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:

o обработку пропусков в данных;

o кодирование категориальных признаков;

o 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()