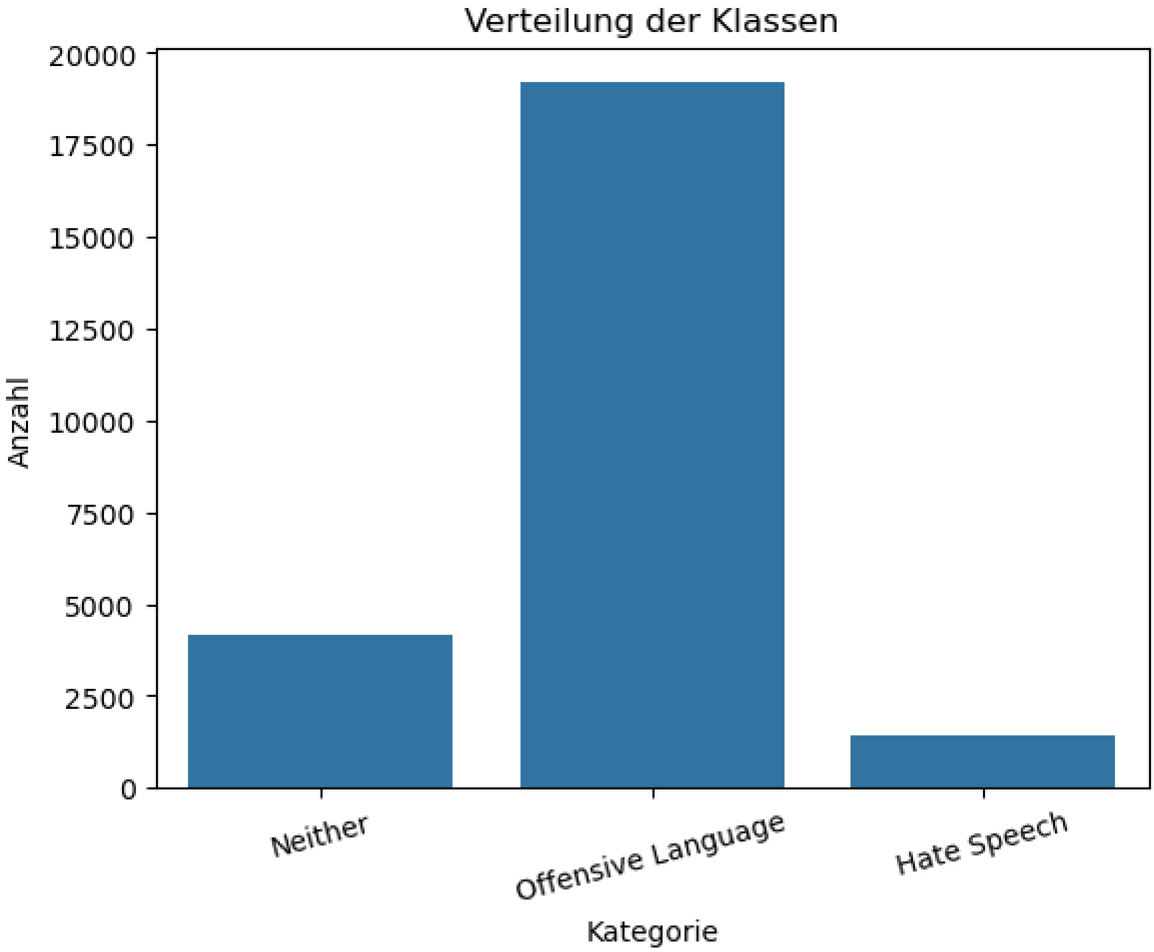
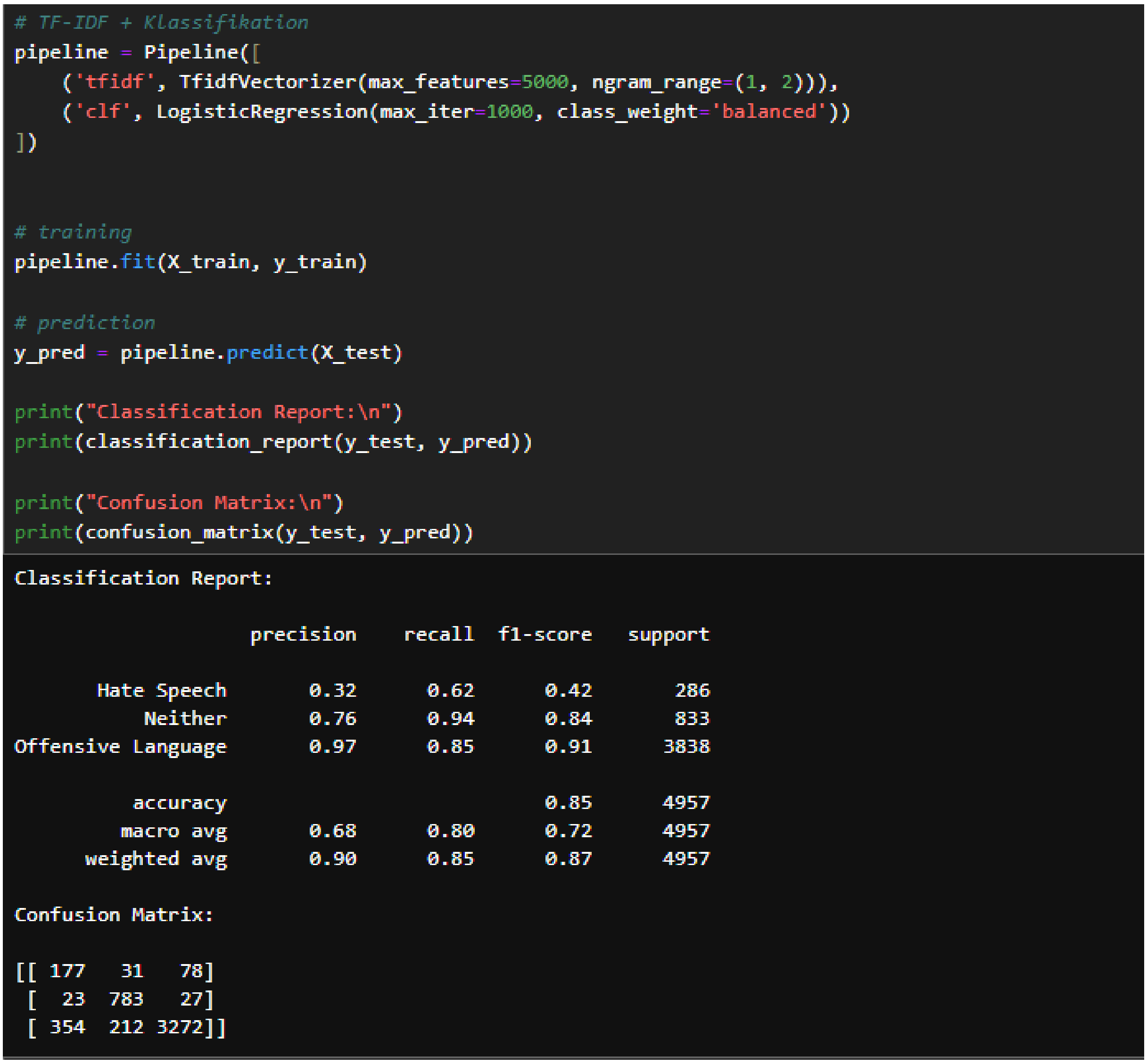
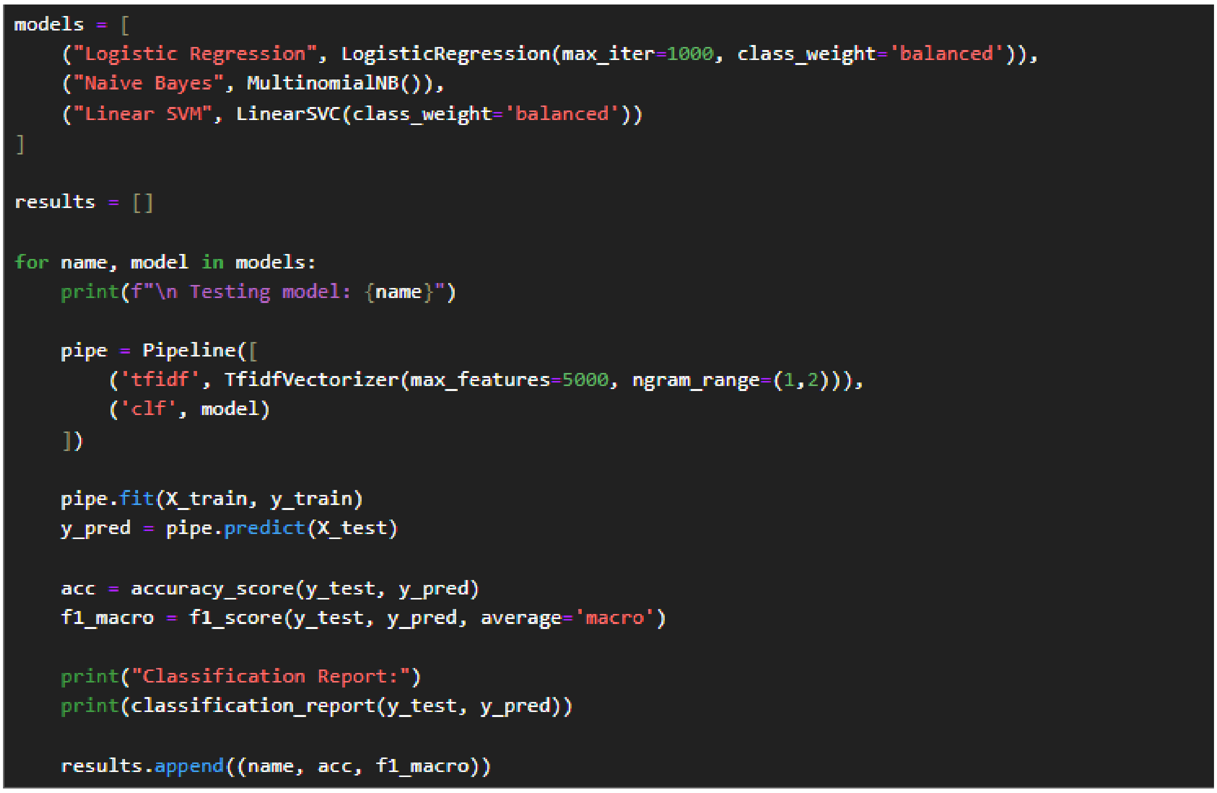

Verteilung der Klassen
Anzahl
0
2500
5000
7500
10000
12500
15000
17500
20000
Neither
Offensive Language
Hate Speech
Kategorie

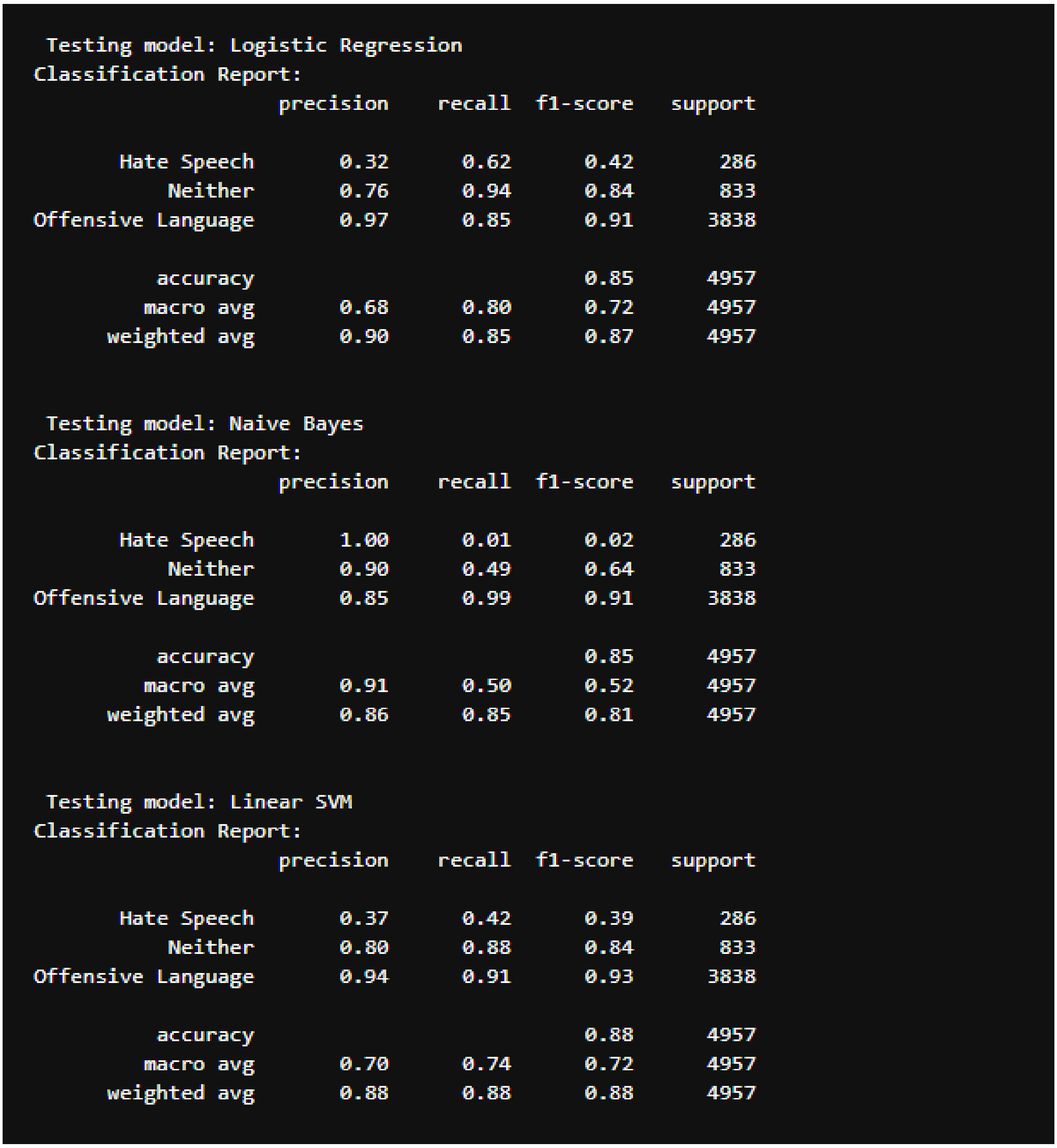


```python
# TF-IDF + Klassifikation
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(max_features=5000, ngram_range=(1, 2))),
    ('clf', LogisticRegression(max_iter=1000, class_weight='balanced'))
])


# training
pipeline.fit(X_train, y_train)

# prediction
y_pred = pipeline.predict(X_test)

print("Classification Report:\n")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:\n")
print(confusion_matrix(y_test, y_pred))
```

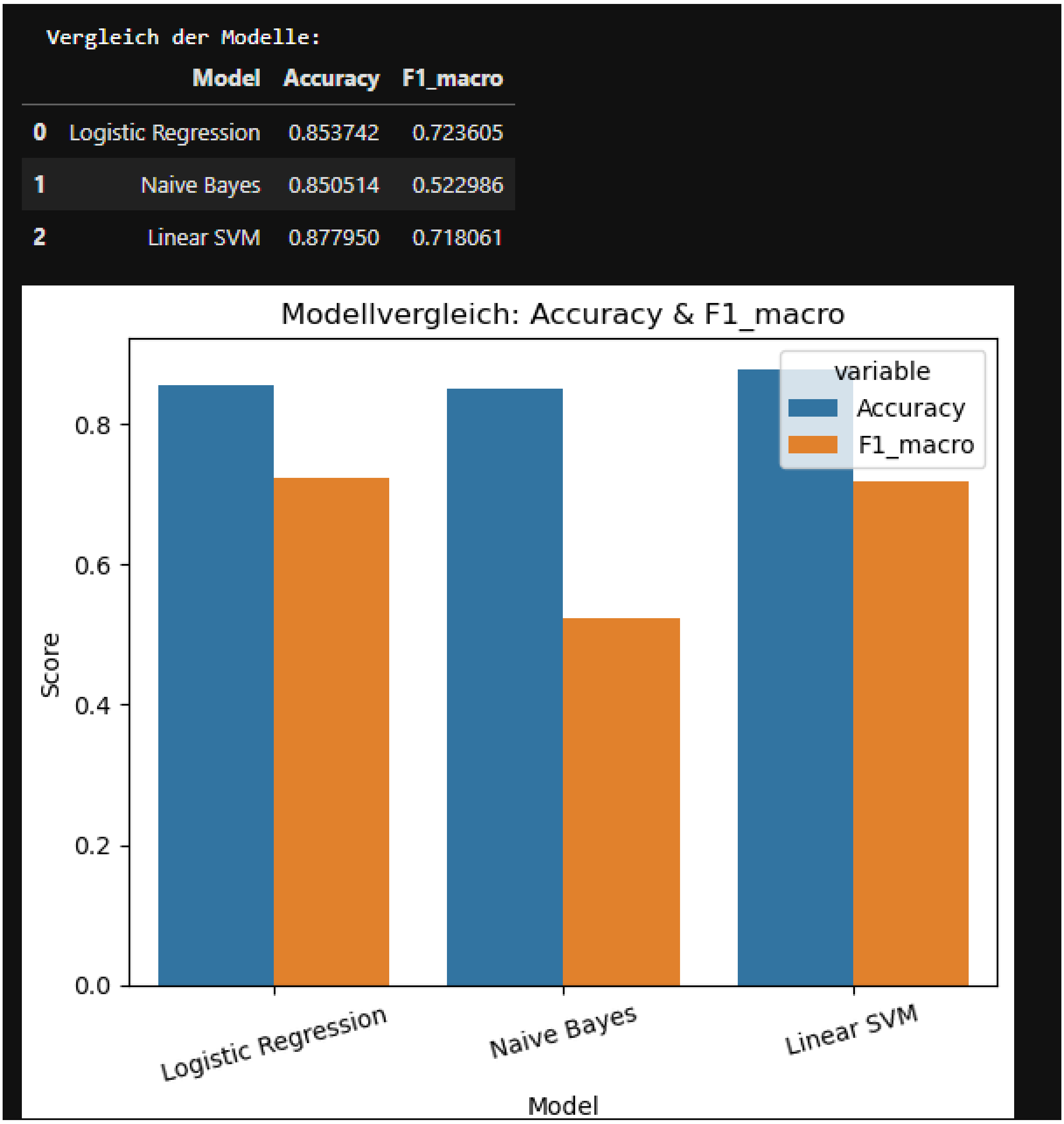
```
Classification Report:

                    precision    recall  f1-score   support

       Hate Speech       0.32      0.62      0.42       286
           Neither       0.76      0.94      0.84       833
Offensive Language       0.97      0.85      0.91      3838

          accuracy                           0.85      4957
         macro avg       0.68      0.80      0.72      4957
      weighted avg       0.90      0.85      0.87      4957

Confusion Matrix:

[[ 177   31   78]
 [  23  783   27]
 [ 354  212 3272]]
```



```python
models = [
    ("Logistic Regression", LogisticRegression(max_iter=1000, class_weight='balanced')),
    ("Naive Bayes", MultinomialNB()),
    ("Linear SVM", LinearSVC(class_weight='balanced'))
]

results = []

for name, model in models:
    print(f"\n Testing model: {name}")

    pipe = Pipeline([
        ('tfidf', TfidfVectorizer(max_features=5000, ngram_range=(1,2))),
        ('clf', model)
    ])

    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    f1_macro = f1_score(y_test, y_pred, average='macro')

    print("Classification Report:")
    print(classification_report(y_test, y_pred))

    results.append((name, acc, f1_macro))
```

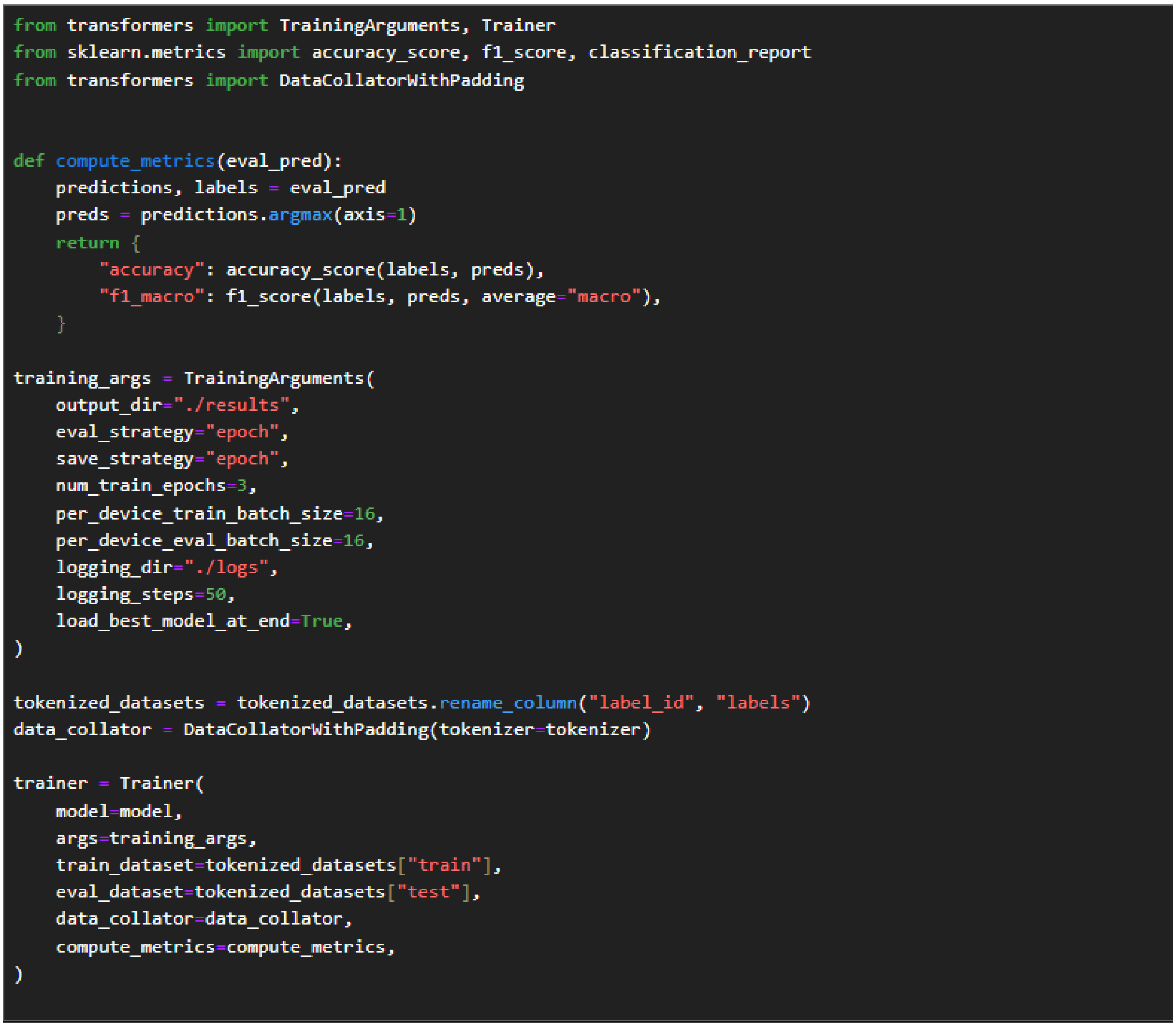


```
 Testing model: Logistic Regression
Classification Report:
                    precision    recall  f1-score   support

       Hate Speech       0.32      0.62      0.42       286
           Neither       0.76      0.94      0.84       833
Offensive Language       0.97      0.85      0.91      3838

          accuracy                           0.85      4957
         macro avg       0.68      0.80      0.72      4957
      weighted avg       0.90      0.85      0.87      4957


 Testing model: Naive Bayes
Classification Report:
                    precision    recall  f1-score   support

       Hate Speech       1.00      0.01      0.02       286
           Neither       0.90      0.49      0.64       833
Offensive Language       0.85      0.99      0.91      3838

          accuracy                           0.85      4957
         macro avg       0.91      0.50      0.52      4957
      weighted avg       0.86      0.85      0.81      4957


 Testing model: Linear SVM
Classification Report:
                    precision    recall  f1-score   support

       Hate Speech       0.37      0.42      0.39       286
           Neither       0.80      0.88      0.84       833
Offensive Language       0.94      0.91      0.93      3838

          accuracy                           0.88      4957
         macro avg       0.70      0.74      0.72      4957
      weighted avg       0.88      0.88      0.88      4957
```

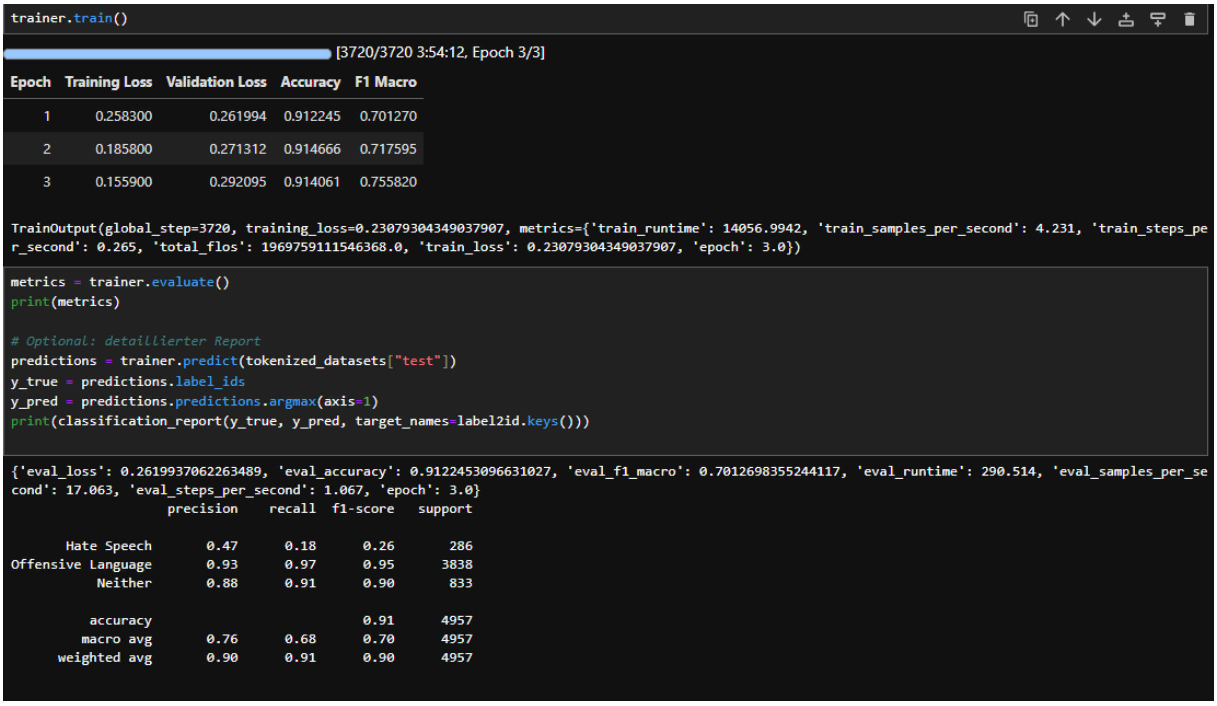


```
Vergleich der Modelle:
```

	Model	Accuracy	F1_macro
0	Logistic Regression	0.853742	0.723605
1	Naive Bayes	0.850514	0.522986
2	Linear SVM	0.877950	0.718061



```python
from transformers import TrainingArguments, Trainer
from sklearn.metrics import accuracy_score, f1_score, classification_report
from transformers import DataCollatorWithPadding


def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    preds = predictions.argmax(axis=1)
    return {
        "accuracy": accuracy_score(labels, preds),
        "f1_macro": f1_score(labels, preds, average="macro"),
    }

training_args = TrainingArguments(
    output_dir="./results",
    eval_strategy="epoch",
    save_strategy="epoch",
    num_train_epochs=3,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    logging_dir="./logs",
    logging_steps=50,
    load_best_model_at_end=True,
)

tokenized_datasets = tokenized_datasets.rename_column("label_id", "labels")
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)
```



```python
trainer.train()
```

[3720/3720 3:54:12, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Macro
1	0.258300	0.261994	0.912245	0.701270
2	0.185800	0.271312	0.914666	0.717595
3	0.155900	0.292095	0.914061	0.755820

```
TrainOutput(global_step=3720, training_loss=0.23079304349037907, metrics={'train_runtime': 14056.9942, 'train_samples_per_second': 4.231, 'train_steps_pe
r_second': 0.265, 'total_flos': 1969759111546368.0, 'train_loss': 0.23079304349037907, 'epoch': 3.0})
```

```python
metrics = trainer.evaluate()
print(metrics)

# Optional: detaillierter Report
predictions = trainer.predict(tokenized_datasets["test"])
y_true = predictions.label_ids
y_pred = predictions.predictions.argmax(axis=1)
print(classification_report(y_true, y_pred, target_names=label2id.keys()))
```

```
{'eval_loss': 0.2619937062263489, 'eval_accuracy': 0.9122453096631027, 'eval_f1_macro': 0.7012698355244117, 'eval_runtime': 290.514, 'eval_samples_per_se
cond': 17.063, 'eval_steps_per_second': 1.067, 'epoch': 3.0}
                    precision    recall  f1-score   support

       Hate Speech       0.47      0.18      0.26       286
Offensive Language       0.93      0.97      0.95      3838
           Neither       0.88      0.91      0.90       833

          accuracy                           0.91      4957
         macro avg       0.76      0.68      0.70      4957
      weighted avg       0.90      0.91      0.90      4957
```