Московский государственный технический университет им. Н.Э.Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №3 по
дисциплине
«Методы машинного обучения» на
тему
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»

Выполнил:
студент группы ИУ5-21М

Наинг Ко Ко Линн

Москва — 2020 г.

1. Цель лабораторной работы

Изучить способы предварительной обработки данных для дальнейшего формирования моделей [1].

2. Задание

Требуется [1]:

1. Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных.
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   - обработку пропусков в данных; • кодирование категориальных признаков;
   - масштабирование данных.

3. Ход выполнения работы

Подключим все необходимые библиотеки и настроим отображение графиков [2,3]:

```
In [1]:    !pip install pandasql
           from pandasql import sqldf
           import pandas as pd
```

```
In [2]: import numpy as np import
        pandas as pd import seaborn as
        sns import sklearn.impute import
        sklearn.preprocessing

        # Enable inline plots
        %matplotlib inline

        # Set plot style
        sns.set(style="ticks")

        # Set plots formats to save high resolution
        PNG from IPython.display import
        set_matplotlib_formats
        set_matplotlib_formats("retina")
```

Зададим ширину текстового представления данных, чтобы в дальнейшем текст в отчёте влезал на А4 [4]:

```
In [3]: pd.set_option("display.width", 70)
```

Для выполнения данной лабораторной работы возьмём набор данных по приложениям в Google Play Store [5]:

```
In [4]: data = pd.read_csv("fundamentals.csv")
```

Посмотрим на эти наборы данных:

```
In [5]: data.head()
```

```
[ ]  1  data.head()
```

	Unnamed: 0	Ticker Symbol	Period Ending	Accounts Payable	Accounts Receivable	Add'l income/expense items	After Tax ROE	Capital Expenditures	Capital Surplus	Cash Ratio	Cash and Cash Equivalents	Changes in Inventories
0	0	AAL	2012-12-31	3.068000e+09	-222000000.0	-1.961000e+09	23.0	-1.888000e+09	4.695000e+09	53.0	1.330000e+09	0.0
1	1	AAL	2013-12-31	4.975000e+09	-93000000.0	-2.723000e+09	67.0	-3.114000e+09	1.059200e+10	75.0	2.175000e+09	0.0
2	2	AAL	2014-12-31	4.668000e+09	-160000000.0	-1.500000e+08	143.0	-5.311000e+09	1.513500e+10	60.0	1.768000e+09	0.0
3	3	AAL	2015-12-31	5.102000e+09	352000000.0	-7.080000e+08	135.0	-6.151000e+09	1.159100e+10	51.0	1.085000e+09	0.0
4	4	AAP	2012-12-29	2.409453e+09	-89482000.0	6.000000e+05	32.0	-2.711820e+08	5.202150e+08	23.0	5.981110e+08	-260298000.0

```
In [5]: data.shape
```

```
1  data.dtypes
```

```
Unnamed: 0                          int64
Ticker Symbol                      object
Period Ending                      object
Accounts Payable                  float64
Accounts Receivable               float64
                                    ...
Total Revenue                     float64
Treasury Stock                    float64
For Year                          float64
Earnings Per Share                float64
Estimated Shares Outstanding      float64
Length: 79, dtype: object
```

3.1. Обработка пропусков в данных

Найдем все пропуски в данных:

```
In [7]: data.isnull().sum()
```

```
[ ]  1  data.isnull().sum()
```

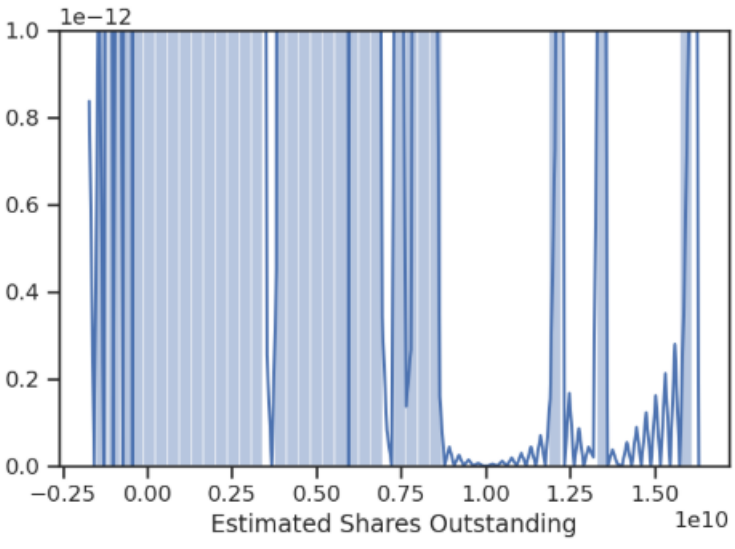
```
Unnamed: 0                          0
Ticker Symbol                       0
Period Ending                       0
Accounts Payable                    0
Accounts Receivable                 0
                                   ...
Total Revenue                       0
Treasury Stock                      0
For Year                          173
Earnings Per Share                219
Estimated Shares Outstanding      219
Length: 79, dtype: int64
```

Очевидно, что мы будем работать с колонкой `Rating`.

Самый простой вариант — заполнить пропуски нулями:

```
In [8]: sns.distplot(data["Estimated Shares Outstanding"].fillna(0));
```

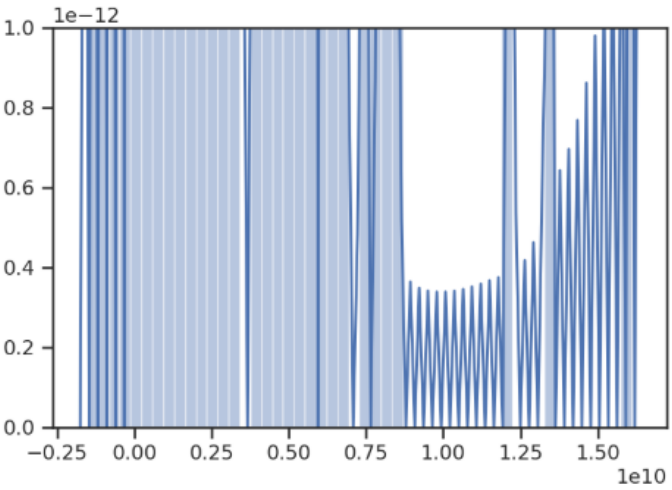
```
1  sns.distplot(data["Estimated Shares Outstanding"].fillna(0));
```



Видно, что в данной ситуации это приводит к выбросам. Логичнее было бы приложениям без рейтинга присваивать средний рейтинг:

```
In [9]: mean_imp = sklearn.impute.SimpleImputer(strategy="mean")
mean_rat = mean_imp.fit_transform(data[["Estimated Shares Outstanding"]])
sns.distplot(mean_rat);
```

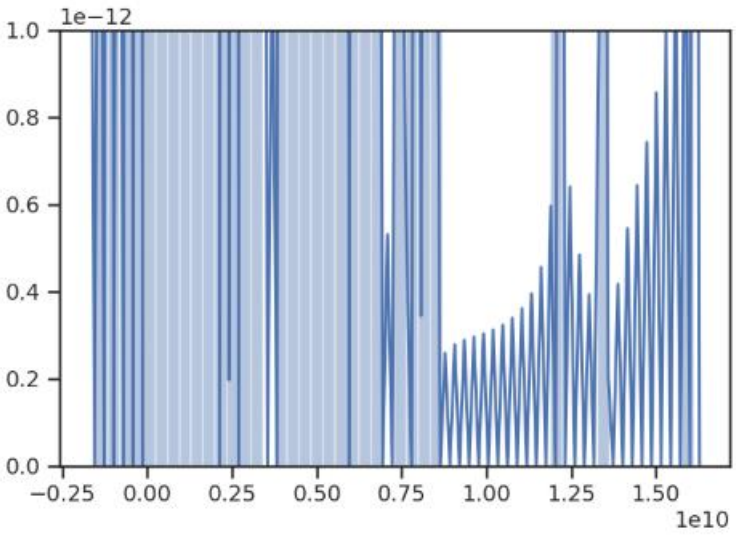
```
1  mean_imp = sklearn.impute.SimpleImputer(strategy="mean")
2  mean_rat = mean_imp.fit_transform(data[["Estimated Shares Outstanding"]])
3  sns.distplot(mean_rat);
```



Попробуем также медианный рейтинг и самый частый рейтинг:

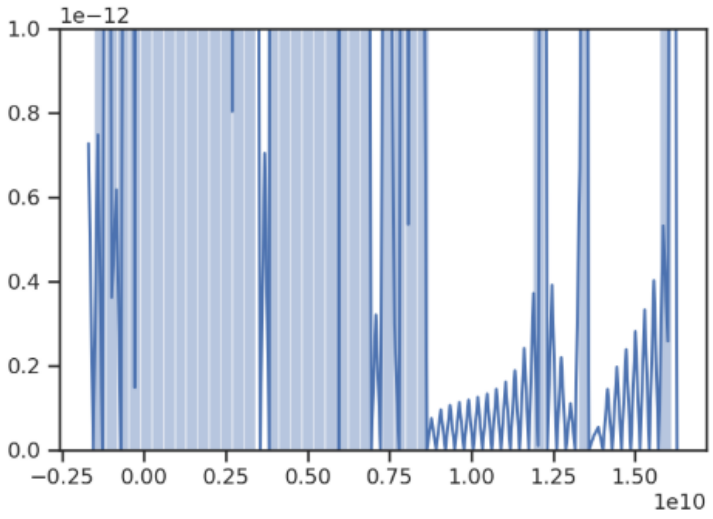
```
In [10]: med_imp = sklearn.impute.SimpleImputer(strategy="median")
med_rat = med_imp.fit_transform(data[["Estimated Shares Outstanding"]])
sns.distplot(med_rat);
```

```
[ ]  1  med_imp = sklearn.impute.SimpleImputer(strategy="median")
     2  med_rat = med_imp.fit_transform(data[["Estimated Shares Outstanding"]])
     3  sns.distplot(med_rat);
```



```
In [11]: freq_imp = sklearn.impute.SimpleImputer(strategy="most_frequent")
freq_rat = freq_imp.fit_transform(data[["Estimated Shares Outstanding"]])
sns.distplot(freq_rat);
```

```
1  freq_imp = sklearn.impute.SimpleImputer(strategy="most_frequent")
2  freq_rat = freq_imp.fit_transform(data[["Estimated Shares Outstanding"]])
3  sns.distplot(freq_rat);
```



Видно, что самый близкий к нормальному распределению график дало обычное среднее значение. Остановимся на нём:

```
In [12]: data["Estimated Shares Outstanding"] = mean_rat
```

3.2. Кодирование категориальных признаков

Рассмотрим колонку `Type`:

```
In [13]: types = data["For Year"].dropna().astype(str)
types.value_counts()
```

```
[ ]  1  data["Estimated Shares Outstanding"] = mean_rat
```

```
1  types = data["For Year"].dropna().astype(str)
2  types.value_counts()
```

```
2014.0    428
2013.0    425
2015.0    425
2012.0    244
2016.0     85
1215.0      1
Name: For Year, dtype: int64
```

Выполним кодирование категорий целочисленными значениями:

```
In [14]: le = sklearn.preprocessing.LabelEncoder()
type_le = le.fit_transform(types)
print(np.unique(type_le))
le.inverse_transform(np.unique(type_le))
```

```
1  le = sklearn.preprocessing.LabelEncoder()
2  type_le = le.fit_transform(types)
3  print(np.unique(type_le))
4  le.inverse_transform(np.unique(type_le))
```

```
[0 1 2 3 4 5]
array(['1215.0', '2012.0', '2013.0', '2014.0', '2015.0', '2016.0'],
      dtype=object)
```

```
[ ]  1  type_oh = pd.get_dummies(types)
     2  type_oh.head()
```

Выполним кодирование категорий наборами бинарных значений:

```
In [15]: type_oh = pd.get_dummies(types)
type_oh.head()
```

```
1  type_oh = pd.get_dummies(types)
2  type_oh.head()
```

	1215.0	2012.0	2013.0	2014.0	2015.0	2016.0
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	0	0	0	1	0	0
3	0	0	0	0	1	0
4	0	1	0	0	0	0

```
In [16]: type_oh[type_oh["2014.0"] == 1].head()
```

```
1  type_oh[type_oh["2014.0"] == 1].head()
```

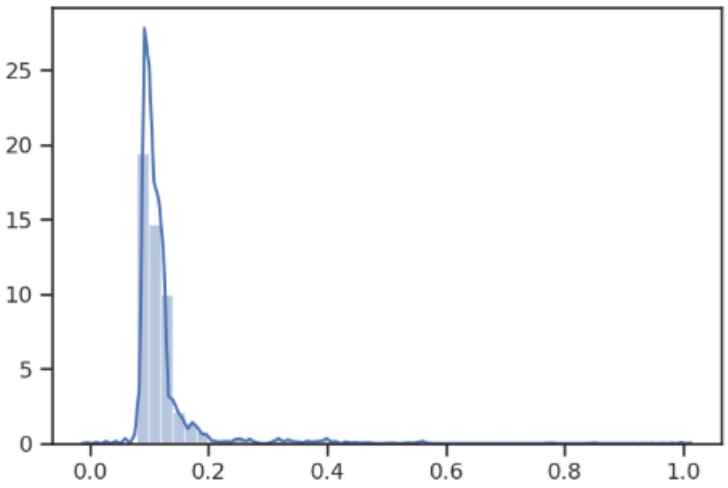
	1215.0	2012.0	2013.0	2014.0	2015.0	2016.0
2	0	0	0	1	0	0
6	0	0	0	1	0	0
9	0	0	0	1	0	0
14	0	0	0	1	0	0
17	0	0	0	1	0	0

3.3. Масштабирование данных

Для начала попробуем обычное MinMax-масштабирование:

```
In [17]: mm = sklearn.preprocessing.MinMaxScaler()
sns.distplot(mm.fit_transform(data[["Estimated Shares Outstanding"]]));
```

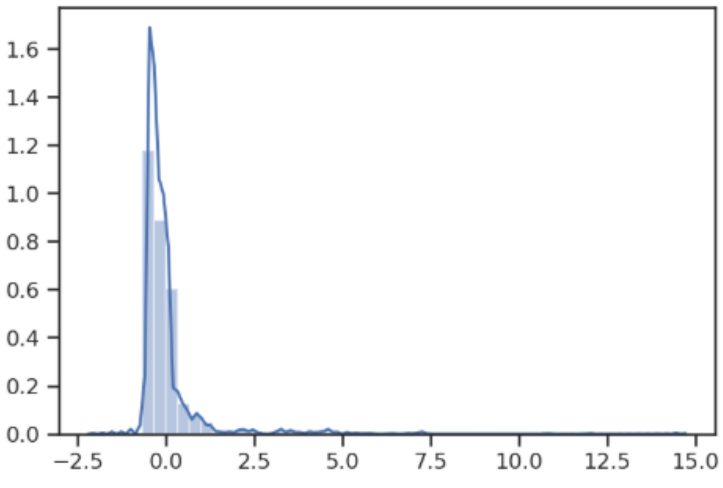
```
1  mm = sklearn.preprocessing.MinMaxScaler()
2  sns.distplot(mm.fit_transform(data[["Estimated Shares Outstanding"]]));
```



Результат вполне ожидаемый и вполне приемлемый. Но попробуем и другие варианты, например, масштабирование на основе Z-оценки:

```
In [18]: ss = sklearn.preprocessing.StandardScaler()
sns.distplot(ss.fit_transform(data[["Estimated Shares Outstanding"]]));
```

```
1  ss = sklearn.preprocessing.StandardScaler()
2  sns.distplot(ss.fit_transform(data[["Estimated Shares Outstanding"]]));
```



Также результат ожидаемый, но его применимость зависит от дальнейшего использования.

Также была опробована нормализация данных, но единственным результатом была ошибка `LinAlgError: singular matrix`. С чем она связана не до конца очевидно,

вероятно, метод `sklearn.preprocessing.Normalizer` плохо рассчитан на одноколоночные данные.

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных» [Электронный ресурс] // GitHub. – – 2019. — Режим доступа: `https://github.com/ugapanyuk/ml_course/wiki/LAB_MISSING` (дата обращения: 05.04.2019).

[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: `https://ipython.readthedocs.io/en/ stable/` (online; accessed: 20.02.2019).

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: `https://seaborn.pydata.org/` (online; accessed: 20.02.2019).

[4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: `http://pandas.pydata.org/pandas-docs/stable/` (online; accessed: 20.02.2019).

[5] Gupta L. Google Play Store Apps [Electronic resource] // Kaggle. — 2019. — Access mode: `https://www.kaggle.com/lava18/google-play-store-apps` (online; accessed:
05.04.2019).