数据处理1

首先对数据进行清洗，去除重复值和空值，然后对数据进行分析，得到以下结果：

```
    df = pd.read_csv('MyData.csv', encoding='utf-8-sig')
    df = df.drop_duplicates()  # 去重
    df = df.dropna()  # 去空值
    zones_chinese = ['东城', '西城', '朝阳', '海淀']
```

（1）四个区的平均总价、最高总价、最低总价；

由于总价单位很统一，所以直接计算平均值即可

```
print("四个区的平均总价、最高总价、最低总价；")
    for zone in zones_chinese:
        print(zone)
        df_zone = df[df['zone_name'] == zone]
        print(df_zone['total_price'].mean())
        print(df_zone['total_price'].max())
        print(df_zone['total_price'].min())
```

结果如下

```
四个区的平均总价、最高总价、最低总价；
东城
892.6786666666666
3125.0
396.0
西城
670.68
2960.0
140.0
朝阳
838.0266666666666
2980.0
206.0
海淀
899.8333333333334
1789.0
438.0
```

（2）四个区的平均单价、最高单价、最低单价；

由于单价单位不统一，所以需要先将单价的单位统一，然后再计算平均值

```
    print("四个区的平均单价、最高单价、最低单价；")
    for zone in zones_chinese:
        print(zone)
        df_zone = df[df['zone_name'] == zone]
        df_zone['price_per_area'] = df_zone['price_per_area'].apply(lambda x:
x.replace('元/平', ''))
        df_zone['price_per_area'] = df_zone['price_per_area'].apply(lambda x:
x.replace(',', ''))
        df_zone['price_per_area'] = df_zone['price_per_area'].astype('int64')
        #print("sdsaasd",df_zone)
        print(df_zone['price_per_area'].mean())
        print(df_zone['price_per_area'].max())
        print(df_zone['price_per_area'].min())
```

结果如下

```
四个区的平均单价、最高单价、最低单价；
东城
112971.31333333334
179731
31172
西城
77636.77333333333
159600
33760
朝阳
88382.22666666667
152812
35100
海淀
91716.2
136364
32971
```

（3）按照房屋建成的年份，计算2000年以前、2000-2009.12.31、2010-至今，这三个时间段的平均单价。

```
 <div class="address">
                <div class="houseInfo"><span class="houseIcon"></span>2室1厅 |
61.85平米 | 南 北 | 精装 | 中楼层(共5层) | 2002年 | 板楼
                </div>
```

可以看到，房屋建成的年份在houseInfo中，所以需要先将houseInfo中的年份提取出来，然后再进行计算,但是其中有一些没有年份的数据，所以需要先将这些数据去除，然后再进行计算

```
    print("按照房屋建成的年份，计算2000年以前、2000-2009.12.31、2010-至今，这三个时间段
的平均单价。")
    def get_year(x):
        x=str(x).split("|")
        for i in x:
            if "年" in i:
                return i.replace("年","")
    df['houseInfo'] = df['houseInfo'].apply(get_year)
    #print(df['houseInfo'])
    #删除None
    df = df[df['houseInfo'] != 'None']
    df_cleaned = df.dropna()
    print(df_cleaned)
    df_cleaned['houseInfo'] = df_cleaned['houseInfo'].astype('int64')
    df_cleaned['price_per_area'] = df_cleaned['price_per_area'].apply(lambda x:
x.replace('元/平', ''))
    df_cleaned['price_per_area'] = df_cleaned['price_per_area'].apply(lambda x:
x.replace(',', ''))
    df_cleaned['price_per_area'] = df_cleaned['price_per_area'].astype('int64')
    df_2000 = df_cleaned[df_cleaned['houseInfo'] < 2000]
    df_2000_2009 = df_cleaned[(df_cleaned['houseInfo'] >= 2000) &
(df_cleaned['houseInfo'] <= 2009)]
    df_2010 = df_cleaned[df_cleaned['houseInfo'] >= 2010]
    print("2000 :",df_2000['price_per_area'].mean())
    print("2000-2009 :",df_2000_2009['price_per_area'].mean())
    print("2010 :",df_2010['price_per_area'].mean())
```

结果如下，然而奇怪的是，2010年的平均单价竟然比2000年的还要低，这可能是因为数据量太少，导致计算出来的平均值不准确。

```
2000 : 97191.45138888889
2000-2009 : 91281.98936170213
2010 : 71113.85714285714
```

2. 处理北京空气质量数据

使用pandas库读取北京空气质量数据

```
    df = pd.read_csv('beijing_17_18_aq.csv', encoding='utf-8-sig')
    # df = df.drop_duplicates()  # 去重
    # df = df.dropna()  # 去空值
    #这次不用去空值了，因为后面会处理
    print(df)
```

对HUMI、PRES、TEMP三列，进行线性插值处理。修改cbwd列中值为“cv”的单元格，其值用后项数据填充。

首先找出所有空值

```
    df_ads=df[df['HUMI'].isna()]
    print("HUMI :\n",df_ads)
    df_ads = df[df['PRES'].isna()]
    print("PRES :\n",df_ads)
    df_ads = df[df['TEMP'].isna()]
    print("TEMP :\n",df_ads)
```

结果如下

```
C:\Users\86182\AppData\Local\Programs\Python\Python311\python.exe
D:\workSpace\python\test1\bejingair.py
HUMI :
           No  year  month  day  hour  ...  TEMP  cbwd    Iws  precipitation
Iprec
45922  45923  2015      3   29    10  ...   NaN   NaN    NaN            0.0    0.0
47954  47955  2015      6   22     2  ...   NaN   NaN    NaN            0.0    0.0
49271  49272  2015      8   15    23  ...   NaN   NaN    NaN            0.0    0.0
51257  51258  2015     11    6    17  ...   2.0    NE  15.64            0.0    0.0
51258  51259  2015     11    6    18  ...   2.0    NE  20.56            0.0    0.0
...      ...   ...    ...  ...   ...  ...   ...   ...    ...            ...    ...
51605  51606  2015     11   21     5  ...   2.0    cv   0.45            0.1    1.1
51606  51607  2015     11   21     6  ...   1.0    SE   1.79            0.1    1.2
51607  51608  2015     11   21     7  ...   1.0    SE   4.92            0.2    1.4
51608  51609  2015     11   21     8  ...   1.0    SE   8.05            0.2    1.6
51891  51892  2015     12    3     3  ...   NaN   NaN    NaN            0.0    0.0

[339 rows x 18 columns]
PRES :
           No  year  month  day  hour  ...  TEMP  cbwd    Iws  precipitation
Iprec
45922  45923  2015      3   29    10  ...   NaN   NaN    NaN            0.0    0.0
47954  47955  2015      6   22     2  ...   NaN   NaN    NaN            0.0    0.0
49271  49272  2015      8   15    23  ...   NaN   NaN    NaN            0.0    0.0
51257  51258  2015     11    6    17  ...   2.0    NE  15.64            0.0    0.0
51258  51259  2015     11    6    18  ...   2.0    NE  20.56            0.0    0.0
...      ...   ...    ...  ...   ...  ...   ...   ...    ...            ...    ...
51605  51606  2015     11   21     5  ...   2.0    cv   0.45            0.1    1.1
51606  51607  2015     11   21     6  ...   1.0    SE   1.79            0.1    1.2
51607  51608  2015     11   21     7  ...   1.0    SE   4.92            0.2    1.4
51608  51609  2015     11   21     8  ...   1.0    SE   8.05            0.2    1.6
51891  51892  2015     12    3     3  ...   NaN   NaN    NaN            0.0    0.0

[339 rows x 18 columns]
TEMP :
           No  year  month  day  hour  ...  TEMP  cbwd  Iws  precipitation  Iprec
45922  45923  2015      3   29    10  ...   NaN   NaN  NaN            0.0    0.0
47954  47955  2015      6   22     2  ...   NaN   NaN  NaN            0.0    0.0
49271  49272  2015      8   15    23  ...   NaN   NaN  NaN            0.0    0.0
```

```
51328  51329  2015     11    9    16  ...   NaN    NaN   NaN            0.0    0.0
51891  51892  2015     12    3     3  ...   NaN    NaN   NaN            0.0    0.0
```

然后进行线性插值处理

```
    df['HUMI'] = df['HUMI'].interpolate()
    df['PRES'] = df['PRES'].interpolate()
    df['TEMP'] = df['TEMP'].interpolate()
```

再次输出后结果如下

```
[5 rows x 18 columns]
after interpolate
HUMI :
 Empty DataFrame
Columns: [No, year, month, day, hour, season, PM_Dongsi, PM_Dongsihuan,
PM_Nongzhanguan, PM_US Post, DEWP, HUMI, PRES, TEMP, cbwd, Iws, precipitation,
Iprec]
Index: []
PRES :
 Empty DataFrame
Columns: [No, year, month, day, hour, season, PM_Dongsi, PM_Dongsihuan,
PM_Nongzhanguan, PM_US Post, DEWP, HUMI, PRES, TEMP, cbwd, Iws, precipitation,
Iprec]
Index: []
TEMP :
 Empty DataFrame
Columns: [No, year, month, day, hour, season, PM_Dongsi, PM_Dongsihuan,
PM_Nongzhanguan, PM_US Post, DEWP, HUMI, PRES, TEMP, cbwd, Iws, precipitation,
Iprec]
Index: []
```

然后修改cbwd列中值为“cv”的单元格，其值用后项数据填充。

```
print("before bfill")
    df_ads = df[df['cbwd'].isna()]
    print("cbwd :\n", df_ads)
    df['cbwd'] = df['cbwd'].fillna(method='bfill')
    print("after bfill")
    df_ads = df[df['cbwd'].isna()]
    print("cbwd :\n",df_ads)
```

结果如下

```
before bfill
cbwd :
           No  year  month  day  hour  ...  TEMP  cbwd  Iws  precipitation  Iprec
45922  45923  2015      3   29    10  ...  16.0   NaN  NaN            0.0    0.0
47954  47955  2015      6   22     2  ...  22.0   NaN  NaN            0.0    0.0
49271  49272  2015      8   15    23  ...  24.0   NaN  NaN            0.0    0.0
51328  51329  2015     11    9    16  ...   6.0   NaN  NaN            0.0    0.0
51891  51892  2015     12    3     3  ...  -1.5   NaN  NaN            0.0    0.0

[5 rows x 18 columns]
after bfill
cbwd :
 Empty DataFrame
Columns: [No, year, month, day, hour, season, PM_Dongsi, PM_Dongsihuan,
PM_Nongzhanguan, PM_US Post, DEWP, HUMI, PRES, TEMP, cbwd, Iws, precipitation,
Iprec]
Index: []
```

能看到空值已经被填充了。