МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

Система прогнозирования оттока клиентов

Студент 23.06.2026 Ю.Е. Митрошин
(подпись, дата) (ФИО полностью)

Руководитель 23.06.2026 Т.Ю. Леонова
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 23.06.2026 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от 24 июня 2026 г. № 49)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ М.А. Митрохин
«07» 04 2026 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Митрошин Юрий Евгеньевич гр. 22ВВП1 факультета ВТ

<Фамилия, имя, отчество полностью>

2.Тема работы Система прогнозирования оттока клиентов

Тема утверждена приказом ПГУ № 347/о от "07" 04 2026 г.

3. Руководитель бакалаврской работы Леонова Т.Ю

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: Обеспечение прогнозирования вероятности отказа клиентов от использования подписного сервиса на основе анализа пользовательской активности с возможностью формирования рекомендаций по удержанию клиентов

Основные функции системы:

1.Загрузка и хранение данных о клиентах и их активности;
2.Предварительная обработка и подготовка данных;
3.Кодирование и нормализация признаков;
4.Обучение модели машинного обучения;
5.Прогнозирование вероятности оттока клиентов;
6.Классификация клиентов по уровню риска ухода;
7.Визуализация результатов анализа и прогнозирования;
8.Формирование отчетов по результатам прогнозирования;
9.Сравнение качества различных моделей машинного обучения;
10.Хранение результатов прогнозирования ианализа.

Технология разработки: объектно-ориентированный и модульный подход с

использованием технологий анализа данных и машинного обучения.
Языки программирования: Python
Среда программирования: PyCharm

5. Перечень подлежащих разработке вопросов:
1.анализ предметной области и постановка задачи;
2.выбор и анализ методов прогнозирования оттока клиентов;
3.проектирование программных средств;
4.разработка и обучение нейросетевой модели;
5. реализация программных средств;
6.тестирование и оценка качества модели;
7.анализ полученных результатов;
8.оформление ВКР

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания "07" 04 2026 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Л.Р. Фисенко
24.06.2026 *(подпись, дата)*

Реферат

Бакалаврская работа содержит 99 листов, 27 рисунков, 6 таблиц, 8 формул, 18 использованных источников и 3 приложения.

ПРОГНОЗИРОВАНИЕ ОТТОКА КЛИЕНТОВ, ПОДПИСНЫЕ СЕРВИСЫ, МАШИННОЕ ОБУЧЕНИЕ, НЕЙРОННЫЕ СЕТИ, МНОГОСЛОЙНЫЙ ПЕРЦЕПТРОН, БИНАРНАЯ КЛАССИФИКАЦИЯ, PYTORCH, АНАЛИЗ ДАННЫХ, ДИСБАЛАНС КЛАССОВ

Объектом исследования является процесс оттока клиентов в подписных сервисах, предметом исследования — методы и модели машинного обучения, позволяющие прогнозировать вероятность ухода клиента на основе данных о его профиле и характере потребления услуг.

Цель работы — разработать и экспериментально исследовать систему прогнозирования оттока клиентов подписного сервиса на основе искусственной нейронной сети, реализованной средствами библиотеки PyTorch.

В ходе работы проведён анализ предметной области и существующих методов прогнозирования оттока, выбран и подготовлен набор данных, выполнен разведочный анализ данных, спроектирована и обучена нейронная сеть, проведено сравнение с базовыми моделями машинного обучения, исследовано влияние балансировки классов, архитектуры сети и порога классификации на качество прогноза, а также проведён анализ важности признаков.

В результате разработана модель, обеспечивающая значение метрики ROC-AUC 0,845 на отложенной тестовой выборке, что соответствует уровню лучших решений для использованного набора данных.

Содержание

Перечень сокращений и обозначений 5
Введение 6
1. Система прогнозирования оттока клиентов 10
1.1 Анализ предметной области 10
1.1.1 Понятие оттока клиентов и показатели подписного бизнеса 10
1.1.2 Значимость прогнозирования оттока 11
1.1.3 Обзор существующих решений 12
1.1.4 Анализ требований к разрабатываемой системе 13
1.1.5 Обзор методов ML для прогнозирования оттока 14
1.1.6 Обоснование выбора нейронной сети 16
1.1.7 Выбор языка программирования 17
1.1.8 Выбор библиотеки машинного обучения 17
1.1.9 Выбор среды разработки 18
1.1.10 Выбор и настройка инструментов разработки 18
1.1.11 Математические основы искусственных нейронных сетей 22
1.1.12 Формальная постановка задачи 25
1.2 Проектирование и разработка 26
1.2.1 Выбор и характеристика набора данных 26
1.2.2 Разведочный анализ данных 28
1.2.3 Предобработка данных 33
1.2.4 Проектирование архитектуры нейронной сети 35
1.2.5 Функция потерь и учёт дисбаланса классов 37
1.2.6 Алгоритм обучения модели 38
1.2.7 Реализация базовых моделей для сравнения 41
1.2.8 Метрики оценки качества 41
1.3 Анализ результатов 43
1.3.1 Результаты обучения нейронной сети 43

1.3.2 Оценка качества на тестовой выборке 45
1.3.3 Сравнение с базовыми моделями 48
1.3.4 Исследование влияния архитектуры сети 50
1.3.5 Исследование влияния балансировки классов 52
1.3.6 Подбор порога классификации 53
1.3.7 Анализ важности признаков 54
1.3.8 Исследование гиперпараметров обучения 56
1.3.9 Итоговая оценка результатов 57
1.3.10 Анализ ошибок и практическое применение модели 58
1.3.11 Оценка вычислительной эффективности модели 60
2 Охрана труда 62
2.1 Анализ условий труда при работе с ЭВМ 62
2.2 Требования к микроклимату и освещению 63
2.3 Требования к организации рабочего места 64
2.4 Электробезопасность 65
2.5 Пожарная безопасность 66
2.6 Выводы по разделу 67
Заключение 68
Список использованных источников 70
ПриложениеА – UML-диаграммы 722
Приложение Б – Листинг программы 777
Приложение В – Презентация 955

Перечень сокращений и обозначений

ИНС - искусственная нейронная сеть

МО - машинное обучение

МСП - многослойный прецептрон (multilayer perceptron, MLP)

API - программный интерфейс приложения (application programming interface)

BCE - бинарная кросс-энтропия (binary cross-entropy)

CRM - система управления взаимоотношениями с клиентами (customerrelationshipmanagement)

CSV - формат хранения табличных данных (comma-separatedvalues)

EDA - разведочный анализ данных (exploratory data analysis)

GPU - графический процессор (graphics processing unit)

LTV - пожизненная ценность клиента (lifetimevalue)

ReLU - блок линейной ректификации (rectifiedlinearunit)

ROC-AUC - площадь под ROC-кривой (areaundertheROCcurve)

SaaS - программное обеспечение как услуга (softwareasaservice)

Введение

За последнее десятилетие подписная (subscription-based) модель монетизации стала одной из доминирующих в цифровой экономике. По этой модели работают стриминговые видеои аудиосервисы, онлайн-кинотеатры, облачные хранилища, программное обеспечение как услуга (SaaS), телекоммуникационные операторы, сервисы доставки и многие другие компании. Ключевая особенность подписной модели состоит в том, что выручка компании формируется не за счёт разовых покупок, а за счёт регулярных периодических платежей, поступающих от действующих подписчиков. В силу этого устойчивость и прибыльность бизнеса напрямую зависят от того, насколько долго клиент остаётся подписчиком сервиса.

Распространение подписной модели обусловлено её взаимной выгодой: потребитель получает доступ к услуге без крупных единовременных затрат, а компания — прогнозируемый и стабильный поток выручки. Объём мирового рынка подписных сервисов на протяжении последних лет демонстрирует устойчивый рост, охватывая всё новые отрасли — от развлечений и образования до программного обеспечения и товаров повседневного спроса. В этих условиях конкуренция за подписчика обостряется, и способность удерживать клиентов становится одним из ключевых факторов успеха компании.

Вместе с тем подписная модель порождает специфическую проблему: в любой момент клиент может отказаться от продления подписки, и чем ниже барьеры выхода, тем проще ему это сделать. Поэтому управление оттоком клиентов превращается в самостоятельную и постоянную задачу, требующую систематического подхода и применения современных аналитических инструментов.

Уход клиента из сервиса принято называть оттоком (англ. churn). Высокий уровень оттока означает, что компания вынуждена постоянно тратить

значительные средства на привлечение новых клиентов лишь для того, чтобы компенсировать потерю ушедших. При этом, согласно многочисленным маркетинговым исследованиям, привлечение нового клиента обходится бизнесу в несколько раз дороже, чем удержание существующего. Следовательно, задача своевременного выявления клиентов, склонных к уходу, и проведения с ними адресных удерживающих мероприятий является экономически значимой и актуальной для любого подписного сервиса.

Решение этой задачи лежит в области интеллектуального анализа данных и машинного обучения. Накопленные сервисом данные о клиентах – срок обслуживания, набор подключённых услуг, тип и условия контракта, способ оплаты, размер платежей и история взаимодействия – содержат закономерности, по которым можно с определённой вероятностью предсказать, уйдёт клиент в ближайшее время или останется. Построение прогнозной модели, выявляющей такие закономерности, позволяет перевести работу по удержанию клиентов из реактивного режима (реакция на уже состоявшийся уход) в проактивный (предотвращение ухода).

Среди методов машинного обучения, применяемых для прогнозирования оттока, особое место занимают искусственные нейронные сети. Благодаря способности аппроксимировать сложные нелинейные зависимости между признаками, нейронные сети демонстрируют высокое качество в задачах бинарной классификации, к которым относится и задача прогнозирования оттока. Развитие специализированных библиотек, таких как PyTorch, и доступность облачных вычислительных сред, таких как GoogleColab, делают разработку и обучение нейросетевых моделей доступными без необходимости в дорогостоящем оборудовании.

Целью настоящей работы является разработка и экспериментальное исследование системы прогнозирования оттока клиентов подписного сервиса на основе искусственной нейронной сети.

Для достижения поставленной цели в работе решаются следующие задачи:

– провести анализ предметной области, понятия оттока клиентов и его влияния на показатели подписного бизнеса;

– выполнить обзор существующих методов и решений в области прогнозирования оттока;

– обосновать выбор инструментальных средств — языка программирования, библиотеки машинного обучения и среды разработки;

– выбрать и подготовить набор данных, провести его разведочный анализ и предобработку;

– спроектировать архитектуру нейронной сети и реализовать алгоритм её обучения с учётом дисбаланса классов;

– провести обучение модели, оценить её качество по совокупности метрик и сравнить с базовыми моделями машинного обучения;

– исследовать влияние архитектуры сети, балансировки классов и порога классификации на качество прогноза, выполнить анализ важности признаков.

Объектом исследования является процесс оттока клиентов в подписных сервисах. Предметом исследования являются методы и модели машинного обучения для прогнозирования оттока.

В работе применялись методы интеллектуального анализа данных, машинного обучения, теории искусственных нейронных сетей, математической статистики, а также методы экспериментального исследования и сравнительного анализа.

Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и двух приложений. В первом разделе проведён анализ предметной области, выполнено проектирование и разработка системы прогнозирования оттока, а также анализ полученных результатов. Второй раздел посвящён вопросам охраны труда при разработке программного обеспечения. В заключении сформулированы основные результаты и выводы. В приложениях приведены листинги разработанных программ.

1. Система прогнозирования оттока клиентов

1.1 Анализ предметной области

1.1.1 Понятие оттока клиентов и показатели подписного бизнеса

Отток клиентов (англ. customerchurn, customerattrition) – это прекращение клиентом отношений с компанией, выражающееся в отказе от продления подписки или расторжении договора на оказание услуг. В подписной бизнес-модели отток является одним из важнейших показателей, поскольку напрямую определяет, какая доля выручки теряется компанией за расчётный период.

Принято различать несколько видов оттока. Добровольный (voluntary) отток происходит по инициативе самого клиента, который сознательно отказывается от услуги – например, из-за неудовлетворённости качеством, высокой цены или перехода к конкуренту. Вынужденный (involuntary) отток наступает по причинам, не зависящим напрямую от воли клиента, — чаще всего из-за неуспешного списания платежа. Кроме того, различают полный отток, при котором клиент полностью прекращает пользоваться сервисом, и частичный, при котором клиент сокращает объём потребляемых услуг. В настоящей работе рассматривается задача прогнозирования факта оттока как такового, без разделения на его подвиды.

Основной количественной мерой оттока служит коэффициент оттока (churnrate) – отношение числа клиентов, ушедших за период, к общему числу клиентов на начало периода. Коэффициент оттока тесно связан с коэффициентом удержания (retentionrate): сумма этих двух показателей равна единице. Так, если за период из сервиса ушло 27% клиентов, коэффициент удержания составляет 73%.

С оттоком напрямую связан показатель пожизненной ценности клиента (lifetimevalue, LTV) — суммарного дохода, который клиент приносит компании за всё время пользования сервисом. Чем ниже коэффициент оттока, тем дольше

клиент остаётся подписчиком и тем выше его пожизненная ценность. Таким образом, снижение оттока даже на несколько процентных пунктов способно существенно повысить совокупную выручку компании, что и обуславливает экономическую значимость задачи прогнозирования оттока.

Помимо прямых потерь выручки, высокий отток влечёт и косвенные негативные последствия: рост удельных затрат на маркетинг, ухудшение репутации сервиса, снижение эффекта сетевого взаимодействия между пользователями. Кроме того, ушедшие клиенты нередко делятся негативным опытом, что затрудняет привлечение новых. Таким образом, отток клиентов представляет собой комплексную проблему, затрагивающую как финансовые, так и репутационные аспекты деятельности компании, а его прогнозирование и предотвращение являются важной составляющей стратегии устойчивого развития подписного бизнеса.

1.1.2 Значимость прогнозирования оттока

Прогнозирование оттока представляет собой задачу заблаговременного определения клиентов, имеющих высокую вероятность ухода в ближайшем будущем. Практическая ценность такого прогноза заключается в возможности проведения превентивных удерживающих мероприятий: персональных предложений, скидок, улучшения условий обслуживания, своевременной технической поддержки.

Экономическая целесообразность прогнозирования оттока опирается на несколько закономерностей, выявленных в маркетинговых исследованиях. Во-первых, стоимость привлечения нового клиента в среднем в пять и более раз превышает стоимость удержания существующего. Во-вторых, согласно принципу Парето, основная доля прибыли формируется относительно небольшой долей лояльных клиентов, потеря которых особенно болезненна. В-третьих, вероятность продажи дополнительной услуги существующему клиенту

значительно выше, чем новому. Из этих закономерностей следует, что инвестиции в удержание клиентов, опирающиеся на точный прогноз оттока, обладают высокой отдачей.

Однако удерживающие мероприятия сами по себе требуют затрат, поэтому применять их ко всей клиентской базе нерационально. Прогнозная модель позволяет сфокусировать усилия именно на тех клиентах, которые действительно находятся в группе риска, и тем самым повысить эффективность расходования бюджета на удержание. Это определяет требование к качеству модели: она должна не просто быть точной в среднем, но обладать высокой полнотой (recall) в отношении уходящих клиентов, чтобы не упускать тех, кого ещё можно удержать.

1.1.3 Обзор существующих решений

Задача прогнозирования оттока решается в составе систем управления взаимоотношениями с клиентами (CRM) и платформ клиентской аналитики. На рынке представлен ряд решений, реализующих предиктивную аналитику оттока.

Платформа Salesforce в составе модуля EinsteinAnalytics предоставляет инструменты автоматического построения прогнозных моделей оттока на основе данных CRM. Облачные платформы машинного обучения, такие как AmazonSageMaker, GoogleCloudAIPlatform и MicrosoftAzureMachineLearning, предлагают типовые шаблоны и автоматизированные конвейеры (AutoML) для решения задачи прогнозирования оттока без необходимости ручной разработки моделей. Специализированные сервисы клиентской аналитики, напримерChurnZero и Gainsight, ориентированы преимущественно на сегмент SaaS и предоставляют готовые показатели «здоровья» клиента и сигналы риска ухода.

Общим недостатком готовых коммерческих решений является их закрытость, высокая стоимость и ориентация на типовые сценарии, что затрудняет их адаптацию под особенности конкретного бизнеса и данных. Кроме того, такие решения, как правило, не раскрывают деталей применяемых моделей. В связи с этим разработка собственной прогнозной модели на основе открытых инструментов остаётся востребованной как с практической, так и с исследовательской точки зрения. Кроме того, готовые платформы автоматизированного машинного обучения, скрывая детали построения модели, не позволяют разработчику в полной мере понять природу прогноза и проверить обоснованность принятых моделью решений, что важно как для доверия к системе, так и для её совершенствования. Самостоятельная разработка модели, напротив, обеспечивает полный контроль над всеми этапами – от предобработки данных до интерпретации результатов. Именно такой подход – самостоятельная разработка и исследование нейросетевой модели прогнозирования оттока средствами открытой библиотеки – реализован в настоящей работе.

1.1.4 Анализ требований к разрабатываемой системе

На основании анализа предметной области сформулированы требования к разрабатываемой системе прогнозирования оттока. Система должна:

– принимать на вход структурированные данные о клиентах подписного сервиса, включающие числовые и категориальные признаки;

– выполнять предобработку данных, приводя их к виду, пригодному для подачи на вход нейронной сети;

– реализовывать модель бинарной классификации, относящую каждого клиента к одному из двух классов – «склонен к оттоку» или «лоялен»;

– учитывать дисбаланс классов, характерный для реальных данных об оттоке, при котором доля уходящих клиентов существенно меньше доли остающихся;

– предоставлять не только бинарное решение, но и оценку вероятности оттока, позволяющую ранжировать клиентов по степени риска;

– обеспечивать качество прогноза, оцениваемое по совокупности метрик классификации, на уровне, пригодном для практического применения;

– быть реализуемой в условиях ограниченных вычислительных ресурсов, в том числе в бесплатной облачной среде.

Перечисленные требования определили выбор инструментальных средств и архитектурных решений, рассматриваемый в последующих пунктах.

1.1.5 Обзор методов ML для прогнозирования оттока

Прогнозирование оттока относится к задачам обучения с учителем (supervisedlearning), а именно к бинарной классификации: по набору признаков клиента требуется отнести его к одному из двух классов. Для решения подобных задач применяется широкий спектр методов машинного обучения, обладающих различными свойствами.

Логистическая регрессия – один из базовых методов бинарной классификации. Она моделирует вероятность принадлежности объекта к классу как логистическую функцию от линейной комбинации признаков. Метод отличается простотой, высокой скоростью обучения и хорошей интерпретируемостью: коэффициенты модели прямо указывают на вклад каждого признака. Основное ограничение – предположение о линейной разделимости классов, из-за чего метод плохо улавливает сложные нелинейные зависимости.

Деревья решений строят иерархию правил вида «если – то», последовательно разбивая пространство признаков. Они хорошо

интерпретируемы и способны улавливать нелинейности, однако склонны к переобучению. Этот недостаток преодолевается в ансамблевых методах. Случайный лес (randomforest) строит множество деревьев на различных подвыборках данных и признаков, усредняя их прогнозы, что повышает устойчивость и точность. Градиентный бустинг (gradientboosting) строит деревья последовательно, каждое из которых корректирует ошибки предыдущих, и нередко обеспечивает наилучшее качество среди классических методов на табличных данных.

Помимо перечисленных, в задачах классификации применяются метод опорных векторов, строящий разделяющую гиперплоскость с максимальным зазором между классами; метод k ближайших соседей, относящий объект к преобладающему классу среди ближайших обучающих примеров; наивный байесовский классификатор, основанный на теореме Байеса и предположении о независимости признаков. Метод опорных векторов эффективен в пространствах высокой размерности, но чувствителен к выбору ядра; метод k ближайших соседей прост, но вычислительно затратен на этапе применения и чувствителен к масштабу признаков; наивный байесовский классификатор быстр, однако его предположение о независимости признаков на практике часто нарушается. Выбор конкретного метода определяется характером данных, требованиями к качеству и интерпретируемости, а также вычислительными ограничениями.

Искусственные нейронные сети моделируют зависимость между признаками и целевой переменной с помощью совокупности связанных вычислительных элементов – нейронов, организованных в слои. За счёт нелинейных функций активации и многослойной структуры нейронные сети способны аппроксимировать функции произвольной сложности, что подтверждается теоремой об универсальной аппроксимации. Это делает их

мощным инструментом классификации, в том числе и для задачи прогнозирования оттока. К особенностям нейронных сетей относится необходимость предварительного масштабирования признаков, подбора гиперпараметров и большего, по сравнению с линейными моделями, объёма данных для обучения.

В рамках настоящей работы в качестве основной модели выбрана искусственная нейронная сеть, а логистическая регрессия и случайный лес используются как базовые модели (baseline) для сравнительной оценки качества.

1.1.6 Обоснование выбора нейронной сети

Выбор искусственной нейронной сети в качестве основной модели обусловлен несколькими причинами. Во-первых, тема настоящей работы предполагает проведение эксперимента именно с нейронной сетью как с современным и перспективным инструментом машинного обучения. Во-вторых, данные об оттоке содержат сложные нелинейные взаимосвязи между признаками – например, совместное влияние срока обслуживания и типа контракта на вероятность ухода, – которые нейронная сеть способна улавливать эффективнее линейных моделей.

В-третьих, нейронная сеть выдаёт на выходе непрерывную оценку вероятности оттока, что соответствует требованию ранжирования клиентов по степени риска. В-четвёртых, архитектура нейронной сети легко масштабируется и допускает гибкую настройку под объём данных и доступные вычислительные ресурсы, что важно при работе в облачной среде. Для решаемой задачи, оперирующей структурированными табличными данными, целесообразно применение полносвязной нейронной сети – многослойного перцептрона, рассматриваемого далее.

1.1.7 Выбор языка программирования

В качестве языка программирования для реализации системы выбран язык Python. На сегодняшний день Python является фактическим стандартом в области машинного обучения и анализа данных. Его выбор обусловлен наличием обширной экосистемы специализированных библиотек, лаконичным и читаемым синтаксисом, а также широкой поддержкой со стороны сообщества разработчиков.

Для решаемой задачи ключевое значение имеют библиотеки экосистемы Python: NumPy для эффективных операций с многомерными массивами, pandas для загрузки и обработки табличных данных, scikit-learn для предобработки и реализации базовых моделей, Matplotlib для визуализации результатов, а также PyTorch для построения и обучения нейронной сети. Альтернативные языки, такие как R или Julia, также применяются в анализе данных, однако уступают Python по широте поддержки библиотек глубокого обучения и распространённости в облачных средах.

1.1.8 Выбор библиотеки машинного обучения

Для построения и обучения нейронной сети выбрана библиотека PyTorch. На рынке библиотек глубокого обучения доминируют два решения –PyTorch, развиваемый компанией Meta, и TensorFlow, развиваемый компанией Google. Обе библиотеки обладают сопоставимой функциональностью и производительностью.

Выбор в пользу PyTorch обусловлен следующими соображениями. PyTorch использует динамический вычислительный граф, что делает процесс отладки модели более наглядным и приближенным к обычному программированию на Python. Библиотека предоставляет понятный объектно-ориентированный интерфейс описания моделей, при котором архитектура сети задаётся как класс, наследуемый от базового модуля. PyTorch получил широкое

распространение в исследовательской среде, обладает подробной документацией и активным сообществом. Кроме того, библиотека автоматически вычисляет градиенты с помощью механизма autograd и поддерживает вычисления как на центральном, так и на графическом процессоре, что позволяет при необходимости ускорить обучение без изменения кода модели.

1.1.9 Выбор среды разработки

В качестве среды разработки и проведения экспериментов выбрана облачная платформа GoogleColaboratory (GoogleColab). Colab представляет собой бесплатный облачный сервис, предоставляющий интерактивную среду выполнения кода Python в формате блокнотов Jupyter непосредственно в браузере.

Преимущества GoogleColab для решаемой задачи заключаются в следующем. Среда не требует установки программного обеспечения на локальный компьютер и поставляется с предустановленными библиотеками машинного обучения, включая PyTorch. Платформа бесплатно предоставляет доступ к графическому процессору, что позволяет при необходимости ускорить обучение нейронной сети. Блокнотный формат удобен для проведения экспериментов, так как объединяет в одном документе исполняемый код, результаты его выполнения, графики и пояснительный текст. Учитывая, что разрабатываемая модель компактна, для её обучения достаточно ресурсов центрального процессора, предоставляемых средой по умолчанию. Это согласуется с требованием реализуемости системы в условиях ограниченных вычислительных ресурсов.

1.1.10 Выбор и настройка инструментов разработки

Для разработки и проведения экспериментального исследования использовался стек инструментов на языке Python. Базой для выполнения программ является интерпретатор Python с изолированным виртуальным

окружением churn_env. Такое окружение обеспечивает воспроизводимость экспериментов и исключает влияние версий библиотек, установленных в других проектах.

Для локальной работы с исходным кодом использовалась интегрированная среда разработки PyCharm. В настройках проекта выбран интерпретатор виртуального окружения; в нём установлены библиотеки pandas и NumPy для обработки данных, scikit-learn для подготовки выборок и расчёта базовых моделей, matplotlib для визуализации, а также PyTorch для построения и обучения нейронной сети.Нарисунке1представленасхемаархитектурысервера.

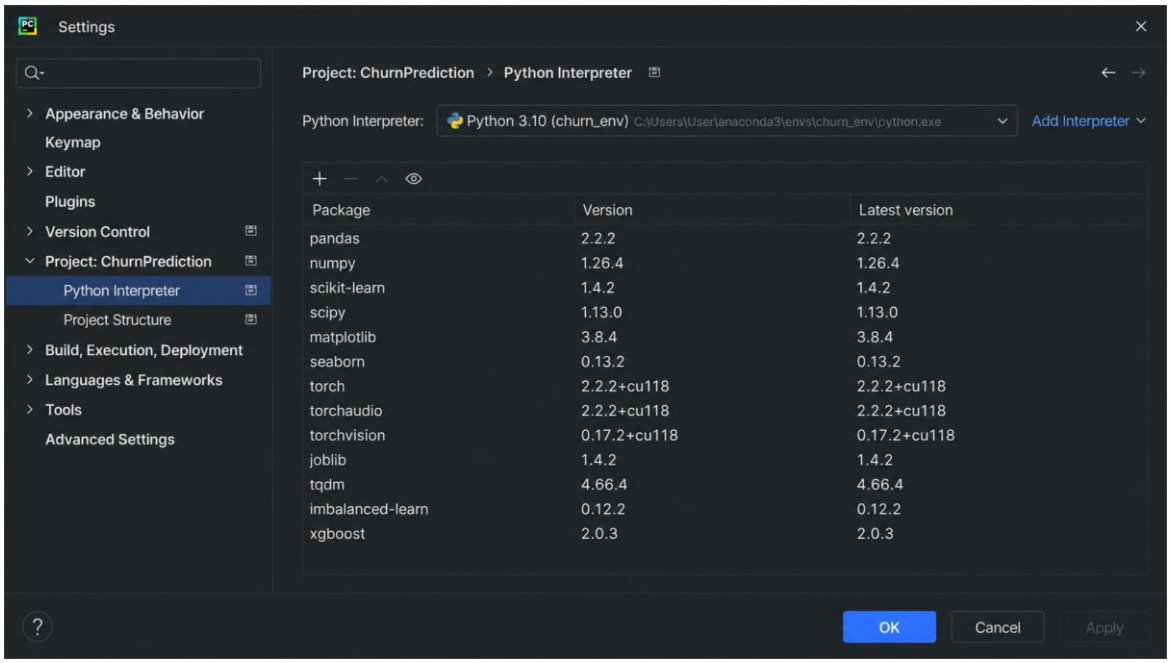


Рисунок 1 - Настройка интерпретатора и библиотек в среде разработки PyCharm

Пакеты устанавливались с помощью менеджера pip. После установки выполнялась проверка состава окружения командой piplist. В рабочем окружении доступны библиотеки torch, torchvision и torchaudio, а также пакеты для обработки табличных данных и визуализации. Проверка версий позволяет фиксировать состав программной среды, необходимый для повторного запуска

экспериментов. На рисунке 2 представлена проверка установленных пакетов в виртуальном окружении Python

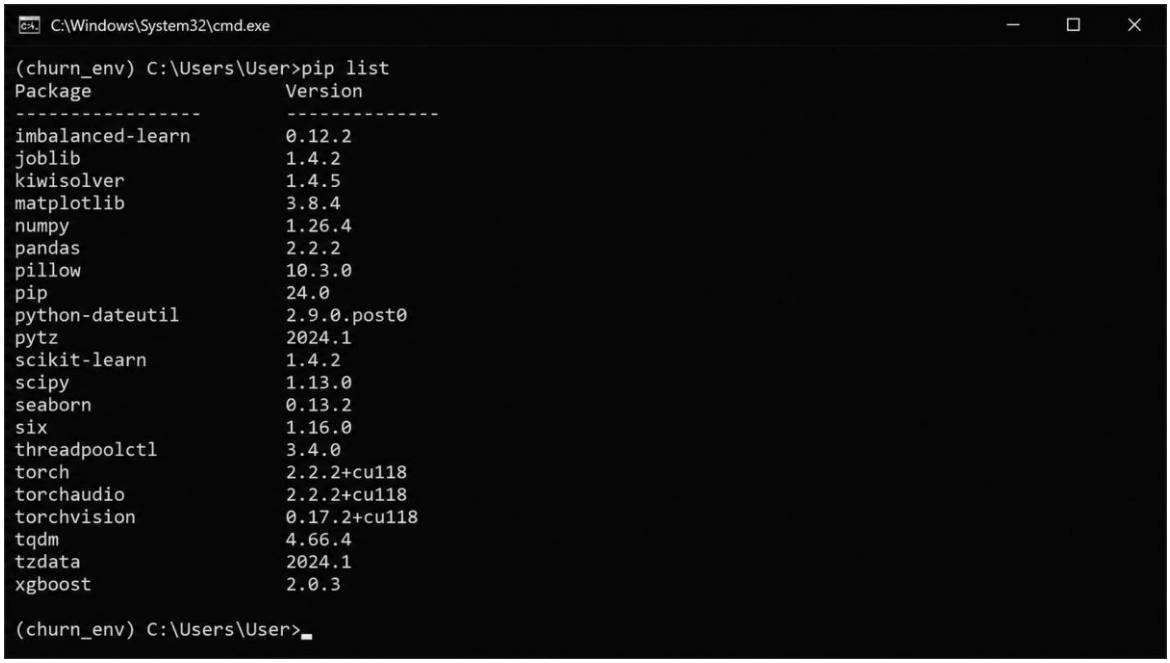


Рисунок 2 - Проверка установленных пакетов в виртуальном окружении Python

Для запуска вычислительных экспериментов применялась облачная платформа GoogleColab. Она предоставляет Jupyter-совместимый интерфейс, позволяет запускать код без настройки локального сервера и при необходимости использовать графический процессор. В блокноте выполнялись загрузка набора данных, предобработка признаков, обучение модели ChurnNet и расчёт метрик качества.НарисункеЗпредставленарабочая среда GoogleColab для выполнения экспериментов.

Для нейросетевой части системы использовалась библиотека PyTorch. При её установке выбирались параметры, соответствующие операционной системе Windows, менеджеру пакетов pip, языку Python и поддерживаемой версии CUDA.

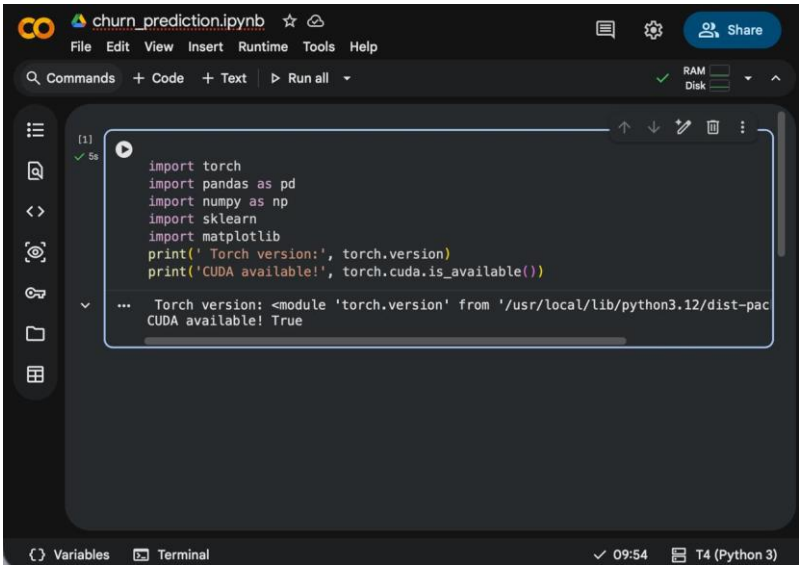


Рисунок 3 - Рабочая среда GoogleColab для выполнения экспериментов

После установки выполнялась проверка версии библиотеки и доступности ускорения CUDA. При отсутствии GPU модель может обучаться на CPU без изменения исходного кода. Нарисунке4представлен выбор параметров установки библиотеки PyTorch.

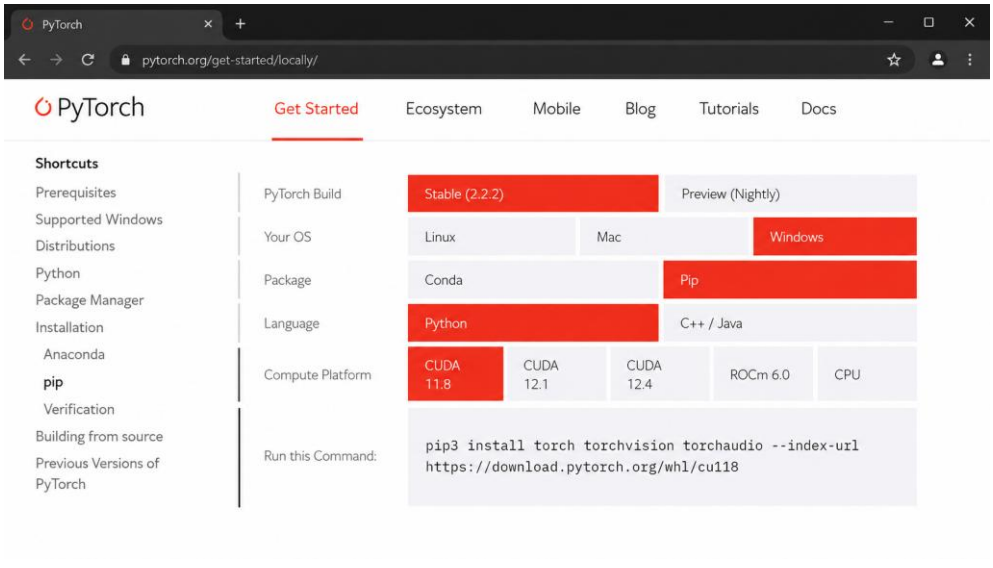


Рисунок 4 - Выбор параметров установки библиотеки PyTorch

1.1.11 Математические основы искусственных нейронных сетей

Искусственная нейронная сеть представляет собой математическую модель, построенную по аналогии с биологическими нейронными сетями. Базовым элементом сети является искусственный нейрон, который вычисляет взвешенную сумму входных сигналов, прибавляет к ней смещение и пропускает результат через нелинейную функцию активации. Математически работа одного нейрона описывается выражением:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \quad (1)$$

где *xi*– входные признаки;

wi – весовые коэффициенты, настраиваемые в процессе обучения;

b – смещение (bias);

f – функция активации;

y – выходной сигнал нейрона.

Весовые коэффициенты определяют вклад каждого входного признака в выходной сигнал и являются обучаемыми параметрами модели.

Нелинейная функция активации является ключевым элементом, позволяющим сети моделировать сложные зависимости. Без неё последовательность слоёв сводилась бы к одному линейному преобразованию. В современных нейронных сетях наиболее распространена функция ReLU (rectifiedlinearunit):

$$\mathrm{ReLU}(x) = \max(0, x) \quad (2)$$

Функция ReLU обнуляет отрицательные значения и пропускает положительные без изменений. Она вычислительно проста, способствует разрежённости активаций и в значительной мере устраняет проблему затухания

градиента, свойственную сигмоидальным функциям. В выходном слое для задачи бинарной классификации применяется логистическая (сигмоидальная) функция, преобразующая произвольное вещественное число в значение из интервала от нуля до единицы, интерпретируемое как вероятность:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

Графики наиболее употребительных функций активации приведены на рисунке 5.

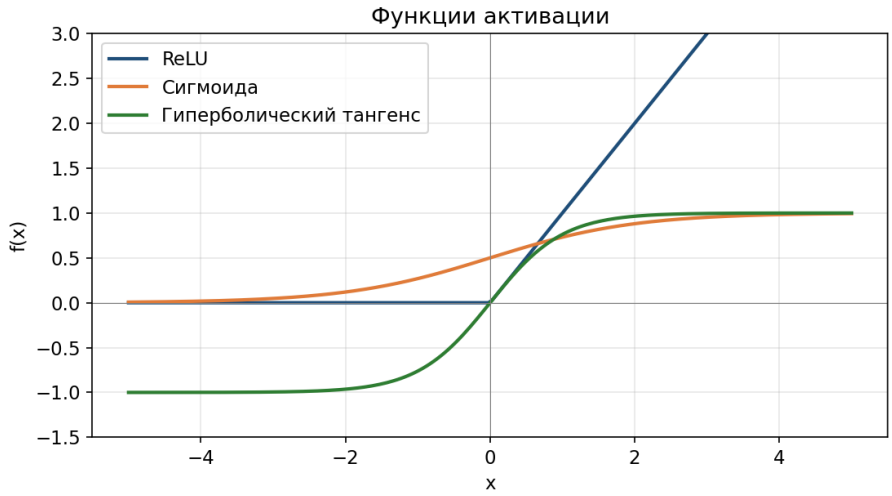


Рисунок 5 – Функции активации нейронов

Объединённые в слои нейроны образуют многослойную сеть. Вычисление выходного сигнала сети по входному вектору признаков называется прямым распространением (forwardpropagation) и состоит в последовательном применении преобразований каждого слоя: выход предыдущего слоя служит входом для следующего. Таким образом, сеть реализует сложную композицию линейных преобразований и нелинейных функций активации, что и обеспечивает её способность аппроксимировать функции произвольной сложности.

Обучение сети состоит в подборе таких значений весов, при которых её прогнозы наилучшим образом соответствуют истинным меткам обучающей выборки. Мерой несоответствия служит функция потерь. Для бинарной классификации используется функция бинарной кросс-энтропии:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \ln \hat{y}_i + (1-y_i)\ln(1-\hat{y}_i)] \quad (4)$$

где N – число объектов;

yi– истинная метка класса (0 или 1);

$\hat{y}i$– прогнозируемая моделью вероятность.

Функция потерь принимает тем меньшие значения, чем ближе прогнозы к истинным меткам.

Минимизация функции потерь выполняется методом градиентного спуска. Для этого вычисляется градиент функции потерь по всем весам сети, после чего веса корректируются в направлении, противоположном градиенту:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w} \quad (5)$$

где η — скорость обучения (learningrate), определяющая величину шага корректировки.

Градиенты функции потерь по весам всех слоёв вычисляются эффективно с помощью алгоритма обратного распространения ошибки (backpropagation), основанного на применении правила дифференцирования сложной функции (цепного правила). Алгоритм последовательно распространяет ошибку от выходного слоя к входному, вычисляя на каждом слое необходимые частные производные. В настоящей работе вычисление градиентов выполняется автоматически средствами механизма автоматического дифференцирования библиотеки PyTorch, а для обновления весов применяется

усовершенствованный метод градиентного спуска Adam, рассматриваемый в пункте 1.2.6.

1.1.12 Формальная постановка задачи

На основании проведённого анализа задача прогнозирования оттока формализуется как задача бинарной классификации с учителем. Каждый клиент описывается вектором признаков, объединяющим его демографические характеристики, набор подключённых услуг, условия обслуживания и платёжную информацию. Этому вектору сопоставляется целевая метка, принимающая значение единицы, если клиент ушёл из сервиса (отток), и нуля в противном случае.

Имеется обучающая выборка, состоящая из пар «вектор признаков — метка класса», полученных из исторических данных сервиса, для которых факт оттока уже известен. Требуется построить решающую функцию, которая по вектору признаков произвольного клиента вычисляет оценку вероятности его оттока — число из интервала от нуля до единицы. Сравнивая эту оценку с заданным порогом, клиента относят к одному из двух классов. Качество решающей функции оценивается на отдельной тестовой выборке, не участвовавшей в обучении, по совокупности метрик классификации.

Обучение модели сводится к подбору её параметров, минимизирующих функцию потерь на обучающей выборке, при условии сохранения обобщающей способности на новых данных. Поскольку доля уходящих клиентов в данных значительно меньше доли остающихся, задача осложняется дисбалансом классов, который необходимо учитывать как при обучении модели, так и при выборе метрик её оценки. Целевым показателем качества выбрана метрика ROC-AUC как не зависящая от порога классификации, а в качестве важнейшего прикладного показателя рассматривается полнота выявления уходящих клиентов.

Таким образом, по результатам анализа предметной области определены постановка задачи, набор требований к системе и инструментальные средства её реализации. Выбрана модель — искусственная нейронная сеть в форме многослойного перцептрона; обоснованы язык программирования Python, библиотека PyTorch и облачная среда GoogleColab; рассмотрены математические основы нейронных сетей. Это создаёт необходимую базу для перехода к проектированию и разработке системы, рассматриваемым во втором пункте.

1.2 Проектирование и разработка

1.2.1 Выбор и характеристика набора данных

Для обучения и оценки модели использован открытый набор данных TelcoCustomerChurn, опубликованный компанией IBM и широко применяемый в качестве эталонного в задачах прогнозирования оттока. Набор описывает клиентов телекоммуникационного оператора, предоставляющего услуги связи и интернета по подписной модели с помесячной и контрактной оплатой, что в полной мере соответствует теме настоящей работы.

Набор данных содержит сведения о 7043 клиентах, каждый из которых описывается 21 полями. Среди признаков представлены как числовые, так и категориальные характеристики клиента, отражающие его демографические данные, набор подключённых услуг, условия обслуживания и платёжную информацию. Целевым признаком является поле Churn, принимающее значение «Yes», если клиент ушёл из сервиса, и «No» в противном случае.

Состав основных признаков набора данных приведён в таблице 1.

Таблица 1 – Описание признаков набора данных TelcoCustomerChurn

Признак	Тип	Описание
gender	категор.	пол клиента
SeniorCitizen	бинарн.	является ли клиент пенсионером
Partner	категор.	наличие супруга(и)
Dependents	категор.	наличие иждивенцев
tenure	числов.	срок обслуживания, месяцев
PhoneService	категор.	подключена ли телефонная связь
InternetService	категор.	тип интернет-подключения
OnlineSecurity	категор.	услуга сетевой безопасности
TechSupport	категор.	услуга технической поддержки
StreamingTV	категор.	услуга стримингового ТВ
Contract	категор.	тип контракта (помесячный, годовой, двухлетний)
PaperlessBilling	категор.	безбумажный расчёт
PaymentMethod	категор.	способ оплаты
MonthlyCharges	числов.	ежемесячный платёж, $
TotalCharges	числов.	суммарный платёж за всё время, $
Churn	целевой	факт оттока клиента (Yes/No)

Признак Тип Описание gender категор. пол клиента SeniorCitizen бинарн. является ли клиент пенсионером Partner категор. наличие супруга(и) Dependents категор. наличие иждивенцев tenure числов. срок обслуживания, месяцев PhoneService категор. подключена ли телефонная связь InternetService категор. тип интернет-подключения OnlineSecurity категор. услуга сетевой безопасности TechSupport категор. услуга технической поддержки StreamingTV категор.

услуга стримингового ТВ Contract категор. тип контракта (помесячный, годовой, двухлетний) PaperlessBilling категор. безбумажный расчёт PaymentMethod категор. способ оплаты MonthlyCharges числов. ежемесячный платёж, $ TotalCharges числов. суммарный платёж за всё время, $ Churn целевой факт оттока клиента (Yes/No) Для числовых признаков набора данных вычислены описательные статистики, приведённые в таблице 2. Они позволяют оценить диапазоны значений и характер распределения признаков.

Таблица 2 – Описательные статистики числовых признаков

Признак	Мин.	Макс.	Среднее	Медиана	Ст. откл.
Tenure(срок,мес.)	0	72	32.4	29.0	24.6
MonthlyCharges	18.2	118.8	64.8	70.4	30.1
TotalCharges	0.0	8684.8	2279.7	1394.6	2266.8

Из таблицы 2 видно, что срок обслуживания клиентов изменяется от 0 до 72 месяцев при среднем значении около 32 месяцев, а ежемесячные платежи варьируются в широком диапазоне от 18 до 119 долларов. Большой разброс значения суммарных платежей (стандартное отклонение сопоставимо со средним) отражает неоднородность клиентской базы по сроку обслуживания и стоимости тарифов. Существенное различие масштабов числовых признаков подтверждает необходимость их стандартизации перед подачей на вход нейронной сети.

1.2.2 Разведочный анализ данных

Перед построением модели проведён разведочный анализ данных (EDA), целью которого является изучение структуры набора, выявление

закономерностей и проверка качества данных. Прежде всего исследовано распределение клиентов по целевому признаку — факту оттока (рисунок 6).

Рисунок 6– Распределение клиентов по факту оттока

Как видно из рисунка 6, классы несбалансированы: доля ушедших клиентов составляет около 26.5%, тогда как оставшихся — около 73.5%. Преобладание класса лояльных клиентов является типичным для задач прогнозирования оттока и требует специальных мер при обучении модели, рассматриваемых далее.

Далее исследовано влияние типа контракта на вероятность оттока (рисунок 7). Тип контракта отражает срок обязательств клиента перед оператором.

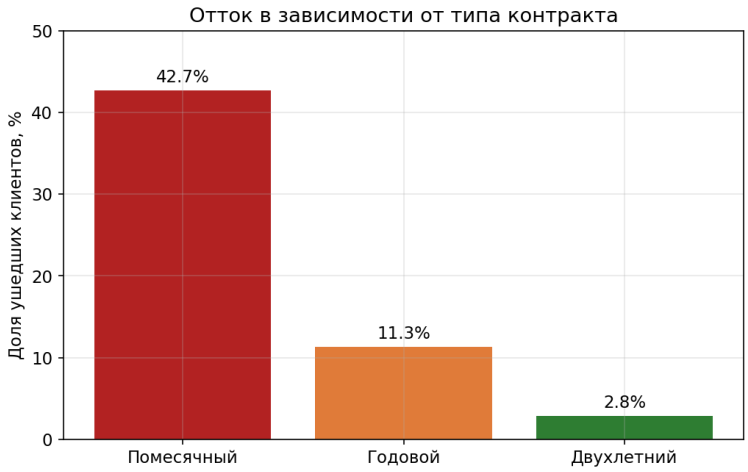


Рисунок 7 – Доля оттока в зависимости от типа контракта

Анализ рисунка 7 выявляет выраженную зависимость: среди клиентов с помесячным контрактом доля ушедших достигает 42.7%, тогда как среди клиентов с двухлетним контрактом она составляет лишь 2.8%. Это объясняется тем, что долгосрочный контракт повышает барьер выхода и отражает изначально более высокую лояльность клиента. Тип контракта, таким образом, является сильным предиктором оттока.

Существенное влияние на отток оказывает и срок обслуживания клиента (tenure). На рисунке 8 представлена зависимость доли оттока от срока обслуживания, сгруппированного по интервалам.

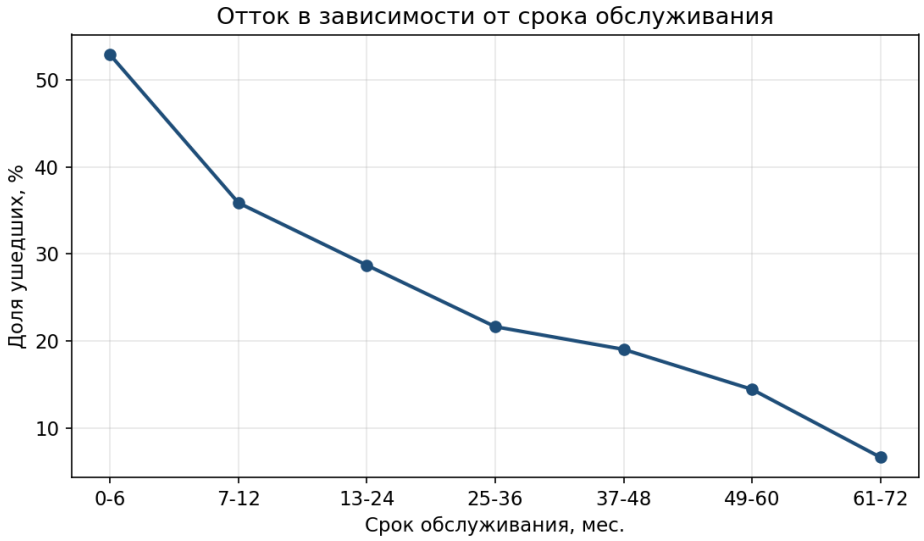


Рисунок 8 – Доля оттока в зависимости от срока обслуживания

Из рисунка 8 следует, что наибольшая доля оттока наблюдается среди клиентов с малым сроком обслуживания (до полугода), после чего она монотонно снижается с увеличением срока. Новые клиенты ещё не сформировали привычку пользования сервисом и уходят значительно чаще, чем клиенты со стажем. Срок обслуживания является, таким образом, ещё одним важным предиктором.

Распределение ежемесячных платежей в разрезе факта оттока показано на рисунке 9.

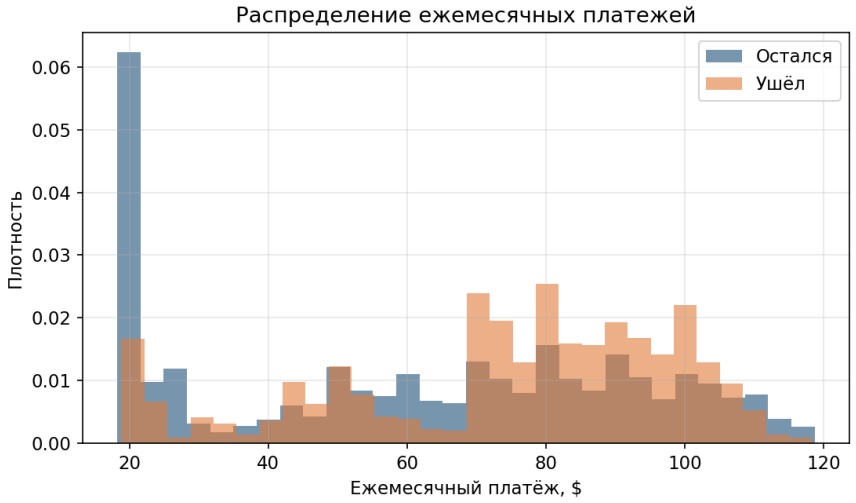


Рисунок 9 – Распределение ежемесячных платежей

Рисунок 9 показывает, что ушедшие клиенты в среднем имеют более высокие ежемесячные платежи: их распределение смещено в область больших значений. Это может объясняться большей чувствительностью к цене у клиентов с дорогими тарифами. Наконец, для оценки взаимосвязей числовых признаков построена корреляционная матрица (рисунок 10).

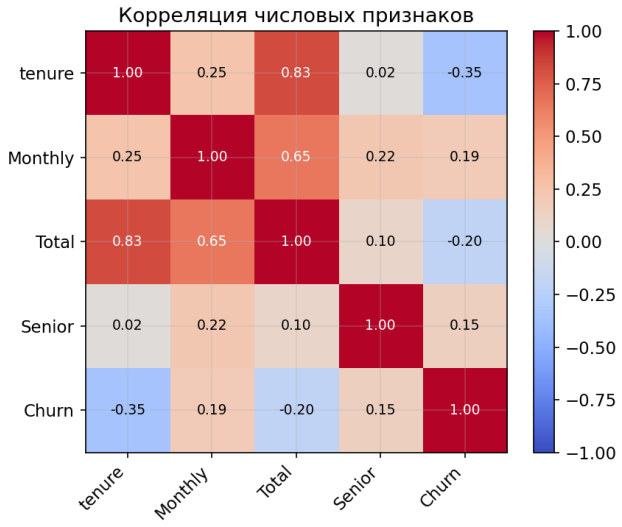


Рисунок 10 – Корреляционная матрица числовых признаков

Корреляционная матрица подтверждает выявленные закономерности: срок обслуживания отрицательно коррелирует с фактом оттока, а суммарный платёж сильно положительно коррелирует со сроком обслуживания, что естественно. Сильной взаимной корреляции, способной существенно ухудшить обучение модели, между признаками не наблюдается.

Дополнительно исследовано влияние способа оплаты и типа интернет-подключения на отток (рисунки 11 и 12). Способ оплаты косвенно отражает степень вовлечённости и финансовую дисциплину клиента.

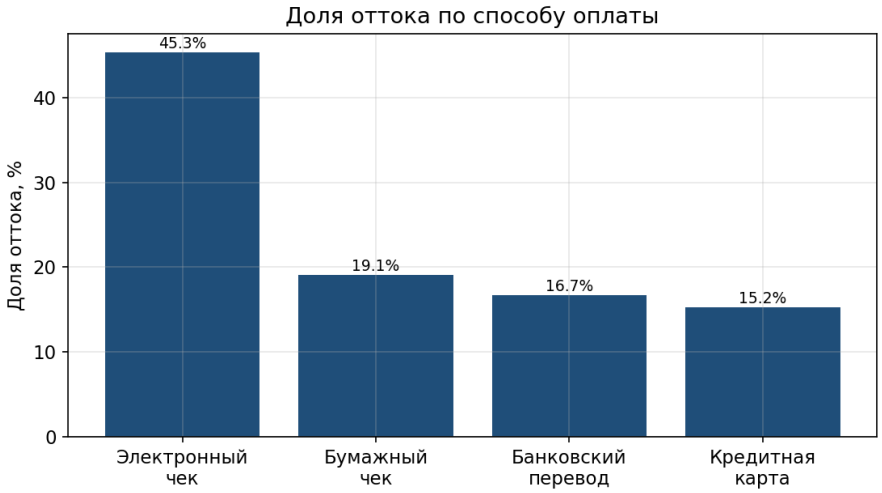


Рисунок 11 – Доля оттока в зависимости от способа оплаты

Из рисунка 11 видно, что клиенты, использующие электронные чеки, уходят существенно чаще, чем клиенты с автоматическими способами оплаты (банковский перевод, кредитная карта). Автоматическое списание платежей повышает удобство пользования сервисом и снижает вероятность ухода. Влияние типа интернет-подключения показано на рисунке 8.

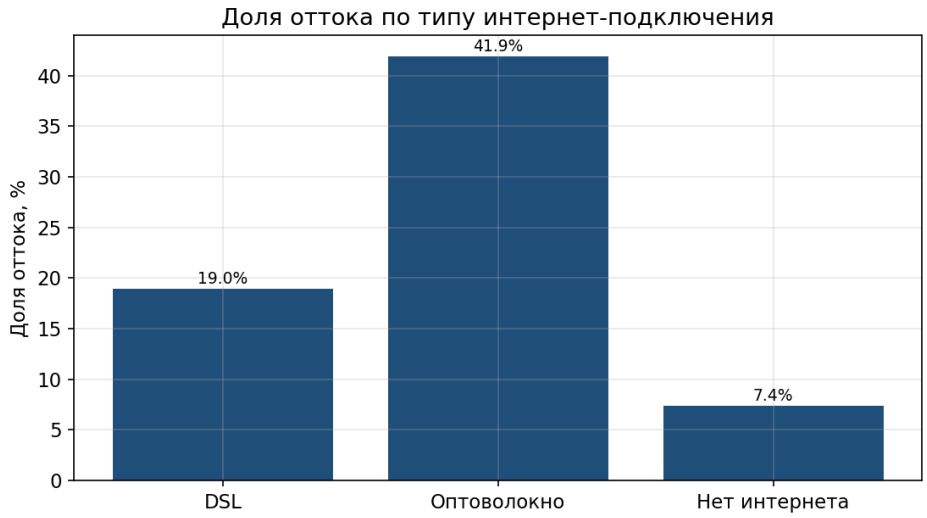


Рисунок 12 – Доля оттока в зависимости от типа интернет-подключения

Рисунок 12 выявляет, что клиенты с оптоволоконным подключением уходят заметно чаще, чем клиенты с подключением DSL или без интернета. Это может быть связано с более высокой стоимостью оптоволоконных тарифов и соответствующими ожиданиями клиентов относительно качествауслуги. Выявленные в ходе разведочного анализа закономерности используются далее при интерпретации важности признаков обученной модели (пункт 1.3.7).

1.2.3 Предобработка данных

Перед подачей на вход нейронной сети исходные данные подвергаются предобработке, включающей несколько последовательных этапов.

На первом этапе из набора удаляется поле customerID — уникальный идентификатор клиента, не несущий предсказательной информации и способный привести к переобучению модели.

На втором этапе устраняется дефект данных в поле TotalCharges. Несмотря на числовую природу, это поле было загружено как текстовое, поскольку у 11 клиентов с нулевым сроком обслуживания (новых клиентов) оно содержит пробел вместо значения. Поле преобразуется в числовой тип, а отсутствующие значения заполняются нулём, что корректно отражает отсутствие накопленных платежей у новых клиентов.

На третьем этапе целевой признак Churn кодируется в бинарную форму: значению «Yes» (отток) сопоставляется единица, значению «No» (лояльность) — ноль.

На четвёртом этапе выполняется кодирование категориальных признаков. Поскольку нейронная сеть оперирует числовыми данными, категориальные признаки преобразуются методом прямого кодирования (one-hotencoding), при котором каждой категории признака сопоставляется отдельный бинарный столбец. Для исключения линейной зависимости столбцов первая категория каждого признака отбрасывается (drop_first). В результате кодирования общее число признаков, подаваемых на вход сети, составило 30.

На пятом этапе набор данных разбивается на три непересекающиеся выборки: обучающую (4930 записей, 70%), валидационную (1056 записей, 15%) и тестовую (1057 записей, 15%). Разбиение выполняется со стратификацией по целевому признаку, что сохраняет исходное соотношение классов во всех трёх выборках. Обучающая выборка используется для настройки весов сети, валидационная — для контроля переобучения и ранней остановки, тестовая — для итоговой независимой оценки качества.

На шестом этапе числовые признаки подвергаются стандартизации (приведению к нулевому среднему и единичному стандартному отклонению). Стандартизация ускоряет и стабилизирует обучение нейронной сети. Важной деталью является то, что параметры стандартизации (среднее и стандартное

отклонение) вычисляются исключительно по обучающей выборке и затем применяются к валидационной и тестовой. Такой порядок исключает утечку информации (dataleakage) из тестовых данных в процесс обучения и обеспечивает объективность итоговой оценки.

Выбор именно такой последовательности предобработки обусловлен требованиями корректного применения нейронной сети. Прямое кодирование категориальных признаков предпочтительнее порядкового, поскольку не вводит искусственного отношения порядка между категориями, не имеющими естественной упорядоченности (например, между способами оплаты). Заполнение пропусков нулём, а не средним значением, в данном случае содержательно оправдано, так как пропуски соответствуют именно отсутствию накопленных платежей у новых клиентов. Совокупность перечисленных преобразований приводит исходные разнородные данные к согласованному числовому представлению, пригодному для эффективного обучения модели.

1.2.4 Проектирование архитектуры нейронной сети

Для решения задачи спроектирована полносвязная нейронная сеть прямого распространения — многослойный перцептрон. Выбор данной архитектуры обусловлен структурированным табличным характером данных, для которого свёрточные и рекуррентные сети не дают преимуществ.

Сеть состоит из входного слоя, двух скрытых слоёв и выходного слоя. Входной слой принимает вектор из 30 признаков. Первый скрытый слой содержит 64 нейрона, второй — 32 нейрона. Выходной слой состоит из одного нейрона, формирующего логит — ненормированную оценку, которая преобразуется в вероятность оттока с помощью логистической (сигмоидальной) функции на этапе оценки. Постепенное сужение слоёв (64 → 32 → 1) реализует иерархическое сжатие информации от признаков к итоговому решению.

В качестве функции активации скрытых слоёв применяется блок линейной ректификации (ReLU), определяемый как f(x) = max(0, x). ReLU вносит в модель нелинейность, вычислительно эффективна и не подвержена проблеме затухания градиента в той мере, в какой ей подвержены сигмоидальные активации.

Для повышения устойчивости и качества обучения в каждый скрытый слой включены два дополнительных механизма. Пакетная нормализация (batchnormalization) нормализует выходы слоя в пределах обучающего пакета, что ускоряет сходимость и снижает чувствительность к выбору начальных весов. Прореживание (dropout) с вероятностью 0,3 случайным образом отключает часть нейронов на каждом шаге обучения, препятствуя переобучению и повышая обобщающую способность сети.

Обоснование конкретного числа нейронов в скрытых слоях приведено в пункте 1.3.4, где экспериментально сравниваются различные варианты архитектуры. Выбор двух скрытых слоёв является компромиссом между выразительной способностью сети и риском переобучения: одного скрытого слоя может быть недостаточно для улавливания сложных взаимодействий признаков, тогда как чрезмерное увеличение числа слоёв на ограниченном объёме данных ведёт к переобучению и усложняет обучение. Структура спроектированной сети схематически представлена на рисунке 13.

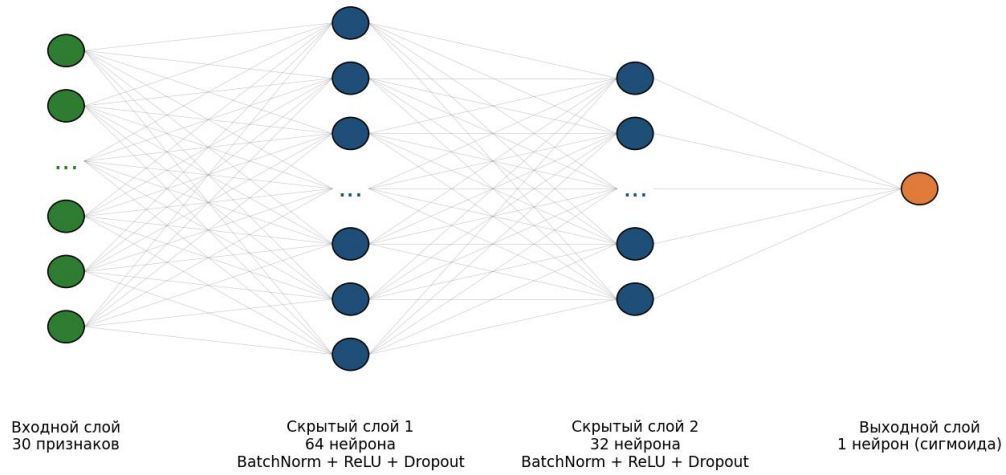


Рисунок 13 – Структура многослойного перцептрона

1.2.5 Функция потерь и учёт дисбаланса классов

В задачах бинарной классификации в качестве функции потерь применяется бинарная кросс-энтропия (binarycross-entropy, BCE), измеряющая расхождение между прогнозируемой вероятностью и истинной меткой класса. В реализации используется вариант функции BCEWithLogitsLoss, объединяющий применение сигмоиды и вычисление кросс-энтропии в одной операции, что обеспечивает большую численную устойчивость по сравнению с раздельным вычислением.

Как было показано при разведочном анализе, классы в данных существенно несбалансированы: доля уходящих клиентов составляет лишь около 27%. При обучении на несбалансированных данных без специальных мер модель склонна «занижать» прогноз для редкого класса, поскольку высокую точность можно формально получить, относя почти всех клиентов к преобладающему классу лояльных. Однако для решаемой задачи именно редкий класс уходящих клиентов представляет наибольший интерес.

Для устранения этой проблемы применяется взвешивание классов: редкому классу оттока в функции потерь присваивается больший вес через параметр pos_weight, равный отношению числа объектов преобладающего класса к числу объектов редкого класса. В результате ошибка на уходящих клиентах вносит больший вклад в функцию потерь, что побуждает модель уделять им большее внимание и повышает полноту прогноза. Количественная оценка влияния балансировки на качество модели приведена в пункте 1.3.5.

Следует отметить, что взвешивание классов в функции потерь является лишь одним из возможных способов борьбы с дисбалансом. Альтернативные подходы включают увеличение числа объектов редкого класса путём их дублирования или генерации синтетических примеров, а также уменьшение числа объектов преобладающего класса. Взвешивание функции потерь выбрано как наиболее простой и эффективный метод, не изменяющий исходный объём и состав обучающей выборки и не вносящий в неё искусственных данных, что снижает риск искажения закономерностей.

1.2.6 Алгоритм обучения модели

Обучение нейронной сети представляет собой итеративный процесс настройки её весовых коэффициентов с целью минимизации функции потерь на обучающей выборке. В работе применяется оптимизатор Adam — адаптивный метод стохастической градиентной оптимизации, сочетающий преимущества методов с импульсом и адаптивной настройкой скорости обучения для каждого параметра. Начальная скорость обучения принята равной 0,001. Для дополнительной регуляризации применяется убывание весов (weightdecay) с коэффициентом 0,0001.

Обучение проводится пакетами (batch) по 64 объекта в течение заданного числа эпох, где эпоха соответствует одному полному проходу по обучающей выборке. На каждом шаге выполняется прямой проход (вычисление прогноза),

вычисление функции потерь, обратное распространение ошибки (вычисление градиентов методом автоматического дифференцирования) и обновление весов оптимизатором.

Для предотвращения переобучения применяется механизм ранней остановки (earlystopping). После каждой эпохи модель оценивается на валидационной выборке, и запоминается состояние весов, при котором валидационная потеря минимальна. Если в течение заданного числа эпох (15) улучшения валидационной потери не происходит, обучение прекращается, а в качестве итоговой принимается модель с наилучшими весами. Это позволяет автоматически определить оптимальную длительность обучения и избежать переобучения. Обобщённая блок-схема алгоритма обучения приведена на рисунке 14.

Рисунок 14 – Блок-схема алгоритма обучения нейронной сети

Обобщённая последовательность действий при обучении модели может быть представлена в виде следующего алгоритма.

Алгоритм обучения нейронной сети Вход: обучающая выборка, валидационная выборка, число эпох E, терпение P Выход: обученная модель с наилучшими весами:

- инициализировать веса сети, оптимизатор Adam, функцию потерь BCE;

-best_loss := +бесконечность; счётчик_без_улучшений := 0 ;

- для epoch от 1 до E выполнять;

- для каждого пакета (X, y) обучающей выборки;

-y_pred := прямой_проход(X) // вычисление прогноза;

-loss := BCE(y_pred, y) // функция потерь;

- вычислить градиенты loss по весам // backpropagation;

- обновить веса оптимизатором Adam;

-val_loss := оценить потерю на валидационной выборке;

-еслиval_loss<best_lossто;

-best_loss := val_loss; сохранитьвеса; счётчик := 0;

-иначе: счётчик := счётчик + 1;

- если счётчик >= P то: прервать цикл // ранняя остановка;

- восстановить наилучшие сохранённые веса.

Применение ранней остановки и сохранение наилучших весов гарантируют, что итоговая модель соответствует точке наилучшей обобщающей способности, а не последней эпохе обучения, на которой возможно переобучение. Такой подход позволяет задавать заведомо большое число эпох, не опасаясь ухудшения качества, поскольку фактическая длительность обучения определяется автоматически по динамике валидационной потери.

1.2.7 Реализация базовых моделей для сравнения

Для объективной оценки качества разработанной нейронной сети необходимо сравнить её с альтернативными методами машинного обучения, выступающими в роли базовых моделей (baseline). В качестве таковых выбраны логистическая регрессия и случайный лес, реализованные средствами библиотеки scikit-learn.

Логистическая регрессия представляет линейный метод и служит нижней границей качества: если сложная нейронная сеть не превосходит простую линейную модель, её применение не оправдано. Случайный лес представляет мощный ансамблевый метод на основе деревьев решений и является сильным конкурентом для табличных данных. Обе базовые модели обучаются на тех же предобработанных данных, что и нейронная сеть, и для них также активирован режим взвешивания классов (class_weight = balanced) для корректного сравнения в условиях дисбаланса. Результаты сравнения приведены в пункте 1.3.3.

1.2.8 Метрики оценки качества

Для всесторонней оценки качества классификации в условиях несбалансированных классов используется совокупность метрик, вычисляемых на основе матрицы ошибок. Матрица ошибок содержит четыре величины: число истинно положительных (TP), истинно отрицательных (TN), ложноположительных (FP) и ложноотрицательных (FN) прогнозов. На их основе определяются следующие метрики: доля верных ответов (accuracy) — отношение числа правильно классифицированных объектов к их общему числу. В условиях дисбаланса классов эта метрика недостаточно информативна, так как может быть высокой даже у модели, игнорирующей редкий класс – точность (precision) — доля действительно ушедших клиентов среди всех, кого модель отнесла к уходящим. Показывает, насколько можно доверять

положительному прогнозу; полнота (recall) — доля выявленных моделью ушедших клиентов среди всех фактически ушедших. Для задачи удержания эта метрика особенно важна, так как отражает способность модели не упускать клиентов из группы риска.

F1-мера — гармоническое среднее точности и полноты, дающее сбалансированную оценку качества по обоим показателям.ROC-AUC — площадь под ROC-кривой, отражающая способность модели разделять классы при всех возможных порогах. Значение 0,5 соответствует случайному угадыванию, 1,0 — идеальной классификации. Метрика не зависит от выбора конкретного порога и потому удобна для сравнения моделей.

Точность и полнота вычисляются на основе элементов матрицы ошибок по формулам:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (6)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (7)$$

где TP — число истинно положительных;

FP — ложноположительных;

FN — ложноотрицательных прогнозов.

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (8)$$

F1-мера определяется как их гармоническое среднее.

Кроме перечисленных метрик, при анализе результатов используются графические представления: ROC-кривая, кривая «точность — полнота» (precision-recall) и матрица ошибок в виде тепловой карты.

Применение совокупности метрик, а не одного показателя, позволяет получить разностороннюю и объективную оценку качества модели. В условиях дисбаланса классов опора на одну лишь долю верных ответов способна ввести в заблуждение, тогда как совместный анализ точности, полноты, F1-меры и ROC-AUC даёт полное представление о поведении модели в отношении как преобладающего, так и редкого класса. Выбор приоритетной метрики при этом определяется бизнес-контекстом: для задачи удержания клиентов, где цена пропуска уходящего клиента высока, в качестве приоритетной рассматривается полнота при контроле приемлемого уровня точности.

1.3 Анализ результатов

1.3.1 Результаты обучения нейронной сети

Обучение основной модели проводилось в течение не более 100 эпох с ранней остановкой. Фактически обучение завершилось на 22-й эпохе, после чего улучшения качества на валидационной выборке не наблюдалось. Динамика функции потерь на обучающей и валидационной выборках представлена на рисунке 15, динамика доли верных ответов — на рисунке 16.

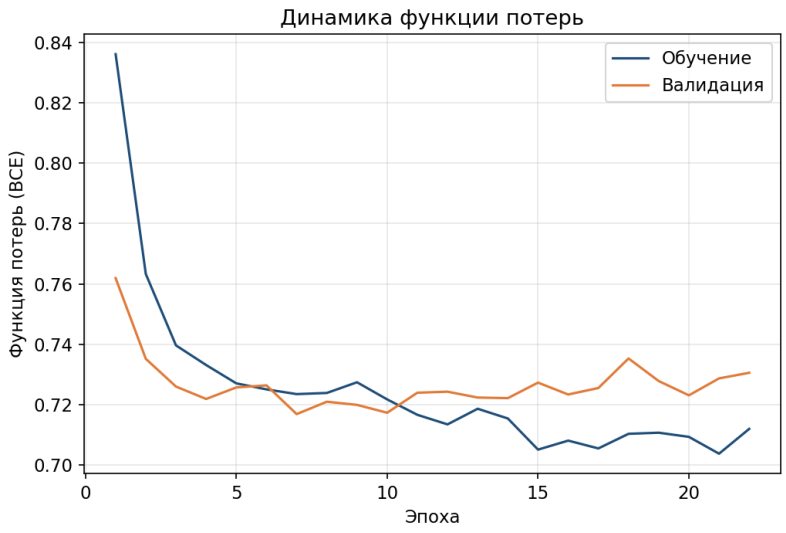


Рисунок 15 – Динамика функции потерь в процессе обучения

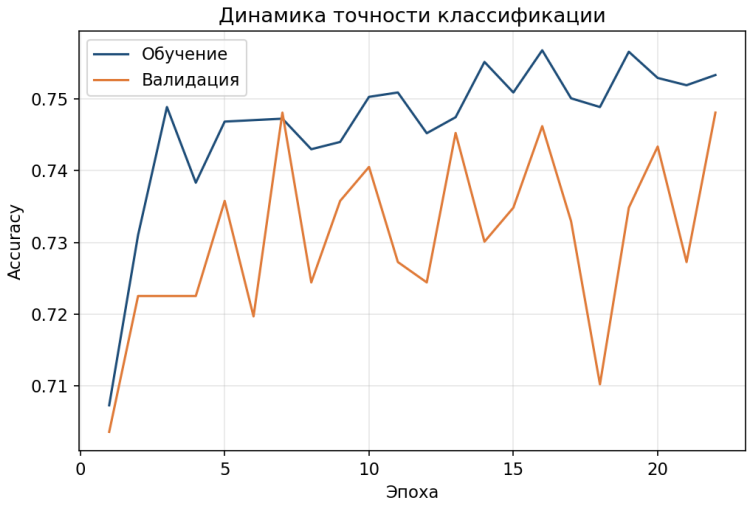


Рисунок 16 – Динамика доли верных ответов в процессе обучения

Из рисунка 15 видно, что функция потерь на обучающей выборке монотонно убывает, тогда как валидационная потеря снижается до определённого момента, после чего стабилизируется и начинает незначительно возрастать. Этот момент соответствует началу переобучения и фиксируется механизмом ранней остановки, который возвращает модель к состоянию с минимальной валидационной потерей. Совпадение характера кривых обучающей и валидационной выборок на начальном этапе и отсутствие большого разрыва между ними свидетельствуют об адекватной регуляризации модели и отсутствии сильного переобучения.

Сравнительно небольшое число эпох, потребовавшихся для обучения, объясняется применением пакетной нормализации и оптимизатора Adam, обеспечивающих быструю сходимость, а также компактностью модели и умеренным объёмом данных. Механизм ранней остановки позволил автоматически определить оптимальную длительность обучения без необходимости её ручного подбора, что повышает воспроизводимость результатов и снижает риск как недообучения, так и переобучения модели.

1.3.2 Оценка качества на тестовой выборке

Итоговое качество модели оценивалось на отложенной тестовой выборке, не участвовавшей в обучении. Полученные значения метрик приведены в таблице 3.

Таблица 3 – Метрики качества нейронной сети на тестовой выборке

Метрика	Значение
Доля верных ответов	0.744
Точность	0.556
Полнота	0.744
F1-мера	0.636
ROC-AUC	0.845

Метрика Значение Доля верных ответов (Accuracy) 0.774 Точность (Precision) 0.556 Полнота (Recall) 0.744 F1-мера 0.636 ROC-AUC 0.845 Ключевым показателем является ROC-AUC, составивший 0.845. Это значение соответствует уровню лучших опубликованных решений для данного набора данных и свидетельствует о хорошей разделяющей способности модели. Полнота модели составила 0.744, то есть модель выявляет около 74% всех фактически ушедших клиентов, что отвечает основному практическому требованию задачи удержания. ROC-кривая модели приведена на рисунке 17.

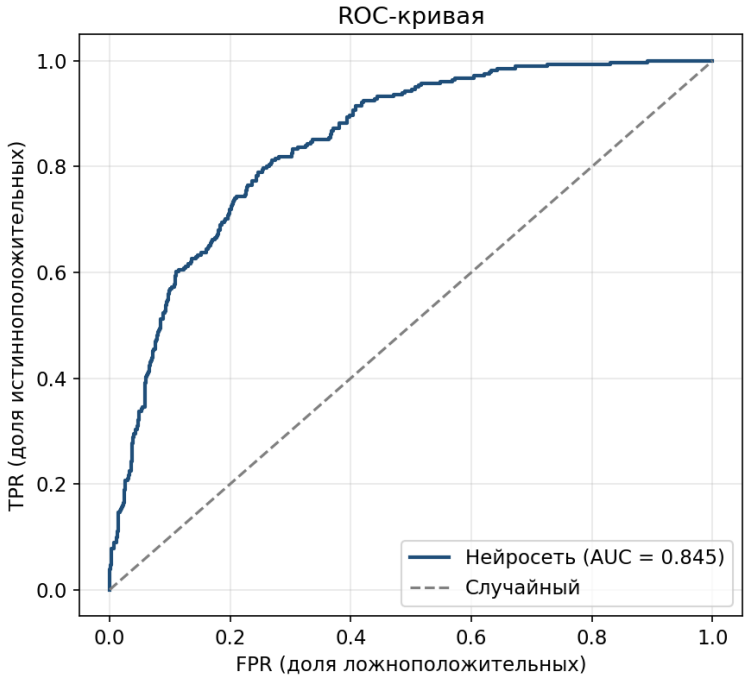


Рисунок 17 – ROC-кривая нейронной сети

ROC-кривая расположена значительно выше диагонали случайного классификатора, что подтверждает высокое качество разделения классов. Матрица ошибок модели представлена на рисунке 18.

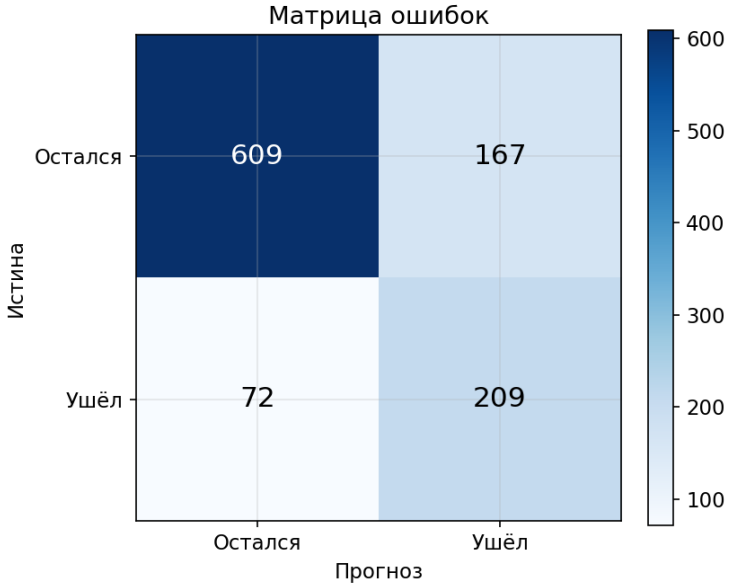


Рисунок 18 – Матрица ошибок нейронной сети

Согласно матрице ошибок, модель верно определила 209 ушедших клиентов из 281 и 609 лояльных из 776. Большинство ошибок приходится на ложноположительные срабатывания, при которых лояльный клиент ошибочно относится к группе риска. Для задачи удержания такие ошибки менее критичны, чем пропуск действительно уходящего клиента, поскольку в худшем случае приводят лишь к избыточному удерживающему предложению. Дополнительно на рисунке 19 приведена кривая «точность — полнота».

Случайный лес продемонстрировал более высокую точность (0.638), но значительно меньшую полноту (0.452), то есть пропускает существенную часть уходящих клиентов, что для задачи удержания нежелательно. В результате его F1-мера (0.529) и ROC-AUC (0.822) оказались ниже, чем у нейронной сети.

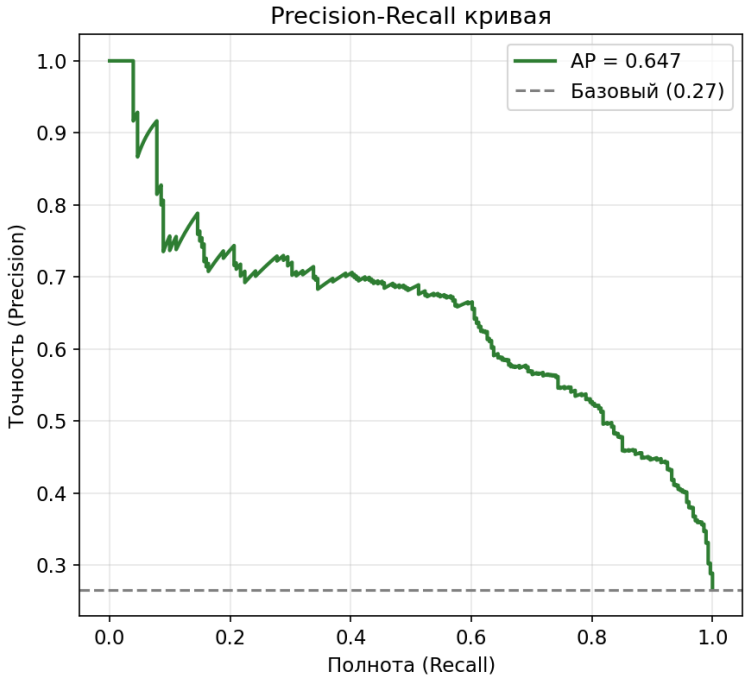


Рисунок 19 – Кривая «точность — полнота»

Площадь под кривой «точность — полнота» (averageprecision) составила 0.647, что существенно превышает уровень случайного классификатора, равный доле положительного класса (0.27).

1.3.3 Сравнение с базовыми моделями

Для оценки целесообразности применения нейронной сети проведено её сравнение с базовыми моделями — логистической регрессией и случайным лесом. Все модели обучались и оценивались на одних и тех же выборках. Сводные результаты приведены в таблице 4.

Таблица 4 – Сравнение качества моделей на тестовой выборке

Модель	Accuracy	Precision	Recall	F1	ROC-AUC
Логистическая регрессия	0.747	0.516	0.779	0.621	0.845
Случайный лес	0.786	0.638	0.452	0.529	0.822
Нейронная сеть	0.774	0.556	0.744	0.636	0.845

Модель AccuracyPrecisionRecallF1 ROC-AUC Логистическая регрессия 0.747 0.516 0.7790.621 0.845 Случайный лес 0.786 0.638 0.452 0.529 0.822 Нейронная сеть 0.774 0.556 0.744 0.636 0.845 Из таблицы 4 видно, что нейронная сеть и логистическая регрессия показали практически одинаковое значение ROC-AUC (0.845 против 0.845), однако нейронная сеть превзошла логистическую регрессию по F1-мере (0.636 против 0.621) за счёт более сбалансированного соотношения точности и полноты. Сравнение ROC-кривых всех трёх моделей приведено на рисунке 20, а столбчатая диаграмма метрик F1 и ROC-AUC — на рисунке 21.

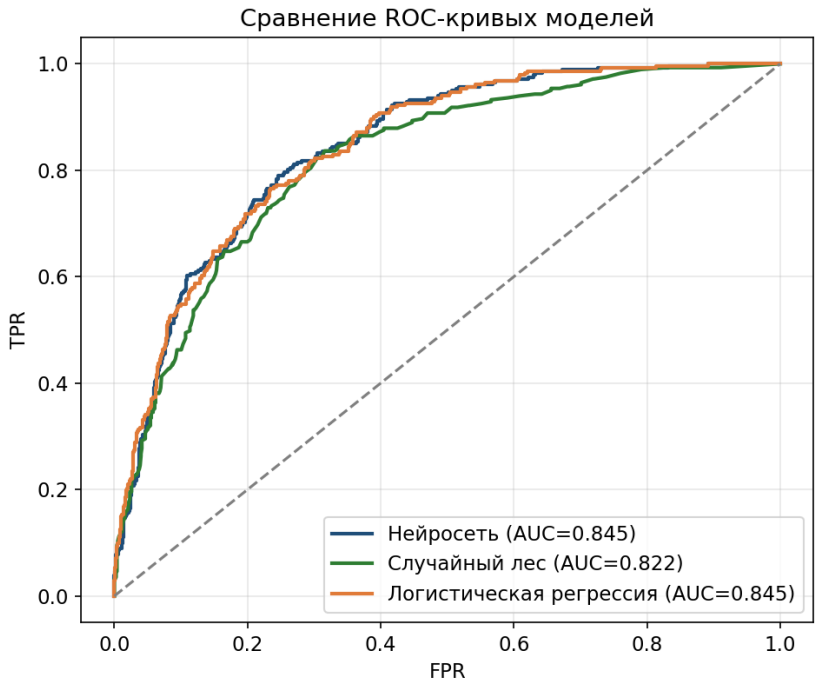


Рисунок 20 – Сравнение ROC-кривых моделей

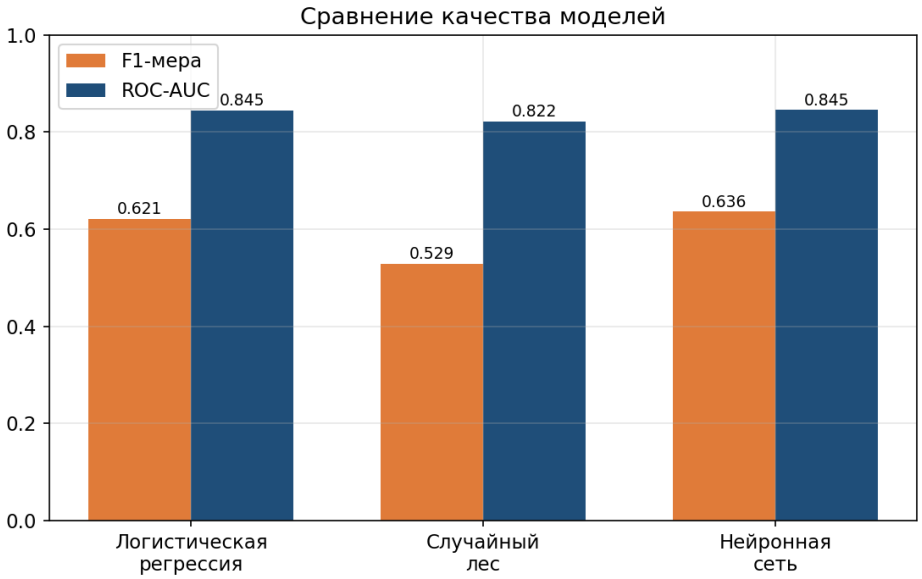


Рисунок 21 – Сравнение моделей по F1-мере и ROC-AUC

Проведённое сравнение показывает, что нейронная сеть обеспечивает наилучший баланс качества среди рассмотренных моделей и её применение для решения поставленной задачи оправдано.

Примечательно, что простая логистическая регрессия показала качество, близкое к нейронной сети по метрике ROC-AUC. Это объясняется тем, что значительная часть зависимостей в данных об оттоке носит близкий к линейному характер, который улавливается и линейной моделью. Преимущество нейронной сети проявляется в способности дополнительно учитывать нелинейные взаимодействия признаков, что выражается в более высокой и сбалансированной F1-мере. Случайный же лес, несмотря на высокую точность, продемонстрировал склонность к осторожным прогнозам в пользу преобладающего класса, что для задачи удержания снижает его практическую ценность. Полученные результаты согласуются с известными в литературе наблюдениями о том, что на табличных данных умеренного объёма различные методы машинного обучения нередко показывают сопоставимое качество, и выбор между ними определяется дополнительными требованиями — к интерпретируемости, скорости и характеру ошибок.

1.3.4 Исследование влияния архитектуры сети

С целью обоснования выбора архитектуры проведена серия экспериментов, в которых сравнивались различные конфигурации скрытых слоёв — от одного слоя с 32 нейронами до двух слоёв с 256 и 128 нейронами. Результаты приведены в таблице 5.

Анализ таблицы 5 показывает, что увеличение размера сети сверх конфигурации с двумя скрытыми слоями (64 и 32 нейрона) не приводит к росту качества: значения ROC-AUC у всех конфигураций близки, а наиболее крупные сети даже несколько уступают по F1-мере и доле верных ответов, что объясняется склонностью к переобучению на ограниченном объёме данных.

Таблица 5 – Влияние архитектуры на качество и сложность модели

Скрытые слои	Число параметров	Accuracy	F1	ROC-AUC
32	1089	0.749	0.624	0.845
64	2177	0.760	0.630	0.845
64-32	4289	0.774	0.636	0.845
128-64	12673	0.753	0.625	0.844
256-128	41729	0.747	0.619	0.847

При этом число обучаемых параметров возрастает многократно. Конфигурация 64–32, обладающая наилучшим сочетанием F1-меры и доли верных ответов при умеренном числе параметров, выбрана в качестве основной. Зависимость метрик от архитектуры наглядно представлена на рисунке 22.

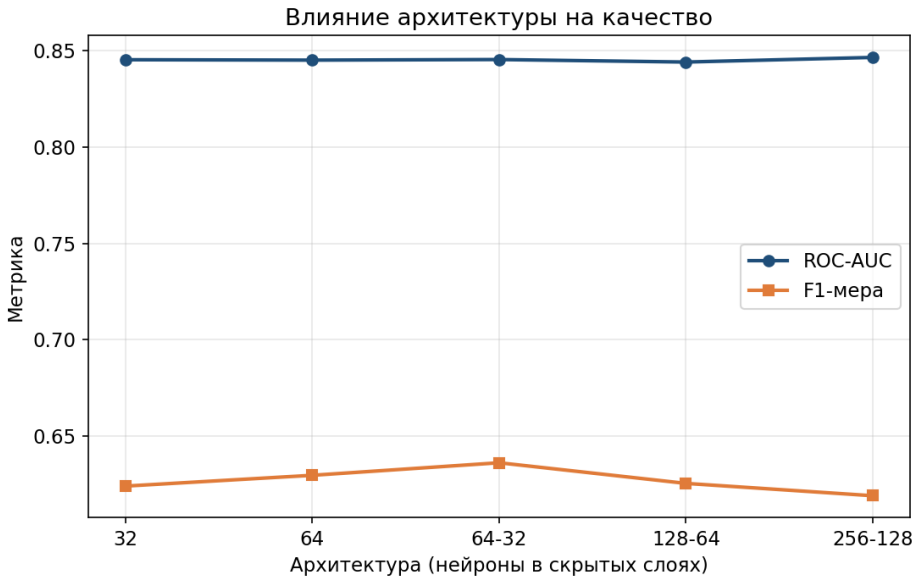


Рисунок 22 – Влияние архитектуры сети на качество

1.3.5 Исследование влияния балансировки классов

Для оценки значимости учёта дисбаланса классов проведено сравнение двух вариантов обучения нейронной сети: с балансировкой классов (взвешиванием через pos_weight) и без неё. Результаты сравнения представлены на рисунке 23.

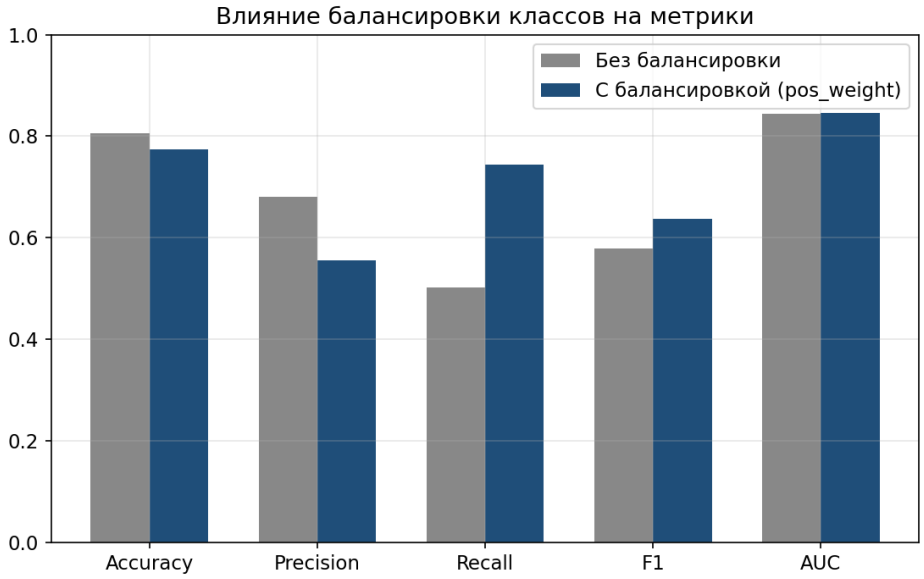


Рисунок 23 – Влияние балансировки классов на метрики качества

Рисунок 16 наглядно демонстрирует ключевой эффект балансировки. Без взвешивания классов модель достигает несколько более высокой доли верных ответов (0.805 против 0.774) и точности, однако её полнота падает с 0.744 до 0.502. Это означает, что несбалансированная модель выявляет лишь около 50% уходящих клиентов, пропуская примерно половину из них. Для задачи удержания такой результат неприемлем, поскольку именно выявление уходящих клиентов является целью системы. Балансировка классов повышает полноту ценой умеренного снижения точности, что отражается в росте F1-меры с 0.578 до 0.636. При этом значение ROC-AUC при балансировке практически не меняется, поскольку эта метрика инвариантна к порогу классификации.

Таким образом, применение балансировки классов является обоснованным и необходимым для решаемой задачи.

1.3.6 Подбор порога классификации

Нейронная сеть выдаёт на выходе вероятность оттока, которая для принятия окончательного решения сравнивается с порогом. По умолчанию порог принимается равным 0,5, однако его значение можно настраивать в зависимости от приоритета точности или полноты. Зависимость метрик от порога классификации представлена на рисунке 24.

Как видно из рисунка 24, с увеличением порога точность растёт, а полнота снижается, что отражает естественный компромисс между этими метриками. Максимум F1-меры, равный 0.638, достигается при пороге 0.51.

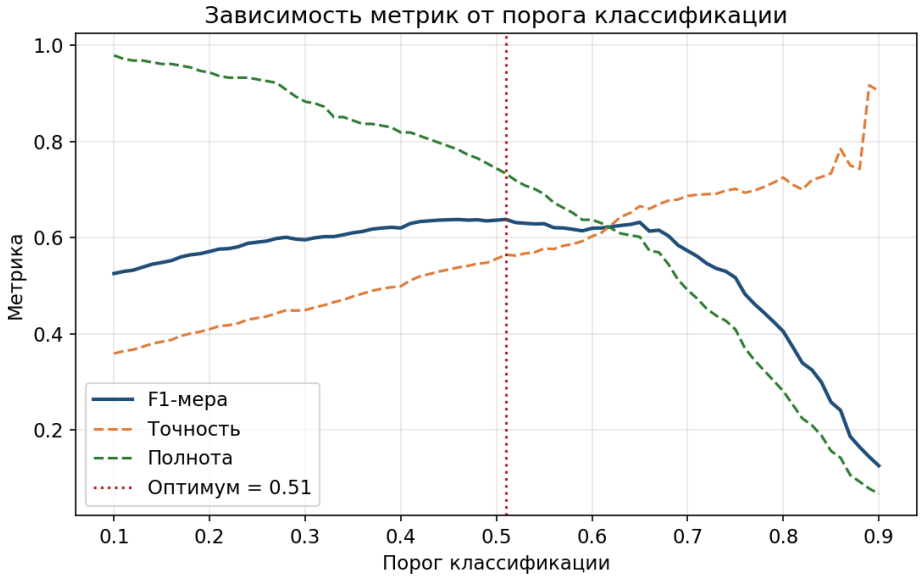


Рисунок 24 – Зависимость метрик от порога классификации

Поскольку оптимальный порог близок к стандартному значению 0,5, прирост качества от его настройки в данном случае незначителен. Тем не менее возможность регулирования порога предоставляет практическую гибкость: при

необходимости охватить как можно больше уходящих клиентов порог можно понизить, повысив полноту, а при ограниченном бюджете на удержание — повысить, сократив число ложных срабатываний.

Близость оптимального порога к стандартному значению 0,5 является косвенным свидетельством хорошей калибровки выходных вероятностей модели, достигнутой в том числе благодаря балансировке классов при обучении. На практике выбор рабочего порога целесообразно осуществлять не только по формальному максимуму F1-меры, но и с учётом экономических соображений — соотношения стоимости удерживающего мероприятия и ценности удержанного клиента. Такой подход позволяет настроить систему под конкретные бизнес-условия и добиться максимальной отдачи от её применения.

1.3.7 Анализ важности признаков

Для интерпретации модели и выявления факторов, в наибольшей степени влияющих на отток, проведён анализ важности признаков методом перестановок (permutationimportance). Метод заключается в поочерёдном случайном перемешивании значений каждого признака на тестовой выборке и измерении вызванного этим падения метрики ROC-AUC: чем сильнее падение, тем важнее признак для модели. Результаты приведены на рисунке 25.

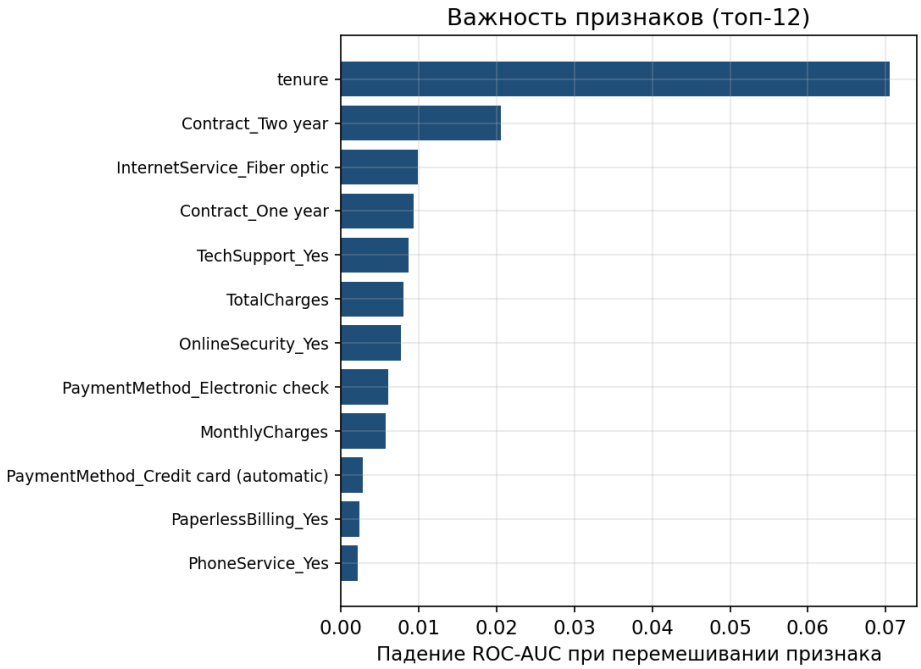


Рисунок 25 – Важность признаков по методу перестановок

Наибольшее влияние на прогноз оттока оказывает срок обслуживания (tenure), перемешивание которого снижает ROC-AUC на 0.070. Это согласуется с выводами разведочного анализа, согласно которым новые клиенты уходят значительно чаще. Высокую важность имеют также признаки, связанные с типом контракта (двухлетний и годовой контракт), типом интернет-подключения (оптоволокно) и наличием технической поддержки. Полученные результаты интерпретируемы и совпадают с практическими представлениями о поведении клиентов, что повышает доверие к модели.

Согласованность результатов анализа важности признаков с выводами разведочного анализа данных является важным свидетельством адекватности модели: сеть опирается на те же факторы, которые при предварительном изучении данных были определены как значимые предикторы оттока. Это подтверждает, что высокое качество модели достигнуто не за счёт случайных закономерностей, а благодаря улавливанию содержательных зависимостей. Полученные сведения о важности признаков имеют и самостоятельную

практическую ценность: они указывают, на какие характеристики клиента следует обращать внимание в первую очередь при разработке программ удержания, и позволяют формировать адресные предложения с учётом конкретных факторов риска.

1.3.8 Исследование гиперпараметров обучения

Качество нейронной сети зависит от ряда гиперпараметров, значения которых задаются до начала обучения. Исследовано влияние двух важнейших из них — вероятности прореживания (dropout) и скорости обучения. Влияние вероятности прореживания на качество модели представлено на рисунке 26.

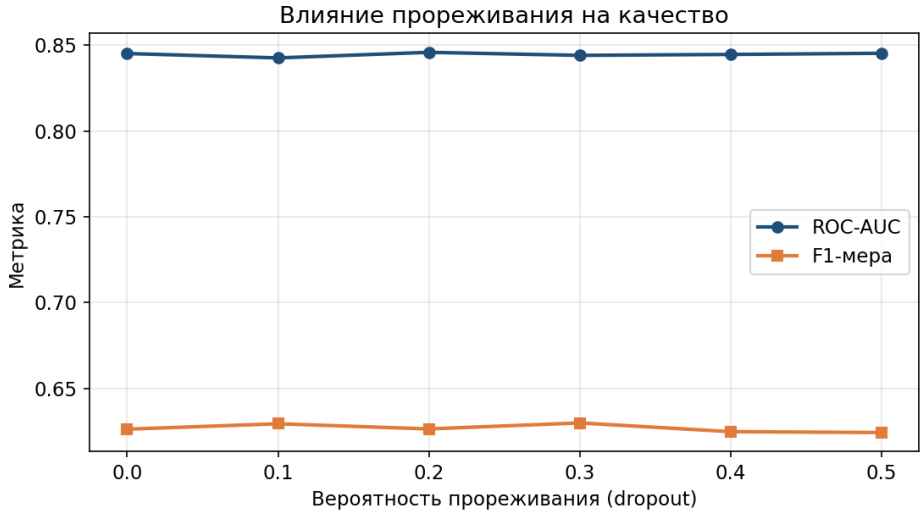


Рисунок 26 – Влияние вероятности прореживания на качество

Как видно из рисунка 26, в исследованном диапазоне от 0 до 0,5 вероятность прореживания слабо влияет на итоговое качество: значения ROC-AUC и F1-меры изменяются незначительно. Это свидетельствует об устойчивости модели и умеренной склонности к переобучению на данном наборе данных. Значение 0,3, принятое в основной модели, обеспечивает разумный компромисс между регуляризацией и сохранением обучаемости сети. Влияние скорости обучения показано на рисунке 27.

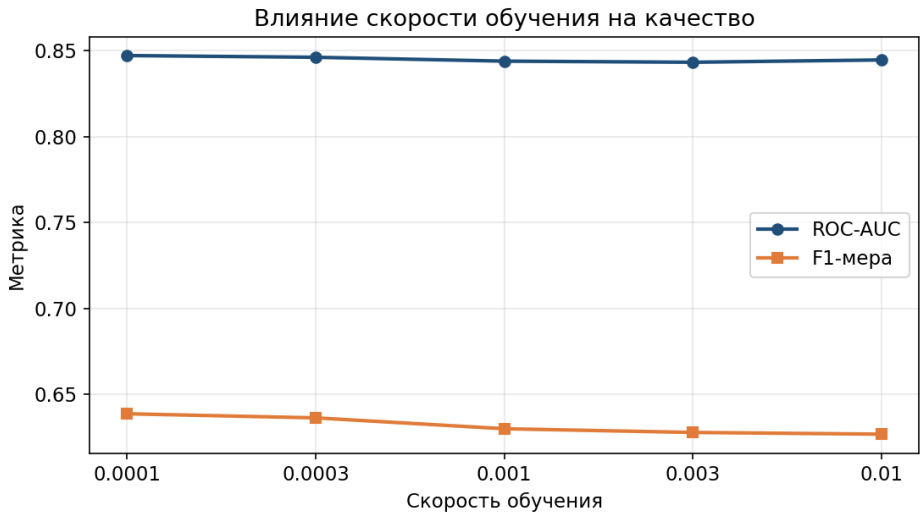


Рисунок 27 – Влияние скорости обучения на качество

Анализ рисунка 27 показывает, что наилучшее качество достигается при малых значениях скорости обучения порядка 0.0001; при её увеличении качество незначительно снижается из-за менее точной сходимости. В целом модель сохраняет устойчиво высокое качество в широком диапазоне скоростей обучения, что упрощает её настройку. Принятое в основной модели значение 0,001 находится в области устойчивой работы и обеспечивает приемлемую скорость сходимости при сохранении качества.

1.3.9 Итоговая оценка результатов

Проведённые эксперименты позволяют сделать вывод о том, что разработанная нейросетевая модель успешно решает поставленную задачу прогнозирования оттока клиентов. Модель обеспечивает значение ROC-AUC 0.845 и полноту 0.744 на независимой тестовой выборке, что соответствует уровню лучших известных решений для использованного набора данных и отвечает практическим требованиям задачи удержания.

Серия дополнительных экспериментов подтвердила обоснованность принятых проектных решений: выбранная архитектура из двух скрытых слоёв является оптимальной по соотношению качества и сложности; балансировка

классов необходима для достижения высокой полноты; нейронная сеть превосходит базовые модели по сбалансированности качества. Анализ важности признаков выявил ключевые факторы оттока, что придаёт модели практическую интерпретируемость.

Разработанная модель и методика её обучения могут быть применены для построения систем удержания клиентов в реальных подписных сервисах. Для внедрения в промышленную эксплуатацию модель целесообразно дообучать на актуальных данных конкретного сервиса и интегрировать в контур работы с клиентами через программный интерфейс.

1.3.10 Анализ ошибок и практическое применение модели

Для понимания характера работы модели проведён анализ допускаемых ею ошибок. Ошибки классификации делятся на два типа, имеющих различные последствия для бизнеса. Ложноположительные ошибки (модель относит лояльного клиента к группе риска) приводят к избыточным затратам на удержание клиента, который и так не собирался уходить. Ложноотрицательные ошибки (модель не распознаёт уходящего клиента) приводят к потере клиента, которого можно было бы удержать, и потому в задаче удержания считаются более критичными.

На тестовой выборке модель допустила 167 ложноположительных и 72 ложноотрицательных ошибок. Преобладание ложноположительных ошибок над ложноотрицательными является следствием применённой балансировки классов и соответствует приоритету полноты над точностью, обоснованному спецификой задачи. Сводка типов ошибок и их интерпретация приведены в таблице 6.

Таблица 6 – Типы ошибок классификации и их интерпретация

Ситуация	Прогноз	Факт	Бизнес-инерпретация
Истинно положительный	уйдет	ушел	Клиент верно выявлен, удержание оправдано
Истинно отрицательный	останется	остался	Ресурсы на удержание не расходуются
Ложноположительный	уйдет	остался	Избыточные затраты на удержание
Ложноотрицательный	останется	ушел	Упущенный клиент, потеря выручки

Ситуация Прогноз Факт Бизнес-интерпретация Истинно положительный уйдёт ушёл клиент верно выявлен, удержание оправдано Истинно отрицательный останется остался ресурсы на удержание не расходуются Ложноположительный уйдёт остался избыточные затраты на удержание Ложноотрицательный останется ушёл упущенный клиент, потеря выручки Практическое применение разработанной модели предполагает её интеграцию в информационную систему подписного сервиса. Обученная модель периодически применяется к актуальной базе клиентов, формируя для каждого клиента оценку вероятности оттока. Клиенты с высокой оценкой передаются в пунктение по работе с клиентами для проведения удерживающих мероприятий. При этом возможность регулирования порога классификации позволяет управлять числом клиентов, попадающих в группу риска, в зависимости от доступного бюджета на удержание.

Типовая схема внедрения включает следующие этапы: сбор и предобработку данных о клиентах, периодическое дообучение модели на актуальных данных, пакетное применение модели для оценки вероятности оттока, передачу результатов в систему управления взаимоотношениями с клиентами и проведение адресных удерживающих кампаний. Поскольку модель обучается быстро и не требует специализированного оборудования, её дообучение на свежих данных может выполняться регулярно, что обеспечивает поддержание актуальности прогнозов при изменении поведения клиентов с течением времени.

Важным преимуществом разработанного решения является его прозрачность: проведённый анализ важности признаков позволяет не только прогнозировать отток, но и указывать на вероятные его причины, что даёт основу для содержательных удерживающих предложений. Например, для клиентов с малым сроком обслуживания и помесячным контрактом эффективной мерой может стать предложение перейти на годовой контракт на льготных условиях.

1.3.11 Оценка вычислительной эффективности модели

Помимо качества прогноза, для практического применения модели важна её вычислительная эффективность. Разработанная нейронная сеть отличается компактностью: общее число её обучаемых параметров составляет 4289, что является малой величиной по меркам современных нейронных сетей. Благодаря этому модель не требует специализированного оборудования и обучается за считанные секунды на обычном центральном процессоре, без привлечения графического ускорителя.

Малое число параметров обеспечивает и высокую скорость применения модели: вычисление прогноза для одного клиента сводится к нескольким операциям умножения матриц небольшого размера и выполняется практически

мгновенно. Это позволяет применять модель как в пакетном режиме – для периодической оценки всей клиентской базы, так и в режиме реального времени – для оценки отдельного клиента по запросу. Низкие требования к вычислительным ресурсам делают возможным регулярное дообучение модели на актуальных данных без существенных затрат.

Сравнение с альтернативными конфигурациями, проведённое в пункте 1.3.4, показало, что увеличение размера сети не приводит к росту качества, но многократно увеличивает число параметров и время обучения. Таким образом, выбранная компактная архитектура является оптимальной не только по качеству прогноза, но и по вычислительной эффективности, что подтверждает обоснованность принятых проектных решений и пригодность модели для практического применения в условиях ограниченных ресурсов.

2 Охрана труда

2.1 Анализ условий труда при работе с ЭВМ

Разработка системы прогнозирования оттока клиентов осуществляется с применением электронно-вычислительных машин (ЭВМ). Труд разработчика программного обеспечения относится к категории работ, выполняемых с использованием персональных компьютеров, и сопряжён с воздействием на работника ряда вредных производственных факторов. Требования к организации таких рабочих мест регламентируются санитарными правилами и нормами, в частности СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания», а также соответствующими государственными стандартами системы стандартов безопасности труда (ССБТ).

К основным вредным факторам при работе с ЭВМ относятся: повышенный уровень электромагнитных излучений, статическое электричество, неблагоприятные параметры микроклимата, недостаточная или избыточная освещённость, повышенный уровень шума, зрительное и нервно-эмоциональное напряжение, а также монотонность труда и длительное пребывание в фиксированной рабочей позе. Кроме того, ЭВМ являются электроустановками, что обуславливает опасность поражения электрическим током.

Среди вредных факторов особое внимание уделяется электромагнитным излучениям и шуму. Современные жидкокристаллические мониторы характеризуются существенно более низкими уровнями электромагнитного излучения по сравнению с устаревшими электронно-лучевыми и удовлетворяют действующим гигиеническим нормативам. Дополнительными мерами защиты являются соблюдение рекомендуемого расстояния до экрана и ограничение непрерывного времени работы. Уровень шума, создаваемого вентиляторами

систем охлаждения и периферийным оборудованием, на рабочих местах, связанных с программированием, согласно гигиеническим нормативам не должен превышать 50 дБА; для его снижения применяются малошумное оборудование и звукопоглощающие материалы.

2.2 Требования к микроклимату и освещению

Параметры микроклимата рабочего помещения оказывают существенное влияние на работоспособность и самочувствие работника. Работа разработчика программного обеспечения по уровню энергозатрат относится к категории Ia (работы с интенсивностью энергозатрат до 139 Вт, производимые сидя и сопровождающиеся незначительным физическим напряжением). Для данной категории в холодный период года оптимальная температура воздуха составляет 22–24 °C, в тёплый период — 23–25 °C. Относительная влажность воздуха должна находиться в пределах 40–60%, скорость движения воздуха не должна превышать 0,1 м/с. Для поддержания требуемых параметров микроклимата помещение оборудуется системами отопления, вентиляции и кондиционирования воздуха.

Рациональное освещение рабочего места является важным условием сохранения зрения и поддержания работоспособности. Освещение должно быть смешанным — естественным и искусственным. Естественное освещение обеспечивается через оконные проёмы и характеризуется коэффициентом естественной освещённости. Искусственное освещение должно обеспечивать освещённость рабочей поверхности на уровне 300–500 лк. Для исключения слепящего действия и бликов на экране монитора светильники располагают сбоку от рабочего места, а рабочий стол размещают так, чтобы экран располагался под углом к оконным проёмам. Применяются светильники с рассеивателями, обеспечивающие равномерное освещение без резких теней и блёскости.

2.3 Требования к организации рабочего места

Эргономичная организация рабочего места снижает утомляемость и предупреждает развитие заболеваний опорно-двигательного аппарата. Конструкция рабочего стола должна обеспечивать оптимальное размещение оборудования с учётом его количества и конструктивных особенностей. Высота рабочей поверхности стола должна составлять около 725 мм, рабочий стул должен быть подъёмно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки.

Экран монитора располагается на расстоянии 600–700 мм от глаз работника, верхняя граница экрана должна находиться на уровне глаз или несколько ниже. Клавиатура размещается на поверхности стола на расстоянии 100–300 мм от края. Для снижения зрительного утомления необходимо периодически переводить взгляд на удалённые объекты и соблюдать режим труда и отдыха.

Согласно требованиям, при работе с ЭВМ предусматриваются регламентированные перерывы. При восьмичасовой рабочей смене суммарное время перерывов составляет от 50 до 90 минут в зависимости от категории и тяжести работы. Перерывы целесообразно проводить с выполнением комплексов упражнений для глаз, кистей рук и опорно-двигательного аппарата.

В зависимости от характера работы с ЭВМ устанавливаются три категории трудовой деятельности: категория А — работа по считыванию информации с экрана, категория Б — работа по вводу информации, категория В — творческая работа в режиме диалога с ЭВМ, к которой относится труд разработчика программного обеспечения. Для каждой категории нормируется суммарное число обрабатываемых знаков за смену и продолжительность регламентированных перерывов. Соблюдение установленного режима труда и

отдыха является необходимым условием сохранения работоспособности и предупреждения профессиональных заболеваний.

2.4 Электробезопасность

Персональный компьютер и периферийное оборудование питаются от электрической сети переменного тока напряжением 220 В, что создаёт опасность поражения электрическим током. По степени опасности поражения электрическим током помещения с ЭВМ, как правило, относятся к категории помещений без повышенной опасности при условии соблюдения требований по влажности, температуре и отсутствию токопроводящих полов и пыли.

Для обеспечения электробезопасности применяется комплекс мер: защитное заземление и зануление оборудования, применение устройств защитного отключения, изоляция токоведущих частей, использование оборудования, соответствующего требованиям безопасности. Запрещается эксплуатация оборудования с повреждённой изоляцией питающих кабелей, а также самостоятельное вскрытие и ремонт находящегося под напряжением оборудования. Все работники должны быть проинструктированы по правилам электробезопасности.

Помимо технических мер, важную роль играют организационные мероприятия: назначение ответственных за электрохозяйство лиц, периодическая проверка состояния изоляции и заземляющих устройств, ведение эксплуатационной документации, а также регулярный инструктаж и проверка знаний работников. Совокупность технических и организационных мер обеспечивает требуемый уровень электробезопасности на рабочем месте оператора ЭВМ.

2.5 Пожарная безопасность

Помещения с ЭВМ по пожарной и взрывопожарной опасности относятся к категории В (пожароопасные), поскольку содержат твёрдые горючие материалы (мебель, изоляция кабелей, корпуса оборудования, бумага). Источниками возгорания могут служить короткие замыкания в электропроводке, перегрузка сети, неисправность оборудования.

Для предупреждения пожаров применяются следующие меры: соблюдение правил эксплуатации электрооборудования, своевременное техническое обслуживание, оснащение помещений системами автоматической пожарной сигнализации и первичными средствами пожаротушения. Для тушения пожаров в помещениях с ЭВМ применяются углекислотные или порошковые огнетушители, не повреждающие электронное оборудование и безопасные при тушении электроустановок под напряжением. Применение воды и пенных огнетушителей для тушения находящегося под напряжением оборудования недопустимо. Помещения оборудуются эвакуационными путями и выходами, а работники проходят инструктаж по действиям в случае возникновения пожара.

Здание, в котором расположены помещения с ЭВМ, должно быть оборудовано системой оповещения и управления эвакуацией людей при пожаре, а пути эвакуации — обозначены указателями и поддерживаться свободными. Расчёт времени эвакуации и числа эвакуационных выходов производится исходя из площади помещений и числа работающих. Регулярное проведение учебных эвакуаций и поддержание первичных средств пожаротушения в исправном состоянии существенно повышают готовность персонала к действиям в случае возникновения пожара и снижают возможный ущерб.

2.6 Выводы по разделу

В разделе проведён анализ условий труда при разработке системы прогнозирования оттока клиентов с применением ЭВМ. Рассмотрены основные вредные производственные факторы и сформулированы требования к микроклимату, освещению, организации рабочего места, электро и пожарной безопасности. Соблюдение перечисленных требований и нормативов обеспечивает безопасные и комфортные условия труда разработчика, способствует сохранению его здоровья и поддержанию высокой работоспособности.

Создание безопасных условий труда является не только требованием законодательства, но и фактором повышения производительности и качества работы. Комфортная и безопасная рабочая среда снижает утомляемость, уменьшает число ошибок и способствует сохранению здоровья работников в долгосрочной перспективе. Поэтому учёт требований охраны труда должен осуществляться уже на этапе организации рабочего места разработчика, а их соблюдение — контролироваться на протяжении всего процесса разработки программного обеспечения.

Заключение

В ходе выполнения бакалаврской работы разработана и экспериментально исследована система прогнозирования оттока клиентов в подписных сервисах на основе искусственной нейронной сети. Поставленная цель работы достигнута, все задачи решены в полном объёме.

Проведён анализ предметной области, в рамках которого рассмотрены понятие оттока клиентов, его виды и показатели, обоснована экономическая значимость прогнозирования оттока для подписного бизнеса, выполнен обзор существующих решений и методов машинного обучения. На основании анализа обоснован выбор искусственной нейронной сети в качестве основной модели, а также выбраны инструментальные средства разработки: язык Python, библиотека PyTorch и облачная среда GoogleColab.

Выбран и подготовлен открытый набор данных TelcoCustomerChurn, описывающий клиентов телекоммуникационного подписного сервиса. Проведён разведочный анализ данных, выявивший ключевые закономерности оттока, и выполнена предобработка данных, включающая очистку, кодирование категориальных признаков, стандартизацию и стратифицированное разбиение на выборки.

Спроектирована и реализована нейронная сеть — многослойный перцептрон с двумя скрытыми слоями, пакетной нормализацией и прореживанием. Реализован алгоритм обучения с применением оптимизатора Adam, учётом дисбаланса классов и ранней остановкой. Качество модели оценено по совокупности метрик на независимой тестовой выборке.

Разработанная модель обеспечила значение метрики ROC-AUC 0.845 и полноту 0.744, что соответствует уровню лучших известных решений для использованного набора данных. Проведена серия дополнительных экспериментов: сравнение с базовыми моделями (логистической регрессией и

случайным лесом), исследование влияния архитектуры сети, балансировки классов и порога классификации, а также анализ важности признаков. Эксперименты подтвердили обоснованность принятых проектных решений и показали, что нейронная сеть обеспечивает наилучший баланс качества среди рассмотренных моделей.

Практическая значимость работы заключается в том, что разработанная модель и методика её обучения могут быть использованы для построения систем удержания клиентов в реальных подписных сервисах. Направлениями дальнейшего развития работы являются применение более сложных архитектур и методов ансамблирования, учёт временной динамики поведения клиентов, а также интеграция модели в программный контур работы с клиентами через программный интерфейс.

Полученные в работе результаты подтверждают, что применение искусственных нейронных сетей является эффективным подходом к решению задачи прогнозирования оттока клиентов в подписных сервисах. Разработанная методика, включающая предобработку данных, проектирование архитектуры сети, учёт дисбаланса классов и комплексную оценку качества, носит универсальный характер и может быть адаптирована для различных подписных сервисов — телекоммуникационных операторов, стриминговых платформ, облачных и иных сервисов, работающих по подписной модели. Практическая значимость работы состоит в готовности предложенного решения к адаптации и внедрению, а теоретическая — в систематизации методов и экспериментальном обосновании принятых проектных решений.

Список использованных источников

1. Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль ; пер. с англ. — Москва : ДМК Пресс, 2018. — 652 с.

2. Николенко, С. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. — Санкт-Петербург : Питер, 2018. — 480 с.

3. Рашка, С. Python и машинное обучение / С. Рашка ; пер. с англ. — Москва : ДМК Пресс, 2017. — 418 с.

4. Джулли, А. Библиотека Keras — инструмент глубокого обучения / А. Джулли, С. Пал ; пер. с англ. — Москва : ДМК Пресс, 2018. — 294 с.

5. Хайкин, С. Нейронные сети: полный курс / С. Хайкин ; пер. с англ. — 2-е изд. — Москва : Вильямс, 2006. — 1104 с.

6. PyTorchdocumentation [Электронный ресурс]. — Режим доступа: https://pytorch.org/docs (дата обращения: 15.05.2025).

7. scikit-learn: machinelearninginPython [Электронный ресурс]. — Режим доступа: https://scikit-learn.org (дата обращения: 15.05.2025).

8. TelcoCustomerChurn [Электронный ресурс] : набор данных. — Режим доступа: https://www.kaggle.com/datasets/blastchar/telco-customer-churn (дата обращения: 10.05.2025).

9. pandasdocumentation [Электронный ресурс]. — Режим доступа: https://pandas.pydata.org/docs (дата обращения: 15.05.2025).

10. Matplotlib: VisualizationwithPython [Электронный ресурс]. — Режим доступа: https://matplotlib.org (дата обращения: 15.05.2025).

11. Kingma, D. P. Adam: A Method for Stochastic Optimization / D. P. Kingma, J. Ba // arXiv preprint arXiv:1412.6980. — 2014.

12. Ioffe, S. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / S. Ioffe, C. Szegedy // arXiv preprint arXiv:1502.03167. — 2015.

13. Srivastava, N. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / N. Srivastava [et al.] // Journal of Machine Learning Research. — 2014. — Vol. 15. — P. 1929–1958.

14. Provost, F. Data Science for Business / F. Provost, T. Fawcett. — Sebastopol : O'Reilly Media, 2013. — 414 p.

15. Фоулер, Дж. Машинное обучение в бизнесе / Дж. Фоулер ; пер. с англ. — Москва : ДМК Пресс, 2020. — 320 с.

16. GoogleColaboratory [Электронный ресурс]. — Режим доступа: https://colab.research.google.com (дата обращения: 15.05.2025).

17. СанПиН 1.2.3685-21. Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания. — Москва, 2021. — 469 с.

18. ГОСТ 12.2.032-78. Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования. — Москва : Издательство стандартов, 1986. — 9 с.

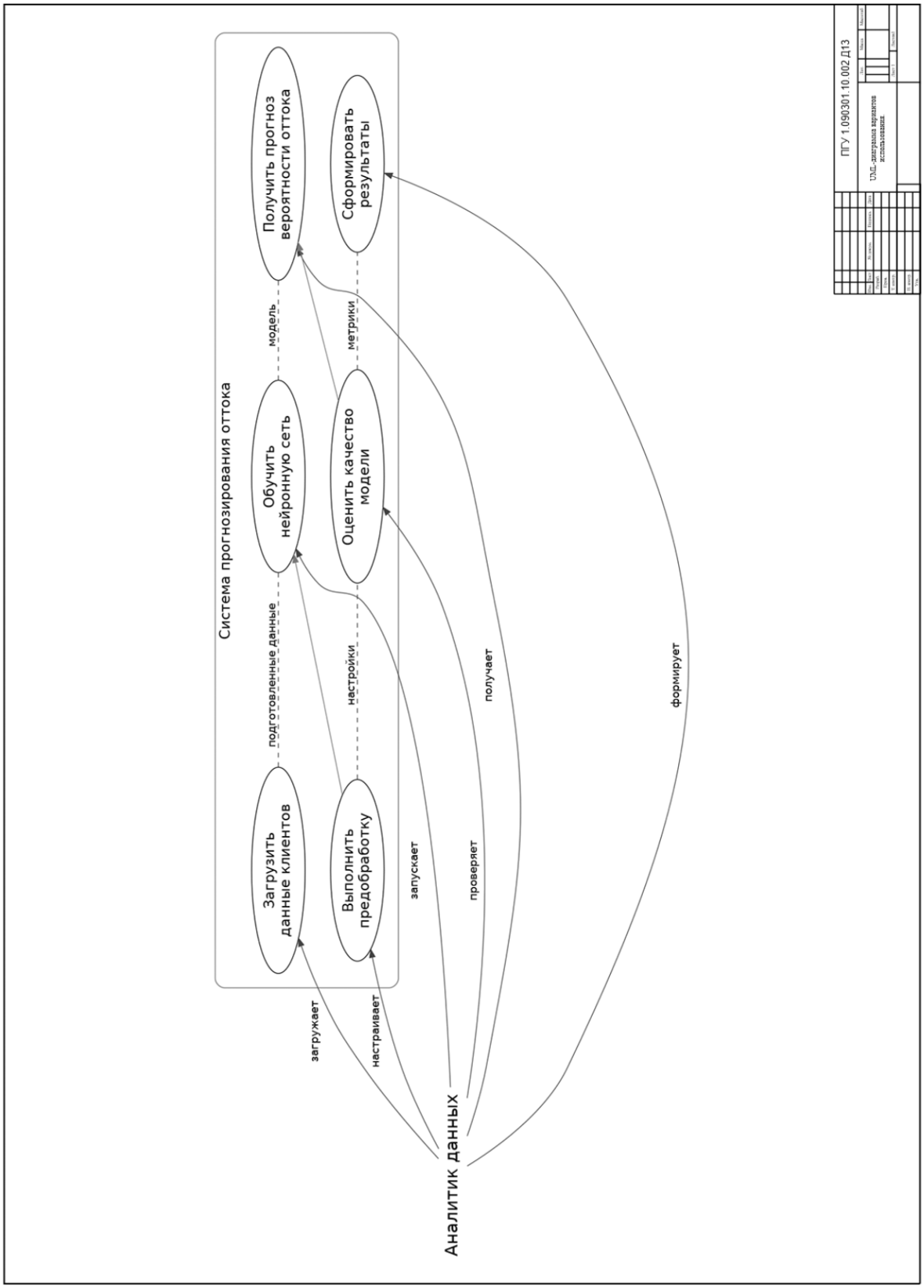
Приложение А

(обязательное)

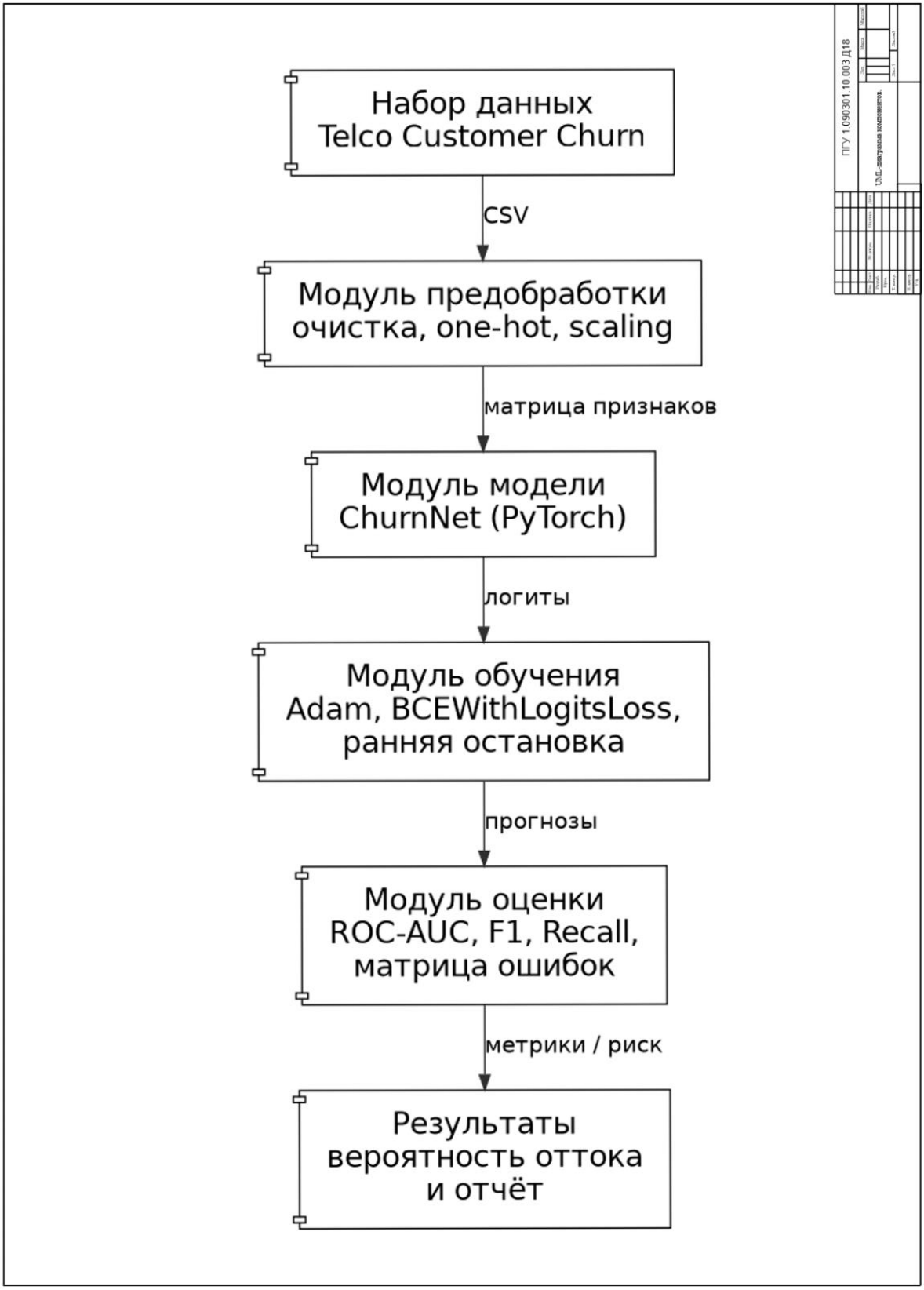
Система прогнозирования оттока клиентов

UML-диаграммы

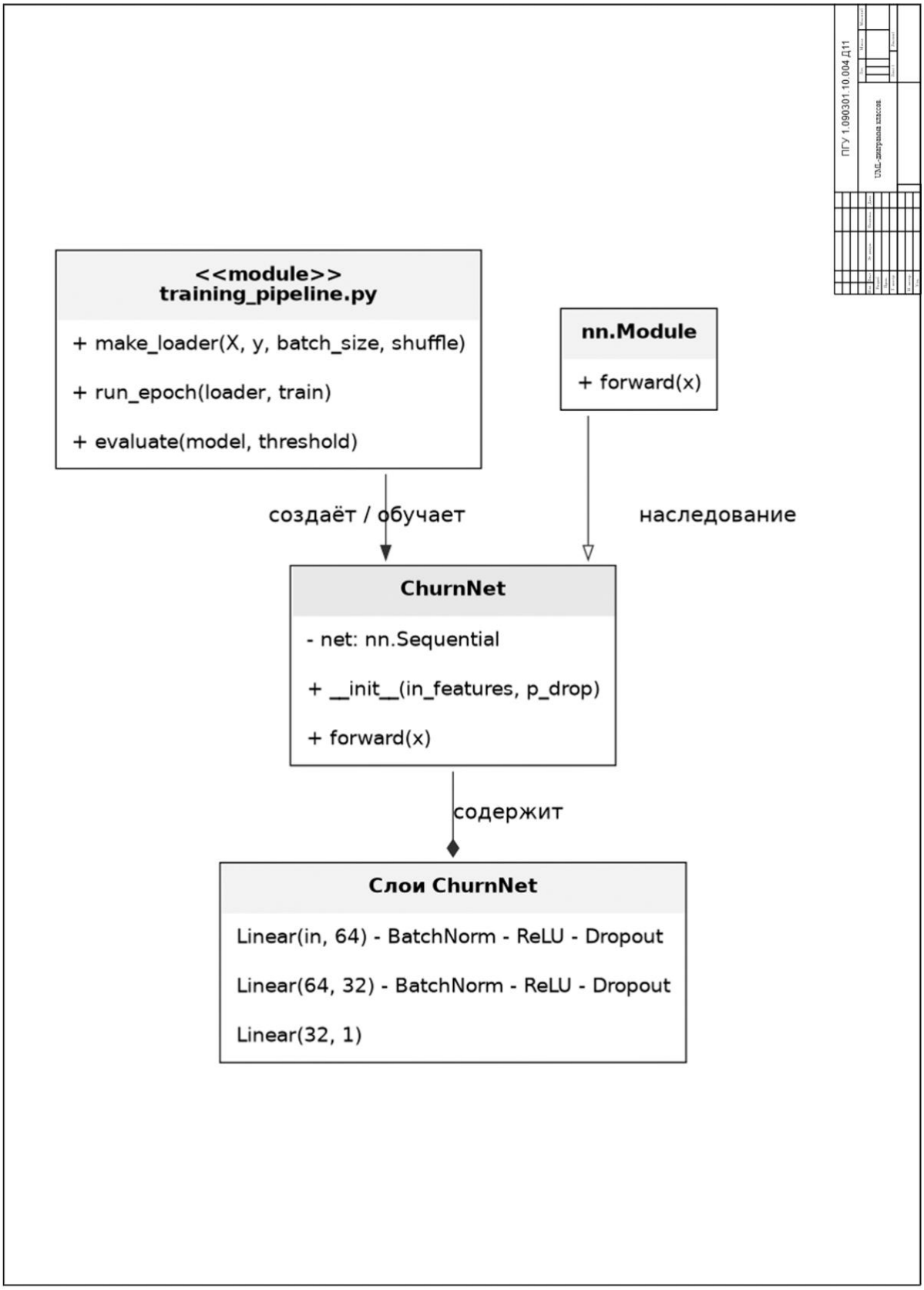
UML-диаграмма вариантов использования.



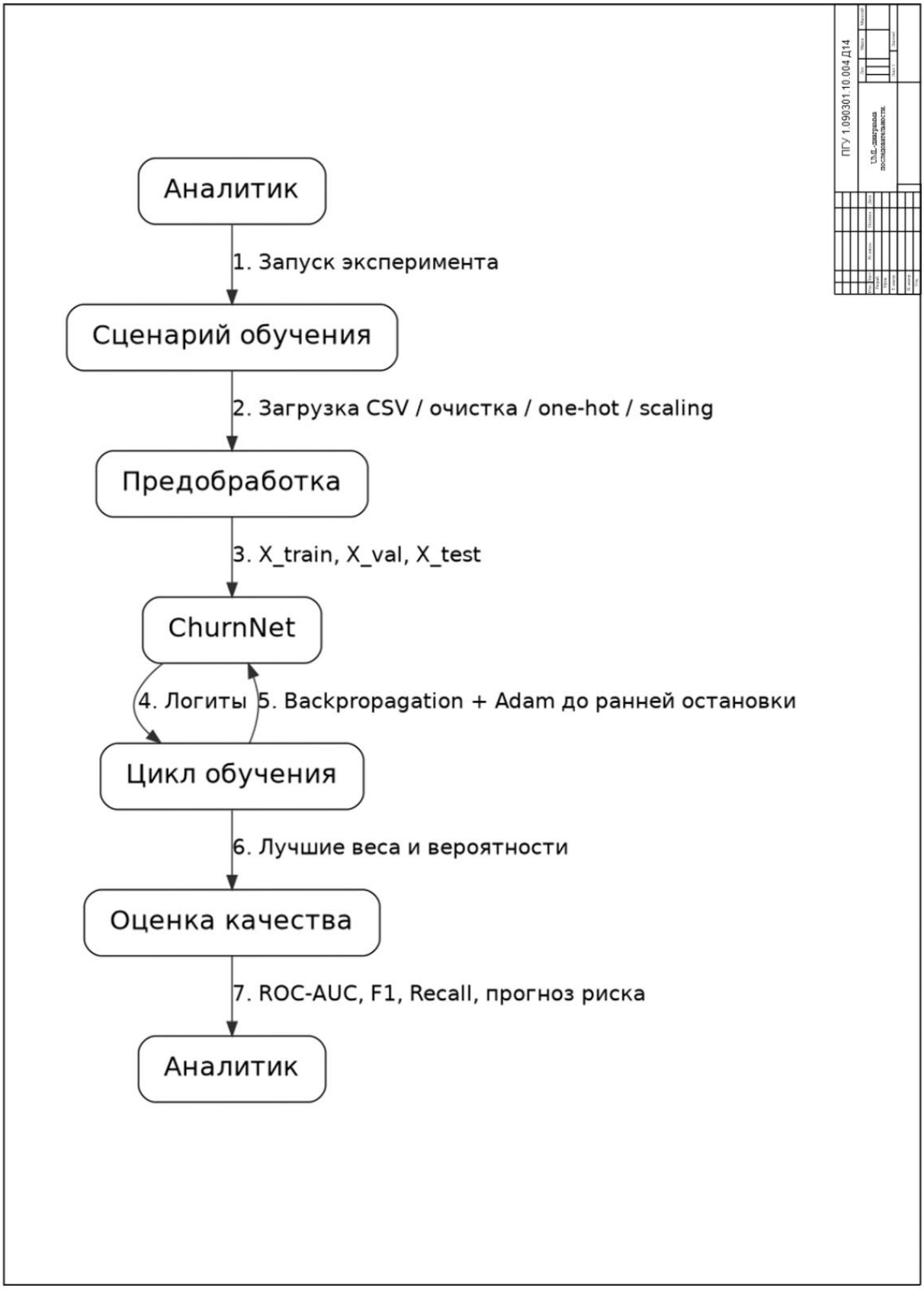
UML-диаграмма компонентов.



UML-диаграмма классов



UML-диаграмма последовательности.



Приложение Б

(обязательное)

Система прогнозирования оттока клиентов

Листинг программы

```
# -*- coding: utf-8

import numpy as np
import pandas as pd
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, roc_curve, confusion_matrix,
precision_recall_curve,
)

import matplotlib.pyplot as plt

# ------------------------------------------------------
# 0. Воспроизводимость и устройство
# ------------------------------------------------------
SEED = 42
np.random.seed(SEED)
torch.manual_seed(SEED)

device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
print(f"Используемое устройство: {device}")

# ------------------------------------------------------
# 1. Загрузкаданных
# ------------------------------------------------------
DATA_URL = (
    "https://raw.githubusercontent.com/IBM/telco-customer-churn-on-
icp4d/"
    "master/data/Telco-Customer-Churn.csv"
)
df = pd.read_csv(DATA_URL)
print(f"Размернабораданных: {df.shape[0]} клиентов, {df.shape[1]}
признаков")

# ------------------------------------------------------# 2.
Предобработка
# ------------------------------------------------------

# 2.1 Удаляем идентификатор клиента — он не несёт предсказательной
информации.
df = df.drop(columns=["customerID"])
```

```
# 2.2 Столбец TotalCharges считан как текст: у 11 новых клиентов
(tenure=0)
#     там пробел. Преобразуем в число, пустые значения заполняем 0.
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"],
errors="coerce").fillna(0.0)

# 2.3 Целевая переменная: Yes -> 1 (ушёл), No -> 0 (остался).
y = (df["Churn"] == "Yes").astype(np.float32).values
df = df.drop(columns=["Churn"])

# 2.4 Числовые и категориальные признаки.
numeric_cols = ["tenure", "MonthlyCharges", "TotalCharges",
"SeniorCitizen"]
categorical_cols = [cforcindf.columnsifcnotinnumeric_cols]

# 2.5 One-hot кодирование категориальных признаков.
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
X = df.values.astype(np.float32)
feature_names = df.columns.tolist()
print(f"Признаков после кодирования: {X.shape[1]}")

# 2.6 Разбиение: 70% обучение, 15% валидация, 15% тест (со
стратификацией,
#     чтобы доля ушедших клиентов сохранялась во всех выборках).
X_train, X_tmp, y_train, y_tmp = train_test_split(
    X, y, test_size=0.30, random_state=SEED, stratify=y
)
X_val, X_test, y_val, y_test = train_test_split(
    X_tmp, y_tmp, test_size=0.50, random_state=SEED, stratify=y_tmp
)

# 2.7 Масштабирование числовых признаков. Параметры считаем ТОЛЬКО
по
#     обучающей выборке, чтобы не было утечки данных.
scaler = StandardScaler()
num_idx = [feature_names.index(c) for c in numeric_cols]
X_train[:, num_idx] = scaler.fit_transform(X_train[:, num_idx])
X_val[:, num_idx] = scaler.transform(X_val[:, num_idx])
X_test[:, num_idx] = scaler.transform(X_test[:, num_idx])

print(f"Обучение: {len(y_train)} | Валидация: {len(y_val)} | Тест:
{len(y_test)}")
print(f"Доля ушедших клиентов (обучение): {y_train.mean():.3f}")

# 2.8 Тензоры и загрузчики данных.
def make_loader(Xa, ya, batch_size=64, shuffle=False):
    ds = TensorDataset(torch.from_numpy(Xa),
torch.from_numpy(ya).unsqueeze(1))
    return DataLoader(ds, batch_size=batch_size, shuffle=shuffle)
```

```
train_loader = make_loader(X_train, y_train, batch_size=64,
shuffle=True)
val_loader = make_loader(X_val, y_val, batch_size=256)

# ----------------------------------------------------------
# 3. Модель — компактныймногослойныйперцептрон.
# ----------------------------------------------------------
class ChurnNet(nn.Module):
    def __init__(self, in_features, p_drop=0.3):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(in_features, 64),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.Dropout(p_drop),

            nn.Linear(64, 32),
            nn.BatchNorm1d(32),
            nn.ReLU(),
            nn.Dropout(p_drop),

            nn.Linear(32, 1),  # одинвыход (логит),
сигмоидуприменяемприоценке
        )

    def forward(self, x):
        return self.net(x)

model = ChurnNet(X.shape[1]).to(device)
n_params = sum(p.numel() for p in model.parameters())
print(f"Параметроввмодели: {n_params}")

# ----------------------------------------------------------
# 4. Функцияпотерьиоптимизатор.
# ----------------------------------------------------------
pos_weight = torch.tensor([(y_train == 0).sum() / (y_train ==
1).sum()],
                          dtype=torch.float32, device=device)
criterion = nn.BCEWithLogitsLoss(pos_weight=pos_weight)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3,
weight_decay=1e-4)

# ----------------------------------------------------------
# 5. Цикл обучения с ранней остановкой по валидационной потере.
# ----------------------------------------------------------
EPOCHS = 80
PATIENCE = 12  # сколько эпох ждать улучшения перед остановкой
```

```
history = {"train_loss": [], "val_loss": [], "train_acc": [],
"val_acc": []}
best_val_loss = float("inf")
best_state = None
epochs_no_improve = 0


def run_epoch(loader, train=True):
    model.train() if train else model.eval()
    total_loss, correct, total = 0.0, 0, 0
    with torch.set_grad_enabled(train):
        for xb, yb in loader:
            xb, yb = xb.to(device), yb.to(device)
            logits = model(xb)
            loss = criterion(logits, yb)
            if train:
                optimizer.zero_grad()
                loss.backward()
                optimizer.step()
            total_loss += loss.item() * xb.size(0)
            preds = (torch.sigmoid(logits) >= 0.5).float()
            correct += (preds == yb).sum().item()
            total += xb.size(0)
    return total_loss / total, correct / total


for epoch in range(1, EPOCHS + 1):
    tr_loss, tr_acc = run_epoch(train_loader, train=True)
    va_loss, va_acc = run_epoch(val_loader, train=False)

    history["train_loss"].append(tr_loss)
    history["val_loss"].append(va_loss)
    history["train_acc"].append(tr_acc)
    history["val_acc"].append(va_acc)

    if va_loss < best_val_loss:
        best_val_loss = va_loss
        best_state = {k: v.cpu().clone() for k, v in
model.state_dict().items()}
        epochs_no_improve = 0
    else:
        epochs_no_improve += 1

    if epoch % 5 == 0 or epoch == 1:
        print(f"Эпоха {epoch:3d} | "
              f"loss: {tr_loss:.4f}/{va_loss:.4f} | "
              f"acc: {tr_acc:.3f}/{va_acc:.3f}")

    if epochs_no_improve >= PATIENCE:
```

```
        print(f"Ранняя остановка на эпохе {epoch} (нет улучшения
{PATIENCE} эпох)")
        break

# Восстанавливаем лучшие веса.
model.load_state_dict(best_state)

# ----------------------------------------------------------
# 6. Оценканатестовойвыборке.
# ----------------------------------------------------------
model.eval()
withtorch.no_grad():
logits_test = model(torch.from_numpy(X_test).to(device))
    probs_test = torch.sigmoid(logits_test).cpu().numpy().ravel()

preds_test = (probs_test >= 0.5).astype(int)

acc = accuracy_score(y_test, preds_test)
prec = precision_score(y_test, preds_test)
rec = recall_score(y_test, preds_test)
f1 = f1_score(y_test, preds_test)
auc = roc_auc_score(y_test, probs_test)

print("\n===== Результаты на тестовой выборке =====")
print(f"Accuracy  (доля верных):     {acc:.3f}")
print(f"Precision (точность):        {prec:.3f}")
print(f"Recall    (полнота):         {rec:.3f}")
print(f"F1-score:                    {f1:.3f}")
print(f"ROC-AUC:                     {auc:.3f}")

# ----------------------------------------------------------
# 7. Графики (matplotlib).
# ----------------------------------------------------------
epochs_range = range(1, len(history["train_loss"]) + 1)

# 7.1 Кривые функции потерь.
plt.figure(figsize=(7, 5))
plt.plot(epochs_range, history["train_loss"], label="Обучение")
plt.plot(epochs_range, history["val_loss"], label="Валидация")
plt.xlabel("Эпоха"); plt.ylabel("Функция потерь (BCE)")
plt.title("Динамика функции потерь")
plt.legend(); plt.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig("fig_loss.png", dpi=150);
plt.close()

# 7.2 Кривые точности.
plt.figure(figsize=(7, 5))
plt.plot(epochs_range, history["train_acc"], label="Обучение")
plt.plot(epochs_range, history["val_acc"], label="Валидация")
```

```
plt.xlabel("Эпоха"); plt.ylabel("Accuracy")
plt.title("Динамика точности классификации")
plt.legend(); plt.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig("fig_accuracy.png", dpi=150);
plt.close()

# 7.3 ROC-кривая.
fpr, tpr, _ = roc_curve(y_test, probs_test)
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, label=f"ROC (AUC = {auc:.3f})")
plt.plot([0, 1], [0, 1], "--", color="gray", label="Случайный
классификатор")
plt.xlabel("Доля ложноположительных (FPR)")
plt.ylabel("Доля истинноположительных (TPR)")
plt.title("ROC-кривая")
plt.legend(); plt.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig("fig_roc.png", dpi=150);
plt.close()

# 7.4 Матрица ошибок.
cm = confusion_matrix(y_test, preds_test)
plt.figure(figsize=(5.5, 5))
plt.imshow(cm, cmap="Blues")
plt.title("Матрица ошибок")
plt.colorbar()
plt.xticks([0, 1], ["Остался", "Ушёл"])
plt.yticks([0, 1], ["Остался", "Ушёл"])
plt.xlabel("Прогноз"); plt.ylabel("Истина")
thresh = cm.max() / 2
for i in range(2):
    for j in range(2):
        plt.text(j, i, str(cm[i, j]), ha="center", va="center",
                 color="white" if cm[i, j] > thresh else "black",
fontsize=14)
plt.tight_layout(); plt.savefig("fig_confusion.png", dpi=150);
plt.close()

# 7.5 Precision-Recall кривая (полезна при несбалансированных
классах).
pr_p, pr_r, _ = precision_recall_curve(y_test, probs_test)
plt.figure(figsize=(6, 6))
plt.plot(pr_r, pr_p)
plt.xlabel("Полнота (Recall)"); plt.ylabel("Точность (Precision)")
plt.title("Precision-Recall кривая")
plt.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig("fig_pr.png", dpi=150); plt.close()

print("\nГрафики сохранены: fig_loss.png, fig_accuracy.png,
```

```
fig_roc.png, "
      "fig_confusion.png, fig_pr.png")



# -*- coding: utf-8 -*-

import json, warnings, numpy as np, pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import torch, torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (accuracy_score, precision_score,
recall_score,
    f1_score, roc_auc_score, roc_curve, confusion_matrix,
precision_recall_curve,
    average_precision_score)
warnings.filterwarnings("ignore")

plt.rcParams.update({"font.size": 11, "figure.dpi": 150,
                     "axes.grid": True, "grid.alpha": 0.3})
C_BLUE, C_ORANGE, C_GREEN, C_RED = "#1f4e79", "#e07b39", "#2e7d32",
"#b22222"

SEED = 42
np.random.seed(SEED); torch.manual_seed(SEED)
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
R = {}  # сюда складываем все числовые результаты

# ============================================================
# 1. ДАННЫЕ
# ============================================================
df = pd.read_csv("telco.csv")
R["n_rows"], R["n_cols_raw"] = int(df.shape[0]), int(df.shape[1])
raw = df.copy()
df = df.drop(columns=["customerID"])
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"],
```

```
errors="coerce")
R["n_empty_totalcharges"] = int(df["TotalCharges"].isna().sum())
df["TotalCharges"] = df["TotalCharges"].fillna(0.0)
y = (df["Churn"] == "Yes").astype(np.float32).values
R["churn_rate"] = float(y.mean())
df_feat = df.drop(columns=["Churn"])

numeric_cols = ["tenure", "MonthlyCharges", "TotalCharges",
"SeniorCitizen"]
categorical_cols = [c for c in df_feat.columns if c not in
numeric_cols]
df_enc = pd.get_dummies(df_feat, columns=categorical_cols,
drop_first=True)
X = df_enc.values.astype(np.float32)
feat_names = df_enc.columns.tolist()
R["n_features"] = int(X.shape[1])

X_tr, X_tmp, y_tr, y_tmp = train_test_split(X, y, test_size=0.30,
                                            random_state=SEED,
stratify=y)
X_va, X_te, y_va, y_te = train_test_split(X_tmp, y_tmp,
test_size=0.50,
                                          random_state=SEED,
stratify=y_tmp)
scaler = StandardScaler()
num_idx = [feat_names.index(c) for c in numeric_cols]
X_tr[:, num_idx] = scaler.fit_transform(X_tr[:, num_idx])
X_va[:, num_idx] = scaler.transform(X_va[:, num_idx])
X_te[:, num_idx] = scaler.transform(X_te[:, num_idx])
R["n_train"], R["n_val"], R["n_test"] = len(y_tr), len(y_va),
len(y_te)

# ============================================================
# 2. РАЗВЕДОЧНЫЙ АНАЛИЗ (EDA)
# ============================================================
# 2.1 Баланс классов

fig, ax = plt.subplots(figsize=(5.5, 5))
counts = pd.Series(y).map({0: "Остался", 1: "Ушёл"}).value_counts()
ax.pie(counts.values, labels=counts.index, autopct="%1.1f%%",
       colors=[C_BLUE, C_ORANGE], startangle=90,
       wedgeprops=dict(edgecolor="white", linewidth=2))
ax.set_title("Распределение клиентов по факту оттока")
```

```
plt.tight_layout(); plt.savefig("figs/eda_balance.png");
plt.close()

# 2.2 Отток по типу контракта
fig, ax = plt.subplots(figsize=(7, 4.5))
ct = raw.groupby("Contract")["Churn"].apply(lambda s: (s ==
"Yes").mean()).reindex(
    ["Month-to-month", "One year", "Two year"])
ct.index = ["Помесячный", "Годовой", "Двухлетний"]
bars = ax.bar(ct.index, ct.values * 100, color=[C_RED, C_ORANGE,
C_GREEN])
ax.set_ylabel("Доля ушедших клиентов, %")
ax.set_title("Отток в зависимости от типа контракта")
for b, v in zip(bars, ct.values):
    ax.text(b.get_x() + b.get_width()/2, v*100 + 1,
f"{v*100:.1f}%", ha="center")
ax.set_ylim(0, 50)
plt.tight_layout(); plt.savefig("figs/eda_contract.png");
plt.close()
R["churn_by_contract"] = {k: round(float(v), 3) for k, v in zip(
    ["month", "year", "two_year"], ct.values)}

# 2.3 Отток по сроку обслуживания (tenure)
fig, ax = plt.subplots(figsize=(7.5, 4.5))
bins = [0, 6, 12, 24, 36, 48, 60, 72]
labels = ["0-6", "7-12", "13-24", "25-36", "37-48", "49-60", "61-
72"]
raw["tenure_bin"] = pd.cut(raw["tenure"], bins=bins, labels=labels,
include_lowest=True)
tb = raw.groupby("tenure_bin")["Churn"].apply(lambda s: (s ==
"Yes").mean())
ax.plot(range(len(tb)), tb.values * 100, marker="o", color=C_BLUE,
linewidth=2)
ax.set_xticks(range(len(tb))); ax.set_xticklabels(tb.index)
ax.set_xlabel("Срок обслуживания, мес."); ax.set_ylabel("Доля
ушедших, %")
ax.set_title("Отток в зависимости от срока обслуживания")
plt.tight_layout(); plt.savefig("figs/eda_tenure.png"); plt.close()

# 2.4 Распределение ежемесячных платежей
fig, ax = plt.subplots(figsize=(7.5, 4.5))
ax.hist(raw[raw.Churn == "No"]["MonthlyCharges"], bins=30,
alpha=0.6,
        label="Остался", color=C_BLUE, density=True)
ax.hist(raw[raw.Churn == "Yes"]["MonthlyCharges"], bins=30,
alpha=0.6,
        label="Ушёл", color=C_ORANGE, density=True)
ax.set_xlabel("Ежемесячный платёж, $"); ax.set_ylabel("Плотность")
ax.set_title("Распределение ежемесячных платежей")
```

```
ax.legend()
plt.tight_layout(); plt.savefig("figs/eda_monthly.png");
plt.close()

# 2.5 Корреляционная матрица числовых признаков
fig, ax = plt.subplots(figsize=(6, 5))
corr_df = raw.copy()
corr_df["TotalCharges"] = pd.to_numeric(corr_df["TotalCharges"],
errors="coerce").fillna(0)
corr_df["ChurnNum"] = (corr_df["Churn"] == "Yes").astype(int)
cm_corr = corr_df[["tenure", "MonthlyCharges", "TotalCharges",
"SeniorCitizen", "ChurnNum"]].co
rr()
im = ax.imshow(cm_corr, cmap="coolwarm", vmin=-1, vmax=1)
lab = ["tenure", "Monthly", "Total", "Senior", "Churn"]
ax.set_xticks(range(5)); ax.set_xticklabels(lab, rotation=45,
ha="right")
ax.set_yticks(range(5)); ax.set_yticklabels(lab)
for i in range(5):
    for j in range(5):
        ax.text(j, i, f"{cm_corr.values[i,j]:.2f}", ha="center",
va="center",
                color="white" if abs(cm_corr.values[i,j]) > 0.5
else "black", fontsize=9)
ax.set_title("Корреляция числовых признаков")
fig.colorbar(im, fraction=0.046)
plt.tight_layout(); plt.savefig("figs/eda_corr.png"); plt.close()

print("EDA готов")

# ============================================================
# 3. МОДЕЛЬ И ОБУЧЕНИЕ
# ============================================================
class ChurnNet(nn.Module):
    def __init__(self, in_f, h1=64, h2=32, p=0.3):
        super().__init__()
        layers = [nn.Linear(in_f, h1), nn.BatchNorm1d(h1),
nn.ReLU(), nn.Dropout(p)]
        if h2 > 0:
            layers += [nn.Linear(h1, h2), nn.BatchNorm1d(h2),
nn.ReLU(), nn.Dropout(p)]
            last = h2
        else:
            last = h1
        layers += [nn.Linear(last, 1)]
        self.net = nn.Sequential(*layers)
    def forward(self, x): return self.net(x)

def loaders(bs=64):
```

```
    tr = DataLoader(TensorDataset(torch.from_numpy(X_tr),
torch.from_numpy(y_tr).unsqueeze(1)),
                    batch_size=bs, shuffle=True)
    va = DataLoader(TensorDataset(torch.from_numpy(X_va),
torch.from_numpy(y_va).unsqueeze(1)),
                    batch_size=256)
    return tr, va

def train_model(h1=64, h2=32, p=0.3, balanced=True, epochs=100,
patience=15,
                lr=1e-3, seed=SEED, track=True):
    torch.manual_seed(seed); np.random.seed(seed)
    model = ChurnNet(X.shape[1], h1, h2, p).to(device)
    if balanced:
        pw = torch.tensor([(y_tr == 0).sum() / (y_tr == 1).sum()],
                          dtype=torch.float32, device=device)
        crit = nn.BCEWithLogitsLoss(pos_weight=pw)
    else:
        crit = nn.BCEWithLogitsLoss()
    opt = torch.optim.Adam(model.parameters(), lr=lr,
weight_decay=1e-4)
    tr, va = loaders()
    hist = {"tl": [], "vl": [], "ta": [], "vaacc": []}
    best, best_state, noimp = 1e9, None, 0
    for ep in range(epochs):
        for mode, ld in [("tr", tr), ("va", va)]:
            model.train() if mode == "tr" else model.eval()
            tot, corr, n = 0., 0, 0
            with torch.set_grad_enabled(mode == "tr"):
                for xb, yb in ld:
                    xb, yb = xb.to(device), yb.to(device)
                    lo = model(xb); loss = crit(lo, yb)
                    if mode == "tr":
                        opt.zero_grad(); loss.backward();
opt.step()
                    tot += loss.item()*xb.size(0)
                    corr += ((torch.sigmoid(lo) >= .5).float() ==
yb).sum().item()
                    n += xb.size(0)
            if mode == "tr": tl, ta = tot/n, corr/n
            else: vl, vaacc = tot/n, corr/n
        if track:
            hist["tl"].append(tl); hist["vl"].append(vl)
            hist["ta"].append(ta); hist["vaacc"].append(vaacc)
        if vl < best:
            best, best_state, noimp = vl, {k: v.cpu().clone() for
k, v in model.state_dict().it
ems()}, 0
        else:
```

```
            noimp += 1
        if noimp >= patience: break
    model.load_state_dict(best_state)
    return model, hist, ep + 1

def evaluate(model, thr=0.5):
    model.eval()
    with torch.no_grad():
        p =
torch.sigmoid(model(torch.from_numpy(X_te).to(device))).cpu().numpy
().ravel()
    pred = (p >= thr).astype(int)
    return {"acc": accuracy_score(y_te, pred), "prec":
precision_score(y_te, pred),
            "rec": recall_score(y_te, pred), "f1": f1_score(y_te,
pred),
            "auc": roc_auc_score(y_te, p)}, p, pred

# 3.1 Основная модель
model, hist, n_ep = train_model()
R["n_epochs_trained"] = int(n_ep)
m_main, probs, preds = evaluate(model)
R["nn_main"] = {k: round(v, 4) for k, v in m_main.items()}
print("Основная NN:", R["nn_main"])

# Кривые потерь и точности
ep_r = range(1, len(hist["tl"]) + 1)
fig, ax = plt.subplots(figsize=(7, 4.8))
ax.plot(ep_r, hist["tl"], label="Обучение", color=C_BLUE)
ax.plot(ep_r, hist["vl"], label="Валидация", color=C_ORANGE)
ax.set_xlabel("Эпоха"); ax.set_ylabel("Функция потерь (BCE)")
ax.set_title("Динамика функции потерь"); ax.legend()
plt.tight_layout(); plt.savefig("figs/nn_loss.png"); plt.close()

fig, ax = plt.subplots(figsize=(7, 4.8))
ax.plot(ep_r, hist["ta"], label="Обучение", color=C_BLUE)
ax.plot(ep_r, hist["vaacc"], label="Валидация", color=C_ORANGE)
ax.set_xlabel("Эпоха"); ax.set_ylabel("Accuracy")
ax.set_title("Динамика точности классификации"); ax.legend()
plt.tight_layout(); plt.savefig("figs/nn_acc.png"); plt.close()

# ROC
fpr, tpr, _ = roc_curve(y_te, probs)
fig, ax = plt.subplots(figsize=(6, 5.5))
ax.plot(fpr, tpr, color=C_BLUE, lw=2, label=f"Нейросеть (AUC =
{m_main['auc']:.3f})")
ax.plot([0, 1], [0, 1], "--", color="gray", label="Случайный")
ax.set_xlabel("FPR (доля ложноположительных)"); ax.set_ylabel("TPR
(доля истинноположительных)"
```

```
)
ax.set_title("ROC-кривая"); ax.legend(loc="lower right")
plt.tight_layout(); plt.savefig("figs/nn_roc.png"); plt.close()

# Confusion
cm = confusion_matrix(y_te, preds)
fig, ax = plt.subplots(figsize=(5.5, 4.8))
im = ax.imshow(cm, cmap="Blues")
ax.set_xticks([0, 1]); ax.set_xticklabels(["Остался", "Ушёл"])
ax.set_yticks([0, 1]); ax.set_yticklabels(["Остался", "Ушёл"])
ax.set_xlabel("Прогноз"); ax.set_ylabel("Истина");
ax.set_title("Матрица ошибок")
th = cm.max()/2
for i in range(2):
    for j in range(2):
        ax.text(j, i, str(cm[i, j]), ha="center", va="center",
fontsize=15,
                color="white" if cm[i, j] > th else "black")
fig.colorbar(im, fraction=0.046)
plt.tight_layout(); plt.savefig("figs/nn_confusion.png");
plt.close()
R["nn_confusion"] = cm.tolist()

# PR-кривая
pr_p, pr_r2, _ = precision_recall_curve(y_te, probs)
ap = average_precision_score(y_te, probs)
fig, ax = plt.subplots(figsize=(6, 5.5))
ax.plot(pr_r2, pr_p, color=C_GREEN, lw=2, label=f"AP = {ap:.3f}")
ax.axhline(y_te.mean(), ls="--", color="gray", label=f"Базовый
({y_te.mean():.2f})")
ax.set_xlabel("Полнота (Recall)"); ax.set_ylabel("Точность
(Precision)")
ax.set_title("Precision-Recall кривая"); ax.legend()
plt.tight_layout(); plt.savefig("figs/nn_pr.png"); plt.close()
R["nn_ap"] = round(float(ap), 4)

# ============================================================
# 4. БАЗОВЫЕ МОДЕЛИ
# ============================================================
logreg = LogisticRegression(max_iter=1000, class_weight="balanced")
logreg.fit(X_tr, y_tr)
lp = logreg.predict_proba(X_te)[:, 1]; lpred = (lp >=
.5).astype(int)
R["logreg"] = {"acc": round(accuracy_score(y_te, lpred), 4),
    "prec": round(precision_score(y_te, lpred), 4),
    "rec": round(recall_score(y_te, lpred), 4),
    "f1": round(f1_score(y_te, lpred), 4),
    "auc": round(roc_auc_score(y_te, lp), 4)}
```

```
rf = RandomForestClassifier(n_estimators=200,
class_weight="balanced",
                            random_state=SEED, n_jobs=-1)
rf.fit(X_tr, y_tr)
rp = rf.predict_proba(X_te)[:, 1]; rpred = (rp >= .5).astype(int)
R["rf"] = {"acc": round(accuracy_score(y_te, rpred), 4),
    "prec": round(precision_score(y_te, rpred), 4),
    "rec": round(recall_score(y_te, rpred), 4),
    "f1": round(f1_score(y_te, rpred), 4),
    "auc": round(roc_auc_score(y_te, rp), 4)}
print("LogReg:", R["logreg"]); print("RF:", R["rf"])

# Сравнительная ROC
fig, ax = plt.subplots(figsize=(6.5, 5.5))
for prob, name, col in [(probs, "Нейросеть", C_BLUE), (rp,
"Случайный лес", C_GREEN),
                        (lp, "Логистическая регрессия", C_ORANGE)]:
    f, t, _ = roc_curve(y_te, prob)
    ax.plot(f, t, lw=2, label=f"{name} (AUC={roc_auc_score(y_te,
prob):.3f})", color=col)
ax.plot([0, 1], [0, 1], "--", color="gray")
ax.set_xlabel("FPR"); ax.set_ylabel("TPR"); ax.set_title("Сравнение
ROC-кривых моделей")
ax.legend(loc="lower right")
plt.tight_layout(); plt.savefig("figs/compare_roc.png");
plt.close()

# Сравнительная столбчатая по F1/AUC
fig, ax = plt.subplots(figsize=(7.5, 4.8))
models = ["Логистическая\nрегрессия", "Случайный\nлес",
"Нейронная\nсеть"]
f1s = [R["logreg"]["f1"], R["rf"]["f1"], R["nn_main"]["f1"]]
aucs = [R["logreg"]["auc"], R["rf"]["auc"], R["nn_main"]["auc"]]
x = np.arange(len(models)); w = 0.35
ax.bar(x - w/2, f1s, w, label="F1-мера", color=C_ORANGE)
ax.bar(x + w/2, aucs, w, label="ROC-AUC", color=C_BLUE)
ax.set_xticks(x); ax.set_xticklabels(models); ax.set_ylim(0, 1)
ax.set_title("Сравнение качества моделей"); ax.legend()
for i, (a, b) in enumerate(zip(f1s, aucs)):
    ax.text(i - w/2, a + .01, f"{a:.3f}", ha="center", fontsize=9)
    ax.text(i + w/2, b + .01, f"{b:.3f}", ha="center", fontsize=9)
plt.tight_layout(); plt.savefig("figs/compare_bars.png");
plt.close()

# ============================================================
# 5. ИССЛЕДОВАНИЕ АРХИТЕКТУРЫ
# ============================================================
archs = [("32", 32, 0), ("64", 64, 0), ("64-32", 64, 32),
         ("128-64", 128, 64), ("128-64-32*", 128, 64)]  # последний
```

```
помечен отдельно ниже
arch_res = []
for name, h1, h2 in [("32", 32, 0), ("64", 64, 0), ("64-32", 64,
32),
                     ("128-64", 128, 64), ("256-128", 256, 128)]:
    mdl, _, ep = train_model(h1=h1, h2=h2, track=False)
    met, _, _ = evaluate(mdl)
    nparam = sum(p.numel() for p in mdl.parameters())
    arch_res.append({"name": name, "params": int(nparam),
                     "f1": round(met["f1"], 4), "auc":
round(met["auc"], 4),
                     "acc": round(met["acc"], 4)})
R["arch"] = arch_res
print("Архитектуры:", arch_res)

fig, ax = plt.subplots(figsize=(7.5, 4.8))
names = [a["name"] for a in arch_res]
ax.plot(names, [a["auc"] for a in arch_res], marker="o",
label="ROC-AUC", color=C_BLUE, lw=2)
ax.plot(names, [a["f1"] for a in arch_res], marker="s", label="F1-
мера", color=C_ORANGE, lw=2)
ax.set_xlabel("Архитектура (нейроны в скрытых слоях)");
ax.set_ylabel("Метрика")
ax.set_title("Влияние архитектуры на качество"); ax.legend()
plt.tight_layout(); plt.savefig("figs/arch.png"); plt.close()

# ============================================================
# 6. ВЛИЯНИЕ БАЛАНСИРОВКИ КЛАССОВ
# ============================================================
mdl_unbal, _, _ = train_model(balanced=False, track=False)
met_unbal, _, pred_unbal = evaluate(mdl_unbal)
R["nn_unbalanced"] = {k: round(v, 4) for k, v in met_unbal.items()}
print("Без балансировки:", R["nn_unbalanced"])

fig, ax = plt.subplots(figsize=(7.5, 4.8))
metrics_names = ["Accuracy", "Precision", "Recall", "F1", "AUC"]
bal_vals = [m_main["acc"], m_main["prec"], m_main["rec"],
m_main["f1"], m_main["auc"]]
unbal_vals = [met_unbal["acc"], met_unbal["prec"],
met_unbal["rec"], met_unbal["f1"], met_unbal
["auc"]]
x = np.arange(len(metrics_names)); w = 0.35
ax.bar(x - w/2, unbal_vals, w, label="Без балансировки",
color="#888")
ax.bar(x + w/2, bal_vals, w, label="С балансировкой (pos_weight)",
color=C_BLUE)
ax.set_xticks(x); ax.set_xticklabels(metrics_names); ax.set_ylim(0,
1)
ax.set_title("Влияние балансировки классов на метрики");
```

```
ax.legend()
plt.tight_layout(); plt.savefig("figs/balance_effect.png");
plt.close()

# ============================================================
# 7. ПОДБОР ПОРОГА КЛАССИФИКАЦИИ
# ============================================================
ths = np.linspace(0.1, 0.9, 81)
f1_by_th = [f1_score(y_te, (probs >= t).astype(int)) for t in ths]
prec_by_th = [precision_score(y_te, (probs >= t).astype(int),
zero_division=0) for t in ths]
rec_by_th = [recall_score(y_te, (probs >= t).astype(int)) for t in
ths]
best_th = float(ths[int(np.argmax(f1_by_th))])
R["best_threshold"] = round(best_th, 3)
R["best_threshold_f1"] = round(float(max(f1_by_th)), 4)
met_bt, _, _ = evaluate(model, thr=best_th)
R["nn_best_threshold"] = {k: round(v, 4) for k, v in
met_bt.items()}

fig, ax = plt.subplots(figsize=(7.5, 4.8))
ax.plot(ths, f1_by_th, label="F1-мера", color=C_BLUE, lw=2)
ax.plot(ths, prec_by_th, label="Точность", color=C_ORANGE, lw=1.5,
ls="--")
ax.plot(ths, rec_by_th, label="Полнота", color=C_GREEN, lw=1.5,
ls="--")
ax.axvline(best_th, color=C_RED, ls=":", label=f"Оптимум =
{best_th:.2f}")
ax.set_xlabel("Порог классификации"); ax.set_ylabel("Метрика")
ax.set_title("Зависимость метрик от порога классификации");
ax.legend()
plt.tight_layout(); plt.savefig("figs/threshold.png"); plt.close()
print("Лучший порог:", best_th, "->", R["nn_best_threshold"])

# ============================================================
# 8. ВАЖНОСТЬ ПРИЗНАКОВ (permutation importance на нейросети)
# ============================================================
def auc_of(Xmat):
    model.eval()
    with torch.no_grad():
        p =
torch.sigmoid(model(torch.from_numpy(Xmat).to(device))).cpu().numpy
().ravel()
    return roc_auc_score(y_te, p)

base_auc = auc_of(X_te)
imps = []
rng = np.random.default_rng(SEED)
for i in range(X_te.shape[1]):
```

```
    drops = []
    for _ in range(5):
        Xp = X_te.copy()
        rng.shuffle(Xp[:, i])
        drops.append(base_auc - auc_of(Xp))
    imps.append(np.mean(drops))
imp_idx = np.argsort(imps)[::-1][:12]
fig, ax = plt.subplots(figsize=(7.5, 5.5))
yp = np.arange(len(imp_idx))[::-1]
ax.barh(yp, [imps[i] for i in imp_idx], color=C_BLUE)
ax.set_yticks(yp); ax.set_yticklabels([feat_names[i] for i in
imp_idx], fontsize=9)
ax.set_xlabel("Падение ROC-AUC при перемешивании признака")
ax.set_title("Важность признаков (топ-12)")
plt.tight_layout(); plt.savefig("figs/importance.png"); plt.close()
R["top_features"] = [(feat_names[i], round(float(imps[i]), 4)) for
i in imp_idx[:8]]
print("Топ признаки:", R["top_features"][:5])

# ============================================================
json.dump(R, open("results.json", "w", encoding="utf-8"),
ensure_ascii=False, indent=2)
print("\nВсе эксперименты завершены. results.json сохранён.")
```

Приложение В

(обязательное)

Система прогнозирования оттока клиентов

Презентация

Система прогнозирования оттока клиентов

Автор работы:

студент группы 22ВВП1 Митрошин Юрий Евгеньевич

Научный руководитель:

к.т.н., доцент кафедры ВТ Леонова Татьяна Юрьевна

Пенза 2026

АКТУАЛЬНОСТЬ

Почему прогнозирование оттока важно

Подписная (subscription) модель — одна из доминирующих в цифровой экономике: стриминг, SaaS, телеком, облачные сервисы.

Выручка формируется регулярными платежами, поэтому устойчивость бизнеса напрямую зависит от того, как долго клиент остаётся подписчиком.

Отток (churn) — отказ от продления подписки. Своевременное выявление клиентов «группы риска» позволяет перейти от реакции к проактивному удержанию.

ЦЕЛЬ И ЗАДАЧИ

Что требовалось сделать

Цель. Разработать и экспериментально исследовать систему прогнозирования оттока клиентов подписного сервиса на основе искусственной нейронной сети, реализованной средствами PyTorch.

1. Анализ предметной области: понятие оттока и его влияние на показатели подписного бизнеса
2. Обзор существующих методов и решений прогнозирования оттока
3. Обоснование выбора инструментов: язык, библиотека ML, среда разработки
4. Выбор и подготовка набора данных, разведочный анализ и предобработка
5. Проектирование архитектуры сети и реализация обучения с учётом дисбаланса классов
6. Оценка качества по совокупности метрик и сравнение с базовыми моделями
7. Исследование влияния архитектуры, балансировки и порога; анализ важности признаков

ДАННЫЕ · EDA

Набор данных и разведочный анализ

Telco Customer Churn
открытый набор, телеком-сервис

7 043 клиента
30 признаков после кодирования

70 / 15 / 15 %
train / valid / test, стратификация

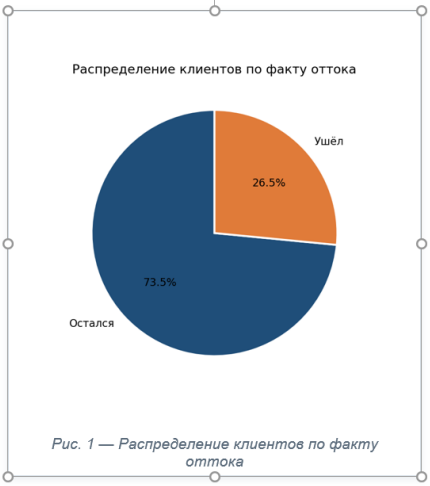


Рис. 1 — Распределение клиентов по факту оттока

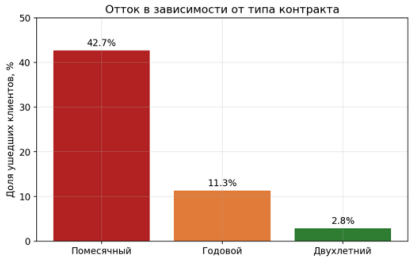


Рис. 2 — Доля оттока по типу контракта

Ключевые предикторы

- Классы несбалансированы: ~27 % оттока
- Короткий срок обслуживания (tenure) → выше отток
- Помесячный контракт резко повышает риск ухода
- Электронные чеки и оптоволокно связаны с оттоком

АРХИТЕКТУРА

Многослойный перцептрон (MLP)

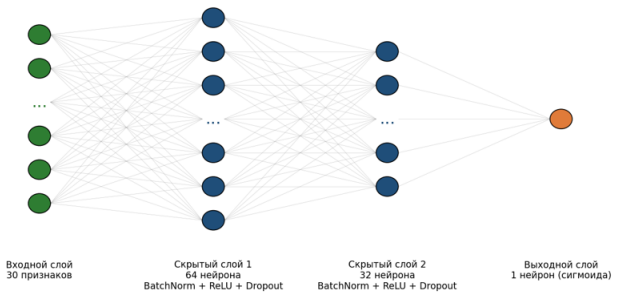


Рис. 3 — Структура сети: 30 → 64 → 32 → 1

Полносвязная сеть

30 признаков → 64 → 32 → 1 логит; сужение слоёв сжимает информацию к решению

ReLU + BatchNorm

нелинейность и ускорение сходимости, устойчивость к выбору весов

Dropout 0.3

случайное отключение нейронов против переобучения

Учёт дисбаланса

BCEWithLogitsLoss с весом pos_weight для редкого класса

ПРЕДОБРАБОТКА

Подготовка данных к обучению

1 Очистка

Удаление customerID; исправление поля TotalCharges (пробелы → 0 у новых клиентов)

2 Кодирование цели

Churn → бинарная метка: «Yes» = 1 (отток), «No» = 0 (лояльность)

3 One-hot кодирование

Категориальные признаки → бинарные столбцы, drop_first; итого 30 признаков

4 Разбиение

Стратифицированно на train (4930) / valid (1056) / test (1057)

5 Стандартизация

Числовые признаки к нулевому среднему; параметры считаются только по train

6 Без утечки данных

Порядок исключает data leakage — объективная итоговая оценка

ОБУЧЕНИЕ

Алгоритм обучения и баланс классов

Настройки обучения

- **Оптимизатор**
  Adam, lr = 0.001, weight decay = 1e-4
- **Функция потерь**
  BCEWithLogitsLoss + pos_weight
- **Пакеты**
  batch = 64 объекта
- **Ранняя остановка**
  patience = 15; обучение остановилось на 22-й эпохе
- **Регуляризация**
  BatchNorm, Dropout 0.3, weight decay

Зачем балансировка классов

Без взвешивания модель «занижает» прогноз для редкого класса — пропускает уходящих клиентов.

Recall без баланса	→	Recall с балансом
0.502		0.744

F1-мера выросла с 0.578 до 0.636; ROC-AUC при этом не меняется (метрика инвариантна к порогу).

РЕЗУЛЬТАТЫ

Качество на отложенной тестовой выборке

ROC-AUC	Recall	F1-мера	Accuracy
0.845	0.744	0.636	0.774

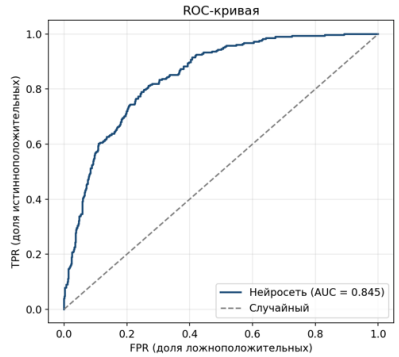


Рис. 4 — ROC-кривая нейронной сети

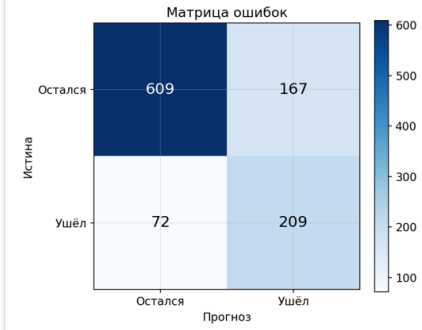


Рис. 5 — Матрица ошибок

Что это значит

- Модель выявляет ~74 % всех уходящих клиентов
- Верно: 209 из 281 ушедших, 609 из 776 лояльных
- Ошибки в основном ложноположительные — менее критичны для удержания
- ROC-AUC на уровне лучших решений для этого набора

АНАЛИЗ

Сравнение моделей и факторы оттока

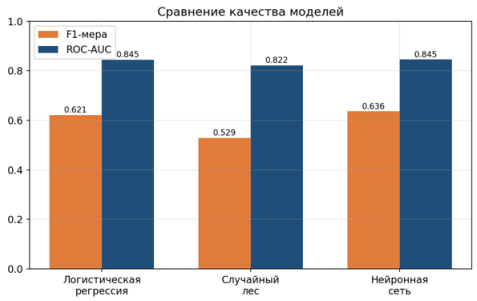


Рис. 6 — F1-мера и ROC-AUC по моделям

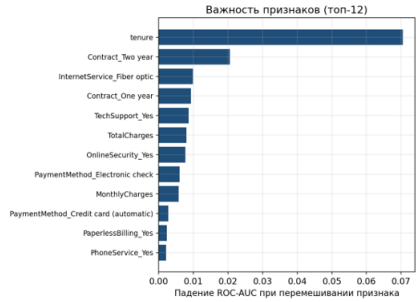


Рис. 7 — Важность признаков (permutation)

Сравнение с базовыми моделями

Нейросеть и логистическая регрессия дают ROC-AUC 0.845; случайный лес точнее (0.638), но пропускает половину уходящих (recall 0.452). Нейросеть — лучший баланс качества.

Главные факторы оттока

Сильнее всего на прогноз влияет срок обслуживания (tenure), затем тип контракта, оптоволоконное подключение и наличие техподдержки. Выводы совпадают с EDA — модель интерпретируема.

Цель достигнута, все задачи решены

Рабочая модель

Нейросеть даёт ROC-AUC 0.845 и полноту 0.744 — уровень лучших решений для набора Telco Churn

Обоснованные решения

Архитектура 64-32 оптимальна; балансировка классов необходима; нейросеть превосходит baseline по балансу

Интерпретируемость

Анализ важности признаков выявил ключевые факторы оттока, совпадающие с разведочным анализом

Эффективность

4 289 параметров: обучение за секунды на CPU, пригодна для регулярного дообучения

0.845

ROC-AUC
на тестовой выборке

- учёт временной динамики поведения
- ансамбли и более сложные архитектуры
- интеграция в CRM через API

Спасибо за внимание!