文章
Hao Ma · 三月 25, 2021 阅读大约需 分钟

Open Exchange

为什么 COVID-19 对机器学习也有危险？(Part II)

继上一部分，现在要利用 IntegratedML VALIDATION MODEL 语句提供信息以监视您的 ML 模型。
您可以在 此处观看实际运作。

此处所示代码衍生自 InterSystems IntegragedML 模板或 IRIS 文档 提供的示例，我主要是把代码混合了起来。
这是一个简单的示例，目的是为进一步讨论和未来工作提供一个起点。

注：此处提供的代码仅作说明之用。如果您想尝试，我开发了一个 Open Exchange 报示例应用
(iris-integratedml-monitor-example)，并将其提交到 InterSystems IRIS AI Contest。
读完这篇文章后您可以去投，如果喜欢，就请 投一票吧！:)

目录

第一部分：

- IRIS IntegratedML 和 ML 系统
- 新旧常态之间

第二部分：

- 监视 ML 性能
- 简单用例
- 未来工作

监视 ML 性能

要监视 ML 模型，至少需要两个功能：

1) 性能指标提供程序 2) 监视和通知服务

幸运的是，IRIS 为我们提供了这两个必要的功能。

获取 ML 模型性能指标

如上一部分所示，IntegratedML 提供了 VALIDATE MODEL 语句来计算以下参数：

- 准确率：模型的好坏（值接近 1 表示正确答案率高）
- 精度：模型处理误报的能力如何（值接近 1 表示 **无**误报率高）
- 召回率：模型处理漏报的能力如何（值接近 1 表示 **无**漏报率高）
- F 度量：另一种测量准确率的方法，用于准确率表现不佳的情况（值接近 1 表示正确答案率高）

注：这些定义并不是正式的，而且非常浅显！我推荐您一些时间 了解它们。

最妙的是，每次调用 VALIDATE MODEL 时，IntegrateML
都会存储它的指标，这样的功能可以很好地用于监视。

监视引擎

InterSystems IRIS 提供 System Monitor 框架用于处理监视任务。
它还允许您定义自定义规则，以根据这些指标上应用的谓词触发通知。

默认提供磁盘、内存、进程、网络等一系列指标。此外，System Monitor
还可以让您扩展监视器，覆盖无限的可能性这样的自定义监视器在系统监视器术语中称为应用监视器。

您可以在 此处了解有关 System Monitor 的更多信息。

整合

现在，有了一种获取这模型验证指标值的方法，还有一个可以根据应用于自定义指标源的自定义规则触发警报的
工具。那么，是时候把它们结合起来了。

首先，我们需要通过扩展 %Monitor.Abstract 类创建自定义应用监视器类，并实现 *Initialize* 和 *GetSample* 方法。

```
Class MyMetric.IntegratedMLModelsValidation Extends %Monitor.Adaptor
{

/// Initialize the list of models validation metrics.
Method Initialize() As %Status
{
    Return $$$OK
}

/// Get routine metric sample.
/// A return code of $$$OK indicates there is a new sample instance.
/// Any other return code indicates there is no sample instance.
Method GetSample() As %Status
{
    Return $$$OK
}

}
```

系统监视器会定期发出调用以监视类，获取一组称为样本的指标。
这样的样本可以仅用于监视，也可用于检查是否必须提高警报规则。
您可以通过在监视器类中定义标准的非内部属性来定义此类样本的结构。需注意的是，必须在参数 INDEX
中指定其中一个属性为每个样本的主键，否则将抛出键重复错误。

```
Class MyMetric.IntegratedMLModelsValidation1 Extends %Monitor.Adaptor
{

Parameter INDEX = "ModelTrainedName";

/// Name of the model definition
Property ModelName As %Monitor.String;

/// Name of the trained model being validated
Property ModelTrainedName As %Monitor.String;
```

```
/// Validation error (if encountered)
Property StatusCode As %Monitor.String;

/// Precision
Property ModelMetricPrecision As %Monitor.Numeric;

/// Recall
Property ModelMetricRecall As %Monitor.Numeric;

/// F-Measure
Property ModelMetricFMeasure As %Monitor.Numeric;

/// Accuracy
Property ModelMetricAccuracy As %Monitor.Numeric;

...

}
```

Initialize 方法在每次监视器调用时被调用一次，*GetSample* 方法则被调用到返回 0 为止。

因此，我们现在 IntegrateML 验证历史上设置 SQL，向监视器提供指标信息，实现 *Initialize* 和 *GetSample* 方法：

```
/// Initialize the list of models validation metrics.
Method Initialize() As %Status
{
    // Get the latest validation for each model validated by VALIDATION MODEL stateme
nt
    Set sql =
    "SELECT MODEL_NAME, TRAINED_MODEL_NAME, STATUS_CODE, %DLIST(pair) AS METRICS_LIST
 FROM ("_
        "SELECT m.*, $LISTBUILD(m.METRIC_NAME, m.METRIC_VALUE) pair, r.STATUS_CODE "_
        "FROM INFORMATION_SCHEMA.ML_VALIDATION_RUNS r "_
        "JOIN INFORMATION_SCHEMA.ML_VALIDATION_METRICS m "_
        "ON m.MODEL_NAME = r.MODEL_NAME "_
            "AND m.TRAINED_MODEL_NAME = r.TRAINED_MODEL_NAME "_
            "AND m.VALIDATION_RUN_NAME = r.VALIDATION_RUN_NAME "_
        "GROUP BY m.MODEL_NAME, m.METRIC_NAME "_
        "HAVING r.COMPLETED_TIMESTAMP = MAX(r.COMPLETED_TIMESTAMP)"_
    ") "_
    "GROUP BY MODEL_NAME"
    Set stmt = ##class(%SQL.Statement).%New()
    $$$THROWONERROR(status, stmt.%Prepare(sql))
    Set ..Rspec = stmt.%Execute()
    Return $$$OK
}

/// Get routine metric sample.
/// A return code of $$$OK indicates there is a new sample instance.
/// Any other return code indicates there is no sample instance.
Method GetSample() As %Status
{
    Set stat = ..Rspec.%Next(.sc)
    $$$THROWONERROR(sc, sc)

    // Quit if we have done all the datasets
```

```
    If 'stat {
        Quit 0
    }

    // populate this instance
    Set ..ModelName = ..Rspec.%Get("MODEL_NAME")
    Set ..ModelTrainedName = ..Rspec.%Get("TRAINED_MODEL_NAME")_" ["_$zdt($zts,3)_"]"
    Set ..StatusCode = ..Rspec.%Get("STATUS_CODE")
    Set metricsList = ..Rspec.%Get("METRICS_LIST")
    Set len = $LL(metricsList)
    For iMetric = 1:1:len {
        Set metric = $LG(metricsList, iMetric)
        Set metricName = $LG(metric, 1)
        Set metricValue = $LG(metric, 2)
        Set:(metricName = "PRECISION") ..ModelMetricPrecision = metricValue
        Set:(metricName = "RECALL") ..ModelMetricRecall = metricValue
        Set:(metricName = "F-MEASURE") ..ModelMetricFMeasure = metricValue
        Set:(metricName = "ACCURACY") ..ModelMetricAccuracy = metricValue
    }

    // quit with return value indicating the sample data is ready
    Return $$$OK
}
```

编译监视器类后，您需要重新启动系统监视器，使系统意识到一个新的监视器已经创建并可以使用。您可以使用 ^%SYSMONMGR 例程或 %SYS.Monitor 类来完成这一步。

简单用例

这样就有了所需的工具来收集、监视和发布 ML 性能指标的警报。
接下来要做的是定义自定义警报规则，模拟已部署的 ML 模型开始对性能造成负面影响的场景。

首先，我们必须配置电子邮件警报及其触发规则。这可以使用 ^%SYSMONMGR 例程完成
不过，为了方便创建了一个设置方法，它可以设置所有电子邮件配置和警报规则。您需要将 <>
之间的值替换为您的电子邮件服务器和帐户参数。

```
ClassMethod NotificationSetup()
{
    // Set E-mail parameters
    Set sender = "<your e-mail address>"
    Set password = "<your e-mail password>"
    Set server = "<SMTP server>"
    Set port = "<SMTP server port>"
    Set sslConfig = "default"
    Set useTLS = 1
    Set recipients = $LB("<comma-separated receivers for alerts>")
    Do ##class(%Monitor.Manager).AppEmailSender(sender)
    Do ##class(%Monitor.Manager).AppSmtpServer(server, port, sslConfig, useTLS)
    Do ##class(%Monitor.Manager).AppSmtpUserName(sender)
    Do ##class(%Monitor.Manager).AppSmtpPassword(password)
    Do ##class(%Monitor.Manager).AppRecipients(recipients)

    // E-mail as default notification method
    Do ##class(%Monitor.Manager).AppNotify(1)
```

```
    // Enable e-mail notifications
    Do ##class(%Monitor.Manager).AppEnableEmail(1)

    Set name  = "perf-model-appointments-prediction"
    Set appname = $namespace
    Set action = 1
    Set nmethod = ""
    Set nclass = ""
    Set mclass = "MyMetric.IntegratedMLModelsValidation"
    Set prop = "ModelMetricAccuracy"
    Set expr = "%1 < .8"
    Set once = 0
    Set evalmethod = ""
    // Create an alert
    Set st = ##class(%Monitor.Alert).Create(name, appname, action, nmethod, nclass, m
class, prop, expr, once, evalmethod)
    $$$THROWONERROR(st, st)

    // Restart monitor
    Do ##class(MyMetric.Install).RestartMonitor()
}
```

在以前的方法中，警报在监视器获得的准确率值小于 90% 时发出。

现在，设置警报规则后，让我们用前 500 条记录创建、训练和验证履约/失约预测模型，并通过前 600 条记录进行验证。

注：*种子*参数只是为了保证可重复性（没有随机值），通常在生产中必须避免。

```
-- Creates the model
CREATE MODEL AppointmentsPredection PREDICTING (Show) FROM MedicalAppointments USING
{\"seed\": 3}
-- Train it using first 500 records from dataset
TRAIN MODEL AppointmentsPredection FROM MedicalAppointments WHERE ID <= 500 USING {\"
seed\": 3}
-- Show model information
SELECT * FROM INFORMATION_SCHEMA.ML_TRAINED_MODELS
```

```
|   | MODEL_NAME             | TRAINED_MODEL_NAME      | PROVIDER | TRAINED_TIMESTAMP
       | MODEL_TYPE     | MODEL_INFO                                              |
|---|------------------------|-------------------------|----------|-----------------
-------|----------------|---------------------------------------------------------|
| 0 | AppointmentsPredection | AppointmentsPredection2 | AutoML   | 2020-07-12 04:46:
00.615 | classification | ModelType:Logistic Regression, Package:sklearn... |
```

注意：意味着，使用 AutoML 作为提供程序（PROVIDER 列），IntegrateML 从提供的数据集中采用逻辑回归算法从 scikit-learn 库（MODEL_INFO 列）推断出分类模型（MODEL_TYPE 列）。这里必须强调“垃圾进，垃圾出”规则，即模型质量与数据质量直接相关。

接下来继续进行模型验证。

```
-- Calculate performace metrics of model using first 600 records (500 from trainning
set + 100 for test)
VALIDATE MODEL AppointmentsPredection FROM MedicalAppointments WHERE ID < 600 USING {
```

```
\"seed\": 3}
-- Show validation metrics
SELECT * FROM INFORMATION_SCHEMA.ML_VALIDATION_METRICS WHERE MODEL_NAME = '%s'


| METRIC_NAME              | Accuracy | F-Measure | Precision | Recall |
|--------------------------|----------|-----------|-----------|--------|
| AppointmentsPredection21 | 0.9      | 0.94      | 0.98      | 0.91   |
```

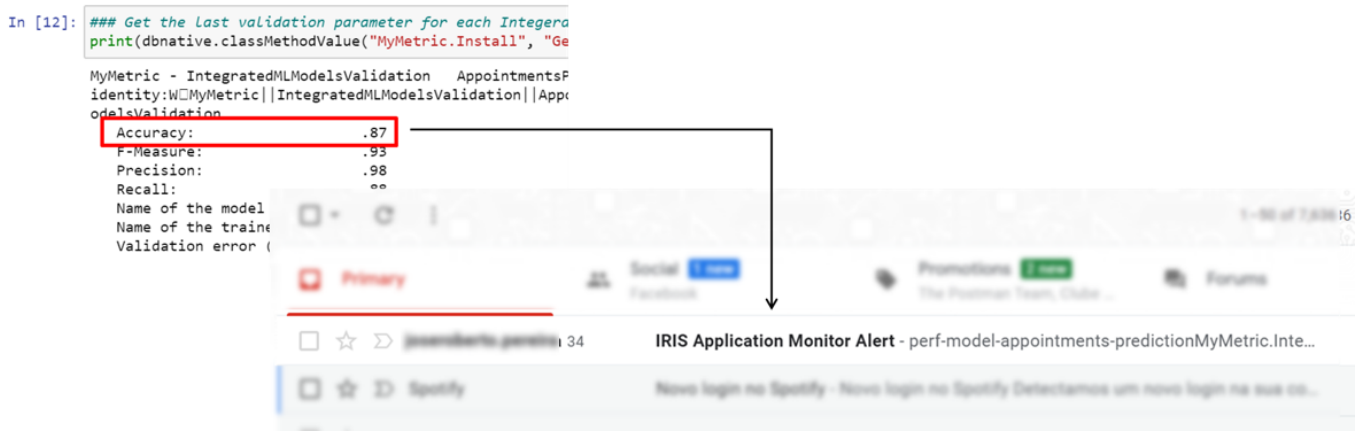
模型可用于通过 PREDICT 语句执行预测：

```
SELECT PREDICT(AppointmentsPredection) As Predicted, Show FROM MedicalAppointments  W
HERE ID <= 500


|     | Predicted | Show  |
|-----|-----------|-------|
| 0   | 0         | False |
| 1   | 0         | False |
| 2   | 0         | False |
| 3   | 0         | False |
| 4   | 0         | False |
| ... | ...       | ...   |
| 495 | 1         | True  |
| 496 | 0         | True  |
| 497 | 1         | True  |
| 498 | 1         | True  |
| 499 | 1         | True  |
```

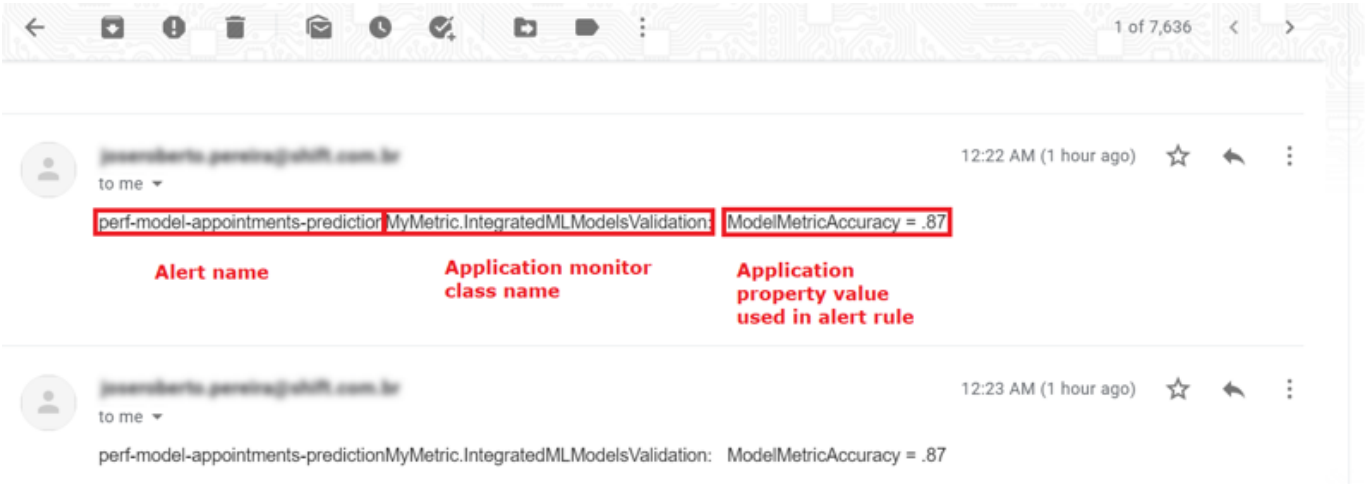
然后，我们来模拟在模型中添加 200 条新记录（共 800 条记录），使模型准确率降低到 87%。

```
-- Calculate performace metrics of model using first 800 records
VALIDATE MODEL AppointmentsPredection FROM MedicalAppointments WHERE ID < **800** USI
NG {\"seed\": 3}
-- Show validation metrics
SELECT * FROM INFORMATION_SCHEMA.ML_VALIDATION_METRICS WHERE MODEL_NAME = '%s'


| METRIC_NAME              | Accuracy | F-Measure | Precision | Recall |
|--------------------------|----------|-----------|-----------|--------|
| AppointmentsPredection21 | 0.9      | 0.94      | 0.98      | 0.91   |
| AppointmentsPredection22 | 0.87     | 0.93      | 0.98      | 0.88   |
```

由于我们一些时候设置了当准确率低于 90%
时发出电子邮件通知的规则，系统监视器意识到是时候向相关电子邮件帐户发出警报了。



在电子邮件正文中，您可以找到有关警报的信息，例如它的名称、应用监视器和触发警报的指标值。



这样一来，人们可以在收到通知后及时采取应对措施。
例如，一个操作可能只是简单的重新训练模型，但在某些情况下可能需要更详细的方法。

当然，您可以详细说明监视指标并创建更好的警报。例如，假设您有多个 ML 模型，每个模型由不同的人员负责运行。
您可以使用模型名称指标，并为特定的电子邮件接收者设置特定的警报规则。

系统监视器还允许您用 ClassMethod 代替电子邮件。
也就是说您可以在引发警报时执行复杂的逻辑，例如自动重新训练模型。

需注 意的是，由于系统监视器定期运行 Initialize 和 GetSample 方法，所以这些方法需仔细设计，以免占用太多系统资源。

未来工作

正如 Benjamin De Boe 注 意到的，IRIS 引入了一种自定义监视任务的新方法，即 SAM 工具。
我的第一印象非常积极，因为 SAM 与 Grafana 和 Prometheus 等市面上的标准监视技术相集成
那么，为什么不去测试如何用这样的新功能来改进这项工作呢？因为这是留给未来工作的材料 :)

那么到这里了！ 望本文对您有一定的帮助。再见！

#AI #IntegratedML #分析 #机器学习 #竞赛 #InterSystems IRIS #Open Exchange
在 InterSystems Open Exchange 上检查相关应用程序

源 URL: https://cn.community.intersystems.com/post/%E4%B8%BA%E4%BB%80%E4%B9%88-covid-19-%E5%AF%B9%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E4%B9%9F%E6%9C%89%E5%8D%B1%E9%99%A9%EF%BC%9F-part-ii