```
#pandas API 参考
import pandas as pd
df = pd.read_csv() pd.read_csv("C.csv")
df[[]].mean(numeric_only=True) 这个代码有问题，需要改为df.mean(numeric_only=True)
df.groupby().mean(numeric_only=True) df.groupby('Status').mean(numeric_only=True)
df.merge(df2, how='', left_on='', right_on='')  (how='left', left_on='Id', right_on='Id')寻找两表中适合的列

#串口  API 参考
import serial
ser = serial.Serial('', 9600, timeout=1)  要查看端口号并修改参数，例如("COM4", 19200, timeout=1)
f = open('', 'wb')  ("A.csv", 'wb')
ser.write(b'')  (b'startA')
while True:
        r = ser.readline()
        if r == b'': r==b'end':
             break
        f.write(r)
        print(r)

f.close()
ser.close()

#torch 参考
import pandas as pd
import numpy as np
import torch
import matplotlib.pyplot as plt

df = pd.read_csv() pd.read_csv("2train.csv')
ds = df.to_numpy()

Y, X = np.split(ds, (    ), axis=1 ) (ds, (1, ), axis=1)
Y = Y.squeeze()
X = X.reshape( () ) X.reshape((-1, 28, 28, 1))
X = X / 255
X = X.transpose( () ) X.transpose(0, 3, 1, 2)
X = X.astype(np.float32)

import torch
from torch.utils.data import DataLoader, random_split

class my_dataset(torch.utils.data.Dataset):
         def __init__(self, ): (self, X, Y):
                pass 删掉pass, 改为self.X, self.Y = X, Y
```

```
    def __getitem__(self, index):
        return 改为return self.X[index], self.Y[index]

    def __len__(self):
        return 改为return len(self.X)

train, test = random_split(my_dataset(), () ) random_split(my_dataset(X, Y), (0.7, 0.3))
data_loader = torch.utils.data.DataLoader(dataset=train,batch_size=64, shuffle=True)
data_loader2 = torch.utils.data.DataLoader(dataset=test,batch_size=64, shuffle=True)

import torch.nn as nn

class MyNet(nn.Module):

    def __init__(self):
        super(MyNet, self).__init__()

        self.conv1 = nn.Conv2d(1, 32, 3)
        self.bn1 = nn.BatchNorm2d(32)
        self.L = nn.Linear(32*26*26, 10)


    def forward(self, x):
        out = self.conv1(x)
        out = self.bn1(out)
        out = torch.relu(out)

        out = torch.flatten(out, start_dim = 1)
        out = self.L(out)

        return out

model = MyNet()

for batchX, batchY in data_loader:
    out = model(batchX)
    break

import torchmetrics

lossf = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
metrics = torchmetrics.Accuracy(task='multiclass', num_classes=10)
```

```
for i in range():
    for batchX, batchY in data_loader:
        score = model(batchX)
        score = torch.squeeze(score)
        loss = lossf(score, batchY)

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        metrics(score, batchY)

    print(loss, metrics.compute())
    metrics.reset()

torch.save(model, '2Model')

for batchX, batchY in data_loader2:
    score = model(batchX)
    metrics(score, batchY)

print(metrics.compute())
metrics.reset()

import pandas as pd
import numpy as np
import torch
import matplotlib.pyplot as plt

df = pd.read_csv('2test.csv')
ds = df.to_numpy()
Y, X = np.split(ds, (1, ), axis=1 )
Y = Y.squeeze()
X = X.reshape( (-1, 28, 28, 1) )
X = X / 255
X = X.transpose( (0, 3, 1, 2) )
X = X.astype(np.float32)

from torch.utils.data import DataLoader, random_split

class my_dataset(torch.utils.data.Dataset): 这部分和上面要改的一样
    def __init__(self, ):
        pass
    def __getitem__(self, index):
        return

    def __len__(self):
        return

data_loader = torch.utils.data.DataLoader(dataset=my_dataset(X, Y),batch_size=64, shuffle=True)

import torch.nn as nn

这部分和上面要改的一样
class MyNet(nn.Module):
    def __init__(self):
        super(MyNet, self).__init__()

        self.conv1 = nn.Conv2d()
        self.bn1 = nn.BatchNorm2d()

        self.L = nn.Linear()
    def forward(self, x):
        out = self.conv1(x)
        out = self.bn1(out)
        out = torch.relu(out)

        out = torch.flatten(out, start_dim = 1)
        out = self.L(out)

        return out

model = MyNet()
model = torch.load('2Model')

import torchmetrics
metrics = torchmetrics.Accuracy(task='multiclass', num_classes=10)

for batchX, batchY in data_loader:
    score = model(batchX)
    metrics(score, batchY)

print(metrics.compute())
metrics.reset()

#取几张图片，让模型分类，把分类的结果写在图片上，图片保存成文件
YP = score.argmax(axis=1)
```

```
X = X * 255
X = X.astype(np.int32)
X = X.transpose( (0, 2, 3, 1) )
for i in range(10):
        plt.imshow(X[i], cmap='gray')
        plt.text(0, 0, YP[i], fontsize=30, color='red')
        plt.savefig(str(i) + '.png')
        plt.close()
print(Y[:10])
```