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 и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по Лабораторной работе №3

«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.»

по дисциплине «Технология машинного обучения»

Выполнил:
студент группы ИУ5-61Б
И.А. Калинников

Проверил:
Ю.Е. Гапанюк

2023 г.

In [27]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import *
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style ="ticks")


from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer, MissingIndicator
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler, StandardScaler, Normalizer
```

In [2]:
```
df = pd.read_csv('bank_dataset.csv')
```

In [3]:
```
data = df.copy()
```

In [4]:
```
df.head()
```

Out[4]:

	userid	score	City	Gender	Age	Objects	Balance	Products	CreditCard	Loyalty	estimated_salary	Churn
0	15677338	619	Ярославль	Ж	42	2	NaN	1	1	1	101348.88	1
1	15690047	608	Рыбинск	Ж	41	1	83807.86	1	0	1	112542.58	0
2	15662040	502	Ярославль	Ж	42	8	159660.80	3	1	0	113931.57	1
3	15744090	699	Ярославль	Ж	39	1	NaN	2	0	0	93826.63	0
4	15780624	850	Рыбинск	Ж	43	2	125510.82	1	1	1	79084.10	0

In [5]:
```
#Кодирование категориальных признаков

df["City"] = df["City"].astype('category')

df["Gender"] = df["Gender"].astype('category')


#Назначить закодированную переменную новосу столбцу с помощью метода доступа
df["City_cat"] = df["City"].cat.codes
df["Gender_cat"] = df["Gender"].cat.codes
```

In [6]:
```
df = df.drop(['City', 'Gender'], axis=1)
```

In [7]:
```
df.head()
```

Out[7]:

	userid	score	Age	Objects	Balance	Products	CreditCard	Loyalty	estimated_salary	Churn	City_cat	Gender_cat
0	15677338	619	42	2	NaN	1	1	1	101348.88	1	2	0
1	15690047	608	41	1	83807.86	1	0	1	112542.58	0	1	0
2	15662040	502	42	8	159660.80	3	1	0	113931.57	1	2	0
3	15744090	699	39	1	NaN	2	0	0	93826.63	0	2	0
4	15780624	850	43	2	125510.82	1	1	1	79084.10	0	1	0

In [8]:
```
df.isna().sum()
```

Out[8]:
```
userid              0
score               0
Age                 0
Objects             0
Balance          3617
Products            0
CreditCard          0
Loyalty             0
estimated_salary    0
Churn               0
City_cat            0
Gender_cat          0
dtype: int64
```

In [9]:
```
df = df.dropna()
```

In [10]:
```
df.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
userid	6383.0	1.573310e+07	71929.130555	15608437.00	1.567094e+07	15732262.00	1.579584e+07	15858426.00
score	6383.0	6.511385e+02	96.934609	350.00	5.840000e+02	652.00	7.180000e+02	850.00
Age	6383.0	3.919771e+01	10.476208	18.00	3.200000e+01	38.00	4.400000e+01	92.00
Objects	6383.0	4.979633e+00	2.909514	0.00	2.000000e+00	5.00	8.000000e+00	10.00
Balance	6383.0	1.198275e+05	30095.056462	3768.69	1.001820e+05	119839.69	1.395123e+05	250898.09
Products	6383.0	1.386025e+00	0.577011	1.00	1.000000e+00	1.00	2.000000e+00	4.00
CreditCard	6383.0	6.992010e-01	0.458641	0.00	0.000000e+00	1.00	1.000000e+00	1.00
Loyalty	6383.0	5.135516e-01	0.499855	0.00	0.000000e+00	1.00	1.000000e+00	1.00
estimated_salary	6383.0	1.007174e+05	57380.316584	11.58	5.173685e+04	101139.30	1.495966e+05	199970.74
Churn	6383.0	2.407959e-01	0.427600	0.00	0.000000e+00	0.00	0.000000e+00	1.00
City_cat	6383.0	1.013630e+00	0.894271	0.00	0.000000e+00	1.00	2.000000e+00	2.00
Gender_cat	6383.0	5.473915e-01	0.497788	0.00	0.000000e+00	1.00	1.000000e+00	1.00

In [11]:
```
y = df['Churn']  #Наименования признаков
X = df.drop('Churn', axis=1) # Значения признаков
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state= 45)
# random_state позволяет задавать базовое значение для генератора случайных чисел, чтобы сделать выборку неслучайной
```

In [13]:
```
# Размер обучающей выборки
X_train.shape, y_train.shape
```

Out[13]:
```
((4787, 11), (4787,))
```

In [14]:
```
# Размер тестовой выборки
X_test.shape, y_test.shape
```

Out[14]:
```
((1596, 11), (1596,))
```

In [15]:
```
np.unique(y_train)
```

Out[15]:
```
array([0, 1])
```

In [16]:
```
np.unique(y_test)
```

Out[16]:
```
array([0, 1])
```

In [17]:
```
#Масштабирование данных
scaler = MinMaxScaler().fit(X_train)
X_train = pd.DataFrame(scaler.transform(X_train), columns = X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns = X_train.columns)
X_train.describe()
```

Out[17]:

	userid	score	Age	Objects	Balance	Products	CreditCard	Loyalty	estimated_salary	City_cat	Gender_cat
count	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000	4787.000000
mean	0.497183	0.602214	0.287423	0.498997	0.439925	0.129030	0.697514	0.512847	0.508251	0.506476	0.544809
std	0.287851	0.193475	0.140587	0.291413	0.128051	0.192955	0.459382	0.499887	0.285996	0.447354	0.498040
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.246767	0.466934	0.189189	0.200000	0.356705	0.000000	0.000000	0.000000	0.261767	0.000000	0.000000
50%	0.492614	0.609218	0.270270	0.500000	0.439208	0.000000	1.000000	1.000000	0.513190	0.500000	1.000000
75%	0.749771	0.739479	0.351351	0.800000	0.523497	0.333333	1.000000	1.000000	0.750887	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [24]:
```
def test_model(model):
    print("mean_absolute_error:",
          mean_absolute_error(y_test, model.predict(X_test)))
    print("mean_squared_error:",
          mean_squared_error(y_test, model.predict(X_test)))
    print("median_absolute_error:",
          median_absolute_error(y_test, model.predict(X_test)))
    print("r2_score:",
          r2_score(y_test, model.predict(X_test)))
```

Попробуем метод ближайших соседей с гиперпараметром K = 10:

In [31]:
```
reg_10 = KNeighborsRegressor(n_neighbors=10)
reg_10.fit(X_train, y_train)
```

Out[31]:
```
KNeighborsRegressor(n_neighbors=10)
```

In [32]:
```
test_model(reg_10)
```

```
mean_absolute_error: 0.2895363408521303
mean_squared_error: 0.15360275689223057
median_absolute_error: 0.2
r2_score: 0.12241210313232487
```