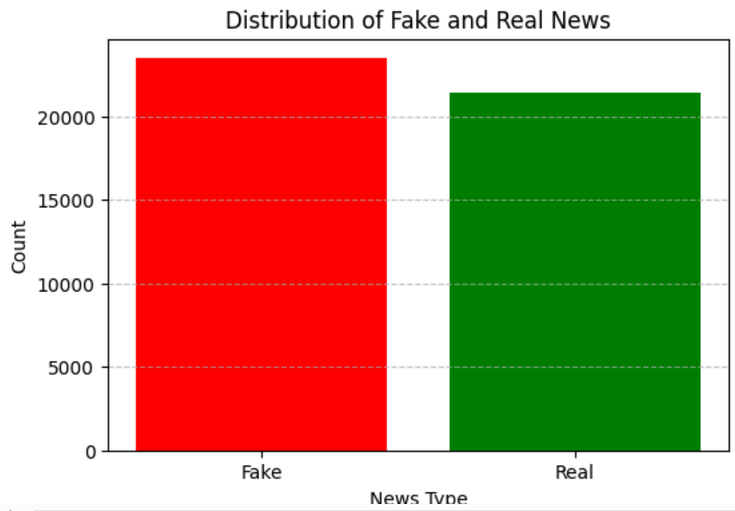
```
from google.colab import files
files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable

```
import pandas as pd
import numpy as np
import re
import string

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load files that are now uploaded
df_fake = pd.read_csv('Fake.csv')
df_true = pd.read_csv('True.csv')

df_fake['label'] = 0
df_true['label'] = 1

df = pd.concat([df_fake, df_true], ignore_index=True)
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

def preprocess(text):
    text = str(text).lower()
    text = re.sub(f'[{re.escape(string.punctuation)}]', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

df['text'] = df['title'].apply(preprocess)

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

vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_vec = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("🎯 Accuracy:", accuracy_score(y_test, y_pred))
print("\n📄 Classification Report:\n", classification_report(y_test, y_pred))
```

```
🎯 Accuracy: 0.9492204899777282

📄 Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95      4710
           1       0.93      0.96      0.95      4270

    accuracy                           0.95      8980
   macro avg       0.95      0.95      0.95      8980
weighted avg       0.95      0.95      0.95      8980
```

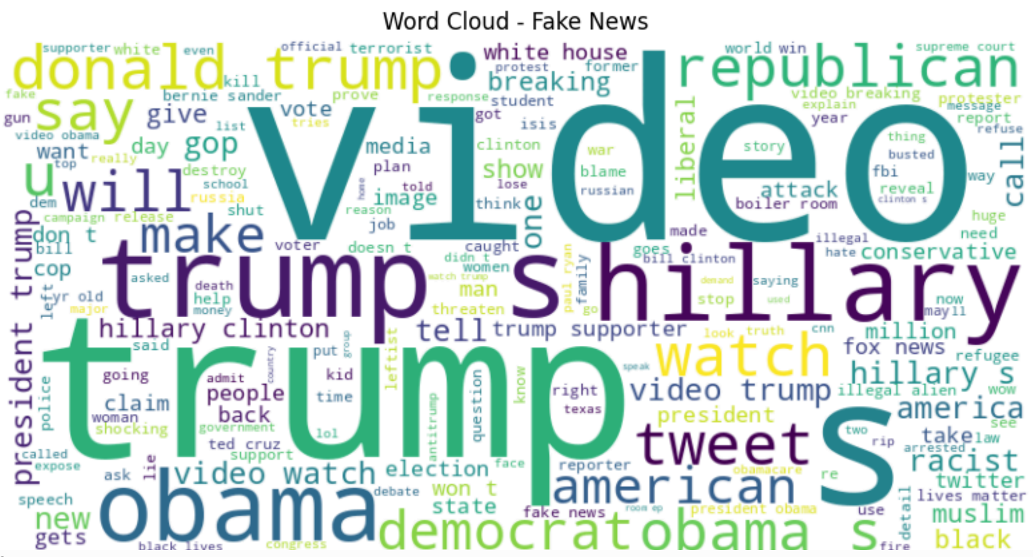
```
import matplotlib.pyplot as plt

# Count labels
label_counts = df['label'].value_counts()
labels = ['Fake', 'Real']

plt.figure(figsize=(6,4))
plt.bar(labels, label_counts, color=['red', 'green'])
plt.title('Distribution of Fake and Real News')
plt.xlabel('News Type')
plt.ylabel('Count')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-packages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from wordcloud) (11.2.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (2.9.0.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->wordclou
```

```
from wordcloud import WordCloud

# Word cloud for Fake news
fake_words = ' '.join(df[df['label'] == 0]['text'])

wordcloud = WordCloud(width=800, height=400, background_color='white').generate(fake_words)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Fake News')
plt.show()
```



```
# Predict on all data
df['prediction'] = model.predict(vectorizer.transform(df['text']))

# Map label names
df['label_name'] = df['label'].map({0: 'Fake', 1: 'Real'})
```

```
df['predicted_label'] = df['prediction'].map({0: 'Fake', 1: 'Real'})

# Export
df[['title', 'label_name', 'predicted_label']].to_csv('news_predictions.csv', index=False)

from google.colab import files
files.download('news_predictions.csv')
```

```
!pip install fpdf
```

```
Collecting fpdf
  Downloading fpdf-1.7.2.tar.gz (39 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: fpdf
  Building wheel for fpdf (setup.py) ... done
  Created wheel for fpdf: filename=fpdf-1.7.2-py2.py3-none-any.whl size=40704 sha256=fe682a9381821183a1eb86099e1a55103fcc7f6eaf6fe4
  Stored in directory: /root/.cache/pip/wheels/65/4f/66/bbda9866da446a72e206d6484cd97381cbc7859a7068541c36
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
```

```
from fpdf import FPDF
from datetime import datetime

# Get accuracy score again
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)

# Count fake and real
fake_count = df['label'].value_counts()[0]
real_count = df['label'].value_counts()[1]

# Create PDF
pdf = FPDF()
pdf.add_page()
pdf.set_font("Arial", size=12)

# Title
pdf.set_font("Arial", 'B', 16)
pdf.cell(200, 10, txt="Fake News Detection Project Report", ln=True, align='C')
pdf.ln(10)

# Timestamp
pdf.set_font("Arial", size=12)
pdf.cell(200, 10, txt=f"Generated on: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}", ln=True)
pdf.ln(10)

# Dataset Details
pdf.set_font("Arial", 'B', 14)
pdf.cell(200, 10, txt="Dataset Summary:", ln=True)
pdf.set_font("Arial", size=12)
pdf.cell(200, 10, txt=f"Total Articles: {fake_count + real_count}", ln=True)
pdf.cell(200, 10, txt=f"Fake News Articles: {fake_count}", ln=True)
pdf.cell(200, 10, txt=f"Real News Articles: {real_count}", ln=True)
pdf.ln(10)

# Preprocessing
pdf.set_font("Arial", 'B', 14)
pdf.cell(200, 10, txt="Text Preprocessing Applied:", ln=True)
pdf.set_font("Arial", size=12)
pdf.multi_cell(0, 10, txt="- Lowercasing\n- Removing punctuation\n- Removing digits\n- Removing extra spaces")
pdf.ln(5)

# Model
pdf.set_font("Arial", 'B', 14)
pdf.cell(200, 10, txt="Model Used:", ln=True)
pdf.set_font("Arial", size=12)
pdf.cell(200, 10, txt="Logistic Regression", ln=True)
pdf.ln(5)

# Accuracy
pdf.set_font("Arial", 'B', 14)
pdf.cell(200, 10, txt="Model Accuracy:", ln=True)
pdf.set_font("Arial", size=12)
pdf.cell(200, 10, txt=f"{accuracy:.2%}", ln=True)
pdf.ln(10)
```

```
# Save it
pdf.output("FakeNews_Project_Report.pdf")

# Download
from google.colab import files
files.download("FakeNews_Project_Report.pdf")
```

```
with open('app.py', 'w') as f:
    f.write('''
import streamlit as st
import pandas as pd
import re
import string
import joblib

# Load model and vectorizer
model = joblib.load("fake_news_model.pkl")
vectorizer = joblib.load("vectorizer.pkl")

# Preprocess input
def preprocess(text):
    text = str(text).lower()
    text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)
    text = re.sub(r"\\d+", "", text)
    text = re.sub(r"\\s+", " ", text).strip()
    return text

# UI
st.title("📰 Fake News Detector")

headline = st.text_input("Enter a news headline:")

if st.button("Check"):
    if headline.strip() == "":
        st.warning("Please enter a headline.")
    else:
        processed = preprocess(headline)
        vec_input = vectorizer.transform([processed])
        result = model.predict(vec_input)[0]
        label = "🟢 Real News" if result == 1 else "🔴 Fake News"
        st.success(f"Prediction: {label}")
''')
```

```
from google.colab import files
files.download('app.py')
```

```
import joblib

# Save model
joblib.dump(model, 'fake_news_model.pkl')

# Save vectorizer
joblib.dump(vectorizer, 'vectorizer.pkl')
```

```
['vectorizer.pkl']
```

```
from google.colab import files

files.download('fake_news_model.pkl')
files.download('vectorizer.pkl')
```