📰 Detecting Political Bias in Indian News Media

This project analyzes political bias in Indian news articles by scraping real-time data, cleaning and visualizing it, and training MiniLM model to detect leanings (left, right, center).

- Project By: Prashant Pramod Patil

🎯 Objective

- Detect and classify political bias in Indian news using NLP.
- Use transformer models like MiniLM for robust classification.
- Visualize patterns in headlines and sentiment across different media outlets.

🤔 Why This Project?

- India has a diverse and polarized media landscape.
- Political bias in reporting can shape public opinion.
- Analyzing it with data and ML can uncover patterns and trends.

```
#Data handling
import pandas as pd
import numpy as np


#Data visualization
import matplotlib.pyplot as plt
import seaborn as sns

#Train/test split and evaluation
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score,ConfusionMatrixDisplay

#Hugging face transformers
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
import transformers

#Hugging face datasets (for tokenization)
from datasets import Dataset


from google.colab import drive
drive.mount('/content/drive')

print("Libraries loaded successfully.")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Libraries loaded successfully.
```

Loading the data

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/NewsArticleData.csv')
print(df.shape)
df.head()
```

```
(600, 6)
```

	title	description	published_date	source	bias_text	label
0	India Ranks 5th in Global GDP List	IMF figures indicate positive trajectory.	27-06-2025	Mint	center	1
1	RBI Keeps Repo Rate Unchanged	Monetary policy committee evaluates inflation ...	24-06-2025	PTI	center	1
2	Refugees Face Statelessness Under New Policy	UN says new rules violate humanitarian princip...	11-06-2025	Scroll.in	left	0
3	SEBI Introduces New IPO Norms	Reforms aim to protect retail investors.	19-06-2025	Indian Express	center	1
4	Disability Pension Budget Slashed	Critics say social justice is being undermined.	27-06-2025	The Hindu	left	0

Next steps: Generate code with df · View recommended plots · New interactive sheet

Checking for null values

```
df.isnull().sum()
```

	0
title	0
description	0
published_date	0
source	0
bias_text	0
label	0

dtype: int64

Checking the size of each category

```
print(df['bias_text'].value_counts())
```

```
bias_text
center    197
left      197
right     195
Right       5
Center      3
Left        3
Name: count, dtype: int64
```

There are inconsistencies in spellings of bias_text so we will lower all the fields from that column to Lowercase

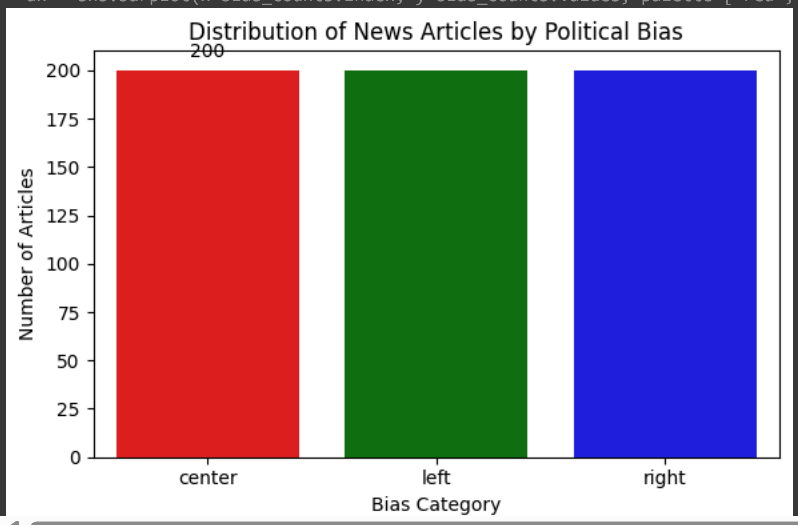
```
df['bias_text'] = df['bias_text'].str.lower()
print(df['bias_text'].value_counts())
```

```
bias_text
center    200
left      200
right     200
Name: count, dtype: int64
```

The data is well distributed among each category so there is no need for downsampling or upsampling

```
bias_counts = df['bias_text'].value_counts().sort_index()

plt.figure(figsize=(6, 4))

ax = sns.barplot(x=bias_counts.index, y=bias_counts.values, palette=["red", "green", "blue"])

ax.bar_label(ax.containers[0], padding=5, fontsize=10)

ax.set_title("Distribution of News Articles by Political Bias", fontsize=12)
ax.set_xlabel("Bias Category", fontsize=10)
ax.set_ylabel("Number of Articles", fontsize=10)

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-12-2015718283.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.barplot(x=bias_counts.index, y=bias_counts.values, palette=["red", "green", "blue"])
```



We will create a label column which will indicate the bias as (left=0, center=1, right=2)

```
label_map = {'left': 0, 'center': 1, 'right': 2}

df['label'] = df['bias_text'].map(label_map)
df.head()
print(df['label'].value_counts())
```

```
label
1    200
0    200
2    200
Name: count, dtype: int64
```

We will merge the title and description column by creating a text column.(it will help to tokenize the data)

```
df['text'] = df['title'] +" "+df['description']

#we will lower the text from text column also

df['text'] = df['text'].str.lower()
df.head()
```

	title	description	published_date	source	bias_text	label	text
0	India Ranks 5th in Global GDP List	IMF figures indicate positive trajectory.	27-06-2025	Mint	center	1	india ranks 5th in global gdp list imf figures...
1	RBI Keeps Repo Rate Unchanged	Monetary policy committee evaluates inflation ...	24-06-2025	PTI	center	1	rbi keeps repo rate unchanged monetary policy ...
2	Refugees Face Statelessness Under New Policy	UN says new rules violate humanitarian princip...	11-06-2025	Scroll.in	left	0	refugees face statelessness under new policy u...
3	SEBI Introduces New IPO Norms	Reforms aim to protect retail investors.	19-06-2025	Indian Express	center	1	sebi introduces new ipo norms reforms aim to p...
4	Disability Pension Budget Slashed	Critics say social justice is being undermined.	27-06-2025	The Hindu	left	0	disability pension budget slashed critics say ...

Next steps: Generate code with df | View recommended plots | New interactive sheet

Now we will split the data to train and test. we will only need the following columns for the model:

- text
- label

```
X = df['text']
y = df['label']

train_texts, val_texts, train_labels, val_labels = train_test_split(
    df['text'], df['label'], test_size=0.2, random_state=42, stratify=df['label']
)
print("Training examples: ",train_texts.shape)
print("Test examples: ",val_texts.shape)
```

```
Training examples:  (480,)
Test examples:  (120,)
```

Tokenize Text Using MiniLM Tokenizer

We'll use Hugging Face's AutoTokenizer to tokenize the input text using the MiniLM model

```
model_checkpoint = "microsoft/MiniLM-L12-H384-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)

#Tokenize train and Test sets
def tokenize(batch):
    return tokenizer(batch, padding='max_length', truncation=True, max_length=128)

train_texts = list(train_texts)
val_texts = list(val_texts)

#Apply tokenization
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=128)
val_encodings = tokenizer(val_texts, truncation=True, padding=True, max_length=128)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%  2.00/2.00 [00:00<00:00, 191B/s]
config.json: 100%  385/385 [00:00<00:00, 45.2kB/s]
vocab.txt:  232k/? [00:00<00:00, 10.4MB/s]
special_tokens_map.json: 100%  112/112 [00:00<00:00, 8.04kB/s]
```

Convert Tokenized Data into Hugging Face Dataset Format

Because Hugging face trainer expects data in a special format to train the model

```
#Creating training dataset
train_dataset = Dataset.from_dict({
    'input_ids': train_encodings['input_ids'],
    'attention_mask': train_encodings['attention_mask'],
    'label': train_labels
})

#Creating test dataset
val_dataset = Dataset.from_dict({
    'input_ids': val_encodings['input_ids'],
    'attention_mask': val_encodings['attention_mask'],
    'label': val_labels
})
```

Load MiniLM Model & Define Training Arguments

```
# Load MiniLM model with 3 output labels for bias classification
model = AutoModelForSequenceClassification.from_pretrained(
    "microsoft/MiniLM-L12-H384-uncased",
    num_labels=3
)

training_args = TrainingArguments(
    output_dir="./results",              # Where to save model checkpoints
    num_train_epochs=4,                  # Number of epochs
    per_device_train_batch_size=16,      # Training batch size
    per_device_eval_batch_size=64,       # Evaluation batch size
    weight_decay=0.01,                   # Regularization
    logging_dir="./logs",                # Logging directory
    logging_strategy="epoch",           # Log once per epoch
    logging_steps=1,
    report_to="none"
)

# Initializing Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at microsoft/MiniLM-L12-H384-uncased and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

Training the model

```
trainer.train()
```

[120/120 00:12, Epoch 4/4]

Step	Training Loss
30	0.999100
60	0.493500
90	0.259700
120	0.156900

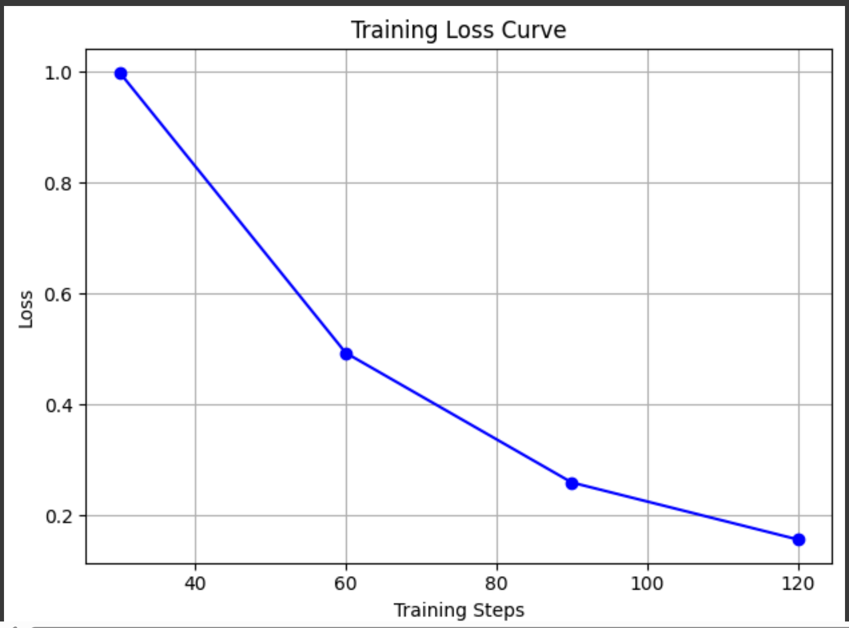
```
TrainOutput(global_step=120, training_loss=0.47728791236877444, metrics={'train_runtime': 12.5203, 'train_samples_per_second': 153.351, 'train_steps_per_second': 9.584, 'total_flos': 6175679328000.0,
'train_loss': 0.47728791236877444, 'epoch': 4.0})
```

This shows how well the model learned from the data

How to Interpret:

- Decreasing line = the model is learning
- Flat line = no further learning
- Upward trend = possible overfitting or unstable training

```
# Extract loss values from training logs
logs = trainer.state.log_history

train_steps = [log["step"] for log in logs if "loss" in log]
train_loss = [log["loss"] for log in logs if "loss" in log]

plt.plot(train_steps, train_loss, marker='o', linestyle='-', color='blue')
plt.title("Training Loss Curve")
plt.xlabel("Training Steps")
plt.ylabel("Loss")
plt.grid(True)
plt.tight_layout()
plt.show()
```



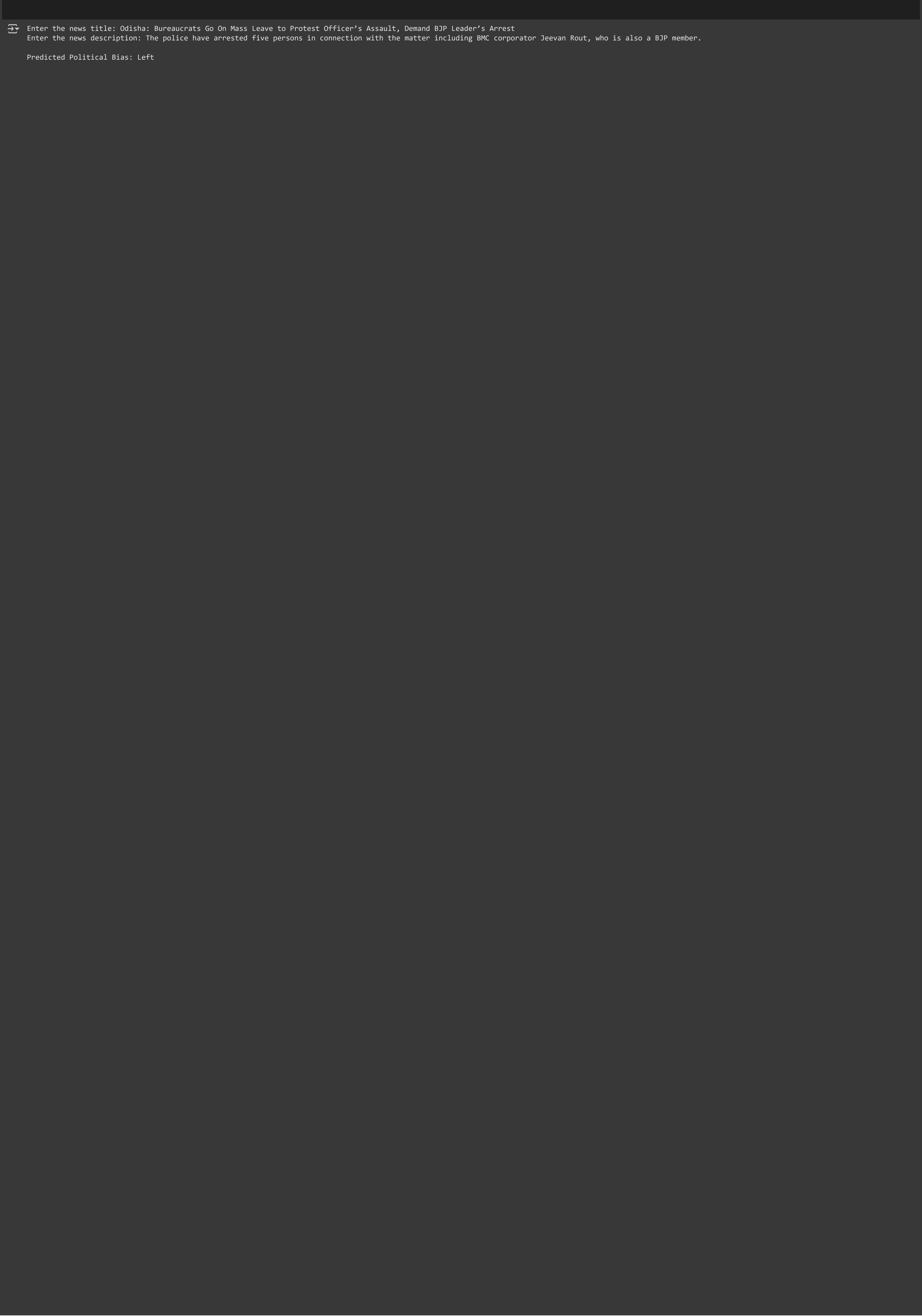
Model Accuracy on Test data

```
predictions = trainer.predict(val_dataset)

preds = predictions.predictions.argmax(axis=1)
true = predictions.label_ids

accuracy = accuracy_score(true, preds)
print("test Accuracy:", round(accuracy * 100, 2), "%")

#the reason of 100% accuracy can be repeated fields as i collected data of the same news from different sources. so the wording can be same
```

```
test Accuracy: 100.0 %
```

Simple prediction model

```
def predict_bias(title, description):

    input_text = title + ". " + description


    inputs = tokenizer(
        input_text,
        truncation=True,
        padding=True,
        max_length=128,
        return_tensors="pt"
    ).to(model.device)


    outputs = model(**inputs)
    predicted_label = outputs.logits.argmax(dim=1).item()


    label_map = {0: "Left", 1: "Center", 2: "Right"}
    return label_map[predicted_label]

title_input = input("Enter the news title: ")
description_input = input("Enter the news description: ")

predicted_bias = predict_bias(title_input, description_input)
print("\nPredicted Political Bias:", predicted_bias)
```



```
Enter the news title: Odisha: Bureaucrats Go On Mass Leave to Protest Officer’s Assault, Demand BJP Leader’s Arrest
Enter the news description: The police have arrested five persons in connection with the matter including BMC corporator Jeevan Rout, who is also a BJP member.

Predicted Political Bias: Left
```