Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления» – ИУ5
Факультет «Радиотехнический» – РТ5

Отчёт по лабораторной работе №2 по курсу Технологии машинного обучения

6
(количество листов)

Исполнитель

студент группы РТ5-61б _____ Нижаметдинов М. Ш.

“___” _____ 2023 г.

Проверил

Преподаватель кафедры ИУ5 _____ Гапанюк Ю. Е.

“___” _____ 2023 г.

Москва, 2023 г.

Задание

1. Выбрать набор данных (датасет), содержащий категориальные 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.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   - обработку пропусков в данных;
   - кодирование категориальных признаков;
   - масштабирование данных.

Набор данных

https://www.kaggle.com/datasets/georgescutelnicu/top-100-popular-movies-from-2003-to-2022-imdb

Исходный текст проекта

```
# Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

## Загрузка и первичный анализ данных

Используем данные из соревнования Top 100 popular movies from 2003 to 2022 (iMDB)
https://www.kaggle.com/datasets/georgescutelnicu/top-100-popular-movies-from-2003-to-2022-imdb

data = pd.read_csv('https://raw.githubusercontent.com/mansurik1/MLT/master/LW%202/Project/data/movies.csv', sep=";")

# размер набора данных
data.shape

# типы колонок
data.dtypes

# проверка на пропущенные значения
data.isnull().sum()
```

```
# Первые 5 строк датасета
data.head()

total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))

## Обработка пропусков в данных

### Обработка пропусков в числовых данных

# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)

# Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)

# Заполнение всех пропущенных значений нулями
# В данном случае это некорректно, так как нулями заполняются в том числе
категориальные колонки
data_new_3 = data.fillna(0)
data_new_3.head()

# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col,
dt, temp_null_count, temp_perc))

# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num

# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
```

```
    plt.show()
```

Будем использовать встроенные средства импьютации библиотеки scikit-learn - https://scikit-learn.org/stable/modules/impute.html

Пропуски данных в колонке Rating заполним на медианное значение, в колонке Runtime - на наиболее повторяющееся

```
data_num_Rating = data_num[['Rating']]
data_num_Rating.head()

from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator

# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_Rating_only = indicator.fit_transform(data_num_Rating)
mask_missing_Rating_only

imp_num = SimpleImputer(strategy='median')
data_num_imp_Rating = imp_num.fit_transform(data_num_Rating)
data_num_imp_Rating[mask_missing_Rating_only]

np.unique(data_num_imp_Rating)

data_num_Runtime = data_num[['Runtime']]
data_num_Runtime.head()

from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator

indicator = MissingIndicator()
mask_missing_Runtime_only = indicator.fit_transform(data_num_Runtime)
mask_missing_Runtime_only

imp_num = SimpleImputer(strategy='most_frequent')
data_num_imp_Runtime = imp_num.fit_transform(data_num_Runtime)
data_num_imp_Runtime[mask_missing_Runtime_only]

np.unique(data_num_imp_Runtime)

### Обработка пропусков в категориальных данных

# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
```

```
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col,
dt, temp_null_count, temp_perc))

cat_temp_data = data[['Certificate']]
cat_temp_data.head()

cat_temp_data['Certificate'].unique()

cat_temp_data[cat_temp_data['Certificate'].isnull()].shape

# Импьютация константой
imp = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')
data_imp = imp.fit_transform(cat_temp_data)
data_imp

np.unique(data_imp)

data_imp[data_imp=='NA'].size
```

Преобразование категориальных признаков в числовые

Используем OrdinalEncoder, который ориентирован на применение к матрице объект-признак, то есть для кодирования матрицы нецелевых признаков.

```
from sklearn.preprocessing import OrdinalEncoder

data_oe = data[['Title', 'Month', 'Certificate', 'Directors', 'Stars', 'Genre', 'Filming_location',
'Country_of_origin']]
data_oe.head()

data_oe['Title'].unique()

imp2 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')
data_oe_filled = imp2.fit_transform(data_oe)
data_oe_filled

oe = OrdinalEncoder()
cat_enc_oe = oe.fit_transform(data_oe_filled)
cat_enc_oe
```

```
# Уникальные значения 1 признака
np.unique(cat_enc_oe[:, 0])

# Уникальные значения 2 признака
np.unique(cat_enc_oe[:, 1])

# Уникальные значения 3 признака
np.unique(cat_enc_oe[:, 2])

# Наименования категорий в соответствии с порядковыми номерами
oe.categories_

# Обратное преобразование
oe.inverse_transform(cat_enc_oe)

## Масштабирование данных

from sklearn.preprocessing import MinMaxScaler, StandardScaler

### MinMax масштабирование

sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Runtime']])

plt.hist(data['Runtime'], 50)
plt.show()

plt.hist(sc1_data, 50)
plt.show()

### Масштабирование данных на основе Z-оценки - StandardScaler

sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Runtime']])

plt.hist(sc2_data, 50)
plt.show()
```