**Министерство науки и высшего образования 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 Радиотехнический (РТ)

КАФЕДРА Системы обработки информации и управления (ИУ5)

# Рубежный контроль №1
# по курсу «Технологии машинного обучения»
# 15 вариант

Студентка РТ5-61Б (Группа) Стадник Е.Р. (Фамилия И.О.)

Преподаватель: Гапанюк Ю. Е. (Фамилия И.О.)

2021 г.

# Рубежный контроль 1

Стадник Елена, 15 вариант, задание 2, номер датасета 7

## Загрузка и первичный анализ датасета

```
[1]  import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from google.colab import files
```

```
[8]  files.upload()
```

2 мин.

Выбрать файлы Файл не выбран Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving restaurant-scores-lives-standard.csv to restaurant-scores-lives-standard.csv
{'restaurant-scores-lives-standard.csv': b'business_id,business_name,business_addres
```

Описание:

business_id - id ресторана;

business_name - название;

business_address - адрес;

business_city - город;

business_state - штат;

business_postal_code - почтовый индекс;

business_latitude - широта;

business_longitude - долгота;

business_location - расположение;

business_phone_number - номер телефона;

inspection_id - id инспекции;

inspection_date - дата инспекции;

inspection_score - оценка;

inspection_type - тип инспекции;

violation_id - id нарушения;

violation_description - описание нарушения;

risk_category - категория риска

```
[9]  data = pd.read_csv('restaurant-scores-lives-standard.csv')
```

Размер набора данных:

```
[10]  data.shape
```

```
(53973, 23)
```

Типы колонок:

```
[11]  data.dtypes
```

```
business_id                  int64
business_name               object
business_address            object
business_city               object
business_state              object
business_postal_code        object
business_latitude          float64
business_longitude         float64
business_location           object
business_phone_number      float64
inspection_id               object
inspection_date             object
inspection_score           float64
inspection_type             object
violation_id                object
violation_description       object
risk_category               object
Neighborhoods (old)        float64
Police Districts           float64
Supervisor Districts       float64
Fire Prevention Districts  float64
Zip Codes                  float64
Analysis Neighborhoods     float64
dtype: object
```

Проверим, есть ли пропущенные значения:

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

```
[12] business_id                    0
     business_name                  0
     business_address               0
     business_city                  0
     business_state                 0
     business_postal_code        1018
     business_latitude          19556
     business_longitude         19556
     business_location          19556
     business_phone_number      36938
     inspection_id                  0
     inspection_date                0
     inspection_score           13610
     inspection_type                0
     violation_id               12870
     violation_description      12870
     risk_category              12870
     Neighborhoods (old)        19594
     Police Districts           19594
     Supervisor Districts       19594
     Fire Prevention Districts  19646
     Zip Codes                  19576
     Analysis Neighborhoods     19594
     dtype: int64
```

Первые 5 строк датасета:

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

[23]

| | business_id | business_name | business_address | business_city | business_state | busi |
|---|---|---|---|---|---|---|
| **0** | 101192 | Cochinita #2 | 2 Marina Blvd Fort Mason | San Francisco | CA | |
| **1** | 97975 | BREADBELLY | 1408 Clement St | San Francisco | CA | |
| **2** | 92982 | Great Gold Restaurant | 3161 24th St. | San Francisco | CA | |
| **3** | 101389 | HOMAGE | 214 CALIFORNIA ST | San Francisco | CA | |
| **4** | 85986 | Pronto Pizza | 798 Eddy St | San Francisco | CA | |

5 rows × 23 columns

К сожалению, из первичного анализа датасета выходит, что нет какого-либо категориального или количественного признака, который можно было бы восполнить без потери для смысла. Например, есть рестораны, у которых пропущены почтовые индексы - как их восполнить, не обращая внимание на адрес без потери смысла, непонятно. Таким образом было решено добавить больше "нарушений" для ресторанов - будет добавлено самое частое id и его описание, а из числовых значений добавим оценку инспекции из числа средних.

## Обработка пропусков в данных

### Обработка пропусков в числовых данных:

Итак, пропуски в числовых значениях будет произведено с помощью средних значений.

Создадим функцию, которая позволит задавать колонку и вид импьютации:

```
[18] from sklearn.impute import SimpleImputer
     from sklearn.impute import MissingIndicator
```

```
[16] def test_num_impute_col(dataset, column, strategy_param):
         temp_data = dataset[[column]]

         indicator = MissingIndicator()
         mask_missing_values_only = indicator.fit_transform(temp_data)

         imp_num = SimpleImputer(strategy=strategy_param)
         data_num_imp = imp_num.fit_transform(temp_data)

         filled_data = data_num_imp[mask_missing_values_only]

         return column, strategy_param, filled_data.size, filled_data[0], filled_data[fi
```

Применим функцию для выбранной нами колонки, а также посмотрим, сколько и какие значения в итоге туда подставили:

```
[19] test_num_impute_col(data, 'inspection_score', 'mean')
```

0 сек.

```
('inspection_score', 'mean', 13610, 86.22679186383569, 86.22679186383569)
```

### Обработка пропусков в категориальных данных:

Импьютация наиболее частыми значениями в две колонки. Делаем так, поскольку id нарушения должен соответстсовать его описанию.

```
[21] imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
     data_imp2 = imp2.fit_transform(data[['violation_id']])
     data_imp2
```

0 сек.

```
array([['2659_20180327_103103'],
       ['97975_20190725_103124'],
       ['2659_20180327_103103'],
       ...,
       ['84541_20190506_103133'],
       ['91572_20190506_103116'],
       ['89569_20190506_103157']], dtype=object)
```

```
[22] imp3 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
     data_imp3 = imp3.fit_transform(data[['violation_description']])
     data_imp3
```

0 сек.

```
array([['Unclean or degraded floors walls or ceilings'],
       ['Inadequately cleaned or sanitized food contact surfaces'],
       ['Unclean or degraded floors walls or ceilings'],
       ...,
       ['Foods not protected from contamination'],
       ['Inadequate food safety knowledge or lack of certified food safety manager']
       ['Food safety certificate or food handler card not available']],
      dtype=object)
```

Вывод: выполнила импьютацию числового значения (оценки инспекции) в качестве среднего, а качественных значений (id и описание нарушения) в качестве наиболее частого (для совпадения id и описания оба выбраны частыми).