Data loading

Load the two CSV files into pandas DataFrames and add a label column.

```
import pandas as pd

df_fake = pd.read_csv('Fake.csv')
df_true = pd.read_csv('True.csv')

df_fake['label'] = 'fake'
df_true['label'] = 'true'
```

Data preparation

Concatenate the two dataframes, preprocess the text data, and handle missing values.

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string

# Concatenate the dataframes
df_combined = pd.concat([df_fake, df_true], ignore_index=True)

# Text preprocessing
def preprocess_text(text):
    if isinstance(text, str):  # Check if the value is a string
        text = text.lower()
        text = ''.join([char for char in text if char not in string.punctuation])
        stop_words = set(stopwords.words('english'))
        text = ' '.join([word for word in text.split() if word not in stop_words])
        stemmer = PorterStemmer()
        text = ' '.join([stemmer.stem(word) for word in text.split()])
    return text


df_combined['text'] = df_combined['text'].apply(preprocess_text)

# Handle missing values (replace with empty string)
df_combined['text'].fillna('', inplace=True)

display(df_combined.head())
```

```
<ipython-input-8-94c88dffe795>:25: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df_combined['text'].fillna('', inplace=True)
```

	title	text	subject	date	label
0	Donald Trump Sends Out Embarrassing New Year'...	donald trump wish american happi new year leav...	News	December 31, 2017	fake
1	Drunk Bragging Trump Staffer Started Russian ...	hous intellig committe chairman devin nune go ...	News	December 31, 2017	fake
2	Sheriff David Clarke Becomes An Internet Joke...	friday reveal former milwauke sheriff david cl...	News	December 30, 2017	fake
3	Trump Is So Obsessed He Even Has Obama's Name...	christma day donald trump announc would back w...	News	December 29, 2017	fake
4	Pope Francis Just Called Out Donald Trump Dur...	pope franci use annual christma day messag reb...	News	December 25, 2017	fake

Feature engineering

Convert the preprocessed text data in `df_combined` into numerical features using TF-IDF.

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
```

```
# Create a TfidfVectorizer object
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))

# Fit and transform the text data
tfidf_matrix = tfidf_vectorizer.fit_transform(df_combined['text'])

# Convert the sparse matrix to a dense array
tfidf_array = tfidf_matrix.toarray()

# Create a new DataFrame with TF-IDF features and labels
df_tfidf = pd.DataFrame(tfidf_array, columns=tfidf_vectorizer.get_feature_names_out())
df_tfidf['label'] = df_combined['label']

display(df_tfidf.head())
```

	10	10 percent	10 year	100	1000	10000	100000	11	12	13	...	young peopl	younger	youth	youtub	zealand	zero	zika	zimbabw	zone	zuma
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Data splitting

Split the data into training and testing sets.

```
from sklearn.model_selection import train_test_split

# Assuming 'label' is the target variable and the rest are features
X = df_tfidf.drop('label', axis=1)
y = df_tfidf['label']

# Split 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)
```

Model training

Train a Naive Bayes and a Logistic Regression classifier.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

# Instantiate the models
nb_model = MultinomialNB()
lr_model = LogisticRegression(max_iter=1000)  # Increased max_iter

# Train the models
nb_model.fit(X_train, y_train)
lr_model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

Model evaluation

Evaluate the performance of the trained Naive Bayes and Logistic Regression models.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Predict the labels for the test set using both models
nb_predictions = nb_model.predict(X_test)
```

```
lr_predictions = lr_model.predict(X_test)

# Calculate evaluation metrics
nb_accuracy = accuracy_score(y_test, nb_predictions)
lr_accuracy = accuracy_score(y_test, lr_predictions)

nb_precision = precision_score(y_test, nb_predictions, average='weighted')
lr_precision = precision_score(y_test, lr_predictions, average='weighted')

nb_recall = recall_score(y_test, nb_predictions, average='weighted')
lr_recall = recall_score(y_test, lr_predictions, average='weighted')

nb_f1 = f1_score(y_test, nb_predictions, average='weighted')
lr_f1 = f1_score(y_test, lr_predictions, average='weighted')

# Print the evaluation metrics
print("Naive Bayes:")
print(f"Accuracy: {nb_accuracy:.4f}")
print(f"Precision: {nb_precision:.4f}")
print(f"Recall: {nb_recall:.4f}")
print(f"F1-score: {nb_f1:.4f}")

print("\nLogistic Regression:")
print(f"Accuracy: {lr_accuracy:.4f}")
print(f"Precision: {lr_precision:.4f}")
print(f"Recall: {lr_recall:.4f}")
print(f"F1-score: {lr_f1:.4f}")
```

```
Naive Bayes:
Accuracy: 0.9532
Precision: 0.9533
Recall: 0.9532
F1-score: 0.9532

Logistic Regression:
Accuracy: 0.9906
Precision: 0.9907
Recall: 0.9906
F1-score: 0.9906
```

Summary:

1. Q&A

- **Which model performed better?** The Logistic Regression model significantly outperformed the Naive Bayes model across all evaluation metrics (accuracy, precision, recall, and F1-score).

2. Data Analysis Key Findings

- **Logistic Regression Superior Performance:** The Logistic Regression model achieved an accuracy of 0.9906, precision of 0.9907, recall of 0.9906, and F1-score of 0.9906 on the test set, outperforming the Naive Bayes model.
- **Naive Bayes Performance:** The Naive Bayes model achieved an accuracy of 0.9532, precision of 0.9533, recall of 0.9532, and F1-score of 0.9532 on the test set.
- **TF-IDF Vectorization:** TF-IDF vectorization with a vocabulary size of 5000 features (using unigrams and bigrams) was used to convert text data into numerical representations for model training.

3. Insights or Next Steps

- **Hyperparameter Tuning:** Explore hyperparameter tuning for both models to potentially improve their performance further.
- **Investigate Misclassifications:** Analyze the instances where the models made incorrect predictions to gain insights into their weaknesses and potential areas for improvement in data preprocessing or feature engineering.