In [124]:
```
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\COMPUMARTS\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\COMPUMARTS\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out [124]:
```
True
```

In [ ]:
```
!pip install xgboost
```

In [9]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import accuracy_score
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassi
from xgboost import XGBClassifier

data = pd.read_csv('dataset.csv')

def preprocess_text(text):
    return ' '.join([WordNetLemmatizer().lemmatize(word) for word in text.l

data['question1'] = data['question1'].apply(preprocess_text)
data['question2'] = data['question2'].apply(preprocess_text)

x = data[['question1', 'question2']]
y = data['is_duplicate']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, ra

train_pairs_q1 = x_train['question1']
train_pairs_q2 = x_train['question2']
test_pairs_q1 = x_test['question1']
test_pairs_q2 = x_test['question2']

vectorizer = TfidfVectorizer(ngram_range=(1, 2))
tfidf_train_q1 = vectorizer.fit_transform(train_pairs_q1)
tfidf_train_q2 = vectorizer.transform(train_pairs_q2)
tfidf_test_q1 = vectorizer.transform(test_pairs_q1)
tfidf_test_q2 = vectorizer.transform(test_pairs_q2)

cosine_sim_train = cosine_similarity(tfidf_train_q1, tfidf_train_q2).diagon
```

```
cosine_sim_test = cosine_similarity(tfidf_test_q1, tfidf_test_q2).diagonal(

X_train_cosine = cosine_sim_train.reshape(-1, 1)
X_test_cosine = cosine_sim_test.reshape(-1, 1)

models = {
    'Support Vector Classifier': SVC(C=0.2,gamma ="scale"),
    'Random Forest Classifier': RandomForestClassifier(n_estimators=49, min
    'Gradient Boosting Classifier': GradientBoostingClassifier(n_estimators
    'XGBoost Classifier': XGBClassifier(n_estimators=12, learning_rate=0.5)
}
results = {}
for name, model in models.items():
    model.fit(X_train_cosine, y_train)
    y_pred = model.predict(X_test_cosine)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy*100
print("Accuracy:")
for name, accuracy in results.items():
    print(f"{name}: {accuracy:.2f}%")

print("*********************************")
def compute_similarity(text1, text2):
    text1 = preprocess_text(text1)
    text2 = preprocess_text(text2)

    vectorizer = TfidfVectorizer(ngram_range=(1, 2))
    tfidf_matrix = vectorizer.fit_transform([text1, text2])
    cosine_sim = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])
    return cosine_sim[0][0]

text1 = input("Enter the first text: ")
text2 = input("Enter the second text: ")

similarity = compute_similarity(text1, text2)
similarity = similarity*100
print(f"Similarity: {similarity:.2f} %")
```

```
Accuracy:
Support Vector Classifier: 66.00%
Random Forest Classifier: 73.00%
Gradient Boosting Classifier: 68.00%
XGBoost Classifier: 71.00%
*********************************

Enter the first text:  HELL
Enter the second text:  HELL

Similarity: 100.00 %
```

```
In [ ]:
```