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 домашнему заданию по курсу “Методы машинного обучения” вариант №1

Исполнитель:
Студент группы ИУ5-22М
Желанкина А.С.
20.05.2021

Москва, 2021

Постановка задачи машинного обучения

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. Для построения модели была использована база стартапов Crunchbase. Из неё был сформирован датасет, состоящий из 3987 строк и 19 столбцов.

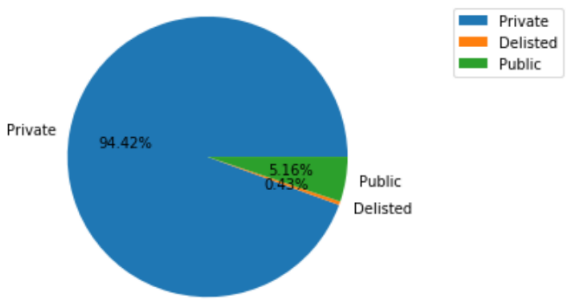
Необходимо провести исследование для определения факторов, влияющих на успешность стартапов на рынке, и разработать модель, которая могла бы предсказать возможные успешные стартапы по имеющемуся набору характеристик.

Решение задачи

Для начала было решено просмотреть состав двух переменных, из которых собирается целевая. Первой была рассмотрена переменная 'IPO Status'.

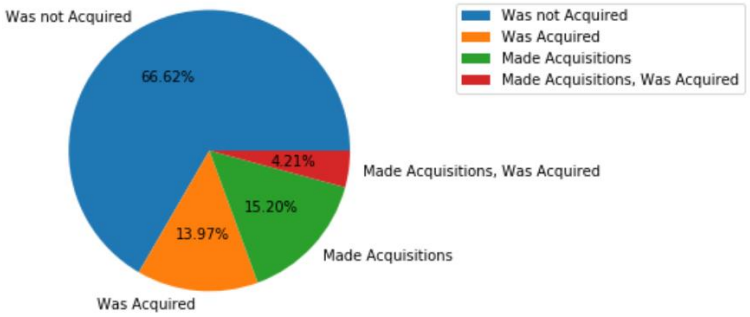
IPO может принимать такие значения, как «Private», «Delisted» и «Public». В случае успешного IPO повышается рыночная стоимость компании. Рассмотрим подробнее принимаемые переменной значения. «Public» статус означает, что стартапу удалось успешно разместить акции на бирже, «Private» – компания ещё не успела провести IPO и до этого момента считается частной. «Delisted» – компания перестала выставляться как публичная, возможно ей не удалось выйти на IPO или же она предпочла вид частного капитала. Обзор распределения в представленных данных показал, что большую часть рынка (94,42%) занимают компании с «Private» статусом, то есть еще не разместившие свои акции на бирже (рис. 1). Публичных компаний значительно меньше – 5,16%, в то время как стартапов со статусом «Delisted» всего 0,43%. Успешными стартапами в данном случае будут считаться компании, вышедшие на IPO («Public»).

In [20]:
```
fig1, ax1 = plt.subplots()

wedges, texts, autotexts = ax1.pie([3771, 17, 206], labels=labels, autopct='%1.2f%%')
ax1.axis('equal')
ax1.legend(loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



Вторая переменная, Acquisition Status, обозначает статус приобретения стартапа и так же имеет четыре значения: не была продана (“Was not Acquired”), была продана (“Was Acquired”), приобрела другую компанию (“Made Acquisitions”), приобрела другую компанию и была куплена (“Made Acquisitions, Was Acquired”). Большую часть рынка (66,62%) занимают стартапы, которые еще не были приобретены (рис. 2). Приобретенные компании составляют 13,97% от общего числа. Стартапы, совершившие покупку других компаний составляют 15,20%, а стартапы с обеими операциями насчитывают всего лишь 4,21%. Статус «Was Acquired» используется в случае, если компания была продана, что является одним из параметров оценки успешности стартапа. Также можно считать успешной компанию, чей статус равен “Made Acquisitions, Was Acquired”, так как это означает, что компания была продана и при этом успела приобрести стартап. Статус «Made Acquisitions» как правило связан с покупкой другого стартапа.

In [22]:
```
fig1, ax1 = plt.subplots()

wedges, texts, autotexts = ax1.pie([2661, 558, 607, 168], labels=labels, autopct='%1.2f%%')
ax1.axis('equal')
ax1.legend(loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```

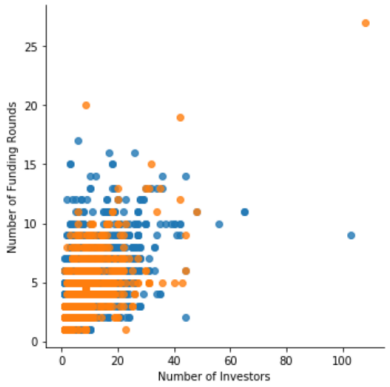


Переменные IPO Status и Acquisition Status будут рассмотрены в паре. Поэтому из них будет создана целевая переменная ‘target’. Посмотрим, есть ли явная корреляция целевой переменной с какой-либо другой из набора. Можно заметить, что целевая переменная ни с одной другой не имеет сильной связи. Однако сильно взаимосвязаны оказались число раундов инвестиций и количество инвесторов, а также зарегистрированные торговые марки и патенты, которыми владеет компания.

```
In [31]:  1  sns.heatmap(corr)
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1ca394ef7b8>
```

Рассмотрим найденные корреляции поподробнее. Большинство успешных стартапов имеет число раундов инвестиций не более 10 и количество инвесторов до 20.

```
In [36]:  1  #sns.regplot(x=data['Number of Investors'], y=data['Number of Funding Rounds'])
          2  sns.lmplot(x='Number of Investors', y='Number of Funding Rounds', data=data, fit_reg=False, hue='target', legend=False)
Out[36]: <seaborn.axisgrid.FacetGrid at 0x14c7ec88160>
```



Целевая переменная имеет только два значения: 0 – неуспешный стартап, 1 – успешный стартап. С помощью следующего графика можно проиллюстрировать, что успешных стартапов в несколько раз меньше.

```
In [29]:  1  sns.distplot(data['target'], hist=True, kde=False, rug=False)
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x14c7f4bc7f0>
```

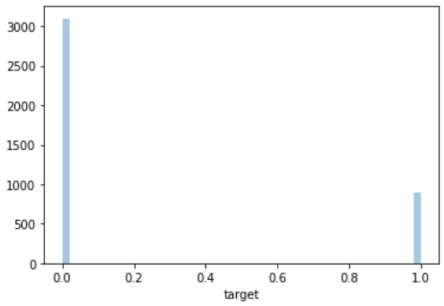
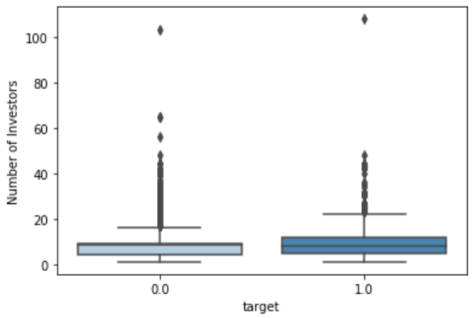


Диаграмма ящик с усами в удобной форме показывает медиану, нижний и верхний квартили, минимальное и максимальное значение выборки и выбросы. Рассмотрим ящики для целевой переменной по числу раундов инвестиций и количеству инвесторов. Медиана успешных стартапов по числу инвесторов находится в районе 10, а количеству инвесторов – 4. В обоих случаях имеются выбросы вверх, что требует дальнейшего изучения.

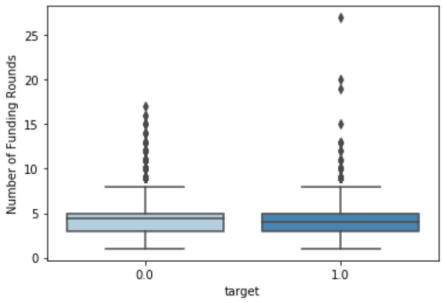
In [42]:
```
#sns.boxplot( x=data['target'], y=label_enc.inverse_transform(data['Funding Status']), palette="Blues")
sns.boxplot(x=data['target'], y=data['Number of Investors'], palette="Blues")
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x14c01d86550>



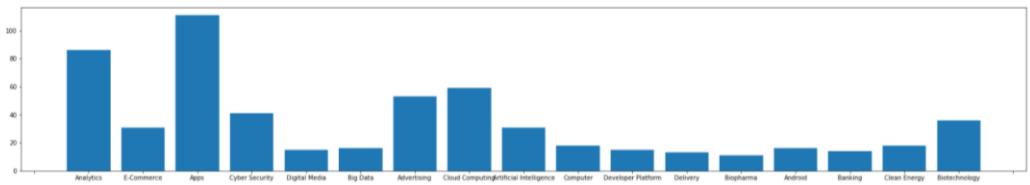
In [43]:
```
#sns.boxplot( x=data['target'], y=label_enc.inverse_transform(data['Funding Status']), palette="Blues")
sns.boxplot(x=data['target'], y=data['Number of Funding Rounds'], palette="Blues")
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x14c01dfc320>



Отношение успешных компаний по целевой переменной к индустриям и группам. На график выведем десятку лидеров. Лидирующую позицию с большим отрывом от остальных занимают стартапы, которые занимаются разработкой приложений.

In [37]:
```
x = most_fr.keys()
y = most_fr.values()
fig, ax = plt.subplots()
ax.bar(x, y, linewidth = 3)

#  Устанавливаем интервал основных делений:
ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
#  Устанавливаем интервал вспомогательных делений:
ax.xaxis.set_minor_locator(ticker.MultipleLocator(1))

fig.set_figwidth(30)
fig.set_figheight(5)

plt.show()
```



Такая информация выводится о датасете.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 41 columns):
 #    Column                                        Non-Null Count  Dtype
```

```
---  ------                                          --------------  -----
 0   Organization Name                               3999 non-null   object
 1   Organization Name URL                           3999 non-null   object
 2   Headquarters Location                           3994 non-null   object
 3   Total Equity Funding Amount                     281 non-null    float6
4
 4   Total Equity Funding Amount Currency            281 non-null    object
 5   Total Equity Funding Amount Currency (in USD)   281 non-null    float6
4
 6   Exit Date                                       922 non-null    object
 7   Exit Date Precision                             922 non-null    object
 8   Founded Date                                    3994 non-null   object
 9   Founded Date Precision                          3994 non-null   object
 10  Investor Type                                   25 non-null     object
 11  Industry Groups                                 3964 non-null   object
 12  Number of Employees                             3881 non-null   object
 13  Last Equity Funding Amount                      2931 non-null   float6
4
 14  Last Equity Funding Amount Currency             2931 non-null   object
 15  Last Equity Funding Amount Currency (in USD)    2931 non-null   float6
4
 16  Funding Status                                  2675 non-null   object
 17  Total Funding Amount                            3297 non-null   float6
4
 18  Total Funding Amount Currency                   3297 non-null   object
 19  Total Funding Amount Currency (in USD)          3297 non-null   float6
4
 20  Last Funding Date                               3369 non-null   object
 21  Last Funding Amount                             2960 non-null   float6
4
 22  Last Funding Amount Currency                    2960 non-null   object
 23  Last Funding Amount Currency (in USD)           2960 non-null   float6
4
 24  Number of Funding Rounds                        3369 non-null   float6
4
 25  Number of Investors                             3125 non-null   float6
4
 26  Acquisition Status                              1333 non-null   object
 27  Acquired by                                     726 non-null    object
 28  Acquired by URL                                 726 non-null    object
 29  Announced Date                                  726 non-null    object
 30  Announced Date Precision                        726 non-null    object
 31  IPO Date                                        223 non-null    object
 32  IPO Status                                      3994 non-null   object
 33  Delisted Date                                   16 non-null     object
 34  Delisted Date Precision                         17 non-null     object
 35  SimilarWeb - Monthly Visits                     3354 non-null   object
 36  IPqwery - Patents Granted                       2573 non-null   object
 37  IPqwery - Trademarks Registered                 2573 non-null   object
 38  Website                                         2996 non-null   object
 39  Description                                     2000 non-null   object
 40  Industries                                      2992 non-null   object
dtypes: float64(10), object(31)
```

Так как присутствует большое количество колонок, в которых достаточно сложно заполнить пропуски (больше 50%), или коррелирующих между собой колонок, то такие данные было решено удалить.

In [6]:
```
data = data.drop('Organization Name URL', 1)
data = data.drop('Total Equity Funding Amount', 1)
data = data.drop('Total Equity Funding Amount Currency', 1)
data = data.drop('Total Equity Funding Amount Currency (in USD)', 1)
data = data.drop('Exit Date', 1)
data = data.drop('Exit Date Precision', 1)
data = data.drop('Founded Date Precision', 1)
data = data.drop('Investor Type', 1)
data = data.drop('Last Equity Funding Amount Currency', 1)
data = data.drop('Last Equity Funding Amount Currency (in USD)', 1)
data = data.drop('Last Funding Amount Currency', 1)
data = data.drop('Last Funding Amount Currency (in USD)', 1)
data = data.drop('Acquired by', 1)
data = data.drop('Acquired by URL', 1)
data = data.drop('Announced Date', 1)
data = data.drop('Announced Date Precision', 1)
data = data.drop('IPO Date', 1)
data = data.drop('Delisted Date', 1)
data = data.drop('Delisted Date Precision', 1)
data = data.drop('Description', 1)
data = data.drop('Total Funding Amount Currency', 1)
data = data.drop('Total Funding Amount Currency (in USD)', 1)
data = data.drop('Last Equity Funding Amount', 1)
data = data.drop('Total Funding Amount', 1)
```

In [7]:
```
data = data.dropna(subset=['IPO Status'])
```

Новый датасет имеет такие данные:

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3994 entries, 0 to 3998
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Organization Name               3994 non-null   object 
 1   Headquarters Location           3994 non-null   object 
 2   Founded Date                    3994 non-null   object 
 3   Industry Groups                 3964 non-null   object 
 4   Number of Employees             3881 non-null   object 
 5   Funding Status                  2675 non-null   object 
 6   Last Funding Date               3369 non-null   object 
 7   Last Funding Amount             2960 non-null   float64
 8   Number of Funding Rounds        3369 non-null   float64
 9   Number of Investors             3125 non-null   float64
 10  Acquisition Status              1333 non-null   object 
 11  IPO Status                      3994 non-null   object 
 12  SimilarWeb - Monthly Visits     3354 non-null   object 
 13  IPqwery - Patents Granted       2573 non-null   object 
 14  IPqwery - Trademarks Registered 2573 non-null   object 
 15  Website                         2991 non-null   object 
 16  Industries                      2992 non-null   object 
dtypes: float64(3), object(14)
memory usage: 561.7+ KB
```

Пропуски в числовых значениях заменяли на значение среднего в данном столбце.

In [9]:
```
def repl(col):
    new_col = []
    for n in col:
        if type(n) == str:
            #print('{} = {}, {}'.format(n, n.replace(',', ''), i))
            n = n.replace(',', '')
            #print(n)
            n = float(n)
        new_col.append(n)
    new_col = pd.Series(new_col)
    return new_col
```

In [10]:
```
array = ['Last Funding Amount',
         'Number of Funding Rounds',
         'Number of Investors',
         'SimilarWeb - Monthly Visits',
         'IPqwery - Patents Granted',
         'IPqwery - Trademarks Registered']
for i in array:
    data[i] = repl(data[i])
    data[i] = data[i].fillna(data[i].mean())
```

Заполнение категориальных пропусков зависит от столбца, в котором есть пропуски. Заполнялось либо наиболее вероятным значением, либо наиболее часто встречающимся, либо ничего не значащим.

In [11]:
```
data['Acquisition Status'] = data['Acquisition Status'].fillna('Was not Acquired')
```

In [12]:
```
simp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data[['Industry Groups']] = simp.fit_transform(data[['Industry Groups']])
data[['Number of Employees']] = simp.fit_transform(data[['Number of Employees']])
data[['Industries']] = simp.fit_transform(data[['Industries']])
```

In [13]:
```
simp2 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='!!!')
data[['Funding Status']] = simp.fit_transform(data[['Funding Status']])
data[['Last Funding Date']] = simp.fit_transform(data[['Last Funding Date']])
data[['Website']] = simp.fit_transform(data[['Website']])
```

Кодирование категориальных значений с помощью LabelEncoder.

In [15]:
```
label_enc = LabelEncoder()
obj_columns = ['Organization Name', 'Headquarters Location', 'Founded Date',
               'Industry Groups', 'Number of Employees', 'Funding Status',
               'Last Funding Date', 'Website', 'Industries']

for obj in obj_columns:
    data[obj] = label_enc.fit_transform(data[obj])
```

Нормализация обучающей выборки из датасета производилась с помощью MinMaxScaler.

In [37]:
```
min_max_sc = MinMaxScaler()

x_train = min_max_sc.fit_transform(x_train)
X_test = min_max_sc.transform(X_test)
```

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

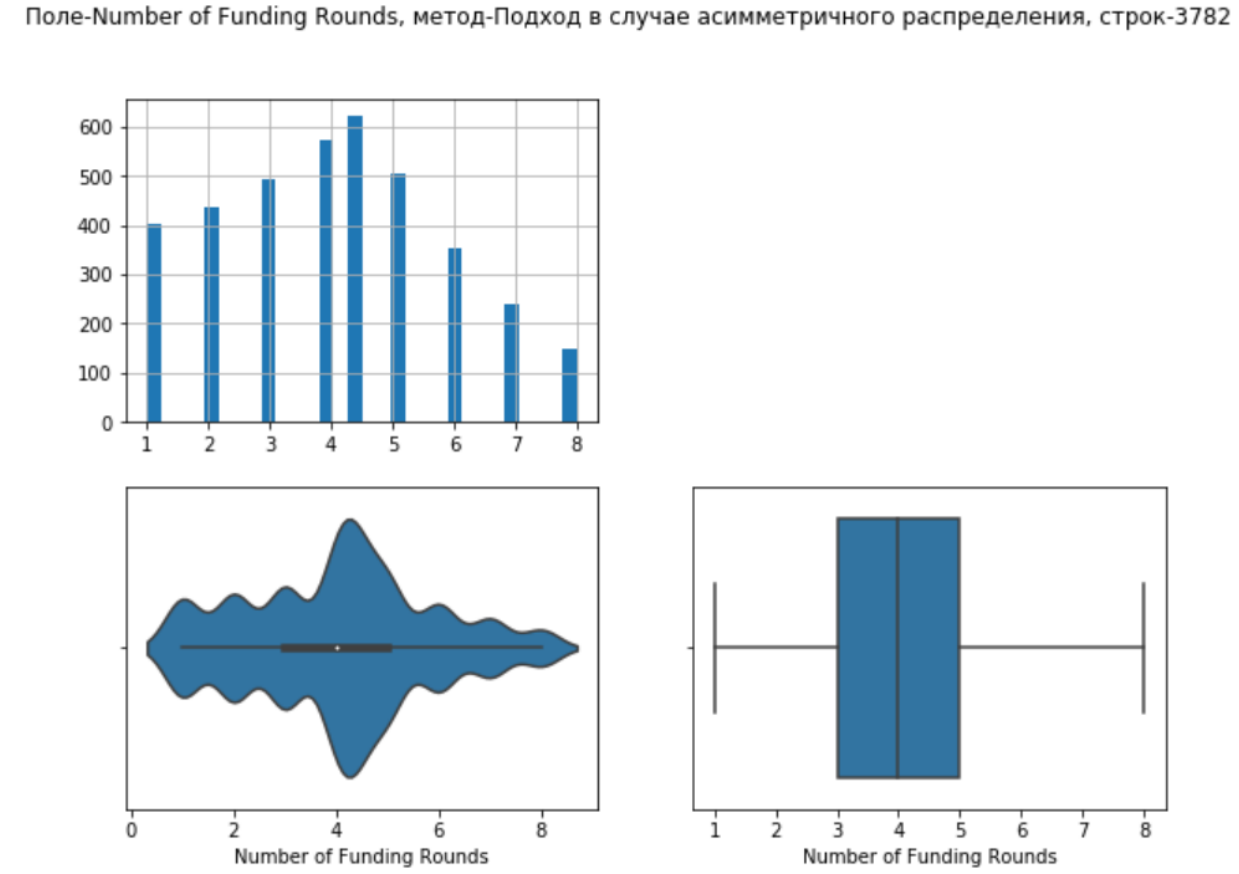
```
#MinMax-масштабирование
min_max_sc = MinMaxScaler()

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)]
```

Затем к обработанным данным были применены различные методы машинного обучения.

```
In [40]:
1 target_logistic_regression = logistic_regression.predict(X_test)
2 target_random_forest = random_forest.predict(X_test)
3 target_naive_bayes = naive_bayes.predict(X_test)
4 target_gradient_boosting = gradient_boosting.predict(X_test)
```

```
In [38]:
1 print_accuracy(target_logistic_regression, Y_test)

accuracy = 0.7642140468227425, balanced accuracy = 0.5012260036935301,
precision = 0.5, F1-score = 0.007042253521126761
```

```
In [39]:
1 print_accuracy(target_random_forest, Y_test)

accuracy = 0.7918060200668896, balanced accuracy = 0.5707706752331735,
precision = 0.8113207547169812, F1-score = 0.25671641791044775
```

```
In [40]:
1 print_accuracy(target_naive_bayes, Y_test)

accuracy = 0.7583612040133779, balanced accuracy = 0.5047527041916912,
precision = 0.3333333333333333, F1-score = 0.04620462046204621
```

```
In [41]:
1 print_accuracy(target_gradient_boosting, Y_test)

accuracy = 0.802675585284281, balanced accuracy = 0.5950463243167745,
precision = 0.8382352941176471, F1-score = 0.32571428571428573
```

Для улучшения точности их работы был использован подбор гиперпараметров с помощью GridSearchCV. Сравнение результатов до его использования и после представлено далее.

```
Сравнение
Логистическая регрессия
accuracy = 0.7642140468227425, balanced accuracy = 0.5012260036935301,
precision = 0.5, F1-score = 0.007042253521126761
accuracy = 0.7633779264214047, balanced accuracy = 0.49945295404814005,
precision = 0.0, F1-score = 0.0
Случайный лес
accuracy = 0.7918060200668896, balanced accuracy = 0.5707706752331735,
precision = 0.8113207547169812, F1-score = 0.25671641791044775
accuracy = 0.794314381270903, balanced accuracy = 0.5760898241693437,
precision = 0.8214285714285714, F1-score = 0.27218934911242604
Градиентный бустинг
accuracy = 0.802675585284281, balanced accuracy = 0.5950463243167745,
precision = 0.8382352941176471, F1-score = 0.32571428571428573
accuracy = 0.7959866220735786, balanced accuracy = 0.5735059049924732,
precision = 0.8958333333333334, F1-score = 0.2606060606060606
Наивный Байес
accuracy = 0.7583612040133779, balanced accuracy = 0.5047527041916912,
precision = 0.3333333333333333, F1-score = 0.04620462046204621
```

Выводы

Лучшие результаты показал градиентный бустинг, что говорит о возможном наличии сложных взаимосвязей в датасете.