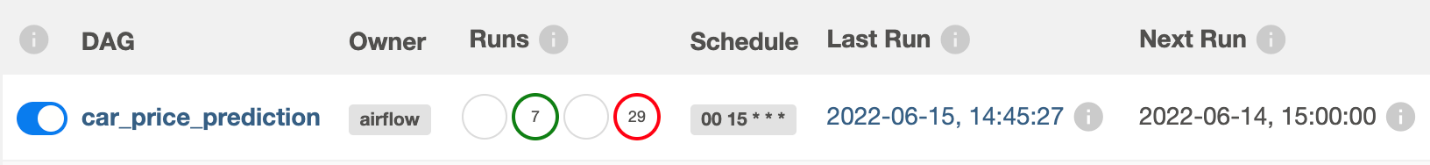
**33. 5 Практическая работа**

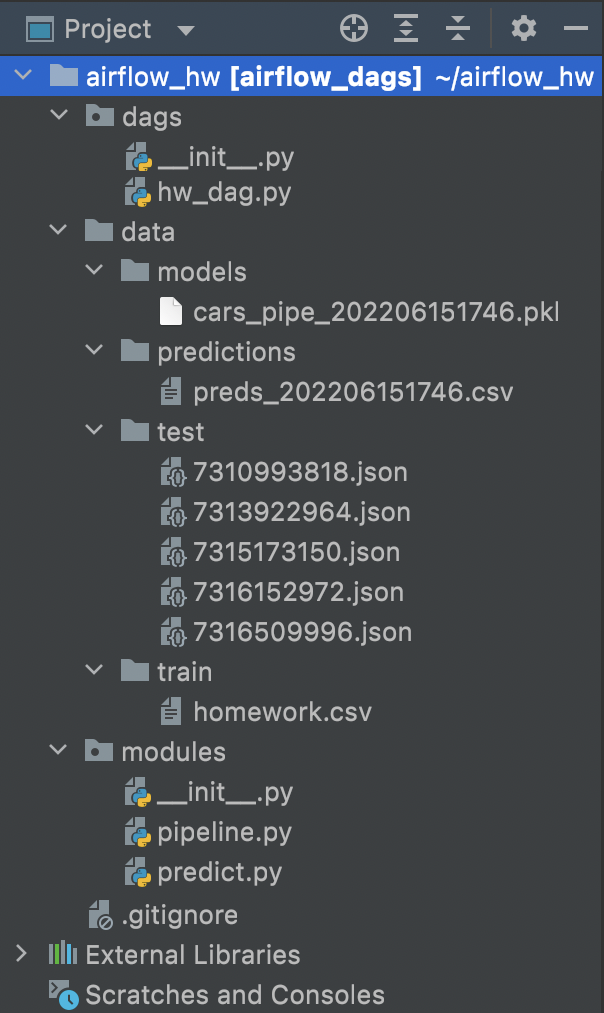
Цель задания

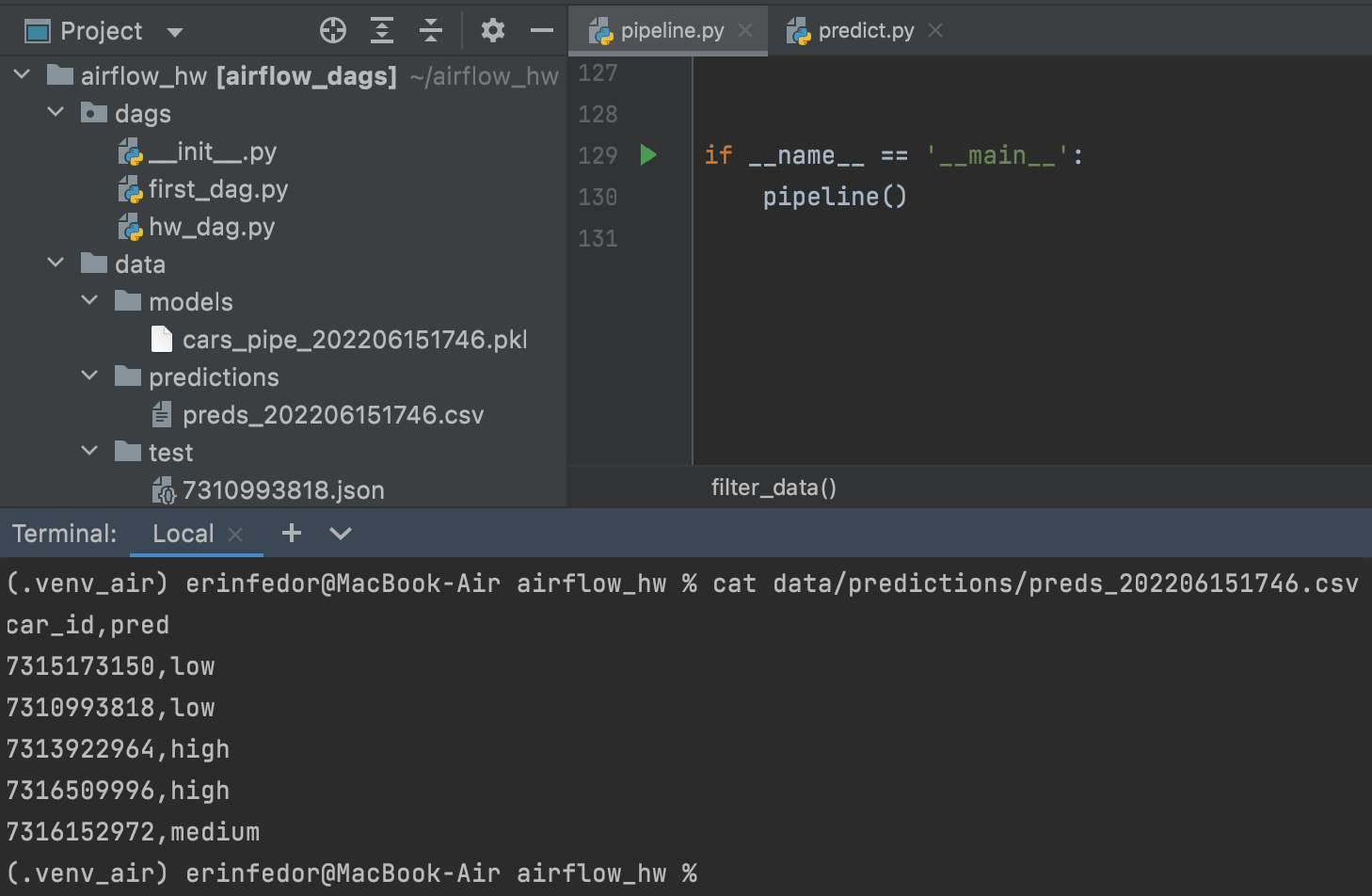
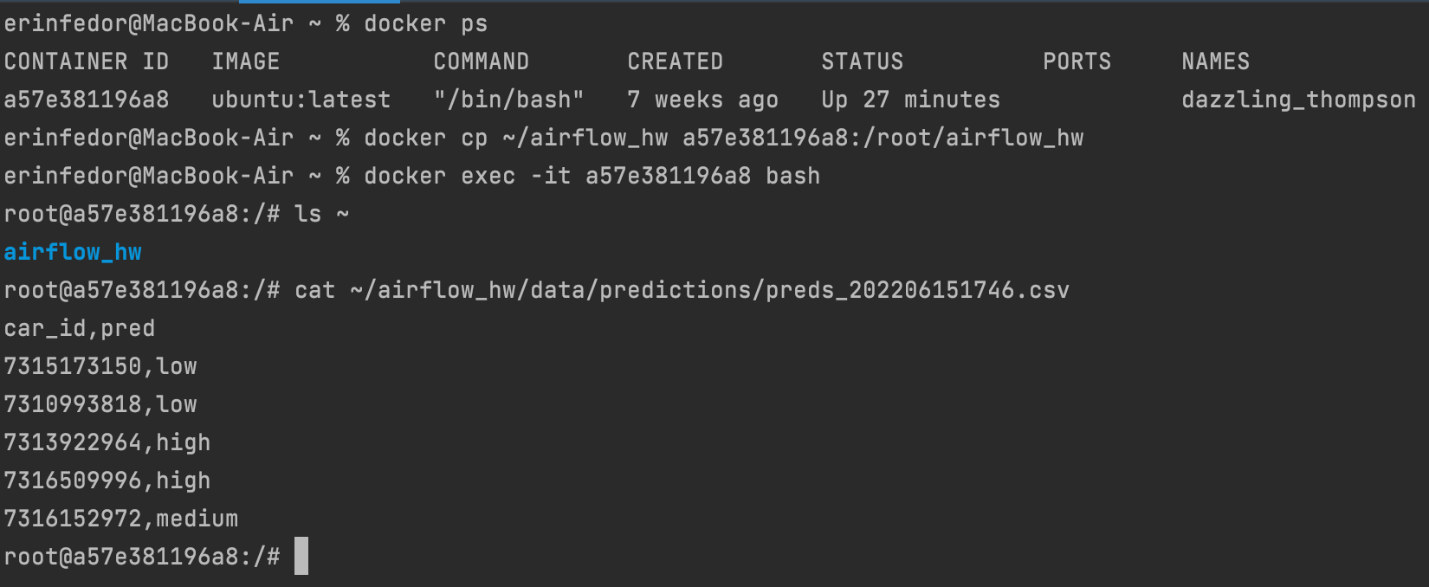
Разработать пайплайн обучения ML-модели.

Что нужно сделать

1. Скачать архив с проектом airflow\_hw, внутри него:
   * шаблон DAG’а (*dags/hw\_dag.py*),
   * готовый код ML-модели (*modules/pipeline.py*),
   * шаблон скрипта для прогноза моделью (*modules/predict.py*),
   * данные для обучения и тестирования (*data/train, data/test*),
   * пустые папки под сохранение ML-модели и предсказаний.
2. Положить папку *airflow\_hw* в домашнюю директорию (*~/*) и открыть её **в Pycharm**.
3. Запустить пайплайн с моделью локально и в Airflow, это обучит и сохранит объект с пайплайном лучшей модели в **pickle** формате:
   * локально: **python3***modules/pipeline.py* (из терминала Pycharm).
   * в Airflow: скопировать файл *hw\_dag.py* в папку *$AIRFLOW\_HOME/dags*.  
       
     После этого в интерфейсе отобразится новый DAG:  
     
4. Написать код в файле *modules/predict.py*, который при вызове функции *predict()*:
   * загружает обученную модель,
   * делает предсказания для всех объектов в папке *data/test*,
   * объединяет предсказания в один Dataframe и сохраняет их в **csv**-формате в папку *data/predictions*.  
       
     Не забываем разбивать код на смысловые части в виде отдельных функций.
5. Проверить корректность кода, запустив его локально: **python3***modules/predict.py*(из терминала Pycharm)
6. Встроить прогноз моделью в пайплайн, в котором будет 2 шага:
   * *pipeline* — здесь выполняется функция pipeline,
   * *predict* — здесь делается предикт для всех объектов и сохраняется в папку *data/predictions*.
7. Запустить пайплайн в интерфейсе Airflow и получить предикты модели.

Советы и рекомендации  
Целевая структура проекта выглядит так:



1. Работа с файлами проекта будет зависеть от способа развертывания Airflow:
   * **локально:**Airflow живёт в домашней директории (~/airflow) вашей операционной системы, папка с проектом располагается рядом (~/airflow\_hw). При запуске DAG результаты будут записываться сразу в папку проекта, дополнительных действий не потребуется; содержимое файла предсказания можно посмотреть так:  
     
   * **в Docker’е:** код DAG запускается в контейнере worker, а значит, код проекта нужно будет скопировать туда до запуска полного пайплайна;
   * (1) посмотреть список контейнеров,
   * (2) скопировать файлы из локальной папки в контейнер с заданным id,
   * (3) войти в терминал контейнера,
   * (4) и посмотреть предикты после отработки DAG  
     можно так:  
     
2. В пайплайне с моделью из предыдущего модуля нет работы с API, пайплайн сохраняет только саму модель и тайм-стамп времени запуска. В текущей задаче вам нужно показать работу с моделью как с некоторым периодически запускаемым в Airflow пайплайном обработки данных.
3. Обратите внимание на работу с переменными окружения (sys.path, os.environ) и областью видимости модулей и функций в коде проекта (в коде есть пояснительные комментарии). Как мы помним из теоретического занятия, в общем случае файл с DAG может запускаться на изолированной машине (удалённый сервер / docker-контейнер), поэтому импорт всех нужных модулей проекта должен быть возможен.
4. Окружений (или виртуальных сред) в данной задаче два: окружение, в котором ведётся разработка в PyCharm (он нужен только для написания кода и подсветки синтаксиса), и окружение, в котором запускаются DAG’и. Создавать дополнительные окружения не стоит, в них можно легко запутаться.

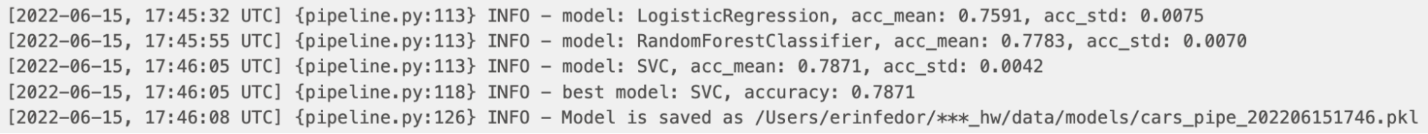
Критерии оценивания

1. Структура проекта и логика файлов modules/predict и dags/hw\_dag соответствуют заданию.
2. DAG с пайплайном модели из двух шагов (pipeline, predict) работает и сохраняет предсказания в файл (приложен скриншот с содержимым файла с предиктами модели).
3. Папка Airflow содержит только файлы самого Airflow. В папку dags скопированы файлы DAG’ов из папки с проектом. (Файлы проекта и папка Airflow существуют раздельно.)

Как отправить работу на проверку

1. Код проекта (папка airflow\_hw) выложить на Github в личный репозиторий.
2. Для возможности проверки сделать его публичным (public).
3. Ссылку на репозиторий прикрепить через форму ниже, туда же скриншот с содержимым файла с предиктами модели, как результат работы итогового DAG.

P. S. Интересные факты, которые вы могли заметить при работе с Airflow:

1. В файле modules/pipeline.py вы наверняка заметите, что отладочная информация о качестве ML-моделей теперь выводится не через print(), а с помощью модуля[logging.](https://docs.python.org/3/library/logging.html) В реальных задачах справочную информацию принято логировать таким образом. Благодаря этому, в логах задач в Airflow мы сможем читать всю нужную нам информацию (время, файл и так далее):  
   
2. Также вы можете заметить в коде [typing](https://docs.python.org/3/library/typing.html) — это конструкции вида func(value: int) -> str:.  
   Это означает, что функция func принимает аргумент типа int, а возвращает значение типа string. Это помогает понимать тип данных при написании кода. При этом надо помнить, что Python обладает динамической типизацией, и такая аннотация — это лишь подсказка для удобства разработчика.
3. В папке проекта есть файл .gitignore. Он поможет вам не отслеживать файлы с данными при выкладывании кода в ваш репозиторий GitHub: