TensorFlow2.0 线性回归实战（收入与受教育程度的关系）

链接：https://blog.csdn.net/qq\_41627642/article/details/104690186

# 综述

本文使用 **TensorFlow 2.0** 框架，搭建了一个最简单的全连接[神经网络](https://so.csdn.net/so/search?q=%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C&spm=1001.2101.3001.7020" \t "_blank)模型。使用 **ANN（人工神经网络）** 解决了 **收入与受教育程度** 之间的线性关系。

通过本章的学习，你能较清楚的掌握使用 **TensorFlow** 搭建 **DNN（全连接神经网络）**，并解决一系列的一元问题。

编程环境与工具：

| **名称** | **版本** |
| --- | --- |
| 系统 | Windows10 |
| Python版本 | Python 3.7 |
| IDE | jupyter notebook |
| 使用库 | TensorFlow 2.0、Matplotlib、Pandas |

# 数据展示

本文的部分数据如下，

| **Education** | **Income** |
| --- | --- |
| 10 | 26.66 |
| 10.4 | 27.31 |
| 10.84 | 22.13 |
| 11.24 | 21.17 |
| …… | …… |

完整数据与代码可在我的GitHub中找到，[GitHub链接在此](https://github.com/18996677975/DeepLearning-Demo/tree/master/TensorFlow2.0%20%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92%E5%AE%9E%E6%88%98%EF%BC%88%E6%94%B6%E5%85%A5%E4%B8%8E%E5%8F%97%E6%95%99%E8%82%B2%E7%A8%8B%E5%BA%A6%E7%9A%84%E5%85%B3%E7%B3%BB%EF%BC%89)

# 代码与解释

导入库函数

import tensorflow as tf

import pandas as pd

import matplotlib.pyplot as plt

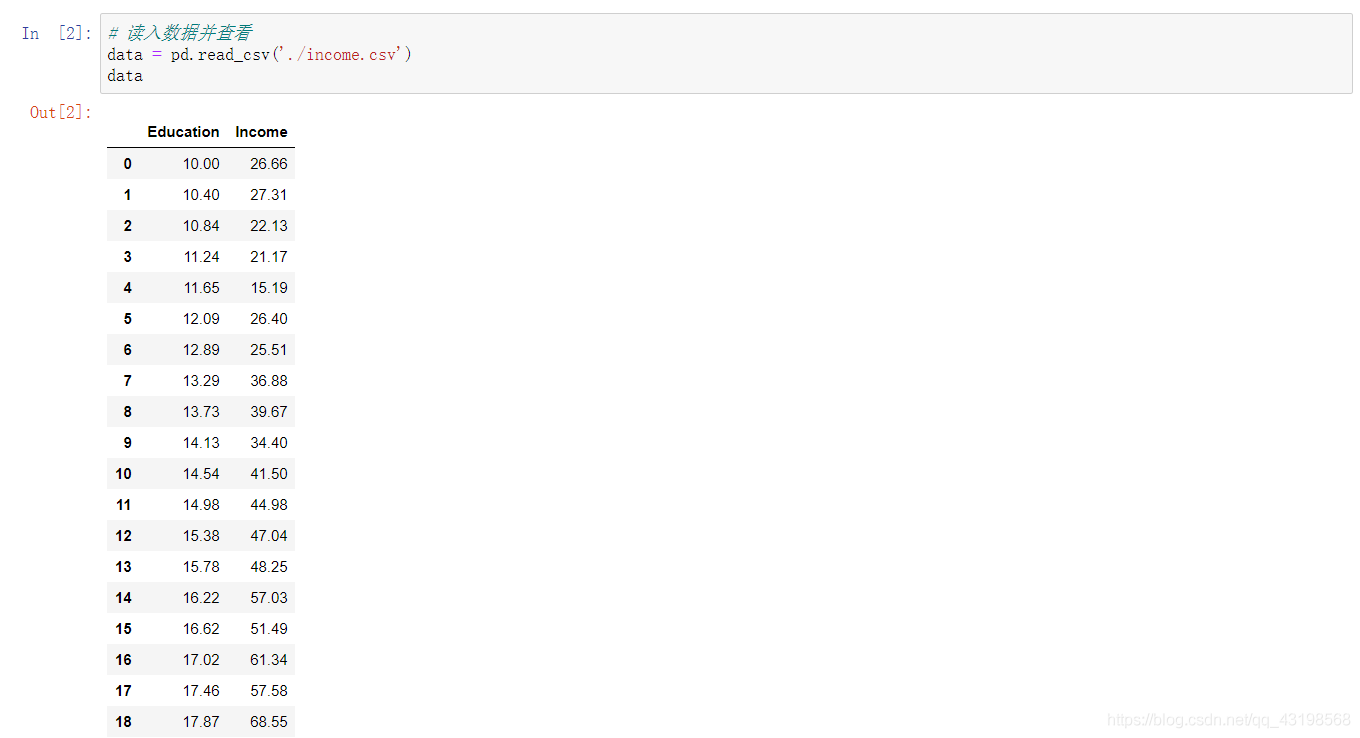
%matplotlib inline

读入数据并查看

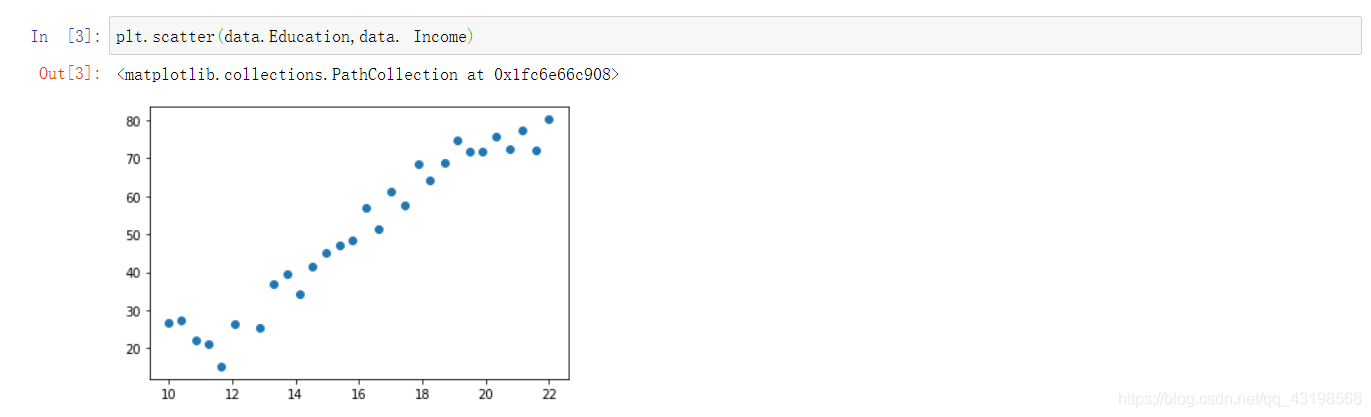
# 读入数据并查看

data = pd.read\_csv('./income.csv')

data

  
画散点图，查看数据分布情况

plt.scatter(data.Education,data. Income)

  
数据赋值

x = data.Education

y = data.Income

搭建网络，并查看网络参数

# 搭建网络

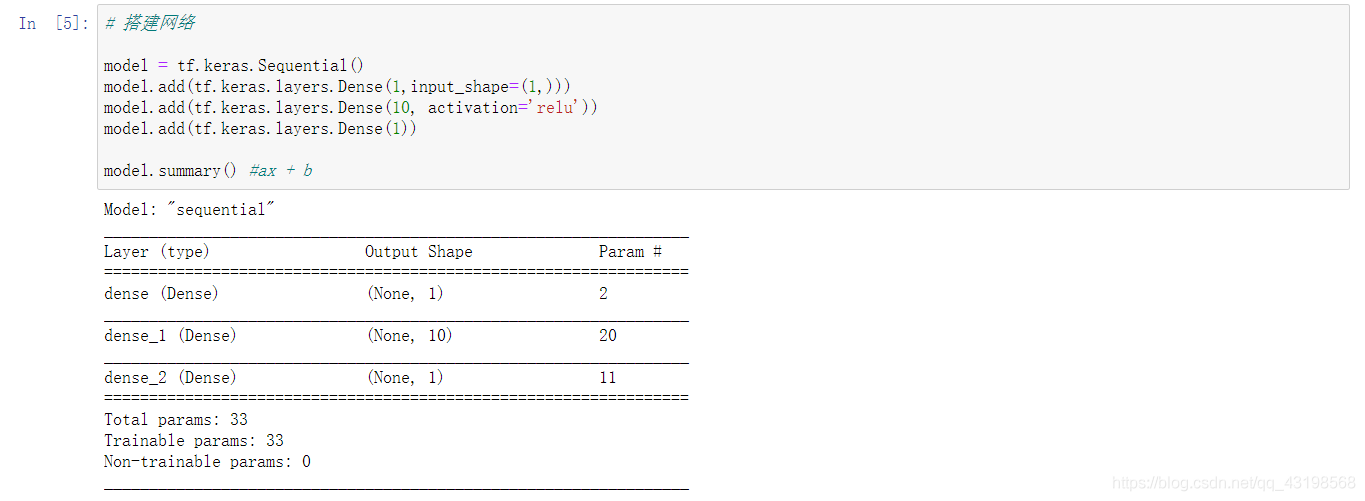
model = tf.keras.Sequential()

model.add(tf.keras.layers.Dense(1,input\_shape=(1,)))

model.add(tf.keras.layers.Dense(10, activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.summary() #ax + b



其中，使用的是顺序模型。网络总三层，第一层结点数为1的全连接层，输入数据维度为1维；第二层结点数为10的全连接层，激活函数为 ReLU；最后一层为结点数为1的全连接层，为输出层。

模型编译

model.compile(optimizer='adam',loss='mse')

* 1

其中，优化器选择为 **adam**，损失函数为 **mse（均方误差）**

模型训练，训练次数为5000次

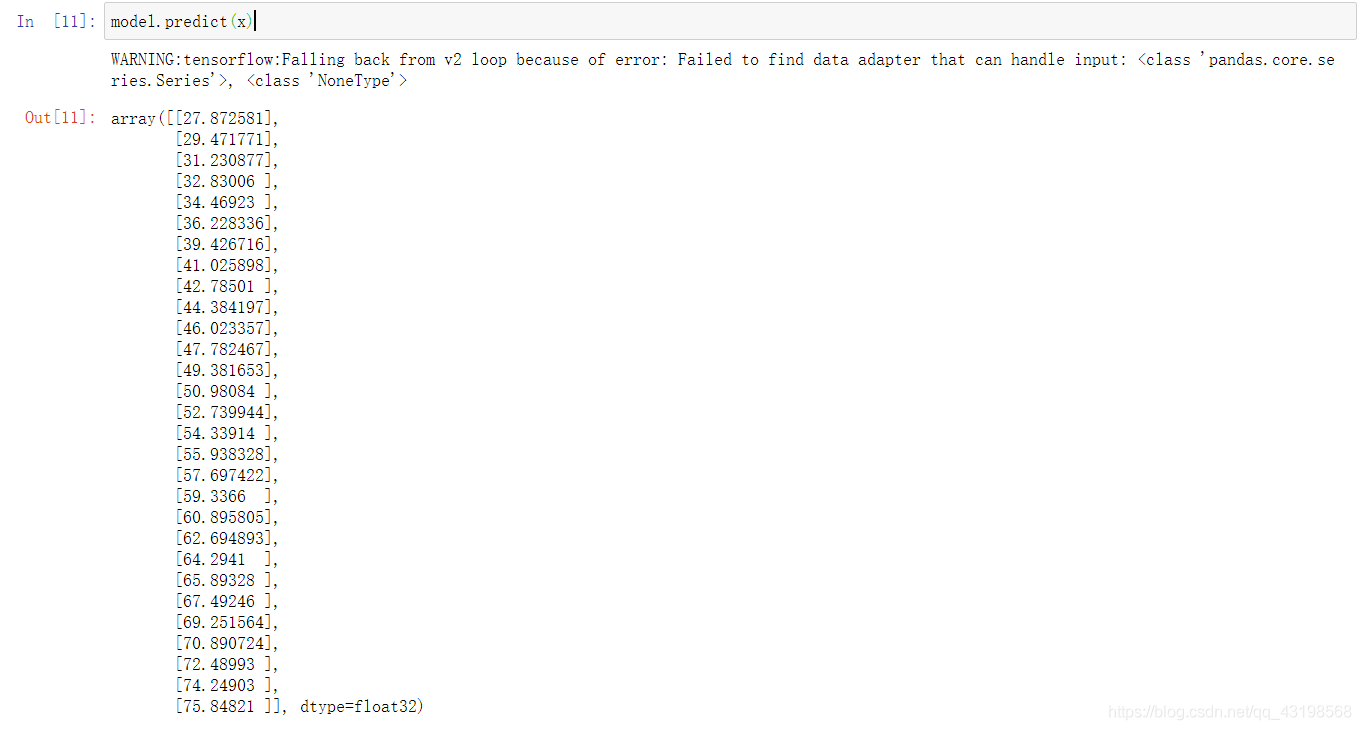
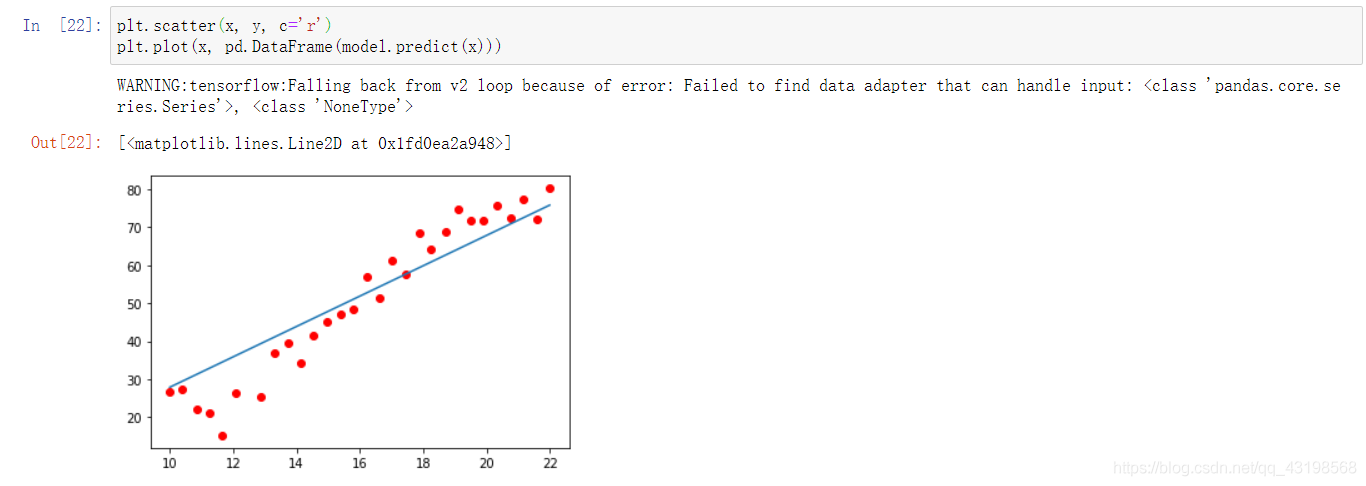
history = model.fit(x, y, epochs=5000)

* 1

以训练后的模型对 **x** 进行测试，得到以后结果

model.predict(x)

* 1

  
最终结果如下  


# 结语

最终可以得到，**收入与受教育程度** 成正比，虽然不乏有特例，但是总体为，受教育程度越高者，收入也越高。

# 必看关键点

# 搭建网络

model = tf.keras.Sequential()

* 1
* 2

每一次搭建网络的第一步就是这样，这叫 顺序模型 或者 序贯模型，对于初学者，你可以不必过多理解，只需要记住，它的功能强大且使用简单，如想要更多的了解它，请自行查找资料。

model.add(tf.keras.layers.Dense(1,input\_shape=(1,)))