In [1]:
```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from gensim.models import KeyedVectors
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [2]:
```
df_train = pd.read_csv('Banks.csv')
```

In [3]:
```
df_train.dropna(inplace=True)
```

In [4]:
```
def convert_to_lowercase(text):
    return text.lower() if isinstance(text, str) else text

def remove_special_chars(text):
    text = re.sub('[^A-Za-zğüşıöçĞÜŞİÖÇ]', ' ', text)
    return text

def tokenize_text(text):
    return word_tokenize(text)

def removeUnnessWords(tokens):
    stop_words = set(stopwords.words('turkish'))
    return [word for word in tokens if word not in stop_words]

def preprocess_text(text):
    text = remove_special_chars(text)
    text = convert_to_lowercase(text)
    tokens = tokenize_text(text)
    tokens = removeUnnessWords(tokens)
    return tokens
```

In [5]:
```
df_train['text'] = df_train['text'].apply(lambda x: preprocess_text(x))
```

In [6]:
```
df_train['sentiment'] = df_train['star'].apply(lambda x: 'positive' if x in [4, 5] else 'negative')

# Separating the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df_train[['text', 'like']], df_train['sentiment'], test_size=0.2, random_state=42)

# Text and like vectorization process (we use TF-IDF and number of likes)
vectorizer = TfidfVectorizer()
X_train_text_vectorized = vectorizer.fit_transform(X_train['text'].apply(lambda x: ' '.join(x)))
X_test_text_vectorized = vectorizer.transform(X_test['text'].apply(lambda x: ' '.join(x)))

# Vectorize like numbers
X_train_like = X_train[['like']].values
X_test_like = X_test[['like']].values

# Merge vectors
X_train_combined = pd.concat([pd.DataFrame(X_train_text_vectorized.toarray()), pd.DataFrame(X_train_like)], axis=1)
X_test_combined = pd.concat([pd.DataFrame(X_test_text_vectorized.toarray()), pd.DataFrame(X_test_like)], axis=1)

# Train emotion prediction model using Random Forest classifier
clf = RandomForestClassifier()
clf.fit(X_train_combined, y_train)

#  Make predictions on the test set
y_pred = clf.predict(X_test_combined)

# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9536330223353124
              precision    recall  f1-score   support

    negative       0.95      1.00      0.98      3317
    positive       0.98      0.26      0.41       220

    accuracy                           0.95      3537
   macro avg       0.97      0.63      0.69      3537
weighted avg       0.95      0.95      0.94      3537
```

In [ ]: