Source code

```python
import pandoas as pd

import string

import re

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report,

confusion_matrix

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import LabelEncoder

import joblib

import seaborn as sns

import matplotlib.pyplot as plt

file_path = "fake_and_real_news(1).csv"

news_df = pd.read_csv(file_path)

label_encoder = LabelEncoder()

news_df['label_encoded'] = label_encoder.fit_transform(news_df['label'])

def clean_text(text):

text = text.lower()

text = re.sub(r'<.*?>', '', text)

text = re.sub(r'http\S+', '', text)
```

```python
    text = re.sub(r'[^a-z\s]', '', text)
    return text

news_df['clean_text'] = news_df['Text'].apply(clean_text)

X_train, X_test, y_train, y_test = train_test_split(
    news_df['clean_text'], news_df['label_encoded'], test_size=0.2,
    random_state=42
)

model = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_df=0.7)),
    ('clf', LogisticRegression())
])

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

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

print("Classification Report:\n", classification_report(y_test, y_pred,
    target_names=label_encoder.classes_))

cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')
```

```python
plt.show()

label_counts = news_df['label'].value_counts()

sns.barplot(x=label_counts.index, y=label_counts.values)

plt.title('Label Distribution')

plt.ylabel('Count')

plt.xlabel('Label')

plt.show()

joblib.dump(model, "fake_news_model.pkl")

joblib.dump(label_encoder, "label_encoder.pkl")

def predict_news(text):

clean = clean_text(text)

pred = model.predict([clean])[0]

return label_encoder.inverse_transform([pred])[0]
```