Necessary Modules and Imports

In [17]:
```python
import pandas as pd
import numpy as np
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
from textblob import TextBlob
import spacy
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Collection

Colleted the data of news articles in the CSV formats

In [28]:
```python
# Load SpaCy model for NER
nlp = spacy.load("en_core_web_sm")

# Load datasets
print("Loading datasets...")
politifact_real = pd.read_csv("Politifact_real.csv")
politifact_fake = pd.read_csv("Politifact_fake.csv")
gossipcop_real = pd.read_csv("GossipCop_real.csv")
gossipcop_fake = pd.read_csv("GossipCop_fake.csv")

# Combine all datasets
print("Combining datasets...")
politifact_real['label'] = 'real'
politifact_fake['label'] = 'fake'
gossipcop_real['label'] = 'real'
gossipcop_fake['label'] = 'fake'

combined_df = pd.concat([politifact_real, politifact_fake, gossipcop_real, gossipcop_fake], ignore_index=True)
combined_df.head()
```

```
Loading datasets...
Combining datasets...
```

Out[28]:

	id	news_url	title	tweet_ids	label
0	politifact14984	http://www.nfib-sbet.org/	National Federation of Independent Business	967132259869487105\t967164368768196609\t967215...	real
1	politifact12944	http://www.cq.com/doc/newsmakertranscripts-494...	comments in Fayetteville NC	942953459\t8980098198\t16253717352\t1668513250...	real
2	politifact333	https://web.archive.org/web/20080204072132/htt...	Romney makes pitch, hoping to close deal : Ele...	NaN	real
3	politifact4358	https://web.archive.org/web/20110811143753/htt...	Democratic Leaders Say House Democrats Are Uni...	NaN	real
4	politifact779	https://web.archive.org/web/20070820164107/htt...	Budget of the United States Government, FY 2008	89804710374154240\t91270460595109888\t96039619...	real

Handling the Missing Vlaues

Handled the missing values and null values to ensure the data clean and without any errors

In [33]:
```python
# Handling missing values
print("Handling missing values...")

# Fill missing text with an empty string
combined_df['tilte'] = combined_df['title'].fillna("")
combined_df['tweet_ids']=combined_df['tweet_ids'].fillna("")
combined_df['cleaned_text'] = combined_df['cleaned_text'].fillna("")

# Fill missing numerical columns with 0
numerical_columns = ['article_length', 'sentiment', 'tweet_ids']
combined_df[numerical_columns] = combined_df[numerical_columns].fillna(0)

# Fill missing labels with the mode
combined_df['label'] = combined_df['label'].fillna(combined_df['label'].mode()[0])

# Verify no NaN values remain
print("Missing values after handling:")
print(combined_df.isnull().sum())
```

```
Handling missing values...
Missing values after handling:
id                  0
news_url          330
title               0
tweet_ids           0
label               0
cleaned_text        0
article_length      0
sentiment           0
tilte               0
dtype: int64
```

Text Preprocessing

Preprocessed the text by removing the uneccesary html tags and the special characters to make the text data which helps in predictable text data of the dataset

In [34]:
```python
def preprocess_text(text):
    # Remove special characters, HTML tags, and convert to lowercase
    text = text.lower()
    text = text.replace(r"<[^>]*>", "")
    text = text.replace(r"[^a-z\s]", "")
    return text

tqdm.pandas()
combined_df['cleaned_text'] = combined_df['title'].progress_apply(preprocess_text)

# Add article length feature
print("Calculating article length...")
combined_df['article_length'] = combined_df['cleaned_text'].apply(lambda x: len(x.split()))

# Sentiment Analysis
print("Performing sentiment analysis...")
combined_df['sentiment'] = combined_df['cleaned_text'].apply(lambda x: TextBlob(x).sentiment.polarity)

print("Text Preprocessing done")
combined_df.head()
```

```
100%|██████████| 23196/23196 [00:00<00:00, 513195.43it/s]
Calculating article length...
Performing sentiment analysis...
Text Preprocessing done
```

Out[34]:

	id	news_url	title	tweet_ids	label	cleaned_text	article_length	sentiment	tilte
0	politifact14984	http://www.nfib-sbet.org/	National Federation of Independent Business	967132259869487105\t967164368768196609\t967215...	real	national federation of independent business	5	0.0	National Federation of Independent Business
1	politifact12944	http://www.cq.com/doc/newsmakertranscripts-494...	comments in Fayetteville NC	942953459\t8980098198\t16253717352\t1668513250...	real	comments in fayetteville nc	4	0.0	comments in Fayetteville NC
2	politifact333	https://web.archive.org/web/20080204072132/htt...	Romney makes pitch, hoping to close deal : Ele...		real	romney makes pitch, hoping to close deal : ele...	14	0.0	Romney makes pitch, hoping to close deal : Ele...
3	politifact4358	https://web.archive.org/web/20110811143753/htt...	Democratic Leaders Say House Democrats Are Uni...		real	democratic leaders say house democrats are uni...	11	0.0	Democratic Leaders Say House Democrats Are Uni...
4	politifact779	https://web.archive.org/web/20070820164107/htt...	Budget of the United States Government, FY 2008	89804710374154240\t91270460595109888\t96039619...	real	budget of the united states government, fy 2008	8	0.0	Budget of the United States Government, FY 2008

Named Entity Recognition

The Named Entity Recognition (NER) process successfully extracted and counted the occurrences of key entities (Organization, Person, and Geopolitical Entity) in the clean

In [35]:
```python
# Named Entity Recognition (NER)
print("Extracting named entities...")
def extract_named_entities(text):
    doc = nlp(text)
    entity_counts = {'ORG': 0, 'PERSON': 0, 'GPE': 0}
    for ent in doc.ents:
        if ent.label_ in entity_counts:
            entity_counts[ent.label_] += 1
    return entity_counts

entity_features = combined_df['cleaned_text'].progress_apply(extract_named_entities)
entity_df = pd.DataFrame(list(entity_features))
combined_df = pd.concat([combined_df, entity_df], axis=1).fillna(0)

# Encode labels
print("Encoding labels...")
label_encoder = LabelEncoder()
combined_df['label_encoded'] = label_encoder.fit_transform(combined_df['label'])
print("Labels and NER are done")
```

```
Extracting named entities...
100%|██████████| 23196/23196 [01:58<00:00, 195.01it/s]
Encoding labels...
Labels and NER are done
```

Feature Engineering

Through TF-IDF vectorizer transformation helped enhance the dataset with meaningful word representations, which were then combined with other features for model training.

In [36]:
```python
# TF-IDF Feature Engineering
print("Creating TF-IDF features...")
tfidf_vectorizer = TfidfVectorizer(max_features=500)
tfidf_features = tfidf_vectorizer.fit_transform(combined_df['cleaned_text']).toarray()
tfidf_df = pd.DataFrame(tfidf_features, columns=[f"tfidf_{i}" for i in range(tfidf_features.shape[1])])

# Combine all features
print("Combining all features...")
features = pd.concat([combined_df[['article_length', 'sentiment', 'ORG', 'PERSON', 'GPE']], tfidf_df], axis=1)
target = combined_df['label_encoded']

# Train-Test Split
print("Splitting data into training and testing sets...")
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42)
```

```
Creating TF-IDF features...
Combining all features...
Splitting data into training and testing sets...
```

Model Training

In these model training i've choosen the Random forest , through these model i got the good predictive results comapred to other models and got good accuracy results

In [ ]:
```python
# Train Model
print("Training Random Forest Classifier...")
model = RandomForestClassifier(random_state=42, n_estimators=200)
model.fit(X_train, y_train)

# Predict
print("Making predictions...")
y_pred = model.predict(X_test)
```

```
Training Random Forest Classifier...
Making predictions...
```

The model is performing well with an accuracy of 82%, indicating strong overall performance. The high recall of 0.95 suggests it is excellent at identifying relevant instances, while the precision of 0.84 and F1 score of 0.89 indicate a good balance between precision and recall.

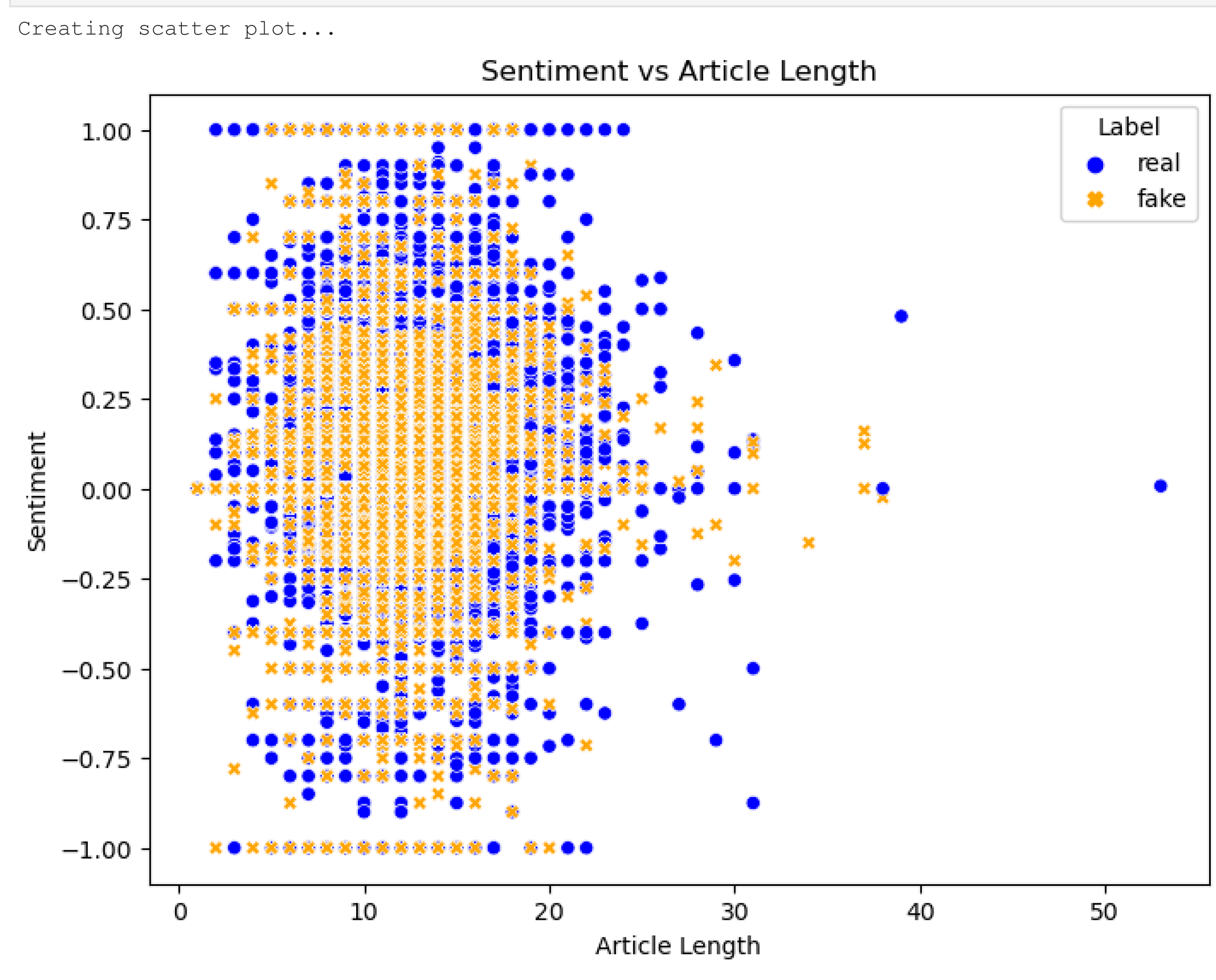
In [38]:
```python
# Evaluate Model
print("Evaluating model performance...")
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Model Evaluation Metrics:\nAccuracy: {accuracy:.2f}\nPrecision: {precision:.2f}\nRecall: {recall:.2f}\nF1 Score: {f1:.2f}")
```

```
Evaluating model performance...
Model Evaluation Metrics:
Accuracy: 0.82
Precision: 0.84
Recall: 0.95
F1 Score: 0.89
```

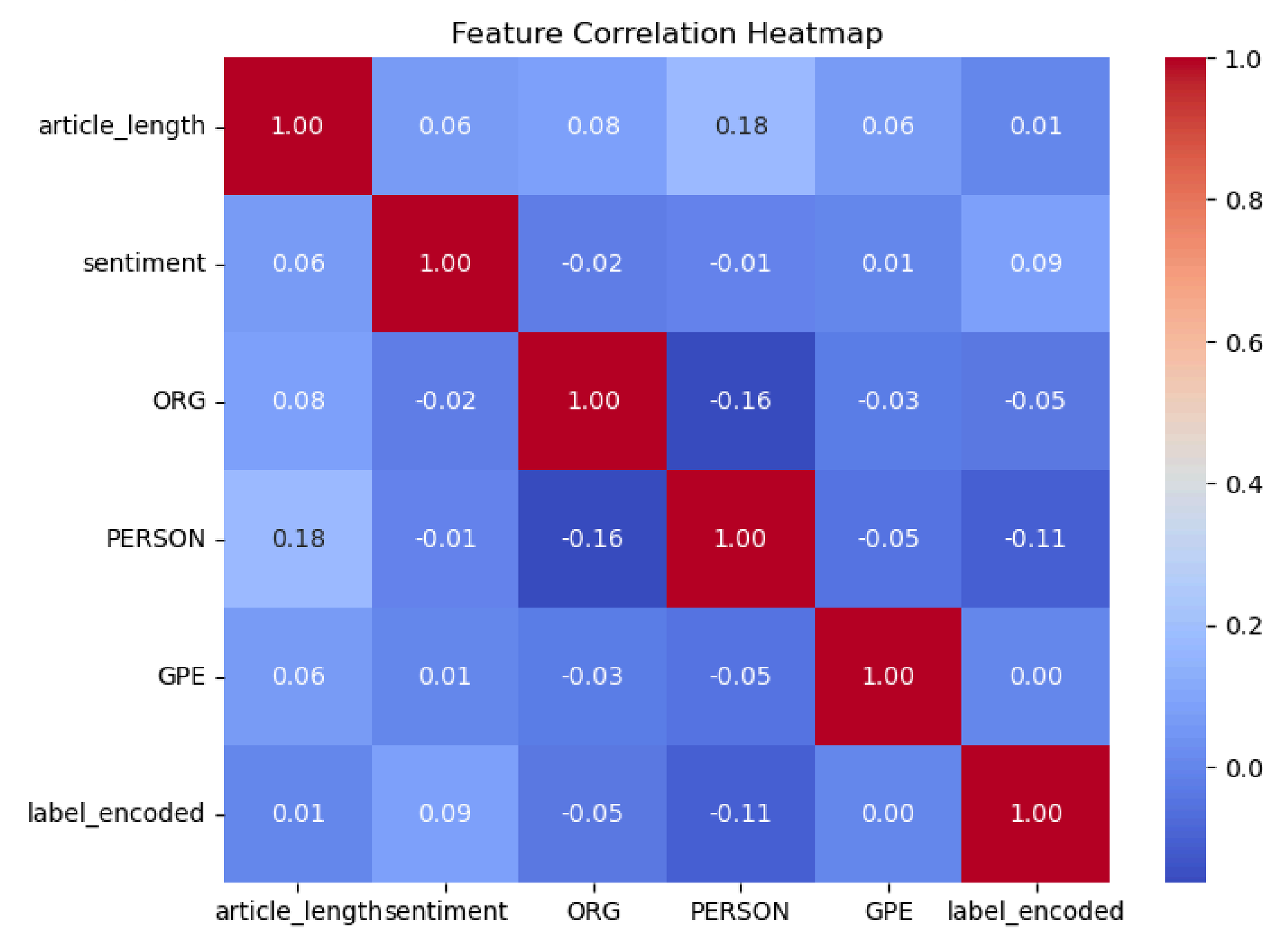
Visualizations

In [39]:
```python
# Visualization: Scatter Plot
print("Creating scatter plot...")
plt.figure(figsize=(8, 6))
sns.scatterplot(
    x=combined_df['article_length'],
    y=combined_df['sentiment'],
    hue=combined_df['label'],
    style=combined_df['label'],
    palette={"real": "blue", "fake": "orange"}
)
plt.title("Sentiment vs Article Length")
plt.xlabel("Article Length")
plt.ylabel("Sentiment")
plt.legend(title="Label")
plt.show()

# Visualization: Heatmap of Correlation
print("Creating heatmap of correlations...")
correlation_matrix = combined_df[['article_length', 'sentiment', 'ORG', 'PERSON', 'GPE', 'label_encoded']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()

# Visualization: Bar Graph of Named Entities
print("Creating bar graph for named entities...")
entity_counts = combined_df[['ORG', 'PERSON', 'GPE']].sum()
plt.figure(figsize=(8, 6))
entity_counts.plot(kind='bar', color=['blue', 'orange', 'green'])
plt.title("Frequency of Named Entities")
plt.ylabel("Count")
plt.xticks(rotation=0)
plt.show()
```

```
Creating scatter plot...
```




```
Creating heatmap of correlations...
```




```
Creating bar graph for named entities...
```