Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования
Направление подготовки: 01.04.02 – «Прикладная математика и информатика»
Направленность (профиль) образовательной программы – «Математическое и программное обеспечение информационных систем»

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2025 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: «Деревья принятия решений в задаче прогнозирования оттока клиентов банка»

Исполнитель
студент группы 3101ом _____ Д.Н. Четыркин
(подпись, дата)

Научный руководитель
доцент, канд. физ.-мат. наук _____ Н.Н. Максимова
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
доцент, канд. физ.-мат. наук _____ Н.Н. Максимова
(подпись, дата)

Нормоконтроль
ассистент _____ В.О. Салмиянов
(подпись, дата)

Рецензент
доцент, канд. тех. наук _____ Т.А. Галаган
(подпись, дата)

Благовещенск 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2025 г.

З А Д А Н И Е

К магистерской диссертации студента Дениса Николаевича Четыркина

1. Тема магистерской диссертации: Деревья принятия решений в задаче прогнозирования оттока клиентов банка.

(утверждены приказом от 24.01.2025 г. № 162-уч)

2. Срок сдачи студентом законченной проекта: 24 июня 2025 года.

3. Исходные данные к магистерской диссертации: набор данных «Bank Churn Dataset»; библиотека машинного обучения Scikit-learn языка Python; Google Colab – среда разработки, используемая для написания кода и анализа данных; методы машинного обучения – дерево решений (Decision Tree) и случайный лес (Random Forest) для построения моделей прогнозирования оттока клиентов; отчеты по производственной практике (научно-исследовательской работе) за 1-3 семестры; отчет по преддипломной практике.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): обзор методов искусственного интеллекта и машинного обучения, их развития и областей применения, анализ задач, решаемых методами машинного обучения, с акцентом на банковскую сферу.

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): листинги программного кода для построения моделей Decision Tree и Random Forest; графики

и таблицы, отображающие результаты предварительной обработки данных.

6. Консультанты по магистерской диссертации: рецензент – Галаган Т.А., доцент кафедры информационных и управляющих систем, ФГБОУ ВО «АмГУ», канд. техн. наук, доцент; нормоконтроль – Салмиянов В.О., ассистент.

7. Дата выдачи задания: 03.02.2025 г.

Руководитель магистерской диссертации: Максимова Надежда Николаевна, доцент, канд. физ.-мат. наук, доцент

Задание принял к исполнению (03.02.2025): _____ Д.Н. Четыркин

РЕФЕРАТ

Магистерская диссертация содержит 64 с., 13 рисунков, 7 таблиц, 6 приложений, 22 источника.

МАШИННОЕ ОБУЧЕНИЕ, ДЕРЕВО РЕШЕНИЙ, ПРОГНОЗИРОВАНИЕ ОТТОКА КЛИЕНТОВ, АНАЛИЗ ДАННЫХ, МОДЕЛИРОВАНИЕ, КАЧЕСТВО МОДЕЛИ, PYTHON, SKLEARN.

В работе исследуется задача прогнозирования оттока клиентов банка с использованием методов машинного обучения, в частности деревьев принятия решений и их ансамблей. Основное внимание уделяется построению моделей на основе Decision Tree и Random Forest, анализу их точности и применимости в банковской сфере.

Цель магистерской диссертации – изучение и реализация моделей машинного обучения (дерево решений и случайный лес) для прогнозирования оттока клиентов банка, а также анализ качества построенных моделей и их сравнение.

Для достижения цели были поставлены следующие задачи:

- изучение научной и методической литературы по вопросам машинного обучения и прогнозирования оттока клиентов;
- анализ существующих методов классификации, в том числе деревьев решений и ансамблевых методов;
- описание и подготовка данных из набора «Bank Churn Dataset»;
- построение модели на основе Decision Tree и подбор ее гиперпараметров;
- построение модели Random Forest, настройка параметров и сравнение с моделью Decision Tree;
- оценка качества моделей с использованием различных метрик (Accuracy, Precision, Recall, F1-score, ROC-AUC);
- анализ результатов моделирования и выбор наиболее эффективного метода для прогнозирования оттока клиентов.

Основу методологии исследования составляют методы машинного обучения, в частности деревья решений и случайный лес, реализованные с использованием библиотеки scikit-learn языка Python. В ходе работы была проведена обработка данных, построены модели, выполнен их сравнительный анализ, а также предложены рекомендации по их применению в банковской сфере.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Деревья решений представляют собой один из основных методов машинного обучения, используемых для классификации и регрессии на табличных данных. Этот метод создает модель в виде дерева, где каждый внутренний узел соответствует проверке определенного признака, каждая ветвь представляет результат этой проверки, а каждый лист – конечное решение или предсказание. Деревья решений обладают рядом преимуществ, включая интерпретируемость, способность работать с данными разной природы и устойчивость к выбросам. Компьютерные системы используют алгоритмы деревьев решений для обработки больших объемов данных и выявления шаблонов, что позволяет эффективно решать задачи классификации и регрессии.

Использование деревьев решений может быть полезно для аналитиков данных и специалистов в различных областях для автоматической классификации объектов, выявления важных признаков и принятия решений на основе имеющейся информации.

Современная банковская сфера сталкивается с проблемой оттока клиентов, что оказывает значительное влияние на финансовые показатели организаций. Привлечение новых клиентов зачастую требует больших затрат, чем удержание уже существующих, поэтому прогнозирование вероятности оттока является важной задачей для банков. В условиях высокой конкуренции кредитные организации стремятся минимизировать потери за счет анализа поведения клиентов и внедрения эффективных стратегий удержания.

Актуальность данной работы обусловлена растущей ролью искусственного интеллекта и машинного обучения в сфере финансовых технологий. Банки ежедневно обрабатывают большие объемы данных, содержащих информацию о транзакциях, кредитных историях, платежеспособности и поведении клиентов. Внедрение методов машинного обучения в процесс анализа этих данных позволяет повысить точность прогнозирования, минимизировать финансовые потери и разрабатывать индивидуальные стратегии удержания клиентов.

Использование деревьев решений и ансамблей моделей позволяет эффективно решать задачу классификации клиентов на остающихся и уходящих, что дает возможность своевременно предпринимать меры по удержанию. В то же время, в отличие от более сложных моделей, таких как нейронные сети, деревья решений обеспечивают хорошую интерпретируемость, что делает их удобными для практического применения в банковской сфере.

Работа направлена на исследование методов прогнозирования оттока клиентов банка с использованием алгоритмов дерева решений и случайного леса. Основное внимание уделяется построению, настройке и сравнению моделей, а также анализу их точности и эффективности.

Объектом исследования является набор данных, содержащий информацию о различных объектах и их характеристиках. **Предмет** исследования – использование деревьев решений для классификации на табличных данных.

Целью работы является создание моделей машинного обучения на основе деревьев решений для классификации объектов на табличных данных, а также оценка их эффективности и точности.

Для достижения поставленной цели необходимо решить следующие **задачи**:

- Провести обзор научной литературы по машинному обучению и его применению в банковской сфере.
- Изучить теоретические основы деревьев решений и их ансамблей.
- Ознакомиться с набором данных «Bank Churn Dataset» и провести его предварительную обработку.
- Разработать модель на основе Decision Tree, настроить гиперпараметры и проанализировать результаты.
- Разработать модель на основе Random Forest, провести настройку параметров и сравнить её эффективность с моделью Decision Tree.
- Оценить качество моделей с использованием различных метрик.
- Сравнить полученные модели и определить наиболее эффективный метод для прогнозирования оттока клиентов.

Программные средства, используемые для реализации и анализа моделей:

- язык программирования Python;
- библиотека машинного обучения scikit-learn;
- среда разработки Google Colab;
- библиотеки для обработки и визуализации данных (pandas, matplotlib, seaborn).

Новизна исследования заключается в сравнительном анализе методов построения деревьев решений и случайного леса для прогнозирования оттока клиентов. Особое внимание уделяется влиянию гиперпараметров на точность моделей, а также интерпретации полученных результатов.

Теоретическая значимость работы заключается в систематизации знаний о методах машинного обучения, применяемых в задачах прогнозирования оттока клиентов. Также работа вносит вклад в изучение влияния различных параметров моделей на их точность и устойчивость.

Практическая значимость исследования заключается в разработке и тестировании моделей, которые могут быть использованы в реальных банковских системах для прогнозирования оттока клиентов. Полученные результаты помогут финансовым организациям вырабатывать стратегии удержания клиентов, минимизировать риски и повышать лояльность пользователей.

Работа состоит из введения, трех глав, заключения, списка литературы и приложений.

В первой главе рассматриваются основные понятия искусственного интеллекта и машинного обучения, их применение в банковской сфере, а также ключевые метрики оценки моделей.

Вторая глава посвящена изучению деревьев решений и ансамблевых методов, включая метод случайного леса. Рассматриваются основные этапы построения моделей, а также программная реализация на Python с использованием библиотеки scikit-learn.

В третьей главе проводится практическое моделирование: рассматривается набор данных «Bank Churn Dataset», выполняется его предварительная обработка, строятся модели Decision Tree и Random Forest, проводится настройка гиперпараметров и анализируются результаты.

По результатам работы **опубликованы** две работы [21-22].

Результаты работы докладывались на двух **научных конференциях**:

- XXXIII научная конференция «День науки» (Амурский государственный университет, 2024 гг.);

- XXXIV научная конференция «День науки» (Амурский государственный университет, 2025 гг.);

1 ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И МАШИННОЕ ОБУЧЕНИЕ

1.1 Основные понятия и этапы развития искусственного интеллекта

1.1.1 Понятие искусственного интеллекта

Искусственный интеллект (ИИ) представляет собой динамичную и быстро развивающуюся область компьютерных наук, посвященную созданию систем, способных анализировать данные, извлекать закономерности, принимать решения и решать задачи, которые ранее считались прерогативой человеческого интеллекта [1-2].

Главными концепциями в области искусственного интеллекта являются обучение, рассуждение и самокоррекция.

Обучение – это процесс, в рамках которого система приобретает знания и опыт из имеющихся данных. Обучение может происходить различными способами, такими как обучение с учителем, обучение без учителя и обучение с подкреплением. В процессе обучения система анализирует данные, выделяет закономерности и строит модели, которые позволяют делать прогнозы или принимать решения на основе новых данных.

Рассуждение – этот аспект искусственного интеллекта касается способности системы применять правила и логику для формирования выводов или принятия решений. Рассуждение может быть символическим, статистическим или логическим, в зависимости от конкретной задачи и методологии. Системы искусственного интеллекта могут использовать рассуждение для решения сложных проблем, анализа данных или предоставления рекомендаций.

Самокоррекция – это способность системы адаптироваться к новым данным или ситуациям и корректировать свое поведение или модели в соответствии с этими изменениями. Эта характеристика особенно важна в сферах, где данные или условия могут изменяться со временем. Системы искусственного интеллекта должны быть способным обновлять свои знания и модели, чтобы оставаться актуальными и эффективными [3-4].

Искусственный интеллект применяется в различных областях, включая медицину, финансы, производство, транспорт и многое другое. Его применение способствует автоматизации процессов, повышению производительности и созданию новых возможностей для решения сложных задач, которые ранее казались недостижимыми без участия человека.

1.1.2 Этапы развития искусственного интеллекта

Развитие искусственного интеллекта можно разделить на несколько ключевых этапов:

1) 1950-е годы: Эпоха зарождения идеи искусственного интеллекта. Этот период отмечен работой таких выдающихся ученых, как Алан Тьюринг и Джон Маккарти, которые внесли значительный вклад в создание основных концепций и методов искусственного интеллекта. В это время были разработаны первые программы, способные решать некоторые ограниченные задачи, такие как игра в шахматы и решение алгебраических проблем.

2) 1960-е - 1970-е годы: Развитие теории и практики искусственного интеллекта. В этот период были созданы первые роботы и экспертные системы, способные выполнять определенные задачи с использованием заранее определенных правил и знаний. Ученые активно изучали методы символьного и статистического вывода, а также разрабатывали алгоритмы для решения проблем в области машинного видения и обработки естественного языка.

3) 1980-е годы: Всплеск интереса к искусственному интеллекту. С развитием компьютерных технологий и вычислительной мощности начался новый этап в истории ИИ. Были созданы более сложные алгоритмы и модели, позволяющие решать более широкий спектр задач. В этот период активно развивались методы машинного обучения, включая нейронные сети и генетические алгоритмы.

4) 1990-е – 2010-е годы: Расцвет машинного обучения и Big Data. В этот период произошел значительный скачок в развитии искусственного интеллекта благодаря росту вычислительных мощностей и распространению интернета.

Начали активно развиваться методы машинного обучения, особенно статистические методы и алгоритмы, такие как SVM (метод опорных векторов) и ансамблевые модели (Random Forest, Gradient Boosting). В 2000-х годах объем данных резко увеличился, что способствовало росту интереса к Big Data и углубленному анализу информации.

5) 2010-е годы – настоящее время: Эпоха глубокого обучения и повсеместная интеграция ИИ. Современные достижения в области искусственного интеллекта связаны с развитием глубоких нейронных сетей (Deep Learning), особенно благодаря архитектурам CNN (сверточные нейронные сети), RNN (рекуррентные нейронные сети) и трансформерам (например, GPT). Расширение облачных вычислений, разработка специализированных процессоров (GPU, TPU) и доступ к огромным объемам данных привели к тому, что ИИ теперь применяется практически во всех сферах жизни: от автоматизированных систем здравоохранения и финансовых технологий до автономных транспортных средств и генеративных моделей контента.

1.2 Области применения искусственного интеллекта

Искусственный интеллект охватывает множество различных областей, включая:

1) Машинное зрение: Эта область фокусируется на разработке систем, способных анализировать и понимать визуальную информацию, такую как изображения и видео. Машинное зрение используется в различных задачах, включая распознавание объектов, обнаружение и классификацию лиц, анализ медицинских изображений, автоматическое видеонаблюдение и многое другое.

2) Обработка естественного языка (NLP): Эта область, посвященная разработке систем, способных понимать, интерпретировать и генерировать текстовую информацию на естественных языках. Применения NLP включают в себя машинный перевод, анализ тональности текста, извлечение информации из текстов, создание чат-ботов и многое другое.

3) робототехника: искусственный интеллект используется для управления и координации действий роботов в различных сферах, включая производство,

медицину, обслуживание клиентов и т.д. Роботы, оснащенные ИИ, могут выполнять широкий спектр задач, от манипуляции предметами до навигации в неструктурированных средах.

4) экспертные системы: это программные системы, созданные для имитации знаний и решений человеческих экспертов в определенной области. Они используются для решения сложных проблем и поддержки принятия решений в различных областях, таких как медицина, финансы, юриспруденция и т.д. Экспертные системы могут анализировать данные, формулировать диагнозы, предоставлять рекомендации и многое другое, что обеспечивает значительное улучшение процессов в таких областях.

5) автоматизация процессов и управление: искусственный интеллект применяется для автоматизации различных бизнес-процессов и управления ресурсами. Это включает в себя создание систем управления запасами, оптимизацию производственных процессов, автоматизацию бухгалтерии и т.д. ИИ позволяет улучшить эффективность и точность таких процессов, сокращая время и затраты.

6) медицина и здравоохранение: в этой области искусственный интеллект используется для диагностики заболеваний, анализа медицинских изображений, прогнозирования распространения заболеваний, разработки индивидуализированных лечебных схем и многое другое. ИИ помогает врачам и медицинским специалистам в принятии более точных диагнозов и назначении более эффективного лечения.

7) финансы и инвестиции: в финансовой сфере искусственный интеллект применяется для анализа рынков, прогнозирования трендов, оптимизации инвестиционных портфелей, выявления мошеннических операций и многое другое. Алгоритмы машинного обучения и нейронные сети помогают финансовым аналитикам принимать более обоснованные и выгодные решения.

8) образование: в сфере образования искусственный интеллект используется для персонализации обучения, создания интеллектуальных образовательных платформ, автоматизации оценки знаний студентов, адаптации учебных ма-

териалов под индивидуальные потребности и многое другое. Это помогает студентам получать более качественное образование и эффективнее осваивать учебный материал.

1.3 Задачи, решаемые методами машинного обучения

Машинное обучение помогает решать разнообразные задачи:

Классификация

Эта задача заключается в присвоении объектам одной или нескольких категорий на основе их характеристик. Примерами могут служить классификация электронных писем как спама или не спама, классификация изображений как кошек или собак, определение типа опухоли на медицинских изображениях и многое другое. Алгоритмы классификации, такие как метод опорных векторов (SVM), наивный Байесовский классификатор, логистическая регрессия и нейронные сети, широко используются для решения таких задач.

Регрессия

Эта задача состоит в прогнозировании количественных значений на основе набора входных данных. Например, регрессия может использоваться для прогнозирования цены на недвижимость на основе характеристик домов, прогнозирования выручки компании на следующий квартал, предсказания погоды на основе метеорологических данных и т.д. Методы регрессии включают линейную регрессию, регрессионные деревья, метод опорных векторов для регрессии и др.

Кластеризация

Эта задача состоит в группировке объектов на основе их сходства без заранее определенных категорий. Кластеризация используется, когда нет явных меток классов для объектов. Примерами могут служить сегментация клиентов по их покупательским предпочтениям, группировка новостных статей по темам, кластеризация географических областей по климатическим характеристикам и т.д. Алгоритмы кластеризации включают k-средних, иерархическую кластеризацию, алгоритмы DBSCAN и многие другие [5-6].

1.4 Искусственный интеллект в банковской сфере

Современные технологии ИИ широко применяются в различных направлениях банковского бизнеса, создавая конкурентные продукты и повышая эффективность внутренних процессов.

Основные направления применения ИИ в банковской сфере:

- Клиентское обслуживание и чат-боты.

ИИ активно используется для улучшения взаимодействия с клиентами. Чат-боты и виртуальные ассистенты, созданные на основе технологий обработки естественного языка (NLP), помогают автоматизировать ответы на типовые запросы, проводить консультации по продуктам и услугам банка, а также оперативно решать возникающие проблемы. Например, такие решения позволяют снизить нагрузку на контактные центры и обеспечить круглосуточную поддержку.

- Кредитный скоринг и оценка рисков.

Модели машинного обучения позволяют банкам более точно оценивать кредитоспособность заемщиков, анализируя широкий спектр данных: от финансовой истории до поведения клиентов.

- Обнаружение мошенничества (фрод-мониторинг).

ИИ играет ключевую роль в обеспечении безопасности банковских операций. Алгоритмы анализа транзакционных данных выявляют аномалии в режиме реального времени, сигнализируя о возможных случаях мошенничества. Благодаря этим технологиям банки могут минимизировать финансовые потери и защитить своих клиентов от киберугроз.

- Персонализация продуктов и услуг.

На основе анализа больших данных ИИ помогает банкам предлагать клиентам персонализированные продукты и услуги. Это может быть рекомендации по инвестициям, подбор оптимального кредитного предложения или управление личными финансами. Подобный подход увеличивает лояльность клиентов и способствует росту доходности банка.

- Автоматизация внутренних процессов.

Технологии ИИ также используются для оптимизации внутренних процессов в банках, таких как обработка документов, управление персоналом и финансовое планирование. Роботизированные автоматизированные системы (RPA) ускоряют выполнение рутинных задач и минимизируют вероятность ошибок.

Инструменты и решения на базе ИИ

Для реализации указанных направлений применяются разнообразные инструменты и платформы. Среди них можно выделить:

- Системы обработки естественного языка (NLP) для создания чат-ботов и анализа текстовых данных.
- Платформы машинного обучения, такие как TensorFlow, PyTorch или отечественные аналоги, используемые для разработки и обучения моделей.
- Программные комплексы для фрод-мониторинга, способные обрабатывать миллионы транзакций в реальном времени.
- Аналитические системы для работы с большими данными, включая облачные платформы и решения для визуализации данных.

Примеры использования ИИ в российских банках

Российские банки активно внедряют ИИ в свою деятельность. Примеры некоторых из них:

- Сбербанк: использует технологии ИИ для работы с клиентами. Виртуальный ассистент "Салют" помогает пользователям управлять продуктами банка, а также проводить консультации.
- Тинькофф Банк: применяет ИИ для персонализации продуктов, управления рисками и анализа транзакций. Один из заметных проектов - система прогнозирования расходов клиентов.
- Альфа-Банк: внедрил чат-боты на основе ИИ, которые помогают в обслуживании клиентов и выполнении простых операций, таких как переводы или консультации по кредитам.
- ВТБ: использует технологии ИИ для автоматизации процессов, включая проверку документов и оценку заявок на кредиты.

1.5 Метрики качества в задачах машинного обучения

Для оценки эффективности моделей машинного обучения используются различные метрики:

- Точность (Accuracy): доля правильно классифицированных объектов от всех объектов.

Accuracy измеряет долю правильно классифицированных объектов относительно общего числа объектов. Эта метрика вычисляется по формуле:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

где:

TP (True Positive) – количество истинно положительных предсказаний (модель верно классифицировала положительные примеры).

TN (True Negative) – количество истинно отрицательных предсказаний (модель верно классифицировала отрицательные примеры).

FP (False Positive) – количество ложноположительных предсказаний (модель ошибочно классифицировала отрицательные примеры как положительные).

FN (False Negative) – количество ложноотрицательных предсказаний (модель ошибочно классифицировала положительные примеры как отрицательные).

- Точность (Precision) и Полнота (Recall): метрики, оценивающие качество классификации в задачах с несбалансированными классами.

Precision измеряет долю истинно положительных предсказаний среди всех предсказаний, сделанных моделью как положительные. Формула:

$$Precision = \frac{TP}{TP + FP}$$

Высокое значение Precision говорит о том, что модель почти не делает ложноположительных ошибок.

Recall измеряет долю истинно положительных объектов, которые модель смогла корректно идентифицировать среди всех объектов с положительным классом. Формула:

$$Recall = \frac{TP}{TP + FN}$$

Высокое значение Recall говорит о том, что модель практически не пропускает положительные предсказания.

- F-мера: гармоническое среднее точности и полноты.

F-мера объединяет Precision и Recall в одно значение, представляя собой их гармоническое среднее. Формула:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Гармоническое среднее более чувствительно к низким значениям, чем арифметическое. Таким образом, F-мера будет низкой, если хотя бы одна из метрик (Precision или Recall) низкая.

- ROC-AUC: площадь под кривой ошибок.

Метрика показывает зависимость между долей истинно положительных предсказаний (True Positive Rate, TPR) и долей ложноположительных предсказаний (False Positive Rate, FPR) при разных порогах классификации.

TPR (или Recall)**:**

$$TPR = \frac{TP}{TP + FN}$$

FPR**:**

$$FRP = \frac{FP}{FP + TN}$$

Показывает, насколько хорошо модель разделяет положительные и отрицательные примеры.

Матрица ошибок – это таблица, которая используется для оценки качества модели классификации. Она показывает количество правильно и ошибочно классифицированных объектов для каждого класса.

Пример матрицы ошибок для задачи прогнозирования оттока клиентов (табл. 2). Предположим, что у нас есть 1000 клиентов, и мы предсказываем, уйдут ли они из банка (1 – ушел, 0 – остался).

В данном случае:

- **700 клиентов (TN)** модель правильно классифицировала как оставшихся.
- **150 клиентов (TP)** модель правильно предсказала как ушедших.

• **50 клиентов (FP)** были ошибочно предсказаны как ушедшие, хотя на самом деле остались.

• **100 клиентов (FN)** модель ошибочно предсказала как оставшихся, но они на самом деле ушли.

Таблица 1 – Пример матрицы ошибок.

	Предсказанный класс: Положительный	Предсказанный класс: Отрицательный
Фактический класс: Положительный	True Positive (TP) – Верно предсказанные положительные	False Negative (FN) – Ошибочно предсказанные как отрицательные
Фактический класс: Отрицательный	False Positive (FP) – Ошибочно предсказанные как положительные	True Negative (TN) – Верно предсказанные отрицательные

Таблица 2 – Пример матрицы ошибок.

	Предсказанный: Остался (0)	Предсказанный: Ушел (1)
Фактический: Остался (0)	TN = 700	FP = 50
Фактический: Ушел (1)	FN = 100	TP = 150

1.6 Программные средства для обучения моделей и построения систем искусственного интеллекта

Развитие искусственного интеллекта (ИИ) и машинного обучения (МО) невозможно без мощных программных инструментов. Эти инструменты включают в себя специализированные библиотеки, платформы, облачные решения и интегрированные среды разработки (IDE), которые позволяют разрабатывать, обучать и внедрять модели искусственного интеллекта. Широкий спектр программных средств предоставляет разработчикам и исследователям возможность работать с данными, создавать сложные архитектуры моделей и оптимизировать процесс их обучения.

Одним из ключевых языков программирования, используемых в машинном обучении, является Python. Его популярность обусловлена удобством синтаксиса, богатой экосистемой и обилием специализированных библиотек. Среди этих библиотек особое место занимает TensorFlow, разработанный компанией Google, который предоставляет мощные инструменты для создания нейросетей и их обучения на графических процессорах (GPU). Для более традиционных задач машинного обучения активно применяется Scikit-learn, который включает алгоритмы классификации, регрессии, кластеризации и методы предобработки данных.

Помимо библиотек, для эффективной работы с моделями ИИ используются интегрированные среды разработки и облачные платформы. Jupyter Notebook – одна из наиболее распространенных сред, позволяющая выполнять код в интерактивном режиме и визуализировать результаты. Google Colab предоставляет аналогичные возможности, но при этом позволяет использовать облачные вычислительные ресурсы, включая графические процессоры (GPU) и тензорные процессоры (TPU), что значительно ускоряет процесс обучения моделей. Также широкое распространение получили среды разработки, такие как Visual Studio Code и PyCharm, которые предоставляют мощные инструменты для отладки, тестирования и оптимизации кода.

Современные технологии предлагают также автоматизированные платформы для машинного обучения, известные как AutoML. Эти платформы позволяют автоматизировать процесс выбора гиперпараметров моделей, оптимизировать их структуру и оценивать качество полученных результатов без необходимости ручной настройки. Среди таких решений можно выделить Google AutoML, H2O.ai и Auto-sklearn. Подобные системы значительно упрощают процесс обучения моделей и делают его доступным для более широкого круга пользователей.

Еще одной важной категорией программных средств являются облачные платформы для машинного обучения. Они позволяют разрабатывать, обучать и

развертывать модели без необходимости использования локального оборудования. Например, Google Cloud AI предоставляет набор инструментов для работы с TensorFlow и AutoML, Amazon SageMaker предлагает мощные средства для обучения и развертывания моделей, а Microsoft Azure Machine Learning предоставляет удобные инструменты для автоматизированного машинного обучения и мониторинга моделей. Эти платформы значительно упрощают процесс работы с большими данными и обеспечивают возможность масштабирования моделей для работы с реальными промышленными нагрузками.

Развитие программных инструментов для искусственного интеллекта также включает создание специализированных аппаратных решений, таких как графические процессоры (GPU) и тензорные процессоры (TPU), разработанные для ускоренного вычисления. Например, современные GPU от NVIDIA широко используются в задачах глубокого обучения, так как они позволяют параллельно выполнять сложные вычисления, что значительно ускоряет обучение моделей. Компания Google разработала TPU, которые оптимизированы для работы с TensorFlow и обеспечивают высокую производительность при обучении нейросетей.

Выбор программных инструментов зависит от конкретной задачи и требований к системе искусственного интеллекта. Для задач, связанных с обработкой больших объемов данных, подходят облачные решения и распределенные вычисления, тогда как для исследований и прототипирования удобнее использовать локальные среды разработки и открытые библиотеки. Развитие технологий машинного обучения и искусственного интеллекта приводит к постоянному совершенствованию программных инструментов, появлению новых алгоритмов и повышению эффективности работы с моделями. В результате современные программные средства позволяют не только обучать модели, но и развертывать их в реальных приложениях, обеспечивая интеграцию ИИ в повседневную жизнь и промышленность.

2 ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ И ИХ АНСАМБЛИ

2.1 Деревья решений

2.1.1 Основные понятия

Деревья принятия решений (Decision Trees) представляют собой мощный инструмент в области машинного обучения, который используется для решения задач классификации и регрессии. Они представляют собой графическую модель принятия решений в виде древовидной структуры, где каждый узел представляет собой тест на характеристику данных, каждая ветвь представляет собой возможный результат теста, а каждый лист дерева представляет собой конечный результат или прогноз (рис. 2.1).

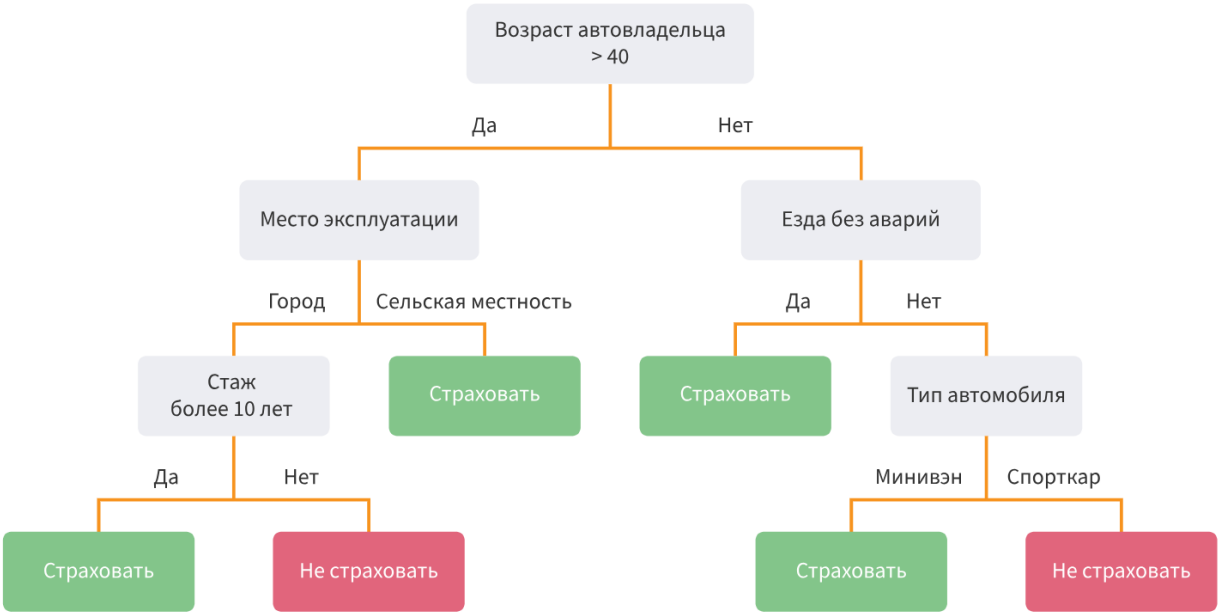

Рисунок 2.1 – Схематичное изображение дерева решений

Узлы, Ветви и Листья

Вершины дерева называются узлами. Узлы, которые не имеют детей, называются листьями, а узлы, имеющие детей, называются внутренними узлами. Ветви соединяют узлы между собой и представляют условия или правила, которые определяют, куда должны переходить данные.

Решающие правила

Каждый узел дерева представляет собой решающее правило, которое используется для принятия решения о разделении данных на более мелкие группы. Решающие правила могут быть любого вида, но обычно они представляют собой проверку определенного признака данных на соответствие определенному порогу или условию.

Классификация и Регрессия

В зависимости от типа задачи, деревья принятия решений могут использоваться как для классификации, так и для регрессии. В задачах классификации, листья дерева представляют собой классы или категории, в которые классифицируются объекты, в то время как в задачах регрессии, листья содержат числовые значения, которые представляют собой прогнозируемую переменную.

Процесс построения дерева

Процесс построения дерева начинается с корневого узла, который содержит всю обучающую выборку. Затем происходит рекурсивное разбиение данных на более мелкие подгруппы в каждом узле с целью минимизации некоторого критерия неопределенности или ошибки. Для каждого разбиения выбирается признак и порог, которые наилучшим образом разделяют данные, и этот процесс продолжается до тех пор, пока не будет выполнен какой-то критерий остановки, такой как максимальная глубина дерева или минимальное количество объектов в листьях.

Критерии разделения

Для определения того, как разбить данные на подгруппы, используются различные критерии разделения, такие как критерий Джини, энтропийный критерий или критерий информационного выигрыша (Information Gain). Каждый из этих критериев оценивает степень неопределенности в данных перед и после разбиения, и выбирается разбиение, которое минимизирует эту неопределенность.

Преимущества и Ограничения

Деревья принятия решений имеют несколько преимуществ, включая простоту интерпретации, способность работать с категориальными и числовыми

данными, а также хорошую масштабируемость. Однако у них также есть некоторые ограничения, такие как склонность к переобучению, особенно при большой глубине дерева, и недостаточная способность к обобщению, особенно при наличии шума или недостающих данных.

Деревья принятия решений широко применяются в различных областях, включая медицину, финансы, маркетинг, биологию и многое другое. Они используются для прогнозирования заболеваний, классификации клиентов по их поведению, выявления мошенничества в финансовых операциях, анализа геномных данных и многое другое.

2.1.2 Этапы построения деревьев решений

Построение деревьев решений является важной задачей в области машинного обучения и анализа данных. Этот процесс включает в себя несколько этапов, начиная с подготовки данных и заканчивая оценкой и оптимизацией модели. Деревья решений обладают уникальными свойствами, включая простоту интерпретации и способность обрабатывать как числовые, так и категориальные данные. В этой главе мы подробно рассмотрим основные этапы построения деревьев решений, начиная с подготовки данных и заканчивая оценкой модели.

1) Предварительная обработка данных

a) Сбор данных

Первым этапом в построении деревьев решений является сбор данных. Данные могут быть получены из различных источников, таких как базы данных, файлы, веб-сайты и сенсоры. Важно, чтобы данные были релевантными и качественными, так как качество данных напрямую влияет на качество модели.

b) Очистка данных

Данные, собранные на предыдущем этапе, часто содержат ошибки, пропуски и шум. Очистка данных включает в себя следующие шаги:

- Удаление или корректировка пропущенных значений.
- Обнаружение и исправление ошибок в данных.
- Удаление дубликатов.
- Фильтрация шума.

c) Преобразование данных

На этом этапе данные преобразуются в формат, удобный для анализа и построения моделей. Преобразование данных включает:

- Нормализацию и стандартизацию числовых признаков.
- Кодирование категориальных признаков (например, one-hot encoding).
- Создание новых признаков на основе существующих данных (feature engineering).

2) Выбор признаков

Выбор признаков (feature selection) является критически важным этапом в построении деревьев решений. Хорошо выбранные признаки могут значительно улучшить производительность модели.

a) Важность признаков

Для выбора наиболее значимых признаков используются различные методы, такие как:

- Методы фильтрации (filter methods), которые оценивают признаки на основе статистических критериев.
- Методы встраивания (embedded methods), которые оценивают важность признаков в процессе построения модели.
- Методы отбора (wrapper methods), которые оценивают признаки путем построения и оценки моделей.

b) Редукция размерности

Если количество признаков слишком велико, это может привести к переобучению модели. Редукция размерности помогает снизить количество признаков без значительной потери информации. Методы редукции включают:

- Метод главных компонент (PCA).
- Линейный дискриминантный анализ (LDA).
- Автоэнкодеры.

3) Построение дерева решений

Построение дерева решений включает в себя несколько этапов, каждый из которых направлен на создание модели, способной точно предсказывать или классифицировать данные.

a) Выбор корневого узла

Корневой узел является начальной точкой дерева решений. Выбор корневого узла осуществляется на основе критерия разбиения, такого как энтропия или индекс Джини. Выбирается признак и порог, которые максимизируют информационный выигрыш или минимизируют неопределенность.

b) Разбиение узлов

Каждый внутренний узел дерева представляет собой разбиение данных на основе одного из признаков. Разбиение продолжается рекурсивно, пока не будут выполнены условия останова. Основные критерии разбиения включают:

- Информационный выигрыш.
- Индекс Джини.
- Ошибку классификации.

c) Условия останова

Чтобы предотвратить переобучение, разбиение узлов должно остановиться при выполнении определенных условий. Условия останова могут включать:

- Максимальная глубина дерева.
- Минимальное количество объектов в узле.
- Минимальное улучшение критериев разбиения.

4) Постобработка и оптимизация дерева

После построения дерева решений необходимо выполнить постобработку и оптимизацию модели для улучшения её производительности и интерпретируемости.

a) Обрезка дерева (pruning)

Обрезка дерева является важным шагом для предотвращения переобучения и повышения обобщающей способности модели. Существуют два основных подхода к обрезке дерева:

Пост-обрезка (post-pruning): Обрезка выполняется после полного построения дерева. Сначала строится полное дерево, а затем удаляются узлы, которые не приносят значимого улучшения.

Пред-обрезка (pre-pruning): Обрезка выполняется на этапе построения дерева. Разбиение прекращается, если оно не приносит значимого улучшения.

b) Выбор гиперпараметров

Выбор оптимальных гиперпараметров, таких как максимальная глубина дерева, минимальное количество объектов в узле и критерий разбиения, является важным этапом оптимизации модели. Для этого используются методы кросс-валидации и сеточного поиска.

c) Интерпретируемость модели

Деревья решений имеют преимущество в своей интерпретируемости. Важно представлять результаты модели в виде, понятном для пользователей. Это может включать визуализацию дерева, выделение важных признаков и предоставление объяснений для каждого решения.

5) Оценка модели

Оценка модели является ключевым этапом в процессе построения деревьев решений, позволяющим определить качество и точность модели на новых данных.

a) Разделение данных

Для оценки модели данные делятся на обучающую и тестовую выборки. Обычно используется метод кросс-валидации для более точной оценки производительности модели.

b) Метрики оценки

Для оценки качества модели используются различные метрики, такие как:

- Точность (accuracy): Доля правильно классифицированных объектов.
- Точность (precision) и полнота (recall): Метрики для оценки качества классификации в задачах с несбалансированными классами.
- F-мера: Гармоническое среднее точности и полноты.
- ROC-AUC: Площадь под кривой ошибок.

2.1.3 Метод DecisionTreeClassifier библиотеки scikit-learn языка Python

Метод DecisionTreeClassifier из библиотеки scikit-learn [10] является одним из наиболее популярных и широко используемых инструментов для построения деревьев решений в задачах классификации. В этой главе рассмотрим основные концепции, связанные с этим методом, его параметры, процесс обучения и применения, а также примеры использования на практике.

Основные концепции

Метод DecisionTreeClassifier в scikit-learn основан на алгоритме CART (Classification and Regression Trees), который строит бинарные деревья решений, используя критерии разделения, такие как индекс Джини или энтропия. Этот алгоритм предназначен для создания модели, которая предсказывает метки классов целевой переменной на основе значений входных признаков.

Индекс Джини

Индекс Джини используется для оценки "чистоты" разбиений в дереве. Он рассчитывается как вероятность того, что случайно выбранный элемент из набора данных будет неправильно классифицирован, если он будет случайно помечен по распределению меток классов в подмножестве. Индекс Джини минимизируется при выборе разбиений, которые лучше всего отделяют классы друг от друга.

Энтропия

Энтропия измеряет уровень неопределенности или беспорядка в данных. Чем выше энтропия, тем более неоднородны данные. При построении дерева решений стремятся к минимизации энтропии, выбирая разбиения, которые уменьшают беспорядок и улучшают однородность подмножеств.

Параметры DecisionTreeClassifier

Метод DecisionTreeClassifier предоставляет множество параметров, которые позволяют настроить процесс построения дерева решений в зависимости от специфики задачи и данных:

- criterion: Критерий для оценки качества разбиений. Возможные значения: "gini" (по умолчанию) и "entropy".

- splitter: Стратегия разбиения в каждом узле. Возможные значения: "best" (по умолчанию) и "random".
- max_depth: Максимальная глубина дерева. Если не задано, дерево будет расти до тех пор, пока все листья не будут чистыми или пока в каждом листе не останется меньше min_samples_split образцов.
- min_samples_split: Минимальное количество образцов, необходимых для разбиения узла. По умолчанию значение равно 2.
- min_samples_leaf: Минимальное количество образцов в листе. По умолчанию значение равно 1.
- max_features: Максимальное количество признаков, используемых для поиска наилучшего разбиения. Возможные значения: None (используются все признаки), "auto", "sqrt", "log2" или целое число.
- random_state: Случайное состояние для контролирования случайности разбиений и инициализации алгоритма [7-10].

2.1.4 Пример построения дерева решений на табличных данных

Рассмотрим пример, который демонстрирует процесс построения и оценки модели дерева решений для классификации цветков из датасета iris (программный код представлен в приложении А).

Импортируются необходимые библиотеки и модули: pandas для работы с данными, набор данных iris из sklearn, функции для разделения данных и оценки модели, модуль для создания и визуализации дерева решений, а также модуль для метрик.

Загружается встроенный датасет iris и выводится его содержимое.

Данные и целевые значения разделяются на обучающую и тестовую выборки в соотношении 80/20 с фиксированным случайным состоянием для воспроизводимости.

Создается объект классификатора дерева решений с максимальной глубиной 2, который затем обучается на обучающем наборе данных.

Визуализируется обученное дерево решений с указанием названий признаков, и график отображается.

Модель предсказывает метки для тестового набора данных, затем вычисляется и выводится точность модели, а также строится и выводится матрица неточностей (confusion matrix), показывающая качество предсказаний.

В результате получаем построенное дерево решений (рис. 2.4).

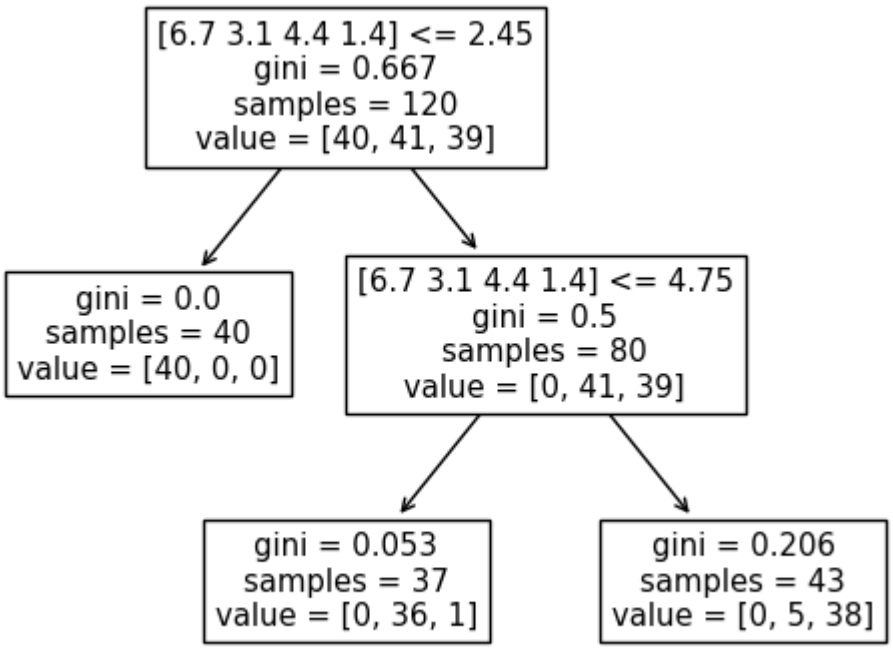

Рисунок 2.4 – Дерево решений

И получаем такой вывод результатов:

0.9666666666666667

[[10 0 0]

[0 8 1]

[0 0 11]]

Точность модели составляет примерно 0.967 (или 96.67%). Это означает, что 96.67% предсказаний модели верны. Высокая точность указывает на то, что модель хорошо справляется с задачей классификации данного набора данных.

Матрица неточностей показывает, как предсказания модели соотносятся с истинными метками классов. Строки представляют истинные классы, а столбцы - предсказанные классы.

- Первая строка [10 0 0] означает, что из 10 образцов, которые на самом деле принадлежат к классу 0, все 10 были правильно классифицированы как класс 0.
- Вторая строка [0 8 1] означает, что из 9 образцов, которые на самом деле принадлежат к классу 1, 8 были правильно классифицированы как класс 1, а 1 образец был ошибочно классифицирован как класс 2.
- Третья строка [0 0 11] означает, что из 11 образцов, которые на самом деле принадлежат к классу 2, все 11 были правильно классифицированы как класс 2.

2.2 Метод случайного леса

2.2.1 Описание метода

Случайный лес (Random Forest) – это один из самых популярных алгоритмов ансамблевого обучения, который работает путем создания множества решающих деревьев (Decision Trees) и объединения их результатов для получения более точного и устойчивого предсказания. Он был предложен Лео Брейманом в 2001 году и сочетает в себе две ключевые концепции:

- **Bagging (Bootstrap Aggregating)** – обучение каждого дерева на случайной выборке с возвратом (bootstrap) из обучающего набора данных. Это позволяет снизить дисперсию модели и уменьшить вероятность переобучения.
- **Случайный выбор признаков** – для каждого разбиения узла в дереве случайно выбирается подмножество признаков, что снижает корреляцию между деревьями и делает модель более устойчивой к шуму в данных.

Метод случайного леса применяется как для задач классификации, так и для регрессии. В случае классификации каждое дерево принимает решение относительно того, к какому классу относится наблюдение, и финальный результат выбирается путем голосования (majority voting) среди всех деревьев. В случае регрессии выходное значение получается путем усреднения предсказаний отдельных деревьев, что позволяет получить более стабильные результаты.

Алгоритм построения случайного леса:

1. **Генерация обучающих выборок:**

- Из исходного набора данных случайным образом выбираются подвыборки с возвратом (bootstrap).
- Размер каждой выборки примерно равен размеру исходного набора, но некоторые наблюдения могут встречаться несколько раз, а некоторые – не попадать в выборку вовсе.

2. **Построение множества деревьев решений:**

- Для каждой bootstrap-выборки строится отдельное дерево решений.
- При каждом разбиении узла случайным образом выбирается подмножество признаков, из которых выбирается лучший для разбиения.
- Глубина деревьев либо фиксируется заранее, либо определяется по критериям остановки (например, минимальное количество объектов в листе).

3. **Формирование итогового предсказания:**

- Для классификационных задач применяется голосование по большинству: класс, который предсказало большее число деревьев, становится финальным предсказанием.
- Для регрессионных задач усредняются прогнозируемые значения всех деревьев.

Основные преимущества метода:

- **Устойчивость к переобучению** за счет усреднения результатов множества деревьев, что снижает влияние выбросов и шумов в данных.
- **Эффективность при работе с большими наборами данных**, включая задачи с большим количеством признаков, где метод случайного выбора признаков помогает избежать избыточной корреляции между деревьями.
- **Гибкость** – метод применяется в самых разных задачах, включая медицинские исследования, финансы, обработку изображений, прогнозирование клиентского поведения и многие другие области.
- **Работа с пропущенными значениями** – метод случайного леса может обрабатывать пропущенные данные, используя стратегии, такие как заполнение пропусков на основе наиболее частых значений среди деревьев.

- **Высокая интерпретируемость по сравнению с нейронными сетями** – хотя отдельные деревья могут быть сложными, анализ значимости признаков позволяет оценить вклад каждого параметра в предсказания модели.

Недостатки метода:

Несмотря на многочисленные преимущества, у метода случайного леса есть и некоторые ограничения:

- **Требовательность к вычислительным ресурсам** – при большом количестве деревьев и признаков обучение может занимать значительное время.
- **Сложность интерпретации при большом количестве деревьев** – хотя отдельные деревья решений легко визуализировать, ансамбль из сотен деревьев становится сложным для анализа.
- **Неоптимальность по скорости предсказаний** – для больших моделей случайного леса предсказания могут выполняться медленнее по сравнению с более простыми моделями, такими как логистическая регрессия или отдельные деревья решений.

Метод случайного леса – это инструмент машинного обучения, который широко применяется в задачах классификации и регрессии. Его способность строить ансамбль деревьев решений позволяет повысить точность предсказаний и снизить переобучение. Однако следует учитывать вычислительные затраты при работе с большими объемами данных. В целом, случайный лес остается одним из наиболее популярных и эффективных методов машинного обучения, активно используемым в различных областях науки и бизнеса.

2.2.2 Метод Random Forest библиотеки scikit-learn языка Python

Библиотека scikit-learn является одной из самых популярных библиотек для машинного обучения в языке Python. Она предоставляет удобный интерфейс для работы с различными алгоритмами, включая метод случайного леса (Random Forest). В scikit-learn метод случайного леса реализован в виде классов RandomForestClassifier (для задач классификации) и RandomForestRegressor (для задач регрессии).

Основные параметры метода Random Forest в scikit-learn:

- **n_estimators** – количество деревьев в лесу.
- **criterion** – функция для оценки качества разбиений ('gini' или 'entropy' для классификации, 'mse' или 'mae' для регрессии).
- **max_depth** – максимальная глубина деревьев.
- **min_samples_split** – минимальное количество образцов, необходимое для разбиения узла.
- **min_samples_leaf** – минимальное количество образцов в листе.
- **max_features** – количество признаков, используемых при построении каждого дерева.

Пример использования метода случайного леса на языке Python представлен в Приложении В.

Метод случайного леса в scikit-learn обладает высокой производительностью. В приведенном примере модель обучается на данных Iris Dataset, а затем оценивается на тестовой выборке. Итоговая точность позволяет судить о качестве предсказаний.

На рисунке 3 представлена визуализация дерева решений (программный код представлен в приложении Б).

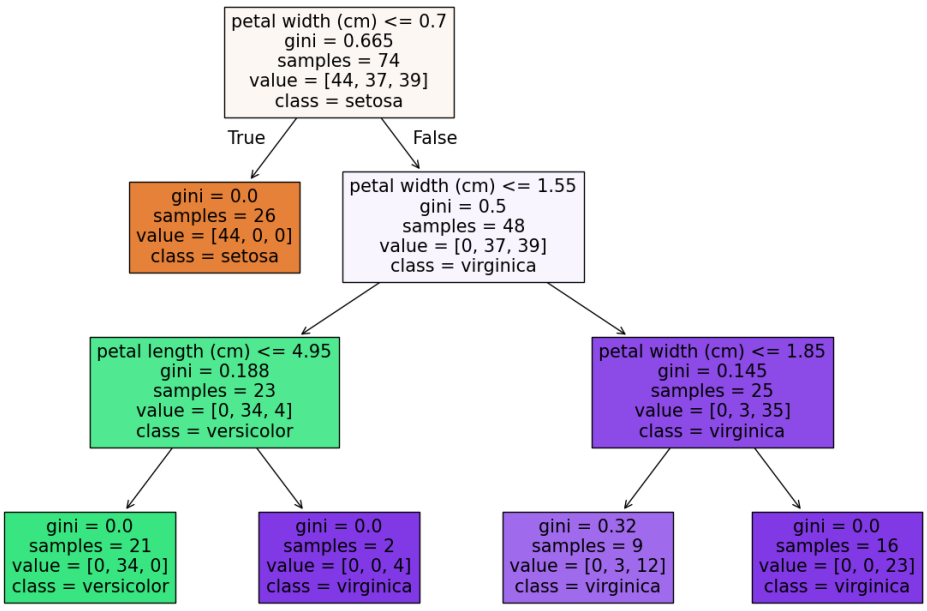

Рисунок 3 – Визуализация дерева решений методом случайного леса

3 ПРОГНОЗИРОВАНИЕ ОТТОКА КЛИЕНТОВ БАНКА

3.1 Описание набора данных «Bank Churn Dataset»

Набор для прогнозирования оттока клиентов банка ABC Multistate представляет собой набор данных 165034 клиентов, а также их статус (клиент покинул банк в течение какого-то периода или нет). Набор находится в свободном доступе в kaggle – системы хранения наборов данных, а также социальной сети для специалистов в области искусственного интеллекта [11]. Описание переменных представлено в табл. 1. В ходе первичного анализа датасета, установили, что переменные «id», «CustomerId», «Surname» не являются значимыми, т.е. не могут существенным образом повлиять на значение целевой переменной «Exited». Данные о первых пяти пациентах представлены в табл. 2.

Все процедуры загрузки и обработки данных, а также построения, обучения дерева решений и анализа результатов проводились средствами библиотек языка Python в среде для разработки и выполнения программного кода Google Colaboratory [12].

Таблица 3 – Описание переменных набора данных

Переменная	Тип данных	Описание	Значимость
id	int64 – целое число		Не значимо
CustomerId	int64 – целое число		Не значимо
Surname	object – текстовая переменная		Не значимо
CreditScore	int64 – целое число	Кредитный рейтинг клиента	Значимо
Geography	object – текстовая переменная	Страна проживания (France, Germany, Spain)	Значимо
Gender	object – текстовая переменная	Пол (Female, Male)	Значимо
Age	float64 – число с плавающей точкой		Значимо

Продолжение таблицы 3

Tenure	int64 – целое число	Срок действия депозита	Значимо
Balance	float64 – число с плавающей точкой	Текущий счет клиента	Значимо
NumOfProducts	int64 – целое число	Количество банковских счетов, связанных с продуктами, которые есть у клиента	Значимо
HasCrCard	float64 – число с плавающей точкой	Наличие у клиента кредитной карты через банк («Да» = 1, «Нет» = 0)	Значимо
IsActiveMember	float64 – число с плавающей точкой	Является ли клиент активным участником? («Да»=1, «Нет»=0)	Значимо
EstimatedSalary	float64 – число с плавающей точкой	Предполагаемая зарплата заказчика	Значимо
Exited (таргетная переменная)	**int64 – целое число**	**Покидал ли клиент банк в течение последних 6 месяцев? («Да»=1, «Нет»=0)**	**Значимо**

На рисунке 4 показан график распределения таргетной переменной, на графике видно, что большинство клиентов (0) остались в банке, тогда как меньшая часть клиентов (1) покинули его.

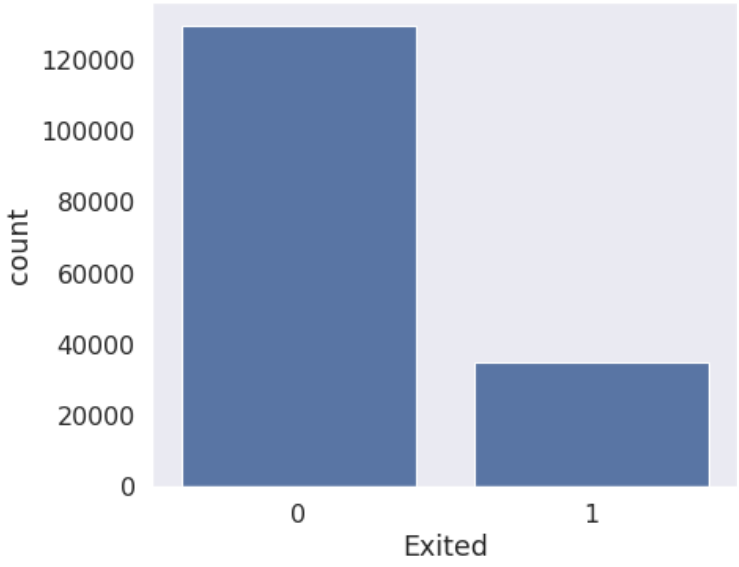

Рисунок 4 – График распределения таргетной переменной

График показывает распределение возрастов клиентов, где видно, что большинство клиентов находятся в возрасте около 35–40 лет.

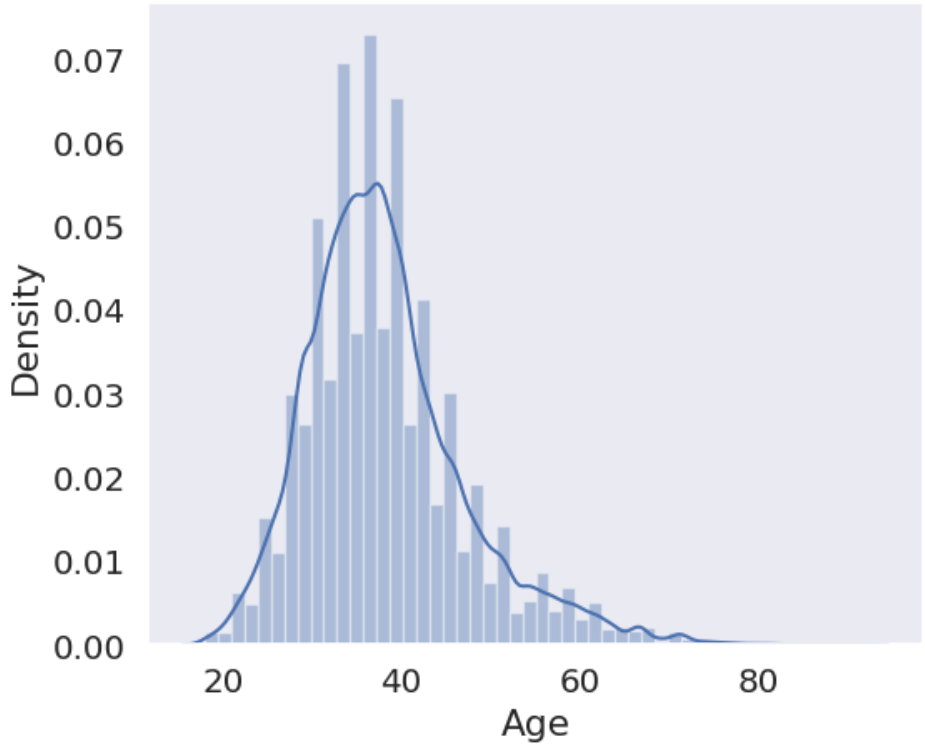

Рисунок 5 – График распределения возрастов клиентов

На рисунке 6 показано соотношение стран проживания клиентов банка.

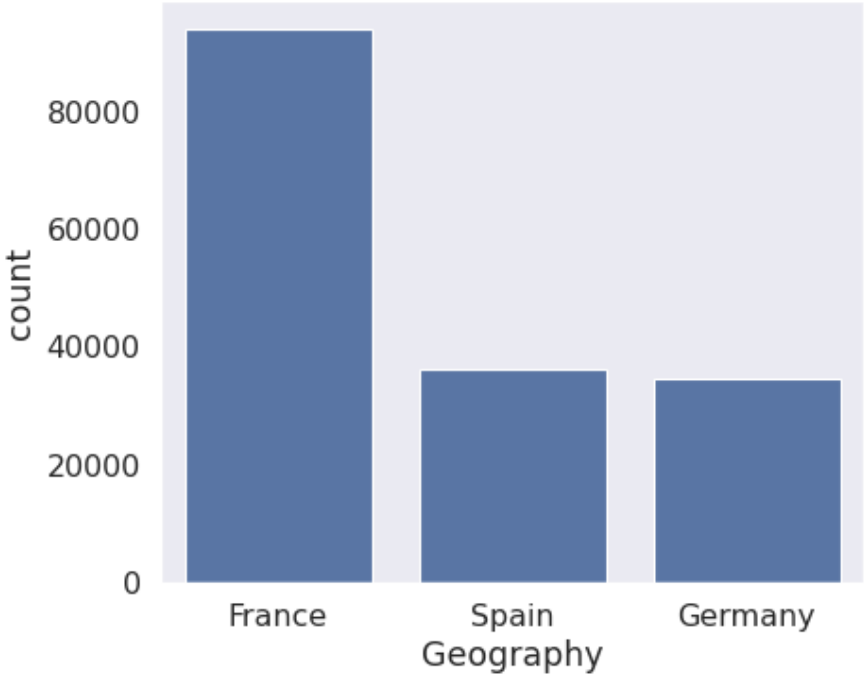

Рисунок 6 – График распределения стран проживания клиентов

3.2 Предварительная обработка и анализ набора данных

Перед построением и обучением алгоритма были выполнены стандартные процедуры обработки имеющихся данных. Из набора были удалены неинформативные столбцы «id», «CustomerId», «Surname»; тем самым, осталось 10 признаковых переменных, из которых 8 являются числовыми переменными, 2 – категориальными. Таргетная переменная (Exited) является бинарной переменной.

Далее из набора данных были удалены 123 повторяющиеся строки. Далее категориальные переменные с помощью функции *LabelEncoder* были перекодированы в числовые. Следует отметить, что процедуру перекодировки категориальных переменных можно осуществлять с помощью других подходов, например, с помощью метода «one hot encoding», однако на обучение в нашем случае способ перекодирования существенно не влияет. Корреляционных анализ (см. рис. 3) показал, что между признаками отсутствует корреляционная зависимость.

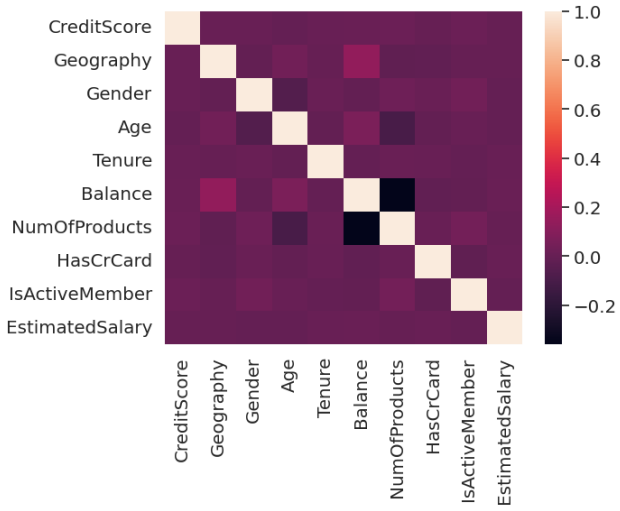

Рисунок 7 – Визуализация матрицы корреляции признаковых переменных

Полученный набор содержит 164 911 примеров, из которых количество примеров, в которых метка «Exited» равна 1, равно 34909. Тем самым, набор является несбалансированным, и это нужно учитывать при анализе качества обучения.

3.3 Модель на основе дерева решений

3.3.1 Описание настройки гиперпараметров

Алгоритм работает путём разделения данных на подмножества на основе значений признаков, выбираемых таким образом, чтобы максимизировать чистоту (гомогенность) подмножеств. Официальная документация метода в библиотеке scikit-learn представлена в [10]. Основные параметра метода, влияющие на качество обучения, следующие:

- максимальная глубина дерева (max_depth);
- минимальное количество объектов в листе (min_samples_leaf);
- минимальное количество объектов для разделения (min_samples_split);
- критерий разделения (criterion);
- максимальное количество признаков (max_features).

Положим во всех экспериментах настройки параметров по умолчанию: min_samples_leaf = 1, min_samples_split = 2, criterion = «gini», max_features = None. Будем изменять глубину дерева и фиксировать метрики качества обучения.

3.3.2 Результаты моделирования и их анализ

Для анализа данных результатов обучения при решении задачи бинарной классификации используются следующие метрики качества [13]:

1) accuracy (точность): показывает долю правильно классифицированных наблюдений от общего числа (полезна, когда классы сбалансированы);

2) recall (полнота): показывает долю истинно положительных результатов среди всех меток класса, которые были определены как положительные;

3) precision (точностъ): показывает долю истинно положительных результатов среди всех положительных меток;

4) F1-score (F1-мера): среднее гармоническое между Precision и Recall;

5) Area Under the ROC Curve (ROC-кривая): численно равна площади под ROC-кривой (Receiver Operating Characteristic). Численная оценка качества модели, учитывающая соотношение истинно положительных и ложно положительных результатов.

Выбор метрик зависит от конкретной задачи и требований к модели. В нашей задаче выявления клиентов, которые могут покинуть банк, важен такой

аспект, как минимизация ложноотрицательных результатов, поэтому при анализе качества обучения следует обращать особое внимание на метрику «recall».

Разобьём имеющийся набора на тренировочный и тестовый в соотношении 80/20. Будем сравнивать результаты обучения при разной глубине деревьев. Результаты сравнения представлены в табл. 3. Как показали эксперименты, оптимальной в данном случае точности можно добиться при глубине дерева, равной 6; дальнейшее увеличение не приведет к существенно лучшему результату обучения. Максимальное значение важной метрики recall, равное, 0.53, можно добиться при глубине дерева 5. На рис. 8 представлено дерево решений глубиной 3 (для наглядности).

Для устранения несбалансированности можно использовать метод «SMOTE» (Synthetic Minority Over-sampling Technique) [14]. Это метод обработки несбалансированных данных, который используется для увеличения количества образцов в меньшинстве классов. Несбалансированные данные возникают тогда, когда одна категория (класс) встречается гораздо реже, чем другая. Это может затруднить обучение моделей машинного обучения, так как они склонны обучаться преимущественно на большинстве классов, игнорируя меньшинство.

В таблице 4 представлены результаты обучения. На рис. 9 представлено соответствующее дерево решения глубиной 3.

Таблица 4 – Значения метрик качества обучения

Глубина дерева	Класс	Метрика				
		precision	recall	f1-score	accuracy	AUC
2	0	**0.86**	**0.93**	**0.89**	**0.83**	**0.79**
	1	**0.62**	**0.45**	**0.52**		
3	0	**0.86**	**0.97**	**0.91**	**0.85**	**0.84**
	1	**0.79**	**0.41**	**0.54**		
4	0	**0.87**	**0.95**	**0.91**	**0.85**	**0.86**
	1	**0.73**	**0.48**	**0.58**		

Продолжение таблицы 4

5	0	**0.88**	**0.94**	**0.91**	**0.86**	**0.87**
	1	**0.71**	**0.53**	**0.61**		
6	0	**0.88**	**0.85**	**0.91**	**0.86**	**0.87**
	1	**0.74**	**0.50**	**0.60**		

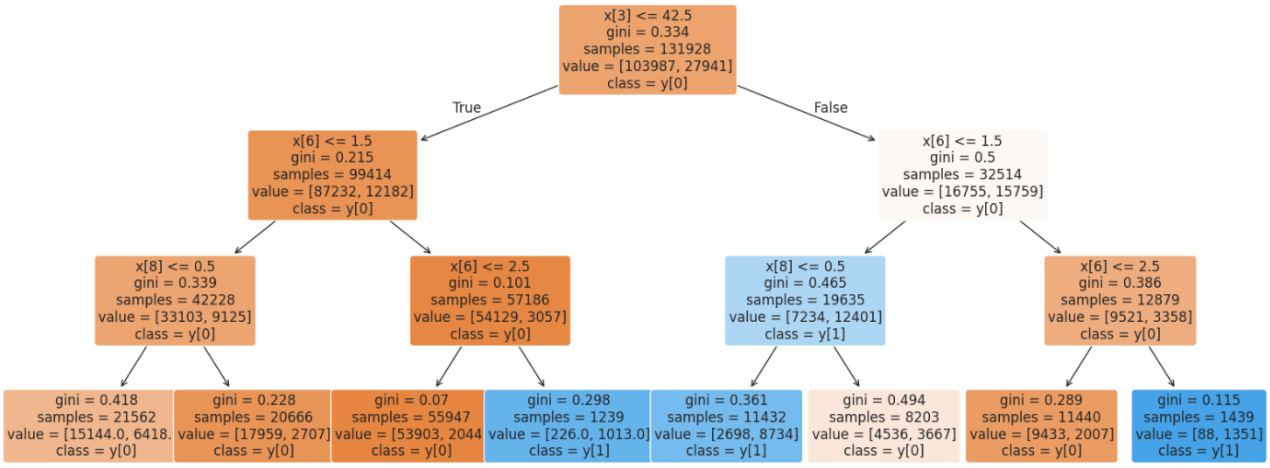

Рисунок 8 – Дерево решений глубиной 3

Таблица 5 – Значения метрик качества обучения (с методом «SMOTE»)

Глубина дерева	Класс	Метрика				
		precision	recall	f1-score	accuracy	AUC
2	0	**0.78**	**0.83**	**0.80**	**0.79**	**0.82**
	1	**0.81**	**0.76**	**0.79**		
3	0	**0.82**	**0.77**	**0.80**	**0.80**	**0.87**
	1	**0.79**	**0.84**	**0.81**		
4	0	**0.79**	**0.89**	**0.84**	**0.83**	**0.91**
	1	**0.87**	**0.77**	**0.82**		
5	0	**0.86**	**0.83**	**0.84**	**0.85**	**0.92**
	1	**0.84**	**0.86**	**0.85**		
6	0	**0.84**	**0.89**	**0.87**	**0.86**	**0.94**
	1	**0.89**	**0.83**	**0.86**		

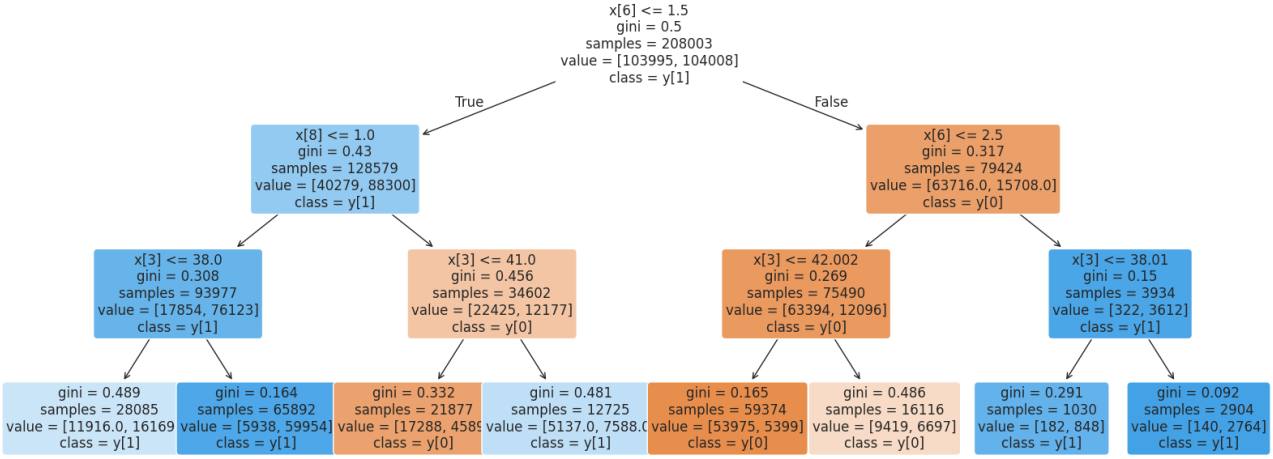

Рисунок 9 – Дерево решений глубиной 3 (с методом «SMOTE»)

На рисунке 10 представлена матрица ошибок для метода деревья решений при глубине дерева 6:

23245 – количество клиентов, которые остались (класс 0) и модель правильно их предсказала как оставшихся;

2762 – клиентов, которые остались (класс 0), но модель ошибочно предсказала их уход (ложные положительные, False Positive);

4471 – клиентов, которые ушли (класс 1), но модель ошибочно предсказала, что они останутся (ложные отрицательные, False Negative);

22523 – количество клиентов, которые ушли (класс 1) и модель правильно предсказала их уход.

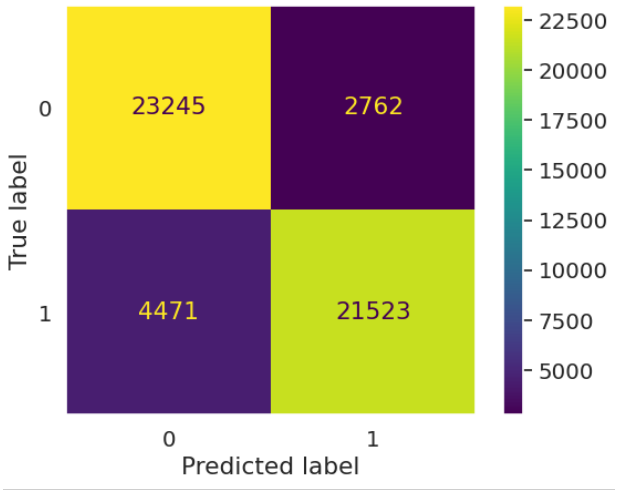

Рисунок 10 – Матрица ошибок

3.4 Модель на основе Random Forest

3.4.1 Описание настройки гиперпараметров

Алгоритм Random Forest строит множество деревьев решений и объединяет их результаты для улучшения качества предсказания и уменьшения переобучения. Модель работает за счёт «усреднения» предсказаний отдельных деревьев, что делает её более устойчивой к шуму в данных.

К основным гиперпараметрам, влияющим на обучение модели, относятся:

- n_estimators: количество деревьев в лесу (большее значение обычно повышает качество, но увеличивает время обучения);
- max_depth: максимальная глубина дерева (помогает контролировать переобучение);
- min_samples_split: минимальное количество образцов для разделения узла;
- min_samples_leaf: минимальное количество образцов в листовом узле;
- criterion: функция для измерения качества разбиения («gini» или «entropy»);
- max_features: максимальное количество признаков, которое используется при построении одного дерева.

В базовом варианте будем использовать параметры по умолчанию: min_samples_split=2, min_samples_leaf=1, criterion=”gini”, max_features=”sqrt”.

Будем изменять количество деревьев (n_estimators) и глубину дерева (max_depth), фиксируя результаты обучения.

3.4.2 Результаты моделирование и их анализ

Анализ качества модели будет проводиться с использованием тех же метрик, что и для дерева решений:

- accuracy (точность),
- recall (полнота),
- precision (точность положительных предсказаний),
- F1-score (среднее гармоническое precision и recall),

• AUC (площадь под ROC-кривой).

Поскольку задача заключается в предсказании ухода клиента из банка, особенно важной является метрика recall, которая показывает, насколько хорошо модель находит клиентов, которые действительно уйдут.

Таблица 6 – Значения метрик качества обучения методом Random Forest

Глубина дерева	Класс	Метрика				
		precision	recall	f1-score	accuracy	AUC
2	0	**0.85**	**0.83**	**0.83**	**0.84**	**0.92**
	1	**0.83**	**0.85**	**0.84**		
3	0	**0.86**	**0.84**	**0.85**	**0.85**	**0.93**
	1	**0.84**	**0.86**	**0.85**		
4	0	**0.87**	**0.85**	**0.86**	**0.86**	**0.94**
	1	**0.85**	**0.87**	**0.86**		
5	0	**0.87**	**0.88**	**0.88**	**0.87**	**0.95**
	1	**0.88**	**0.87**	**0.87**		
6	0	**0.88**	**0.89**	**0.88**	**0.88**	**0.95**
	1	**0.89**	**0.88**	**0.88**		

Максимальное значение равное 0.88 метрики recall получили при глубине дерева 6.

На рисунке 11 представлена матрица ошибок для метода случайного леса при глубине дерева 6:

23204 – количество клиентов, которые остались (класс 0) и модель правильно их предсказала как оставшихся.

2803 – клиентов, которые остались (класс 0), но модель ошибочно предсказала их уход (ложные положительные, False Positive).

3247 – клиентов, которые ушли (класс 1), но модель ошибочно предсказала, что они останутся (ложные отрицательные, False Negative).

22747 – количество клиентов, которые ушли (класс 1) и модель правильно предсказала их уход.

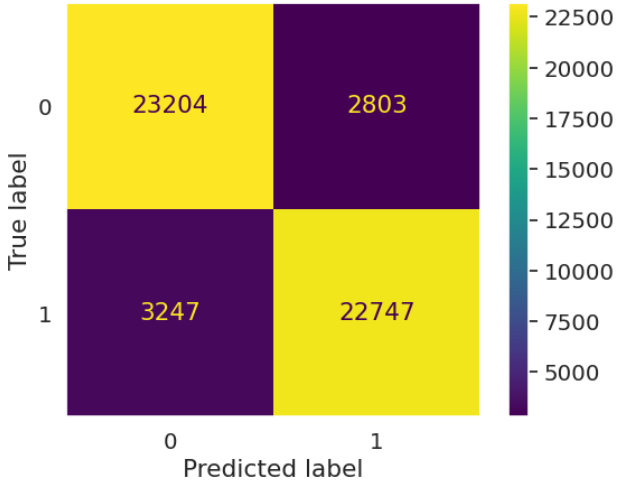

Рисунок 11 – Матрица ошибок

Таблица 7 – Значения метрик качества обучения методом Random Forest с методом SMOTE

Глубина дерева	Класс	Метрика				
		precision	recall	f1-score	accuracy	AUC
2	0	**0.85**	**0.83**	**0.83**	**0.84**	**0.93**
	1	**0.84**	**0.85**	**0.84**		
3	0	**0.86**	**0.84**	**0.85**	**0.85**	**0.94**
	1	**0.84**	**0.85**	**0.85**		
4	0	**0.87**	**0.85**	**0.86**	**0.86**	**0.95**
	1	**0.86**	**0.87**	**0.86**		
5	0	**0.87**	**0.88**	**0.88**	**0.87**	**0.95**
	1	**0.88**	**0.87**	**0.87**		
6	0	**0.88**	**0.89**	**0.88**	**0.89**	**0.97**
	1	**0.89**	**0.88**	**0.88**		

Максимальное значение равное 0.89 метрики recall получили при глубине дерева 6.

На рисунке 12 представлена матрица ошибок для метода случайного леса с методом SMOTE при глубине дерева 6:

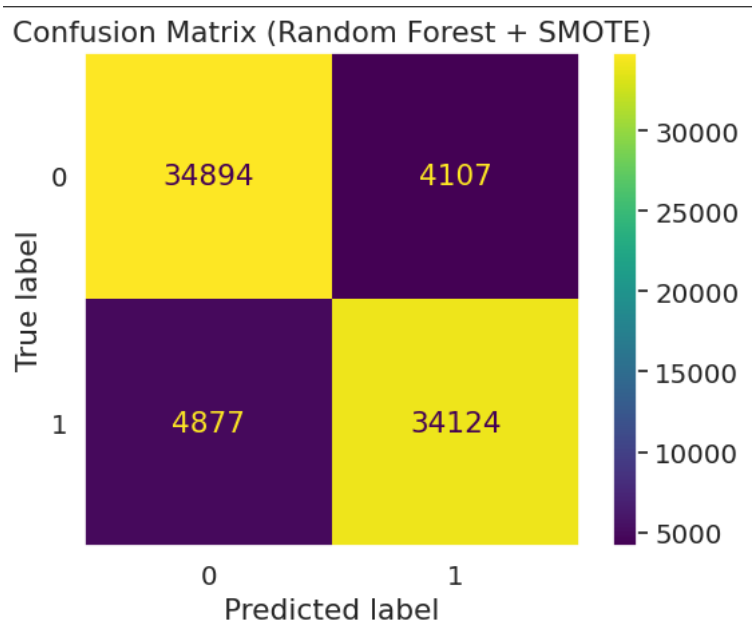

Рисунок 12 – Матрица ошибок

3.5 Сравнение построенных моделей

Для решения задачи бинарной классификации были построены две модели: на основе дерева решений и случайного леса. Оценка качества моделей проводилась по метрикам precision, recall, f1-score, accuracy и AUC.

Анализ результатов показал следующее:

- **Модель дерева решений без применения SMOTE** достигает максимальной точности (accuracy) 86% при глубине дерева 5-6. При этом значение метрики recall для целевого класса (ушедшие клиенты) составляет 0.53 для глубины 5. Значение AUC составляет 0.87.
- **Модель дерева решений с использованием SMOTE** демонстрирует рост качества: метрика recall увеличивается до 0.86 при глубине дерева 5, а точность (accuracy) возрастает до 85%. Площадь под ROC-кривой (AUC) также увеличивается до 0.92.

- **Модель случайного леса** при глубине деревьев 5-6:
  - Accuracy составляет 87–88%.
  - Recall достигает 0.88.
  - F1-score для обоих классов составляет около 0.87–0.88.
  - AUC равен 0.95.
- **Модель случайного леса с методом SMOTE** показывает лучшие результаты по всем основным метрикам. При глубине деревьев 5-6:
  - Accuracy составляет 87–89%.
  - Recall для ушедших клиентов достигает 0.88.
  - F1-score для обоих классов составляет около 0.87–0.88.
  - AUC достигает 0.97, что указывает на высокую способность модели различать классы.

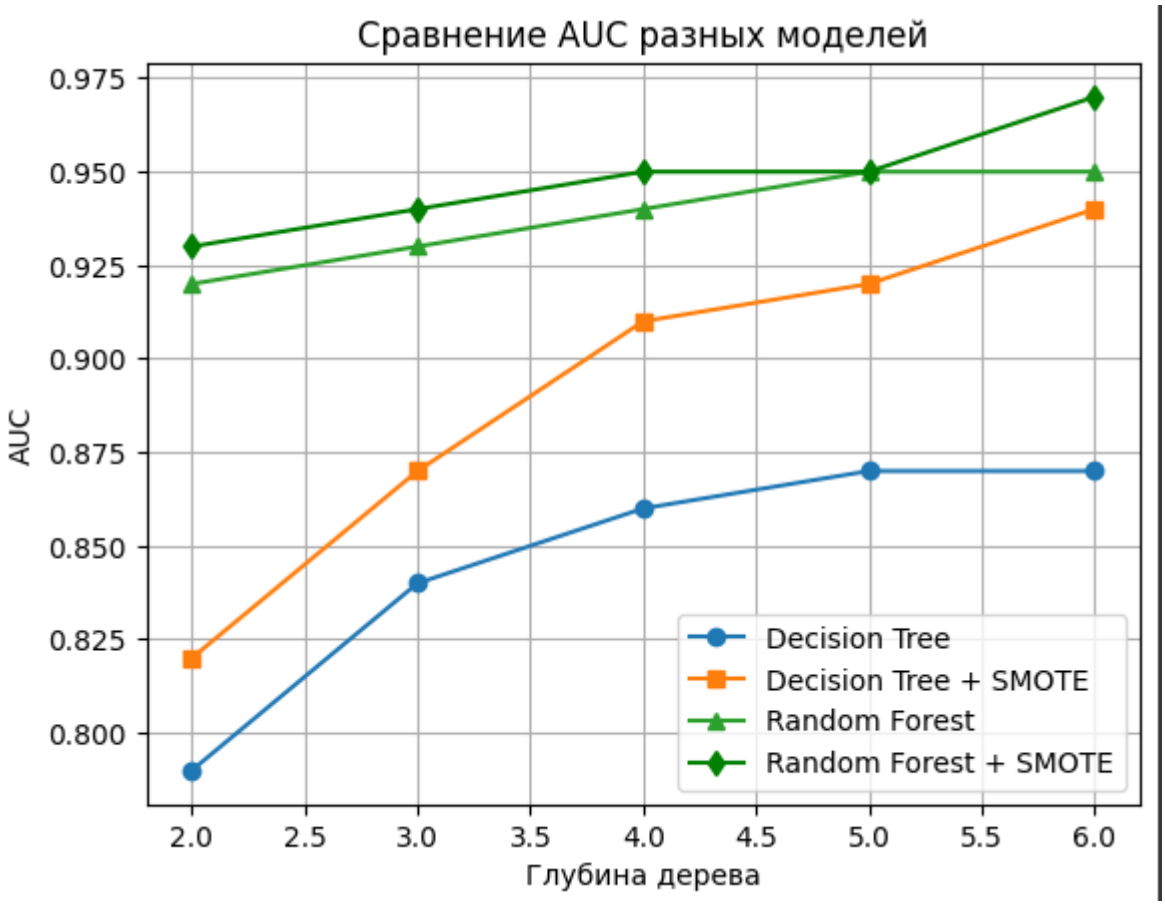

Рисунок 13 – Сравнение AUC разных моделей

Таким образом, случайный лес значительно превосходит одиночное дерево решений как по точности предсказаний, так и по полноте - ключевой метрике для данной задачи. Особенно стоит отметить улучшение метрики recall, которая критична при выявлении клиентов, склонных к уходу.

ЗАКЛЮЧЕНИЕ

В рамках данной работы были исследованы методы прогнозирования оттока клиентов банка с использованием алгоритмов дерева решений и случайного леса. Были построены, настроены и сравнены модели машинного обучения, что позволило оценить их качество и эффективность на основе различных метрик.

В процессе выполнения работы была проведена теоретическая подготовка: рассмотрены основные понятия искусственного интеллекта и машинного обучения, проанализированы этапы развития и применения методов искусственного интеллекта в банковской сфере.

На практике были разработаны две модели: на основе алгоритма Decision Tree и на основе ансамблевого метода Random Forest. Модели были обучены на наборе данных «Bank Churn Dataset», который предварительно был обработан, включая устранение пропусков, кодирование категориальных признаков и балансировку классов методом SMOTE.

В результате моделирования установлено, что модель случайного леса продемонстрировала более высокие показатели качества по сравнению с моделью дерева решений. По итогам анализа метрик (precision, recall, f1-score, accuracy, AUC) модель Random Forest при глубине деревьев 6 достигла наивысших значений точности и полноты, что подтверждается результатами матрицы ошибок и ROC-кривой.

Таким образом, модель случайного леса с методом SMOTE оказалась более эффективной для задачи прогнозирования оттока клиентов банка, обеспечивая лучшее соотношение между точностью и полнотой. Использование ансамблевых методов, таких как Random Forest, позволяет повысить надежность и устойчивость моделей машинного обучения, что особенно важно в банковской сфере при решении задач классификации.

Практическая значимость данной работы заключается в возможности применения построенных моделей для реального прогнозирования оттока клиентов в банковской сфере. Модели машинного обучения, такие как Decision Tree и

Random Forest, позволяют на основе имеющихся клиентских данных выявлять группы клиентов, склонных к уходу, и разрабатывать персонализированные стратегии их удержания. Это способствует повышению лояльности клиентов, снижению потерь финансовых ресурсов и увеличению конкурентоспособности банковских учреждений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Искусственный интеллект. Инноватика: учеб. Пособие / Ю. А. Антохина [и др.] – Санкт-Петербург: ГУАП, 2023. – 320 с. – Режим доступа: https://e.lanbook.com/book/341003

2 Остроух, А.В. Системы искусственного интеллекта: моногр. / А. В. Остроух, Н. Е. Суркова. – 3-е изд., стер. – Санкт-Петербург: Лань, 2023. – 228 с. – Режим доступа: https://e.lanbook.com/book/310199

3 Толмачев, С.Г. Нейросетевые методы обработки информации: учебное пособие / С.Г. Толмачев. – Санкт-Петербург: БГТУ "Военмех" им. Д.Ф. Устинова, 2021. – 103 с. – Режим доступа: https://e.lanbook.com/book/220238

4 habr.com: Нейросеть – обучение без учителя. Метод Policy Gradient [Электронный ресурс]: офиц. сайт. – 26.05.2006 – Режим доступа: https://habr.com/ru/articles/506384/

5 Постолит, А.В. Основы искусственного интеллекта в примерах на Python: самоучитель / А. В. Постолит. – СПб.: БХВ-Петербург, 2021. – 448 с.

6 habr.com: Возможности Искусственного Интеллекта в 2023 году. Эндрю Ын [Электронный ресурс]: офиц. сайт. – 26.05.2006 – Режим доступа: https://habr.com/ru/companies/serverspace/articles/776954/

7 Бастиан, Ш. Крупномасштабное машинное обучение вместе с Python / Ш.Бастиан. – М.: ДМК Пресс, 2017. – 358 с.

8 Саммерфильд, Марк. Python на практике / Марк Саммерфильд. – М.: ДМК Пресс, 2014. – 338 с.

9 Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем / Орельен Жерон. – М.: Вильямс, 2018. – 688 с.

10 scikit-learn.org: sklearn.tree.DecisionTreeClassifier [Электронный ресурс]: офиц. сайт. – Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

11 Bank Churn Dataset [Электронный ресурс]. URL: https://www.kaggle.com/datasets/rangalamahesh/bank-churn (дата обращения: 20.05.2024)

12 Google Colaboratory [Электронный ресурс]. URL: https://colab.google (дата обращения: 09.10.2023)

13 Метрики классификации и регрессии [Электронный ресурс]. URL: https://education.yandex.ru/handbook/ml/article/metriki-klassifikacii-i-regressii (дата обращения: 28.11.2024)

14 SMOTE [Электронный ресурс]. URL: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (дата обращения: 29.10.2024)

15 Han Yi & Chen Jinhao & Dou Meitao & Wang Jiahong & Feng Kangxiao. The Impact of Artificial Intelligence on the Financial Services Industry // Academic Journal of Management and Social Sciences. 2023. 2. 83-85. DOI: 10.54097/ajmss.v2i3.8741

16 Singh Pahul & Anik Fahim & Senapati Rahul & Sinha Arnav & Sakib Nazmus & Hossain Eklas. Investigating customer churn in banking: A machine learning approach and visualization app for data science and management // Data Science and Management. 2023. 7. DOI: 10.1016/j.dsm.2023.09.002

17 Esmaeilpour Charandabi Sina. Prediction of Customer Churn in Banking Industry. 2020. DOI: 10.13140/RG.2.2.24177.56167

18 DecisionTreeClassifier [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (дата обращения: 28.09.2024)

19 Берман, К. Основы Python для Data Science / К. Берман. — СПб.: Питер, 2023. — 272 с.

20 Чистяков, С. П. Случайные леса: обзор / С. П. Чистяков // Труды Карельского научного центра Российской академии наук. – 2013. – № 1. – С. 117-136. – EDN PZBGBL.

Публикации автора по теме работы

21 Четыркин Д.Н. Модель оттока клиентов банка на основе дерева решений / Д.Н. Четыркин, Н.Н. Максимова // Материалы V национальной научной конференции «Far East Math – 2024» (Хабаровск, 2–7 декабря 2024 г.) / [отв. за вып. Е. Г. Агапова] – Хабаровск: Издательство ТОГУ, 2025. – С. 271-276.

22 Четыркин, Д. Н. Деревья принятия решений в задаче прогнозирования оттока клиентов банка / Д.Н. Четыркин // День науки: Материалы XXXIV научной конференции Амурского государственного университета, Благовещенск, 13 марта 2025 года. – Благовещенск: Амурский государственный университет, 2025. – *в печати*

ПРИЛОЖЕНИЕ А
Код примера построения дерева решений

```
# импортирование библиотек и модулей
import pandas
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn import metrics
# загрузка данных
iris = datasets.load_iris()
print(iris)
# разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2,
random_state=42)
# создание и обучение модели дерева решений
dtree = DecisionTreeClassifier(max_depth=2)
dtree = dtree.fit(X_train, y_train)
# визуализация дерева решений
tree.plot_tree(dtree, feature_names=iris.feature_names)
plt.show()
# прогнозирование и оценка модели
y_pred = dtree.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

ПРИЛОЖЕНИЕ Б

Пример использования метода случайного леса на языке Python

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
# Загрузка данных
iris = load_iris()
X, y = iris.data, iris.target
# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=42)
# Создание и обучение модели
rf = RandomForestClassifier(n_estimators=100, max_depth=3, random_state=42)
rf.fit(X_train, y_train)
# Предсказание и оценка качества модели
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Точность модели: {accuracy:.2f}')
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
plt.figure(figsize=(15,10))
plot_tree(rf.estimators_[0], filled=True, feature_names=iris.feature_names,
class_names=iris.target_names)
plt.show()
```

ПРИЛОЖЕНИЕ В
Код модели на основе Decision Tree

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from sklearn import metrics
from numpy import *
from sklearn. model_selection import train_test_split
import seaborn as sns
sns.set(style='dark', font_scale=1.3)
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
import pickle
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('/content/train.csv')
print(df.shape)
df.head()
df.drop(columns=['id', 'CustomerId','Surname'], axis=1,inplace=True)
df
df.duplicated().sum()
df.drop_duplicates(inplace=True)
disp('Размер преобразованной таблицы:')
df.shape
df.describe()
figure(figsize=(6, 5), dpi=80)
sns.countplot(x='Exited',data=df)
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
X = df[['CreditScore',  'Geography',  'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProd-
ucts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']].copy()
X.loc[:, 'Gender'] = labelencoder.fit_transform(X.loc[:, 'Gender'])
X.loc[:, 'Geography'] = labelencoder.fit_transform(X.loc[:, 'Geography'])
X.head()
X.corr()
sns.heatmap(X.corr())
y = df['Exited']
print(y)
y.value_counts()
# метод SMOTE
smote = SMOTE(sampling_strategy='minority')
X, y = smote.fit_resample(X, y)
y.value_counts()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=10)
X_train.shape[0], X_test.shape[0]
X_train = X_train.to_numpy()
X_test = X_test.to_numpy()
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
dtree = DecisionTreeClassifier(max_depth=3)
dtree = dtree.fit(X_train, y_train)
plt.figure(figsize=(20, 8), dpi=100)
tree.plot_tree(dtree,
               #feature_names=X_train,
               feature_names=X_train.columns if hasattr(X_train, 'columns') else None,
```

```
            class_names=True,
            filled=True,
            rounded=True,
            fontsize=12
           )
plt.show()
y_pred = dtree.predict(X_test)
print(y_test)
print(y_pred)
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
print(classification_report(y_test, y_pred))
# определение метрик
y_pred_proba_log = dtree.predict_proba(X_test)[::, 1]
fpr_log, tpr_log, _ = metrics.roc_curve(y_test, y_pred_proba_log)
auc_log = metrics.roc_auc_score(y_test, y_pred_proba_log)
# построение ROC-кривой
plt.plot(fpr_log, tpr_log, label = 'AUC='+str(auc_log))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 4)
plt.grid()
plt.show()
```

ПРИЛОЖЕНИЕ Г
Код модели на основе Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMa-
trixDisplay, roc_auc_score, roc_curve
# Построение модели Random Forest
rfc = RandomForestClassifier(n_estimators=100, max_depth=6, random_state=10)
rfc.fit(X_train, y_train)
# Предсказания
y_pred_rfc = rfc.predict(X_test)
# Отображение матрицы ошибок
cm_rfc = confusion_matrix(y_test, y_pred_rfc)
disp_rfc = ConfusionMatrixDisplay(confusion_matrix=cm_rfc)
disp_rfc.plot()
plt.show()
print(classification_report(y_test, y_pred_rfc))
# Построение ROC-кривой
y_pred_proba_rfc = rfc.predict_proba(X_test)[:,1]
fpr_rfc, tpr_rfc, _ = roc_curve(y_test, y_pred_proba_rfc)
auc_rfc = roc_auc_score(y_test, y_pred_proba_rfc)
plt.plot(fpr_rfc, tpr_rfc, label = 'AUC=' + str(round(auc_rfc, 2)))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.grid()
plt.show()
```

ПРИЛОЖЕНИЕ Д
Код модели на основе Random Forest с методом SMOTE

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, roc_auc_score, roc_curve
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10, stratify=y)
# Применение SMOTE к тренировочным данным
smote = SMOTE(random_state=10)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
rfc_smote = RandomForestClassifier(n_estimators=100, max_depth=6, random_state=10)
rfc_smote.fit(X_train_smote, y_train_smote)
y_pred_smote = rfc_smote.predict(X_test)
# Матрица ошибок
cm_smote = confusion_matrix(y_test, y_pred_smote)
disp_smote = ConfusionMatrixDisplay(confusion_matrix=cm_smote)
disp_smote.plot()
plt.title("Confusion Matrix (Random Forest + SMOTE)")
plt.show()
# Классификационный отчёт
print("Classification Report (Random Forest + SMOTE):")
print(classification_report(y_test, y_pred_smote))
# ROC-кривая и AUC
y_pred_proba_smote = rfc_smote.predict_proba(X_test)[:,1]
```

```
fpr_smote, tpr_smote, _ = roc_curve(y_test, y_pred_proba_smote)
auc_smote = roc_auc_score(y_test, y_pred_proba_smote)
plt.plot(fpr_smote, tpr_smote, label='Random Forest + SMOTE (AUC=' +
str(round(auc_smote, 2)) + ')')
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC-кривая (Random Forest + SMOTE)')
plt.legend(loc=4)
plt.grid()
plt.show()
```

ПРИЛОЖЕНИЕ Е

Таблица Е.1 – Данные о пяти первых клиентах из набора данных

x	id	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	0	15674932	Okwudilichukwu	668	France	Male	33.0	3	0.0	2	1.0	0.0	181449.97	0
1	1	15749177	Okwudiliolisa	627	France	Male	33.0	1	0.0	2	1.0	1.0	49503.5	0
2	2	15694510	Hsueh	678	France	Male	40.0	10	0.0	2	1.0	0.0	184866.69	0
3	3	15741417	Kao	581	France	Male	34.0	2	148882.54	1	1.0	1.0	84560.88	0
4	4	15766172	Chiemenam	716	Spain	Male	33.0	5	0.0	2	1.0	1.0	15068.83	0