УДК 004

Е. Е. Белова, О. В. Толстель

ИСПОЛЬЗОВАНИЕ БИБЛИОТЕК ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ АНАЛИЗА ОТТОКА КЛИЕНТОВ БАНКА

В настоящее время борьба за клиентов среди банков обостряется. Показатель оттока клиентов очень важен для анализа деятельности банка и планировании дальнейшей работы. В статье использованы пять алгоритмов для расчета оттока клиентов банка. Коды программ написаны на Python.

At the present time, competition for customers among banks is escalating. The indicator of customer churn is very important for the analysis and planning bank work. Five algorithms for calculating the outflow of bank customers are used in this article. Program codes are written in Python.

Ключевые слова: клиенты банка, анализ данных, прогнозирование, логистическая регрессия, наивный Байес, деревья решений, алгоритм ближайшего соседа, случайный лес, Python.

Keywords: bank customers, data analysis, forecasting, logistic regression, Guassian naive Bayes, decision trees, KNN, random forest, Python.

Отток клиентов (англ. *churn*) — это потеря клиентов, выраженная в отсутствии покупок или платежей в течение определенного периода времени.

Показатель оттока крайне важен для компаний с подписной и транзакционной моделями бизнеса, подразумевающими регулярные платежи в сторону компании [1].

Данный показатель зависит от выбранной методики расчета. Существенным параметром является величина периода времени неактивности клиента, чтобы считать данного клиента потерянным для банка. Если данный параметр выбрать неверно, то это приведет к искажениям в оценке и трактовке оттока.

Базовая формула для расчета оттока имеет следующий вид:

$$\mu = \frac{l_k}{s_k},$$

где μ — коэффициент оттока клиентов; l_k — количество потерянных клиентов за интервал времени; s_k — общее количество клиентов на начало периода измерения.

Интервал времени, на котором измеряется отток, должен выбираться с учетом специфики бизнеса (сезонность, стадия жизненного цикла продукта, является ли данный товар или услуга новыми и т. д.).

Для полноты картины сам по себе коэффициент оттока является малоинформативным показателем. Важно понимать структуру оттока:

1) какие сегменты клиентов в первую очередь подвержены оттоку и почему;

2) какие факторы чаще всего служат причиной потери клиентов;

3) какова динамика оттока и тенденции по его структуре.

Для полного анализа необходимо дать оценку оттока в денежном выражении. Эта оценка возможна при моделировании денежных потоков, которые получила бы компания, если бы клиент продолжал пользоваться ее услугами. Формула оценки ущерба компании от оттока клиентов имеет вид

$$\gamma = l_k \cdot p \cdot \nu ,$$

где γ — ущерб от оттока; l_k — количество потерянных клиентов; p — средний чек (размер платежа); ν — частота совершения покупок (количество платежей) за интервал времени.

Рассмотрим модель расчета оттока клиентов банка с использованием Python. Задача состоит в том, чтобы найти параметры, максимально влияющие на отток клиентов банка, и создать гипотезу, которая предсказывала бы отток клиентов банка.

В выборку данных для этой модели войдут возраст, пол и количество денежных средств на счету клиента.

Будем использовать следующие инструменты:

— NumPy — это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц;

— Pandas — программная библиотека на языке Python для обработки и анализа данных (строится поверх NumPy);

— Sklearn (Scikit-learn) — бесплатная библиотека машинного обучения для языка Python;

— Matplotlib — библиотека для визуализации данных.

Перечень входных данных [2] приведен в таблице 1.

Таблица 1

Входные данные

Название	Расшифровка
RowNumber	Номер строки
CustomerId	Id пользователя
Surname	Фамилия клиента
CreditScore	Кредитный рейтинг
Geography	Откуда клиент
Gender	Пол клиента
Age	Возраст клиента
Tenure	Срок пребывания клиента в компании
Balance	Баланс клиента
NumOfProducts	Количество продуктов, используемых клиентом
HasCrCard	Имеющаяся кредитная карта
IsActiveMember	Является ли клиент активным пользователем
EstimatedSalary	Оценка заработной платы
Exited	Клиент покинул банк

Импорт библиотек

Перед началом анализа данных необходимо импортировать все нужные библиотеки:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics, svm, datasets
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,
classification_report, accuracy_score, roc_curve, auc
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

Импорт данных

Считаем входные данные. Выведем первые 10 строк данных:

```
dataset = pd.read_csv('Churn_Modelling.csv')
dataset.head(10)
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	E:
1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	
3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	
4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	
6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	
7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	
8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	
9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	
10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	

Можно посмотреть информацию по этим данным, используя команду info(). Полученный список содержит все столбцы, их типы данных и количество ненулевых значений:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
```

```
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```


Следует отметить, что нулевых значений нет, что свидетельствует о полноте данных.

Выведем описательную статистику. С помощью команды describe() можно посмотреть минимальные / максимальные и средние значения для столбцов, а также другую информацию:

```
dataset.describe()
```

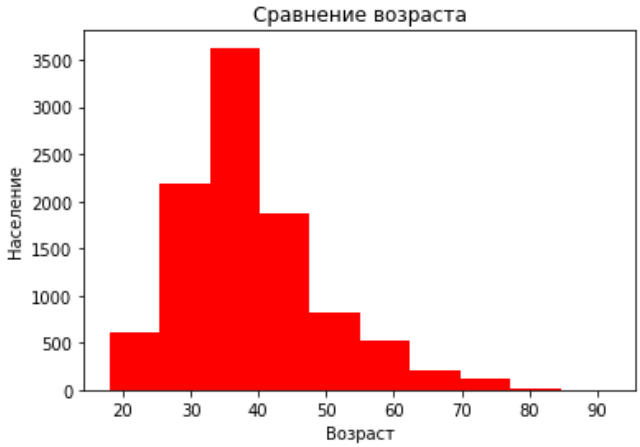
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

Анализ данных

Проанализируем данные. Например, выведем информацию о возрасте:

```
dataset['Age'].value_counts()
```

```
plt.hist(x = dataset.Age, bins = 10, color = 'red')
plt.title('Сравнение возраста')
plt.xlabel('Возраст')
plt.ylabel('Население')
plt.show()
```



Также выведем количество людей, разделенных по географическому признаку:

```
dataset['Geography'].value_counts()
```

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```


Посмотрим, в каком соотношении находятся мужчины и женщины:

```
dataset['Gender'].value_counts()
```

```
Male       5457
Female     4543
Name: Gender, dtype: int64
```

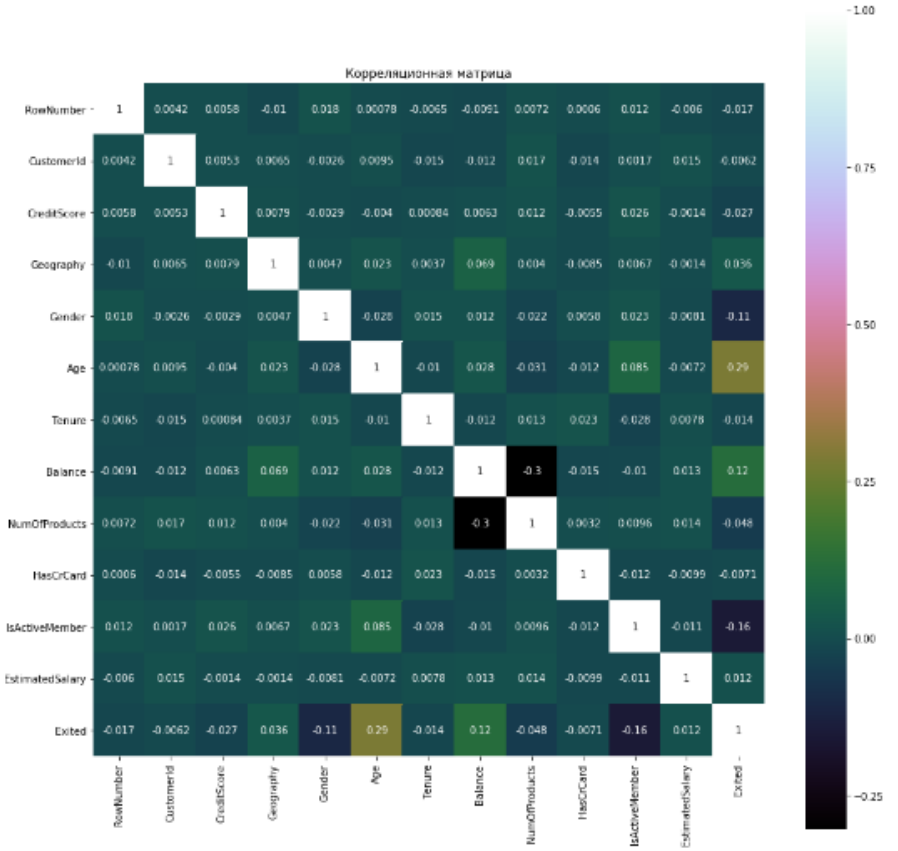
Для правильной работы классификатора необходимо преобразование категориального признака в числовой. Нам нужно изменить 2 столбца: Gender и Geography. Преобразование выглядит следующим образом:

```
dataset['Geography'].replace("France", 1, inplace= True)
dataset['Geography'].replace("Germany", 2, inplace=True)
dataset['Geography'].replace("Spain", 3, inplace = True)
dataset['Gender'].replace("Female", 0, inplace = True)
dataset['Gender'].replace("Male", 1, inplace=True)
```

Создадим корреляционную матрицу, то есть таблицу, в строках и столбцах которой записан коэффициент корреляции между соответствующими параметрами. Данная квадратная матрица симметрична относительно главной диагонали.

```
correlation = dataset.corr()
plt.figure(figsize = (15, 15))
sns.heatmap(correlation, vmax = 1, square = True, annot = True, cmap = 'cubehelix')
plt.title('Корреляционная матрица')
plt.show()
```

Корреляционная матрица показывает, какие параметры будут влиять на результат. Сразу можно выделить 4 положительные корреляции: Balance, Age, Geography и EstimatedSalary.



Предварительная обработка данных

Разобьем набор данных на две части (x — независимые переменные, y — зависимые переменные):

```
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

Разделим наш набор данных. Это поможет избежать проблем с переобучением:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

Моделирование

Данную модель будем реализовывать с помощью 5 алгоритмов.

1. Логистическая регрессия (Logistic Regression) — метод построения линейного классификатора, который позволяет оценивать апостериорные вероятности принадлежности объектов классам.

```
print('Logistic Regression:')
logr = LogisticRegression()
logr.fit(X_train, y_train)
predictions_lr = logr.predict(X_test)
print(classification_report(y_test, predictions_lr))
print(confusion_matrix(y_test, predictions_lr))
print('Logistic Regression =', accuracy_score(y_test, predictions_lr))
```

```
Logistic Regression:
              precision    recall  f1-score   support

           0       0.80      0.97      0.88      1595
           1       0.36      0.07      0.12       405

    accuracy                           0.79      2000
   macro avg       0.58      0.52      0.50      2000
weighted avg       0.71      0.79      0.72      2000

[[1545   50]
 [ 377   28]]
Logistic Regression = 0.7865
```

2. Наивный Байес (Guassian Naive Bayes) — алгоритм классификации, который основан на теореме Байеса с допущением о независимости признаков. Будем использовать нормальное распределение.

```
print('Guassian Naive Bayes:')
gnb = GaussianNB()
predictions_gnb = gnb.fit(X_train, y_train).predict(X_test)
print(classification_report(y_test, predictions_gnb))
print(confusion_matrix(y_test, predictions_gnb))
print('Guassian Naive Bayes =', accuracy_score(y_test, predictions_gnb))
```

```
Guassian Naive Bayes:
              precision    recall  f1-score   support

           0       0.81      0.96      0.88      1595
           1       0.37      0.09      0.14       405

    accuracy                           0.79      2000
   macro avg       0.59      0.52      0.51      2000
weighted avg       0.72      0.79      0.73      2000

[[1535   60]
 [ 370   35]]
Guassian Naive Bayes = 0.785
```

3. *Деревья решений* (Decision Trees).

```
dtree = DecisionTreeClassifier(max_leaf_nodes = 10)
dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=10,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
print('Decision Trees:')
predictions_dtree = dtree.predict(X_test)
print(classification_report(y_test, predictions_dtree))
print(confusion_matrix(y_test, predictions_dtree))
print('Decision Trees =', accuracy_score(y_test, predictions_dtree))
```

```
Decision Trees:
              precision    recall  f1-score   support

           0       0.88      0.95      0.92      1595
           1       0.73      0.50      0.59       405

    accuracy                           0.86      2000
   macro avg       0.80      0.72      0.75      2000
weighted avg       0.85      0.86      0.85      2000

[[1520   75]
 [ 204  201]]
Decision Trees = 0.8605
```

4. *Алгоритм ближайшего соседа* (KNN) выделяет из всех наблюдений k известные объекты (k-ближайших соседей), похожие на новый неизвестный ранее объект. На основе классов ближайших соседей выносится решение касательно нового объекта.

```
knn = KNeighborsClassifier(n_neighbors = 30)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=30, p=2,
                     weights='uniform')
```

```
print('KNN:')
predictions_knn = knn.predict(X_test)
print(classification_report(y_test, predictions_knn))
print(confusion_matrix(y_test, predictions_knn))
print('KNN =', accuracy_score(y_test, predictions_knn))
```

```
KNN:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      1595
           1       0.00      0.00      0.00       405
```

```
       accuracy                           0.80      2000
      macro avg       0.40      0.50      0.44      2000
   weighted avg       0.64      0.80      0.71      2000

   [[1593    2]
    [ 405    0]]
   KNN = 0.7965
```

5. Случайный лес (Random Forest) — множество решающих деревьев.

```
rf = RandomForestClassifier(n_estimators = 100)
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
print('Random Forest:')
predictions_rf = rf.predict(X_test)
print(classification_report(y_test, predictions_rf))
print(confusion_matrix(y_test, predictions_rf))
print('Random Forest =', accuracy_score(y_test, predictions_rf))
```

```
Random Forest:
              precision    recall  f1-score   support

           0       0.89      0.96      0.92      1595
           1       0.77      0.52      0.62       405

    accuracy                           0.87      2000
   macro avg       0.83      0.74      0.77      2000
weighted avg       0.86      0.87      0.86      2000

[[1531   64]
 [ 194  211]]
Random Forest = 0.871
```


Выводы и сравнения алгоритмов

В таблице 2 представлены результаты точности предсказаний.

Таблица 2

Результаты точности предсказаний

Алгоритм	Результат предсказания
Логистическая регрессия	0,7865
Наивный Байес	0,785
Деревья решений	0,8605
Алгоритм ближайшего соседа	0,7965
Случайный лес	0,871

Реальная статистика: из 10 000 клиентов ушло 2037 человек.

Вывод. *И*з пяти представленных алгоритмов наилучший результат показал случайный лес. Точность предсказания составляет 87,1%.

Список литературы

1. *Что* такое отток клиентов и как с ним бороться // NGM. URL: https://ngmsys.com/blog/churn-management (дата обращения: 01.12.2019).

2. *Bank-Customer-Churn-Modeling* // Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling (дата обращения: 23.11.2019).

Об авторах

Елизавета Евгеньевна Белова — магистрант, Балтийский федеральный университет им. И. Канта, Россия.
E-mail: el_liza_belova@mail.ru

Олег Владимирович Толстель — канд. техн. наук, доц., Балтийский федеральный университет им. И. Канта, Россия.
E-mail: oleg77764@mail.ru

The authors

Elizaveta E. Belova, Master's Student, I. Kant Baltic Federal University, Russia.
E-mail: el_liza_belova@mail.ru

Dr Oleg V. Tolstel', Associate Professor, I. Kant Baltic Federal University, Russia.
E-mail: oleg77764@mail.ru