Московский государственный технический университет имени Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5

Отчёт по
лабораторной работе № 2
«Технологии машинного обучения»

Подготовил:

Кан Андрей Дмитриевич

Группа ИУ5-64Б

Подпись_____

Дата_____

Москва
2021г.

Цель лабораторной работы: изучение способов предварительной обработки данных для дальнейшего формирования моделей.

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.

Текст программы:

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


data = pd.read_csv('country_vaccinations.csv', sep=",")


data.shape


data.dtypes


data.isnull().sum()


data.head()
```

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

```
# Обработка пропусков в 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' or dt=='object'):
        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))
```

Сразу удалим колонки с слишком высоким процентом пропуска данных, далее будем работать с оставшимися колонками.

```
# Удаление колонок, содержащих пустые значения
data = data.dropna(axis=1, thresh=3362)
data.shape
```

```
# 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()
```

Как видно из представленных гистограмм, лучше всего использовать мод в качестве значения при заполнении пропусков.

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

```
# 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]
```

```
test_num_impute_col(data, 'total_vaccinations', 'most_frequent')
```

```
test_num_impute_col(data, 'daily_vaccinations', 'most_frequent')
```

```
test_num_impute_col(data, 'total_vaccinations_per_hundred', 'most_frequent')
```

```
test_num_impute_col(data, 'daily_vaccinations_per_million', 'most_frequent')
```

```
# 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))

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



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



cat_temp_data['iso_code'].unique()



cat_temp_data[cat_temp_data['iso_code'].isnull()].shape
```

```
# Импьютация наиболее частыми значениями
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)



# Преобразование категориальных признаков в числовые

cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])



cat_enc['c1'].unique()



np.unique(cat_enc_le)

le.inverse_transform([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,
       13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
       26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
       39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
```

```
       52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
       65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
       78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
       91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
      104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
      117, 118, 119])
```

```
# Масштабирование

from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer


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


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


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

Как видим по графикам выше, масштабирование не изменило гистограмму.