Домашняя работа №6.

В этом задании вам предстоит предсказать "настроение" (sentiment) обзора товара (review).

В этой работе вы воспользуетесь планировщиком Airflow, чтобы реализовать DAG из нескольких задач:

- предобработка тренировочного датасета на Spark (feature engineering)
- обучение модели sklearn на этих данных
- предобработка тестового датасета на Spark (feature engineering)
- предсказание на тестовом датасете с помощью Spark и pandas_udf, используя предобученную модель sklearn

Описание датасета

Датасет состоит из json-строк следующего вида. Заметьте, поля - теже самые, как и в 4-м задании, за исключение целевой переменной. Теперь колонка "overall", содержащая рейтинг товара от 1 до 5, превратилась в колонку label (см. описание)


```
{

  "label": 1,
  "vote": "25",
  "verified": false,
  "reviewTime": "12 19, 2008",
  "reviewerID": "APV13CM0919JD",
  "asin": "B001GXRQW0",
  "reviewerName": "LEH",
  "reviewText": "Amazon,\nI am shopping for Amazon.com gift cards for Christmas gifts and am really so disappointed that out of five choices there isn't one that says \"Merry Christmas\" or mentions Christmas at all!  I am sure I am not alone in wanting a card that reflects the actual \"holiday\" we are celebrating. On principle, I cannot send a Amazon gift card this Christmas. What's up with all the Political Correctness?  Bad marketing decision.\nLynn",
  "summary": "Merry Christmas.",
  "unixReviewTime": 1229644800
}
```


где:

- label - настроение обзора со значениями 1 (позитивный обзор, соответствует рейтингам 4 и 5), и 0 (негативный обзор, соответствует рейтингам 1-3 включительно).
- vote - скольким людям понравился обзор

- verified - True, если известно, что автор обзора купил этот товар на сайте.
- reviewTime - время написания или опубликования обзора
- reviewerID - идентификатор автора
- asin - идентификатор товара
- reviewerName - имя автора
- reviewText - собственно текст обзора
- summary - короткое резюме по товару, или заголовок обзора
- unixReviewTime - время обзора в секундах эпохи.

Целевой переменной является label, все остальные поля - признаки.

Местонахождение датасета

- /datasets/amazon/all_reviews_5_core_train_extra_small_sentiment.json - очень маленький тренировочный датасет на 100000 записей (HDFS).
- /datasets/amazon/all_reviews_5_core_test_extra_small_features.json - маленький тестовый датасет на 1 миллион записей.

В маленьком тренировочном датасете присутствуют не все значения asin, которые есть в тестовом датасете.

Оформление

В вашем приватном репо ozon-masters-bigdata создайте подпапку projects/6. В ней должен быть проект Airflow, то есть файлы:

- <your_nick>_dag.py - файл с графом Airflow, в котором должны быть следующие задачи (task_id): feature_eng_train_task, download_train_task, train_task, model_sensor, feature_eng_test_task, predict_task.
- файл с задачей спарк для предобработки датасета. Название не важно. Он должен принимать две опции: --path-in и --path-out.
- файл с кодом обучения модели sklearn. Название не важно. Опции: --train-in --sklearn-model-out
- файл с задачей спарк для инференса. Название не важно. Опции: --test-in --pred-out --sklearn-model-in

Требования к DAG

DAG должен называться **вашник_dag**.

DAG должен быть без расписания (schedule_interval=None) и без догонки (catchup=False).

Для упрощения, все задачи в DAG должны выполняться последовательно.

Все пути, которые передаются в скрипты по опциям должны содержать шаблонную переменную '{{ dag_run.conf["base_dir"] }}'. Это делается для того, чтобы чекер мог управлять путями к скриптам и путями, куда сохраняются файлы.

Пример использования (фрагмент DAG):

```
base_dir = '{{ dag_run.conf["base_dir"] if dag_run else "" }}'
etl_task = SparkSubmitOperator(
          application=f"{base_dir}etl.py",
...
```

Тогда вы или чекер можете запустить ваш ран с помощью следующей команды:

airflow dags trigger -c '{"base_dir": "/home/users/nick/airflow/"}' nick_dag

Если эта переменная base_dir не дана с запуском, то base_dir в скриптах будет "" и файлы будут сохраняться в текущую директорию (нужно выяснять экспериментально в какую именно, но чаще всего в AIRFLOW_HOME, домашнюю или /).

Требования к задачам

- feature_eng_task - должна сохранять тренировочный преобразованный файл в HDFS по *относительному* пути nick_train_out, а тестовый - по пути nick_test_out
- train_download_task должна сохранять файл nick_train_out_local в base_dir
- train_task должна брать тренировочный файл nick_train_out_local, сохранять модель в файл 6.joblib тоже в base_dir
- model_sensor - должен проверять наличие файла модели.
- predict_task - должна сохранять файл с предсказаниями по *относительному* пути nick_hw6_prediction. Файл в формате CSV с колонками id,prediction без заголовка

Советы по задачам

Используйте SparkSubmitOperator задач спарка. В нем надо задать /usr/bin/spark-submit и передать PYSPARK_PYTHON со значением пути к питону в dsenv. Посмотрите соответствующий синтаксис вызова SparkSubmitOperator.

Используйте BashOperator для запуска тренировки модели sklearn. Это позволяет запустить тот питон, который вам надо (dsenv). Для сенсора файла модели установите приемлемый таймаут.

Советы по работе

Для вас на сервере создана специальная среда afenv. Активируйте командой conda activate afenv. Начните с команды airflow version - она создаст базу данных sqlite и конфиг-файл. В конфиг файле поменяйте порт веб-сервера на 6000 + результат `id -u` - это обеспечит бесконфликтность по портам на сервере.

Для корректной работы SparkSubmitOperator зайдите в UI (через проброс порта) и в меню Connections найдите spark-default и поменяйте в нем очередь root.default на default.

Во время отладки запускайте индивидуальные задачи в тестовом режиме (airflow tasks test). Иногда не все логи появляются в выводе тестового режима. Например, при запуске задачи-оператора спарка пишется что команда вышла с кодом 1, при этом логов нет. Можно запустить означенную команду вручную и посмотреть, что не так.

Чекер

Для просмотра логов и запуска DAG установлен централизованный инстанс Aiflow по адресу https://bigdatamasters.ml:9000.

Перед тем как запустить чекер зайдите туда и авторизуйтесь с вашим паролем от Github используя OAuth. В этот момент автоматически создается пользователь без прав, так что вы ничего там не увидите.

Запустите чекер: checker.sh 6

Внимание! Планировщик установлен на последовательное исполнение всех задач - это сделано для снижения нагрузки на сервер. Ваш запуск может стоять в очереди, хотя через UI вы ее не увидите.

Работа чекера

- выкачивает ваше репо, проверяет наличие dag и других нужных файлов
- проверяет ваш DAG: python nick_dag.py
- копирует nick_dag.py в папку dags централизованного инстанса.
- планировщик подхватит ваш файл.
- запускает DAG run командой: airflow dags trigger -c '{"base_dir": "/home/users/nick/airflow"}' -r checker-N nick_dag инкрементируя N в соответствии с попытками. На этом месте первая попытка может не пройти, если планировщик еще не увидел ваш файл. В таком случае, перезапустите чекер через несколько минут.
- снимает ваш DAG с паузы.
- назначает вам права на ваш DAG. В этот момент ваш DAG становится вам видимым в UI на https://bigdatamasters.ml:9000

- ждет завершения работы DAG.
- в случае успешного завершения скачивает файл с предсказаниями и запускает скорер.
- выводит PASSED 1 при успешном скоринге.