



Geekbrains

**Прогнозирование оттока клиентов в телекоммуникационной
компании TELCO с применением алгоритмов машинного
обучения и составление рекомендаций по их удержанию**

Программа: Разработчик-аналитик
Специализация Data Engineer
Абдулаев Заур Исаевич

Москва
2024



СОДЕРЖАНИЕ

Введение.....	3
Глава 1. Основы машинного обучения: виды и их алгоритмы.....	5
1.1 Что такое машинное обучение.....	5
1.2 Основные типы и виды машинного обучения.....	6
1.3 Основные алгоритмы машинного обучения.....	7
1.4 Инструменты в машинном обучении.....	13
Глава 2. Проведение анализа данных компании TELCO в Jupiter notebook.....	16
2.1 Краткая характеристика компании.....	16
2.2 Создание проекта и активация виртуального окружения.....	16
2.3 Исследовательский анализ данных в Jupiter notebook. Часть 1.....	17
2.4 Исследовательский анализ данных в Jupiter notebook. Часть 2.....	31
Глава 3. Применение моделей машинного обучения для прогнозирования оттока клиентов.....	37
3.1 Выбор модели машинного обучения.....	37
3.2 Модель Decision Tree.....	39
3.3 Модель Random Forest.....	43
3.4 Модель Boosted Trees.....	45
3.5 Модель логистической регрессии.....	50
3.6 Метод опорных векторов (SVM).....	53
3.7 Метод К-ближайших соседей (KNN).....	54
3.8 Общий вывод.....	56
3.9 Общие рекомендации по снижению оттока клиентов в компании TELCO....	56
Заключение.....	59
Список используемой литературы.....	61
Приложения.....	62

Введение

В настоящее время конкуренция на рынке телекоммуникационных услуг достигла своего пика, и удержание клиентов стало одним из ключевых элементов успешной стратегии компании. Одной из основных проблем, с которой сталкиваются телекоммуникационные компании, является отток клиентов. Потеря клиентов может оказать негативное влияние на финансовые показатели компании, а также на её репутацию.

Целью данного проекта является прогнозирование оттока клиентов в телекоммуникационной компании TELCO с применением алгоритмов машинного обучения и разработка рекомендаций по их удержанию. Данный проект будет решать проблему определения клиентов, которые могут уйти от компании, и выявлять ключевые факторы, влияющие на решение клиента остаться или уйти.

Для достижения поставленной цели будут использоваться алгоритмы машинного обучения, такие как деревья принятия решений, случайный лес и градиентный бустинг, модель логистической регрессии, метод опорных векторов (SVM), метод К-ближайших соседей (KNN). Эти алгоритмы позволят проанализировать исторические данные о клиентах компании, идентифицировать паттерны и предсказать вероятность оттока для каждого клиента.

При проведении исследования использовались такие инструменты как Python, Jupiter notebook и различные библиотеки для анализа данных. В проекте использовалось виртуальное окружение conda, с помощью которого можно устанавливать различные версии пакетов и библиотек, не влияя на основное окружение системы. Это позволяет избежать конфликтов между зависимостями и обеспечивает более надежное и удобное управление окружением для разработки проектов на Python.

В качестве исторических данных были взяты данные компании TELCO с сайта www.kaggle.ru, где можно найти много исторических данных различных компаний.

Применение машинного обучения в данном проекте позволит компании оптимизировать свои маркетинговые и удерживающие стратегии, направив усилия на клиентов с наибольшей вероятностью оттока. Благодаря этому компания сможет сохранить большее количество клиентов и увеличить свою доходность.

Актуальность данной темы обусловлена необходимостью телекоммуникационным компаниям улучшить качество обслуживания клиентов, минимизировать отток и увеличить свою конкурентоспособность на рынке. Применение алгоритмов машинного обучения в данном проекте позволит компании совершить качественный прорыв в сфере удержания клиентов и повысить свою эффективность.

Глава 1. Основы машинного обучения: виды и их алгоритмы

1.1 Что такое машинное обучение

Машинное обучение - это раздел искусственного интеллекта, который изучает методы, алгоритмы и модели, позволяющие компьютеру обучаться на основе данных, не явно программируясь на определенные инструкции. В отличие от классического программирования, где набор инструкций задается заранее, в машинном обучении компьютер самостоятельно определяет закономерности и шаблоны в данных, чтобы делать прогнозы или принимать решения.

Используя технологию машинного обучения, программисты больше не обязаны тратить время на написание инструкций, рассматривающих все возможные сценарии и содержащих все решения. Вместо этого они могут встроить в компьютер или программу алгоритм, который самостоятельно находит решения, используя статистические данные для выявления закономерностей и предсказаний[3].

Технология машинного обучения на основе анализа данных впервые появилась в 1950-х годах при разработке программ для игры в шашки. За прошедшие десятилетия этот общий принцип остался неизменным, но благодаря резкому увеличению вычислительной мощности компьютеров, стало возможным создание более сложных закономерностей и прогнозов, а также решение более широкого круга задач с применением машинного обучения.

Для начала процесса машинного обучения необходимо загрузить в компьютер набор данных, на которых алгоритм будет учиться. Например, это могут быть изображения собак и кошек с уже расставленными метками, указывающими на их вид. После обучения программа сможет самостоятельно распознавать собак и кошек на новых фотографиях без меток. Чем больше данных обработала программа, тем точнее будет ее распознавание[4].

Сегодня благодаря машинному обучению компьютеры могут распознавать и фотографии, и изображения, и тексты. Например, программы уже

умеют распознавать не только лица на фотографиях, но и пейзажи, предметы, текст и цифры. Для текстов также необходимо машинное обучение: проверка грамматики в текстовых редакторах и на мобильных устройствах уже стала стандартом. Также существуют программы, способные автоматически создавать новостные статьи на различные темы без участия человека.

1.2 Основные типы и виды машинного обучения

Все задачи, решаемые с помощью ML, относятся к одной из следующих категорий:

1) *Задача регрессии* заключается в предсказании числового значения на основе данных с различными признаками. Например, прогнозирование цены акций через определенный период времени или ожидаемый объем продаж товара на следующий месяц.

2) *Задача классификации* состоит в присвоении объекту определенной категории на основе набора признаков. Например, определение наличия определенного объекта на изображении или диагностика болезни по медицинским показателям.

3) *Задача кластеризации* предполагает разделение данных на группы по их сходству без предварительного определения категорий. Например, классификация покупателей по их покупательским привычкам или разделение текстов по тематике.

4) *Задача уменьшения размерности* заключается в сокращении количества признаков для упрощения анализа данных и улучшения визуализации. Например, сокращение размерности для отображения данных на двумерном графике.

5) *Задача выявления аномалий* заключается в обнаружении нестандартных и редких случаев в данных. Например, выявление мошеннических операций с кредитными картами или выявление необычных поведенческих шаблонов в данных.

Основные виды машинного обучения

Машинное обучение подразделяется на два основных типа: обучение с учителем и обучение без учителя. В обоих случаях машинам предоставляются исходные данные для анализа и выявления закономерностей, но различие заключается в участии человека в обучении.

1) Обучение с учителем

При обучении с учителем машине предоставляются данные вместе с ответами, что позволяет ей проверять свои гипотезы. Задача заключается в том, чтобы создать модель, которая может предсказывать целевую переменную (например, цену квартиры) на основе входных данных (площадь, расположение и т. д.). Примеры задач обучения с учителем включают регрессию (предсказание непрерывных значений, таких как цена) и классификацию (разделение объектов на категории, такие как спам или не спам).

2) Обучение без учителя

В обучении без учителя машине не предоставляются ответы. Она должна самостоятельно обнаруживать шаблоны и структуры в данных. Задачи обучения без учителя включают:

- Кластеризация: Разделение данных на группы на основе сходства, например, в случае распределения людей по размерным группам для пошива рубашек.

- Уменьшение размерности: Уменьшение количества признаков в данных для облегчения их отображения и анализа. Например, отображение данных с сотнями признаков в двух- или трехмерном пространстве.

1.3 Основные алгоритмы машинного обучения

Алгоритмы моделей машинного обучения - это математические процедуры, которые используются для обучения моделей машинного обучения на данных. Они определяют, как модель будет учиться и делать прогнозы.

Существует множество различных алгоритмов машинного обучения, каждый со своими сильными и слабыми сторонами.

Вот некоторые общие категории алгоритмов машинного обучения.

1) Дерево принятия решений для бизнеса

Дерево принятия решений - это метод, который помогает принимать решения с учетом потенциальных последствий, эффективности и затрат. В контексте бизнес-процессов дерево принятия решений создается путем последовательности вопросов "да/нет", которые ведут к правильному выбору.

Этот метод структурирует и систематизирует проблему, обеспечивая логическую основу для принятия решений. Дерево принятия решений особенно полезно, когда необходимо учитывать несколько факторов и оценивать вероятность различных событий[7].

Вот как работает дерево принятия решений для бизнеса:

- Начните с определения проблемы или решения, которое нужно принять.
- Создайте корневой узел, представляющий исходную проблему.
- Добавьте ветви для каждого возможного решения или действия.
- Для каждой ветви добавьте узлы, представляющие возможные последствия, эффективность и затраты.
- Повторяйте этот процесс, создавая дочерние узлы, пока не достигнете конечных узлов, представляющих конкретные решения.
- Оцените вероятность и влияние каждого возможного пути.
- Выберите путь, который приводит к наилучшему результату с учетом всех факторов.

Дерево принятия решений помогает предприятиям принимать обоснованные решения, учитывая все релевантные факторы и возможные последствия. Оно обеспечивает прозрачный и систематический подход к принятию решений, что особенно ценно в сложных и неопределенных ситуациях.

2) Наивная байесовская классификация

Наивная байесовская классификация - это метод машинного обучения, используемый для классификации данных. Он основан на теореме Байеса, которая позволяет рассчитывать вероятность события на основе имеющейся информации.

Наивность в названии метода заключается в предположении, что признаки независимы друг от друга, даже если на практике это может быть и не так. Несмотря на это упрощение, наивные байесовские классификаторы часто показывают хорошие результаты в реальных задачах.

Вот как работает наивная байесовская классификация:

- Рассчитать вероятность того, что объект принадлежит каждому классу на основе имеющихся данных.
- Рассчитать вероятность того, что объект имеет конкретный набор признаков для каждого класса.
- Умножить вероятности из шагов 1 и 2 для каждого класса.
- Классифицировать объект в класс с наибольшим произведением вероятностей.

Наивные байесовские классификаторы широко используются в следующих областях:

- Фильтрация спама
- Классификация новостных статей
- Анализ настроений
- Распознавание лиц и образов

Они популярны благодаря своей простоте, эффективности и возможности работать с большим количеством признаков и данных.

3) Метод наименьших квадратов

Метод наименьших квадратов - это статистический метод, используемый для подгонки прямой к набору данных. Цель метода - найти прямую, которая наиболее точно соответствует точкам данных, минимизируя сумму квадратов расстояний между точками и прямой.

Как работает метод наименьших квадратов:

- Предварительная обработка данных: данные преобразуются в числовой формат, и для каждого объекта создается набор признаков.

- Определение целевой переменной: выбирается целевая переменная, которую необходимо предсказать.

- Подгонка прямой: алгоритм метода наименьших квадратов используется для подгонки прямой к данным. Алгоритм находит значения параметров прямой, которые минимизируют сумму квадратов расстояний между точками данных и прямой.

- Прогнозирование: после подгонки прямой ее можно использовать для прогнозирования целевой переменной для новых объектов.

Метод наименьших квадратов широко используется в машинном обучении для решения задач регрессии. Регрессия - это задача прогнозирования непрерывных целевых переменных (например, цены на акции или продаж).

Преимущества:

- Простой и эффективный алгоритм
- Хорошо подходит для задач регрессии с линейными зависимостями
- Может работать с большим количеством признаков и данных

Недостатки:

- Предположение о линейной зависимости может снизить точность в некоторых случаях

- Может быть чувствителен к шуму и выбросам в данных

Области применения:

- Прогнозирование спроса
- Анализ временных рядов
- Оценка рисков
- Финансовое моделирование

4) Логистическая регрессия

Логистическая регрессия - это метод анализа данных, который используется для предсказания вероятности возникновения определенных событий на основе одной или нескольких независимых переменных. Она

применяется в различных областях, таких как кредитный скоринг, анализ рекламных кампаний, прогнозирование прибыли от продаж товаров и предсказание естественных катастроф. Логистическая регрессия позволяет оценить влияние различных факторов на вероятность наступления события и принять обоснованные решения на его основе.

5) Метод опорных векторов

Метод опорных векторов (SVM) представляет собой набор алгоритмов, которые широко используются для решения задач классификации и регрессионного анализа. SVM строит гиперплоскость в N-мерном пространстве, чтобы разделить объекты на два класса. Гиперплоскость создается таким образом, чтобы максимально удалиться от ближайшей точки каждого класса.

Этот метод помогает решать разнообразные задачи машинного обучения, такие как сплайсинг ДНК, определение пола человека по фотографии, и показ рекламы на сайтах[10].

6) Метод ансамблей

Метод ансамблей в машинном обучении - это подход, при котором используется не один классификатор, а множество классификаторов, которые работают вместе для принятия решения. Этот метод снижает влияние случайных ошибок и уменьшает дисперсию результатов, так как объединение нескольких моделей дает более точные прогнозы, чем одна отдельно взятая.

Подход ансамбль учитывает различные гипотезы и позволяет расширить множество базовых гипотез для более точного прогнозирования.

7) Кластеризация

Кластеризация - это процесс деления группы объектов на подмножества, называемые кластерами, таким образом, чтобы объекты в каждом кластере были более похожи друг на друга, чем на объекты из других кластеров.

Существует несколько алгоритмов для кластеризации объектов. Некоторые из них включают:

- Алгоритмы на основе центра тяжести, которые определяют кластеры на основе расстояния между объектами и их центром тяжести.

- Алгоритмы подключения, которые определяют кластеры на основе связей между объектами.
- Алгоритмы сокращения размерности, которые уменьшают размерность данных для упрощения кластеризации.
- Алгоритмы плотности, которые опираются на плотность объектов в пространстве для определения кластеров.
- Вероятностные алгоритмы, которые используют вероятностные модели для кластеризации.
- Методы машинного обучения, включая нейронные сети, для кластеризации данных.

Алгоритмы кластеризации применяются в различных областях, таких как биология (для анализа геномов и генетических данных), социология (для анализа социологических данных) и информационные технологии (для группировки данных и поиска закономерностей)[1].

8) *Метод главных компонент*

Метод главных компонент (РСА) – это статистический метод ортогонального преобразования, который используется для преобразования наблюдений над переменными, имеющими какие-то взаимосвязи, в новый набор главных компонент – линейно некоррелированных значений.

РСА обычно применяется для визуализации данных и сжатия информации, что позволяет упростить и уменьшить размерность данных для дальнейшего анализа. Однако РСА не подходит для данных, в которых все компоненты имеют высокую дисперсию и слабо упорядочены. Поэтому его применимость зависит от того, насколько хорошо изучена предметная область и какие данные доступны для анализа[9].

9) *Сингулярное разложение*

Сингулярное разложение, или SVD, в линейной алгебре представляет собой разложение прямоугольной матрицы на произведение трех матриц: U , Σ и V , где U и V являются унитарными матрицами, а Σ - диагональной матрицей.

Метод главных компонент является частным случаем сингулярного разложения. В прошлом алгоритмы компьютерного зрения основывались на этом методе, представляя лица или другие объекты как сумму базисных компонент, уменьшая их размерность и сравнивая с образцами из выборки. Современные алгоритмы сингулярного разложения в машинном обучении намного сложнее своих предшественников, но их основной принцип остается неизменным.

10) Анализ независимых компонент

Анализ независимых компонент, или ICA - это метод, который позволяет выявить скрытые факторы, влияющие на случайные величины или сигналы. Он строит модель, объясняющую данные с использованием независимых компонентов, которые считаются негауссовскими сигналами. В отличие от анализа главных компонент, ICA эффективнее в ситуациях, когда стандартные методы неэффективны. Этот метод широко используется в различных областях, таких как астрономия, медицина, распознавание речи, финансовый анализ и другие. Он помогает обнаружить скрытые причины явлений и является мощным инструментом для исследований и анализа данных[2].

1.4 Инструменты в машинном обучении

Инструменты машинного обучения - это программные платформы или библиотеки, которые предоставляют набор функций и алгоритмов для разработки, обучения и развертывания моделей машинного обучения. Эти инструменты позволяют разработчикам и специалистам по данным автоматизировать процессы машинного обучения и создавать более точные и эффективные модели.

Вот некоторые из наиболее популярных инструментов машинного обучения:

- TensorFlow: открытая платформа с открытым исходным кодом, разработанная Google. TensorFlow предоставляет широкий спектр инструментов

и библиотек для создания, обучения и развертывания моделей глубокого обучения и других типов моделей машинного обучения[5].

- `scikit-learn`: библиотека Python с открытым исходным кодом, предоставляющая широкий спектр алгоритмов машинного обучения для классификации, регрессии, кластеризации и других задач.

- `PyTorch`: библиотека Python с открытым исходным кодом, разработанная Facebook. `PyTorch` используется в основном для глубокого обучения и предоставляет гибкую и эффективную среду для создания и обучения моделей.

- `Keras`: высокоуровневый API для `TensorFlow`, разработанный для упрощения создания и обучения моделей глубокого обучения. `Keras` предоставляет простой и удобный интерфейс для создания и запуска моделей с использованием `TensorFlow`.

- `Jupyter Notebook`: веб-приложение с открытым исходным кодом, которое позволяет пользователям создавать и делиться документами, содержащими живой код, уравнения, визуализации и текстовые пояснения. `Jupyter Notebook` широко используется для исследований в области науки о данных и разработки моделей машинного обучения[6].

Другие популярные инструменты машинного обучения:

- `XGBoost`: библиотека с открытым исходным кодом для градиентного бустинга, используемая для классификации и регрессии.

- `LightGBM`: быстрая и эффективная библиотека градиентного бустинга, используемая для классификации и регрессии.

- `CatBoost`: библиотека с открытым исходным кодом для градиентного бустинга, оптимизированная для категориальных данных.

- `H2O.ai`: коммерческая платформа машинного обучения, предоставляющая широкий спектр алгоритмов и инструментов для создания и развертывания моделей машинного обучения.

- Azure Machine Learning: облачная платформа машинного обучения от Microsoft, предоставляющая управляемую среду для разработки, обучения и развертывания моделей машинного обучения.

Эти инструменты машинного обучения предоставляют широкий спектр функциональных возможностей, включая:

- Предварительная обработка и очистка данных
- Выбор признаков и уменьшение размерности
- Обучение и оценка моделей машинного обучения
- Оптимизация гиперпараметров
- Развертывание и мониторинг моделей
- Визуализация и отчетность

Инструменты машинного обучения значительно упрощают и ускоряют процесс разработки и развертывания моделей машинного обучения, позволяя специалистам по данным и разработчикам создавать более точные и эффективные модели для широкого спектра приложений[8].

Глава 2. Проведение анализа данных компании TELCO в Jupiter notebook

2.1 Краткая характеристика компании

TELCO - крупная телекоммуникационная компания, предоставляющая широкий спектр услуг связи для частных лиц и бизнеса. Компания основана в 2005 году и с тех пор завоевала популярность благодаря высокому качеству услуг и инновационным технологиям.

TELCO предлагает такие услуги как мобильная и фиксированная связь, интернет, цифровое телевидение, хостинг, облачные сервисы и другие. Компания активно работает над совершенствованием своей инфраструктуры и предлагает клиентам самые современные технологии.

TELCO имеет широкую сеть филиалов и партнеров, что позволяет обеспечить качественное обслуживание своих клиентов по всей стране. Компания также активно участвует в социальных и благотворительных проектах, стремясь быть ответственным участником рынка и общества в целом.

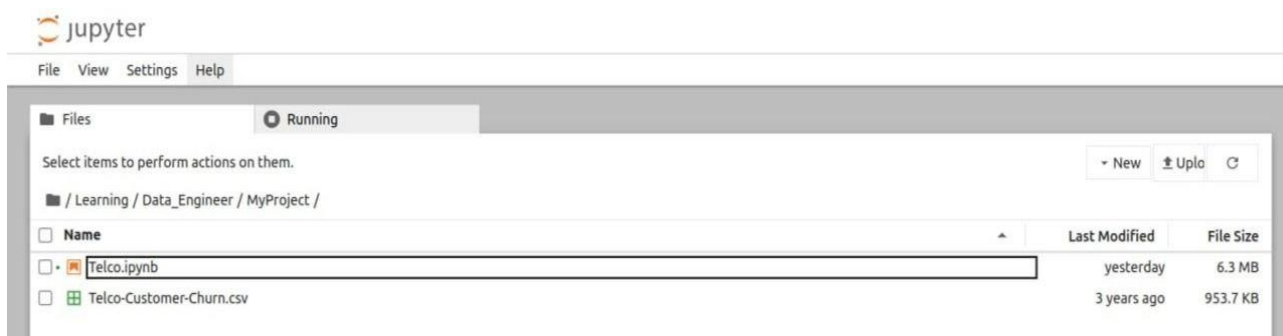
TELCO является лидером в своей отрасли и постоянно совершенствует свои услуги и технологии, чтобы удовлетворить потребности самых требовательных клиентов.

2.2 Создание проекта и активация виртуального окружения

Теперь мы приступили к основной нашей практической работе. Вся наша работа будет вестись в Jupiter notebook с автоматическим использованием языка программирования Python.

Создадим на нашем компьютере директорию MyProject. Поместим туда файл Telco-Customer-Churn.csv, скачанный из сайта www.kaggle.ru. В данном файле содержится информация о различных показателях телекоммуникационной компании TELCO, которые мы будем анализировать.

Запустим Jupiter notebook, перейдём в нашу вновь созданную директорию и создадим там файл Telco.ipynb. В общем, картина будет выглядеть следующим образом, что представлена ниже:



Все файлы будут лежать в одной директории MyProject. По умолчанию будет использоваться виртуальное окружение Conda, являющееся изолированной средой Python, в которой установлены определенные версии пакетов. Оно позволяет пользователям создавать и управлять несколькими средами Python, каждая из которых имеет свой собственный набор установленных пакетов.

2.3 Исследовательский анализ данных в Jupiter notebook. Часть 1

Мы приступаем к исследовательской части нашей работы. Откроем созданный файл Telco.ipynb, и всю работу будем ввести в этом файле. У нас будет очень много кода на языке программирования Python. Поэтому в целях удобства и читаемости мы будем вставлять последовательно фото(скриншот) части нашего кода из Jupiter notebook и описывать его.

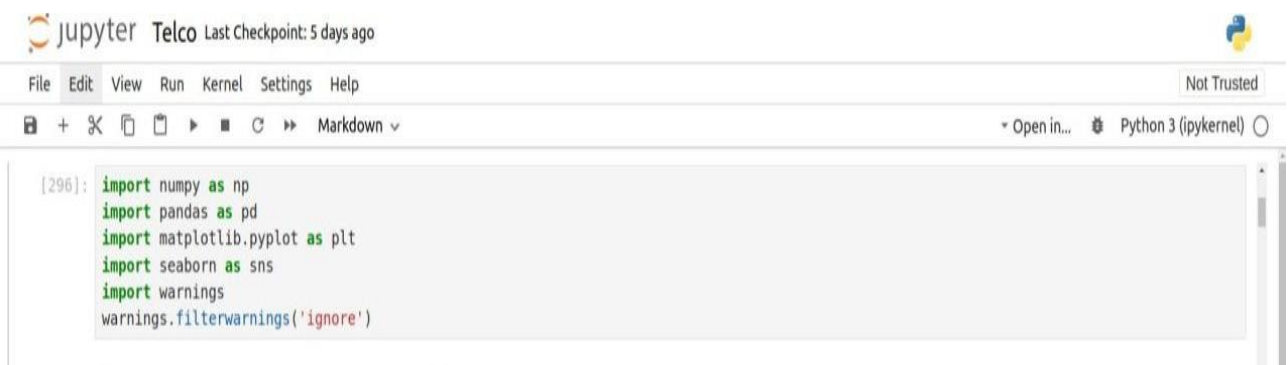
Начнём наше исследование. Ещё раз напомним, что целью нашей работы является создание модели предсказания того, уйдёт ли клиент в отток и выяснить причины оттока. К этому мы будем двигаться постепенно, анализируя различные показатели и метрики.

Опишем обозначения из файла Telco-Customer-Churn.csv.

Информация об атрибутах приведена ниже:

- Customer ID - уникальный номер клиента
- gender - пол
- SeniorCitizen - является ли клиент пенсионером или нет (1, 0)
- Partner - имеется ли у клиента партнер (Да, Нет)
- Dependents - имеет ли клиент иждивенцев или нет (Да, Нет)
- tenure - количество месяцев, которое клиент провел в компании
- PhoneService - имеет ли клиент телефонное обслуживание или нет (Да, Нет)
- MultipleLines - имеет ли клиент несколько линий или нет (Да, Нет, Нет телефонного обслуживания)
- InternetService - поставщик услуг интернет-провайдера (DSL, оптоволокно, нет)
- OnlineSecurity - имеется ли у клиента онлайн-безопасность или нет (Да, Нет, Нет Интернет-сервис)
- OnlineBackup - прибегал ли клиент к резервному копированию в интернете (Да, Нет, Нет телефонного обслуживания)
- DeviceProtection - подключена ли у клиента защита устройства (Да, Нет, Нет телефонного обслуживания)
- TechSupport - имеется ли у клиента техническая поддержка устройства (Да, Нет, Нет телефонного обслуживания)
- StreamingTV - имеется ли у клиента онлайн трансляция телевизионных программ и видео контента через интернет (Да, Нет, Нет телефонного обслуживания)
- StreamingMovies - имеется ли у клиента сервис, который предоставляет доступ к фильмам и телевизионным шоу через интернет для просмотра на устройствах, подключенных к сети (Да, Нет, Нет телефонного обслуживания)
- Contract - срок действия контракта клиента
- PaperlessBilling - отправляет ли клиент счета и квитанции за услуги или товары электронным способом (Да, Нет, Нет телефонного обслуживания)
- PaymentMethod - способ оплаты клиента
- MonthlyCharges - ежемесячные расходы клиента
- TotalCharges - общие расходы клиента
- church - отток клиента (Да, Нет)

Импортируем необходимые библиотеки.



The screenshot shows a Jupyter Notebook window titled "Telco" with a "Last Checkpoint: 5 days ago" status. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and execution. The main area displays a code cell with the following Python code:

```
[296]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Импортируем исследуемый датасет Telco-Customer-Churn.csv, в котором представлена информация о различных переменных(описание выше) компании TELCO.

```
[3]: df = pd.read_csv('Telco-Customer-Churn.csv')
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	
...	
7027	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	
7028	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	
7029	4801-JAZZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	
7030	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	No	
7031	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	Yes	

7032 rows x 21 columns

```
[3]: df = pd.read_csv('Telco-Customer-Churn.csv')
df
```

	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
	Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	No
	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

	Yes	...	Yes	Yes	Yes	Yes	One year	Yes	Mailed check	84.80	1990.50	No
	No	...	Yes	No	Yes	Yes	One year	Yes	Credit card (automatic)	103.20	7362.90	No
	Yes	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.60	346.45	No
	No	...	No	No	No	No	Month-to-month	Yes	Mailed check	74.40	306.60	Yes
	Yes	...	Yes	Yes	Yes	Yes	Two year	Yes	Bank transfer (automatic)	105.65	6844.50	No

Проверим типы данных и есть ли отсутствующие данные в датасете.

```
[298]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   customerID            7032 non-null   object 
 1   gender                7032 non-null   object 
 2   SeniorCitizen         7032 non-null   int64  
 3   Partner               7032 non-null   object 
 4   Dependents            7032 non-null   object 
 5   tenure                7032 non-null   int64  
 6   PhoneService          7032 non-null   object 
 7   MultipleLines         7032 non-null   object 
 8   InternetService       7032 non-null   object 
 9   OnlineSecurity        7032 non-null   object 
10  OnlineBackup          7032 non-null   object 
11  DeviceProtection      7032 non-null   object 
12  TechSupport           7032 non-null   object 
13  StreamingTV           7032 non-null   object 
14  StreamingMovies       7032 non-null   object 
15  Contract              7032 non-null   object 
16  PaperlessBilling       7032 non-null   object 
17  PaymentMethod         7032 non-null   object 
18  MonthlyCharges        7032 non-null   float64 
19  TotalCharges          7032 non-null   float64 
20  Churn                 7032 non-null   object 
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

Видно, что пропусков и отсутствующих данных не имеется.

Выведем статистические метрики для числовых колонок и посмотрим на закономерность.

```
[4]: df.describe()
```

```
[4]:
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000	7032.000000
mean	0.162400	32.421786	64.798208	2283.300441
std	0.368844	24.545260	30.085974	2266.771362
min	0.000000	1.000000	18.250000	18.800000
25%	0.000000	9.000000	35.587500	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.862500	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

Видно, что большинство колонок являются категориальными(нечисловые), поэтому для них будем создавать dummy-переменные.

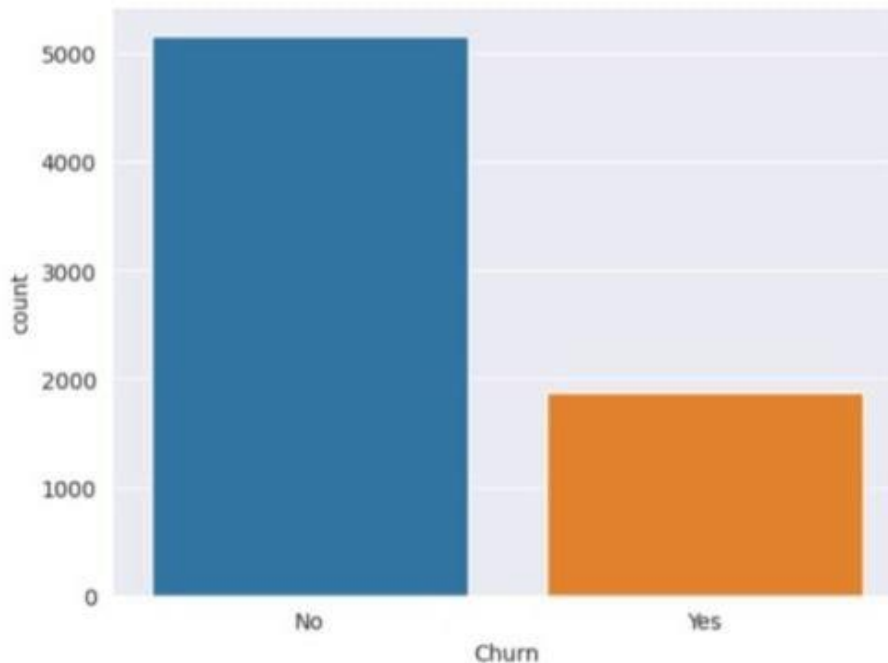
Проведём тщательный анализ. Убедимся ещё раз, что в исследуемых данных нет отсутствующих значений.

```
[300]: df.isnull().sum()
```

```
[300]: customerID      0
gender              0
SeniorCitizen      0
Partner            0
Dependents         0
tenure             0
PhoneService       0
MultipleLines      0
InternetService    0
OnlineSecurity     0
OnlineBackup       0
DeviceProtection   0
TechSupport        0
StreamingTV        0
StreamingMovies    0
Contract           0
PaperlessBilling   0
PaymentMethod      0
MonthlyCharges     0
TotalCharges       0
Churn              0
dtype: int64
```

Мы получили все 0 на всех колонках, что означает отсутствующих значений нет. Теперь перейдём к визуализации данных. Построим график CountPlot для проверки сбалансированности значений колонок с классами(Churn).

```
301]: sns.countplot(x='Churn', data=df, hue='Churn');
```



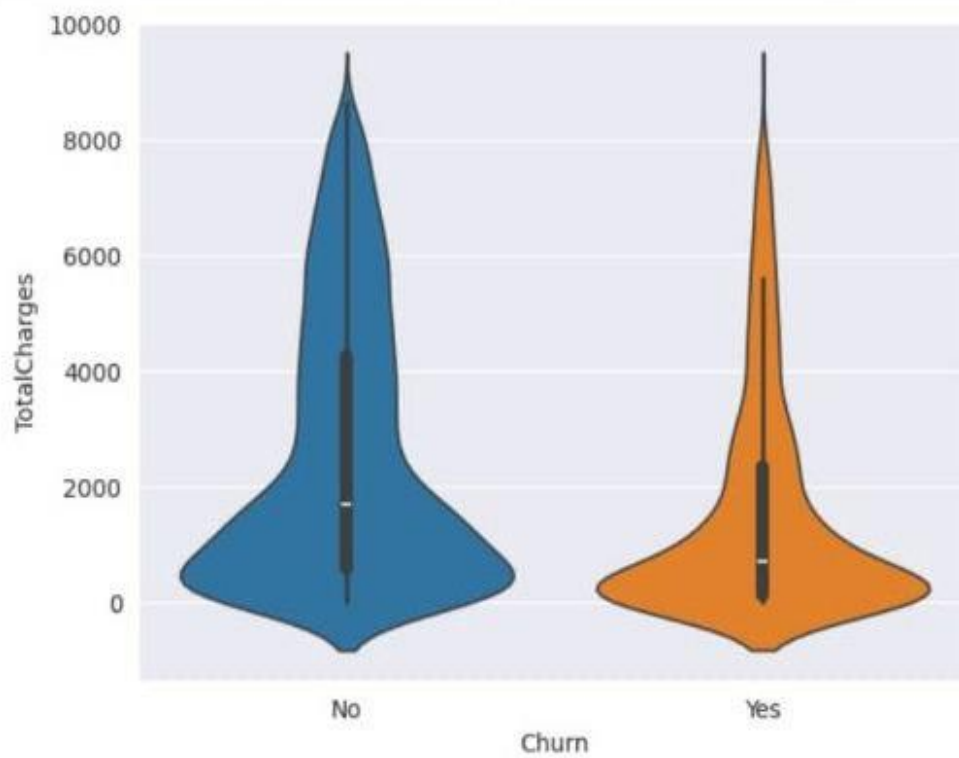
```
302]: df['Churn'].value_counts()
```

```
302]: Churn
No    5163
Yes   1869
Name: count, dtype: int64
```

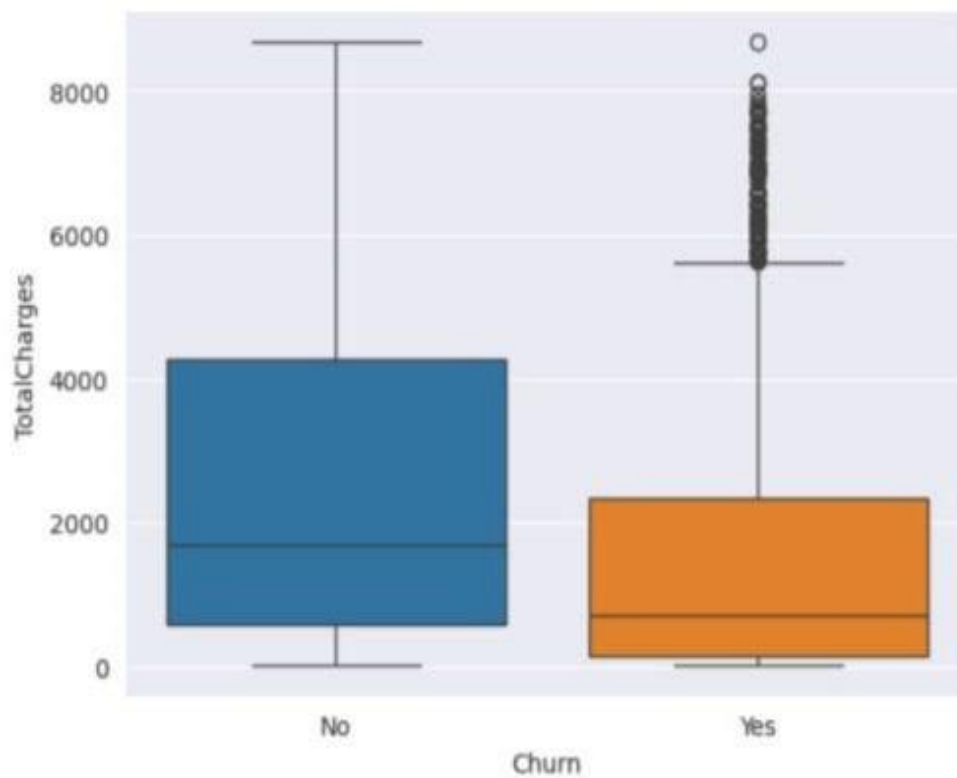
Заметно сразу, что классы немного несбалансированны. Клиентов, которые уходят в отток в 2.8 раз меньше, чем те, кто остаются в компании. Но данная разница между классами не является громадной.

Теперь исследуем распределение колонки TotalCharges по различным категориям Churn с помощью графиков Box Plot и Violin Plot.


```
[303]: sns.violinplot(x='Churn', y='TotalCharges', data=df, hue='Churn');
```



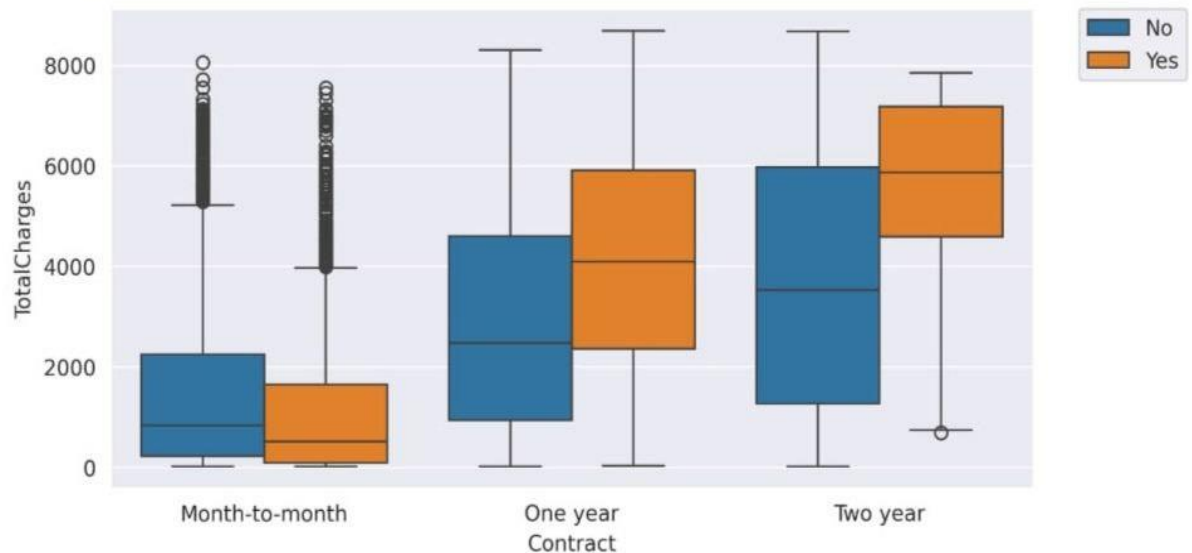
```
[304]: sns.boxplot(x='Churn', y='TotalCharges', data=df, hue='Churn');
```



Здесь также видно, что есть перекос данных и несбалансированность значений. Наблюдаются определённое количество выбросов у тех, кто уходит в отток. К ним вернёмся позже.

Построим график `boxplot` с распределением колонки `TotalCharges` для различных типов контрактов.

```
[305]: plt.figure(figsize=(8,4), dpi=300)
sns.boxplot(data=df, x='Contract', y='TotalCharges', hue='Churn')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.);
```



Если мы посмотрим на тип контракта `Month-to-month`(контракты на определённое количество месяцев, то есть меньше года), то значение `Churn` в них примерно одинаковое. Почти схожая картина и для контрактов `One year contract`, а вот для контрактов `Two year` видно, что меньше клиентов уходят в отток.

Теперь подумаем, что могла бы сделать компания `TELCO`, чтобы меньше людей уходило в отток.

По вышеприведённым графикам видно, что чем больше значение `TotalCharges`(общие расходы) на любые виды контрактов, тем больше людей уходят в отток, и либо находят выгодные условия у конкурентов или, к примеру, не получают качественный сервис обслуживания в компании `TELCO`. Данный факт следует взять на вооружение.

Стоит задуматься также над тем, что почему при больших расходах большая часть клиентов не уходят в отток.

Построим график `barplot` с корреляцией числовыми признаками с целевой переменной (`Churn`). Для категориальных признаков проведём конвертацию их в `dummy`-переменные, так как корреляция вычисляется только для числовых колонок.



```
[7]: corr_df = pd.get_dummies(df[['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
                                'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'InternetService',
                                'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn']]).corr()
```

Отметим, что мы выделим в переменной `Churn` только атрибут `Churn_Yes` (уход в отток) и применим её при построении графика `barplot`.

На основе этого теперь построим график `barplot`.



```
44 rows x 44 columns

[308]: corr_yes_churn = corr_df['Churn_Yes'].sort_values().iloc[1:-1]

[309]: plt.figure(figsize=(10,4), dpi=300)
plt.title('Корреляция признаков с оттоком Churn_Yes')
sns.barplot(x=corr_yes_churn.index, y=corr_yes_churn.values, hue=corr_yes_churn)
plt.xticks(rotation=90);
```

В итоге получим следующий рисунок:



График корреляции с атрибутом Churn_Yes показывает, что для краткосрочных контрактов(Contract_Month-to-month) с большими расходами характерен высокий отток клиентов. Тут прямая зависимость: с возрастанием значения Contract_Month-to-month, возрастает в количественном соотношении Churn_Yes.

Мы выяснили, что длительность контракта по времени оказывает немаловажное значение на отток клиентов и соответственно будем анализировать влияние всех видов контрактов на отток клиентов из компании TELCO.

Проведём анализ оттока и когорт. Разобьём всех абонентов(клиентов) на отдельные сегменты по длительности обслуживания в компании. Это нам поможет понять, как меняется поведение клиентов в зависимости от срока обслуживания.

Посмотрим какие у нас есть виды контрактов.

```
[310]: df['Contract'].unique()
[310]: array(['Month-to-month', 'One year', 'Two year'], dtype=object)
[311]: df['Contract'].value_counts()
[311]: Contract
Month-to-month    3875
Two year          1685
One year          1472
Name: count, dtype: int64
```

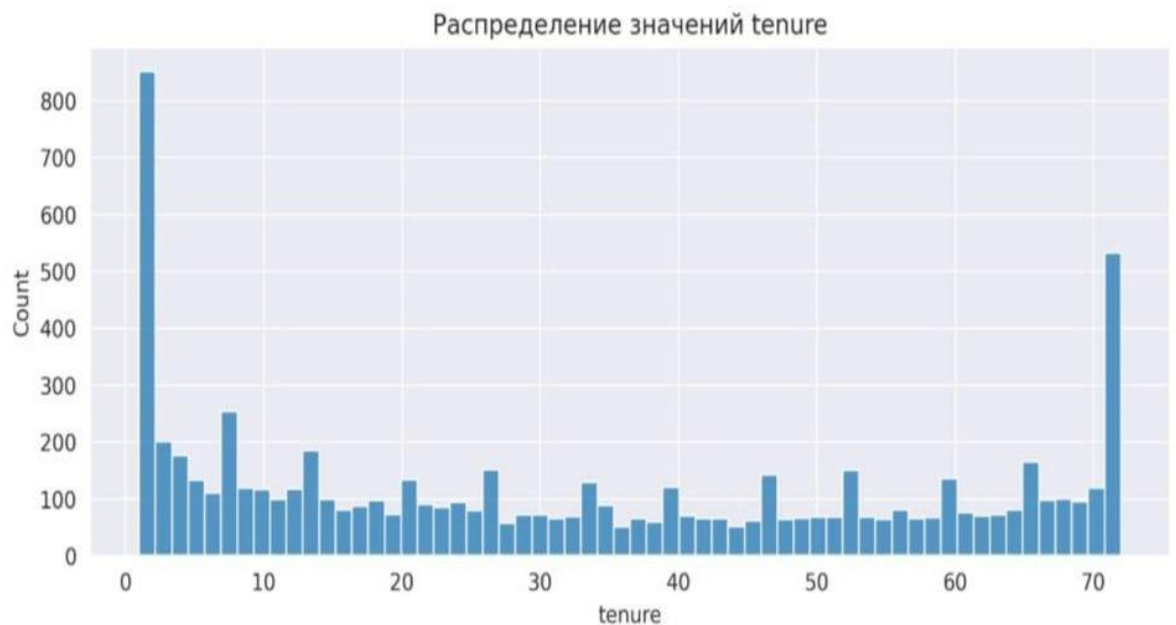
Посмотрим сколько месяцев клиент являлся или является абонентом в компании и также посмотрим какое количество клиентов по каждому месяцу.

```
[16]: df['tenure'].unique()
[16]: array([ 1, 34,  2, 45,  8, 22, 10, 28, 62, 13, 16, 58, 49, 25, 69, 52, 71,
          21, 12, 30, 47, 72, 17, 27,  5, 46, 11, 70, 63, 43, 15, 60, 18, 66,
           9,  3, 31, 50, 64, 56,  7, 42, 35, 48, 29, 65, 38, 68, 32, 55, 37,
          36, 41,  6,  4, 33, 67, 23, 57, 61, 14, 20, 53, 40, 59, 24, 44, 19,
          54, 51, 26, 39])
[17]: df['tenure'].value_counts()
[17]: tenure
1      613
72     362
2      238
3      200
4      176
...
38      59
28      57
39      56
44      51
36      50
Name: count, Length: 72, dtype: int64
```

Видна следующая закономерность по признаку tenure (количество месяцев, которое клиент провёл в компании): чем больше длительность контракта по времени, тем меньше клиентов заключают договор с компанией. Хотя эта зависимость не строгая, наблюдаются определённые цикличности, но всё же следует заострить внимание на краткосрочных контрактах.

Построим гистограмму с распределением значений колонки tenure.

```
[314]: plt.figure(figsize=(10,4), dpi=300)
plt.title('Распределение значений tenure')
sns.histplot(data=df, x='tenure', bins=60);
```



Видно, что очень много клиентов имеют значение tenure от 1 до 2 месяцев. Второе пиковое значение находится в районе 72. На графике отчётливо заметны всплески между первым и вторым пиковыми значениями, что скорее означает продление годовых и двухлетних контрактов.

Теперь построим график `displot` по признаку tenure, колонке Contract, строке Churn и посмотрим на получившийся результат.

Рисунок представлен ниже.

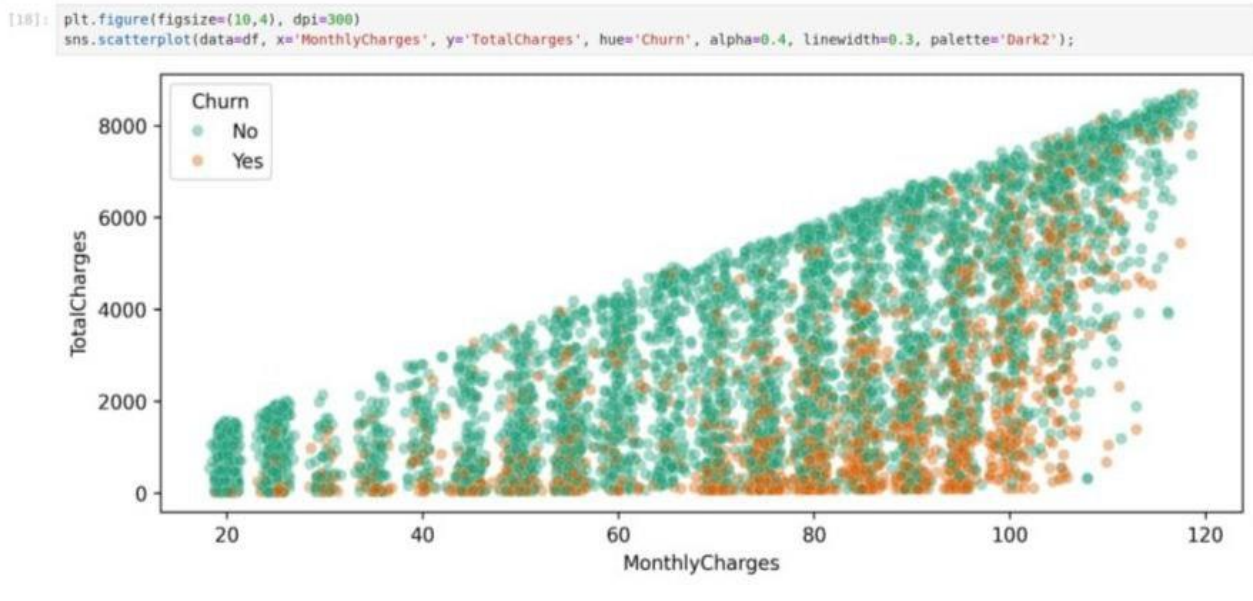


По графику `displot` видно, что на годовых и двухгодичных контрактах, уходящих в отток клиентов мало, поэтому основные усилия можно направить на контракты 'Month-to-month'. Причём для краткосрочных контрактов (меньше года) пик значения, наблюдается от 1 до 2 месяцев для клиентов уходящих в отток. Но затем идёт пологое распределение. Следует учесть тот факт, что ранее мы наблюдали большой отток клиентов из-за больших расходов на долгосрочных контрактах. Возможно, отток на контрактах 'Month-to-month' обусловлен из-за высоких цен (стоимость тарифа) чем на долгосрочных контрактах. В таком случае кампании Telco следовало бы проводить активную маркетинговую акцию, т.е. предлагать клиентам с контрактом 'Month-to-month' перейти на годовые или двухгодичные контракты, тем самым доход от этих абонентов снизится, но в долгосрочной перспективе кампания получит большой суммарный доход от этого абонента.

Таким образом, анализ графика `displot` поможет определить стратегию по снижению оттока клиентов и увеличению доходов компании. Важно учитывать

особенности каждого типа контракта и предлагать клиентам наиболее выгодные и удобные условия для них.

Построим график scatterplot для колонок Total Charges и Monthly Charges, колонки раскрасим по колонке Churn.



Очередной раз подтверждается тот факт, что в отток уходят те, у кого большие расходы. В принципе, это логично, чем больше ежемесячные расходы, тем больше у клиентов возникает причин искать более дешёвые тарифы у конкурентов.

Да, и вообще, это очень дискуссионный вопрос: "Что для компании важнее: либо абонент с ежемесячными высокими платежами(MonthlyCharges), либо абонент с высокими общими платежами(TotalCharges)". Эти два показателя связаны между собой через длительность по времени(tenure).

На графике scatterplot мы подтверждаем, что в отток уходят клиенты с высокими ежемесячными расходами, причём в основном у них небольшие значения(TotalCharges).

Это может быть связано с тем, что клиенты с большими ежемесячными расходами могут быть более ценными для компании в долгосрочной перспективе, если они остаются на услугах компании длительное время. Однако, клиенты с высокими общими платежами могут приносить компании больше прибыли в краткосрочной перспективе.

Таким образом, важно учитывать оба показателя при анализе оттока клиентов и разработке стратегии удержания.

2.4 Исследовательский анализ данных в Jupiter notebook. Часть 2

Создадим когорты(сегменты) по колонке `tenure`. Для начала поместим различные значения `tenure` (1 месяц, 2 месяц, 3 месяца и так далее) в отдельные когорты(сегменты).

Для каждого уникального значения колонки `tenure` в качестве когорты, вычислим процент оттока (`churn rate`) - это количество тех людей, кто ушёл в отток в когорте, как процент от общего количества людей в данной когорте. Вычислим такой процент отдельно для каждой когорты. В итоге должно получиться когорты от 1 до 72 месяцев, и с увеличением количества месяцев уровень оттока должен снижаться. Это разумно, потому что чем дольше клиент пользуется услугами компании, тем вероятнее то, что его/её всё устраивает, и он/она продолжит пользоваться этими услугами.

Чтобы выполнить разбиение отдельные когорты по каждому месяцу, проведём ряд операций и преобразований. Отдельно разобьём кто уходит в отток(`yes_churn`) и тех, кто остаётся в компании(`no_churn`). И по этому разбиению уже определим, процент ушедших по каждому месяцу.

Рисунки представлены ниже.


```
[22]: no_churn = df.groupby(['Churn', 'tenure']).count().transpose()['No']
```

```
[23]: yes_churn = df.groupby(['Churn', 'tenure']).count().transpose()['Yes']
```

```
[24]: churn_rate = 100 * yes_churn / (no_churn + yes_churn)
```

```
churn_rate
```

	tenure	1	2	3	4	5	6	7	8	9	10	...	63	64	65	66
customerID	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
gender	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
SeniorCitizen	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
Partner	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
Dependents	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
PhoneService	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
MultipleLines	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
InternetService	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
OnlineSecurity	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
OnlineBackup	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
DeviceProtection	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
TechSupport	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
StreamingTV	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
StreamingMovies	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
Contract	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
PaperlessBilling	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
PaymentMethod	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
MonthlyCharges	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	
TotalCharges	61.990212	51.680672	47.0	47.159091	48.120301	36.363636	38.931298	34.146341	38.655462	38.793103	...	5.555556	5.0	11.842105	14.606742	

19 rows × 72 columns

```
[25]: churn_rate.transpose()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
tenure											
1	61.990212	61.990212	61.990212	61.990212	61.990212	61.990212	61.990212	61.990212	61.990212	61.990212	61.990212
2	51.680672	51.680672	51.680672	51.680672	51.680672	51.680672	51.680672	51.680672	51.680672	51.680672	51.680672
3	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000
4	47.159091	47.159091	47.159091	47.159091	47.159091	47.159091	47.159091	47.159091	47.159091	47.159091	47.159091
5	48.120301	48.120301	48.120301	48.120301	48.120301	48.120301	48.120301	48.120301	48.120301	48.120301	48.120301
...
68	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000
69	8.421053	8.421053	8.421053	8.421053	8.421053	8.421053	8.421053	8.421053	8.421053	8.421053	8.421053
70	9.243697	9.243697	9.243697	9.243697	9.243697	9.243697	9.243697	9.243697	9.243697	9.243697	9.243697
71	3.529412	3.529412	3.529412	3.529412	3.529412	3.529412	3.529412	3.529412	3.529412	3.529412	3.529412
72	1.657459	1.657459	1.657459	1.657459	1.657459	1.657459	1.657459	1.657459	1.657459	1.657459	1.657459

72 rows × 19 columns

```
[321]: churn_rate.transpose()['customerID']
```

```
[321]: tenure
```

```
1    61.990212
```

```
2    51.680672
```

```
3    47.000000
```

```
4    47.159091
```

```
5    48.120301
```

```
...
```

```
68    9.000000
```

```
69    8.421053
```

```
70    9.243697
```

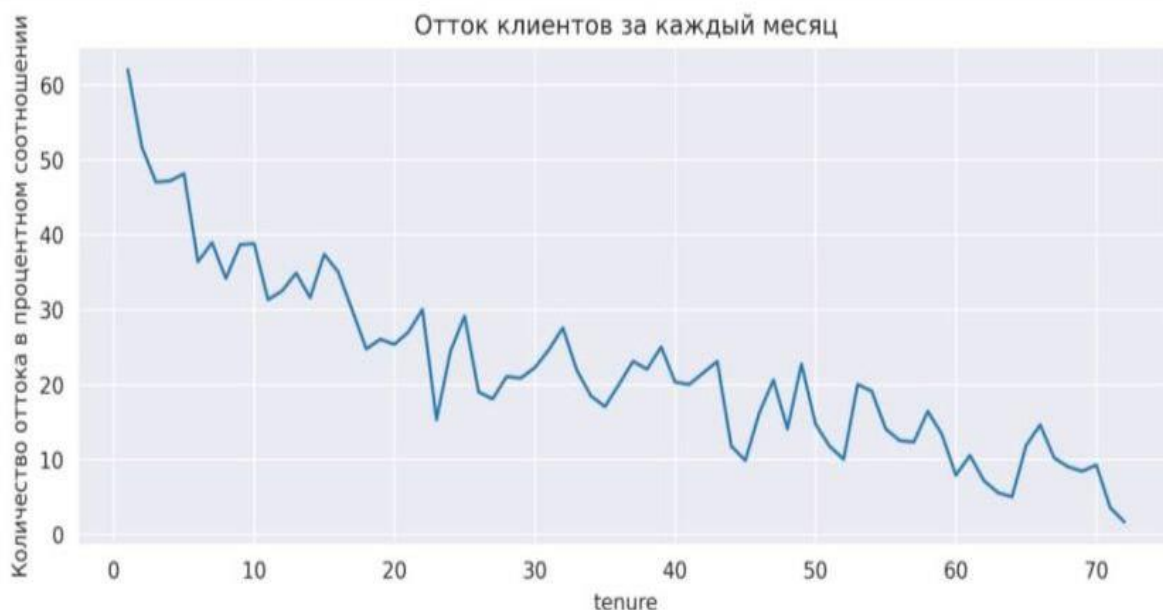
```
71    3.529412
```

```
72    1.657459
```

```
Name: customerID, Length: 72, dtype: float64
```


Построим график, показывающий отток клиентов за каждый месяц.

```
[322]: plt.figure(figsize=(10,4), dpi=300)
plt.title('Отток клиентов за каждый месяц')
plt.xlabel('Количество месяцев')
plt.ylabel('Количество оттока в процентном соотношении')
churn_rate.transpose()['customerID'].plot();
```



Видно, что на графике пологое распределение. Чем больше значение tenure, тем меньше процент оттока в когорте. Отчётливо также видно, до 12 месяцев отток в процентном соотношении большой. После 12 месяцев идёт постепенное снижение оттока. Очередной раз убеждаемся, что упор компании нужно сделать на контракты до года.

Оптимизируем процесс исследования. Будем двигаться в следующем ключе. На основе колонки tenure создадим колонку с названием Tenure Cohort, в которой будут следующие 4 категории:

- 0-12 месяцев
- 12-24 месяца
- 24-48 месяцев
- Более 48 месяцев

На основе этого создадим функцию для преобразования данных, которая будет иметь следующий вид:

```
[323]: def cohort(tenure):
        if tenure < 13:
            return '0-12 месяцев'
        elif tenure < 25:
            return '12-24 месяцев'
        elif tenure < 49:
            return '24-48 месяцев'
        else:
            return 'Более 48 месяцев'
```

```
[324]: df['Tenure Cohort'] = df['tenure'].apply(cohort)
```

```
[325]: df[['Tenure Cohort', 'tenure']]
```

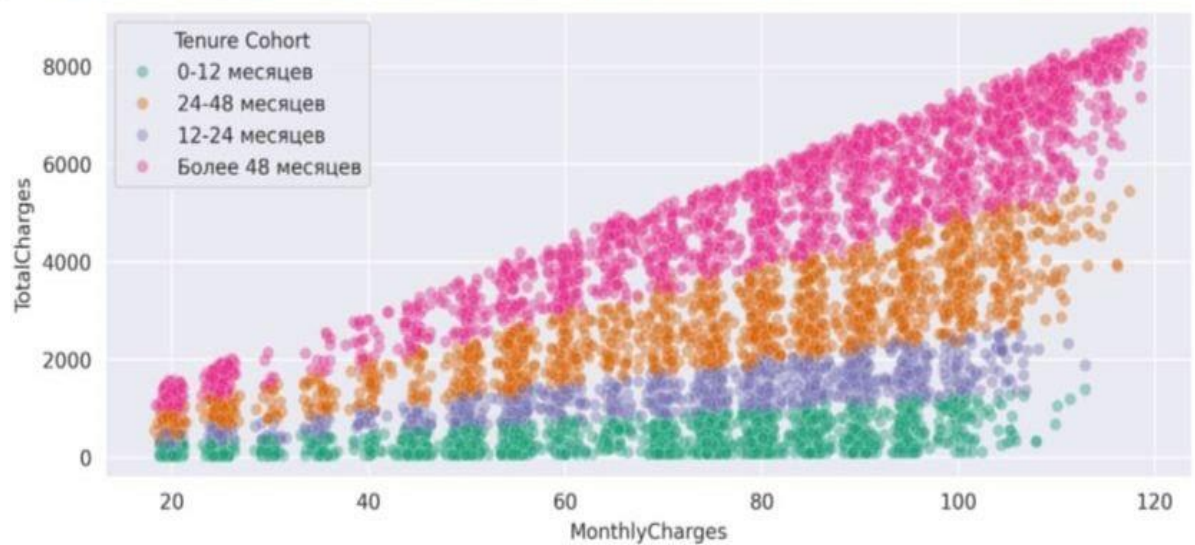
```
[325]:
```

	Tenure Cohort	tenure
0	0-12 месяцев	1
1	24-48 месяцев	34
2	0-12 месяцев	2
3	24-48 месяцев	45
4	0-12 месяцев	2
...
7027	12-24 месяцев	24
7028	Более 48 месяцев	72
7029	0-12 месяцев	11
7030	0-12 месяцев	4
7031	Более 48 месяцев	66

7032 rows x 2 columns

Построим график scatterplot для Total Charges и Monthly Charges, раскрашивая график разными цветами по колонке Tenure Cohort, которую создали ранее.

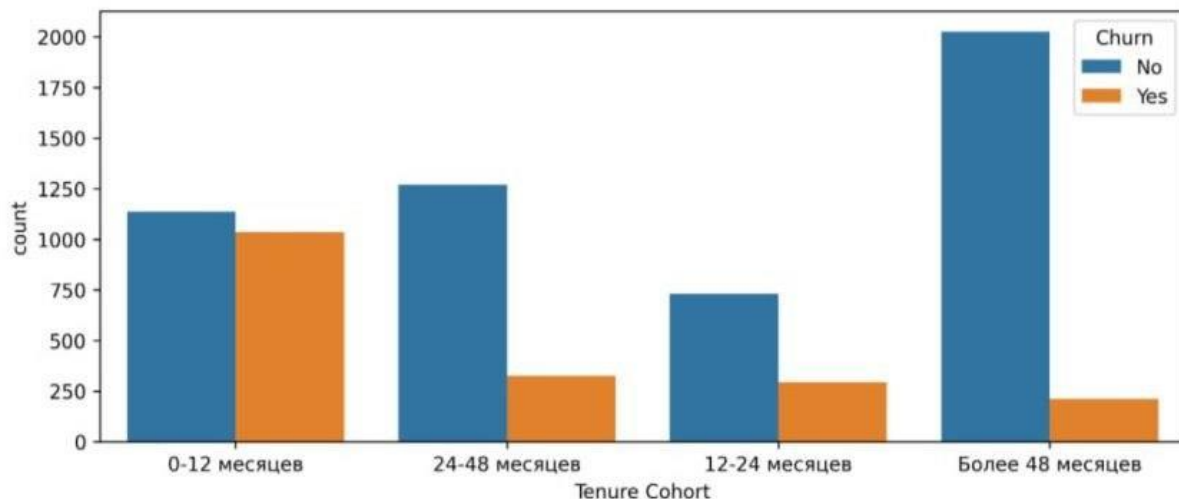
```
[326]: plt.figure(figsize=(10,4), dpi=300)
        sns.scatterplot(data=df, x='MonthlyCharges', y='TotalCharges', hue='Tenure Cohort', alpha=0.4, linewidth=0.3, palette='Dark2');
```



На рисунке всё выглядит логично. Чем больше когорта, тем большее значение TotalCharges она принимает. Сам рисунок является более читаемым и легко воспринимаемым для анализа.

Построим график countplot с количеством ушедших и не ушедших в отток людей в каждой когорте.

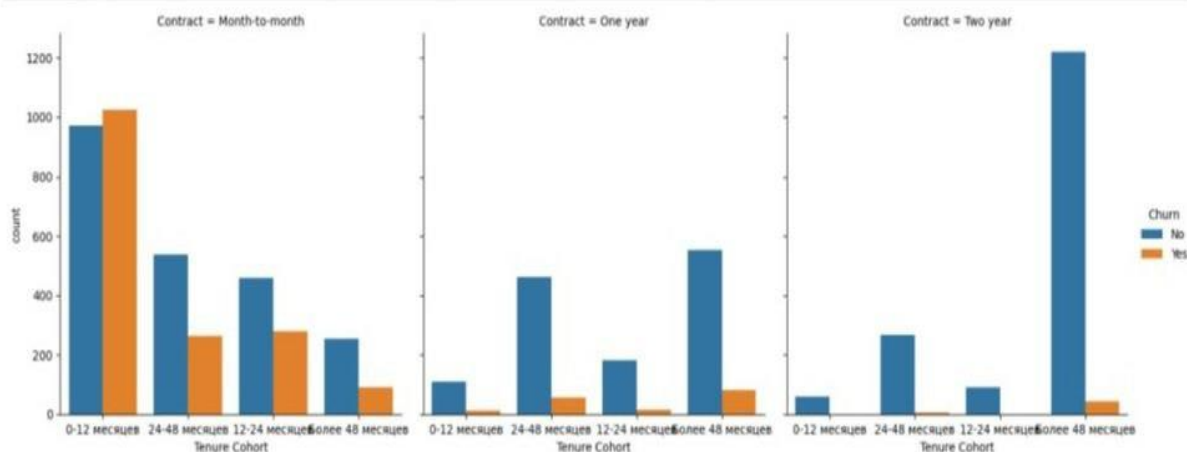
```
[63]: plt.figure(figsize=(10,4), dpi=300)
sns.countplot(data=df, x='Tenure Cohort', hue='Churn');
```



Отчётливо видно, что клиенты с длительностью обслуживания более 48 месяцев менее подвержены к оттоку, а те, кто пользуются услугами компании менее 12 месяцев, то вероятность оттока заметно возрастает.

Создадим набор графиков CountPlot с количеством клиентов по различным когортам Tenure Cohort. Для каждого типа контракта создадим отдельный график, и раскрасим эти графики разными цветами по колонке Churn.

```
[68]: sns.catplot(data=df, x='Tenure Cohort', hue='Churn', kind='count', col='Contract');
```



Вышеприведённый график, как и другие, также подтверждает тот факт, что клиенты на однолетнем и двухлетнем контрактах меньше уходят в отток вне зависимости от длительности пользования услугами. Об этом свидетельствует то, что процент оттока на контрактах продолжительностью в один и два года остается ниже, чем на контрактах другой продолжительности, как на начальных

этапах пользования услугами, так и на более поздних. Вероятно, клиенты, заключившие контракт на год или два, более лояльны и удовлетворены услугами провайдера.

Глава 3. Применение моделей машинного обучения для прогнозирования оттока клиентов

3.1 Выбор модели машинного обучения

Теперь мы подошли к основной части нашей работы – создание предиктивной модели машинного обучения. Возникает невольный вопрос: “Какую модель следует выбирать?”

Сделаем небольшое напоминание, что при выборе модели машинного обучения необходимо учитывать следующие факторы:

1. Цель задачи: определить, какую задачу мы хотим решить (классификация, регрессия, кластеризация и т. д.).
2. Объем и характер данных: учесть размер выборки, количество признаков, их тип (категориальные или числовые) и распределение.
3. Время обучения и предсказания: учесть требования к скорости работы модели.
4. Интерпретируемость: если важно понимать, как модель принимает решения, то лучше выбрать модели с хорошей интерпретируемостью.
5. Регуляризация: если имеется много признаков, которые могут быть нерелевантными, рассмотреть модели с возможностью регуляризации.
6. Разреженность данных: если есть разреженные данные (многие нулевые значения), выбрать модели, которые умеют с ними хорошо работать.
7. Проверка модели: использовать кросс-валидацию и метрики качества, чтобы выбрать наилучшую модель.
8. Наличие предобученных моделей: если у нас ограниченные ресурсы, то лучше рассмотреть использование предобученных моделей для нашей задачи.

Исходя из этих факторов, можно выбрать наиболее подходящую модель машинного обучения для нашей конкретной задачи. Важно также помнить, что иногда эффективно использовать не одну модель, а ансамбль моделей для повышения качества предсказаний.

В нашем случае данные размечены, что означает классифицированы и соотнесены к определённому классу. Здесь уместно использовать обучение с

учителем. Выбор наш падёт на модели на основе деревьев, логистическая регрессия, метод опорных векторов (SVM), метод К-ближайших соседей (KNN).

Данные модели машинного обучения имеют несколько преимуществ:

1. Интерпретируемость: легко понимать и интерпретировать, поскольку они представляют собой простую логическую структуру. Это делает их полезными для принятия решений и объяснения прогнозов.

2. Универсальность: могут быть использованы как для задач классификации, так и для задач регрессии. Они могут быть применены к данным с различными типами признаков, включая категориальные признаки.

3. Работа с нелинейными зависимостями: могут обрабатывать сложные нелинейные зависимости в данных без необходимости их линеаризации.

4. Устойчивость к выбросам: хорошо устойчивы к выбросам и шуму в данных.

Также модели на основе деревьев, логистическая регрессия, метод опорных векторов (SVM), метод К-ближайших соседей (KNN) могут быть применены в различных областях, включая:

1. Классификация: часто используются для классификации объектов на основе их признаков. Например, они могут быть применены в медицине для диагностики заболеваний, в финансах для прогнозирования рисков и мошенничества, или в маркетинге для выявления целевой аудитории.

2. Регрессия: могут использоваться для прогнозирования непрерывных значений. Например, они могут быть применены в финансовой аналитике для прогнозирования цен на акции или в промышленности для прогнозирования времени до отказа оборудования.

3. Кластеризация: могут использоваться для кластеризации данных и выявления групп похожих объектов.

4. Ранжирование: могут быть использованы для ранжирования объектов в зависимости от их значимости или релевантности.

Использование этих моделей даст целый спектр ответов, которые мы хотим узнать.

Исследуем шесть моделей на основе классификации: одно дерево решений, случайный лес, адаптивный бустинг, градиентный бустинг, логистическая регрессия, метод опорных векторов (SVM), метод К-ближайших соседей (KNN), а далее сравним результаты и выберем из них наилучшую с меньшей ошибкой предсказания (в нашем случае метрика recall).

3.2 Модель Decision Tree

Модель Decision Tree (одно дерево решений) - метод машинного обучения, который использует структуру дерева для принятия решений. Каждый узел дерева представляет собой условие, которое определяет, какие данные нужно анализировать, а каждое ребро соответствует возможному результату этого условия. Путем прохождения по дереву от корня к листьям можно получить прогноз для конкретного наблюдения. Decision Tree является простым и интерпретируемым методом, который может быть эффективно использован для задач классификации и регрессии.

Разделим все данные на признаки X и целевую переменную Y. Создадим dummy-переменные, где это необходимо, а также посмотрим, есть ли какие-то признаки, которые можно было бы удалить.

```
[329]: X = df.drop(['Churn', 'customerID'], axis=1)
```

```
[330]: X = pd.get_dummies(X, drop_first=True)
```

```
[331]: y = df['Churn']
```

Выполним разбиение данных на обучающий и тестовый наборы (train_test_split), откладывая в сторону 10% данных для тестирования. В решениях мы используем random_state=101.

```
[77]: from sklearn.model_selection import train_test_split

[78]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)

[ ]:

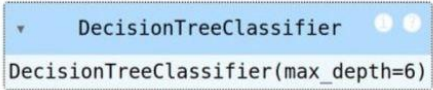
[ ]:
```

Построим модель Decision Tree, оценим её работу и выполним её обучение.

```
[83]: from sklearn.tree import DecisionTreeClassifier

[84]: dt = DecisionTreeClassifier(max_depth=6)

[85]: dt.fit(X_train, y_train)
```

[85]: 

Оценим метрики работы модели дерева решений, выведем отчёт "classification report" и нарисуем график с матрицей ошибок (confusion matrix).

```
[86]: from sklearn.metrics import classification_report, confusion_matrix

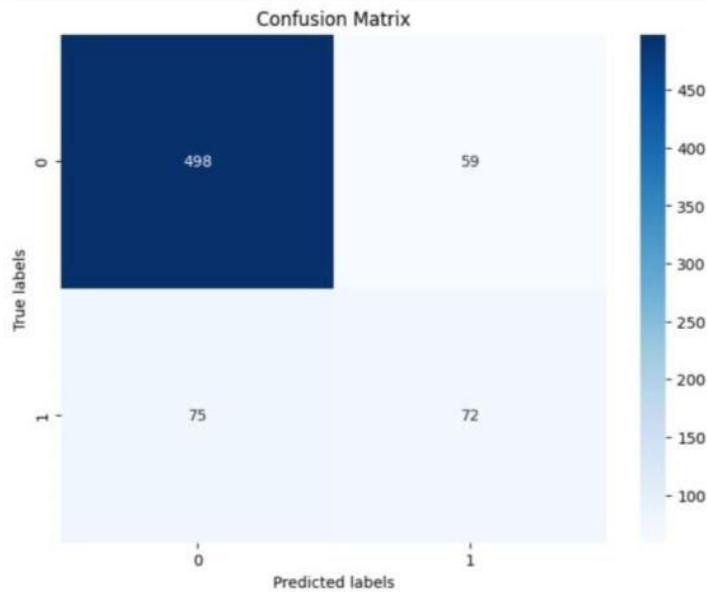
[87]: preds = dt.predict(X_test)

[88]: print(classification_report(y_test, preds))
```

	precision	recall	f1-score	support
No	0.87	0.89	0.88	557
Yes	0.55	0.49	0.52	147
accuracy			0.81	704
macro avg	0.71	0.69	0.70	704
weighted avg	0.80	0.81	0.81	704

По данным видно, что модель лучше работает для класса Churn['No'] чем Churn['Yes']. Это логично и разумно, так как в классе Churn['No'] содержится больше данных.


```
[90]: conf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show();
```



Мы видим, что неправильно классифицированы 75 абонентов, когда наша модель предсказала, что оттока не будет, но отток случился и именно это число мы хотели бы минимизировать, так как здесь мы упускаем тех, кто уходит в отток. Кроме этого, есть ещё 59 людей, для которых модель предсказала, что они уйдут в отток, но на самом деле они не ушли в отток.

Оценим, какие признаки играют наибольшую роль в принятии решений моделью.

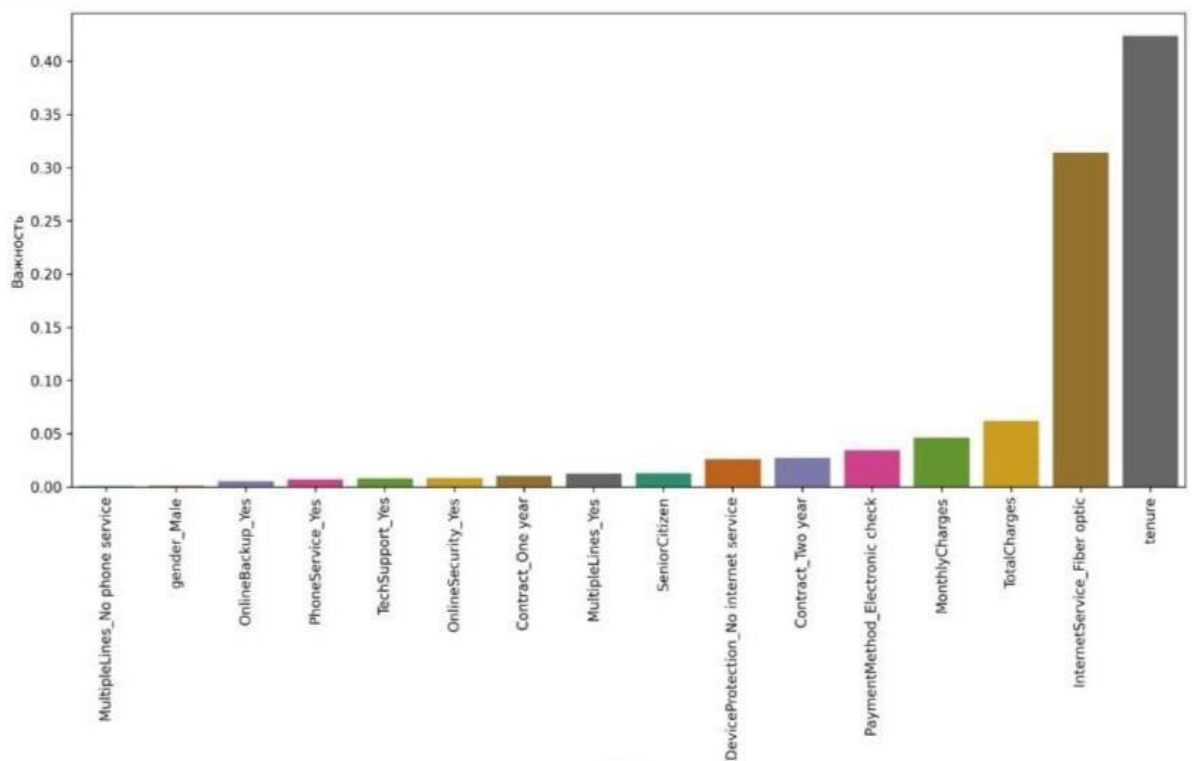
```
[95]: imp_feats = pd.DataFrame(data=dt.feature_importances_, index=X.columns, columns=['Важность'])
[96]: imp_feats = imp_feats.sort_values('Важность')
[97]: imp_feats = imp_feats[imp_feats['Важность'] > 0]
imp_feats
```

```
[97]:
```

	Важность
MultipleLines_No phone service	0.000890
gender_Male	0.001237
OnlineBackup_Yes	0.005341
PhoneService_Yes	0.006962
TechSupport_Yes	0.007868
OnlineSecurity_Yes	0.008376
Contract_One year	0.010021
MultipleLines_Yes	0.012432
SeniorCitizen	0.012680
DeviceProtection_No internet service	0.026290
Contract_Two year	0.027065
PaymentMethod_Electronic check	0.034436
MonthlyCharges	0.046115
TotalCharges	0.062313
InternetService_Fiber optic	0.314060
tenure	0.423914

Визуализируем полученные данные.

```
[98]: plt.figure(figsize=(14, 6), dpi=300)
sns.barplot(data=imp_feats, x=imp_feats.index, y='Важность', palette="Dark2")
plt.xticks(rotation=90);
```



Ранее мы видели, что признак `tenure` хорошо коррелирует с целевой переменной. Однако, есть ещё один интересный признак `InternetService_Fiber optic`, который ранее не встречали нигде в исследовании. В этом случае нужно выяснять отдельно, почему клиенты, которые нуждаются в интернете по

оптоволокну, могут уходить в отток. Возможно у компании TELCO выше цены, чем у конкурентов, именно по этой услуге.

К примеру, для проверки данной гипотезы можно провести анализ цен на услуги интернет - провайдера TELCO и его конкурентов, сравнить их тарифные планы и узнать, предлагают ли конкуренты более привлекательные условия для клиентов, нуждающихся в интернете по оптоволокну.

Также стоит провести опрос клиентов, уходящих в отток, чтобы узнать, почему они решают сменить интернет - провайдера. Вопросы опроса могут касаться стоимости услуг, качества обслуживания, скорости интернета и других факторов, которые могут повлиять на принятие решения об уходе.

Исходя из результатов анализа цен, тарифных планов и опроса клиентов, можно будет сделать вывод о том, является ли цена услуг интернета по оптоволокну одной из причин ухода клиентов в отток от TELCO. Если это подтвердится, компания сможет принять меры для улучшения своего конкурентного положения на рынке и удержания клиентов.

3.3 Модель Random Forest

Модель Random Forest (случайный лес) – это алгоритм машинного обучения, который использует комбинацию нескольких деревьев принятия решений для решения задач классификации или регрессии. Он работает путем создания леса случайных деревьев во время обучения и совершения предсказаний на основе голосования или усреднения результатов всех деревьев. Random Forest обладает высокой точностью, устойчив к переобучению и способен обрабатывать как категориальные, так и количественные данные.

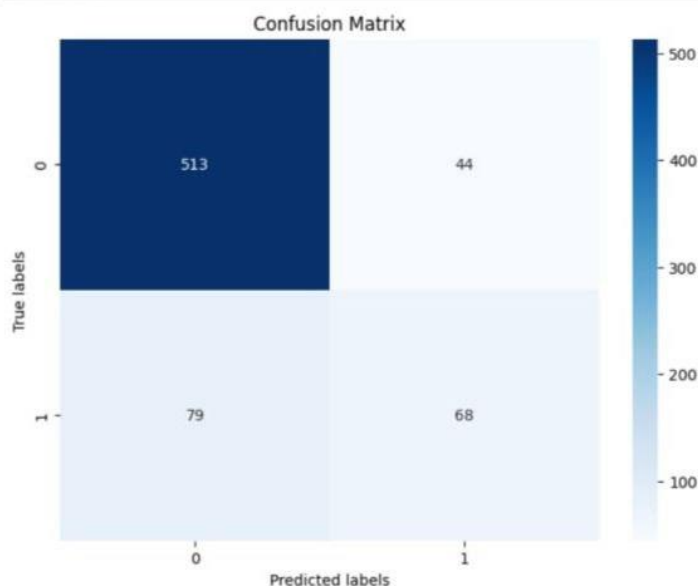
```
[101]: from sklearn.ensemble import RandomForestClassifier
[102]: rf = RandomForestClassifier(max_depth=6)
[103]: rf.fit(X_train, y_train)
[103]: ▼ RandomForestClassifier
RandomForestClassifier(max_depth=6)
[104]: preds = rf.predict(X_test)
[105]: print(classification_report(y_test, preds))
```

	precision	recall	f1-score	support
No	0.87	0.92	0.89	557
Yes	0.61	0.46	0.53	147
accuracy			0.83	704
macro avg	0.74	0.69	0.71	704
weighted avg	0.81	0.83	0.82	704

Видно, что эта модель работает немного хуже чем модель Decision Tree (важно счесть, какая метрика учитывается и сравнивается). Причина в том, что дерево решений получилось слишком глубоким, и из-за этого модель может получится переобученной(overfitting), т.е. хорошо работать на обученных данных и плохо работать на тестовых данных. Поэтому в определённых ситуациях полезно ограничивать глубину дерева(max_depth).

Построим матрицу ошибок и посмотрим на результат.

```
[106]: conf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show();
```



В матрице ошибок (Confusion Matrix) мы видим интересную картину. Эта модель имеет более высокие метрики accuracy и precision с поправкой на ограничение(max_depth=6). Однако, если опираться на нашу задачу, то мы хотим минимизировать количество тех абонентов, которые по факту ушли в отток, но наша модель думает, что они не уйдут в отток. В данной матрице ошибок это значение равно 79, хотя раньше это значение было равно 75 для модели одного дерева. И за это число отвечает метрика recall, и эту метрику нам необходимо минимизировать. Вывод: модель случайного леса показывает худшее значение метрики recall, чем модель дерево решений.

Поэтому, если основной задачей является минимизация количества ложно отрицательных результатов (когда модель предсказывает, что клиент останется, но он на самом деле уходит), то лучше использовать модель дерева решений с глубиной равной 6. В данном случае accuracy и precision могут быть не такими высокими, как у модели случайного леса, но recall будет лучше.

Но делать поспешных выводов пока не стоит. Рассмотрим ещё две модели, но уже с применением гиперпараметров.

3.4 Модель Boosted Trees

Модель Boosted Trees (расширяемые деревья) – это ансамблевый метод машинного обучения, который объединяет несколько слабых моделей (деревьев решений) в одну сильную модель.

Мы построим поочерёдно модели AdaBoost и Gradient Boosting.

Создадим модель AdaBoost с применением гиперпараметров, оценим её работу и выполним обучение.

```
[113]: from sklearn.ensemble import AdaBoostClassifier
[114]: ada_model = AdaBoostClassifier(n_estimators=118)
[115]: ada_model.fit(X_train, y_train)
[115]: * AdaBoostClassifier
AdaBoostClassifier(n_estimators=118)
[116]: ada_preds = ada_model.predict(X_test)
[117]: print(classification_report(y_test, ada_preds))
```

	precision	recall	f1-score	support
No	0.88	0.92	0.90	557
Yes	0.63	0.52	0.57	147
accuracy			0.84	704
macro avg	0.76	0.72	0.74	704
weighted avg	0.83	0.84	0.83	704

Оценим важность признаков. Посмотрим какие из признаков оказывают наибольшее влияние на целевую переменную в данной модели.

```
[121]: ada_model.feature_importances_
[121]: array([0.00847458, 0.16949153, 0.22033898, 0.43220339, 0.
, 0.
, 0.01694915, 0.
, 0.01694915,
0.02542373, 0.00847458, 0.
, 0.01694915, 0.
,
0.
, 0.
, 0.
, 0.
, 0.00847458,
0.
, 0.00847458, 0.
, 0.00847458, 0.00847458,
0.01694915, 0.00847458, 0.
, 0.01694915, 0.00847458,
0.
, 0.
, 0.
])
[122]: ada_model.feature_importances_.argmax()
[122]: 3
[123]: X.columns[3]
[123]: 'TotalCharges'
```

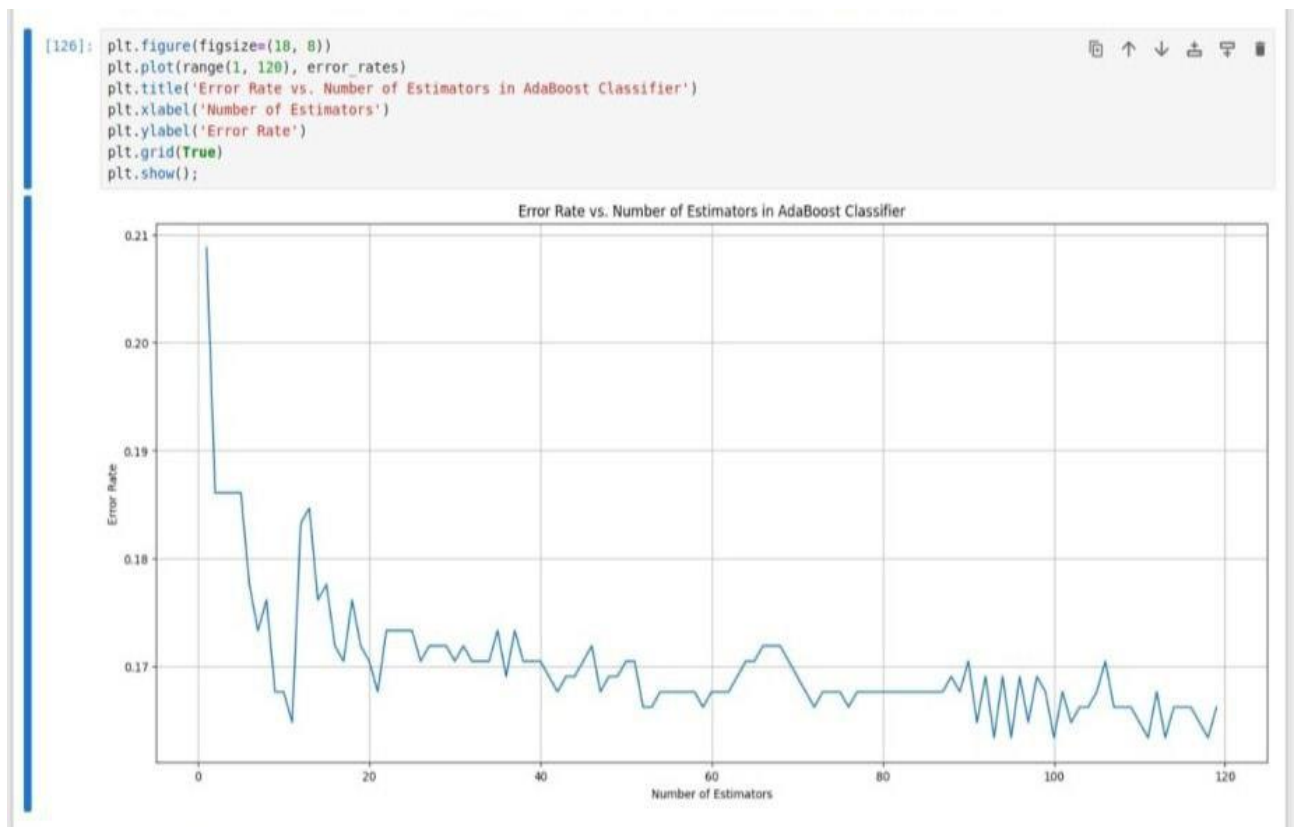
Получили, что в модели AdaBoost наиболее важное значение имеет признак TotalCharges. И это справедливо, так как показывает сколько всего расходов понёс клиент в кампании TELCO, и также данный признак достаточно хорошо коррелирует с целевой переменной Churn. Поэтому можно сделать вывод, что суммарные расходы клиента являются важным показателем для прогнозирования оттока клиентов.

Оптимизируем модель.

```
[124]: from sklearn.metrics import accuracy_score
error_rates = []
for n in range(1, 120):
    best_ada_model = AdaBoostClassifier(n_estimators=n)
    best_ada_model.fit(X_train, y_train)
    best_ada_preds = best_ada_model.predict(X_test)
    err = 1 - accuracy_score(y_test, best_ada_preds)
    error_rates.append(err)
```

При увеличении количества деревьев ошибка модели должна уменьшаться, так как увеличение количества деревьев в модели может помочь уменьшить ошибку на тренировочном наборе данных, что может привести к уменьшению ошибки на тестовом наборе данных. Однако, если количество деревьев становится слишком большим, модель может начать переобучаться и ошибаться на новых данных. Поэтому всегда важно подбирать оптимальное количество деревьев в модели.

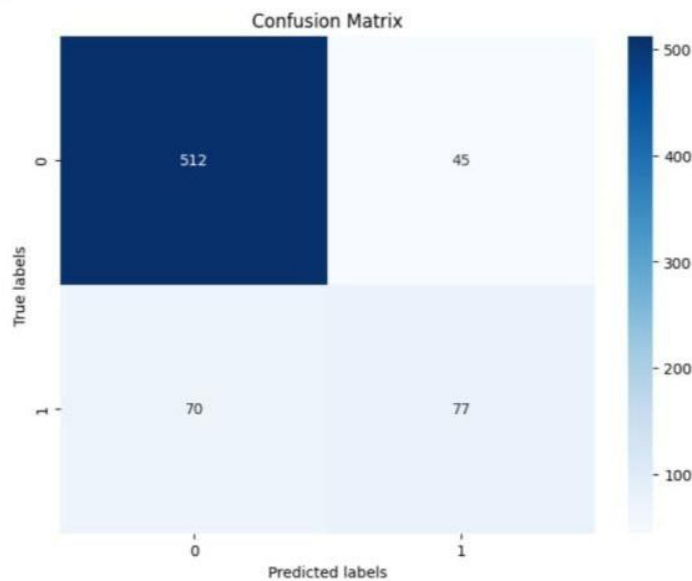
Визуализируем наши данные и посмотрим на результат.



По графику минимальное значение ошибки находится где-то в районе 118. Это значение мы и указали при обучении модели выше. Это значение ошибки является оптимальным для данной модели и набора данных, так как дальнейшее уменьшение ошибки может привести к переобучению модели и ухудшению ее предсказательной способности на новых данных. Поэтому важно найти баланс между снижением ошибки и предотвращением переобучения при обучении модели.

Построим матрицу ошибок для данной модели.

```
[127]: conf_matrix = confusion_matrix(y_test, ada_preds)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show();
```



Мы уменьшили значение метрики recall. И это значение равно 70. Это наилучшее значение из всех, что мы получали ранее на других моделях. Пока данная модель имеет лучшую предсказательную способность.

Создадим модель Gradient Boosting с применением гиперпараметров, оценим её работу и выполним обучение.


```
[69]: from sklearn.ensemble import GradientBoostingClassifier
      from sklearn.model_selection import GridSearchCV

[70]: param_grid = {'n_estimators': [50, 60, 70, 80, 90, 100, 110, 120, 130],
                  'learning_rate': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9],
                  'max_depth': [3, 4, 5, 6, 7, 8, 9]}

[71]: gb_model = GradientBoostingClassifier()

[72]: grid = GridSearchCV(estimator=gb_model, param_grid=param_grid)

[73]: grid.fit(X_train, y_train)
```

```
[73]: > GridSearchCV
      > estimator: GradientBoostingClassifier
          > GradientBoostingClassifier
              GradientBoostingClassifier()
```

```
[79]: gb_preds = grid.predict(X_test)
```

```
[80]: grid.best_estimator_
```

```
[80]: > GradientBoostingClassifier
      GradientBoostingClassifier(n_estimators=80)
```

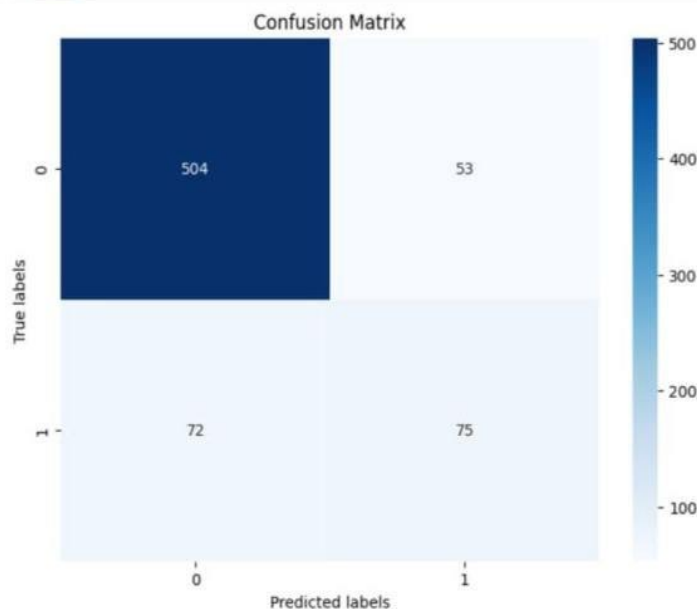
```
[81]: grid.best_params_
```

```
[81]: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 80}
```

```
[82]: print(classification_report(y_test, gb_preds))
```

	precision	recall	f1-score	support
No	0.88	0.90	0.89	557
Yes	0.59	0.51	0.55	147
accuracy			0.82	704
macro avg	0.73	0.71	0.72	704
weighted avg	0.81	0.82	0.82	704

```
[83]: conf_matrix = confusion_matrix(y_test, gb_preds)
      plt.figure(figsize=(8, 6))
      sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
      plt.xlabel('Predicted labels')
      plt.ylabel('True labels')
      plt.title('Confusion Matrix')
      plt.show();
```



Следует отметить, что обучение данной модели проходит очень долго по сравнению с другими тремя моделями. Скорее это связано, что поиск оптимальных значений происходит по сетке (GridSearchCV), и в модели мы указали много входных данных в качестве гиперпараметров.

В итоге модель Gradient Boosting отработала чуть хуже, чем предыдущая модель.

3.5 Модель логистической регрессии

Модель логистической регрессии - это статистический метод, который используется для предсказания вероятности того, что наблюдение принадлежит к определенной категории, исходя из одного или нескольких предикторов.

Построим модель логистической регрессии и обучим её.

```
[83]: from sklearn.preprocessing import StandardScaler
      from sklearn.linear_model import LogisticRegressionCV

[84]: scaler = StandardScaler()

[85]: scaled_X_train = scaler.fit_transform(X_train)

[86]: scaled_X_test = scaler.transform(X_test)

[87]: log_model = LogisticRegressionCV()

[88]: log_model.fit(scaled_X_train, y_train)

[88]: LogisticRegressionCV()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

Посмотрим на оптимальные значения данной модели.

```
[89]: log_model.Cs_

[89]: array([1.00000000e-04, 7.74263683e-04, 5.99484250e-03, 4.64158883e-02,
          3.59381366e-01, 2.78255940e+00, 2.15443469e+01, 1.66810054e+02,
          1.29154967e+03, 1.00000000e+04])

[90]: log_model.C_#выбор наилучшего значения

[90]: array([0.04641589])

[91]: log_model.get_params()

[91]: {'Cs': 10,
      'class_weight': None,
      'cv': None,
      'dual': False,
      'fit_intercept': True,
      'intercept_scaling': 1.0,
      'l1_ratios': None,
      'max_iter': 100,
      'multi_class': 'auto',
      'n_jobs': None,
      'penalty': 'l2',
      'random_state': None,
      'refit': True,
      'scoring': None,
      'solver': 'lbfgs',
      'tol': 0.0001,
      'verbose': 0}
```

Посмотрим на коэффициенты модели.

```
[92]: log_model.coef_

[92]: array([[ 8.50978351e-02, -7.35111542e-01,  1.29221171e-01,
           -1.56351333e-02,  4.79987953e-04, -1.50747632e-02,
           -4.43580136e-02, -8.03040647e-02,  8.03040647e-02,
           1.30196342e-01,  3.91037246e-01, -4.17105411e-02,
           -4.17105411e-02, -1.79349219e-01, -4.17105411e-02,
           -6.87981541e-02, -4.17105411e-02, -4.86559668e-04,
           -4.17105411e-02, -1.56171995e-01, -4.17105411e-02,
           1.17666197e-01, -4.17105411e-02,  9.53287254e-02,
           -2.96495051e-01, -5.95550305e-01,  1.66288540e-01,
           -4.32903180e-02,  1.47591543e-01, -2.23400002e-02,
           -1.67639596e-01, -1.51422564e-01, -1.83869135e-02]])

[93]: coefs = pd.Series(index=X.columns, data=log_model.coef_[0]).sort_values()
      coefs

[93]: tenure                                -0.735112
      Contract_Two year                    -0.595550
      Contract_One year                    -0.296495
      OnlineSecurity_Yes                   -0.179349
      Tenure Cohort 12-24 месяцев          -0.167640
      TechSupport_Yes                      -0.156172
      Tenure Cohort 24-48 месяцев          -0.151423
      PhoneService_Yes                     -0.080304
      OnlineBackup_Yes                     -0.068798
      Dependents_Yes                       -0.044358
      PaymentMethod_Credit card (automatic) -0.043290
      DeviceProtection_No internet service -0.041711
      OnlineBackup_No internet service      -0.041711
      StreamingMovies_No internet service   -0.041711
      InternetService_No                   -0.041711
      StreamingTV_No internet service       -0.041711
      OnlineSecurity_No internet service    -0.041711
      TechSupport_No internet service       -0.041711
      PaymentMethod_Mailed check           -0.022340
      Tenure Cohort_Более 48 месяцев        -0.018387
      TotalCharges                         -0.015635
      Partner_Yes                          -0.015075
      DeviceProtection_Yes                  -0.000487
      gender_Male                           0.000480
      MultipleLines_No phone service        0.000304
      SeniorCitizen                         0.005098
      StreamingMovies_Yes                   0.005329
      StreamingTV_Yes                       0.117666
      MonthlyCharges                       0.129221
      MultipleLines_Yes                     0.130196
      PaymentMethod_Electronic check        0.147592
      PaperlessBilling_Yes                  0.166289
      InternetService_Fiber optic           0.391037
      dtype: float64

[94]: y_pred = log_model.predict(scaled_X_test)
```

0 1 Python 3 (ipykernel) | Idle

Выполним предсказание модели и построим отчёт классификации.

```
[94]: y_pred = log_model.predict(scaleed_X_test)
```

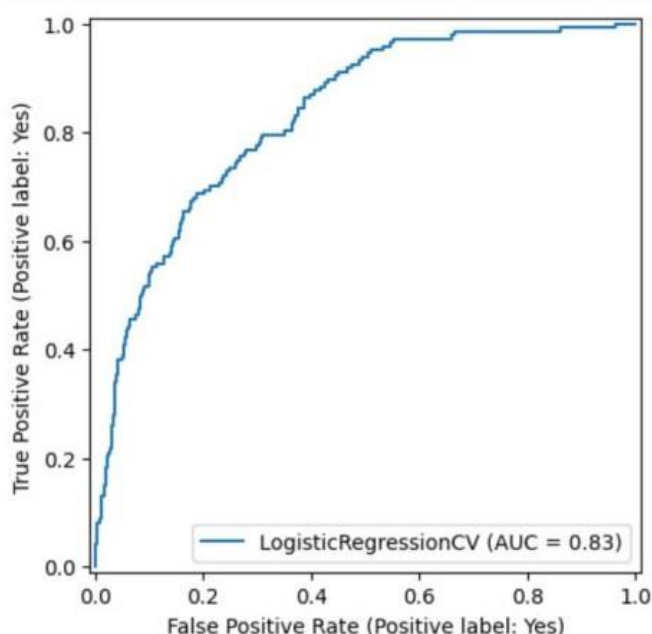
```
[95]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
No	0.87	0.91	0.89	557
Yes	0.61	0.50	0.55	147
accuracy			0.83	704
macro avg	0.74	0.71	0.72	704
weighted avg	0.82	0.83	0.82	704

Построим график кривой логистической регрессии (ROC CURVE)

```
[96]: from sklearn.metrics import roc_curve, RocCurveDisplay
```

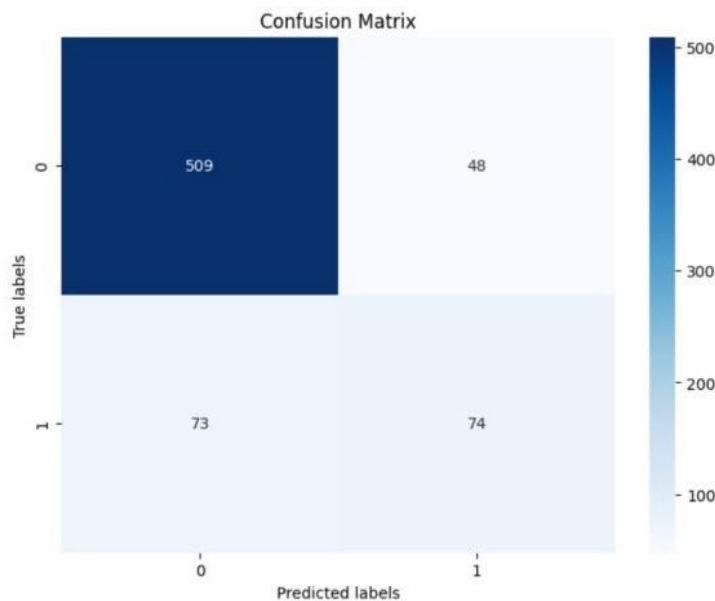
```
[97]: RocCurveDisplay.from_estimator(log_model, scaleed_X_test, y_test);
```



LogisticRegressionCV (AUC=0.83) в нашем случае означает, что была применена модель логистической регрессии с кросс-валидацией, и значение площади под кривой ROC (AUC) для данной модели составляет 0.83. AUC является метрикой, которая измеряет качество классификационной модели, где значение ближе к 1 указывает на более точную модель. Таким образом, значение AUC=0.83 указывает на хорошее качество модели логистической регрессии.

Построим матрицу ошибок и посмотрим на значение метрик.

```
[98]: conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show();
```



Видно что логистическая регрессия немного хуже отрабатывает, чем модель AdaBoost. Метрика recall равна 73. Ранее наилучшее значение мы получали 70.

3.6 Метод опорных векторов (SVM)

Метод опорных векторов (SVM) — это алгоритм машинного обучения, используемый для классификации и регрессии. Он особенно эффективен для задач с высокой размерностью и малым количеством обучающих данных.

Выполним обучение модели.

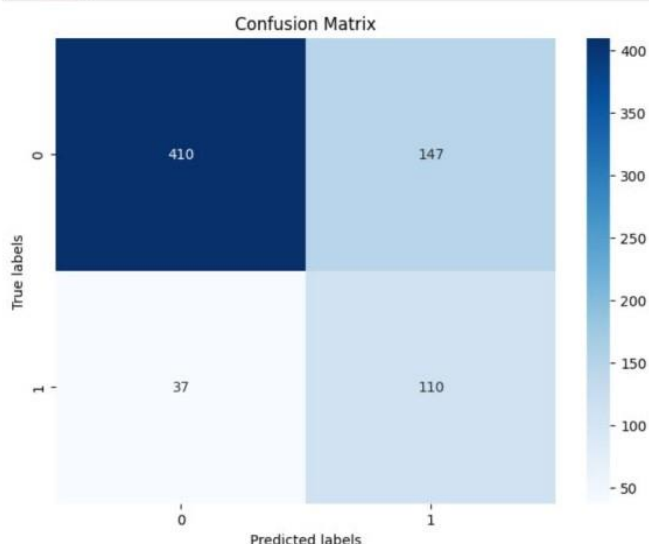
```
[99]: from sklearn.svm import SVC
[100]: svc = SVC(class_weight='balanced')
[101]: param_grid = {'C': [0.001, 0.01, 0.1, 0.5, 1]}
[102]: grid = GridSearchCV(svc, param_grid)
[103]: grid.fit(scaled_X_train, y_train)
[103]: GridSearchCV(estimator=SVC(class_weight='balanced'),
                  param_grid={'C': [0.001, 0.01, 0.1, 0.5, 1]})
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

Определим лучшие параметры данной модели.

```
[104]: grid.best_params_  
[104]: {'C': 1}  
[105]: grid_preds = grid.predict(scaled_X_test)  
[106]: confusion_matrix(y_test, grid_preds)  
[106]: array([[410, 147],  
            [ 37, 110]])
```

Построим матрицу ошибок и сделаем вывод.

```
[109]: conf_matrix = confusion_matrix(y_test, grid_preds)  
plt.figure(figsize=(8, 6))  
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')  
plt.xlabel('Predicted labels')  
plt.ylabel('True labels')  
plt.title('Confusion Matrix')  
plt.show();
```



Мы получили наилучшее значение recall равной 37. Уменьшили ошибку предсказания для этой метрики почти в 2 раза, по сравнению с предыдущими моделями. Предсказание данной модели в целях решения нашей задачи является пока лучшей.

3.7 Метод К-ближайших соседей (KNN)

Метод К-ближайших соседей (KNN) — это алгоритм машинного обучения, который используется для классификации или регрессии. Он основан на идее, что точки данных, которые расположены близко друг к другу в пространстве признаков, скорее всего, относятся к одному и тому же классу.

Построим данную модель для наших данных.

Метод К-ближайших соседей(KNN)

```
[110]: from sklearn.neighbors import KNeighborsClassifier
[111]: knn = KNeighborsClassifier()
[114]: operations = [('scaler', scaler), ('knn', knn)]
[115]: pipe = Pipeline(operations)
[116]: k_values = list(range(1, 30))
[117]: param_grid = {'knn_n_neighbors': k_values}
[129]: full_cv_classifier = GridSearchCV(pipe, param_grid, cv=5, scoring='accuracy')
[130]: full_cv_classifier.fit(X_train, y_train)
[130]: GridSearchCV(cv=5,
                  estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                             ('knn', KNeighborsClassifier())]),
                  param_grid={'knn_n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                                  12, 13, 14, 15, 16, 17, 18, 19,
                                                  20, 21, 22, 23, 24, 25, 26, 27,
                                                  28, 29]},
                  scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Выполним обучение модели и построим матрицу ошибок.

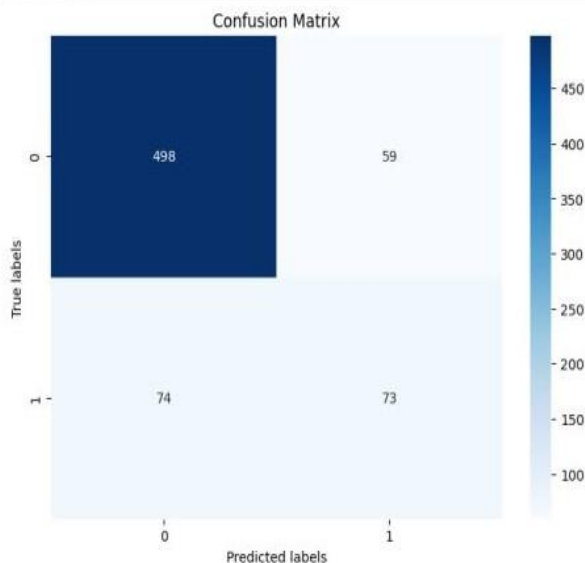
```
[137]: knn_pred = full_cv_classifier.predict(X_test)
[138]: print(classification_report(y_test, knn_pred))

              precision    recall  f1-score   support

     No         0.87         0.89         0.88         557
     Yes         0.55         0.50         0.52         147

 accuracy         0.81         0.81         0.81         704
 macro avg         0.71         0.70         0.70         704
 weighted avg         0.80         0.81         0.81         704

[139]: conf_matrix = confusion_matrix(y_test, knn_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show();
```



Значение recall (равное 74) получилось достаточно большим. Это хуже, чем у многих ранее рассмотренных моделей.

3.8 Общий вывод

Если сделаем вывод, то можно сказать, что модель метод опорных векторов (SVM) с использованием гиперпараметров работает лучше остальных моделей. Значение метрики recall у неё наименьшее чем у других моделей. И соответственно, предсказательная способность у неё намного лучше. Эту модель можно использовать в нашей задаче, ведь у неё и ошибка предсказания меньше. И наша модель готова. Её можно использовать в дальнейшем (при добавлении и появлении новых данных в датасете, модель нужно будет корректировать).

Было выяснено, что наибольшее влияние на отток клиентов влияет показатель TotalCharges (общие расходы), и больше оттока наблюдается на краткосрочных контрактах. В связи с этим, компании TELCO рекомендуется пересмотреть условия краткосрочных контрактов (меньше года), чтобы удержать абонентов у себя. Исходя из этого, следует разработать некоторые мероприятия, которые удовлетворяли бы клиентов на контрактах меньше года.

3.9 Общие рекомендации по снижению оттока в компании TELCO

Дадим некоторые рекомендации как телекоммуникационная компания TELCO могла бы значительно уменьшить отток клиентов, особенно на краткосрочных контрактах.

1. Анализ и Сегментация:

Создание профилей клиентов: разработка детального профиля каждого клиента, включающего демографические данные, историю использования услуг, уровень удовлетворенности, размер трат, и т.д.

Сегментация клиентов: разделение клиентов на группы с похожими характеристиками, чтобы лучше понимать их потребности и предлагать персонализированные решения.

Идентификация факторов риска оттока: анализ данных, чтобы выявить факторы, которые чаще всего приводят к оттоку клиентов (например, высокая стоимость услуг, плохой интернет, неудобные условия договора).

2. Программы Лояльности и Удержания:

Специальные предложения для краткосрочных клиентов: разработка индивидуальных предложений, учитывающих потребности и предпочтения клиентов с краткосрочными контрактами.

Бонусные программы: введение бонусных систем за продолжительность использования услуг, активное потребление и рекомендации.

Программа "Happy Customer": создание программы обратной связи, где клиенты могут делиться своим опытом и получать вознаграждение за активное участие.

Персонализированные предложения: использование данных клиента для предложения индивидуальных пакетов услуг, скидок и акций.

3. Улучшение Качества Обслуживания:

Оптимизация call-центра: улучшение качества обслуживания клиентов по телефону, сокращение времени ожидания, повышение уровня компетентности сотрудников.

Онлайн-чат и поддержка: введение удобного онлайн-чата или бота для быстрого решения проблем и ответов на вопросы.

Проактивный подход к решению проблем: связь с клиентами, испытывающими трудности, с предложениями решений и помощью.

Удобство использования сайта и приложения: оптимизация веб-сайта и мобильного приложения для удобного управления услугами, оплаты счетов, получения информации.

4. Упрощение Договорных Отношений:

Прозрачные условия контракта: четкая формулировка условий договора, без скрытых платежей и ограничений.

Гибкие условия: предоставление возможности продления контракта на более длительный период, возможность смены тарифа и услуг без лишних сложностей.

Автоматизация процесса продления: создание системы автоматического продления контракта, чтобы избежать его случайной отмены.

5. Повышение Значимости Бренда:

Укрепление позиционирования бренда: создание положительного имиджа компании, акцентируя внимание на ценностях и преимуществах услуг.

Повышение лояльности сотрудников: мотивирование сотрудников, обеспечение им возможности профессионального развития, создание комфортных условий работы.

Создание сообщества: взаимодействие с клиентами через социальные сети, создание форумов и групп для общения.

Дополнительные советы:

Анализ конкурентов: изучение конкурентной среды, выявление лучших практик и адаптация их к специфике TELCO.

Тестирование и оптимизация: постоянный анализ результатов и внесение корректировок в бизнес-процессы для повышения эффективности.

Инвестирование в технологии: использование CRM-систем, аналитических инструментов и других технологий для автоматизации процессов, улучшения качества обслуживания и персонализации взаимодействия с клиентами.

Важно: разработка успешных бизнес-процессов требует комплексного подхода и постоянного мониторинга результатов. Необходимо адаптировать решения к специфике компании, ее клиентов и рыночных условий.

Заключение

В данном исследовании было проведено прогнозирование оттока клиентов в телекоммуникационной компании TELCO с использованием алгоритмов машинного обучения. Целью исследования было выявить факторы, влияющие на отток клиентов, и разработать рекомендации по их удержанию.

Для достижения этой цели был проведен анализ данных клиентов, включающий информацию о демографических характеристиках, услугах, использовании ресурсов и удовлетворенности клиентов. Затем были применены различные алгоритмы машинного обучения, такие как модели на основе деревьев, логистическая регрессия, метод опорных векторов (SVM), метод К-ближайших соседей (KNN) для построения моделей прогнозирования оттока.

Результаты исследования показали, что на отток клиентов влияют несколько ключевых факторов. Одним из них является тип услуги, которую клиент использует. Например, клиенты, использующие только интернет-услуги, имеют более высокую вероятность оттока. Кроме того, длительность пользования услугами компании и уровень удовлетворенности клиентов также оказывают значительное влияние на отток. Наибольший отток наблюдался на краткосрочных контрактах.

На основе этих результатов были составлены рекомендации по удержанию клиентов. Во-первых, компания должна обратить особое внимание на клиентов, использующих только интернет-услуги, и предложить им дополнительные услуги, чтобы повысить их удовлетворенность и уменьшить вероятность оттока. Во-вторых, компания должна проводить регулярные опросы клиентов для выявления и устранения возможных проблем и недовольств. Кроме того, компания может предлагать персонализированные скидки и бонусы клиентам, которые пользуются услугами компании в течение длительного времени, чтобы стимулировать их оставаться.

Таким образом, применение алгоритмов машинного обучения для прогнозирования оттока клиентов и составление рекомендаций по их удержанию может значительно помочь телекоммуникационным компаниям, таким как

TELCO, в улучшении своей стратегии удержания клиентов и повышении своей конкурентоспособности на рынке.

Список используемой литературы

1. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. – М.: ДМК Пресс, 2015
2. Саттон Р.С., Барто Э.Дж. Обучение с подкреплением: Введение. 2-е изд. – М.: ДМК Пресс, 2020
3. Шалев-Шварц Ш., Бен-Давид Ш. Идеи машинного обучения: от теории к алгоритмам. – М.: ДМК Пресс, 2019.
4. Agrawal R. et al. Fast discovery of association rules // Advances in Knowledge Discovery and Data Mining. AAAI Press, 1996. P. 307–328.
5. Han J. et al. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach // Data Mining and Knowledge Discovery. 2004. Vol. 8, № 1. P. 53–87.
6. Tipping M. The Relevance Vector Machine // Advances in Neural Information Processing Systems / ed. Solla S., Leen T., Müller K. MIT Press, 2000. Vol. 12.
7. Метод релевантных векторов [Электронный ресурс]. – Режим доступа: <http://www.machinelearning.ru/wiki/index.php?title=RVM>
8. Quinlan J.R. C4.5: programs for machine learning. San Mateo, Calif: Morgan Kaufmann Publishers, 1993. 302 p.
9. Breiman L. et al. Classification and Regression Trees. Wadsworth, Belmont, California, 1984
10. Rabiner L.R. A tutorial on hidden Markov models and selected applications in speech recognition // Proc. IEEE. 1989. Vol. 77, № 2. P. 257–286.
11. Документация Python <https://docs.python.org/3.10/>
12. Документация Pandas <https://pandas.pydata.org/docs/>
13. Документация Python <https://scikit-learn.ru/>

Приложение 1. Содержание файла Telco-Customer-Churn
<https://drive.google.com/file/d/1WKkCoQfKsHVzfWs158FIQ0f-ZnlVIEpL/view?usp=sharing>

Приложение 2. Содержание файла Telco.ipynb
<https://drive.google.com/file/d/1buAMu48yFS-YSbt5--DateSryQJ2zEPS/view?usp=sharing>