KNN

2018 年 2 月 23 日

0.1 数据读取

```
In [1]: import pandas as pd
        dc_all_listings = pd.read_csv('listings.csv')
        print(dc_all_listings.shape)
        dc_all_listings.head()
```

```
(3723, 92)
```

Out[1]:

	id	listing_url	scrape_id	last_scraped	\
0	7087327	https://www.airbnb.com/rooms/7087327	20151002231825	2015-10-03	
1	975833	https://www.airbnb.com/rooms/975833	20151002231825	2015-10-03	
2	8249488	https://www.airbnb.com/rooms/8249488	20151002231825	2015-10-03	
3	8409022	https://www.airbnb.com/rooms/8409022	20151002231825	2015-10-03	
4	8411173	https://www.airbnb.com/rooms/8411173	20151002231825	2015-10-03	

	name	\
0	Historic DC Condo-Walk to Capitol!	
1	Spacious Capitol Hill Townhouse	
2	Spacious/private room for single	
3	A wonderful bedroom with library	
4	Downtown Silver Spring	

	summary	\
0	Professional pictures coming soon! Welcome to ...	
1	NaN	
2	This is an ideal room for a single traveler th...	
3	Prime location right on the Potomac River in W...	
4	Hi travellers! I live in this peaceful spot, b...	

	space	\

```
0                                                 NaN
1  Beautifully renovated Capitol Hill townhouse. ...
2                                                 NaN
3                                                 NaN
4  This is a 750 sq ft 1 bedroom 1 bathroom.  Whi...

                                          description experiences_offered  \
0  Professional pictures coming soon! Welcome to ...                 none
1  Beautifully renovated Capitol Hill townhouse. ...                 none
2  This is an ideal room for a single traveler th...                 none
3  Prime location right on the Potomac River in W...                 none
4  Hi travellers! I live in this peaceful spot, b...                 none

                               neighborhood_overview        ...         \
0                                                 NaN       ...
1                                                 NaN       ...
2                                                 NaN       ...
3                                                 NaN       ...
4  Silver Spring is booming.  You can walk to a n...        ...

  review_scores_value requires_license license  \
0                 NaN                f     NaN
1                 9.0                f     NaN
2                 NaN                f     NaN
3                 NaN                f     NaN
4                 NaN                f     NaN

                jurisdiction_names instant_bookable cancellation_policy  \
0  DISTRICT OF COLUMBIA, WASHINGTON                f            flexible
1  DISTRICT OF COLUMBIA, WASHINGTON                f              strict
2                               NaN                f            flexible
3  DISTRICT OF COLUMBIA, WASHINGTON                f            flexible
4                               NaN                f            flexible

   require_guest_profile_picture require_guest_phone_verification  \
0                              f                                 f
1                              f                                 f
2                              f                                 f
3                              f                                 f
```

```
4                                f                                f

  calculated_host_listings_count reviews_per_month
0                             18               NaN
1                              1              2.11
2                              1              1.00
3                              1               NaN
4                              1               NaN

[5 rows x 92 columns]
```

```
In [2]: features = ['accommodates','bedrooms','bathrooms','beds','price','minimum_nights','maximum
        dc_listings = dc_all_listings[features]
        print(dc_listings.shape)
        dc_listings.head()
```

```
(3723, 8)
```

Out[2]:

	accommodates	bedrooms	bathrooms	beds	price	minimum_nights	maximum_nights	number_of_reviews
0	4	1.0	1.0	2.0	$160.00	1	1125	0
1	6	3.0	3.0	3.0	$350.00	2	30	65
2	1	1.0	2.0	1.0	$50.00	2	1125	1
3	2	1.0	1.0	1.0	$95.00	1	1125	0
4	4	1.0	1.0	1.0	$50.00	7	1125	0

数据特征：

- accommodates: 可以容纳的旅客
- bedrooms: 卧室的数量
- bathrooms: 厕所的数量
- beds: 床的数量
- price: 每晚的费用
- minimum_nights: 客人最少租了几天
- maximum_nights: 客人最多租了几天
- number_of_reviews: 评论的数量

0.2 各个特征的描述统计

In [3]:
```
# dc_listings.insert (5,'price_num',dc_listings['price'])
# print ("insert")

#import string
#for i in dc_listings['price_num']:
#    dc_listings[5,i].split('$')[0]

dc_listings['price'] = dc_listings.price.str.replace("\$|,",'').astype(float)

#dc_listings['price_num'].replace('$','')
print (dc_listings.head())

print (dc_listings.describe())
```

```
   accommodates  bedrooms  bathrooms  beds  price  minimum_nights  \
0             4       1.0        1.0   2.0  160.0               1
1             6       3.0        3.0   3.0  350.0               2
2             1       1.0        2.0   1.0   50.0               2
3             2       1.0        1.0   1.0   95.0               1
4             4       1.0        1.0   1.0   50.0               7

   maximum_nights  number_of_reviews
0            1125                  0
1              30                 65
2            1125                  1
3            1125                  0
4            1125                  0
       accommodates     bedrooms    bathrooms         beds        price  \
count   3723.000000  3702.000000  3696.000000  3712.000000  3723.000000
mean       3.195004     1.210157     1.256358     1.643319   149.165995
std        2.012216     0.839851     0.585539     1.182117   140.110699
min        1.000000     0.000000     0.000000     1.000000    10.000000
25%        2.000000     1.000000     1.000000     1.000000    85.000000
50%        2.000000     1.000000     1.000000     1.000000   115.000000
75%        4.000000     1.000000     1.000000     2.000000   165.000000
max       16.000000    10.000000     8.000000    16.000000  2822.000000

       minimum_nights  maximum_nights  number_of_reviews
count     3723.000000    3.723000e+03        3723.000000
```

```
mean          2.250067    5.803069e+05          15.306742
std           3.622879    3.519552e+07          29.645586
min           1.000000    1.000000e+00           0.000000
25%           1.000000    1.200000e+02           1.000000
50%           2.000000    1.125000e+03           4.000000
75%           3.000000    1.125000e+03          16.000000
max         180.000000    2.147484e+09         362.000000
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#in
```

```
In [4]: import numpy as np

        our_acc_value = 3

        dc_listings['distance']= np.abs(dc_listings.accommodates - our_acc_value)
        dc_listings.distance.value_counts().sort_index()
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#in
  """
```

```
Out[4]: 0      461
        1     2294
        2      503
        3      279
        4       35
        5       73
        6       17
        7       22
        8        7
        9       12
```

```
10       2
11       4
12       6
13       8
Name: distance, dtype: int64
```

0.3 各个特征与价格的相关性

```
In [5]: dc_listings.corr()
```

Out[5]:

	accommodates	bedrooms	bathrooms	beds	price	\
accommodates	1.000000	0.719362	0.580176	0.811764	0.503120	
bedrooms	0.719362	1.000000	0.697185	0.715324	0.521759	
bathrooms	0.580176	0.697185	1.000000	0.585579	0.495966	
beds	0.811764	0.715324	0.585579	1.000000	0.463642	
price	0.503120	0.521759	0.495966	0.463642	1.000000	
minimum_nights	0.013727	0.024945	0.033149	0.003847	0.014055	
maximum_nights	0.022959	0.015532	-0.007106	0.018866	0.014744	
number_of_reviews	0.009725	-0.051872	-0.037448	-0.009338	-0.084339	
distance	0.766304	0.672660	0.580009	0.711561	0.405729	

	minimum_nights	maximum_nights	number_of_reviews	distance
accommodates	0.013727	0.022959	0.009725	0.766304
bedrooms	0.024945	0.015532	-0.051872	0.672660
bathrooms	0.033149	-0.007106	-0.037448	0.580009
beds	0.003847	0.018866	-0.009338	0.711561
price	0.014055	0.014744	-0.084339	0.405729
minimum_nights	1.000000	-0.001159	-0.020709	0.004866
maximum_nights	-0.001159	1.000000	-0.001862	0.018091
number_of_reviews	-0.020709	-0.001862	1.000000	-0.021972
distance	0.004866	0.018091	-0.021972	1.000000

```
In [6]: dc_listings = dc_listings.sample(frac=1,random_state=0)
        dc_listings = dc_listings.sort_values('distance')
        dc_listings.price.head()
```

```
Out[6]: 2645     75.0
        2825    120.0
        2145     90.0
        2541     50.0
        3349    105.0
        Name: price, dtype: float64
```

计算平均价格

```
In [7]: mean_price = dc_listings.price.iloc[:5].mean()
        mean_price
```

```
Out[7]: 88.0
```

0.4 模型的评估

首先制定好训练集和测试集
loc 是根据 dataframe 的具体标签选取列，而 iloc 是根据标签所在的位置

```
In [8]: dc_listings.drop("distance",axis = 1)

        #2793 行前的作为训练集，之后剩余做为测试集
        train_df = dc_listings.copy().iloc[:2792]
        test_df = dc_listings.copy().iloc[2792:]

        #train_df.head()
```

基于单变量预测价格

```
In [9]: def predict_price(new_listing_value,feature_column):
            temp_df = train_df
            temp_df['distance']= np.abs(dc_listings[feature_column] - new_listing_value)
            temp_df = temp_df.sort_values('distance')
            knn_5 = temp_df.price.iloc[:5]
            predict_price = knn_5.mean()
            return (predict_price)

        test_df['predicted_price']=test_df.accommodates.apply(predict_price,feature_column = 'acco

        test_df['squared_error'] = (test_df['predicted_price'] - test_df['price'])**(2)
        mse = test_df['squared_error'].mean()
        rmse = mse**(1/2)
        rmse
```

```
Out[9]: 212.98927967051543
```

0.5 不同的变量效果会不会不同呢？

test_df[‘predicted_price’] = test_df.accommodates.apply(predict_price,feature_column=feature)

```
In [10]: for feature in ['accommodates','bedrooms','bathrooms','number_of_reviews']:
             test_df['predicted_price'] = test_df[feature].apply(
                 predict_price,feature_column=feature)
             test_df['squared_error'] = (test_df['predicted_price'] - test_df['price'])**(2)
             mse = test_df['squared_error'].mean()
             rmse = mse ** (1/2)
             print("RMSE for the {} column: {}".format(feature,rmse))
```

```
RMSE for the accommodates column: 212.98927967051543
RMSE for the bedrooms column: 199.80935328065033
RMSE for the bathrooms column: 230.24716705684227
RMSE for the number_of_reviews column: 235.91327066995507
```

0.6 综合利用所有的信息来一起进行测试

StandardScaler 作用：去均值和方差归一化。且是针对每一个特征维度来做的，而不是针对样本。
normalized_listings 就是标准化后的数据集

```
In [11]: import pandas as pd
         from sklearn.preprocessing import StandardScaler
         features = ['accommodates','bedrooms','bathrooms','beds','price','minimum_nights','maximu

         dc_listings = pd.read_csv('listings.csv')

         dc_listings = dc_listings[features]

         dc_listings['price'] = dc_listings.price.str.replace("\$|,",'').astype(float)

         dc_listings = dc_listings.dropna()


         dc_listings[features] = StandardScaler().fit_transform(dc_listings[features])

         normalized_listings = dc_listings

         print(dc_listings.shape)

         normalized_listings.head()
```

```
(3671, 8)
```

```
Out[11]:    accommodates  bedrooms  bathrooms      beds     price  minimum_nights  \
         0      0.401420 -0.249501  -0.439211  0.297386  0.081119       -0.341421
         1      1.399466  2.129508   2.969551  1.141704  1.462622       -0.065047
         2     -1.095648 -0.249501   1.265170 -0.546933 -0.718699       -0.065047
         3     -0.596625 -0.249501  -0.439211 -0.546933 -0.391501       -0.341421
         4      0.401420 -0.249501  -0.439211 -0.546933 -0.718699        1.316824

            maximum_nights  number_of_reviews
         0       -0.016575          -0.516779
         1       -0.016606           1.706767
         2       -0.016575          -0.482571
         3       -0.016575          -0.516779
         4       -0.016575          -0.516779
```

```
In [12]: norm_train_df = normalized_listings.copy().iloc[0:2792]
         norm_test_df = normalized_listings.copy().iloc[2792:]
```

scipy 中已经有现成的距离的计算工具了

```
In [13]: from scipy.spatial import distance

         first_listing = normalized_listings.iloc[0][['accommodates', 'bathrooms']]
         fifth_listing = normalized_listings.iloc[20][['accommodates', 'bathrooms']]
         first_fifth_distance = distance.euclidean(first_listing, fifth_listing)
         first_fifth_distance
```

```
Out[13]: 3.7230196040170322
```

0.7 多变量 KNN 模型

语法：scipy.spatial.distance.cdist(XA, XB, metric='euclidean', p=None, V=None, VI=None, w=None)，该函数用于计算两个输入集合的距离，通过 metric 参数指定计算距离的不同方式得到不同的距离度量值

metric 的取值如下：

- braycurtis
- canberra
- chebyshev：切比雪夫距离
- cityblock
- correlation：相关系数
- cosine：余弦夹角
- dice

- euclidean：欧式距离
- hamming：汉明距离
- jaccard：杰卡德相似系数
- kulsinski
- mahalanobis：马氏距离
- matching
- minkowski：闵可夫斯基距离
- rogerstanimoto
- russellrao
- seuclidean：标准化欧式距离
- sokalmichener
- sokalsneath
- sqeuclidean
- wminkowski
- yule

常见的欧氏距离计算：

```
In [14]: from scipy.spatial.distance import cdist
         import numpy as np
         x1 =np.array([(1,3),(2,4),(5,6)])
         x2 =[(3,7),(4,8),(6,9)]
         cdist(x1,x2,metric='euclidean')
```

```
Out[14]: array([[4.47213595, 5.83095189, 7.81024968],
                [3.16227766, 4.47213595, 6.40312424],
                [2.23606798, 2.23606798, 3.16227766]])
```

解析上述计算过程：结果数组中的第一行数据表示的是 x1 数组中第一个元素点与 x2 数组中各个元素点的距离，计算两点之间的距离

```
In [15]: def predict_price_multivariate(new_listing_value,feature_columns):
             temp_df = norm_train_df
             temp_df['distance'] = distance.cdist(temp_df[feature_columns],[new_listing_value[feat
             temp_df = temp_df.sort_values('distance')
             knn_5 = temp_df.price.iloc[:5]
             predicted_price = knn_5.mean()
             return(predicted_price)
```

```
In [16]: cols = ['accommodates', 'bathrooms']
         norm_test_df['predicted_price'] = norm_test_df[cols].apply(predict_price_multivariate,fea
         norm_test_df['squared_error'] = (norm_test_df['predicted_price'] - norm_test_df['price'])
```

```
mse = norm_test_df['squared_error'].mean()
rmse = mse ** (1/2)
print(rmse)
```

```
0.7894063922577537
```

0.8 使用 Sklearn 来完成 KNN

```
In [17]: from sklearn.neighbors import KNeighborsRegressor
         cols = ['accommodates','bedrooms']
         knn = KNeighborsRegressor()
         knn.fit(norm_train_df[cols], norm_train_df['price'])
         two_features_predictions = knn.predict(norm_test_df[cols])
```

```
In [18]: from sklearn.metrics import mean_squared_error

         two_features_mse = mean_squared_error(norm_test_df['price'], two_features_predictions)
         two_features_rmse = two_features_mse ** (1/2)
         print(two_features_rmse)
```

```
0.8426824704818202
```

加入更多的特征

```
In [19]: knn = KNeighborsRegressor()

         cols = ['accommodates','bedrooms','bathrooms','beds','minimum_nights','maximum_nights','n

         knn.fit(norm_train_df[cols], norm_train_df['price'])
         four_features_predictions = knn.predict(norm_test_df[cols])
         four_features_mse = mean_squared_error(norm_test_df['price'], four_features_predictions)
         four_features_rmse = four_features_mse ** (1/2)
         four_features_rmse
```

```
Out[19]: 0.8243838530880285
```