题目 1：球星投篮命中率预测任务（此题 100 分）

一. 题目概述

给定一份球星投篮的 csv 文件，文件名为“shot-data.csv”，

其中有意义的变量命名如下：

loc_x: 投篮时球星在球场上的横向位置，已经归一化，不用管单位

loc_y: 投篮时球星在球场上的纵向位置，已经归一化，不用管单位

minutes_remaining: 当前节剩余的分钟数，已经归一化，不用管单位

shot_distance: 投篮距离篮筐的距离，已经归一化，不用管单位

shot_made_flag: 投篮是否命中，如果命中，值为 1，否则值为 0

shot_id: 样本唯一标识

现在希望你能够根据该球星的历史数据，建立模型预测该球星的投篮命中情况，并进行可视化。

二. 提交要求

请基于数据集完成以下 5 道小题，按照要求把答案复制到答题卡的指定位置，并提交整个作答过程中的 Notebook 文档（会影响评分）。一共需要提交以下两个文件，请按照命名规则命名。

1. jupyter notebook 运行文件（包括运行结果的运行文件），命名规则为 NOAI1_学生编号.ipynb

（1）不提交 jupyter notebook 文件的，此题计 0 分。

（2）在作答过程中如使用了多个 jupyter notebook，请提供清晰的命名，例如用来做哪道题的。

2. NOAI1 答题卡的.docx 文档，命名规则为 NOAI1_学生编号.docx

三. 参考用库

1. 可以将下表用库直接复制到 notebook 中

```
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib
matplotlib.rcParams['font.family']='simhei'     #保证能够显示中文
matplotlib.rcParams['font.sans-serif']=['simhei']   #保证能够显示英文

%matplotlib inline
```

2. 在整个答题过程中，为了避免卡在代码细节，你可以参考组委会提供的 pandas、numpy、matplotlib、PyTorch 的参考代码，可以根据自己需要进行复制和修改。

四. 题目主体

一共有 5 道小题，每道 20 分。

1. 数据预处理（20 分）

在主程序中实现如下要求。读取投篮文件“shot-data.csv”，首先过滤掉 shot_made_flag 为 nan（值为空）的行，打印行数(num_rows)和列数(num_columns)，并打印列名(column_names)。

提示：

① 用 pd.read_csv(文件名)可以读取.csv 数据文件；

② 用 data = data[~data.xxx.isna()] 命令可以过滤掉 data 中 xxx 列中取值为 nan 对应的行；

请将球星投篮的行数、列数、列名填到答题卡对应位置。

2. 画一个篮球场的半场图（20 分）

提示：此问比较耗时，不作答不影响第 4 问和第 5 问作答，但会影响第 3 问作答。

提示：

将 fig, ax = plt.subplots(figsize=(9.4, 10.0)) 放在图片绘制之前，以确定图片尺寸

为了让场地图大小显示正确，请使用类似如下命令将篮球场尺寸限制。

ax.set_xlim(-250, 250)

ax.set_ylim(-47.5, 422.5)

篮球场的具体要求如下：

所有线宽设定为 2，篮球场整体用黑线表示。注意，线型或者颜色不对，此题 0 分。

（1）创建一个表示篮筐的圆形，位于原点(0,0)，半径为 7.5

（2）创建一个表示篮板的矩形，左下角坐标为(-30, -7.5)，宽度为 60，高度为 1

（3）创建两个表示罚球区的矩形:

① 外框，左下角坐标为(-80, -47.5)，宽度为 160，高度为 190

② 内框，左下角坐标为(-60, -47.5)，宽度为 120，高度为 190

（4）创建两个表示罚球弧的弧线:

① 顶部弧线，中心点在(0, 142.5)，宽度和高度均为 120，从 0 度到 180 度

② 底部弧线，中心点相同，从 180 度到 0 度，虚线样式

（5）创建一个表示限制区的弧线，中心点在(0, 0)，宽度和高度均为 80，从 0 度到 180 度

（6）创建三分线:

① 两个直线部分，分别在左侧(-220, -47.5)和右侧(220, -47.5)，高度均为 140;

② 一个弧线部分，中心点在(0, 0)，宽度和高度均为 475，从 22 度到 158 度

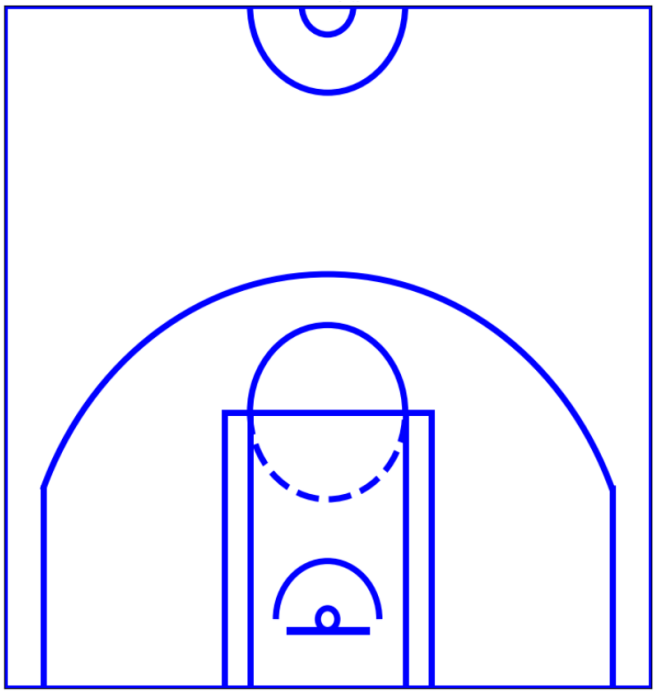
（7）创建两个表示中圈的弧:

① 外弧线，中心点在(0, 422.5)，宽度和高度均为 120，从 180 度到 0 度

② 内弧线，中心点相同，宽度和高度均为 40，从 180 度到 0 度

（8）创建一个表示外边界线的矩形，左下角坐标为(-250, -47.5)，宽度为 500，高度为 470

下图提供了一个篮球场的绘制样例（注意：下图仅为示意图，线宽和颜色与题目要求不一致）效果如下：

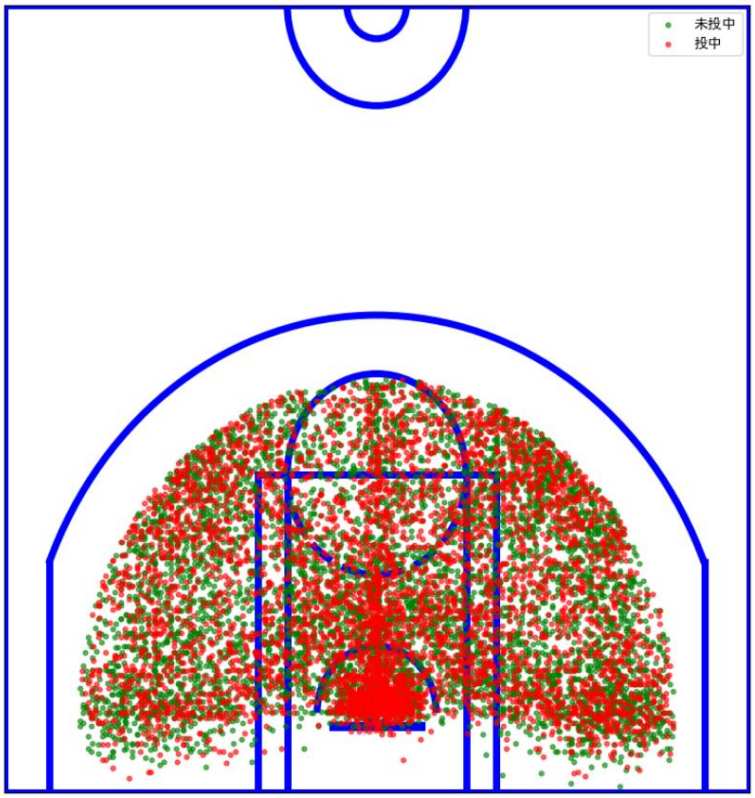


请将绘制好的球场图片复制到答题卡相应位置。

3. 将球星的投篮点绘制在图上（20 分）

在第 2 问的篮球场上，绘**制球星在时间剩余 6 分钟(minutes_remaining<=6)以内及投篮距离>20(shot_distance > 20)时的投篮点**，投中为红色，未投中为绿色。此题点的位置或者颜色不对，得 0 分。

下图提供了一个投篮距离<20（你要绘制的是投篮距离>20）命中位置的绘制样例。



提示：为了让场地图大小显示正确，请使用类似如下命令将篮球场尺寸限制。

如果你在上一问没有把绘图过程封装到函数中，建议部分复制上一问的代码。

ax.set_xlim(-250, 250)

ax.set_ylim(-47.5, 422.5)

请将你绘制好的带有球星投篮位置的图片复制到答题卡相应位置。

4. 使用逻辑斯蒂回归模型预测球星是否能投中（20 分）

模型训练过程和过程如下：

（1）将数据划分为训练集（Training set）和验证集（Validation set）测试集（Testing set），划分比例为 **6:2:2**。

（2）基于投篮数据中的投篮位置（loc_x，loc_y），预测投篮是否命中（shot_made_flag 取值为 1 还是 0）。即 Feature 为投篮位置，Label 为是否命中，这可以看作一个含有 Label 取值为 2 的二分类问题。

（3）使用 PyTorch 中的**逻辑斯蒂回归**模型进行训练和预测。

具体要求如下：

① model：逻辑回归模型，可以看作一个单层线性网络，激活函数为 sigmoid function.

② optimizer：优化器选择 Adam，learning_rate 设定为 0.003

③ criterion：损失函数选择 BCELoss

（4）**训练 500 个 epochs；**

在此过程中可能用到的过程性参考代码如下：

函数① 用于将 Data 中的 Feature 和 Label 分开的函数，输入为 dataframe 格式的数据，输出中 X 表示 Feature 对应的 dataframe，y 表示 label 对应的 dataframe

```
def extract_feature(data):
    X = data[['loc_x', 'loc_y']].values.astype(np.float32)  #提取 Feature
    y = data['shot_made_flag'].values.astype(np.float32)  #提取 Label
    return X, y
```

函数② 用于按比例分割数据集的函数，输入为两个 dataframe，一个是 Feature，一个是 Label，输出为 6 个 dataframe，分表表示训练集（Train）、验证集（Validation）、和测试集（Test），其中 X 表示 Feature，y 表示 Label：

```
def train_val_test_split(X, y, train_size=0.6, val_size=0.2, test_size=0.2, random_state=None):
    assert len(X) == len(y)
    assert train_size + val_size + test_size == 1.0
    if random_state is not None:
        np.random.seed(random_state)
    # 混洗数据
    indices = np.random.permutation(len(X))
    X = X[indices]
    y = y[indices]
    # 计算划分索引
    train_end = int(train_size * len(X))
    val_end = train_end + int(val_size * len(X))
    X_train, y_train = X[:train_end], y[:train_end]
    X_val, y_val = X[train_end:val_end], y[train_end:val_end]
    X_test, y_test = X[val_end:], y[val_end:]
return X_train, X_val, X_test, y_train, y_val, y_test
```

调用函数①和函数②可以实现数据集的分割，参考代码如下

```
X, y = extract_feature(data)
X_train, X_val, X_test, y_train, y_val, y_test = train_val_test_split(X, y, train_size=0.6, val_size=0.2, test_size=0.2, random_state=42) #这里的 random_state 的值可以不用修改
print(f"Training set size: {len(X_train)}")
print(f"Validation set size: {len(X_val)}")
print(f"Test set size: {len(X_test)}")
```

在调用完函数后，可以使用一下命令将训练集、验证集、测试集转为 PyTorch 可以输入的部分

```
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
```

```
y_train_tensor = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)

X_val_tensor = torch.tensor(X_val, dtype=torch.float32)
y_val_tensor = torch.tensor(y_val, dtype=torch.float32).view(-1, 1)

X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)
```

最终你需要编写模型，完成训练和测试过程，需要自己编写的部分如下：

① class LogisticRegressionModel(nn.Module): 模型

② def train 训练过程函数，包括前向训练和反向传播

提示：训练函数需要导入的参数如下（也可以不按照提示做）

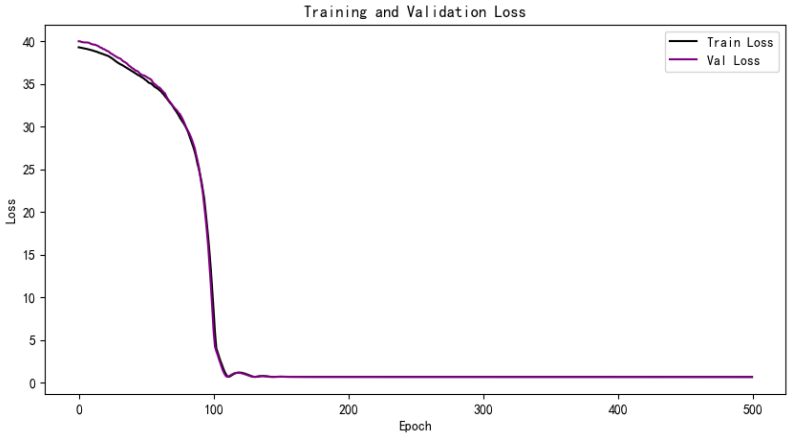
def train(model, optimizer, criterion, X_train_tensor, X_val_tensor, y_train_tensor, y_val_tensor):

③ def draw_loss 函数，绘制损失函数值在 Training set 和 Testing set 随着 epoch 增加的变化过程

最终，在主函数中绘制每个 epoch 之后的训练集（Training set）和验证集（Validation set）的损失函数（Loss Function）的变化情况，其中，Training set 的 Loss 用蓝线表示，Validation set 的 Loss 用红线表示。并在模型训练完成后，导入测试集（Testing Set）并最终打印测试集（Testing set）的测试准确率。

提示，整个过程可以参考组委会提供的 PyTorch 样例代码，直接复制可以使用的部分。

参考效果如下（注意：该图片与你的图片的颜色要求不一样，请注意颜色要求）。



请将下列部分填写到答题卡相应位置

（1）请将模型框架定义的 class 复制到答题卡，展现出如何使用 Pytorch 搭建的模型；

```
class LogisticRegressionModel(nn.Module):
    def __init__(self):
```

（2）最终的损失函数 Loss Function 的变化图；

（3）测试集（Testing Set）中的测试准确率。

注：本题的损失（Loss）和提升准确率（Accuracy）的数值只作为是否按照题目结构要求搭建网络的评分依据，其高低不作为评分依据，不需要纠结于降低损失（Loss）和提升准确率（Accuracy）。

5. 将逻辑斯蒂回归模型改为多层感知机（神经网络）模型，重复上述过程（20 分）

感知机的结构特征如下；

（1）输入为 2 个特征（loc_x，loc_y），输出为 1 个标签（shot_made_flag）；

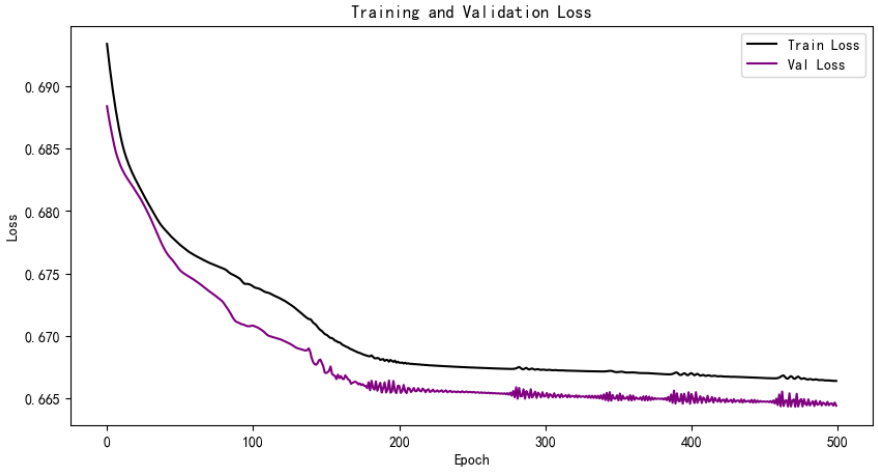
（2）一共包含三个线性层，分别是（2,8），（8,4），（4,1），括号中前一个参数变量代表该层输入大小，后一个参数代表该层输出大小；

（3）三层网络均设定激活函数，分别选用 tanh function，tanh function，和 sigmoid function；

（4）训练过程的损失函数和优化器与上一问一致，依旧是 BCELoss 和 Adam，learning_rate 设定为 0.003。

最终，在主函数中绘制每个 epoch 之后的训练集（Training set）和验证集（Validation set）的损失函数（Loss Function）的变化情况，其中，Training set 的 Loss 用**蓝线**表示，Validation set 的 Loss 用**红线**表示。并在模型训练完成后，导入测试集（Testing Set）并最终打印测试集（Testing set）的测试准确率。

参考效果如下（注意：该图片与你要绘制的图片的颜色要求不一样，请注意颜色要求）。



请将下列部分填写到答题卡相应位置

（1）请将模型框架定义的 class 复制到答题卡，展现出如何使用 Pytorch 搭建的模型；

```
class MLPModel(nn.Module):
    def __init__(self):
```

（2）最终的损失函数 Loss Function 的变化图

（3）测试集（Testing Set）中的测试准确率

注：本题的损失（Loss）和提升准确率（Accuracy）的数值只作为是否按照题目结构要求搭建网络的评分依据，其高低不作为评分依据，不需要纠结于降低损失（Loss）和提升准确率（Accuracy）。