Weathe Analysis By Python

> *对于本次的 气象分析报告，大致上，我完成了所有的代码以及绘图，实现了任务书上的一系列任务。此刻，我就针对 气象分析过程 进行 一个分析以及 总结。*

- 本次 使用到的知识点：
  - numpy的 一些 向量运算，以及矩阵的使用
  - pandas的数据分析 ，包括Series、DataFrame还有loc等切片切片操作
  - matplotlib的绘图，例如线性，柱状图 等，包括如何 设置标题，散点，颜色等等
  - 程序设计 --封装思想

> *这是第一次的个人书写分析项目，因此难免会有不足，在 完成的 过程中，我们需要善用**浏览器***

Weather Analysis

前提准备

> *在开始项目之前，我们需要对需要分析的 十个城市进行数据抽取，因此，我们在特定的网站上下载获取了 对应的数据集，以及分析 得到十个城市距海的物理距离：*

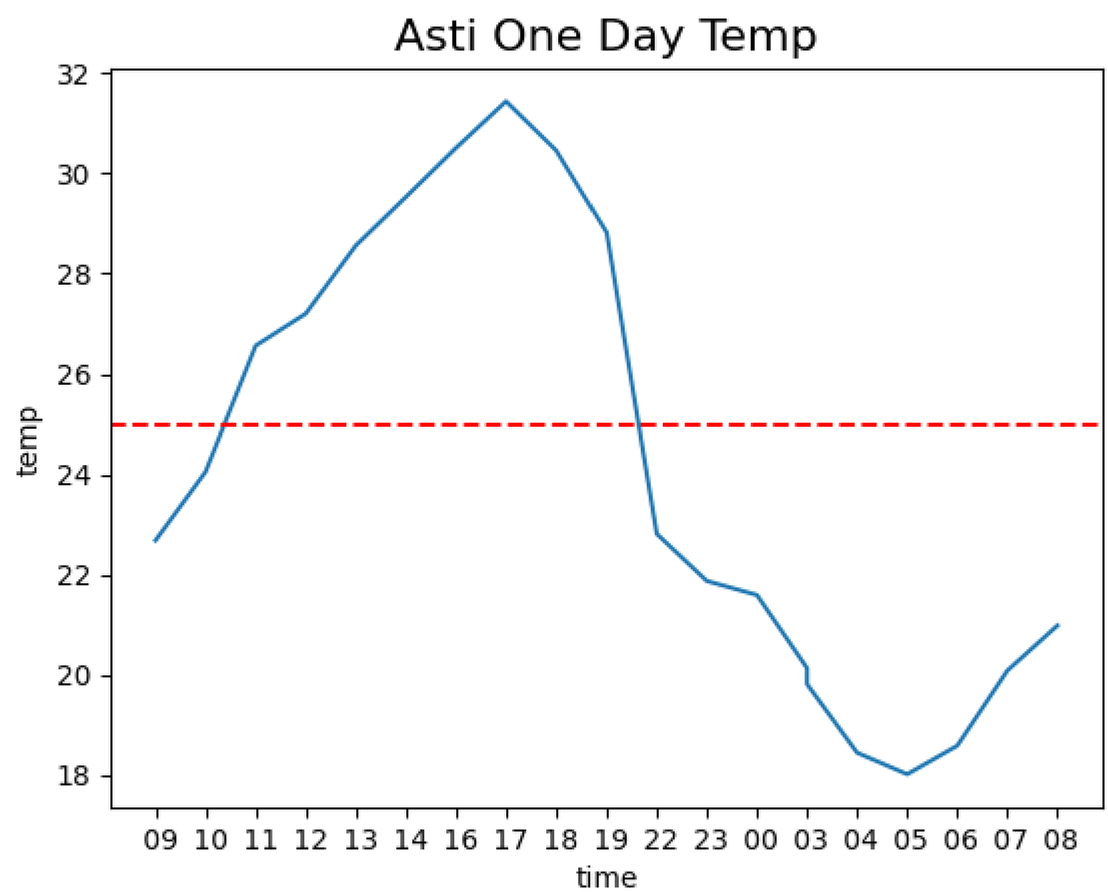
```python
def Distance() :
    '''
        为了方便，我们采取了用一个城市到一个沿海城市的距离近似作为距海距离
        本次，我们选取意大利的Comacchio作为参考城市
        同时，直接使用城市名代表距离(km)
    '''
     dis = {
        "Ferrara"   : 47,
        "Torino"    : 358,
        "Mantova"   : 121,
        "Milano"    : 251,
        "Ravenna"   : 31,
        "Asti"      : 316,
        "Bologna"   : 71,
        "Piacenza"  : 200,
        "Cesena"    : 62,
        "Faenza"    : 52
    }
    return dis
```

> *当这一些数据准备完成之后，我们就可以 开始项目的 第一个板块内容*

温度数据分析

绘制某个城市一天的气温变化

```
def Asti() :
    df = pd.read_csv("WeatherData/asti_270615.csv")
    temp = df[:]["temp"]
    time = df[:]["day"]
    dtime = []
    for i in range(0, 20) :
        dtime.append(time[i][10:13])
    dtime = pd.Series(dtime)
    print(dtime)
    plt.plot(dtime, temp)
    plt.title("Asti One Day Temp", fontsize = 16)
    plt.xlabel("time")
    plt.ylabel("temp")
    plt.axhline(y=25, ls='--', c='red')
    plt.savefig("Asti.png")
    plt.show()
```



> *经过分析，我们发现**从早上5点开始到下午的17时气温逐步上升**，直到大概 17时达到峰值， 随后开始下降*

六个城市之间的 气温变化关系

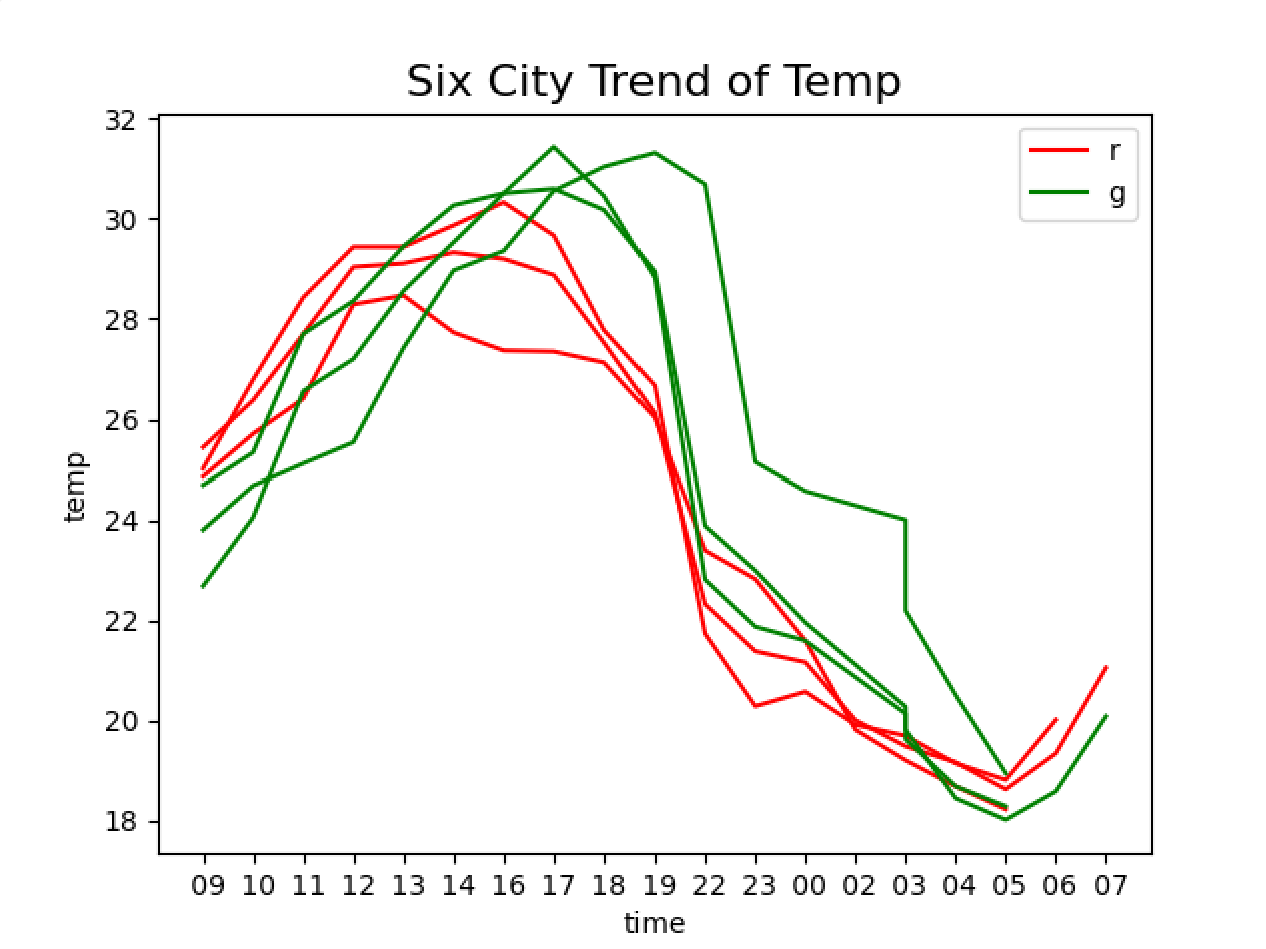
> *通过排列组合，我们可以得出，三个离海最近的以及 三个离海最远的*
> *分别分析后统一画出线性关系图*

```
def sort(x) :
    '''
        字典进行排序，查看距离最近的三个以及距离最远的三个
        可以知道:              最近的三个城市: Ravenna、Ferrara、Faenza
            最远的三个城市: Milano、Asti、Torino
    '''    sort_list = zip(x.values(), x.keys())
    sort_list = sorted(sort_list, reverse=False)
    return sort_list
```

```
def compareWeather() :
    '''
        探究三个最远的城市和三个最近的城市天气
    '''
```

```
'''
# 三个最近的城市
ravenna = pd.read_csv("WeatherData/ravenna_270615.csv")
temp1 = ravenna[:]["temp"]
time1 = ravenna[:]["day"]
dtime1 = []
for i in range(0, 17):
    dtime1.append(time1[i][10:13])
dtemp1 = temp1[0:17]
dtime1 = pd.Series(dtime1)

ferrara = pd.read_csv("WeatherData/ferrara_270615.csv")
temp2 = ferrara[:]["temp"]
time2 = ferrara[:]["day"]
dtime2 = []
for j in range(1, 19):
    dtime2.append(time2[j][10:13])
dtemp2 = temp2[1:19]
dtime2 = pd.Series(dtime2)

faenza  = pd.read_csv("WeatherData/faenza_270615.csv")
temp3 = faenza[:]["temp"]
time3 = faenza[:]["day"]
dtime3 = []
for k in range(0, 19):
    dtime3.append(time3[k][10:13])
dtemp3 = temp3[0:19]
dtime3 = pd.Series(dtime3)

# 绘制对应的图像，最近的用红色，最远的用绿色
plt.plot(dtime1, dtemp1, color = 'red', label = 'r')
plt.plot(dtime2, dtemp2, color = 'r')
plt.plot(dtime3, dtemp3, color = 'r')

# 三个最远的城市
milano = pd.read_csv("WeatherData/milano_270615.csv")
temp1 = milano[:]["temp"]
time1 = milano[:]["day"]
dtime1 = []
for i in range(0, 17):
    dtime1.append(time1[i][10:13])
dtemp1 = temp1[0:17]
dtime1 = pd.Series(dtime1)

asti = pd.read_csv("WeatherData/asti_270615.csv")
temp2 = asti[:]["temp"]
time2 = asti[:]["day"]
dtime2 = []
for j in range(0, 19):
```

```
for i in range(0, 19):
    dtime2.append(time2[i][10:13])
dtemp2 = temp2[0:19]
dtime2 = pd.Series(dtime2)

torino = pd.read_csv("WeatherData/torino_270615.csv")
temp3 = torino[:]["temp"]
time3 = torino[:]["day"]
dtime3 = []
for i in range(0, 17):
    dtime3.append(int(time3[i][10:13]))
dtemp3 = temp3[0:17]
dtime3 = pd.Series(dtime1)

# 绘制对应的图像，最近的用红色，最远的用绿色
plt.plot(dtime1, dtemp1, color='green', label = 'g')
plt.plot(dtime2, dtemp2, color='g')
plt.plot(dtime3, dtemp3, color='g')
plt.title("Six City Trend of Temp", fontsize = 16)
plt.xlabel("time")
plt.ylabel("temp")
plt.legend(loc = 'upper right')
plt.savefig("SixCityTrendofTemp.png")
plt.show()
```



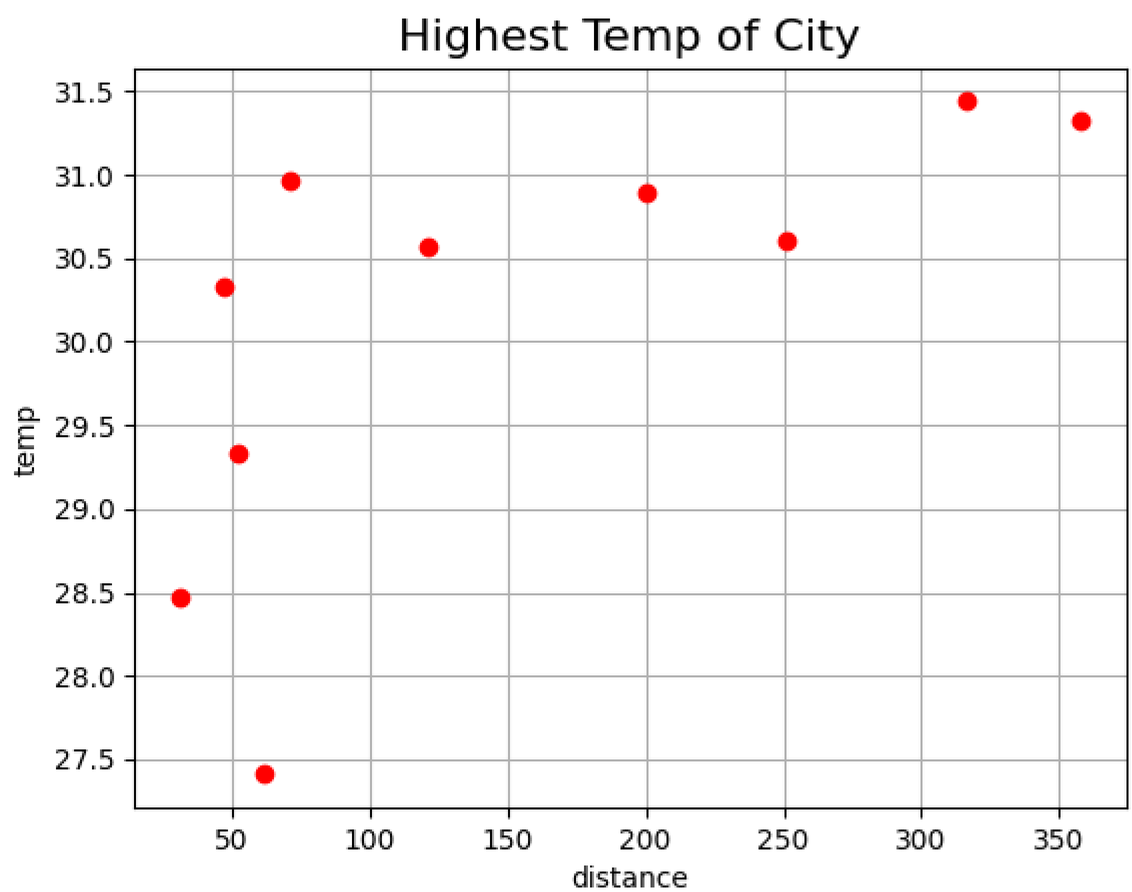
> *据图 分析，可见，绿色线条的最高值总是高于红色线条，因此说明了离海的 远近对气温是有影响的*

研究最值点与离海远近之间的关系

> *我们通过获取到的十个数据集，可以轻易的分析出 十个城市一天中气温的最值*

```
def CityHighTemp(path) :
    temp = pd.read_csv(path)
    temp = temp.sort_values(by="temp", ascending=False)
    temp = temp[:]["temp"]
    temp = temp.iloc[0:1] # 取出特定行
    return temp

def highTemp(x) :
    x = dict(sort(x))
    dis = pd.Series(x.keys())
    rank = pd.Series(x.values())
    print(rank)

    # 此时应该按照rank的远近顺序来提取出各自的温度最大值
    # 由于相似代码太多，直接封装函数，传入csv文件路径即可
```

```
    temp1 = CityHighTemp("WeatherData/ravenna_270615.csv")
    temp2 = CityHighTemp("WeatherData/ferrara_270615.csv")
    temp3 = CityHighTemp("WeatherData/faenza_270615.csv")
    temp4 = CityHighTemp("WeatherData/cesena_270615.csv")
    temp5 = CityHighTemp("WeatherData/bologna_270615.csv")
    temp6 = CityHighTemp("WeatherData/mantova_270615.csv")
    temp7 = CityHighTemp("WeatherData/piacenza_270615.csv")
    temp8 = CityHighTemp("WeatherData/milano_270615.csv")
    temp9 = CityHighTemp("WeatherData/asti_270615.csv")
    temp10 = CityHighTemp("WeatherData/torino_270615.csv")
    temp = [temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8, temp9, temp
    dis = list(dis)
    print(temp)
    print(dis)
    plt.plot(dis, temp, 'ro')
    plt.title("Highest Temp of City", fontsize = 16)
    plt.xlabel("distance")
    plt.ylabel("temp")
    plt.grid()
    plt.savefig("HighestTemOfCity.png")
    plt.show()
    return temp, dis
```



> *可以看见，当distance越大，temp值就 越高，因此离海距离对最高温度的影响也是十分显著的*

使用线性回归对我们上述的结论进行验证

- 对于线性回归，如果是多元的，那么有：

$$y = a_2x^2 + a_1x + a_0 \to \begin{bmatrix} x_1^2 & x_1 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

- 假若$(x_i, y_i), i = 1, 2, \ldots, n$，那么可以得到：

$$\begin{bmatrix} x_i^2 & x_i & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = y_i$$

- 所以，可以化为矩阵的形式：

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

- 假设$\begin{bmatrix} x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix}$为A, $\begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix}$为x, $\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$为T,那么:

$$Ax = T \implies A^TAx = A^TT \implies (A^TA)^{-1}A^TAx = (A^TA)^{-1}A^TT$$

- 因此:

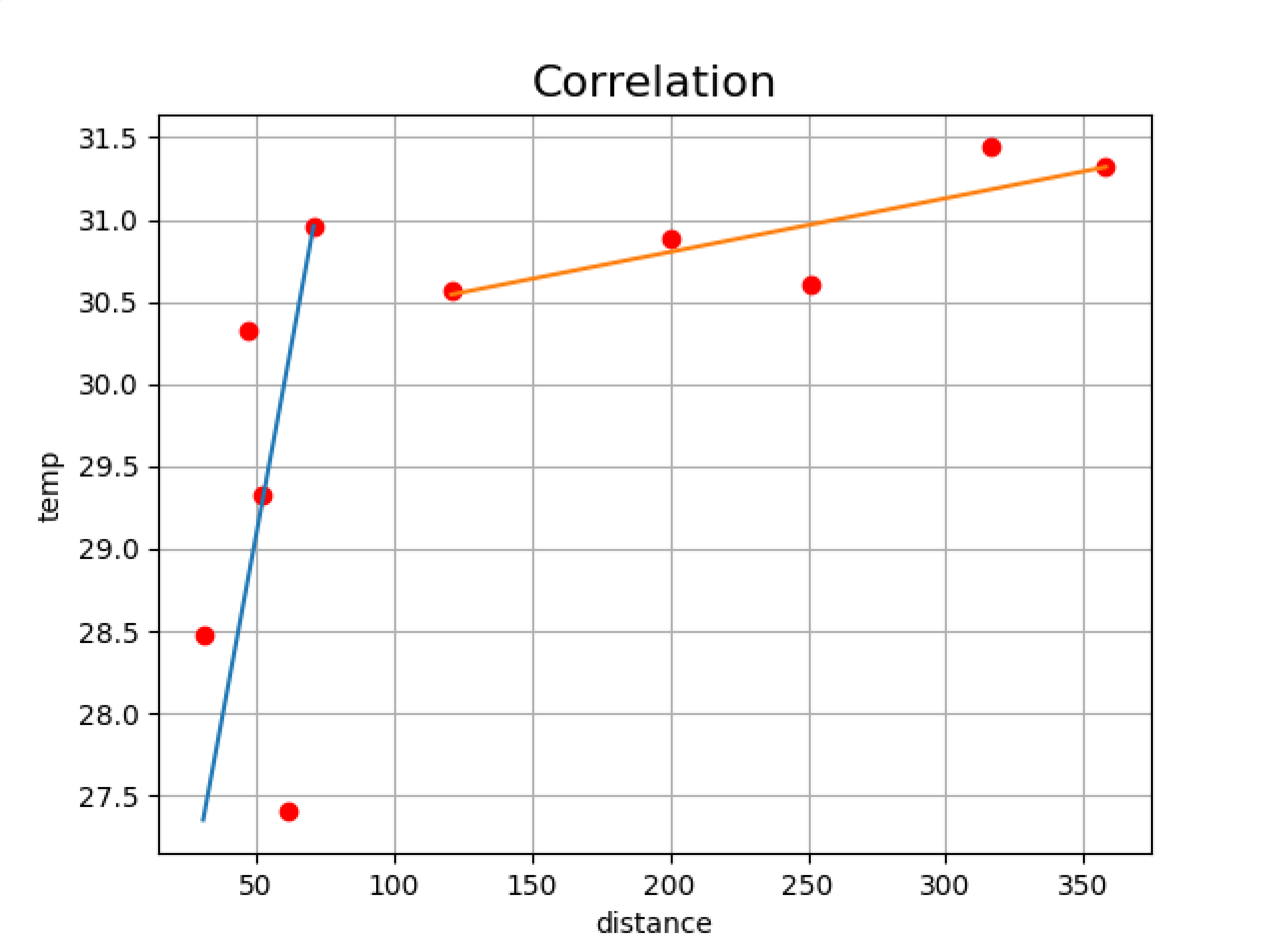
$$x = (A^TA)^{-1}A^TT$$

> *但是，因为本次直接为一元线性，因此设$y = ax + b$:*

$$a = \frac{n\sum_{i=1}^n x_iy_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n\sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}, b = \vec{y} - a\vec{x}$$

```
def Correlation(temp, dis) :
    x = np.array(dis)
    y = np.array(temp)
    # 设 y = ax + b , 已知x,y通过拟合来求解a,b系数
    '''
        cost = min(sum_i^n (y_hat - y_i)^2)        y_hat = ax + b          cost =
        da = -1 * 2 * sum_i^n (y_i  - b - ax)^2 = 0         db = -x_i * 2 * sum_
            a = [sum(x_i^2) * sum(y_i) - sum(x_i) * sum(x_i * y_i)] / [n * sum(
    x_avr = x.sum() / len(x)
    y_avr = y.sum() / len(y)

    for i in x :
        for j in y :
            t = h = 0
            t += (i - x_avr) * (j - y_avr)
            h += (i - x_avr) ** 2

    a = t / h
    b = y_avr - a * x_avr

    #a = ((x - x_avr) * (y - y_avr)) / ((x - x_avr)**2)
    #b = y_avr - a * x_avr
    return a, b

    '''
        有公式:
            a_0 * N + a_1 * sum_{i = 1} ^ {N} x_i = sum_{i = 1}^N y_i
    N = len(x)    sumx = sum(x)    sumy = sum(y)    sumx1 = sum(x**2)    sumxy =
    N = len(x)    sumx = x.sum()    sumy = y.sum()    X = x**2    sumx1 = X.sum
    return np.linalg.solve(A, b)    '''
def CorrelationTrendGraph(x) :
    '''
```

```
    我们已经在highTemp中分析出了距离对于温度最大值的散点图关系
    我们在highTemp函数中将得到的距离和温度进行返回
    根据要求，我们将dis<60的作为近海，dis>60的作为远海，分为两组
    但根据实际，应该从71km作为分界线，分为两组
'''    temp, dis = highTemp(x)
# 近海组
temp1 = temp[0:5]
dis1 = dis[0:5]

# 远海组
temp2 = temp[5:10]
dis2 = dis[5:10]

# 由于进行线性回归的过程是一致的，因此封装一个函数进行线性分析
a1, b1 = Correlation(temp1, dis1)
a2, b2 = Correlation(temp2, dis2)

# 进行数据分析
dis1 = np.array(dis1)
y1 = a1 * dis1 + b1
print(a1)
print(dis1)
print(y1)

dis2 = np.array(dis2)
y2 = a2 * dis2 + b2

# 进行图像处理
plt.plot(dis, temp, 'ro')
plt.plot(dis1, y1)
plt.plot(dis2, y2)
plt.plot()
plt.title("Correlation", fontsize=16)
plt.xlabel("distance")
plt.ylabel("temp")
plt.grid()
plt.savefig("Correlation.png")
plt.show()
```



> *可以看出，线性回归也是依次增大的，因此，我们的猜测**离海的远近对气温有影响**是正确的*

湿度数据分析

> *考察三个近海，三个远海城市，那么我们先要进行数据分析和处理*

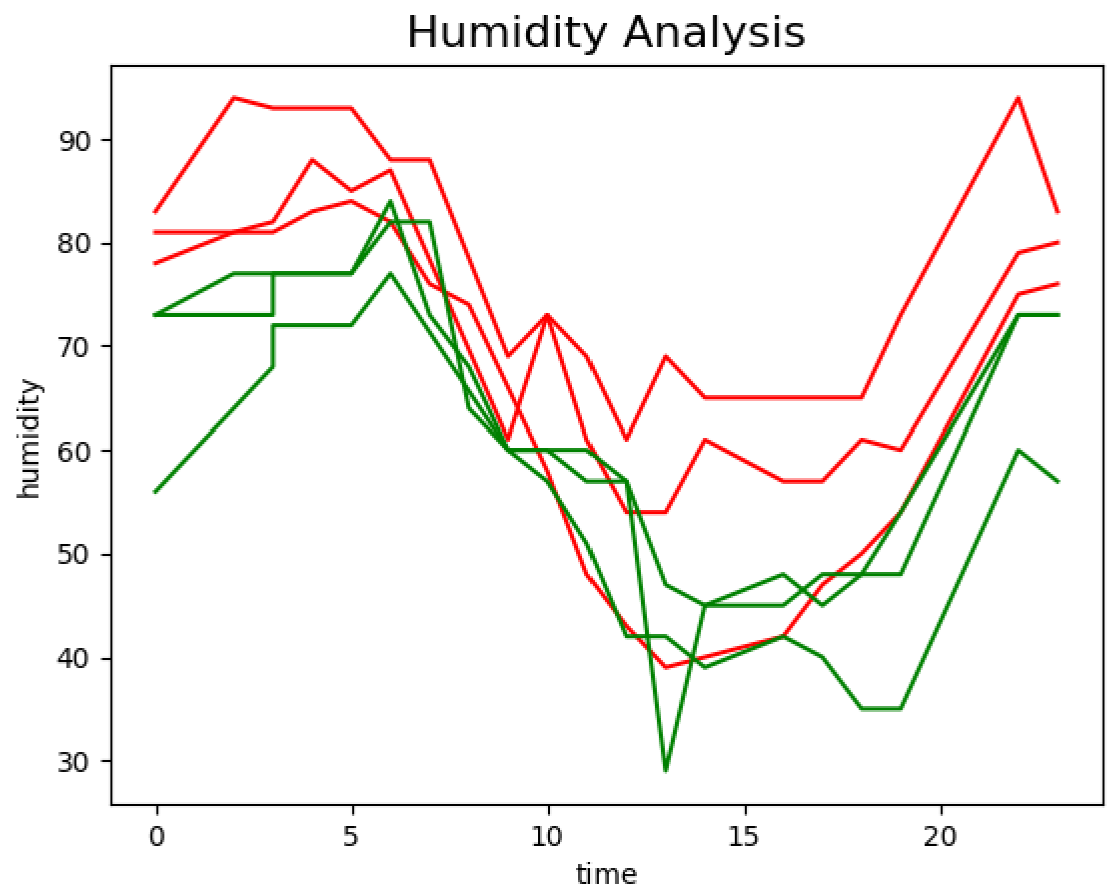
```
def cityHumidity(path) :
    humidity = pd.read_csv(path)
    cityhumidity = humidity[:]["humidity"]
    time = humidity[:]["day"]
    dtime = []
    for i in range(0, len(humidity)) :
        dtime.append(int(time[i][10:13]))
    return cityhumidity, dtime
```

```
def dataSovle(humidity, time) :
    humidity = list(humidity)
    key = sorted(zip(time, humidity))
```

```
    humidity = list(zip(*key))
    return humidity
```

```
def HumidityAnalysis(x) :
    '''
        分析十个城市中的湿度情况，选取六个城市，分别为三个近海，三个远海
        近海的采用红色，远海的采用绿色
    '''    x = dict(sort(x))
    dis = pd.Series(x.keys())
    rank = pd.Series(x.values())
    print(rank)
    # 根据分析，近海的三个城市为：Rvaenna、Ferrara、Faenza
    # 远海的三个城市为：Milano、Asti、Torino

    # 获取湿度数据,作为封装使用
    # 近海湿度数据
    rvaenna, time1 = cityHumidity("WeatherData/ravenna_270615.csv")
    ferrara, time2 = cityHumidity("WeatherData/ferrara_270615.csv")
    faenza, time3 = cityHumidity("WeatherData/faenza_270615.csv")

    # 远海湿度数据
    milano, time4 = cityHumidity("WeatherData/milano_270615.csv")
    asti, time5 = cityHumidity("WeatherData/asti_270615.csv")
    torino, time6 = cityHumidity("WeatherData/torino_270615.csv")

    # 分析处理数据做出图表
    rvaenna = dataSovle(rvaenna, time1)
    ferrara = dataSovle(ferrara, time2)
    faenza = dataSovle(faenza, time3)
    milano = dataSovle(milano, time4)
    asti = dataSovle(asti, time5)
    torino = dataSovle(torino, time6)
    print(rvaenna[0])
    print(ferrara[0])
    print(faenza[0])

    plt.plot(rvaenna[0], rvaenna[1], c='r')
    plt.plot(ferrara[0], ferrara[1], c='r')
    plt.plot(faenza[0], faenza[1], c='r')
    plt.plot(milano[0], milano[1], c='g')
    plt.plot(asti[0], asti[1], c='g')
    plt.plot(torino[0], torino[1], c='g')
    plt.title("Humidity Analysis", fontsize = 16)
    plt.xlabel("time")
    plt.ylabel("humidity")
```

```
    plt.savefig("Humidity.png")
    plt.show()
```



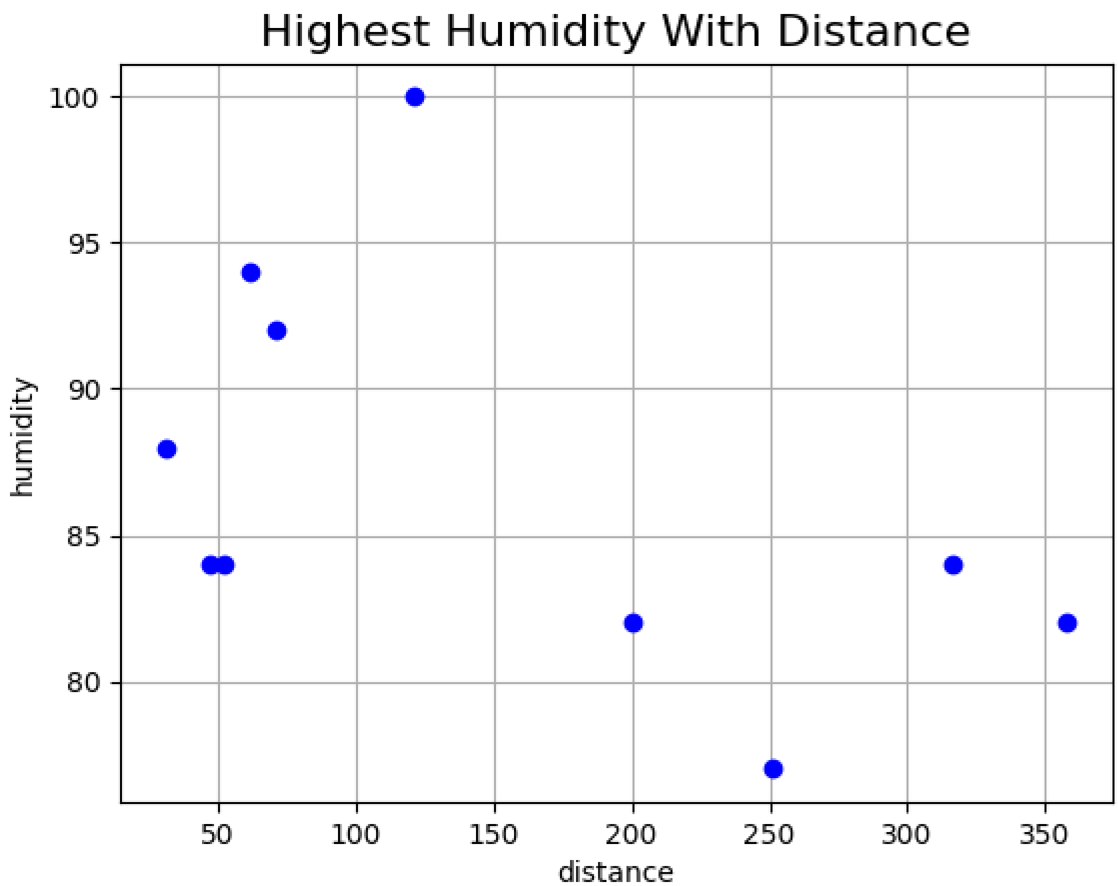
> *看上去，红色的湿度比绿色的湿度普遍偏高，接下来我们用湿度的最大值来判断*

最大(最小)湿度与离海距离关系

```
def CityHumidity(path, flag) :
    temp = pd.read_csv(path)
    temp = temp.sort_values(by="humidity", ascending=flag)
    temp = temp[:]["humidity"]
    temp = temp.iloc[0:1] # 取出特定行
    return temp
```

```
def MaxHumidity(x) :
    x = dict(sort(x))
    dis = pd.Series(x.keys())
    rank = pd.Series(x.values())
    print(rank)
```

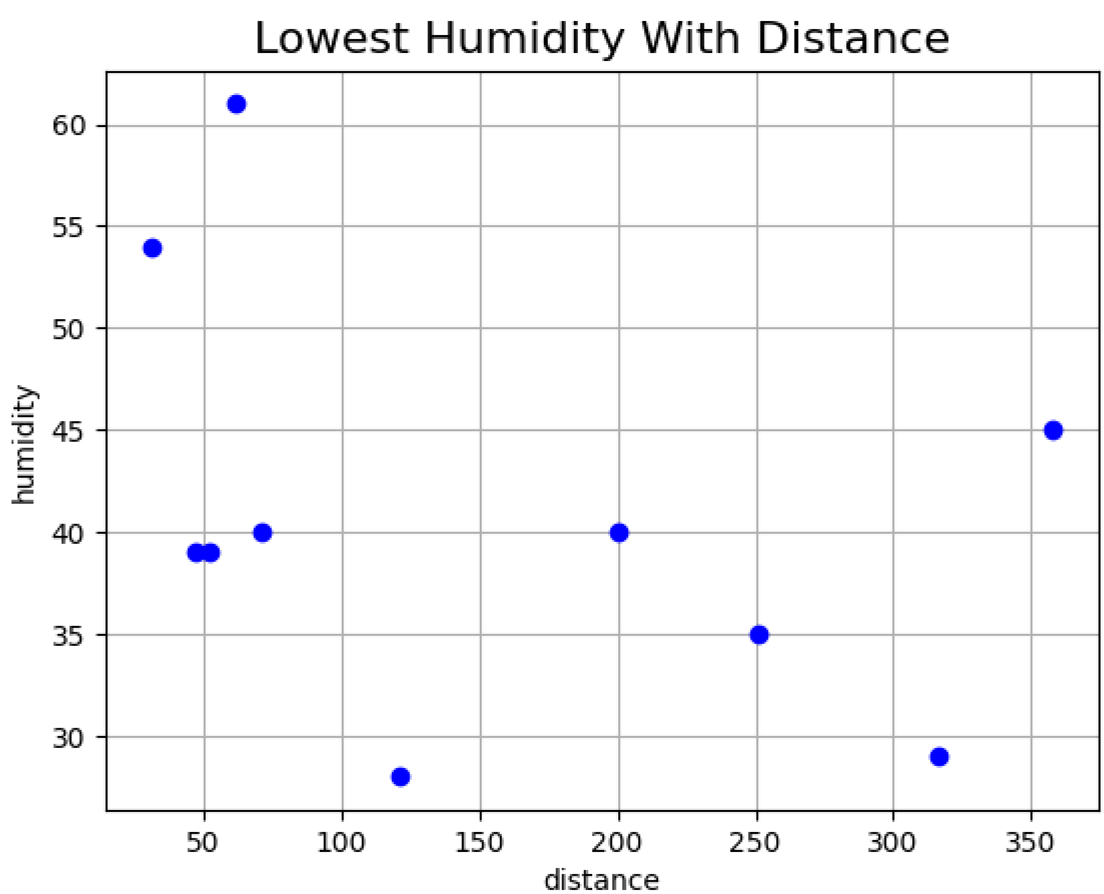
```
# 根据远近关系，找出各城市对应的最大湿度
ravenna = CityHumidity("WeatherData/ravenna_270615.csv", False)
ferrara = CityHumidity("WeatherData/ferrara_270615.csv", False)
faenza  = CityHumidity("WeatherData/ferrara_270615.csv", False)
cesena  = CityHumidity("WeatherData/cesena_270615.csv", False)
bologna = CityHumidity("WeatherData/bologna_270615.csv", False)
manatova = CityHumidity("WeatherData/mantova_270615.csv", False)
piacenza = CityHumidity("WeatherData/piacenza_270615.csv", False)
milano   = CityHumidity("WeatherData/milano_270615.csv", False)
asti     = CityHumidity("WeatherData/asti_270615.csv", False)
torino   = CityHumidity("WeatherData/torino_270615.csv", False)

maxhumi = [ravenna, ferrara, faenza, cesena, bologna, manatova, piacenza, m

# 处理数据图像
dis = list(dis)
plt.plot(dis, maxhumi, 'bo')
plt.title("Highest Humidity With Distance", fontsize = 16)
plt.xlabel("distance")
plt.ylabel("humidity")
plt.grid()
plt.savefig("HighestHumidityWithDistance.png")
plt.show()
```



> *可以发现，最大湿度随着距离的增加而大致降低*

```
def MinHumidity(x) :
    x = dict(sort(x))
    dis = pd.Series(x.keys())
    rank = pd.Series(x.values())
    print(rank)

    # 根据远近关系，找出各城市对应的最小湿度
    ravenna = CityHumidity("WeatherData/ravenna_270615.csv", True)
    ferrara = CityHumidity("WeatherData/ferrara_270615.csv", True)
    faenza = CityHumidity("WeatherData/ferrara_270615.csv", True)
    cesena = CityHumidity("WeatherData/cesena_270615.csv", True)
    bologna = CityHumidity("WeatherData/bologna_270615.csv", True)
    manatova = CityHumidity("WeatherData/mantova_270615.csv", True)
    piacenza = CityHumidity("WeatherData/piacenza_270615.csv", True)
    milano = CityHumidity("WeatherData/milano_270615.csv", True)
    asti = CityHumidity("WeatherData/asti_270615.csv", True)
    torino = CityHumidity("WeatherData/torino_270615.csv", True)

    minhumi = [ravenna, ferrara, faenza, cesena, bologna, manatova, piacenza, m
```

```
    # 处理数据图像
    dis = list(dis)
    plt.plot(dis, minhumi, 'bo')
    plt.title("Lowest Humidity With Distance", fontsize=16)
    plt.xlabel("distance")
    plt.ylabel("humidity")
    plt.grid()
    plt.savefig("LowestHumidityWithDistance.png")
    plt.show()
```



> *可以发现，最低湿度也是随着距离的增加而逐渐下降的*

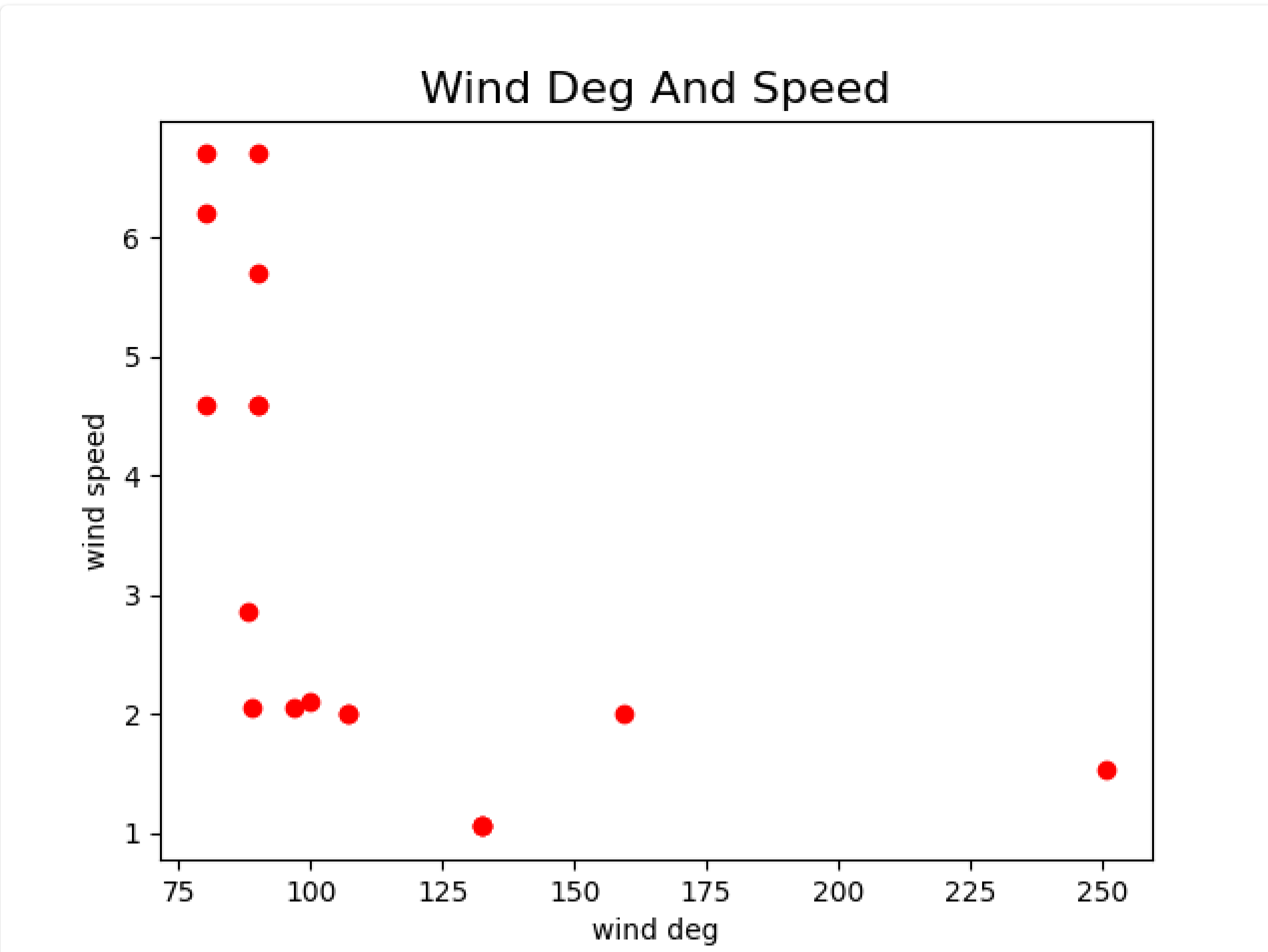
> *因此，对于湿度关于离海距离的影响，是**随着距离的增加湿度逐渐下降的***

风向分析

风向散点图

```
def windDirection(x) :
    '''
        上面已经分析了温度以及湿度，现在开始分析风速、风向
```

```
    取出数据集中的wind_deg、wind_speed
    '''    ravenna = pd.read_csv("WeatherData/ravenna_270615.csv")
    ravenna = ravenna[:][["wind_deg", "wind_speed", "day"]]

    # 数据处理
    plt.plot(ravenna["wind_deg"], ravenna["wind_speed"], 'ro')
    plt.title("Wind Deg And Speed", fontsize = 16)
    plt.xlabel("wind deg")
    plt.ylabel("wind speed")
    plt.savefig("WindDegAndSpeed.png")
    plt.show()
```



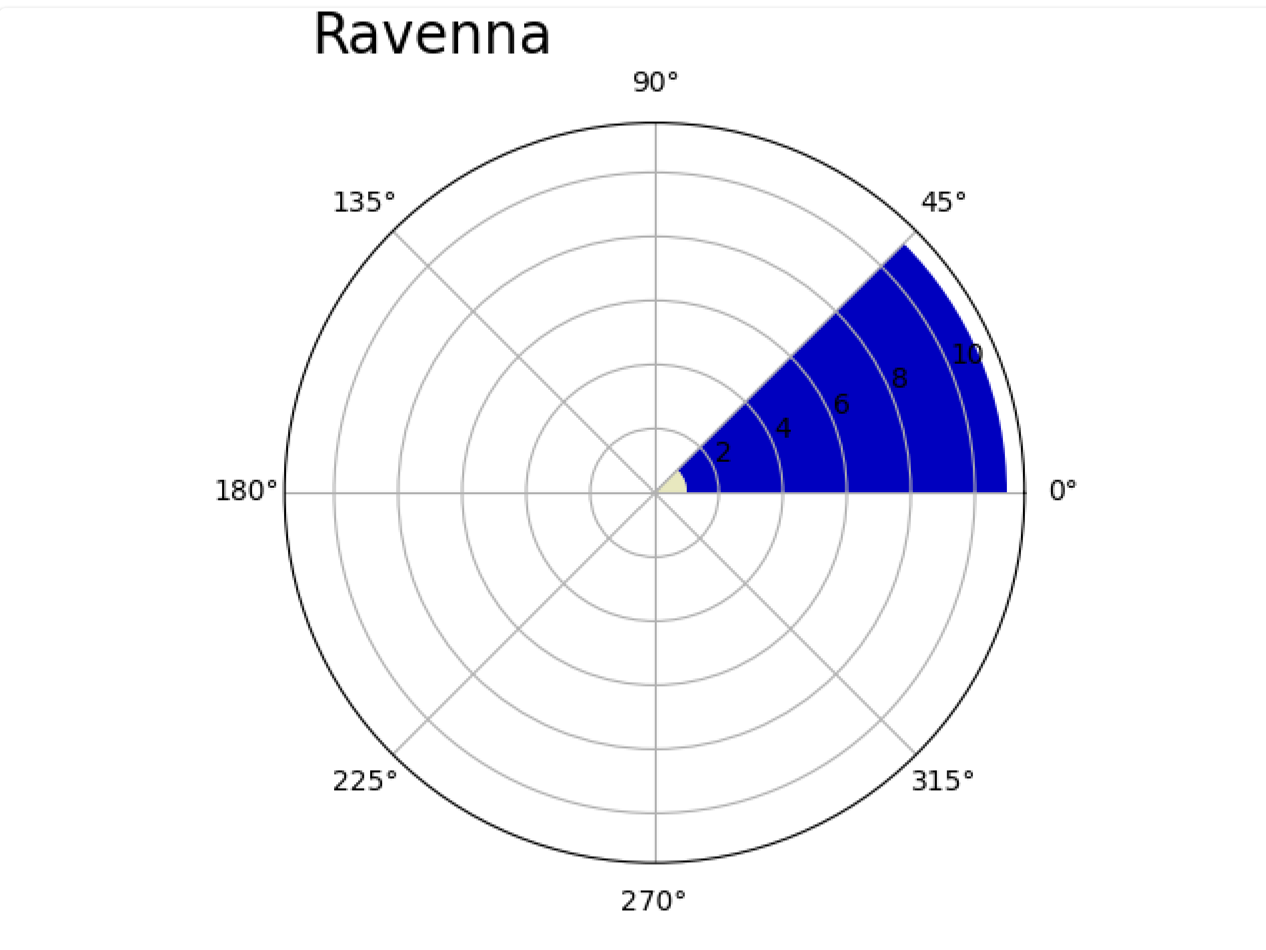
> *可以发现，这一些数据无法支撑实验结论，因此，还需要多方位的考虑*

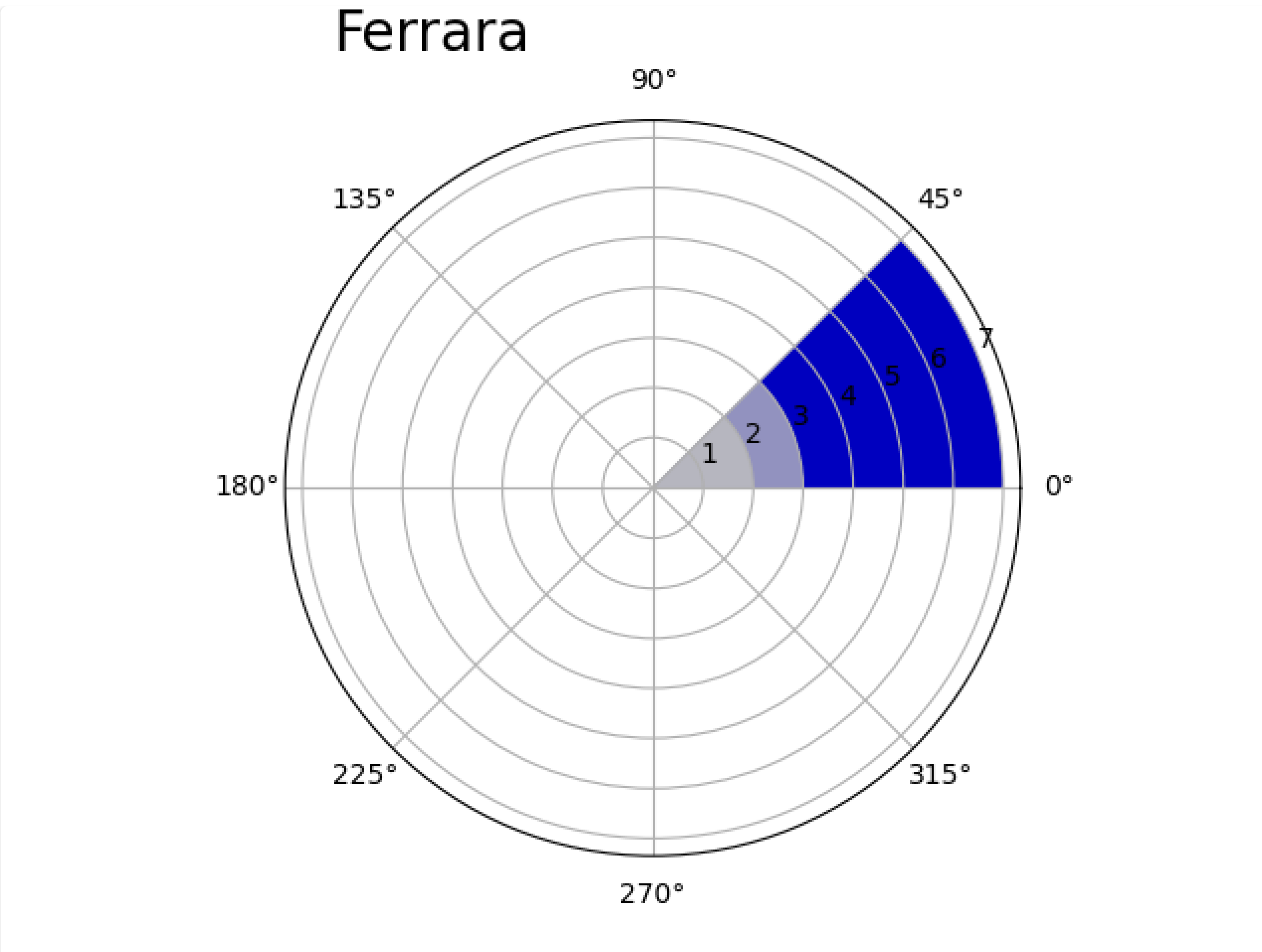
风向玫瑰图

```
def showRoseWind(var, city, max_var) :

    N = 8 # 一共八个区间
    theta = np.arange(2 * np.pi / 16, 2 * np.pi / 8)
    #theta = np.arange(0. + np.pi / 8, 2 * np.pi / 8, 2 * np.pi / 8)
    radii = np.array(var)
```

```
    plt.axes(polar=True)
    colors = [(1 - x / max_var, 1 - x / max_var, 0.75) for x in radii]
    plt.bar(theta, radii, width=(2 * np.pi / N), bottom=0.0, color = colors)
    plt.title(city, x = 0.2, fontsize = 20)
    plt.savefig(city + ".png")
    plt.show()
```

```
def PolarMap() :
    ravenna = pd.read_csv("WeatherData/ravenna_270615.csv")
    hist, bins = np.histogram(ravenna["wind_deg"], 8, [0, 360])
    showRoseWind(hist, "Ravenna", max(hist))

    ferrara = pd.read_csv("WeatherData/ferrara_270615.csv")
    hist, bins = np.histogram(ferrara["wind_deg"], 8, [0, 360])
    showRoseWind(hist, "Ferrara", max(hist))
```



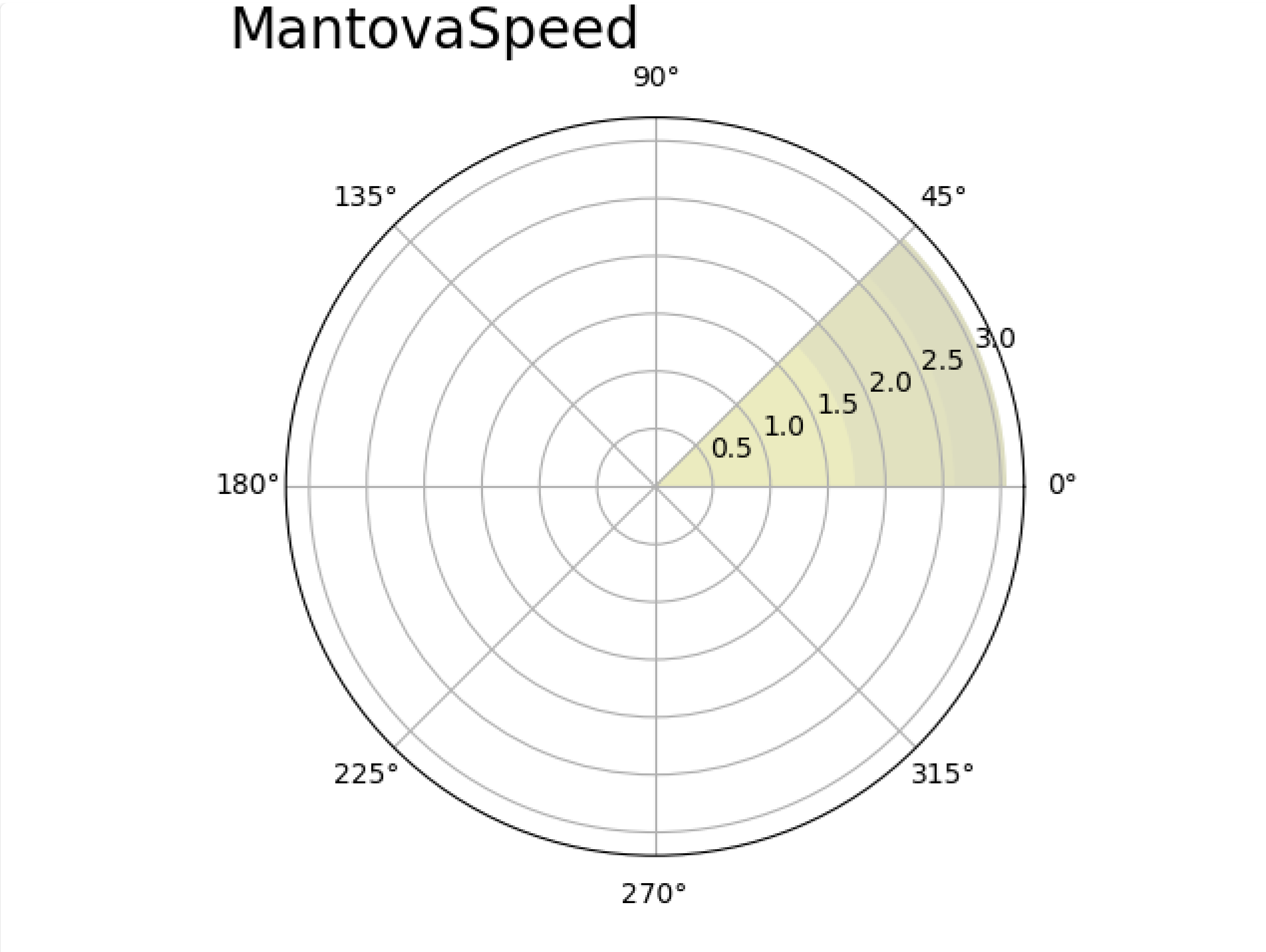


> *但是，事实上，画出了玫瑰图也会发现，数据不足而导致图像结果不明确*

平均风速柱状图

```
def RoseWindSpeed(city) :
    deg = np.arange(45, 361, 45)
    speed_avr = []
    for i in deg:
        speed_avr.append(city[(city["wind_deg"] > (i - 46)) & (city["wind_deg"]
    return speed_avr
```

```
def meanWindSpeed() :
    ravenna = pd.read_csv("WeatherData/mantova_270615.csv", usecols=["wind_deg"
    hist, bins = np.histogram(ravenna, 8, [0, 360])
    showRoseWind(RoseWindSpeed(ravenna), "MantovaSpeed", max(hist))
```



> *以上就是本实验的所有数据分析以及结论*