mtool 使用指引

导入mtool模块

```
# -*- coding: utf-8 -*-
import mtool
```

获取hive对象

```
# -*- coding: utf-8 -*-
import mtool

hive = mtool.get_hive()                       #使用mr引擎
hiveonyarn = mtool.get_hive(engine='mr')      #使用mr引擎
hiveonspark = mtool.get_hive(engine='spark') #使用spark引擎（对数据倾斜非常敏感，投入生产环境前请反复测试）
```

批量执行hive sql

```
# -*- coding: utf-8 -*-
import mtool

hive = mtool.get_hive()

#使用列表型参数
v_sql="""
drop table if exists tmp.test_{0};
create table tmp.test_{1} as
    select * from tmp.test;
"""
hive.executeUpdate(v_sql, '20181107', '20181108')

#使用字典型型参数
v_sql="""
drop table if exists tmp.test_{dt1};
create table tmp.test_{dt2} as
    select * from tmp.test;
"""
hive.executeUpdate(v_sql, dt1='20181107', dt2='20181108')
```

执行hive查询

```
# -*- coding: utf-8 -*-
import mtool

hive = mtool.get_hive()

v_sql="""
select * from tmp.test_{0}
"""
#返回list(tuple)型数据
rows = hive.executeQueryLimit10K(v_sql, '20181108')      #批量取数请使用后文的方法，此处最多返回10000条数据
print(rows)

#返回list(dict)型数据
rows = hive.executeQueryLimit10K(v_sql, dictify=True, '20181108')      #批量取数请使用后文的方法，此处最多返回10000条数据
print(rows)
```

将hive查询sql的数据下载至本地目录

```
# -*- coding: utf-8 -*-
import mtool

mtool.download_data("select * from tmp.test", "/tmp/test")
```

注：“mtool.download_data”仅能导出实体表的数据，如dsst、dmdt库的视图，请先执行“hive.executeUpdate”将所需数据转移到专用分析库的临时表中再导出

加载本地目录中的数据至hive表

```
# -*- coding: utf-8 -*-
import mtool

#覆盖表的已有数据（默认）
mtool.upload_data_to_table("/tmp/test", "tmp.test_20181108")
#追加数据文件到表
mtool.upload_data_to_table("/tmp/test", "tmp.test_20181108", overwrite=False)
```

加载本地目录中的数据至hive表分区

```
# -*- coding: utf-8 -*-
import mtool

#覆盖分区的已有数据（默认）
mtool.upload_data_to_partition("/tmp/test", "tmp.test_20181108", "dt=20181108")
#追加数据文件到表分区
mtool.upload_data_to_partition("/tmp/test", "tmp.test_20181108", "dt=20181108",
overwrite=False)
```

读取本地目录的数据文件，并返回pandas DataFrame

```
# -*- coding: utf-8 -*-
import mtool

df = mtool.read_dir_as_frame("/tmp/test")
df.show()
```

mtool api文档

分析类应用基础工具，目前提供对hive操作的功能封装：

1. 执行hive update sql、hive select sql
2. 将hive select sql的数据下载到本地目录
3. 将本地目录的数据上传到hive表
4. 将本地目录的数据上传到hive表的分区
5. 读取本地目录的数据，并返回pandas DataFrame对象

更新记录：

1. 2018-11-22 :

- 增加hive on spark sql运行报错时的自动降级处理，此时使用hive on yarn再运行一遍错误sql
- MHive.executeQueryLimit10K(self, sql_select, dictify=False, *args, **kwargs)增加dictify参数、True时返回list(dict)数据

MHive

```
MHive(self, user=None, database=None, auth_mechanism=None, kerberos_service_name=None,
engine=None)
```

MHive 对hive server2的操作封装、并提供应用接口

基于impyla模块封装

executeUpdate

```
MHive.executeUpdate(self, sql_execute, *args, **kwargs)
```

执行hive跑批sql，sql间用分号分隔

支持变量替换，可使用{0}、{1}、{var1}、{var2}的形式

Args:

```
sql_execute (str): 批量sql语句
*args (list):列表型参数
**kwargs (dict):字典型参数
```

executeQueryLimit10K

```
MHive.executeQueryLimit10K(self, sql_select, dictify=False, *args, **kwargs)
```

执行hive sql查询并返回结果

此方法并不适用于大规模取数，所以限制取数条数为1w条

Args:

```
sql_select (str): 待执行sql查询
dictify(boolean): 是否返回dict型数据，默认为False
```

Returns:

```
[list]: 结果数组
```

get_hive

```
get_hive(engine=None)
```

获取hive操作对象

传入engine参数以获取hive操作对象

```
engine (str, optional): Defaults to 'mr'

默认使用mr引擎，engine='spark'使用spark引擎
```

Returns:

```
[obj]: hive操作对象
```

upload_data_to_table

```
upload_data_to_table(dir_name, table_name, overwrite=True)
```

加载本地目录中的数据至hive表

模型输出一般为csv格式（','分隔），故用于装载数据的hive表需建成如下形式：

```
    CREATE TABLE tmp.tmp_mydata(
        acct_nbr bigint,
        credit_lmt decimal(15,2)
    )
    ROW FORMAT SERDE
    'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
    WITH SERDEPROPERTIES (
    'field.delim'=','
    )
    STORED AS textfile
```

Args:

```
dir_name (str): 本地数据目录
table_name (str): hive表名（tmp.tmp01的形式）
overwrite (bool, optional): Defaults to True. 是否覆盖已存在的数据，默认为覆盖
    False表示追加数据
```

upload_data_to_partition

```
upload_data_to_partition(dir_name, table_name, partition_name, overwrite=True)
```

加载本地目录中的数据至hive表分区

模型输出一般为csv格式（','分隔），故用于装载数据的hive表需建成如下形式：

```
    CREATE TABLE tmp.tmp_mydata(
        acct_nbr bigint,
        credit_lmt decimal(15,2)
    )
    ROW FORMAT SERDE
    'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
    WITH SERDEPROPERTIES (
    'field.delim'=','
    )
    STORED AS textfile
```

Args:

```
dir_name (str): 本地数据目录
table_name (str): hive表名 (tmp.tmp01的形式)
partition_name (str): hive表分区名 (data_dt='2018-08-31'的形式)
overwrite (bool, optional): Defaults to True. 是否覆盖已存在的数据，默认为覆盖
    False表示追加数据
```

download_data

```
download_data(sql_select, dir_name, fieldsby=',', nullas="''")
```

将hive表的数据下载至本地目录

Args:

```
sql_select (str): 筛选数据的sql语句
dir_name (str): 本地数据目录
fieldsby (str, optional): Defaults to ",". 字段分隔符
nullas (str, optional): Defaults to "''". null值的导出形式
```

run_system_command

```
run_system_command(command)
```

执行操作系统命令

Args:

```
command (str): 操作系统命令
```

Raises:

```
Exception: 受限的操作系统命令异常 (如'rm -fr')
CalledProcessError: 操作系统命令执行异常
```

read_dir_as_frame

```
read_dir_as_frame(dir_name, fieldsby=',')
```

读取本都数据目录下的文件（csv格式），并返回pandas的DataFrame对象

因hive的分布式导出会将数据拆分为多个数据文件，故此方法将多个数据文件整合成1个DataFrame对象

Args:

```
dir_name (str): 数据目录
fieldsby (str, optional): Defaults to ",". 字段分隔符
```

Raises:

```
Exception: 数据目录不存在异常
```

Returns:

```
[DataFrame]: DataFrame对象
```