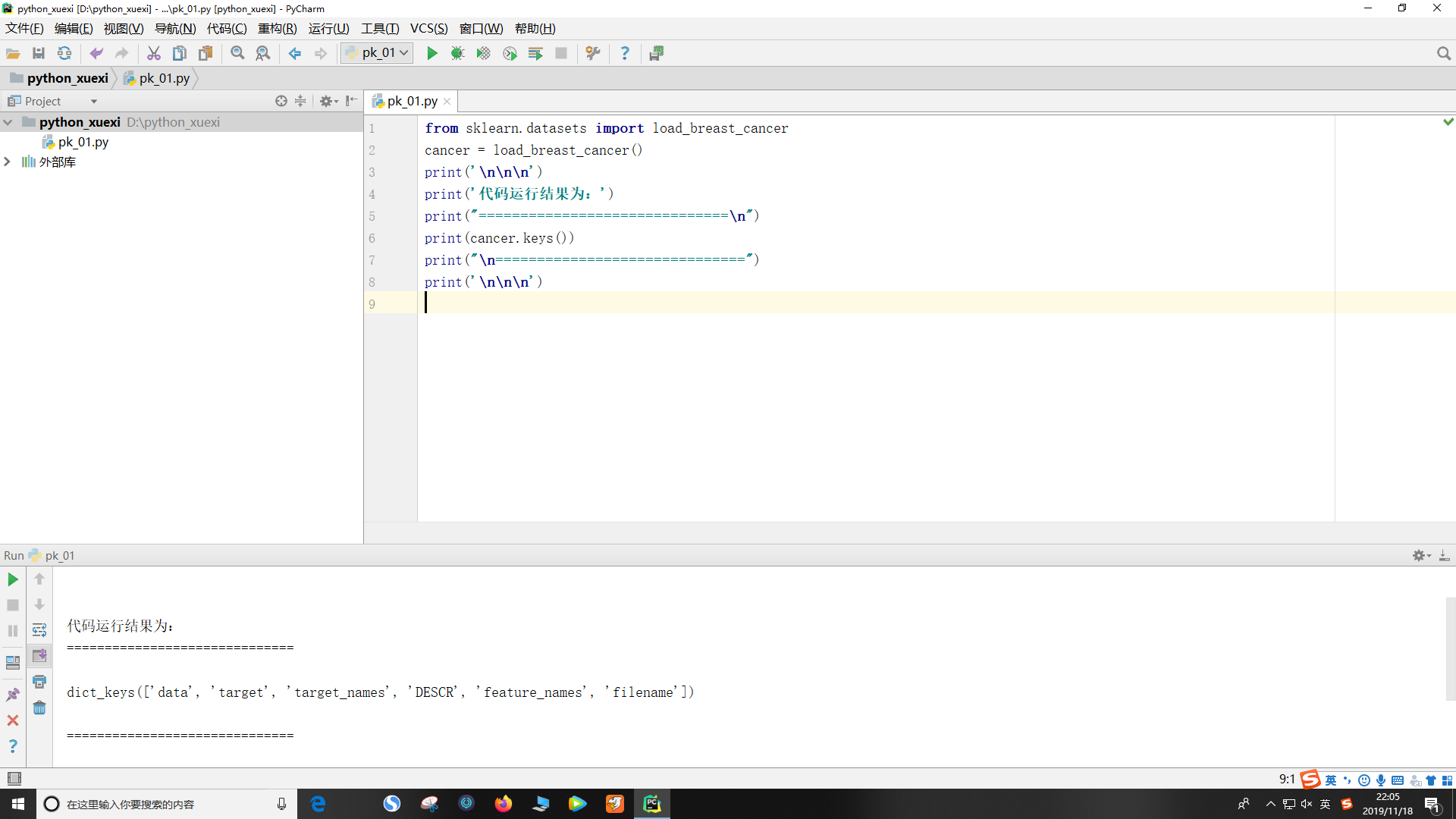
**1.威斯康星乳腺肿瘤数据集是一个非常经典的用于医疗病情分析的数据集，他包括500个病例样本，每个样本有三十个特征值，而样本分为两类，恶性和良性。**

**1.使用高斯贝叶斯**

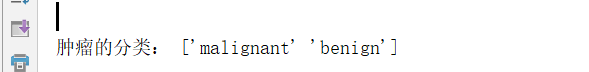
**#导入数据集并打印数据集键值**

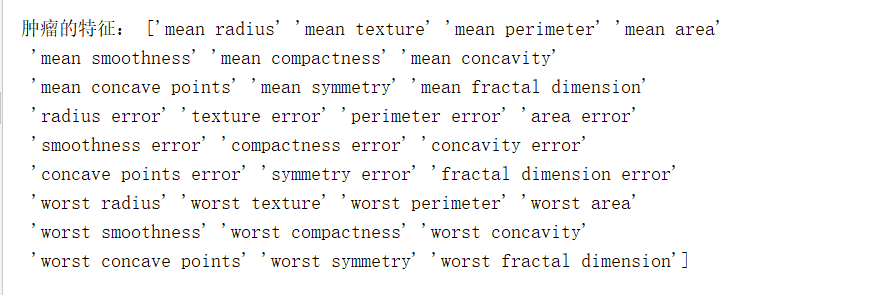
**from** sklearn.datasets **import** load\_breast\_cancer  
**from** sklearn.model\_selection **import** train\_test\_split*#用model\_selection模块导入train\_test\_split***from** sklearn.naive\_bayes **import** GaussianNB*#调用高斯贝叶斯***from** sklearn.model\_selection **import** ShuffleSplit  
**from** matplotlib **import** pyplot **as** plt  
**import** numpy **as** np  
**from** sklearn.model\_selection **import** learning\_curve  
fig = plt.figure()

cancer = load\_breast\_cancer()  
print(**'\n\n\n'**)  
print(**'代码运行结果为：'**)  
print(**"==============================\n"**)  
print(cancer.keys())  
print(**"\n=============================="**)  
print(**'\n\n\n'**)

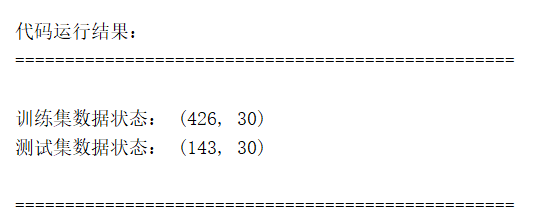


*#打印数据集中标注好的肿瘤分类*print(**"肿瘤的分类："**,cancer[**"target\_names"**])

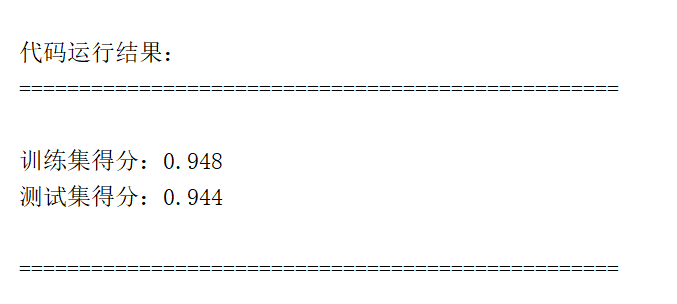
  
*#打印数据集中的肿瘤特征名称*print(**"\n肿瘤的特征："**,cancer[**"feature\_names"**])



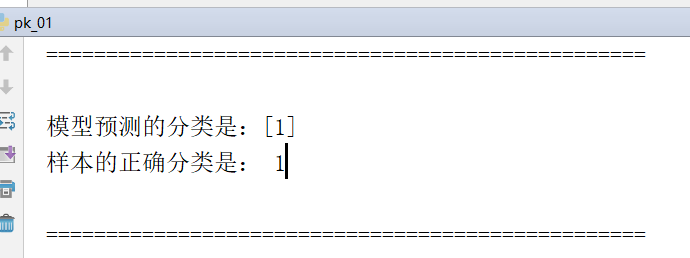
*#将数据集中的数值和分类目标分别赋值给x和y*X,y = cancer.data,cancer.target  
*#使用数据集拆分工具拆分为训练集和测试集*X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,random\_state=38)  
print(**"\n\n\n"**)  
print(**"代码运行结果："**)  
print(**"==================================================\n"**)  
*#打印训练集和测试集的数据状态*print(**"训练集数据状态："**,X\_train.shape)  
print(**"测试集数据状态："**,X\_test.shape)  
print(**"\n=================================================="**)  
print(**"\n\n\n"**)



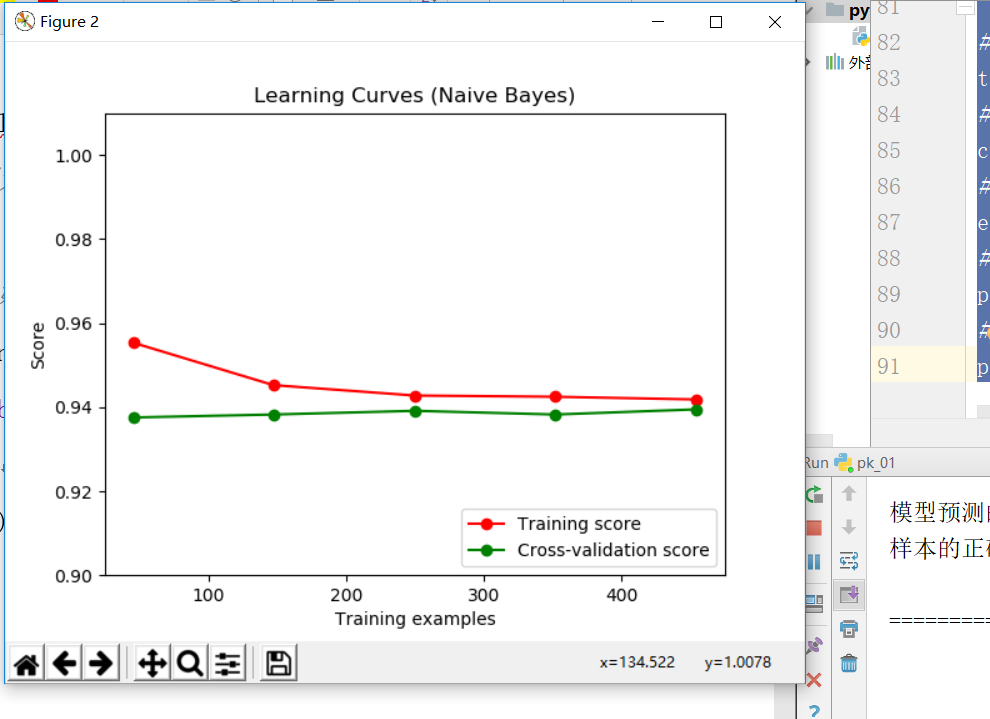
*#使用高斯朴素贝叶斯拟合数据*gnb = GaussianNB()  
gnb.fit(X\_train,y\_train)  
print(**"\n\n\n"**)  
print(**"代码运行结果："**)  
print(**"==================================================\n"**)  
*#打印模型评分*print(**"训练集得分：{:.3f}"**.format(gnb.score(X\_train,y\_train)))  
print(**"测试集得分：{:.3f}"**.format(gnb.score(X\_test,y\_test)))  
print(**"\n=================================================="**)  
print(**"\n\n\n"**)



*#随便使用一个模型进行预测*print(**"\n\n\n"**)  
print(**"代码运行结果："**)  
print(**"==================================================\n"**)  
*#打印模型预测的分类和真实的分类*print(**"模型预测的分类是：{}"**.format(gnb.predict([X[312]])))  
print(**"样本的正确分类是："**,y[312])  
print(**"\n=================================================="**)  
print(**"\n\n\n"**)



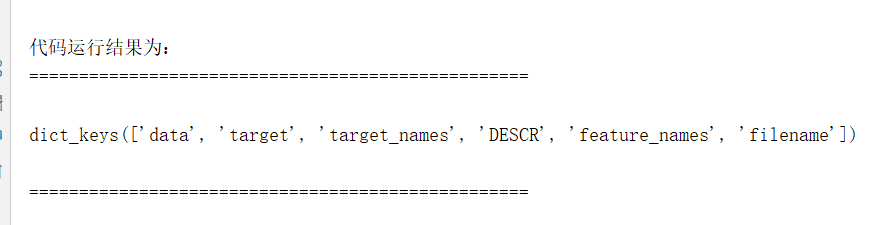
*#定义一个函数绘制学习曲线***def** plot\_learning\_curve(estimator,title,X,y,ylim=**None**,cv=**None**,  
 n\_jobs=1,train\_sizes=np.linspace(.1,1.0,5)):  
 plt.figure()  
 plt.title(title)  
 **if** ylim **is not None**:  
 plt.ylim(\*ylim)  
  
*#设定横轴标签* plt.xlabel(**"Training examples"**)  
*#设定纵轴标签* plt.ylabel(**"Score"**)  
 train\_sizes,train\_scores,test\_scores = learning\_curve(  
 estimator,X,y,cv=cv,n\_jobs=n\_jobs,train\_sizes=train\_sizes  
 )  
 train\_scores\_mean = np.mean(train\_scores,axis=1)  
 test\_scores\_mean = np.mean(test\_scores,axis=1)  
 plt.plot(train\_sizes,train\_scores\_mean,**'o-'**,color=**"r"**,  
 label=**"Training score"**)  
 plt.plot(train\_sizes,test\_scores\_mean,**'o-'**,color=**"g"**,  
 label=**"Cross-validation score"**)  
 plt.legend(loc=**"lower right"**)  
 **return** plt  
*#设定图题*title = **"Learning Curves (Naive Bayes)"***#设定拆分量*cv = ShuffleSplit(n\_splits=100,test\_size=0.2,random\_state=0)  
*#设定模型为高斯朴素贝叶斯*estimator = GaussianNB()  
*#调用定义好的函数*plot\_learning\_curve(estimator,title,X,y,ylim=(0.9,1.01),cv=cv,n\_jobs=4)  
*#显示图片*plt.show()



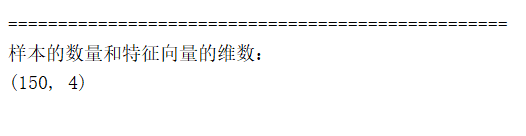
**2.使用贝叶斯分析器分析鸢尾花数据**

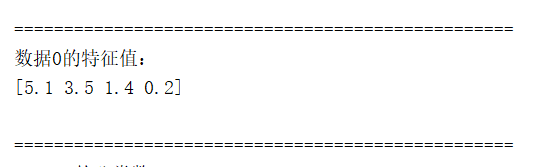
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** sklearn **import** datasets  
**from** sklearn.naive\_bayes **import** GaussianNB  
**from** sklearn.datasets **import** load\_iris

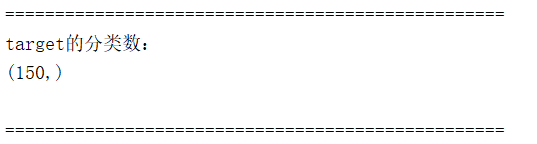
*#打印数据集的关键值*  
iris = load\_iris()  
print(**'\n\n\n'**)  
print(**'代码运行结果为：'**)  
print(**"==================================================\n"**)  
print(iris.keys())  
print(**"\n=================================================="**)  
print(**'\n\n\n'**)

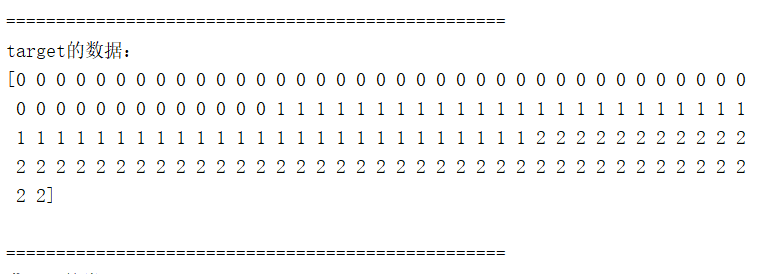


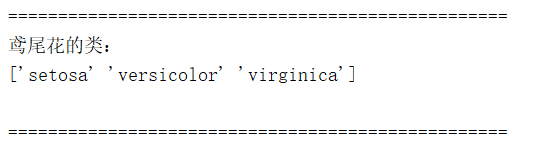
*#打印数据集中标注好的分类*n\_samples, n\_features = iris.data.shape  
  
print(**"\n=================================================="**)  
print(**"样本的数量和特征向量的维数："**)  
print((n\_samples, n\_features))  
print(**"\n=================================================="**)

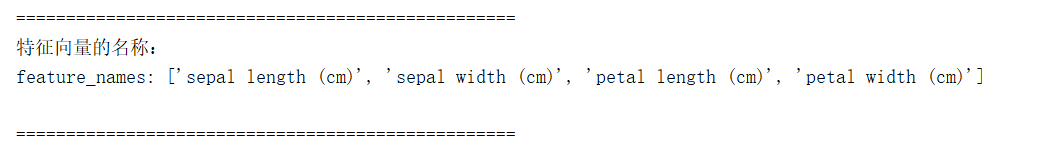
  
print(**"数据0的特征值："**)  
print(iris.data[0])  
print(**"\n=================================================="**)

  
print(**"target的分类数："**)  
print(iris.target.shape)  
print(**"\n=================================================="**)

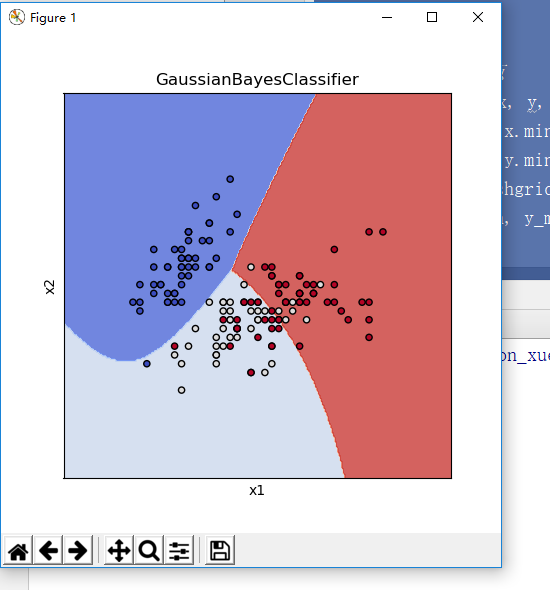
  
print(**"target的数据："**)  
print(iris.target)  
print(**"\n=================================================="**)

  
print(**"鸢尾花的类："**)  
print(iris.target\_names)  
print(**"\n=================================================="**)

  
print(**"特征向量的名称："**)  
print(**"feature\_names:"**,iris.feature\_names)  
print(**"\n=================================================="**)



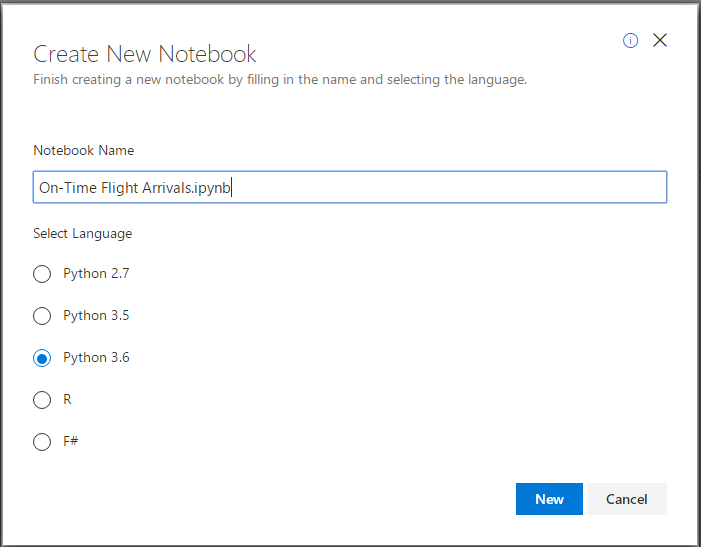
*# 生成所有测试样本点***def** make\_meshgrid(x, y, h=.02):  
 x\_min, x\_max = x.min() - 1, x.max() + 1  
 y\_min, y\_max = y.min() - 1, y.max() + 1  
 xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),  
 np.arange(y\_min, y\_max, h))  
 **return** xx, yy  
  
*# 对测试样本进行预测，并显示***def** plot\_test\_results(ax, clf, xx, yy, \*\*params):  
 Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])  
 Z = Z.reshape(xx.shape)  
 ax.contourf(xx, yy, Z, \*\*params)  
  
*# 载入iris数据集*iris = datasets.load\_iris()  
*# 只使用前面连个特征*X = iris.data[:, :2]  
*# 样本标签值*y = iris.target  
  
*# 创建并训练正态朴素贝叶斯分类器*clf = GaussianNB()  
clf.fit(X,y)  
  
title = (**'GaussianBayesClassifier'**)  
  
fig, ax = plt.subplots(figsize = (5, 5))  
plt.subplots\_adjust(wspace=0.4, hspace=0.4)  
  
X0, X1 = X[:, 0], X[:, 1]  
*# 生成所有测试样本点*xx, yy = make\_meshgrid(X0, X1)  
  
*# 显示测试样本的分类结果*plot\_test\_results(ax, clf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)  
*# 显示训练样本*ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors=**'k'**)  
ax.set\_xlim(xx.min(), xx.max())  
ax.set\_ylim(yy.min(), yy.max())  
ax.set\_xlabel(**'x1'**)  
ax.set\_ylabel(**'x2'**)  
ax.set\_xticks(())  
ax.set\_yticks(())  
ax.set\_title(title)  
plt.show()



**3.在Azure上的机器学习**

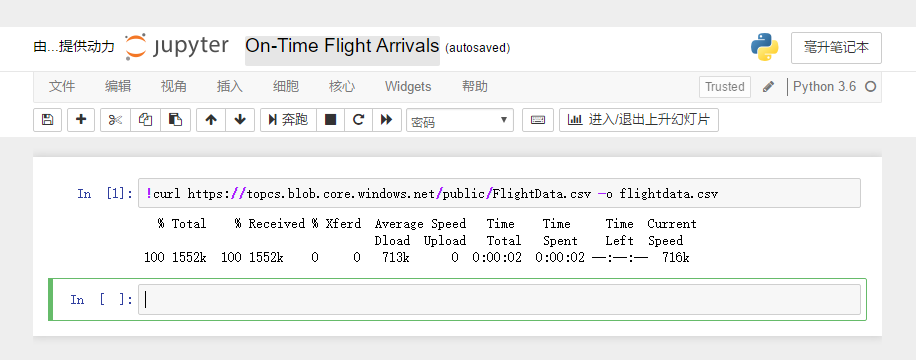
Azure Notebooks 是一个基于云的平台，用于生成和运行 Jupyter Notebook。 Jupyter 是基于 IPython 的环境，它有助于使用 Python 和其他编程语言进行交互式编程和数据分析。 Azure Notebooks 免费提供 Jupyter 服务。 它是无需安装和管理 Jupyter 服务器便能编写 Python 代码的便捷方法。 它基于 Web，是联机协作的理想解决方案。

在本模块中，将创建一个 Azure Notebook，导入包含美国一家主要航空公司的**准点到达**信息的数据集，并将数据集加载到笔记本中。 然后将使用 Pandas 清理数据集，使用 Scikit-learn 构建机器学习模型，并使用 Matplotlib 可视化模型的输出。



!curl [https://topcs.blob.core.windows.net/public/FlightData.csv -o flightdata.csv](https://topcs.blob.core.windows.net/public/FlightData.csv%20-o%20flightdata.csv)

Curl是bash命令，可以用jupyter笔记本中通过为其添加感叹号前缀来执行bash命令。此命令从azure blob存储下载csv文件，并用flightdata.csv来保存它

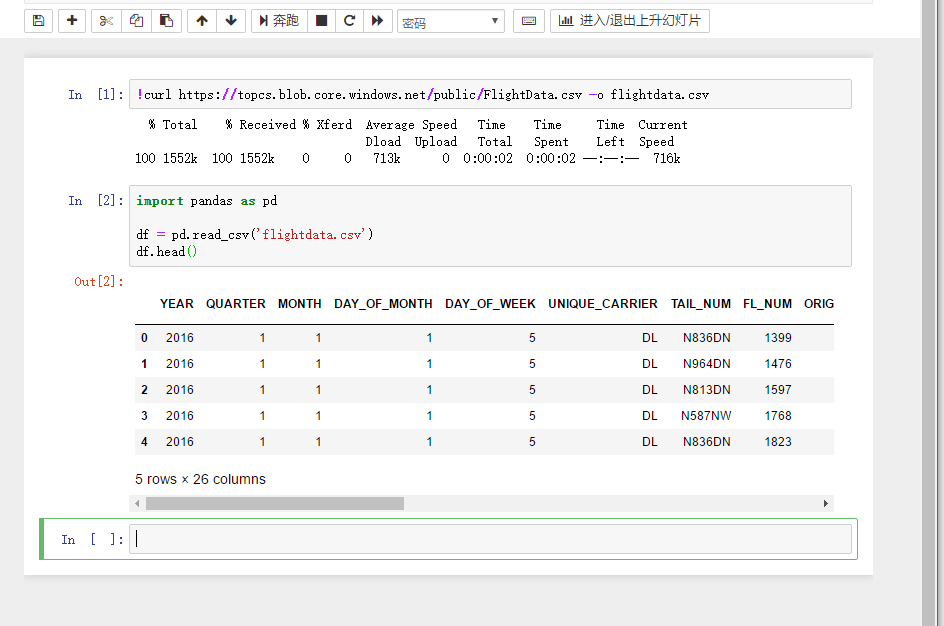


读取数据，使用head函数返回前五行数据

import pandas as pd

df = pd.read\_csv('flightdata.csv')

df.head()



使用df.shape来查看数据的行数和列数

df.shape



数据中列的含义

| **列** | **说明** |
| --- | --- |
| YEAR | 航班飞行的年份 |
| QUARTER | 航班飞行的季度 (1-4) |
| MONTH | 航班飞行的月份 (1-12) |
| DAY\_OF\_MONTH | 航班飞行的当月日期 (1-31) |
| DAY\_OF\_WEEK | 航班飞行的当周日期（1 = 星期一，2 = 星期二等） |
| UNIQUE\_CARRIER | 航空公司代码（例如，DL） |
| TAIL\_NUM | 飞机机尾编号 |
| FL\_NUM | 航班号 |
| ORIGIN\_AIRPORT\_ID | 出发地机场 ID |
| ORIGIN | 出发地机场代码（ATL、DFW、SEA 等） |
| DEST\_AIRPORT\_ID | 目的地机场 ID |
| DEST | 目的地机场代码（ATL、DFW、SEA 等） |
| CRS\_DEP\_TIME | 计划出发时间 |
| DEP\_TIME | 实际出发时间 |
| DEP\_DELAY | 出发晚点分钟数 |
| DEP\_DEL15 | 0 = 出发晚点不到 15 分钟，1 = 出发晚点 15 分钟或更长时间 |
| CRS\_ARR\_TIME | 计划到达时间 |
| ARR\_TIME | 实际到达时间 |
| ARR\_DELAY | 航班到达晚点分钟数 |
| ARR\_DEL15 | 0 = 到达晚点不到 15 分钟，1 = 到达晚点 15 分钟或更长时间 |
| CANCELLED | 0 = 航班未取消，1 = 航班已取消 |
| DIVERTED | 0 = 航班未转移，1 = 航班已转移 |
| CRS\_ELAPSED\_TIME | 以分钟为单位的计划飞行时间 |
| ACTUAL\_ELAPSED\_TIME | 以分钟为单位的实际飞行时间 |
| DISTANCE | 以英里为单位的飞行距离 |

将清除无关的列并替换剩余列中的缺失值。

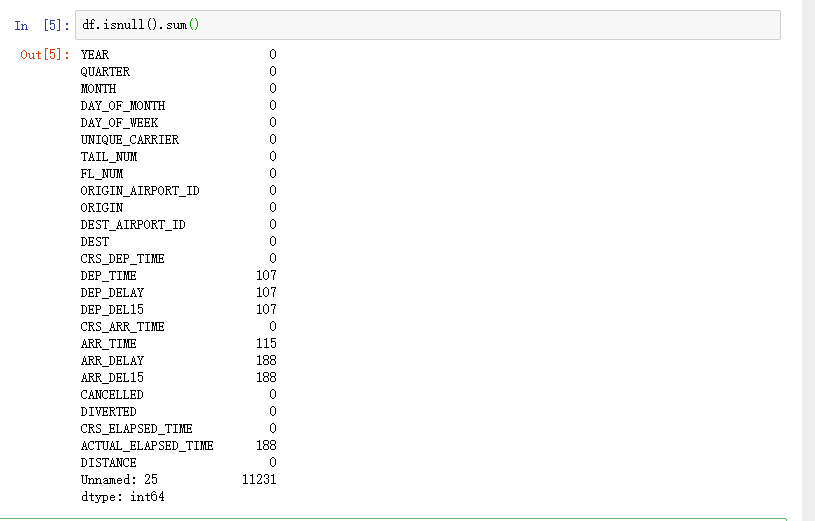
df.isnull().values.any()

当输出为true时，表示数据中某处至少有一个缺失值



查找缺失值的所在位置

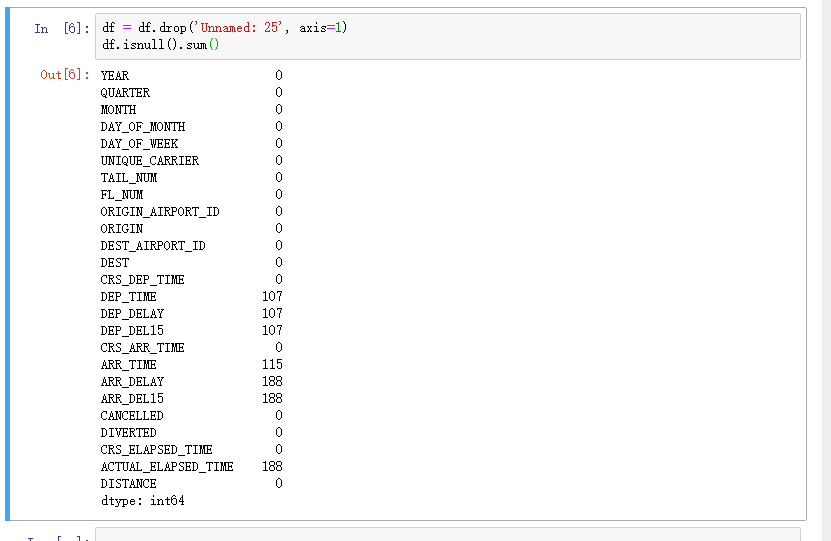
df.isnull().sum()



Unamed包含11231个缺失值，这等于数据集中的行数，说明此列呗错误的创建，实际上是数据集中每行句末的一个逗号，接下来清除该列

df = df.drop('Unnamed: 25', axis=1)

df.isnull().sum()



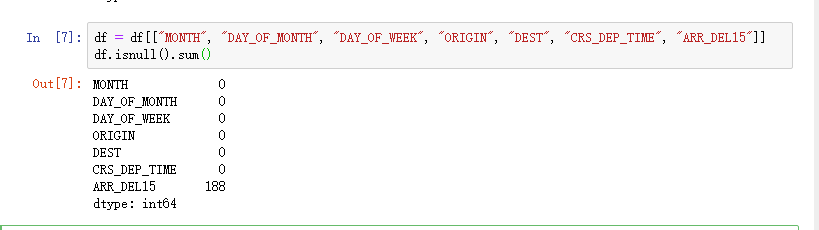
dataFrame中仍然包含许多缺失值，但是其中一些缺失值是无用的，因为包含它们的列与你正在构建的模型无关。 该模型的目标是预测你考虑预订的航班是否可能准点到达。因此，下一步是筛选数据集以清除与预测模型无关的列。

例如，飞机的机尾编号可能与航班是否准点到达几乎没有关联，而在你预订机票时，你无法知道航班是否会取消、转移或晚点。 相比之下，计划出发时间可能与准点到达有很大关联。 由于大多数航空公司使用轮辐式系统，早上的航班往往比下午或晚上的航班更准时。 在一些主要机场，客流量在白天堆积，增加了后期航班晚点的可能性。

使用pandas提供的一种简单放大筛选掉不需要的列

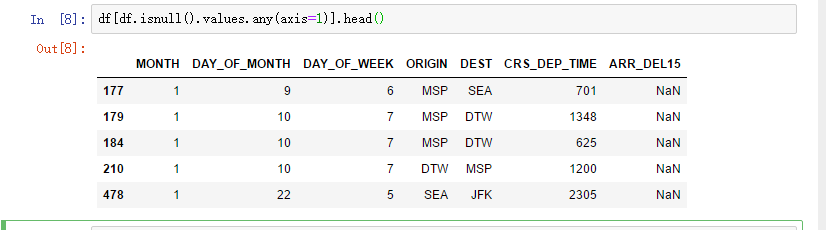
df = df[["MONTH", "DAY\_OF\_MONTH", "DAY\_OF\_WEEK", "ORIGIN", "DEST", "CRS\_DEP\_TIME", "ARR\_DEL15"]]

df.isnull().sum()



现在，唯一包含缺失值的列是 ARR\_DEL15 列，该列使用 0 来标识准点到达的航班，使用 1 来标识未准点到达的航班。 使用以下代码显示包含缺失值的前五行：

df[df.isnull().values.any(axis=1)].head()

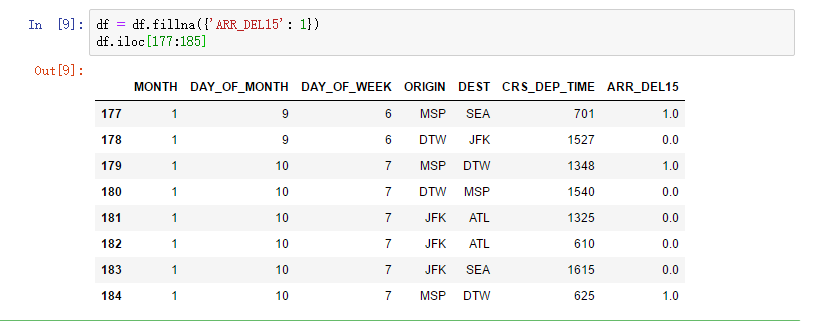


其中NaN表示缺失值，意思为非数字

这些行缺失 ARR\_DEL15 值的原因在于它们全都对应于已取消或转移的航班。 你可以在 DataFrame 上调用 dropna来删除这些行。 但是，由于取消或转移到另一机场的航班可能被视为“晚点”，让我们使用 filla方法将缺失值替换为 1。再查看117-184行，其中NaN已经被替换为1

df = df.fillna({'ARR\_DEL15': 1})

df.iloc[177:185]



数据集现在是“干净的”，因为缺失值已被替换，且列的列表已将范围缩小到与模型最相关的列。

正在使用的数据集的 CRS\_DEP\_TIME 列表示计划出发时间。 此列中数字的粒度可能会对机器学习模型的准确率产生负面影响。 这可以使用名为分箱或量化的技术来解决。将此列中的每个数字除以 100 并向下舍入到最近的整数。1030 将变为 10，1925 将变为 19，依此类推，并且将在此列中最多留下 24 个离散值。

直观地说，这是有道理的，因为航班是在上午 10:30 还是在上午 10:40 出发可能无关紧要。 重点在于它是在上午 10:30 出发 还是在下午 5:30 出发。

此外，数据集的 ORIGIN 和 DEST 列包含代表分类机器学习值的机场代码。 这些列需要转换为包含指示变量（有时称为“虚拟”变量）的离散列。 换句话说，包含五个机场代码的 ORIGIN 列需要转换为五列，每个机场一列，且每列都包含指示航班是否从该列所表示的机场出发的 1 和 0。 DEST 列需要以类似的方式进行处理。

CRS\_DEP\_TIME 列包含表示军事时间的 0 到 2359 之间的值。

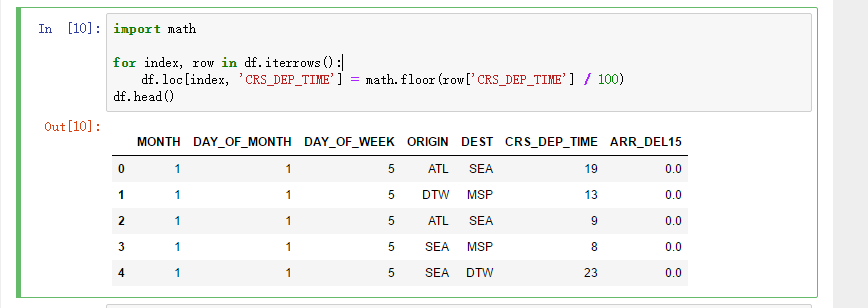
使用以下方法来讲数据分箱，并且使用pandas的get dummies方法从origin和dest列创建指示列

import math

for index, row in df.iterrows():

df.loc[index, 'CRS\_DEP\_TIME'] = math.floor(row['CRS\_DEP\_TIME'] / 100)

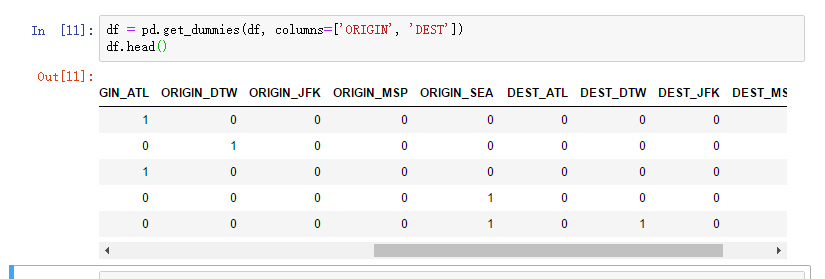
df.head()



现在使用语句生成指示列并且删除origin和dest列

df = pd.get\_dummies(df, columns=['ORIGIN', 'DEST'])

df.head()



Origin为1的是从此处起飞，而dest为1的是到此处降落

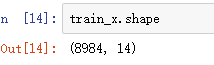
from sklearn.model\_selection import train\_test\_split

train\_x, test\_x, train\_y, test\_y = train\_test\_split(df.drop('ARR\_DEL15', axis=1), df['ARR\_DEL15'], test\_size=0.2, random\_state=42)

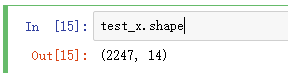
第一个语句导入 Scikit-learn 的 train test split helper 函数。 第二行使用该函数将 DataFrame 拆分为包含 80% 原始数据的训练集，以及包含剩余 20% 原始数据的测试集。 random\_state 参数对用于执行拆分的随机数生成器播种，而第一个和第二个参数是包含特征列和标签列的 DataFrame。

查看行数和列数

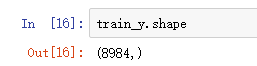
train\_x.shape



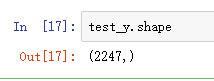
test\_x.shape



train\_y.shape



test\_y.shape



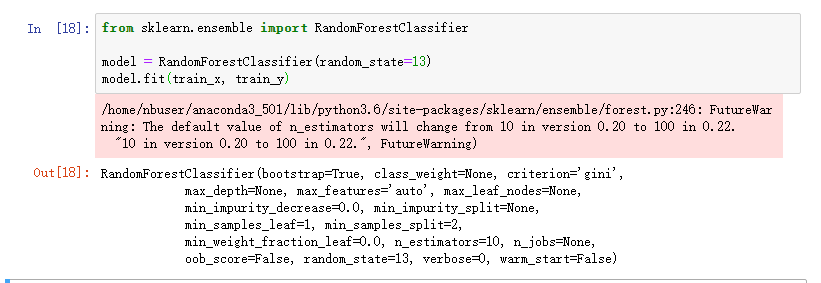
RandomForestClassifier使多个决策树拟合数据，并使用平均方法来提高整体准确率及限制过拟合。

通过fit方法进行训练

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random\_state=13)

model.fit(train\_x, train\_y)

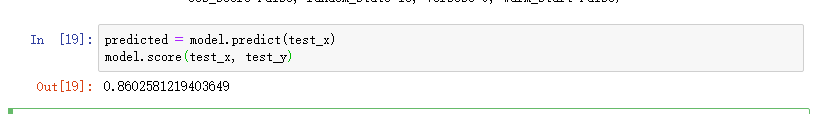


输出显示分类器中使用的参数，包括 n\_estimators（指定每个决策树林中的树的数量），以及 max\_depth（指定决策树的最大深度）。 显示的值为默认值，但可以在创建 RandomForestClassifier 对象时对任意值进行替代。

现在，调用 predict方法以使用 test\_x 中的值测试模型，然后使用score 方法来确定模型的平均准确率

predicted = model.predict(test\_x)

model.score(test\_x, test\_y)



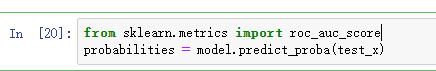
平均准确率为 86%，从表面上看似乎很好。 然而，平均准确率并不总是分类模型准确率的可靠指示器。 让我们更深入地探讨并确定模型的真实准确率，即它在确定航班是否准点到达方面的擅长程度。

有多种方法可用于度量分类模型的准确率。 二进制分类模型的最佳整体度量值之一是接受者工作特征曲线下面积（有时称为“ROC AUC”），它实质上量化了模型进行正确预测的频率，而不管结果如何。 在本单元中，你将计算之前构建的模型的 ROC AUC 分数，并了解该分数低于 score 方法输出的平均准确率的部分原因。 你还将了解估计模型准确率的其他方法。

在计算 ROC AUC 之前，你必须为测试集生成预测概率。 这些概率是模型可以预测的每个类或答案的估计值。 例如，[0.88199435, 0.11800565] 表示航班准点到达的几率为 89% (ARR\_DEL15 = 0)，不能准点到达的几率为 12% (ARR\_DEL15 = 1)。 两个概率的总和为 100%。

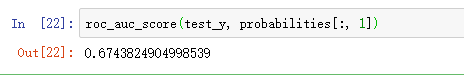
from sklearn.metrics import roc\_auc\_score

probabilities = model.predict\_proba(test\_x)



现在，使用以下语句通过 Scikit-learn 的roc auc score 方法从概率生成 ROC AUC 分数

roc\_auc\_score(test\_y, probabilities[:, 1])



score 方法的输出反映模型可以对测试集中的多少项进行正确预测。 此分数受到以下事实的影响：用于对模型定型和测试的数据集包含的代表准点到达的行数多于代表晚点到达的行数。 由于数据中的这种不平衡，预测航班准点到达可能会比预测航班晚点到达的正确率要高。

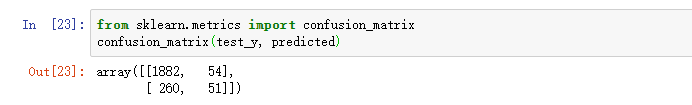
ROC AUC 将这一点考虑在内，并提供准点或晚点的预测将为正确预测的可能性的更准确指示。

可以通过生成混淆矩阵（也称为错误矩阵）来了解有关模型行为的详细信息。 混淆矩阵量化每个答案被正确分类或错误分类的次数。 具体而言，它量化了假正、假负、真正和真负的数量。 这很重要，因为如果使用包含 95% 的狗的数据集测试训练用于识别猫和狗的二进制分类模型，那么每次只需猜测“狗”，分数就可达到 95%。 但如果该模型根本不能识别猫，那就没有任何意义。

使用以下代码为模型生成混淆矩阵

from sklearn.metrics import confusion\_matrix

confusion\_matrix(test\_y, predicted)



输出中的第一行表示准点到达的航班。 该行的第一列显示正确预测为准点到达的航班的数量，而第二列则显示预测为晚点但实际未晚点的航班的数量。 由此可以看出，该模型似乎擅长预测航班准点到达。

但是在第二行，它代表晚点的航班。 第一列显示被错误预测为准点到达的晚点航班的数量。 第二列显示正确预测为晚点的航班的数量。 显然，该模型并不像预测航班将准点到达那样善于预测航班将晚点。 在混淆矩阵中所需的内容是左上角和右下角的大数字，以及右上角和左下角的小数字（最好是零）。

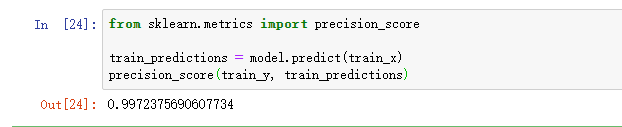
分类模型的其他准确率度量值包括精确率和召回率。 假设向该模型提供了三个准点到达的航班和三个晚点到达的航班，并且它正确预测了其中两个准点到达的航班，但错误地将其中两个晚点到达的航班预测为准点到达。 在这种情况下，精确率将为 50%（在其分类为准点到达的四个航班中，实际上有两个航班准点到达），而其召回率为 67%（它正确识别了三个准点到达的航班中的两个）。

Scikit-learn 包含名为 precision score 的便捷方法，可用于计算精确率。

from sklearn.metrics import precision\_score

train\_predictions = model.predict(train\_x)

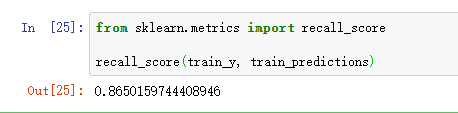
precision\_score(train\_y, train\_predictions)



Scikit-learn 还包含名为 recall score 的方法，可用于计算召回率。

from sklearn.metrics import recall\_score

recall\_score(train\_y, train\_predictions)

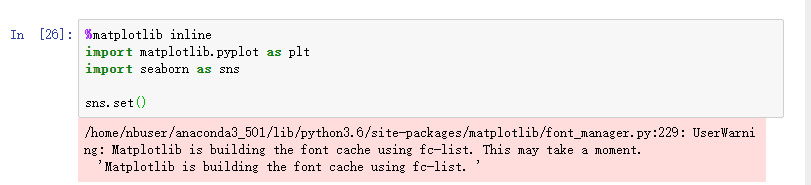


%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()



第一个语句是在创建笔记本时选择的 Python 内核支持的多个 magic命令之一。 它使 Jupyter 能够在笔记本中呈现 Matplotlib 输出，而无需执行重复调用来显示。 它必须出现在任何对 Matplotlib 本身的引用之前。 最后一个语句配置 seaborn来增强 Matplotlib 的输出。

查看 Matplotlib 的工作状况，执行以下代码，为构建的机器学习模型绘制 roc曲线

from sklearn.metrics import roc\_curve

fpr, tpr, \_ = roc\_curve(test\_y, probabilities[:, 1])

plt.plot(fpr, tpr)

plt.plot([0, 1], [0, 1], color='grey', lw=1, linestyle='--')

plt.xlabel('False Positive Rate')

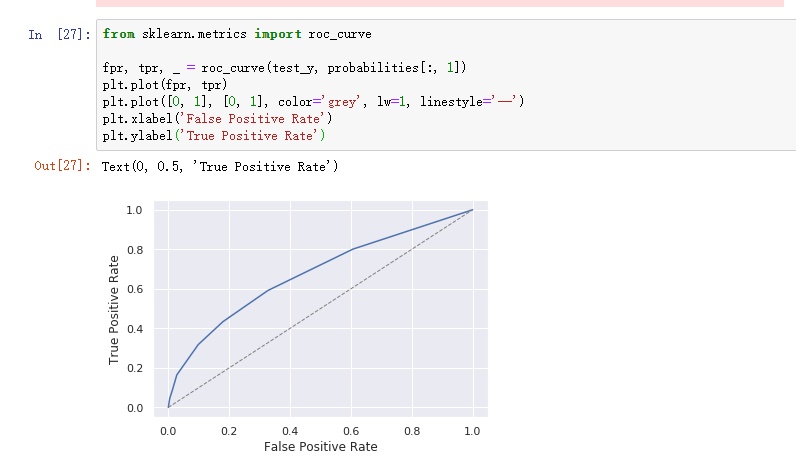
plt.ylabel('True Positive Rate')

此函数使用日期和时间、出发地机场代码和目的地机场代码作为输入，并返回介于 0.0 和 1.0 之间的值，指示航班准点到达目的地的概率。 它使用构建的机器学习模型来计算概率。 为了调用模型，它会将包含输入值的 DataFrame 传递到 predict\_proba。 DataFrame 的结构与我们之前使用的 DataFrame 的结构完全匹配。

predict\_delay 函数的日期输入使用国际日期格式 dd/mm/year。

计算 10 月 1 日晚上从纽约飞往亚特兰大的航班准点到达的概率。 输入的年份无关紧要，因为模型不使用它。

predict\_delay('1/10/2018 21:45:00', 'JFK', 'ATL')



图中间的虚线表示获得正确答案的几率为 50%。 蓝色曲线表示模型的准确率。

构建机器学习模型的原因在于预测航班是会准点到达还是晚点到达。 在本练习中，编写一个 Python 函数，该函数调用构建的机器学习模型，以计算航班准点到达的可能性。 然后将使用该函数分析多个航班。

def predict\_delay(departure\_date\_time, origin, destination):

from datetime import datetime

try:

departure\_date\_time\_parsed = datetime.strptime(departure\_date\_time, '%d/%m/%Y %H:%M:%S')

except ValueError as e:

return 'Error parsing date/time - {}'.format(e)

month = departure\_date\_time\_parsed.month

day = departure\_date\_time\_parsed.day

day\_of\_week = departure\_date\_time\_parsed.isoweekday()

hour = departure\_date\_time\_parsed.hour

origin = origin.upper()

destination = destination.upper()

input = [{'MONTH': month,

'DAY': day,

'DAY\_OF\_WEEK': day\_of\_week,

'CRS\_DEP\_TIME': hour,

'ORIGIN\_ATL': 1 if origin == 'ATL' else 0,

'ORIGIN\_DTW': 1 if origin == 'DTW' else 0,

'ORIGIN\_JFK': 1 if origin == 'JFK' else 0,

'ORIGIN\_MSP': 1 if origin == 'MSP' else 0,

'ORIGIN\_SEA': 1 if origin == 'SEA' else 0,

'DEST\_ATL': 1 if destination == 'ATL' else 0,

'DEST\_DTW': 1 if destination == 'DTW' else 0,

'DEST\_JFK': 1 if destination == 'JFK' else 0,

'DEST\_MSP': 1 if destination == 'MSP' else 0,

'DEST\_SEA': 1 if destination == 'SEA' else 0 }]

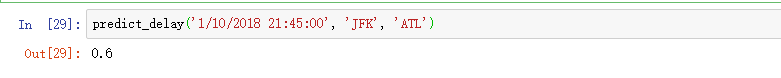
return model.predict\_proba(pd.DataFrame(input))[0][0]



此函数使用日期和时间、出发地机场代码和目的地机场代码作为输入，并返回介于 0.0 和 1.0 之间的值，指示航班准点到达目的地的概率。 它使用在上一实验中构建的机器学习模型来计算概率。 为了调用模型，它会将包含输入值的 DataFrame 传递到 predict\_proba。 DataFrame 的结构与我们之前使用的 DataFrame 的结构完全匹配。

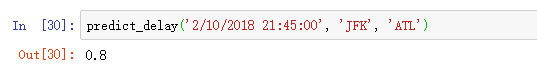
计算 10 月 1 日晚上从纽约飞往亚特兰大的航班准点到达的概率。 输入的年份无关紧要，因为模型不使用它。

predict\_delay('1/10/2018 21:45:00', 'JFK', 'ATL')



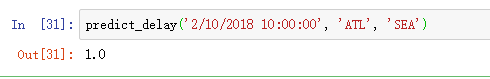
修改代码以计算一天后同一航班准点到达的概率

predict\_delay('2/10/2018 21:45:00', 'JFK', 'ATL')



修改代码以计算同一天从亚特兰大飞往西雅图的早班航班准点到达的概率

predict\_delay('2/10/2018 10:00:00', 'ATL', 'SEA')



绘制数天内从 JFK 飞往 ATL 的晚间航班准点到达的概率

import numpy as np

labels = ('Oct 1', 'Oct 2', 'Oct 3', 'Oct 4', 'Oct 5', 'Oct 6', 'Oct 7')

values = (predict\_delay('1/10/2018 21:45:00', 'JFK', 'ATL'),

predict\_delay('2/10/2018 21:45:00', 'JFK', 'ATL'),

predict\_delay('3/10/2018 21:45:00', 'JFK', 'ATL'),

predict\_delay('4/10/2018 21:45:00', 'JFK', 'ATL'),

predict\_delay('5/10/2018 21:45:00', 'JFK', 'ATL'),

predict\_delay('6/10/2018 21:45:00', 'JFK', 'ATL'),

predict\_delay('7/10/2018 21:45:00', 'JFK', 'ATL'))

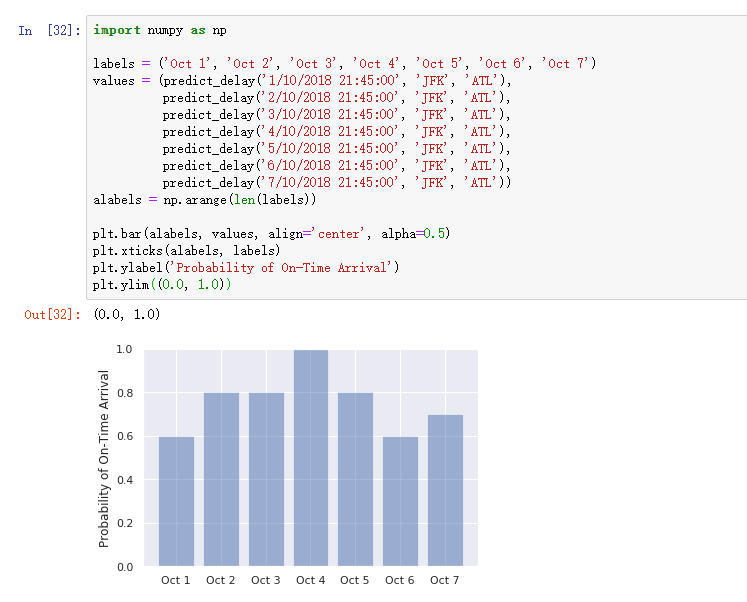
alabels = np.arange(len(labels))

plt.bar(alabels, values, align='center', alpha=0.5)

plt.xticks(alabels, labels)

plt.ylabel('Probability of On-Time Arrival')

plt.ylim((0.0, 1.0))



**5.气候变化研究（线性回归）**

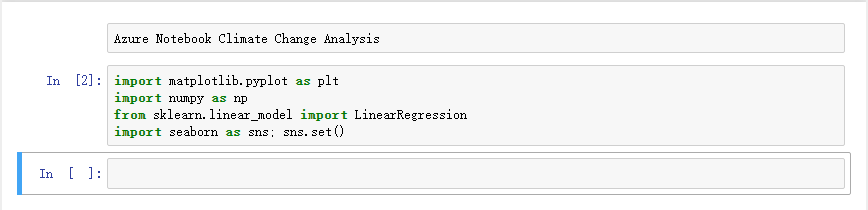
**加载需要用到的包**

import matplotlib.pyplot as plt

import numpy as np

from sklearn.linear\_model import LinearRegression

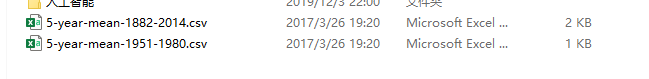
import seaborn as sns; sns.set()



将要用到的数据导入内存中，并且调用

yearsBase, meanBase = np.loadtxt('5-year-mean-1951-1980.csv', delimiter=',', usecols=(0, 1), unpack=True)

years, mean = np.loadtxt('5-year-mean-1882-2014.csv', delimiter=',', usecols=(0, 1), unpack=True)





使用Matplotlib创建散点图

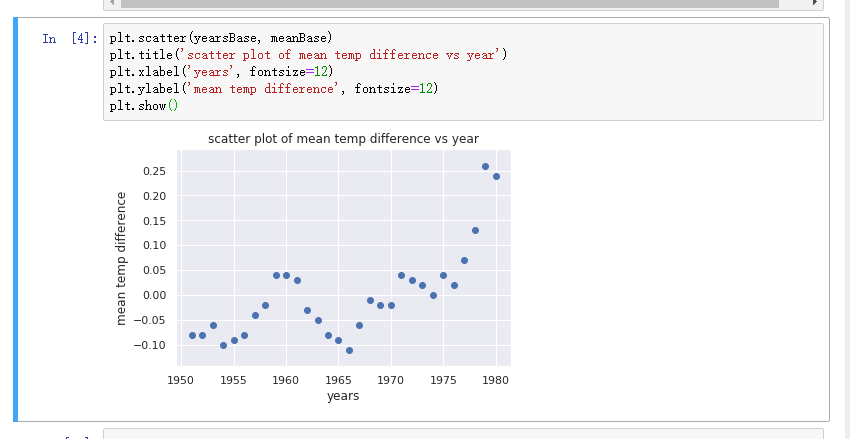
plt.scatter(yearsBase, meanBase)

plt.title('scatter plot of mean temp difference vs year')

plt.xlabel('years', fontsize=12)

plt.ylabel('mean temp difference', fontsize=12)

plt.show()



使用Numpy执行线性回归

# Creates a linear regression from the data points

m,b = np.polyfit(yearsBase, meanBase, 1)

# This is a simple y = mx + b line function

def f(x):

return m\*x + b

# This generates the same scatter plot as before, but adds a line plot using the function above

plt.scatter(yearsBase, meanBase)

plt.plot(yearsBase, f(yearsBase))

plt.title('scatter plot of mean temp difference vs year')

plt.xlabel('years', fontsize=12)

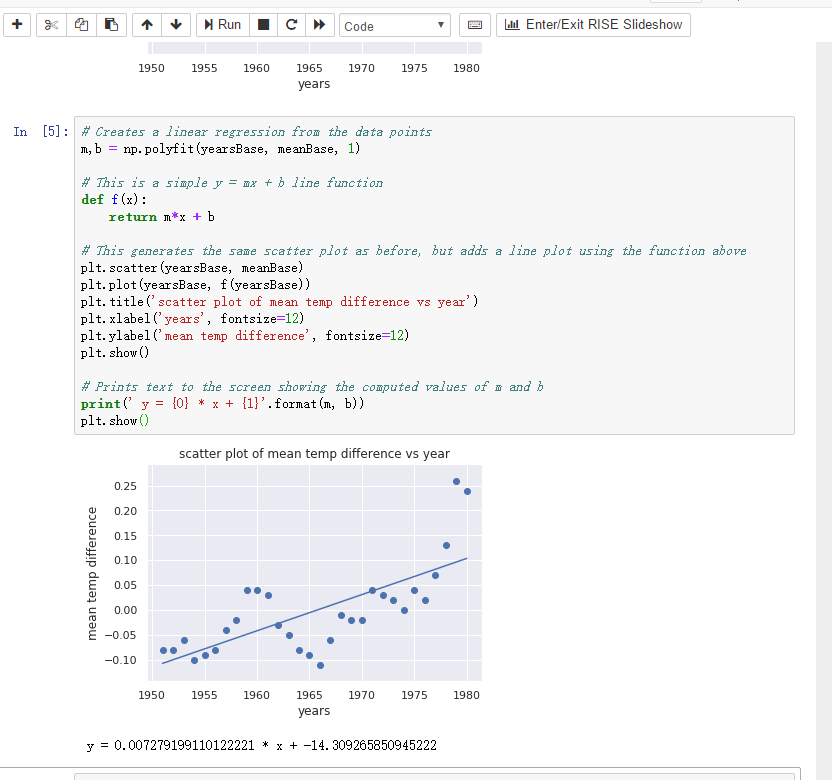
plt.ylabel('mean temp difference', fontsize=12)

plt.show()

# Prints text to the screen showing the computed values of m and b

print(' y = {0} \* x + {1}'.format(m, b))

plt.show()



使用Scikit Learn执行线性回归

# Pick the Linear Regression model and instantiate it

model = LinearRegression(fit\_intercept=True)

# Fit/build the model

model.fit(yearsBase[:, np.newaxis], meanBase)

mean\_predicted = model.predict(yearsBase[:, np.newaxis])

# Generate a plot like the one in the previous exercise

plt.scatter(yearsBase, meanBase)

plt.plot(yearsBase, mean\_predicted)

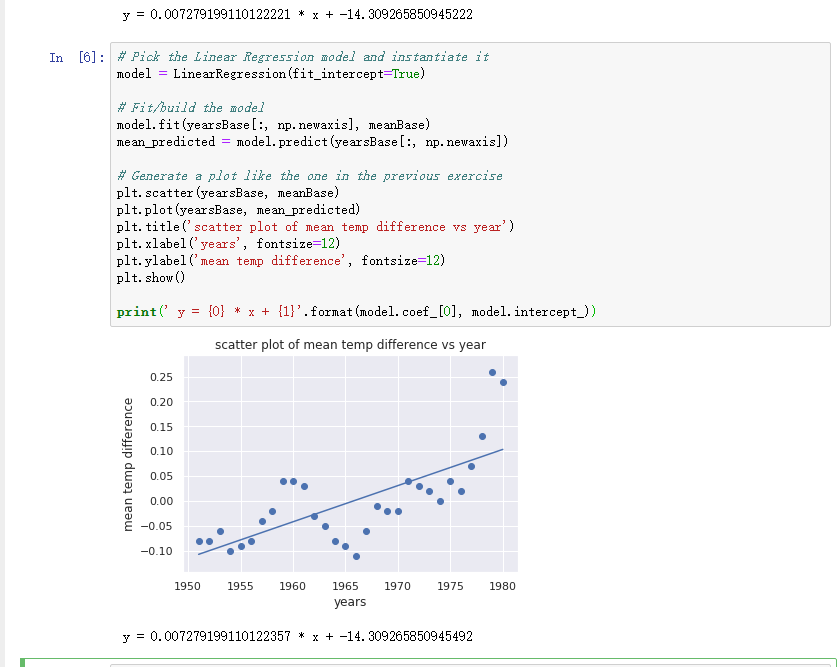
plt.title('scatter plot of mean temp difference vs year')

plt.xlabel('years', fontsize=12)

plt.ylabel('mean temp difference', fontsize=12)

plt.show()

print(' y = {0} \* x + {1}'.format(model.coef\_[0], model.intercept\_))



在scikit learn 中 lineraRegression代替了在numpy在中编写的线性函数

使用Seaborn为第二个数据集（1882-2014）绘图，可以创建一条伴有投影的回归线，通过一个简单的函数调用根据回归显示数据点应位于什么位置。

plt.scatter(years, mean)

plt.title('scatter plot of mean temp difference vs year')

plt.xlabel('years', fontsize=12)

plt.ylabel('mean temp difference', fontsize=12)

sns.regplot(yearsBase, meanBase)

plt.show()

