看水位预测代码

我们拿到行业服务科发来的 python 脚本

名称	修改日期	类型	大小
__pycache__	2021/5/7 14:45	文件夹	
data	2021/5/7 14:53	文件夹	
output	2021/5/7 11:22	文件夹	
use_model	2021/5/7 11:23	文件夹	
clean_data.py	2021/5/7 11:21	Python 源文件	3 KB
config.ini	2021/5/7 11:21	配置设置	1 KB
get_data.py	2021/5/7 14:42	Python 源文件	4 KB
main.py	2021/5/7 11:21	Python 源文件	2 KB
prediction.py	2021/5/10 16:39	Python 源文件	4 KB
save.py	2021/5/7 11:28	Python 源文件	1 KB

我们是不能直接用的
有几个问题

1. 模型是用测点名区分的并且还使用了大量使用了中文
   为了防止一些编码上的问题或其他一些不必要的问题这里需要修改为使用测点id

prediction.py 8 | config.ini | Dockerfile

config.ini

```
[bottom]
北山电排站水位 = 15300
南新楼前电排站水位 = 14700
后河电排站水位 = 13750
山头电排站水位 = 14400
柘林电排站水位 = 15800
泾口电排站水位 = 15100
滁槎电排站水位 = 17300
玉丰电排站水位 = 15000
瑶溪肖家电排站水位 = 14850
红旗大泵电排站水位 = 13400

[diff_threshold]
threshold = 182.8

[test_type]
if_online = 1
```

```
十水位测点
ls = {'北山电排站水位':[615,3376],
      '南新楼前电排站水位':[623,3399],
      '后河电排站水位':[617,3379],
      '山头电排站水位':[649,3452],
      '柘林电排站水位':[686,3653],
      '泾口电排站水位':[549,3375],
      '滁槎电排站水位':[683,3649],
      '玉丰电排站水位':[620,3428],
      '瑶溪肖家电排站水位':[648,3451],
      '红旗大泵电排站水位':[616,3378]}
```

2. 平台api 和 天气接口是写死的，平台api token 也是写死的

```
data = { "stations":[value[1]]}
url = 'https://openapi.anxinyun.cn/api/v1/structures/{ids}/factors/283/stations/data?startTime={startTime}&endTime={endTime}&token=8dd412f4-411d-4533-b4f3-67b2bfa8ef
r = requests.post(url,headers=headers,data=json.dumps(data))
```

```
l = 'https://weatherssj.anxinyun.cn/weatherApp/weather/getWeatherData'
rain = requests.post(url,data = json.dumps(post_data),headers = headers)
```

3. 多处读取配置文件

```
cf = configparser.ConfigParser()
cf.read('config.ini')
threshold = cf.getfloat('diff_threshold','threshold')
```

```
cf = configparser.ConfigParser()
cf.read('con  int  ni')
test_type = cf.getint('test_type','if_online')
if(test_type):
    result_list = []
```

```
14
15  def offline_prediction(use_data, data_array, name):
16      cf = configparser.ConfigParser()
17      cf.read('config.ini')
18      result_df = pd.DataFrame(columns=['name', 'error'])
19      sw = np.append(np.array(use_data['water_level']), 0)
20      label = tf.keras.preprocessing.sequence.TimeseriesGenerator(
21          data=sw, targets=sw, length=12, sampling_rate=1, stride=1, batch_size=1)
22      label_list = []
```

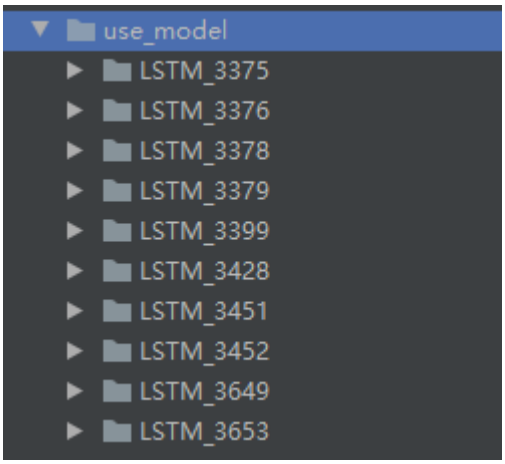
```
75
76
77  def online_prediction(data_array, name):
78      cf = configparser.ConfigParser()
79      cf.read('config.ini')
80      model = tf.keras.models.load_model('use_model/LSTM_'+name)
81      last_num = data_array[0, -1, 2].copy()
82      sca = StandardScaler()
```

4. 数据存储需要改为 ES

```
5   #name: str
6   #score: array like
7   def save_data(name,score):
8       db = pymysql.connect(host="127.0.0.1",user="root",password="123456",database="weather_data")
9       cursor = db.cursor()
0       now_time = datetime.datetime.now().strftime('%Y-%m-%d %H')+':00:00'
1       value = {}
2       for i in range(len(score)):
3           value[str(i)] = float(score[i])
4
```

另外 我们部署环境是k8s， 需要自己编写`Dockerfile`来构建镜像

开始修改代码

1. 

   把每个模型文件夹命名改为测点id,并修改相关引用代码，

```
def online_prediction(data_array, sid):
    model = tf.keras.models.load_model('app/use_model/LSTM_{0}'.format(sid))
    last_num = data_array[0, -1, 2].copy()
    sca = StandardScaler()
```

配置文件和相关代码修改：

```
[bottom]
level_sid_3376 = 15300
level_sid_3399 = 14700
level_sid_3379 = 13750
level_sid_3452 = 14400
level_sid_3653 = 15800
level_sid_3375 = 15100
level_sid_3649 = 17300
level_sid_3428 = 15000
level_sid_3451 = 14850
level_sid_3378 = 13400
```

```
    score = score - (score[0] - last_num)
    score = np.where(score < config.getint('bottom', "level_sid_{0}".format(sid)),
                     config.getint('bottom', "level_sid_{0}".format(sid)), score)
```

10水位测点修改：

```
stations = {
    3376: {
        'name': '北山电排站水位',
        'struct': 615,
        'sid': 3376
    },
    3399: {...},
    3379: {...},
    3452: {
        'name': '山头电排站水位',
        'struct': 649,
        'sid': 3452
    },
    3653: {...},
    3375: {...},
    3649: {
        'name': '滁槎电排站水位',
        'struct': 683,
        'sid': 3649
    },
    3428: {...},
    3451: {...},
    3378: {...}
}
```

2. 把天气和anxinyun平台 api 改为读取配置

```
        token = fetch_token()
        url = '{0}/structures/{1}/factors/283/stations/data?startTime={2}&endTime={3}&token={4}'.format(
            config.CLOUD_API,
            station['struct'], start_time, now_time, token)
        r = requests.post(url, headers=headers, data=json.dumps(data))
        if 200 == r.status_code:...

    # 进贤雨量
    post_data = {...}

    rain_start_time = (now_time_s - datetime.timedelta(hours=155)).strftime('%Y-%m-%dT%H') + ':00:00'
    rain_end_time = now_time_s.strftime('%Y-%m-%dT%H') + ':00:00'
    cut_time = (now_time_s - datetime.timedelta(hours=155)).strftime('%Y-%m-%d %H') + ':00:00'

    post_data['startTime'] = rain_start_time
    post_data['endTime'] = rain_end_time

    url = '{0}/weatherApp/weather/getWeatherData'.format(config.WEATHER_API)
    r_rain = requests.post(url, data=json.dumps(post_data), headers=headers)
    data_json = r_rain.json()
    data_rain = pd.DataFrame(columns=['time', 'precip'])
    for d in data_json['data']:...
```

3. 配置管理我们单独写一个文件，具备从配置文件和环境变量获取参数

```
def _init_args(self):
    config_path = os.path.join(basedir, config_dir)
    if os.path.exists(config_path):
        self._init_add_args()
        self._file_parser = configparser.ConfigParser()
        self._file_parser.read(os.path.join(basedir, config_dir), encoding='utf-8')
        self._read_input_args()
        self.init_args()
    else:
        raise Exception('file not exists:{}'.format(config_path))

def _init_add_args(self):...

def _read_input_args(self):...

def _set(self, section, key, value):
    self._file_parser.set(section, key, value)

def get(self, section, key):...

def getfloat(self, section, key):...

def getint(self, section, key):...

def init_args(self):
    self.ES_REST_NODES = getenv("ES_REST_NODES", self.get('es', 'nodes'))
    self.CLOUD_API = getenv("CLOUD_API", self.get('api', 'cloud_api'))
    self.WEATHER_API = getenv("WEATHER_API", self.get('api', 'weather_api'))
    self.DB_ANXINCLOUD_HOST = getenv("DB_ANXINCLOUD_HOST", self.get('pg', 'host'))
    self.DB_ANXINCLOUD_USER = getenv("DB_ANXINCLOUD_USER", self.get('pg', 'user'))
    self.DB_ANXINCLOUD_PASSWD = getenv("DB_ANXINCLOUD_PASSWD", self.get('pg', 'passwd'))
    self.DB_ANXINCLOUD_PORT = getenv("DB_ANXINCLOUD_PORT", self.get('pg', 'port'))
    self.DB_ANXINCLOUD_NAME = getenv("DB_ANXINCLOUD_NAME", self.get('pg', 'name'))
```

在每个使用参数的地方添加引用 `config` ，

> 这里说下 `ConfigParser` 这个配置管理用的工具类，会把读取的配置存到对象实例，没必要每次都去读配置文件

4. 数据存到ES ,会用到 `elasticsearch6`
   ES 保存也很简单：

```
es = Elasticsearch(hosts=config.ES_REST_NODES.split(','))
for index, value in enumerate(score):
 data['forecast_entire'].append({
     "forecast_time": '{}:00:00'.format((pre_now +
timedelta(hours=index)).strftime('%Y-%m-%d %H')),
     "value": float('%.4f' % ((float(value) - param) / 1000))
 })
```

可以看到上面对数据结果做了简单计算和精度处理。
从数据库读取参数：

```
class PostgresDbHelper(object):
 def __init__(self):
     self._logger = logging.getLogger('PostgresDbHelper')
     self._connection = psycopg2.connect(
         user=config.DB_ANXINCLOUD_USER,
         password=config.DB_ANXINCLOUD_PASSWD,
         host=config.DB_ANXINCLOUD_HOST,
         port=config.DB_ANXINCLOUD_PORT,
         database=config.DB_ANXINCLOUD_NAME
      )

 def get_formal_params(self, sids):
     try:
         cursor = self._connection.cursor()
         str_ids = ','.join('%s' % id for id in sids)
         query = "SELECT sensor,params from t_device_sensor where sensor in
({0});".format(str_ids)
         cursor.execute(query)
         records = dict((record[0], float(record[1].get('DAQ0', 0))) for
record in cursor.fetchall())
         return records
     except Exception as ex:
         self._logger.warning("select formula params error", ex)
     finally:
         self._connection.commit()
```

5. 编写 `Dockerfile`

```
FROM repository.anxinyun.cn/base-images/python-slim:9.21-05-25

WORKDIR /apps

COPY . .

RUN chmod u+x ./start.sh

CMD ["./start.sh" ]
```

上面基础镜像是定做的，因为网络环境不是很稳定，直接把虚拟环境和依赖打到基础镜像里了.