Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Отчет**

**Лабораторная работа № 3**

**По курсу «Технологии машинного обучения»**

## «Обработка пропусков данных, кодирование категориальных признаков, масштабирование данных»

**ИСПОЛНИТЕЛЬ:**

Сергеев И.В.

Группа ИУ5-64Б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

Москва 2020

**Описание задания**

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

**Задание -** 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.). Для выбранного датасета (датасетов) на основе материалов [лекции](https://github.com/ugapanyuk/ml_course_2020/blob/master/common/notebooks/missing/handling_missing_norm.ipynb) решить следующие задачи:

* обработку пропусков в данных;
* кодирование категориальных признаков;
* масштабирование данных.

**Текст программы**

Программа разрабатывалась в IDE PyCharm. Ниже приведён полный листинг программы:

#%%  
  
import numpy  
import pandas  
import seaborn  
import matplotlib.pyplot as plt  
%matplotlib inline   
seaborn.set(style=**"ticks"**)  
data = pandas.read\_csv(**'dataset.csv'**, sep=**","**)  
data.head()  
  
#%%  
  
print(data.shape)  
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer  
  
#%%  
  
data.isnull().sum()  
  
#%%  
  
# Here select empty numeric columns  
total\_count = data.shape[0]  
num\_cols = []  
for col in data.columns:  
 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))  
  
#%%  
  
data\_num = data[num\_cols]  
data\_num  
  
#%%  
  
for col in data\_num:  
 plt.hist(data[col], 50)  
 plt.xlabel(col)  
 plt.show()  
  
#%%  
  
from sklearn.impute import SimpleImputer  
from sklearn.impute import MissingIndicator  
cat\_temp\_data = data[[**'Product\_Category\_2'**]]  
cat\_temp\_data.head()  
  
#%%  
  
cat\_temp\_data[cat\_temp\_data[**'Product\_Category\_2'**].isnull()].shape  
  
#%%  
  
  
  
#%%  
  
imp3 = SimpleImputer(missing\_values=numpy.nan, strategy=**'constant'**, fill\_value=0)  
data\_imp3 = imp3.fit\_transform(cat\_temp\_data)  
data\_imp3  
  
#%%  
  
numpy.unique(data\_imp3)  
  
#%%  
  
data\_imp3[data\_imp3==0].size  
  
#%%  
  
sc1 = MinMaxScaler()  
sc1\_data = sc1.fit\_transform(data[[**'Purchase'**]])  
  
#%%  
  
plt.hist(data[**'Purchase'**], 50)  
plt.show()  
  
#%%  
  
plt.hist(sc1\_data, 50)  
plt.show()  
  
#%%  
  
sc2 = StandardScaler()  
sc2\_data = sc2.fit\_transform(data[[**'Purchase'**]])  
  
#%%  
  
plt.hist(sc2\_data, 50)  
plt.show()  
  
#%%  
  
temp\_data = data[[**'Gender'**]]  
temp\_data.head()  
  
#%%  
  
temp\_data[**'Gender'**].unique()  
  
#%%  
  
temp\_data[temp\_data[**'Gender'**].isnull()].shape  
  
#%%  
  
from sklearn.impute import SimpleImputer  
imp2 = SimpleImputer(missing\_values=numpy.nan, strategy=**'most\_frequent'**)  
data\_imp2 = imp2.fit\_transform(temp\_data)  
data\_imp2  
  
#%%  
  
numpy.unique(data\_imp2)  
  
#%%  
  
cat\_enc = pandas.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()  
  
#%%  
  
numpy.unique(cat\_enc\_le)  
  
#%%  
  
le.inverse\_transform([0, 1])  
  
#%%  
  
ohe = OneHotEncoder()  
cat\_enc\_ohe = ohe.fit\_transform(cat\_enc[[**'c1'**]])  
  
#%%  
  
cat\_enc.shape  
  
#%%  
  
cat\_enc\_ohe.shape  
  
#%%  
  
cat\_enc\_ohe.todense()[0:10]  
  
#%%  
  
cat\_enc.head(10)  
  
#%%  
  
pandas.get\_dummies(cat\_enc).head()

**Примеры выполнения программы**

Выполнение программы, а также наглядная демонстрация входных и выходных данных (таблиц, графиков и тд) осуществлялась на базе Jupyter Notebook, сервер которого запускался из-под PyCharm. Ниже приведены скриншоты, отражающие работу программы:



















