Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Новосибирский государственный технический университет»

Кафедра прикладной математики

Отчет о проекте

Разработка программной системы машинного обучения для прогнозирования оттока клиентов

Факультет: ПМИ

Группа: ПМ-51

Студент: Москалев Д.И.

Преподаватель: Тракимус Ю.В.

Новосибирск

2019

Рассмотренные в проекте датасеты банков А, В взяты из открытых источников.

Банк А:

https://bit.ly/Bank_A_dataset

Банк В:

https://bit.ly/Bank_B_dataset

Последовательные решения задачи представлены в файлах на ЯП Python:

Bank_A.ipynb, *Bank_B.ipynb*.

Адаптивная визуализация полученных данных доступна в файле на ЯП C#:

Visualization.exe, ***Visualization_v2.exe*** (в отчете используется ***финальная*** версия).

Код всех программ и визуализации доступен в отчете (“Приложения А, Б, В”).

Данные для визуализации берутся из файлов

Банк А: Bank_A_RF.txt, Bank_A_CB.txt, *Bank_A_XGB.txt*.

Банк В: Bank_B_RF.txt, Bank_B_CB.txt, *Bank_B_XGB.txt*.

Структурные части работы:

1) Обзор практического применения задачи.

2) Сравнение методов классификации, выбранных для прогнозирования.

3) Подготовка и анализ данных для создания модели.

4) Реализация выбранных методов классификации для решения задачи прогнозирования оттока клиентов в банках.

5) Визуализация полученных результатов с помощью основных метрик.

6) Составление рекомендаций по выбору оптимального метода решения задачи.

АННОТАЦИЯ

Отчёт 68 с., 31 рис., 5 ч., 20 табл., 16 источников, 3 прил.

МАШИННОЕ ОБУЧЕНИЕ, DATA MINING, BIG DATA, ЗАДАЧА КЛАССИФИКАЦИИ, ПРОГНОЗИРОВАНИЕ ОТТОКА КЛИЕНТОВ.

Объектом исследования является набор данных о банках, полученный из общедоступных источников.

Цель работы – создание моделей прогнозирования оттока клиентов в банковской сфере используя методы классификации и проведение сравнительного анализа точности полученных результатов с помощью визуализации метрик качества.

В процессе работы были изучены эффективные методы классификации, выполнена обработка больших статистических данных, созданы модели классификации, разработано визуальное моделирование для проверки результатов задачи.

Результатом работы является реализация методов классификации, визуализация проверки точности созданной модели и рекомендации относительно области применения каждого из исследуемых методов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время актуальность прогнозирования оттока клиентов обуславливается тем, что большинство организаций получают прибыль благодаря лояльности клиентов. В условиях постоянной конкуренции компании предлагают различные акции, тем самым генерируя постоянный отток потребителей из других организаций либо из своих, если для клиентов данное предложение оказалось невыгодным.

Постепенно, объем данных для анализа стратегии максимизации прибыли становится настолько большим, что его невозможно обработать вручную и появляется необходимость в автоматизации процесса. Именно поэтому сейчас активно развивается сфера машинного обучения, позволяющая обрабатывать большие данные (Big Data) эффективно и точно. Актуальность и новизну теме добавляет и тот факт, что после решения задачи не требуется хранить первоначальные файлы большого размера. Машинное обучение позволяет перевести и улучшить исходные данные, при этом существенно сэкономив ресурсы, а полученная модель полностью адаптивна для дальнейшего использования или обучения на новых, либо уже предобученных данных. Каждый год появляются новые статьи и методы для улучшения точности в разных отраслях прогнозирования оттока клиентов, что показывает существенный интерес к задаче.

В качестве сферы прогнозирования выбрана банковская отрасль, так как она является одной из самых крупных и не подвержена изменениям в зависимости от страны банка и его внутренних условий. Для одного из банков было проведено исследование в статьях [7,12] и найдены наиболее точные классификаторы с помощью метрик. В качестве подтверждения полученных результатов дополнительно было проведено повторное исследование и визуальное представление результатов для банков A и B с целью доказательства преимущества одного из методов для всей банковской отрасли в целом.

Своевременное определение возможного оттока позволяет скорректировать предложение на рынке, определить причины и принять меры по удержанию

клиентов в будущем, а также адаптировать предложения для данного сегмента людей различными доступными способами.

В рамках выпускной квалификационной работы было необходимо разработать программную систему машинного обучения для прогнозирования оттока клиентов. Для этого требовалось:

1. Найти общедоступные корректные наборы данных.
2. Произвести подготовку данных для дальнейшего обучения модели.
3. Создать программные реализации методов классификации.
4. Обучить модель на обработанных данных.
5. Сравнить эффективные методы машинного обучения в задаче бинарной классификации с помощью метрик.
6. Определить сегмент клиентов, которые склонны отказаться от использования какого-либо продукта или услуги банка.
7. Разработать программу для визуального моделирования.
8. Визуализировать результаты работы методов.

В первом разделе выпускной квалификационной работы описывается практическое применение решаемой задачи и основные понятия, используемые в работе.

Во втором разделе выпускной квалификационной работы приведены методы решения задачи, обзор источников информации, преимущества и недостатки выбранных методов классификации.

В третьем разделе выпускной квалификационной работы рассмотрены выбранные языки программирования, обработка данных и подробное описание программ.

В четвертом разделе выпускной квалификационной работы проведен анализ данных, представлен процесс обучения моделей и полученные результаты.

В пятом разделе выпускной квалификационной работы выполнена визуализация результатов работы алгоритмов и произведена оценка точности полученных моделей.

1. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ

Главным достоинством прогнозирования оттока клиентов является его универсальность. Любая организация на основании данных за прошедший отчетный период времени может получить достаточно точное представление о текущем состоянии компании и принять меры по удержанию определенного сегмента клиентов.

1.1. ОСНОВНЫЕ ПРОБЛЕМЫ ПРОГНОЗИРОВАНИЯ

Практическое применение прогнозирования оттока клиентов позволяет ответить на множество интересующих компанию вопросов.

1.1.1. Причины ухода клиентов

Среди основных причин, по которым клиенты банка уходят либо переходят к другим организациям, наиболее распространены: выгода, безопасность и функциональные возможности банка. Если банк не готов предоставить условия выгоднее, чем в конкурирующей организации, то клиент начинает поиск и с некоторой вероятностью теряет интерес к данному банку.

1.1.2. Типы клиентов, которые больше всего подвержены оттоку

Существуют четыре основных типа клиентов:

- Имеющие желание и возможность для покупки.
- Имеющие желание для покупки, но не имеющие такой возможности.
- Имеющие возможность для покупки, но не имеющие желания.
- Не имеющие ни желания, ни возможности для покупки.

Далеко не всегда можно сказать, что именно последняя категория клиентов больше всего подвержена перестать пользоваться предложениями компании, так как существует разные виды мотиваций, например, акции, которые вместе с

появлением желания дают и возможность. Поэтому в модели предусмотрено множество признаков, которые позволяют сказать об этом более точно.

1.1.3. Подсчет оттока клиентов

Рассчитать отток клиентов всегда можно по формуле:

$$Churn = \frac{CustomersChurned}{AllCustomers},$$

где $CustomersChurned$ – количество клиентов, покинувших банк за отчетный период времени, $AllCustomers$ – общее количество клиентов.

1.1.4. Эффективность мер по борьбе с оттоком клиентов

Для понимания эффективности принятых мер необходимо взять показатели за новый период, совпадающего по длительности, признакам и сегменту пользователей.

1.1.5. Основные причины оттока

Среди причин оттока клиентов есть ряд универсальных для любой из сфер деятельности:

- Более привлекательные предложения конкурентов
- Низкий уровень клиентоориентированности
- Недовольство качеством продуктов и услуг компании
- Появление аналогичных продуктов
- Снижение лояльности к бренду
- Персональные факторы

Поскольку прогнозирование можно построить для любой отрасли, в которой ведется база данных, то однозначно ответить на этот вопрос можно лишь проанализировав соответствующую организацию и построив модель.

Для банковской сферы наиболее характерна разница в выгоде предложений.

1.2. ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ И ОБОЗНАЧЕНИЯ

Анализ данных (Data Mining) – инструмент обработки данных на подготовительном и завершающем этапах создания модели.

Ансамбль – композиция алгоритмов с целью получения более точного прогнозирования в задачах машинного обучения.

Большие данные (Big Data) – серия подходов, инструментов и методов для работы с данными больших объемов.

Бустинг (Boosting) – технология последовательного построения композиции решающих деревьев машинного обучения, где каждое следующее дерево стремится компенсировать недостатки композиции всех предыдущих.

Бэггинг (Bagging) – технология классификации, где все элементарные классификаторы вычисляются параллельно перед построением деревьев решений.

Выборка (Dataset) – набор данных.

Гиперпараметры – параметры модели.

График AUC ROC (Area Under Curve Receiver Operating Characteristic) – кривая оценки качества модели классификации, показывающий баланс между TPR *(True Positive Rate)* и FPR *(False Positive Rate)* [6].

Классификация – совокупность методов машинного обучения, в которых предсказывается принадлежность целевого признака к определенному классу.

Машинное обучение (Machine Learning) – обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.

Метрика – числовое или визуальное представление результатов для оценки качества модели и сравнения алгоритмов.

Признак – входной параметр, который может иметь числовой или категориальный тип.

Точность (Accuracy) – метрика точности решения модели классификатора на тестовой выборке.

Целевой признак – признак, который определяется в процессе решения задачи.

Шум – отсутствие, неверный формат или не уникальные данные.

2. ЗАДАЧА КЛАССИФИКАЦИИ

2.1. ПОСТАНОВКА ЗАДАЧИ

Формальная постановка задачи классификации представляет собой неизвестную целевую зависимость:

$$y^*: X \to Y, \tag{2.1}$$

где X – множество описаний объектов, Y – конечное множество номеров классов. Значения отображения (2.1) известны только на объектах конечной обучающей выборки набора данных:

$$X^n = \{(x_1, y_1), \dots, (x_n, y_n)\}, \tag{2.2}$$

где n – количество строк объектов.

Существует вероятностная постановка задачи, которая предполагает некое пространство, состоящее из множества пар с неизвестной вероятностной мерой P: $X \times Y$, где X – множество описаний объектов, Y – конечное множество номеров классов. При этом имеется обучающая выборка зависимостей (2.2), сгенерированная согласно вероятностной мере P.

В обеих постановках задачи необходимо построить алгоритм a: $X \to Y$, способный классифицировать произвольный объект $x \in X$ из набора данных.

Таким образом, задача решается методами классификации, которые по признакам, опираясь на обучающую выборку и вероятностную меру, с определенной вероятностью, определяют принадлежность к одному из классов. Признаком называется отображение f: $X \to D_f$, где D_f — множество допустимых значений признака. Если заданы признаки $features = \{f_1, \dots, f_n\}$, то вектор $x = (f_1(x), \dots, f_n(x))$называется признаковым описанием объекта $x \in X$, а множество $X = Df_1 \times \cdots \times Df_n$называют признаковым пространством [1].

В задаче бинарной классификации допустимое множество значений признаков D_f имеет бинарный признак: $D_f = \{f_1, f_2\}$. Обычно $f_1 = 0, f_2 = 1$.

Основные этапы решения задачи можно разделить на построение модели и визуализацию результатов (рисунок 1).

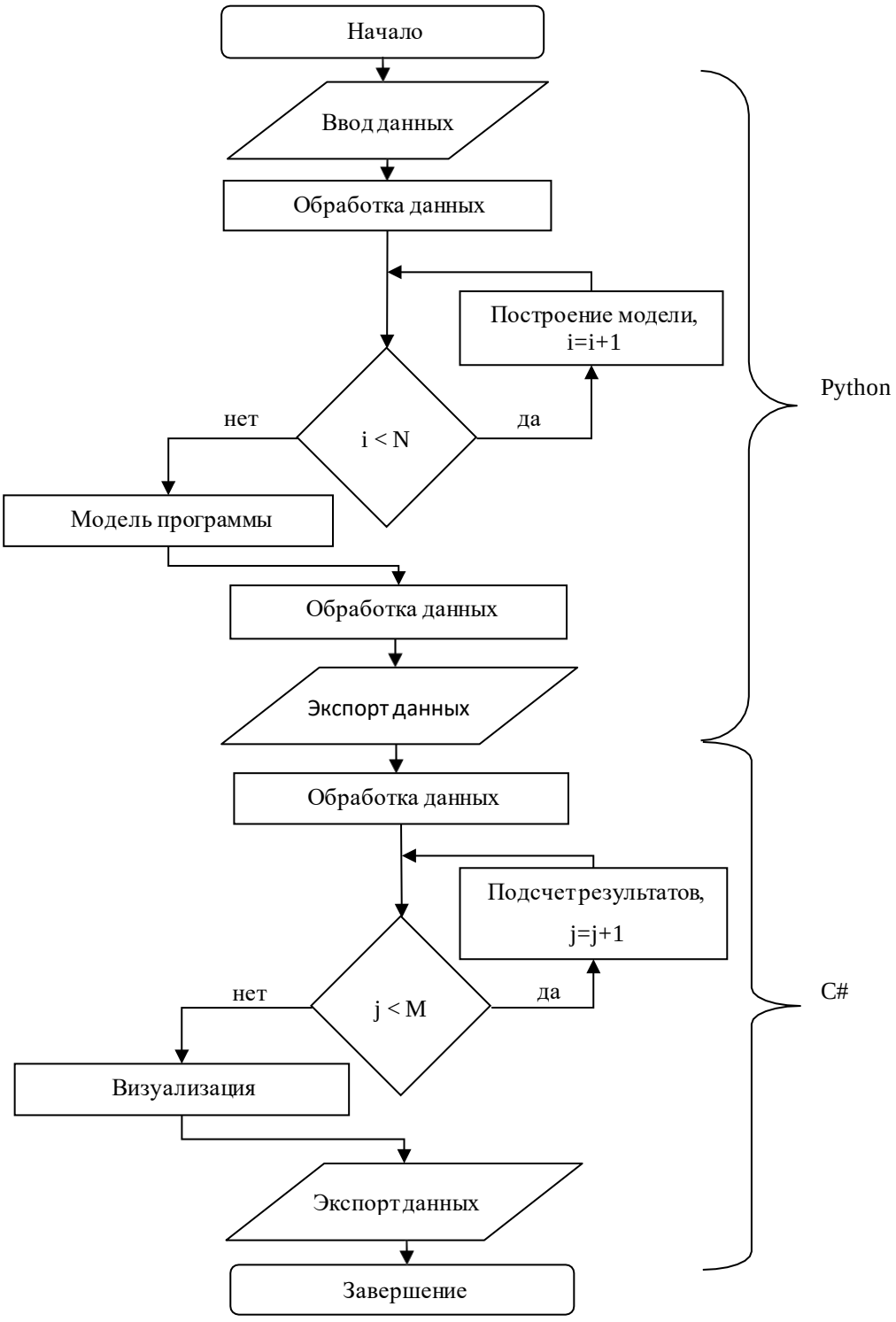

Рисунок 1 – Алгоритм решения задачи

Таким образом исходные и полученные результаты целевого признака на языке программирования Python являются входными данными для визуализации на языке C#.

2.2. ОБЗОР ИСТОЧНИКОВ ИНФОРМАЦИИ

Проблема прогнозирования оттока клиентов является одной из основополагающих задач в сфере машинного обучения, поэтому существует множество статей и наборов входных данных банков на специальных открытых ресурсах для создания собственных моделей решения и применения на основе обученных данных.

В качестве набора исходных данных для банка А был выбран общедоступный датасет *"Bank Marketing Data Set"* из репозитория *The UCI Machine Learning Repository* [10]. Основным преимуществом выбранного датасета можно назвать большой объем информации о клиентах и признаках.

Входными данными для банка В являются общедоступный набор данных *"Predicting Churn for Bank Customers"* с платформы *Kaggle* [14], который также имеет все необходимые атрибуты для построения модели.

Для разделения на классы по известным признакам был выбран дискриминантный анализ, использующий методы классификации, так как он показал более точный результат по сравнению с кластерным анализом [8].

2.3. ВЫБРАННЫЕ МЕТОДЫ РЕШЕНИЯ

Выбор методов обусловлен исследованием эффективности решения задач классификации в статье [12] для разных сфер работы с клиентами, а также актуальностью применения в современных сложных задачах классификации.

Выбранные методы основаны на деревьях решений и относятся к ансамблю алгоритмов с целью получения наиболее точного результата.

Общая формула дерева решений:
где b_j – константный прогноз, x – объект, в котором считаем прогноз, $R_1, \dots, R_j$ – области пространства объектов b_i, соответствующие листьям деревьев.

Таким образом, дерево решений (решающее дерево) – сумма прогнозов по всем областям, в которые попал объект x.

В качестве первого метода был выбран Random Forest (случайный лес), основанный на технологии параллельного построения решающих деревьев (Bagging). В общем случае случайный лес использует в ансамбле множество деревьев решений, каждое из которых по отдельности не всегда дает точные результаты, но за счет голосования деревьев относительно классифицируемых объектов, алгоритм позволяет найти точные значения. Значение считается принятым, если за него проголосовало большинство деревьев [15].

Алгоритм построения деревьев решений методом случайного леса с N решающими деревьями и глубиной дерева d представлен на рисунке 2.

Итоговый классификатор $a(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} a_i(x)$.

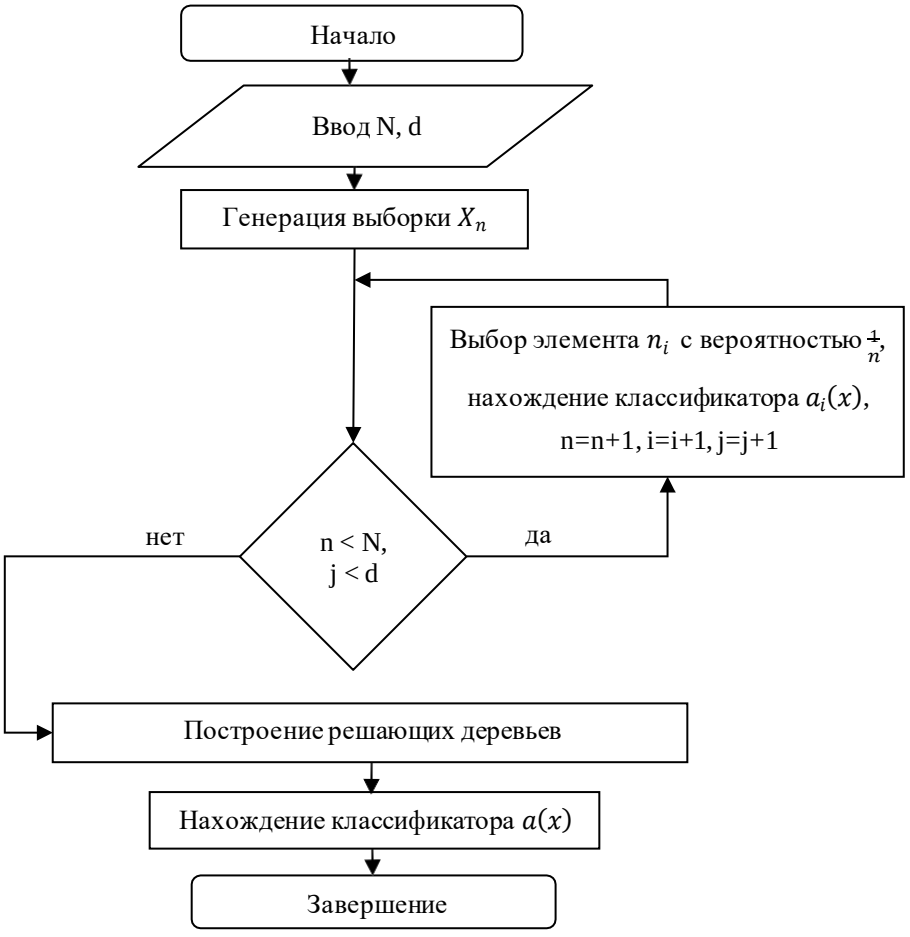

Рисунок 2 – Алгоритм построения случайного леса

Методы XGBoost и CatBoost основаны на градиентном бустинге и похожи между собой за исключением этапа построения деревьев. Метод XGBoost предоставляет возможность строить несбалансированные деревья, а CatBoost составляет только симметричные и сбалансированные. Алгоритм построения деревьев решений методами градиентного бустинга представлен на рисунке 3.

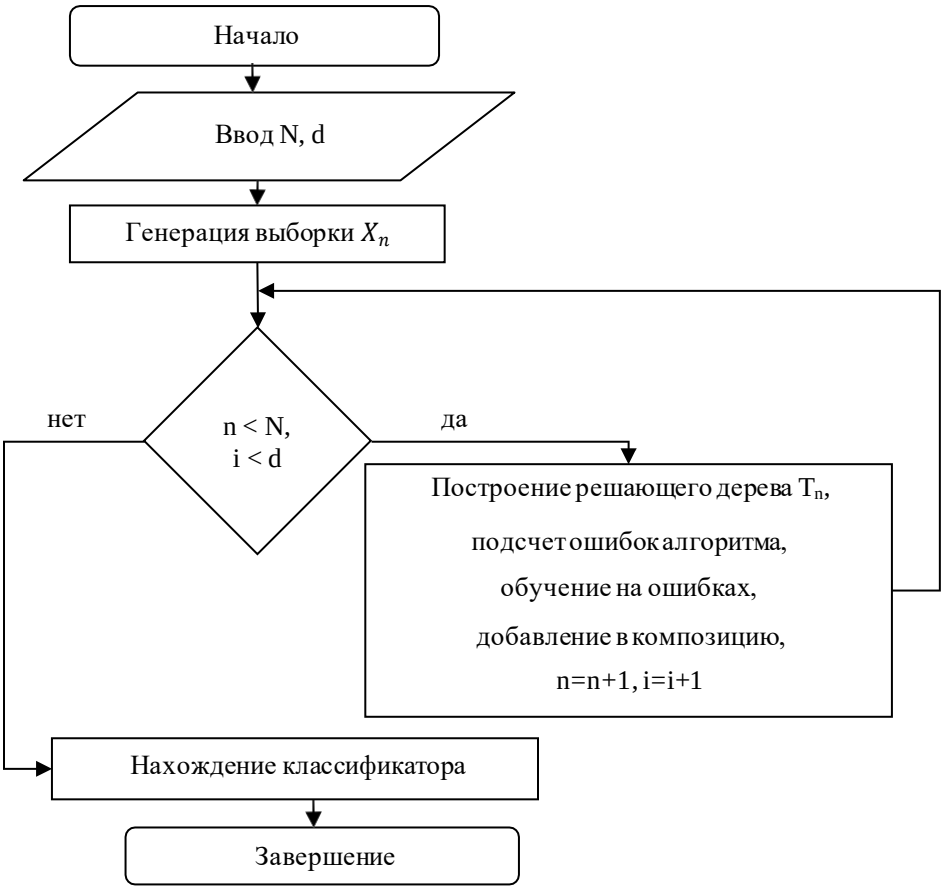

Рисунок 3 – Алгоритм построения градиентного бустинга

К основным преимуществам случайного леса можно отнести меньшее время работы за счет параллельного построения деревьев решений, точность классификатора и ряд других особенностей [3]. Главный недостаток метода – отсутствие возможности исправления ошибок прогнозирования во время обучения модели (таблица 1).

Для методов градиентного бустинга XGBoost и CatBoost к преимуществам можно отнести возможность исправления неточного прогнозирования за счет последовательного построения деревьев решений [4,11], но при этом стоит отметить длительное время обучения модели (таблицы 2-3).

Таблица 1 – Преимущества и недостатки метода Random Forest

Преимущества	***Недостатки***
Редко переобучается	Категориальные признаки нужно переводить в числовые
Высокая точность	Деревья не исправляют ошибки друг друга
Не требует тщательной настройки параметров	
Эффективно обрабатывает данные с большим числом признаков	
Обучение модели за сравнительно короткое время	
Параллельное построение деревьев решений	

Таблица 2 – Преимущества и недостатки метода XGBoost

Преимущества	***Недостатки***
Редко переобучается	Категориальные признаки нужно переводить в числовые
Высокая точность	Последовательное построение деревьев решений
Эффективно обрабатывает данные с большим числом признаков	
Деревья могут исправлять ошибки друг друга	

Таблица 3 – Преимущества и недостатки метода CatBoost

Преимущества	***Недостатки***
Редко переобучается	Последовательное построение деревьев решений
Высокая точность	Обучение модели за сравнительно долгое время
Может работать с категориальными признаками	
Эффективно обрабатывает данные с большим числом признаков	
Деревья могут исправлять ошибки друг друга	

3. ОПИСАНИЕ РАЗРАБОТАННЫХ ПРОГРАММ

3.1. МОДУЛЬ ПОСТРОЕНИЯ МОДЕЛЕЙ

Наименование программ: Bank_A.ipynb, Bank_B.ipynb.

Программное обеспечение, необходимое для функционирования программы: программы Anaconda Navigator, Jupiter Notebook и неустановленные библиотеки Python.

Язык программирования: для решения задач машинного обучения существуют два основных языка программирования: R и Python. Отметим тот факт, что R является узконаправленным языком программирования, созданным специально для статистической обработки данных. Что касается языка программирования Python, то он получил большую известность в сфере задач машинного обучения за счет своей универсальности и предустановленных библиотек классификаторов, визуальных решений и анализа данных. В настоящее время Python находится на 3 месте среди всех языков программирования, ввиду своей универсальности и удобства. В 2018 году язык был признан победителем по количеству прироста рейтинга [16]. Исходя из этого, для решения задачи прогнозирования оттока клиентов был выбран язык программирования Python версии 3.6.6.

Назначение программы: построение модели на основе выбранных классификаторов для прогнозирования оттока клиентов методами машинного обучения для задачи классификации.

Сведения о функциональных ограничениях на применение: наличие доступа в сеть Интернет для скачивания и обновления необходимых компонентов программы.

Способ вызова программы: запуск программы осуществляется из программы Jupiter Notebook расположенной по адресу *http://localhost:8888* нажатием кнопки запуска.

Алгоритм программы представлен на рисунке 4.

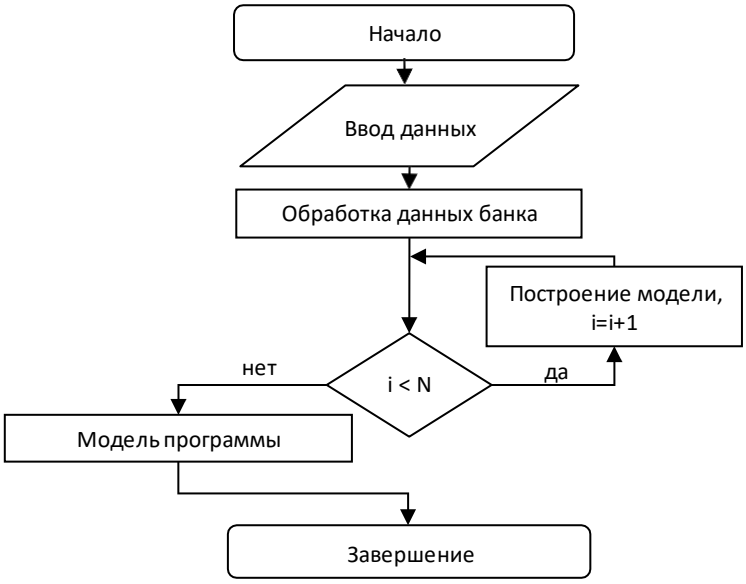

Рисунок 4 – Алгоритм программы для построения моделей

Используемые методы: в качестве методов решения задачи были выбраны *Random Forest, XGBoost*, которые показали одни из лучших результатов в задачах бинарной классификации среди 179 классификаторов [12] и были протестированы авторами статьи *"A Comparison of Classical Statistical & Machine Learning Techniques in Binary Classification"* на датасете банка А [7], а также *CatBoost* компании Яндекс, использующий градиентный бустинг на решающих деревьях [11].

Связи программы с другими программами: данная программа связана с программой для визуализации результатов работы алгоритмов на языке программирования C#.

3.1.1. Входные и выходные данные банка А

Описание входных данных: исходные данные содержатся в файле *Bank-full.csv* и представляют собой признаковое описание.

Формат: матрица входных данных имеет размер 45211×17, где 45211 – количество клиентов для обучения модели и прогнозирования, 17 – количество признаков, которые предлагается использовать для анализа данных и построения модели.

На рисунке 5 приведена часть таблицы входных данных банка А.

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

Рисунок 5 – Входные данные банка А

В таблице 4 представлены признаки банка А по которым строится модель. Входными данными являются признаки числового и категориального типа.

Таблица 4 – Признаки банка А

Описание	***Признак***
Возраст	age
Тип профессии	job
Семейное положение	marital
Образование клиента	education
Есть ли просроченные кредиты?	default
Среднегодовой баланс	balance
Есть ли ипотечный счет?	housing
Есть ли индивидуальный кредит?	loan
Способ связи	contact
День связи	day
Месяц связи	month
Время разговора в секундах	duration
Количество звонков компании	campaign
Сколько дней прошло после связи с прошлой компанией?	pdays
Количество разговоров до выбранной компании	previous
Результаты предыдущей компании	poutcome
Лояльность клиента банка (целевой признак)	y

Для корректной работы методов потребуется преобразовать категориальные признаки в числовые.

Прежде чем приступать к анализу данных и обучению модели, необходимо проанализировать исходные наборы данных для определения качества выборки и составления прогнозов.

Организация и предварительная подготовка входных данных: для анализа используется проверка на пустые, повторяющиеся и некорректные значения, которые могут создавать шум. В случае присутствия пустых значений удаляется строка таблицы входных данных. Дополнительно проверяются уникальные значения каждого важного признака, чтобы исключить повторения ключевых

атрибутов (таблица 5). Анализ таблицы 5 позволяет сделать вывод, что пустых значений нет, идентификаторы клиентов не повторяются.

Таблица 5 – Проверка корректности значений признаков банка А

Признак	***Значения***	
	Непустые	***Уникальные***
age	45211	77
job	45211	12
marital	45211	3
education	45211	4
default	45211	2
balance	45211	7168
housing	45211	2
loan	45211	2
contact	45211	3
day	45211	31
month	45211	12
duration	45211	1573
campaign	45211	48
pdays	45211	559
previous	45211	41
poutcome	45211	4
y	45211	2

После проверки корректности данных разделяем выборку на подвыборки. Для банка А соотношение обучающей и тестовой выборок взято 2 к 1.

На рисунке 6 представлена обучающая выборка банка А.

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
58	1	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3	0
44	2	2	1	0	29	1	0	2	5	8	151	1	-1	0	3	0
33	1	1	1	0	2	1	1	2	5	8	76	1	-1	0	3	0
47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3	0
33	2	2	3	0	1	0	0	2	5	8	198	1	-1	0	3	0

Рисунок 6 – Обучающая выборка банка А

В дальнейшем используем нормализированные переменные. Нормализация необходима для перевода признаков в один и тот же масштаб, иначе признак с большим числовым значением может сильно влиять на решение.

Описание и организация выходных данных: в качестве ответа для банка А определяется целевая переменная лояльности клиентов в виде вектора “y”, где

считается бинарная принадлежность клиента к классу “yes” или “no”, где “yes” – клиент остался, “no” – клиент ушел.

3.1.2. Входные и выходные данные банка В

Описание входных данных: исходные данные содержатся в файле *Churn_Modelling.csv* и представляют собой признаковое описание.

Формат: матрица входных данных банка В имеет размер 10000×14, где 10000 – количество клиентов для обучения модели и прогнозирования, 14 – количество признаков, которые предлагается использовать для анализа данных и построения модели.

На рисунке 7 приведена часть таблицы входных данных банка В.

CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Рисунок 7 – Входные данные банка В

В таблице 6 представлены признаки банка В по которым строится модель. Входными данными являются признаки числового и категориального типа.

Таблица 6 – Признаки банка В

Описание	***Признак***
Номер строки	RowNumber
Идентификатор клиента	CustomerId
Фамилия	Surname
Кредитный балл	CreditScore
Страна	Geography
Пол	Gender
Возраст	Age
Период кредитования	Tenure
Баланс счета	Balance
Количество продуктов банка	NumOfProducts
Есть ли кредитная карта банка?	HasCrCard
Активный клиент?	IsActiveMember
Предполагаемая зарплата	EstimatedSalary
Клиент не остался (целевой признак)	Exited

Для корректной работы методов потребуется преобразовать категориальные признаки в числовые.

Прежде чем приступать к анализу данных и обучению модели, необходимо проанализировать исходные наборы данных для определения качества выборки и составления прогнозов.

Организация и предварительная подготовка входных данных: для анализа используется проверка на пустые, повторяющиеся и некорректные значения, которые могут создавать шум. В случае присутствия пустых значений удаляется строка таблицы входных данных. Дополнительно проверяются уникальные значения каждого важного признака, чтобы исключить повторения ключевых атрибутов (таблица 7). Таблица позволяет сделать вывод, что пустых значений нет, строки с номерами клиентов не повторяются.

Таблица 7 – Проверка корректности значений признаков банка В

Признак	***Значения***	
	Непустые	***Уникальные***
RowNumber	10000	10000
CustomerId	10000	10000
Surname	10000	2932
CreditScore	10000	460
Geography	10000	3
Gender	10000	2
Age	10000	70
Tenure	10000	11
Balance	10000	6382
NumOfProducts	10000	4
HasCrCard	10000	2
IsActiveMember	10000	2
EstimatedSalary	10000	9999
Exited	10000	2

После проверки корректности данных разделяем выборку на подвыборки. Для банка В 70% данных являются обучающей выборкой и 30% – тестовой.

На этапе построения модели убираем столбцы “*RowNumber*”, “*CustomerId*”, “*Surname*”, так как они не содержат важной информации в плане оттока клиентов (рисунок 8).

Exited	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
1	680	1	0	48	8	115115.38	1	1	0	139558.60
0	525	0	0	25	6	0.00	2	1	0	89566.64
1	705	1	0	37	3	109974.22	1	1	1	36320.87
1	793	0	1	32	2	0.00	2	1	0	193817.63
0	490	0	1	29	4	0.00	2	1	0	32089.57

Рисунок 8 – Обучающая выборка банка B

В дальнейшем используем только нормализированные переменные. Нормализация необходима для перевода признаков в один и тот же масштаб, иначе признак с большим числовым значением может сильно влиять на решение.

Описание и организация выходных данных: в качестве ответов для банка B определяется целевая переменная оттока клиентов “*Exited*” в виде вектора с бинарной принадлежностью к классам “0” или “1”, где “0” – клиент остался, “1” – клиент ушел.

3.2. МОДУЛЬ ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ

Наименование программы: Visualization.exe.

Программное обеспечение, необходимое для функционирования программы: Visual Studio, библиотеки C#.

Язык программирования: в качестве визуализации результатов программы был выбран язык программирования графический интерфейс Windows Forms языка программирования C# на платформе .NET Framework версии 4.6.1. Основным преимуществом выбранного языка является его кроссплатформенность.

Назначение программы: программа визуализирует полученные ранее результаты классификаторов с помощью метрики Accuracy.

Сведения о функциональных ограничениях на применение: наличие доступа в сеть Интернет для скачивания и обновления необходимых компонентов программы.

Способ вызова программы: запуск программы осуществляется исполняемым файлом “Visualization.exe” путем двойного нажатия ЛКМ из директории файла, расположенной на жестком диске.

Алгоритм программы представлен на рисунке 9.

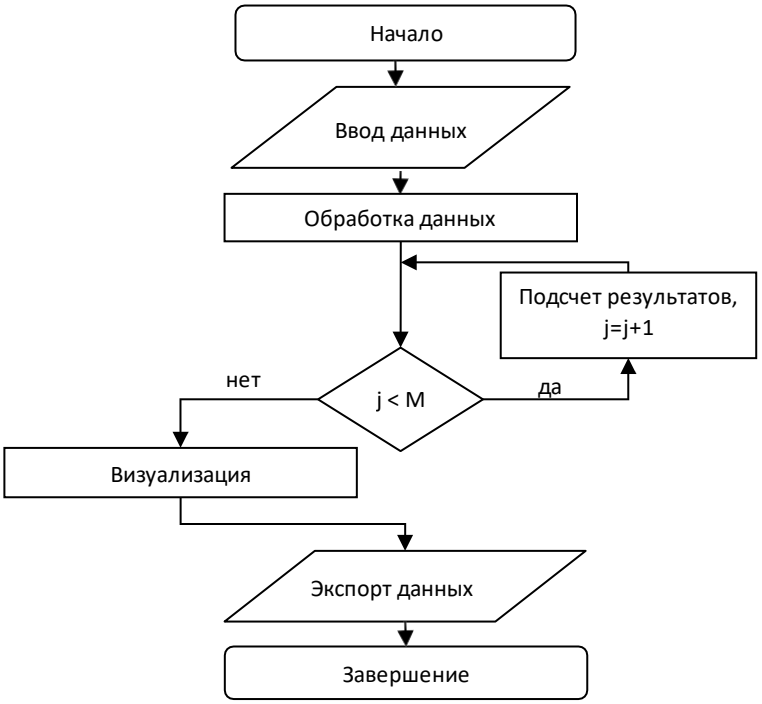

Рисунок 9 – Алгоритм визуализации

Используемые методы: в качестве методов решения выбрано построение метрики точности Accuracy.

Связи программы с другими программами: данная программа связана с программами для построения моделей методами машинного обучения на языке программирования Python.

Описание входных данных: входные данные представляют собой текстовые файлы: *Bank_A_RF.txt*, *Bank_A_CB.txt*, *Bank_A_XGB.txt*, *Bank_B_RF.txt*, *Bank_B_CB.txt* и *Bank_B_XGB.txt*.

Формат: содержание файлов представлено двумя столбцами векторов, слева – заранее известный целевой признак входных данных тестовой выборки, справа – посчитанный алгоритмом. Значения $D_f = \{0, 1\}$.

Описание и организация выходных данных: выходные данные представляют собой матрицу ошибок, диаграмму, время визуализации и метрику точности Accuracy.

4. ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ ОТТОКА КЛИЕНТОВ

4.1. АНАЛИЗ ДАННЫХ

На этапе анализа данных (*Data Mining*) происходит визуализация выборок для более точного представления и проверки корректности значений. Для начала узнаем, сколько клиентов остались или ушли из банков (рисунок 10). На рисунке 10 видно, что в банке А больший процент оттока клиентов.

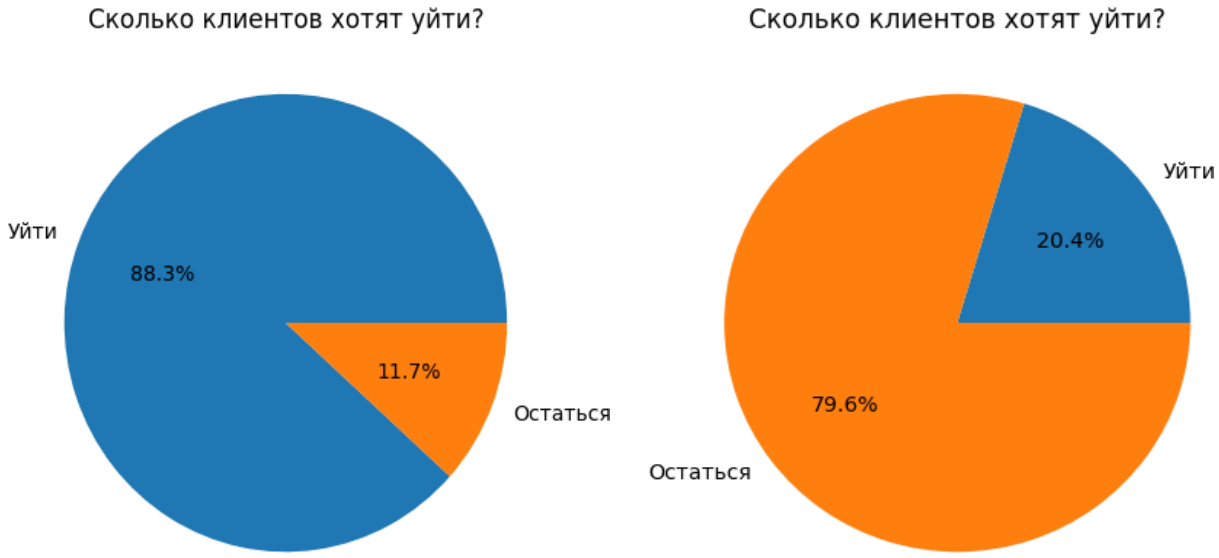

Рисунок 10 – Соотношение оттока клиентов в банках

Данные по банку А содержат в несколько раз больше информации относительно банка В, что позволяет составить более точную модель, несмотря на несбалансированность классов.

Рассмотрим распределение числовых показателей возраста, баланса, кредитного балла и предполагаемой зарплаты для понимания целевой аудитории банков (рисунки 11-12). Отметим тот факт, что многие клиенты банка А имеют близкий к нулевому баланс, а также примерно равный возраст клиентов банков.

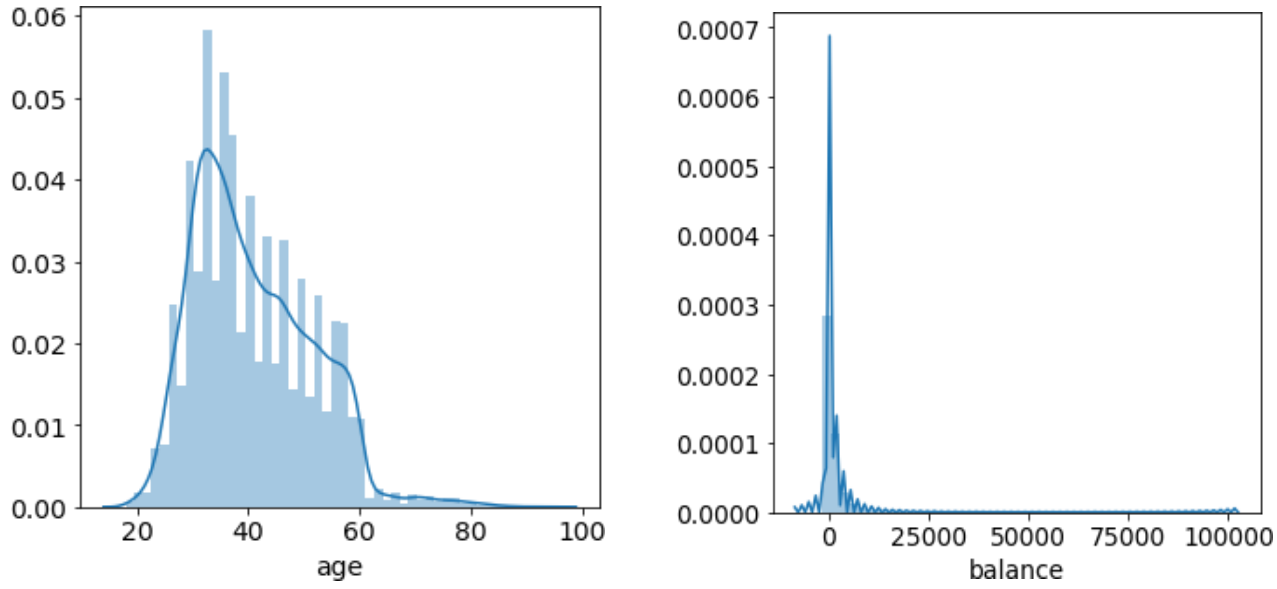

Рисунок 11 – Визуализация основных признаков банка А

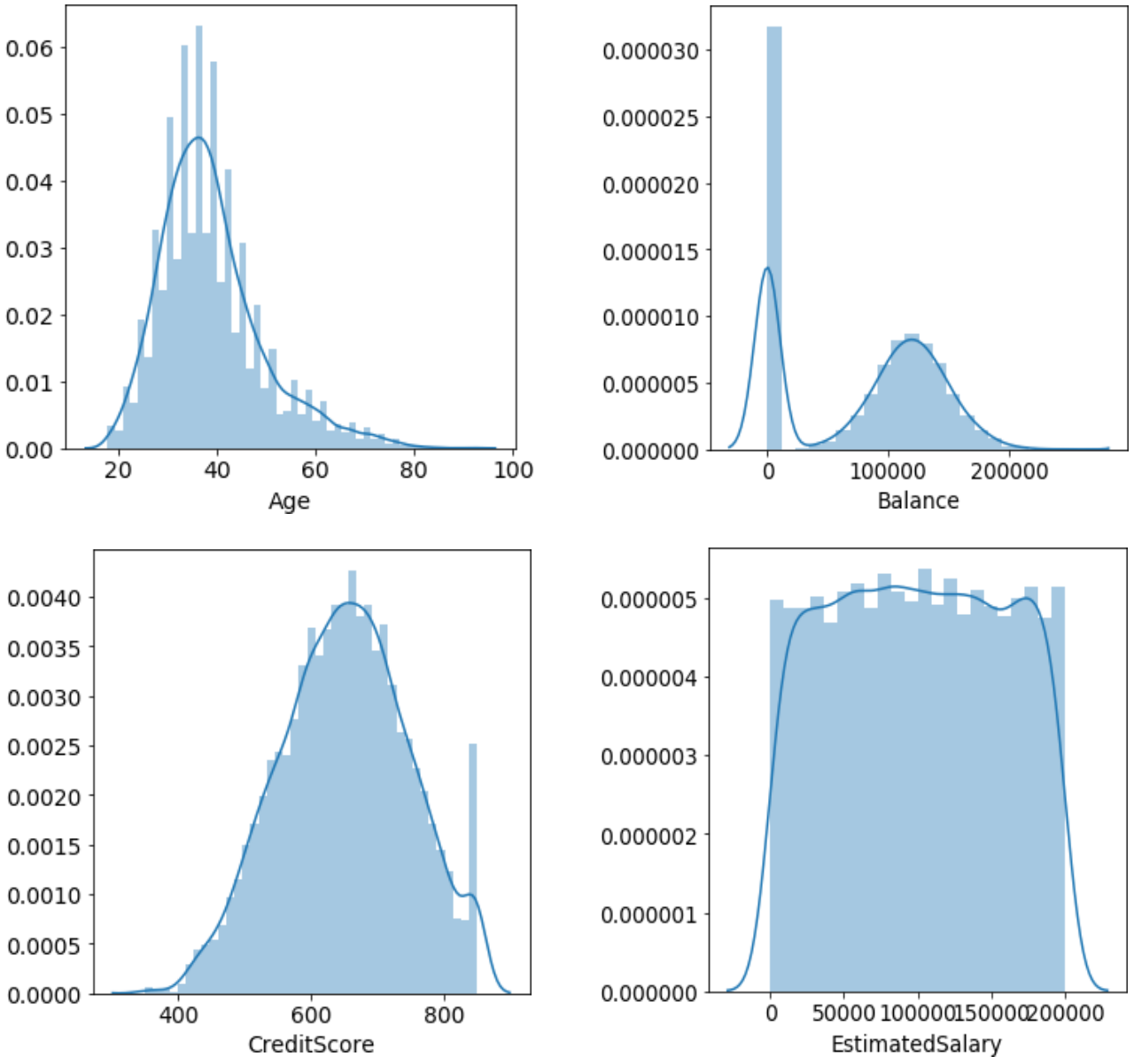

Рисунок 12 - Визуализация основных признаков банка В

На рисунке 13 показана взаимосвязь возраста, семейного положения, образования, времени разговора и баланса клиентов с нулевой и положительной лояльностью для банка А. Рисунок 13 показывает, что не удается выделить определенный сегмент пользователей, склонных к оттоку, ввиду равномерности распределения.

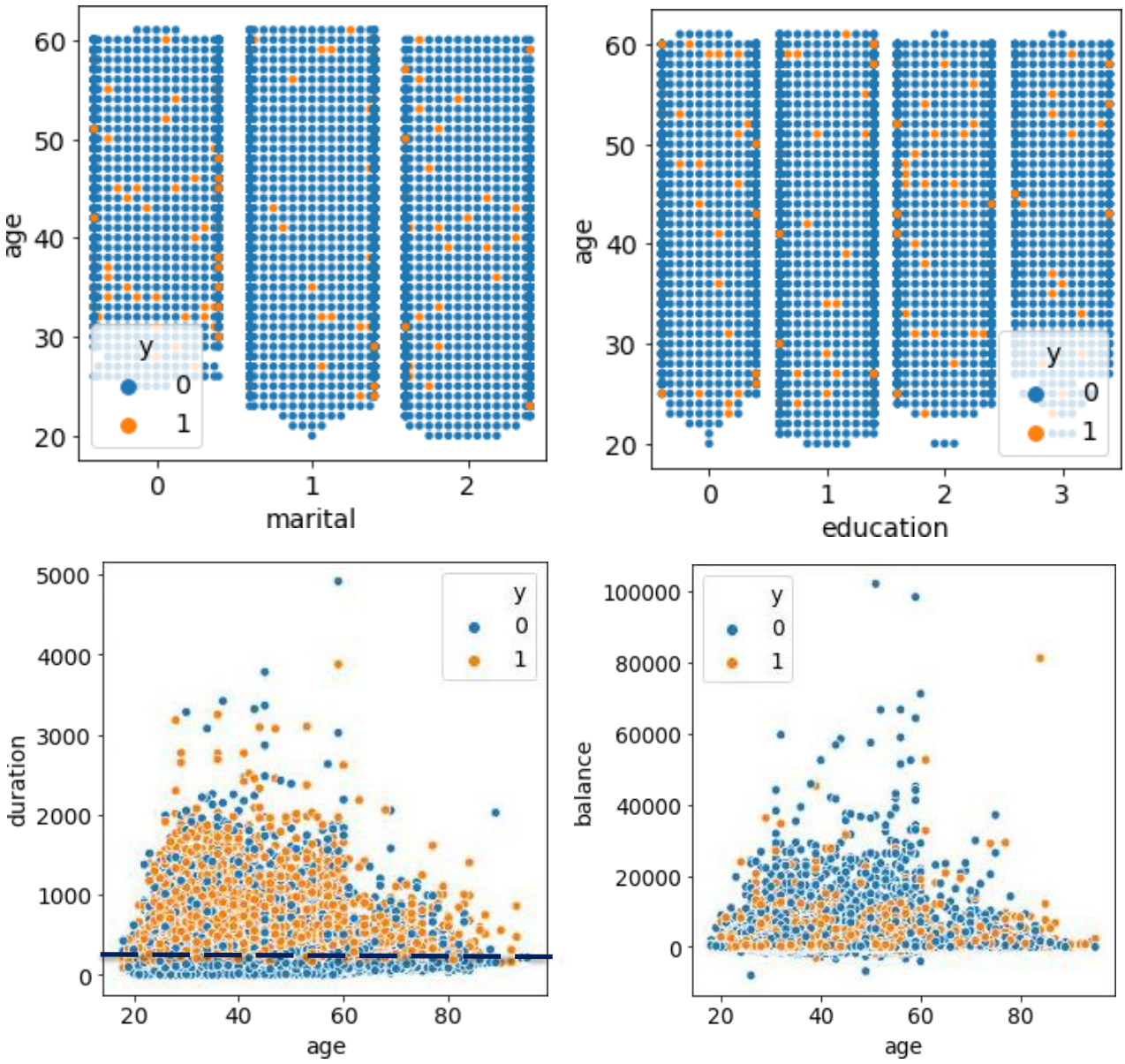

Рисунок 13 – Отток клиентов банка в зависимости от значений признаков

На рисунке 14 показаны распределения числовых данных по признакам: возраст, количество продуктов банка, активность клиента, баланс и кредитный балл для банка В. Можно заметить, что средний возраст клиентов обоих банков 35-40 лет.

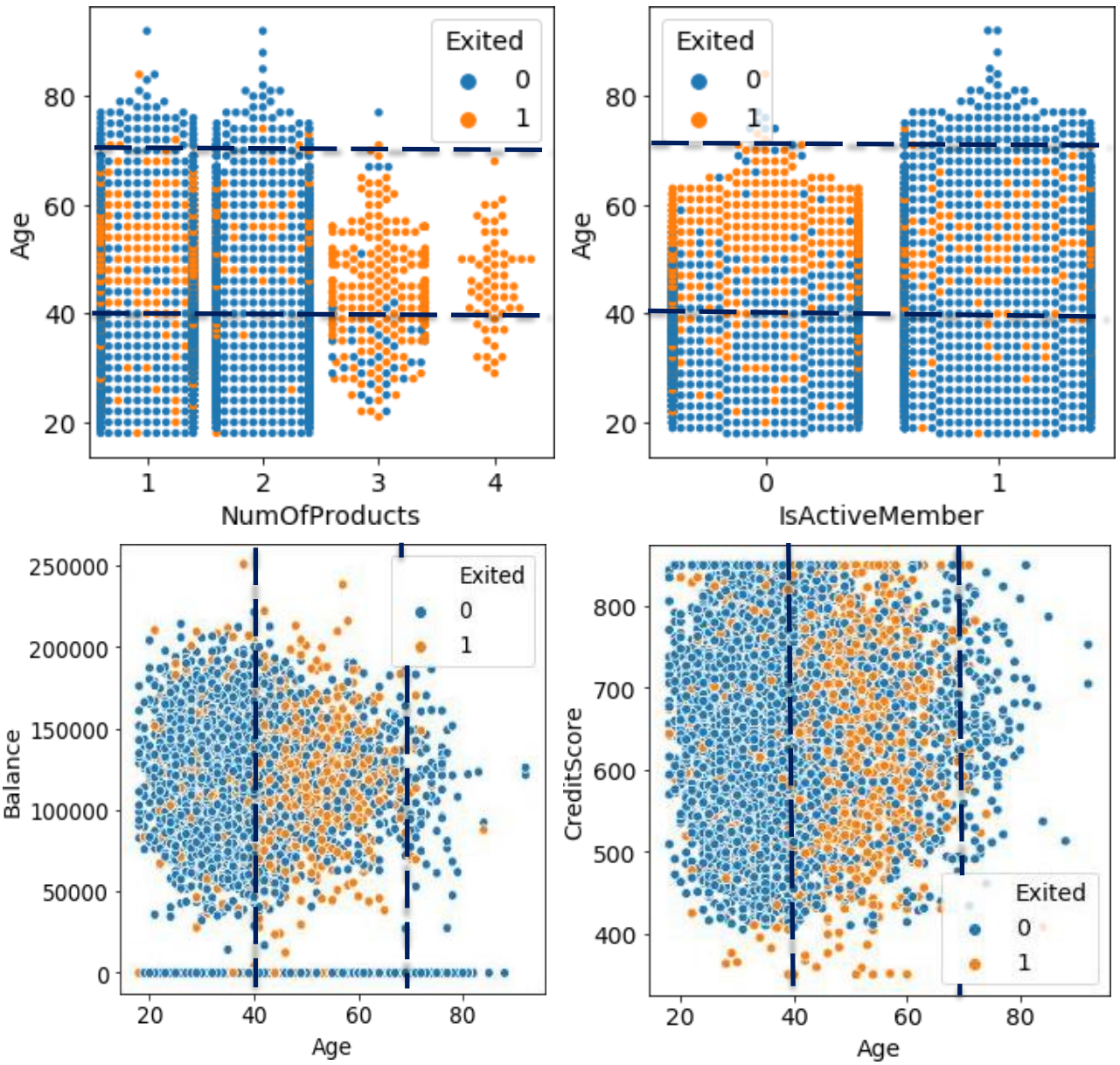

Рисунок 14 – Отток клиентов банка в зависимости от значений признаков

Глядя на рисунки 13-14 можно заметить, что от услуг банка А чаще отказываются клиенты с малым временем разговора, а от услуг банка В – малоактивные клиенты и те, которые пользуются 1, 3 и 4 продуктами банка в возрасте от 40 до 70 лет.

Для более точного определения сегмента оттока клиентов воспользуемся усредненными данными по числовым признакам (таблица 8). В таблице видно, что больше всего различается значение признака *"Duration"*. У действующих клиентов продолжительнее время разговора, клиент был заинтересован и задавал вопросы. При этом у текущих клиентов выше среднегодовой баланс.

Таблица 8 – Усредненные значения признаков банка А

Признак	***Клиент остался***	***Клиент ушел***
	Среднее значение	***Среднее значение***
age	41,67	40,84
job	1,24	1,16
marital	1,24	1,16
education	1,36	1,21
default	0,01	0,02
balance	1804,27	1303,71
housing	0,37	0,58
loan	0,09	0,17
contact	0,27	0,69
day	15,16	15,89
month	5,32	5,55
duration	537,29	221,18
campaign	2,14	2,85
pdays	68,70	36,42
previous	1,17	0,50
poutcome	2,35	2,59

На рисунке 15 изображена двумерная диаграмма средних значений признаков банка A. Диаграмма наглядно показывает основные различия между группами клиентов банка.

При больших различиях порог принятия решения алгоритма увеличивается и позволяет более точно классифицировать целевой признак лояльности клиента банка.

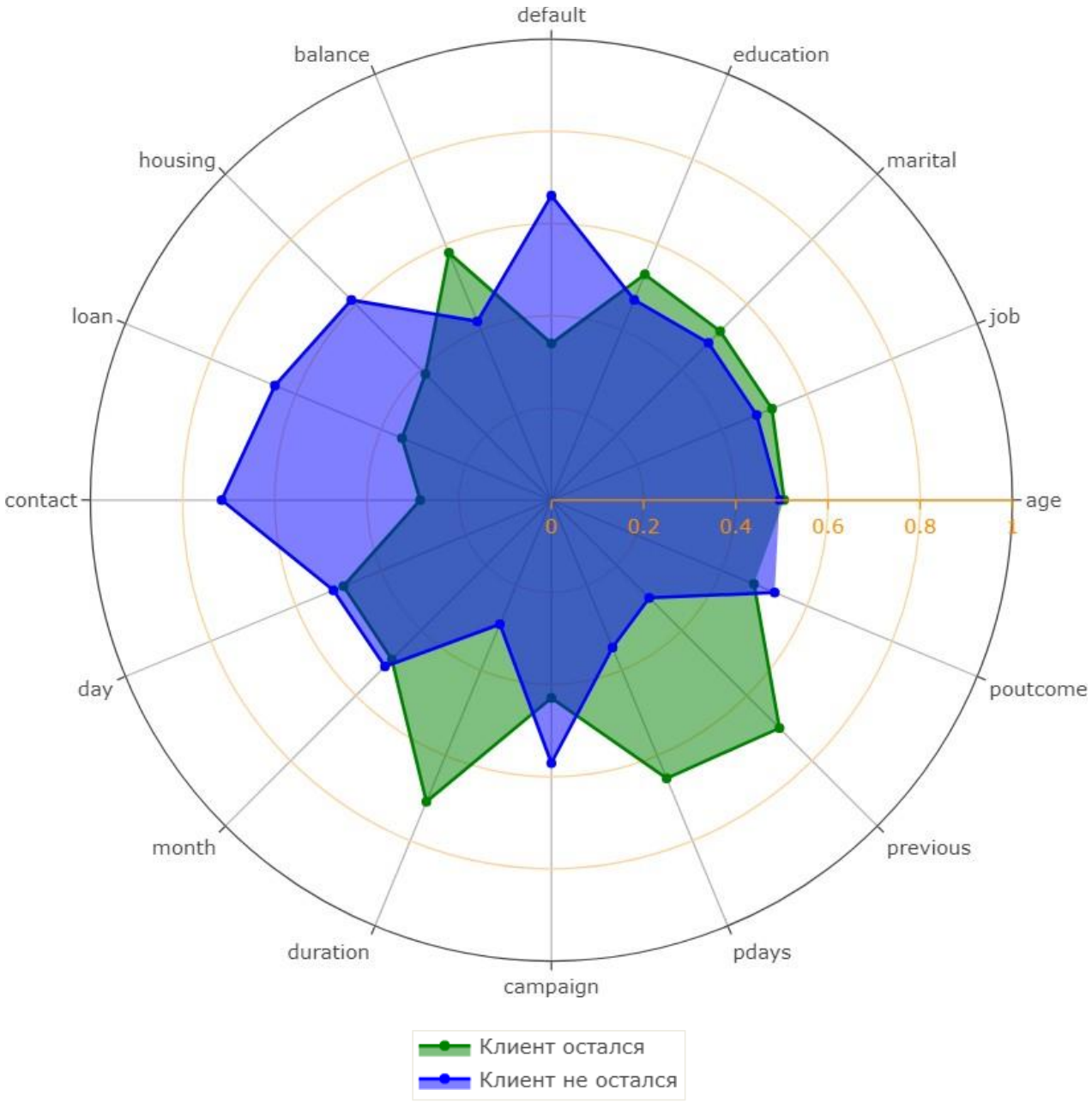

Рисунок 15 – Распределение значений признаков банка A

В таблице 9 рассмотрены минимальные и максимальные значения признаков банка A и проведена нормализация данных. В таблице 10 представлены средние значения признаков банка B. Больше всего различаются признаки *“IsActiveMember”*, *“Gender”*, *“Tenure”*, *“Balance”*. У действующих клиентов чаще статус *“IsActiveMember”*. Клиенты, которые перестали пользоваться продуктами банка имеют более высокий *“Balance”* и *“EstimatedSalary”*.

Таблица 9 – Минимальные и максимальные значения признаков банка А

Признак	***Min***	***Min ϵ [0,1]***	***Max***	***Max ϵ [0,1]***
age	18	0	95	1
job	0	0	2	1
marital	0	0	2	1
education	0	0	3	1
default	0	0	1	1
balance	-8019	0	102127	1
housing	0	0	1	1
loan	0	0	1	1
contact	0	0	2	1
day	1	0	31	1
month	0	0	11	1
duration	0	0	4918	1
campaign	1	0	63	1
pdays	-1	0	871	1
previous	0	0	275	1
poutcome	0	0	3	1

Таблица 10 – Усредненные значения признаков банка В

Признак	***Клиент остался***	***Клиент ушел***
	Среднее значение	***Среднее значение***
CreditScore	651,85	645,35
Geography	0,73	0,81
Gender	0,57	0,44
Age	37,41	44,84
Tenure	5,03	4,93
Balance	72745,30	91108,54
NumOfProducts	1,54	1,48
HasCrCard	0,71	0,70
IsActiveMember	0,55	0,36
EstimatedSalary	99738,39	101465,68

На рисунке 16 показаны средние значения признаков клиентов банка В.

В отличие от диаграммы банка А, в данном случае значения двух классов клиентов имеют меньше различий.

Рисунок 16 – Распределение значений признаков банка B

В таблице 11 рассмотрены минимальные и максимальные значения признаков банка B и проведена нормализация данных.

Таблица 11 – Минимальные и максимальные значения банка B

Признак	***Min***	***Min ϵ [0,1]***	***Max***	***Max ϵ [0,1]***
CreditScore	350	0	850	1
Geography	0	0	2	1
Gender	0	0	1	1
Age	18	0	92	1
Tenure	0	0	10	1
Balance	0	0	250898	1
NumOfProducts	1	0	4	1
HasCrCard	0	0	1	1
IsActiveMember	0	0	1	1
EstimatedSalary	11,58	0	199992	1

Еще один из способов анализа данных – построение матрицы корреляции, которая показывает зависимость признаков друг от друга. На рисунке 17 изображена матрица корреляции банка А. Особенностью матрицы является ее симметричность и единичная главная диагональ.

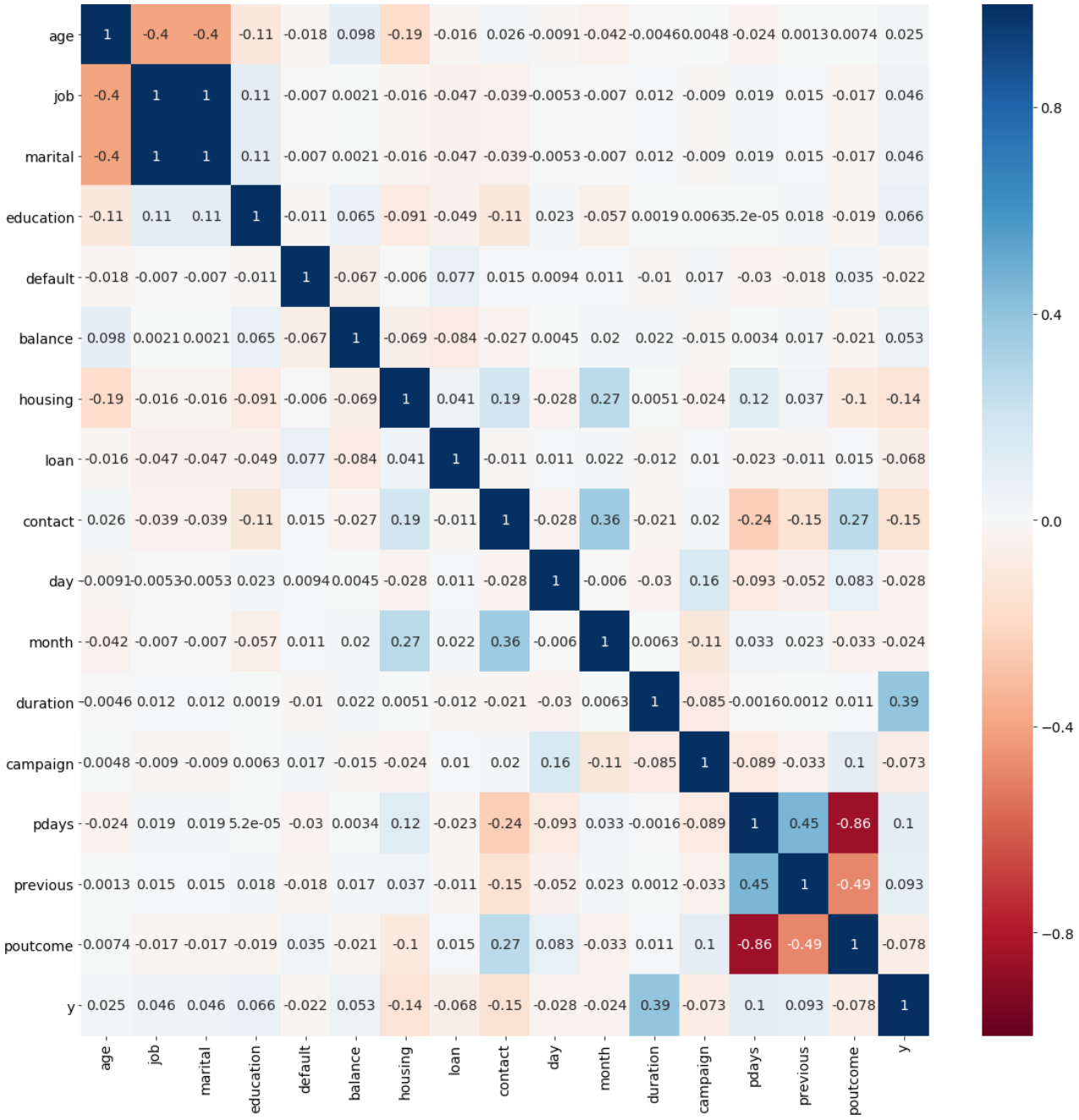

Рисунок 17 – Корреляционная матрица банка А

Матрица корреляции признаков банка В (рисунок 18).

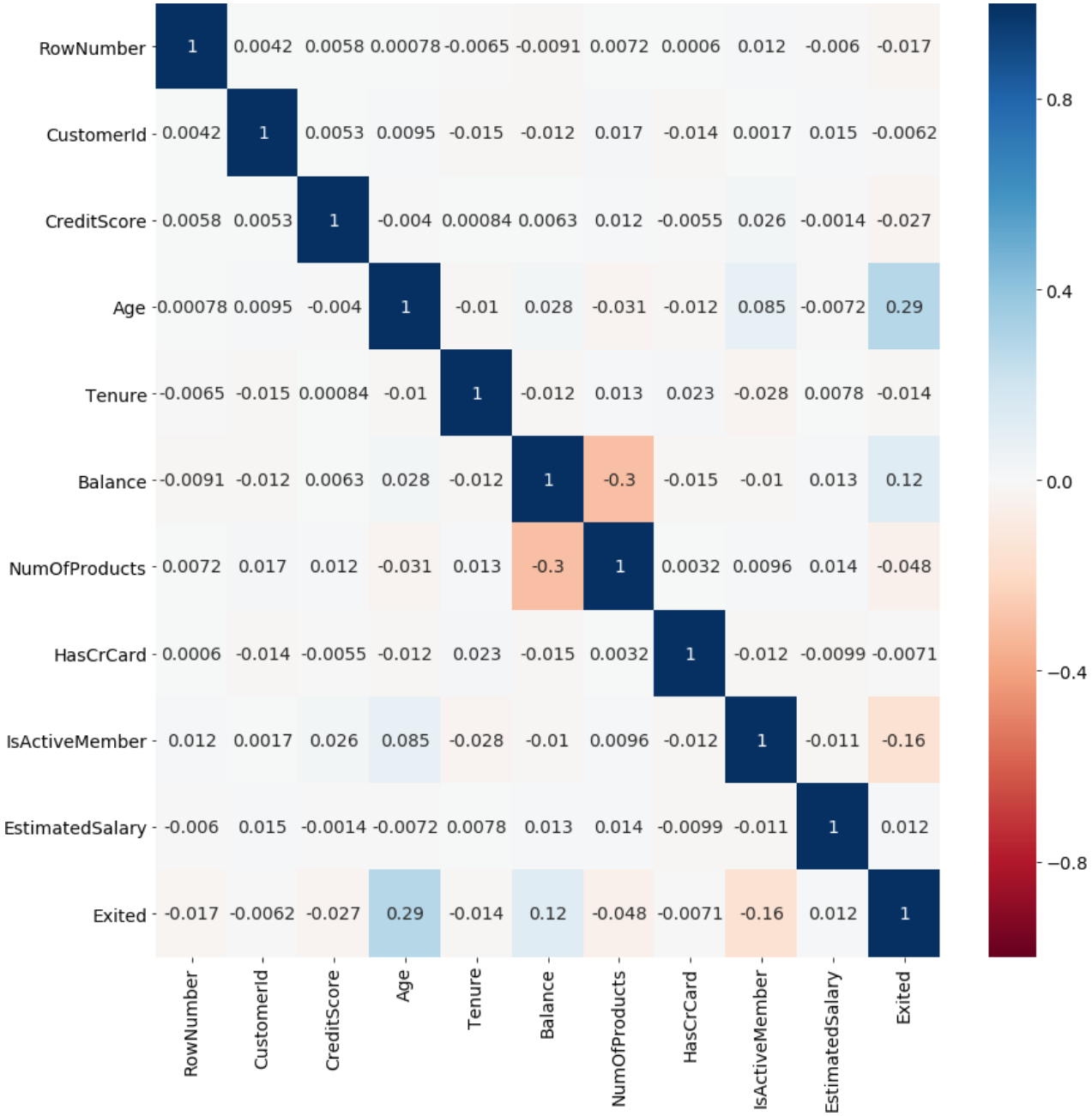

Рисунок 18 – Корреляционная матрица банка В

Для интерпретации результатов приведем таблицу взаимосвязи коэффициента Пирсона и значений признаков (таблица 12).

Таблица 12 – Коэффициент Пирсона

Коэффициент Пирсона	***Описание линейной связи***	X, Y ***– значение признаков***
1	Строгая прямая связь	$X \to X + i, Y \to Y + i, i = const$
0.5	Слабая прямая связь	$X \to X + i, Y \to Y + j, i = const, j = const, i \neq j$
0	Нет связи	$X \to X \pm i, Y \to Y \pm j, i = const, j = const, i \neq j$
-0.5	Слабая обратная связь	$X \to X + i, Y \to Y - j, i = const, j = const, i \neq j$
-1	Строгая обратная связь	$X \to X + i, Y \to Y - i, i = const$

Таким образом, можно сказать, что ближе всего к слабой прямой связи с оттоком клиентов в банке А находится признак продолжительности разговора с сотрудником банка, а в банке В – возраст клиентов [5].

4.2. ОПТИМИЗАЦИЯ ГИПЕРПАРАМЕТРОВ

Для оптимизации и выбора оптимальных локальных параметров алгоритмов используется функция *GridSearchCV* и параметры: *clf* – классификатор, *parameter_grid* – гиперпараметры, *verbose* – лог выполнения, *cv* – кросс-валидация, которая разделяет обучающую и тестовые выборки на части и тестирует каждую из них.

В ходе тестирования были выделены основные параметры и выбраны оптимальные с помощью функции *GridSearchCV*.

Для выбранных алгоритмов наиболее важными гиперпараметрами являются количество деревьев решений и их глубина (таблицы 13-15).

Методу случайного леса требуется больше деревьев и их уровней при увеличении объема данных, в отличие от методов градиентного бустинга.

Таблица 13 – Гиперпараметры Random Forest

Параметр	***Значения***	***Оптимальные значения***	
		Банк А	***Банк В***
Число решающих деревьев	100	500	200
	200		
	500		
Максимальная глубина дерева	8	14	12
	10		
	12		
	14		

Таблица 14 – Гиперпараметры CatBoost

Параметр	***Значения***	***Оптимальные значения***	
		Банк А	***Банк В***
Число решающих деревьев	100	200	200
	200		
	500		
Максимальная глубина дерева	8	8	8
	10		
	12		
	14		

Таблица 15 – Гиперпараметры XGBoost

Параметр	***Значения***	***Оптимальные значения***	
		Банк А	***Банк В***
Число решающих деревьев	100	100	100
	200		
	500		
Максимальная глубина дерева	8	8	8
	10		
	12		
	14		

Рассмотрим работы алгоритмов с точки зрения важности признаков (рисунки 19-21). На рисунках 19-21 видно, что наиболее значимые признаки в банке А – *“Duration”*, в банке В – *“CreditScore”*. Чем важнее признак, тем больший вес он принимает при голосовании решающих деревьев. Важность значений алгоритма Random Forest близка к значениям в матрице корреляции, что позволяет алгоритму работать более точно.

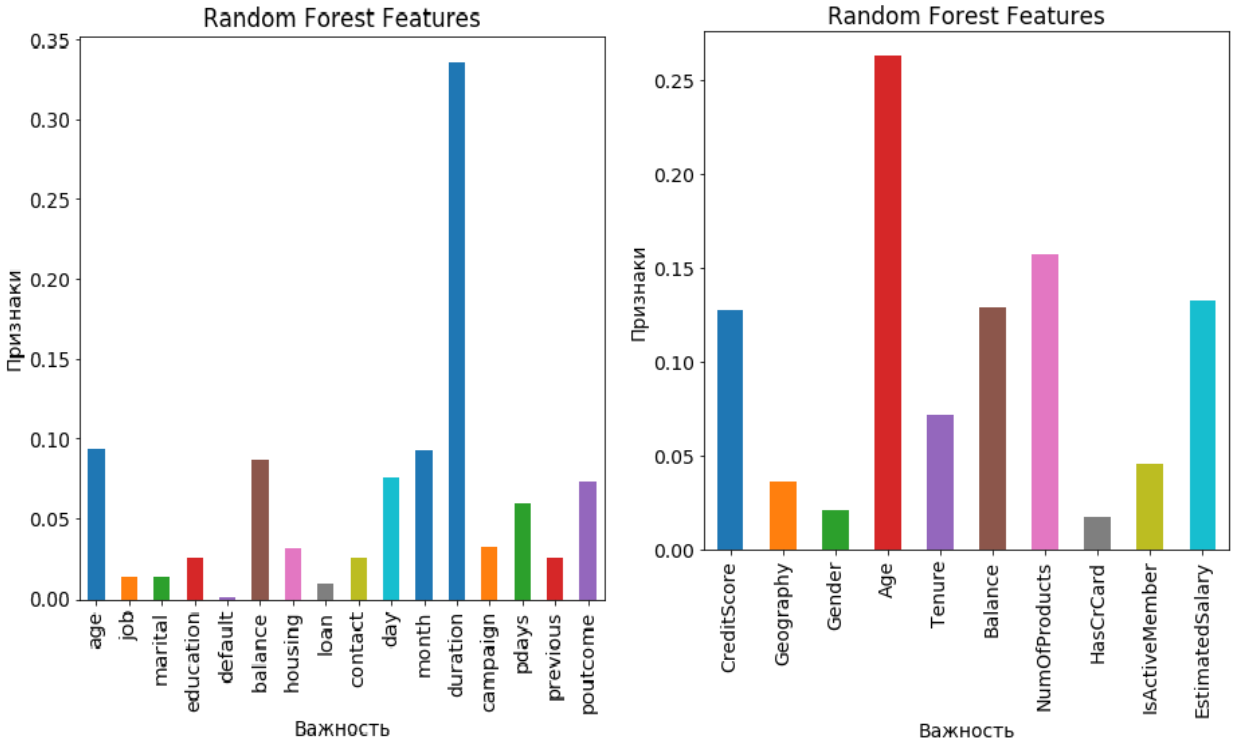

Рисунок 19 – График важности признаков Random Forest

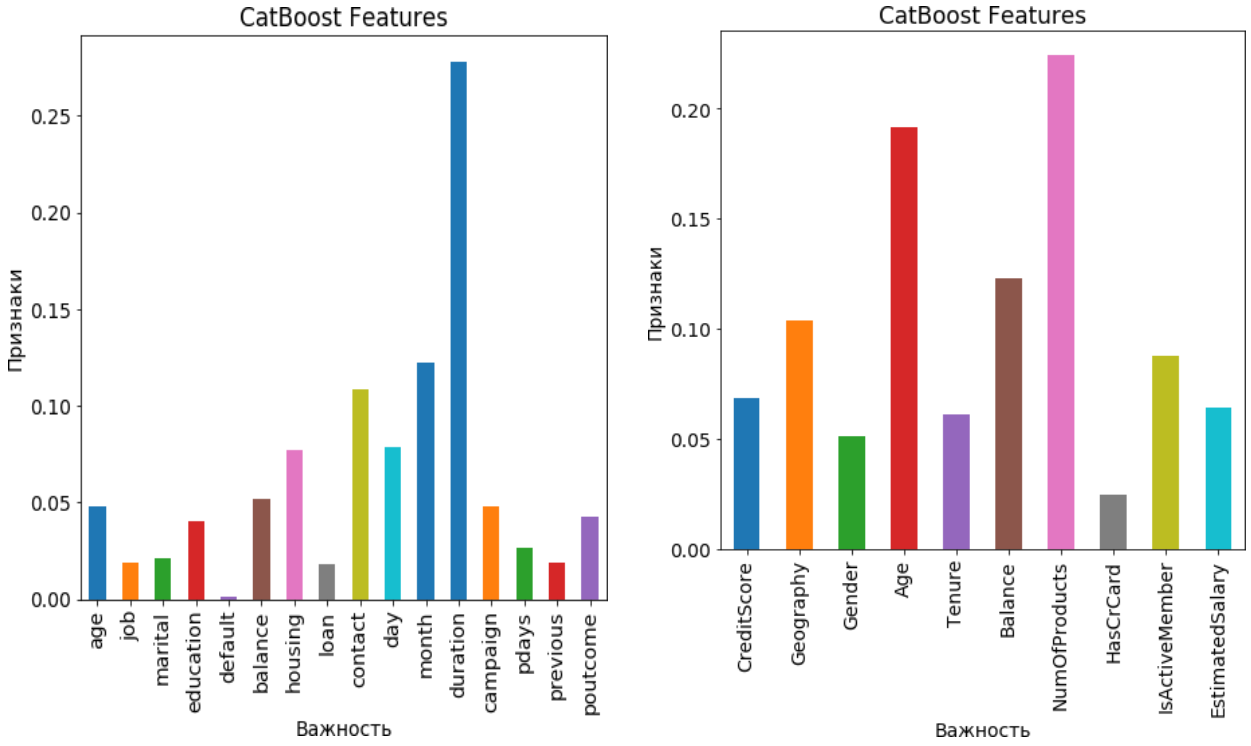

Рисунок 20 – График важности признаков CatBoost

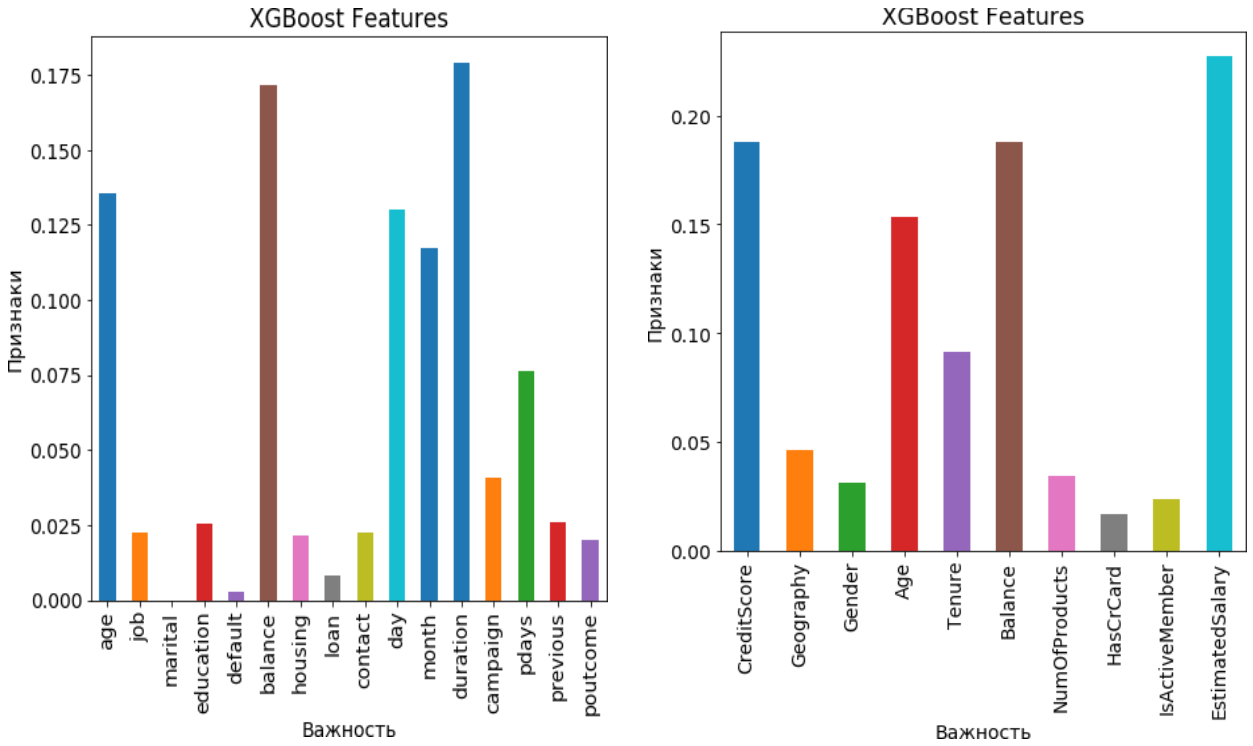

Рисунок 21 – График важности признаков XGBoost

В таблице 16 приведено сравнение времени работы алгоритмов. Как и ожидалось, параллельное построение деревьев решений дает серьезное преимущество по времени относительно последовательного.

Таблица 16 – Сравнение времени работы алгоритмов

Алгоритм	***Время работы, секунд***	
	Банк А	***Банк В***
Random Forest	848	233
CatBoost	2816	826
XGBoost	2150	454

4.3. РЕЗУЛЬТИРУЮЩИЕ МЕТРИКИ

В качестве основных метрик будут использоваться график кривой AUC ROC, а также точность Accuracy в процессе визуализации экспортированных данных модели. Визуальный пример модели можно получить с помощью ROC-кривой, которая позволяет провести оценку качества модели классификатора и качество бинарной классификации.

Построение ROC-кривой: составление сетки из тестовой выборки размера $m \times n$, где m – число единиц, n – число нулей. Движение из точки (0,0) прорисовывая на сетке прямоугольника линии и переходя из одного узла в другой. Если значение метки класса в просматриваемой строке “1”, то делаем шаг вверх; если “0” – вправо. В итоге мы попадём в точку (1, 1), так как сделаем в сумме m шаговвверх и n шагов вправо.

Оптимальному методу соответствует ROC-кривая, проходящая через точку (0, 1), площадь кривой равна единице. Наихудшему – ROC-кривая, проходящая через точку (0.5, 0.5) – диагональ квадрата, площадь которого равна 0.5. Если график проходит через точку (1, 0), то можно реверсировать алгоритм [9].

Формулы расчета TPR и FPR имеют следующий вид:

$$TPR = \frac{TP}{P}, FPR = \frac{FP}{N}, \tag{4.1}$$

где P – количество положительных объектов, N – количество отрицательных объектов [13].

Рисунки 22-23 показывают баланс между максимизацией истинно положительных (TPR) и минимизацией ложноположительных (FPR) результатов и вычисляются по формуле (4.1).

Результаты работы алгоритмов для банка А оказались более точными. Это объяснятся большим количеством данных для обучения модели, а также более четким диапазоном для определения целевого класса.

В таблице 17 приведены параметры для расчета метрики.

В банковской сфере широко применяется метрика качества, называемая коэффициентом Джини, которая показывает отклонение фактического распределения доходов в обществе от их абсолютного равенства между населением и позволяет достаточно точно оценить неравномерность распределения доходов в обществе (рисунок 24).

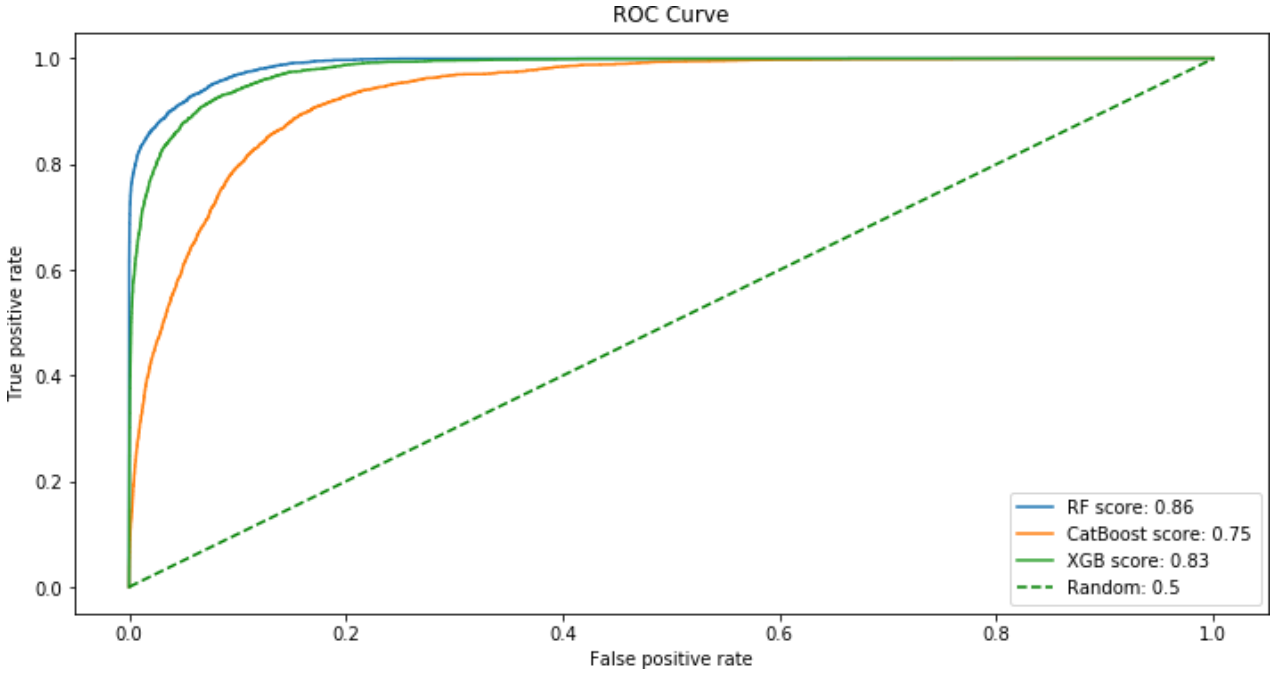

Рисунок 22 – Результаты работы алгоритмов для банка A

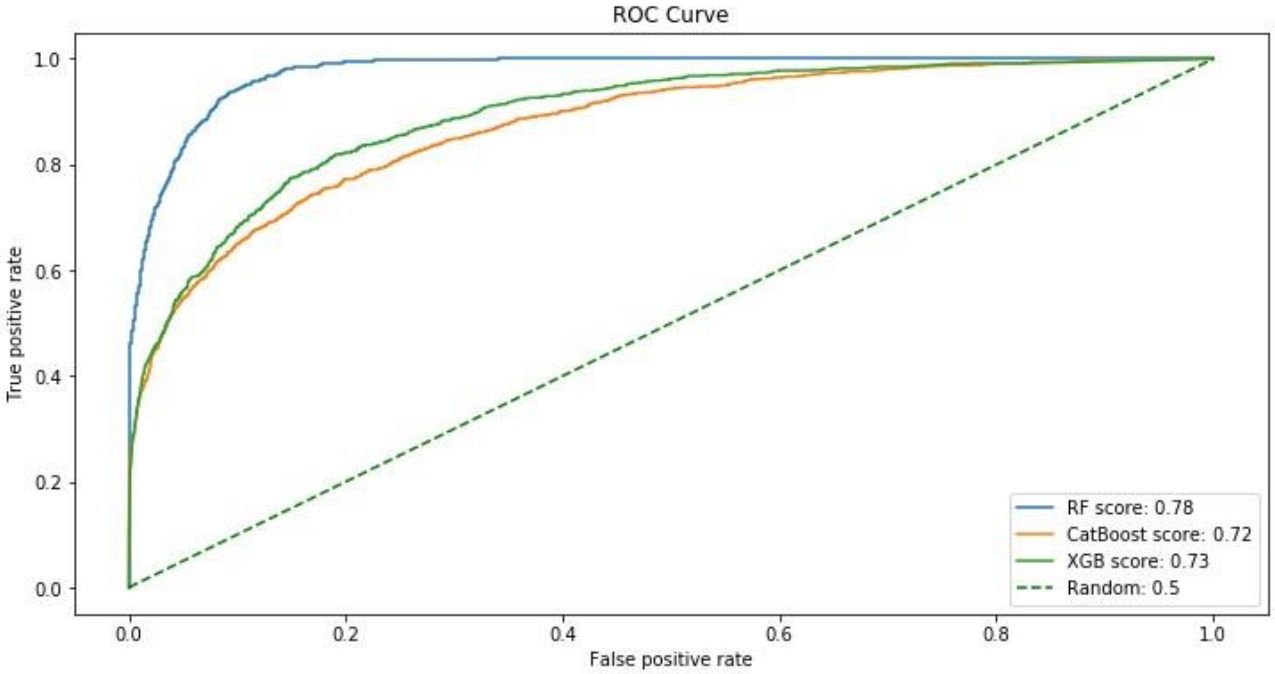

Рисунок 23 – Результаты работы алгоритмов для банка B

Таблица 17 – Параметры метрики

Класс	***Предполагаемый***		
Текущий		P	N
	P	TP (истинно положительные)	FN (ложно отрицательные)
	N	FP (ложно положительный)	TN (истинно отрицательные)

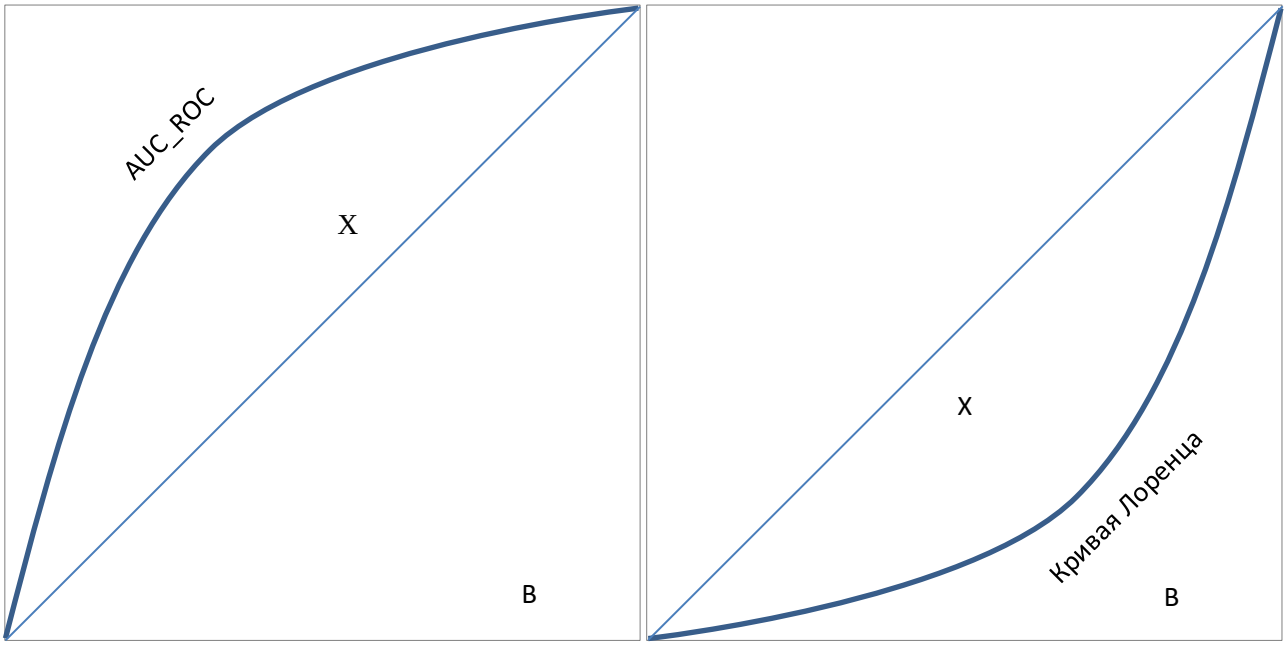

Рисунок 24 – График AUC_ROC и кривая Лоренца

Известно, что кривая Лоренца является графическим отображением коэффициента Gini [2].

Система взаимосвязи коэффициента Gini и кривой Лоренца имеет вид:

$$\begin{cases} Gini = \frac{x}{x+B} \\ B = 0{,}5 \end{cases} <=> \begin{cases} Gini = 2 \cdot x \\ x \neq -0{,}5 \\ B = 0{,}5 \end{cases}, \tag{4.2}$$

где B – половина площади квадрата, x – абсолютное равенство для кривой Лоренца и диагональ квадрата для AUC_ROC.

С другой стороны,

$$x + B = AUC\_ROC => \begin{cases} Gini = \frac{x}{AUC\_ROC} \\ B = 0{,}5 \end{cases}. \tag{4.3}$$

Из формул (4.2) и (4.3) получаем, что:

$$Gini = 2 \cdot x = \frac{x}{AUC\_ROC} => Gini = \begin{cases} 2 = \frac{1}{AUC\_ROC} \\ x \neq 0 \end{cases} <=>$$

$$<=> \begin{cases} Gini = 2 \cdot AUC\_ROC - 1 \\ AUC\_ROC \neq 0 \\ x \neq 0 \end{cases} => Gini = 2 \cdot AUC\_ROC - 1. \tag{4.4}$$

Пользуясь формулой (4.4) найдем коэффициент Джини для алгоритмов решения (таблица 18).

Таблица 18 – Коэффициент Джини

Алгоритм	***Коэффициент Джини***			
	Банк А	***Среднее значение***	***Банк В***	***Среднее значение***
Random Forest	0,72	0,63	0,56	0,49
CatBoost	0,5		0,44	
XGBoost	0,66		0,46	

Среднее значение коэффициента Джини больше в банке А, следовательно, там выше неравномерность распределения доходов в обществе. Это подтверждается распределением баланса на рисунках 11-12.

5. ИССЛЕДОВАНИЕ ТОЧНОСТИ МОДЕЛЕЙ

5.1. ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ МОДЕЛИРОВАНИЯ

Визуальное представление результатов работы позволяет оценить точность полученных моделей.

Элементы представляют собой набор текстовых полей, кнопок, диаграмму сравнения результатов и обрабатываются при различных взаимодействиях пользователя с формой.

В результате визуализации пользователю предлагается посмотреть основные параметры модели в окне формы и сохранить изображение в формате *PNG* для более удобного просмотра диаграммы.

Для удобства работы с программой предусмотрено динамическое изменение размеров окна формы в зависимости от параметров, а также проверка входных данных для начала работы программы либо ее завершения в случае некорректных значений файла.

Метрика для визуализации имеет вид:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (5.1)$$

Зная, что $P = TP + FN,\ N = TN + FP$ по формуле (5.1), получаем:

$$Accuracy = \frac{TP + TN}{P + N}. \quad (5.2)$$

Данная метрика показывает соотношение верного и неверного прогнозов и считается одной из самых важных показателей моделей.

После подсчета результатов происходит вывод полученных данных в элемент типа RichTextBox, который позволяет более наглядно представить значения файла. Также введены пороговые значения, в зависимости от которых определяется цвет элементов Label.

5.1.1. Алгоритм Random Forest

На рисунках 25-26 визуализирована метрика Accuracy, произведен подсчет времени решения задачи и показана матрица ошибок.

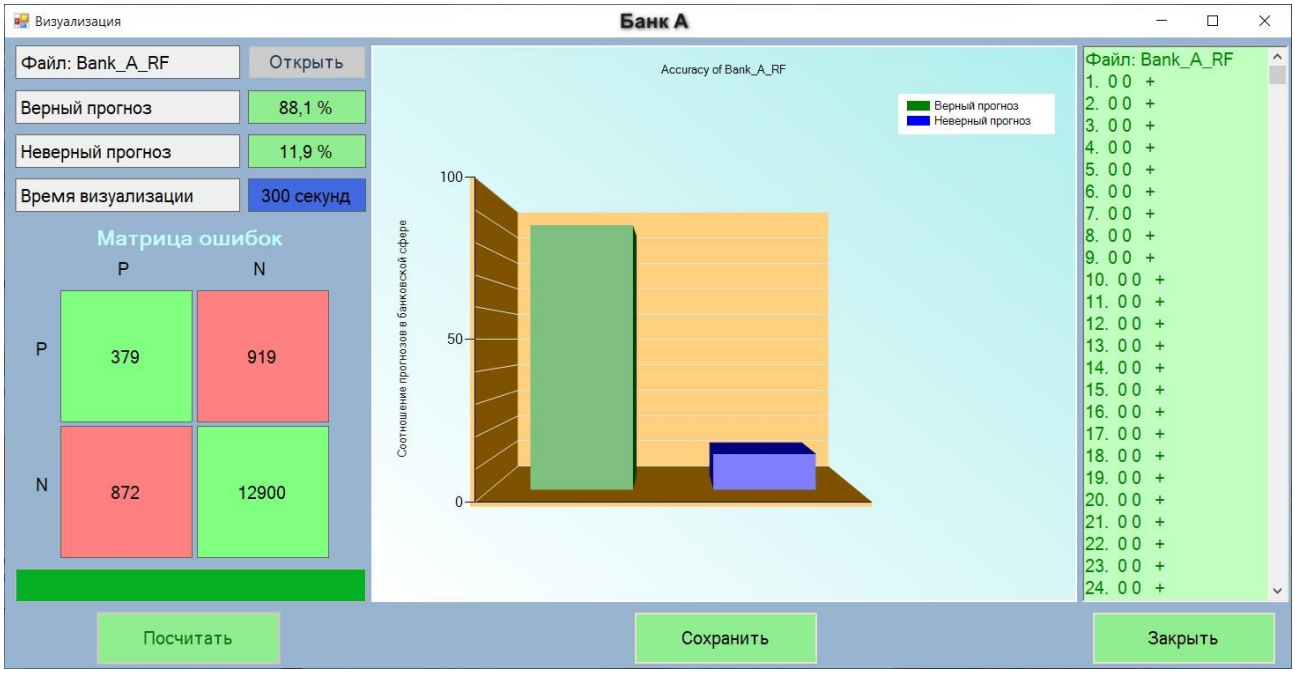

Рисунок 25 – Метрика Accuracy алгоритма Random Forest банка А

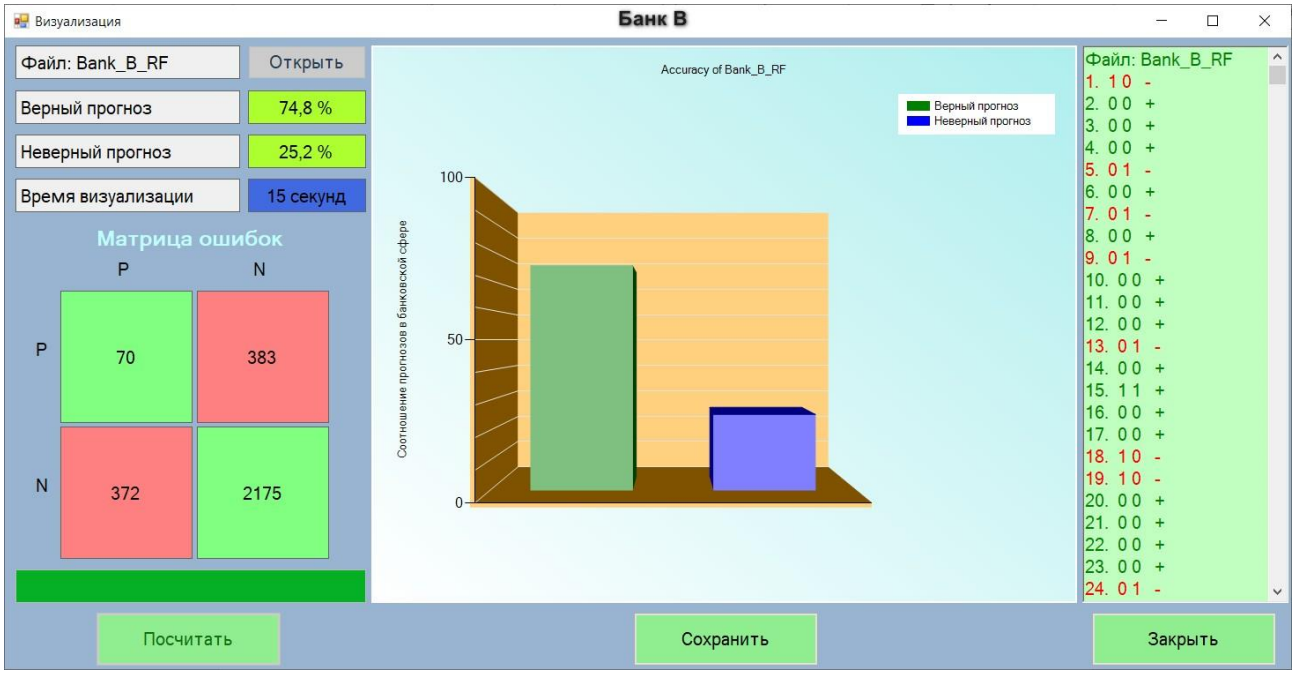

Рисунок 26 – Метрика Accuracy алгоритма Random Forest банка В

5.1.2. Алгоритм XGBoost

На рисунках 27-28 визуализирована метрика Accuracy, произведен подсчет времени решения задачи и показана матрица ошибок.

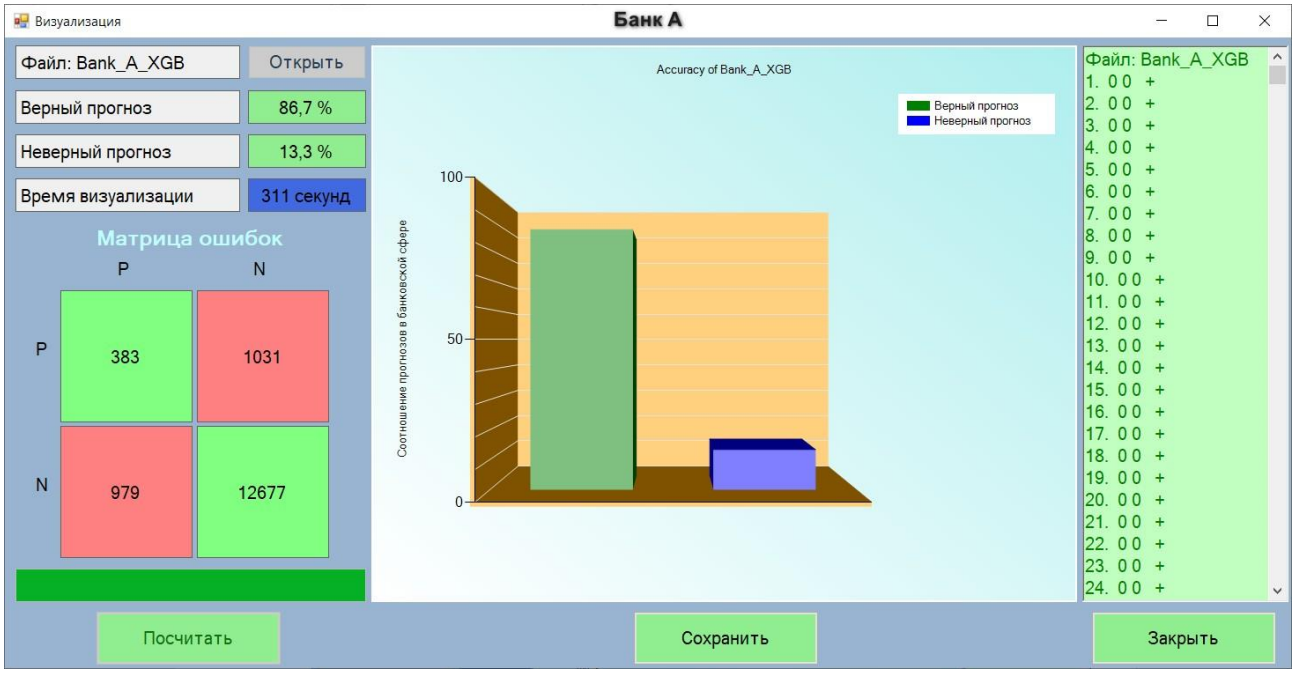

Рисунок 27 – Метрика Accuracy алгоритма XGBoost банка А

Рисунок 28 – Метрика Accuracy для алгоритма XGBoost банка В

5.1.3. Алгоритм CatBoost

На рисунках 29-30 визуализирована метрика Accuracy, произведен подсчет времени решения задачи и показана матрица ошибок.

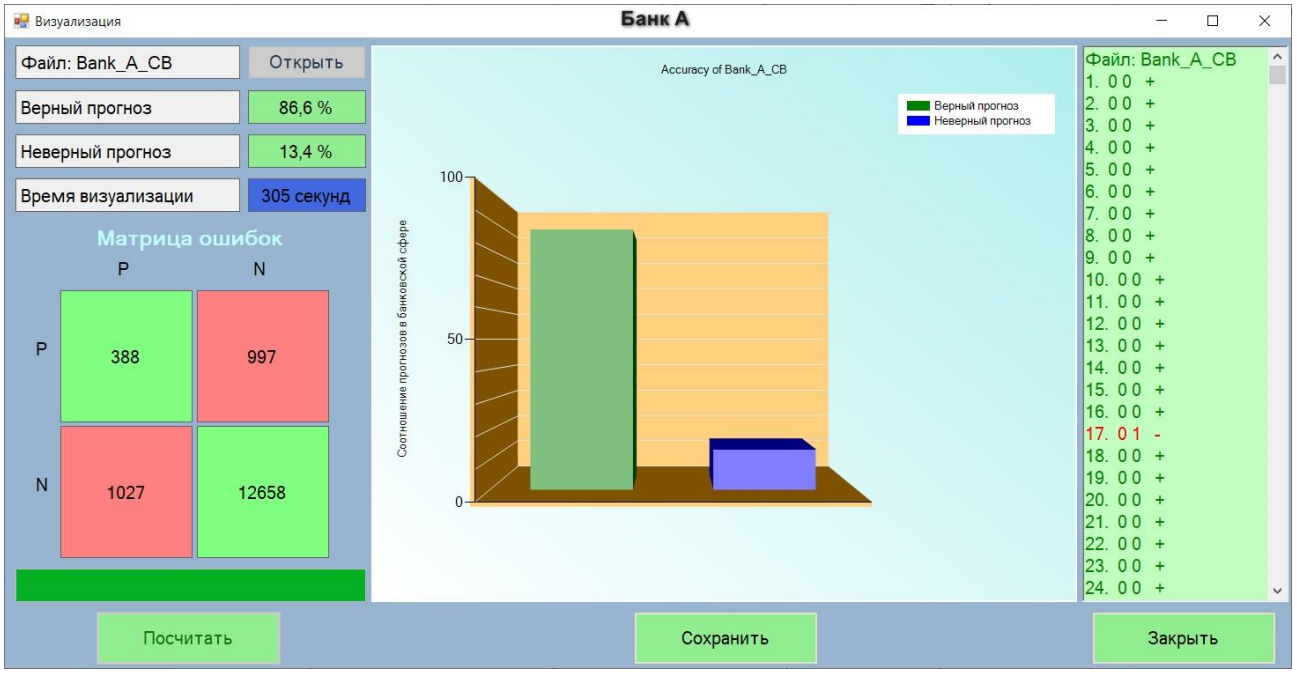

Рисунок 29 – Метрика Accuracy для алгоритма CatBoost банка А

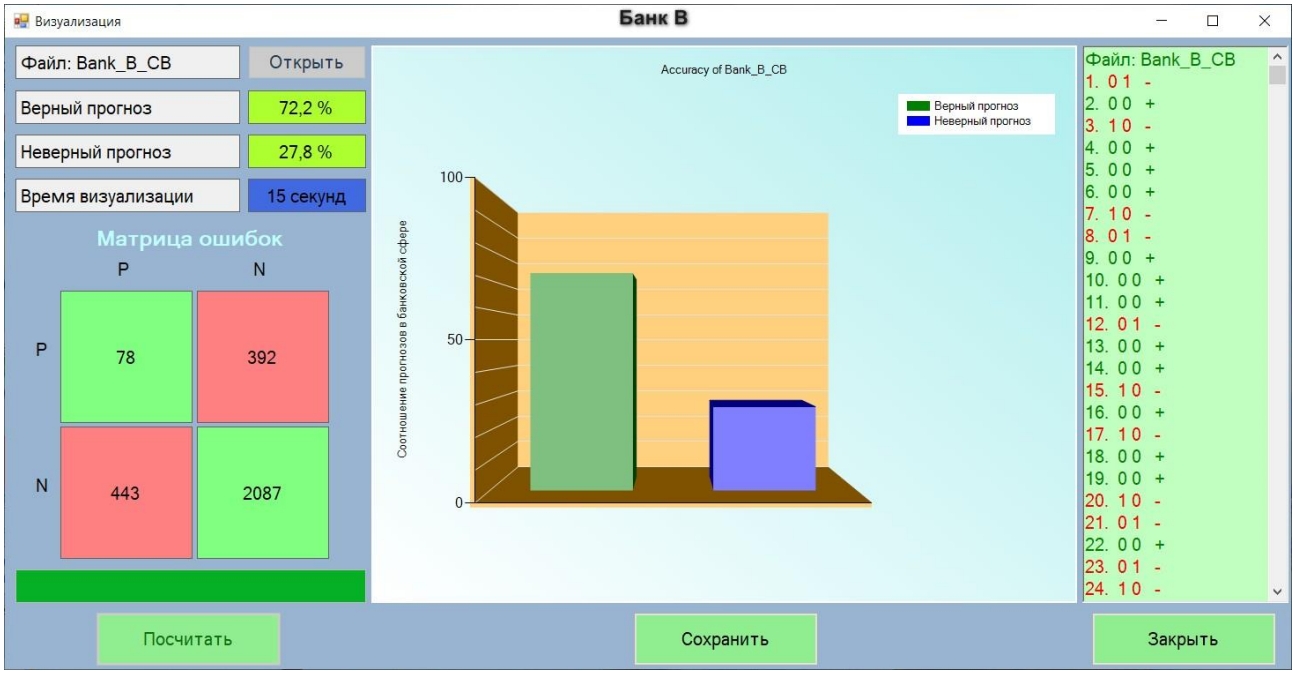

Рисунок 30 – Метрика Accuracy для алгоритма CatBoost банка В

5.2. СРАВНЕНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Визуализация показывает, что более точные результаты для всех рассматриваемых методов имеет банк А (таблица 19).

Время визуализации классификаторов зависит от объема входных данных (таблица 20).

Суммируя значения таблиц 16 и 20, получаем общее время обучения и визуализации моделей (рисунок 31).

Таблица 19 – Результаты метрики Accuracy

	Accuracy, %		
	Random Forest	***XGBoost***	***CatBoost***
Банк А	88,1	86,7	86,6
Банк В	74,8	72,6	72,2

Таблица 20 – Время визуализации алгоритмов

	Время визуализации, секунд		
	Random Forest	***XGBoost***	***CatBoost***
Банк А	300	311	305
Банк В	15	16	15

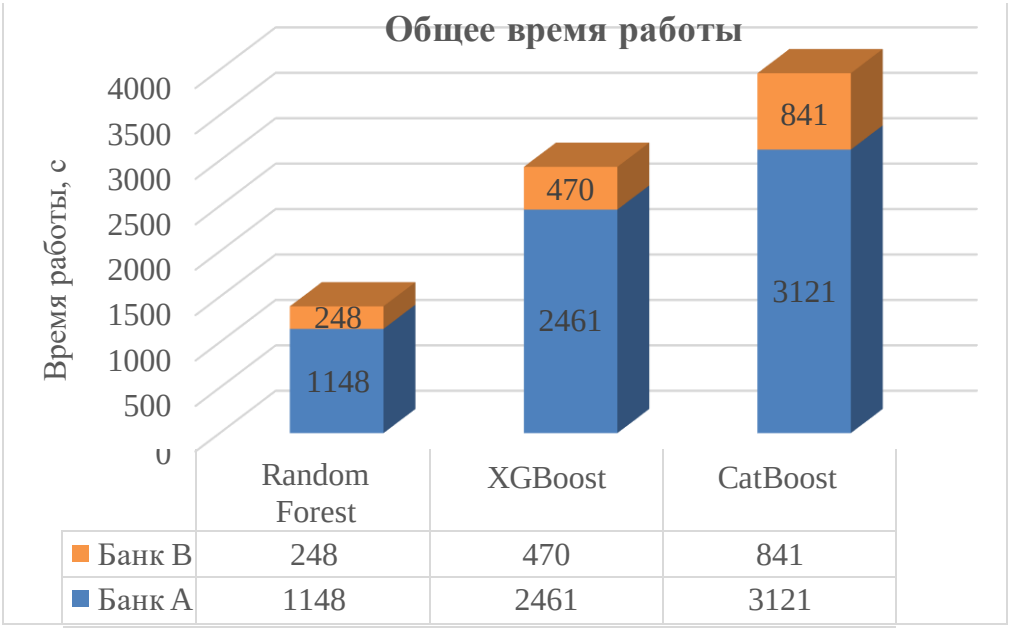

Рисунок 31 – Общее время работы алгоритмов программы

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе были сделаны программные реализации трех эффективных методов классификации машинного обучения для решения задачи прогнозирования оттока клиентов и визуализация полученных моделей.

Точность всех моделей в банке А оказалась выше из-за большей амплитуды средних значений признаков, что позволяет эффективнее разделять сегменты пользователей на классы. В процессе решения задачи выяснили, что существует линейная зависимость некоторых параметров и оттока клиентов банков А и В.

Сравнительный анализ методов показал, что все классификаторы имеют хорошую точность в пределах от 72,2 до 88,1 процентов, что доказывает эффективность методов для решения задачи классификации машинного обучения. Лучшие результаты на тестовой выборке показал метод *Random Forest (88,1%)*, что подтверждает выводы из статьи [12], на втором месте – *XGBoost* и на третьем – классификатор *CatBoost* от компании Яндекс. Также время работы метода *Random Forest* оказалось наименьшим за счет возможности параллельного построения деревьев решений. Таким образом, метод случайного леса можно применять во всех сферах прогнозирования оттока клиентов.

СПИСОК ЛИТЕРАТУРЫ

1. Классификация: [Электронный ресурс] // MachineLearning, 2011. URL: http://www.machinelearning.ru/wiki/index.php?title=Классификация (дата обращения: 27.10.2018).

2. Коэффициент Джини. Из экономики в машинное обучение [Электронный ресурс] // TechMedia, 2019. URL: https://habr.com/ru/company/ods/blog/350440 (дата обращения: 15.12.2018).

3. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес: [Электронный ресурс] // TechMedia, 2017. URL: https://habr.com/ru/company/ods/blog/324402 (дата обращения: 29.04.2019).

4. Открытый курс машинного обучения. Тема 10. Градиентный бустинг: [Электронный ресурс] // TechMedia, 2017. URL: https://habr.com/ru/company/ods/blog/327250 (дата обращения: 29.04.2019).

5. Гласс Дж., Стэнли Дж. Статистические методы в педагогике и психологии.: Прогресс, 1976. – 410 с.

6. Ын Анналин, Су Кеннет. Теоретический минимум по Big Data. Всё, что нужно знать о больших данных. – СПб.: Питер, 2019. – 208 с.

7. KVU Perera, SD Viswakula. A Comparison of Classical Statistical & Machine Learning Techniques in Binary Classification // Department of Statistics, University of Colombo, Sri Lanka. – 2017. – №1. – С. 1-7.

8. Dejana Pavlović, Marija Reljić, Sonja Jaćimović. Application of Data Mining in Direct Marketing in Banking Sector // Ministry of Science and Technological Development of the Republic of Serbia. – 2014. – №1. – С. 189-201.

9. AUC ROC: [Электронный ресурс] // WordPress, 2017. URL: https://dyakonov.org/2017/07/28/auc-roc-площадь-под-кривой-ошибок (дата обращения: 15.12.2018).

10. Bank Marketing Data Set: [Электронный ресурс] // The UCI Machine Learning Repository, 2012. URL: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing (дата обращения: 10.12.2018).

11. CatBoost is a fast, scalable, high performance open-source gradient boosting on decision trees library: [Электронный ресурс] // Yandex, 2019.
URL: https://catboost.ai (дата обращения: 07.03.2019).
12. Manuel Fernández-Delgado, Eva Cernadas, Senén Barro. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? // Journal of Machine Learning Research. – 2014. – №15. – С. 3133-3181.
13. Gini & ROC & Precision-Recall: [Электронный ресурс] // FutureBanking, 2019.
URL: http://futurebanking.ru/post/3761 (дата обращения: 15.12.2018).
14. Predicting Churn for Bank Customers: [Электронный ресурс] // Kaggle Inc, 2018.
URL: https://www.kaggle.com/adammaus/predicting-churn-for-bank-customers (дата обращения: 19.12.2018).
15. L. Breiman, “Random Forests”, Machine Learning, 45(1), 2001. – c. 5-32.
16. TIOBE Index: [Электронный ресурс] // TIOBE Software BV, 2019.
URL: https://www.tiobe.com/tiobe-index (дата обращения: 02.06.2019).

ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ СОЗДАНИЯ МОДЕЛЕЙ БАНКА А

Bank_A.ipynb:

```
# Отключить предупреждения
import warnings
warnings.filterwarnings('ignore')
# Библиотеки
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import plotly
import plotly.graph_objs as go
from plotly.offline import *
from subprocess import call
from sklearn.model_selection import GridSearchCV
%matplotlib inline
from pylab import *
from sklearn.metrics import *
# Классификаторы
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
# Файл df содержит датасет банка A
df = pd.read_csv('Bank-full.csv', delimiter=';')
print (
    "Клиентов: ",df.shape[0],"\n",
    "Атрибутов: ",df.shape[1],"\n")
df.head(5)
# Информация о входных данных, проверка на пустые значения и тип атрибутов,
использовании памяти
print(df.info())
# Количество уникальных значений каждого признака
df.nunique()
# Преобразование категориальных переменных в числовые
le = LabelEncoder()
le.fit(df.marital)
df['job']=le.transform(df.marital)
le.fit(df.marital)
df['marital']=le.transform(df.marital)
le.fit(df.education)
df['education']=le.transform(df.education)
le.fit(df.default)
df['default']=le.transform(df.default)
le.fit(df.housing)
df['housing']=le.transform(df.housing)
le.fit(df.loan)
df['loan']=le.transform(df.loan)
le.fit(df.poutcome)
df['poutcome']=le.transform(df.poutcome)
le.fit(df.contact)
df['contact']=le.transform(df.contact)
le.fit(df.month)
df['month']=le.transform(df.month)
le.fit(df.y)
df['y']=le.transform(df.y)
df.head(5)
# Графики признаков
```

```
plt.rcParams['font.size'] = 14.0
figure, location = plt.subplots(2, 2, figsize=(10, 10))
sns.distplot(df[df.columns[0]], ax=location[0][0])
sns.distplot(df[df.columns[1]], ax=location[0][1])
sns.distplot(df[df.columns[2]], ax=location[1][0])
sns.distplot(df[df.columns[3]], ax=location[1][1])
figure, location = plt.subplots(2, 2, figsize=(10, 10))
sns.distplot(df[df.columns[5]], ax=location[0][0])
sns.distplot(df[df.columns[10]], ax=location[0][1])
sns.distplot(df[df.columns[11]], ax=location[1][0])
sns.distplot(df[df.columns[12]], ax=location[1][1])
# Круговая диаграмма лояльности клиентов банка
figure, location = plt.subplots(figsize=(7, 7))
labels = ['Уйти', 'Остаться']
sizes = [1-df['y'].mean(axis=0), df['y'].mean(axis=0)]
plt.title("Сколько клиентов хотят уйти?")
location.pie(sizes, labels=labels, autopct='%.1f%%')
plt.show()
# Лояльность клиентов в зависимости от признаков
_, ax =  plt.subplots(2, 2, figsize=(10,10))
sns.swarmplot(x = "marital", y = "age", hue="y", sizes = (20, 20), data = df[:],
ax=ax[0][0])
sns.swarmplot(x = "education", y = "age", hue = "y", sizes = (20, 20), data =
df[:], ax=ax[0][1])
sns.scatterplot(x = "age", y = "duration", hue="y",sizes = (20, 20), data = df[:],
ax=ax[1][0])
sns.scatterplot(x = "age", y = "balance", hue = "y", sizes = (20, 20), data =
df[:], ax=ax[1][1])
# Матрица корреляции
plt.subplots(figsize=(19,19))
sns.heatmap(df.corr(method='pearson'), vmin=-1, vmax=1, annot=True, cmap='RdBu')
# Средние значения для клиентов, которые не ушли (1) и ушли (0)
print(round(df[df['y'] == 1].mean(),2), "\n\n",
      round(df[df['y'] == 0].mean(),2))
# Лепестковая диаграмма
data_row = [
    go.Scatterpolar(
      r = [df.age[df['y'] == 1].mean()/(df.age[df['y'] == 1].mean() +
df.age[df['y'] == 0].mean()),
           df.job[df['y'] == 1].mean()/(df.job[df['y'] == 1].mean()+ df.job[df['y']
== 0].mean()),
           df.marital[df['y'] == 1].mean()/(df.marital[df['y'] == 1].mean() +
df.marital[df['y'] == 0].mean()),
           df.education[df['y'] == 1].mean()/(df.education[df['y'] ==
1].mean()+df.education[df['y'] == 0].mean()),
           df.default[df['y'] == 1].mean()/(df.default[df['y'] ==
1].mean()+df.default[df['y'] == 0].mean()),
           df.balance[df['y'] == 1].mean()/(df.balance[df['y'] == 1].mean() +
df.balance[df['y'] == 0].mean()),
           df.housing[df['y'] == 1].mean()/(df.housing[df['y'] == 1].mean() +
df.housing[df['y'] == 0].mean()),
           df.loan[df['y'] == 1].mean()/(df.loan[df['y'] == 1].mean() +
df.loan[df['y'] == 0].mean()),
           df.contact[df['y'] == 1].mean()/(df.contact[df['y'] == 1].mean()+
df.contact[df['y'] == 0].mean()),
           df.day[df['y'] == 1].mean()/(df.day[df['y'] == 1].mean() +
df.day[df['y'] == 0].mean()),
           df.month[df['y'] == 1].mean()/(df.month[df['y'] == 1].mean() +
df.month[df['y'] == 0].mean()),
           df.duration[df['y'] == 1].mean()/(df.duration[df['y'] == 1].mean() +
df.duration[df['y'] == 0].mean()),
           df.campaign[df['y'] == 1].mean()/(df.campaign[df['y'] == 1].mean() +
df.campaign[df['y'] == 0].mean()),
           df.pdays[df['y'] == 1].mean()/(df.pdays[df['y'] == 1].mean() +
df.pdays[df['y'] == 0].mean()),
```

```
            df.previous[df['y'] == 1].mean()/(df.previous[df['y'] == 1].mean() +
df.previous[df['y'] == 0].mean()),
            df.poutcome[df['y'] == 1].mean()/(df.poutcome[df['y'] == 1].mean() +
df.poutcome[df['y'] == 0].mean())],
          theta = [df.columns[0],df.columns[1],df.columns[2],
df.columns[3],df.columns[4],df.columns[5],df.columns[6],df.columns[7],df.columns[8]
,df.columns[9],df.columns[10],df.columns[11],df.columns[12],df.columns[13],df.colum
ns[14],df.columns[15]],
        fill = 'toself',
        name = 'Клиент остался',
          line =  dict(
              color = 'green')),
      go.Scatterpolar(
        r = [df.age[df['y'] == 0].mean()/(df.age[df['y'] == 1].mean() +
df.age[df['y'] == 0].mean()),
            df.job[df['y'] == 0].mean()/(df.job[df['y'] == 1].mean()+ df.job[df['y']
== 0].mean()),
            df.marital[df['y'] == 0].mean()/(df.marital[df['y'] == 1].mean() +
df.marital[df['y'] == 0].mean()),
            df.education[df['y'] == 0].mean()/(df.education[df['y'] ==
1].mean()+df.education[df['y'] == 0].mean()),
            df.default[df['y'] == 0].mean()/(df.default[df['y'] ==
1].mean()+df.default[df['y'] == 0].mean()),
            df.balance[df['y'] == 0].mean()/(df.balance[df['y'] == 1].mean() +
df.balance[df['y'] == 0].mean()),
            df.housing[df['y'] == 0].mean()/(df.housing[df['y'] == 1].mean() +
df.housing[df['y'] == 0].mean()),
            df.loan[df['y'] == 0].mean()/(df.loan[df['y'] == 1].mean() +
df.loan[df['y'] == 0].mean()),
            df.contact[df['y'] == 0].mean()/(df.contact[df['y'] == 1].mean()+
df.contact[df['y'] == 0].mean()),
            df.day[df['y'] == 0].mean()/(df.day[df['y'] == 1].mean() +
df.day[df['y'] == 0].mean()),
            df.month[df['y'] == 0].mean()/(df.month[df['y'] == 1].mean() +
df.month[df['y'] == 0].mean()),
            df.duration[df['y'] == 0].mean()/(df.duration[df['y'] == 1].mean() +
df.duration[df['y'] == 0].mean()),
            df.campaign[df['y'] == 0].mean()/(df.campaign[df['y'] == 1].mean() +
df.campaign[df['y'] == 0].mean()),
            df.pdays[df['y'] == 0].mean()/(df.pdays[df['y'] == 1].mean() +
df.pdays[df['y'] == 0].mean()),
            df.previous[df['y'] == 0].mean()/(df.previous[df['y'] == 1].mean() +
df.previous[df['y'] == 0].mean()),
            df.poutcome[df['y'] == 0].mean()/(df.poutcome[df['y'] == 1].mean() +
df.poutcome[df['y'] == 0].mean())],
        theta = [df.columns[0],df.columns[1],df.columns[2],
df.columns[3],df.columns[4],df.columns[5],df.columns[6],df.columns[7],df.columns[8]
,df.columns[9],df.columns[10],df.columns[11],df.columns[12],df.columns[13],df.colum
ns[14],df.columns[15]],
          fill = 'toself',
        name = 'Клиент не остался',
          line =  dict(
              color = 'blue'))]
layout = go.Layout(
  polar = dict(
    radialaxis = dict(
        color = 'darkorange',
      visible = True,
      range = [0, 1])),
  showlegend = True)
fig = go.Figure(data=data_row, layout=layout)
plotly.offline.plot(fig, filename='Bank_A.html')
# Минимальные и максимальные значения признаков в датасете
print(df.apply(np.min), "\n\n", df.apply(np.max))
# Обучающая и тестовая выборки
```

```
df_train = df.sample(frac=2/3)
df_test = df.drop(df_train.index)
print("Обучающая выборка: ", len(df_train),"\n", "Тестовая выборка: ",len(df_test))
df.head(5)
# Группировка признаков по типу переменной
necateg_p =
['age','job','marital','education','default','balance','housing','loan',
'contact','day','month','duration','campaign','pdays','previous','poutcome']
df_train = df_train[['y'] + necateg_p]
df_train.head(5)
# Нормализация переменных (от 0 до 1)
minVec = df_train[necateg_p].min().copy()
maxVec = df_train[necateg_p].max().copy()
df_train[necateg_p] = (df_train[necateg_p]-minVec)/(maxVec-minVec)
df_train.head(5)
# Функция для проверки данных
def Df_score(df_predict,df_train_Cols,minVec,maxVec):
    # Численные признаки для обучения модели
    necateg_p =
['age','job','marital','education','default','balance','housing','loan',
'contact','day','month','duration','campaign','pdays','previous','poutcome']
    df_predict = df_predict[['y'] + necateg_p]
    L = list(set(df_train_Cols) - set(df_predict.columns))
    for l in L:
        df_predict[str(l)] = -1
    df_predict[necateg_p] = (df_predict[necateg_p]-minVec)/(maxVec-minVec)
    df_predict = df_predict[df_train_Cols]
    return df_predict
# Функция для определения оптимальных параметров и метрики
def best_model(model):
    print(model.best_score_)
    print(model.best_params_)
    print(model.best_estimator_)
def get_auc_scores(y_actual, method,method2):
    auc_score = roc_auc_score(y_actual, method);
    fpr_df, tpr_df, _ = roc_curve(y_actual, method2);
    return (auc_score, fpr_df, tpr_df)
from time import time
# Классификатор Random Forest
parameter_grid = {
            'n_estimators': [100,200,500],
            'max_depth': [8,10,12,14]
        }
classifier1 = RandomForestClassifier() # Классификатор
# Выбор гиперапаметров
grid_searcher = GridSearchCV(classifier1, parameter_grid, verbose=0, cv=5)
start = time()
# Обучение модели
grid_searcher.fit(df_train.loc[:, df_train.columns != 'y'],df_train.y)
finish = time()
classifier1_best = grid_searcher.best_estimator_
print (best_model(grid_searcher))
print("Время работы алгоритма:", round(finish - start), "секунд")
RForest=classifier1_best
y = df_train.y
X = df_train.loc[:, df_train.columns != 'y']
# График важности признаков
features_value = pd.DataFrame({'features': RForest.feature_importances_, 'value':
X.columns})
features_value.plot(kind='bar', x='value', y='features', figsize=(7, 7),
legend=False)
plt.title('Random Forest Features')
plt.ylabel('Признаки')
plt.xlabel('Важность')
plt.show()
```

```
print ("Признаки: ", RForest.feature_importances_, "\n", "Сумма: ",
RForest.feature_importances_.sum())
# Метрики
print(classification_report(df_train.y,  RForest.predict(df_train.loc[:,
df_train.columns != 'y'])))
auc_RForest, fpr_RForest, tpr_RForest = get_auc_scores(y,
RForest.predict(X),RForest.predict_proba(X)[:,1])
# AUC ROC метрика
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RForest, tpr_RForest, label = 'RF score: ' +
str(round(auc_RForest,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Классификатор CatBoost
parameter_grid = {
            'iterations': [100,200,500],
            'depth': [8,10,12,14],
            'verbose': [False],
            }
CTB = CatBoostClassifier(learning_rate=0.03) # Классификатор
# Выбор гиперапаметров
grid_searcher = GridSearchCV(CTB, parameter_grid, cv=5)
start = time()
# Обучение модели
grid_searcher.fit(df_train.loc[:, df_train.columns != 'y'],df_train.y)
finish = time()
CTB_best = grid_searcher.best_estimator_
print (best_model(grid_searcher))
print("Время работы алгоритма: ", round(finish - start), " секунд")
CatBoost=CTB_best
# График важности признаков
features_value = pd.DataFrame({'features': CatBoost.feature_importances_/100,
'value': X.columns}) #Так как от 0 до 1
features_value.plot(kind='bar', x='value', y='features', figsize=(7, 7),
legend=False)
plt.title('CatBoost Features')
plt.ylabel('Признаки')
plt.xlabel('Важность')
plt.show()
print ("Признаки: ", CatBoost.feature_importances_/100, "\n", "Сумма: ",
CatBoost.feature_importances_.sum()/100)
# Метрики
print(classification_report(df_train.y,  CatBoost.predict(df_train.loc[:,
df_train.columns != 'y'])))
auc_CatBoost, fpr_CatBoost, tpr_CatBoost = get_auc_scores(y,
CatBoost.predict(X),CatBoost.predict_proba(X)[:,1])
# AUC ROC метрика
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_CatBoost, tpr_CatBoost, label = 'CatBoost score: ' +
str(round(auc_CatBoost,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Классификатор XGBoost
parameter_grid = {
    'n_estimators':[100,200,500],
    'max_depth': [8,10,12,14],
        }
```

```
XGB = XGBClassifier() # Классификатор
# Выбор гиперапаметров
grid_searcher = GridSearchCV(XGB, parameter_grid, verbose=0, cv=5, refit=True)
start = time()
# Обучение модели
grid_searcher.fit(df_train.loc[:, df_train.columns != 'y'],df_train.y)
finish = time()
XGB_best = grid_searcher.best_estimator_
print (best_model(grid_searcher))
print("Время работы алгоритма: ", round(finish - start), " секунд")
XGboosting=XGB_best
# График важности признаков
features_value = pd.DataFrame({'features': XGboosting.feature_importances_,
'value': X.columns})
features_value.plot(kind='bar', x='value', y='features', figsize=(7, 7),
legend=False)
plt.title('XGBoost Features')
plt.ylabel('Признаки')
plt.xlabel('Важность')
plt.show()
print ("Признаки: ", XGboosting.feature_importances_, "\n", "Сумма: ",
XGboosting.feature_importances_.sum())
# Метрики
print(classification_report(df_train.y,  XGboosting.predict(df_train.loc[:,
df_train.columns != 'y'])))
auc_XGboosting, fpr_XGboosting, tpr_XGboosting = get_auc_scores(y,
XGboosting.predict(X),XGboosting.predict_proba(X)[:,1])
# AUC ROC метрика
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_XGboosting, tpr_XGboosting, label = 'XGB score: ' +
str(round(auc_XGboosting,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Тестовая выборка
df_test = Df_score(df_test,df_train.columns,minVec,maxVec)
df_test = df_test.mask(np.isinf(df_test))
df_test = df_test.dropna()
print ("Клиентов: ",df_test.shape[0],"\n", "Атрибутов: ",df_test.shape[1],"\n")
df_test.head(5)
# Метрики
print(classification_report(df_test.y,  RForest.predict(df_test.loc[:,
df_test.columns != 'y'])))
# AUC ROC метрика
auc_RForest_test, fpr_RForest_test, tpr_RForest_test = get_auc_scores(df_test.y,
RForest.predict(df_test.loc[:, df_test.columns != 'y']),
RForest.predict_proba(df_test.loc[:, df_test.columns != 'y'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RForest_test, tpr_RForest_test, label = 'RF score: ' +
str(round(auc_RForest_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Метрика Accuracy
accuracy_score(df_test.y, RForest.predict(df_test.loc[:, df_test.columns != 'y']))
# Матрица ошибок
confusion_matrix(df_test.y, RForest.predict(df_test.loc[:, df_test.columns !=
'y']))
# Метрики
```

```
print(classification_report(df_test.y, CatBoost.predict(df_test.loc[:,
df_test.columns != 'y'])))
# AUC ROC метрика
auc_CatBoost_test, fpr_CatBoost_test, tpr_CatBoost_test = get_auc_scores(df_test.y,
CatBoost.predict(df_test.loc[:, df_test.columns != 'y']),
CatBoost.predict_proba(df_test.loc[:, df_test.columns != 'y'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_CatBoost_test, tpr_CatBoost_test, label = 'CatBoost score: ' +
str(round(auc_CatBoost_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Метрика Accuracy
accuracy_score(df_test.y, CatBoost.predict(df_test.loc[:, df_test.columns != 'y']))
# Матрица ошибок
confusion_matrix(df_test.y, CatBoost.predict(df_test.loc[:, df_test.columns !=
'y']))
# Метрики
print(classification_report(df_test.y,  XGboosting.predict(df_test.loc[:,
df_test.columns != 'y'])))
# AUC ROC метрика
auc_XGboosting_test, fpr_XGboosting_test, tpr_XGboosting_test =
get_auc_scores(df_test.y, XGboosting.predict(df_test.loc[:, df_test.columns !=
'y']), XGboosting.predict_proba(df_test.loc[:, df_test.columns != 'y'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_XGboosting_test, tpr_XGboosting_test, label = 'XGB score: ' +
str(round(auc_XGboosting_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Метрика Accuracy
accuracy_score(df_test.y, XGboosting.predict(df_test.loc[:, df_test.columns !=
'y']))
# Матрица ошибок
confusion_matrix(df_test.y, XGboosting.predict(df_test.loc[:, df_test.columns !=
'y']))
# Сравнение результатов работы программы
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RForest_test, tpr_RForest_test, label = 'RF score: ' +
str(round(auc_RForest_test,2)))
plt.plot(fpr_CatBoost_test, tpr_CatBoost_test, label = 'CatBoost score: ' +
str(round(auc_CatBoost_test,2)))
plt.plot(fpr_XGboosting_test, tpr_XGboosting_test, label = 'XGB score: ' +
str(round(auc_XGboosting_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Вывод результатов в 2 столбца
def print2 ( *args ):
    num = len(args)
    x = 0
    while x < num:
        try:
            print(str(int(args[x])) + " " + str(int(args[x + 1])))
        except:
            print(str(int(args[x])))
```

```
        x += 2
# Экспорт в txt
import sys
sys.stdout = open('Bank_A_RF.txt', 'w')
print2(*df_test.y[:], *RForest.predict(df_test.loc[:, df_test.columns != 'y'][:]))
sys.stdout.close()
# Экспорт в txt
sys.stdout = open('Bank_A_CB.txt', 'w')
print2(*df_test.y[:], *CatBoost.predict(df_test.loc[:, df_test.columns != 'y'][:]))
sys.stdout.close()
# Экспорт в txt
sys.stdout = open('Bank_A_XGB.txt', 'w')
print2(*df_test.y[:], *XGboosting.predict(df_test.loc[:, df_test.columns !=
'y'][:]))
sys.stdout.close()
```

ПРИЛОЖЕНИЕ Б. ТЕКСТ ПРОГРАММЫ СОЗДАНИЯ МОДЕЛЕЙ БАНКА В

Bank_B.ipynb:

```
# Отключить предупреждения
import warnings
warnings.filterwarnings('ignore')
# Библиотеки
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import plotly
import plotly.graph_objs as go
from plotly.offline import *
from subprocess import call
from sklearn.model_selection import GridSearchCV
%matplotlib inline
from pylab import *
from sklearn.metrics import *
# Классификаторы
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
# Файл df содержит датасет банка B
df = pd.read_csv('Churn_Modelling.csv', delimiter=',')
print (
    "Клиентов: ",df.shape[0],"\n",
    "Атрибутов: ",df.shape[1],"\n")
df.head(5)
# Информация о входных данных, проверка на пустые значения и тип атрибутов,
использовании памяти
print(df.info())
# Количество уникальных значений каждого признака
df.nunique()
# Графики признаков
plt.rcParams['font.size'] = 14.0
figure, location = plt.subplots(2, 2, figsize=(10, 10))
sns.distplot(df[df.columns[3]], ax=location[0][0])
sns.distplot(df[df.columns[6]], ax=location[0][1])
sns.distplot(df[df.columns[7]], ax=location[1][0])
sns.distplot(df[df.columns[8]], ax=location[1][1])
figure, location = plt.subplots(2, 2, figsize=(10, 10))
sns.distplot(df[df.columns[9]], ax=location[0][0])
sns.distplot(df[df.columns[10]], ax=location[0][1])
sns.distplot(df[df.columns[11]], ax=location[1][0])
sns.distplot(df[df.columns[12]], ax=location[1][1])
# Круговая диаграмма лояльности клиентов банка
figure, location = plt.subplots(figsize=(7, 7))
labels = ['Уйти', 'Остаться']
sizes = [df['Exited'].mean(axis=0), 1-df['Exited'].mean(axis=0)]
plt.title("Сколько клиентов хотят уйти?")
location.pie(sizes, labels=labels, autopct='%.1f%%')
plt.show()
# Лояльность клиентов в зависимости от признаков
_, ax =  plt.subplots(2, 2, figsize=(10,10))
sns.swarmplot(x = "NumOfProducts", y = "Age", hue="Exited", sizes = (20, 20), data
= df[:], ax=ax[0][0])
sns.swarmplot(x = "IsActiveMember", y = "Age", hue="Exited", sizes = (20, 20), data
```

```
= df[:], ax=ax[0][1])
sns.scatterplot(x = "Age", y = "Balance", hue = "Exited", sizes = (20, 20), data =
df[:], ax=ax[1][0])
sns.scatterplot(x = "Age", y = "CreditScore", hue = "Exited", sizes = (20, 20),
data = df[:], ax=ax[1][1])
# Матрица корреляции
plt.subplots(figsize=(14,14))
sns.heatmap(df.corr(method='pearson'), vmin=-1, vmax=1, annot=True, cmap='RdBu')
# Преобразование категориальных переменных в числовые
le = LabelEncoder()
le.fit(df.Geography)
df['Geography']=le.transform(df.Geography)
le.fit(df.Gender)
df['Gender']=le.transform(df.Gender)
df.head(5)
# Средние значения для клиентов, которые ушли (1) и не ушли (0)
print(round(df[df['Exited'] == 1].mean(),2), "\n\n",
      round(df[df['Exited'] == 0].mean(),2))
# Лепестковая диаграмма
data_row = [
    go.Scatterpolar(
      r = [df.CreditScore[df['Exited'] == 1].mean()/(df.CreditScore[df['Exited'] ==
1].mean() + df.CreditScore[df['Exited'] == 0].mean()),
           df.Geography[df['Exited'] == 1].mean()/(df.Geography[df['Exited'] ==
1].mean()+ df.Geography[df['Exited'] == 0].mean()),
           df.Gender[df['Exited'] == 1].mean()/(df.Gender[df['Exited'] == 1].mean()
+ df.Gender[df['Exited'] == 0].mean()),
           df.Age[df['Exited'] == 1].mean()/(df.Age[df['Exited'] ==
1].mean()+df.Age[df['Exited'] == 0].mean()),
           df.Tenure[df['Exited'] == 1].mean()/(df.Tenure[df['Exited'] ==
1].mean()+df.Tenure[df['Exited'] == 0].mean()),
           df.Balance[df['Exited'] == 1].mean()/(df.Balance[df['Exited'] ==
1].mean() + df.Balance[df['Exited'] == 0].mean()),
           df.NumOfProducts[df['Exited'] ==
1].mean()/(df.NumOfProducts[df['Exited'] == 1].mean() +
df.NumOfProducts[df['Exited'] == 0].mean()),
           df.HasCrCard[df['Exited'] == 1].mean()/(df.HasCrCard[df['Exited'] ==
1].mean() + df.HasCrCard[df['Exited'] == 0].mean()),
           df.IsActiveMember[df['Exited'] ==
1].mean()/(df.IsActiveMember[df['Exited'] == 1].mean()+
df.IsActiveMember[df['Exited'] == 0].mean()),
           df.EstimatedSalary[df['Exited'] ==
1].mean()/(df.EstimatedSalary[df['Exited'] == 1].mean() +
df.EstimatedSalary[df['Exited'] == 0].mean())],
        theta =
[df.columns[3],df.columns[4],df.columns[5],df.columns[6],df.columns[7],df.columns[8
],df.columns[9],df.columns[10],df.columns[11],df.columns[12]],
        fill = 'toself',
        name = 'Клиент не остался',
        line =  dict(
            color = 'blue')),
    go.Scatterpolar(
      r = [df.CreditScore[df['Exited'] == 0].mean()/(df.CreditScore[df['Exited'] ==
1].mean() + df.CreditScore[df['Exited'] == 0].mean()),
           df.Geography[df['Exited'] == 0].mean()/(df.Geography[df['Exited'] ==
1].mean()+ df.Geography[df['Exited'] == 0].mean()),
           df.Gender[df['Exited'] == 0].mean()/(df.Gender[df['Exited'] == 1].mean()
+ df.Gender[df['Exited'] == 0].mean()),
           df.Age[df['Exited'] == 0].mean()/(df.Age[df['Exited'] ==
1].mean()+df.Age[df['Exited'] == 0].mean()),
           df.Tenure[df['Exited'] == 0].mean()/(df.Tenure[df['Exited'] ==
1].mean()+df.Tenure[df['Exited'] == 0].mean()),
           df.Balance[df['Exited'] == 0].mean()/(df.Balance[df['Exited'] ==
1].mean() + df.Balance[df['Exited'] == 0].mean()),
           df.NumOfProducts[df['Exited'] ==
```

```
0].mean()/(df.NumOfProducts[df['Exited'] == 1].mean() +
df.NumOfProducts[df['Exited'] == 0].mean()),
           df.HasCrCard[df['Exited'] == 0].mean()/(df.HasCrCard[df['Exited'] ==
1].mean() + df.HasCrCard[df['Exited'] == 0].mean()),
           df.IsActiveMember[df['Exited'] ==
0].mean()/(df.IsActiveMember[df['Exited'] == 1].mean()+
df.IsActiveMember[df['Exited'] == 0].mean()),
           df.EstimatedSalary[df['Exited'] ==
0].mean()/(df.EstimatedSalary[df['Exited'] == 1].mean() +
df.EstimatedSalary[df['Exited'] == 0].mean())],
        theta =
[df.columns[3],df.columns[4],df.columns[5],df.columns[6],df.columns[7],df.columns[8
],df.columns[9],df.columns[10],df.columns[11],df.columns[12]],
        fill = 'toself',
        name = 'Клиент остался',
        line =  dict(
            color = 'green'))]
layout = go.Layout(
  polar = dict(
    radialaxis = dict(
        color = 'darkorange',
      visible = True,
      range = [0, 1])),
  showlegend = True)
fig = go.Figure(data=data_row, layout=layout)
plotly.offline.plot(fig, filename='Bank_B.html')
# Минимальные и максимальные значения признаков в датасете
print(df.apply(np.min), "\n\n", df.apply(np.max))
# Удалить столбцы
df = df.drop(columns=["RowNumber", "CustomerId", "Surname"], axis = 1)
# Обучающая и тестовая выборки
df_train = df.sample(frac=0.7)
df_test = df.drop(df_train.index)
print("Обучающая выборка: ", len(df_train),"\n", "Тестовая выборка: ",len(df_test))
df.head(5)
# Группировка признаков по типу переменной
necateg_p =
['CreditScore','Geography','Gender','Age','Tenure','Balance','NumOfProducts',
'HasCrCard','IsActiveMember','EstimatedSalary']
df_train = df_train[['Exited'] + necateg_p]
df_train.head(5)
# Нормализация переменных (от 0 до 1)
minVec = df_train[necateg_p].min().copy()
maxVec = df_train[necateg_p].max().copy()
df_train[necateg_p] = (df_train[necateg_p]-minVec)/(maxVec-minVec)
df_train.head(5)
# Функция для проверки данных
def Df_score(df_predict,df_train_Cols,minVec,maxVec):
    # Численные признаки для обучения модели
    necateg_p =
['CreditScore','Geography','Gender','Age','Tenure','Balance','NumOfProducts',
'HasCrCard','IsActiveMember','EstimatedSalary']
    df_predict = df_predict[['Exited'] + necateg_p]
    L = list(set(df_train_Cols) - set(df_predict.columns))
    for l in L:
        df_predict[str(l)] = -1
    df_predict[necateg_p] = (df_predict[necateg_p]-minVec)/(maxVec-minVec)
    df_predict = df_predict[df_train_Cols]
    return df_predict
# Функция для определения оптимальных параметров и метрики
def best_model(model):
    print(model.best_score_)
    print(model.best_params_)
    print(model.best_estimator_)
def get_auc_scores(y_actual, method,method2):
```

```
    auc_score = roc_auc_score(y_actual, method);
    fpr_df, tpr_df, _ = roc_curve(y_actual, method2);
    return (auc_score, fpr_df, tpr_df)
from time import time
# Классификатор Random Forest
parameter_grid = {
            'n_estimators': [100,200,500],
            'max_depth': [8,10,12,14]
        }
clf = RandomForestClassifier() # Классификатор
# Выбор гиперапаметров
grid_searcher = GridSearchCV(clf, parameter_grid, verbose=0, cv=5)
start = time()
# Обучение модели
grid_searcher.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
finish = time()
clf_best = grid_searcher.best_estimator_
print (best_model(grid_searcher))
print("Время работы алгоритма: ", round(finish - start), " секунд")
RForest=clf_best
y = df_train.Exited
X = df_train.loc[:, df_train.columns != 'Exited']
# График важности признаков
features_value = pd.DataFrame({'features': RForest.feature_importances_, 'value':
X.columns})
features_value.plot(kind='bar', x='value', y='features', figsize=(7, 7),
legend=False)
plt.title('Random Forest Features')
plt.ylabel('Признаки')
plt.xlabel('Важность')
plt.show()
print ("Признаки: ", RForest.feature_importances_, "\n", "Сумма: ",
RForest.feature_importances_.sum())
# Метрики
print(classification_report(df_train.Exited,  RForest.predict(df_train.loc[:,
df_train.columns != 'Exited'])))
auc_RForest, fpr_RForest, tpr_RForest = get_auc_scores(y,
RForest.predict(X),RForest.predict_proba(X)[:,1])
# AUC ROC метрика
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RForest, tpr_RForest, label = 'RF score: ' +
str(round(auc_RForest,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Классификатор CatBoost
parameter_grid = {
            'iterations': [100,200,500],
            'depth': [8,10,12,14],
            'verbose': [False],
            }
CTB = CatBoostClassifier(learning_rate=0.03) # Классификатор
# Выбор гиперапаметров
grid_searcher = GridSearchCV(CTB, parameter_grid, cv=5)
start = time()
# Обучение модели
grid_searcher.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
finish = time()
CTB_best = grid_searcher.best_estimator_
print (best_model(grid_searcher))
print("Время работы алгоритма: ", round(finish - start), " секунд")
CatBoost=CTB_best
```

```
# График важности признаков
features_value = pd.DataFrame({'features': CatBoost.feature_importances_/100,
'value': X.columns}) # Так как от 0 до 1
features_value.plot(kind='bar', x='value', y='features', figsize=(7, 7),
legend=False)
plt.title('CatBoost Features')
plt.ylabel('Признаки')
plt.xlabel('Важность')
plt.show()
print ("Признаки: ", CatBoost.feature_importances_/100, "\n", "Сумма: ",
CatBoost.feature_importances_.sum()/100)
# Метрики
print(classification_report(df_train.Exited,  CatBoost.predict(df_train.loc[:,
df_train.columns != 'Exited'])))
auc_CatBoost, fpr_CatBoost, tpr_CatBoost = get_auc_scores(y,
CatBoost.predict(X),CatBoost.predict_proba(X)[:,1])
# AUC ROC метрика
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_CatBoost, tpr_CatBoost, label = 'CatBoost score: ' +
str(round(auc_CatBoost,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Классификатор XGBoost
parameter_grid = {
    'n_estimators':[100,200,500],
    'max_depth': [8,10,12,14],
        }
XGB = XGBClassifier() # Классификатор
# Выбор гиперапаметров
grid_searcher = GridSearchCV(XGB, parameter_grid, verbose=0, cv=5, refit=True)
start = time()
# Обучение модели
grid_searcher.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
finish = time()
XGB_best = grid_searcher.best_estimator_
print (best_model(grid_searcher))
print("Время работы алгоритма: ", round(finish - start), " секунд")
XGboosting=XGB_best
# График важности признаков
features_value = pd.DataFrame({'features': XGboosting.feature_importances_,
'value': X.columns})
features_value.plot(kind='bar', x='value', y='features', figsize=(7, 7),
legend=False)
plt.title('XGBoost Features')
plt.ylabel('Признаки')
plt.xlabel('Важность')
plt.show()
print ("Признаки: ", XGboosting.feature_importances_, "\n", "Сумма: ",
XGboosting.feature_importances_.sum())
# Метрики
print(classification_report(df_train.Exited,  XGboosting.predict(df_train.loc[:,
df_train.columns != 'Exited'])))
auc_XGboosting, fpr_XGboosting, tpr_XGboosting = get_auc_scores(y,
XGboosting.predict(X),XGboosting.predict_proba(X)[:,1])
# AUC ROC метрика
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_XGboosting, tpr_XGboosting, label = 'XGB score: ' +
str(round(auc_XGboosting,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
```

```
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Тестовая выборка
df_test = Df_score(df_test,df_train.columns,minVec,maxVec)
df_test = df_test.mask(np.isinf(df_test))
df_test = df_test.dropna()
print ("Клиентов: ",df_test.shape[0],"\n", "Атрибутов: ",df_test.shape[1],"\n")
df_test.head(5)
# Метрики
print(classification_report(df_test.Exited,  RForest.predict(df_test.loc[:,
df_test.columns != 'Exited'])))
# AUC ROC метрика
auc_RForest_test, fpr_RForest_test, tpr_RForest_test =
get_auc_scores(df_test.Exited, RForest.predict(df_test.loc[:, df_test.columns !=
'Exited']), RForest.predict_proba(df_test.loc[:, df_test.columns !=
'Exited'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RForest_test, tpr_RForest_test, label = 'RF score: ' +
str(round(auc_RForest_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Метрика Accuracy
accuracy_score(df_test.Exited, RForest.predict(df_test.loc[:, df_test.columns !=
'Exited']))
# Матрица ошибок
confusion_matrix(df_test.Exited, RForest.predict(df_test.loc[:, df_test.columns !=
'Exited']))
# Метрики
print(classification_report(df_test.Exited, CatBoost.predict(df_test.loc[:,
df_test.columns != 'Exited'])))
# AUC ROC метрика
auc_CatBoost_test, fpr_CatBoost_test, tpr_CatBoost_test =
get_auc_scores(df_test.Exited, CatBoost.predict(df_test.loc[:, df_test.columns !=
'Exited']), CatBoost.predict_proba(df_test.loc[:, df_test.columns !=
'Exited'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_CatBoost_test, tpr_CatBoost_test, label = 'CatBoost score: ' +
str(round(auc_CatBoost_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Метрика Accuracy
accuracy_score(df_test.Exited, CatBoost.predict(df_test.loc[:, df_test.columns !=
'Exited']))
# Матрица ошибок
confusion_matrix(df_test.Exited, CatBoost.predict(df_test.loc[:, df_test.columns !=
'Exited']))
# Метрики
print(classification_report(df_test.Exited,  XGboosting.predict(df_test.loc[:,
df_test.columns != 'Exited'])))
# AUC ROC метрика
auc_XGboosting_test, fpr_XGboosting_test, tpr_XGboosting_test =
get_auc_scores(df_test.Exited, XGboosting.predict(df_test.loc[:, df_test.columns !=
'Exited']), XGboosting.predict_proba(df_test.loc[:, df_test.columns !=
'Exited'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_XGboosting_test, tpr_XGboosting_test, label = 'XGB score: ' +
```

```
str(round(auc_XGboosting_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Метрика Accuracy
accuracy_score(df_test.Exited, XGboosting.predict(df_test.loc[:, df_test.columns !=
'Exited']))
# Матрица ошибок
confusion_matrix(df_test.Exited, XGboosting.predict(df_test.loc[:, df_test.columns
!= 'Exited']))
# Сравнение результатов работы программы
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RForest_test, tpr_RForest_test, label = 'RF score: ' +
str(round(auc_RForest_test,2)))
plt.plot(fpr_CatBoost_test, tpr_CatBoost_test, label = 'CatBoost score: ' +
str(round(auc_CatBoost_test,2)))
plt.plot(fpr_XGboosting_test, tpr_XGboosting_test, label = 'XGB score: ' +
str(round(auc_XGboosting_test,2)))
plt.plot([0,1], [0,1], 'g--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
# Вывод результатов в 2 столбца
def print2 ( *args ):
    num = len(args)
    x = 0
    while x < num:
        try:
            print(str(int(args[x])) + " " + str(int(args[x + 1])))
        except:
            print(str(int(args[x])))
        x += 2
# Экспорт в txt
import sys
sys.stdout = open('Bank_B_RF.txt', 'w')
print2(*df_test.Exited[:], *RForest.predict(df_test.loc[:, df_test.columns !=
'Exited'][:]))
sys.stdout.close()
# Экспорт в txt
sys.stdout = open('Bank_B_CB.txt', 'w')
print2(*df_test.Exited[:], *CatBoost.predict(df_test.loc[:, df_test.columns !=
'Exited'][:]))
sys.stdout.close()
# Экспорт в txt
sys.stdout = open('Bank_B_XGB.txt', 'w')
print2(*df_test.Exited[:], *XGboosting.predict(df_test.loc[:, df_test.columns !=
'Exited'][:]))
sys.stdout.close()
```

ПРИЛОЖЕНИЕ В. ТЕКСТ ПРОГРАММЫ ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ

Form1.cs:

```
using System;
using System.IO;
using System.Drawing;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.Diagnostics;
namespace Visualization
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Open_file_Click(object sender, EventArgs e) // Нажатие кнопки
"Открыть"
        {
            using (OpenFileDialog openDialog = new OpenFileDialog())
            {
                openDialog.Filter = "Текстовые файлы|*.txt"; // Импорт текстового
файла
                if (openDialog.ShowDialog(this) == DialogResult.OK)
                {
                    // Динамическое изменение формы для удобства пользователя
                    file.Text = openDialog.FileName;
                    filename.Text +=
Path.GetFileNameWithoutExtension(openDialog.FileName);
                    file_log.Text = filename.Text;
                    open_file.Enabled = false;
                    true_predict.Visible = true;
                    false_predict.Visible = true;
                    time_visualization.Visible = true;
                    equal.Visible = true;
                    unequal.Visible = true;
                    visualization_time.Visible = true;
                    matrix.Visible = true;
                    p2.Visible = true;
                    p1.Visible = true;
                    n1.Visible = true;
                    n2.Visible = true;
                    matrix_TP.Visible = true;
                    matrix_FN.Visible = true;
                    matrix_FP.Visible = true;
                    matrix_TN.Visible = true;
                    start.Enabled = true;
                    start.Visible = true;
                    progress.Visible = true;}}}
        private void Start_Click(object sender, EventArgs e) // Нажатие кнопки
"Посчитать"
        {
            int o1 = 1; // Номер строки
            int TP = 0;
            int TN = 0;
            int FP = 0;
            int FN = 0;
            string line; // Строка
```

```
            Stopwatch findTime = new Stopwatch();
            findTime.Start(); // Начало счета визуализации
            using (var streamReader = new StreamReader(file.Text,
System.Text.Encoding.Default))
            {
                while ((line = streamReader.ReadLine()) != null) // Пока есть
строки в файле
                {
                    var stringNumbers = line.Split(' '); // Разделитель цифр
                    string expression = o1 + ".  " + line;
                    // Проверка на корректные значения строки
                    if (stringNumbers[0].Length > 1 || stringNumbers[1].Length > 1
|| Convert.ToInt32(stringNumbers[0]) > 1 || Convert.ToInt32(stringNumbers[1]) > 1)
                    {
                        MessageBox.Show("Неверные данные");
                        Application.Exit();
                    }
                    if (stringNumbers[0] == stringNumbers[1]) // Если совпало (TP,
TN)
                        if (Convert.ToInt32(stringNumbers[0]) > 0) // TP
                        {
                            TP++;
                            // Визуализация строки
                            file_log.Text += Environment.NewLine + expression + "
+";
                        }
                        else // TN
                        {
                            TN++;
                            // Визуализация строки
                            file_log.Text += Environment.NewLine + expression + "
+";
                        }
                    else // Иначе (FP, FN)
                        if (Convert.ToInt32(stringNumbers[0]) > 0) // FN
                    {
                        FN++;
                        // Визуализация строки
                        file_log.Text += Environment.NewLine + expression + "   -";
                    }
                    else // FP
                    {
                        FP++;
                        // Визуализация строки
                        file_log.Text += Environment.NewLine + expression + "   -";
                    }
                    o1 += 1;
                    // Динамическое изменение максимума прогресс-бара
                    progress.Maximum += progress.Step;
                    progress.Value += progress.Step;}
                streamReader.Close(); // Закрыть файл
                for (int i = 1; i < file_log.Lines.Length; i++) // Закрашиваем
строки
                {
                    string text = file_log.Lines[i];
                    file_log.Select(file_log.GetFirstCharIndexFromLine(i),
text.Length);
                    if (text.Contains("-")) // Если не совпали
                    {
                        file_log.SelectionColor = Color.Red;
                    }}
                float P = TP + FN; // Количество полож. объектов
                float N = TN + FP; // Количество отриц. объектов
                // Верный прогноз
```

```
                equal.Text = Convert.ToString(String.Format("{0:0.0}", ((TP + TN) /
(P + N)) * 100));
                // Неверный прогноз
                unequal.Text = Convert.ToString(String.Format("{0:0.0}", (FN / (N +
P) + FP / (N + P)) * 100));
                // Заполняем матрицу ошибок (confusion matrix)
                matrix_TP.Text = Convert.ToString(TP); // TP (1 1)
                matrix_FN.Text = Convert.ToString(FN); // FN (1 0)
                matrix_FP.Text = Convert.ToString(FP); // FP (0 1)
                matrix_TN.Text = Convert.ToString(TN); // TN (0 0)
                // Условия изменения цвета для верного прогнозирования
                if (Convert.ToDouble(equal.Text) <= 100 &&
Convert.ToDouble(equal.Text) > 90)
                equal.BackColor = Color.Green;
                else if (Convert.ToDouble(equal.Text) <= 90 &&
Convert.ToDouble(equal.Text) > 80)
                equal.BackColor = Color.LightGreen;
                else if (Convert.ToDouble(equal.Text) <= 80 &&
Convert.ToDouble(equal.Text) > 70)
                equal.BackColor = Color.GreenYellow;
                else if (Convert.ToDouble(equal.Text) <= 70 &&
Convert.ToDouble(equal.Text) >= 60)
                equal.BackColor = Color.Yellow;
                else equal.BackColor = Color.Red;
                // Условия изменения цвета для неверного прогнозирования
                if (Convert.ToDouble(unequal.Text) >= 0 &&
Convert.ToDouble(unequal.Text) < 10)
                unequal.BackColor = Color.Green;
                else if (Convert.ToDouble(unequal.Text) >= 10 &&
Convert.ToDouble(unequal.Text) < 20)
                unequal.BackColor = Color.LightGreen;
                else if (Convert.ToDouble(unequal.Text) >= 20 &&
Convert.ToDouble(unequal.Text) < 30)
                unequal.BackColor = Color.GreenYellow;
                else if (Convert.ToDouble(unequal.Text) >= 30 &&
Convert.ToDouble(unequal.Text) <= 40)
                unequal.BackColor = Color.Yellow;
                else unequal.BackColor = Color.Red;}
            // Построение диаграммы
            string[] seriesArray = { true_predict.Text, false_predict.Text }; //
Строки диаграммы
            double[] pointsArray = { Convert.ToDouble(equal.Text),
Convert.ToDouble(unequal.Text) }; // Столбцы
            diagram.BorderlineDashStyle = ChartDashStyle.Solid;
            diagram.ChartAreas[0].AxisX.Enabled = AxisEnabled.False;
            diagram.ChartAreas[0].AxisY.MajorGrid.LineColor =
SystemColors.ControlLight;
            diagram.ChartAreas[0].AxisY.Minimum = 0;
            diagram.ChartAreas[0].AxisY.Maximum = 100;
            diagram.ChartAreas[0].AxisY.MajorGrid.Interval = 10;
            diagram.ChartAreas[0].AxisY.Title = "Соотношение прогнозов в банковской
сфере";
            diagram.BackColor = Color.PaleTurquoise;
            diagram.BackSecondaryColor = Color.White;
            diagram.BackGradientStyle = GradientStyle.DiagonalRight;
            diagram.BorderlineColor = Color.White;
            diagram.ChartAreas[0].BackColor = Color.Orange;
            diagram.Palette = ChartColorPalette.Bright;
            diagram.Titles.Add("Accuracy of" +
Path.GetFileNameWithoutExtension(filename.Text));
            for (int i = 0; i < seriesArray.Length; i++)
            {
                Series series = diagram.Series.Add(seriesArray[i]);
                series.Points.Add(pointsArray[i]);
                diagram.ChartAreas[0].Area3DStyle.Enable3D = true;
```

```
                diagram.ChartAreas[0].Area3DStyle.IsRightAngleAxes = false;
                diagram.ChartAreas[0].Area3DStyle.Inclination = 0;
                diagram.ChartAreas[0].Area3DStyle.Rotation = -90;
                diagram.ChartAreas[0].Area3DStyle.LightStyle =
LightStyle.Realistic;
                diagram.ChartAreas[0].Area3DStyle.Perspective = 10;}
            findTime.Stop(); // Окончание счета визуализации
            equal.Text += " %";
            unequal.Text += " %";
            visualization_time.Text = (findTime.Elapsed.Ticks) / 10000000 + "
секунд";
            // Динамическое изменение формы для удобства пользователя
            diagram.Visible = true;
            file_log.Visible = true;
            start.Enabled = false;
            save_diagram.Visible = true;
            close.Visible = true;}

        private void Form1_Load(object sender, EventArgs e) // Загрузка формы
        {
            this.AutoSize = true; // Динамическое изменение формы для удобства
пользователя
        }
        private void Close_Click(object sender, EventArgs e) // Нажатие кнопки
"Закрыть"
        {
            this.Close();
        }
        private void Save_Click(object sender, EventArgs e) // Нажатие кнопки
"Сохранить"
        {
            // Экспорт в Png
            save_diagram.Enabled = false;
            var image = Path.ChangeExtension(file.Text, "png");
            diagram.SaveImage(image, System.Drawing.Imaging.ImageFormat.Png);
        }
    }
}
```