Отчет о практическом задании «Градиентные методы обучения линейных моделей. Применение линейных моделей для определения токсичности комментария».

Практикум 317 группы, ММП ВМК МГУ.

Булкин Антон Павлович.

Ноябрь 2024.

Содержание

1 Вступление

Данное задание посвящено ознакомлению с линейными моделями и градиентными методами обучения на примере задачи классификации токсичности комментария при помощи модели линейного классификатора.
Задачами задания являлись:

1. Реализовать линейный классификатор с произвольной функцией потерь на языке Python без использования библиотеки `scikit-learn`.

2. Реализовать *oracle*, использующий логистическую функцию потерь.

3. Вывод необходимых формул и теоретических выкладок для понимания работы алгоритма.

4. Выполнить 7 экспериментов на датасете токсичных комментариев в рамках решения задачи бинарной классификации.

5. Провести предварительную обработку и преобразование выборки для упрощения решения задачи классификации.

2 Теоритическая часть

2.1 Формулы логистической регрессии

Линейная модель классификации определяется как: $a(x) = \text{sign}(\langle w, x\rangle + b)$, где $w \in R^d$ — вектор весов, $b \in R$ — сдвиг.

Величина $M_i(w, b)$, называемая отступом объекта x_i относительно алгоритма $a(x)$, определяется как: $M_i(w, b) = y_i(\langle w, x_i\rangle + b)$.

Формула функции потерь логистической регрессии имеет вид:
$\mathcal{L}(M) = \log(1 + \exp(-M)) = \log(1 + \exp(-(\langle w, x\rangle + b)))$.

Тогда функция потерь выглядит: $Q(X, w) = \frac{1}{l}\sum_{i=1}^{l} \log\left(1 + e^{-y_i(w,x_i)}\right)$

2.2 Вывод формулы градиента функции потерь для задачи бинарной логистической регрессии

$$\nabla_w Q(X, w) = \nabla_w \left(\frac{1}{l}\sum_{i=1}^{l} \log\left(1 + e^{-y_i(w,x_i)}\right)\right)$$

Найдем производные по w_j:

$$\frac{\partial}{\partial w_j}(y_i(\langle w, x_i\rangle + b)) = y_i x_{ij}$$

$$\frac{\partial}{\partial w_j} L(w, x_i, b) = \frac{\partial}{\partial w_j}\left(\log\left(1 + e^{-y_i(\langle w,x_i\rangle+b)}\right)\right) = \frac{-e^{-y_i(\langle w,x_i\rangle+b)}}{1 + e^{-y_i(\langle w,x_i\rangle+b)}} \cdot \frac{\partial}{\partial w_j}(y_i(\langle w, x_i\rangle+b))$$

$$\frac{\partial}{\partial w_j} Q(X, w) = \frac{\partial}{\partial w_j}\left(\frac{1}{l}\sum_{i=1}^{l} \log\left(1 + e^{-y_i\langle w,x_i\rangle}\right)\right) = \frac{1}{l}\sum_{i=1}^{l} (y_i x_{ij}) \cdot \frac{-e^{-y_i\langle w,x_i\rangle}}{1 + e^{-y_i\langle w,x_i\rangle}} =$$

$$= -\frac{1}{l}\sum_{i=1}^{l} \frac{y_i x_{ij}}{1 + e^{y_i(w,x_i)}}$$

Следовательно, $\nabla_w Q(X, w) = -\frac{1}{l}\sum_{i=1}^{l} \frac{y_i x_i}{1+e^{y_i\langle w,x_i\rangle}}$.

2.3 Вывод формулы градиента функции потерь для задачи многоклассовой (мультиномиальной) логистической регрессии

Функция потерь для задачи многоклассовой (мультиномиальной) логистической регрессии: $(w \in \mathbb{R}^{d\times K})$

$$L = -\frac{1}{l}\sum_{i=1}^{l} \ln P(y_i \mid x_i),$$

где $P(y = j \mid x) = \frac{e^{\langle w_j x\rangle}}{\sum_{k=1}^{K} e^{\langle w_k, x\rangle}}$.

Посчитаем частные производные: $\frac{\partial L}{\partial w_j} = -\frac{1}{l}\sum_{i=1}^{l}\frac{1}{P_i}\frac{\partial P_i}{\partial w_j} = -\frac{1}{l}\sum_{i=1}^{l}\frac{1}{P_i}\frac{\partial}{\partial w_j}(\frac{e^{\langle w_{y_i}x_i\rangle}}{\sum_{k=1}^{K}e^{\langle w_k,x_i\rangle}}) =$
$= -\frac{1}{l}\sum_{i=1}^{l}\frac{1}{P_i}(\frac{x_i e^{\langle w_j x_i\rangle}\sum_{k=1}^{K}e^{\langle w_k,x_i\rangle} - e^{\langle w_{y_i},x_i\rangle}e^{\langle w_j,x_i\rangle}}{(\sum_{k=1}^{K}e^{\langle w_k,x_i\rangle})^2}) = -\frac{1}{l}\sum_{i=1}^{l}(x_i - x_i\cdot\frac{e^{\langle w_j,x_i\rangle}}{\sum_{k=1}^{K}e^{\langle w_k,x_i\rangle}}) =$
$= -\frac{1}{l}\sum_{i=1}^{l}(x_i - P(y=j \mid x_i)\cdot x_i)$

3 Обработка данных

3.1 Предварительная обработка текста

Проведем предварительную обработку текста комментариев:

1. Приведем все тексты к нижнему регистру при помощи *str.lower()*
2. Заменим все символы, не являющиеся цифрами или буквами, на пробелы

3.2 Преобразование выборки

Проведем преобразование выборки в разреженную матрицу *scipy.sparse.csr_matrix*, где значению в (i, j) позиции соответствует число встреченных слов j в документе i, при помощи *sklearn.feature_extraction.text.CountVectorizer*, который реализует модель представления текста *Bag of Words*, в которой каждый текстовый документ представляется вектором, который отражает частоты слов, присутствующих в документе. Столбцы вектора соответствуют уникальным словам из всего корпуса документов. При применении *CountVectorizer* укажем параметр *min_df*, равный 0.001, чтобы уменьшить размерность данных и ускорить проведение экспериментов.

4 Сравнение численного подсчета градиента функции потерь с вычислением по аналитической формуле

4.1 Постановка задачи

Сравните численный подсчет градиента функции потерь из модуля utils.py с вычислением по аналитической формуле.

4.2 Реализация

Для сравнения алгоритмов подсчета градиента функции потерь сгенерируем выборки размера $(500000, x)$, где $x \in \{5, 25, 75, 100, 125, 175, 250, 350, 500, 1000\}$,

чтобы сравнить работу алгоритмов на данных с разными размерами признакового пространства.

На рисунках 1 и 2 представлены графики зависимости времени работы алгоритмов и значения RMSE (евклидовой нормы вектора разности результатов работы алгоритмов, деленной на $\sqrt{n}$) от размерности признакового пространства выборки.

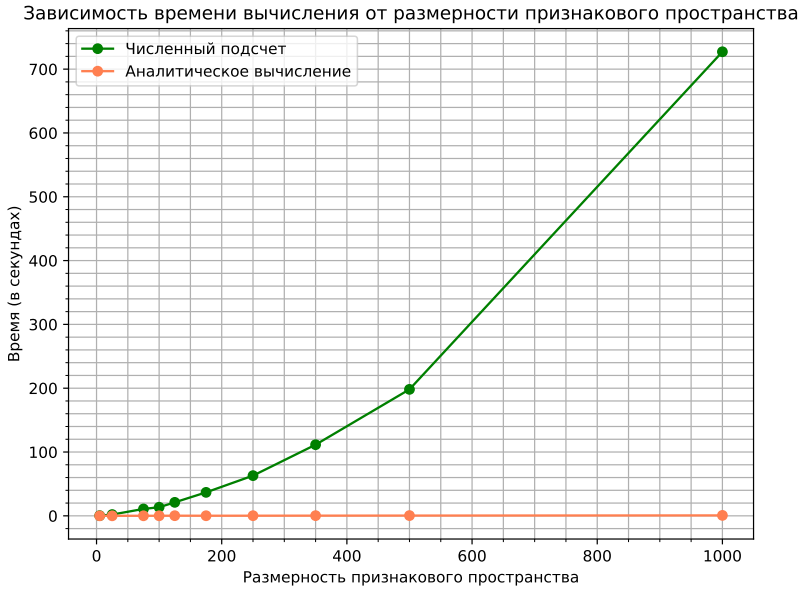


Рис. 1: Время работы алгоритмов подсчета градиента функции в зависимости от размера признакового пространства

Рис. 2: Значение RMSE в зависимости от размера признакового пространства

При сравнении аналитического и численного вычисления градиента функции потерь использовался лосс для бинарной логистической задачи классификации (*Binary Log-Loss*), т.е. $Q(X,w) = \frac{1}{l}\sum_{i=1}^{l}\log\left(1+e^{y_i<\mathbf{w},\mathbf{x}_i>}\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$.

Аналитический метод подсчета градиента выполнялся по формуле:
$\nabla_{\mathbf{w}} Q(\mathbf{X},\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \cdot y_i \cdot \sigma\left(-y_i \cdot (\mathbf{w}^\top \mathbf{x}_i)\right) + \lambda\mathbf{w}$

Численный же метод подсчета градиента представлял собой:
$\nabla_{\mathbf{w}} Q(X,w) = \left(\frac{\partial}{\partial w_0}, \frac{\partial}{\partial w_1}, \ldots, \frac{\partial}{\partial w_l}\right),$

где $\frac{\partial}{\partial w_j} = \frac{Q(X,w+\epsilon*\mathbf{e}_j)-Q(X,w)}{\epsilon}$, $e_{ij} = \begin{cases}1, & \text{если } i=j,\\ 0, & \text{если } i\neq j,\end{cases}$ $\epsilon = 10^{-8}$.

4.3 Выводы

Исследование показало, что алгоритм численного подсчета лишь немного уступает аналитическому методу в вычислении градиента функции потерь, так как значение RMSE методов почти не превосходит 10^{-5}.

Однако, как можно заметить из графиков, время вычисления градиента функции потерь значительно растет с ростом размерности признакового пространства, в отличие от аналитического метода вычисления, скорость выполнения которого практически не изменяется с ростом размерности признакового пространства.

Из перечисленного выше, можно сделать вывод, что аналитический метод работает эффективнее по времени, поэтому в дальнейшим будем использовать его.

5 Исследование поведения градиентного спуска для задачи логистической регрессии в зависимости от *step_ alpha*, *step_ beta* и начального приближения

5.1 Постановка задачи

Исследовать поведение градиентного спуска для задачи логистической регрессии в зависимости от следующих параметров:

- параметр размера шага `step_alpha`
- параметр размера шага `step_beta`
- начального приближения

Исследование поведения метода подразумевало анализ следующих зависимостей:

- зависимость значения функции потерь от итерации метода
- зависимость точности (accuracy) итерации метода

5.2 Реализация

Во всех экспериментах до подбора наилучшего параметра при прочих равных увсловиях брались как тестирующие значения параметры:

- $step\_beta = 0$
- $l2\_coef = 0.001$
- $max\_iter = 1000$

После нахождения наилучшего параметра - использовался он.

5.2.1 Подбор параметра *step_ alpha*

Будем рассматривать значения $step\_alpha$, равномерно расположенные по логарифмической шкале от 10^{-4} до 10.

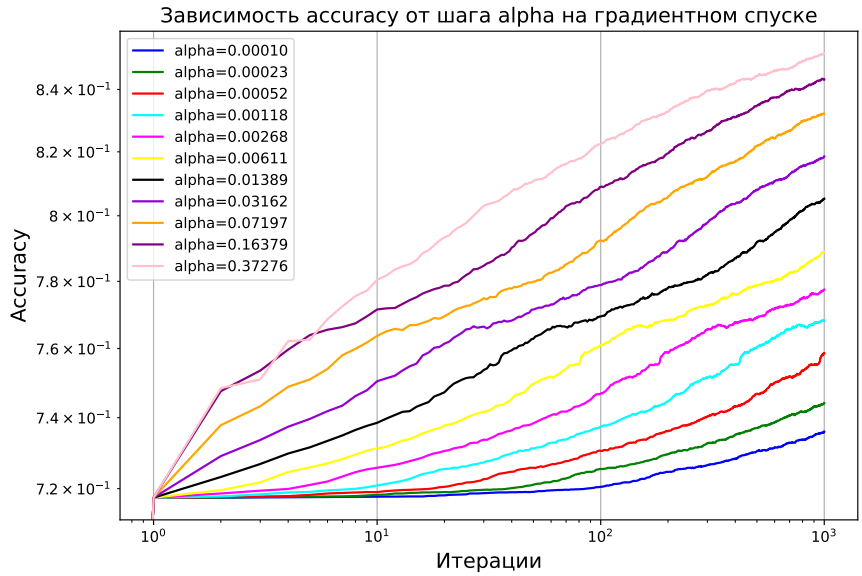


Рис. 3: Accuracy на итерациях модели в зависимости от параметра *step_alpha*

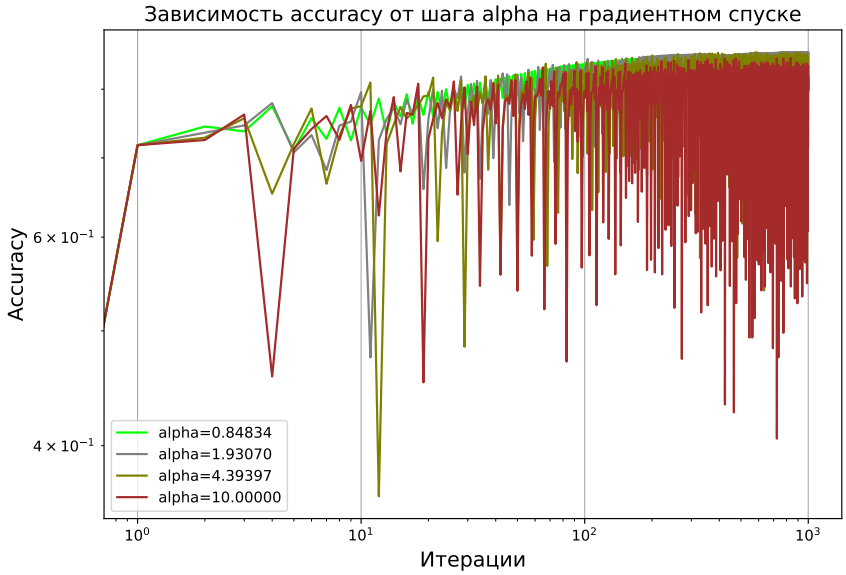


Рис. 4: Accuracy на итерациях модели в зависимости от параметра *step_alpha*

На рисунках 3 и 4 представлены графики зависимости accuracy модели на каждой итерации градиентного спуска от параметра *step_alpha*. Как можно заметить из графиков, наилучшая точность модели достигается при *step_alpha* = *0.372759372*.

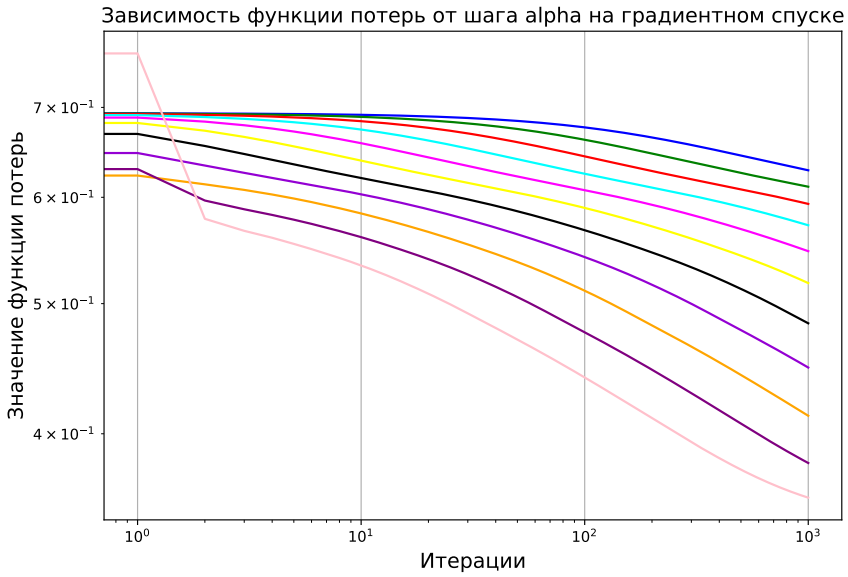


Рис. 5: Значение функции потерь на итерациях модели в зависимости от параметра *step_alpha*

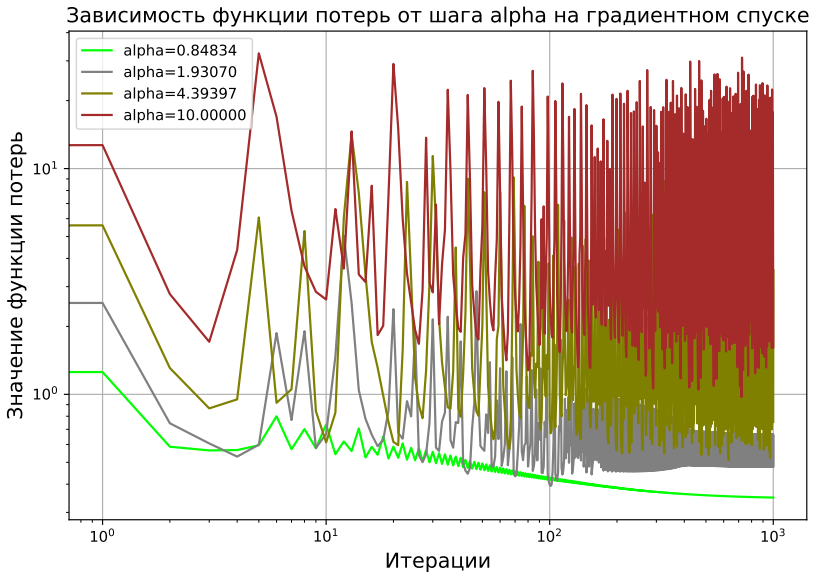


Рис. 6: Значение функции потерь на итерациях модели в зависимости от параметра *step_alpha*

На рисунках 5 и 6 представлены графики зависимости значения функции потерь на каждой итерации градиентного спуска от параметра *step_alpha*. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при *step_alpha* = *0.372759372*.

5.2.2 Подбор параметра *step_ beta*

Будем рассматривать значения *step_ beta* среди следующего множества значений: {0, 0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, 2, 3, 4, 5, 10, 15, 25, 50, 75, 100}.

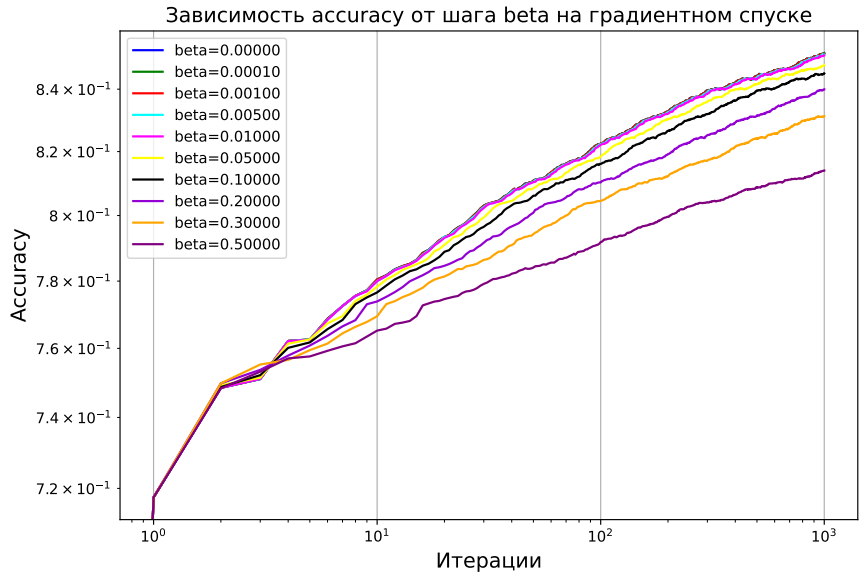


Рис. 7: Accuracy на итерациях модели в зависимости от параметра *step_ beta*

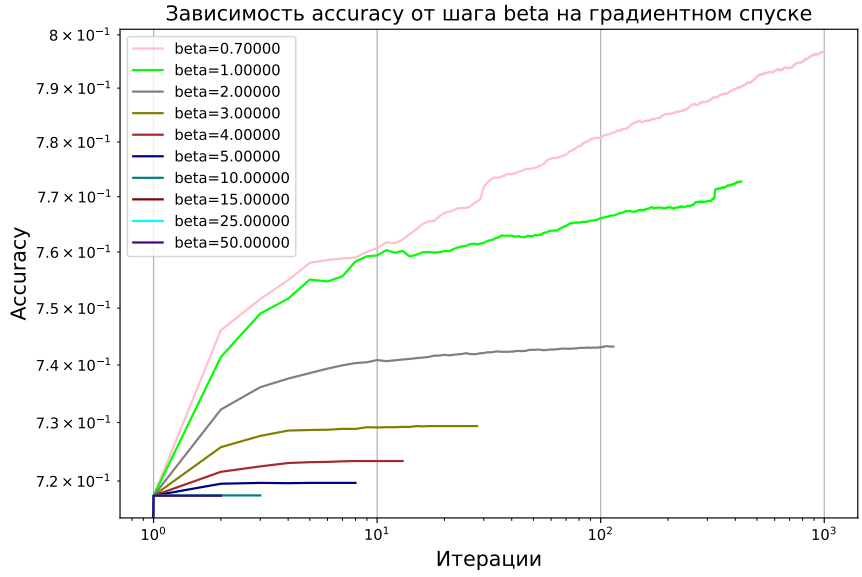


Рис. 8: Accuracy на итерациях модели в зависимости от параметра *step_ beta*

На рисунках 7 и 8 представлены графики зависимости accuracy модели на каждой итерации градиентного спуска от параметра *step_ beta*. Как можно заметить

из графиков, наилучшая точность модели достигается при $step\_beta = 0$.

Также можно заметить, что часть графиков на рисунке 8 имеют отличный характер зависимости от других, так как алгоритм градиентного спуска заканчивается раньше, чем проходит все итерации, так как модуль разницы последовательных на итерациях значений функции потерь становится меньше, чем *tolerance*, что приводит к остановке алгоритма.

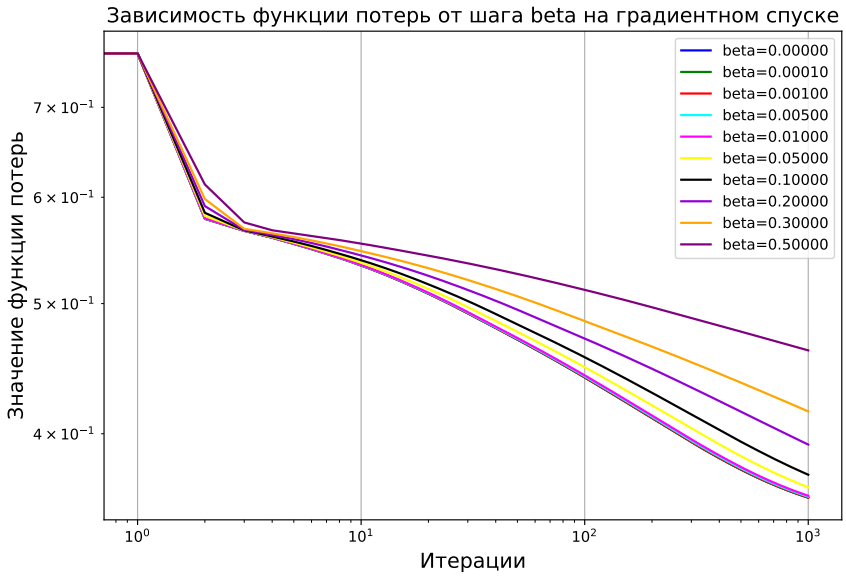


Рис. 9: Значение функции потерь на итерациях модели в зависимости от параметра $step\_beta$

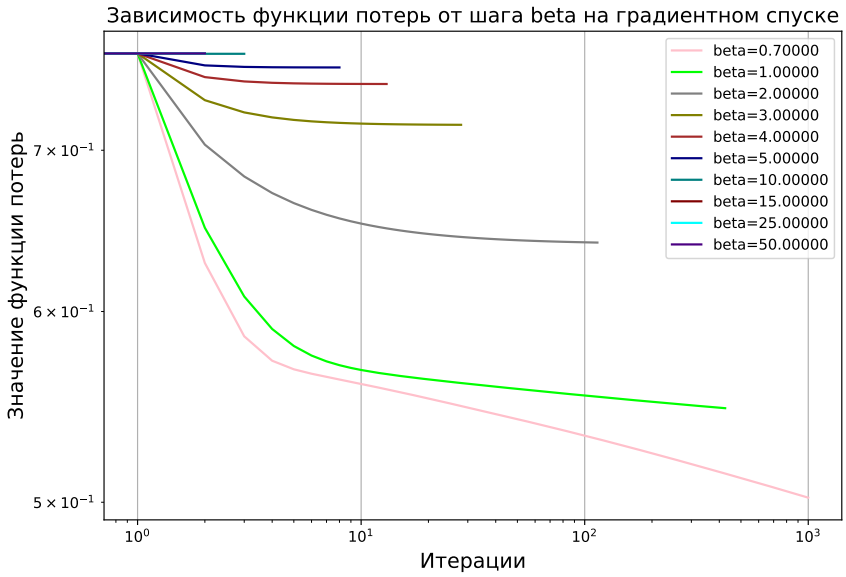


Рис. 10: Значение функции потерь на итерациях модели в зависимости от параметра *step_ beta*

На рисунках 9 и 10 представлены графики зависимости значения функции потерь на каждой итерации градиентного спуска от параметра *step_ beta*. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при *step_ beta = 0*, причём алгоритм проходит все итерации, а точность (accuracy) оказывается выше, чем при других значениях *step_ beta*.

5.2.3 Подбор начального приближения весов w_0

Будем рассматривать значения начального приближения весов w_0, используя функции *Numpy*, такие как *np.random.normal* и *np.random.uniform*, соответствующие нормальному и равномерному распределениям. Рассмотрим значения начального приближения весов, соответствующие данным распределениям со следующими параметрами:

- normal 0, 0.1
- normal 0, 0.25
- normal 0, 0.5
- normal 0, 0.75
- normal 0, 1
- uniform -0.1, 0.1

- uniform -0.5, 0.5
- uniform -1, 1
- uniform -2, 2
- uniform -5, 5
- uniform -10, 10
- uniform -50, 50

А также начальное приближение весов: нулями и по формуле: $w_j = \frac{\langle f_j, y \rangle}{\langle f_j, f_j \rangle}$, где y — вектор меток, f_j — столбец признака, а w_j - j-й элемент вектора весов w_0

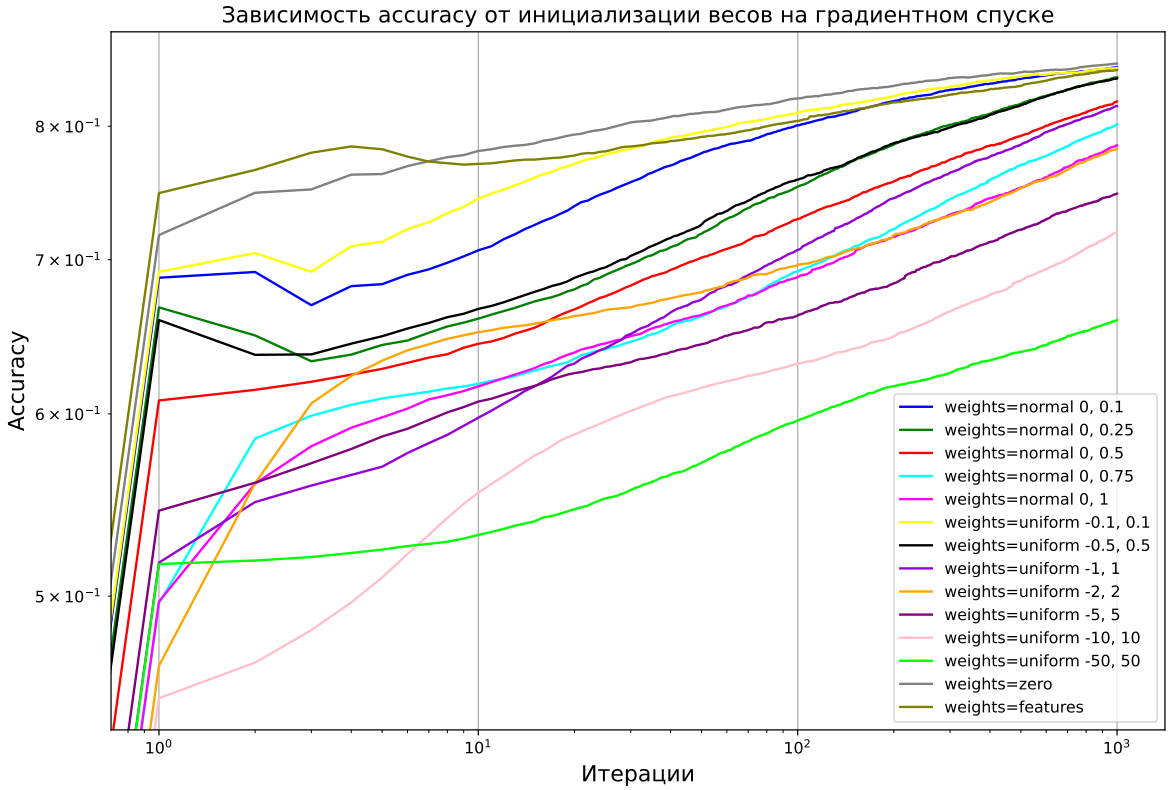


Рис. 11: Accuracy на итерациях модели в зависимости от начального приближения весов

На рисунке 11 представлены графики зависимости accuracy модели на каждой итерации градиентного спуска от начального приближения весов w_0. Как можно заметить из графиков, наилучшая точность модели достигается при $w_0 = (0, ..., 0)$.

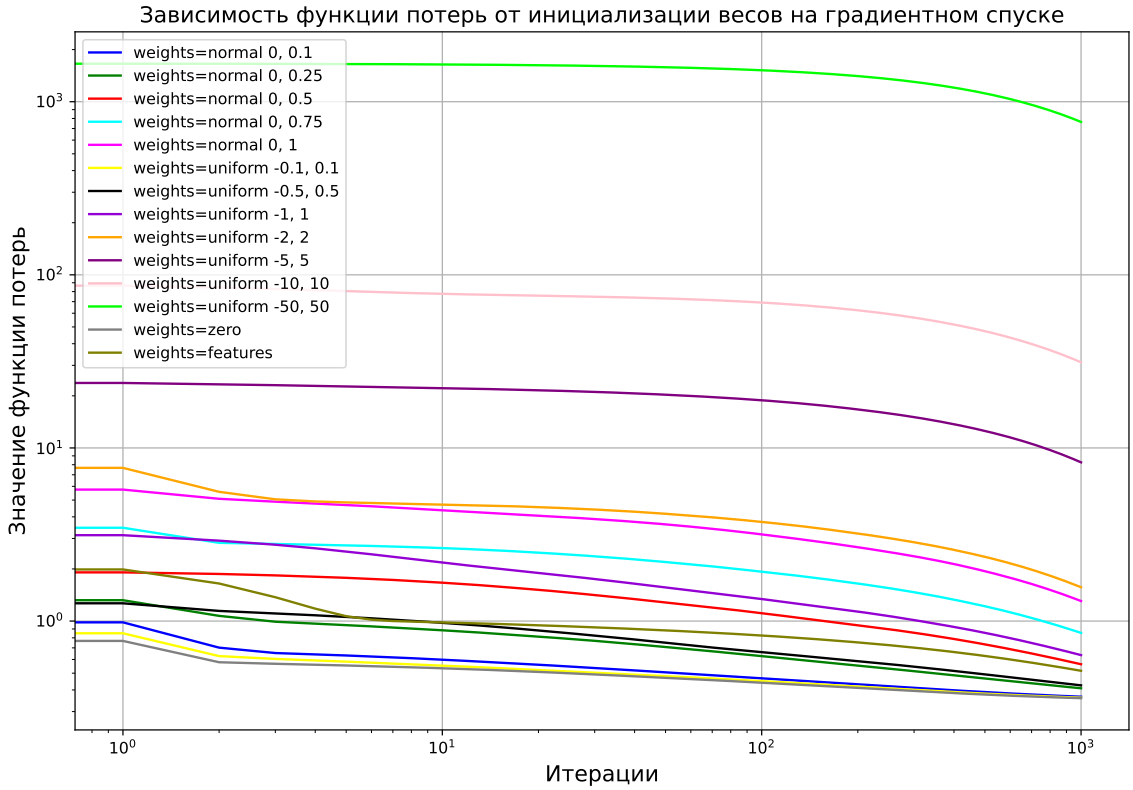


Рис. 12: Значение функции потерь на итерациях модели в зависимости от начального приближения весов

На рисунке 12 представлены графики зависимости значения функции потерь на каждой итерации градиентного спуска от начального приближения весов w_0. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при $w_0 = (0, ..., 0)$.

5.2.4 Подбор коэффициента L_2-регуляризации

Будем рассматривать значения *l2_coef*, равномерно расположенные по логарифмической шкале от 10^{-4} до 10^4.

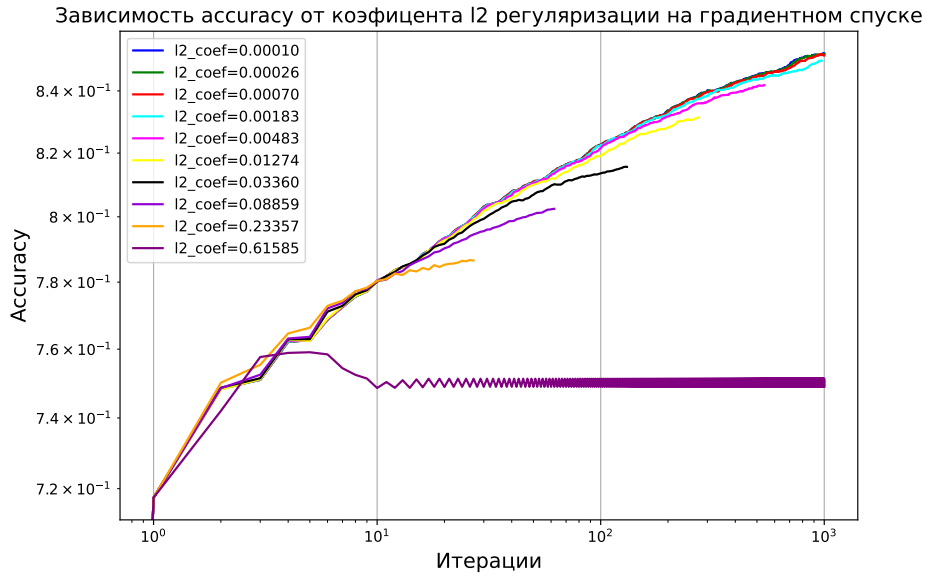


Рис. 13: Accuracy на итерациях модели в зависимости от параметра *l2_ coef*

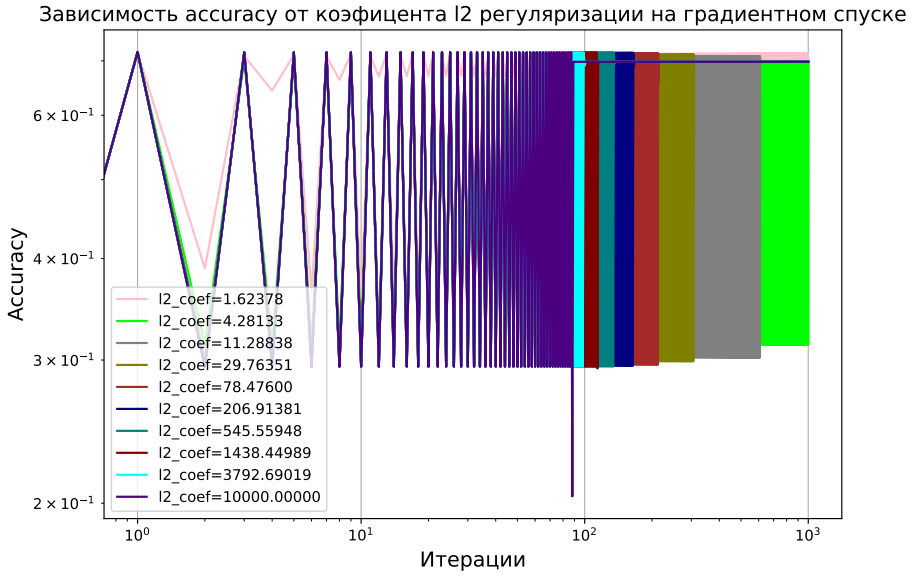


Рис. 14: Accuracy на итерациях модели в зависимости от параметра *l2_ coef*

На рисунках 13 и 14 представлены графики зависимости accuracy модели на каждой итерации градиентного спуска от параметра *l2_ coef*. Как можно заметить из графиков, наилучшая точность модели достигается при *l2_ coef* = *0.0001*.

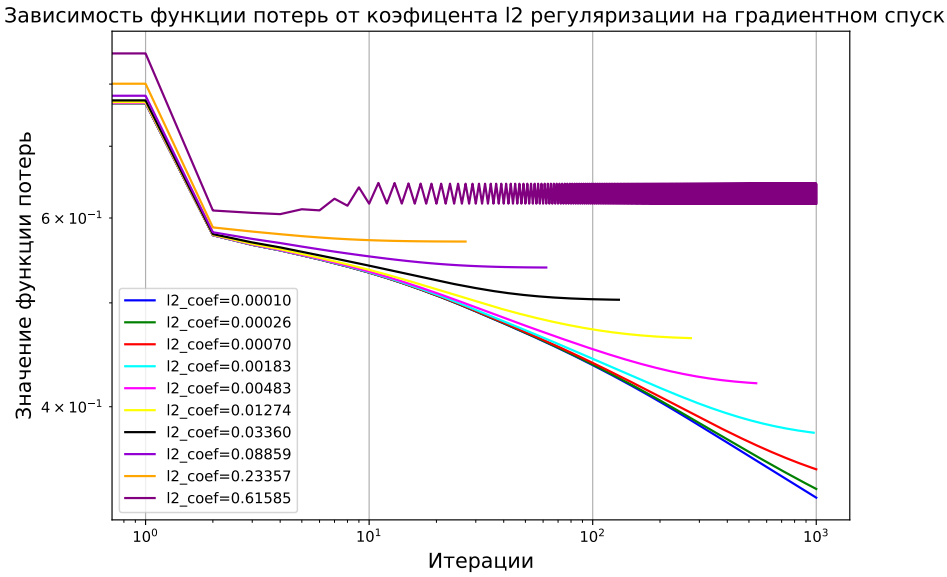


Рис. 15: Значение функции потерь на итерациях модели в зависимости от параметра *l2_ coef*

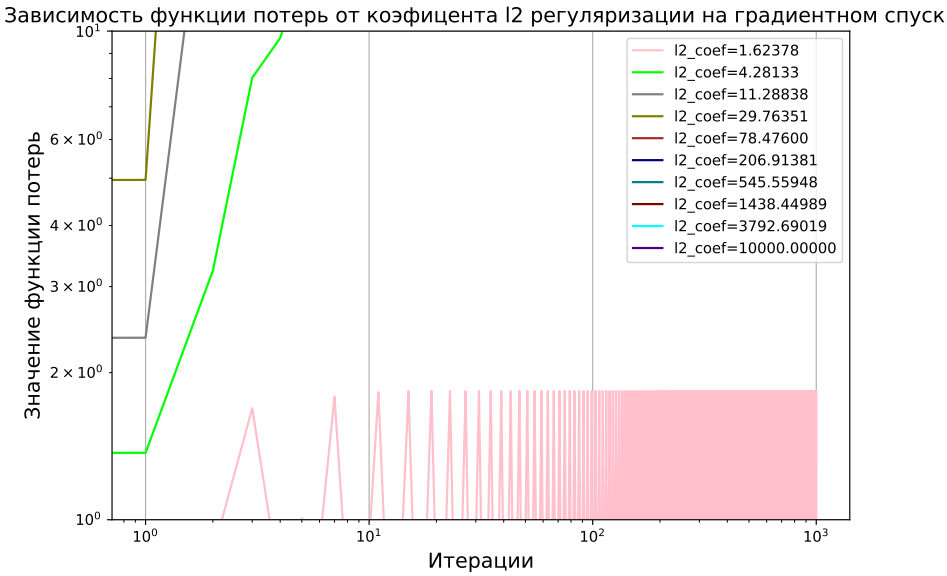


Рис. 16: Значение функции потерь на итерациях модели в зависимости от параметра *l2_ coef*

На рисунках 15 и 16 представлены графики зависимости значения функции потерь на каждой итерации градиентного спуска от параметра *l2_ coef*. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при *l2_ coef = 0.0001*.

5.3 Выводы

Проведенные эксперименты показали следующие лучшие параметры градиентного спуска для задачи логистической регрессии при прочих равных условиях:

- *step_ alpha = 0.372759372*
- *step_ beta = 0*
- $w_0 = (0, ..., 0)$
- *l2_ coef = 0.0001*

6 Исследование поведения стохастического градиентного спуска для задачи логистической регрессии в зависимости от *step_ alpha*, *step_ beta* и начального приближения

6.1 Постановка задачи

Исследовать поведение стохастического градиентного спуска для задачи логистической регрессии в зависимости от следующих параметров:

- параметр размера шага `step_alpha`
- параметр размера шага `step_beta`
- начального приближения
- размера подвыборки *batch_ size*

Исследование поведения метода подразумевало анализ следующих зависимостей:

- зависимость значения функции потерь от эпохи метода
- зависимость точности (accuracy) эпохи метода

6.2 Реализация

Во всех экспериментах до подбора наилучшего параметра при прочих равных увсловиях брались как тестирующие значения параметры:

- $step\_beta = 0$
- $batch\_size = 500$
- $l2\_coef = 0.001$
- $max\_iter = 1000$

После нахождения наилучшего параметра - использовался он.

6.2.1 Подбор параметра *step_alpha*

Будем рассматривать значения $step\_alpha$, равномерно расположенные по логарифмической шкале от 10^{-4} до 10.

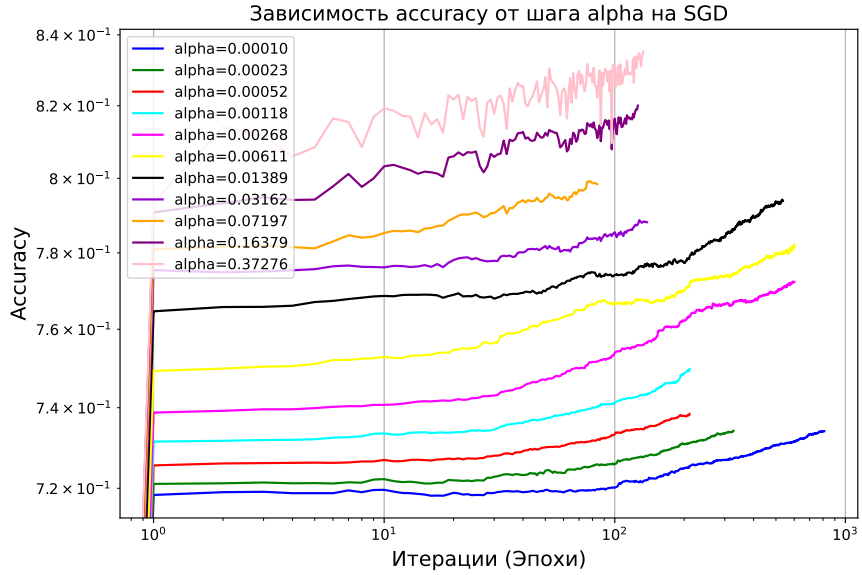


Рис. 17: Accuracy на эпохах модели в зависимости от параметра $step\_alpha$

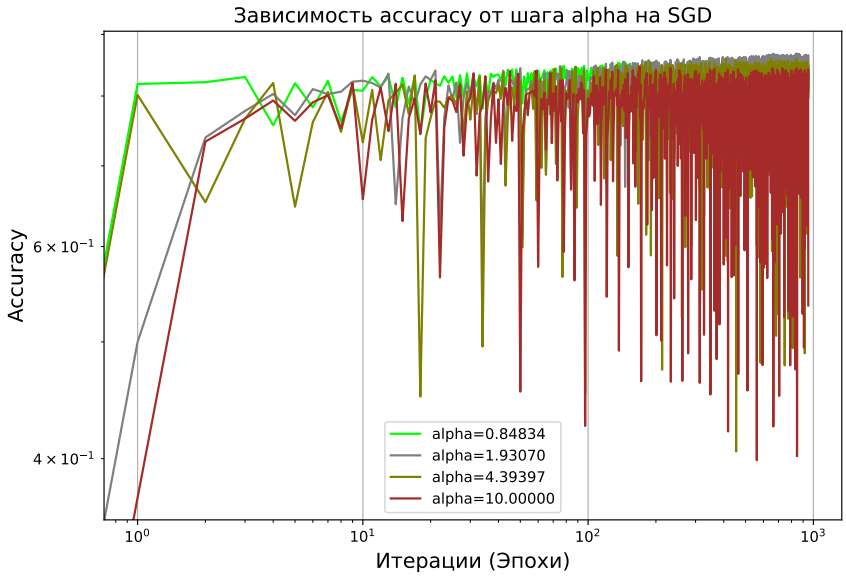


Рис. 18: Accuracy на эпохах модели в зависимости от параметра *step_alpha*

На рисунках 17 и 18 представлены графики зависимости accuracy модели на каждой эпохе стохастического градиентного спуска от параметра *step_alpha*. Как можно заметить из графиков, наилучшая точность модели достигается при *step_alpha* = *0.372759372*.

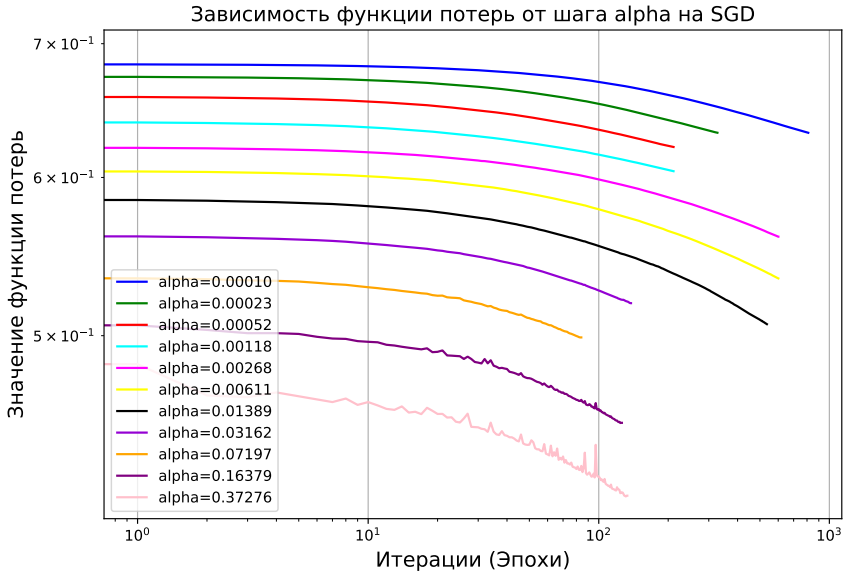


Рис. 19: Значение функции потерь на эпохах модели в зависимости от параметра *step_alpha*

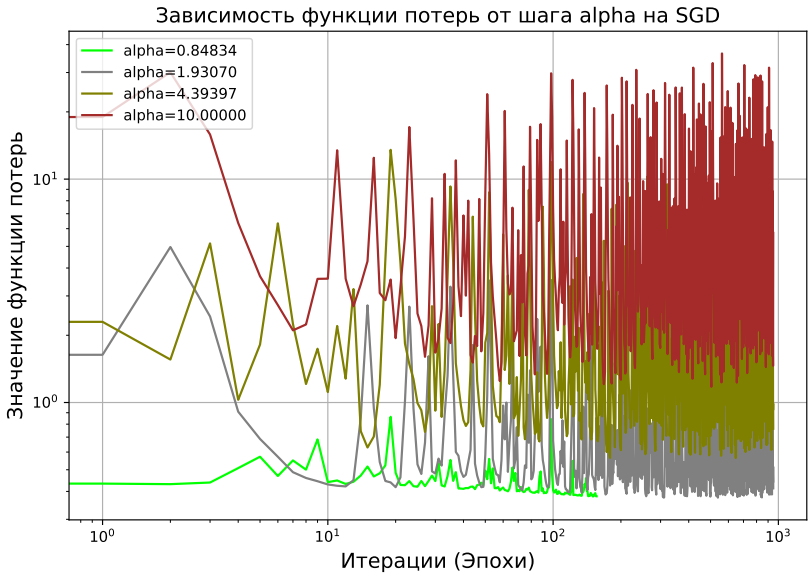


Рис. 20: Значение функции потерь на эпохах модели в зависимости от параметра *step_ alpha*

На рисунках 19 и 20 представлены графики зависимости значения функции потерь на каждой эпохе стохастического градиентного спуска от параметра *step_ alpha*. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при *step_ alpha = 0.372759372*.

6.2.2 Подбор параметра *step_ beta*

Будем рассматривать значения *step_ beta* среди следующего множества значений: {0, 0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, 2, 3, 4, 5, 10, 15, 25, 50, 75, 100}.

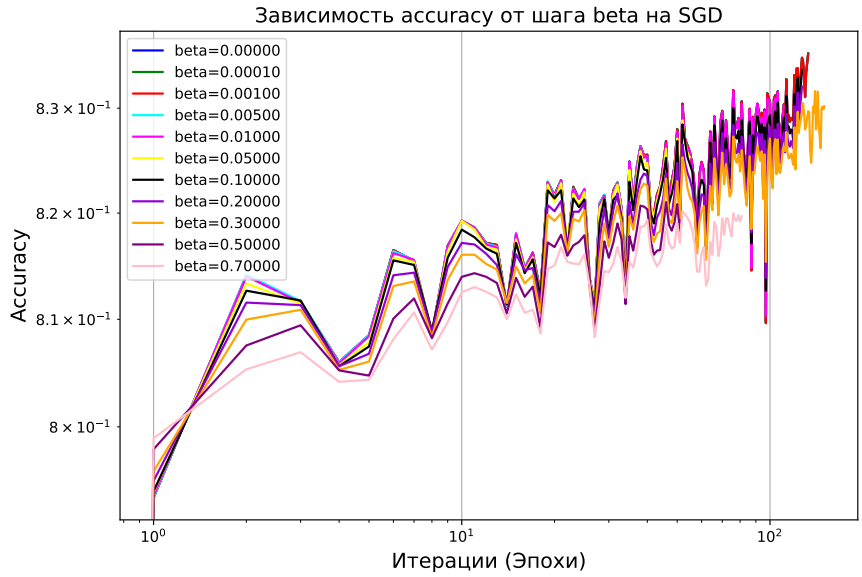


Рис. 21: Accuracy на эпохах модели в зависимости от параметра *step_beta*

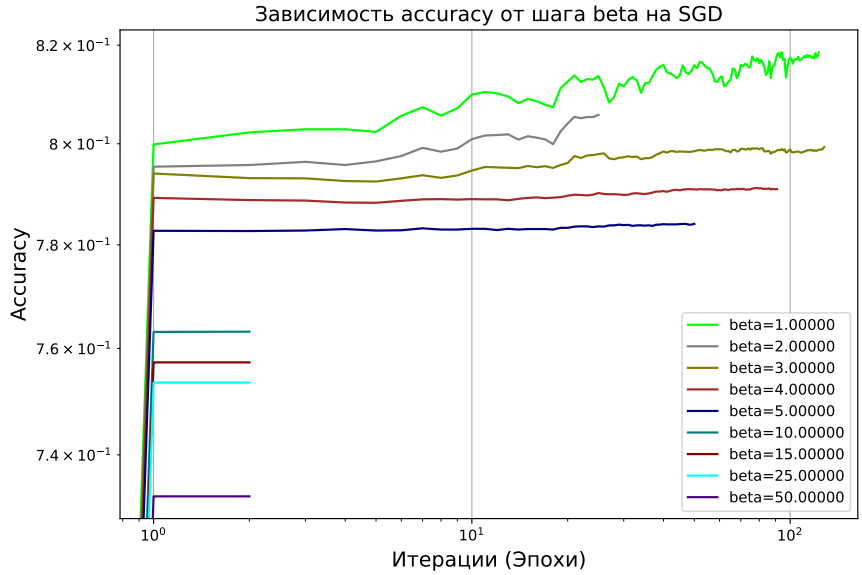


Рис. 22: Accuracy на эпохах модели в зависимости от параметра *step_beta*

На рисунках 21 и 22 представлены графики зависимости accuracy модели на каждой эпохе стохастического градиентного спуска от параметра *step_beta*. Как можно заметить из графиков, наилучшая точность модели достигается при *step_beta = 0.001*.

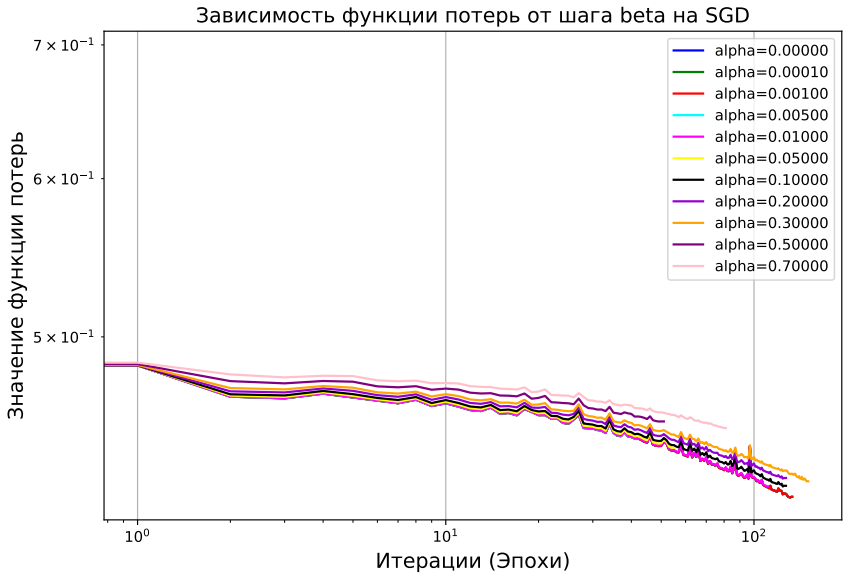


Рис. 23: Значение функции потерь на эпохах модели в зависимости от параметра *step_ beta*

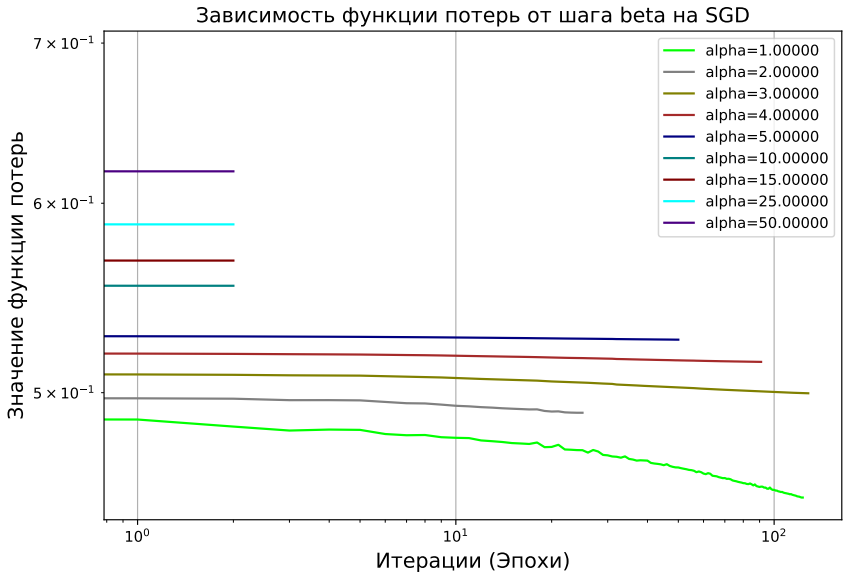


Рис. 24: Значение функции потерь на эпохах модели в зависимости от параметра *step_ beta*

На рисунках 23 и 24 представлены графики зависимости значения функции потерь на каждой эпохе стохастического градиентного спуска от параметра *step_ beta*. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при *step_ beta = 0.001*.

6.2.3 Подбор начального приближения весов w_0

Будем рассматривать значения начального приближения весов w_0, используя функции *Numpy*, такие как *np.random.normal* и *np.random.uniform*, соответствующие нормальному и равномерному распределениям. Рассмотрим значения начального приближения весов, соответствующие данным распределениям со следующими параметрами:

- normal 0, 0.1
- normal 0, 0.25
- normal 0, 0.5
- normal 0, 0.75
- normal 0, 1
- uniform -0.1, 0.1
- uniform -0.5, 0.5
- uniform -1, 1
- uniform -2, 2
- uniform -5, 5
- uniform -10, 10
- uniform -50, 50

А также начальное приближение весов: нулями и по формуле: $w_j = \frac{\langle f_j, y \rangle}{\langle f_j, f_j \rangle}$, где y — вектор меток, f_j — столбец признака, а w_j - j-й элемент вектора весов w_0

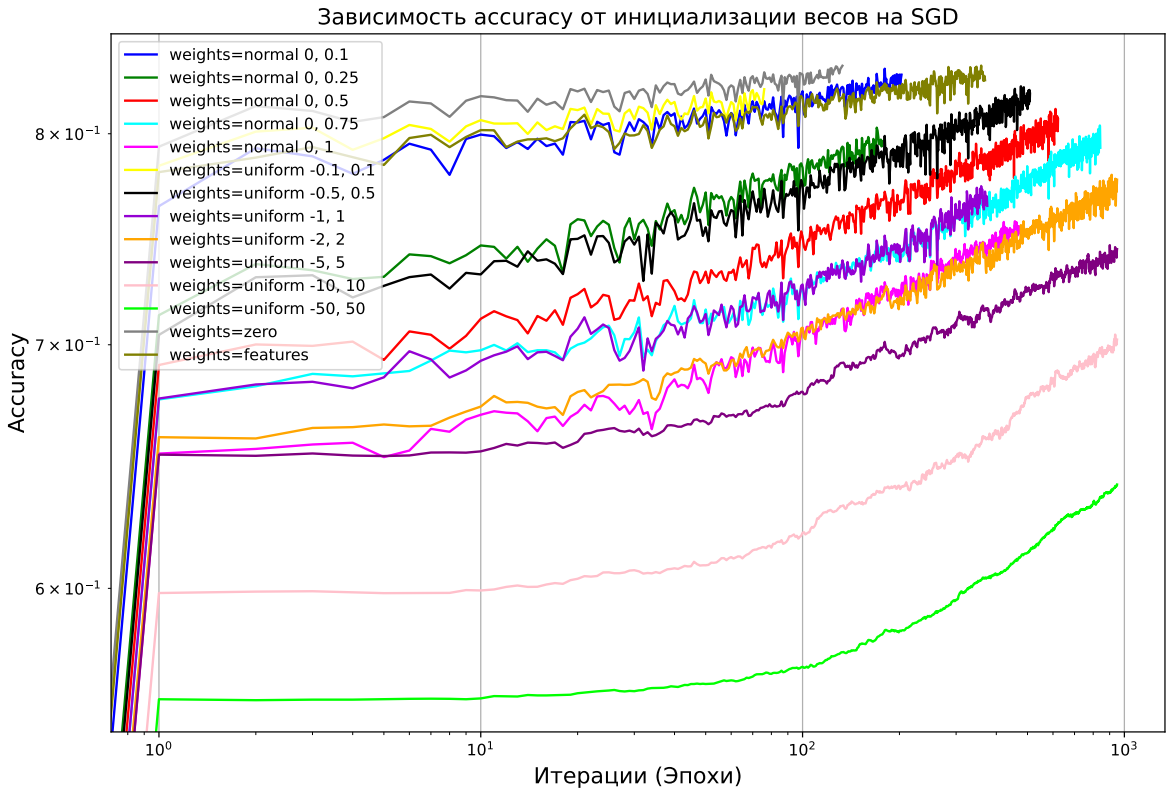


Рис. 25: Accuracy на эпохах модели в зависимости от начального приближения весов

На рисунке 25 представлены графики зависимости accuracy модели на каждой эпохе стохастического градиентного спуска от начального приближения весов w_0. Как можно заметить из графиков, наилучшая точность модели достигается при $w_0 = (0, ..., 0)$.

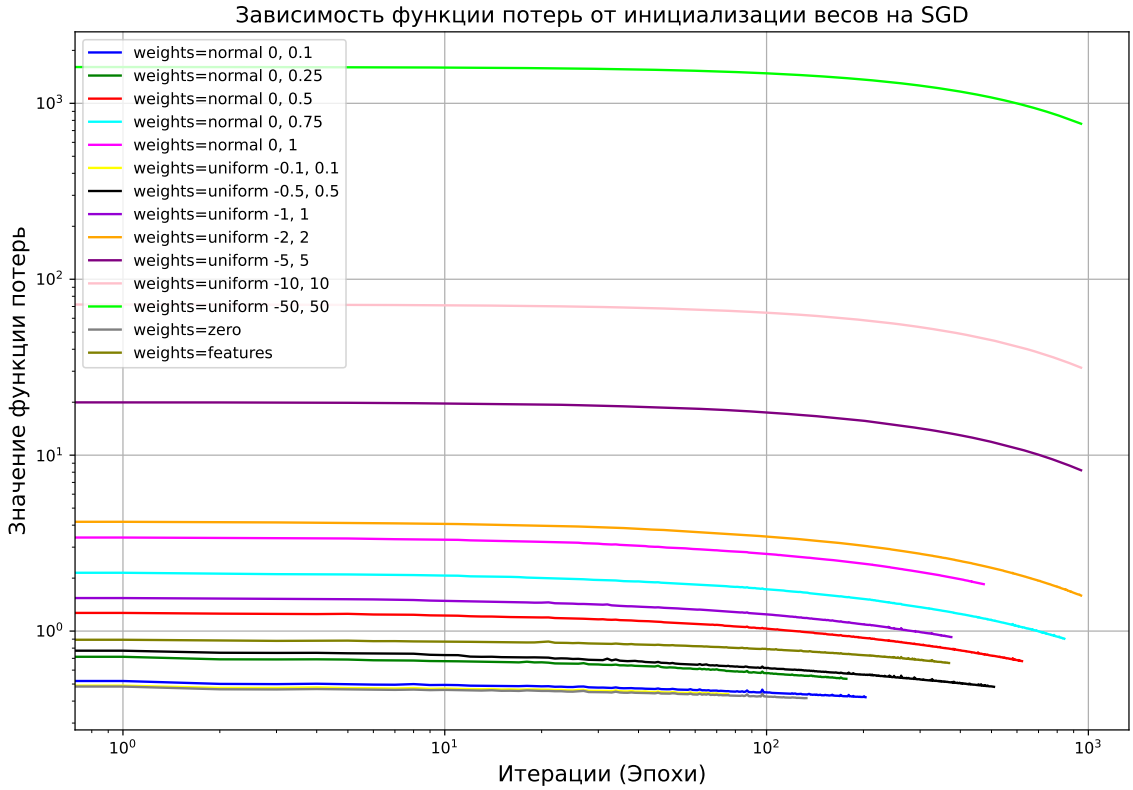


Рис. 26: Значение функции потерь на эпохах модели в зависимости от начального приближения весов

На рисунке 26 представлены графики зависимости значения функции потерь на каждой эпохе стохастического градиентного спуска от начального приближения весов w_0. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при $w_0 = (0, ..., 0)$.

6.2.4 Подбор размера подвыборки *batch_size*

Будем рассматривать значения *batch_size* среди следующего множества значений: {100, 200, 300, 400, 500, 750, 1000, 1500, 2000, 5000, 10000, 20000}.

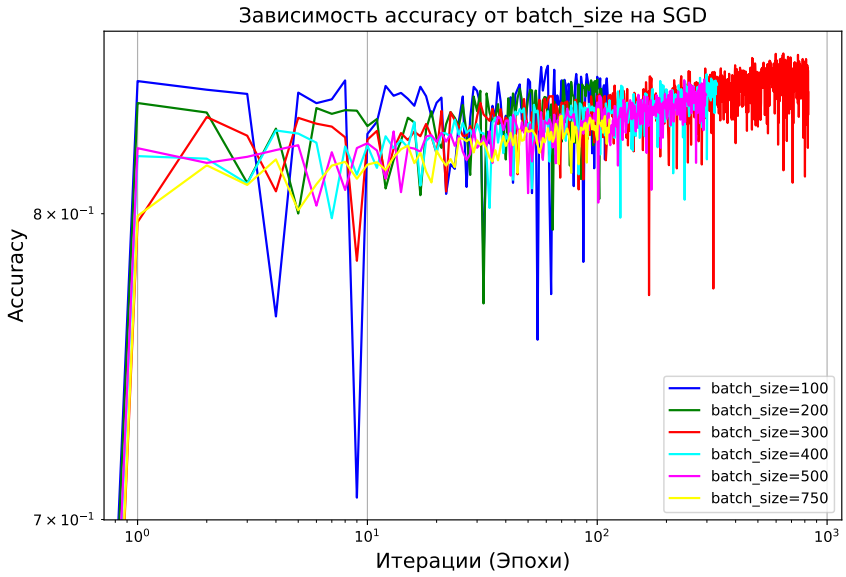


Рис. 27: Accuracy на эпохах модели в зависимости от параметра *batch_size*

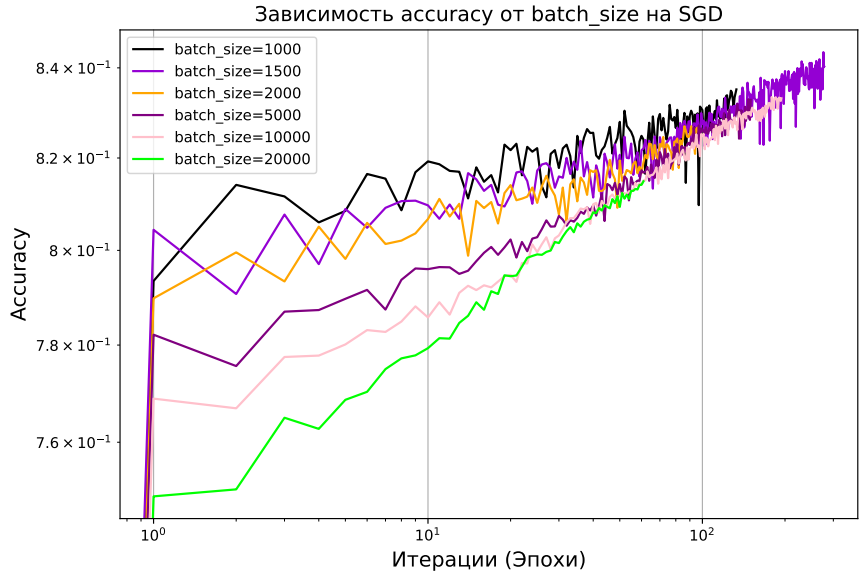


Рис. 28: Accuracy на эпохах модели в зависимости от параметра *batch_size*

На рисунках 27 и 28 представлены графики зависимости accuracy модели на каждой эпохе стохастического градиентного спуска от параметра *batch_size*. Как можно заметить из графиков, наилучшая точность модели достигается при *batch_size* = *300*.

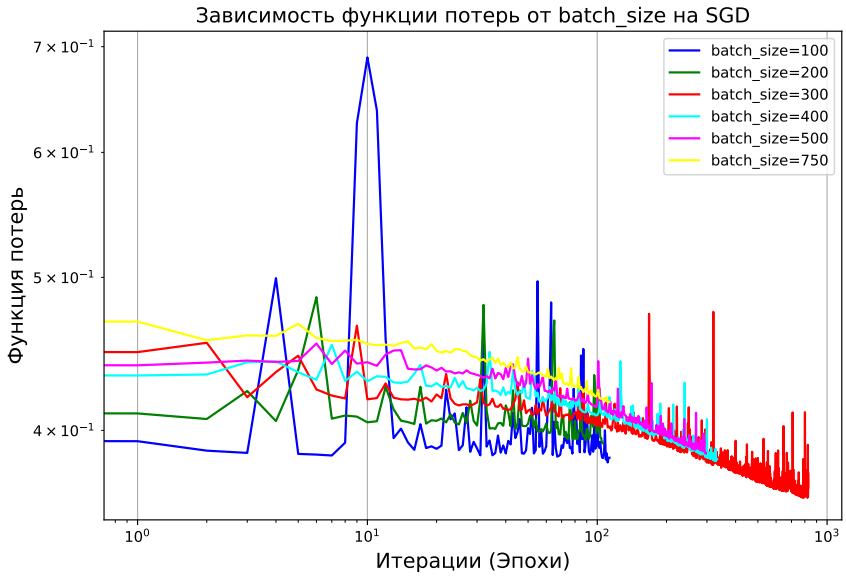


Рис. 29: Значение функции потерь на эпохах модели в зависимости от параметра *batch_size*

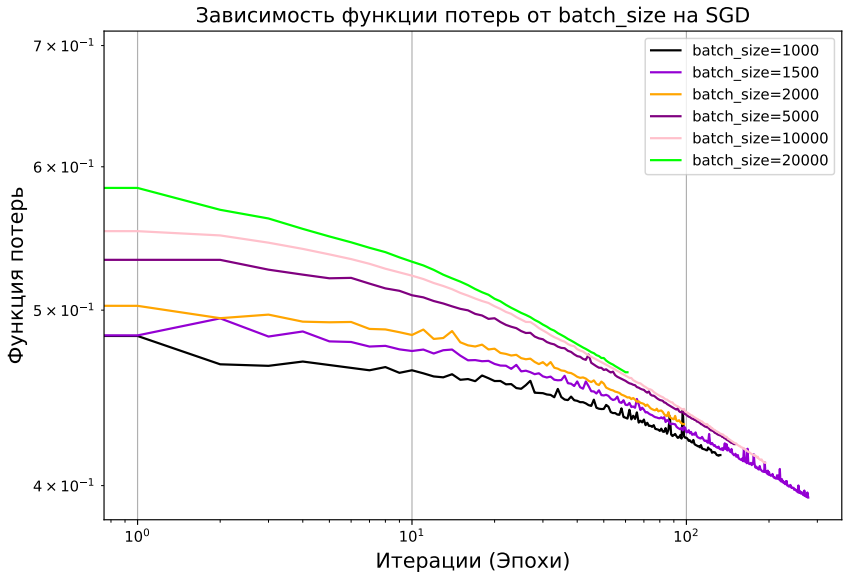


Рис. 30: Значение функции потерь на эпохах модели в зависимости от параметра *batch_size*

На рисунках 29 и 30 представлены графики зависимости значения функции потерь на каждой эпохе стохастического градиентного спуска от параметра *batch_size*. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при $batch\_size = 300$.

6.2.5 Подбор коэффициента L_2-регуляризации

Будем рассматривать значения *l2_ coef*, равномерно расположенные по логарифмической шкале от 10^{-4} до 10^4.

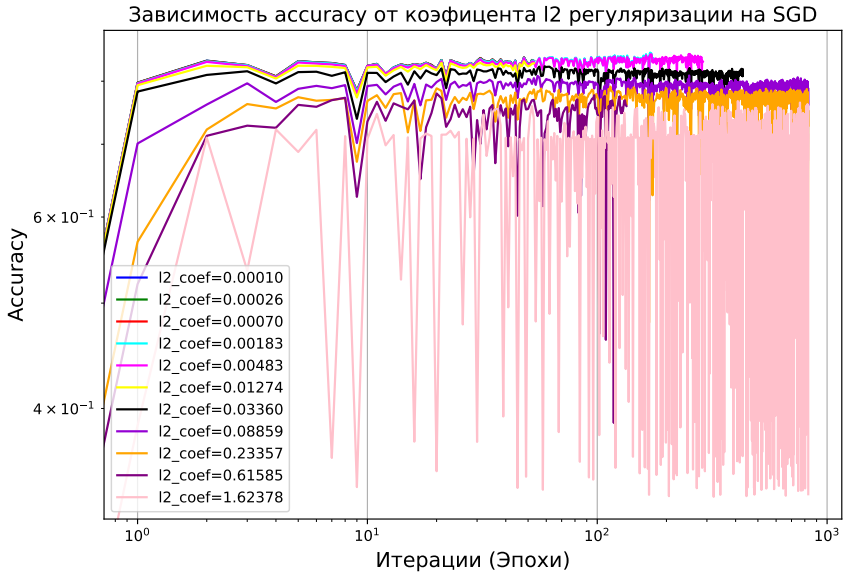


Рис. 31: Accuracy на эпохах модели в зависимости от параметра *l2_ coef*

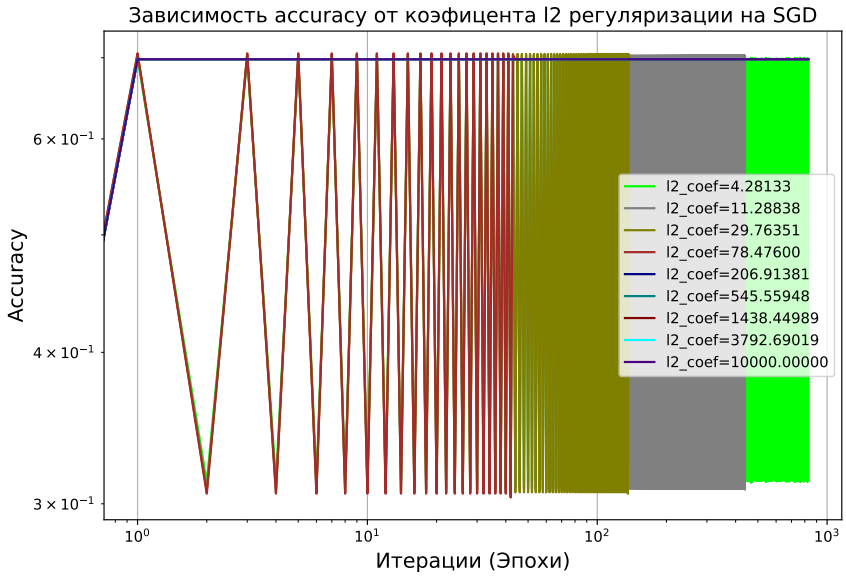


Рис. 32: Accuracy на эпохах модели в зависимости от параметра *l2_ coef*

На рисунках 31 и 32 представлены графики зависимости accuracy модели на каждой эпохе стохастического градиентного спуска от параметра *l2_ coef*. Как

можно заметить из графиков, наилучшая точность модели достигается при $l2\_coef = 0.00183298071$.

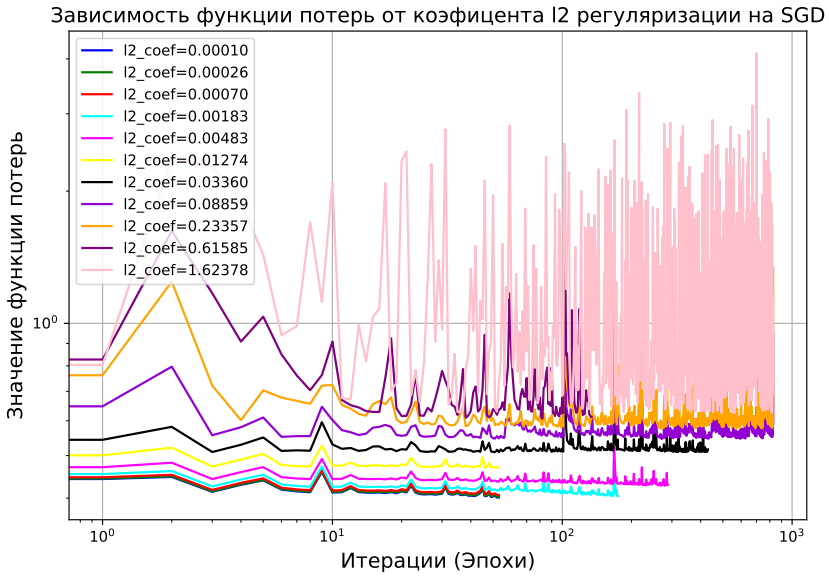


Рис. 33: Значение функции потерь на эпохах модели в зависимости от параметра $l2\_coef$

На рисунке 33 представлены графики зависимости значения функции потерь на каждой эпохе стохастического градиентного спуска от параметра $l2\_coef$. Как можно заметить из графиков, наилучшая сходимость значений функции потерь модели достигается при $l2\_coef = 0.00183298071$.

6.3 Выводы

Проведенные эксперименты показали следующие лучшие параметры стохастического градиентного спуска для задачи логистической регрессии при прочих равных условиях:

- $step\_alpha = 0.372759372$
- $step\_beta = 0.001$
- $w_0 = (0, ..., 0)$
- $batch\_size = 300$
- $l2\_coef = 0.00183298071$

7 Сравнение градиентного и стохастического градиентного спуска для задачи логистической регрессии

7.1 Постановка задачи

Сравнить поведение градиентного и стохастического градиентного спуска для задачи логистической регрессии.

7.2 Реализация

Обучим каждый метод на лучших параметрах, полученных в предыдущих разделах (5 и 6), и сравним их.

Рассмотрим accuracy и значения функции потерь на каждом шаге алгоритма метода, время работы каждой итерации (эпохи), а также общее время работы и итоговую точность.

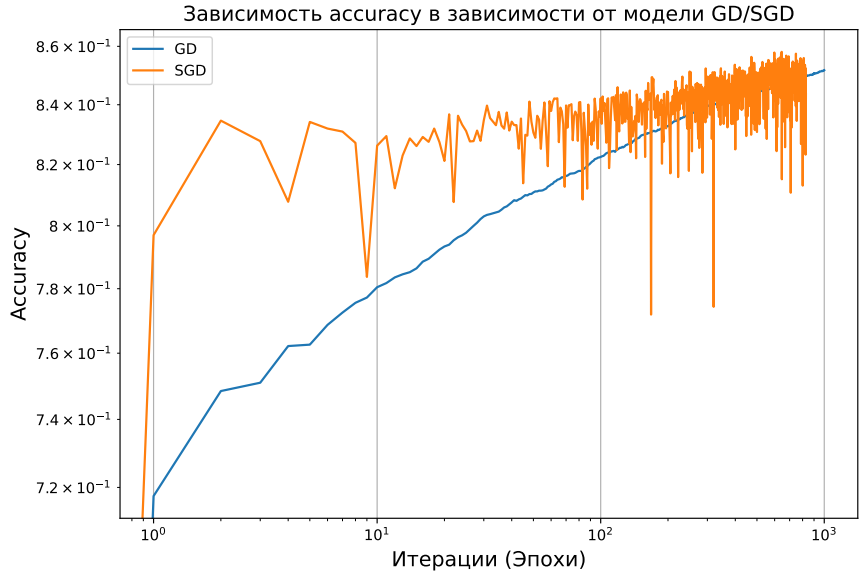


Рис. 34: Accuracy на итерациях (эпохах) методов GD/SGD

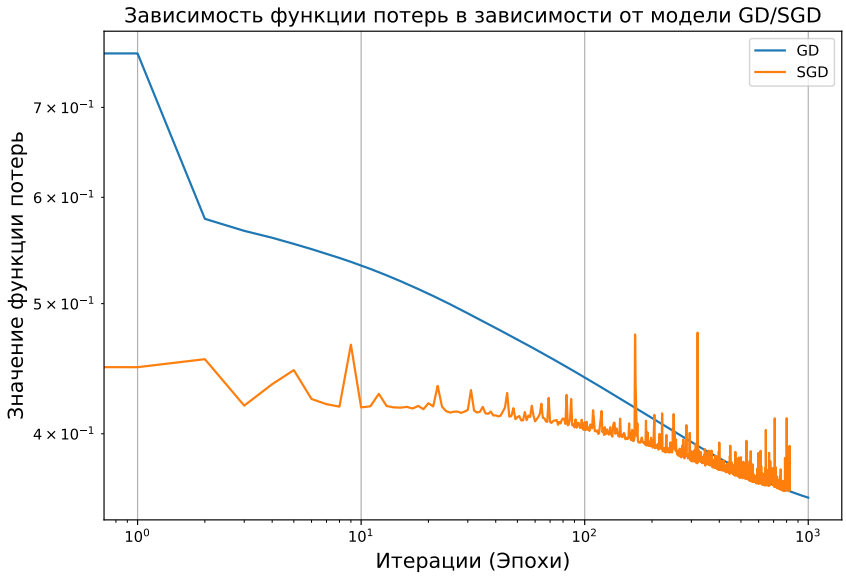


Рис. 35: Значения функции потерь на итерациях (эпохах) методов GD/SGD

На рисунках 34 и 35 представлены графики зависимости accuracy и значения функции потерь на каждой итерации (эпохе) градиентного и стохастического градиентного спуска. Как можно заметить из графиков, метод градиентного спуска дает лучшую точность, однако, значение функции потерь сходится быстрее на методе стохастического градиентного спуска.

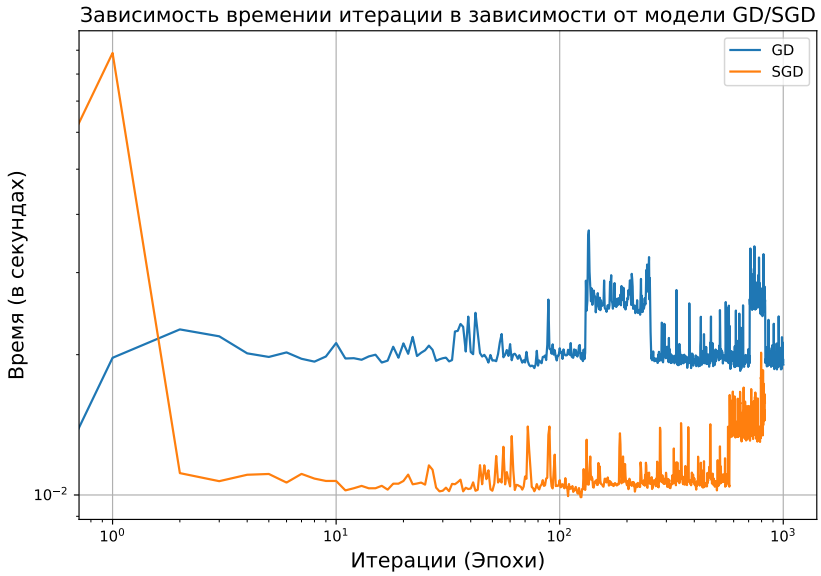


Рис. 36: Время работы итераций (эпох) методов GD/SGD

На рисунке 36 представлен время работы итераций (эпох) методов градиентного и стохастического градиентного спуска. Как можно заметить из графиков,

метод стохастического градиентного спуска затрачивает на одну эпоху меньше времени, чем метод градиентного спуска на одну итерацию.

В таблице 1 ниже представлены точности на тестовой выборке и общее время работы методов градиентного и стохастического градиентного спуска. Как можно заметить из таблицы, стохастичесский работает в разы быстрее градиентного спуска, причем accuracy двух методов отличается не больше, чем на 10^{-2} .

Метод	Accuracy	Время выполнения(сек.)
GD	0.851760495260205	21.50232481956482
SGD	0.8435867672663958	9.866184711456299

Таблица 1: Точность на тестовой выборке и общее время работы методов градиентного и стохастического градиентного спуска

7.3 Выводы

Анализ представленных данных показывает, что выбор между методом градиентного и стохастического градиентного спуска зависит от приоритетов решаемой задачи:

- Градиентный спуск демонстрирует лучшую точность, что может быть критически важным для задач, где даже небольшое улучшение точности играет значимую роль.
- Стохастический градиентный спуск обеспечивает более быстрое время сходимости функции потерь и значительно снижает затраты по времени, что делает его предпочтительным для задач, где вычислительные ресурсы ограничены или требуется быстрая обработка больших объемов данных.

Таким образом, если точность критична, предпочтителен градиентный спуск, однако, если важнее время обработки и ресурсоёмкость, более эффективным будет стохастический градиентный спуск. Отличие между точностями обоих методов минимально (менее 10^{-2}), что делает стохастический метод более практичным для большинства задач.

8 Лемматизация

8.1 Постановка задачи

Применить алгоритм лемматизации *WordNetLemmatizer* из библиотеки *nltk* к коллекции комментариев. Удалить из текста стоп-слова, используя список стоп-

слов из *nltk*. Исследовать, как предобработка корпуса повлияла на точность классификации, время работы алгоритма и размерность признакового пространства.

8.2 Реализация

Произведем лемматизацию текстов комментариев, используя класс *WordNetLemmatizer*. В процессе лемматизации удаляются стоп-слова — слова и символы, которые не несут значимой смысловой нагрузки. Остальные слова преобразуются в их начальную форму, т.е. в инфинитив.

Сравним размерности признакового пространства выборки до и после лемматизации:

Выборка	Размерность признакового пространства
До лемматизации	3736
После лемматизации	3195

Таблица 2: Размерность признакового пространства выборки до и после лемматизации

В таблице 2 приведены размерности признакового пространства выборки до и после лемматизации. Как можно заметить из таблицы, лемматизация позволила сократить признаковое пространство на 541 признак.

Рассмотрим, как проведение лемматизации повлияло на изменение точности, значения функции потерь на каждой итерации (эпохе), а также времени ее выполнения.

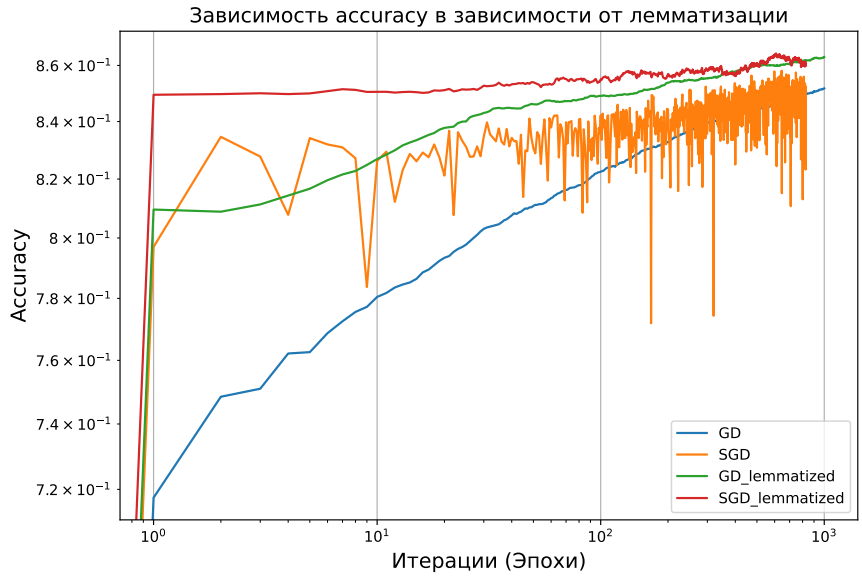


Рис. 37: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от проведения лемматизации

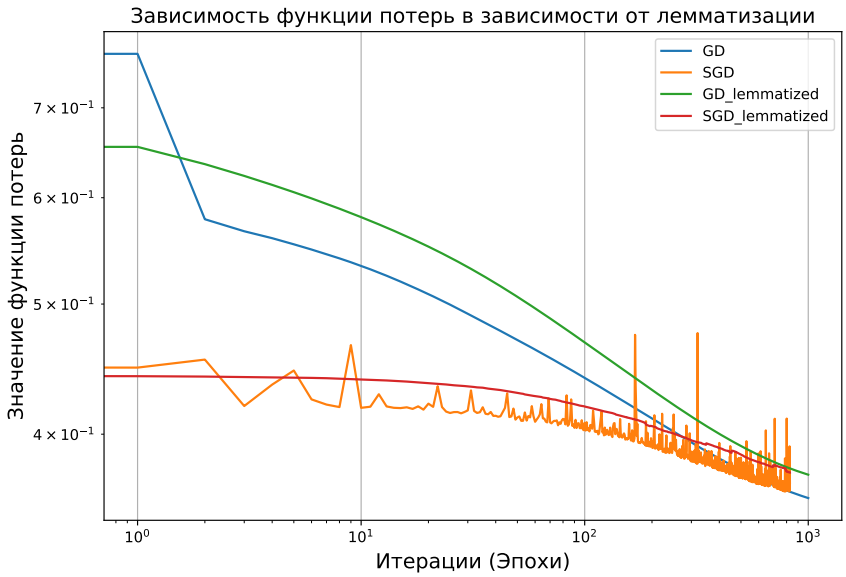


Рис. 38: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от проведения лемматизации

На рисунках 37 и 38 представлены графики зависимости accuracy и значения функции потерь на каждой итерации (эпохе) градиентного и стохастического градиентного спуска от проведения лемматизации. Как можно заметить из графиков, лемматизация улучшает эффективность обоих методов, повышая accuracy с первых итераций (эпох) методов и ускоряя сходимость функции потерь. Метод

стохастического градиентного спуска показывает на лемматизированной выборке намного больше стабильность.

В таблице 3 ниже представлены точности на тестовой выборке и общее время работы методов градиентного и стохастического градиентного спуска в зависимости от проведения лемматизации.

Метод	Accuracy	Время выполнения(сек.)
GD	0.851760495260205	21.50232481956482
SGD	0.8435867672663958	9.866184711456299
GD_lemmatized	0.8630295995356936	27.30692982673645
SGD_lemmatized	0.8599825885084156	10.892971515655518

Таблица 3: Точность на тестовой выборке и общее время работы методов градиентного и стохастического градиентного спуска в зависимости от проведения лемматизации

Исходя из вышеприведенных данных, лемматизация улучшает точность обоих методов, однако требует увеличения времени выполнения. Метод GD показывает лучшую точность после лемматизации, тогда как SGD сохраняет преимущество в скорости работы.

8.3 Выводы

Лемматизация положительно сказывается на эффективности обоих методов, улучшая accuracy с первых итераций и ускоряя сходимость функции потерь. Несмотря на увеличение времени выполнения, её применение обеспечивает значительное повышение стабильности метода SGD и улучшение итоговой точности для метода GD.

9 Сравнение представлений *BagOfWords* и *Tfidf* и подбор параметров *min_ df* и *max_ df* конструкторов

9.1 Постановка задачи

Исследовать качество, время работы алгоритма и размер признакового пространства в зависимости от следующих факторов:

- использовалось представление *BagOfWords* или *Tfidf*
- параметров *min_ df* и *max_ df* конструкторов.

9.2 Реализация

Для построения представлений *BagOfWords* и *Tfidf* использовались *CountVectorizer* и *TfidfVectorizer* из библиотеки *sklearn*, соответственно.

9.2.1 Сравнение представлений *BagOfWords* и *Tfidf* с параметрами по умолчанию

Построим представления *BagOfWords* и *Tfidf* с параметрами по умолчанию. Сравним точность, значения функции потерь на каждой итерации (эпохе), а также время ее выполнения в зависимости от представления. В сравнении использовались модели с ранее подобранными лучшими параметрами.

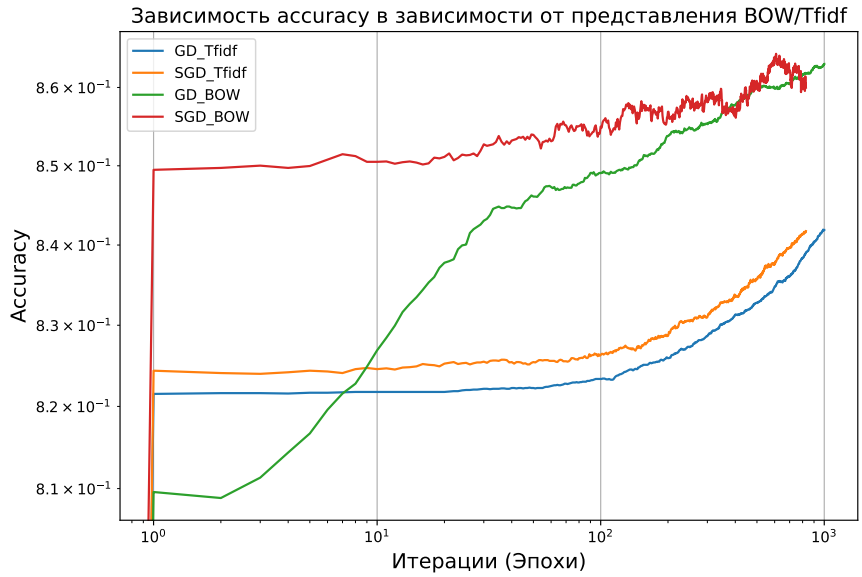


Рис. 39: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf*

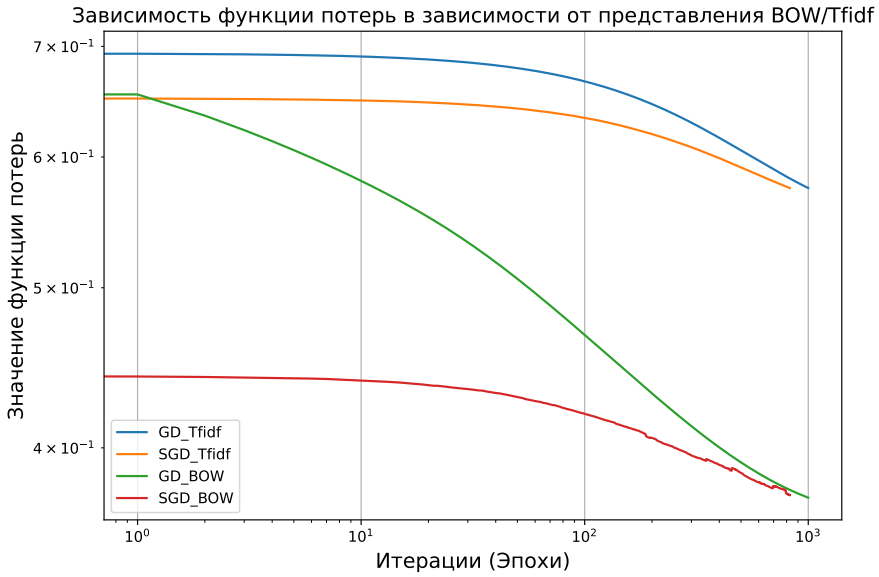


Рис. 40: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf*

На рисунках 39 и 40 представлены графики зависимости accuracy и значения функции потерь на каждой итерации (эпохе) градиентного и стохастического градиентного спуска от представления *BOW/Tfidf*. Как можно заметить из графиков, представление *BOW* улучшает точность и ускоряет сходимость функции потерь для методов GD и SGD, делая *BOW* предпочтительным выбором по сравнению с *Tfidf*.

В таблице 4 ниже представлены точности на тестовой выборке и общее время работы методов градиентного и стохастического градиентного спуска в зависимости от представления *BOW/Tfidf*.

Метод	Accuracy	Время выполнения(сек.)
GD_Tfidf	0.8418939833623524	12.866596937179565
SGD_Tfidf	0.8417488875991488	5.978341341018677
GD_BOW	0.8630295995356936	27.30692982673645
SGD_BOW	0.8599825885084156	10.892971515655518

Таблица 4: Точность на тестовой выборке и общее время работы методов градиентного и стохастического градиентного спуска в зависимости от представления *BOW/Tfidf*

Исходя из вышеприведенных данных, представление *BOW* обеспечивает более высокую точность, но требует больше времени выполнения, а *Tfidf* позволяет значительно ускорить работу методов, но с небольшим снижением точности.

9.2.2 Сравнение представлений *BOW/Tfidf* по параметрам *min_ df* и *max_ df* конструкторов

Сравним работу моделей по параметрам *min_ df* и *max_ df* конструкторов представлений *BOW/Tfidf*.

Будем рассматривать параметры *min_ df*, равномерно расположенные по логарифмической шкале от 10^{-10} до 10^{-1}, а также проанализируем параметры *max_ df*, равномерно расположенные по линейной шкале от 0.4 до 1.

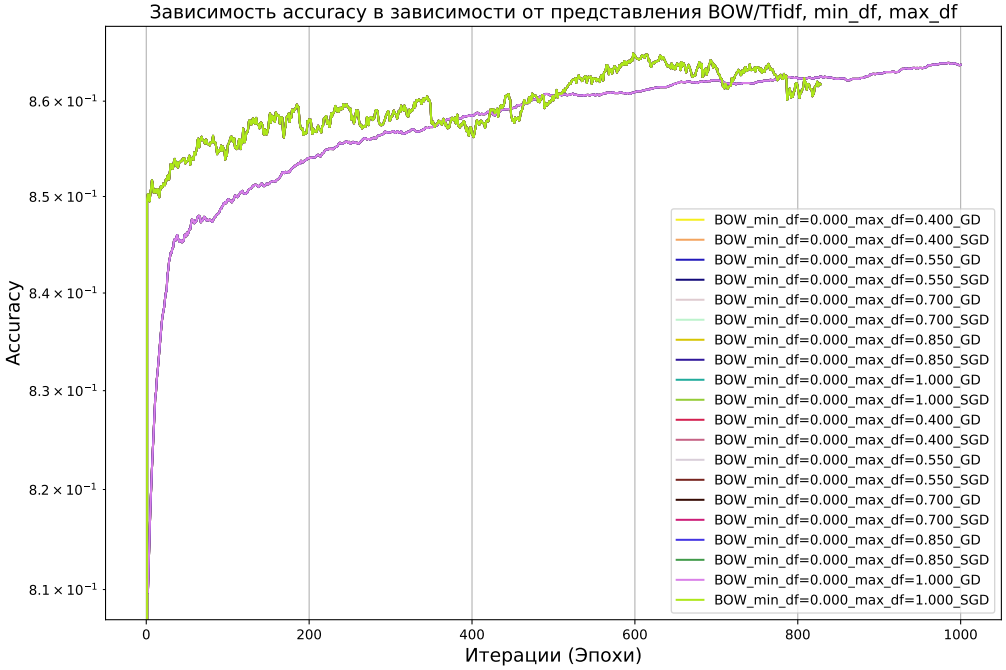


Рис. 41: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

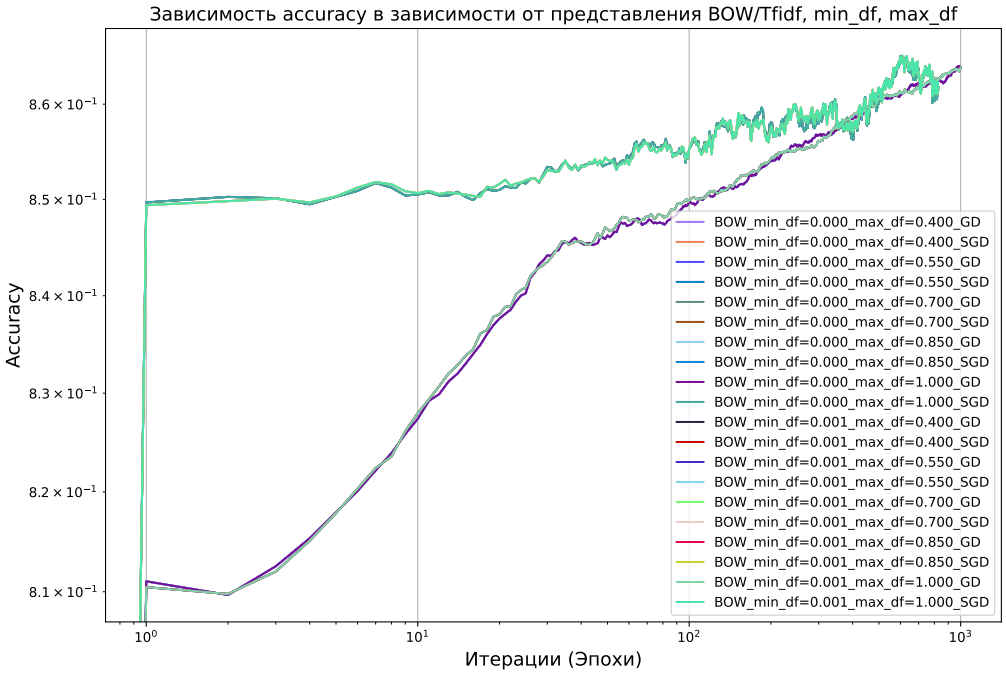


Рис. 42: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

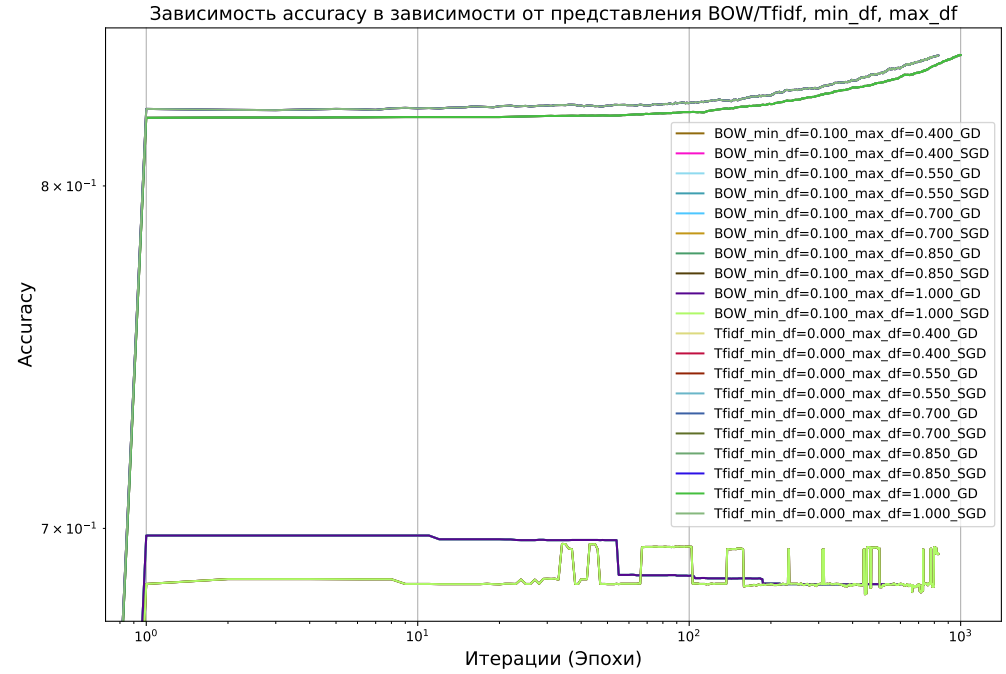


Рис. 43: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

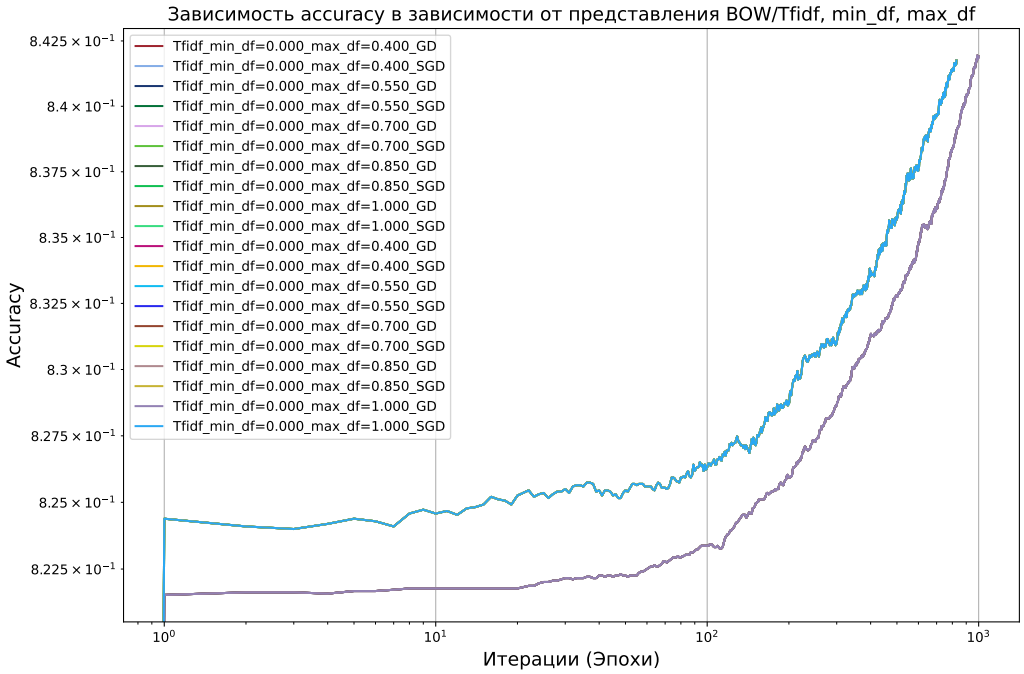


Рис. 44: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

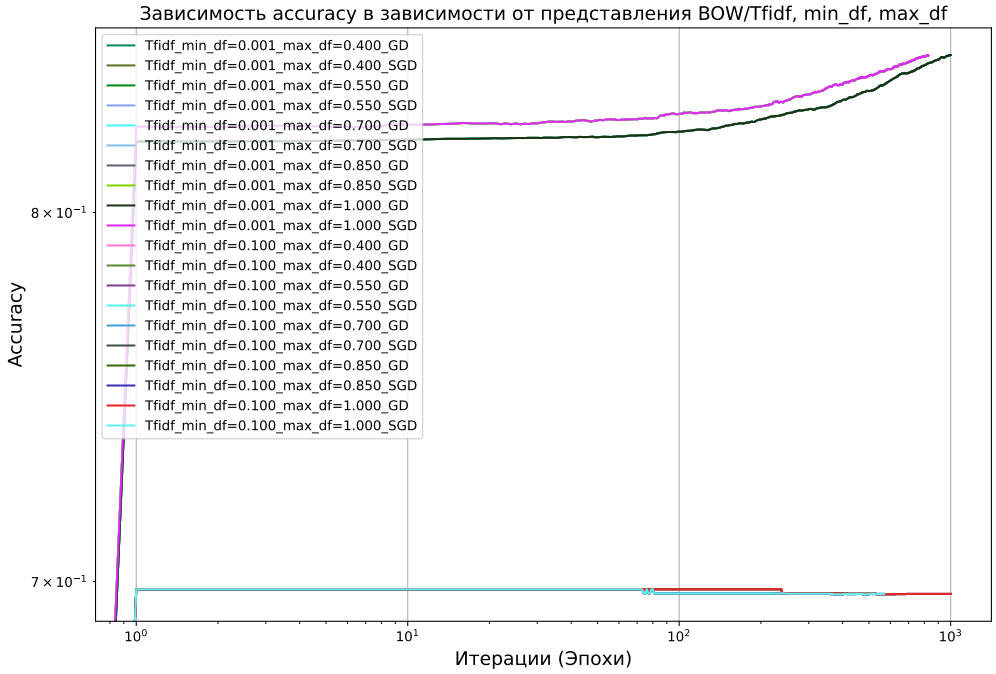


Рис. 45: Accuracy на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

На рисунках 41, 42, 43, 44, 45 представлены графики зависимости accuracy на каждой итерации (эпохе) градиентного и стохастического градиентного спуска от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*. Как можно заметить из графиков, увеличение параметра *min_ df* или уменьшение *max_ df* приводит к стабилизации и улучшению результатов. Более строгие ограничения на частотность позволяют исключить незначимые признаки, что ускоряет сходимость и повышает итоговую точность.

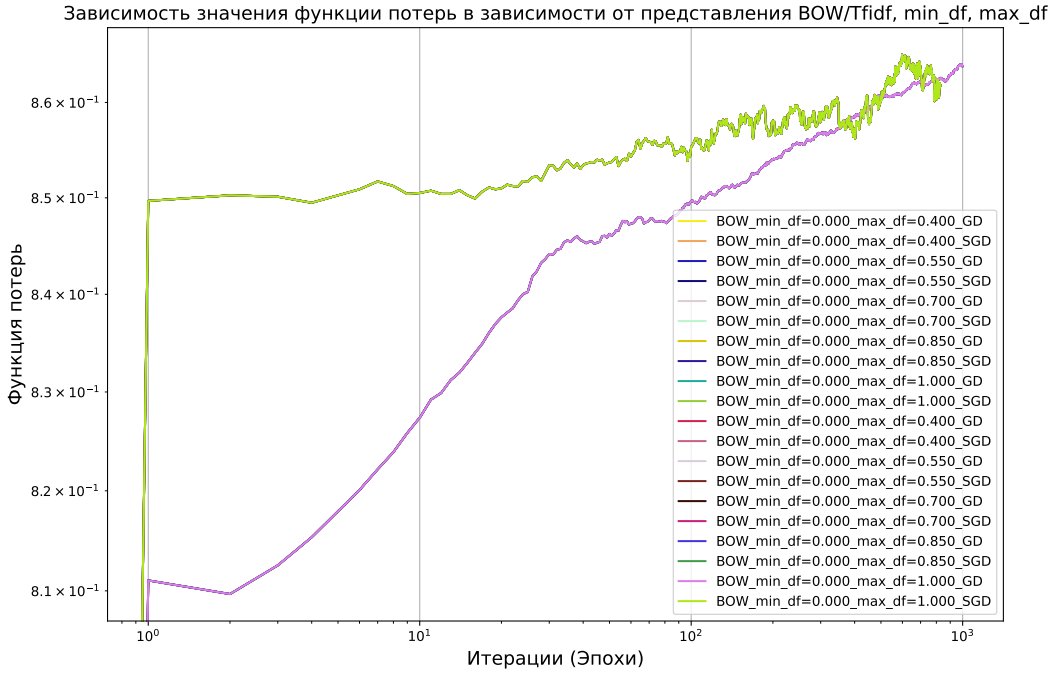


Рис. 46: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

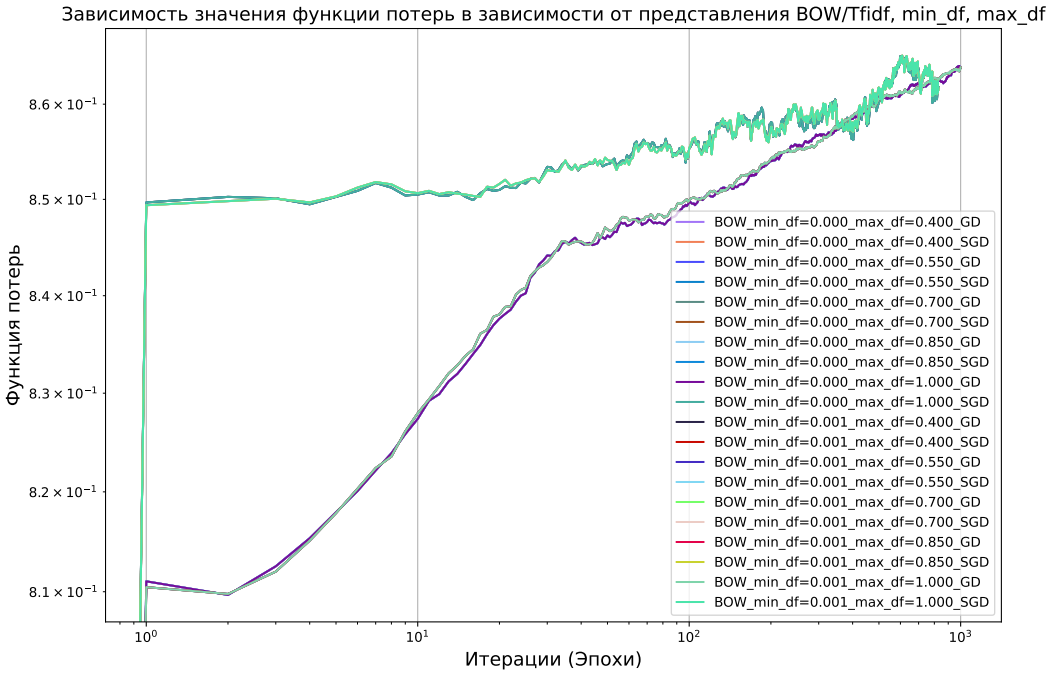


Рис. 47: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

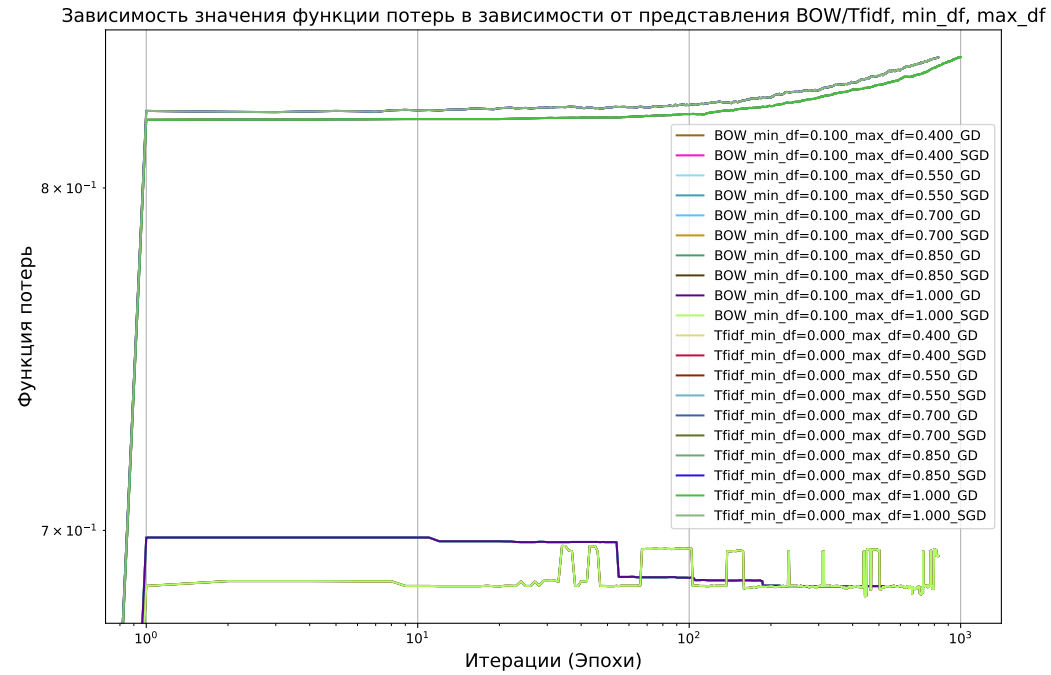


Рис. 48: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

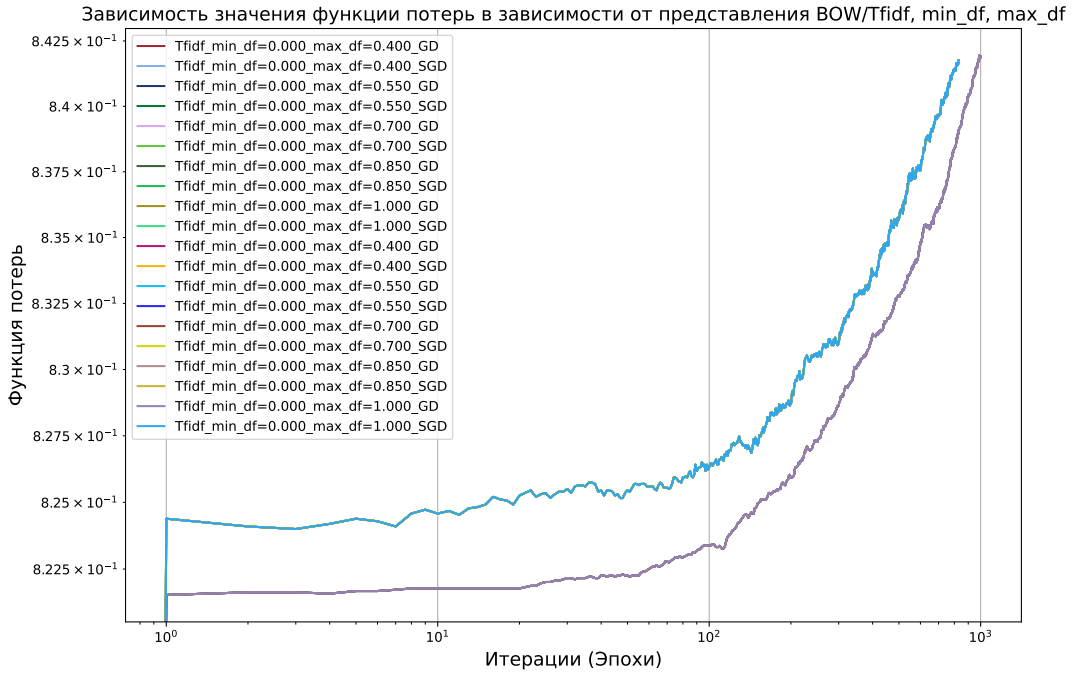


Рис. 49: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

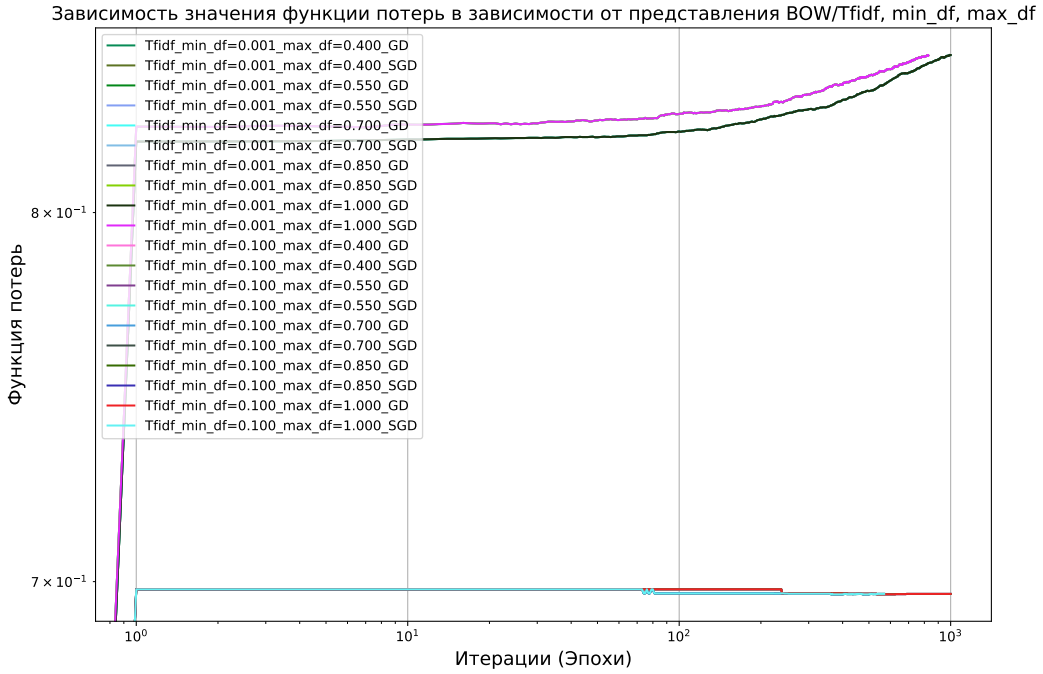


Рис. 50: Значения функции потерь на итерациях (эпохах) методов GD/SGD в зависимости от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*

На рисунках 46, 47, 48, 49, 50 представлены графики зависимости значения функции потерь на каждой итерации (эпохе) градиентного и стохастического градиентного спуска от представления *BOW/Tfidf* и параметров *min_ df* и *max_ df*.

Проанализируем размерность признакового пространства выборки в зависимости от параметров *min_ df* и *max_ df*.

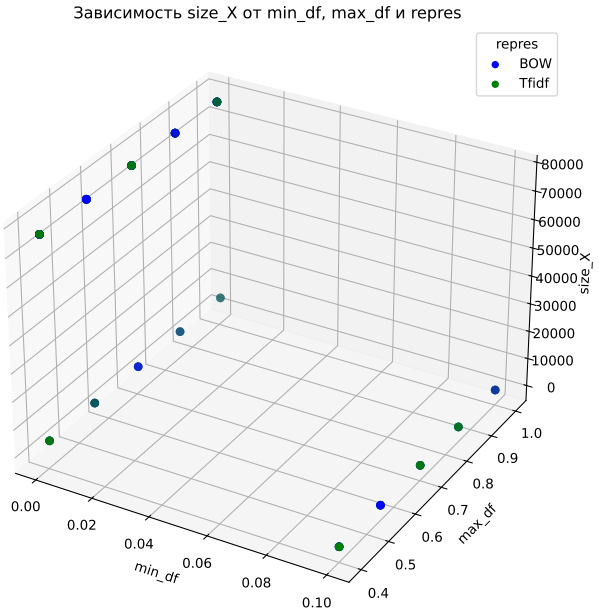


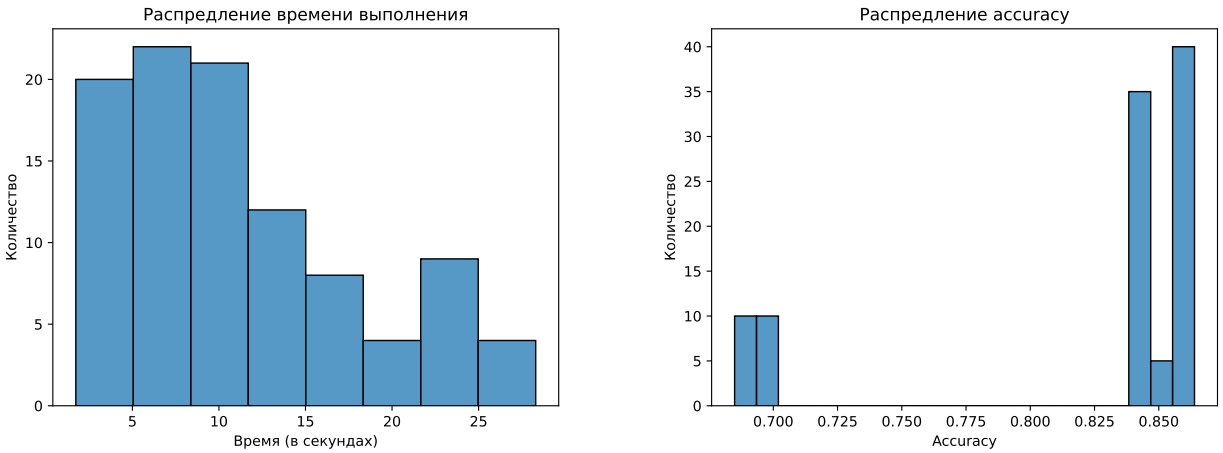
Рис. 51: Зависимости размерность признакового пространства выборки от параметров *min_ df* и *max_ df*

На рисунке 51 представлен график зависимости размерности признакового пространства выборки от параметров *min_ df* и *max_ df*. Как можно заметить из графика, сильное уменьшение *max_ df* или сильное увеличение *min_ df* приводит к слишком большой размерности сокращению признакового пространства, а сильное увеличение *max_ df* или сильное уменьшение *min_ df* - к слишком большой размерности признакового пространства.

9.3 Выводы

Рассмотрим распределение времени выполнения одного метода и accuracy (См. рис. 52) для определения наиболее оптимальных представления и параметров:

Выберем те модели, которые выполняются быстрее 10 секунд, имеют точность больше 0.86 и размер признакового пространства выборки, на которой обучалась модель больше 100, но меньше 7000 (См. табл. 5):



(a) Распределение времени выполнения

(b) Распределение accuracy

Рис. 52: Распределения времени выполнения и accuracy для методов, обученных на различных представлениях *BOW/Tfidf* и параметрах *min_df* и *max_df*

model	repres	min_df	max_df	accuracy	X.shape[1]	time
SGD	*BOW*	0.000562	0.40	0.86153	4724	7.022379
SGD	*BOW*	0.000562	0.55	0.86153	4724	7.479241
SGD	*BOW*	0.000562	0.70	0.86153	4724	7.314365
SGD	*BOW*	0.000562	0.85	0.86153	4724	7.462992
SGD	*BOW*	0.000562	1.00	0.86153	4724	7.316502

Таблица 5: Модели, соответствующие параметрам

Выберем наилучшие параметры и представление, основываясь на балансе скорости работы и точности.

Таким образом, наилучшими будут:

- Представление - *BOW*
- *min_df* - 0.000562
- *max_df* - 0.7

10 Выбор лучшего алгоритма и анализ ошибок

10.1 Постановка задачи

- Выбрать лучший алгоритм для тестовой выборки.
- Проанализировать ошибки алгоритма.

- Проанализировать и указать общие черты объектов, на которых были допущены ошибки.

10.2 Реализация

10.2.1 Создание лучшей модели

На основании проведённого анализа в предыдущих пунктах можно сделать вывод, что модель стохастического градиентного спуска уступает методу градиентного спуска по точности (accuracy), но значительно превосходит его по скорости работы. В связи с этим, обучим модель SGD с использованием ранее определённых оптимальных параметров, а также с применением лемматизации выборки и представления данных, параметры которого также были выбраны как наилучшие в ходе анализа. Хотя предложенная модель может уступать градиентному спуску по точности, она будет демонстрировать значительное преимущество в скорости работы.

Лучшие параметры и представления, полученные ранее:

- $step\_alpha = 3.72759372e\text{-}01$
- $step\_beta = 0.001$
- $w_0 = (0, ..., 0)$
- $batch\_size = 300$
- $l2\_coef = 0.00183298071$
- представление - *BagOfWords*
- параметр конструктора представления $min\_df = 0.000562$
- параметр конструктора представления $max\_df = 0.7$

Ниже приведено время обучения данной модели и accuracy на тестовой выборке (см. табл. 6), а также графики зависимости accuracy и значения функции потерь от эпохи метода (см. рис. 53 и 54):

Accuracy	Время обучения модели(сек.)
0.8626910427548848	11.371453762054443

Таблица 6: Точность на тестовой выборке и время обучения лучшей модели

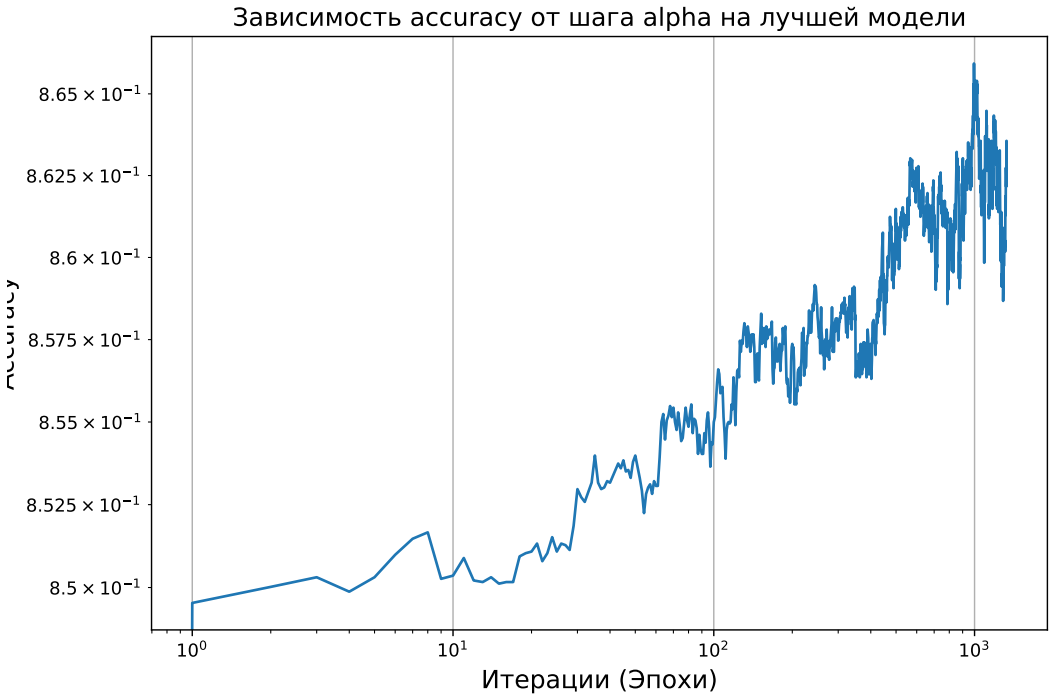


Рис. 53: Зависимость accuracy от эпохи получившегося метода

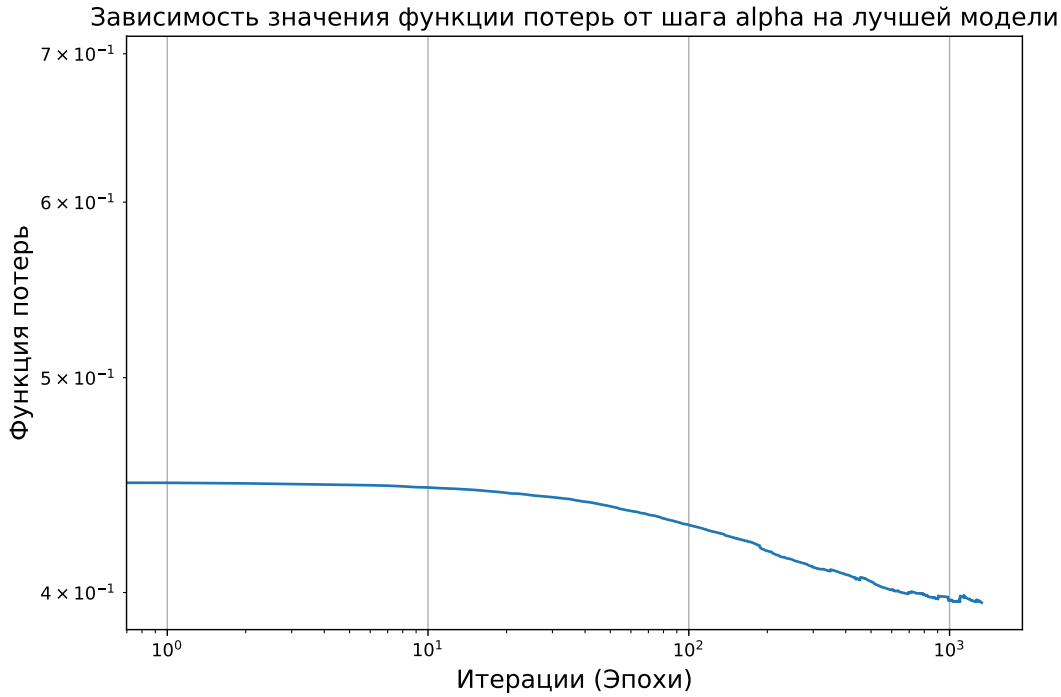


Рис. 54: Зависимость значения функции потерь от эпохи получившегося метода

10.2.2 Анализ ошибок

Для анализа ошибок нашей модели построим матрицу ошибок (См. рис. 55):

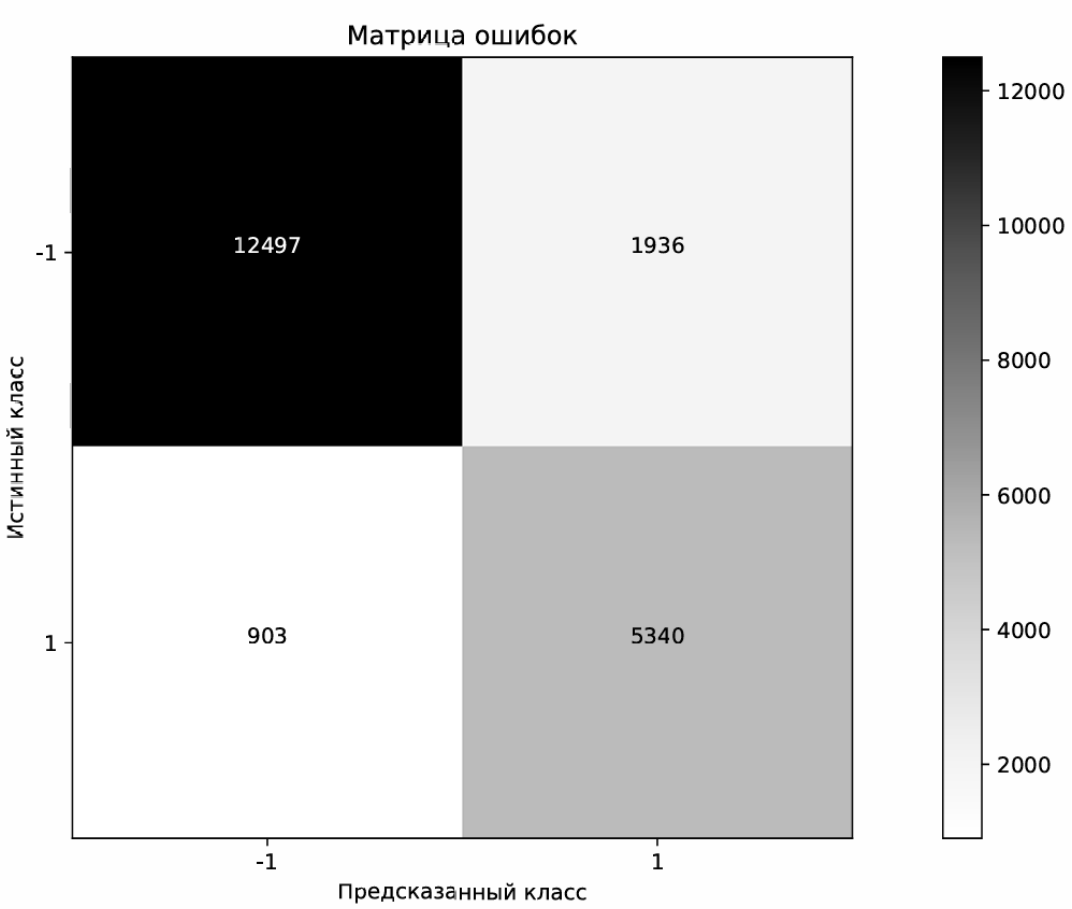


Рис. 55: Матрица ошибок алгоритма

Ложно положительных объектов - 1936, ложно отрицательных - 903.

Проанализируем ошибки на примерах:

- False Negative объекты:
  - uncivil b ignorance persistence think saying correct already provoking provoking already act incivility offence idiot would given talking oh ironic still talk page afraid willing discus anything future well nice feel exact way halleluiah agreement something lol hope weird see said ton time one deluded people think right time whenever proved otherwise shutdown try turn table person wrong uncivil
  - pigsonthewing pigsonthewing aka andy mabbett obese stink booze need wear bikini often thanks leonig mig

- hey kendricks pittsburgh compton wack motherfucker nwa would ashamed boyyyyyyyyyyyyyy
- barnstar thanks barnstar big sexy pimp
- see left part supposed discussing quickly reverted shabazz wrongly claimed done good deleting certainly anyone honest non biased situation would disagree merely placed paragraph question different area trimmed bit unless someone determined make paragraph important action jidf claiming statement something bad deleting would true therefore feel hostile intention concerned people respect feel good intention ganging handle also thinking since shabazz essentially told get lost think going stick around get louder also going get many friend respect noble intention start getting involved f ridiculous
- sick idiocy going de watchlist whole lot ai article never ending pointless battle fancruft always win good riddance
- dumbass called also responsible
- user page please stop putting crap thanks
- support merge sucker
- lake bob would someone please stop changing reference lake bob farris sea care name referenced old news article simply called current student personally attest fact universally called lake bob student faculty alike also would whatever student responsible recent vandalism please quit wikipedia place play stupid prank air grudge school end making look like fool thanks

- False Positive объекты:
  - know get around watching soon know seen punk stuff reason really really annoys know good wrestler like oh get started wwe treatment paul london seen load indy work stuff roh amazing fucking awesome wwe fucked big time went arounf rumor site got released planning quitting wrestling moving acting something would suck stupid wwe penchant asshole look like steroid like cena idiot like batista bitched moved raw got beat jericho title take deep breath sorry ranting
  - enjoy stay
  - brainloss swedish word hj rnsl pp loosely translated brainloss braindrop brain drop head meaning feeling get realise act utter stupidity done even understand could stupid tagging article deletion wrong title moved different title good thing resourceful brain dept

- posted sharon mooney
- teacher really dumb people think people first last name people mr williams teacher
- nice attempt look horrible crammed way much information small as area honestly fix far lazy
- care people caring read one care coming either treelo going hire lawyer either sued dare delete message message give period edt
- support someone calling stupid registered editor nice know please give warning deserve
- spell kris draper without kid raper
- die teutsche wikipedia ist voll von menschen wie ihnen einer mehr macht allerdings auch nichts au

10.3 Выводы

- Построен лучший алгоритм по ранее определенным оптимальным параметрам с точностью, равной *0.8626910427548848*, и временем работы *11.37145376205444*3 секунд.

- Ошибки False Negative связаны с неспособностью модели обнаруживать скрытую токсичность, сарказм или пассивно-агрессивные высказывания. Такие комментарии часто содержат оскорбительные слова, но иногда токсичность проявляется на уровне подтекста или скрытой агрессии, что требует более сложного анализа контекста. Кроме того, грамматические ошибки и нестандартные синтаксические конструкции в тексте могут приводить к тому, что модель не распознает явную токсичность. Сложные многосоставные предложения также затрудняют правильную интерпретацию комментария. В случае False Positive, где нетоксичные комментарии классифицируются как токсичные, проблемы возникают из-за эмоциональной окраски текста, использования сложных слов с потенциально двойным смыслом, а также из-за нейтральной критики, которая может быть ошибочно воспринята как токсичная. Модель также может неверно интерпретировать фразы на других языках или сложные конструктивные комментарии, которые содержат отдельные слова, ошибочно распознаваемые как токсичные.
  Таким образом, основными проблемами являются трудности в учёте контекста, распознавании скрытой токсичности и правильной интерпретации эмоционально окрашенных текстов.

11 Итог

В ходе выполнения работы проведен анализ линейных моделей на примере задачи классификации токсичности комментариев. Реализованы методы градиентного и стохастического градиентного спуска, проведено их сравнение с учетом различных параметров, таких как размер шага, начальное приближение весов, размер подвыборки и коэффициент регуляризации. Выявлено, что градиентный спуск показывает немного более высокую точность, но стохастический градиентный спуск значительно превосходит его по скорости работы. Лемматизация данных продемонстрировала улучшение точности обоих методов за счет сокращения размерности признакового пространства, хотя и увеличила общее время выполнения алгоритмов.

Также выполнено сравнение различных представлений данных — *BagOfWords* и *Tfidf*. Представление *BagOfWords* оказалось более точным, однако *Tfidf* обеспечило более быстрое выполнение задач.

В рамках анализа ошибок алгоритма было установлено, что модель иногда ошибается на комментариях с сарказмом, скрытой агрессией и нетипичными синтаксическими конструкциями, что связано с ограничениями выбранных моделей.

Также, был составлен лучший алгоритм по ранее определенным оптимальным параметрам с точностью, равной *0.8627*, и временем работы *11.3715* секунд.

Таким образом, лучшие результаты по точности достигнуты с использованием метода градиентного спуска, лемматизации и представления *BagOfWords*. В то же время, стохастический градиентный спуск с представлением TF-IDF является более эффективным выбором для задач, требующих быстродействия.

Список литературы

[1] Материалы семинаров ММРО и Практикума 3 курса ВМК МГУ, https://github.com/mmp-practicum-team/mmp_practicum_fall_2024/blob/main/Seminars/08-text-processing-and-logreg/seminar.pdf

[2] kaggle, https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challen protect\mathrel{|}\protect\mathrel{\mkern-3mu}\protect\mathrel\std@ equal