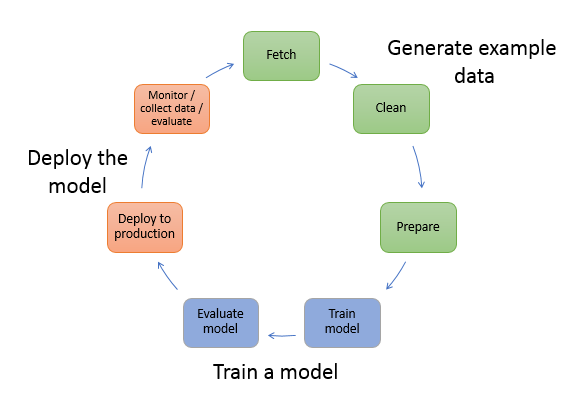
AWS Sagemaker



原理图

在机器学习中，您"教"计算机进行预测或推论。首先，您使用算法和示例数据来训练模型。然后，将模型集成到应用程序中，以实时和大规模生成推论。在生产环境中，模型通常从数百万个示例数据项目中学习，并在数百到 20 毫秒内生成推论。

实验网址：<https://docs.aws.amazon.com/zh_cn/sagemaker/latest/dg/gs-console.html>

介绍如何创建 SageMaker 笔记本实例、使用 Conda 环境打开预配置内核的 Jupyter 笔记本以进行机器学习，以及启动 SageMaker 会话以运行端到端的 ML 周期。您将了解如何将数据集保存到与 SageMaker 会话自动配对的默认 Amazon S3 存储桶，向 Amazon EC2 提交 ML 模型的培训作业，以及如何通过 Amazon EC2 托管或批量推理来部署经过培训的模型以进行预测。

第 1 步：创建 Amazon SageMaker 笔记本实例

选择笔记本实例，然后选择创建笔记本实例。

在 Create notebook instance (创建笔记本实例) 页面上，提供以下信息（如果未提到某个字段，请保留默认值）：

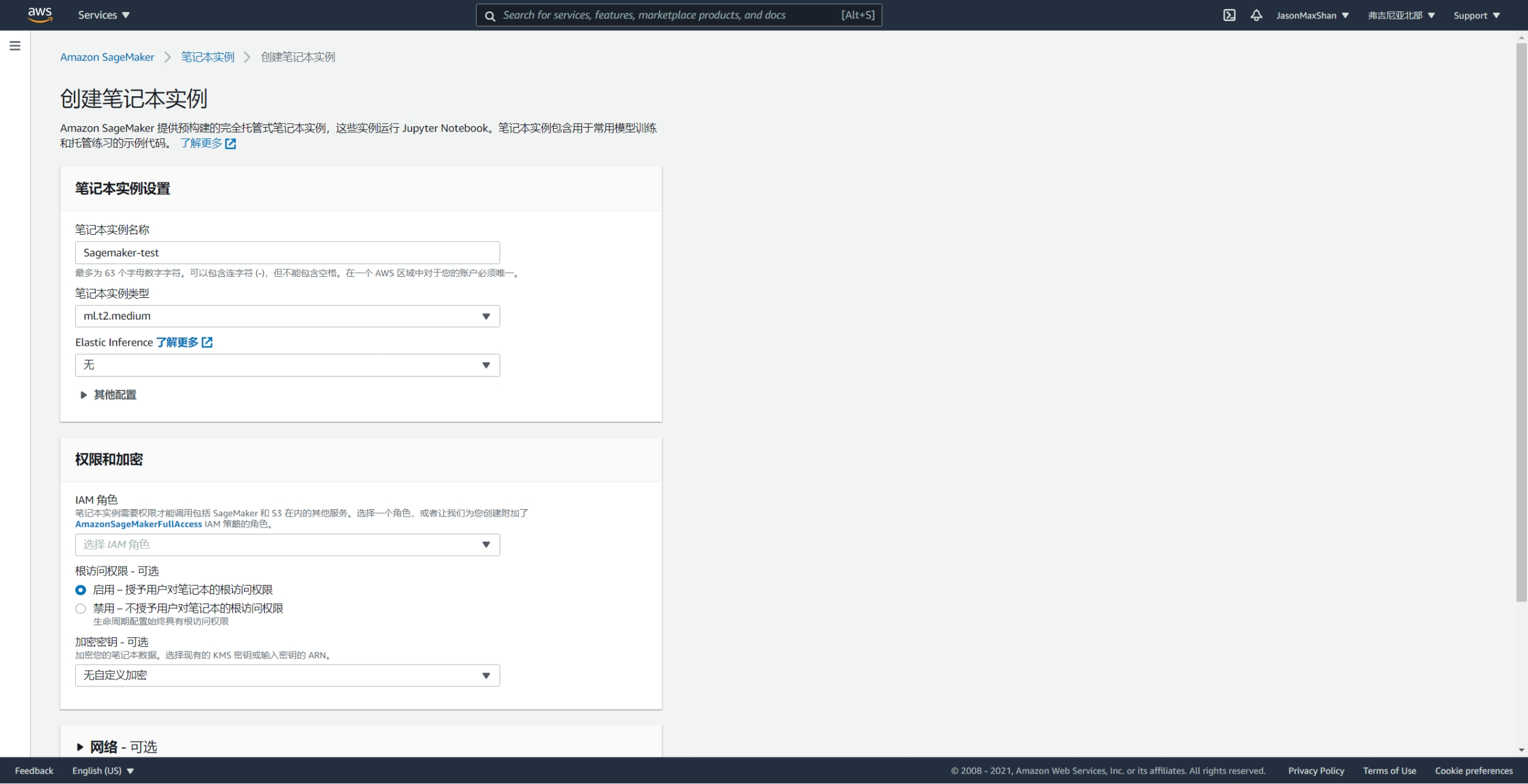
对于 Notebook instance name (笔记本实例名称)，请键入您的笔记本实例的名称。

对于 Instance type (实例类型)，选择 ml.t2.medium。这是笔记本实例支持的成本最低的实例类型，并且足以完成本练习。如果ml.t2.medium实例类型在您的当前AWS区域，选择ml.t3.medium。

适用于IAM 角色中，选择创建新角色，然后选择创建角色。此 IAM 角色自动获取任何具有的 S3 存储桶的访问权限。sagemaker在名称中。它通过获取这些权限AmazonSageMakerFullAccess策略，SageMaker 将该策略附加到角色。

选择创建笔记本实例。

在几分钟内，SageMaker 启动 ML 计算实例（在本例中为笔记本实例），并将 5 GB 的 Amazon EBS 存储卷附加到该实例。笔记本实例有一个预配置的 Jupyter 笔记本服务器 SageMaker 和AWSSDK 库和一组 Anaconda 库。

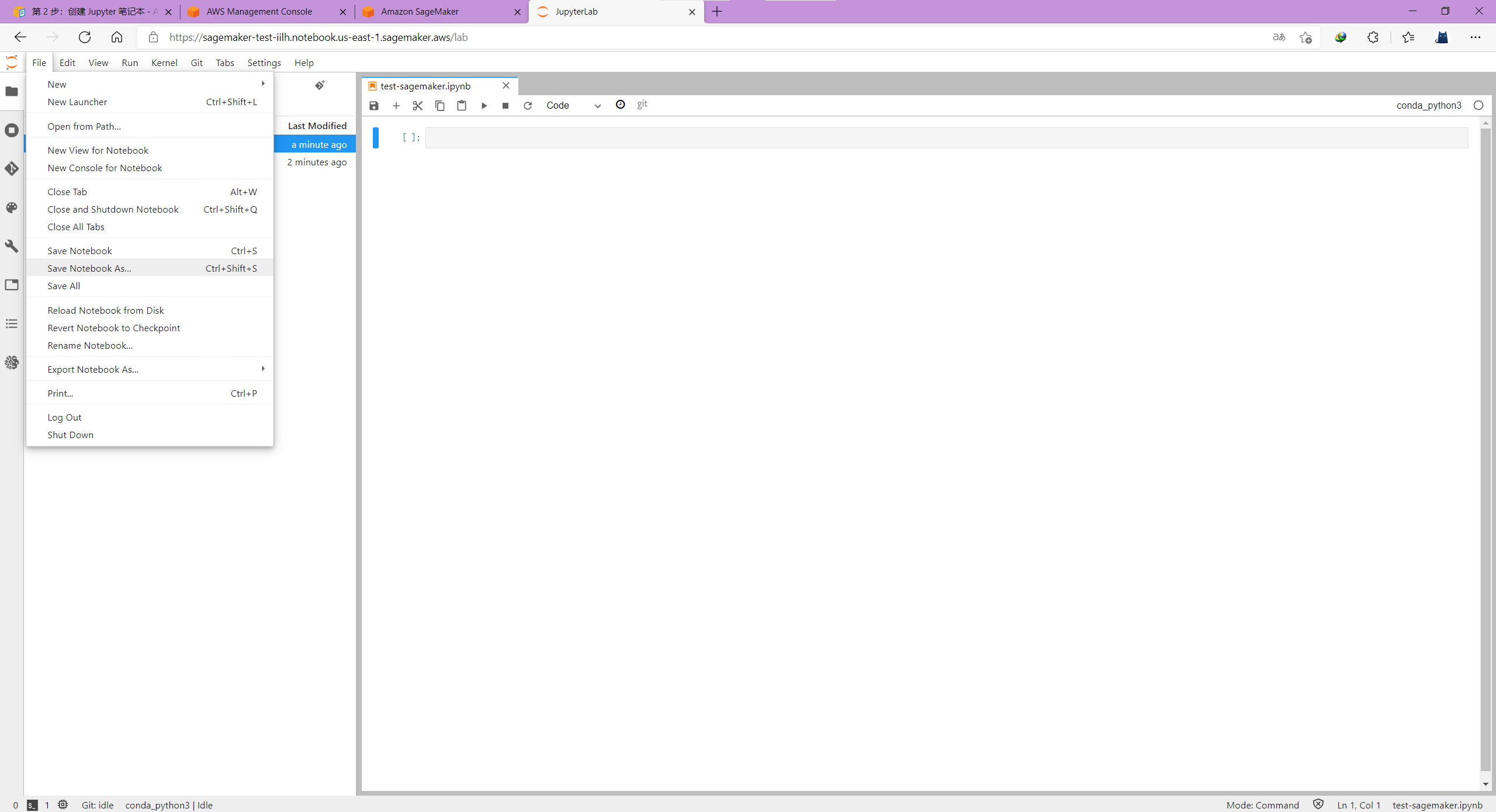


第 2 步：创建 Jupyter 笔记本

创建笔记本，如下所示：

在 JupyterLab 视图中打开了笔记本，请在 JupyterLab 视图中的File (文件)菜单上，选择New，然后选择笔记本。对于 Select Kernel (选择内核)，请选择 conda\_python3。该预装环境包括默认的 Anaconda 安装和 Python 3。

在 “JupyterLab” 视图中，选择File (文件)中，选择将笔记本另存为...，然后重命名笔记本。



第 3 步：下载、浏览和转换数据集

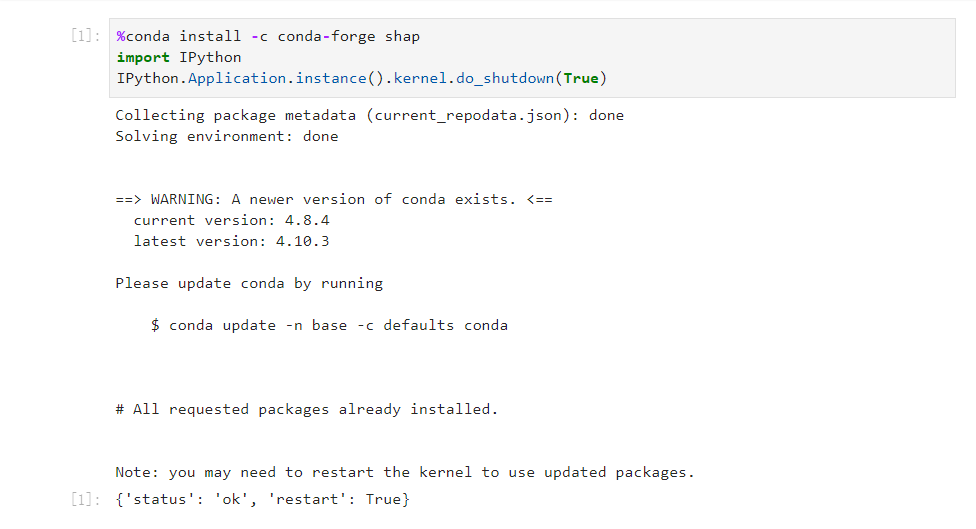
如果当前 Jupyter 内核没有 SHAP 库，请通过运行以下conda命令：

%conda install -c conda-forge shap

如果您使用的是 JupyterLab，则必须在安装和更新完成后手动刷新内核。运行以下 IPython 脚本关闭内核（内核将自动重新启动）：

import IPython

IPython.Application.instance().kernel.do\_shutdown(True)



使用 SHAP 库，导入成人普查数据集，如下所示：

import shap

X, y = shap.datasets.adult()

X\_display, y\_display = shap.datasets.adult(display=True)

feature\_names = list(X.columns)

feature\_names

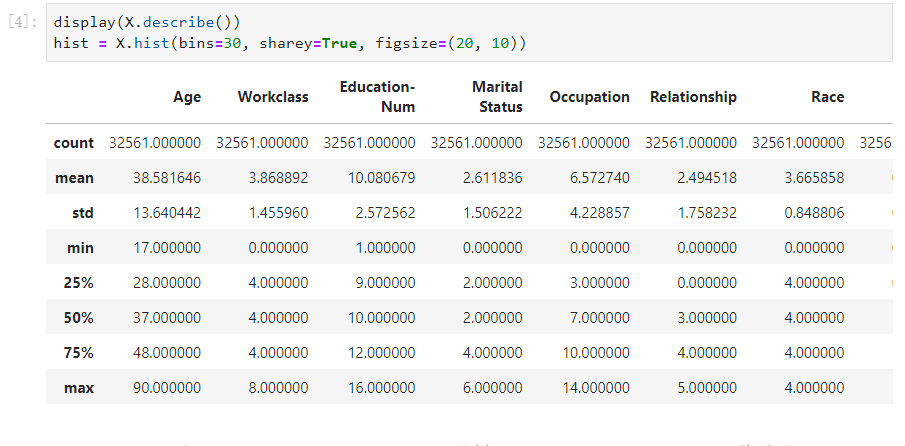


概述数据集

运行以下脚本以显示数字要素的数据集和直方图的统计概述。

display(X.describe())

hist = X.hist(bins=30, sharey=True, figsize=(20, 10))



将数据集拆分为训练数据集、验证数据集和测试数据集

使用 Sklearn 可以将数据集拆分为一个训练集和一个测试集。使用固定随机种子对数据集进行随机排序：训练集合占数据集的 80%，测试集合占 20%。

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

X\_train\_display = X\_display.loc[X\_train.index]

拆分培训集以分离验证集。75% 的培训集将成为最终训练集，其余为验证集。

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.25, random\_state=1)

X\_train\_display = X\_display.loc[X\_train.index]

X\_val\_display = X\_display.loc[X\_val.index]

使用熊猫包，通过将数字要素与真实标注串联起来显式对齐每个数据集。

import pandas as pd

train = pd.concat([pd.Series(y\_train, index=X\_train.index,

name='Income>50K', dtype=int), X\_train], axis=1)

validation = pd.concat([pd.Series(y\_val, index=X\_val.index,

name='Income>50K', dtype=int), X\_val], axis=1)

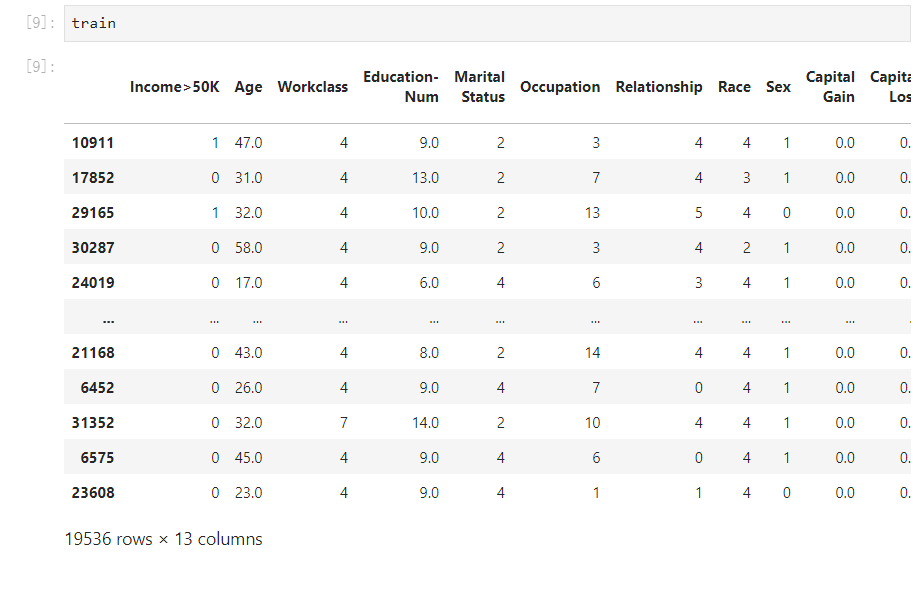
test = pd.concat([pd.Series(y\_test, index=X\_test.index,

name='Income>50K', dtype=int), X\_test], axis=1)

检查数据集是否按预期进行拆分和结构化：

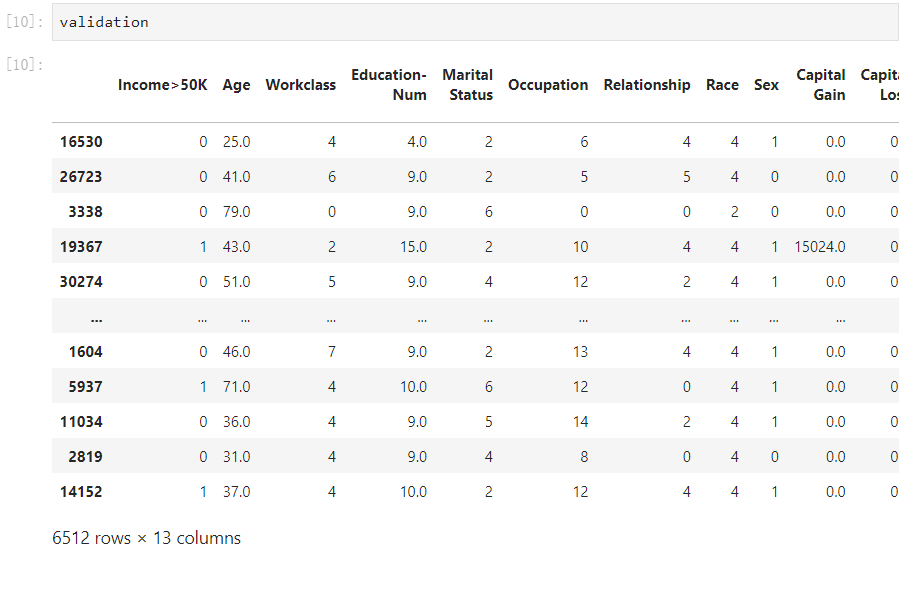
train

训练数据集。

****

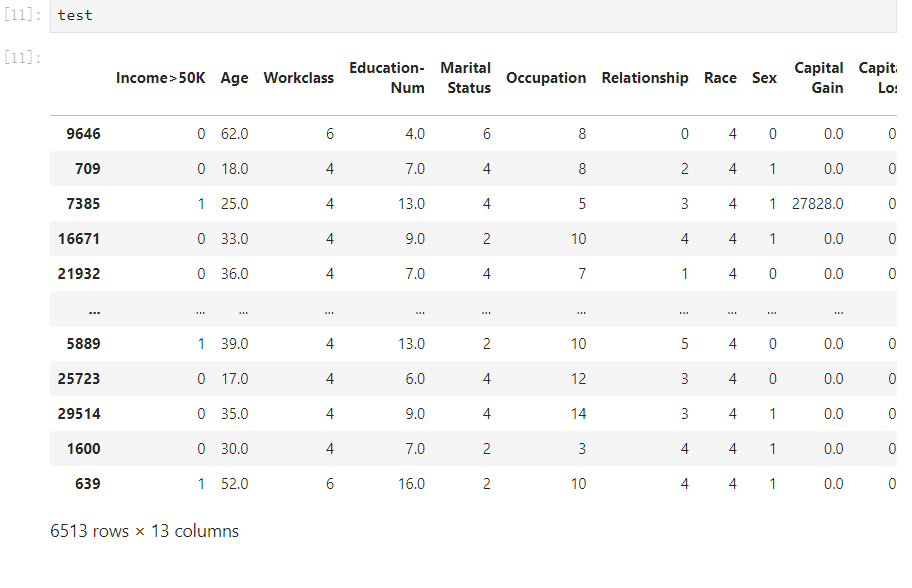
validation

验证数据集。



test

测试数据集。



将列车和验证数据集转换为 CSV 文件

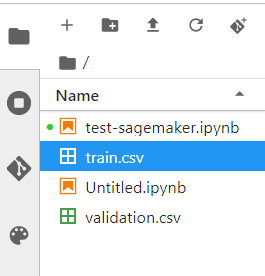
转换train和validation数据框对象转换为 CSV 文件，以匹配 XGBoost 算法的输入文件格式。

# Use 'csv' format to store the data

# The first column is expected to be the output column

train.to\_csv('train.csv', index=False, header=False)

validation.to\_csv('validation.csv', index=False, header=False)



将数据集上传到 Amazon S3

使用 SageMaker 和 Bot3，将培训和验证数据集上传到默认的 Amazon S3 存储桶。S3 存储桶中的数据集将被 Amazon EC2 上计算优化的 SageMaker 实例用于培训。

以下代码为当前 SageMaker 会话设置默认 S3 存储桶 URI，创建一个新的demo-sagemaker-xgboost-adult-income-prediction文件夹，然后将训练和验证数据集上传到data子文件夹。

import sagemaker, boto3, os

bucket = sagemaker.Session().default\_bucket()

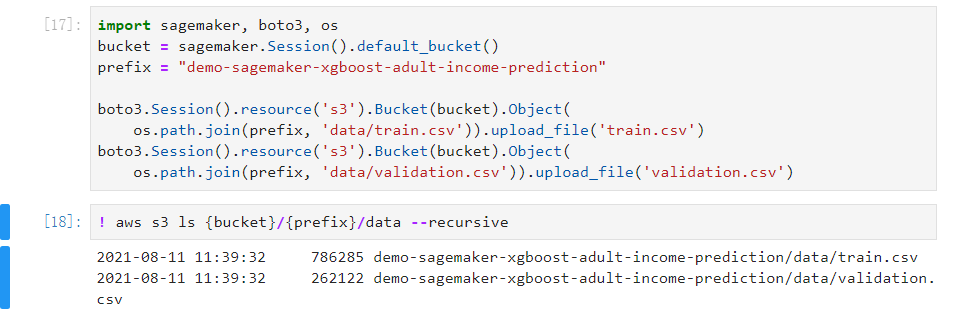
prefix = "demo-sagemaker-xgboost-adult-income-prediction"

boto3.Session().resource('s3').Bucket(bucket).Object(

os.path.join(prefix, 'data/train.csv')).upload\_file('train.csv')

boto3.Session().resource('s3').Bucket(bucket).Object(

os.path.join(prefix, 'data/validation.csv')).upload\_file('validation.csv')



第 4 步：训练模型

选择训练算法

要为数据集选择正确的算法，通常需要评估不同的模型，以找到最适合您的数据的模型。为了简单起见，SageMakerXGBoost 算法内置算法在本教程中使用，而不需要对模型进行预先评估。

创建并运行训练 Job

找出要使用的模型后，开始构建一个 SageMaker 估计器进行培训。本教程使用 XGBoost 内置算法作为 SageMaker 通用估计器。

运行模型训练作业

导入Amazon SageMaker Python 开发工具包，并从当前 SageMaker 会话中检索基本信息开始。

import sagemaker

region = sagemaker.Session().boto\_region\_name

print("AWS Region: {}".format(region))

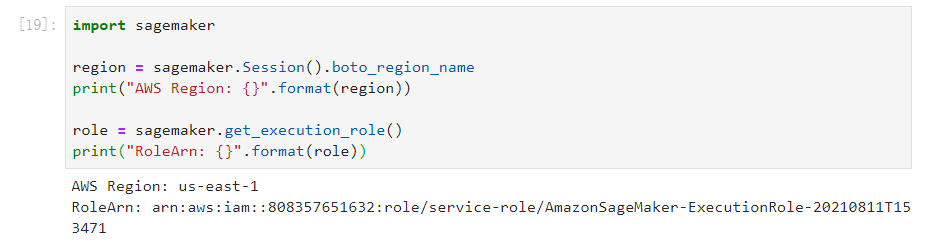
role = sagemaker.get\_execution\_role()

print("RoleArn: {}".format(role))

此操作将返回以下信息：

region— 当前的AWS正在运行 SageMaker 笔记本实例的区域。

role— 笔记本实例所使用的 IAM 角色。



创建一个 XGBoost 估计器

使用sagemaker.estimator.Estimator类。在下面的示例代码中，XGBoost 估计器被命名为xgb\_model。

from sagemaker.debugger import Rule, rule\_configs

from sagemaker.session import TrainingInput

s3\_output\_location='s3://{}/{}/{}'.format(bucket, prefix, 'xgboost\_model')

container=sagemaker.image\_uris.retrieve("xgboost", region, "1.2-1")

print(container)

xgb\_model=sagemaker.estimator.Estimator(

image\_uri=container,

role=role,

instance\_count=1,

instance\_type='ml.m4.xlarge',

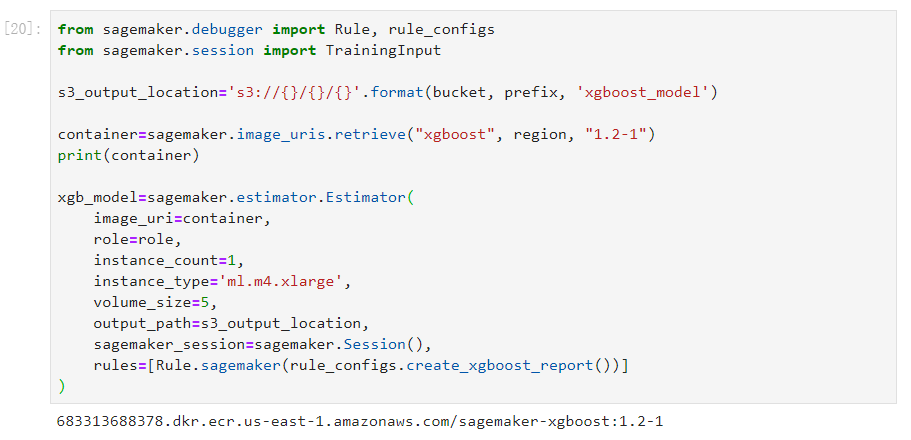
volume\_size=5,

output\_path=s3\_output\_location,

sagemaker\_session=sagemaker.Session(),

rules=[Rule.sagemaker(rule\_configs.create\_xgboost\_report())]

)



设置 XGBoost 算法的超参数

方法是调用set\_hyperparameters方法为评估程序。有关 XGBoost 超参数的完整列表。

xgb\_model.set\_hyperparameters(

max\_depth = 5,

eta = 0.2,

gamma = 4,

min\_child\_weight = 6,

subsample = 0.7,

objective = "binary:logistic",

num\_round = 1000

)

使用TrainingInput类配置用于训练的数据输入流。

以下示例代码显示了如何配置TrainingInput对象以使用您上传到 Amazon S3 的训练和验证数据集将数据集拆分为训练数据集、验证数据集和测试数据集部分。

from sagemaker.session import TrainingInput

train\_input = TrainingInput(

"s3://{}/{}/{}".format(bucket, prefix, "data/train.csv"), content\_type="csv"

)

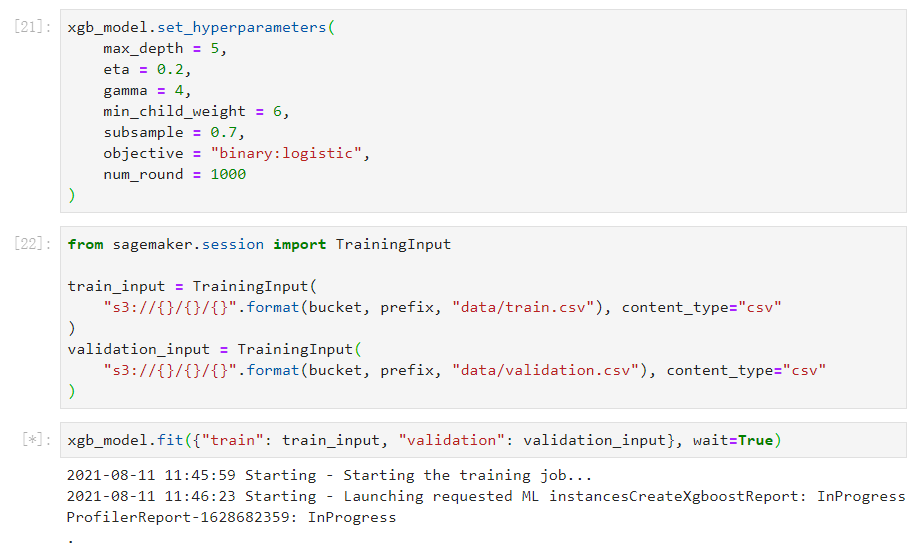
validation\_input = TrainingInput(

"s3://{}/{}/{}".format(bucket, prefix, "data/validation.csv"), content\_type="csv"

)

要启动模型训练，请调用估算器的fit方法与训练和验证数据集一起使用。通过设置wait=True，fit方法显示进度日志，并等到训练完成。

xgb\_model.fit({"train": train\_input, "validation": validation\_input}, wait=True)



有关模型训练的更多信息，请参阅通过 Amazon SageMaker 来训练模型。本教程培训作业最多可能需要 10 分钟。

运行以下代码以指定生成调试器培训报告的 S3 存储桶 URI，并检查报告是否存在。

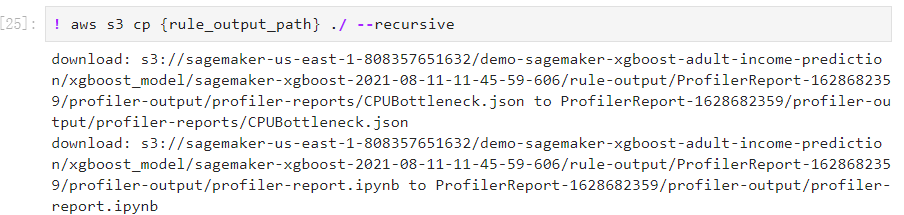
rule\_output\_path = xgb\_model.output\_path + "/" + xgb\_model.latest\_training\_job.name + "/rule-output"

! aws s3 ls {rule\_output\_path} –recursive



将调试器 XGBoost 培训和分析报告下载到当前工作区：

! aws s3 cp {rule\_output\_path} ./ --recursive



运行以下 IPython 脚本以获取 XGBoost 培训报告的文件链接：

from IPython.display import FileLink, FileLinks

display("Click link below to view the XGBoost Training report



以下 IPython 脚本返回调试器分析报告的文件链接，该链接显示 EC2 实例资源利用率的摘要和详细信息、系统瓶颈检测结果和 python 操作分析结果：

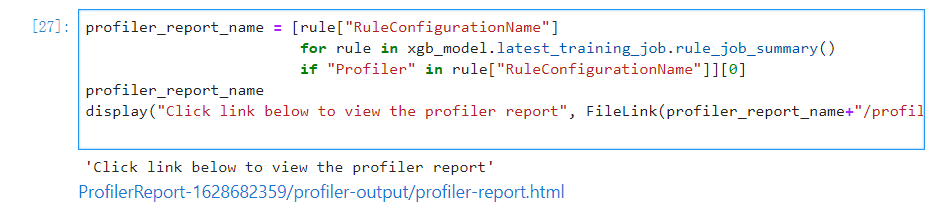
profiler\_report\_name = [rule["RuleConfigurationName"]

for rule in xgb\_model.latest\_training\_job.rule\_job\_summary()

if "Profiler" in rule["RuleConfigurationName"]][0]

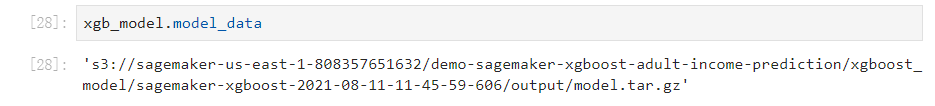
profiler\_report\_name

display("Click link below to view the profiler report", FileLink(profiler\_report\_name+"/profiler-output/profiler-report.html"))



您现在有一个训练有素的 XGBoost 模型。SageMaker 将模型构件存储在 S3 存储桶中。要查找模型工件的位置，请运行以下代码以打印xgb\_model估计器：

xgb\_model.model\_data



第 5 步：将模型部署到 Amazon EC2

要使用 Amazon SageMaker 通过 Amazon EC2 托管模型，请部署您在创建并运行训练 Job通过调用deploy的方法xgb\_model估计。当您调用deploy方法，您必须指定要用于托管终端节点的 EC2 ML 实例的数量和类型。

import sagemaker

from sagemaker.serializers import CSVSerializer

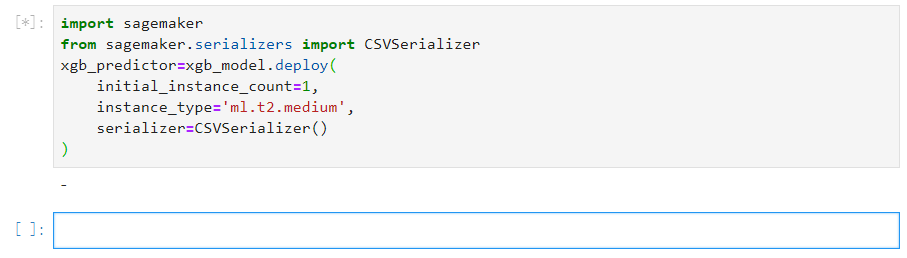
xgb\_predictor=xgb\_model.deploy(

initial\_instance\_count=1,

instance\_type='ml.t2.medium',

serializer=CSVSerializer()

)



第 6 步：评估模型

评估部署到 SageMaker 托管服务的模型

要评估模型并在生产中使用它，请使用测试数据集调用该终端节点，并检查您获取的推理是否返回要实现的目标精度。

评估模型

设置以下函数来预测测试集的每一行。在以下示例代码中，rows参数是指定一次要预测的行数。您可以更改它的值，以执行充分利用实例硬件资源的批量推理。

import numpy as np

def predict(data, rows=1000):

split\_array = np.array\_split(data, int(data.shape[0] / float(rows) + 1))

predictions = ''

for array in split\_array:

predictions = ','.join([predictions, xgb\_predictor.predict(array).decode('utf-8')])

return np.fromstring(predictions[1:], sep=',')

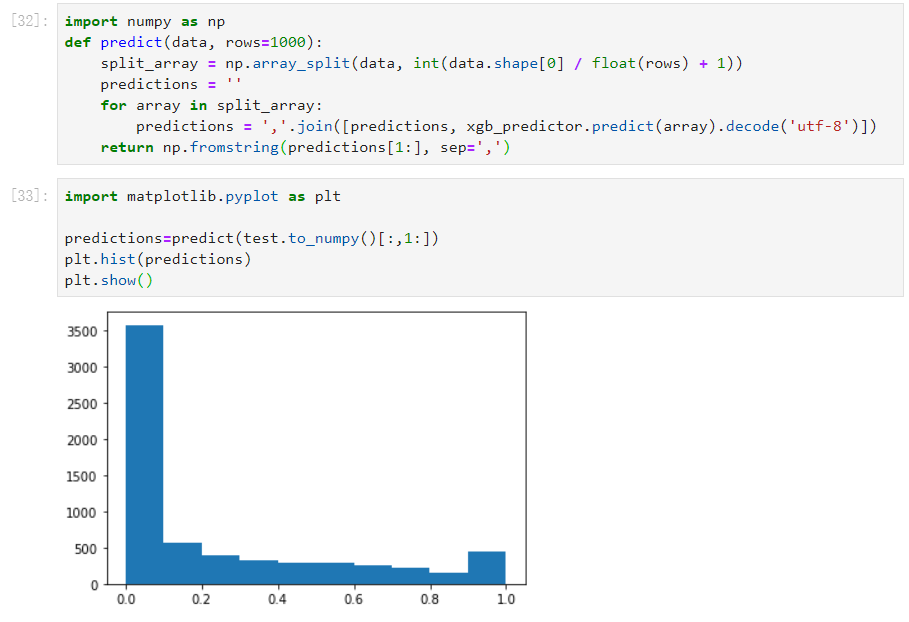
运行以下代码对测试数据集进行预测并绘制直方图。您只需获取测试数据集的要素列，不包括实际值的第 0 列。

import matplotlib.pyplot as plt

predictions=predict(test.to\_numpy()[:,1:])

plt.hist(predictions)

plt.show()



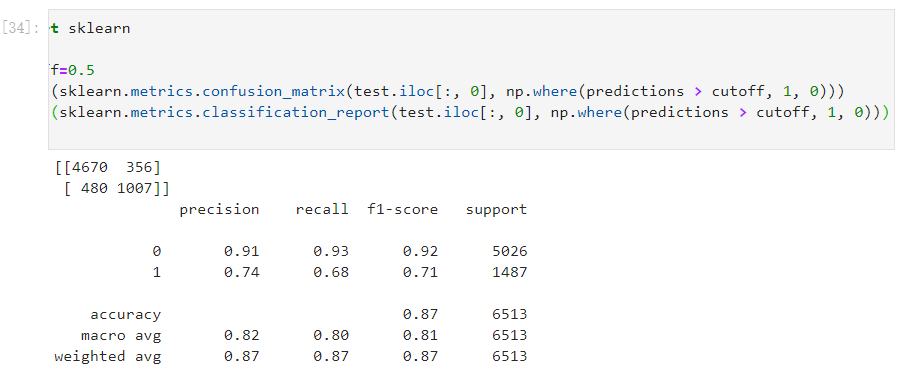
预测值为浮点型。确定True或者False基于浮点值，则需要设置一个截止值。如以下示例代码所示，使用 Scikit-Learning 库返回截止值为 0.5 的输出混淆度量和分类报告。

import sklearn

cutoff=0.5

print(sklearn.metrics.confusion\_matrix(test.iloc[:, 0], np.where(predictions > cutoff, 1, 0)))

print(sklearn.metrics.classification\_report(test.iloc[:, 0], np.where(predictions > cutoff, 1, 0)))



要使用给定的测试集找到最佳截止值，请计算 Logistic 回归的对数损失函数。对数丢失函数被定义为返回地面真相标签的预测概率的 Logistic 模型的负对数似然。以下示例代码以数字和迭代方式计算对数丢失值（-(y\*log(p)+(1-y)log(1-p)），其中，y是真正的标签，p是相应检验样本的概率估计值。它返回对数丢失与截止图。

import matplotlib.pyplot as plt

cutoffs = np.arange(0.01, 1, 0.01)

log\_loss = []

for c in cutoffs:

log\_loss.append(

sklearn.metrics.log\_loss(test.iloc[:, 0], np.where(predictions > c, 1, 0))

)

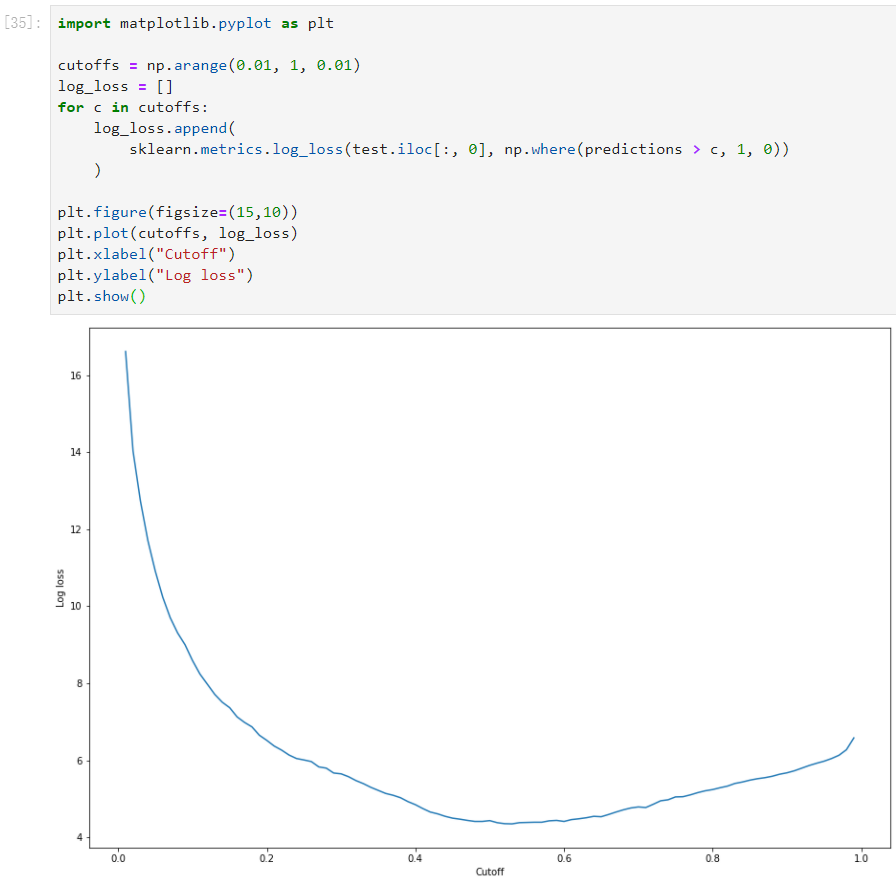
plt.figure(figsize=(15,10))

plt.plot(cutoffs, log\_loss)

plt.xlabel("Cutoff")

plt.ylabel("Log loss")

plt.show()



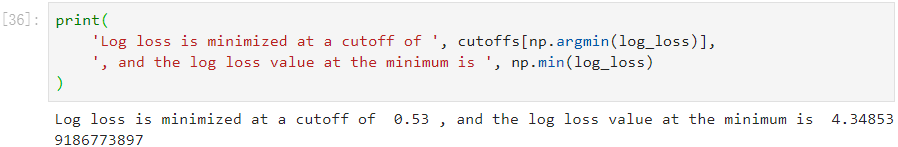
使用 NumPy 查找误差曲线的最小点argmin和min函数：

print(

'Log loss is minimized at a cutoff of ', cutoffs[np.argmin(log\_loss)],

', and the log loss value at the minimum is ', np.min(log\_loss)

)



您可以估计成本函数作为替代方法，而不是计算和最小化日志丢失函数。例如，如果要训练模型以对业务问题（如客户流失预测问题）执行二进制分类，则可以为混淆矩阵元素设置权重，并相应地计算成本函数。

**现在，您在 SageMaker 中训练、部署并评估了第一个模型。**