TensorFlow实现线性回归：颜值评分作业

In [2]:

```
import pandas as pd # 载入pandas包并命名为pd
MasterFile=pd.read_csv('./FaceScore.csv') # 从当前notebook所在文件夹读入FaceScore.csv并重命名为MasterFile
print(MasterFile.shape) # 输出MasterFile数据集的行数与列数
MasterFile[0:5] # 输出MasterFile的前五条记录
```

(5500, 2)

Out[2]:

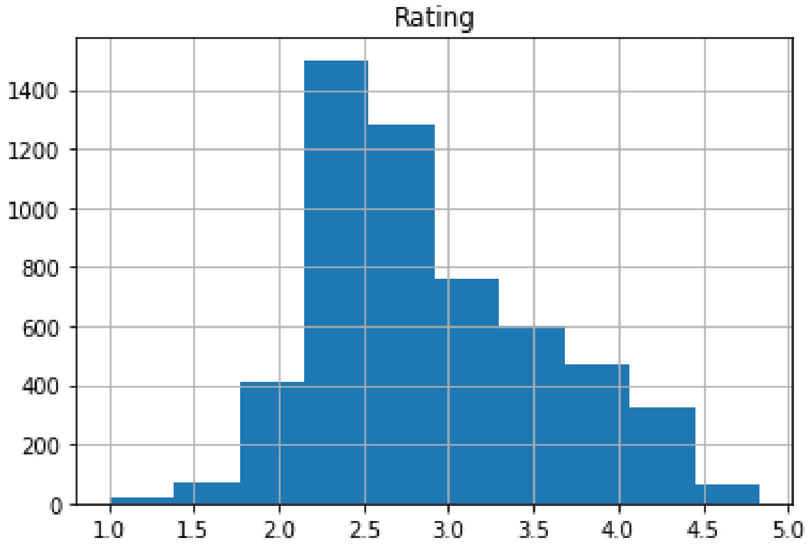
	Filename	Rating
0	ftw1.jpg	4.083333
1	ftw10.jpg	3.666667
2	ftw100.jpg	1.916667
3	ftw101.jpg	2.416667
4	ftw102.jpg	3.166667

In [3]:

```
MasterFile.hist() # 绘制MasterFile中Rating变量的频数分布直方图
```

Out[3]:

```
array([[<AxesSubplot:title={'center':'Rating'}>]], dtype=object)
```



In [4]:

```
import numpy as np # 载入numpy包并重命名为np
FileNames=MasterFile['Filename'] # 将MasterFile数据框中的Filename所在列的数据存在FileNames变量中
N=len(FileNames) # 记录FileNames文件中的记录条数
Y=np.array(MasterFile['Rating']).reshape([N,1]) # 选取MaterFile数据框中的Rating变量对应数据并将其存储形式改为含有N个小list的大list，每个小list只含有一个元素，最后转化为nparray
#Y=(Y-np.mean(Y))/np.std(Y) # 对Y做标准化处理
```

In [5]:

```
from PIL import Image # 从PIL包中载入Image类

IMSIZE=128 # 限定图片大小参数为128像素
X=np.zeros([N,IMSIZE,IMSIZE,3]) # 生成一个N*IMSIZE*IMSIZE*3大小的高维纯0矩阵并命名为X
for i in range(N): # 生成0到N-1的连续正整数
    MyFile=FileNames[i] # 选择FileNames中下标为i的元素记录为MyFile
    Im=Image.open('./images/'+MyFile) # 打开上层文件夹中的images文件夹并取出名为MyFile的图片重命名为Im
    Im=Im.resize([IMSIZE,IMSIZE]) # 将Im的大小调整为128*128
    Im=np.array(Im)/255 # 将Im对应像素矩阵中的0-255的色值元素变换到0-1之间
    X[i,]=Im # 将Im添加到X矩阵第一维度的i位置
```

In [71]:

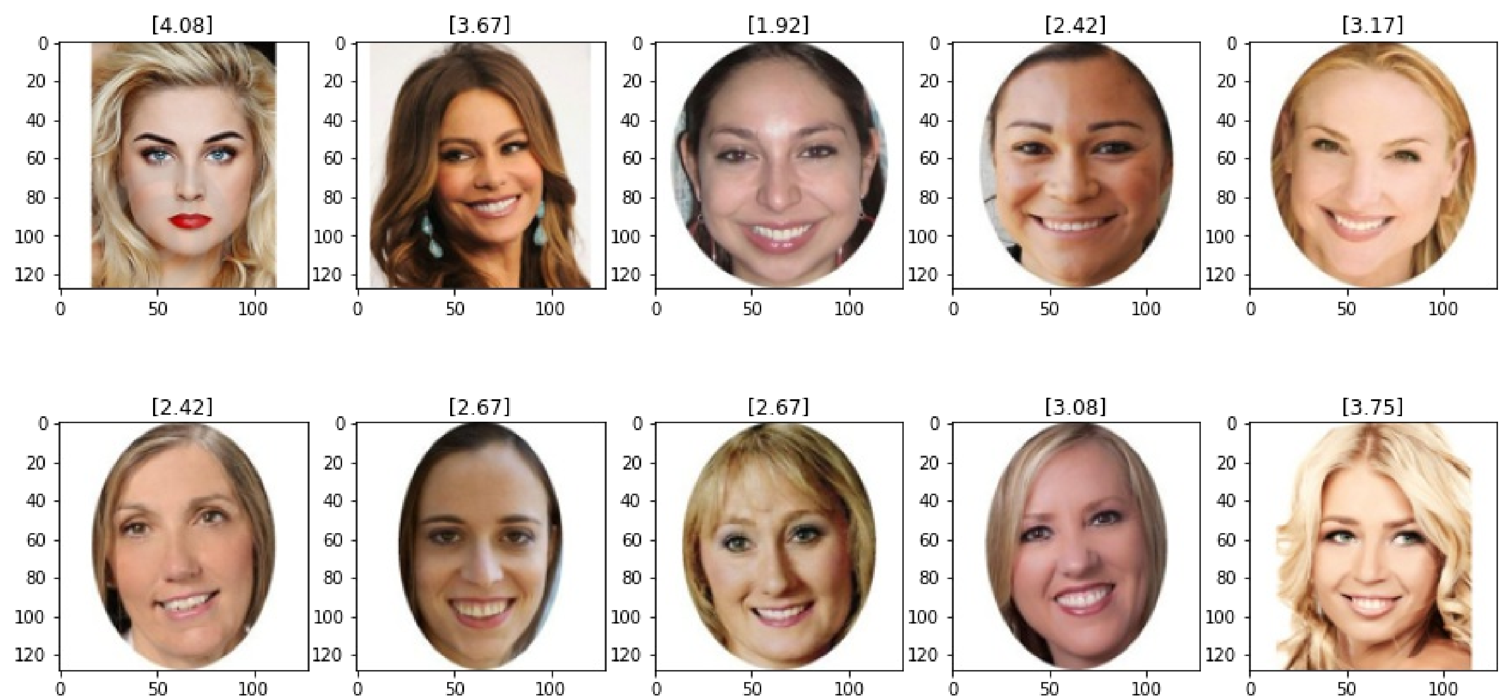
```
X_pred=np.zeros([N,IMSIZE,IMSIZE,3]) # 生成一个N*IMSIZE*IMSIZE*3大小的高维纯0矩阵并命名为X_pred
for i in range(8): # 生成0到7的连续正整数
    name = str(i+1)
    Im=Image.open('./predict_images/'+name+'.png') # 打开上层文件夹中的images文件夹并取出名为MyFile的图片重命名为Im
    Im=Im.resize([IMSIZE,IMSIZE]) # 将Im的大小调整为128*128
    Im=np.array(Im)/255 # 将Im对应像素矩阵中的0-255的色值元素变换到0-1之间
    X_pred[i,]=Im # 将Im添加到X矩阵第一维度的i位置
```

In [6]:

```
from matplotlib import pyplot as plt # 从matplotlib包里载入pyplot类并重命名为plt
plt.figure() # 初始化一个Figure类的实例作为所有绘图元素的最高级容器
fig,ax=plt.subplots(2,5) # 为当前的Figure类实例添加子图，以行数为2，列数为5的排列方式呈现，并返回axes类型变量命名为fig或ax，由于subplot()缺少参数index，此处的axes是整个子图的坐标轴
fig.set_figheight(7.5) # 限制图像高度为7.5
fig.set_figwidth(15) # 限制图像宽度为15
ax=ax.flatten() # 将ax由n*m的Axes组展平成1*nm的Axes组，以便通过下标循环取用
for i in range(10): # 生成0-9的连续正整数
    ax[i].imshow(X[i,:,:,:]) # 将X中第一纬度下标为i的三维色值元素矩阵绘制出来
    ax[i].set_title(np.round(Y[i],2)) # 为axes组中的每一个子图axes设置标题为对应Rating保留两位小数
```

```
d:\anaconda3\envs\tensorflow_gpu_36\lib\site-packages\matplotlib\text.py:1163: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  if s != self._text:

<Figure size 432x288 with 0 Axes>
```



In [7]:

In [7]:

```
from sklearn.model_selection import train_test_split # 从sklearn包中的model_selection模块载入 train_test_split函数
X0,X1,Y0,Y1=train_test_split(X,Y,test_size=0.5,random_state=0) # 按照0.5的比例划分测试集[X1,Y1]与训练集[X0,Y0]
```

In [37]:

```
from keras.layers import Dense, Flatten, Input # 从keras包中的layers模块载入Dense, Flatten, Input类
from keras import Model # 从keras包中载入Model类

input_layer=Input([IMSIZE,IMSIZE,3]) # 用IMSIZE(length),IMSIZE(width),3(channel)初始化输入层对象
x=input_layer # 将input_layer命名为x
x=Flatten()(x) # 将输入层从矩阵形式降维成一维向量
x=Dense(1)(x) # 设置一个输出元素数量为1的全连接层
output_layer=x # 将x重命名为out_layer
model=Model(input_layer,output_layer) # 用input_layer,output_layer来初始化一个model实例
model.summary() # 输出model的基本信息
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_7 (Flatten)          (None, 49152)             0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 49153
=================================================================
Total params: 49,153
Trainable params: 49,153
Non-trainable params: 0
_________________________________________________________________
```

In [38]:

```
from keras.optimizers import Adam # 从keras包的optimizer模块中载入Adam类
model.compile(loss='mse',optimizer=Adam(lr=0.001),metrics=['mse']) # 调用model类的compile方法设置损失函数为均方损失，优化算法为学习率为0.001的Adam算法，设置测试集上的评估标准为均方损失
```

In [39]:

```
model.fit(X0,Y0,validation_data=(X1,Y1),batch_size=100,epochs=100) #训练模型，输入训练集X0，Y0，验证集[X1,Y1]，每批处理100张图片，一共完成100轮批处理训练
```

```
Epoch 1/100
28/28 [==============================] - 6s 228ms/step - loss: 121.2110 - mse: 121.2110 - val_loss: 17.6222 - val_mse: 17.6222
Epoch 2/100
28/28 [==============================] - 1s 28ms/step - loss: 6.3022 - mse: 6.3022 - val_loss: 1.2458 - val_mse: 1.2458
Epoch 3/100
28/28 [==============================] - 1s 26ms/step - loss: 0.9062 - mse: 0.9062 - val_loss: 0.6136 - val_mse: 0.6136
Epoch 4/100
28/28 [==============================] - 1s 27ms/step - loss: 0.4550 - mse: 0.4550 - val_loss: 0.4508 - val_mse: 0.4508
Epoch 5/100
28/28 [==============================] - 1s 26ms/step - loss: 0.4151 - mse: 0.4151 - val_loss: 0.4554 - val_mse: 0.4554
Epoch 6/100
28/28 [==============================] - 1s 26ms/step - loss: 0.3774 - mse: 0.3774 - val_loss: 0.3901 - val_mse: 0.3901
Epoch 7/100
```

```
28/28 [==============================] - 1s 26ms/step - loss: 0.3540 - mse: 0.3540 - val_loss: 0.3642 - v
al_mse: 0.3642
Epoch 8/100
28/28 [==============================] - 1s 26ms/step - loss: 0.3380 - mse: 0.3380 - val_loss: 0.3533 - v
al_mse: 0.3533
Epoch 9/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3388 - mse: 0.3388 - val_loss: 0.3728 - v
al_mse: 0.3728
Epoch 10/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3284 - mse: 0.3284 - val_loss: 0.3755 - v
al_mse: 0.3755
Epoch 11/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3214 - mse: 0.3214 - val_loss: 0.3351 - v
al_mse: 0.3351
Epoch 12/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3107 - mse: 0.3107 - val_loss: 0.3541 - v
al_mse: 0.3541
Epoch 13/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3096 - mse: 0.3096 - val_loss: 0.3530 - v
al_mse: 0.3530
Epoch 14/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3536 - mse: 0.3536 - val_loss: 0.3278 - v
al_mse: 0.3278
Epoch 15/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3102 - mse: 0.3102 - val_loss: 0.4012 - v
al_mse: 0.4012
Epoch 16/100
28/28 [==============================] - 1s 27ms/step - loss: 0.3094 - mse: 0.3094 - val_loss: 0.3233 - v
al_mse: 0.3233
Epoch 17/100
28/28 [==============================] - 1s 27ms/step - loss: 0.3157 - mse: 0.3157 - val_loss: 0.3547 - v
al_mse: 0.3547
Epoch 18/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3158 - mse: 0.3158 - val_loss: 0.3799 - v
al_mse: 0.3799
Epoch 19/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2960 - mse: 0.2960 - val_loss: 0.3228 - v
al_mse: 0.3228
Epoch 20/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2824 - mse: 0.2824 - val_loss: 0.3540 - v
al_mse: 0.3540
Epoch 21/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2816 - mse: 0.2816 - val_loss: 0.3491 - v
al_mse: 0.3491
Epoch 22/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2862 - mse: 0.2862 - val_loss: 0.3398 - v
al_mse: 0.3398
Epoch 23/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2795 - mse: 0.2795 - val_loss: 0.3119 - v
al_mse: 0.3119
Epoch 24/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3242 - mse: 0.3242 - val_loss: 0.3170 - v
al_mse: 0.3170
Epoch 25/100
28/28 [==============================] - 1s 25ms/step - loss: 0.2969 - mse: 0.2969 - val_loss: 0.3373 - v
al_mse: 0.3373
Epoch 26/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2908 - mse: 0.2908 - val_loss: 0.3359 - v
al_mse: 0.3359
Epoch 27/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3336 - mse: 0.3336 - val_loss: 0.4568 - v
al_mse: 0.4568
```

```
Epoch 28/100
28/28 [==============================] - 1s 26ms/step - loss: 0.3775 - mse: 0.3775 - val_loss: 0.3111 - v
al_mse: 0.3111
Epoch 29/100
28/28 [==============================] - 1s 27ms/step - loss: 0.2717 - mse: 0.2717 - val_loss: 0.3224 - v
al_mse: 0.3224
Epoch 30/100
28/28 [==============================] - 1s 29ms/step - loss: 0.2852 - mse: 0.2852 - val_loss: 0.3227 - v
al_mse: 0.3227
Epoch 31/100
28/28 [==============================] - 1s 36ms/step - loss: 0.3350 - mse: 0.3350 - val_loss: 0.4817 - v
al_mse: 0.4817
Epoch 32/100
28/28 [==============================] - 1s 30ms/step - loss: 0.3468 - mse: 0.3468 - val_loss: 0.3229 - v
al_mse: 0.3229
Epoch 33/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2678 - mse: 0.2678 - val_loss: 0.3163 - v
al_mse: 0.3163
Epoch 34/100
28/28 [==============================] - 1s 25ms/step - loss: 0.2814 - mse: 0.2814 - val_loss: 0.3352 - v
al_mse: 0.3352
Epoch 35/100
28/28 [==============================] - 1s 26ms/step - loss: 0.2898 - mse: 0.2898 - val_loss: 0.3384 - v
al_mse: 0.3384
Epoch 36/100
28/28 [==============================] - 1s 29ms/step - loss: 0.2784 - mse: 0.2784 - val_loss: 0.3076 - v
al_mse: 0.3076
Epoch 37/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3295 - mse: 0.3295 - val_loss: 0.4840 - v
al_mse: 0.4840
Epoch 38/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2603 - mse: 0.2603 - val_loss: 0.3220 - v
al_mse: 0.3220
Epoch 39/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2535 - mse: 0.2535 - val_loss: 0.4018 - v
al_mse: 0.4018
Epoch 40/100
28/28 [==============================] - 1s 25ms/step - loss: 0.2779 - mse: 0.2779 - val_loss: 0.3090 - v
al_mse: 0.3090
Epoch 41/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3205 - mse: 0.3205 - val_loss: 0.3142 - v
al_mse: 0.3142
Epoch 42/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2839 - mse: 0.2839 - val_loss: 0.3076 - v
al_mse: 0.3076
Epoch 43/100
28/28 [==============================] - 1s 25ms/step - loss: 0.2775 - mse: 0.2775 - val_loss: 0.4695 - v
al_mse: 0.4695
Epoch 44/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2917 - mse: 0.2917 - val_loss: 0.3679 - v
al_mse: 0.3679
Epoch 45/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2699 - mse: 0.2699 - val_loss: 0.3186 - v
al_mse: 0.3186
Epoch 46/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2920 - mse: 0.2920 - val_loss: 0.3092 - v
al_mse: 0.3092
Epoch 47/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2430 - mse: 0.2430 - val_loss: 0.3258 - v
al_mse: 0.3258
Epoch 48/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3347 - mse: 0.3347 - val_loss: 0.3538 - v
```

```
28/28 [==============================] - 1s 23ms/step - loss: 0.3847 - mse: 0.3847 - val_loss: 0.3538 - v
al_mse: 0.3538
Epoch 49/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2631 - mse: 0.2631 - val_loss: 0.3433 - v
al_mse: 0.3433
Epoch 50/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3464 - mse: 0.3464 - val_loss: 0.3721 - v
al_mse: 0.3721
Epoch 51/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2578 - mse: 0.2578 - val_loss: 0.4220 - v
al_mse: 0.4220
Epoch 52/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2559 - mse: 0.2559 - val_loss: 0.4301 - v
al_mse: 0.4301
Epoch 53/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4865 - mse: 0.4865 - val_loss: 0.5660 - v
al_mse: 0.5660
Epoch 54/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3714 - mse: 0.3714 - val_loss: 0.4648 - v
al_mse: 0.4648
Epoch 55/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2664 - mse: 0.2664 - val_loss: 0.4847 - v
al_mse: 0.4847
Epoch 56/100
28/28 [==============================] - 1s 23ms/step - loss: 0.5140 - mse: 0.5140 - val_loss: 1.0380 - v
al_mse: 1.0380
Epoch 57/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3854 - mse: 0.3854 - val_loss: 0.5994 - v
al_mse: 0.5994
Epoch 58/100
28/28 [==============================] - 1s 23ms/step - loss: 0.5175 - mse: 0.5175 - val_loss: 0.3307 - v
al_mse: 0.3307
Epoch 59/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3718 - mse: 0.3718 - val_loss: 0.4636 - v
al_mse: 0.4636
Epoch 60/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3225 - mse: 0.3225 - val_loss: 0.7949 - v
al_mse: 0.7949
Epoch 61/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3148 - mse: 0.3148 - val_loss: 0.3130 - v
al_mse: 0.3130
Epoch 62/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2539 - mse: 0.2539 - val_loss: 0.3122 - v
al_mse: 0.3122
Epoch 63/100
28/28 [==============================] - 1s 23ms/step - loss: 0.4707 - mse: 0.4707 - val_loss: 0.3194 - v
al_mse: 0.3194
Epoch 64/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3117 - mse: 0.3117 - val_loss: 1.5285 - v
al_mse: 1.5285
Epoch 65/100
28/28 [==============================] - 1s 23ms/step - loss: 0.6381 - mse: 0.6381 - val_loss: 0.9031 - v
al_mse: 0.9031
Epoch 66/100
28/28 [==============================] - 1s 23ms/step - loss: 0.4440 - mse: 0.4440 - val_loss: 0.4893 - v
al_mse: 0.4893
Epoch 67/100
28/28 [==============================] - 1s 24ms/step - loss: 0.5369 - mse: 0.5369 - val_loss: 0.4022 - v
al_mse: 0.4022
Epoch 68/100
28/28 [==============================] - 1s 23ms/step - loss: 1.1537 - mse: 1.1537 - val_loss: 1.1930 - v
al_mse: 1.1930
Epoch 69/100
```

```
Epoch 69/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4718 - mse: 0.4718 - val_loss: 1.1469 - v
al_mse: 1.1469
Epoch 70/100
28/28 [==============================] - 1s 23ms/step - loss: 0.7144 - mse: 0.7144 - val_loss: 0.8159 - v
al_mse: 0.8159
Epoch 71/100
28/28 [==============================] - 1s 23ms/step - loss: 1.0891 - mse: 1.0891 - val_loss: 0.8203 - v
al_mse: 0.8203
Epoch 72/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4381 - mse: 0.4381 - val_loss: 0.4048 - v
al_mse: 0.4048
Epoch 73/100
28/28 [==============================] - 1s 23ms/step - loss: 0.7277 - mse: 0.7277 - val_loss: 0.4734 - v
al_mse: 0.4734
Epoch 74/100
28/28 [==============================] - 1s 23ms/step - loss: 1.0810 - mse: 1.0810 - val_loss: 1.3928 - v
al_mse: 1.3928
Epoch 75/100
28/28 [==============================] - 1s 22ms/step - loss: 0.6862 - mse: 0.6862 - val_loss: 0.4407 - v
al_mse: 0.4407
Epoch 76/100
28/28 [==============================] - 1s 23ms/step - loss: 0.7392 - mse: 0.7392 - val_loss: 0.8241 - v
al_mse: 0.8241
Epoch 77/100
28/28 [==============================] - 1s 26ms/step - loss: 1.2409 - mse: 1.2409 - val_loss: 4.1613 - v
al_mse: 4.1613
Epoch 78/100
28/28 [==============================] - 1s 31ms/step - loss: 1.2048 - mse: 1.2048 - val_loss: 0.6738 - v
al_mse: 0.6738
Epoch 79/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4424 - mse: 0.4424 - val_loss: 0.3500 - v
al_mse: 0.3500
Epoch 80/100
28/28 [==============================] - 1s 23ms/step - loss: 1.1555 - mse: 1.1555 - val_loss: 2.9902 - v
al_mse: 2.9902
Epoch 81/100
28/28 [==============================] - 1s 23ms/step - loss: 1.1777 - mse: 1.1777 - val_loss: 0.9963 - v
al_mse: 0.9963
Epoch 82/100
28/28 [==============================] - 1s 23ms/step - loss: 1.2453 - mse: 1.2453 - val_loss: 1.3101 - v
al_mse: 1.3101
Epoch 83/100
28/28 [==============================] - 1s 26ms/step - loss: 0.7920 - mse: 0.7920 - val_loss: 0.6791 - v
al_mse: 0.6791
Epoch 84/100
28/28 [==============================] - 1s 24ms/step - loss: 0.9833 - mse: 0.9833 - val_loss: 0.4360 - v
al_mse: 0.4360
Epoch 85/100
28/28 [==============================] - 1s 25ms/step - loss: 0.8656 - mse: 0.8656 - val_loss: 0.8352 - v
al_mse: 0.8352
Epoch 86/100
28/28 [==============================] - 1s 35ms/step - loss: 0.9705 - mse: 0.9705 - val_loss: 1.2777 - v
al_mse: 1.2777
Epoch 87/100
28/28 [==============================] - 1s 33ms/step - loss: 1.2071 - mse: 1.2071 - val_loss: 0.3723 - v
al_mse: 0.3723
Epoch 88/100
28/28 [==============================] - 1s 33ms/step - loss: 0.9982 - mse: 0.9982 - val_loss: 0.3478 - v
al_mse: 0.3478
Epoch 89/100
28/28 [==============================] - 1s 32ms/step - loss: 0.7202 - mse: 0.7202 - val_loss: 0.7995 - v
al_mse: 0.7995
```

```
al_mse: 0.7995
Epoch 90/100
28/28 [==============================] - 1s 28ms/step - loss: 0.3307 - mse: 0.3307 - val_loss: 0.3735 - v
al_mse: 0.3735
Epoch 91/100
28/28 [==============================] - 1s 33ms/step - loss: 0.5907 - mse: 0.5907 - val_loss: 0.4968 - v
al_mse: 0.4968
Epoch 92/100
28/28 [==============================] - 1s 25ms/step - loss: 0.9736 - mse: 0.9736 - val_loss: 0.6237 - v
al_mse: 0.6237
Epoch 93/100
28/28 [==============================] - 1s 23ms/step - loss: 0.4401 - mse: 0.4401 - val_loss: 0.5410 - v
al_mse: 0.5410
Epoch 94/100
28/28 [==============================] - 1s 25ms/step - loss: 0.6622 - mse: 0.6622 - val_loss: 0.5406 - v
al_mse: 0.5406
Epoch 95/100
28/28 [==============================] - 1s 24ms/step - loss: 0.5690 - mse: 0.5690 - val_loss: 3.6290 - v
al_mse: 3.6290
Epoch 96/100
28/28 [==============================] - 1s 23ms/step - loss: 1.6157 - mse: 1.6157 - val_loss: 0.3450 - v
al_mse: 0.3450
Epoch 97/100
28/28 [==============================] - 1s 23ms/step - loss: 0.5238 - mse: 0.5238 - val_loss: 0.4859 - v
al_mse: 0.4859
Epoch 98/100
28/28 [==============================] - 1s 23ms/step - loss: 0.9879 - mse: 0.9879 - val_loss: 2.1161 - v
al_mse: 2.1161
Epoch 99/100
28/28 [==============================] - 1s 23ms/step - loss: 0.9819 - mse: 0.9819 - val_loss: 0.4606 - v
al_mse: 0.4606
Epoch 100/100
28/28 [==============================] - 1s 23ms/step - loss: 0.5247 - mse: 0.5247 - val_loss: 1.9373 - v
al_mse: 1.9373
```

Out[39]:

```
<tensorflow.python.keras.callbacks.History at 0x28bb166ec88>
```

In [73]:

```python
# MyPic=(np.array(Image.open('xiongda.jpg').resize((IMSIZE,IMSIZE)))/255).reshape((1,IMSIZE,IMSIZE,3))
# 合并为一行代码
MyPic=Image.open('xiongda.jpg') # 载入熊大图片并命名为Mypic变量
MyPic # 展示Mypic内容
MyPic=MyPic.resize((IMSIZE,IMSIZE)) # 调整Mypic大小为标准大小128*128
MyPic=np.array(MyPic)/255 # 将Mypic图像转化为元素为色素值的三维矩阵并将0-255的取值变换到0-1

MyPic=MyPic.reshape((1,IMSIZE,IMSIZE,3)) # 将Mypic色素值矩阵进一步调整为符合模型输入的四维矩阵

model.predict(MyPic) # 利用训练过的模型为熊大的颜值打分
```

Out[73]:

```
array([[0.3103053]], dtype=float32)
```

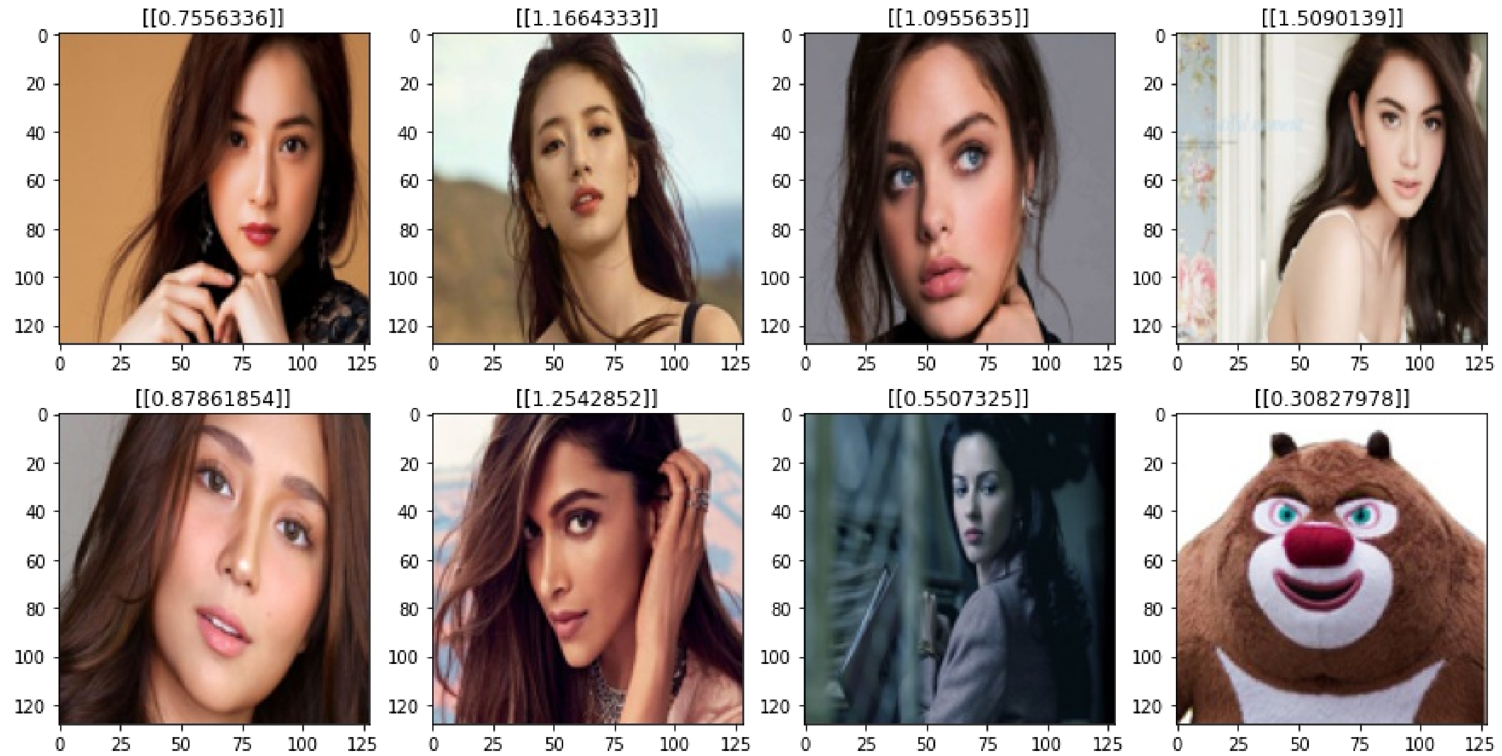
如果给我们朋友们的颜值打分，他们会不高兴

In [72]:

```python
plt.figure() # 初始化一个Figure类的实例作为所有绘图元素的最高级容器
fig,ax=plt.subplots(2,4) # 为当前的Figure类实例添加子图，以行数为2，列数为5的排列方式呈现，并返回axes
类型变量命名为fig或ax，由于subplot()的少参数index，此处的axes是整个子图的坐标轴
```

```
实空变量命名为fig或ax，由于subplot()缺少参数index，此处的axes是整个子图的坐标轴
fig.set_figheight(7.5) # 限制图像高度为7.5
fig.set_figwidth(15) # 限制图像宽度为15
ax=ax.flatten() # 将ax由n*m的Axes组展平成1*nm的Axes组，以便通过下标循环取用
for i in range(8): # 生成0-9的连续正整数
   ax[i].imshow(X_pred[i,:,:,:]) # 将X中第一纬度下标为i的三维色值元素矩阵绘制出来
   ax[i].set_title(model.predict((np.array(X_pred[i,:,:,:]).reshape((1,IMSIZE,IMSIZE,3))))) # 为axes组中的每一个子图axes设置标题为对应Rating保留两位小数
```

<Figure size 432x288 with 0 Axes>



我的预测模型（利用Sequential）

In [74]:

```
from keras.models import Sequential # 我的预测模型

my_model= Sequential([Input([IMSIZE,IMSIZE,3]),Flatten(),Dense(units=1,use_bias=True)]) # 使用sequential容器（有点类似dplyr的管道函数）配置训练模型
my_model.summary() # 输出model的基本信息
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_8 (Flatten)          (None, 49152)             0         
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 49153     
=================================================================
Total params: 49,153
Trainable params: 49,153
Non-trainable params: 0
_________________________________________________________________
```

In [75]:

```
from keras.optimizers import Adam # 从keras包的optimizer模块中载入Adam类
my_model.compile(loss='mse',optimizer=Adam(lr=0.001),metrics=['mse']) # 调用model类的compile方法设置损失函数为均方损失，优化算法为学习率为0.001的Adam算法，设置测试集上的评估标准为均方损失
```

In [76]:

```
my_model.fit(X0,Y0,validation_data=(X1,Y1),batch_size=100,epochs=100) #训练模型，输入训练集X0，Y0，
验证集[X1,Y1]，每批处理100张图片，一共完成100轮批处理训练
```

```
Epoch 1/100
28/28 [==============================] - 3s 93ms/step - loss: 107.4273 - mse: 107.4273 - val_loss: 2.64
00 - val_mse: 2.6400
Epoch 2/100
28/28 [==============================] - 1s 28ms/step - loss: 5.9141 - mse: 5.9141 - val_loss: 2.8223 - v
al_mse: 2.8223
Epoch 3/100
28/28 [==============================] - 1s 26ms/step - loss: 0.8898 - mse: 0.8898 - val_loss: 0.4742 - v
al_mse: 0.4742
Epoch 4/100
28/28 [==============================] - 1s 27ms/step - loss: 0.4438 - mse: 0.4438 - val_loss: 0.4381 - v
al_mse: 0.4381
Epoch 5/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3898 - mse: 0.3898 - val_loss: 0.4092 - v
al_mse: 0.4092
Epoch 6/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3796 - mse: 0.3796 - val_loss: 0.4012 - v
al_mse: 0.4012
Epoch 7/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3662 - mse: 0.3662 - val_loss: 0.4866 - v
al_mse: 0.4866
Epoch 8/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3640 - mse: 0.3640 - val_loss: 0.4186 - v
al_mse: 0.4186
Epoch 9/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3835 - mse: 0.3835 - val_loss: 0.4490 - v
al_mse: 0.4490
Epoch 10/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3377 - mse: 0.3377 - val_loss: 0.3652 - v
al_mse: 0.3652
Epoch 11/100
28/28 [==============================] - 1s 27ms/step - loss: 0.3284 - mse: 0.3284 - val_loss: 0.3583 - v
al_mse: 0.3583
Epoch 12/100
28/28 [==============================] - 1s 28ms/step - loss: 0.3236 - mse: 0.3236 - val_loss: 0.3609 - v
al_mse: 0.3609
Epoch 13/100
28/28 [==============================] - 1s 26ms/step - loss: 0.3111 - mse: 0.3111 - val_loss: 0.3518 - v
al_mse: 0.3518
Epoch 14/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3259 - mse: 0.3259 - val_loss: 0.3439 - v
al_mse: 0.3439
Epoch 15/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3102 - mse: 0.3102 - val_loss: 0.3923 - v
al_mse: 0.3923
Epoch 16/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3504 - mse: 0.3504 - val_loss: 0.3799 - v
al_mse: 0.3799
Epoch 17/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3369 - mse: 0.3369 - val_loss: 0.3402 - v
al_mse: 0.3402
Epoch 18/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3000 - mse: 0.3000 - val_loss: 0.6007 - v
al_mse: 0.6007
Epoch 19/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3955 - mse: 0.3955 - val_loss: 0.4750 - v
al_mse: 0.4750
```

```
al_mse: 0.4750
Epoch 20/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3824 - mse: 0.3824 - val_loss: 0.4376 - v
al_mse: 0.4376
Epoch 21/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3172 - mse: 0.3172 - val_loss: 0.3295 - v
al_mse: 0.3295
Epoch 22/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2805 - mse: 0.2805 - val_loss: 0.3645 - v
al_mse: 0.3645
Epoch 23/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2809 - mse: 0.2809 - val_loss: 0.3308 - v
al_mse: 0.3308
Epoch 24/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2877 - mse: 0.2877 - val_loss: 0.3252 - v
al_mse: 0.3252
Epoch 25/100
28/28 [==============================] - 1s 23ms/step - loss: 0.3244 - mse: 0.3244 - val_loss: 0.4499 - v
al_mse: 0.4499
Epoch 26/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2774 - mse: 0.2774 - val_loss: 0.3494 - v
al_mse: 0.3494
Epoch 27/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2914 - mse: 0.2914 - val_loss: 0.3444 - v
al_mse: 0.3444
Epoch 28/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3790 - mse: 0.3790 - val_loss: 0.3951 - v
al_mse: 0.3951
Epoch 29/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4608 - mse: 0.4608 - val_loss: 0.3342 - v
al_mse: 0.3342
Epoch 30/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4024 - mse: 0.4024 - val_loss: 0.5568 - v
al_mse: 0.5568
Epoch 31/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3503 - mse: 0.3503 - val_loss: 0.3371 - v
al_mse: 0.3371
Epoch 32/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3005 - mse: 0.3005 - val_loss: 0.3405 - v
al_mse: 0.3405
Epoch 33/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2730 - mse: 0.2730 - val_loss: 0.3298 - v
al_mse: 0.3298
Epoch 34/100
28/28 [==============================] - 1s 23ms/step - loss: 0.2661 - mse: 0.2661 - val_loss: 0.3326 - v
al_mse: 0.3326
Epoch 35/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2642 - mse: 0.2642 - val_loss: 0.3842 - v
al_mse: 0.3842
Epoch 36/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2768 - mse: 0.2768 - val_loss: 0.7570 - v
al_mse: 0.7570
Epoch 37/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3279 - mse: 0.3279 - val_loss: 0.4890 - v
al_mse: 0.4890
Epoch 38/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2962 - mse: 0.2962 - val_loss: 0.3921 - v
al_mse: 0.3921
Epoch 39/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2906 - mse: 0.2906 - val_loss: 0.3908 - v
al_mse: 0.3908
Epoch 40/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2907 - mse: 0.2907 - val_loss: 0.5781 - v
```

```
28/28 [==============================] - 1s 24ms/step - loss: 0.2907 - mse: 0.2907 - val_loss: 0.5781 - v
al_mse: 0.5781
Epoch 41/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3173 - mse: 0.3173 - val_loss: 0.3265 - v
al_mse: 0.3265
Epoch 42/100
28/28 [==============================] - 1s 30ms/step - loss: 0.2645 - mse: 0.2645 - val_loss: 0.3354 - v
al_mse: 0.3354
Epoch 43/100
28/28 [==============================] - 1s 28ms/step - loss: 0.2978 - mse: 0.2978 - val_loss: 0.4445 - v
al_mse: 0.4445
Epoch 44/100
28/28 [==============================] - 1s 27ms/step - loss: 0.2708 - mse: 0.2708 - val_loss: 0.3614 - v
al_mse: 0.3614
Epoch 45/100
28/28 [==============================] - 1s 28ms/step - loss: 0.4314 - mse: 0.4314 - val_loss: 0.7607 - v
al_mse: 0.7607
Epoch 46/100
28/28 [==============================] - 1s 30ms/step - loss: 0.3775 - mse: 0.3775 - val_loss: 0.4197 - v
al_mse: 0.4197
Epoch 47/100
28/28 [==============================] - 1s 25ms/step - loss: 0.2962 - mse: 0.2962 - val_loss: 0.3172 - v
al_mse: 0.3172
Epoch 48/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2971 - mse: 0.2971 - val_loss: 0.4526 - v
al_mse: 0.4526
Epoch 49/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3679 - mse: 0.3679 - val_loss: 0.3150 - v
al_mse: 0.3150
Epoch 50/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2684 - mse: 0.2684 - val_loss: 0.6752 - v
al_mse: 0.6752
Epoch 51/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3217 - mse: 0.3217 - val_loss: 0.3154 - v
al_mse: 0.3154
Epoch 52/100
28/28 [==============================] - 1s 25ms/step - loss: 0.4998 - mse: 0.4998 - val_loss: 0.4881 - v
al_mse: 0.4881
Epoch 53/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2845 - mse: 0.2845 - val_loss: 0.4428 - v
al_mse: 0.4428
Epoch 54/100
28/28 [==============================] - 1s 27ms/step - loss: 0.2687 - mse: 0.2687 - val_loss: 0.3213 - v
al_mse: 0.3213
Epoch 55/100
28/28 [==============================] - 1s 25ms/step - loss: 0.4996 - mse: 0.4996 - val_loss: 0.3489 - v
al_mse: 0.3489
Epoch 56/100
28/28 [==============================] - 1s 30ms/step - loss: 0.4114 - mse: 0.4114 - val_loss: 0.4538 - v
al_mse: 0.4538
Epoch 57/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3045 - mse: 0.3045 - val_loss: 0.7987 - v
al_mse: 0.7987
Epoch 58/100
28/28 [==============================] - 1s 28ms/step - loss: 0.4881 - mse: 0.4881 - val_loss: 0.4096 - v
al_mse: 0.4096
Epoch 59/100
28/28 [==============================] - 1s 28ms/step - loss: 0.7144 - mse: 0.7144 - val_loss: 0.3284 - v
al_mse: 0.3284
Epoch 60/100
28/28 [==============================] - 1s 25ms/step - loss: 0.3660 - mse: 0.3660 - val_loss: 1.2868 - v
al_mse: 1.2868
Epoch 61/100
```

```
Epoch 61/100
28/28 [==============================] - 1s 25ms/step - loss: 0.6644 - mse: 0.6644 - val_loss: 0.3505 - v
al_mse: 0.3505
Epoch 62/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3928 - mse: 0.3928 - val_loss: 0.5139 - v
al_mse: 0.5139
Epoch 63/100
28/28 [==============================] - 1s 25ms/step - loss: 0.7906 - mse: 0.7906 - val_loss: 0.4132 - v
al_mse: 0.4132
Epoch 64/100
28/28 [==============================] - 1s 24ms/step - loss: 0.9307 - mse: 0.9307 - val_loss: 1.2611 - v
al_mse: 1.2611
Epoch 65/100
28/28 [==============================] - 1s 25ms/step - loss: 0.9829 - mse: 0.9829 - val_loss: 0.4161 - v
al_mse: 0.4161
Epoch 66/100
28/28 [==============================] - 1s 27ms/step - loss: 0.4647 - mse: 0.4647 - val_loss: 0.4190 - v
al_mse: 0.4190
Epoch 67/100
28/28 [==============================] - 1s 26ms/step - loss: 0.3406 - mse: 0.3406 - val_loss: 2.1776 - v
al_mse: 2.1776
Epoch 68/100
28/28 [==============================] - 1s 26ms/step - loss: 0.5506 - mse: 0.5506 - val_loss: 1.1426 - v
al_mse: 1.1426
Epoch 69/100
28/28 [==============================] - 1s 24ms/step - loss: 1.0515 - mse: 1.0515 - val_loss: 1.9352 - v
al_mse: 1.9352
Epoch 70/100
28/28 [==============================] - 1s 24ms/step - loss: 2.4646 - mse: 2.4646 - val_loss: 1.6372 - v
al_mse: 1.6372
Epoch 71/100
28/28 [==============================] - 1s 24ms/step - loss: 2.1913 - mse: 2.1913 - val_loss: 2.2914 - v
al_mse: 2.2914
Epoch 72/100
28/28 [==============================] - 1s 24ms/step - loss: 0.6282 - mse: 0.6282 - val_loss: 0.7847 - v
al_mse: 0.7847
Epoch 73/100
28/28 [==============================] - 1s 24ms/step - loss: 0.2954 - mse: 0.2954 - val_loss: 0.3190 - v
al_mse: 0.3190
Epoch 74/100
28/28 [==============================] - 1s 25ms/step - loss: 0.8439 - mse: 0.8439 - val_loss: 0.9897 - v
al_mse: 0.9897
Epoch 75/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4457 - mse: 0.4457 - val_loss: 0.9127 - v
al_mse: 0.9127
Epoch 76/100
28/28 [==============================] - 1s 23ms/step - loss: 0.5321 - mse: 0.5321 - val_loss: 0.3299 - v
al_mse: 0.3299
Epoch 77/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4280 - mse: 0.4280 - val_loss: 0.8650 - v
al_mse: 0.8650
Epoch 78/100
28/28 [==============================] - 1s 24ms/step - loss: 1.6704 - mse: 1.6704 - val_loss: 0.5772 - v
al_mse: 0.5772
Epoch 79/100
28/28 [==============================] - 1s 23ms/step - loss: 0.4194 - mse: 0.4194 - val_loss: 0.4003 - v
al_mse: 0.4003
Epoch 80/100
28/28 [==============================] - 1s 24ms/step - loss: 0.8833 - mse: 0.8833 - val_loss: 0.5307 - v
al_mse: 0.5307
Epoch 81/100
28/28 [==============================] - 1s 24ms/step - loss: 2.8574 - mse: 2.8574 - val_loss: 0.9742 - v
```

```
al_mse: 0.9742
Epoch 82/100
28/28 [==============================] - 1s 24ms/step - loss: 2.1058 - mse: 2.1058 - val_loss: 1.4449 - v
al_mse: 1.4449
Epoch 83/100
28/28 [==============================] - 1s 24ms/step - loss: 1.1015 - mse: 1.1015 - val_loss: 1.4230 - v
al_mse: 1.4230
Epoch 84/100
28/28 [==============================] - 1s 24ms/step - loss: 0.6621 - mse: 0.6621 - val_loss: 0.3657 - v
al_mse: 0.3657
Epoch 85/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4082 - mse: 0.4082 - val_loss: 0.5416 - v
al_mse: 0.5416
Epoch 86/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3594 - mse: 0.3594 - val_loss: 0.3514 - v
al_mse: 0.3514
Epoch 87/100
28/28 [==============================] - 1s 24ms/step - loss: 0.5013 - mse: 0.5013 - val_loss: 0.8550 - v
al_mse: 0.8550
Epoch 88/100
28/28 [==============================] - 1s 24ms/step - loss: 1.3684 - mse: 1.3684 - val_loss: 0.6653 - v
al_mse: 0.6653
Epoch 89/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3342 - mse: 0.3342 - val_loss: 0.3568 - v
al_mse: 0.3568
Epoch 90/100
28/28 [==============================] - 1s 24ms/step - loss: 0.4057 - mse: 0.4057 - val_loss: 0.7849 - v
al_mse: 0.7849
Epoch 91/100
28/28 [==============================] - 1s 25ms/step - loss: 0.5115 - mse: 0.5115 - val_loss: 0.3435 - v
al_mse: 0.3435
Epoch 92/100
28/28 [==============================] - 1s 26ms/step - loss: 0.3283 - mse: 0.3283 - val_loss: 0.3474 - v
al_mse: 0.3474
Epoch 93/100
28/28 [==============================] - 1s 27ms/step - loss: 0.4266 - mse: 0.4266 - val_loss: 0.5525 - v
al_mse: 0.5525
Epoch 94/100
28/28 [==============================] - 1s 24ms/step - loss: 0.3482 - mse: 0.3482 - val_loss: 0.6396 - v
al_mse: 0.6396
Epoch 95/100
28/28 [==============================] - 1s 25ms/step - loss: 0.7979 - mse: 0.7979 - val_loss: 0.6573 - v
al_mse: 0.6573
Epoch 96/100
28/28 [==============================] - 1s 25ms/step - loss: 3.0829 - mse: 3.0829 - val_loss: 2.3367 - v
al_mse: 2.3367
Epoch 97/100
28/28 [==============================] - 1s 24ms/step - loss: 1.4284 - mse: 1.4284 - val_loss: 0.5517 - v
al_mse: 0.5517
Epoch 98/100
28/28 [==============================] - 1s 24ms/step - loss: 0.7573 - mse: 0.7573 - val_loss: 0.3603 - v
al_mse: 0.3603
Epoch 99/100
28/28 [==============================] - 1s 24ms/step - loss: 0.6350 - mse: 0.6350 - val_loss: 0.6866 - v
al_mse: 0.6866
Epoch 100/100
28/28 [==============================] - 1s 23ms/step - loss: 0.5546 - mse: 0.5546 - val_loss: 1.6267 - v
al_mse: 1.6267
```

Out[76]:

```
<tensorflow.python.keras.callbacks.History at 0x28adc9b6470>
```

In [78]:

```
plt.figure() # 初始化一个Figure类的实例作为所有绘图元素的最高级容器
fig,ax=plt.subplots(2,4) # 为当前的Figure类实例添加子图，以行数为2，列数为5的排列方式呈现，并返回axes类型变量命名为fig或ax，由于subplot()缺少参数index，此处的axes是整个子图的坐标轴
fig.set_figheight(7.5) # 限制图像高度为7.5
fig.set_figwidth(15) # 限制图像宽度为15
ax=ax.flatten() # 将ax由n*m的Axes组展平成1*nm的Axes组，以便通过下标循环取用
for i in range(8): # 生成0-9的连续正整数
    ax[i].imshow(X_pred[i,:,:,:]) # 将X中第一纬度下标为i的三维色值元素矩阵绘制出来
    ax[i].set_title(my_model.predict((np.array(X_pred[i,:,:,:]).reshape((1,IMSIZE,IMSIZE,3))))) # 为axes组中的每一个子图axes设置标题为对应Rating保留两位小数
```

<Figure size 432x288 with 0 Axes>

In [ ]:

```
```