```
from google.colab import files
```

```
import pandas as pd

df = pd.read_csv("/content/Fake.csv")  # Use uploaded file name
df.head()
```

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | Donald Trump Sends Out Embarrassing New Year'... | Donald Trump just couldn t wish all Americans ... | News | December 31, 2017 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | News | December 31, 2017 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwauk... | News | December 30, 2017 |
| 3 | Trump Is So Obsessed He Even Has Obama's Name... | On Christmas day, Donald Trump announced that ... | News | December 29, 2017 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | News | December 25, 2017 |

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

```
df.info()
df.describe()
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9826 entries, 0 to 9825
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   title    9826 non-null   object
 1   text     9825 non-null   object
 2   subject  9825 non-null   object
 3   date     9825 non-null   object
dtypes: object(4)
memory usage: 307.2+ KB
Index(['title', 'text', 'subject', 'date'], dtype='object')
```

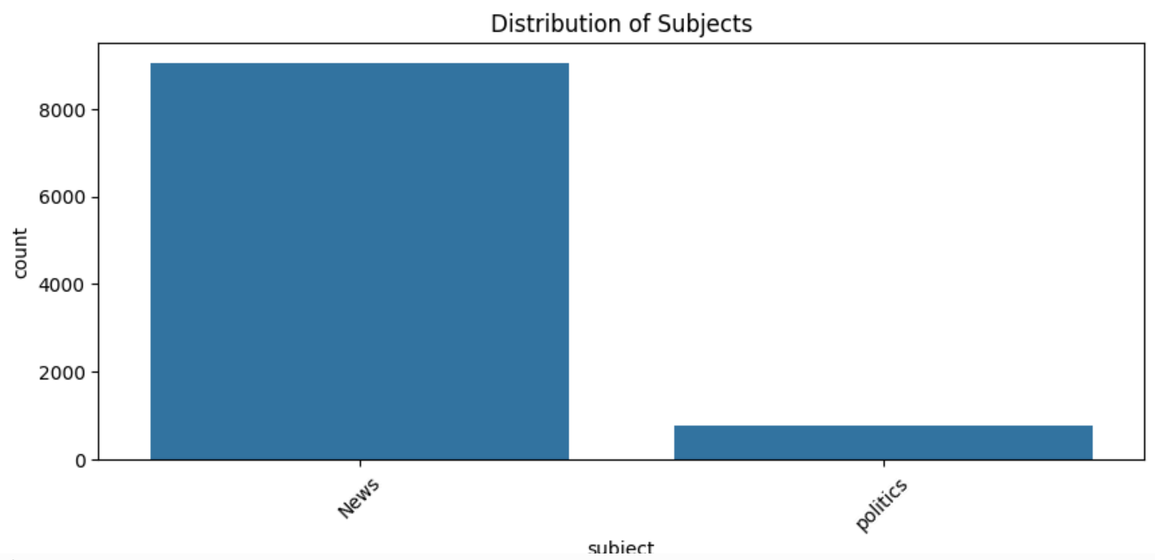
```
print("Missing values:\n", df.isnull().sum())
print("\nDuplicates:", df.duplicated().sum())
```

```
Missing values:
 title      0
text       1
subject    1
date       1
dtype: int64

Duplicates: 0
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,4))
sns.countplot(x='subject', data=df)
plt.title("Distribution of Subjects")
plt.xticks(rotation=45)
plt.show()
```



```
df['label'] = 0
```

```
df['text'] = df['title'].astype('category').cat.codes
```

```
import pandas as pd
df = pd.read_csv("/content/Fake.csv")
```

```
df = pd.get_dummies(df, columns=['subject'], drop_first=True)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the CSV file
df = pd.read_csv("/content/Fake.csv")

# Create the 'label' column and assign 0 to all rows
df['label'] = 0

# Now, 'subject' column should be present.
# Perform one-hot encoding on the 'subject' column
df = pd.get_dummies(df, columns=['subject'], drop_first=True)

# Convert 'title' column to numerical representation using category codes
df['text'] = df['title'].astype('category').cat.codes

# Define features (X) and target (y)
X = df['text']
y = df['label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
!pip install --upgrade scikit-learn
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Load the CSV file for fake news
df_fake = pd.read_csv("/content/Fake.csv")

# Create the 'label' column and assign 0 to all rows for fake news
df_fake['label'] = 0

# Load or create a dataset for real news (replace with your real news data)
# Here's an example if you have a CSV file for real news:
# df_real = pd.read_csv("/content/real_news.csv")
# Or, if you want to generate some dummy real news data:
df_real = pd.DataFrame({'title': ['Real News 1', 'Real News 2', 'Real News 3'],
                         'text': ['This is a real news article.', 'Another real news story.', 'Breaking real news.'],
                         'subject': ['News', 'Politics', 'World']})
df_real['label'] = 1  # Assign 1 to real news

# Concatenate the fake and real news dataframes
df = pd.concat([df_fake, df_real], ignore_index=True)

# Perform one-hot encoding on the 'subject' column
df = pd.get_dummies(df, columns=['subject'], drop_first=True)

# Keep the original 'title' column for text processing
# df['text'] = df['title'].astype('category').cat.codes # Comment out or remove this line

# Define features (X) and target (y)
X = df[['title']]  # Use 'title' column for text features
y = df['label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = vectorizer.fit_transform(X_train['title']) # Use 'title' column here
X_test_tfidf = vectorizer.transform(X_test['title']) # Use 'title' column here

model = LogisticRegression()
model.fit(X_train_tfidf, y_train)
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
▾ LogisticRegression
LogisticRegression()
```

```
from sklearn.metrics import accuracy_score, classification_report

y_pred = model.predict(X_test_tfidf)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4697

    accuracy                           1.00      4697
   macro avg       1.00      1.00      1.00      4697
weighted avg       1.00      1.00      1.00      4697
```

```
sample_text = ["The government has declared a national emergency amid new reports."]
sample_vec = vectorizer.transform(sample_text)
prediction = model.predict(sample_vec)

print("Prediction:", "Fake" if prediction[0] == 0 else "Real")
```

```
Prediction: Fake
```

```
user_df = pd.DataFrame({'text': sample_text})
user_df['text_vector'] = vectorizer.transform(user_df['text']).toarray().tolist() # Convert to list of arrays
```

```
!pip install gradio
import gradio as gr
```

```
Collecting gradio
  Downloading gradio-5.29.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
```

```
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.0 (from gradio)
  Downloading gradio_client-1.10.0-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.31.1)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.18)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.4)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (2025.3.2
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->g
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->g
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradic
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.
```

```python
def predict_fake_news(text):
    vector = vectorizer.transform([text])
    prediction = model.predict(vector)[0]
    return "Fake News ❌" if prediction == 0 else "Real News ✅"
```

```python
iface = gr.Interface(fn=predict_fake_news,
                     inputs="text",
                     outputs="text",
                     title="📰 Fake News Detector",
                     description="Enter a news article to determine if it's real or fake.")
iface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automati

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://3de32953f0e3ec976b.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the worki
```

## 📰 Fake News Detector

Enter a news article to determine if it's real or fake.

text

house

output

Fake News ❌

Clear Submit

Flag