Аннотация

В данной работе рассмотрена проблема недостаточной защищённости конфиденциальных данных, используемых банком для машинного обучения в облаке. Актуальность решения проблемы заключается в том, что обеспечение безопасности данных клиентов является одним из ключевых аспектов деятельности банка, так как возможные утечки к снижению уровня доверия клиентов, репутационным потерям и юридической ответственности.

В качестве решения проблемы выступает технология полностью гомоморфного шифрования, которое позволяет не только шифровать данные, но и производить вычисления с зашифрованными данными, что обеспечивает безопасность данных и даёт возможность разворачивать модели в ненадежных средах, в том числе, в облаке.

В результате анализа источников и литературы по теме исследования было выяснено, что наиболее подходящей для машинного обучения библиотекой полностью гомоморфного шифрования является TenSEAL, а большинство работ посвящены решению задачи аппроксимации сигмоидной функции, поскольку полностью гомоморфное шифрование не позволяет вычислять напрямую функции, которые невозможно представить с помощью сложения и умножения. Также во всех рассмотренных работах процесс шифрования данных не выделяется на отдельный этап, то есть в рамках клиент-серверной архитектуры загрузка и шифрование данных происходит на сервере, а не на клиенте, что также было решено в данной работе путём разработки дополнительных функций для библиотеки TenSEAL.

Применение полностью гомоморфного шифрования было продемонстрировано на примере решения задачи кредитного скоринга, которая является классической задачей для банка, а качестве модели была выбрана логистическая регрессия, рассматриваемая как простая однослойная нейронная сеть.

В качестве схемы шифрования была использована схема CKKS, так как она позволяет работать с вещественными числами, с помощью которой

данные были зашифрованы на клиенте и переданы на сервер для обучения на них модели.

В результате обучения модели логистической регрессии на зашифрованных данных была достигнута точность 86.27 %, что совпало с точностью модели логистической регрессии, обученной на незашифрованных данных, и что демонстрирует применимость технологии полностью гомоморфного шифрования для обеспечения безопасности данных в машинном обучении, так как оно не влияет на качество прогноза, но при этом обеспечивает максимальный уровень защиты данных, так как их содержимое никогда не раскрывается перед сервером.

Тем не менее, необходимо отметить, что использование полностью гомоморфного шифрования влечёт за собой дополнительные вычислительные затраты, например, увеличение размера данных и времени обучения модели.

Ключевые слова: полностью гомоморфное шифрование, CKKS, логистическая регрессия, полиномиальная аппроксимация, конфиденциальное машинное обучение.

1 Введение

В данной работе рассматривается ситуация, в которой банк, использующий в своей деятельности машинное обучение, столкнулся с нехваткой вычислительной мощности, поэтому было принято решение арендовать для этих целей облачный сервер. Однако так как данные передаются на сторонний сервер, то возникает проблема обеспечения безопасности данных, используемых для машинного обучения, актуальность решения которой заключается в том, что уязвимости в самом облаке, возможности несанкционированного доступа или даже неправильная настройка доступа к облаку могут привести к угрозам информационной безопасности, например, утечке данных, что снизит уровень доверия клиентов, а также приведёт к репутационным потерям и штрафам за нарушение правил информационной безопасности.

Современные тенденции в области обеспечения безопасности машинного обучения включают в себя широкий класс методов для обеспечения безопасности и конфиденциальности данных. Например, многосторонние вычисления позволяют нескольким участникам совместно выполнять вычисления над общими данными без раскрытия этих данных друг другу [1], а концепция дифференциальной приватности позволяет уменьшить связь данных с пользователями за счёт создания ограничений [2].

Такие методы достаточно практичны и могут быть легко интегрированы в существующие системы машинного обучения, но при их использовании содержание данных в том или ином виде всё равно раскрывается перед сервером, и предотвратить это можно только с использованием полностью гомоморфного шифрования, которое позволяет выполнять вычисления, в частности, сложение и умножение, над зашифрованными данными без их расшифровки, что обеспечивает наивысший уровень безопасности [3], так как содержание данных никогда не раскрывается. Такое свойство гомоморфного шифрования, из-за которого его также называют «святым Граалем криптографии» [4], делает его применение лучшим выбором для защиты данных в машинном обучении.

Целью данной работы является разработка модели машинного обучения на зашифрованных данных на примере решения задачи кредитного скоринга, а новизна работы заключается в реализации клиент-серверного сценария использования библиотеки TenSEAL путём передачи на сервер уже зашифрованных на клиенте данных.

Для того, чтобы показать, насколько эффективно обучение модели логистической регрессии на зашифрованных данных, предлагается предварительно обучить эту же модель, но не на зашифрованных данных.

Для решения поставленной цели необходимо решить следующие задачи:

- Проанализировать источники и литературу по применению полностью гомоморфного шифрования для машинного обучения, в частности, по решению задачи аппроксимации сигмоидной функции.

- Найти и подготовить данные для решения задачи кредитного скоринга.
- Определить схему и параметры шифрования, после чего зашифровать подготовленные данные.
- Разработать механизм выгрузки зашифрованных данных и их отправку на сервер.
- Обучить модель логистической регрессии на зашифрованных и незашифрованных данных.
- Проанализировать результаты обучения модели на зашифрованных данных.

2 Анализ источников и литературы по применению полностью гомоморфного шифрования для машинного обучения

Большинство работ [15, 16, 17, 18, 19], посвященных использованию полностью гомоморфного шифрования для машинного обучения, описывают только шифрование на фазе вывода работы нейронной сети , и не касаются этапа обучения. Например, работа [15] посвящена аддитивно-гомоморфному шифрованию с интерактивным протоколом, в результате чего вывод небольшой нейронной сети занял всего 10 секунд, в то время как в следующей работе [16] для набора данных MNIST вывод занял уже 30 мс.

Предположительно, небольшое внимание к обучению на зашифрованных данных обусловлено тем, что оно занимает слишком много времени, однако уже в 2019 году в работе [12] с помощью библиотеки HELib была обучена нейронная сеть на основе стохастического градиентного спуска (SGD), что продемонстрировало эффективность обучения на зашифрованных данных, а также авторы предложили несколько методов для улучшения обучения, например, анализ сети, выбор подходящего представления данных и оптимизация упаковки данных в зашифрованных текстах.

В работе [14] была реализована логистическая регрессия и аппроксимированная функция логистической регрессии с помощью схемы CKKS и с использованием Microsoft SEAL. Процент правильных прогнозов

совпал для обучения на открытых и зашифрованных данных, однако после шифрования размер входного набора данных увеличился в 272.5. раз, а время обработки данных увеличилось примерно в 4865 раз.

Также не все работы используют существующие схемы полностью гомоморфного шифрования. В работе [13] был представлен вариант бесшумной схемы гомоморфного шифрования на основе матрицы (MORE), которая требует достаточно небольших вычислительных затрат и, что важно, позволяет выполнять операции непосредственно с числами с плавающей запятой, что собой критическое свойство для нейронных сетей. Однако схема шифрования MORE обеспечивает меньшую безопасность по сравнению со стандартными схемами, так как основывается на линейных функциях и существует возможность определения секретного ключа, имея доступ к достаточно большому количеству пар зашифрованных и незашифрованных значений.

Для дальнейшего анализа существующих работ по теме исследования, важно отметить, что так как при использовании полностью гомоморфного шифрования над зашифрованными данными возможны только операции сложения и умножения, то с помощью них должны быть представлены все вычисляемые функции. В то время как в машинном обучении требуются неполиномиальные функции, в частности, логистической регрессии, применяемой для решения поставленной задачи кредитного скоринга, необходима сигмоидная функция:

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

Как видно из формулы (1), сигмоида не может быть эффективно представлена с помощью только сложения и умножения, и поэтому не может быть вычислена гомоморфно, поэтому разработка эффективного способа поддержки неполиномиальных функций является активной темой исследований в последние годы.

Для поддержки неполиномиальных функций, в частности, логистической сигмоиды, в гомоморфных вычислениях выделяются следующие подходы:

- использование полиномиальной замены;
- предварительное вычисление функции для дискретных значений и сохранение этих значений в справочных таблицах;
- использование полиномов низкой степени для аппроксимации неполиномиальных функций.

Подход, заключающийся в использовании полиномиальной замены, был самым первым решением проблемы поддержки неполиномиальных функций, и был представлен в 2016 году в работе [5], посвящённой CryptoNets, в которой авторы предложили замену сигмоиды квадратичным полиномом:

$$f(x) = x^2 \tag{2}$$

Но квадратичный полином растет экспоненциально и вызывает нестабильность во время обучения нейросети [6]. Как следствие, CryptoNets не может поддерживать более одного уровня активации, что оказывает значительное влияние на точность.

Другой подход был представлен в 2018 году в работе [7], заключающийся в предварительном вычислении дискретных выборок неполиномиальной функции активации и сохранении этих дискретных значений в справочной таблице. Во время гомоморфных вычислений программа выполняет поиск по таблице, однако низкие частоты дискретизации снизят точность вычислений с плавающей точкой и окажут прямое влияние на производительность нейронных сетей. И наоборот, увеличение частоты выборки может привести к созданию слишком больших справочных таблиц.

Более эффективный и часто используемый подход заключается в аппроксимации неполиномиальных функций полиномами низкой степени. Использование низкой степени обусловлено тем, что схемы гомоморфного шифрования ограничивают количество операций умножения.

Рассмотрим наиболее используемые методы полиномиальной аппроксимации:

1. Численный метод

Данный метод заключается в том, что «коэффициенты полинома $p(x)$ степени N подбираются при помощи логистической регрессии таким образом, чтобы минимизировать среднюю квадратичную ошибку (mean squared error – MSE) в узлах интерполяции» [10].

В работе [8] был исследован данный метод, для чего авторы сгенерировали набор вычисленных значений точек функции ReLU, который передали функции аппроксимации вместе со степенью полинома. Эксперименты проводились с полиномами степени от 3 до 13, в результате чего было выяснено, что чем ниже степень полинома, тем ниже точность, то есть для достижения достаточной точности требуется увеличивать степень полинома, что не является эффективным для гомоморфных вычислений. Кроме этого, использование численного метода продемонстрировало низкую производительность.

2. Ряд Тейлора

Аппроксимация функции строится численным методом, и заключается в построении полинома, приближающего исходную функцию на заданном отрезке, методом Крога [10], и разложении функции $f(x)$ в ряд Тейлора в окрестностях заданной точки c:

$$f(x) = f(c) + \sum_{k=1}^{\infty} \frac{f^{(k)} \cdot (c)}{k!} \cdot (x - c)^k \quad (3)$$

Например, функция ReLU была аппроксимирована с помощью разложения в ряд Тейлора в работе [9], и в сочетании с другими методами, такими как пакетная нормализация [11], устанавливающей границу входных данных для каждого слоя активации, авторы смогли достичь точность 99.3 %

В работе [8] также исследовалась аппроксимация функции ReLU с помощью разложения в ряд Тейлора, для чего были использованы полиномы разных степеней, однако метод столкнулся с двумя основными проблемами,

делающими его неэффективным. Во-первых, проблемой является высокая степень полиномиальной аппроксимации, хотя она ниже, чем у численного метода, но всё еще выше для использования в схемах полностью гомоморфного шифрования. Во-вторых, в данном методе существует проблема интервала аппроксимации, так как разложение в ряд Тейлора заключается в аппроксимации функции в окрестностях какой-то точки, и для точек, не из рассматриваемой окрестности, ошибка аппроксимации значительно выше, чем для точки, принадлежащей ей.

3. Полиномы Чебышёва

В отличии от разложения в ряд Тейлора использование полиномов Чебышёва позволяет аппроксимировать функцию на интервале, а не в окрестности точки.

Аппроксимация функции $f(x)$ на отрезке $[a, b]$ строится в форме полинома следующего вида [10]:

$$f(x) = \sum_{i=0}^{N} a_i \cdot T_i \cdot \left(2 \cdot \frac{x-a}{b-a} - 1\right) \tag{4}$$

Сами полиномы Чебышёва определяются как:

$$T_i(x) = \cos \cdot \left(i \cdot arccos(x)\right), x \in [-1, 1] \tag{5}$$

где a_i – соответствующие коэффициенты.

В работе [8] было рассмотрено использование как стандартных полиномов Чебышёва, которые ортогональны на отрезке $[-1, 1]$ с весом:

$$h = \frac{1}{\sqrt{1-x^2}} \tag{6}$$

Так и их модифицированная версия, в которой было предложено рассчитывать вес следующим образом:

$$h = e^{-\frac{1}{1e-5+x^2}} \tag{7}$$

В результате чего было выяснено, что использование стандартных полиномов Чебышёва позволила достичь точности только в 69.98 %, в то время как их модицифированная версия достигла точность 88.53 %.

Большая точность связана с тем, что вес, рассчитанный специально для аппроксимируемой функции, приводит к меньшим колебаниям в результирующем приближении и, следовательно, к большей сходство производных с сигмоидной функцией.

4. Производная аппроксимируемой функции

Данный метод предполагает использование аппроксимации полиномами низкой степени одним из методов, описанных ранее, но вместо самой функции $f(x)$ аппроксимировать её производную $f'(x)$. Тогда аппроксимацией функции $f(x)$ на отрезке $[a, b]$ является полином $p(x)$ [8]:

$$p(x) = \int p_1(x)dx \tag{8}$$

где $p_1(x)$ – полиномиальное приближение для производной $f'(x)$ на отрезке $[a, b]$.

Сигмоидная функция является непрерывной и бесконечно дифференцируемой, и в работе [8] было доказано, что это позволяет аппроксимировать ее более точно, чем, например, функцию ReLU, которая не дифференцируема в точке 0.

5. Минимаксная аппроксимация

При использовании данного метода максимальное отклонение аппроксимирующей функции от исходной на заданном интервале минимизируется. Такая задача обычно решается с помощью численных методов оптимизации, например, полиномов Чебышёва.

В работе [20] была рассмотрена минимаксная аппроксимация сигмоидной функции с использованием алгоритма Ремезы, а также сочетание начальной загрузки с операцией масштабирования для вычислений чисел с фиксированной точкой, в результате чего была достигнута точность 97.8 % при затратах в 0,4-3,2 часа на одну итерацию градиентного спуска.

Как видно из существующих работ, подход полиномиальной аппроксимации более надежен, чем подход поиска по таблице, и дает более высокую точность, чем простое использование функции замены низкой

степени. Но существующие работы также показывают, что определение наиболее эффективного полинома является наиболее сложной и важной задачей.

Поэтому, рассмотрев пять подходов к аппроксимации сигмоидной функции, сведём примеры полиномов, аппроксимирующих сигмоидную функцию, в таблицу 1, используя полиномы для первых четырёх подходов из работы [10], а для минимаксной аппроксимации – из работы [20]. При этом ограничимся только полиномами третьей степени, так как полиномы второй степени не обеспечивают достаточную точность [14], а использование полиномов высоких степеней ограничивается возможным количеством умножений в гомоморфном шифровании [6].

Для оценки качества аппроксимации используем среднюю квадратичную ошибку. В работе [10] значения MSE уже рассчитаны, в то время как для минимаксной аппроксимации её требуется рассчитать по формуле:

$$MSE(f,p) = \frac{1}{m} \cdot \sum_{i=1}^{m} (f(x_i) - p(x_i))^2 \tag{9}$$

Таблица 1 – Полиномиальные аппроксимации для сигмоидной функции

Метод	**Полином**	**MSE**
Численный метод [10] $x_i \sim U(-200, 200)$	$\sigma(x) = -1.378 \cdot 10^{-7} \cdot x^3 - 3.097 \cdot 10^{-7} \cdot x^2 + 0.007042 \cdot x + 0.5127$	0.0328
Ряд Тейлора [10]	$\sigma(x) = -2.437 \cdot 10^{-8} \cdot x^3 - 2.054 \cdot 10^{-19} \cdot x^2 + 0.003363 \cdot x + 0.5$	0.0634
Полиномы Чебышёва [10]	$\sigma(x) = -1.061 \cdot 10^{-7} \cdot x^3 - 2.141 \cdot 10^{-20} \cdot x^2 + 0.006365 \cdot x + 0.5$	0.0340
Аппроксимация производной [10]	$\sigma(x) = -5.305 \cdot 10^{-8} \cdot x^3 + 0.004774 \cdot x + 0.5305$	0.0445
Минимаксная аппроксимация [20] $x \in [-5; 5]$	$\sigma(x) = -0.004 \cdot x^3 + 0.197 \cdot x + 0.5$	0.0114

Из таблицы 1 видно, что минимаксная аппроксимация сигмоидной функции показывает наименьшую среднюю квадратичную ошибку, но, чтобы убедиться в этом, построим графики аппроксимирующей и сигмоидной функций, как показано на рисунке 1.

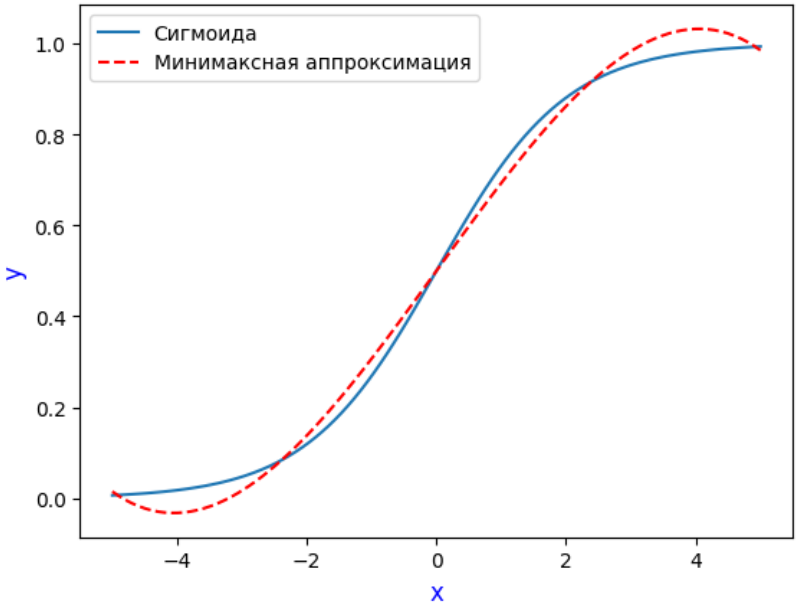

Рисунок 1. Графики аппроксимирующей (минимаксная аппроксимация) и сигмоидной функций

Таким образом, в результате проведённого анализа, для решения задачи кредитного скоринга предлагается использовать аппроксимирующую функцию, полученную с помощью минимаксной аппроксимации, так как она наиболее точно аппроксимирует сигмоидную функцию.

Также следует отметить, что во всех рассмотренных работах процесс шифрования данных не выделяется на отдельный этап, то есть в рамках клиент-серверной архитектуры загрузка и шифрование данных происходит на сервере, а не на клиенте, что подвергает данные определённым угрозам информационной безопасности, и что предлагается решить в рамках данной работы путём разработки дополнительного функционала для библиотеки TenSEAL.

3 Разработка модели машинного обучения с использованием полностью гомоморфного шифрования

3.1 Модель логистической регрессии для решения задачи кредитного скоринга

Задача кредитного скоринга в банке заключается в оценке способности потенциального заёмщика выполнить свои кредитные обязательства – то есть, в определении вероятности того, что заёмщик своевременно будет выплачивать кредит. Решение этой задачи помогает банку управлять рисками кредитования, минимизировать потери от невозврата кредитов и увеличить прибыль.

В настоящее время данная задача решается с помощью методов машинного обучения, и так как при её решении требуется классифицировать два взаимоисключающих класса (вернёт заёмщик кредит или не вернёт), то она относится к задачам бинарной классификации.

Одной из популярных моделей для решения таких задач является логистическая регрессия, которая моделирует вероятность принадлежности заёмщика к одному из двух классов, которые можно определить как «Надёжный заёмщик» и «Ненадёжный заёмщик». Логистическая регрессия предсказывает вероятность того, что заёмщик окажется надёжным, на основе набора входных переменных (например, возраст заёмщика, его доход, кредитная история, сумма кредита и т. д.).

В отличие от многих других алгоритмов классификации, логистическая регрессия предоставляет не только прогноз класса, но и вероятность принадлежности к каждому из классов, что полезно при оценке рисков. Также она основана на относительно простых математических операциях, что делает её вычислительно эффективной даже для больших объёмов данных, в том числе, зашифрованных.

Несмотря на существование более сложных и мощных алгоритмов машинного обучения, логистическая регрессия остаётся популярным выбором для решения задачи кредитного скоринга благодаря своей простоте и

интерпретируемости и способности, что делает её идеальным первым шагом в построении более сложных моделей оценки кредитного риска.

Логистическую регрессию можно рассматривать как простую однослойную нейронную сеть, использующую сигмоидальную функцию активации (1).

Модель логистической регрессии вычисляет взвешенную сумму входных переменных (плюс смещение), и выводит вероятность результата:

$$\hat{p} = h_\theta(x) = \sigma(\theta^T x) \tag{10}$$

где h_θ – функция гипотезы с использованием параметров модели;

σ – логистическая функция (1);

θ^T – транспонированный вектор параметров модели.

Если предполагаемая вероятность больше 50 %, то модель предсказывает, что экземпляр принадлежит положительному классу «1» (заёмщик вернёт кредит), а если меньше 50 %, то экземпляр принадлежит отрицательному классу «0» (заёмщик не вернёт кредит):

$$\hat{y} = \begin{cases} 0, \hat{p} < 0.5 \\ 1, \hat{p} \geq 0.5 \end{cases} \tag{11}$$

Так как решаемая задача кредитного скоринга относится к задачам бинарной классификации, то в качестве функции потери используется бинарная потеря кросс-энтропии, которая измеряет несоответствие между предсказанными вероятностями и истинными бинарными метками, что позволяет оценить, насколько хорошо модель предсказывает вероятность принадлежности к положительному классу.

$$L = \frac{1}{m} \cdot \sum_{i=1}^{m} [y^{(i)} \cdot \log\left(\hat{y}^{(i)}\right) + (1 - y^{(i)}) \cdot \log\left(1 - \hat{y}^{(i)}\right)] \tag{12}$$

где m – количество экземпляров в наборе данных.

$y^{(i)}$ – фактическая метка для каждого экземпляра (0 или 1)

$\hat{y}^{(i)}$ – предсказанная вероятность принадлежности экземпляра положительному классу

В качестве оптимизатора логистической регрессии используется стохастический градиентный спуск, так как он требует меньше вычислений на итерацию, используя только один обучающий пример за раз при обновлении веса модели. Это делает его особенно подходящим для обучения на больших данных, где полный проход может быть очень затратным по времени.

3.2 Подготовка данных

Исходные данные представляют собой набора данных «Credit score classification», скачанный с Kaggle, который включает в себя практически полную информацию о заёмщике (28 столбцов), например:

- Annual_Income – годовой доход.
- Monthly_Inhand_Salary – ежемесячный доход.
- Num_Bank_Accounts – количество банковских счетов.
- Num_Credit_Card – количество кредитных карт.
- Num_of_Loan – количество кредитов.

Этап подготовки данных включает в себя удаление лишних символов, обработку не имеющих смысла значения и выбросов, а также нормализацию категориальных данных. Также, так как используемая функция аппроксимации сигмоиды определена только на отрезке [-5; 5], то и все данные, в том числе, числовые, нормализуем в рамках заданного диапазона.

Для проведения корреляционного анализа используем тепловую карту, с помощью которой определим поля с высоким коэффициентом корреляции (больше 0.6), которые удалим для того, чтобы избежать мультиколлинеарности, то есть высокой степени взаимосвязи между признаками, которая может стать причиной переобучения модели. В результате чего удалим поля Monthly_Balance, Total_EMI_per_month и Payment_of_Min_Amount, так как о них можно делать выводы по полям Monthly_Inhand_Salary, Num_of_Loan и Credit_Mix соответственно.

Также в целях избегания переобучения модели, и, как следствие, повышения её качества, разделим данные на тестовую и обучающую выборки.

3.3 Шифрование данных

В качестве библиотеки полностью гомоморфного шифрования выбрана TenSEAL, основанная на Microsoft SEAL, которая в настоящее время является наиболее используемой библиотекой полностью гомоморфного шифрования. С помощью API она обеспечивает простоту использования языка Python, при этом сохраняя эффективность за счет реализации большинства операций с использованием C++.

TenSEAL, основанная на Microsoft SEAL, в настоящее время является наиболее используемой библиотекой полностью гомоморфного шифрования. Например, данная библиотека использовалась в работах [15, 17, 18]. С помощью API она обеспечивает простоту использования языка Python, при этом сохраняя эффективность за счет реализации большинства операций с использованием C++.

TenSEAL позволяет производить поэлементное сложение, вычитание и умножение зашифрованных векторов, а шифрование и дешифрование данных осуществляет по схеме BFV для целых чисел или CKKS для вещественных чисел, которая и используется в разрабатываемой модели, так использование вещественных чисел позволяет добиться большей точности.

Для указания схемы и параметров шифрования в TenSEAL используется специальный объект – TenSEALContext.

Параметры шифрования θ схемы CKKS включают в себя:

- степень полиномиального модуля N;
- размеры модуля коэффициента q.

Прежде чем приступить к определению параметрам шифрования, следует определить количество необходимых операций умножения:

- для операции скалярного произведения – 1;
- для аппроксимации сигмоидной функции – 2;
- для осуществления обратного распространения ошибки – 3.

Таким образом, мультипликативная глубина равна 1 + 2 + 3 = 6, что определяет количество чисел, составляющих модуль коэффициента.

Степень полиномиального модуля, которую ещё называют коэффициентом масштабирования, должна быть степенью двойки, так как она определяет точность кодирования для двоичного представления числа. Также эта степень напрямую влияет на количество коэффициентов в полиноме открытого текста, размер элементов зашифрованного текста, вычислительную производительность схемы и уровень безопасности. При этом, чем выше степень полиномиального модуля, тем ниже производительность, но выше уровень безопасности.

Приняв желаемый уровень защищенности, эквивалентный AES, равный 128 битам, степень полиномиального модуля следует определить не ниже 8192, что позволит группировать до 4096 значений в одном зашифрованном тексте.

Размеры модуля коэффициента представляют собой список двоичных размеров, используя который TenSEAL генерирует список простых чисел этих двоичных размеров. Модуль коэффициента влияет на размер элементов зашифрованного текста, а длина списка указывает уровень схемы, то есть количество поддерживаемых умножений. При этом, чем длиннее данный список, тем выше уровень безопасности.

Так как желаемый уровень защищённости составляет 128 бит, количество умножений равно 6, а 128 / 6 = 21.3, то в качестве двоичного размера следует использовать 21 бит. При этом двоичные размеры в количестве, равном количеству умножений, определяются в списке размеров со второй по предпоследнюю позицию, в то время как первый и последний размер в данном списке должны быть больше их по размеру.

Поэтому размеры модуля коэффициента [40, 21, 21, 21, 21, 21, 21, 40] обозначают то, что модуль коэффициента будет содержать 8 простых чисел: первое и последнее по 40 бит и остальные по 21 бит.

Таким образом, выбранные параметры шифрования схемы CKKS приведены в таблице 2.

Таблица 2 – Выбранные параметры шифрования схемы CKKS

Степень полиномиального модуля	**Размеры модуля коэффициента**
8192	[40, 21, 21, 21, 21, 21, 21, 40]

При создании TenSEALContext по умолчанию создаются ключи релинеаризации, а также включена автоматическая релинеаризация и масштабирование, однако для поддержки выполнения операции скалярного произведения векторов необходимо отдельно создать ключи Галуа с помощью метода generate_galois_keys().

Для обеспечения безопасности данных на сервер нужно загружать уже зашифрованные на клиенте данные, для чего для каждой из выборок создадим зашифрованные вектора, которые сериализуем, то есть представим в виде двоичных данных, и запишем в файлы с расширением «hex». Каждый из таких файлов соответствует одному зашифрованному вектору и весит 429 Кбайт, поэтому для экономии места архивируем получившиеся файлы, как показано в листинге 1.

Листинг 1 – Функция шифрования данных

```
def encrypt_data(data, filename, ctx_training):
    t_start = time()
    rows = x_train.shape[0]
    zip_name = f'encrypted data//{filename}.zip'

    with ZipFile(zip_name, 'w') as filezip:
        for i in range(rows):
            vector = ts.ckks_vector(ctx_training,
data[i].tolist()).serialize()
            filename_with_num = f'{filename}_{i}.hex'
            with open(filename_with_num, 'wb') as file:
                file.write(vector)
            filezip.write(filename_with_num)
            os.remove(filename_with_num)
    t_end = time()
    print(f'Шифрование заняло {int(t_end - t_start)} секунд')
```

Таким же образом сериализуем объект TenSEALContext для того, чтобы развернуть его на сервере и не настраивать параметры шифрования ещё раз.

Отметим, что обычно для отправления данных на сервер используется формат JSON, однако для записи сериализованных векторов в такой файл их придётся преобразовать в строки, что кратно увеличит размер JSON файла, поэтому такой формат хранения зашифрованных данных подходит на этапе использования, а не обучения модели.

3.4 Разработка модели логистической регрессии, обучаемой на зашифрованных данных

Перед тем, как приступить к обучению модели на сервере, следует загрузить и восстановить на нём сериализованный TenSEALContext, содержащий схему и параметры шифрования, настроенные на клиенте, а также сериализованные зашифрованные векторы.

Для восстановления TenSEALContext можно воспользоваться функцией context_from(), но для зашифрованных векторов следует сначала распаковать архив, а затем последовательно открыть каждый из двоичных файлов, из которого уже восстанавливается зашифрованный вектор в виде объекта CKKSVector, как показано в листинге 2.

Листинг 2 – Функция восстановления зашифрованных векторов

```
def get_ckks_vector(filename, ctx_training):
    zip_name = f'encrypted data//{filename}.zip'
    with ZipFile(zip_name, "r") as myzip:
        myzip.extractall(path='encrypted data')

    ckks_vectors_list = []
    i = 1
    while True:
        filename_with_num = f'encrypted data//{filename}_{i}.hex'
        try:
            with open(filename_with_num, 'rb') as file:
                bytes_vector = file.read()
            ckks_vector = ts.ckks_vector_from(ctx_training,
bytes_vector)
            ckks_vectors_list.append(ckks_vector)
```

Продолжение листинга 2

```
            os.remove(filename_with_num)
            i += 1
        except:
            break

    return ckks_vectors_list
```

Как уже отмечалось ранее, модель логистической регрессии в качестве функции потери использует бинарную потерю кросс-энтропии, однако для борьбы с переобучением модели следует использовать регуляризацию, которая действует путем добавления штрафа к функции потерь.

Величина штрафа меняется в зависимости от типа регуляризации. L1 регуляризация основана на добавлении штрафа, равного абсолютному значению коэффициентов модели, а L2 регуляризация добавляет штрафную функцию, являющуюся суммой квадратов весов модели, умноженных на гиперпараметр регуляризации.

L1 регуляризация может быть полезна в ситуациях, когда необходимо уменьшить количество признаков для упрощения модели, так как она может обнулять коэффициенты для менее значимых признаков. Однако для принятия решения о выдаче кредита следует учитывать все признаки, поэтому следует использовать L2 регуляризацию, которая не обнуляет коэффициенты, а лишь уменьшает их величину.

Тогда функция потерь (12) примет вид:

$$L = \frac{1}{m} \cdot \sum_{i=1}^{m} \left[y^{(i)} \cdot \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \cdot \log\left(1 - \hat{y}^{(i)}\right) \right] + \frac{\lambda}{2 \cdot m} \cdot \sum_{j=1}^{n} \theta_j^2 \quad (13)$$

где λ – параметр регуляризации, который контролирует степень влияния штрафа на функцию потерь;

θ – вектор параметров модели.

Для обновления параметров применяется следующее правило:

$$\theta_j = \theta_j - \alpha \cdot [\frac{1}{m} \cdot \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \cdot x^{(i)} + \frac{\lambda}{m} \cdot \theta_j] \quad (14)$$

Однако, учитывая ограничения полностью гомоморфного шифрования, примем $\alpha = 1$ и $\frac{\lambda}{m} = 0.05$, и тогда правило обновление параметров примет следующий вид:

$$\theta_j = \theta_j - [\frac{1}{m} \cdot \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \cdot x^{(i)} + 0.05 \cdot \theta_j] \quad (15)$$

Поскольку полностью гомоморфное шифрование не позволяет напрямую вычислить сигмоидную функцию, так как её невозможно представить с помощью сложения и умножения, то её необходимо аппроксимировать с помощью полинома низкой степени. В результате проведённого анализа источников и литературы по теме исследования, в качестве аппроксимирующей функции была выбрана следующая функция:

$$\sigma(x) = -0.004 \cdot x^3 + 0.197 \cdot x + 0.5 \quad (16)$$

Данная функция была получена с помощью минимаксной аппроксимации, и она аппроксимирует сигмоидную функцию на отрезке [-5; 5], поэтому на этапе подготовки данные были нормализованы в соответствии с данным диапазоном.

Используя PyTorch, создадим модель логистической регрессии, использующей зашифрованные данные.

Листинг 3 – Модель обучения логистической регрессии на зашифрованных данных

```
class EncryptedLogisticRegression:

    def __init__(self, torch_lr):
        self.weight = torch_lr.lr.weight.data.tolist()[0]
        self.bias = torch_lr.lr.bias.data.tolist()
        # накапливаем градиент и подсчитываем количество итераций
        self._delta_w = 0
        self._delta_b = 0
        self._count = 0
```

Продолжение листинга 3

```
    def forward(self, enc_x):
        enc_out = enc_x.dot(self.weight) + self.bias
        enc_out = EncryptedLogisticRegression.sigmoid(enc_out)
        return enc_out

    def backward(self, enc_x, enc_out, enc_y):
        out_minus_y = (enc_out - enc_y)
        self._delta_w += enc_x * out_minus_y
        self._delta_b += out_minus_y
        self._count += 1

    def update_parameters(self):
        if self._count == 0:
            raise RuntimeError('Вы должны выполнить по крайней мере одну прямую
итерацию')
        # обновляем параметры в соответствии с L2 регуляризацией, приняв α=1 и
λ/m=0.05
        self.weight -= self._delta_w * (1 / self._count) + self.weight * 0.05
        self.bias -= self._delta_b * (1 / self._count)
        # обнуляем накапливание градиента и счётчик итераций
        self._delta_w = 0
        self._delta_b = 0
        self._count = 0

    @staticmethod
    def sigmoid(enc_x):
        return enc_x.polyval([0.5, 0.197, 0, -0.004])

    def encrypt(self, context):
        self.weight = ts.ckks_vector(context, self.weight)
        self.bias = ts.ckks_vector(context, self.bias)

    def decrypt(self):
        self.weight = self.weight.decrypt()
        self.bias = self.bias.decrypt()

    def __call__(self, *args, **kwargs):
        return self.forward(*args, **kwargs)
```

Данная модель учитывает обновление параметров в соответствии с L2 регуляризацией, вычисляя вес с учётом принятого коэффициента, равного 0.05, а также переопределяет сигмоидальную функцию как полином с коэффициентами 0.5, 0.197, 0 и -0.004.

3.5 Анализ результатов

Основные результаты, полученные в ходе эксперимента по обеспечению безопасности данных при обучении модели логистической регрессии на сервере, приведены в таблице 3.

Таблица 3 – Характеристика модели логистической регрессии для решения задачи кредитного скоринга

Параметр	**Значение**
Вес исходного файла с набором данных (csv)	30 407 Кбайт
Вес одного зашифрованного вектора (hex)	429 Кбайт
Время шифрования одного тензора ([3523, 19])	59 секунд
Точность обучения модели на зашифрованных данных	86.27 %
Точность обучения модели на незашифрованных данных	86.27 %
Среднее время обучения модели на зашифрованных данных	346 секунд

Вес одного зашифрованного вектора составляет 429 Кбайт, в то время как файл, содержащий набор из 20000 строк, весит всего 30 407 Кбайт, что свидетельствует о значительном увеличении объема данных после шифрования, что может оказывать влияние на требования к хранению и передаче данных. Также время шифрования одного тензора, составляющее 59 секунд, демонстрирует то, что процесс шифрования может быть достаточно времязатратным, что необходимо учитывать при планировании операций обработки больших данных.

При этом точность модели логистической регрессии, обученной на зашифрованных данных, составила 86%, что совпадает с точностью модели, обученной на незашифрованных данных. Этот результат подтверждает эффективность применения технологии полностью гомоморфного шифрования для защиты конфиденциальности данных в процессе их обработки, без потери качества моделирования.

Однако среднее время обучения модели на зашифрованных данных составляет 346 секунд, что указывает на то, что обучение на зашифрованных

данных занимает больше времени по сравнению с обучением на незашифрованных данных. Однако с точки зрения обеспечения конфиденциальности и безопасности данных, затраты времени могут считаться оправданными.

Заключение

В данной работе рассматривалась проблема обеспечения безопасности данных, используемых банком для машинного обучения, так как утечка или раскрытие данных могут привести к репутационным потерям и штрафам за нарушение правил информационной безопасности.

В рамках проведенного исследования было осуществлено решение задачи кредитного скоринга путём разработки модели логистической регрессии, обучаемой на данных, зашифрованных с использованием технологии полностью гомоморфного шифрования. Для этого были проанализированы источники и литература по применению полностью гомоморфного шифрования для машинного обучения, в частности, было отмечено, что большинство работ посвящены решению задачи аппроксимации сигмоидной функции, поскольку полностью гомоморфное шифрование не позволяет вычислять напрямую функции, которые невозможно представить с помощью сложения и умножения. В результате проведённого анализа, было выяснено, что наилучшее приближение показывает полином третьей степени с коэффициентами 0.5, 0.197, 0 и -0.004.

Также во всех рассмотренных работах процесс шифрования данных не выделяется на отдельный этап, то есть в рамках клиент-серверной архитектуры загрузка и шифрование данных происходит на сервере, а не на клиенте, что подвергает данные определённым угрозам информационной безопасности, поэтому в данной работе путём разработки дополнительных функций encrypt_data() и get_ckks_vector() для библиотеки TenSEAL.

Так как полностью гомоморфное шифрование ограничивает количество операций умножения, то было выяснено, что для решения задачи кредитного

скоринга с помощью модели логистической регрессии требуется минимум 6 операций умножения, и в соответствии с этим были настроены параметры шифрования в виде степени полиномиального модуля, равной 8192 и двоичного размера, равного 21 бит, а в качестве схемы шифрования выбрана схема CKKS, так как она позволяет работать с вещественными числами, которыми и представлено большинство данных.

На клиенте данные были зашифрованы и сериализованы, после чего записаны в отдельные файлы, которые были архивированы в соответствии с шифруемым тензором, с помощью разработанной функции encrypt_data(). После загрузки этих архивов на сервер, сериализованные данные были восстановлены в зашифрованные векторы CKKSVector с помощью функции get_ckks_vector().

Модель логистической регрессии, обучаемая на зашифрованных данных, была определена с помощью PyTorch, и в качестве функции потери была выбрана бинарная потеря кросс-энтропии, однако с учётом использования L2 регуляризации для борьбы с переобучением, веса в ней были вычислены с учётом принятого коэффициента, равного 0.05.

В результате обучения данной модели была продемонстрирована точность в 86.27 %, что совпадает с точностью модели логистической регрессии, обученной на незашифрованных данных.

Таким образом, исследование показало, что использование полностью гомоморфного шифрования не вносит искажений в качество прогнозов, а также демонстрирует его применимость для сложных задач машинного обучения и открывает новые горизонты для проведения анализа данных с соблюдением высоких стандартов обеспечения безопасности данных, исключая возможность их раскрытия, так как на сервер загружаются уже зашифрованные данные.

Тем не менее, необходимо отметить, что использование полностью гомоморфного шифрования влечёт за собой дополнительные вычислительные затраты по сравнению с традиционными методами обработки данных, так как

кратно возрастает как время обучения, так и размер данных. Однако обеспечение безопасности данных клиентов для банка является ключевым аспектом, так как неспособность защитить личные данные может подорвать доверие клиентов и привести к их оттоку, а также утечка данных может привести к штрафам и репутационным потерям. Однако ни один из существующих способов защиты данных не может обеспечить такой уровень безопасности, как полностью гомоморфное шифрование, так как только оно позволяет производить вычисления над зашифрованными данными, не раскрывая их содержимое перед сервером.

Направления для дальнейшего исследования могут включать в себя оптимизацию процесса отправки данных на сервер и получения данных с него, так как сейчас это выполняется вручную. Кроме того, в данной работе сейчас происходит расшифровка и новое шифрование весов после каждой эпохи, что необходимо из-за ограниченного количества операций умножений. Однако данный процесс можно оптимизировать путём отправки весов обратно на клиент для расшифровки и повторного шифрования. В этом случае это приведет к передаче всего нескольких килобайт в эпоху.

Также высокие требования к времени обработки и объему хранимых данных указывают на необходимость дальнейших исследований и улучшений в области оптимизации процесса полностью гомоморфного шифрования, что может включать в себя разработку более эффективных алгоритмов шифрования, а также использование аппаратных решений для ускорения процессов.

Список литературы

1. Xu R., Nathalie Baracaldo N., Joshi J. Privacy-Preserving Machine Learning: Methods, Challenges and Directions [Электронный ресурс] // arXiv.org. 2024. Дата обновления: 22.09.2021. URL: https://arxiv.org/abs/2108.04417 (дата обращения: 05.03.2024).

2. Guerra-Manzanares A., L. Lopez J.L., Maniatakos M., Shamout F.Privacy-Preserving Machine Learning for Healthcare: Open Challenges and Future Perspectives. – First International Workshop, TML4H 2023, Virtual Event, 2023. – P. 25–40.

3. Варновский, Н. П., Шокуров, А. В. Гомоморфное шифрование. – Труды ИСП РАН. 2007.

4. Martins P., Sousa L., Mariano A. A survey on fully homomorphic encryption: An engineering perspective. – ACM Computing Surveys, Volume 50, Number 6, 2018. – P. 83:1–83:33.

5. Gilad-Bachrach R., Dowlin N., Laine K., Lauter E. K., Naehrig M., Wernsing J. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. – JMLR Workshop and Conference Proceedings, Volume 48. – P. 201–210.

6. Bourse F., Sanders O., Traoré J. Improved Secure Integer Comparison via Homomorphic Encryption. In Topics in Cryptology. – The Cryptographers' Track at the RSA Conference 2020, Lecture Notes in Computer Science, Volume 12006. – P. 391–416.

7. Crawford J. L. H., Gentry C., Halevi S., Platt D., Shoup V. Doing real work with FHE: The case of logistic regression. – Proc. 6th Workshop Encrypted Comput. Appl. Homomorphic Cryptogr, 2018. – 1–12.

8. Hesamifard E., Takabi H., Ghasemi M. CryptoDL: Deep neural networks over encrypted data. [Электронный ресурс] // arXiv.org. 2024. Дата обновления: 01.06.2017. URL: https://arxiv.org/abs/1711.05189 (дата обращения: 05.03.2024).

9. Chabanne H., Wargny A., Milgram J., Morel C., Prouff E. Privacy-preserving classification on deep neural network. – IACR Cryptol. ePrint Arch, 2017. – P. 1–35.

10. Маршалко Г. Б., Труфанова Ю. А., Полиномиальные аппроксимации некоторых функций активации нейронных сетей, Информатика и автоматизация, 2022, выпуск 21, том 1, 161–180

11. Ioffe S., Szegedy C., Batch normalization: Accelerating deep network training by reducing internal covariate shift [Электронный ресурс] // arXiv.org. 2024. Дата обновления: 03.01.2016. URL: https://arxiv.org/abs/1502.03167 (дата обращения: 05.03.2024).

12. Nandakumar K., Ratha N., Pankanti S., Halevi S. Towards Deep Neural Network Training on Encrypted Data. – CVPR Workshops, 2019. – P. 40–48.

13. Vizitiu A., Nita C. I., Puiu A., Suciu C., Itu L. M. Applying Deep Neural Networks over Homomorphic Encrypted Medical Data [Электронный ресурс] // doi.org. 2024. Дата обновления: 12.04.2020. URL: https://doi.org/10.1155/2020/3910250 (дата обращения: 05.03.2024).

14. Kahya A. Machine Learning over Encrypted Data With Fully Homomorphic Encyption. – Master of Science, Middle East Technical University. 2022.

15. Barni M., Orlandi C., Piva A. A privacypreserving protocol for neural-network-based computation. – 8th Workshop on Multimedia and Security, 2006. – P. 146–151.

16. Vaikuntanathan C. J. V., Chandrakasan A. GAZELLE: a low latency framework for secure neural network inference [Электронный ресурс] // usenix.org. 2024. Дата обновления: 10.08.2018. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar (дата обращения: 05.03.2024).

17. Liu J., Juuti M., Lu Y., Asokan N. Oblivious neural network predictions via minionn transformations. ACM SIGSAC Conference on Computer and Communications Security, October 30 – November 03, 2017. – P. 619–631.

18. Mohassel P., Zhang Y. Secureml: A system for scalable privacy-preserving machine learning. – IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017. – P. 19–38.

19. Riazi M. S., Weinert C., Tkachenko O., Songhori E. M., Schneider T., Koushanfar F. Chameleon: A hybrid secure computation framework for machine

learning applications. – 13 ACM Asia Conference on Information, Computer and Communications Security, ACM, Jun 4–8, 2018.

20. Chen H, Gilad-Bachrach R., Han K., Huang Z., Jalali A., Laine K., Lauter K. Logistic regression over encrypted data from fully homomorphic encryption. – BMC Medical Genomics, Volume 11, Article number 81. – P. 56–67.