In [1]:
```
import pandas as pd
import numpy as np
import re
import string

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report
```

In [2]:
```
fake = pd.read_csv("Fake.csv")
true = pd.read_csv("True.csv")
```

In [3]:
```
fake["class"] = 0
true["class"] = 1
```

In [4]:
```
fake.dropna(inplace=True)
true.dropna(inplace=True)
```

In [5]:
```
data = pd.concat([fake, true], axis=0)
data = data.drop(["title", "subject", "date"], axis=1)
data = data.sample(frac=1).reset_index(drop=True)
```

In [6]:
```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'<.*?>+', '', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub(r'\n', ' ', text)
    text = re.sub(r'\w*\d\w*', '', text)
    return text
```

In [7]:
```
data['text'] = data['text'].apply(clean_text)
```

In [8]:
```
x = data['text']
y = data['class']
```

In [9]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_st
```

In [10]:
```
vectorizer = TfidfVectorizer()
xv_train = vectorizer.fit_transform(x_train)
xv_test = vectorizer.transform(x_test)
```

In [11]:
```
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier()
}

for name, model in models.items():
    model.fit(xv_train, y_train)
    acc = model.score(xv_test, y_test)
    print(f"{name} Accuracy: {acc:.4f}")
    print(classification_report(y_test, model.predict(xv_test)))
```

```
Logistic Regression Accuracy: 0.9875
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5879
           1       0.98      0.99      0.99      5346

    accuracy                           0.99     11225
   macro avg       0.99      0.99      0.99     11225
weighted avg       0.99      0.99      0.99     11225

Decision Tree Accuracy: 0.9963
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5879
           1       1.00      1.00      1.00      5346

    accuracy                           1.00     11225
   macro avg       1.00      1.00      1.00     11225
weighted avg       1.00      1.00      1.00     11225

Random Forest Accuracy: 0.9857
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5879
           1       0.99      0.98      0.99      5346

    accuracy                           0.99     11225
   macro avg       0.99      0.99      0.99     11225
weighted avg       0.99      0.99      0.99     11225

Gradient Boosting Accuracy: 0.9958
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      5879
           1       0.99      1.00      1.00      5346

    accuracy                           1.00     11225
   macro avg       1.00      1.00      1.00     11225
weighted avg       1.00      1.00      1.00     11225
```

In [12]:
```python
def predict_fake_or_real(news_text):
    cleaned = clean_text(news_text)
    vector = vectorizer.transform([cleaned])

    print("\nPrediction from all models:")
    for name, model in models.items():
        pred = model.predict(vector)[0]
        label = "Fake News" if pred == 0 else "Real News"
        print(f"{name}: {label}")
```

In [16]:
```python
if __name__ == "__main__":
    user_input = input("\nEnter a news paragraph to test if it's fake or real:\n")
    predict_fake_or_real(user_input)
```

```
Enter a news paragraph to test if it's fake or real:
A federal appeals court ruled that California ammunition background checks were un
constitutional.

Prediction from all models:
Logistic Regression: Fake News
Decision Tree: Fake News
Random Forest: Fake News
Gradient Boosting: Fake News
```