Task 2

In [27]:
```
import pandas as pd
import numpy as np

df1=pd.read_csv('/Users/snehungsu/Desktop/Encryptix/Tasks/Task 2/dataset/archive/fraudTrain.csv')
df2=pd.read_csv('/Users/snehungsu/Desktop/Encryptix/Tasks/Task 2/dataset/archive/fraudTest.csv')


df1_cleaned = df1.drop(columns=['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'first', 'last', 'street','city', 'state', 'zip', 'trans_num', 'dob'])
df2_cleaned = df2.drop(columns=['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'first', 'last', 'street','city', 'state', 'zip', 'trans_num', 'dob'])

from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in df1_cleaned.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df1_cleaned[column] = label_encoders[column].fit_transform(df1_cleaned[column])

for column in df2_cleaned.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df2_cleaned[column] = label_encoders[column].fit_transform(df2_cleaned[column])

from sklearn.model_selection import train_test_split
x_train = df1_cleaned.drop(columns=['is_fraud']).values
y_train = df1_cleaned['is_fraud'].values
x_test = df1_cleaned.drop(columns=['is_fraud']).values
y_test = df1_cleaned['is_fraud'].values

from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(random_state=1)
decision_tree.fit(x_train, y_train)

y_pred = decision_tree.predict(x_test)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.3f}")
```

```
Accuracy: 1.000
```

Task 3

In [26]:
```
import pandas as pd
import numpy as np

df3=pd.read_csv('/Users/snehungsu/Desktop/Encryptix/Tasks/Task 3/Churn_Modelling.csv')

df3_cleaned = df3.drop(columns=['RowNumber','CustomerId','Surname','Geography'])

from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in df3_cleaned.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df3_cleaned[column] = label_encoders[column].fit_transform(df3_cleaned[column])

x=df3_cleaned.drop(columns=['Exited']).values
y=df3_cleaned['Exited'].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.25, random_state=0)

from sklearn.ensemble import RandomForestClassifier
```

```
classifier = RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=0)

classifier.fit(x_train, y_train)

y_pred=classifier.predict(x_test)

from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
[[1907   84]
 [ 298  211]]
0.8472
```

Task 4

In [24]:

```
import pandas as pd
import numpy as np
import nltk
nltk.download('punkt')

df4 = pd.read_csv('spam.csv',encoding='ISO-8859-1')
df4_cleaned = df4.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'])
df4_cleaned.columns = ['label','message']

from nltk.tokenize import word_tokenize
df4_cleaned['Tokenized_Message'] = df4_cleaned['message'].apply(word_tokenize)

from gensim.models import Word2Vec
model = Word2Vec(sentences=df4_cleaned['Tokenized_Message'], vector_size=100, window=5, min_count=1, workers=4)
word_embedding = model.wv['word']

def get_sentence_embedding(sentence):
    vectors = [model.wv['word'] for word in sentence if word in model.wv]
    return np.mean(vectors, axis=0) if vectors else np.zeros(100)

df4_cleaned['Embedding'] = df4_cleaned['Tokenized_Message'].apply(get_sentence_embedding)
x = np.array(df4_cleaned['Embedding'].tolist())
y=df4_cleaned.iloc[:,0:1].values

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=1)

from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(x_train,y_train)

y_pred=classifier.predict(x_test)

from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
[nltk_data] Downloading package punkt to /Users/snehungsu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
/opt/anaconda3/lib/python3.11/site-packages/sklearn/preprocessing/_label.py:116: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
[[976   0]
 [139   0]]
0.8753363228699551
```