```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from flask import Flask, request, jsonify
```

```
encodings_to_try = ['utf-8', 'latin-1', 'ISO-8859-1', 'cp1252']
for encoding in encodings_to_try:
    try:
        data = pd.read_csv("/content/test.csv", encoding=encoding)
        break
    except UnicodeDecodeError:
        print(f"Failed to decode using {encoding} encoding")
```

```
Failed to decode using utf-8 encoding
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4815 entries, 0 to 4814
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   textID            3534 non-null   object
 1   text              3534 non-null   object
 2   sentiment         3534 non-null   object
 3   Time of Tweet     3534 non-null   object
 4   Age of User       3534 non-null   object
 5   Country           3534 non-null   object
 6   Population -2020  3534 non-null   float64
 7   Land Area (Km²)   3534 non-null   float64
 8   Density (P/Km²)   3534 non-null   float64
dtypes: float64(3), object(6)
memory usage: 338.7+ KB
```

```
data.isna().sum()
```

```
textID              1281
text                1281
sentiment           1281
Time of Tweet       1281
Age of User         1281
Country             1281
Population -2020    1281
Land Area (Km²)     1281
Density (P/Km²)     1281
dtype: int64
```

```
data = data.dropna()
```

```
data.isna().sum()
```

```
textID              0
text                0
sentiment           0
Time of Tweet       0
Age of User         0
Country             0
Population -2020    0
Land Area (Km²)     0
Density (P/Km²)     0
dtype: int64
```

```
#check for duplicates
duplicates = data[data.duplicated(keep=False)]
if not duplicates.empty:
  print("Duplicate rows found:")
  print(duplicates)
else:
  print("No duplicates found.")
```

```
No duplicates found.
```

```
#encoding sentiment column
encoder = LabelEncoder()
data["sentiment"] = encoder.fit_transform(data["sentiment"])
```

```
data["sentiment"].unique()
```

```
array([1, 2, 0])
```

```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
# Define the data cleaning function
def clean_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word not in stop_words]
    cleaned_text = ' '.join(filtered_tokens)
    return cleaned_text
```

```
data['cleaned_text'] = data['text'].apply(clean_text)
print(data['cleaned_text'])
```

```
0                    last session day httptwitpiccom67ezh
1       shanghai also really exciting precisely skyscr...
2       recession hit veronique branquinho quit compan...
3                                               happy bday
4                                 httptwitpiccom4w75p like
                               ...
3529                              3 im tired cant sleep try
3530    alone old house thanks net keeps alive kicking...
3531    know mean little dog sinking depression wants ...
3532          sutra next youtube video gon na love videos
3533           httptwitpiccom4woj2 omgssh ang cute ng bby
Name: cleaned_text, Length: 3534, dtype: object
```

```
# Feature extraction using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=10)
tfidf_matrix = tfidf_vectorizer.fit_transform(data['cleaned_text'])
```

```
X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix, data['sentiment'], test_size=0.2, random_state=42)

# Train the Naive Bayes classifier
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)

y_pred = naive_bayes.predict(X_test)

# Print classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.01      0.02       207
           1       0.43      0.95      0.59       286
           2       0.71      0.27      0.39       214

    accuracy                           0.47       707
   macro avg       0.72      0.41      0.33       707
weighted avg       0.68      0.47      0.36       707
```

```
app = Flask(__name__)

@app.route('/predict_sentiment', methods=['POST'])
def predict_sentiment():
    try:
        data = request.json
        text = data['text']

        # Clean and preprocess text data
        def clean_text(text):
            text = text.lower()
            text = text.translate(str.maketrans('', '', string.punctuation))
            tokens = word_tokenize(text)
            stop_words = set(stopwords.words('english'))
            filtered_tokens = [word for word in tokens if word not in stop_words]
            cleaned_text = ' '.join(filtered_tokens)
            return cleaned_text

        cleaned_text = clean_text(text)

        # Convert cleaned text to TF-IDF features
        tfidf_features = tfidf_vectorizer.transform([cleaned_text])

        # Predict sentiment using the trained Naive Bayes classifier
        sentiment = naive_bayes.predict(tfidf_features)[0]

        response = {'sentiment': sentiment}
        return jsonify(response)

    except Exception as e:
        response = {'error': str(e)}
        return jsonify(response), 500

if __name__ == '__main__':

    # Initialize TF-IDF vectorizer
    tfidf_vectorizer = TfidfVectorizer(max_features=10)
    tfidf_matrix = tfidf_vectorizer.fit_transform(data['cleaned_text'])

    # Train the Naive Bayes classifier
    naive_bayes = MultinomialNB()
    naive_bayes.fit(tfidf_matrix, data['sentiment'])

    # Run the Flask app
    app.run(host='0.0.0.0', port=5000)
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.28.0.12:5000
INFO:werkzeug:Press CTRL+C to quit
```