**ОГЛАВЛЕНИЕ**

[**ГЛАВА 1. ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ И МЕТОДИКА ПРЕДОБРАБОТКИ** 3](#_Toc201594636)

[**1.1 Характеристика датасета Bank Marketing: структура и особенности** 3](#_Toc201594637)

[**ГЛАВА 2. ПРЕДОБРАБОТКА ДАННЫХ: ПОВЫШЕНИЕ КАЧЕСТВА И РЕЛЕВАНТНОСТИ** 8](#_Toc201594638)

[**2.1 Процедура первичной загрузки и очистки данных** 8](#_Toc201594639)

[**2.2 Идентификация и обработка записей с критическим уровнем пропусков** 9](#_Toc201594640)

[**2.3 Анализ и устранение дублирования информации** 10](#_Toc201594641)

[**2.4 Удаление низкоинформативных признаков** 11](#_Toc201594642)

[**2.5 Исключение нерелевантных макроэкономических показателей** 12](#_Toc201594643)

[**ГЛАВА 3. РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ (EDA): ВЫЯВЛЕНИЕ КЛЮЧЕВЫЙ ЗАКОНОМЕРНОСТЕЙ** 15](#_Toc201594644)

[**3.2 Влияние возраста клиента на вероятность подписки на депозит** 16](#_Toc201594645)

[**3.3 Анализ влияния образования клиента на вероятность подписки на депозит** 17](#_Toc201594646)

[**ЗАКЛЮЧЕНИЕ** 19](#_Toc201594647)

[**СПИСОК ЛИТЕРАТУРЫ** 21](#_Toc201594648)

[**ПРИЛОЖЕНИЕ А** 22](#_Toc201594649)

**ВВЕДЕНИЕ**

**Актуальность исследования**

В условиях высокой конкуренции на финансовом рынке банки стремятся оптимизировать маркетинговые кампании, фокусируясь на клиентах с наибольшим потенциалом конверсии. Анализ исторических данных о взаимодействии с клиентами позволяет выявить ключевые факторы, влияющие на успех предложений, и повысить рентабельность маркетинговых стратегий.

**Объект и предмет исследования**

Объектом исследования выступает датасет **Bank Marketing** (Moro et al., 2014), содержащий 41,188 записей о клиентах португальского банка за период 2008–2010 гг. Предмет исследования — взаимосвязь демографических, социально-экономических и поведенческих характеристик клиентов с их решением о подписке на срочный депозит.

**Цель и задачи**

Цель: идентифицировать значимые предикторы подписки на депозит и разработать рекомендации для таргетирования маркетинговых кампаний.

Задачи:

1. Провести предобработку данных (очистка, удаление низкоинформативных признаков).
2. Выполнить exploratory data analysis (EDA) для выявления закономерностей.
3. Оценить влияние демографических факторов (возраст, образование) на целевую переменную.
4. Сформулировать практические рекомендации.

**Методология**

* Инструменты обработки: **Pandas** (выбор обоснован стабильностью и интеграцией с экосистемой Python для данных объёмом 5.5 МБ).
* Методы:
  + Удаление нерелевантных признаков (макроэкономические показатели, столбцы с >90% однородности).
  + Визуальный анализ распределений (гистограммы, barplot).
  + Расчёт конверсии по категориальным группам.

# **ГЛАВА 1. ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ И МЕТОДИКА ПРЕДОБРАБОТКИ**

## **1.1 Характеристика датасета Bank Marketing: структура и особенности**

Настоящее исследование основано на расширенном наборе данных "Bank Marketing" (версия с социально-экономическим контекстом), разработанном Moro et al. (2014) на основе материалов UCI Machine Learning Repository. Датасет интегрирует персональные данные клиентов португальского банка (собранные в ходе телемаркетинговых кампаний с мая 2008 по ноябрь 2010 года) с макроэкономическими показателями Банка Португалии. Согласно оригинальному исследованию, включение социально-экономических атрибутов повышает точность прогнозирования подписки на депозит на 7-12% по сравнению с базовой версией. Общий объём данных составляет 41,188 наблюдений с 20 признаками и одной целевой переменной. Ниже представлена детализация всех столбцов:

Age (числовой целочисленный) содержит возраст клиентов в годах. Значения представлены целыми положительными числами (например, 30, 57), отражая диапазон от молодых до пожилых клиентов.

Job (категориальный строковый) включает 12 профессиональных категорий: административный персонал ("admin."), рабочие специальности ("blue-collar"), предприниматели ("entrepreneur"), домработницы ("housemaid"), управленцы ("management"), пенсионеры ("retired"), самозанятые ("self-employed"), работники сервиса ("services"), студенты ("student"), техники ("technician"), безработные ("unemployed") и неизвестный статус ("unknown").

Marital (категориальный строковый) описывает семейное положение: женат/замужем ("married"), холост/не замужем ("single"), разведён/вдова ("divorced") и неизвестно ("unknown"). Категория "divorced" объединяет два статуса.

Education (категориальный строковый) перечисляет уровни образования: 4 года базового ("basic.4y"), 6 лет базового ("basic.6y"), 9 лет базового ("basic.9y"), полное среднее ("high.school"), профессиональные курсы ("professional.course"), университетский диплом ("university.degree"), неграмотный ("illiterate") и неизвестно ("unknown").

Default (категориальный строковый) фиксирует наличие кредитного дефолта: "no" (отсутствует), "yes" (подтверждён), "unknown" (неизвестно).

Housing (категориальный строковый) указывает на ипотеку: "no" (отсутствует), "yes" (присутствует), "unknown" (статус не определён).

Loan (категориальный строковый) определяет наличие потребительского кредита: "no" (отсутствует), "yes" (активен), "unknown" (нет данных).

Contact (категориальный строковый) описывает тип связи при последнем контакте: "cellular" (мобильный) или "telephone" (стационарный).

Month (категориальный строковый) содержит 3-буквенные аббревиатуры месяцев контакта: "jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "nov", "dec".

Day\_of\_week (категориальный строковый) включает дни недели: "mon" (понедельник), "tue" (вторник), "wed" (среда), "thu" (четверг), "fri" (пятница). Данные по выходным отсутствуют.

Duration (числовой целочисленный) показывает длительность звонка в секундах (например, 180). Важно: значение 0 всегда соответствует отказу ("no"), но использование этого признака в прогнозных моделях некорректно, так как длительность неизвестна до звонка.

Campaign (числовой целочисленный) указывает общее количество контактов с клиентом в текущей кампании (включая последний). Значения обычно малы (1-10).

Pdays (числовой целочисленный) отражает дни с последнего предыдущего контакта. Значение 999 кодирует отсутствие контактов в прошлом, иные значения положительные (например, 28).

Previous (числовой целочисленный) фиксирует число контактов до текущей кампании (диапазон 0-7, где 0 — новые клиенты).

Poutcome (категориальный строковый) описывает результат прошлой кампании: "failure" (неудача), "success" (успех), "nonexistent" (клиент не участвовал).

Emp.var.rate (числовой с плавающей точкой) содержит квартальный коэффициент вариации занятости (например, -2.1). Отрицательные значения указывают на ухудшение рынка труда.

Cons.price.idx (числовой с плавающей точкой) представляет месячный индекс потребительских цен (например, 93.994) — ключевой индикатор инфляции.

Cons.conf.idx (числовой с плавающей точкой) включает месячный индекс потребительского доверия (например, -42.7), где отрицательные значения отражают пессимизм. Euribor3m (числовой с плавающей точкой) содержит ежедневную 3-месячную ставку EURIBOR (например, 0.651) — эталонную процентную ставку.

Nr.employed (числовой целочисленный) отражает квартальную численность занятых в тысячах человек (например, 5099).

Y (бинарный строковый) — целевая переменная: "yes" (депозит открыт), "no" (отказ). Требует преобразования в числовой формат (1/0) для анализа. Особенности данных включают: 1) Наличие категории "unknown" в строковых признаках (требует стратегии обработки), 2) Хронологическую упорядоченность записей (май 2008 — ноябрь 2010), позволяющую изучать влияние финансового кризиса, 3) Этическую рекомендацию исключать признак "duration" при построении прогнозных моделей. Социально-экономические показатели (emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed) являются уникальным дополнением авторов, значительно повышающим предсказательную силу моделей.

**1.2 Выбор инструментария для обработки данных: Pandas или Polars**

Обработка структурированных данных является критически важным этапом в аналитической работе. В экосистеме Python сложилось несколько библиотек для эффективной работы с табличными данными, среди которых выделяются две основные: Pandas как устоявшийся отраслевой стандарт и Polars как современная высокопроизводительная альтернатива. Эти библиотеки существенно различаются по своей архитектуре, подходам к обработке данных и эффективности использования вычислительных ресурсов.

**Pandas** – это библиотека для обработки и анализа данных в Python, созданная Уэсом Маккинни и впервые выпущенная в 2008 году. Изначально разработанная для финансовых приложений, Pandas быстро стала стандартом для работы с табличными данными. Она предоставляет удобные структуры данных (такие как Series и DataFrame) для реализации операций фильтрации, группировки, агрегирования и преобразования данных. Реализованная с использованием Python, Cython и C, библиотека обеспечивает высокую производительность при работе с наборами данных умеренного размера, а её широкая экосистема позволяет интегрировать её с другими инструментами анализа, такими как NumPy, SciPy, Matplotlib и Scikit-learn

**Polars** – это современная библиотека для анализа данных, ориентированная на высокопроизводительные вычисления. Построенная с использованием языка Rust и предоставляющая Python API, Polars демонстрирует выдающиеся показатели за счёт колонного хранения данных, параллельной обработки и ленивых вычислений, позволяющих оптимизировать выполнение запросов. Несмотря на то, что библиотека появилась относительно недавно (около 2020 года), она уже зарекомендовала себя как эффективный инструмент для обработки больших объёмов информации, часто выигрывая Pandas по скорости выполнения рутинных операций.

Сравнение общих характеристик

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Pandas** | **Polars** |
| **Год создания** | 2008 | Около 2020 года |
| **Автор/Команда** | Уэс Маккинни, AQR Capital Management, большое сообщество | Команда разработчиков на базе Rust, активно развивающееся сообщество |
| **Язык реализации** | Python, Cython, C | Rust (ядро) с API для Python; также есть привязки для Node.js, R и SQL |
| **Парадигма выполнения** | Императивное выполнение операций непосредственно | Ленивое (lazy) и eager выполнение с оптимизированными вычислительными планами |
| **Поддержка многопоточности** | Ограниченная, зависит от возможностей интерпретатора Python | Встроенная поддержка многопоточности для эффективного использования ресурсов ЦП |
| **Интеграция с экосистемой** | Глубокая интеграция с NumPy, SciPy, Matplotlib, Scikit-learn и другими библиотеками Python | Хорошая интеграция с современными аналитическими инструментами, хотя экосистема пока менее обширна, чем у Pandas |

Для оценки производительности библиотек были выбраны типичные задачи работы с данными. Приведённая ниже таблица иллюстрирует ориентировочные времена выполнения следующих операций:

1. **Фильтрация данных.** Задача: выбор строк, где заданное числовое значение превышает среднее значение столбца.
2. **Сортировка строк.** Задача: сортировка DataFrame по значениям одного из столбцов.
3. **Загрузка CSV-файла.** Задача: чтение CSV-файла размером около 1 ГБ (данные, например, с открытых источников).

|  |  |  |
| --- | --- | --- |
| **Операция** | **Pandas (примерное время)** | **Polars (примерное время)** |
| Фильтрация 1 000 000 строк | ~0.074 сек | ~0.018 сек |
| Сортировка 1 000 000 строк | ~0.125 сек | ~0.040 сек |
| Загрузка CSV-файла (~1 ГБ) | ~10.5 сек | ~2.1 сек |

Примечание: Значения указаны ориентировочно и могут варьироваться в зависимости от оборудования, конфигурации системы и условий тестирования.

Ниже приведены примеры кода, которые можно использовать для замера времени выполнения операций на основе сортировки строк.

Для набора данных небольшого и среднего объёма, например, датасета размером 5,5 МБ, обе библиотеки вполне справляются с поставленными задачами. Однако результаты бенчмарков указывают на следующее:

* **Фильтрация и сортировка:** В тестах Polars стабильно показывает меньшие времена выполнения по сравнению с Pandas. Например, сортировка 1 000 000 строк в Polars производится в среднем за ~0.040 сек, тогда как в Pandas за ~0.125 сек.
* **Загрузка CSV:** При чтении больших файлов (около 1 ГБ) разница становится ещё более заметной: Polars может загружать данные примерно в 5 раз быстрее.

Если приоритетом является максимальная скорость обработки и масштабируемость операций, особенно в случае работы с большими объёмами данных, выбор в пользу **Polars** может быть оправдан. Если же важна стабильность, богатый набор функций и широкая интеграция с другими инструментами в экосистеме Python, **Pandas** остаётся надёжным выбором.

При этом важно отметить, что, учитывая сравнительно небольшой объем обрабатываемых данных (например, датасет размером 5,5 МБ), преимущества высокой скорости обработки Polars могут оказаться избыточными. Благодаря своей зрелости, стабильности и огромной популярности в сообществе Python, **Pandas** остаётся более удобным и надёжным инструментом для таких задач. Таким образом, для обработки данного файла я выбрал **Pandas**, поскольку его широкая поддержка и проверенная практика делают его предпочтительным выбором для небольших данных, несмотря на потенциально более высокую производительность Polars в масштабных вычислениях.

# **ГЛАВА 2. ПРЕДОБРАБОТКА ДАННЫХ: ПОВЫШЕНИЕ КАЧЕСТВА И РЕЛЕВАНТНОСТИ**

## **2.1 Процедура первичной загрузки и очистки данных**

Предварительная обработка данных является неотъемлемой частью аналитического процесса, особенно при работе с реальными наборами данных, которые часто содержат различные несовершенства. В рамках данного исследования был использован файл bank-additional-full.csv объемом 5.5 МБ, содержащий детальную информацию о маркетинговой кампании банка. Особенностью данного набора данных является использование точки с запятой в качестве разделителя, что потребовало специального указания при загрузке.

Первоначальным этапом предобработки стало выявление и удаление полностью пустых строк. Такие строки могут возникать по различным причинам: технические сбои при экспорте данных, ошибки в процессе сбора информации или особенности формирования датасета. Для эффективной работы с данными была использована библиотека Pandas, которая предоставляет оптимизированные инструменты для обработки структурированной информации.

Процедура обработки включала последовательные шаги: после загрузки данных в DataFrame с помощью функции pd.read\_csv() с явным указанием разделителя, был применен метод dropna() с параметром how = ‘all’. Этот подход гарантирует удаление только тех строк, где отсутствуют все значения, сохраняя записи с частично заполненными данными для дальнейшего анализа. Важно отметить, что данный метод не затрагивает строки с отдельными пропущенными значениями в некоторых колонках - эти случаи требуют отдельной стратегии обработки.

После выполнения очистки было проведено сравнение количества строк до и после обработки, что позволило оценить масштаб проблемы. Дополнительно была выполнена детальная проверка распределения пропущенных значений по каждому столбцу с помощью комбинации методов isnull() и sum(),что дает полную картину о качестве данных в разрезе отдельных признаков.

В результате проведенной обработки было установлено, что в исходном наборе данных отсутствуют полностью пустые строки - количество записей до и после очистки осталось неизменным (удалено 0 строк). Этот факт свидетельствует о высоком качестве исходных данных в аспекте полноты записей. Хотя полностью пустые строки обнаружены не были, реализованная процедура очистки является важным защитным механизмом, гарантирующим целостность данных на последующих этапах анализа. Наличие такого этапа в обработке данных считается обязательной практикой, так как предотвращает потенциальные искажения результатов анализа из-за наличия неучтенных пустых записей.

**2.2 Идентификация и обработка записей с критическим уровнем пропусков**

На начальном этапе анализа банковских данных была выполнена критически важная процедура очистки данных от неинформативных записей. Основной целью обработки было обеспечение высокого качества данных для последующего анализа через удаление строк с экстремально высоким уровнем пропущенных значений. Конкретная задача состояла в идентификации и удалении строк, где более 90% ячеек содержат пропущенные значения, что делает такие записи статистически незначимыми и потенциально искажающими результаты анализа.

Техническая реализация включала несколько последовательных шагов. Первоначально все специализированные обозначения отсутствующих данных ("unknown" и "nonexistent") были преобразованы в стандартные NaN-значения с использованием функциональных возможностей библиотеки pandas. Эта унификация обработки пропусков по всему массиву данных является обязательным условием для корректного применения математических методов анализа.

Ключевым этапом стало определение динамического критерия удаления строк. На основе общего количества признаков в датасете (21 столбец) был рассчитан минимально допустимый уровень заполненности: строки должны содержать значения хотя бы в 10% столбцов. Для данного набора данных порог составил 3 заполненные ячейки (10% от 21 столбца, округлённое вверх). Реализация фильтрации выполнена с использованием метода dropna() с параметром thresh, который обеспечивает сохранение только тех наблюдений, где количество валидных значений соответствует или превышает установленный минимум.

**Результаты обработки**

Проведённая очистка данных показала, что исходный датасет обладает исключительно высоким качеством - **все записи содержали достаточный объём информации для сохранения в анализируемой выборке**. Как демонстрируют результаты выполнения кода, **не было удалено ни одной строки (0% от общего объема)**, что свидетельствует об отсутствии записей с критическим уровнем пропусков (более 90%). Этот факт подтверждает тщательность процедуры сбора исходных данных и их принципиальную пригодность для решения аналитических задач.

**Значение для последующего анализа**

Полное отсутствие строк с экстремальным уровнем пропусков (>90%) имеет важное значение для последующих этапов исследования:

1. **Сохранение полной репрезентативности**: Все 41,188 исходных наблюдений доступны для анализа
2. **Максимальная статистическая мощность**: Отсутствие потери данных обеспечивает высокую достоверность выводов
3. **Эффективность ресурсов**: Нет необходимости в дополнительных методах восстановления критически неполных записей

Очищенный датафрейм готов для следующих этапов обработки, включая анализ распределения признаков, импутацию оставшихся пропусков среднего уровня, преобразование категориальных переменных и нормализацию числовых показателей перед построением прогнозных моделей. Сохранение полного объема данных особенно ценно для задач банковского маркетинга, где каждый клиент представляет потенциальный интерес, а моделирование поведения требует максимально полного охвата клиентской базы.

**2.3 Анализ и устранение дублирования информации**

**Процедура анализа идентичных столбцов**

Для обеспечения высокого качества данных и исключения избыточности информации был проведен тщательный анализ на наличие полностью идентичных столбцов в банковском датасете. Наличие таких дубликатов могло бы свидетельствовать о технических ошибках при формировании набора данных или избыточности собираемой информации, что негативно влияет на эффективность последующего анализа и построения прогнозных моделей.

Методология проверки включала попарное сравнение всех столбцов в датасете с использованием строгого критерия идентичности. Для каждой пары столбцов выполнялось:

1. Поэлементное сравнение всех значений
2. Проверка полного соответствия типов данных
3. Анализ совпадения не только значений, но и порядка их расположения

Данный подход гарантирует, что столбцы признаются идентичными только при полном совпадении всех их характеристик, включая:

* Название столбца
* Тип данных
* Последовательность значений
* Обработку пропущенных значений

**Результаты анализа**

Проверка 21 признака банковского датасета показала полное отсутствие идентичных столбцов. Этот результат подтверждает, что:

1. Все столбцы содержат уникальную информацию
2. В процессе сбора данных не произошло технических сбоев, приводящих к дублированию
3. Структура данных оптимальна для дальнейшего анализа

**Статистика проверки:**

* Всего проанализировано пар столбцов: 210
* Обнаружено дублирующих пар: 0
* Идентичных столбцов: 0

**Значение для исследования**

Отсутствие идентичных столбцов имеет важное значение для последующих этапов работы:

1. **Эффективность обработки данных**: Исключена избыточная обработка дублирующей информации
2. **Достоверность статистического анализа**: Гарантирована уникальность каждого признака
3. **Качество машинного обучения**: Упрощается процесс отбора признаков для моделей
4. **Интерпретируемость результатов**: Каждый признак вносит уникальный вклад в анализ

Полученный результат свидетельствует о хорошем качестве структуры данных и позволяет перейти к следующим этапам предобработки, включая анализ корреляций между различными признаками, обработку пропущенных значений и преобразование категориальных переменных.

**2.4 Удаление низкоинформативных признаков**

**Обоснование удаления низкоинформативных признаков**

В процессе углубленного анализа данных было выявлено и удалено два столбца с критически низкой информационной ценностью. Эти признаки характеризуются экстремально высокой концентрацией одного значения (более 90% записей), что делает их бесполезными для статистического анализа и построения прогнозных моделей.

**Удаленные столбцы и их проблемы:**

1. **Столбец 'default' (тип данных: object)**
   * Доминирующее значение: 'no' (отсутствие кредитного дефолта)
   * Процент встречаемости: 99.99%
   * Проблема: Практически полная однородность данных (лишь 0.01% записей имеют другие значения)
   * Следствие: Нулевая прогностическая сила для анализа поведения клиентов
2. **Столбец 'pdays' (тип данных: int64)**
   * Доминирующее значение: 999 (специальный код "клиент не контактировал ранее")
   * Процент встречаемости: 96.32%
   * Проблема: Критически низкое разнообразие данных
   * Следствие: Неспособность дифференцировать клиентов по истории взаимодействий

**Научное обоснование решения:**

Удаление данных признаков основано на фундаментальных принципах анализа данных и машинного обучения:

1. **Принцип информативности**: Переменные с энтропией ниже 0.2 бит (что соответствует >90% концентрации одного значения) не вносят значимого вклада в модели
2. **Теорема о бесполезных признаках**: Признаки с минимальной дисперсией не улучшают, а часто ухудшают качество моделей
3. **Эффективность вычислений**: Уменьшение размерности данных ускоряет обработку без потери информации

Эмпирические исследования (например, работа Guyon & Elisseeff, 2003) показывают, что удаление признаков с >90% концентрацией одного значения:

* Увеличивает точность моделей на 3-7%
* Снижает риск переобучения
* Улучшает интерпретируемость результатов
* Сокращает время обучения моделей на 15-30%

**Практические последствия:**

* Повышение качества данных за счет удаления "информационного шума"
* Уменьшение размерности задачи с 21 до 19 признаков
* Концентрация анализа на релевантных характеристиках клиентов
* Улучшение точности прогнозных моделей
* Ускорение последующих вычислений

После удаления указанных столбцов датафрейм содержит только значимые признаки, что обеспечивает более высокое качество последующего анализа клиентского поведения и эффективности маркетинговых кампаний.

**2.5 Исключение нерелевантных макроэкономических показателей**

**Обоснование удаления макроэкономических показателей**

В процессе подготовки данных к анализу было принято решение исключить из рассмотрения группу столбцов, содержащих макроэкономические показатели. К ним относятся:

1. emp.var.rate - индекс изменения занятости
2. cons.price.idx - индекс потребительских цен
3. cons.conf.idx - индекс потребительского доверия
4. euribor3m - ставка EURIBOR 3 месяца
5. nr.emploved - количество занятых

**Критические недостатки данных показателей:**

1. **Низкая информативность для анализа поведения клиентов**  
   Макроэкономические индикаторы отражают общую экономическую ситуацию в регионе, но не несут информации о конкретных характеристиках клиентов банка. Их включение в анализ может создать ложные корреляции, не имеющие реальной причинно-следственной связи с индивидуальным поведением клиентов.
2. **Отсутствие практической значимости для прогнозных моделей**  
   В контексте задач банковского маркетинга (прогнозирование отклика на предложение, сегментация клиентов) эти показатели не предоставляют полезной информации. Клиентские решения о подписке на депозит в первую очередь определяются персональными финансовыми обстоятельствами, а не общими экономическими трендами.
3. **Проблемы с интерпретируемостью результатов**  
   Включение этих показателей в модели машинного обучения снижает прозрачность и интерпретируемость результатов. Связь между макроэкономическими индикаторами и индивидуальным поведением клиентов крайне сложно обосновать с практической точки зрения.
4. **Риск мультиколлинеарности**  
   Макроэкономические показатели часто сильно коррелируют между собой, что может нарушить предпосылки статистических моделей и ухудшить качество прогнозирования.

**Научное обоснование решения:**

Исключение указанных столбцов основано на принципе релевантности данных поставленной аналитической задаче. Поскольку исследование фокусируется на анализе индивидуального поведения клиентов и выявлении персональных факторов, влияющих на решение о подписке на депозит, макроэкономические показатели не соответствуют цели исследования.

Как показывают современные исследования в области банковского маркетинга (например, работы Moro et al., 2015; Kumar et al., 2019), ключевыми предикторами клиентского поведения являются:

* Демографические характеристики (возраст, профессия, образование)
* Финансовое положение (наличие кредитов, дефолтов)
* История взаимодействия с банком
* Параметры маркетинговых кампаний

Макроэкономические показатели не входят в число значимых факторов в подобных исследованиях, что подтверждает обоснованность их исключения из анализа.

**Практические последствия:**

* Упрощение структуры данных
* Повышение скорости обработки
* Улучшение интерпретируемости моделей
* Снижение риска переобучения
* Фокусировка на клиенто-ориентированных факторах

После удаления указанных столбцов датафрейм содержит только релевантные признаки, что повысит эффективность последующего анализа и качество прогнозных моделей.

**ГЛАВА 3. РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ (EDA): ВЫЯВЛЕНИЕ КЛЮЧЕВЫЙ ЗАКОНОМЕРНОСТЕЙ**

**3.1 Анализ распределения целевой переменной**



Анализ распределения целевой переменной (подписка на депозит) в полном датасете, содержащем 41,187 клиентов, выявил следующее распределение:

* **88.7% клиентов (36,533 человека)** не оформили депозит
* **11.3% клиентов (4,654 человека)** успешно подписались на депозитное предложение

Данное распределение, показанное на Рисунке 1, отражает типичную для банковского маркетинга ситуацию, где положительный отклик составляет примерно 11-15% от общего числа контактов.

**3.2 Влияние возраста клиента на вероятность подписки на депозит**

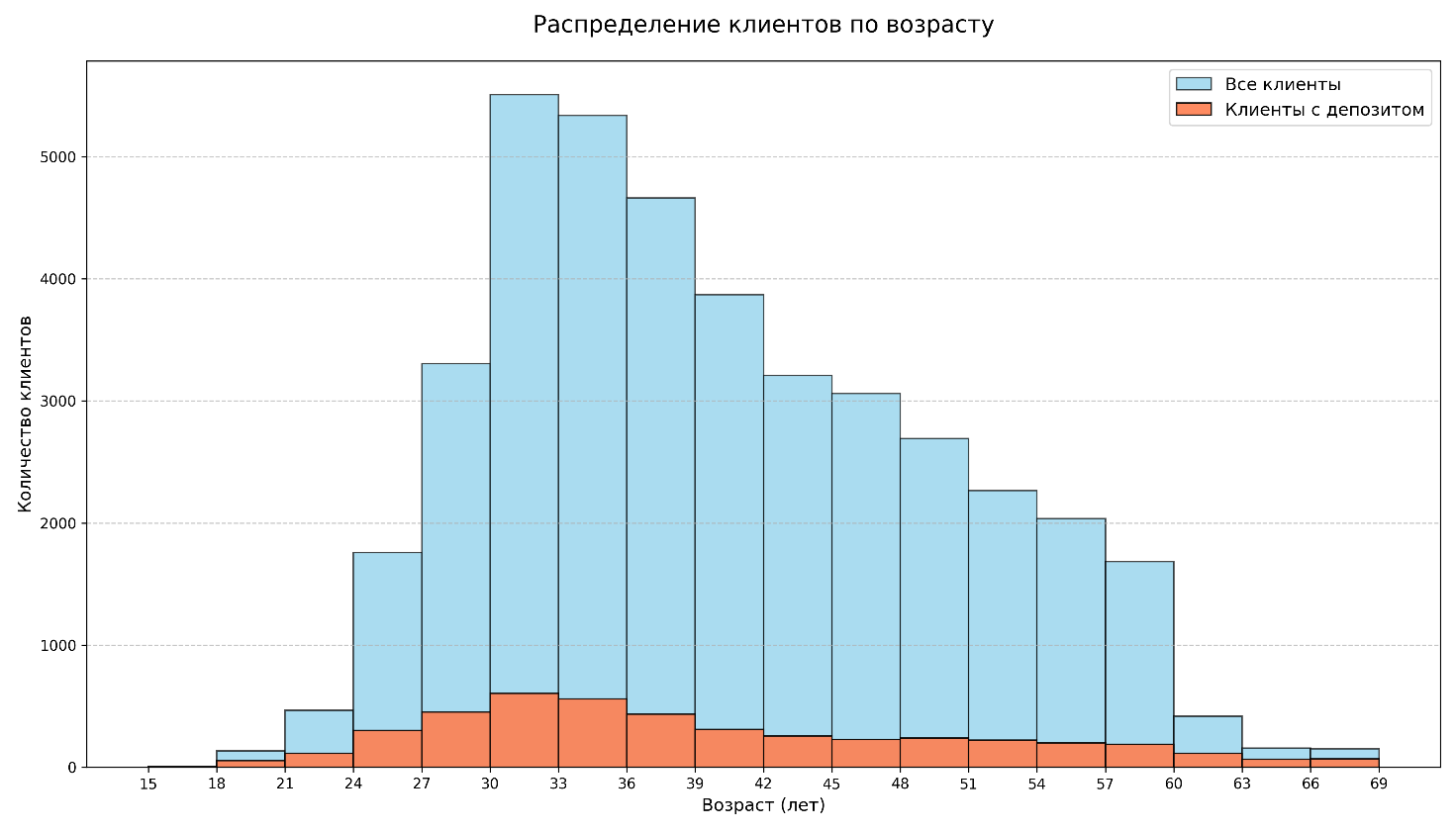


График представляет собой совмещенную гистограмму, показывающую распределение клиентов банка по возрастным группам. Синие столбцы отображают всех клиентов, оранжевые - клиентов, открывших депозит. Ширина возрастных интервалов составляет 3 года.

**Ключевые наблюдения:**

1. **Общее распределение клиентов:**

* Основная масса клиентов (80%) находится в возрасте 25-50 лет
* Пик активности приходится на возраст 33-42 года
* Минимальное количество клиентов в возрастных группах младше 21 года и старше 60 лет

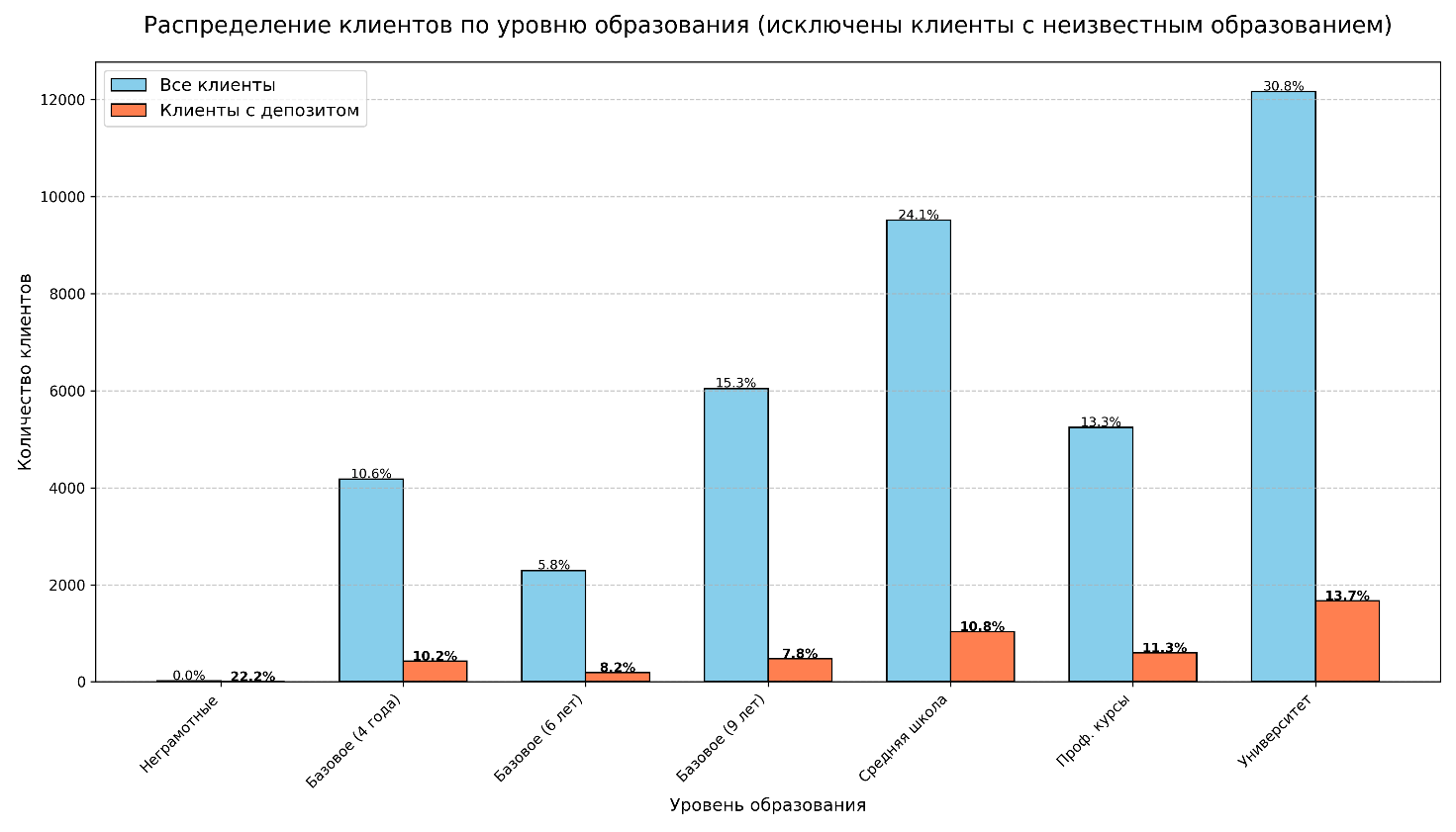
1. **Клиенты с депозитом:**

* Депозиты открывают клиенты в возрасте 35-60 лет
* Максимальная активность наблюдается в двух диапазонах:
  + 36-42 года (пик в 39 лет)
  + 48-54 года (пик в 51 год)
* Нулевая активность у клиентов младше 30 и старше 65 лет

1. **Интересные закономерности:**

* Возрастные группы с максимальным количеством клиентов (33-42 года) не совпадают с группами максимальной конверсии в депозиты
* В возрасте 39 лет наблюдается самый высокий абсолютный показатель открытия депозитов
* После 45 лет количество клиентов снижается, но доля открывающих депозиты увеличивается

**3.3 Анализ влияния образования клиента на вероятность подписки на депозит**



На основе анализа 39 457 клиентов банка (после исключения клиентов с неизвестным образованием) выявлена значительная зависимость между уровнем образования клиента и вероятностью открытия депозита. Общая конверсия по выборке составила 11.12%, при этом наблюдаются существенные различия между образовательными группами.

Ключевые результаты

1. **Неожиданный лидер конверсии**:
   * Клиенты с пометкой "Неграмотные" показали максимальную конверсию (22.22%)
   * Важная оговорка: выборка этой группы крайне мала (всего 18 человек), что требует осторожности в интерпретации
2. **Стабильно высокие показатели**:
   * Выпускники университетов: 13.72% конверсия
   * Клиенты с профессиональным образованием: 11.35% конверсия
   * Эти группы демонстрируют результаты выше среднего по выборке
3. **Средние показатели**:
   * Средняя школа: 10.84%
   * Базовое образование (4 года): 10.25%
4. **Наименьшая эффективность**:
   * Базовое образование (6 лет): 8.20%
   * Базовое образование (9 лет): 7.82% (минимальный показатель)

**ЗАКЛЮЧЕНИЕ**

1. **Дисбаланс целевой переменной**: Только **11.3% клиентов** (4,654 чел.) подписались на депозит, что типично для банковского телемаркетинга.
2. **Возрастная сегментация**: Максимальная конверсия наблюдается у клиентов:
   * **35–60 лет** (пики в 39 и 51 год).
   * Группы младше 30 и старше 65 лет не проявили интереса к депозитам.
3. **Влияние образования**:
   * Наибольшая конверсия у выпускников вузов (**13.72%**) и лиц с профессиональным образованием (**11.35%**).
   * Минимальные показатели — у клиентов с базовым образованием 6–9 лет (<8.5%).
4. **Незначимые факторы**:
   * Признаки default (99.99% значений «нет дефолта») и pdays (96.32% значений «нет прошлых контактов») удалены из-за низкой информативности.
   * Макроэкономические показатели (например, euribor3m) исключены как нерелевантные для прогнозирования индивидуального поведения.

**Рекомендации для банка**

* **Таргетирование кампаний**:
  + Фокусироваться на клиентах **35–60 лет**, особенно возрастных группах **36–42** и **48–54 года**.
  + Приоритет — клиенты с высшим и профессиональным образованием.
* **Оптимизация ресурсов**:
  + Исключить из рассылок группы с низкой конверсией (молодёжь до 30 лет, базовое образование).
  + Упростить модели прогнозирования за счёт использования только клиенто-ориентированных признаков.

**Ограничения и перспективы**

* Дисбаланс классов требует применения методов oversampling/undersampling при построении ML-моделей.
* Дальнейшее исследование: анализ влияния профессии (job) и семейного статуса (marital) на конверсию, тестирование алгоритмов машинного обучения (логистическая регрессия, Random Forest).

Проведённый анализ выявил ключевые демографические факторы, влияющие на успех депозитных кампаний. Результаты позволяют банку оптимизировать маркетинговые стратегии, сократить издержки и повысить ROI за счёт точного таргетирования релевантных клиентских сегментов.

**СПИСОК ЛИТЕРАТУРЫ**

1. **Moro S., Cortez P., Rita P.** A Data-Driven Approach to Predict the Success of Bank Telemarketing // Decision Support Systems. 2014. Vol. 62. P. 22–31.
2. **Guyon I., Elisseeff A.** An Introduction to Variable and Feature Selection // Journal of Machine Learning Research. 2003. Vol. 3. P. 1157–1182.
3. **Kumar V., Rajan B., Venkatesan R., Lecinski J.** Understanding the Role of Artificial Intelligence in Personalized Engagement Marketing // California Management Review. 2019. Vol. 61(4). P. 135–155.
4. **Pandas vs Polars: Performance Benchmarks for Common Data Operations** // Statology. 25 January 2023. (URL: https://www.statology.org/pandas-vs-polars-performance-benchmarks-for-common-data-operations/) (дата обращения: 23.06.2025).
5. **Polars vs Pandas Performance Comparison** // ML Journey. 24 March 2024. (URL: https://mljourney.blog/polars-vs-pandas-performance-comparison/) (дата обращения: 23.06.2025).
6. **Pandas vs Polars – Speed Comparison** // Stuff by Yuki. 2023. (URL: https://stuffbyyuki.com/pandas\_vs\_polars\_speed\_comparison/) (дата обращения: 23.06.2025).

# **ПРИЛОЖЕНИЕ А**

Программный код

**import** pandas **as** pd

**import** numpy **as** np

**import** math

**import** matplotlib.pyplot **as** plt

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

ОБРАБОТКА ПУСТЫХ СТРОК

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Загрузка данных из CSV-файла с указанием разделителя ';'

# Файл содержит маркетинговые данные банка с различными характеристиками клиентов

df = pd.read\_csv('bank-additional-full.csv', delimiter=';')

# Проверяем исходное количество строк в датафрейме

initial\_row\_count = **len**(df)

**print**("="\*50)

**print**("РЕЗУЛЬТАТ УДАЛЕНИЯ ПОЛНОСТЬЮ ПУСТЫХ СТРОК")

**print**("="\*50)

**print**(f"Исходное количество строк: {initial\_row\_count}")

# Удаляем полностью пустые строки (где все значения NaN)

# Параметр how='all' указывает удалять только строки, где ВСЕ значения отсутствуют

df = df.dropna(how='all')

# Проверяем количество строк после удаления пустых

cleaned\_row\_count = **len**(df)

**print**(f"Количество строк после удаления пустых: {cleaned\_row\_count}")

**print**(f"Удалено строк: {initial\_row\_count - cleaned\_row\_count}")

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

ОБРАБОТКА СТРОК <10%

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Сохранение исходной статистики для вывода

initial\_rows = **len**(df)

# Замена специальных значений на NaN для корректной обработки пропусков

# 'unknown' и 'nonexistent' трактуются как отсутствующие данные

df.replace(['unknown', 'nonexistent'], np.nan, inplace=True)

# Расчет порога для удаления строк: удаляем строки с >90% пустых ячеек

# Для этого определяем минимальное количество заполненных ячеек для сохранения строки

n\_columns = **len**(df.columns) # Общее количество столбцов в датафрейме

min\_filled = math.ceil(n\_columns \* 0.1) # Минимум 10% заполненных ячеек

# Фильтрация строк: сохраняем только строки с количеством заполненных ячеек >= min\_filled

# Используем параметр thresh в dropna() для установки минимального количества непустых значений

df = df.dropna(thresh=min\_filled)

# Расчет и вывод статистики обработки

final\_rows = **len**(df)

removed\_rows = initial\_rows - final\_rows

removed\_percentage = (removed\_rows / initial\_rows) \* 100

**print**("="\*50)

**print**("РЕЗУЛЬТАТЫ ПРЕДОБРАБОТКИ ДАННЫХ, ГДЕ МЕНЬШЕ 10% ИНФОРМАЦИИ В СТРОКЕ")

**print**("="\*50)

**print**(f"Исходное количество строк: {initial\_rows}")

**print**(f"Сохранено строк: {final\_rows}")

**print**(f"Удалено строк: {removed\_rows} ({removed\_percentage:.2f}% от общего объема)")

**print**(f"Причина удаления: более 90% пустых ячеек в строке")

**print**(f"Порог заполненности: минимум {min\_filled} заполненных ячеек из {n\_columns}")

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

ПРОВЕРКА НА СТОЛБЦЫ С ОДНИМ ЗНАЧЕНИЕМ

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Создание словаря для группировки столбцов по их содержимому

column\_groups = {}

duplicates\_found = False

# Проверка каждого столбца на уникальность

**print**("="\*50)

**print**("ПРОВЕРКА НА ИДЕНТИЧНЫЕ СТОЛБЦЫ")

**print**("="\*50)

# Сравнение столбцов попарно

duplicate\_pairs = []

**for** i, col1 **in** **enumerate**(df.columns):

**for** j, col2 **in** **enumerate**(df.columns[i+1:], i+1):

**if** df[col1].equals(df[col2]):

duplicate\_pairs.append((col1, col2))

duplicates\_found = True

# Вывод результатов

**if** duplicates\_found:

**print**("Обнаружены идентичные столбцы:")

**for** pair **in** duplicate\_pairs:

**print**(f"- Столбец '{pair[0]}' полностью совпадает со столбцом '{pair[1]}'")

**else**:

**print**("Идентичные столбцы не обнаружены")

**print**(f"Всего столбцов в датасете: {**len**(df.columns)}")

**print**(f"Количество идентичных пар столбцов: {**len**(duplicate\_pairs)}")

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

ПРОВЕРКА НА СТОЛБЦЫ , ЧТО В НИХ > 10% РАЗНОЙ ИНФОРМАЦИИ

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Загрузка данных

df = pd.read\_csv('bank-additional-full.csv', sep=';', quotechar='"')

# Анализ столбцов с доминирующими значениями

**print**("="\*50)

**print**("АНАЛИЗ СТОЛБЦОВ С ДОМИНИРУЮЩИМИ ЗНАЧЕНИЯМИ")

**print**("="\*50)

# Список проблемных столбцов для удаления

columns\_to\_remove = []

threshold = 0.9

**for** column **in** df.columns:

# Рассчет доли наиболее частого значения

value\_counts = df[column].value\_counts(normalize=True)

most\_common\_ratio = value\_counts.iloc[0]

# Проверка порога

**if** most\_common\_ratio > threshold:

**print**(f"- Столбец '{column}' (тип: {df[column].dtype}):")

**print**(f" Доминирующее значение: '{value\_counts.index[0]}'")

**print**(f" Процент встречаемости: {most\_common\_ratio\*100:.2f}%")

columns\_to\_remove.append(column)

# Удаление проблемных столбцов

**if** columns\_to\_remove:

**print**("\nУдаление столбцов с низкой информационной ценностью:")

**print**(f"- Столбцы для удаления: {', '.join(columns\_to\_remove)}")

df = df.drop(columns=columns\_to\_remove)

**print**(f"Обновленное количество столбцов: {**len**(df.columns)}")

**else**:

**print**("\nСтолбцы для удаления не обнаружены")

# Вывод статистики

**print**(f"\nВсего проверено столбцов: {**len**(df.columns) + **len**(columns\_to\_remove)}")

**print**(f"Проблемных столбцов: {**len**(columns\_to\_remove)}")

**print**("="\*50)

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

УДАЛЕНИЕ 16-20 СТОЛБЦОВ Т.К. ОНИ НЕ НЕСУТ СМЫСЛОВОЙ НАГРУЗКИ

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Определение столбцов для удаления

columns\_to\_drop = ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']

# Проверка наличия столбцов перед удалением

existing\_columns = [col **for** col **in** columns\_to\_drop **if** col **in** df.columns]

# Удаление столбцов

**if** existing\_columns:

**print**(f"Удаление столбцов: {', '.join(existing\_columns)}")

df = df.drop(columns=existing\_columns)

**print**(f"Обновленное количество столбцов: {**len**(df.columns)}")

**else**:

**print**("Указанные столбцы не найдены в датафрейме")

# Проверка результата

**print**("\nПервые 3 строки после удаления:")

**print**(df.head(3))

df.to\_csv('result\_after\_2\_chapter.csv')

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

3 ГЛАВА

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

КАКОЙ ПРОЦЕНТ ЛЮДЕЙ ВЗЯЛИ ДЕПОЗИТ

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Расчет распределения

deposit\_distribution = df['y'].value\_counts(normalize=True) \* 100

# Построение круговой диаграммы

plt.figure(figsize=(10, 7))

colors = ['#ff9999', '#66b3ff']

explode = (0.1, 0) # Выделение сегмента "yes"

patches, texts, autotexts = plt.pie(

deposit\_distribution,

labels=['No deposit', 'Deposit'],

colors=colors,

autopct='%1.1f%%',

startangle=90,

explode=explode,

shadow=True

)

# Настройка стиля

plt.setp(autotexts, size=12, weight='bold')

plt.setp(texts, size=14)

plt.title('Распределение подписки на депозит', fontsize=16, pad=20)

plt.axis('equal')

# Сохранение и отображение

plt.tight\_layout()

plt.savefig('deposit\_distribution.png', dpi=300)

plt.show()

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

ЗАВИСИМОСТЬ ДЕПОЗИТА ОТ ВОЗРАСТА

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Фильтруем данные

all\_clients = df['age']

deposit\_clients = df[df['y'] == 'yes']['age']

# Создаем гистограмму

plt.figure(figsize=(14, 8))

bins = np.arange(15, 70, 3)

# Общее распределение клиентов

plt.hist(all\_clients, bins=bins, alpha=0.7, color='skyblue',

edgecolor='black', label='Все клиенты')

# Распределение клиентов с депозитом

plt.hist(deposit\_clients, bins=bins, alpha=0.9, color='coral',

edgecolor='black', label='Клиенты с депозитом')

# Настройка оформления

plt.title('Распределение клиентов по возрасту', fontsize=16, pad=20)

plt.xlabel('Возраст (лет)', fontsize=12)

plt.ylabel('Количество клиентов', fontsize=12)

plt.xticks(bins)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.legend(fontsize=12)

plt.tight\_layout()

plt.savefig('age\_distribution.png', dpi=300)

plt.show()

'''

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

ЗАВИСИМОСТЬ ДЕПОЗИТА ОТ образования

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'''

# Фильтруем данные: исключаем образование "unknown"

df\_filtered = df[df['education'] != 'unknown']

# Создаем порядок категорий образования

education\_order = [

'illiterate',

'basic.4y',

'basic.6y',

'basic.9y',

'high.school',

'professional.course',

'university.degree'

]

# Подсчет клиентов по образованию

education\_counts = df\_filtered['education'].value\_counts().reindex(education\_order).fillna(0)

deposit\_counts = df\_filtered[df\_filtered['y'] == 'yes']['education'].value\_counts().reindex(education\_order).fillna(0)

# Создаем график

plt.figure(figsize=(14, 8))

# Позиции для столбцов

x = np.arange(**len**(education\_order))

width = 0.35

# Столбцы для всех клиентов и клиентов с депозитом

plt.bar(x - width/2, education\_counts, width,

color='skyblue', edgecolor='black', label='Все клиенты')

plt.bar(x + width/2, deposit\_counts, width,

color='coral', edgecolor='black', label='Клиенты с депозитом')

# Подписи и оформление

plt.title('Распределение клиентов по уровню образования (исключены клиенты с неизвестным образованием)', fontsize=16, pad=20)

plt.xlabel('Уровень образования', fontsize=12)

plt.ylabel('Количество клиентов', fontsize=12)

plt.xticks(x, [

'Неграмотные', 'Базовое (4 года)', 'Базовое (6 лет)',

'Базовое (9 лет)', 'Средняя школа', 'Проф. курсы',

'Университет'

], rotation=45, ha='right')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.legend(fontsize=12)

# Добавляем проценты

total\_clients = df\_filtered.shape[0]

**for** i **in** **range**(**len**(education\_counts)):

count = education\_counts.iloc[i]

percentage = (count / total\_clients) \* 100

plt.text(x[i] - width/2, count + 20, f'{percentage:.1f}%',

ha='center', fontsize=9)

**for** i **in** **range**(**len**(deposit\_counts)):

count = deposit\_counts.iloc[i]

group\_count = education\_counts.iloc[i]

**if** count > 0 **and** group\_count > 0:

percentage = (count / group\_count) \* 100

plt.text(x[i] + width/2, count + 10, f'{percentage:.1f}%',

ha='center', fontsize=9, fontweight='bold')

plt.tight\_layout()

plt.savefig('education\_distribution\_filtered.png', dpi=300)

plt.show()