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

Отчёт по рубежному контролю №1 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

Для заданного набора данных проведите обработку пропусков в данных для одного категориального и одного количественного признака. Какие способы обработки пропусков в данных для категориальных и количественных признаков Вы использовали? Какие признаки Вы будете использовать для дальнейшего построения моделей машинного обучения и почему?

Набор данных

https://www.kaggle.com/lava18/google-play-store-apps

Исходный текст проекта

```
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

data =
pd.read_csv('https://raw.githubusercontent.com/mansurik1/MLT/master/MC%201/Project/data/googleplaystore.csv', sep=",")

# размер набора данных
data.shape

# типы колонок
data.dtypes

# проверим есть ли 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

## Простые стратегии - удаление или заполнение нулями

Удаление колонок, содержащих пустые значения
```

`res = data.dropna(axis=1, how='any')`

Удаление строк, содержащих пустые значения
`res = data.dropna(axis=0, how='any')`

Удаление может производиться для группы строк или колонок.

```
# 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)

data.head()

# 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()
```

"Внедрение значений" - импьютация (imputation)

Обработка пропусков в числовых данных

```
# 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()

data_num_MasVnrArea = data_num[['Rating']]
data_num_MasVnrArea.head()

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

# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_MasVnrArea)
mask_missing_values_only

С помощью класса [SimpleImputer](https://scikit-
learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html#sklearn.impute.SimpleI
mputer) можно проводить импьютацию различными [показателями центра
распределения](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D0%B0%D0%B
7%D0%B0%D1%82%D0%B5%D0%BB%D0%B8_%D1%86%D0%B5%D0%BD%D1%82%D
1%80%D0%B0_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5
%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)

strategies=['mean', 'median', 'most_frequent']

def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_MasVnrArea)
    return data_num_imp[mask_missing_values_only]

strategies[0], test_num_impute(strategies[0])

strategies[1], test_num_impute(strategies[1])

strategies[2], test_num_impute(strategies[2])

# Более сложная функция, которая позволяет задавать колонку и вид импьютации
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[filled_data.size-1]

data[['Rating']].describe()

test_num_impute_col(data, 'Rating', strategies[0])

test_num_impute_col(data, 'Rating', strategies[1])

test_num_impute_col(data, 'Rating', strategies[2])
```

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[['Current Ver']]
cat_temp_data.head()

cat_temp_data['Current Ver'].unique()

cat_temp_data[cat_temp_data['Current Ver'].isnull()].shape

# Импьютация наиболее частыми значениями (модой)
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2

# Пустые значения отсутствуют
np.unique(data_imp2)

# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

```
np.unique(data_imp3)

data_imp3[data_imp3=='NA'].size

## Подключение другого набора данных (для получения большего количества числовых значений)

from sklearn.datasets import *

wine = load_wine()

wine['data'].shape

data1 = pd.DataFrame(data= np.c_[wine['data'], wine['target']],
                     columns= wine['feature_names'] + ['target'])

data1
```

Чтобы определить, какие признаки я буду использовать для дальнейшего построения моделей машинного обучения и почему, нужно сделать проверку корреляции признаков, которая поможет определить, какие признаки наиболее сильное коррелируют с целевым признаком.

```
data1.corr()
```

На основе корреляционной матрицы можно сделать следующие выводы:

- Целевой признак наиболее сильно коррелирует с пропиленом (-0.63), OD280/OD315 разбавленных вин (-0.79), общим кол-во фенолов (-0,71) и флавоноидами (-0,84). Эти признаки обязательно следует оставить в модели.
- Целевой признак отчасти коррелирует с алкоголем (-0.33). Этот признак стоит также оставить в модели.
- Целевой признак слабо коррелирует с золой (-0.05) и магнием (-0.2). Скорее всего эти признаки стоит исключить из модели, возможно они только ухудшат качество модели.