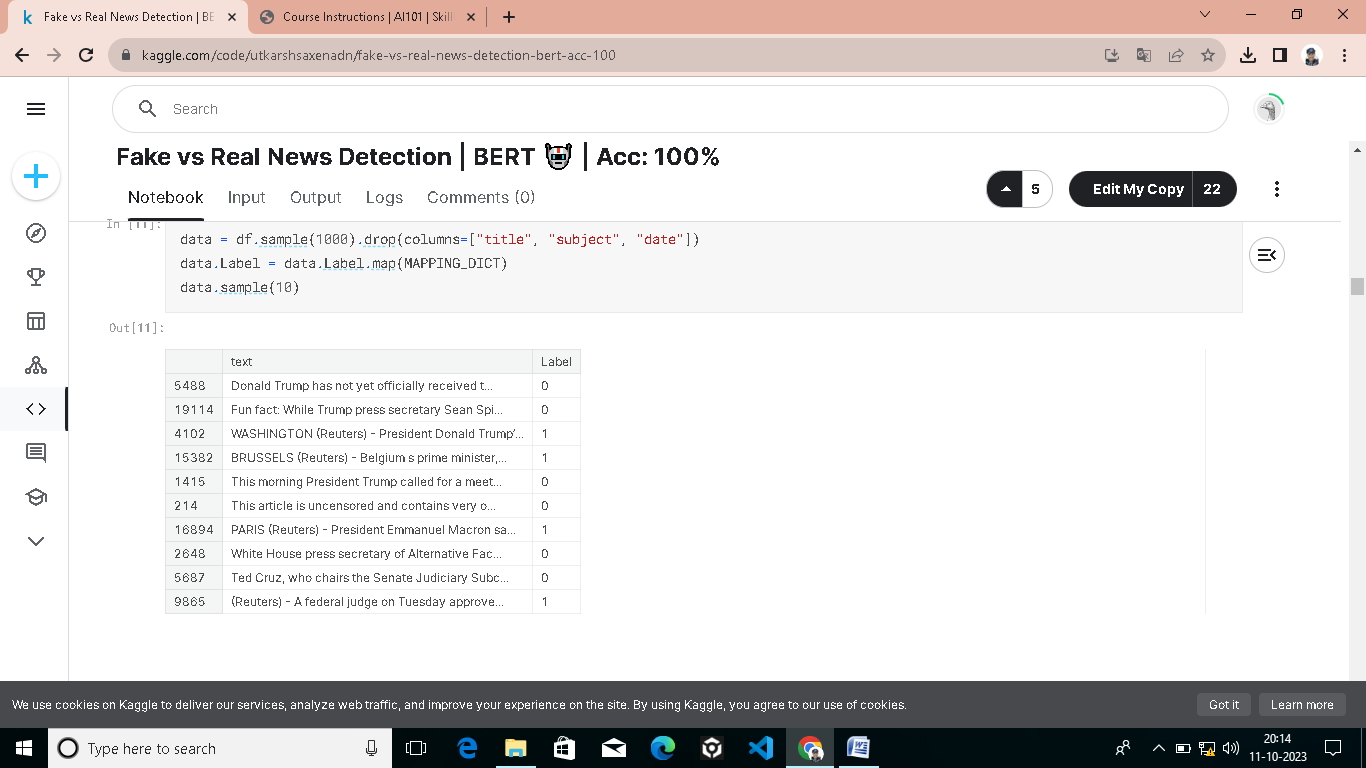
In [11]:

linkcode

data = df.sample(1000).drop(columns=["title", "subject", "date"])

data.Label = data.Label.map(MAPPING\_DICT)

data.sample(10)

**.**

The data presented above can be **valuable for visualization purposes**. However, when it comes to **model building and training**, we need a **streamlined version of the data**. Specifically, we can exclude the **title, subject, and date columns** as features for our model, focusing solely on the **essential text content**. Additionally, we'll need to convert the **categorical labels** into **numeric format** to facilitate **model training and evaluation.**

**Data Visualization:**

Before delving into the **classification task**, it's crucial to address **class imbalance**. This **initial analysis** is paramount because it can **significantly influence our model's performance**. Let's begin by assessing the **distribution of classes**, as this forms a **fundamental step** in our **model-building process**.

In [12]:

linkcode

class\_dis = px.histogram(

data\_frame = df,

y = "Label",

color = "Label",

title = "Fake & Real Samples Distribution",

text\_auto=True

)

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

class\_dis.show()

FAKE & REAL SAMPLE DISTRIBUTION

It's evident that there's a **slight class imbalance** in our dataset, with a **higher number of fake samples compared to real samples**. However, the imbalance is **relatively low**, with **approximately 23,000 samples** for **fake news** and **21,000 samples for real news**. (On original Data)

While this level of **class imbalance** is **not expected** to **significantly impact our model's performance**, we'll take a **precautionary approach** and use a **stratified split** to ensure a **balanced distribution** of classes in our **training and testing sets**. This will help us maintain **model stability and mitigate** any potential **bias introduced** by the **class distribution**.

IN[13]:

subject\_dis = px.histogram(

data\_frame = df,

x = "subject",

color = "subject",

facet\_col = "Label",

title = "Fake & Real Subject Distribution",

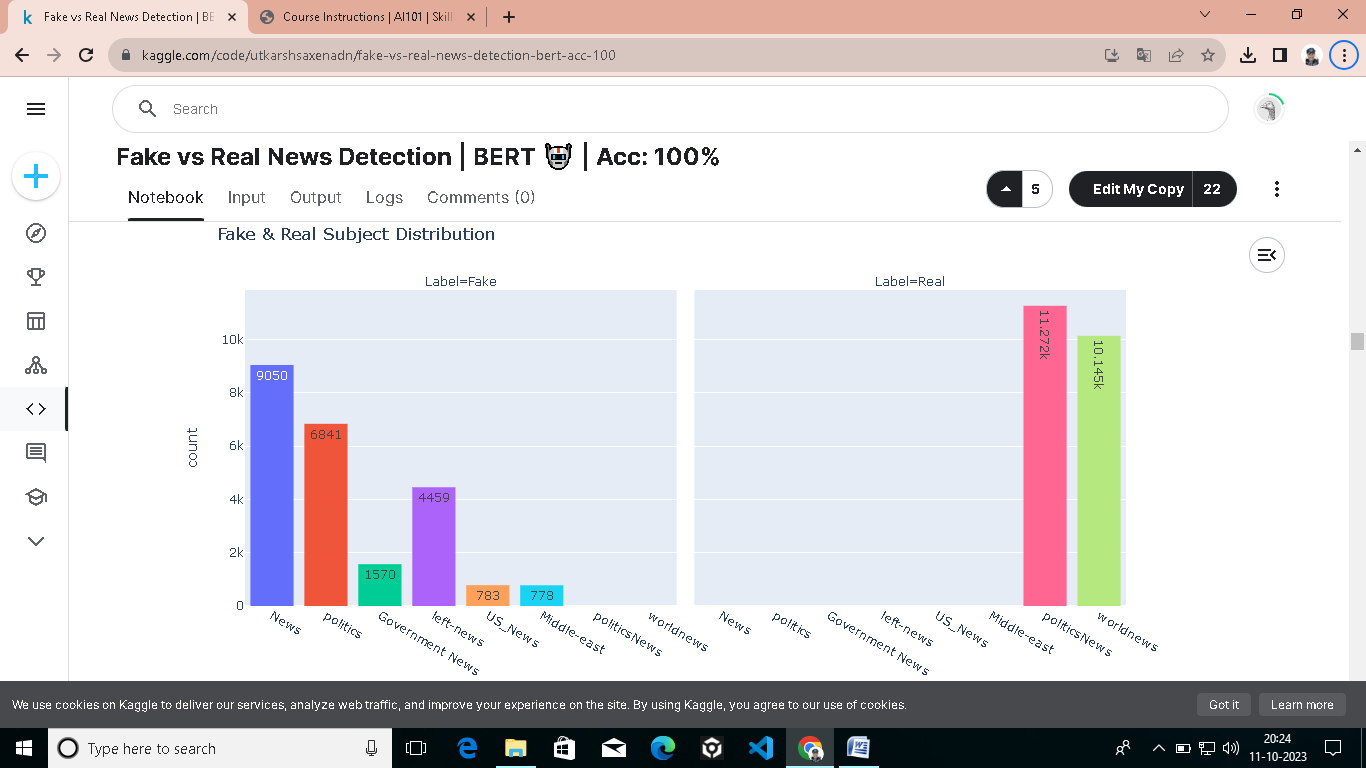
text\_auto=True

)

subject\_dis.update\_layout(showlegend=False)

subject\_dis.show()

FAKE & REAL SUBJECT DISTRIBUTION:



Indeed, the **distribution of subjects or categories** in our dataset poses a **significant challenge** for using the **'subject' column** as a **feature** for **our model**. It's clear that the **all of fake news articles fall** under various subjects such as **politics, government news, left news, US news, and the Middle East**, while **real news articles** are primarily categorized under **political news and world news**. Utilizing the **'subject' column** as a feature could lead the model to **over-rely on this information**, potentially resulting in a **biased prediction** pattern where it **simply associates real news** with these **two subjects** and **makes guesses** based on that association.

Ideally, a **more balanced distribution** of **subjects between fake and real news** would have provided a **better learning environment** for the model. However, given the **dataset's inherent structure**, it's prudent to **exclude the 'subject' column** from our **feature set** and **focus on the textual content itself**. By doing so, we allow the model to **learn from the rich linguistic features** present in the text, enabling it to make **more nuanced and accurate predictions**.

IN[14]:

list(filter(lambda x: len(x)>20, df.date.unique()))

Out[14]:

['https://100percentfedup.com/served-roy-moore-vietnamletter-veteran-sets-record-straight-honorable-decent-respectable-patriotic-commander-soldier/',

'https://100percentfedup.com/video-hillary-asked-about-trump-i-just-want-to-eat-some-pie/',

'https://100percentfedup.com/12-yr-old-black-conservative-whose-video-to-obama-went-viral-do-you-really-love-america-receives-death-threats-from-left/',

'https://fedup.wpengine.com/wp-content/uploads/2015/04/hillarystreetart.jpg',

'https://fedup.wpengine.com/wp-content/uploads/2015/04/entitled.jpg',

'MSNBC HOST Rudely Assumes Steel Worker Would Never Let His Son Follow in His Footsteps…He Couldn’t Be More Wrong [Video]']

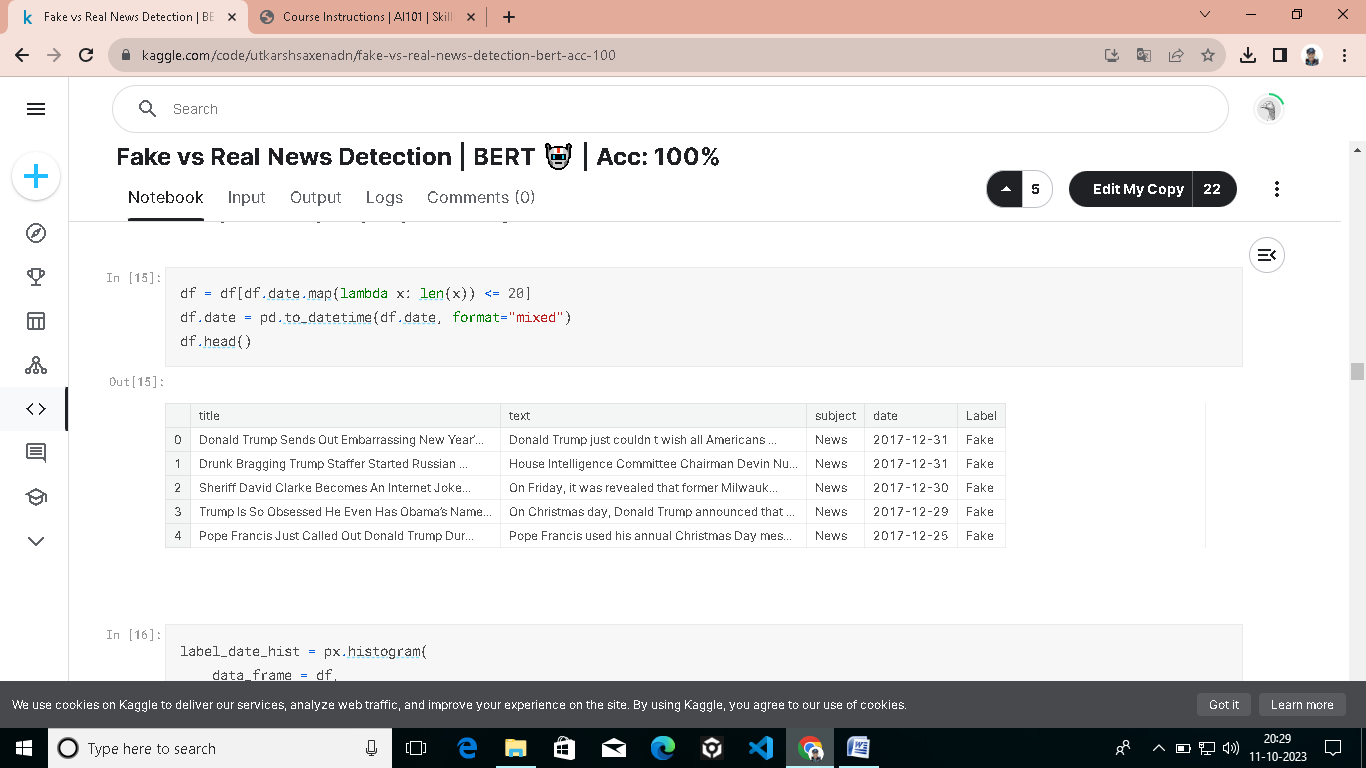
It does **seem unusual** to encounter **text and links** in a **date column**, especially in a **dataset** where one would expect the **'date' column to strictly contain date-related information**. Discovering such discrepancies highlights the **importance of data quality and integrity**. Anomalies like this can potentially **impact the accuracy and reliability of any analysis** or modeling conducted on the dataset.

In [15]:

df = df[df.date.map(lambda x: len(x)) <= 20]

df.date = pd.to\_datetime(df.date, format="mixed")

df.head()



IN[16]:

label\_date\_hist = px.histogram(

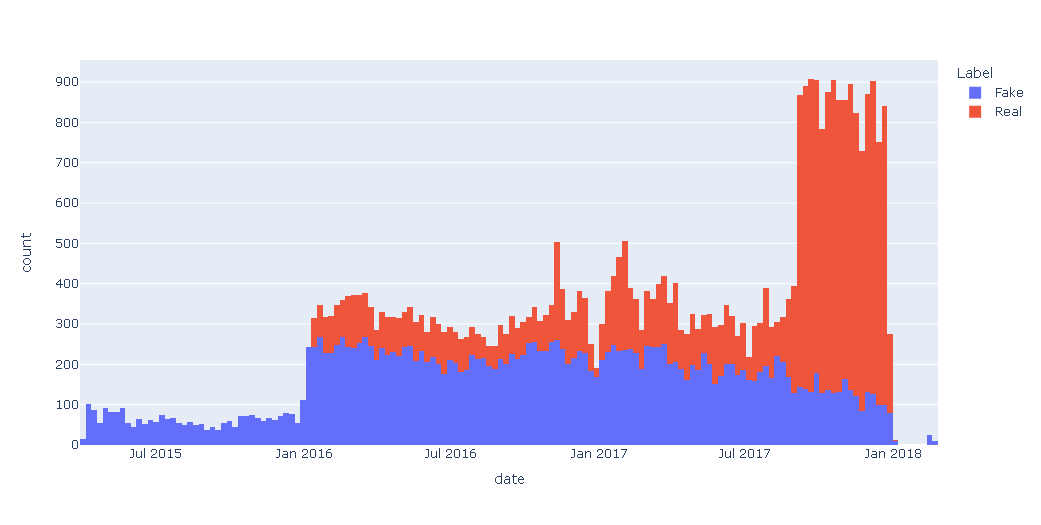
data\_frame = df,

x = 'date',

color = "Label",

)

label\_date\_hist.show()



The **observed patterns** in **data collection over time** indeed **raise intriguing questions** about the **dynamics of fake and real news** in the dataset. As you've pointed out, there are **two potential explanations for this phenomenon**.

The first explanation suggests that **real news might be more prevalent** in **recent times**. This could be due to **various factors**, including **improved government measures** to combat **fake news or a shift in public perception** and **consumption of news**. In this scenario, the **data collection reflects a real-world trend** where genuine **news articles** are on the rise.

The **second explanation** is related to **data collection strategies**. Since the dataset doesn't **start collecting both fake and real news** from the **same date**, it's possible that there are **external factors influencing** the **data collection process**. For instance, the increase in the **total number of real news articles** in **2018 might** be due to a **deliberate effort to balance the class distribution** and **reduce class imbalance** issues in the dataset. This approach can help create a **more representative dataset for machine learning purposes.**

The histogram indeed provides valuable insights into the **temporal distribution of fake and real news articles**, showcasing a decrease in **fake news** and a **significant increase in real news** as we approach **2018**.

IN[17]:

real\_sub\_hist = px.histogram(

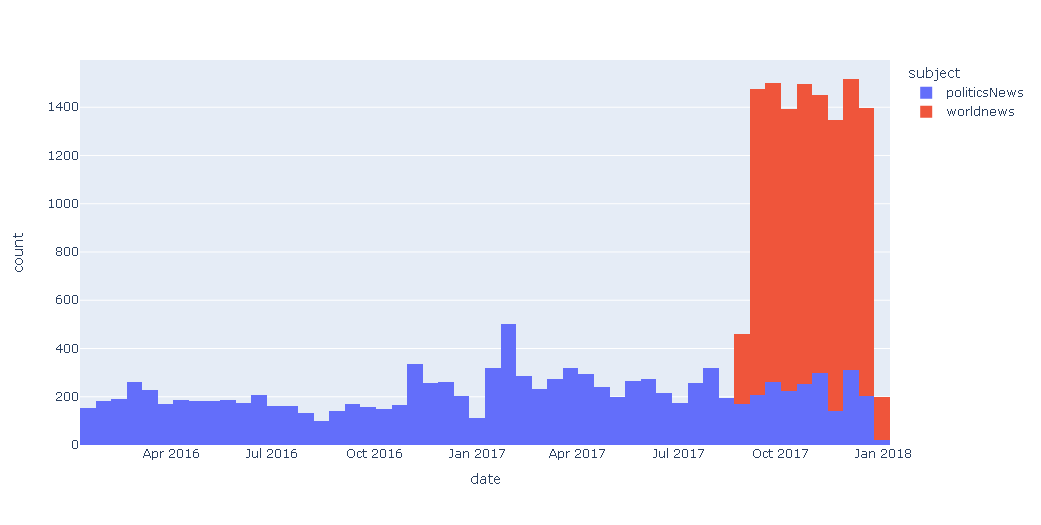
data\_frame = df[df.Label == "Real"],

x = 'date',

color = "subject",

)

real\_sub\_hist.show()



The observation that the spike in the **total number** of **real news articles** in recent years is **predominantly attributed to world news articles** collected starting in **August 2017** is an **interesting finding**.

It suggests that a **significant amount** of **world news** data, particularly from **August 2017 onwards**, was **incorporated into the dataset**. This clearly shows that the data is not collected in an **balanced manner**.

IN[18]:

subject\_hist = px.histogram(

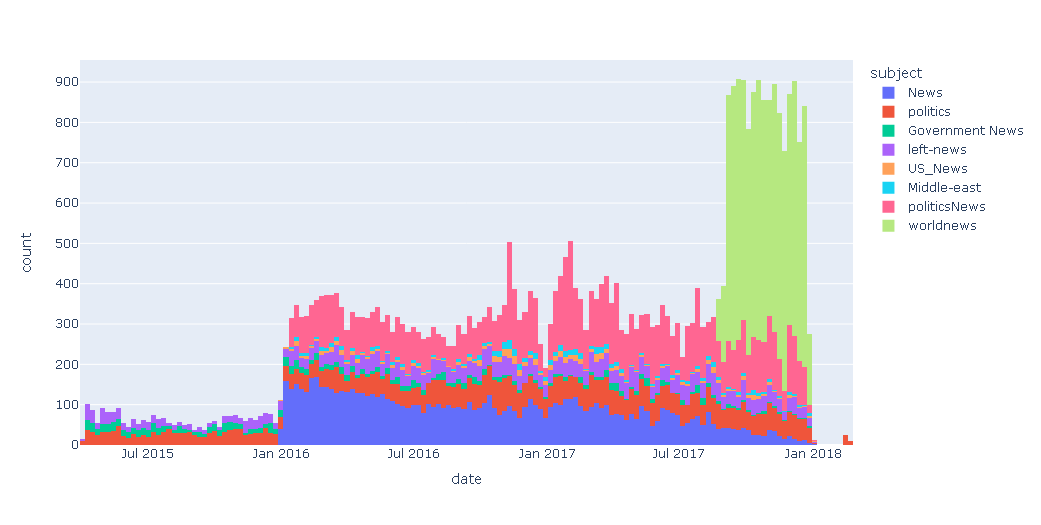
data\_frame = df,

x = 'date',

color = "subject",

)

subject\_hist.show()



In our dataset, the **distinct spikes and variations** in **news distribution** become clearer upon **closer examination**. These fluctuations can be traced back to the **timing of data collection** for **different news categories**. Notably, **'World News'** articles were **predominantly collected in the year 2017**, while other categories were **documented as far back as 2015**. Additionally, the **'Political News' category** joined the dataset in **2016**. Understanding these **temporal dynamics** is crucial for our analysis, as it sheds light on the **composition and origins of the data**, offering a structured foundation for **further exploration and modeling**

**Text Processing** :

Before proceeding to the next steps, it's essential to apply preprocessing to our data. This includes converting the text to **lowercase**, **eliminating stopwords**, and **removing any punctuation marks.**

In [19]:

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

def text\_processing(text):

words = text.lower().split()

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

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

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

return clean\_text

In [20]:

X = data.text.apply(text\_processing).to\_numpy()

Y = data.Label.to\_numpy().astype('float32').reshape(-1,1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, Y,

train\_size=0.9,

test\_size=0.1,

stratify=Y,

random\_state=42

)

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(

X\_train, y\_train,

train\_size=0.9,

test\_size=0.1,

stratify=y\_train,

random\_state=42

)

**BERT Classification Model**

**BERT**, or **Bidirectional Encoder Representations from Transformers**, is a **cutting-edge natural language processing (NLP) model** developed by **Google**. What sets **BERT** apart is its ability to **understand the context of words** in a sentence by considering both the words that come **before and after** them, allowing it to **grasp nuances**, **context**, and **meaning in language more effectively.** **BERT** has achieved **remarkable success** in various **NLP tasks**, including **text classification**, **sentiment analysis**, and **machine translation**, and it has become a **cornerstone** in the field of **AI** for **understanding and generating human language**.

Before proceeding with the loading of the **pre-trained BERT model**, a crucial step lies ahead: **tokenization** of our data. At present, our input data points remain in their **textual format**, necessitating their **transformation into tokens**. This **transformation** is essential to enable the **subsequent processing** of our data by the **BERT model.**

In [21]:

linkcode

bert\_name = "bert-base-uncased"

tokenizer = AutoTokenizer.from\_pretrained(

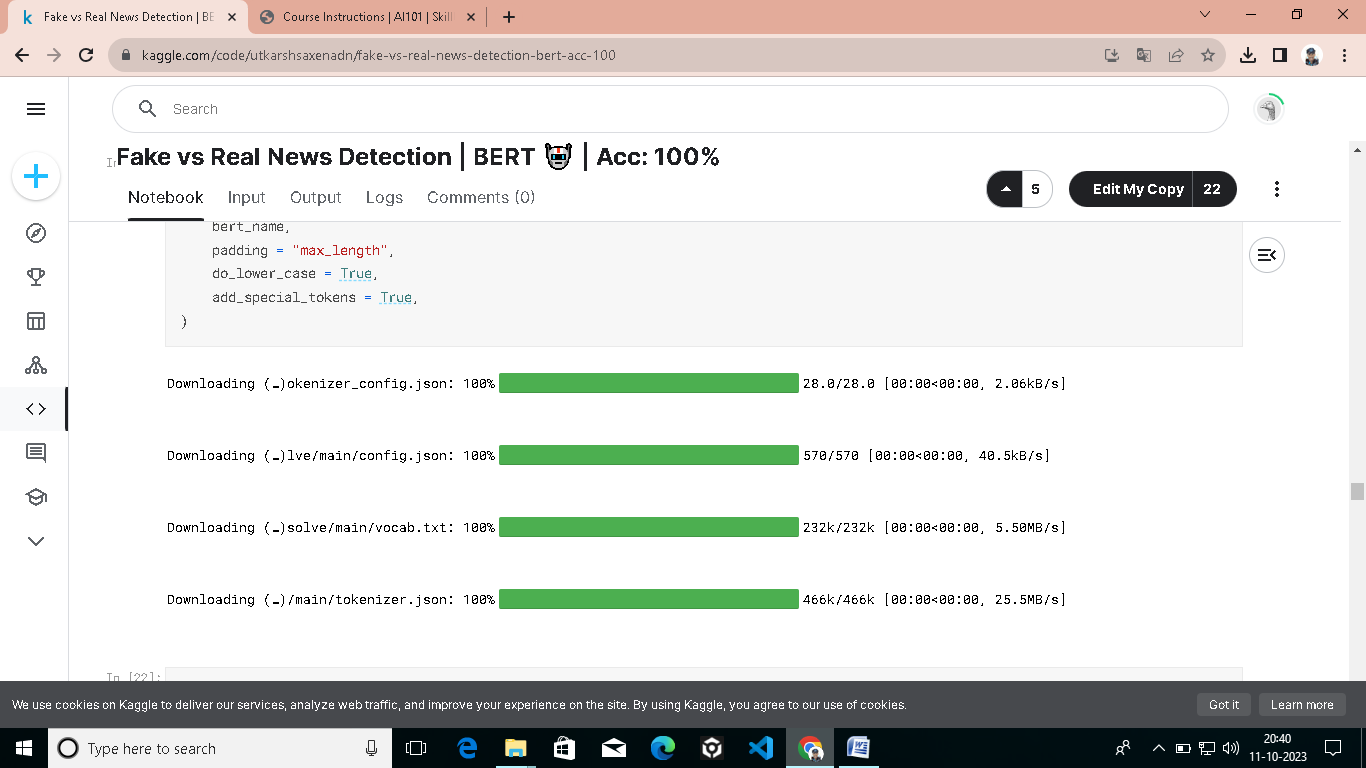
bert\_name,

padding = "max\_length",

do\_lower\_case = True,

add\_special\_tokens = True,

)



X\_train\_encoded = tokenizer(

X\_train.tolist(),

padding = True,

truncation = True,

return\_tensors = "tf"

).input\_ids

X\_valid\_encoded = tokenizer(

X\_valid.tolist(),

padding = True,

truncation = True,

return\_tensors = "tf"

).input\_ids

X\_test\_encoded = tokenizer(

X\_test.tolist(),

padding = True,

truncation = True,

return\_tensors = "tf"

).input\_ids

In [23]:

train\_ds = tf.data.Dataset.from\_tensor\_slices((X\_train\_encoded, y\_train)).shuffle(len(X\_train)).batch(8).prefetch(tf.data.AUTOTUNE)

valid\_ds = tf.data.Dataset.from\_tensor\_slices((X\_valid\_encoded, y\_valid)).shuffle(len(X\_valid)).batch(8).prefetch(tf.data.AUTOTUNE)

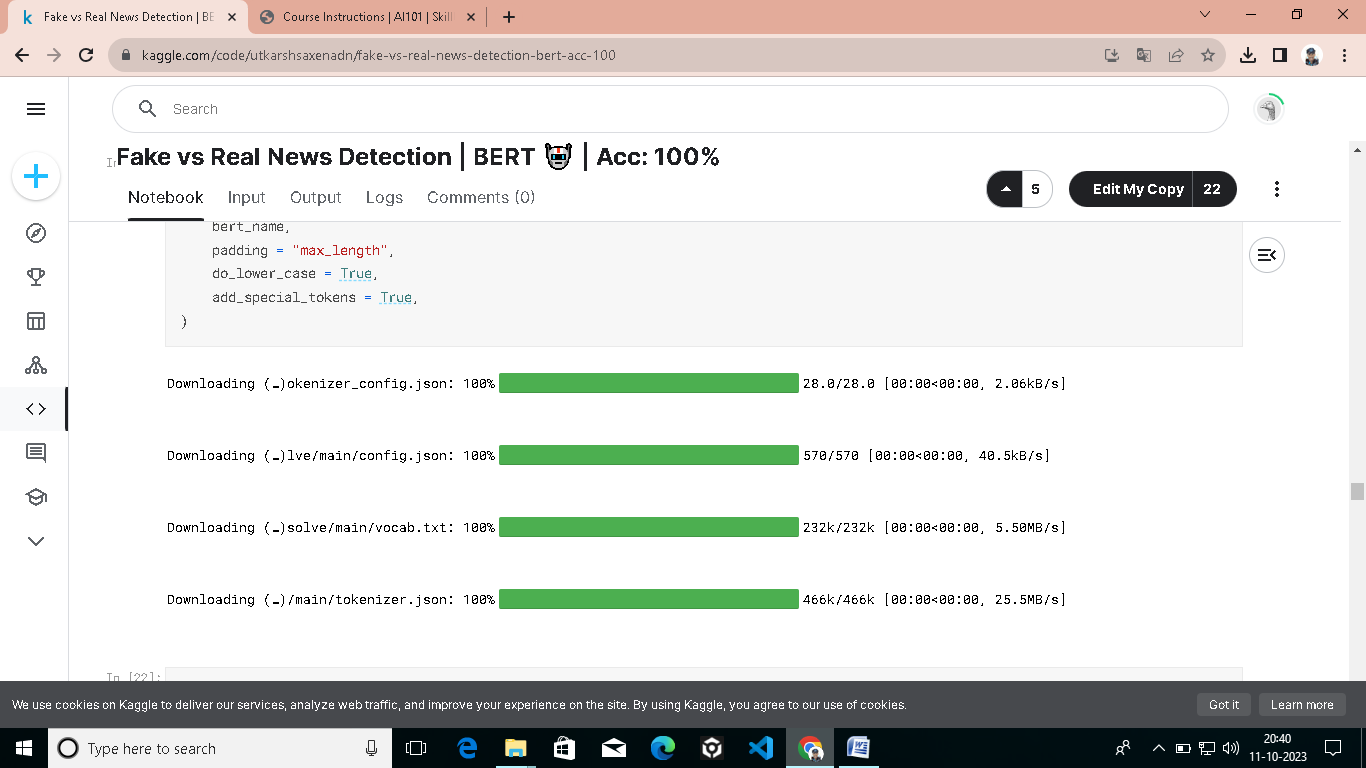
test\_ds = tf.data.Dataset.from\_tensor\_slices((X\_test\_encoded, y\_test)).shuffle(len(X\_test)).batch(8).prefetch(tf.data.AUTOTUNE)

**Fantastic!** Our data is **now fully prepared** for **ingestion by our model**. It has been **successfully tokenized** from its **original textual format**, **enabling compatibility** with the **model's processing requirements**. Additionally, the **data** has been **organized** into **manageable data flows**, **optimizing efficiency**. The next step on our journey is to **load the model**.

In [24]:

linkcode

bert\_model = TFAutoModelForSequenceClassification.from\_pretrained(bert\_name, num\_labels = 1)

Downloading model.safetensors: 100%

440M/440M [00:02<00:00, 223MB/s]

All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are newly initialized: ['classifier.weight', 'classifier.bias']

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

# **Training BERT**

In this section, **our primary focus** will be on the **training** of the **BERT model**. In addition to **monitoring the loss**, we will assess the **model's performance** using **various metrics**, including the **F1 score**, **recall score**, and **precision score**.

In [25]:

.

In [25]:

bert\_model.compile(

optimizer = Adam(learning\_rate = 1e-5),

metrics = [

tf.keras.metrics.BinaryAccuracy(name="Accuracy"),

tf.keras.metrics.Precision(name="Precision"),

tf.keras.metrics.Recall(name="Recall"),

]

)

model\_history = bert\_model.fit(

train\_ds,

validation\_data = valid\_ds,

epochs = 5,

batch\_size = 16,

callbacks = MODEL\_CALLBACKS

)

model\_history = pd.DataFrame(model\_history.history)

Epoch 1/5

102/102 [==============================] - 204s 1s/step - loss: 0.1709 - Accuracy: 0.7617 - Precision: 0.7751 - Recall: 0.6818 - val\_loss: 0.1052 - val\_Accuracy: 0.8667 - val\_Precision: 0.7959 - val\_Recall: 0.9512

Epoch 2/5

102/102 [==============================] - 151s 1s/step - loss: 0.0343 - Accuracy: 0.9753 - Precision: 0.9758 - Recall: 0.9706 - val\_loss: 0.0192 - val\_Accuracy: 0.9667 - val\_Precision: 0.9524 - val\_Recall: 0.9756

Epoch 3/5

102/102 [==============================] - 152s 1s/step - loss: 0.0079 - Accuracy: 0.9988 - Precision: 1.0000 - Recall: 0.9973 - val\_loss: 0.0048 - val\_Accuracy: 1.0000 - val\_Precision: 1.0000 - val\_Recall: 1.0000

Epoch 4/5

102/102 [==============================] - 109s 1s/step - loss: 0.0052 - Accuracy: 1.0000 - Precision: 1.0000 - Recall: 1.0000 - val\_loss: 0.0087 - val\_Accuracy: 0.9778 - val\_Precision: 0.9535 - val\_Recall: 1.0000

Epoch 5/5

102/102 [==============================] - 108s 1s/step - loss: 0.0043 - Accuracy: 1.0000 - Precision: 1.0000 - Recall: 1.0000 - val\_loss: 0.0075 - val\_Accuracy: 0.9778 - val\_Precision: 0.9535 - val\_Recall: 1.0000

In [26]:

*# Save the mdoel*

bert\_model.save(model\_name)

**Learning Curve Visualization**

Let's take a **visual journey** to explore how the **model embarked** on its **path to achieving** its **remarkable final performance.**

In [27]:

linkcode

import plotly.graph\_objs as go

from plotly.subplots import make\_subplots

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

*# Add traces to subplots*

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)

*# Customize the layout*

fig.update\_layout(

title='Model Training History',

xaxis\_title='Epoch',

yaxis\_title='Metric Value',

showlegend=False,

)

*# Update subplot axes labels*

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)

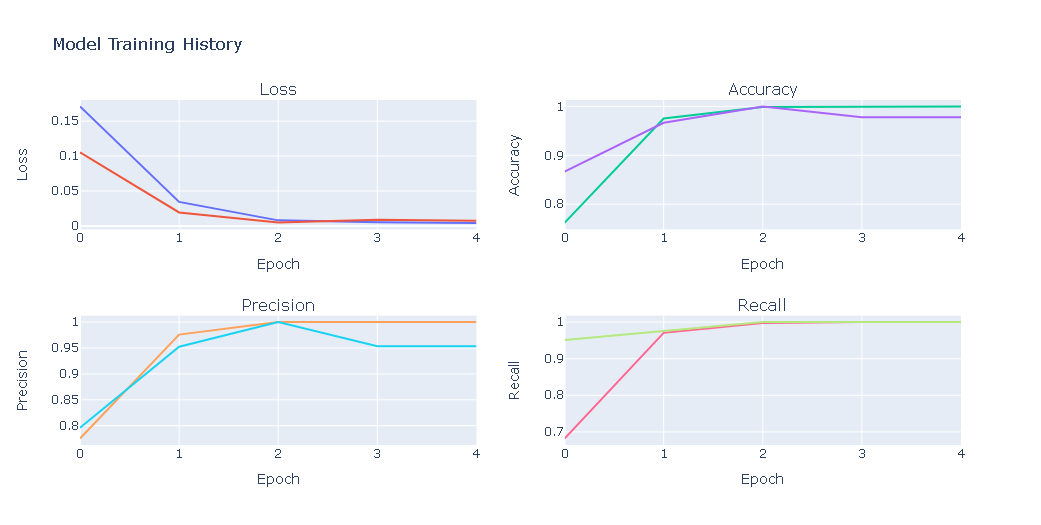
fig.update\_yaxes(title\_text='Recall', row=2, col=2)

*# Display the figure*

fig.show()

The power of a **pre-trained model** like **BERT** truly shines in its ability to **leverage vast amounts of prior knowledge** from a **large text corpus**. This **extensive understanding** of **text allows** it to **excel** in tasks such as **text classification**, like **distinguishing between fake and real news**. As we can witness, the **loss decreases almost to zero**, **precision and recall approach one**, and **accuracy soars**.

This level of **performance** is **truly remarkable**, especially considering the **dataset's size**. However, it's worth noting that **some memory constraints exist**; training on a **single GPU is impractical due to memory limitations**. This is why **I utilized two GPUs**, even though **only one is actively used**. The initialization process demands more than **30GB of memory**. Despite these **constraints**, **fine-tuning the BERT model** yields substantial **performance improvements**, making it a **powerful tool** in the realm of **natural language processing**

**Test Performance Evaluation**

Having evaluated our **model's performance** on both the **training and validation datasets**, it's crucial to now assess how well it **performs** on the **testing dataset**—a critical step in gauging its **real-world applicability and generalization capabilities.**

In [28]:

test\_loss, test\_acc, test\_precision, test\_recall = bert\_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**}**")

Test Loss : 0.0008588206837885082

Test Accuracy : 1.0

Test Precision : 1.0

Test Recall : 1.0

This model is **truly remarkable**, as it maintains its **exceptional performance** even when **tested on new, unseen data**. This level of consistency suggests that the **model possesses high accuracy** and is well-equipped to excel in **real-world scenarios**.

**Model's Prediction Samples**

Beyond the **numerical metrics**, it's beneficial to **examine the model's predictions directly**. This will provide us with a **visual insight** into the **accuracy and reliability** of the **model's classifications.**

In [29]:

def predict\_text(text, model):

tokens = tokenizer(text, return\_tensors = 'tf', padding="max\_length", truncation=True).input\_ids

return np.abs(np.round(model.predict(tokens, verbose = 0).logits))

In [30]:

for \_ **in** range(5):

index = np.random.randint(len(X\_test))

text = X\_test[index]

true = y\_test[index]

model\_pred = predict\_text(text, model = bert\_model)[0]

print(f"ORGINAL TEXT:**\n\n{**text**}\n\n**TRUE: **{**CLASS\_NAMES[int(true)]**}\t**PREDICTED: **{**CLASS\_NAMES[int(model\_pred)]**}\n{**'-'\*100**}\n**")

ORGINAL TEXT:

washington reuters president barack obama friday signed law measure pledges greater efforts protect drugdependent newborns assist parents comprehensive addiction recovery act also stresses drug treatment overdose prevention help stanch nation’s heroin opioid drug epidemic obama said statement 78 americans die opioid overdose every day noted legislation included modest steps address epidemic “i deeply disappointed republicans failed provide real resources seeking addiction treatment get care need” obama said “in fact blocked efforts democrats include 920 million treatment funding” bill passed nearly unanimously house representatives senate efforts enforce provisions protect newborns help parents come response reuters investigation last year titled “helpless hooked” new law requires federal government every state follow 2003 law routinely ignored law called states require hospitals social services report track assist drugdependent newborns families reuters found nine states following requirement children born addicted mothers including many mothers taking prescribed methadone reported hospitals required law often medical workers feared involving child protective services existing law requires cases reported social services reuters found efforts protect child help parents often limited failures came cost reuters found 110 babies since 2010 died preventable circumstances sent home families illequipped care them experts said far children likely died gone uncounted new law promises nonpunitive approach includes “safe care plans” aimed keeping newborns home parents receive additional help “this step forward vulnerable babies who due opioid dependency begin lives facing enormous challenges” said senator bob casey pennsylvania ranking democrat senate subcommittee children families “reuters’ initial reporting shined light darkness enveloped far many lives much work genuine step forward” representative john kline minnesota republican chairs house committee education workforce initiated measure said track state actions “these reforms important part broader efforts combat nation’s opioid epidemic provide vulnerable families better chance brighter future” kline said statement 2013 latest year nationwide hospital reporting 27315 babies diagnosed newborn drug withdrawal syndrome fivefold increase decade earlier reuters found one drugdependent baby born average every 19 minutes united states suffer shaking crying feeding problems battle withdrawal senator ron wyden oregon ranking democrat senate finance committee said broader addiction law “no half measure” without funding wyden cosponsored measure setting aside money substance abuse treatment parents danger losing children passed house stalled senate jim greenwood former pennsylvania congressman championed 2003 law said deaths reuters revealed represent “a national disgrace glaring failure federal state local level implement plans safe care infants” greenwood president washington dcbased biotechnology group applauded new measure “to improve health safety babies families” stephen patrick assistant professor pediatrics vanderbilt university leading researcher condition said new law “good news” added “wish funding came it”

TRUE: Real PREDICTED: Real

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

ORGINAL TEXT:

washington reuters us government could provide 80 billion aid victims hurricane harvey fraction total impact could storm texas representative pete sessions said thursday “people think federal government going pay this fact may 60 70 80 billion it’s 1 trillion impact” sessions told fox business network specify meant impact damage estimates remain preliminary

TRUE: Real PREDICTED: Real

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

ORGINAL TEXT:

donald trump arguably bizarre week history american presidency careened outrage outrage lie lie lawless action lawless action span five days starting firing fbi director james comey threw massive tantrum trumprussia collusion investigation well one respected american enough trump acting law journalist dan ratherrather took facebook page rip trump shreds can begins talking long lived many presidencies lived including richard nixon moves extraordinary disgrace donald trump presidency rather says partbut never seen week president nation behaved cavalier disregard norms institutions democracy seems like investigation expanding trump business dealings comparisons richard nixon plentiful days even seem untethered basic governance never seen many members political party rally around incompetence intemperance inanitydan rather correct donald trump makes richard nixon look like boy scout trump downright dangerous must restrained congress would jobs speak out act check president supposed be hell happy check president obama even anything lawlessthis republican party putting partisan politics political futures good republic shameful must vote congress 2018mr rather thank speaking truth power need voices like dark dire timesread entire brilliant post belowfeatured image via kirk irwingetty images siriusxm

TRUE: Fake PREDICTED: Fake

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

ORGINAL TEXT:

live wisconsin want working neighbors fund existence may need start peeing cup prove dependency government related dependency drugs governor wisconsin love em hate em kind leader conservatives love making public sector unions pay benefits liberals hate daring stand powerful organized megadonors democrat party governor walker shake things blue state wisconsin liberals gonna happy gov scott walker moving forward effort drug test food stamp recipients testing expected begin little year absent action lawmakers federal governmentwisconsin republican governor submitted plan state lawmakers drug testing ablebodied recipients state food share program state legislature object within 120 days plan go effect though take least year actual testing beginthe program necessarily massive effect however walker administration estimated october 220 food stamp recipients statewide 03 ablebodied adults would test positive first year employers jobs available need skilled workers pass drug test walker said statement rule change means people battling substance use disorders able get help need get healthy get back workforce year ago walker asked presidentelect donald trump incoming administration clear way change food stamp program overseen state largely funded federal taxpayers far happened walker spokesman said monday governor believes state proceed without federal action position authority implement rule spokesman tom evenson saidthe nowdeparted appointees president barack obama see way january 2017 right trump took white house former us official charge replacement program food stamps said testing would require change federal law law clearly allow it said kevin concannon undersecretary federal food nutrition service within us department agriculture walker office forwarded request us clear consulted legal counsels law absolutely allow it trump administration however may see issue light journal sentinel

TRUE: Fake PREDICTED: Fake

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

ORGINAL TEXT:

washington reuters us government could provide 80 billion aid victims hurricane harvey fraction total impact could storm texas representative pete sessions said thursday “people think federal government going pay this fact may 60 70 80 billion it’s 1 trillion impact” sessions told fox business network specify meant impact damage estimates remain preliminary

TRUE: Real PREDICTED: Real

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

Linkcode

Unquestionably, the **model's performance** stands as **nothing short** of an **awe-inspiring marvel**, casting **aside any shadow of doubt** that might dare to linger around its **inferences**. Its predictions **unfailingly unveil** a **realm of unmatched**, staggering **accuracy**—a testament to its **sheer brilliance and prowess.**

Concluding this notebook, I welcome your valuable suggestions and comments. Please don't hesitate to pinpoint any specific aspects or areas for improvement. Thank you for accompanying us on this journey to the end.