Code:

```
#Importing the Dependencies
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

import nltk
nltk.download('stopwords')

# printing the stopwords in English
print(stopwords.words('english'))

#Data Pre-processing
# loading the dataset to a pandas DataFrame
news_dataset = pd.read_csv('/content/train.csv')
news_dataset.shape

# print the first 5 rows of the dataframe
news_dataset.head()
```

```
# counting the number of missing values in the dataset
news_dataset.isnull().sum()

# replacing the null values with empty string
news_dataset = news_dataset.fillna('')

# merging the author name and news title
news_dataset['content'] = news_dataset['author']+' '+news_dataset['title']
print(news_dataset['content'])

# separating the data & label
X = news_dataset.drop(columns='label', axis=1)
Y = news_dataset['label']
print(X)
print(Y)

#Stemming
port_stem = PorterStemmer()
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ',content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if
not word in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
news_dataset['content'] = news_dataset['content'].apply(stemming)
```

```
print(news_dataset['content'])

#separating the data and label
X = news_dataset['content'].values
Y = news_dataset['label'].values
print(X)
print(Y)
Y.shape

# converting the textual data to numerical data
vectorizer = TfidfVectorizer()
vectorizer.fit(X)
X = vectorizer.transform(X)
print(X)

#Splitting the dataset to training & test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
stratify=Y, random_state=2)

#Training the Model: Logistic Regression
model = LogisticRegression()
model.fit(X_train, Y_train)

#Evaluation-accuracy score
# accuracy score on the training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
#Accuracy score of the training data :  0.9865985576923076

# accuracy score on the test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

print('Accuracy score of the test data : ', test_data_accuracy)
#Accuracy score of the test data :  0.9790865384615385

#Making a Predictive System
X_new = X_test[3]
prediction = model.predict(X_new)
print(prediction)
if (prediction[0]==0):
  print('The news is Real')
else:
  print('The news is Fake')
#[0]
#The news is Real

print(Y_test[3])
#0
```