

In [3]:

```
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['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['Begünstigter/Zahlungspflichtiger'] = df['Begünstigter/Zahlungspflichtiger'].str.replace(r'\W', ' ', regex=True)
df['Begünstigter/Zahlungspflichtiger'] = df['Begünstigter/Zahlungspflichtiger'].str.replace(r'\d', '', regex=True)

df.head()
```

Out[3]:

	Auftragskonto	Buchungstag	Valutadatum	Buchungstext	Verwendungszweck	Begünstigter/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	BylademSbt Miete Beuthener Str End End Ref...	Georg Tasche	DE31251900019123456780 VOHA
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	Buchungstg	Valutadatum	Buchungstext	Verwendungszweck	Begünstigter/Zahlungspflichtiger	Kontonummer
---------------	------------	-------------	--------------	------------------	----------------------------------	-------------

In [4]:

```
#####
# 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['Begünstigter/Zahlungspflichtiger']

data['message']
```

Out[4]:

0	Lohn	Gehalt	Gehalt	Adorsys GmbH	End	En...
1	Miete	BylademSbt	Miete	Beuthener Str	End	E...
2	Bargeld	Uhr	Nuernberg	All Eur	Geb	Eur
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 [5]:

```
#####
# 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
0	1.00	1.00	1.00	8
1	1.00	1.00	1.00	4
2	0.94	1.00	0.97	16
3	1.00	0.71	0.83	7
4	1.00	0.67	0.80	6
5	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)

