Xây Dựng Hàm Phụ Trợ PipeLine

Trong bài này sử dụng tập dữ liệu Penguins để đào tạo một mô hình phân loại các loài chim cánh cụt.

I. DatasetMixin Class

tại thư mục gốc tạo pipelines/common.py

```
import logging
import logging.config
import sys
import time
from io import StringIO
from pathlib import Path

import pandas as pd
from metaflow import IncludeFile, current

PYTHON = "3.12.8"

PACKAGES = {
    "keras": "3.8.0",
    "scikit-learn": "1.6.1",
    "mlflow": "2.20.2",
    "tensorflow": "2.18.0",
}



class DatasetMixin:
    """A mixin for loading and preparing a dataset.

    This mixin is designed to be combined with any pipeline that requires accessing
    a dataset.
```

```
    """

    dataset = IncludeFile(
        "dataset",
        is_text=True,
        help="Dataset that will be used to train the model.",
        default="data/penguins.csv",
    )

    def load_dataset(self):
        """Load and prepare the dataset."""
        import numpy as np

        # The raw data is passed as a string, so we need to convert it into a DataFrame.
        data = pd.read_csv(StringIO(self.dataset))

        # Replace extraneous values in the sex column with NaN. We can handle missing
        # values later in the pipeline.
        data["sex"] = data["sex"].replace(".", np.nan)

        # We want to shuffle the dataset. For reproducibility, we can fix the seed value
        # when running in development mode. When running in production mode, we can use
        # the current time as the seed to ensure a different shuffle each time the
        # pipeline is executed.
        seed = int(time.time() * 1000) if current.is_production else 42
        generator = np.random.default_rng(seed=seed)
        data = data.sample(frac=1, random_state=generator)

        logging.info("Loaded dataset with %d samples", len(data))

        return data


def packages(*names: str):
    """Return a dictionary of the specified packages and their corresponding version.

    This function is useful to set up the different pipelines while keeping the
    package versions consistent and centralized in a single location.

    Any packages that should be locked to a specific version will be part of the
```

```
    `PACKAGES` dictionary. If a package is not present in the dictionary, it will be
    installed using the latest version available.
    """
    return {name: PACKAGES.get(name, "") for name in names}


def configure_logging():
    """Configure logging handlers and return a logger instance."""
    if Path("logging.conf").exists():
        logging.config.fileConfig("logging.conf")
    else:
        logging.basicConfig(
            format="%(asctime)s [%(levelname)s] %(message)s",
            handlers=[logging.StreamHandler(sys.stdout)],
            level=logging.INFO,
        )


def build_features_transformer():
    """Build a Scikit-Learn transformer to preprocess the feature columns."""
    from sklearn.compose import ColumnTransformer, make_column_selector
    from sklearn.impute import SimpleImputer
    from sklearn.pipeline import make_pipeline
    from sklearn.preprocessing import OneHotEncoder, StandardScaler

    numeric_transformer = make_pipeline(
        SimpleImputer(strategy="mean"),
        StandardScaler(),
    )

    categorical_transformer = make_pipeline(
        SimpleImputer(strategy="most_frequent"),
        # We can use the `handle_unknown="ignore"` parameter to ignore unseen categories
        # during inference. When encoding an unknown category, the transformer will
        # return an all-zero vector.
        OneHotEncoder(handle_unknown="ignore"),
    )

    return ColumnTransformer(
        transformers=[
```

```
            (
                "numeric",
                numeric_transformer,
                # We'll apply the numeric transformer to all columns that are not
                # categorical (object).
                make_column_selector(dtype_exclude="object"),
            ),
            (
                "categorical",
                categorical_transformer,
                # We want to make sure we ignore the target column which is also a
                # categorical column. To accomplish this, we can specify the column
                # names we only want to encode.
                ["island", "sex"],
            ),
        ],
    )


def build_target_transformer():
    """Build a Scikit-Learn transformer to preprocess the target column."""
    from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OrdinalEncoder

    return ColumnTransformer(
        transformers=[("species", OrdinalEncoder(), ["species"])],
    )


def build_model(input_shape, learning_rate=0.01):
    """Build and compile the neural network to predict the species of a penguin."""
    from keras import Input, layers, models, optimizers

    model = models.Sequential(
        [
            Input(shape=(input_shape,)),
            layers.Dense(10, activation="relu"),
            layers.Dense(8, activation="relu"),
            layers.Dense(3, activation="softmax"),
        ],
```

```
    )

    model.compile(
        optimizer=optimizers.SGD(learning_rate=learning_rate),
        loss="sparse_categorical_crossentropy",
        metrics=["accuracy"],
    )

    return model
```

Class `DatasetMixin`

Là một **mixin** — class phụ dùng để thêm khả năng `load_dataset()` vào các pipeline Metaflow.

```
dataset = IncludeFile(
    "dataset",
    is_text=True,
    help="Dataset that will be used to train the model.",
    default="data/penguins.csv",
)
```

- Cho phép bạn **đưa một file vào flow** như biến `self.dataset`.
- Dữ liệu được load dưới dạng **text**, bạn sẽ cần dùng `StringIO` để convert sang CSV.

load_dataset()

```
def load_dataset(self):
    ...
```

- Đọc file CSV từ `self.dataset`.
- Chuẩn hóa giá trị `.`, NaN trong cột `"sex"`.
- Shuffle dữ liệu bằng `numpy`:
  - Dùng `seed=42` nếu đang ở development.
  - Dùng `seed = time` nếu ở production (`current.is_production`).

Hàm `packages(*names)`

```
def packages(*names: str):
    return {name: PACKAGES.get(name, "") for name in names}
```

Khi bạn viết Metaflow `@conda_base` hoặc `@conda` decorator, bạn có thể truyền gọn:

```
@conda(packages=packages("keras", "scikit-learn"))
```

Hàm `configure_logging()`

```
def configure_logging():
    ...
```

- Nếu có file `logging.conf` thì dùng cấu hình từ file.
- Nếu không có thì thiết lập mặc định:
  - Format log cơ bản
  - In log ra terminal (stdout)
  - Mức `INFO`

build_features_transformer()

```
def build_features_transformer():
    ...
```

Trả về một **`ColumnTransformer`** để xử lý:

- Các cột số (`int`, `float`):
  - Impute bằng trung bình (`SimpleImputer(strategy="mean")`)
  - Chuẩn hóa (`StandardScaler()`)
- Các cột phân loại (`object`):
  - Impute bằng mode (`most_frequent`)
  - One-hot encode (`OneHotEncoder(handle_unknown="ignore")`)

```
["island", "sex"]  # là các cột categorical cụ thể
```

`handle_unknown="ignore"` giúp model không crash khi gặp category mới trong inference.

build_target_transformer()

```
def build_target_transformer():
    ...
```

- Dùng `OrdinalEncoder` để mã hóa target `"species"` thành số nguyên (0, 1, 2).
- Gói trong `ColumnTransformer` để có API nhất quán với `fit_transform`, `inverse_transform`.

build_model(input_shape, learning_rate)

```
def build_model(input_shape, learning_rate=0.01):
    ...
```

- Xây dựng model với Keras Sequential API:
  - Input: `input_shape` chiều (sau preprocessing)

    - 2 hidden layers: 10 và 8 neurons
    - Output: 3 class (penguin species) → softmax
- Compile model:
    - Optimizer: `SGD` với learning rate
    - Loss: `sparse_categorical_crossentropy`
    - Metrics: `accuracy`

Tổng Kết

Phần	Mục đích
`DatasetMixin`	Load CSV dataset dùng `IncludeFile`, shuffle, xử lý giá trị thiếu
`packages()`	Tập trung quản lý version package
`configure_logging()`	Ghi log ra terminal hoặc file
`build_features_transformer()`	Chuẩn hóa, encode dữ liệu đầu vào
`build_target_transformer()`	Encode target sang số
`build_model()`	Khởi tạo mô hình MLP với Keras

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #1
Được tạo 19 tháng 4 2025 15:58:27 bởi Đỗ Ngọc Tú
Được cập nhật 22 tháng 4 2025 17:42:41 bởi Đỗ Ngọc Tú