In [12]:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

data_set = 'SAKI Exercise 1 - Transaction Classification - Data Set.csv'
df = pd.read_csv(data_set, sep=';', index_col=0, na_values='?')

################################################################################
# 1. clean and prepare data                                                    #
################################################################################

df['Buchungstext'] = df['Buchungstext'].str.replace(r'\W', ' ', regex=True)
df['Buchungstext'] = df['Buchungstext'].str.lower()
df['Verwendungszweck'] = df['Verwendungszweck'].str.replace(r'\W', ' ', regex=True)
df['Verwendungszweck'] = df['Verwendungszweck'].str.replace(r'\d', '', regex=True)
df['Verwendungszweck'] = df['Verwendungszweck'].str.replace(r'\b\w{1,2}\b', '', regex=True)
df['Verwendungszweck'] = df['Verwendungszweck'].str.lower()
df['Beguenstigter/Zahlungspflichtiger'] = df['Beguenstigter/Zahlungspflichtiger'].str.replace(r'\W', ' ', regex=True
df['Beguenstigter/Zahlungspflichtiger'] = df['Beguenstigter/Zahlungspflichtiger'].str.replace(r'\d', '', regex=True)
df['Beguenstigter/Zahlungspflichtiger'] = df['Beguenstigter/Zahlungspflichtiger'].str.lower()

df.head()
```

Out[12]:

	Auftragskonto	Buchungstag	Valutadatum	Buchungstext	Verwendungszweck	Beguenstigter/Zahlungspflichtiger	Kontonummer	
0	89990201.0	28.07.2016	28.07.2016	lohn gehalt	gehalt adorsys gmbh end end ref notpro...	adorsys gmbh co kg	7807800780	
1	89990201.0	27.07.2016	27.07.2016	miete	byladem sbt miete beuthener str end end ref...	georg tasche	DE31251900019123456780	VOHAI
2	89990201.0	21.07.2016	21.07.2016	bargeld	uhr nuernberg all eur geb eur einzahlun...	bargeld	9999900780	
3	89990201.0	20.07.2016	20.07.2016	lebensmittel getraenke	edeka neubauer nuernb nuernb kfn la...	kartenzahlung	9736000780	

Auftragskonto	Buchungstag	Valutadatum	Buchungstext	Verwendungszweck	Beguenstigter/Zahlungspflichtiger	Kontonummer

In [7]:

```
###############################################################################
# 2. label data                                                               #
###############################################################################

le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])

###############################################################################
# 3. define features to use                                                   #
###############################################################################

data = pd.DataFrame({'message': []})
data['message'] = df['Buchungstext'] + ' ' + df['Verwendungszweck'] + ' ' + df['Beguenstigter/Zahlungspflichtiger']

data['message']
```

Out[7]:

```
0      lohn   gehalt gehalt adorsys gmbh      end  en...
1      miete byladem sbt  miete beuthener str   end  e...
2      bargeld    uhr nuernberg all eur   geb eur   e...
3      lebensmittel   getraenke  edeka neubauer nuern...
4                 spontanausgabe amazon neue playstation
                           ...
204    mieteinnahmen mietzuschuss end  end ref   notp...
205    geldanlage ccbadexxx  sparen end  end ref   no...
206    lohn   gehalt gehalt adorsys gmbh      end  en...
207    geldanlage sskndexxx  einmalsparen end  end re...
208    lastschrift  einzugsermächtigung  abonnement  ...
Name: message, Length: 209, dtype: object
```

In [11]:

```
##############################################################################
# 4. transform features into a usable format and 5. train model              #
##############################################################################

X_train, X_test, y_train, y_test = train_test_split(data['message'], df['label'], random_state=42)

text_clf = Pipeline([
    ('vect', CountVectorizer()),
    ('clf', MultinomialNB()),
])
parameters = {
    'vect__ngram_range': [(1, 1), (1, 2), (2, 2)],
}
gs_clf = GridSearchCV(text_clf, parameters, cv=5, n_jobs=-1)
gs_clf.fit(X_train, y_train)

##############################################################################
# 6. evaluate model                                                          #
##############################################################################

predictions = gs_clf.predict(X_test)

print(classification_report(y_test, predictions))

for param_name in sorted(parameters.keys()):
    print("%s: %r" % (param_name, gs_clf.best_params_[param_name]))
```

```
                  precision    recall  f1-score   support

         finance       1.00      1.00      1.00         8
          income       1.00      1.00      1.00         4
         leisure       0.94      1.00      0.97        16
          living       1.00      0.71      0.83         7
         private       1.00      0.67      0.80         6
standardOfLiving       0.80      1.00      0.89        12

        accuracy                           0.92        53
       macro avg       0.96      0.90      0.92        53
    weighted avg       0.94      0.92      0.92        53

vect__ngram_range: (2, 2)
```