Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Отчет по лабораторной работе № 3
«Обработка признаков (часть 2)»
по курсу “Методы машинного обучения”

Исполнитель:
Студент группы ИУ5-22М
Желанкина А.С.
01.04.2021

Москва, 2021

Задание лабораторной работы

Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.)

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).

Описание датасета

Рассмотрим статистику стартапов, которые были созданы в промежутке между 2011 и 2012 годами. Выбор этого периода объясняется тем, что за это время часть исследуемых стартапов с большой вероятностью достигла поставленных целей. В то время как стартапы основанные после 2013 года рассматривать рано, так как многие из них еще не успели достичь правильно интерпретируемых результатов. Для построения модели была использована база стартапов Crunchbase. Из неё был сформирован датасет, состоящий из 3987 строк и 19 столбцов.

Экранные формы с текстом программы и примерами её выполнения

Масштабирование данных:

```
#Масштабирование данных на основе Z-оценки
st_sc = StandardScaler()

scaled_train = st_sc.fit_transform(x_train)
scaled_test = st_sc.transform(X_test)
scaled_train = arr_to_df(scaled_train)
scaled_train.describe()
```

	Organization Name	Headquarters Location	Founded Date	Industry Groups	Number of Employees	Funding Status	Last Funding Date	Last Funding Amount	Number of Funding Rounds	Number of Investors
count	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03
mean	6.349536e-17	-3.411108e-16	9.039477e-17	1.068012e-16	3.205630e-17	1.404703e-16	-4.155372e-17	-8.394369e-17	-1.955235e-17	-1.062835e-16
std	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00	1.000179e+00
min	-1.838740e+00	-2.014413e+00	-1.545573e+00	-1.924899e+00	-1.783137e+00	-8.415080e-01	-2.301232e+00	-4.678578e-01	-1.451339e+00	-1.119192e+00
25%	-8.330873e-01	-8.737802e-01	-8.327319e-01	-8.312533e-01	-5.223240e-01	-8.415080e-01	-7.515811e-01	-3.679986e-01	-5.944736e-01	-6.876580e-01
50%	3.182328e-02	1.892180e-01	-2.937096e-02	4.276696e-02	-1.020529e-01	-8.415080e-01	3.417064e-01	-1.564573e-01	-3.786302e-03	-7.531164e-03
75%	8.727312e-01	8.879303e-01	9.776589e-01	9.952250e-01	3.182182e-01	1.115770e+00	6.737271e-01	6.700167e-02	2.623923e-01	1.754097e-01
max	1.691819e+00	1.634418e+00	1.486831e+00	1.490503e+00	1.579031e+00	2.420623e+00	1.524282e+00	2.891284e+01	9.687917e+00	1.427218e+01

```
#Масштабирование "Mean Normalisation"
class MeanNormalisation:

    def fit(self, param_df):
        self.means = x_train.mean(axis=0)
        maxs = x_train.max(axis=0)
        mins = x_train.min(axis=0)
        self.ranges = maxs - mins

    def transform(self, param_df):
        param_df_scaled = (param_df - self.means) / self.ranges
        return param_df_scaled

    def fit_transform(self, param_df):
        self.fit(param_df)
        return self.transform(param_df)

mn= MeanNormalisation()
scaled_train = mn.fit_transform(x_train)
scaled_test = mn.transform(X_test)
scaled_train.describe()
```

	Organization Name	Headquarters Location	Founded Date	Industry Groups	Number of Employees	Funding Status	Last Funding Date	Last Funding Amount	Number of Funding Rounds	Number of Investors
count	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03	2.788000e+03
mean	2.420897e-17	-7.884654e-18	-5.451561e-17	4.035908e-17	-2.326371e-16	1.479368e-17	-2.517713e-17	1.598680e-16	7.069683e-16	-5.158774e-16
std	2.832921e-01	2.741095e-01	3.298305e-01	2.928438e-01	2.974804e-01	3.066031e-01	2.614497e-01	3.404206e-02	8.978870e-02	6.498312e-02
min	-5.208071e-01	-5.520708e-01	-5.096857e-01	-5.635937e-01	-5.303533e-01	-2.579627e-01	-6.015484e-01	-1.592399e-02	-1.302905e-01	-7.271553e-02
25%	-2.359647e-01	-2.394685e-01	-2.746111e-01	-2.433837e-01	-1.553533e-01	-2.579627e-01	-1.964654e-01	-1.252518e-02	-5.336744e-02	-4.467814e-02
50%	9.013665e-03	5.185715e-02	-9.685700e-03	1.252179e-02	-3.035330e-02	-2.579627e-01	8.932299e-02	-5.325174e-03	-3.399062e-04	-4.893108e-04
75%	2.471935e-01	2.433465e-01	3.224039e-01	2.913932e-01	9.464670e-02	3.420373e-01	1.761141e-01	2.280466e-03	2.355564e-02	1.139662e-02
max	4.791929e-01	4.479292e-01	4.903143e-01	4.364063e-01	4.696467e-01	7.420373e-01	3.984516e-01	9.840760e-01	8.697095e-01	9.272845e-01

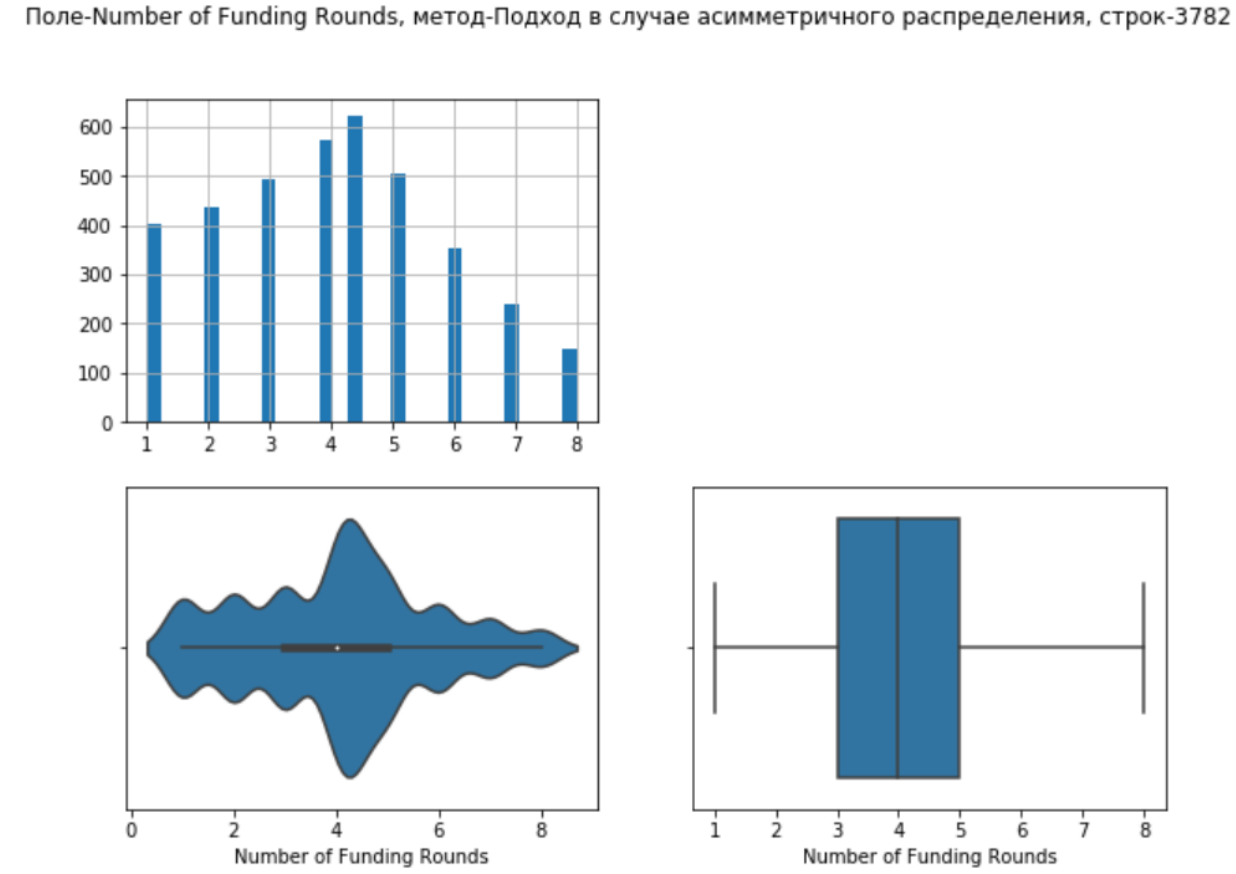
```
#MinMax-масштабирование
min_max_sc = MinMaxScaler()

x_train = min_max_sc.fit_transform(x_train)
X_test = min_max_sc.transform(X_test)
scaled_train = arr_to_df(x_train)
scaled_train.describe()
```

	Organization Name	Headquarters Location	Founded Date	Industry Groups	Number of Employees	Funding Status	Last Funding Date	Last Funding Amount	Number of Funding Rounds	Number of Investors
count	2788.000000	2788.000000	2788.000000	2788.000000	2788.000000	2788.000000	2788.000000	2788.000000	2788.000000	2788.000000
mean	0.520807	0.552071	0.509686	0.563594	0.530353	0.257963	0.601548	0.015924	0.130291	0.072716
std	0.283292	0.274110	0.329830	0.292844	0.297480	0.306603	0.261450	0.034042	0.089789	0.064983
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.284842	0.312602	0.235075	0.320210	0.375000	0.000000	0.405083	0.003399	0.076923	0.028037
50%	0.529821	0.603928	0.500000	0.576115	0.500000	0.000000	0.690871	0.010599	0.129951	0.072226
75%	0.768001	0.795417	0.832090	0.854987	0.625000	0.600000	0.777663	0.018204	0.153846	0.084112
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Обработка выбросов для числовых признаков:

Удаление

```
#Подход в случае асимметричного распределения
K2 = 1.5
IQR = data['Number of Funding Rounds'].quantile(0.75) - data['Number of Funding Rounds'].quantile(0.25)
lower_boundary = data['Number of Funding Rounds'].quantile(0.25) - (K2 * IQR)
upper_boundary = data['Number of Funding Rounds'].quantile(0.75) + (K2 * IQR)
# Флаги для удаления выбросов
outliers_temp = np.where(data['Number of Funding Rounds'] > upper_boundary, True,
                np.where(data['Number of Funding Rounds'] < lower_boundary, True, False))
# Удаление данных на основе флага
data_trimmed = data.loc[~(outliers_temp), ]
title = 'Поле-{}, метод-{}, строк-{}'.format('Number of Funding Rounds', 'Подход в случае асимметричного распределения', data_trimmed.shape[0])
diagnostic_plots(data_trimmed, 'Number of Funding Rounds', title)
```



Замена

```
#Подход в случае асимметричного распределения
K2 = 1.5
IQR = data['Number of Investors'].quantile(0.75) - data['Number of Investors'].quantile(0.25)
lower_boundary = data['Number of Investors'].quantile(0.25) - (K2 * IQR)
```

```
upper_boundary = data['Number of Investors'].quantile(0.75) + (K2 * IQR)
# Флаги для удаления выбросов
data['Number of Investors'] = np.where(data['Number of Investors'] > upper_boundary, True,
        np.where(data['Number of Investors'] < lower_boundary, True, False))
```

Обработка нестандартного признака:

```
1 data['Industries']
```

```
0                    Biotechnology, Health Care, Medical
1                    Biotechnology, Health Care, Medical
2                    Biotechnology, Health Care, Medical
3                    Biotechnology, Health Care, Medical
4                    Biotechnology, Health Care, Medical
                              ...
3994                                        Venture Capital
3995    Biotechnology, Health Care, Life Science, Phar...
3996                  Non Profit, STEM Education, Women's
3997    Biopharma, Biotechnology, Health Care, Pharmac...
3998    Advanced Materials, Health Diagnostics, Pharma...
Name: Industries, Length: 3999, dtype: object
```

```
def industries_cut(col):
    new_text = []
    for text in col:
        sep = ','
        text = text.split(sep, 1)[0]
        new_text.append(text)
    #print(val,' and ', Acquisition_Status[i], ' = ', trgt[i])

    new_text = pd.Series(new_text)
    return new_text
data['first_industry'] = industries_cut(data['Industries'])
data = data.drop('Industries', 1)
data['first_industry']
```

```
0             Biotechnology
1             Biotechnology
2             Biotechnology
3             Biotechnology
4             Biotechnology
                 ...
3994       Venture Capital
3995          Biotechnology
3996             Non Profit
3997              Biopharma
3998    Advanced Materials
Name: first_industry, Length: 3999, dtype: object
```

Отбор признаков методом из группы методов фильтрации (корреляция признаков):

```
# Формирование DataFrame с сильными корреляциями
def make_corr_df(df):
    cr = data.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.8]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
# Обнаружение групп коррелирующих признаков
```

```
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
# Группы коррелирующих признаков
corr_groups(make_corr_df(data))
```

```
[['Total Equity Funding Amount',
  'Total Funding Amount',
  'Total Funding Amount Currency (in USD)',
  'Last Equity Funding Amount Currency (in USD)',
  'Last Equity Funding Amount',
  'Last Funding Amount',
  'Last Funding Amount Currency (in USD)',
  'Total Equity Funding Amount Currency (in USD)']]
```

Отбор признаков методом из группы методов обертывания (алгоритм полного перебора):

```
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
logistic_regression = LogisticRegression()

efs1 = EFS(logistic_regression,
           min_features=2,
           max_features=10,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs1 = efs1.fit(x_train, y_train, custom_feature_names=data.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_)
```

```
Features: 30547/30811
```

```
Best accuracy score: 0.78
Best subset (indices): (2, 3, 4, 5, 6, 14)
Best subset (corresponding names): ('Founded Date', 'Industry Groups', 'Number of Employees', 'Funding Status', 'Last Funding D
ate', 'first_industry')
```

Отбор признаков методом из группы методов вложений (логистическая регрессия):

```
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1500, solver='liblinear', penalty='l1', max_iter=2500, random_state=1)
e_lr1.fit(x_train, y_train)
# Коэффициенты регрессии
e_lr1.coef_
```

```
 array([[ 0.56976329,  0.20003468, -0.57778977, -0.31884258,  0.83466091,
          1.08868251, -0.23318191, -2.7135252 , -1.32793214,  2.2416377 ,
          3.64060283,  0.9481649 , 18.24111279, -0.12290584,  0.48821289]])
```

```
# Все признаки являются "хорошими"
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(x_train, y_train)
list(zip(data.columns, sel_e_lr1.get_support()))
```

```
[('Organization Name', True),
 ('Headquarters Location', True),
 ('Founded Date', True),
 ('Industry Groups', True),
 ('Number of Employees', True),
 ('Funding Status', True),
 ('Last Funding Date', True),
 ('Last Funding Amount', True),
 ('Number of Funding Rounds', True),
 ('Number of Investors', True),
 ('SimilarWeb - Monthly Visits', True),
 ('IPqwery - Patents Granted', True),
 ('IPqwery - Trademarks Registered', True),
 ('Website', True),
 ('first_industry', True)]
```