1.查看原始数据类型

In [ ]:
```
import pandas as pd
path = './主播数据明细.xlsx'
data = pd.read_excel(path,sheet_name='主播数据明细')
modeltitle = pd.read_excel(path,sheet_name='模型字段')
print(data.dtypes)#查看字段列表
# print(data.isnull().values.any())#false没有空值,true存在空值
```

```
主播抖音号                object
主播昵称                object
月均直播时长              float64
月均直播间曝光人数           float64
月均直播间曝光次数           float64
月均直播间观看人数           float64
月均直播间观看次数           float64
直播间曝光观看人数转化率        float64
直播间曝光观看次数转化率        float64
最高在线人数              float64
平均在线人数              float64
人均观看时长分钟            float64
月均评论次数              float64
月均新加直播团人数           float64
月均新增粉丝数             float64
月均取关粉丝数             float64
观看老粉占比              float64
带货商品数                int64
月均直播间商品曝光人数         float64
直播间商品曝光率人数          float64
月均直播间商品点击人数         float64
直播间商品点击率人数          float64
月均直播间商品曝光次数         float64
直播间商品曝光率次数          float64
月均直播间商品点击次数         float64
直播间商品点击率次数          float64
月均直播间成交订单数          float64
年直播间成交金额            float64
月均直播间成交金额           float64
GPM                 float64
月均直播间成交件数           float64
直播间商品点击成交转化率次数      float64
直播间曝光成交转化率次数        float64
月均直播间成交人数           float64
直播间商品点击成交转化率人数      float64
直播间曝光成交转化率人数        float64
月均直播间退款订单数          float64
月均直播间退款金额           float64
月均直播间退款人数           float64
月均预估佣金支出            float64
dtype: object
```

2.建立模型

In [ ]:
```
rowindex = data.columns.to_list# 提取列名
# print(rowindex)

#对列名分组归类
onetitle = ['交易指标','支出指标','人气指标','人数转化','次数转化']
onetwotitle = {}
for i in onetitle:
    uselist = []
    for j in range(len(modeltitle['字段划分'])):
        if modeltitle['字段划分'][j] == i:
            uselist.append(modeltitle['列名'][j])
    onetwotitle[i] = uselist
for item in onetwotitle.items():
    print(item)
```

```
('交易指标', ['年直播间成交金额', '月均直播间成交金额', 'GPM'])
('支出指标', ['月均预估佣金支出', '月均直播间退款金额', '月均直播间退款人数', '月均直播间退款订单数'])
('人气指标', ['月均直播时长', '最高在线人数', '平均在线人数', '人均观看时长分钟', '月均评论次数', '月均新加直播团人数', '月均新增粉丝数', '月均取关粉丝数', '观看老粉占比'])
('人数转化', ['月均直播间曝光人数', '月均直播间观看人数', '直播间曝光观看人数转化率', '月均直播间商品曝光人数', '直播间商品曝光率人数', '月均直播间商品点击人数', '直播间商品点击率人数', '月均直播间成交人数', '直播间商品点击成交转化率人数', '直播间曝光成交转化率人数'])
('次数转化', ['月均直播间曝光次数', '月均直播间观看次数', '直播间曝光观看次数转化率', '月均直播间商品曝光次数', '直播间商品曝光率次数', '月均直播间商品点击次数', '直播间商品点击率次数', '月均直播间成交件数', '直播间商品点击成交转化率次数', '直播间曝光成交转化率次数'])
```

3.创建模型核心算法类

In [ ]:

```
import numpy as np
class AHP:
    # 矩阵参数计算
    def __init__(self, array):
        # 记录矩阵相关信息
        self.array = array
        # 记录矩阵大小
        self.n = array.shape[0]
        # 初始化RI值，用于一致性检验
        self.RI_list = [0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49, 1.51, 1.54, 1.56, 1.58, 1.59]
        # 矩阵的特征值和特征向量
        self.eig_val, self.eig_vector = np.linalg.eig(self.array)
        # 矩阵的最大特征值
        self.max_eig_val = np.max(self.eig_val)
        # 矩阵最大特征值对应的特征向量
        self.max_eig_vector = self.eig_vector[:, np.argmax(self.eig_val)].real
        # 矩阵的一致性指标CI
        self.CI_val = (self.max_eig_val - self.n) / (self.n - 1)
        # 矩阵的一致性比例CR
        self.CR_val = self.CI_val / (self.RI_list[self.n - 1])

    #一致性检验
    def test_consist(self):
        # 打印矩阵的一致性指标CI和一致性比例CR
        # print("判断矩阵的CI值为：" + str(self.CI_val))
        # print("判断矩阵的CR值为：" + str(self.CR_val))
        # 进行一致性检验判断
        if self.n == 2:  # 当只有两个子因素的情况
            print("仅包含两个子因素，不存在一致性问题")
        else:
            if self.CR_val < 0.1:  # CR值小于0.1，可以通过一致性检验
                print("判断矩阵的CR值为" + str(self.CR_val) + ",通过一致性检验")
                return True
            else:  # CR值大于0.1，一致性检验不通过
                print("判断矩阵的CR值为" + str(self.CR_val) + "未通过一致性检验")
                return False

    #算术平均法求权重
    def cal_weight_by_arithmetic_method(self):
        # 求矩阵的每列的和
        col_sum = np.sum(self.array, axis=0)
        # 将判断矩阵按照列归一化
        array_normed = self.array / col_sum
        # 计算权重向量
        array_weight = np.sum(array_normed, axis=1) / self.n
        # 打印权重向量
        print("算术平均法计算得到的权重向量为：\n", array_weight)
        # 返回权重向量的值
        return array_weight

    #几何平均法求权重
    def cal_weight__by_geometric_method(self):
        # 求矩阵的每列的积
        col_product = np.product(self.array, axis=1)
        # 将得到的积向量的每个分量进行开n次方
        array_power = np.power(col_product, 1 / self.n)
        # 将列向量归一化
        array_weight = array_power / np.sum(array_power)
        # 打印权重向量
        print("几何平均法计算得到的权重向量为：\n", array_weight)
        # 返回权重向量的值
        return array_weight

    #特征值法求权重
    def cal_weight__by_eigenvalue_method(self):
        # 将矩阵最大特征值对应的特征向量进行归一化处理就得到了权重
        array_weight = self.max_eig_vector / np.sum(self.max_eig_vector)
        # 打印权重向量
        print("特征值法计算得到的权重向量为：\n", array_weight)
        # 返回权重向量的值
        return array_weight
```

4.创建分组判断矩阵

In [ ]:
```
# 与列名分组一一对应
# onetitle
onematrix = np.array([[1, 3, 5, 6, 7],
                      [1/3, 1, 3, 5, 6],
                      [1/5, 1/3, 1, 3, 5],
                      [1/6, 1/5, 1/3, 1, 3],
                      [1/7, 1/6, 1/5, 1/3, 1]])
# 交易指标
twomatrix = np.array([[1, 2, 3],
                      [1/2, 1, 2],
                      [1/3, 1/2, 1]])
# 支出指标
threematrix = np.array([[1, 3, 4, 5],
                        [1/3, 1, 3, 4],
                        [1/4, 1/3, 1, 3],
                        [1/5, 1/4, 1/3, 1]])
# 人气指标
fourmatrix = np.array([[1, 2, 3, 4, 6, 7, 9, 10, 12],
                       [1/2, 1, 2, 3, 4, 6, 7, 9, 10],
                       [1/3, 1/2, 1, 2, 3, 4, 6, 7, 9],
                       [1/4, 1/3, 1/2, 1, 2, 3, 4, 6, 7],
                       [1/6, 1/4, 1/3, 1/2, 1, 2, 3, 4, 6],
                       [1/7, 1/6, 1/4, 1/3, 1/2, 1, 2, 3, 4],
                       [1/9, 1/7, 1/6, 1/4, 1/3, 1/2, 1, 2, 3],
                       [1/10, 1/9, 1/7, 1/6, 1/4, 1/3, 1/2, 1, 2],
                       [1/12, 1/10, 1/9, 1/7, 1/6, 1/4, 1/3, 1/2, 1]])
# 人数转化
fivematrix = np.array([[1, 2, 3, 5, 6, 7, 8, 9, 11, 13],
                       [1/2, 1, 2, 3, 5, 6, 7, 8, 9, 11],
                       [1/3, 1/2, 1, 2, 3, 5, 6, 7, 8, 9],
                       [1/5, 1/3, 1/2, 1, 2, 3, 5, 6, 7, 8],
                       [1/6, 1/5, 1/3, 1/2, 1, 2, 3, 5, 6, 7],
                       [1/7, 1/6, 1/5, 1/3, 1/2, 1, 2, 3, 5, 6],
                       [1/8, 1/7, 1/6, 1/5, 1/3, 1/2, 1, 2, 3, 5],
                       [1/9, 1/8, 1/7, 1/6, 1/5, 1/3, 1/2, 1, 2, 3],
                       [1/11, 1/9, 1/8, 1/7, 1/6, 1/5, 1/3, 1/2, 1, 2],
                       [1/13, 1/11, 1/9, 1/8, 1/7, 1/6, 1/5, 1/3, 1/2, 1]])
# 次数转化
sixmatrix = np.array([[1, 2, 4, 5, 6, 7, 8, 10, 12, 13],
                      [1/2, 1, 2, 4, 5, 6, 7, 8, 10, 12],
                      [1/4, 1/2, 1, 2, 4, 5, 6, 7, 8, 10],
                      [1/5, 1/4, 1/2, 1, 2, 4, 5, 6, 7, 8],
                      [1/6, 1/5, 1/4, 1/2, 1, 2, 4, 5, 6, 7],
                      [1/7, 1/6, 1/5, 1/4, 1/2, 1, 2, 4, 5, 6],
                      [1/8, 1/7, 1/6, 1/5, 1/4, 1/2, 1, 2, 4, 5],
                      [1/10, 1/8, 1/7, 1/6, 1/5, 1/4, 1/2, 1, 2, 4],
                      [1/12, 1/10, 1/8, 1/7, 1/6, 1/5, 1/4, 1/2, 1, 2],
                      [1/13, 1/12, 1/10, 1/8, 1/7, 1/6, 1/5, 1/4, 1/2, 1]])
```

5.采用算术平均法计算各判断矩阵权重向量

In [ ]:
```
onedict = {}
twodict = {}

if __name__ == "__main__":

    #一致性检验
    one_consistency_check = AHP(onematrix).test_consist()
    # 算术平均法求权重
    oneweight = AHP(onematrix).cal_weight_by_arithmetic_method()
    for i in range(len(onetitle)):
        onedict[onetitle[i]] = oneweight[i]

    #一致性检验
    two_consistency_check = AHP(twomatrix).test_consist()
    # 算术平均法求权重
    twoweight = AHP(twomatrix).cal_weight_by_arithmetic_method()
    for i in range(len(onetwotitle['交易指标'])):
        twodict[onetwotitle['交易指标'][i]] = twoweight[i]

    #一致性检验
    three_consistency_check = AHP(threematrix).test_consist()
    # 算术平均法求权重
    threeweight = AHP(threematrix).cal_weight_by_arithmetic_method()
    for i in range(len(onetwotitle['支出指标'])):
```

```
        twodict[onetwotitle['支出指标'][i]] = threeweight[i]

    #一致性检验
    four_consistency_check = AHP(fourmatrix).test_consist()
    # 算术平均法求权重
    fourweight = AHP(fourmatrix).cal_weight_by_arithmetic_method()
    for i in range(len(onetwotitle['人气指标'])):
        twodict[onetwotitle['人气指标'][i]] = fourweight[i]

    #一致性检验
    five_consistency_check = AHP(fivematrix).test_consist()
    # 算术平均法求权重
    fiveweight = AHP(fivematrix).cal_weight_by_arithmetic_method()
    for i in range(len(onetwotitle['人数转化'])):
        twodict[onetwotitle['人数转化'][i]] = fiveweight[i]

    #一致性检验
    six_consistency_check = AHP(sixmatrix).test_consist()
    # 算术平均法求权重
    sixweight = AHP(sixmatrix).cal_weight_by_arithmetic_method()
    for i in range(len(onetwotitle['次数转化'])):
        twodict[onetwotitle['次数转化'][i]] = sixweight[i]
```

```
判断矩阵的CR值为(0.0711497930619321+0j),通过一致性检验
算术平均法计算得到的权重向量为：
 [0.48297904 0.26142881 0.14145382 0.07390771 0.04023061]
判断矩阵的CR值为(0.007933373029552656+0j),通过一致性检验
算术平均法计算得到的权重向量为：
 [0.53896104 0.2972583  0.16378066]
判断矩阵的CR值为(0.06691728507528544+0j),通过一致性检验
算术平均法计算得到的权重向量为：
 [0.51997713 0.2681975  0.14092085 0.07090452]
判断矩阵的CR值为(0.027021630769513656+0j),通过一致性检验
算术平均法计算得到的权重向量为：
 [0.32528062 0.22491995 0.15506336 0.10588668 0.0713605  0.04755762
 0.03218734 0.02212676 0.01561717]
判断矩阵的CR值为(0.04596169189043939+0j),通过一致性检验
算术平均法计算得到的权重向量为：
 [0.30609236 0.21551577 0.15225511 0.10696504 0.07537883 0.05275675
 0.03642914 0.02468559 0.01737231 0.01254909]
判断矩阵的CR值为(0.06025811682876953+0j),通过一致性检验
算术平均法计算得到的权重向量为：
 [0.31121247 0.21775922 0.14983458 0.104919   0.07452805 0.05256672
 0.03660199 0.02468662 0.01613405 0.01175729]
```

6.对数据进行归一化，计算各项权重得分

In [ ]:

```
# 本文使用最大最小归一化原则，公式为res = (X-Xmin)/(Xmax-Xmin)
disposedata = pd.DataFrame(data)
datamax = disposedata.max().to_dict()
datamin = disposedata.min().to_dict()

for i in onetwotitle:
    for j in range(len(onetwotitle[i])):
        disposedata[onetwotitle[i][j]] = (disposedata[onetwotitle[i][j]] - datamin[onetwotitle[i][j]]) / (
# print(disposedata)

# 计算归一化后数据的权重得分
for i in onetwotitle:
    for j in range(len(onetwotitle[i])):
        disposedata[onetwotitle[i][j]] = disposedata[onetwotitle[i][j]] * twodict[onetwotitle[i][j]]

# 计算第一层数据权重得分和
for i in onetwotitle:
    disposedata = disposedata.assign(**{i:0})
    for j in range(len(onetwotitle[i])):
        disposedata[i] = disposedata[i] + disposedata[onetwotitle[i][j]]

disposedata = disposedata.assign(最终得分=0)
for i in onetitle:
    disposedata['最终得分'] = disposedata['最终得分'] + disposedata[i] * onedict[i]
print(disposedata['最终得分'])
```

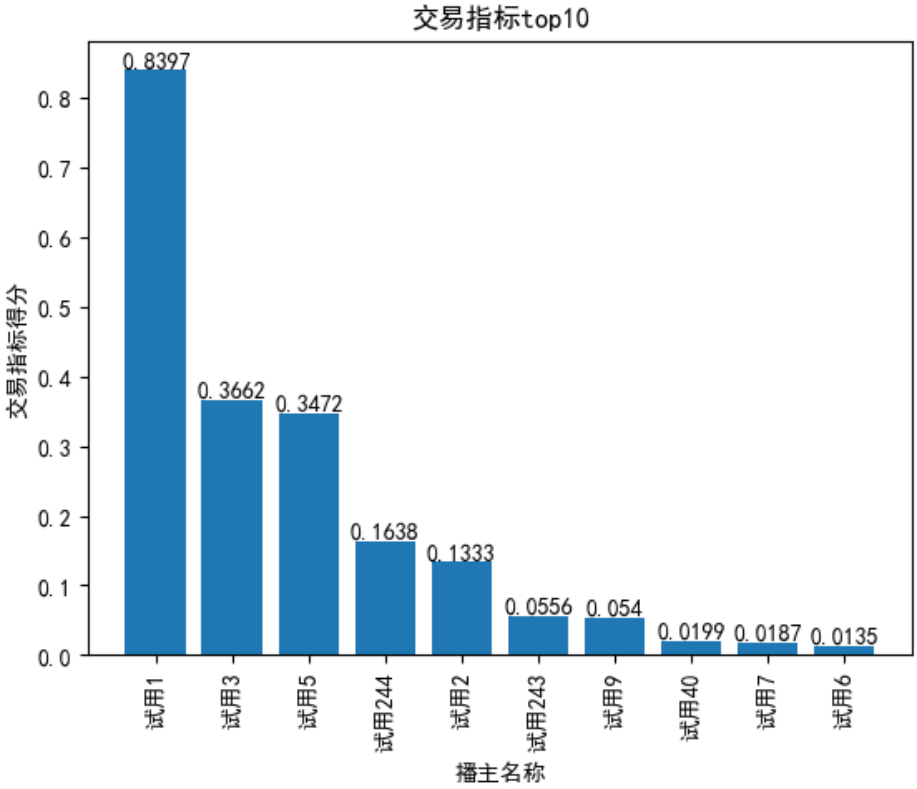
```
0      0.622741
1      0.153741
2      0.301046
3      0.037788
4      0.455411
         ...
```

```
239    0.006428
240    0.008126
241    0.019712
242    0.038812
243    0.096345
Name: 最终得分, Length: 244, dtype: float64
```

7.交易指标得分排名

In [ ]:

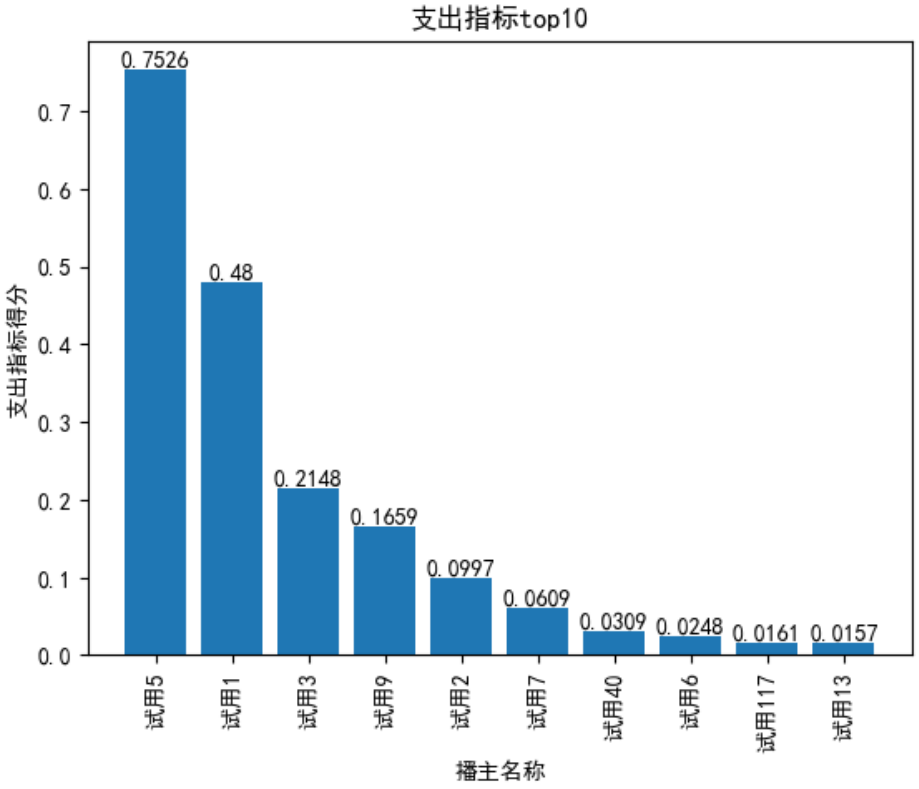
```
import matplotlib.pyplot as plt
import warnings
plt.rcParams['font.sans-serif'] = ['SimHei'] # 显示中文标签
# 绘制结果
warnings.filterwarnings("ignore", category=UserWarning)
#1.交易指标得分排名为
one_sortdata = disposedata.sort_values(by='交易指标',ascending=False)
categories = one_sortdata['主播昵称'][:10].to_list()
one_values = np.round(np.array(one_sortdata['交易指标'][:10].to_list()),4)
one_tag = plt.bar(categories,one_values)
plt.title('交易指标top10')
plt.xlabel('播主名称')
plt.ylabel('交易指标得分')
plt.bar_label(one_tag)
plt.xticks(rotation=90)
plt.show()
```



8.支出指标得分排名

In [ ]:

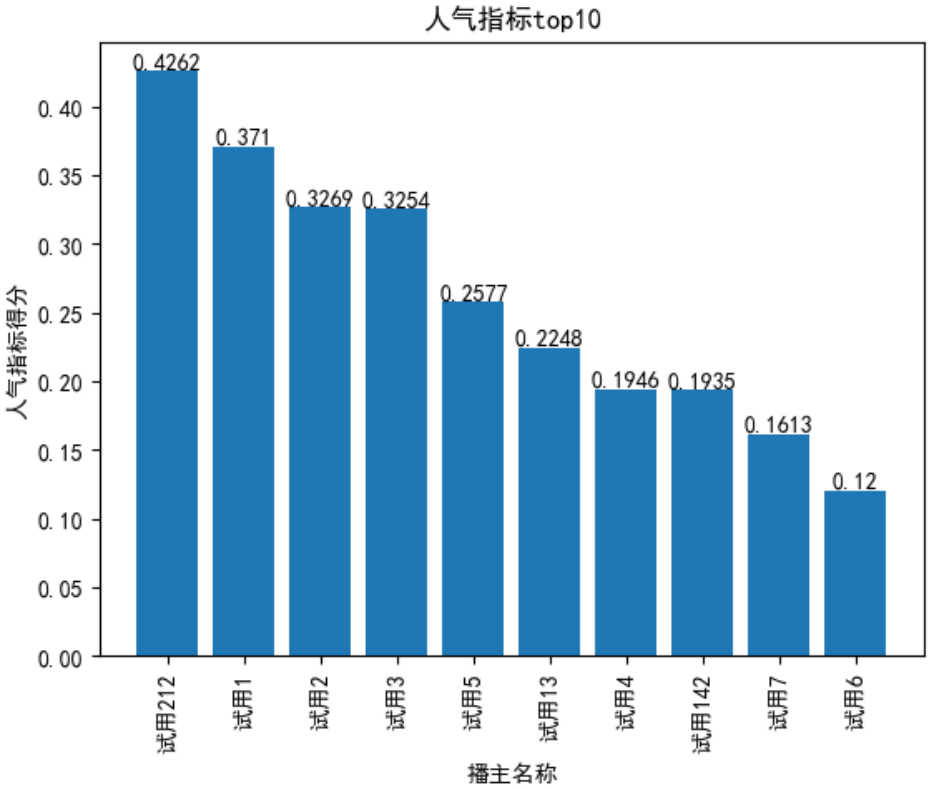
```
#2.支出指标得分排名为
two_sortdata = disposedata.sort_values(by='支出指标',ascending=False)
categories = two_sortdata['主播昵称'][:10].to_list()
two_values = np.round(np.array(two_sortdata['支出指标'][:10].to_list()),4)
two_tag = plt.bar(categories,two_values)
plt.title('支出指标top10')
plt.xlabel('播主名称')
plt.ylabel('支出指标得分')
plt.bar_label(two_tag)
plt.xticks(rotation=90)
plt.show()
```



9.人气指标得分排名

In [ ]:

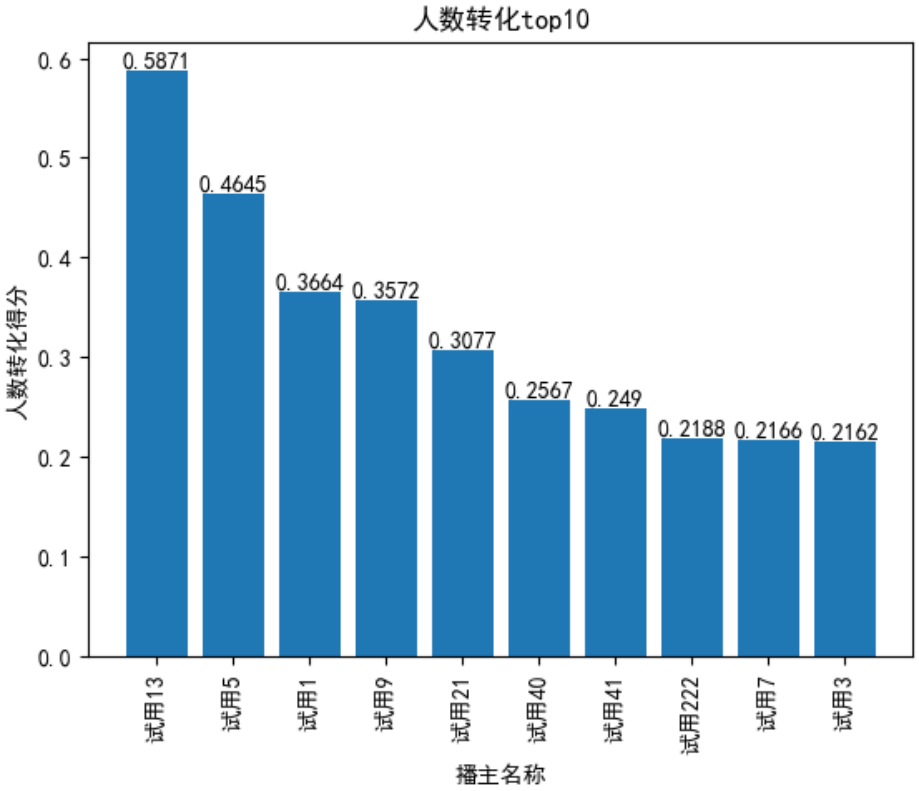
```
#3.人气指标得分排名为
three_sortdata = disposedata.sort_values(by='人气指标',ascending=False)
categories = three_sortdata['主播昵称'][:10].to_list()
three_values = np.round(np.array(three_sortdata['人气指标'][:10].to_list()),4)
three_tag = plt.bar(categories,three_values)
plt.title('人气指标top10')
plt.xlabel('播主名称')
plt.ylabel('人气指标得分')
plt.bar_label(three_tag)
plt.xticks(rotation=90)
plt.show()
```



10.人数转化得分排名

In [ ]:

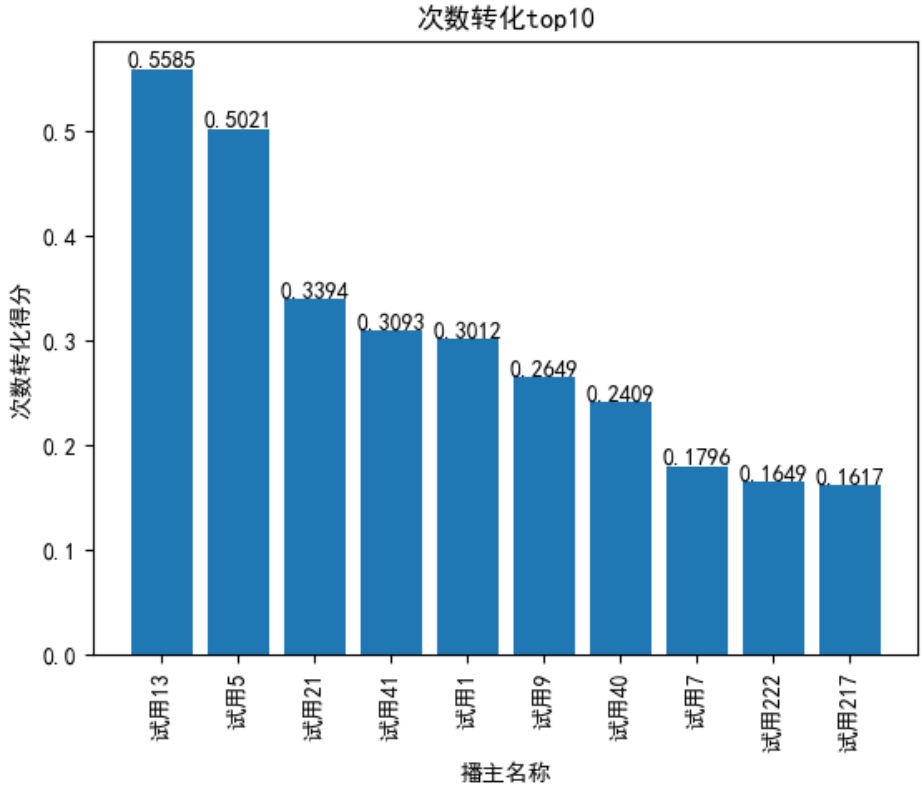
```
#4.人数转化得分排名为
four_sortdata = disposedata.sort_values(by='人数转化',ascending=False)
categories = four_sortdata['主播昵称'][:10].to_list()
four_values = np.round(np.array(four_sortdata['人数转化'][:10].to_list()),4)
four_tag = plt.bar(categories,four_values)
plt.title('人数转化top10')
plt.xlabel('播主名称')
plt.ylabel('人数转化得分')
plt.bar_label(four_tag)
plt.xticks(rotation=90)
plt.show()
```



11.次数转化得分排名

In [ ]:

```
#5.次数转化得分排名为
five_sortdata = disposedata.sort_values(by='次数转化',ascending=False)
categories = five_sortdata['主播昵称'][:10].to_list()
five_values = np.round(np.array(five_sortdata['次数转化'][:10].to_list()),4)
five_tag = plt.bar(categories,five_values)
plt.title('次数转化top10')
plt.xlabel('播主名称')
plt.ylabel('次数转化得分')
plt.bar_label(five_tag)
plt.xticks(rotation=90)
plt.show()
```



12.最终得分排名

In [ ]:

```
#6.最终得分排名为
six_sortdata = disposedata.sort_values(by='最终得分',ascending=False)
categories = six_sortdata['主播昵称'][:10].to_list()
six_values = np.round(np.array(six_sortdata['最终得分'][:10].to_list()),4)
six_tag = plt.bar(categories,six_values)
plt.title('最终得分top10')
plt.xlabel('播主名称')
plt.ylabel('最终得分')
plt.bar_label(six_tag)
plt.xticks(rotation=90)
plt.show()
```