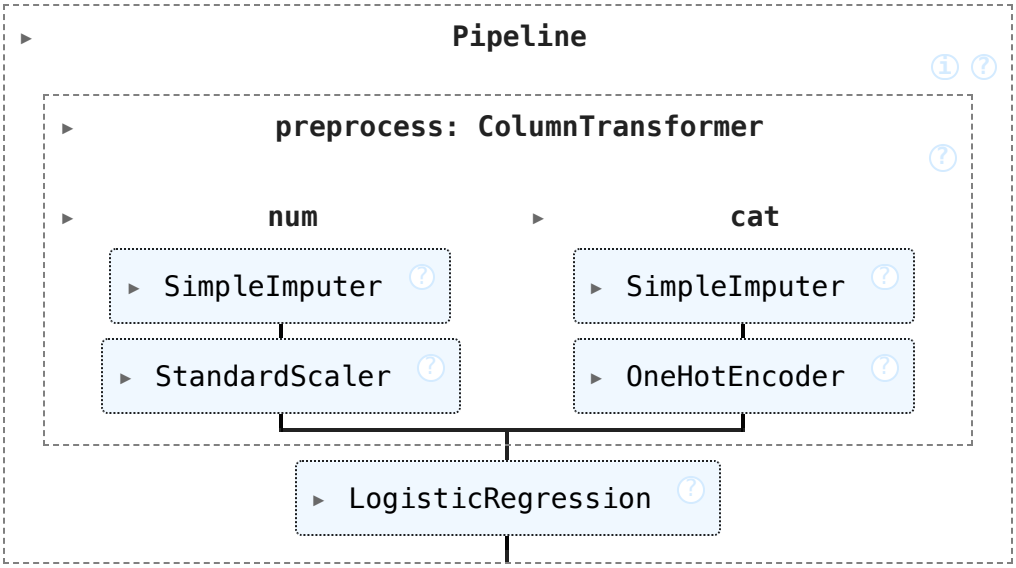
In [3]:
```
"""
Ejemplo mínimo de uso de scikit-learn Pipeline con pasos:
1) Extracción de datos (desde CSV).
2) Filtrado de registros fuera de rango.
3) Tipado y preprocesamiento por tipo de variable (numéricas y categóricas).
4) Separación del dataset para ML y entrenamiento de un modelo.
Requisitos: pandas, scikit-learn.
"""
from pathlib import Path
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# ----------------------------
# 1) EXTRACCIÓN DE DATOS
# ----------------------------
DATA_PATH =  "./synthetic_customers.csv"
df = pd.read_csv(DATA_PATH)

# ----------------------------
# 2) FILTRADO
#    - Mantener edades 18–80
#    - income_gtq > 0
#    - monthly_visits en [0, 30]
# ----------------------------
mask = (
    (df["age"].between(18, 80)) &
    (df["income_gtq"] > 0) &
    (df["monthly_visits"].between(0, 30))
)
df = df.loc[mask].reset_index(drop=True)

# ----------------------------
# 3) TIPOS DE VARIABLES
#    - Definir columnas numéricas vs categóricas
#    - Preprocesar: imputación, escalado y one-hot
# ----------------------------
numeric_features = ["age", "income_gtq", "monthly_visits"]
categorical_features = ["city", "has_kids"]

numeric_pipeline = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_pipeline = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_pipeline, numeric_features),
        ("cat", categorical_pipeline, categorical_features),
    ]
)

# ------------------------------
# 4) SEPARACIÓN DEL DATASET PARA ML
#    - X, y
#    - train/test split
#    - Pipeline completo con modelo final
# ------------------------------
X = df[numeric_features + categorical_features]
y = df["churn"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

clf = Pipeline(steps=[
    ("preprocess", preprocessor),
    ("model", LogisticRegression(max_iter=1000))
])

# Entrenar
clf.fit(X_train, y_train)

# Evaluar
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc:.3f}")
print(classification_report(y_test, y_pred, zero_division=0))
```

```
Accuracy: 0.947
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        13
           1       0.86      1.00      0.92         6

    accuracy                           0.95        19
   macro avg       0.93      0.96      0.94        19
weighted avg       0.95      0.95      0.95        19
```

In [8]:
```
from sklearn import set_config

set_config(display='diagram')

clf
```

Out[8]:


Pipeline
preprocess: ColumnTransformer
num
cat
SimpleImputer
StandardScaler
SimpleImputer
OneHotEncoder
LogisticRegression