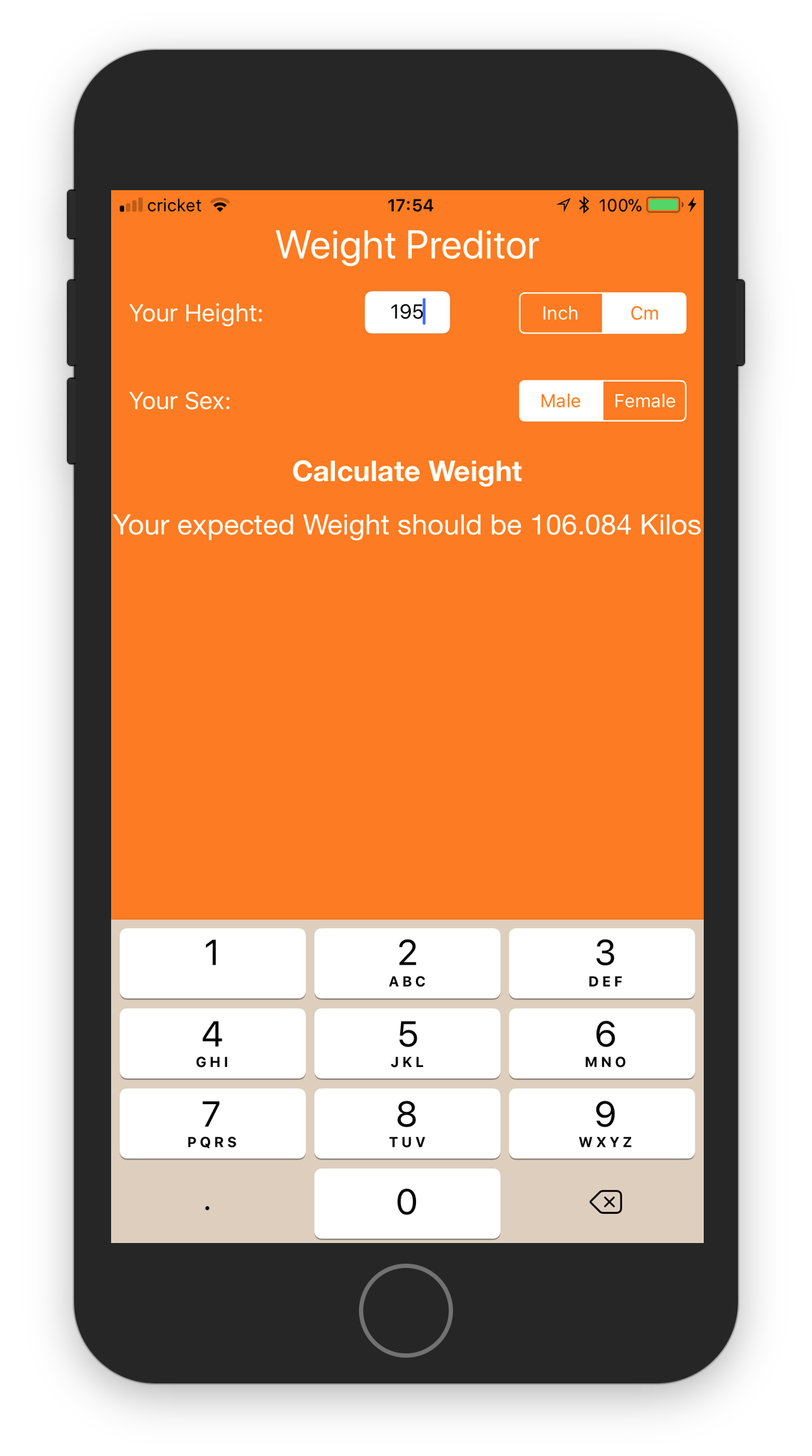
# CoreML、Keras和TensorFlow - 超级简单的端到端测试



这不是一个完整的深入学习教程，只是一个超级简单的端到端测试日志，关于如何将Keras、TensorFlow和CoreML一起使用。

我们将在Swift中构建一个iOS应用程序，该应用程序将使用CoreML从给定高度预测重量，在CoreML中创建并导入Keras/TensorFlow线性回归模型。

应用程序的源代码以及使用CoreML工具创建和导出Keras/Tensorflow模型的Python脚本可从以下站点获得：

[**JacopoMangiavacchi/HeightWeightCoreMLKerasTest** HeightWeightCoreMLKerasTest - Super simple end to end test of Keras, TensorFlow and CoreMLgithub.com](https://github.com/JacopoMangiavacchi/HeightWeightCoreMLKerasTest)

我们将在此测试中建立和使用的极其简单的线性回归模型是基于Udemy平台上非常有用的Keras/Tensorflow课程中的一些身高/体重统计数据：

老实说，在这个非常基本的场景中没有真正的理由使用ML，但无论如何，我决定用文档化的逐步说明来记录这个非常简单但完整的端到端测试，因为它明确地帮助我在学习新事物的同时连接所有的点，并从我的“移动第一”背景发展到“人工智能第一”时代。

首先：在桌面（macOS或Windows）或容器上设置ML环境。

首先要做的是从这个链接下载并安装水蟒

Anaconda是一个包管理器、环境管理器、Python发行版，以及720多个免费且易于安装的开源包的集合，这些包通常对处理Python有很大帮助，但在加载数据集（例如.CSV文件）时尤其如此，轻松绘制图形并轻松导入所需的所有Python包，如Keras、CoreML工具和其他。

当然水蟒不是严格需要的，但我真的推荐这个。

一旦安装了Anaconda，就可以轻松地设置环境，创建一个environment.yml文件，如下所示：

如果需要，可以导入此环境。yml使用以下命令在本地计算机上克隆项目存储库：git clone https://github.com/JacopoMangiavacchi/HeightWeightCoreMLKerasTest.git

一旦有了environment.yml文件，就可以让Anaconda安装Python 2.7、TensorFlow 1.1、Keras 2.0.6、CoreMLTools 0.6.3、Pandas和其他Python使用这个简单的命令：conda env create

创建环境后，您只需在macOS/linux source activate KerasTensorFlowCoreML或Windows activate KerasTensorFlowCoreML上执行此命令即可激活它

现在，您终于可以使用命令Jupyter Notebook启动超级有用的Jupyter笔记本编辑器，并打开浏览器以访问此本地url

如果您已经克隆了Git repo，那么您的文件夹中将已经有一个完整的Jupyter笔记本要打开：createModel.ipynb

另一方面，您需要在新笔记本中复制并粘贴以下Python代码。

在任何情况下，一旦编辑完完整的笔记本，请执行所有单元（Python指令）来创建、测试和导出模型。

### 创建Keras/Tensorflow模型

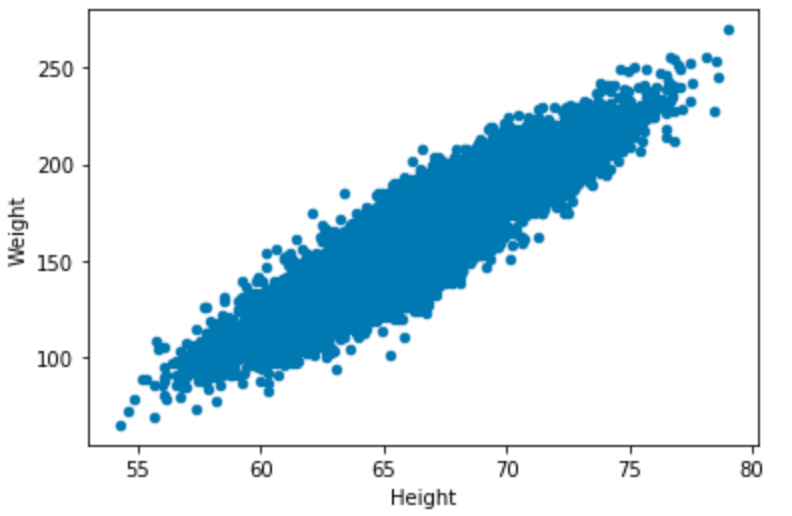
#### Python笔记本说明说明

如前所述，该项目将创建一个非常简单的线性回归模型，用于从给定高度预测体重。

首先要做的是导入一些Python实用程序（如Pandas）并导入HeightWeight统计文件

然后你就可以画出一个体重/身高的图表，用你的眼睛就能理解回归。。

.. 得到这样一个图：



现在，您可以根据此高度和重量数据创建、编译和训练（拟合）简单线性回归模型，方法是按照以下说明创建、编译和训练模型：

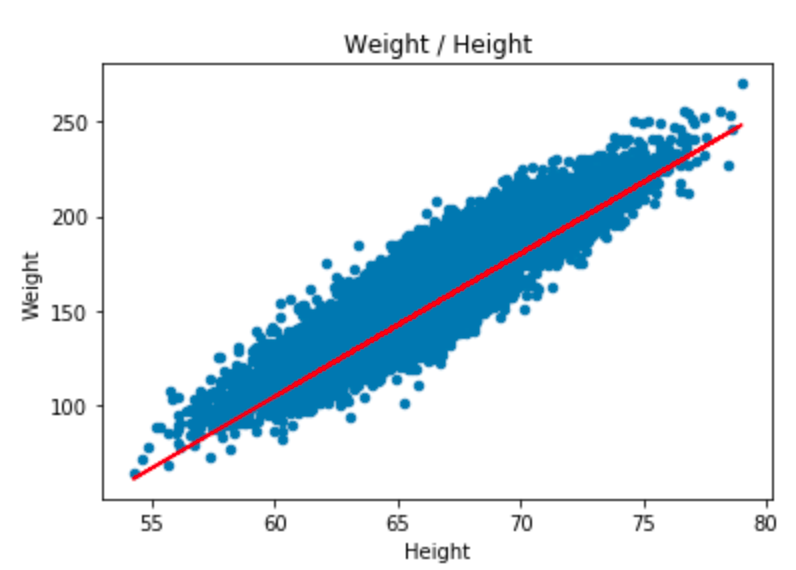
该模型非常容易地训练一个输入，一个浮点数数组（高度）为X，一个浮点数数组（权重）为y。

再次提醒，这不是Keras/Tensorflow教程。我刚刚在这里创建了一个简单的序列线性回归模型，并训练了所有可用的高度/重量统计数据集。这显然不是最佳实践，但对我来说，这只是使用较少的Python代码行创建模型的最佳方法！

现在，如果需要，可以使用Keras前端API在Python中直接测试模型。。

.. 再将体重/身高数据加上与熊猫的线性回归图。。

.. 得到这样一个图：



### 导出CoreML模型

仍然使用Jupyter Notebook或您最喜欢的Python环境，您需要使用Apple开源CoreML工具将模型转换并导出为CoreML模型。

使用CoreMLTools非常容易，您需要在Python中导入它，并基本上调用Keras转换器，如果需要的话，为input（Height）和output（Weight）数组参数命名，并像下面的代码片段中那样保存CoreML模型：

CoreML模型将作为HeightWeight\_Model.mlmodel保存在当前文件夹中

如果需要，可以直接从Python调用CoreML predict API来测试CoreML转换，验证返回的值是否对您使用以下命令有意义：

再次请注意，我们只是简单地训练整个数据集，在这个超级简单的场景中我们不提供任何测试。

成功创建并导出CoreML模型后，您可以使用以下命令继续释放存储空间来清理Python、Keras、TensorFlow、CoreMLTools环境：source deactivate conda remove-y-n KerasTensorFlowCoreML - 所有CoreML模型都不会被删除，它将保留在您的文件夹中

### 构建iOS示例项目

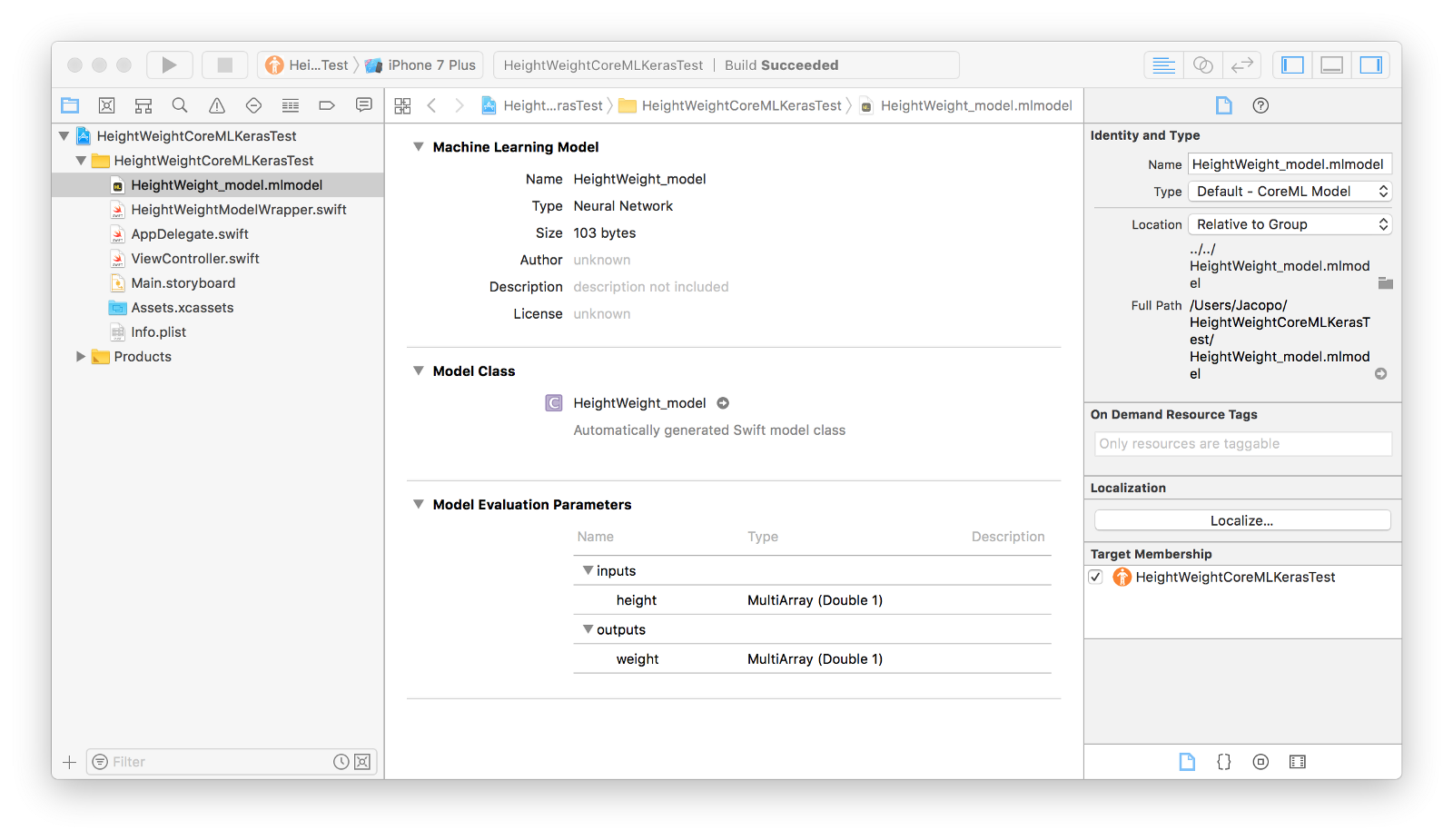
如果您已经克隆了Git repo，现在只需在XCode 9中打开iOS示例项目HeightWeightCoreMLKerasTest.xcodeproj，然后在您的iPhone或模拟器上构建和测试应用程序。

iOS示例项目使用一个Swift包装类（HeightWeightModelWrapper.Swift）来封装所有CoreML API，简化CoreML多数组的使用，并实现一些实用程序，如从厘米到英寸、磅到千克的转换。

使用这个来自Swift的简单模型来预测给定高度的重量，就像执行这两行代码一样简单！

如果你想为这个身高/体重模型创建自己的iOS Swift示例应用程序，你可以打开XCode 9，创建一个空的单视图应用程序项目并开始构建自己的用户界面。

一旦你有了你的用户界面，你就可以在你的XCode 9项目中把CoreML模型拖放到上面（HeightWeight\_model.mlmodel），但是请记住要检查你的项目的目标成员资格是否被选中了项目中的.mlmodel文件，如下图所示：



CoreML和任何苹果技术一样，是一种使用的魔法，它只需在XCode中导入模型并构建解决方案，它就会自动生成一个Swift类，直接在iOS项目中使用该模型并进行预测。

我在这里发现的唯一一个小问题是，有时CoreML MLMultiArray多维数组基本上是管道代码，您可以很容易地避免这些代码在UIViewControllers中或通常在项目Swift类和结构中散布。更一般地说，我发现为CoreML生成的Swift代码开发自己的包装类可能更容易，以便从客户机应用程序代码中强解压模型，直接处理本机数据类型，如果您希望在包装器中实现一些实用程序代码以进一步简化CoreML模型的使用。

在我的超级简单的iOS应用程序中，我为Heigth/Weight CoreML模型创建了以下包装，让您直接调用一个只传递一个Float（Heigth）的predict方法，并最终将其从厘米转换为英寸，磅转换为千克：

当然，这再次是非常基本的代码，只是提供了没有严重的错误处理，抛出的方法和强制的可选展开的想法。

### 扩展模型

现在，一旦您在CoreML中构建并测试了这个超级简单的模型，是时候做一些更复杂的事情了。

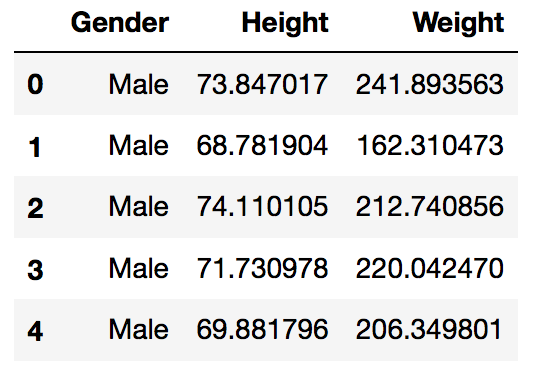
我们将扩展这个体重/身高模型，添加一些参数来区分我们基于性别的预测。

这里的目标不是编写一个关于深度学习/人工智能的教程，而是一步一步地记录Keras（TensorFlow）和CoreML是如何集成到一个通用的移动“人工智能优先”应用程序中的。

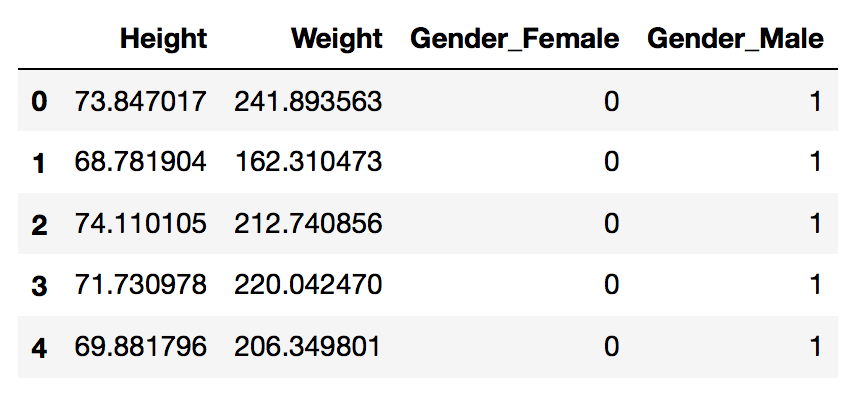
如果您已经克隆了上面的Git repo，那么只需在Jupyter中打开Notebook extendModel.ipynb并执行所有单元格即可生成和导出新的扩展模型。

否则，如下所示创建Python笔记本，导入一些Python实用程序（如Pandas）并再次导入HeightWeight统计文件

现在，如果仔细看一下csv，我们可以看到，我们也有一个列，描述了每一个体重-身高条目的性别。如果执行命令df.head（），我们将看到csv数据的头部，如下所示：



现在为了在Keras/TensorFlow中使用这个性别数据，我们需要将这些性别字符串转换成一些数值。我们在这里使用Python Pandas包来获得一个新的全数字数据集。性别字符串以这种方式转换为对应的女性和男性列中的0或1个数值。



现在我们要重新创建X和y真值，其中y真值仍然是权重数组，但X现在是高度、女性和男性值的三维数组。

与之前完全一样，我们现在将使用TensorFlow后端来创建、编译和训练新模型。这里唯一的区别是，我们将告诉凯拉斯，我们的模型将有一个输入形状3：身高，女性，男性

再次提醒，这不是Keras/Tensorflow教程。我们刚刚在这里训练了所有可用的身高/体重统计数据集。这显然不是最佳实践，但对我来说，这只是使用较少的Python代码行创建模型的最佳方法！

现在和以前一样，我们将使用CoreML工具导出模型

CoreML模型将作为heightweightdextended\_Model.mlmodel保存在当前文件夹中

现在我们已经生成并导出了这个新的CoreML模型，我们将在先前的XCode 9项目中导入它，取代旧的模型，我们将更改UI，以便向用户询问性别。

The CoreML Swift Wrapper we created before will turn back very useful this time to simply adapt to the new 3 dimension MLMultiArray that is used as new input value for our new prediction function.

A simple Swift Enum as in the code below will help indeed a lot to simply adapt the ViewControllers of our App to the new changes.

Have fun now with your Height prediction and please implement and share a real CoreML scenario ;-)