In [73]:
```
import pandas as pd
import numpy as np
#pandas is used for handling tabular data (like CSV files).
#numpy is used for numerical operations (arrays, math, etc).
```

In [74]:
```
df_fake = pd.read_csv(r"C:\Users\HP\OneDrive\Desktop\ML mini project FAKE NEWS
df_true = pd.read_csv(r"C:\Users\HP\OneDrive\Desktop\ML mini project FAKE NEWS
```

In [75]:
```
df_fake.head()
```

Out[75]:

	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

In [76]:
```
df_fake["label"] = 0
df_true["label"] = 1
#Adds a new column label:
#0 for fake news
#1 for real news
#This helps later for classification (supervised learning needs labeled data).
```

In [77]:
```
df_true = df_true[['text','label']]
df_fake = df_fake[['text','label']]
```

In [78]:
```
df = pd.concat([df_fake, df_true], axis=0)
#Combines both fake and true news into a single DataFrame df.
#axis=0 stacks them vertically (one below the other).
```

In [79]:
```
df.shape
```

Out[79]:
```
(44898, 2)
```

In [80]:
```
df.isnull().sum() # no null values
```

Out[80]:
```
text     0
label    0
dtype: int64
```

In [81]:
```
df['label'].value_counts()
```

Out[81]:
```
label
0    23481
1    21417
Name: count, dtype: int64
```

In [82]:
```
df_true.shape # true news
```

Out[82]:
```
(21417, 2)
```

In [83]:
```
df_fake.shape # fake news
```

Out[83]:
```
(23481, 2)
```

In [84]:
```
df = df.sample(frac=1)
#df.sample(frac=1) randomly shuffles all rows in the DataFrame.
#This prevents the model from learning in the order the data was added (first
#frac=1 means "sample 100% of the rows, in random order".
```

In [85]:
```
df.reset_index(inplace=True)
df.drop(["index"], axis=1, inplace=True)
#After shuffling, the old row index is no longer meaningful, so:
#reset_index() assigns new indices (0 to N).
#drop(["index"], axis=1) removes the old index column from the DataFrame.
```

In [86]:
```
df.head()
```

Out[86]:

	text	label
0	WASHINGTON (Reuters) - President Donald Trump ...	1
1	JOHANNESBURG (Reuters) - Nkosazana Dlamini-Zum...	1
2	Jay Dyer 21st Century WireAccording to a recen...	0
3	NEW YORK (Reuters) - If Hillary Clinton decide...	1
4	Sean Spicer might be in a little bit of a trou...	0

In [99]:
```
X = df['text']
y = df['label']
#X is your feature data (news content).
#y is your label (0 = fake, 1 = real).
#This prepares the data for train-test splitting.
```

In [100…
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, randc

#Splits the dataset into 80% training and 20% testing.
#random_state=0 ensures the same split every time (for reproducibility).
```

In [101…
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
#Imports TfidfVectorizer, a tool that converts text into numeric vectors based
#This is essential for feeding text into machine learning models.
```

In [102…

```
vectorizer = TfidfVectorizer()
vec_train = vectorizer.fit_transform(X_train)
vec_test = vectorizer.transform(X_test)
#TfidfVectorizer() creates the vectorizer.
#.fit_transform(X_train):
#Learns vocabulary from the training data.
#Converts training text into sparse numerical vectors.
#.transform(X_test):
#Applies the same vocabulary to the test data (no learning here)
```

In [103…

```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()
clf.fit(vec_train, y_train)
#Imports and initializes a Multinomial Naive Bayes classifier – very effective
#.fit(vec_train, y_train) trains the model using the TF-IDF vectors and their
```

Out[103…

```
▾ MultinomialNB
MultinomialNB()
```

In [104…

```
y_pred = clf.predict(vec_test)
#Uses the trained model to make predictions (y_pred) on the test set (vec_test
#These are the predicted labels: 0 for fake, 1 for real.
```

In [98]:

```
from sklearn.metrics import accuracy_score

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

#Imports a metric to calculate the model's accuracy.
#accuracy_score(y_test, y_pred) compares the predicted labels with the true te
#The result tells you how well your model is performing on unseen data.
```

```
Accuracy: 0.9360801781737194
```

In [94]:

```
from sklearn.metrics import confusion_matrix, classification_report

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
#confusion_matrix: shows the number of:
#True Positives (real news correctly predicted as real)
#True Negatives (fake predicted as fake)
#False Positives (fake predicted as real)
#False Negatives (real predicted as fake)
#classification_report gives:
#Precision: How many predicted positives were actually correct
#Recall: How many actual positives were caught
#F1-score: Harmonic mean of precision and recall
#Support: Number of true instances for each label
```

```
[[8871  473]
 [ 675 7941]]
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      9344
           1       0.94      0.92      0.93      8616

    accuracy                           0.94     17960
   macro avg       0.94      0.94      0.94     17960
weighted avg       0.94      0.94      0.94     17960
```

In [ ]:

In [95]:
```
sample = ["President announces new policy to fight inflation."]
sample_vec = vectorizer.transform(sample)
print(clf.predict(sample_vec))

#Defines a custom news headline or text.
#Transforms it into TF-IDF form using the previously fitted vectorizer.
#Passes it to the model to predict: 0 = fake, 1 = real.
```

```
[1]
```

In [96]:
```
samples = [
    "NASA discovers water on Mars",
    "Click here to win free iPhones",
    "Government collapses under pressure"
]

samples_vec = vectorizer.transform(samples)
print(clf.predict(samples_vec))
#Tests the model on multiple new inputs.
#Again, transforms the input using the same vectorizer.
#Outputs an array of predictions (0 or 1) for each sentence.
```

```
[0 0 1]
```

In [97]:
```
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print text outputs
print("Confusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 📊 Plot confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Fake", "Real"],
            yticklabels=["Fake", "Real"])
```

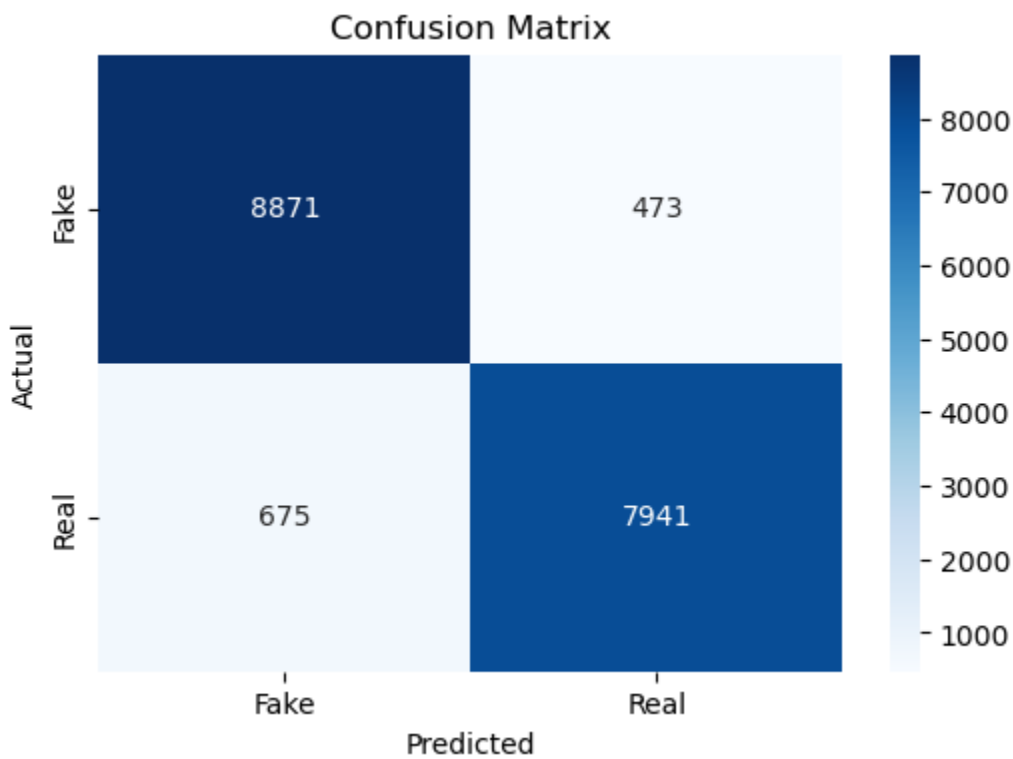
```
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Confusion Matrix:
 [[8871  473]
 [ 675 7941]]

Classification Report:
               precision    recall  f1-score   support

           0       0.93      0.95      0.94      9344
           1       0.94      0.92      0.93      8616

    accuracy                           0.94     17960
   macro avg       0.94      0.94      0.94     17960
weighted avg       0.94      0.94      0.94     17960
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: