```
pip install evaluate
```

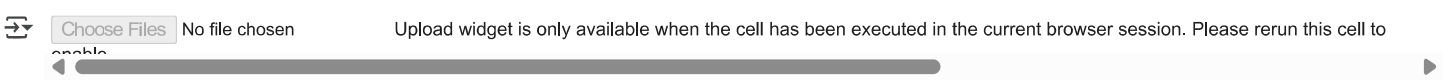
```
Collecting evaluate
  Downloading evaluate-0.4.3-py3-none-any.whl.metadata (9.2 kB)
Collecting datasets>=2.0.0 (from evaluate)
  Downloading datasets-3.5.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from evaluate) (2.0.2)
Collecting dill (from evaluate)
  Downloading dill-0.4.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from evaluate) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.11/dist-packages (from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.11/dist-packages (from evaluate) (4.67.1)
Collecting xxhash (from evaluate)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess (from evaluate)
  Downloading multiprocess-0.70.18-py311-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: fsspec>=2021.05.0 in /usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.05.0->evaluate)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from evaluate) (0.30.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from evaluate) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0->evaluate) (3.18.0)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0->evaluate) (18.1.0)
Collecting dill (from evaluate)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Collecting multiprocess (from evaluate)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec>=2021.05.0 (from fsspec[http]>=2021.05.0->evaluate)
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0->evaluate) (3.11.15)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0->evaluate) (6.0.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.7.0->ev
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate) (2.3.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate) (2025.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->evaluate) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->evaluate) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->evaluate) (2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->eva
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->evaluate) (25
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->evaluat
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets>=2.0.0->evaluate)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->evaluate) (1
Downloading evaluate-0.4.3-py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.0/84.0 kB 8.8 MB/s eta 0:00:00
Downloading datasets-3.5.0-py3-none-any.whl (491 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.2/491.2 kB 37.1 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 12.3 MB/s eta 0:00:00
Downloading fsspec-2024.12.0-py3-none-any.whl (183 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.9/183.9 kB 18.4 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 14.4 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 20.0 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets, evaluate
  Attempting uninstall: fsspec
```

```
pip install --upgrade datasets
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-packages (3.5.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from datasets) (3.18.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.0.2)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (from datasets) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages (from datasets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in /usr/local/lib/python3.11/dist-packages (from fsspec[http]<=2024.12.0,>=2
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.15)
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.30.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
```

```
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.19.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0->data
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (2025.1.3
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17
```

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
df = pd.read_csv("news_articles.csv")
```



USING DISTILBERT FOR SEQUENTIAL CLASSIFICATION

```
import pandas as pd
import torch
import numpy as np
from sklearn.model_selection import train_test_split
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, Trainer, TrainingArguments
from torch.utils.data import Dataset
from evaluate import load
```

```
# Convert labels to binary (1 = Real, 0 = Fake)
df['label'] = df['label'].apply(lambda x: 1 if str(x).lower() == 'real' else 0)

# Combine title and text for better context
df['content'] = df['title'] + " " + df['text']

# Split dataset
train_texts = list(df['content'].astype(str))
train_labels = list(df['label'])

test_texts = list(df['content'].astype(str))
test_labels = list(df['label'])


# Load tokenizer
tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")

# Tokenize texts
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=512)
test_encodings = tokenizer(test_texts, truncation=True, padding=True, max_length=512)

# Define PyTorch Dataset
class NewsDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels
    def __len__(self):
        return len(self.labels)
    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx], dtype=torch.long)  # Ensure correct dtype
        return item

train_dataset = NewsDataset(train_encodings, train_labels)
test_dataset = NewsDataset(test_encodings, test_labels)

# Define GPU usage
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load model
model = DistilBertForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2).to(device)
```

```
# Define training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,  # Reduced batch size for faster training
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    load_best_model_at_end=True,
)

# Load evaluation metric
metric = load("accuracy")

# Define compute_metrics function
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics,
)

# Train model
trainer.train()

# Evaluate model
eval_result = trainer.evaluate()
print("Evaluation Results:", eval_result)

# Save model
model.save_pretrained("fake_news_model")
tokenizer.save_pretrained("fake_news_model")

# Inference function
def predict_fake_news(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512).to(device)
    with torch.no_grad():
        outputs = model(**inputs)
    prediction = torch.argmax(outputs.logits).item()
    return "Real" if prediction == 1 else "Fake"

# Test on a sample
sample_text = "Breaking: Scientists discover a new planet similar to Earth!"
print("Prediction:", predict_fake_news(sample_text))
```

```
/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 secre
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%  48.0/48.0 [00:00<00:00, 4.63kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 1.29MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 25.0MB/s]
config.json: 100%  483/483 [00:00<00:00, 53.7kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better perfc
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to r
model.safetensors: 100%  268M/268M [00:00<00:00, 309MB/s]
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are ne
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWarning: `evaluation_strategy` is deprecated and will
  warnings.warn(
Downloading builder script: 100%  4.20k/4.20k [00:00<00:00, 444kB/s]
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please spe
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ·····
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: sadiqsahilbaigmogal2004 (sadiqsahilbaigmogal2004-vit) to https://api.wandb.ai. Use `wandb login --relogir
Tracking run with wandb version 0.19.8
Run data is saved locally in /content/wandb/run-20250403_192529-a8qkenpx
Syncing run ./results to Weights & Biases (docs)
View project at https://wandb.ai/sadiqsahilbaigmogal2004-vit/huggingface
View run at https://wandb.ai/sadiqsahilbaigmogal2004-vit/huggingface/runs/a8qkenpx
[786/786 06:42, Epoch 3/3]
```

Epoch	Training Loss	Validation Loss	Accuracy
1	0.562400	0.425732	0.819656
2	0.480500	0.234938	0.907920
3	0.169700	0.143192	0.952290

```
[262/262 00:30]
Evaluation Results: {'eval_loss': 0.14319203794002533, 'eval_accuracy': 0.9522900763358778, 'eval_runtime': 30.1762, 'eval_samples_per_s
Prediction: Fake
```

USING SVM FOR SEQUENTIAL CLASSIFICATION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,accuracy_score


# Step 2: Check label distribution

# Step 3: Encode labels ('Fake' = 0, 'Real' = 1)
df = df[df['label'].isin(['Fake', 'Real'])]

# Encode labels
label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])


# Step 4: Stratified train-test split
X_train, X_test, y_train, y_test = train_test_split(
    df['text'], df['label_encoded'],
    test_size=0.2,
    random_state=42
)
```

```
# Confirm label balance in train set)

# Combine X_train and y_train into a single DataFrame
train_df = pd.DataFrame({'text': X_train, 'label': y_train}).dropna()
test_df = pd.DataFrame({'text': X_test, 'label': y_test}).dropna()

X_train = train_df['text']
y_train = train_df['label']
X_test = test_df['text']
y_test = test_df['label']

# Step 5: TF-IDF vectorization
tfidf = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Step 6: Train SVM model
svm_model = LinearSVC()
svm_model.fit(X_train_tfidf, y_train)

# Step 7: Predict and evaluate
y_pred = svm_model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

```
<ipython-input-2-6a0f35e32ad5>:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df['label_encoded'] = label_encoder.fit_transform(df['label'])
              precision    recall  f1-score   support

        Fake       0.77      0.86      0.81       248
        Real       0.74      0.60      0.66       162

    accuracy                           0.76       410
   macro avg       0.75      0.73      0.74       410
weighted avg       0.76      0.76      0.75       410

Accuracy: 0.7585
```

USING SVM AND RANDOM FOREST AS ENSEMBLE MODEL

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,accuracy_score


# Step 2: Filter valid labels
df = df[df['label'].isin(['Fake', 'Real'])]

# Step 3: Drop rows with missing text
df.dropna(subset=['text', 'label'], inplace=True)

# Step 4: Encode labels ('Fake'=0, 'Real'=1)
label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])

# Step 5: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    df['text'], df['label_encoded'],
    test_size=0.2, random_state=42, stratify=df['label_encoded']
)

# Step 6: TF-IDF vectorization
tfidf = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = tfidf.fit_transform(X_train)
```

```
X_test_tfidf = tfidf.transform(X_test)

# Step 7: Define individual models
svm = LinearSVC()
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Step 8: Combine models using VotingClassifier
ensemble_model = VotingClassifier(
    estimators=[
        ('svm', svm),
        ('rf', rf)
    ],
    voting='hard'  # can use 'soft' if using probabilistic models like Logistic Regression
)

# Step 9: Train ensemble model
ensemble_model.fit(X_train_tfidf, y_train)

# Step 10: Evaluate model
y_pred = ensemble_model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

```
              precision    recall  f1-score   support

        Fake       0.72      0.98      0.83       259
        Real       0.90      0.36      0.51       151

    accuracy                           0.75       410
   macro avg       0.81      0.67      0.67       410
weighted avg       0.79      0.75      0.71       410

Accuracy: 0.7488
```

USING ONLY XG-BOOST

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
from xgboost import XGBClassifier


# Step 2: Remove rows with missing text or labels
df = df.dropna(subset=['text', 'label'])

# Step 3: Label encoding
label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])

# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    df['text'], df['label_encoded'],
    test_size=0.2, stratify=df['label_encoded'], random_state=42
)

# Step 5: TF-IDF vectorization
tfidf = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Step 6: XGBoost Classifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_model.fit(X_train_tfidf, y_train)

# Step 7: Predict & evaluate
y_pred = xgb_model.predict(X_test_tfidf)

# Step 8: Classification Report
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))

# Step 9: Accuracy
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [07:42:32] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
              precision    recall  f1-score   support

        Fake       0.79      0.83      0.81       259
        Real       0.68      0.62      0.65       151

    accuracy                           0.75       410
   macro avg       0.73      0.72      0.73       410
weighted avg       0.75      0.75      0.75       410

Accuracy: 0.7512
```

XGBOOST AND NAIVE BAYES USED AS ENSEMBLE MODEL

```
pip install xgboost scikit-learn pandas
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.11/dist-packages (2.1.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.0.2)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.21.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from xgboost) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder

# Step 1: Load and clean data
#df = pd.read_csv('/mnt/data/news_articles.csv')
df = df.dropna(subset=['text', 'label'])

# Step 2: Encode labels
label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])

# Step 3: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    df['text'], df['label_encoded'],
    test_size=0.2,
    stratify=df['label_encoded'],
    random_state=42
)

# Step 4: TF-IDF vectorization
tfidf = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Step 5: Define base models
nb_model = MultinomialNB()
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Step 6: Ensemble with Voting Classifier (soft voting)
ensemble_model = VotingClassifier(
    estimators=[
        ('naive_bayes', nb_model),
        ('xgboost', xgb_model)
    ],
    voting='soft'  # soft voting uses predict_proba
```

```
)

# Step 7: Train ensemble
ensemble_model.fit(X_train_tfidf, y_train)

# Step 8: Predict & evaluate
y_pred = ensemble_model.predict(X_test_tfidf)

# Step 9: Output
print("Classification Report:\n", classification_report(y_test, y_pred, target_names=label_encoder.classes_))
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [07:48:34] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Classification Report:
              precision    recall  f1-score   support

        Fake       0.75      0.95      0.83       259
        Real       0.83      0.45      0.58       151

    accuracy                           0.76       410
   macro avg       0.79      0.70      0.71       410
weighted avg       0.78      0.76      0.74       410

Accuracy: 0.7634
```

USING LSTM AND CUSTOM EMBEDDINGS FOR SEQUENTIAL DATA

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

# Step 1: Load and clean data
#df = pd.read_csv('/mnt/data/news_articles.csv')
df = df.dropna(subset=['text', 'label'])

# Step 2: Label encoding
label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])

# Step 3: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    df['text'], df['label_encoded'], test_size=0.2,
    stratify=df['label_encoded'], random_state=42
)

# Step 4: Tokenization and padding
max_words = 10000
max_len = 300

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(X_train_seq, maxlen=max_len)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len)

# Step 5: Build model with trainable embeddings
model = Sequential([
    Embedding(input_dim=max_words, output_dim=100, input_length=max_len),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Step 6: Train
model.fit(X_train_pad, y_train, batch_size=32, epochs=5, validation_split=0.1)

# Step 7: Evaluate
y_pred_prob = model.predict(X_test_pad)
y_pred = (y_pred_prob > 0.5).astype("int32")

print("Classification Report:\n", classification_report(y_test, y_pred, target_names=label_encoder.classes_))
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
```

```
<ipython-input-3-dfd132960055>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  df['label_encoded'] = label_encoder.fit_transform(df['label'])
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
Epoch 1/5
47/47 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step - accuracy: 0.6099 - loss: 0.6702 - val_accuracy: 0.7012 - val_loss: 0.5886
Epoch 2/5
47/47 ━━━━━━━━━━━━━━━━━━━━ 47s 1s/step - accuracy: 0.7654 - loss: 0.4834 - val_accuracy: 0.6707 - val_loss: 0.6423
Epoch 3/5
47/47 ━━━━━━━━━━━━━━━━━━━━ 78s 933ms/step - accuracy: 0.9267 - loss: 0.2213 - val_accuracy: 0.6951 - val_loss: 0.8183
Epoch 4/5
47/47 ━━━━━━━━━━━━━━━━━━━━ 82s 932ms/step - accuracy: 0.9673 - loss: 0.1060 - val_accuracy: 0.6951 - val_loss: 0.9000
Epoch 5/5
47/47 ━━━━━━━━━━━━━━━━━━━━ 82s 941ms/step - accuracy: 0.9845 - loss: 0.0563 - val_accuracy: 0.7134 - val_loss: 0.8666
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 182ms/step
Classification Report:
              precision    recall  f1-score   support

        Fake       0.78      0.79      0.78       259
        Real       0.63      0.61      0.62       151

    accuracy                           0.72       410
   macro avg       0.70      0.70      0.70       410
weighted avg       0.72      0.72      0.72       410

Accuracy: 0.7220
```