Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения, материалов и транспорта

[**Высшая школа автоматизации и робототехники**](https://www.spbstu.ru/structure/vysshaya_shkola_avtomatizatsii_i_robototekhniki/)

**ЛАБОРАТОРНАЯ РАБОТА №1**

**«Анализ и работа с данными в машинном обучении»**

по дисциплине

«Математические методы интеллектуальных технологий»

Выполнил

студент гр.3331506/00401 <*подпись*> Комаров А.Е.

Руководитель

<*подпись*> Орлова С.Р.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург

2020

***Цель работы:*** освоение приёмов анализа и работы с данными

***Задачи работы:***

* освоить основные методы библиотек p*andas, numpy, matplotlib, seaborn*,
* научиться анализировать данные: находить выбросы, определять актуальность признаков, выявлять закономерности и т. д.,
* освоить методы предобработки признаков и формирования обучающих выборок с библиотеками *scikit-learn* и *pandas*.

# Ход работы и результаты

# Часть 1. Ответы на вопросы

1. Сколько записей в базе?

num\_rec\_table = DataFrame.shape[0] # Кол-во записей в таблице  
print('num\_rec\_table = ', num\_rec\_table)

#Output

num\_rec\_table = 38821

1. Какие типы жилья и районы присутствуют в базе?

type\_of\_room = DataFrame['room\_type'].unique() # Все типы жилья  
type\_of\_neighbourhood\_group = DataFrame['neighbourhood\_group'].unique() # Все типы районов  
print('type\_of\_room = ', type\_of\_room)  
print('type\_of\_neighbourhood\_group = ', type\_of\_neighbourhood\_group)

#Output

type\_of\_room = ['Private room' 'Entire home/apt' 'Shared room']

type\_of\_neighbourhood\_group = ['Brooklyn' 'Manhattan' 'Queens' 'Staten Island' 'Bronx']

1. Статистические значения цены для каждого типа жилья.

room\_type\_stat = DataFrame.groupby(['room\_type']).describe()['price']  
print('room\_type\_stat', '\n', room\_type\_stat)

Таблица 1 - Статистические значения цены для каждого типа жилья

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **room\_type** | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **Entire home/apt** | 20321.0 | 196.315929 | 223.700905 | 0.0 | 116.0 | 151.0 | 220.0 | 10000.0 |
| **Private room** | 17654.0 | 83.985272 | 142.279361 | 0.0 | 50.0 | 70.0 | 94.0 | 10000.0 |
| **Shared room** | 846.0 | 63.213948 | 95.186458 | 0.0 | 33.0 | 45.0 | 70.0 | 1800.0 |

Значения полей:

* count - количество не-NA/null записей.
* mean – среднее значение.
* std - стандартное отклонение.
* min – минимальное значение.
* 25%/50%/75% - соответствующие процентили (мера, в которой процентное значение общих значений равно этой мере или меньше ее).

На основании представленных данных можно сделать следующие выводы: количество комнат, которые делят, (*shared room*) заметно меньше сдающихся целиком комнат (*private room*) и сдающихся апартаментов (*entire home/apt*); стоимость апартаментов в среднем заметно больше стоимости комнат.

1. Статистические значения цены для каждого района.

neighbourhood\_group\_type\_stat = DataFrame.groupby(['neighbourhood\_group']).describe()['price']  
neighbourhood\_group\_type\_stat.to\_excel('4.xlsx')  
print('neighbourhood\_group\_type\_stat', '\n', neighbourhood\_group\_type\_stat)

Таблица 2 – Статистические значения цены для каждого района

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **neighbourhood\_group** | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **Bronx** | 875.0 | 79.558857 | 63.999121 | 0.0 | 45.0 | 65.0 | 95.0 | 800.0 |
| **Brooklyn** | 16439.0 | 121.463289 | 169.887979 | 0.0 | 60.0 | 94.0 | 150.0 | 10000.0 |
| **Manhattan** | 16621.0 | 180.071596 | 226.474061 | 10.0 | 90.0 | 140.0 | 200.0 | 9999.0 |
| **Queens** | 4572.0 | 95.783683 | 168.395210 | 10.0 | 50.0 | 72.0 | 109.0 | 10000.0 |
| **Staten Island** | 314.0 | 89.964968 | 65.864044 | 13.0 | 50.0 | 75.0 | 105.0 | 625.0 |

На основании представленных данных можно сделать следующие выводы: в районах Манхэттен и Бруклин заметно больше количество предложений, чем в остальных районах; более того эти районы в среднем являются самыми дорогими (однако в данном случае Манхэттен заметно превосходит и сам Бруклин в этом параметры).

1. Средняя стоимость жилья в зависимости от типа жилья и района

mean\_price = pandas.pivot\_table(DataFrame,  
 columns='room\_type',  
 index='neighbourhood\_group',  
 values='price',  
 aggfunc=numpy.mean)  
print('mean\_price', '\n', mean\_price)  
mean\_price.to\_excel('5.xlsx')

Таблица 3 – Средней стоимости жилья от типа жилья и района

|  |  |  |  |
| --- | --- | --- | --- |
| **neighbourhood\_group** | **Entire home/apt** | **Private room** | **Shared room** |
| **Bronx** | 120,9383 | 57,09351 | 56,93023 |
| **Brooklyn** | 171,2819 | 73,34956 | 45,45862 |
| **Manhattan** | 230,201 | 106,6075 | 77,98034 |
| **Queens** | 139,2256 | 69,30284 | 64,46711 |
| **Staten Island** | 125,3667 | 57,58491 | 57,6 |

На основании таблицы можно сказать, что выводы, сделанные ранее, по поводу зависимости стоимости жилья от района и от типа жилья подтверждаются.

1. Максимальное значение минимального количества ночей

max\_minimum\_nights = numpy.max(DataFrame['minimum\_nights'])  
print('max\_minimum\_nights = ', max\_minimum\_nights)

#Output

max\_minimum\_nights = 1250

Я считаю, что адекватным стоит считать значение максимального значения минимального количества ночей около 20-30, так как предложения жилья, соответствующие большим значениям этого параметры, стоит относить к долгосрочной аренде жилья, для которых могут действовать другие закономерности.

1. Минимальная, максимальная и средняя цена аренды жилья в трёх

самых дорогих соседствах Бруклина

Ниже представлен результат работы фрагмента программы. Для каждого соседства Бруклина. Порядок вычисленных параметров следующий: минимальное, среднее, максимальное значения.

#Output

neighb\_gr\_1 = [10, 144.4353348729792, 10000]

neighb\_gr\_2 = [35, 176.15833333333333, 8000]

neighb\_gr\_3 = [15, 106.65727699530517, 7500]

1. Рейтинг слов из названий по популярности

in 13282

room 7545

private 5792

bedroom 5640

apartment 4827

cozy 3878

to 3213

the 3086

brooklyn 3012

studio 2952

apt 2867

spacious 2732

with 2460

of 2409

and 2308

manhattan 2307

east 2284

park 2116

sunny 2077

beautiful 1857

williamsburg 1844

near 1800

village 1641

heart 1638

large 1497

# Часть 2. Анализ и преобразование признаков

1. Карта районов

На рисунке 1 представлен график, где по осям *х* и *у* расположены долгота и широта соответственно, а цветом помечены районы. На рисунке 2 представлена маска плотности предложений на карте.

|  |  |
| --- | --- |
|  |  |
| Рисунок 1 | Рисунок 2 |

1. Гистограммы признаков

Гистограммы минимального количества ночей, количества отзывов за месяц, даты последнего отзыва, района жилья представлены на рисунках 3-6 соответственно.

|  |  |
| --- | --- |
|  |  |
| Рисунок 3 | Рисунок 4 |
|  |  |
| Рисунок 5 | Рисунок 6 |

1. Новые признаки

Далее представлен фрагмент кода, реализующий получение трех новых признаков: длина названия в символах *len\_name*, число слов *nb\_words*, число популярных слов *nb\_pop\_words* (из пункта 8).

def search\_for\_popular\_words\_in\_message(data, most\_occur\_names):  
 data = data.lower()  
 number = 0  
 for i in range(25):  
 for word in data:  
 if word == most\_occur\_names[i][0]:  
 number = number + 1  
 return number

DataFrame.loc[:, 'len\_name'] = DataFrame['name'].apply(lambda x: len(str(x)))  
DataFrame.loc[:, 'number\_of\_word'] = DataFrame['name'].apply(lambda x: len(str(x).split()))  
DataFrame.loc[:, 'number\_\_of\_popular\_word'] = \  
 DataFrame['name'].apply(lambda x: af.search\_for\_popular\_words

1. Преобразование цены в категориальный признак

Далее представлен фрагмент кода, реализующий получение признака *price\_group* путем выделения 8-ми ценовых категорий жилья.

column\_without\_nan = DataFrame['price'].fillna(-1)  
labels = ['0 - 10', '10 - 20', '20 - 40', '40 - 80', '80 - 160', '160 - 320', '320 - 640', '640 - 800']  
price\_group = pandas.cut(column\_without\_nan,  
 bins=[0, 10, 20, 40, 80, 160, 320, 640, DataFrame['price'].max()],  
 labels=labels)  
DataFrame.loc[:, 'price\_group'] = price\_group

На рисунке 7 представлена гистограмма признака *price\_group*.

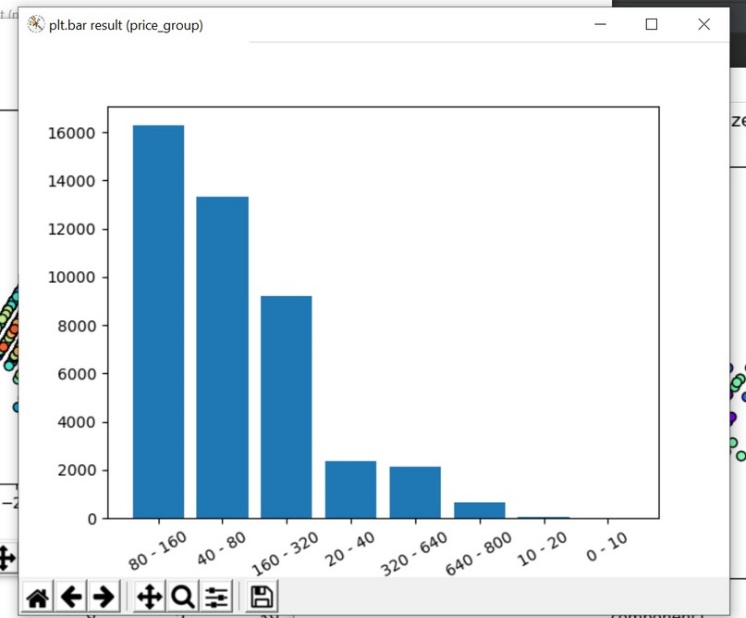


Рисунок 7

1. Метод главных компонент

В качестве целевой переменной выбран район *neighbourhood\_group*, в качестве же признаков взяты: тип жилья, цена, координаты широты, координаты долготы, длина названия в символах, число слов, число популярных слов.

Ниже представлен фрагмент кода, реализующий метод главных компонент для понижения размерности до 2.

DataFrame.loc[:, 'type\_of\_neighbourhood\_group'] = \  
 DataFrame['neighbourhood\_group'].apply(lambda x: af.get\_number\_in\_future(x, type\_of\_neighbourhood\_group))  
DataFrame.loc[:, 'type\_of\_room'] = \  
 DataFrame['room\_type'].apply(lambda x: af.get\_number\_in\_future(x, type\_of\_room))  
  
pca = PCA(n\_components = 2)  
df\_labels = ['type\_of\_room', 'latitude', 'longitude', 'price',  
 'len\_name', 'number\_of\_word', 'number\_\_of\_popular\_word']  
df = DataFrame[df\_labels]  
pca.fit(df)  
data\_reduced = pca.transform(df)  
af.scatter\_plot\_func(DataFrame, data\_reduced, 'neighbourhood\_grou

На рисунке 8 приведена диаграмма рассеяния, в которой по осям расположены получившиеся компоненты, а целевая переменная, район *neighbourhood\_group*, обозначен цветом.

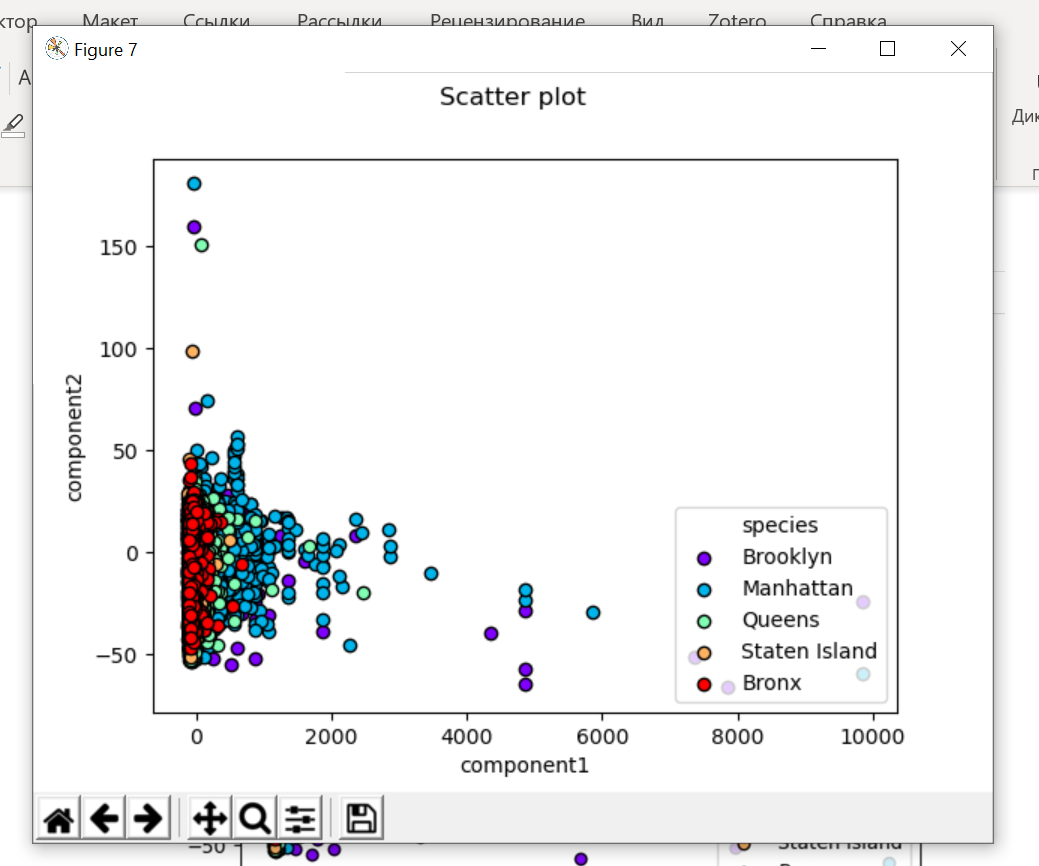


Рисунок 8

1. Нормализация и стандартизация

На рисунках 9 и 10 приведены диаграммы рассеяния, в которой по осям расположены получившиеся компоненты после нормализации и стандартизации соответственно, а целевая переменная, район *neighbourhood\_group*, обозначен цветом.

|  |  |
| --- | --- |
|  |  |
| Рисунок 9 | Рисунок 10 |

По результатам проведения нормализации и стандартизации признаков перед применением метода главных компонент диаграмма рассеяния изменилась. После стандартизации более явно выражены выбросы, которые расположены вдалеке от основного облака объектов. Также после стандартизация лучше можно выделить некоторые границы областей, например, достаточно четко видно граница между объектами типа Бронкс и Стэйтен-Айленд, в то время как после нормализации все объекты разных типов расположены в общем облаке объектов.

1. Построение матрицы корреляции

До построения матрицы корреляции можно сделать предположение, о связи между собой признаков: длина названия в символах, число слов, число популярных слов. Так как из большей длины названия в символах напрямую следует, что название содержит большее число слов и с большей вероятностью название содержит популярные слова. Помимо этого, по результатам статистического анализа типов жилья, представленного в пункте 3, можно предположить о связи цены и типа жилья.

Ниже представлен фрагмент кода, реализующий построение матрицы корреляции признаков с цветовой индикацией.

plt.figure()  
corr = df.corr()  
sns\_hmap = sns.heatmap(corr)  
sns\_hmap.set\_title("correlation PANDAS + SEABORN")

На рисунке 11 представлена матрица корреляции признаков с цветовой индикацией.

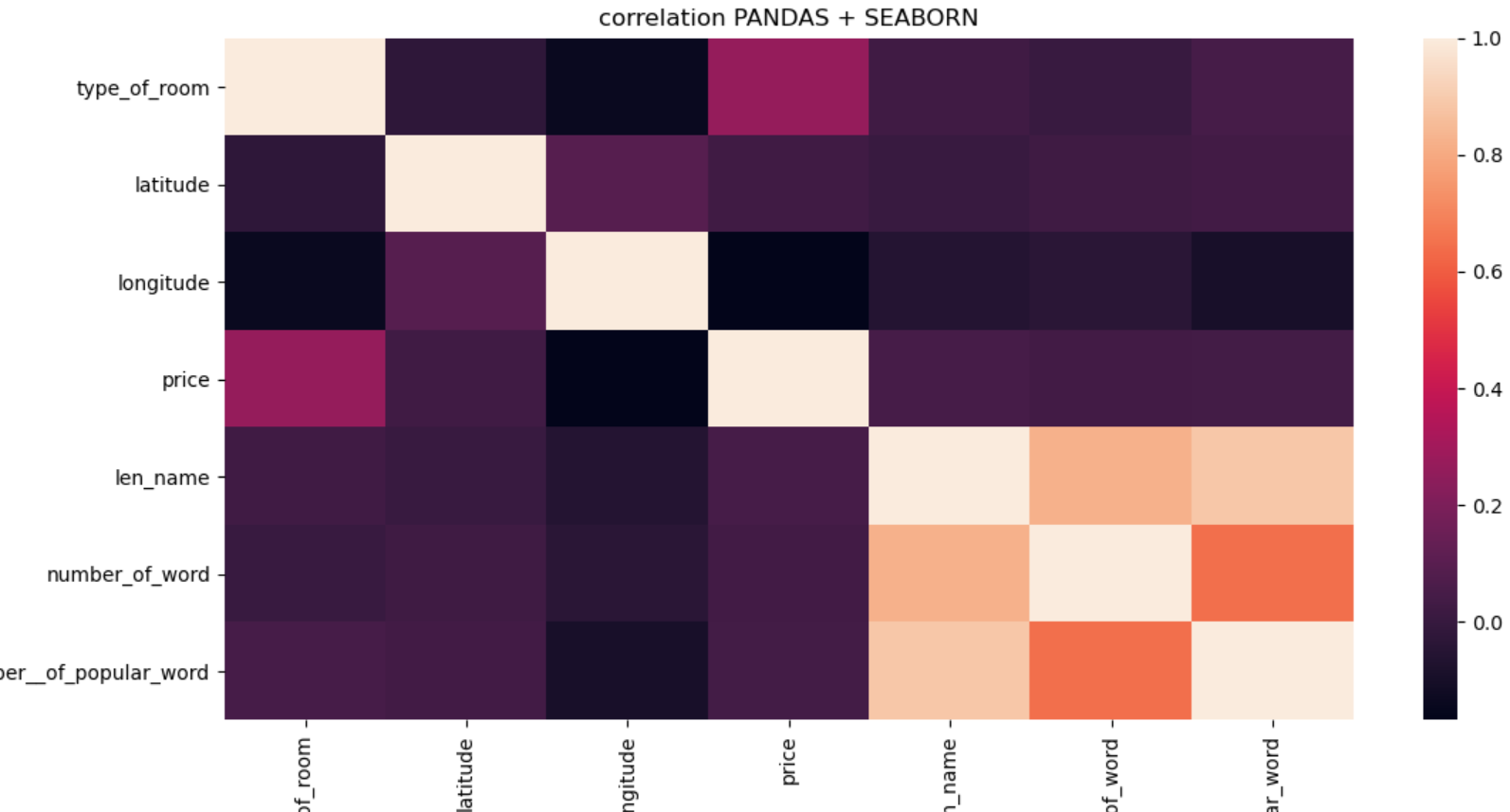


Рисунок 11

На основании представленного рисунка можно подтвердить выдвинутые ранее предложения о корреляции признаков длина названия в символах, число слов, число популярных слов, о чем свидетельствуют светлые ячейки на пресечениях этих признаков. Помимо этого, также есть корреляция между типом комнаты и ценой: меньшая, чем описанная ранее, но большая, чем между всеми остальными признаками, о чем свидетельствует сиреневая ячейка на пересечении между типом комнаты и ценой, в то время как все остальные ячейки окрашены фиолетовым цветом.

# Часть 3. Формирование выборок

1. Разбиения с помощью генератора разбиений *KFold*

Ниже представлен фрагмент кода, реализующий разбиения с помощью генератора разбиений *KFold*.

df\_stand = pandas.DataFrame({df\_labels[0]: df\_stand[:,0],df\_labels[1]: df\_stand[:,1],  
 df\_labels[2]: df\_stand[:, 2], df\_labels[3]: df\_stand[:, 3],  
 df\_labels[4]: df\_stand[:, 4], df\_labels[5]: df\_stand[:, 5],  
 df\_labels[6]: df\_stand[:, 6]})  
df\_target = pandas.DataFrame({'neighbourhood\_group': DataFrame['neighbourhood\_group']})  
  
print('PART 3')  
kf = KFold(n\_splits=3, shuffle=False, random\_state=None)  
for ikf, (train\_index, test\_index) in enumerate(kf.split(df\_stand)):  
 X\_train, X\_test = df\_stand.to\_numpy()[train\_index], df\_stand.to\_numpy()[test\_index]  
 y\_train, y\_test = df\_target.to\_numpy()[train\_index], df\_target.to\_numpy()[test\_index]  
 print('Iteration: ', ikf)  
 print(X\_train, y\_train)  
 print(X\_test, y\_test)  
 af.build\_bar\_chart(df\_stand, 'price', name=str(ikf) + ' ', change\_x\_ticks=False, angle=0)  
 af.build\_bar\_chart(df\_target, 'neighbourhood\_group', name=str(ikf) + ' ', change\_x\_ticks=False, angle=0)

На рисунках 12-14 представлены гистограммы цен и районов каждого разбиения.

|  |  |
| --- | --- |
|  |  |
| Рисунок 13 | |
|  |  |
| Рисунок 14 | |
|  |  |
| Рисунок 15 | |

# Заключение

По результатам проделанной работы можно сделать следующие выводы:

1. Данные требуют качественной фильтрации от некорректных значений (NA, null, смысловые ошибки).

Записи, содержащие эти ошибки, можно не использовать при обучении при достаточном объеме корректных записей, или заменять матожиданием/медианой признака.

1. Данные требуют качественной фильтрации от выбросов.
2. Вероятно, качество работы алгоритмов обучения на стандартизованных данных будет выше, чем на необработанных и нормализованных.
3. Для качественной работы алгоритмов обучения требуется выбирать признаки, хорошо коррелирующие с целевым признаком.