TOPICS Data Science Tutorial ▼

Visual Studio Code中的数据科学

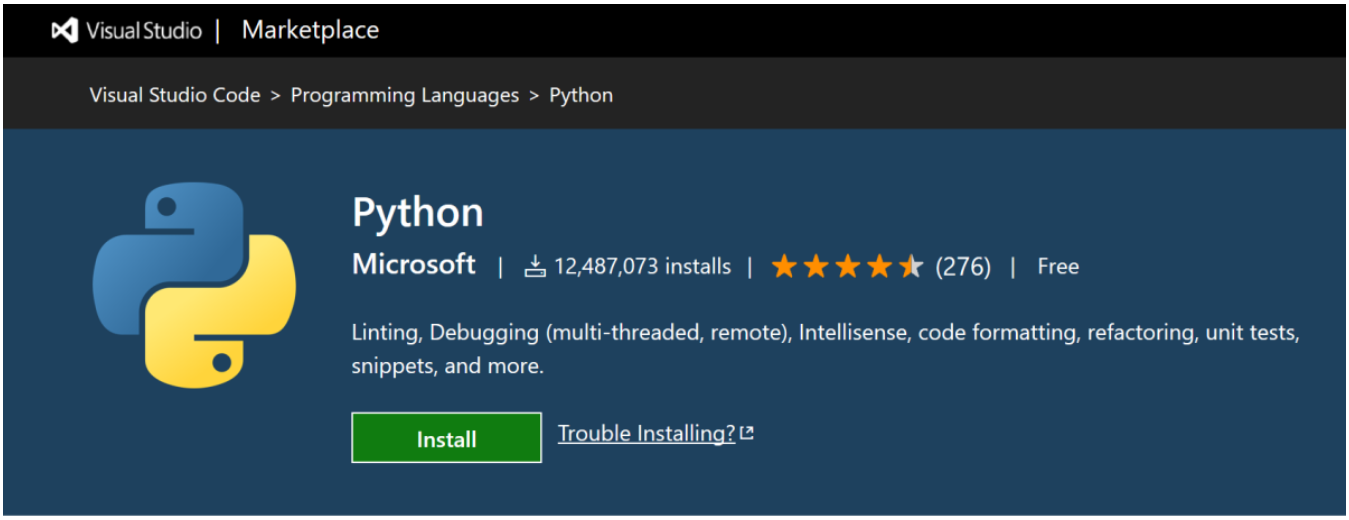
(https://github.com/Microsoft/vscode-docs/blob/master/docs/python/data-science-tutorial.md)

本教程演示了如何使用Visual Studio Code和Microsoft Python扩展以及常见的数据科学库来探索基本的数据科学场景。具体来说，您将使用来自泰坦尼克号的乘客数据，学习如何设置数据科学环境，导入和清除数据，创建机器学习模型以预测泰坦尼克号的生存时间以及评估生成模型的准确性。

先决条件

要完成本教程，需要进行以下安装。如果还没有它们，请在开始安装之前安装它们。

- Visual Studio程式码 (https://code.visualstudio.com/)
- Visual Studio Marketplace中VS Code (https://marketplace.visualstudio.com/items?itemName=ms-python.python)的Python扩展 (https://marketplace.visualstudio.com/items?itemName=ms-python.python)。有关安装扩展的更多详细信息，请参阅Extension Marketplace (/docs/editor/extension-gallery)。Python扩展名为**Python**，由Microsoft发布。



(https://marketplace.visualstudio.com/items?itemName=ms-python.python)

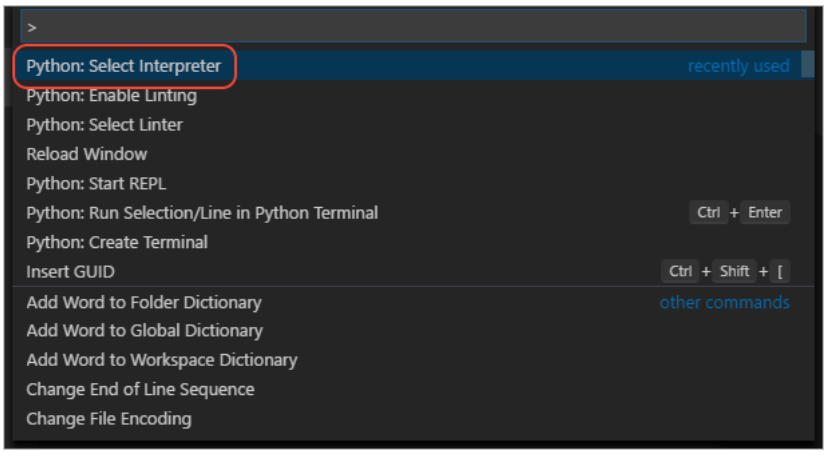
- 带有Python 3.7的Miniconda (https://docs.conda.io/en/latest/miniconda.html)

> **注意**：如果您已经安装了完整的Anaconda发行版，则无需安装Miniconda。另外，如果您不想使用Anaconda或Miniconda，则可以创建一个Python虚拟环境，并使用pip安装本教程所需的软件包。如果走这条路，则需要安装以下软件包：pandas，jupyter，seaborn，scikit-learn，keras和tensorflow。

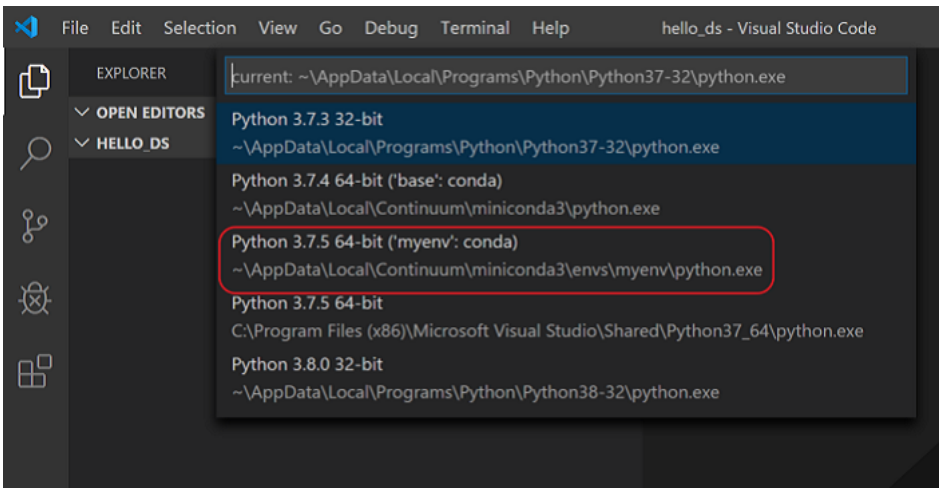
设置数据科学环境

Visual Studio Code和Python扩展为数据科学场景提供了出色的编辑器。通过对Jupyter笔记本电脑和Anaconda的本地支持，可以轻松上手。在本部分中，您将为教程创建一个工作区，使用该教程所需的数据科学模块创建一个Anaconda环境，并创建一个Jupyter笔记本，用于创建机器学习模型。

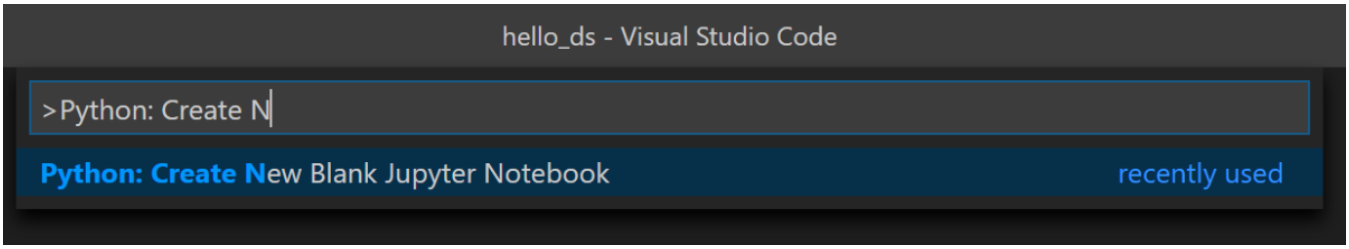
1. 首先为数据科学教程创建Anaconda环境。打开Anaconda命令提示符并运行 `conda create -n myenv python=3.7 pandas jupyter seaborn scikit-learn keras tensorflow` 以创建一个名为**myenv**的环境。有关创建和管理Anaconda环境的其他信息，请参见Anaconda文档 (https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html)。
2. 接下来，在方便的位置创建一个文件夹以用作本教程的VS Code工作区，并将其命名为 `hello_ds` 。
3. 通过运行VS Code并使用“ **文件**”>“ **打开文件夹**”命令，在VS Code中打开项目文件**夹**。
4. VS Code启动后，打开“命令面板”（“ **视图**”>“ **命令面板**”或 Ctrl + Shift + P ）。然后选择**Python：Select Interpreter**命令：



5. “ **Python：选择解释器**”命令显示VS Code能够自动定位的可用解释器的列表（您的列表将与以下所示的列表有所不同；如果看不到所需的解释器，请参阅“ 配置Python环境” (/docs/python/environments)）。从列表中，选择您创建的Anaconda环境，其中应包含文本“ **myenv**”：**conda**。

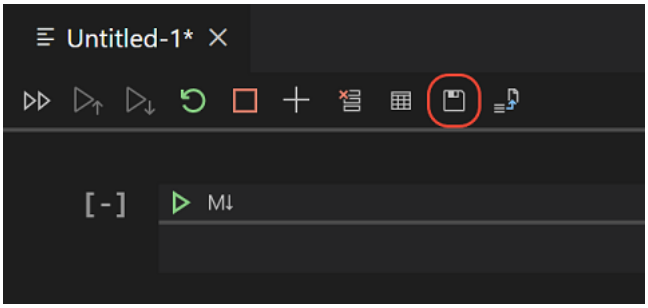


6. 通过环境和VS Code设置，最后一步是创建将用于本教程的Jupyter笔记本。打开命令面板（ Ctrl + Shift + P ）并选择**Python：创建新的空白Jupyter Notebook。**

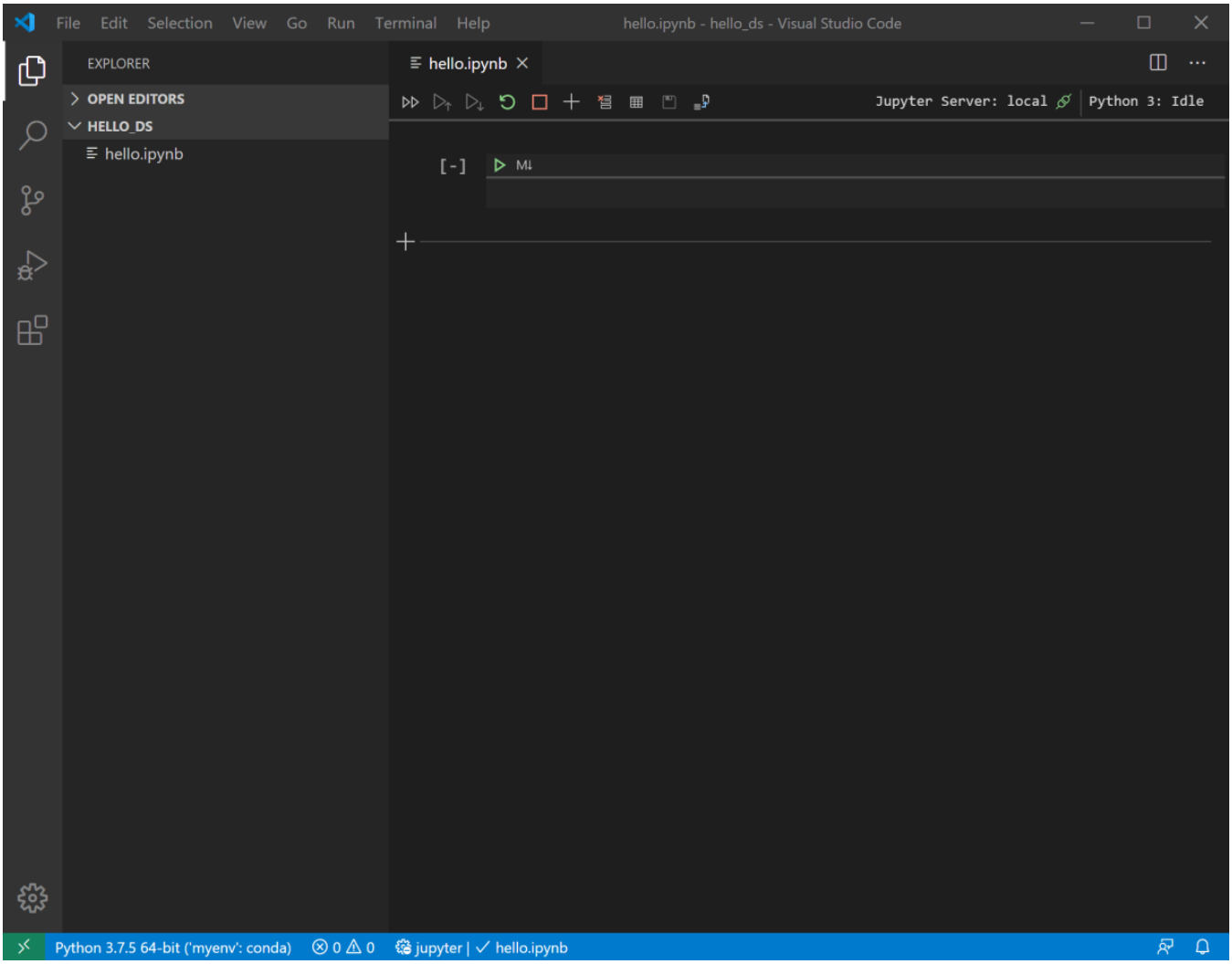


> **注意**：或者，从VS Code File Explorer中，您可以使用New File图标创建名为的Notebook文件 `hello.ipynb` 。

7. 使用笔记本主工具栏上的“保存”图标将笔记本与文件名一起保存 `hello` 。



8. 创建文件后，您应该在本机笔记本编辑器中看到打开的Jupyter笔记本 (https://jupyter.org/)。有关本机Jupyter笔记本支持的其他信息，请参阅文档的此部分 (/docs/python/jupyter-support)。



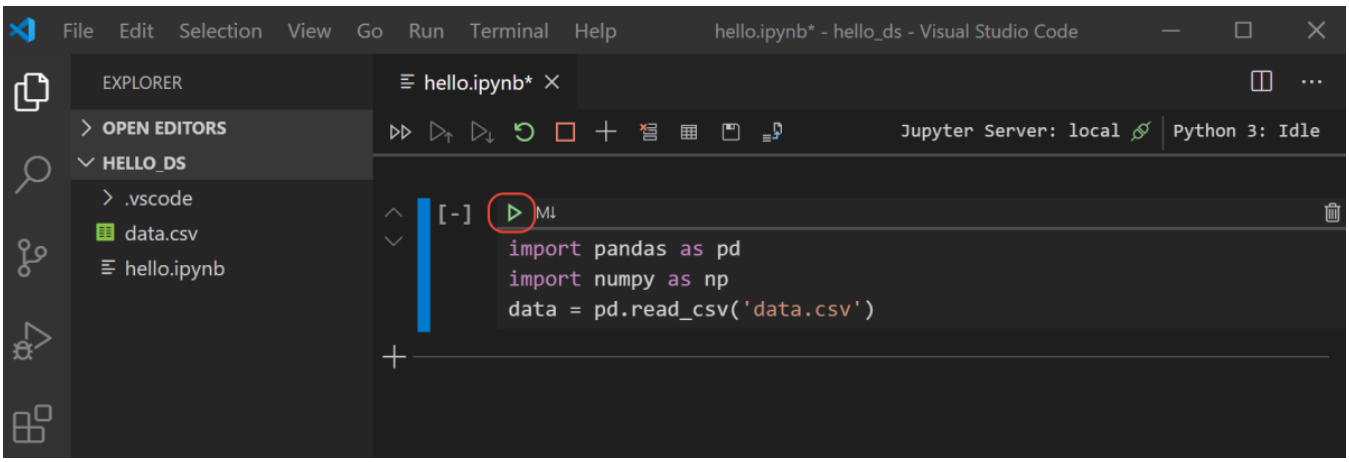
准备数据

本教程使用OpenML.org (https://www.openml.org/d/40945)上的Titanic数据集 (http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.html)，该数据集 (http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.html)可从范德比尔特大学的生物统计学系（http://biostat.mc.vanderbilt.edu/DataSets）获得 (http://biostat.mc.vanderbilt.edu/DataSets)。泰坦尼克号数据可提供有关泰坦尼克号上旅客生存情况的信息，以及有关旅客的年龄和机票等级等特征。使用这些数据，本教程将建立一个模型，用于预测给定的乘客是否会在泰坦尼克号沉没中幸存下来。本节介绍如何在Jupyter笔记本中加载和操作数据。 (https://www.openml.org/d/40945) (http://biostat.mc.vanderbilt.edu/DataSets)

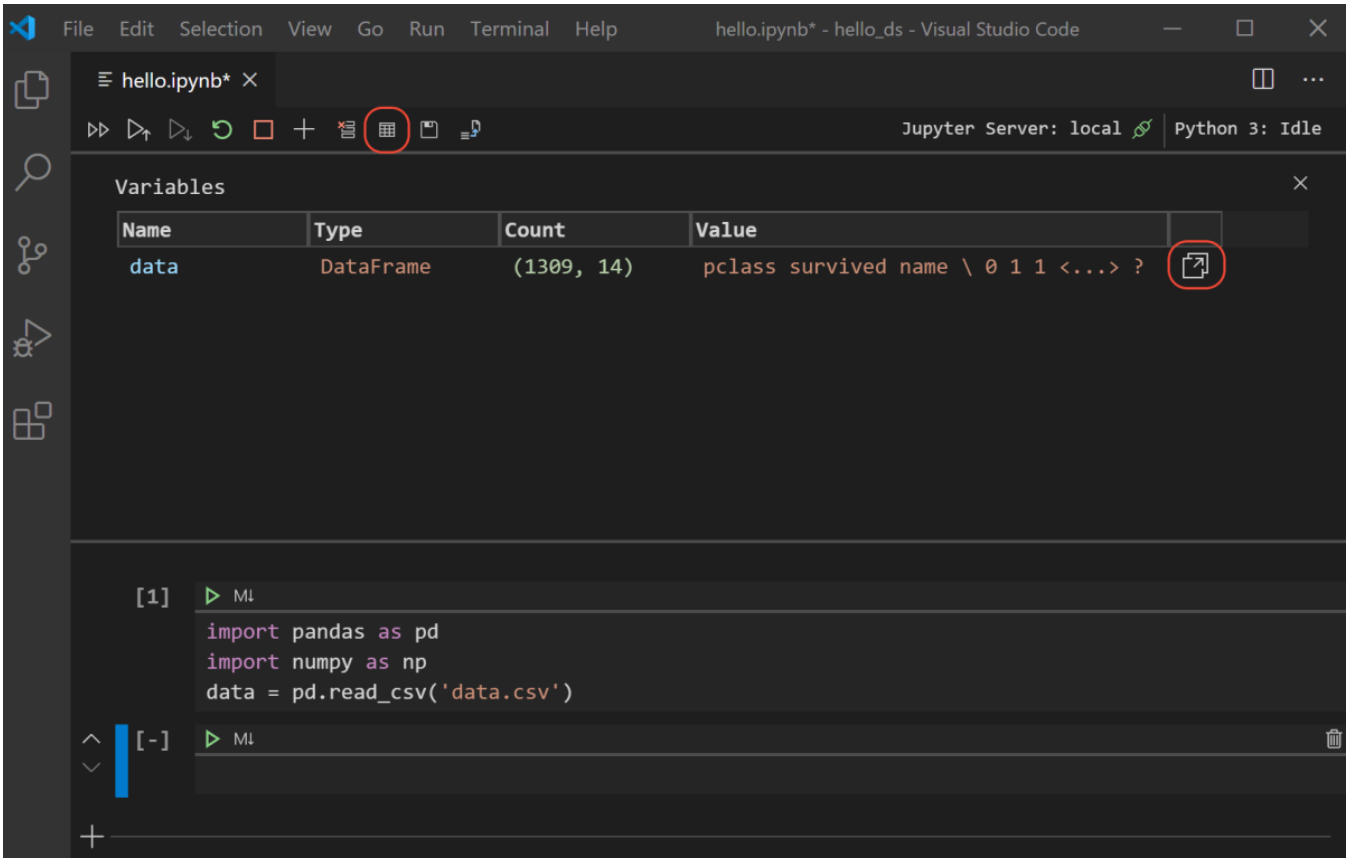
1. 首先，从OpenML.org (https://www.openml.org/d/40945)以名为csv的文件下载Titanic数据 `data.csv` ，并将其保存到 `hello_ds` 上一部分中创建的文件夹中。
2. 在VS Code中，转到**文件** > **打开文件夹**，打开 `hello_ds` 文件夹和Jupyter笔记本（ `hello.ipynb` ）。
3. 在Jupyter笔记本中，首先导入pandas (https://pandas.pydata.org/)和numpy (https://numpy.org/)库，这是两个用于处理数据的常用库，然后将Titanic数据加载到pandas DataFrame中 (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)。为此，将以下代码复制到笔记本的第一个单元格中。有关在VS Code中使用Jupyter Notebook的 (https://code.visualstudio.com/docs/python/jupyter-support)其他指导，请参阅《使用Jupyter Notebooks》(https://code.visualstudio.com/docs/python/jupyter-support)文档。

   ```
   import pandas as pd
   import numpy as np
   data = pd.read_csv('data.csv')
   ```

4. 现在，使用"运行"单元格图标或 Shift + Enter 快捷键运行该单元格。



5. 单元运行完毕后，您可以使用变量资源管理器和数据查看器查看已加载的数据。首先单击笔记本计算机上方工具栏中的图表图标，然后单击 `data` 变量右侧的数据查看器图标。有关数据集的其他信息，请参阅本文档 (http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3info.txt)，了解其构造方式。



然后，您可以使用数据查看器查看，排序和过滤数据行。在检查数据之后，可以对数据的某些方面进行图形化以帮助可视化不同变量之间的关系。

6. 但是，在将数据绘制成图形之前，您需要确保它没有任何问题。如果查看Titanic csv文件，您会注意到一件事，即使用问号（"? "）来指定没有数据的单元格。

   虽然Pandas可以将该值读取到DataFrame中，但Age等列的结果是其数据类型将设置为Object而不是数字数据类型，这对于图形绘制是有问题的。

   可以通过用熊猫能够理解的缺失值替换问号来纠正此问题。将以下代码添加到笔记本的下一个单元格，以numpy NaN (https://docs.scipy.org/doc/numpy/reference/constants.html?highlight=nan#numpy.nan)值替换**age**和**fare**列中的问号。注意，替换值之后，我们还需要更新列的数据类型。 (https://docs.scipy.org/doc/numpy/reference/constants.html?highlight=nan#numpy.nan)

   > **提示**：要添加新的单元格，可以使用现有单元格左下角的插入单元格图标。或者，您也可以使用 Esc 进入命令模式，然后按 B 键。

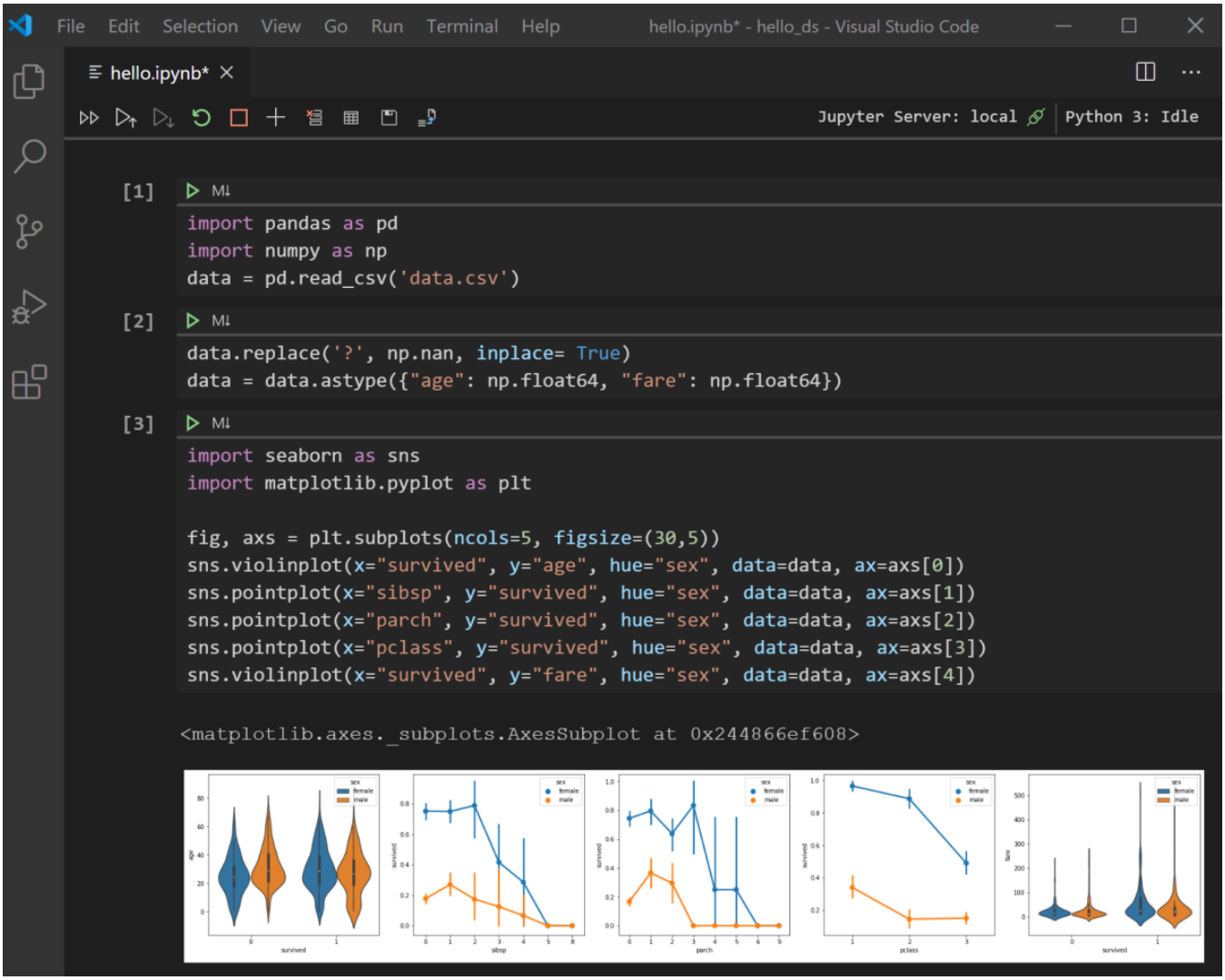
```
data.replace('?', np.nan, inplace= True)
data = data.astype({"age": np.float64, "fare": np.float64})
```

   > **注意**：如果需要查看已用于列的数据类型，则可以使用DataFrame dtypes (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dtypes.html#pandas.DataFrame.dtypes)属性。

7. 现在数据状态良好，您可以使用seaborn (https://seaborn.pydata.org/)和matplotlib (https://matplotlib.org)查看数据集的某些列与生存能力的关系。将以下代码添加到笔记本中的下一个单元格，然后运行它以查看生成的图。

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, axs = plt.subplots(ncols=5, figsize=(30,5))
sns.violinplot(x="survived", y="age", hue="sex", data=data, ax=axs[0])
sns.pointplot(x="sibsp", y="survived", hue="sex", data=data, ax=axs[1])
sns.pointplot(x="parch", y="survived", hue="sex", data=data, ax=axs[2])
sns.pointplot(x="pclass", y="survived", hue="sex", data=data, ax=axs[3])
sns.violinplot(x="survived", y="fare", hue="sex", data=data, ax=axs[4])
```



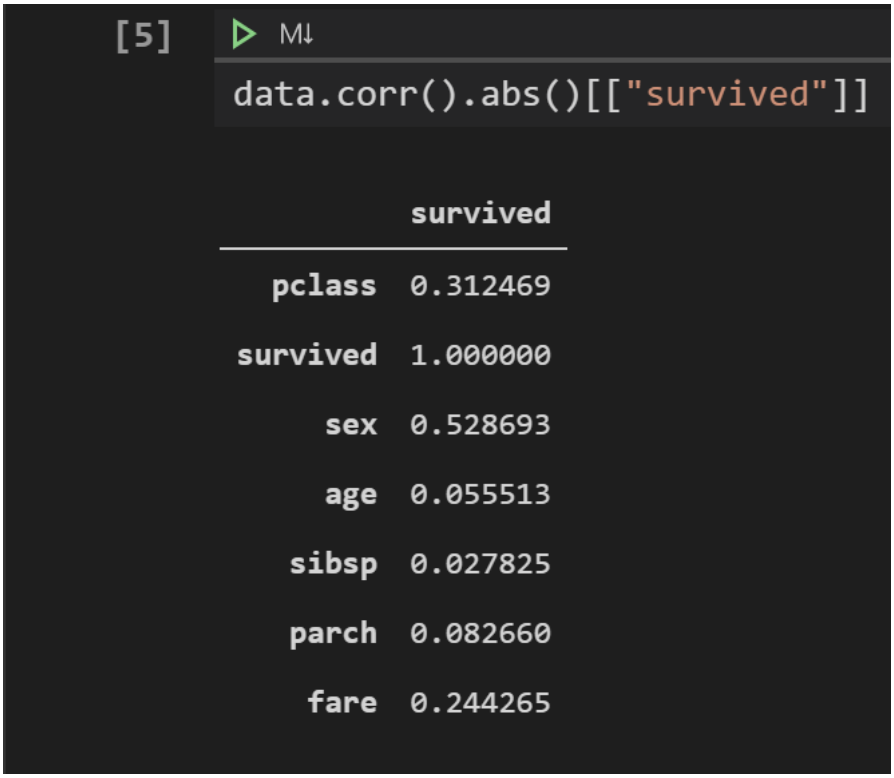
注意：要更好地查看图形的详细信息，可以将鼠标悬停在图形的左上角并单击出现的按钮，以在图查看器中将其打开。

8. 这些图有助于查看生存率与数据输入变量之间的一些关系，但是也可以使用**熊猫**来计算相关性。为此，所有用于关联计算的变量必须是数字，并且当前性别存储为字符串。要将这些字符串值转换为整数，请添加并运行以下代码。

```
data.replace({'male': 1, 'female': 0}, inplace=True)
```

9. 现在，您可以分析所有输入变量之间的相关性，以识别将成为机器学习模型最佳输入的功能。值越接近1，则值与结果之间的相关性越高。使用以下代码关联所有变量和生存之间的关系。

```
data.corr().abs()[["survived"]]
```

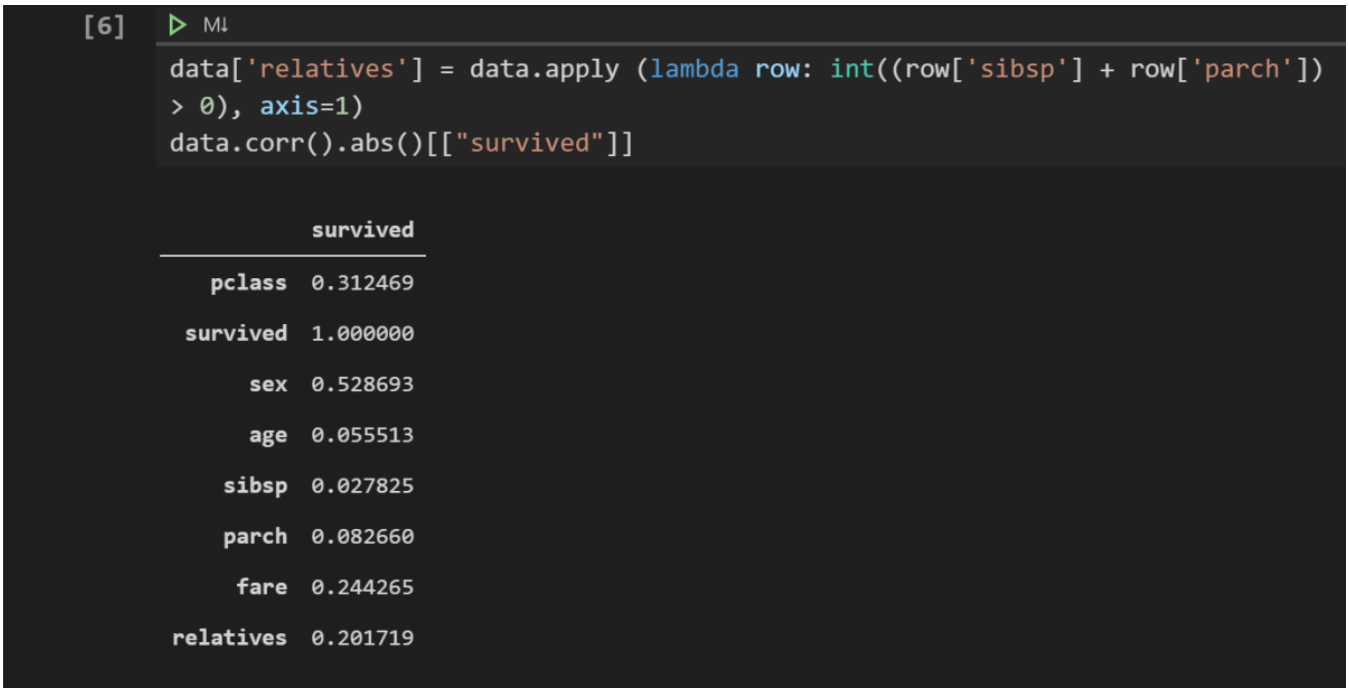


10. 查看相关性结果，您会注意到诸如性别之类的一些变量与生存率具有相当高的相关性，而诸如亲戚（同胞=兄弟姐妹或配偶，parch =父母或子女）之类的其他变量似乎几乎没有相关性。

    让我们假设**sibsp**和**烘干**在它们如何影响生存性是相关的，并把它们组到一个名为"亲戚"新栏看到他们的组合是否具有较高的相关性来生存。要做到这一点，你会检查是否对于给定的乘客数量**sibsp**和**烘干**大于0，如果是这样，你就可以说，他们在船上的亲戚。

    使用以下代码在名为的数据集中创建新变量和列， `relatives` 然后再次检查相关性。

```
data['relatives'] = data.apply (lambda row: int((row['sibsp'] + row['parch']) > 0), axis=1)
data.corr().abs()[["survived"]]
```



11. 您会注意到，实际上，从一个人是否有亲戚（相对于有多少个亲戚）的角度来看，与生存的相关性更高。有了这些信息，你现在可以从数据集中低值下降**sibsp**和**烘干**列，以及为有任何行**的**NaN值，与可用于训练模型的数据集结束。

```
data = data[['sex', 'pclass','age','relatives','fare','survived']].dropna()
```

> **注**：尽管年龄具有较低的直接相关性，但之所以保留，是因为与其他投入物可能仍具有相关性似乎是合理的。

训练和评估模型

准备好数据集后，您现在就可以开始创建模型了。在本节中，您将使用scikit-learn (https://scikit-learn.org/stable/)库（因为它提供了一些有用的辅助函数）来进行数据集的预处理，训练分类模型以确定在Titanic上的生存能力，然后将该模型与测试数据一起使用来确定其准确性。

1. 训练模型的常见第一步是将数据集分为训练和验证数据。这样，您就可以使用一部分数据来训练模型，而使用一部分数据来测试模型。如果您使用所有数据来训练模型，那么您将无法估算该模型相对于尚未看到的数据实际表现如何。scikit-learn库的一个好处是它提供了一种专门用于将数据集拆分为训练和测试数据的方法。

   将具有以下代码的单元格添加并运行到笔记本中以拆分数据。

   ```
   from sklearn.model_selection import train_test_split
   x_train, x_test, y_train, y_test = train_test_split(data[['sex','pclass','age','relatives','fare']], data.survived, test_size=0.2, random
   _state=0)
   ```

2. 接下来，您将对输入进行归一化，以平等对待所有功能。例如，在数据集中，年龄值介于∽0-100之间，而性别仅为1或0。通过对所有变量进行归一化，可以确保值的范围都相同。在新的代码单元格中使用以下代码来缩放输入值。

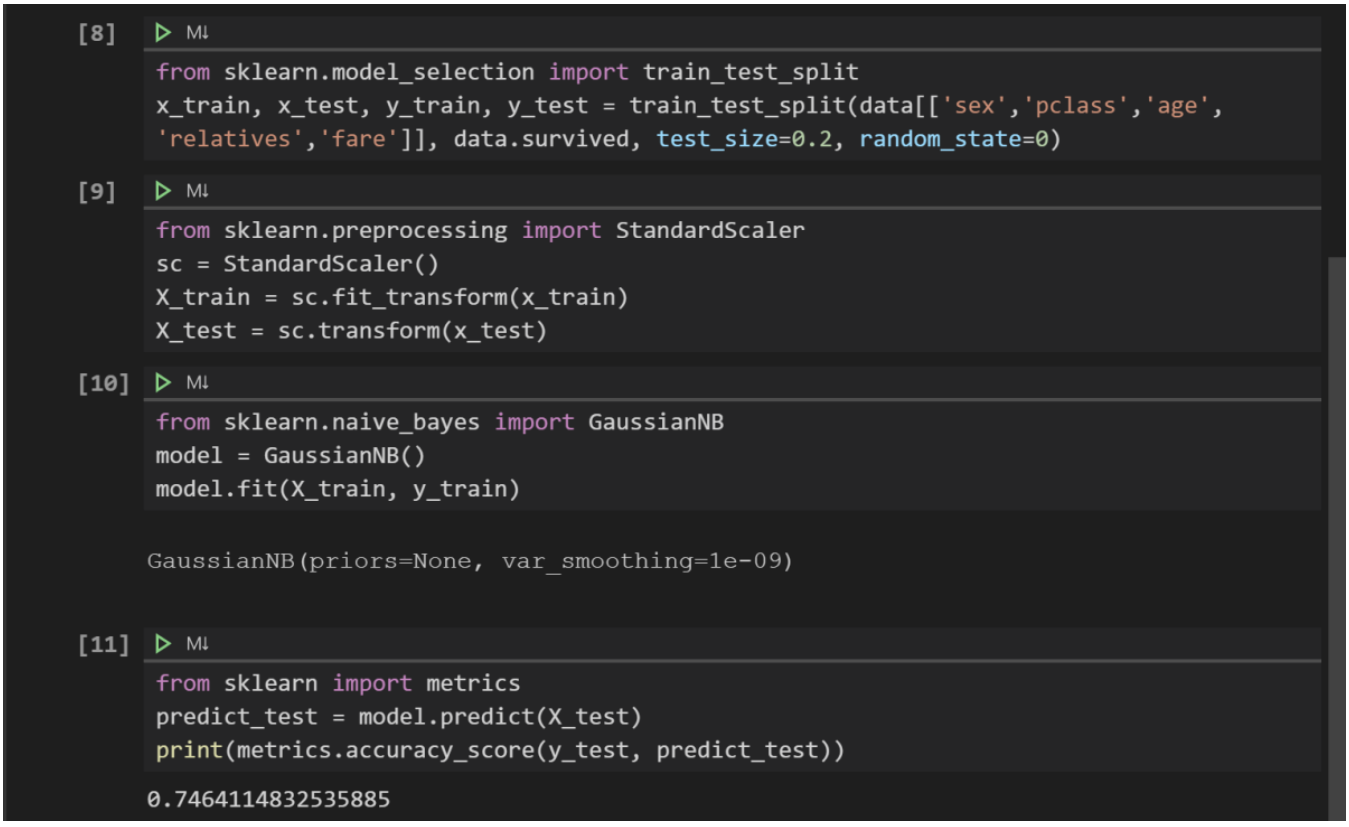
   ```
   from sklearn.preprocessing import StandardScaler
   sc = StandardScaler()
   X_train = sc.fit_transform(x_train)
   X_test = sc.transform(x_test)
   ```

3. 有许多不同的机器学习算法，你可以从中选择数据和scikit学习提供了一些支持模式他们 (https://scikit-learn.org/stable/user_guide.html)，以及一个图表 (https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)来帮助选择一个最适合您的方案。现在，使用朴素贝叶斯算法 (https://scikit-learn.org/stable/modules/naive_bayes.html)，一种用于分类问题的通用算法。添加具有以下代码的单元格以创建和训练算法。

   ```
   from sklearn.naive_bayes import GaussianNB
   model = GaussianNB()
   model.fit(X_train, y_train)
   ```

4. 使用训练有素的模型，您现在可以将其与训练后保留的测试数据集进行对比。添加并运行以下代码，以预测测试数据的结果并计算模型的准确性。

   ```
   from sklearn import metrics
   predict_test = model.predict(X_test)
   print(metrics.accuracy_score(y_test, predict_test))
   ```



查看测试数据的结果，您会发现经过训练的算法在估计生存率方面的成功率约为75％。

（可选）使用神经网络提高准确性

神经网络是一种模型，该模型使用权重和激活函数（对人类神经元的各个方面进行建模）来根据提供的输入确定结果。与您之前看过的机器学习算法不同，神经网络是深度学习的一种形式，您无需提前知道理想的算法来解决问题。它可以用于许多不同的场景，分类就是其中之一。在本节中，您将结合使用TensorFlow (https://www.tensorflow.org/)和Keras (https://keras.io/)库来构建神经网络，并探讨其如何处理Titanic数据集。 (https://www.tensorflow.org/)

1. 第一步是导入所需的库并创建模型。在这种情况下，您将使用顺序 (https://keras.io/getting-started/sequential-model-guide/)神经网络，它是一个分层的神经网络，其中有多个层按顺序相互馈送。

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
```

2. 定义模型后，下一步是添加神经网络的各层。现在，让我们保持简单，仅使用三层。添加以下代码以创建神经网络的各层。

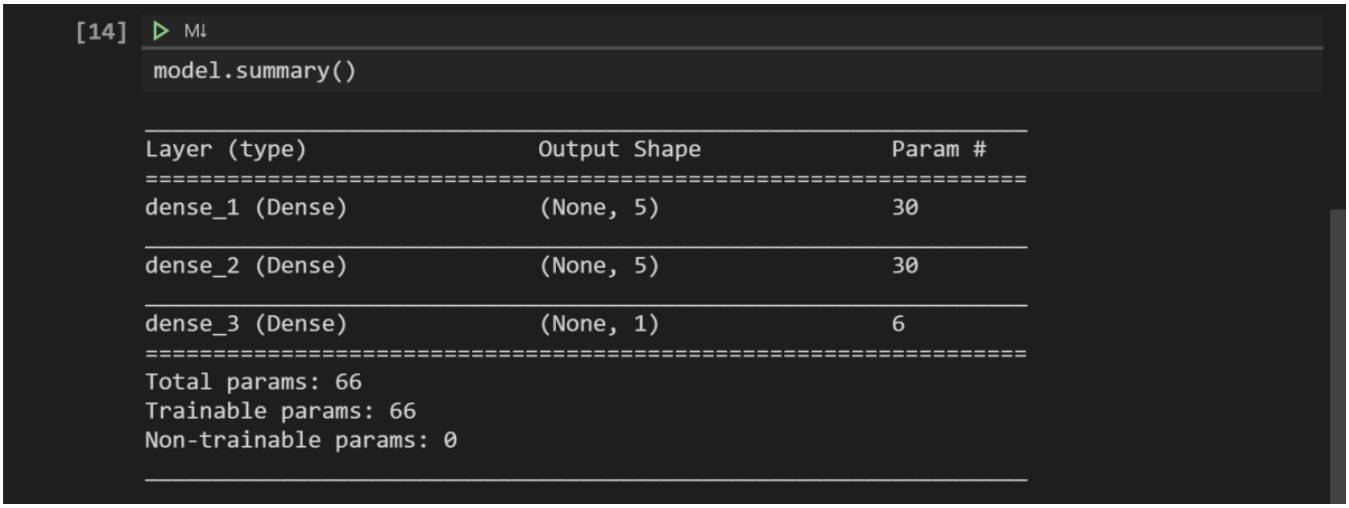
```
model.add(Dense(5, kernel_initializer = 'uniform', activation = 'relu', input_dim = 5))
model.add(Dense(5, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

- 由于您有5个输入：性别，pclass，年龄，亲戚和票价，因此第一层的尺寸将设置为5。
- 最后一层必须输出1，因为您需要一维输出来指示乘客是否可以幸存。
- 为了简单起见，中间层保持为5，尽管该值可能有所不同。

整流的线性单位（relu）激活函数用作前两层的良好通用激活函数，而最后一层需要S型激活函数，因为您想要的输出（乘客是否幸存）需要在0-1（乘客幸存的可能性）范围内缩放。

您还可以查看使用以下代码构建的模型的摘要：

```
model.summary()
```



3. 创建模型后，需要对其进行编译。在此过程中，您需要定义将使用哪种类型的优化器，如何计算损耗以及应针对哪种度量进行优化。添加以下代码以构建和训练模型。您会注意到训练后的准确性约为80％。

> **注意**：根据您的计算机，运行此步骤可能需要几秒钟到几分钟的时间。

```
model.compile(optimizer="adam", loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=32, epochs=50)
```

```
[15] ▷ M↓
model.compile(optimizer="adam", loss='binary_crossentropy', metrics=
['accuracy'])
model.fit(X_train, y_train, batch_size=32, epochs=50)

Epoch 40/50
836/836 [==============================] - 0s 54us/step - loss: 0.4848 - acc: 0.7883
Epoch 41/50
836/836 [==============================] - 0s 53us/step - loss: 0.4807 - acc: 0.7883
Epoch 42/50
836/836 [==============================] - 0s 55us/step - loss: 0.4764 - acc: 0.7859
Epoch 43/50
836/836 [==============================] - 0s 60us/step - loss: 0.4727 - acc: 0.7871
Epoch 44/50
836/836 [==============================] - 0s 54us/step - loss: 0.4692 - acc: 0.7883
Epoch 45/50
836/836 [==============================] - 0s 68us/step - loss: 0.4667 - acc: 0.7883
Epoch 46/50
836/836 [==============================] - 0s 116us/step - loss: 0.4640 - acc: 0.7895
Epoch 47/50
836/836 [==============================] - 0s 77us/step - loss: 0.4621 - acc: 0.7919
Epoch 48/50
836/836 [==============================] - 0s 56us/step - loss: 0.4604 - acc: 0.7847
Epoch 49/50
836/836 [==============================] - 0s 69us/step - loss: 0.4586 - acc: 0.7871
Epoch 50/50
836/836 [==============================] - 0s 63us/step - loss: 0.4571 - acc: 0.7907

<keras.callbacks.History at 0x24b515dbd88>
```

4. 建立并训练模型后，现在就可以查看模型如何针对测试数据进行测试。

```
y_pred = model.predict_classes(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

```
[16] ▷ M↓
y_pred = model.predict_classes(X_test)
print(metrics.accuracy_score(y_test, y_pred))

0.7990430622009569
```

与培训类似，您会注意到，您在预测乘客的生存率方面能够获得接近80%的准确性。这个结果比之前尝试过的朴素贝叶斯分类器的75%准确性要好。

下一步

现在您已经熟悉了在Visual Studio Code中执行机器学习的基础知识，这里是一些其他Microsoft资源和教程供您参考。

- 在Visual Studio Code (https://youtu.be/FSdIoJdSnig)（视频）中了解有关使用Jupyter Notebook的 (https://youtu.be/FSdIoJdSnig)更多信息。
- 开始使用适用于VS Code的Azure机器学习，以使用Azure的功能 (https://docs.microsoft.com/azure/machine-learning/service/how-to-vscode-tools)来部署和优化模型。
- 查找其他数据以探索Azure开放数据集 (https://azure.microsoft.com/services/open-datasets/)。

该文档对您有帮助吗?

是 **没有**

2020年3月23日