LU02.L02 - ML Programmierung

Voraussetzung

```
pip install pandas scikit-learn joblib
```

Python-Skript: ml_basics_shop.py

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import joblib

# ----------------------------
# Daten laden
# ----------------------------
data = pd.read_csv("shop_data.csv")
X = data.drop("buy", axis=1)
y = data["buy"]

# ----------------------------
# Train / Test Split
# ----------------------------
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,
random_state=42)

# ----------------------------
# Modell 1: Logistische Regression
# ----------------------------
log_reg_pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("model", LogisticRegression())
])

log_reg_pipeline.fit(X_train, y_train)
y_pred_lr = log_reg_pipeline.predict(X_test)
#
print("Logistische Regression")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_lr))
print("Classification Report:\n", classification_report(y_test, y_pred_lr))
```

```python
# ------------------------------
# Modell 2: Decision Tree
# ------------------------------
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)

print("\nDecision Tree")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tree))
print("Classification Report:\n", classification_report(y_test,
y_pred_tree))

# ------------------------------
# Bestes Modell speichern
# ------------------------------
joblib.dump(log_reg_pipeline, "best_model.joblib")

# ------------------------------
# Neue Vorhersage
# ------------------------------
new_customer = pd.DataFrame([{
    "age": 32,
    "past_purchases": 5,
    "minutes_on_page": 6.5
}])

loaded_model = joblib.load("best_model.joblib")
prediction = loaded_model.predict(new_customer)

print("\nVorhersage fuer neuen Kunden:", prediction[0])
```

Modellvergleich

Kriterium	**Logistische Regression**	**Decision Tree**
Interpretierbarkeit	hoch	mittel
Overfitting-Gefahr	gering	hoch
Skalierung	nötig ja	nein
Didaktisch	sinnvoll sehr	ja

Fazit

- Bei kleinen, sauberen Datensätzen ist die Logistische Regression meist stabiler.
- Decision Trees sind anschaulich, aber übermotiviert – sie merken sich gern alles bzw. „lernen einen Fall auswendig“.