**End to End Project Workflow**

**设置环境和GITHUB**

图形用户界面, 文本

描述已自动生成

1. **SETUP GITHUB**

配置虚拟环境:

conda create -p venv python==3.8 -y

conda activate venv/

Clone和Sync Github:

Initializing empty git repository:

git init

创建 readme file, 写Description

先add再提交commit上去Github:

add README.md

git commit -m “First commit”

检查branch:

git branch -M main

确保 in sync with repository:

git remote add origin https://github.com/Nick14848/mlproject.git

先检查 global email and ID:

git config global-- user.name “Nick14848”

git config global—user.email “nicktsai1221@gmail.com

最后 Push into repository:

git push -u origin main

拉回来:

git pull

1. **SOURCE FOLDER**

创建 .gitignore, 为了用于指定 Git 仓库中应该忽略的文件或目录。这些被忽略的文件不会被提交到版本控制中

创建 setup.py 和 requirement.txt

Setup.py: 把我们的machine learning application 变成 package

创建 src folder:

\_\_init\_\_.py: 整个 project implement 都在这里, 可以作为一个 package

自动加载 packages:

pip install -r requirements.txt

最后每次更新的流程:

git add .

git status

git commit -m “setup”

git push -u origin main

**PROJECT STRUCTURE**

**项目框架**

**整个Project application 都会在src发生**

在src里创建components folder, components会作为package, 可以被imported exported to other file location, 包括:

data \_ingestion.py: 即从database获取数据的module

data\_transformation.py: Divide into train and test datasets

model\_trainer.py: train model

在src里创建pipeline folder, components会作为package, 可以被imported exported to other file location, 包括:

Train\_pipeline.py

Predict\_pipeline.py

在src里创建:

logger.py

exception.py

utils.py: 用于存放实用函数的模块

当我们遇到exception:

把exception log into logger file

**PROJECT IMPLEMENTATION**

**项目具体实施**

什么样的 datasets ?

EDA 用 Jupyter notebook 更好

modular programming:

在 data\_digestion/data\_transformation/model\_trainer.py 里面调用 utils.py 里的function

**DATA\_INJECTION.py**

class DataIngestionConfig: 接受 any input