**Лабораторная работа №3: "Обработка признаков (часть 2)"**

**ИУ5-21М**

**Ян Тяньци**  
**Задание**

Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных.

Для выполнения следующих пунктов можно использовать несколько различных наборов данных

Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

масштабирование признаков (не менее чем тремя способами);  
обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);  
обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);  
отбор признаков: один метод из группы методов фильтрации (filter methods); один метод из группы методов обертывания (wrapper methods); один метод из группы методов вложений (embedded methods).

[2]

**from** google.colab **import** drive  
drive.mount('/content/drive', force\_remount=True)  
data\_dir = "/content/drive/MyDrive/dataset/bitcoin/BitCoin.csv"

Mounted at /content/drive

[3]

**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**import** seaborn **as** sns  
**import** datetime  
**from** sklearn.datasets **import** load\_boston  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.preprocessing **import** MinMaxScaler  
**from** sklearn.preprocessing **import** RobustScaler  
**import** warnings  
warnings.simplefilter("ignore", UserWarning)

[4]

dataset = pd.read\_csv(data\_dir)

[5]

dataset.head()

[6]

X=dataset.drop('Close',axis=1)  
y=dataset['Close']

[7]

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y)

[8]

X\_train=X\_train.drop(['Unnamed: 0','Date'],axis=1)  
X\_test=X\_test.drop(['Unnamed: 0','Date'],axis=1)  
X\_train

**Масштабирование признаков**

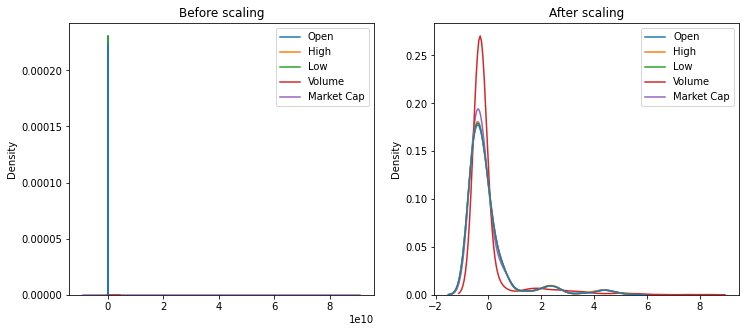
Методом **Z-оценки**:

[9]

**def** arr\_df(a):  
    df = pd.DataFrame(a, columns=X\_train.columns)  
    **return** df  
scaler1 = StandardScaler()  
scaled\_X\_1 = arr\_df(scaler1.fit\_transform(X\_train))  
scaled\_X\_1

[10]

**def** data\_visualize(columns, df1, df2, label1, label2):  
  fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))  
  ax1.set\_title(label1)  
  sns.kdeplot(data=df1[columns], ax=ax1)  
  ax2.set\_title(label2)  
  sns.kdeplot(data=df2[columns], ax=ax2)  
  plt.show()  
  
data\_visualize(X\_train.columns, X\_train, scaled\_X\_1, 'Before scaling', 'After scaling')



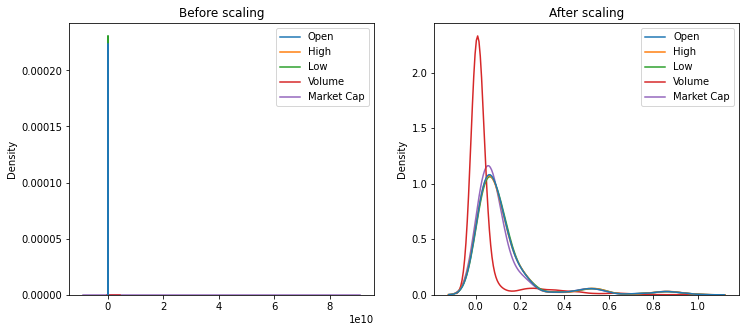
Методом **MinMaxScaler**:

[11]

scaler2 = MinMaxScaler()  
scaled\_X\_2 = arr\_df(scaler2.fit\_transform(X\_train))  
scaled\_X\_2

[12]

data\_visualize(X\_train.columns, X\_train, scaled\_X\_2, 'Before scaling', 'After scaling')



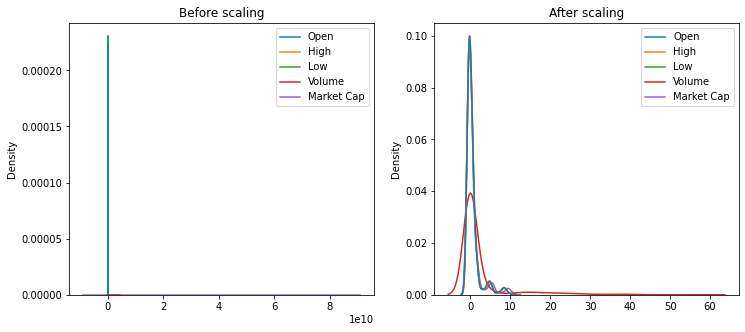
Методом **RobustScaler**:

[13]

scaler3 = RobustScaler()  
scaled\_X\_3 = arr\_df(scaler3.fit\_transform(X\_train))  
scaled\_X\_3

[14]

data\_visualize(X\_train.columns, X\_train, scaled\_X\_3, 'Before scaling', 'After scaling')



**Обработка выбросов для числовых признаков**

**Удаление выборсов**

[15]

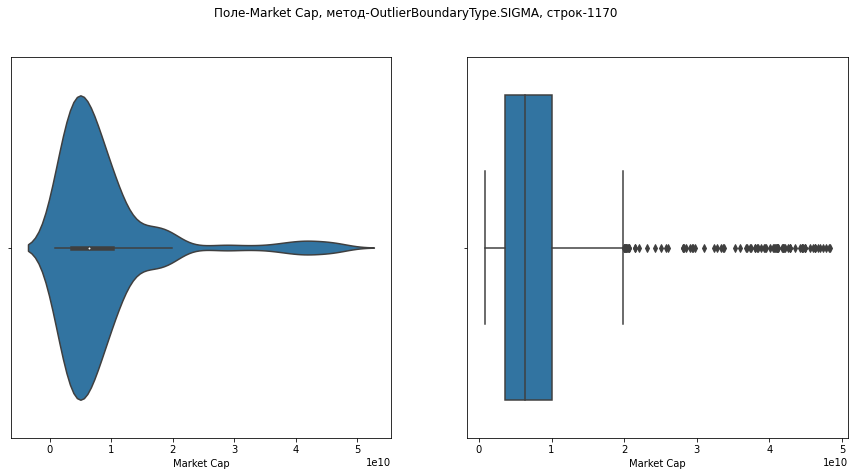
**def** plot\_for\_analys(df, variable, title):  
    fig, ax = plt.subplots(figsize=(15,7))  
    plt.subplot(1, 2, 1)  
    sns.violinplot(x=df[variable])      
    plt.subplot(1, 2, 2)  
    sns.boxplot(x=df[variable])    
    fig.suptitle(title)  
    plt.show()

[16]

**from** enum **import** Enum  
**class** OutlierBoundaryType(Enum):  
    SIGMA = 1  
**def** get\_outlier\_boundaries(df, col, outlier\_boundary\_type: OutlierBoundaryType):  
    **if** outlier\_boundary\_type == OutlierBoundaryType.SIGMA:  
        K1 = 3  
        lower\_boundary = df[col].mean() - (K1 \* df[col].std())  
        upper\_boundary = df[col].mean() + (K1 \* df[col].std())  
  
    **else**:  
        **raise** NameError('Unknown Outlier Boundary Type')  
          
    **return** lower\_boundary, upper\_boundary

[17]

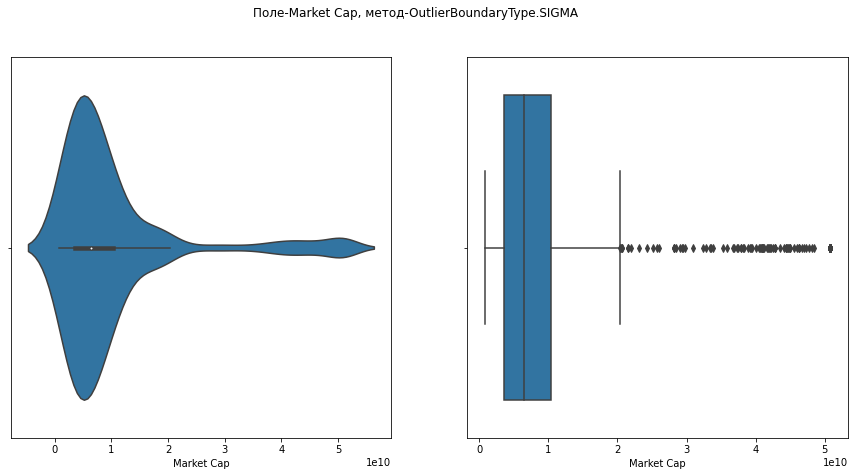
x\_col\_list = ['Market Cap']  
data=X\_train  
**for** col **in** x\_col\_list:      
    **for** obt **in** OutlierBoundaryType:  
        lower\_boundary, upper\_boundary = get\_outlier\_boundaries(data, col, obt)  
        *# Флаги для удаления выбросов*  
        outliers\_temp = np.where(data[col] > upper\_boundary, True,   
                                 np.where(data[col] < lower\_boundary, True, False))  
        *# Удаление данных на основе флага*  
        data\_trimmed = data.loc[~(outliers\_temp), ]    
        title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, data\_trimmed.shape[0])  
        plot\_for\_analys(data\_trimmed, col, title)



**Замена выбросов**

[18]

**for** col **in** x\_col\_list:      
    **for** obt **in** OutlierBoundaryType:  
        lower\_boundary, upper\_boundary = get\_outlier\_boundaries(data, col, obt)  
        data[col] = np.where(data[col] > upper\_boundary, upper\_boundary,   
                                 np.where(data[col] < lower\_boundary, lower\_boundary, data[col]))  
        title = 'Поле-{}, метод-{}'.format(col, obt)  
        plot\_for\_analys(data, col, title)



**Обработка по крайней мере одного нестандартного признака**

[19]

dataset.head()

[20]

dataset['Date'] = dataset.apply(**lambda** x: pd.to\_datetime(x['Date'], format='%Y-%m-%d'), axis=1)

[21]

*# День*  
dataset['day'] = dataset['Date'].dt.day  
*# Месяц*  
dataset['month'] = dataset['Date'].dt.month  
*# Год*  
dataset['year'] = dataset['Date'].dt.year  
*#Неделя года*  
dataset['week'] = dataset['Date'].dt.isocalendar().week  
*#День недели*  
dataset['dayofweek'] = dataset['Date'].dt.dayofweek  
*#Выходной день*  
dataset['day\_name'] = dataset['Date'].dt.day\_name()  
dataset['is\_holiday'] = dataset.apply(**lambda** x: 1 **if** x['Date'].dayofweek **in** [5,6] **else** 0, axis=1)

[22]

dataset.head()

[23]

dataset['now'] = datetime.datetime.today()  
dataset['diff'] = dataset['now'] - dataset['Date']  
dataset.dtypes

Unnamed: 0 int64  
Date datetime64[ns]  
Open float64  
High float64  
Low float64  
Close float64  
Volume float64  
Market Cap int64  
day int64  
month int64  
year int64  
week UInt32  
dayofweek int64  
day\_name object  
is\_holiday int64  
now datetime64[ns]  
diff timedelta64[ns]  
dtype: object

[24]

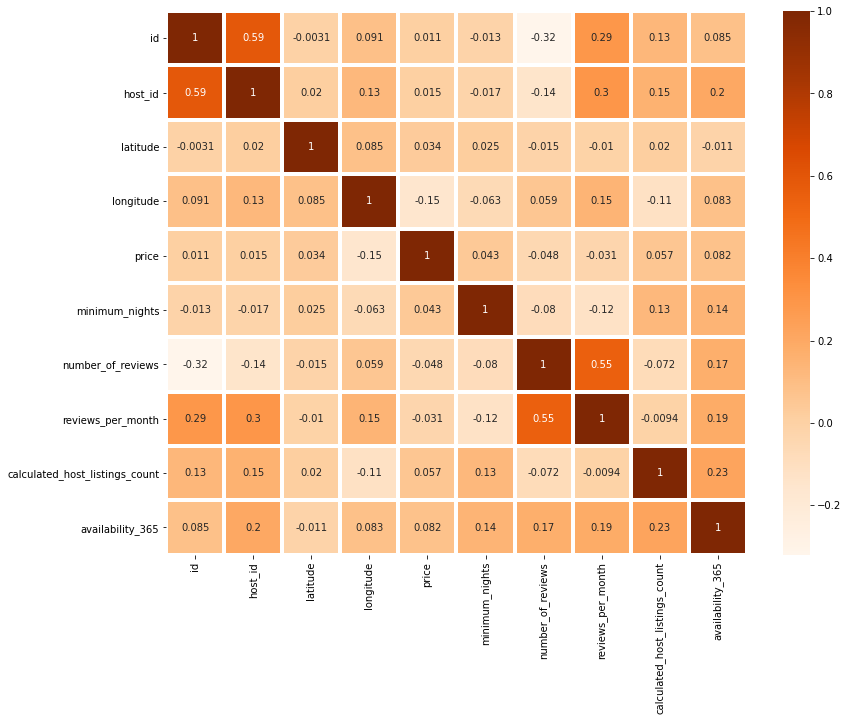
dataset.head()

**Отбор признаков из группы методов фильтрации (корреляция признаков)**

[25]

data\_dir2 = "/content/drive/MyDrive/dataset/housing/AB\_NYC\_2019.csv"  
data=pd.read\_csv(data\_dir2)  
plt.figure(figsize=(13,10))  
sns.heatmap(data.corr(), cmap="Oranges", annot=True, linewidths=3)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f0022995a50>



[30]

**def** make\_corr\_df(df):  
    cr = data.corr()  
    cr = cr.abs().unstack()   
    cr = cr.sort\_values(ascending=False)  
    cr = cr[cr >= 0.53]  
    cr = cr[cr < 1]  
    cr = pd.DataFrame(cr).reset\_index()  
    cr.columns = ['f1', 'f2', 'corr']  
    **return** cr

[31]

make\_corr\_df(data)

[32]

**def** corr\_groups(cr):  
    grouped\_feature\_list = []  
    correlated\_groups = []  
  
    **for** feature **in** cr['f1'].unique():  
        **if** feature **not** **in** grouped\_feature\_list:  
            *# находим коррелирующие признаки*  
            correlated\_block = cr[cr['f1'] == feature]  
            cur\_dups = list(correlated\_block['f2'].unique()) + [feature]  
            grouped\_feature\_list = grouped\_feature\_list + cur\_dups  
            correlated\_groups.append(cur\_dups)  
    **return** correlated\_groups

[33]

*# Группы коррелирующих признаков*  
corr\_groups(make\_corr\_df(data))

[['host\_id', 'id'], ['reviews\_per\_month', 'number\_of\_reviews']]

Видно, что две группы признаков с сильной корреляцией:  
id----host\_id  
number\_of\_reviews----reviews\_per\_month  
Поэтому удалим из датасета **host\_id** и **reviews\_per\_month**

[26]

data=data.drop(['host\_id','reviews\_per\_month'],axis=1)  
data.head()

**Отбор признаков из группы методов обертывания (алгоритм полного перебора)**

[27]

**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** mlxtend.feature\_selection **import** ExhaustiveFeatureSelector **as** EFS  
data=pd.read\_csv(data\_dir2)  
X=data[['latitude','longitude','minimum\_nights','number\_of\_reviews','calculated\_host\_listings\_count','availability\_365']]  
y=data[['price']]  
X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y)  
knn = KNeighborsClassifier(n\_neighbors=3)  
efs1 = EFS(knn,   
           min\_features=2,  
           max\_features=4,  
           scoring='accuracy',  
           print\_progress=True,  
           cv=5)  
  
efs1 = efs1.fit(X\_train, y\_train, custom\_feature\_names=X.columns)  
  
print('Best accuracy score: %.2f' % efs1.best\_score\_)  
print('Best subset (indices):', efs1.best\_idx\_)  
print('Best subset (corresponding names):', efs1.best\_feature\_names\_)

/usr/local/lib/python3.7/dist-packages/sklearn/externals/joblib/\_\_init\_\_.py:15: FutureWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functionality directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.  
 warnings.warn(msg, category=FutureWarning)  
Features: 50/50

Best accuracy score: 0.06  
Best subset (indices): (0, 1, 2, 4)  
Best subset (corresponding names): ('latitude', 'longitude', 'minimum\_nights', 'calculated\_host\_listings\_count')

**Отбор признаков из группы методов вложения (линейная регрессия)**

[35]

**from** sklearn.linear\_model **import** Lasso  
*# Используем L1-регуляризацию*  
e\_ls1 = Lasso(random\_state=1)  
e\_ls1.fit(X\_train, y\_train)  
*# Коэффициенты регрессии*  
list(zip(X\_train.columns, e\_ls1.coef\_))

[('latitude', 0.0),  
 ('longitude', -328.42130692400286),  
 ('minimum\_nights', 0.17313726132768772),  
 ('number\_of\_reviews', -0.3095591266008044),  
 ('calculated\_host\_listings\_count', 0.16891958191548065),  
 ('availability\_365', 0.174944666565565)]

[36]

**from** sklearn.feature\_selection **import** SelectFromModel  
sel\_e\_ls1 = SelectFromModel(e\_ls1)  
sel\_e\_ls1.fit(X\_train, y\_train)  
list(zip(X\_train.columns, sel\_e\_ls1.get\_support()))

[('latitude', False),  
 ('longitude', True),  
 ('minimum\_nights', True),  
 ('number\_of\_reviews', True),  
 ('calculated\_host\_listings\_count', True),  
 ('availability\_365', True)]