工作总结

> 💡 信者行之基，行者人之本。 —— 刘昼

汇报人： **@管中港**

部门：人工智能部

日期：2022年3月24日

关于java端调用sklearn模型的总结

1.sklearn 模型保存为pmml文件

一般可以通过sklearn2pmml或者nyoka这两个包来将sklearn模型的打包为pmml的格式，在由java进行调用。pmml模型具有平台无关性，可以实现跨平台部署。

java调用sklearn模型的一般步骤如下：

- 将数据（文本或非文本）转化为表格数据的形式，即col_name:data
- 使用特定的pipeline包裹每步操作，并在该管道下进行fit，其中两种方式的包裹如下：

Python

```python
from sklearn.pipeline import Pipeline
from sklearn_pandas import DataFrameMapper

lr_pmml = Pipeline([
    ("mapper",DataFrameMapper([
("query",TfidfVectorizer(analyzer='char',min_df=5,max_features=1024,dtype=np.float32)),
        ])),
    ('lr', LogisticRegression())])

print("lr_pmml:",lr_pmml)

lr_pmml.fit(train_df,train_y)



from sklearn2pmml.pipeline import PMMLPipeline
from sklearn_pandas import DataFrameMapper

lr_pmml_ = PMMLPipeline([
    ("mapper",DataFrameMapper([
("query",TfidfVectorizer(analyzer='char',min_df=5,max_features=1024,dtype=np.float32)),
        ])),
    ('lr', LogisticRegression())])

print("lr_pmml_:",lr_pmml_)

lr_pmml_.fit(train_df,train_y)
```

- 可以看到，两种方式的定义格式基本相同只是基础包不一样。其中DataFrameMapper主要用来包裹对特征的处理方式，比如：标准化、归一化等。我在这里是将每条query当作了col_name为query下的一个表格数据，然后就是对query整列数据进行TfidfVectorizer。需要注意的有以下三点：
  - "mapper"是固定的
  - "query"表示表格数据中的query一列
  - TfidfVectorizer中dtype=np.float32表示半精度，默认为64，当精度再下降时TfidfVectorizer会报错
- 然后就是保存，两种定义的保存方式也不一样

Python

```python
from nyoka import skl_to_pmml

skl_to_pmml(lr_pmml,col_names=
['query'],pmml_f_name='classifier_f1_%.3f.pmml'%f1_score)


from sklearn2pmml import sklearn2pmml

sklearn2pmml(lr_pmml_,'classifier_f1_%.3f.pmml'%f1_score,with_repr = True)
```

- 对于使用skl_to_pml的保存没问题，但是使用sklearn2pmml的保存会报以下错误，可以看出是运行时间错误，考虑到进行TfidfVectorizer会使得参数量相当多，所以猜测sklearn2pmml的保存可能有一定的参数上限，相比skl_to_pmml则可以容纳更多的参数量

Python

```
RuntimeError: The JPMML-SkLearn conversion application has failed.
 The Java executable should have printed more information
 about the failure into its standard output and/or standard error streams
```

2.java调用pmml文件

java对pmml文件的调用有以下几步：

- 首先，添加以下依赖

Java

```java
"org.jpmml" % "pmml-evaluator" % "1.4.15",
"org.jpmml" % "pmml-evaluator-extension" % "1.4.15",
"com.huaban" % "jieba-analysis" % "1.0.2"  //如果需要分词的话
```

- 导入以下类

Java

```java
import javax.xml.bind.JAXBException;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.*;

import org.dmg.pmml.FieldName;
import org.dmg.pmml.PMML;
import org.jpmml.evaluator.*;
import org.xml.sax.SAXException;
```

- 读入pmml文件，这里有一点需要注意
  - 第27行modelEvaluatorFactory.newModelEvaluator(pmml)的输入只有pmml，但是这在1.5.x版本中是会报错的，因为1.5.x版本的输入为modelEvaluatorFactory.newModelEvaluator(pmml,model)，所以安装依赖时需要严格控制pmml-evaluator的版本

Java

```java
public static Evaluator loadPmml(String path){
    PMML pmml=new PMML();
    InputStream inputStream=null;
    File file=new File(path);
    //建立文件实例
    try{
        inputStream=new FileInputStream(file);
    }catch (Exception e){
        e.printStackTrace();
    }
    if(inputStream==null){
        return null;
    }
    //读取文件
    try{
        pmml=org.jpmml.model.PMMLUtil.unmarshal(inputStream);
    }catch (JAXBException | SAXException e1){
        e1.printStackTrace();
    }
    //关闭文件
    try{
        inputStream.close();
    }catch (IOException e){
        e.printStackTrace();
    }
    ModelEvaluatorFactory
modelEvaluatorFactory=ModelEvaluatorFactory.newInstance();
    Evaluator evaluator = modelEvaluatorFactory.newModelEvaluator(pmml);
    pmml=null;
    return evaluator;
}
```

- 导入模型参数

Java

```java
Map<FieldName, FieldValue> arguments = new LinkedHashMap<>();//存储模型参数
//导入模型参数
for (InputField inputField:inputFields){
    FieldName inputFieldName=inputField.getName();//参数名
    FieldValue
inputValue=inputField.prepare(map.get(inputFieldName.getValue()));//句子输入
    arguments.put(inputFieldName,inputValue);
}
```

- java调用pmml文件进行预测时就是以其中的参数进行 $y = w^t x + b$ 形式的计算

Java

```java
Map<FieldName,?> result=model.evaluate(arguments);
```

- 取出result中分类出的类别

Java

```java
Map<String, Object> resultMap = new HashMap<>();
for(TargetField targetField : targetFields) {
    FieldName targetFieldName = targetField.getName();
//            System.out.println("targetFieldName"+targetFieldName);
    Object targetFieldValue = result.get(targetFieldName);
    if (targetFieldValue instanceof Computable) {
        Computable computable = (Computable) targetFieldValue;
        resultMap.put(targetFieldName.getValue(), computable.getResult());
    }else {
        resultMap.put(targetFieldName.getValue(), targetFieldValue);
    }
}
```

- 整个过程还有一个重要问题，在调用pmml模型的时候，通常会报错，找不到http://www.dmg.org/PMML-4_4，这一般时pmml版本与java的对应没做好。原始的pmml开头是上面代码这样，只用将4.4改为4.3就可以正常运行

HTML

```html
<?xml version="1.0" encoding="UTF-8"?>
<PMML xmlns="http://www.dmg.org/PMML-4_4" version="4.4.1">
    <Header copyright="Copyright (c) 2021 Software AG" description="Default description">
        <Application name="Nyoka" version="5.2.0"/>
        <Timestamp>2022-03-22 17:54:52.470393</Timestamp>
    </Header>



<?xml version="1.0" encoding="UTF-8"?>
<PMML xmlns="http://www.dmg.org/PMML-4_3" version="4.3.1">
    <Header copyright="Copyright (c) 2021 Software AG" description="Default description">
        <Application name="Nyoka" version="5.2.0"/>
        <Timestamp>2022-03-22 17:54:52.470393</Timestamp>
    </Header>
```

3.碰到的问题

使用java调用sklearn模型可以成功运行，但是在预测时发现输入中文句子，输出的概率很多都一样，输入英文时可以这场预测。查看pmml文件后，发现如下问题：

HTML

```
<NumericPredictor name="tfidf@[query](s)" exponent="1" coefficient="-1.6606425890343206"/>
<NumericPredictor name="tfidf@[query](t)" exponent="1" coefficient="-4.6560710174221311"/>
<NumericPredictor name="tfidf@[query](u)" exponent="1" coefficient="-2.0655156213307730"/>
<NumericPredictor name="tfidf@[query](v)" exponent="1" coefficient="-2.3017803101764049"/>
<NumericPredictor name="tfidf@[query](w)" exponent="1" coefficient="-3.6962048914338621"/>
<NumericPredictor name="tfidf@[query](x)" exponent="1" coefficient="-4.2844472787296892"/>
<NumericPredictor name="tfidf@[query](y)" exponent="1" coefficient="-1.0379443563439172"/>
<NumericPredictor name="tfidf@[query](\xe4\xb8\x81)" exponent="1" coefficient="-2.7510269805589518"/>
<NumericPredictor name="tfidf@[query](\xe4\xb8\x87)" exponent="1" coefficient="-1.8688661028053180"/>
<NumericPredictor name="tfidf@[query](\xe4\xb8\x91)" exponent="1" coefficient="1.1914348323726289"/>
<NumericPredictor name="tfidf@[query](\xe4\xb8\x93)" exponent="1" coefficient="-3.3635811569164296"/>
<NumericPredictor name="tfidf@[query](\xe4\xb8\x96)" exponent="1" coefficient="-2.2035012627585639"/>
<NumericPredictor name="tfidf@[query](\xe4\xb8\x9a)" exponent="1" coefficient="2.2454507982333558"/>
```

- 英文字母是可以正常表示的，但是中文被表示为16进制的格式

想到的解决方案有：

- 保存pmml是否有参数可以控制中文编码
- 读取pmml的时候，是否有参数可以控制中文编码
- 将java端的输入转换为16进制

但是，考虑到这个问题解决后也无法保证没有其他问题的出现，而且，时间上也不允许，所以就中断这个方案了。

4.思考总结

在网上搜集资料时，使用的数据基本都是表格数据，其中的数据类型有数值和英文（用tfidf来处理英文作为其中的一个特征来训练模型），对于这种纯nlp的任务没有找到例子。考虑到，在业务上，对于nlp任务很少使用到sklearn，所以，猜测pmml这种格式不一定支持中文的处理。