mlops-zoomcamp > Pipelines > brilliant_artifact > **Edit** v0.9.76 12:09 UTC Live help Try PRO

ALL FILES CURRENT BLOCK

File Edit Run View Python

Search files

- mlops-zoomcamp
  - 01-intro
  - 02-experiment-tracki
  - 03-training-pipeline
    - Dockerfile
    - README.md
    - docker-compose.yam
    - duration-predictio
    - mage-version.png
  - artifacts
  - custom
  - custom_templates
  - data
  - pipelines
    - brilliant_artifact
    - nyc_taxi_homework
      - __init__.py
      - metadata.yaml
  - training-pipelines-h
  - ~
  - README.md
  - metadata.yaml
  - mlflow.db

PY CUSTOM read_dataframe

```
import pandas as pd

def main(year: int, month: int):
    def load_df(y, m):
        url = f'https://d37ci6vzurychx.cloudfront.r
        df = pd.read_parquet(url)
        df['duration'] = df.lpep_dropoff_datetime -
        df['duration'] = df.duration.dt.total_secor
        df = df[(df.duration >= 1) & (df.duration <
        df[['PULocationID', 'DOLocationID']] = df[[
        df['PU_DO'] = df['PULocationID'] + '_' + df
        return df

    df_train = load_df(year, month)
    next_year = year if month < 12 else year + 1
    next_month = month + 1 if month < 12 else 1
    df_val = load_df(next_year, next_month)

    return df_train, df_val
```

PY CUSTOM create_features

Positional arguments for decorated function:

```
@custom
def transform_custom(data):
    data → read_dataframe
```

```
from sklearn.feature_extraction import DictVectoriz

def main(df_train, df_val):
    def make_X(df, dv=None):
        dicts = df[["PU_DO", "trip_distance"]].to_d
        if dv is None:
            dv = DictVectorizer()
            X = dv.fit_transform(dicts)
        else:
            X = dv.transform(dicts)
        return X, dv

    X_train, dv = make_X(df_train)
    X_val, _ = make_X(df_val, dv)
```

```
    y_train = df_train["duration"].values
    y_val = df_val["duration"].values

    return X_train, X_val, y_train, y_val, dv
```

PY CUSTOM train_model

Positional arguments for decorated function:

```
@custom
def transform_custom(data):
    data → create_features
```

```
import mlflow
import xgboost as xgb
import pickle
from pathlib import Path
from sklearn.metrics import root_mean_squared_error

def main(X_train, X_val, y_train, y_val, dv):
    mlflow.set_tracking_uri("http://localhost:5000"
    mlflow.set_experiment("nyc-taxi-experiment")

    with mlflow.start_run() as run:
        dtrain = xgb.DMatrix(X_train, label=y_train
        dval = xgb.DMatrix(X_val, label=y_val)

        best_params = {
            'learning_rate': 0.09585355369315604,
            'max_depth': 30,
            'min_child_weight': 1.060597050922164,
            'objective': 'reg:linear',
            'reg_alpha': 0.018060244040060163,
            'reg_lambda': 0.011658731377413597,
            'seed': 42
        }

        mlflow.log_params(best_params)

        booster = xgb.train(
            params=best_params,
            dtrain=dtrain,
            num_boost_round=30,
            evals=[(dval, 'validation')],
            early_stopping_rounds=50
        )

        y_pred = booster.predict(dval)
        rmse = root_mean_squared_error(y_val, y_pre
        mlflow.log_metric("rmse", rmse)
```

```
        Path("models").mkdir(exist_ok=True)
        with open("models/preprocessor.b", "wb") as
            pickle.dump(dv, f)
        mlflow.log_artifact("models/preprocessor.b"

        mlflow.xgboost.log_model(booster, artifact_

        return run.info.run_id
```

PY CUSTOM write_run_id

Positional arguments for decorated function:

```
@custom
def transform_custom(data):
    data → train_model
```

```
def main(run_id: str):
    with open("run_id.txt", "w") as f:
        f.write(run_id)

```

All blocks Custom Markdown Search for ⌘ + /

CPU: 0.2% / Memory: 58.5MB Saved a few seconds ago