**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**



**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

**Методические указания к лабораторным работам по курсу «Машинное обучение»**

**Лабораторная работа №3**

**«Обработка признаков(часть2)**

Выполнил Уляшин В.В. (ИУ5-23М)

Москва, 2023 г.

1. **ЗАДАНИЕ**

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

1. Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
   1. масштабирование признаков (не менее чем тремя способами);
   2. обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
   3. обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
   4. отбор признаков:
      * один метод из группы методов фильтрации (filter methods);
      * один метод из группы методов обертывания (wrapper methods);
      * один метод из группы методов вложений (embedded methods).

import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.datasets import load\_wine  
from sklearn.impute import SimpleImputer  
from sklearn.impute import MissingIndicator  
import scipy.stats as stats  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.preprocessing import RobustScaler  
from sklearn.linear\_model import LogisticRegression  
from sklearn.svm import LinearSVC

data = pd.read\_csv("Customer-Churn-Records.csv")  
data.shape

(10000, 18)

data.head()

RowNumber CustomerId Surname CreditScore Geography Gender Age \  
0 1 15634602 Hargrave 619 France Female 42   
1 2 15647311 Hill 608 Spain Female 41   
2 3 15619304 Onio 502 France Female 42   
3 4 15701354 Boni 699 France Female 39   
4 5 15737888 Mitchell 850 Spain Female 43   
  
 Tenure Balance NumOfProducts HasCrCard IsActiveMember \  
0 2 0.00 1 1 1   
1 1 83807.86 1 0 1   
2 8 159660.80 3 1 0   
3 1 0.00 2 0 0   
4 2 125510.82 1 1 1   
  
 EstimatedSalary Exited Complain Satisfaction Score Card Type \  
0 101348.88 1 1 2 DIAMOND   
1 112542.58 0 1 3 DIAMOND   
2 113931.57 1 1 3 DIAMOND   
3 93826.63 0 0 5 GOLD   
4 79084.10 0 0 5 GOLD   
  
 Point Earned   
0 464   
1 456   
2 377   
3 350   
4 425

# Удаление колонок с высоким процентом пропусков (более 25%)  
data.dropna(axis=1, thresh=7500)

RowNumber CustomerId Surname CreditScore Geography Gender Age \  
0 1 15634602 Hargrave 619 France Female 42   
1 2 15647311 Hill 608 Spain Female 41   
2 3 15619304 Onio 502 France Female 42   
3 4 15701354 Boni 699 France Female 39   
4 5 15737888 Mitchell 850 Spain Female 43   
... ... ... ... ... ... ... ...   
9995 9996 15606229 Obijiaku 771 France Male 39   
9996 9997 15569892 Johnstone 516 France Male 35   
9997 9998 15584532 Liu 709 France Female 36   
9998 9999 15682355 Sabbatini 772 Germany Male 42   
9999 10000 15628319 Walker 792 France Female 28   
  
 Tenure Balance NumOfProducts HasCrCard IsActiveMember \  
0 2 0.00 1 1 1   
1 1 83807.86 1 0 1   
2 8 159660.80 3 1 0   
3 1 0.00 2 0 0   
4 2 125510.82 1 1 1   
... ... ... ... ... ...   
9995 5 0.00 2 1 0   
9996 10 57369.61 1 1 1   
9997 7 0.00 1 0 1   
9998 3 75075.31 2 1 0   
9999 4 130142.79 1 1 0   
  
 EstimatedSalary Exited Complain Satisfaction Score Card Type \  
0 101348.88 1 1 2 DIAMOND   
1 112542.58 0 1 3 DIAMOND   
2 113931.57 1 1 3 DIAMOND   
3 93826.63 0 0 5 GOLD   
4 79084.10 0 0 5 GOLD   
... ... ... ... ... ...   
9995 96270.64 0 0 1 DIAMOND   
9996 101699.77 0 0 5 PLATINUM   
9997 42085.58 1 1 3 SILVER   
9998 92888.52 1 1 2 GOLD   
9999 38190.78 0 0 3 DIAMOND   
  
 Point Earned   
0 464   
1 456   
2 377   
3 350   
4 425   
... ...   
9995 300   
9996 771   
9997 564   
9998 339   
9999 911   
  
[10000 rows x 18 columns]

data\_to\_scale = data[['Balance', 'EstimatedSalary']]  
data\_to\_scale.describe()

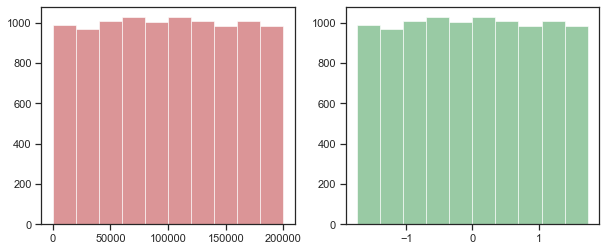
Balance EstimatedSalary  
count 10000.000000 10000.000000  
mean 76485.889288 100090.239881  
std 62397.405202 57510.492818  
min 0.000000 11.580000  
25% 0.000000 51002.110000  
50% 97198.540000 100193.915000  
75% 127644.240000 149388.247500  
max 250898.090000 199992.480000

scaler1 = StandardScaler()

data\_scaled1 = pd.DataFrame(scaler1.fit\_transform(data\_to\_scale), columns=data\_to\_scale.columns)  
data\_scaled1.describe()

Balance EstimatedSalary  
count 1.000000e+04 1.000000e+04  
mean -6.252776e-17 -2.877698e-17  
std 1.000050e+00 1.000050e+00  
min -1.225848e+00 -1.740268e+00  
25% -1.225848e+00 -8.535935e-01  
50% 3.319639e-01 1.802807e-03  
75% 8.199205e-01 8.572431e-01  
max 2.795323e+00 1.737200e+00

fig, axs = plt.subplots(1, 2, figsize=(10,4))  
axs[0].hist(data["EstimatedSalary"], color="r", alpha=0.6)  
axs[1].hist(data\_scaled1["EstimatedSalary"], color="g", alpha=0.6)  
plt.show()

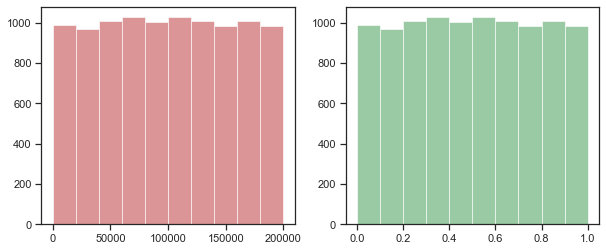


MinMax-масштабирование

scaler2 = MinMaxScaler()  
data\_scaled2 = pd.DataFrame(scaler2.fit\_transform(data\_to\_scale), columns=data\_to\_scale.columns)  
data\_scaled2.describe()

Balance EstimatedSalary  
count 10000.000000 10000.000000  
mean 0.304848 0.500441  
std 0.248696 0.287580  
min 0.000000 0.000000  
25% 0.000000 0.254977  
50% 0.387402 0.500960  
75% 0.508749 0.746955  
max 1.000000 1.000000

fig, axs = plt.subplots(1, 2, figsize=(10,4))  
axs[0].hist(data["EstimatedSalary"], color="r", alpha=0.6)  
axs[1].hist(data\_scaled2["EstimatedSalary"], color="g", alpha=0.6)  
plt.show()

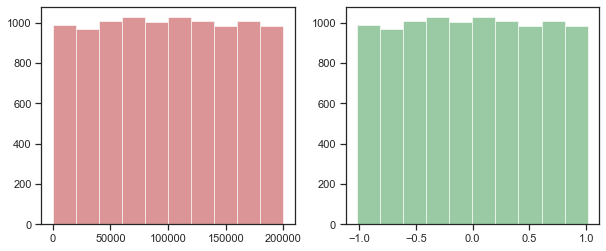


По максимальному значению

scaler3 = RobustScaler()  
data\_scaled3 = pd.DataFrame(scaler3.fit\_transform(data\_to\_scale), columns=data\_to\_scale.columns)  
data\_scaled3.describe()

Balance EstimatedSalary  
count 1.000000e+04 10000.000000  
mean -1.622686e-01 -0.001054  
std 4.888384e-01 0.584539  
min -7.614800e-01 -1.018257  
25% -7.614800e-01 -0.499987  
50% -5.700193e-17 0.000000  
75% 2.385200e-01 0.500013  
max 1.204124e+00 1.014356

fig, axs = plt.subplots(1, 2, figsize=(10,4))  
axs[0].hist(data["EstimatedSalary"], color="r", alpha=0.6)  
axs[1].hist(data\_scaled3["EstimatedSalary"], color="g", alpha=0.6)  
plt.show()



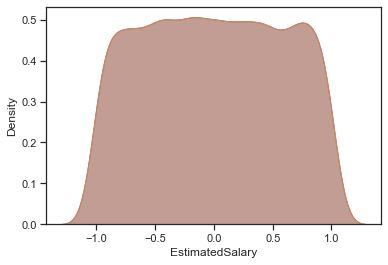
Обработка выбросов

from enum import Enum  
# Тип вычисления верхней и нижней границы выбросов   
class OutlierBoundaryType(Enum):  
 SIGMA = 1  
 QUANTILE = 2  
 IRQ = 3  
  
# Функция вычисления верхней и нижней границы выбросов  
def get\_outlier\_boundaries(df, outlier\_boundary\_type: OutlierBoundaryType):   
 if outlier\_boundary\_type == OutlierBoundaryType.SIGMA:   
 K1 = 3   
 lower\_boundary = df.mean() - (K1 \* df.std())   
 upper\_boundary = df.mean() + (K1 \* df.std())  
 elif outlier\_boundary\_type == OutlierBoundaryType.QUANTILE:  
 lower\_boundary = df.quantile(0.05)   
 upper\_boundary = df.quantile(0.95)  
 elif outlier\_boundary\_type == OutlierBoundaryType.IRQ:  
 K2 = 1.5  
 IQR = df.quantile(0.75) - df.quantile(0.25)   
 lower\_boundary = df.quantile(0.25) - (K2 \* IQR)   
 upper\_boundary = df.quantile(0.75) + (K2 \* IQR)   
 return lower\_boundary, upper\_boundary

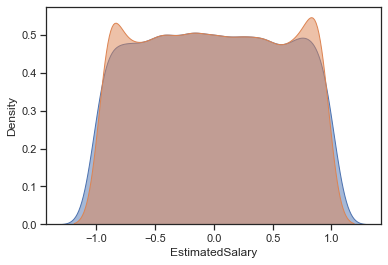
data2 = data\_scaled3.copy()

for obt in OutlierBoundaryType:  
 lower\_boundary, upper\_boundary = get\_outlier\_boundaries(data\_scaled3['EstimatedSalary'], obt)  
  
 data2['EstimatedSalary'] = np.where(data\_scaled3['EstimatedSalary'] > upper\_boundary, upper\_boundary,  
 np.where(data2['EstimatedSalary'] < lower\_boundary, lower\_boundary,  
 data2['EstimatedSalary']))  
 sns.kdeplot(data\_scaled3['EstimatedSalary'], alpha=0.5, fill=True)  
 sns.kdeplot(data2['EstimatedSalary'], alpha=0.5, fill=True)  
 print(obt)  
 plt.show()

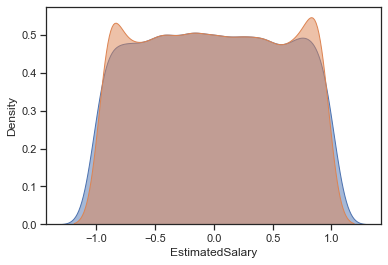
OutlierBoundaryType.SIGMA



OutlierBoundaryType.QUANTILE



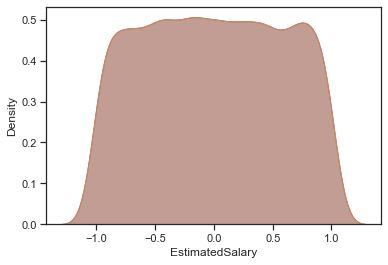
OutlierBoundaryType.IRQ



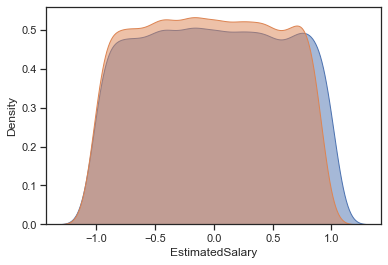
Удаление выбросов

for obt in OutlierBoundaryType:  
 lower\_boundary, upper\_boundary = get\_outlier\_boundaries(data\_scaled3['EstimatedSalary'], obt)  
 temp\_df = np.where(data\_scaled3['EstimatedSalary'] > upper\_boundary, True,  
 np.where(data2['EstimatedSalary'] < lower\_boundary, True, False))  
 data\_trimmed = data\_scaled3.loc[~(temp\_df), ]  
 sns.kdeplot(data\_scaled3['EstimatedSalary'], alpha=0.5, fill=True)  
 sns.kdeplot(data\_trimmed['EstimatedSalary'], alpha=0.5, fill=True)  
 print(obt)  
 plt.show()

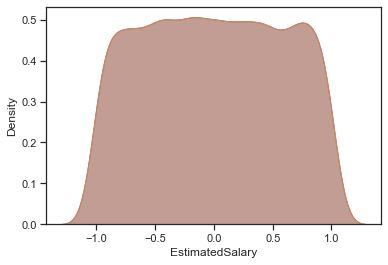
OutlierBoundaryType.SIGMA



OutlierBoundaryType.QUANTILE



OutlierBoundaryType.IRQ



Отбор признаков

# Удаление коррелирующих признаков

wine = load\_wine()  
data = pd.DataFrame(data=wine.data, columns= wine.feature\_names)  
y = wine['target']

data['alcohol\_2'] = data['alcohol'] + np.random.rand(len(data['alcohol']))  
data.head()

alcohol malic\_acid ash alcalinity\_of\_ash magnesium total\_phenols \  
0 14.23 1.71 2.43 15.6 127.0 2.80   
1 13.20 1.78 2.14 11.2 100.0 2.65   
2 13.16 2.36 2.67 18.6 101.0 2.80   
3 14.37 1.95 2.50 16.8 113.0 3.85   
4 13.24 2.59 2.87 21.0 118.0 2.80   
  
 flavanoids nonflavanoid\_phenols proanthocyanins color\_intensity hue \  
0 3.06 0.28 2.29 5.64 1.04   
1 2.76 0.26 1.28 4.38 1.05   
2 3.24 0.30 2.81 5.68 1.03   
3 3.49 0.24 2.18 7.80 0.86   
4 2.69 0.39 1.82 4.32 1.04   
  
 od280/od315\_of\_diluted\_wines proline alcohol\_2   
0 3.92 1065.0 14.450588   
1 3.40 1050.0 13.834093   
2 3.17 1185.0 13.867882   
3 3.45 1480.0 14.458314   
4 2.93 735.0 14.025412

def make\_corr\_df(data):  
 cr = data.corr()  
 cr = cr.abs().unstack()  
 cr = cr.sort\_values(ascending=False)  
 cr = cr[cr >= 0.80]  
 cr = cr[cr < 1]  
 cr = pd.DataFrame(cr).reset\_index()  
 cr.columns = ['f1', 'f2', 'corr']  
 return cr  
  
def corr\_groups(cr):  
 grouped\_feature\_list = []  
 correlated\_groups = []  
 for feature in cr['f1'].unique():  
 if feature not in grouped\_feature\_list:  
 correlated\_block = cr[cr['f1'] == feature]  
 cur\_dups = list(correlated\_block['f2'].unique()) + [feature]  
 grouped\_feature\_list = grouped\_feature\_list + cur\_dups  
 correlated\_groups.append(cur\_dups)  
 return correlated\_groups

corr\_df = make\_corr\_df(data)   
print(corr\_df)   
corr\_groups(corr\_df)

f1 f2 corr  
0 alcohol alcohol\_2 0.937245  
1 alcohol\_2 alcohol 0.937245  
2 flavanoids total\_phenols 0.864564  
3 total\_phenols flavanoids 0.864564

[['alcohol\_2', 'alcohol'], ['total\_phenols', 'flavanoids']]

Метод обёртывания

from sklearn.neighbors import KNeighborsClassifier   
from mlxtend.feature\_selection import ExhaustiveFeatureSelector as EFS   
knn = KNeighborsClassifier(n\_neighbors=3)  
efs1 = EFS(knn,  
 min\_features=2,  
 max\_features=4,  
 scoring='accuracy',  
 print\_progress=True,  
 cv=5)  
efs1 = efs1.fit(data, y)

Features: 1456/1456

print('Best accuracy score: %.2f' % efs1.best\_score\_)  
print('Best subset:', efs1.best\_feature\_names\_)

Best accuracy score: 0.94  
Best subset: ('alcohol', 'total\_phenols', 'flavanoids', 'color\_intensity')

Метод Вложений

e\_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max\_iter=500, random\_state=1)  
e\_lr1.fit(data, y)  
e\_lr1.coef\_

array([[ 7.17648954e-01, 4.54715366e+00, 1.79166689e+01,  
 -2.37347111e+00, 1.06874234e-02, 3.55350512e-01,  
 9.17079642e+00, 0.00000000e+00, -5.33219296e+00,  
 -1.31500548e+00, -6.20093824e+00, 2.32764141e+00,  
 6.22036096e-02, 9.05298436e-01],  
 [-6.78797065e-01, -3.27285624e+00, -1.74065140e+01,  
 1.45943555e+00, -4.27677239e-02, 1.42249410e+00,  
 2.70522866e+00, 1.42977760e+01, 3.50088926e+00,  
 -6.46923572e+00, 2.35765830e+01, -3.98905510e+00,  
 -4.15178762e-02, -7.97731130e-01],  
 [ 1.25661133e-01, 9.75917803e-01, 1.07945627e+00,  
 1.24523093e-01, 2.36929237e-02, -5.38665639e-01,  
 -6.29871199e+00, -5.39669186e+00, -2.08080472e+00,  
 1.51833330e+00, -5.48393053e+00, -2.66846449e+00,  
 2.66907242e-03, 1.29646172e-01]])

from sklearn.feature\_selection import SelectFromModel  
  
sel\_e\_lr1 = SelectFromModel(e\_lr1)  
sel\_e\_lr1.fit(data, y)  
sel\_e\_lr1.get\_support()

array([ True, True, True, True, True, True, True, True, True,  
 True, True, True, True, True])

Обработка нестандартного признака

data3 = pd.read\_csv('countryLockdowndates.csv', sep=",")  
print(data.shape)  
data.head()

(307, 5)

Country/Region Province Date Type \  
0 Afghanistan NaN 24/03/2020 Full   
1 Albania NaN 08/03/2020 Full   
2 Algeria NaN 24/03/2020 Full   
3 Andorra NaN 16/03/2020 Full   
4 Angola NaN 24/03/2020 Full   
  
 Reference   
0 https://www.thestatesman.com/world/afghan-govt...   
1 https://en.wikipedia.org/wiki/2020\_coronavirus...   
2 https://www.garda.com/crisis24/news-alerts/325...   
3 https://en.wikipedia.org/wiki/2020\_coronavirus...   
4 https://en.wikipedia.org/wiki/2020\_coronavirus...

# Сконвертируем дату и время в нужный формат  
data3["Date"] = data3.apply(lambda x: pd.to\_datetime(x["Date"], format='%d/%m/%Y'), axis=1)

data3.head(5)

Country/Region Province Date Type \  
0 Afghanistan NaN 2020-03-24 Full   
1 Albania NaN 2020-03-08 Full   
2 Algeria NaN 2020-03-24 Full   
3 Andorra NaN 2020-03-16 Full   
4 Angola NaN 2020-03-24 Full   
  
 Reference   
0 https://www.thestatesman.com/world/afghan-govt...   
1 https://en.wikipedia.org/wiki/2020\_coronavirus...   
2 https://www.garda.com/crisis24/news-alerts/325...   
3 https://en.wikipedia.org/wiki/2020\_coronavirus...   
4 https://en.wikipedia.org/wiki/2020\_coronavirus...