```
!pip install optuna
```

```
Collecting optuna
  Downloading optuna-4.5.0-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: alembic>=1.5.0 in /usr/local/lib/python3.12/dist-packages (from optuna) (1.16.5
Collecting colorlog (from optuna)
  Downloading colorlog-6.9.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from optuna) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from optuna) (25.0)
Requirement already satisfied: sqlalchemy>=1.4.2 in /usr/local/lib/python3.12/dist-packages (from optuna) (2.0
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from optuna) (4.67.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.12/dist-packages (from optuna) (6.0.2)
Requirement already satisfied: Mako in /usr/local/lib/python3.12/dist-packages (from alembic>=1.5.0->optuna) (
Requirement already satisfied: typing-extensions>=4.12 in /usr/local/lib/python3.12/dist-packages (from alembi
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-packages (from sqlalchemy>=1.4.2-
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.12/dist-packages (from Mako->alembi
Downloading optuna-4.5.0-py3-none-any.whl (400 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 400.9/400.9 kB 24.9 MB/s eta 0:00:00
Downloading colorlog-6.9.0-py3-none-any.whl (11 kB)
Installing collected packages: colorlog, optuna
Successfully installed colorlog-6.9.0 optuna-4.5.0
```

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.8.1-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.12/dist-packages (from category_encoder
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.12/dist-packages (from category_encoder
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from category_encoders
Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from category_e
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from category_encoders
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.12/dist-packages (from category_en
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.5->ca
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.5->
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=1.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-le
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.12/dist-packages (from statsmodels>=0
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.
Downloading category_encoders-2.8.1-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 6.2 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.1
```

```
!pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.8-cp312-cp312-manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-packages (from catboost) (0.21)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.12/dist-packages (from catboost) (
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from catboost) (1.16.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->cat
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->c
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->c
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catbo
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->ca
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->c
Requirement already satisfied: tenacity>=6.2.0 ✦ 'usr/local/lib/python3.12/dist-packages (from plotly->catboo
Downloading catboost-1.2.8-cp312-cp312-manylinu. ₄4_x86_64.whl (99.2 MB)
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 MB 15.9 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.8
```

```
!pip install lifelines
```

```
Collecting lifelines
  Downloading lifelines-0.30.0-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.12/dist-packages (from lifelines) (2.0.
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from lifelines) (1.16.
Requirement already satisfied: pandas>=2.1 in /usr/local/lib/python3.12/dist-packages (from lifelines) (2.2.2)
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.12/dist-packages (from lifelines) (3.
Requirement already satisfied: autograd>=1.5 in /usr/local/lib/python3.12/dist-packages (from lifelines) (1.8.
Collecting autograd-gamma>=0.3 (from lifelines)
  Downloading autograd-gamma-0.5.0.tar.gz (4.0 kB)
  Preparing metadata (setup.py) ... done
Collecting formulaic>=0.2.2 (from lifelines)
  Downloading formulaic-1.2.1-py3-none-any.whl.metadata (7.0 kB)
Collecting interface-meta>=1.2.0 (from formulaic>=0.2.2->lifelines)
  Downloading interface_meta-1.3.0-py3-none-any.whl.metadata (6.7 kB)
Requirement already satisfied: narwhals>=1.17 in /usr/local/lib/python3.12/dist-packages (from formulaic>=0.2.
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.12/dist-packages (from formu
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.12/dist-packages (from formulaic>=0.2.2->l
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.0->
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.0->lif
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotli
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=2.1->life
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=2.1->li
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7-
Downloading lifelines-0.30.0-py3-none-any.whl (349 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 349.3/349.3 kB 22.8 MB/s eta 0:00:00
Downloading formulaic-1.2.1-py3-none-any.whl (117 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.3/117.3 kB 10.8 MB/s eta 0:00:00
Downloading interface_meta-1.3.0-py3-none-any.whl (14 kB)
Building wheels for collected packages: autograd-gamma
  Building wheel for autograd-gamma (setup.py) ... done
  Created wheel for autograd-gamma: filename=autograd_gamma-0.5.0-py3-none-any.whl size=4030 sha256=d1933879df
  Stored in directory: /root/.cache/pip/wheels/50/37/21/0a719b9d89c635e89ff24bd93b862882ad675279552013b2fb
Successfully built autograd-gamma
Installing collected packages: interface-meta, autograd-gamma, formulaic, lifelines
Successfully installed autograd-gamma-0.5.0 formulaic-1.2.1 interface-meta-1.3.0 lifelines-0.30.0
```

```
# train.csv upload

from google.colab import files
train_uploaded = files.upload()
```

Choose Files train.csv

train.csv(text/csv) - 1800971 bytes, last modified: 10/2/2025 - 100% done

```
Saving train.csv to train.csv
```

```
# test.csv upload

test_uploaded = files.upload()
```

Choose Files test.csv

test.csv(text/csv) - 38060 bytes, last modified: 10/2/2025 - 100% done

```
Saving test.csv to test.csv
```

```
TRAIN_FILENAME = next(iter(train_uploaded))
TEST_FILENAME = next(iter(test_uploaded))
```

```
TRAIN_PATH = "/content/" + TRAIN_FILENAME
TEST_PATH = "/content/" + TEST_FILENAME
```

```
# import the necessary libraries

import os
import math
from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from datetime import datetime, timedelta
from dateutil.relativedelta import relativedelta
import matplotlib.pyplot as plt

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.utils.validation import check_is_fitted

import optuna
import shap
import category_encoders as ce

import lightgbm as lgb
import xgboost as xgb
from catboost import CatBoostRegressor, Pool

import lifelines
from lifelines import CoxPHFitter

import warnings
warnings.filterwarnings('ignore')
```

```
def load_data(train_path, test_path):
    df_train = pd.read_csv(train_path)
    df_test  = pd.read_csv(test_path)

    # Standard column names
    df_train = df_train.rename(columns={
        'Customer ID': 'CustomerID',
        'Transaction Date': 'TransactionDate',
        'Product Category': 'ProductCategory',
        'Purchase Amount': 'PurchaseAmount'
    })
    df_test = df_test.rename(columns={
        'Customer ID': 'CustomerID',
        'Transaction Date': 'TransactionDate',
        'Product Category': 'ProductCategory',
        'Purchase Amount': 'PurchaseAmount'
    })

    # Convert date to datetime
    df_train['TransactionDate'] = pd.to_datetime(df_train['TransactionDate'])
    df_test['TransactionDate']  = pd.to_datetime(df_test['TransactionDate'])
```

```
    # Sort
    df_train = df_train.sort_values(['CustomerID','TransactionDate']).reset_index(drop=True)
    df_test  = df_test.sort_values(['CustomerID','TransactionDate']).reset_index(drop=True)

    return df_train, df_test
```

```
df_train, df_test = load_data(TRAIN_PATH, TEST_PATH)
```

```
print(df_train.head(3))
print(df_train.dtypes)
print(f"Train rows: {len(df_train):,} | Test rows: {len(df_test):,}")
```

```
   CustomerID TransactionDate     ProductCategory  PurchaseAmount
0         636      2021-07-25  Non Finished Goods          469.08
1         636      2021-07-25       Personal Care         8413.86
2         636      2021-11-28       Personal Care         8801.68
CustomerID                 int64
TransactionDate    datetime64[ns]
ProductCategory            object
PurchaseAmount            float64
dtype: object
Train rows: 41,984 | Test rows: 889
```

```
def add_basic_customer_ordering(df):
    df = df.sort_values(['CustomerID','TransactionDate']).copy()
    df['TxnIdx'] = df.groupby('CustomerID').cumcount()
    df['TxnCount'] = df.groupby('CustomerID')['CustomerID'].transform('count')
    return df
```

```
def add_gap_features(df):
    df['PrevDate'] = df.groupby('CustomerID')['TransactionDate'].shift(1)
    df['NextDate'] = df.groupby('CustomerID')['TransactionDate'].shift(-1)
    df['DaysSincePrev'] = (df['TransactionDate'] - df['PrevDate']).dt.days
    df['DaysToNext']    = (df['NextDate'] - df['TransactionDate']).dt.days
    for k in [1,2,3,4,5]:
        df[f'GapLag{k}'] = df.groupby('CustomerID')['DaysSincePrev'].shift(k)
    for w in [2,3,5,10]:
        grp = df.groupby('CustomerID')['DaysSincePrev']
        df[f'GapRollMean_w{w}'] = grp.transform(lambda s: s.rolling(w, min_periods=1).mean())
        df[f'GapRollStd_w{w}']  = grp.transform(lambda s: s.rolling(w, min_periods=2).std())
        df[f'GapRollMed_w{w}']  = grp.transform(lambda s: s.rolling(w, min_periods=1).median())
    agg = df.groupby('CustomerID')['DaysSincePrev'].agg(['mean','median','std','min','max']).add_prefix('Cust
    df = df.merge(agg, left_on='CustomerID', right_index=True, how='left')
    return df
```

```
print(df_train.head(3))
```

```
   CustomerID TransactionDate     ProductCategory  PurchaseAmount
0         636      2021-07-25  Non Finished Goods          469.08
1         636      2021-07-25       Personal Care         8413.86
2         636      2021-11-28       Personal Care         8801.68
```

```
def add_amount_features(df):
    for k in [1,2,3]:
        df[f'AmtLag{k}'] = df.groupby('CustomerID')['PurchaseAmount'].shift(k)
    for w in [2,3,5,10]:
        grp = df.groupby('CustomerID')['PurchaseAmount']
        df[f'AmtRollMean_w{w}'] = grp.transform(lambda s: s.rolling(w, min_periods=1).mean())
        df[f'AmtRollStd_w{w}']  = grp.transform(lambda s: s.rolling(w, min_periods=2).std())
        df[f'AmtRollMed_w{w}']  = grp.transform(lambda s: s.rolling(w, min_periods=1).median())
    cust_amt = df.groupby('CustomerID')['PurchaseAmount'].agg(['mean','median','std','sum','max','min']).add_
```

```
    df = df.merge(cust_amt, left_on='CustomerID', right_index=True, how='left')
    return df
```

```
def _entropy(counts):
    total = counts.sum()
    if total <= 0: return 0.0
    p = counts / total
    return float(-(p * np.log(p + 1e-12)).sum())
```

```
def add_category_features(df, top_n_onehot=20):
    cust_cat_counts = df.groupby(['CustomerID','ProductCategory']).size().rename('CustCatCount').reset_index(
    df = df.merge(cust_cat_counts, on=['CustomerID','ProductCategory'], how='left')
    tmp = df.groupby(['CustomerID','ProductCategory']).size().rename('cnt').reset_index()
    div = tmp.groupby('CustomerID')['ProductCategory'].nunique().rename('CustCatUnique')
    ent = tmp.groupby('CustomerID')['cnt'].apply(_entropy).rename('CustCatEntropy')
    df = df.merge(div, on='CustomerID', how='left').merge(ent, on='CustomerID', how='left')

    gfreq = df['ProductCategory'].value_counts(normalize=True).to_dict()
    df['CatGlobalFreq'] = df['ProductCategory'].map(gfreq)
    df['CustCatFreq'] = df['CustCatCount'] / df.groupby('CustomerID')['CustCatCount'].transform('sum')
    topN = df['ProductCategory'].value_counts().head(top_n_onehot).index.tolist()
    df['ProductCategory_OH'] = df['ProductCategory'].where(df['ProductCategory'].isin(topN), 'OTHER')
    oh = pd.get_dummies(df['ProductCategory_OH'], prefix='Cat')
    df = pd.concat([df, oh], axis=1)
    return df
```

```
def add_time_features(df):
    d = df['TransactionDate']
    df['TXN_Year'] = d.dt.year
    df['TXN_Month'] = d.dt.month
    df['TXN_Day'] = d.dt.day
    df['TXN_DayOfWeek'] = d.dt.dayofweek
    df['TXN_IsWeekend'] = df['TXN_DayOfWeek'].isin([5,6]).astype(int)
    df['TXN_WeekOfYear'] = d.dt.isocalendar().week.astype(int)
    df['TXN_Quarter'] = d.dt.quarter
    return df
```

```
def build_features(df):
    df = add_basic_customer_ordering(df)
    df = add_gap_features(df)
    df = add_amount_features(df)
    df = add_category_features(df, top_n_onehot=20)
    df = add_time_features(df)
    return df
```

```
df_train = build_features(df_train)
df_test  = build_features(df_test)
```

```
df_train_labeled = df_train[~df_train['DaysToNext'].isna()].copy()
print("Train labeled rows:", len(df_train_labeled))
```

```
Train labeled rows: 41354
```

```
print(df_train.head(3))
```

```
   CustomerID TransactionDate     ProductCategory  PurchaseAmount  TxnIdx  \
0         636      2021-07-25  Non Finished Goods          469.08       0
1         636      2021-07-25       Personal Care         8413.86       1
2         636      2021-11-28       Personal Care         8801.68       2

   TxnCount   PrevDate   NextDate  DaysSincePrev  DaysToNext  ...  \
0         4        NaT 2021-07-25            NaN         0.0  ...
```

```
1         4 2021-07-25 2021-11-28            0.0       126.0  ...
2         4 2021-07-25 2021-11-28          126.0         0.0  ...

   Cat_Shrink Film  Cat_Specialty Foods - Other  Cat_Unknown  TXN_Year  \
0            False                         False        False      2021
1            False                         False        False      2021
2            False                         False        False      2021

   TXN_Month  TXN_Day  TXN_DayOfWeek  TXN_IsWeekend  TXN_WeekOfYear  \
0          7       25              6              1              29
1          7       25              6              1              29
2         11       28              6              1              47

   TXN_Quarter
0            3
1            3
2            4

[3 rows x 87 columns]
```

```
# Time based split and strong baselines

def time_based_split(df, date_col='TransactionDate', holdout_frac=0.2):
    cutoff = df[date_col].quantile(1 - holdout_frac)
    return df[df[date_col] < cutoff].copy(), df[df[date_col] >= cutoff].copy(), cutoff
```

```
train_part, valid_part, cutoff_date = time_based_split(df_train_labeled, 'TransactionDate', 0.2)
print(f"Cutoff={cutoff_date}, train={len(train_part)}, valid={len(valid_part)}")
```

```
Cutoff=2024-02-04 00:00:00, train=33016, valid=8338
```

```
def evaluate(y_true, y_pred, label):
    mae  = mean_absolute_error(y_true, y_pred)
    rmse = math.sqrt(mean_squared_error(y_true, y_pred))
    print(f"{label:28s} -> MAE: {mae:.3f} | RMSE: {rmse:.3f}")
    return mae, rmse
```

```
last_gap_valid = valid_part['DaysSincePrev'].fillna(valid_part['DaysSincePrev'].median())
cust_mean_gap_valid = valid_part.groupby('CustomerID')['DaysSincePrev'].transform('mean')
cust_mean_gap_valid = cust_mean_gap_valid.fillna(valid_part['DaysSincePrev'].median())
global_median = valid_part['DaysToNext'].median()
global_median_valid = pd.Series(global_median, index=valid_part.index)

_ = evaluate(valid_part['DaysToNext'], last_gap_valid,        "Baseline: last gap")
_ = evaluate(valid_part['DaysToNext'], cust_mean_gap_valid,   "Baseline: customer mean gap")
_ = evaluate(valid_part['DaysToNext'], global_median_valid,   "Baseline: global median gap")
```

```
Baseline: last gap           -> MAE: 16.633 | RMSE: 43.688
Baseline: customer mean gap  -> MAE: 11.190 | RMSE: 27.797
Baseline: global median gap  -> MAE: 10.267 | RMSE: 24.725
```

```
# Feature selection & imputation

base_num = [
    'DaysSincePrev','TxnIdx','TxnCount',
    'GapLag1','GapLag2','GapLag3','GapLag4','GapLag5',
    'GapRollMean_w2','GapRollStd_w2','GapRollMed_w2',
    'GapRollMean_w3','GapRollStd_w3','GapRollMed_w3',
    'GapRollMean_w5','GapRollStd_w5','GapRollMed_w5',
    'GapRollMean_w10','GapRollStd_w10','GapRollMed_w10',
    'CustGap_mean','CustGap_median','CustGap_std','CustGap_min','CustGap_max',
    'PurchaseAmount','AmtLag1','AmtLag2','AmtLag3',
    'AmtRollMean_w2','AmtRollStd_w2','AmtRollMed_w2',
```

```
    'AmtRollMean_w3','AmtRollStd_w3','AmtRollMed_w3',
    'AmtRollMean_w5','AmtRollStd_w5','AmtRollMed_w5',
    'AmtRollMean_w10','AmtRollStd_w10','AmtRollMed_w10',
    'CustAmt_mean','CustAmt_median','CustAmt_std','CustAmt_sum','CustAmt_max','CustAmt_min',
    'CustCatCount','CustCatUnique','CustCatEntropy','CatGlobalFreq','CustCatFreq',
    'TXN_Year','TXN_Month','TXN_Day','TXN_DayOfWeek','TXN_IsWeekend','TXN_WeekOfYear','TXN_Quarter'
]
oh_cols = [c for c in df_train.columns if c.startswith('Cat_')]
feature_cols = base_num + oh_cols

X_train = train_part[feature_cols].copy()
y_train = train_part['DaysToNext'].astype(float).copy()
X_valid = valid_part[feature_cols].copy()
y_valid = valid_part['DaysToNext'].astype(float).copy()

imputer = SimpleImputer(strategy='median')
X_train_imp = pd.DataFrame(imputer.fit_transform(X_train), columns=feature_cols, index=X_train.index)
X_valid_imp = pd.DataFrame(imputer.transform(X_valid), columns=feature_cols, index=X_valid.index)
```

```
def objective(trial):
    params = {
        'objective': 'regression',
        'metric': 'mae',
        'boosting_type': trial.suggest_categorical('boosting', ['gbdt','dart']),
        'num_leaves': trial.suggest_int('num_leaves', 31, 256),
        'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.2, log=True),
        'feature_fraction': trial.suggest_float('feature_fraction', 0.6, 1.0),
        'bagging_fraction': trial.suggest_float('bagging_fraction', 0.6, 1.0),
        'bagging_freq': trial.suggest_int('bagging_freq', 0, 10),
        'min_data_in_leaf': trial.suggest_int('min_data_in_leaf', 20, 200),
        'lambda_l1': trial.suggest_float('lambda_l1', 0.0, 10.0),
        'lambda_l2': trial.suggest_float('lambda_l2', 0.0, 10.0),
        'max_depth': trial.suggest_int('max_depth', -1, 12),
        'verbosity': -1,
        'num_threads': 0
    }

    dtrain = lgb.Dataset(X_train_imp, label=y_train)
    dvalid = lgb.Dataset(X_valid_imp, label=y_valid, reference=dtrain)

    cbs = [
        lgb.early_stopping(stopping_rounds=200),
        lgb.log_evaluation(period=0)
    ]

    try:
        model = lgb.train(
            params,
            dtrain,
            valid_sets=[dtrain, dvalid],
            valid_names=['train','valid'],
            num_boost_round=5000,
            callbacks=cbs
        )
        preds = model.predict(X_valid_imp, num_iteration=model.best_iteration)
    except TypeError:
        model = lgb.LGBMRegressor(
            **{k:v for k,v in params.items() if k not in ['objective','metric','verbosity','num_threads']},
            objective='regression'
        )
        model.fit(
            X_train_imp, y_train,
            eval_set=[(X_valid_imp, y_valid)],
            verbose=False
        )
```

```
            preds = model.predict(X_valid_imp)

        return mean_absolute_error(y_valid, preds)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=40, show_progress_bar=True)

best_params = study.best_trial.params
best_params.update({'objective':'regression','metric':'mae','verbosity':-1,'num_threads':0})
best_params
```

```
[I 2025-10-02 06:14:17,746] A new study created in memory with name: no-name-646cbe76-e452-4312-afb0-b484fcf47
Best trial: 12. Best value: 9.60229: 100%                                40/40 [17:51<00:00, 22.61s/it]
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[432]   train's l1: 11.189  valid's l1: 9.91412
[I 2025-10-02 06:14:21,271] Trial 0 finished with value: 9.914116618531951 and parameters: {'boosting': 'gbdt'
[I 2025-10-02 06:15:45,685] Trial 1 finished with value: 9.74246508207995 and parameters: {'boosting': 'dart',
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[231]   train's l1: 11.3311 valid's l1: 9.76906
[I 2025-10-02 06:15:47,553] Trial 2 finished with value: 9.769063413176474 and parameters: {'boosting': 'gbdt'
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[161]   train's l1: 10.8977 valid's l1: 9.71993
[I 2025-10-02 06:15:52,521] Trial 3 finished with value: 9.719932102508647 and parameters: {'boosting': 'gbdt'
[I 2025-10-02 06:20:12,645] Trial 4 finished with value: 10.102858273385147 and parameters: {'boosting': 'dart
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[435]   train's l1: 11.3795 valid's l1: 9.87316
[I 2025-10-02 06:20:13,581] Trial 5 finished with value: 9.87315750435783 and parameters: {'boosting': 'gbdt',
[I 2025-10-02 06:21:00,422] Trial 6 finished with value: 10.540698528317083 and parameters: {'boosting': 'dart
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[480]   train's l1: 11.2144 valid's l1: 9.86389
[I 2025-10-02 06:21:01,908] Trial 7 finished with value: 9.863894916251235 and parameters: {'boosting': 'gbdt'
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[488]   train's l1: 11.1808 valid's l1: 9.67694
[I 2025-10-02 06:21:03,969] Trial 8 finished with value: 9.676944820023992 and parameters: {'boosting': 'gbdt'
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[726]   train's l1: 11.3045 valid's l1: 9.74807
[I 2025-10-02 06:21:08,761] Trial 9 finished with value: 9.748069305854 and parameters: {'boosting': 'gbdt', '
[I 2025-10-02 06:23:34,311] Trial 10 finished with value: 12.626679640176087 and parameters: {'boosting': 'dar
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[1905]  train's l1: 11.8971 valid's l1: 10.0311
[I 2025-10-02 06:23:37,166] Trial 11 finished with value: 10.031088763922668 and parameters: {'boosting': 'gbd
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[116]   train's l1: 10.555  valid's l1: 9.60229
[I 2025-10-02 06:23:38,217] Trial 12 finished with value: 9.602287085222173 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[68]    train's l1: 10.7389 valid's l1: 9.64844
[I 2025-10-02 06:23:39,144] Trial 13 finished with value: 9.648443277799114 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[67]    train's l1: 10.5117 valid's l1: 9.87467
[I 2025-10-02 06:23:40,015] Trial 14 finished with value: 9.874668800104066 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[46]    train's l1: 9.4943  valid's l1: 9.91388
[I 2025-10-02 06:23:41,117] Trial 15 finished with value: 9.913875980052323 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[38]    train's l1: 11.2902 valid's l1: 10.0093
[I 2025-10-02 06:23:41,803] Trial 16 finished with value: 10.009276733191905 and parameters: {'boosting': 'gbd
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[352]   train's l1: 10.666  valid's l1: 9.79654
[I 2025-10-02 06:23:44,796] Trial 17 finished with value: 9.796540375332468 and parameters: {'boosting': 'gbdt
[I 2025-10-02 06:25:56,709] Trial 18 finished with value: 11.548089034256531 and parameters: {'boosting': 'dar
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[79]    train's l1: 10.7423 valid's l1: 9.84674
[I 2025-10-02 06:25:58,261] Trial 19 finished with value: 9.84674226410931 and parameters: {'boosting': 'gbdt'
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[429]   train's l1: 11.502  valid's l1: 9.94293
[I 2025-10-02 06:25:59,179] Trial 20 finished with value: 9.942929126238585 and parameters: {'boosting': 'gbdt
```

```
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[722]   train's l1: 10.7409 valid's l1: 9.67228
[I 2025-10-02 06:26:02,131] Trial 21 finished with value: 9.672284347596785 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[40]    train's l1: 10.8382 valid's l1: 9.70599
[I 2025-10-02 06:26:02,969] Trial 22 finished with value: 9.705986342619905 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[77]    train's l1: 10.5251 valid's l1: 9.7444
[I 2025-10-02 06:26:03,968] Trial 23 finished with value: 9.744400029575733 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[423]   train's l1: 11.1111 valid's l1: 9.67648
[I 2025-10-02 06:26:06,179] Trial 24 finished with value: 9.676483069042531 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[132]   train's l1: 10.6611 valid's l1: 9.79426
[I 2025-10-02 06:26:07,120] Trial 25 finished with value: 9.79425870388979 and parameters: {'boosting': 'gbdt'
[I 2025-10-02 06:27:59,959] Trial 26 finished with value: 10.582407520468031 and parameters: {'boosting': 'dar
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[144]   train's l1: 11.4157 valid's l1: 9.75957
[I 2025-10-02 06:28:01,171] Trial 27 finished with value: 9.759567454874771 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[34]    train's l1: 11.201  valid's l1: 9.72999
[I 2025-10-02 06:28:01,940] Trial 28 finished with value: 9.729990599959297 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[190]   train's l1: 10.8674 valid's l1: 9.76871
[I 2025-10-02 06:28:02,976] Trial 29 finished with value: 9.76870976373425 and parameters: {'boosting': 'gbdt'
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[216]   train's l1: 10.6299 valid's l1: 9.76527
[I 2025-10-02 06:28:04,591] Trial 30 finished with value: 9.765267181473936 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[488]   train's l1: 11.0652 valid's l1: 9.68386
[I 2025-10-02 06:28:07,113] Trial 31 finished with value: 9.683860658412517 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[289]   train's l1: 10.9798 valid's l1: 9.64414
[I 2025-10-02 06:28:10,191] Trial 32 finished with value: 9.644135031421774 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[522]   train's l1: 10.5264 valid's l1: 9.64967
[I 2025-10-02 06:28:13,195] Trial 33 finished with value: 9.649674622407504 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[297]   train's l1: 10.6383 valid's l1: 9.62165
[I 2025-10-02 06:28:15,056] Trial 34 finished with value: 9.621646872803655 and parameters: {'boosting': 'gbdt
[I 2025-10-02 06:31:30,623] Trial 35 finished with value: 9.943340513182084 and parameters: {'boosting': 'dart
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[578]   train's l1: 11.1482 valid's l1: 9.73273
[I 2025-10-02 06:31:33,169] Trial 36 finished with value: 9.732725493852222 and parameters: {'boosting': 'gbdt
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[480]   train's l1: 11.4178 valid's l1: 9.84213
[I 2025-10-02 06:31:34,367] Trial 37 finished with value: 9.84212711755553 and parameters: {'boosting': 'gbdt'
[I 2025-10-02 06:32:08,483] Trial 38 finished with value: 9.936119591280413 and parameters: {'boosting': 'dart
Training until validation scores don't improve for 200 rounds
Early stopping, best iteration is:
[90]    train's l1: 11.257  valid's l1: 9.83464
[I 2025-10-02 06:32:09,143] Trial 39 finished with value: 9.834641377469502 and parameters: {'boosting': 'gbdt
{'boosting': 'gbdt',
 'num_leaves': 191,
 'learning_rate': 0.06551980915891428,
 'feature_fraction': 0.8259221301813049,
 'bagging_fraction': 0.8994093454795021,
 'bagging_freq': 3,
```

```
dtrain = lgb.Dataset(X_train_imp, label=y_train)
dvalid = lgb.Dataset(X_valid_imp, label=y_valid, reference=dtrain)
cbs = [lgb.early_stopping(stopping_rounds=300), lgb.log_evaluation(period=200)]

try:
    lgb_model = lgb.train(
        params=best_params,
        train_set=dtrain,
        valid_sets=[dtrain, dvalid],
        valid_names=['train', 'valid'],
        num_boost_round=5000,
        callbacks=cbs
    )
    pred_valid = lgb_model.predict(X_valid_imp, num_iteration=lgb_model.best_iteration)

except Exception:
    sk_blocklist = {
        'objective', 'metric', 'verbosity', 'num_threads',
        'num_boost_round', 'num_iterations', 'n_estimators'
    }
    sk_params = {k: v for k, v in best_params.items() if k not in sk_blocklist}

    lgb_model = lgb.LGBMRegressor(
        objective='regression',
        n_estimators=5000,
        n_jobs=best_params.get('num_threads', -1),
        **sk_params
    )
    lgb_model.fit(
        X_train_imp, y_train,
        eval_set=[(X_train_imp, y_train), (X_valid_imp, y_valid)],
        callbacks=cbs
    )
    pred_valid = lgb_model.predict(X_valid_imp,num_iteration=getattr(lgb_model, 'best_iteration_', None))

_ = evaluate(y_valid, pred_valid, "LightGBM (valid)")
```

```
Training until validation scores don't improve for 300 rounds
[200]   train's l1: 10.2141     valid's l1: 9.81861
[400]   train's l1: 9.5796      valid's l1: 10.0744
Early stopping, best iteration is:
[116]   train's l1: 10.555      valid's l1: 9.60229
LightGBM (valid)             -> MAE: 9.602 | RMSE: 19.878
```

```
missing_in_test = [c for c in feature_cols if c not in df_test.columns]
for c in missing_in_test:
    df_test[c] = 0

extra_cats_in_test = [c for c in df_test.columns if c.startswith('Cat_') and c not in feature_cols]
if extra_cats_in_test:
    df_test = df_test.drop(columns=extra_cats_in_test)

for c in feature_cols:
    if c in df_test.columns and df_test[c].dtype == 'bool':
        df_test[c] = df_test[c].astype(int)

X_all  = df_train_labeled[feature_cols].copy()
y_all  = df_train_labeled['DaysToNext'].astype(float).copy()
X_test = df_test[feature_cols].copy()

imputer_final = SimpleImputer(strategy='median')
X_all_imp  = pd.DataFrame(imputer_final.fit_transform(X_all),  columns=feature_cols, index=X_all.index)
X_test_imp = pd.DataFrame(imputer_final.transform(X_test),    columns=feature_cols, index=X_test.index)
best_iter = (
    getattr(lgb_model, 'best_iteration',  None) or
```

```
        getattr(lgb_model, 'best_iteration_', None) or
        best_params.get('num_boost_round') or
        best_params.get('num_iterations') or
        best_params.get('n_estimators') or
        2000
    )
    best_iter = int(best_iter)
    try:
        native_blocklist = {'verbosity', 'num_threads'}
        native_params = {k: v for k, v in best_params.items() if k not in native_blocklist}

        lgb_model_full = lgb.train(
            params=native_params,
            train_set=lgb.Dataset(X_all_imp, label=y_all),
            num_boost_round=best_iter,
            callbacks=[lgb.log_evaluation(period=0)]
        )
        test_pred_days = lgb_model_full.predict(X_test_imp, num_iteration=best_iter)

    except Exception:
        sk_blocklist = {
            'objective','metric','verbosity','num_threads',
            'num_boost_round','num_iterations','n_estimators'
        }
        sk_params = {k: v for k, v in best_params.items() if k not in sk_blocklist}

        lgb_model_full = lgb.LGBMRegressor(
            objective='regression',
            n_estimators=best_iter,
            n_jobs=best_params.get('num_threads', -1),
            **sk_params
        )
        lgb_model_full.fit(X_all_imp, y_all)
        test_pred_days = lgb_model_full.predict(X_test_imp)
    df_submit = df_test[['CustomerID','TransactionDate']].copy()
    df_submit['PredictedDaysToNext'] = np.maximum(test_pred_days, 0)
    df_submit['PredictedDaysToNextRounded'] = np.rint(df_submit['PredictedDaysToNext']).astype(int)

    SUBMIT_PATH = "/content/predictions_lightgbm.csv"
    df_submit.to_csv(SUBMIT_PATH, index=False)
    print(df_submit.head())
    print(f"Saved: {SUBMIT_PATH}")
```

```
   CustomerID TransactionDate  PredictedDaysToNext  PredictedDaysToNextRounded
0         636      2022-11-27             0.000000                           0
1         636      2022-11-27            29.055832                          29
2         637      2022-03-27             0.000000                           0
3         637      2022-03-27            28.077458                          28
4         642      2022-05-29             6.206511                           6
Saved: /content/predictions_lightgbm.csv
```

```
print(df_train.head(3))
```

```
   CustomerID TransactionDate     ProductCategory  PurchaseAmount  TxnIdx  \
0         636      2021-07-25  Non Finished Goods          469.08       0
1         636      2021-07-25       Personal Care         8413.86       1
2         636      2021-11-28       Personal Care         8801.68       2

   TxnCount   PrevDate   NextDate  DaysSincePrev  DaysToNext  ...  \
0         4        NaT 2021-07-25            NaN         0.0  ...
1         4 2021-07-25 2021-11-28            0.0       126.0  ...
2         4 2021-07-25 2021-11-28          126.0         0.0  ...

   Cat_Shrink Film  Cat_Specialty Foods - Other  Cat_Unknown  TXN_Year  \
0            False                         False        False      2021
1            False                         False        False      2021
2            False                         False        False      2021
```