



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ Информатика и системы управления _____

КАФЕДРА _____ «Программное обеспечение ЭВМ и информационные технологии» (ИУ7) _____

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ**

НА ТЕМУ:

***Метод нечеткой классификации
платежей по полнотекстовому
описанию на русском языке***

Студент _____ ИУ7-85Б _____
(Группа)

(Подпись, дата) _____ А.А. Лаврова
(И.О.Фамилия)

Руководитель

(Подпись, дата) _____ Л.Л. Волкова
(И.О.Фамилия)

Нормоконтроллер

(Подпись, дата) _____ Ю.В. Строганов
(И.О.Фамилия)

Москва, 2021 г.

РЕФЕРАТ

В данной работе рассматривается разработка нечёткого метода классификации платежей по полнотекстовому описанию на русском языке.

В тексте работы решаются задачи проведения анализа алгоритмов классификации и отбора терминов, разработка метода классификации платежных документов с помощью нечётких методов, реализация разработанного метода, а также исследование и выбор наиболее эффективного алгоритма.

Отчёт содержит 68 страниц, 17 рисунков, 8 таблиц и 17 источников.

СОДЕРЖАНИЕ

РЕФЕРАТ.....	5
СОДЕРЖАНИЕ.....	6
ОПРЕДЕЛЕНИЯ	9
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	10
ВВЕДЕНИЕ	11
1 Аналитический раздел.....	13
1.1 Формализация задачи	13
1.2 Машинное обучение	13
1.3 Классификация задач машинного обучения	14
1.4 Отбор терминов	15
1.4.1 Документная частота	16
1.4.2 Взаимная информация.....	17
1.4.3 Информационная выгода	17
1.4.4 Критерий хи-квадрат	18
1.5 Классификация текста.....	18
1.6 Алгоритм наивной Байесовской классификации	19
1.7 Алгоритм Роккио	20
1.8 Алгоритм k-ближайших соседей.....	22
1.9 Алгоритм опорных векторов	23
1.10 Алгоритм деревьев принятия решений	25
1.11 Метод логистической регрессии	27
1.12 Вывод	29
2 Конструкторский раздел	30
2.1 Декомпозиция разрабатываемого метода.....	30

2.2 Формирование входных данных	30
2.3 Отбор терминов	32
2.4 Обучение алгоритма классификации.....	33
2.4.1 Алгоритм наивной Байесовской классификации	33
2.4.2 Логистическая регрессия	35
2.5 Тестирование обученной модели	38
2.5.1 Аккуратность.....	38
2.5.2 Точность	38
2.5.3 Полнота	39
2.5.4 F-мера.....	39
2.5.5 Макро-усреднение и взвешенное усреднение	39
2.6 Вывод	40
3 Технологический раздел	41
3.1 Выбор средств разработки	41
3.1.1 Язык программирования	41
3.1.2 Среда разработки	41
3.1.3 Используемые библиотеки	42
3.2 Минимальные требования к вычислительной системе.....	42
3.3 Структура разработанного ПО	43
3.4 Описание классов	43
3.5 Установка ПО.....	46
3.6 Пример работы программы	48
3.7 Тестирование ПО	49
3.8 Вывод	50
4 Исследовательский раздел	52

4.1 Выборка	52
4.2 Параметризация алгоритмов.....	52
4.3 Сравнительный анализ методов классификации.....	55
4.4 Вывод	57
Заключение.....	58
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	59
ПРИЛОЖЕНИЕ А.....	61
ПРИЛОЖЕНИЕ Б	63

ОПРЕДЕЛЕНИЯ

В данной выпускной квалификационной работе использовались термины с следующими определениями:

Стоп-слова – термины, не несущие смысловой нагрузки и подлежащие удалению из документа.

Классификация текста – это процесс разбиения текста на определенные группы.

Нечёткая классификация – классификация, при которой документ может иметь принадлежность к разным классам с разной вероятностью.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В данной выпускной квалификационной работе используются следующие обозначения и сокращения:

- NLP – направление математической лингвистики, которое изучает и анализирует тексты на естественном языке;
- TF-IDF – метод отбора терминов;
- maxDF – максимальное допустимое число документов, в которых встретился термин t ;

ВВЕДЕНИЕ

Классификация текста – это одна из главных задач компьютерной лингвистики, так как к ней сводятся следующие задачи: определение автора или тематики текста, распознавание окраски изречения, определение принадлежности документа к той или иной категории и т.д.

В основе классификации текста лежит машинное обучение, которое за последнее время дало нам практическое распознавание речи, беспилотные автомобили, эффективный поиск в Интернете и значительное улучшение понимания генома человека. Машинное обучение получило настолько широкое распространение, что люди, вероятно, используют его каждый день, даже не подозревая об этом. Также некоторые исследователи считают, что это лучший способ для создания искусственного интеллекта.

Одна из сфер применения задач классификации – это организация определения кода операции платежных документов в банках. В данной отрасли существуют такие проблемы, как задействование большого количества персонала и трата значительных временных ресурсов для обработки информации вручную.

Прогресс в области новейших информационных технологий предопределил широкое распространение обработки большого потока информации. Так как зачастую операции по подобного рода обработке имеют рутинный характер, их можно автоматизировать с помощью алгоритмов классификации.

Целью данной выпускной квалификационной работы является разработка метода нечёткой классификации платежей по полнотекстовому описанию на русском языке. Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть существующие типы алгоритмов машинного обучения;
- проанализировать существующие алгоритмы классификации;
- разработать метод нечёткой классификации платежей по полнотекстовому описанию на русском языке;

- разработать программное обеспечение, демонстрирующее работу метода;
- исследовать разработанный метод на применимость.

1 Аналитический раздел

1.1 Формализация задачи

В соответствии с выбранной темой выпускной квалификационной работы необходимо разработать метод нечёткой классификации платежей по полнотекстовому описанию на русском языке с использованием готовой выборки для сортировки по заранее известным категориям.

1.2 Машинное обучение

Машинное обучение – это раздел теории искусственного интеллекта, предметом которого является поиск методов решения задач путем обучения в процессе решения сходных задач.

Для построения таких методов используются средства алгебры, математической статистики, дискретной математики, теории оптимизации, численных методов, и других разделов математики.

В науке о данных алгоритмом называется последовательность этапов статистической обработки. Алгоритмы обучаются поиску закономерностей в больших объемах данных для принятия решения или проведения прогноза по мере обработки информации.

Выделяют четыре основных шага для создания модели машинного обучения.

1. Выбор и подготовка тренировочных данных.

Обычно данные разделяют на два подмножества: учебное подмножество, которое впоследствии используется для обучения модели, и оценочное подмножество, применяемое для тестирования.

2. Выбор алгоритма.

Необходимо выбрать алгоритм, который обусловлен типом и количеством данных, а также типом решаемой задачи. Такими алгоритмами могут являться алгоритмы регрессии, деревья решений или алгоритмы обучения на примерах, которые подходят для заранее разделенных по категориям данных. Для тех

данных, которые не размечены, подходят алгоритмы кластеризации, алгоритмы ассоциации и нейронные сети.

3. Обучение алгоритма.

Обучение модели является итеративным процессом, который включает в себя запуск алгоритма, сравнение выходных данных с ожидаемым результатом, корректировка весов и смещений в алгоритме для получения более точного результата, повторный запуск алгоритма. Данные действия повторяются до тех пор, пока не будет достигнута необходимая точность. Полученный в результате алгоритм является моделью машинного обучения.

4. Использование и улучшение модели.

Финальным шагом является использование модели, получающей на вход данные для обработки.

Выделяют два основных типа машинного обучения: индуктивный и дедуктивный. Индуктивное машинное обучение основано на выявлении эмпирических закономерностей в поступающих на вход данных. Дедуктивный тип относят к области экспертных систем, где прецеденты (обучающая выборка) – это наборы входных данных и соответствующие им результаты. Решение формируется эмпирически, так как не существует единой формулы для описания зависимости между данными и результатами.

Важным критерием обучаемой системы является способность к обобщению, так как на практике входные данные могут быть неполными или неточными, то есть выходящими за пределы обучающей выборки. Поэтому важно при обучении системы подавать на вход разнообразные данные, чтобы машине было проще найти закономерности и, как следствие, результат.

1.3 Классификация задач машинного обучения

Задачи машинного обучения классифицируются следующим образом:

- обучение с учителем;
- обучение без учителя;
- обучение с частичным привлечением учителя [1].

Машинное обучение с учителем

Машинное обучение с учителем происходит с помощью обучения на данных, заранее разделенных по категориям. Набор данных используется в качестве основы для прогнозирования классификации других немаркированных данных с помощью алгоритмов машинного обучения.

Из задач машинного обучения с учителем можно выделить два типа: задачи линейной регрессии и классификации. Линейная регрессия – это метод обучения с учителем, обычно используемый для прогнозирования и поиска взаимосвязей между количественными данными. Это один из самых ранних методов обучения, который до сих пор широко используется. Под классификацией подразумевается группировка выходных данных по заранее определенным классам [2].

Машинное обучение без учителя

Обучение без учителя – не столько автоматизация решений и прогнозов, сколько определение закономерностей и взаимосвязей в данных. Обучение происходит с помощью немаркированных данных. Алгоритмы обучения без учителя группируют данные в немаркированные наборы, при этом перечень таких наборов заранее не задан.

Машинное обучение с частичным привлечением учителя

Такое обучение предполагает «золотую середину» между обучением с учителем и обучением без учителя. Во время обучения он использует меньший размеченный набор данных для проведения классификации и извлечения признаков из большего неразмеченного набора данных.

1.4 Отбор терминов

Отбор терминов – это процесс уменьшения количества входных переменных при обучении алгоритма и создании модели.

Некоторые задачи прогнозного моделирования имеют большое количество переменных, которые могут замедлить разработку и обучение алгоритма, так как требуют значительного количества системной памяти. Кроме того, производительность некоторых моделей может ухудшиться при включении входных переменных, не относящихся к целевой классу.

Отбор терминов в первую очередь направлен на удаление из модели неинформативных или избыточных предикторов, которые могут помешать корректному обучению и добавить неопределенности к прогнозам, а также снизить общую эффективность модели.

1.4.1 Документная частота

Документная частота – метод отбора терминов, который является одним из наиболее простых и эффективных. Данный метод основан на том соображении, что значительное число терминов коллекции встречаются в малом числе документов, а наибольшую информативность имеют термины со средней или даже высокой частотой, если предварительно были удалены стоп-слова. То есть TF – это частота слова, мера того, насколько важно слово для документа.

$$TF(t, d) = \frac{\text{количество терминов в документе}}{\text{количество всех слов в документе}} \quad (1)$$

Также существует такая мера как DF, которая определяет важность документа во всем корпусе. Она схожа с TF, но определяет количество документов, в которых присутствует интересующий нас термин.

IDF – это величина, обратная частоте документа, которая измеряет информативность термина t . При вычислении IDF будет очень низким для наиболее часто встречающихся слов, таких как стоп-слова.

$$IDF(t) = \log\left(\frac{N}{DF + 1}\right) \quad (2)$$

Скомбинировав TF и IDF, получается TF-IDF – правильная мера для оценки того, насколько важно слово для документа в коллекции или корпусе.

$$TF_{IDF}(t, d) = TF(t, d) * \log\left(\frac{N}{DF + 1}\right) \quad (3)$$

1.4.2 Взаимная информация

Взаимная информация рассчитывается между двумя переменными и измеряет уменьшение неопределенности для одной переменной при известном значении другой переменной.

Величина взаимной информации термина t и категории c :

$$MI(t_k, c_j) = \log_2 \frac{P(t_k, c_j)}{P(t_k) \times P(c_j)} = \log_2 P(t_k | c_j) - \log_2 P(t_k) \quad (4)$$

Один из недостатков метода взаимной информации заключается в подверженности влиянию безусловной вероятности терминов, то есть при условии, что у двух терминов имеется одинаковая условная вероятность, более высокое значение MI будет у более редко встречающегося в тексте. Следовательно, значения взаимной информации нельзя сравнивать для терминов с существенно различающейся частотой встречаемости в документах.

1.4.3 Информационная выгода

IG – это важный метод отбора терминов, который измеряет, насколько термин информативен для класса. Информационная выгода вычисляет количество информации о принадлежности категории c , которое несёт наличие/отсутствие термина t .

$$IG(t_k, c_j) = \sum_{c \in \{c_j, \bar{c}_j\}} \sum_{t \in \{\bar{t}_k, t_k\}} P(t, c) \times \log_2 \frac{P(t, c)}{P(t) \times P(c)}, \quad (5)$$

где \bar{c}_j – все категории, кроме c_j ; \bar{t}_k, t_k – признаки наличия и отсутствия термина t_k соответственно.

IG достигает максимальных значений, когда термин является индикатором категории, то есть присутствует тогда и только тогда, когда документ принадлежит соответствующему классу. Информационная выгода равна нулю в тех случаях, когда распределение термина в категории соответствует распределению термина в коллекции [8].

1.4.4 Критерий хи-квадрат

Хи-квадрат – это статистический метод, используемый для измерения разницы между ожидаемыми частотами и наблюдаемыми частотами для двух событий. Если термин X и класс Y независимы, то $P(XY)=P(X)P(Y)$.

Данная величина вычисляется по формуле:

$$CHI(t_k, c_j) = \sum_{c \in \{c_j, \bar{c}_j\}} \sum_{t \in \{t_k, \bar{t}_k\}} \frac{(P(t, c) - P_{exp}(t, c))^2}{P_{exp}(t, c)} \quad (6)$$

По величине CHI можно сказать, насколько ожидаемая и наблюдаемая вероятности отклоняются друг от друга. Также хи-квадрат принимает значение 0, если термин и категория независимы. Вычисление хи-квадрата изначально производится для каждой категории отдельно, и только потом получают его глобальное значение.

1.5 Классификация текста

Классификация текста (маркировка текста, категоризация текста) – это процесс разбиения текста на определенные группы. Используя обработку естественного языка (NLP), текстовые классификаторы могут автоматически анализировать текст, а затем назначать набор предопределенных тегов или категорий на основе его содержимого [3].

Неструктурированный текст присутствует повсюду, например, в электронных письмах, чатах, на веб-сайтах и в социальных сетях, но извлечь пользу из этих данных сложно, если они не организованы определенным образом [4]. Также классификация текста становится все более важной частью бизнеса, поскольку позволяет легко получать информацию из данных и автоматизировать бизнес-процессы. Раньше это было сложной и дорогостоящей процедурой, так как требовалось время и ресурсы для сортировки данных вручную или создания правил, которые трудно поддерживать. Текстовые классификаторы с NLP оказались отличной альтернативой для быстрого, экономичного и масштабируемого структурирования текстовых данных.

Вот некоторые из наиболее распространенных примеров и вариантов использования автоматической классификации текста:

- анализ тональности текста;
- определение категории документа;
- определение темы текста;
- определение языка, на котором написан текст.

Процесс создания системы классификации очень похож на процесс создания других систем с применением машинного обучения [5]:

1. необходимо собрать коллекцию документов для обучения классификатора;
2. каждый документ из обучающей коллекции нужно представить в виде вектора признаков;
3. для каждого документа нужно указать «правильный ответ», т.е. тип документа, по этим ответам и будет обучаться классификатор;
4. выбор алгоритма классификации и обучение классификатора;
5. использование полученной модели.

1.6 Алгоритм наивной Байесовской классификации

Наивный Байесовский классификатор является простым вероятностным классификатором, основанным на применении теоремы Байеса со строгими (то есть «наивными») предположениями о независимости.

Теорема Байеса представляет собой уравнение, описывающее взаимосвязь условных вероятностей статистических величин. В байесовской классификации интерес представляет нахождение вероятности той или иной категории в зависимости от некоторых наблюдаемых характеристик.

Целью классификации является поиск наилучшего класса для документа, то есть имеющего наибольшую апостериорную вероятность $P(c_i|d_j)$:

$$c^* = \arg_{c_j \in \mathcal{C}} \max P(c_i|d_j), \quad (7)$$

где $d_i \in \Omega, c_j \in \mathcal{C}$

По формуле Байеса:

$$P(c_i|d_j) = \frac{P(c_i)P(d_i|c_j)}{P(d_i)} \approx P(c_j)P(d_i|c_j), \quad (8)$$

где $P(c_j)$ – априорная вероятность, что документ принадлежит c_j , $P(d_i|c_j)$ – вероятность встретить документ типа d_i среди документов, класса c_j .

Поскольку $P(d_i)$ не влияет на выбор класса, итоговое ранжирование классов по априорной вероятности можно провести без учёта знаменателя в формуле (2).

Обучение наивных Байесовских классификаторов может производиться очень эффективно исходя из точной природы вероятностной модели. В качестве одного из методов оценки параметров для наивных Байесовских моделей в прикладных приложениях используется метод максимального правдоподобия, позволяющий проводить работу с байесовской моделью, не обращая внимания на Байесовскую вероятность и байесовские методы.

Несмотря на то, что классификатор имеет очень упрощенные условия и наивный вид, он хорошо показывает себя в работе с самыми разнообразными данными.

Достоинством наивного Байесовского классификатора является малое количество данных, необходимых для обучения, оценки параметров и классификации [6].

1.7 Алгоритм Роккио

Алгоритм Роккио был разработан с использованием модели векторного пространства, в нем документы рассматриваются в векторном пространстве терминов, где происходит поиск границ между классами. Классы представлены множеством точек, каждая из которых равноудалена от центроидов классов. Центроид – усреднённый вектор, центр масс членов класса. Вычисляется центроид как:

$$\overrightarrow{\mu_{c_j}} = \frac{1}{|\mathcal{D}_{c_j}|} \sum_{i:d_i \in c_j} \overrightarrow{d_i}, \quad (9)$$

где \mathcal{D}_{c_j} – множество документов класса c_j

Граница, проходящая между двумя классами в многомерном пространстве терминов, имеет вид гиперплоскости:

$$\vec{w}^T \vec{x} = b, \quad (10)$$

$$\vec{w} = \overrightarrow{\mu_1} - \overrightarrow{\mu_2}, \quad (11)$$

$$b = \frac{1}{2} (\|\overrightarrow{\mu_1}\|^2 - \|\overrightarrow{\mu_2}\|^2), \quad (12)$$

где b – константа, \vec{w} – вектор нормали.

Алгоритм классификации Роккио заключается в классификации точки в соответствии с областью, в которую она попадает. Это эквивалентно поиску центроида, к которому образ нового документа ближе всего [7]. На рисунке 1 представлена иллюстрация работы алгоритма Роккио, где символом «звездочка» обозначен интересующий нас документ, который находится ближе всего к классу «Китай»:

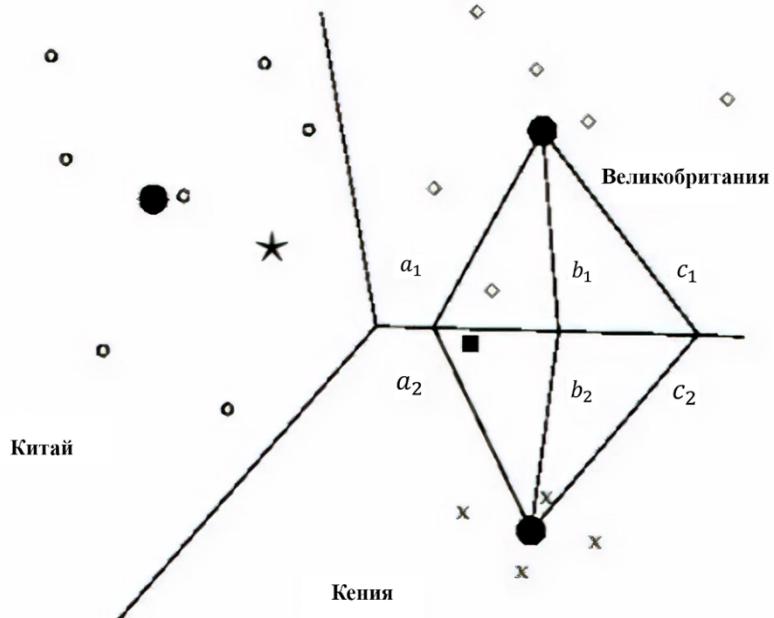


Рис.1 – Иллюстрация работы алгоритма Роккио

Данный метод предполагает, что классы имеют форму сфер с одинаковыми радиусами. Однако при невыполнении этого условия алгоритм может привести к некорректным результатам.

Несмотря на эффективность и простоту реализации, данный алгоритм не может классифицировать мультимодальные классы и их отношения (например, в случае нечетких дубликатов в выборке) [9].

1.8 Алгоритм k-ближайших соседей

Данный метод основывается на гипотезе компактности векторного пространства. Согласно ей, в пространстве терминов образуются компактные непересекающиеся области из документов одного класса. Следовательно, документ будет иметь такую метку класса, как и документы, окружающие его. То есть алгоритм k-ближайших соседей относит документ к классу, находящемуся ближе всего. На рисунке 2 показано, что k – это количество объектов-соседей классов, которые захватываются при рассмотрении [10].

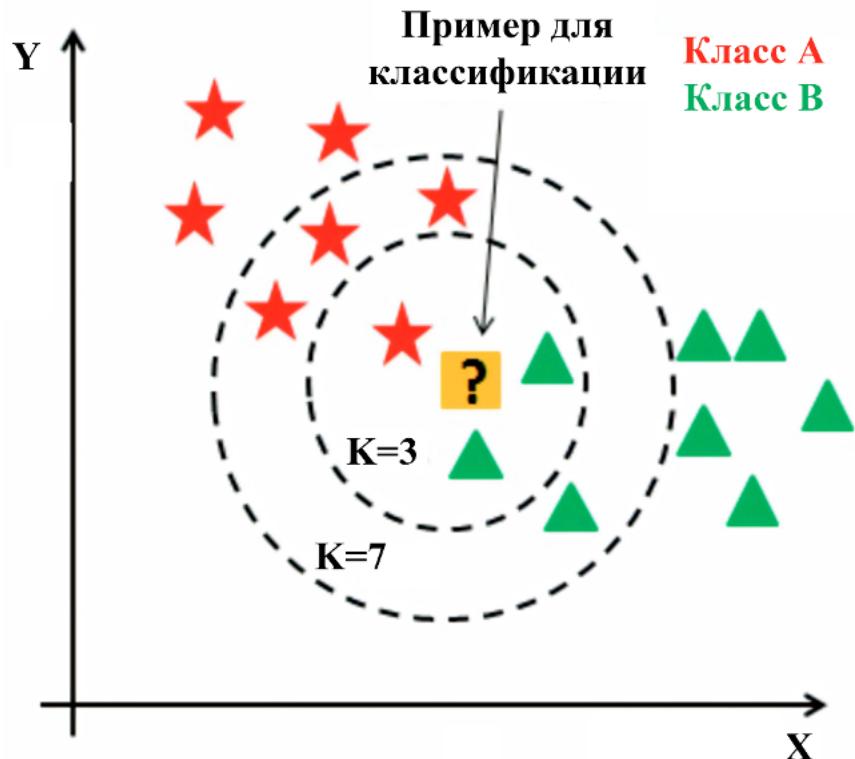


Рис.2 – Количество «соседей», которые рассматриваются единовременно

Стоит учесть, что при $k = 1$ алгоритм неустойчив, так как классификация зависит только от одного документа, что может привести к некорректным результатам вследствие нетипичности. Для повышения точности могут

учитываться веса «голосов» соседей, которые используются для ранжирования классов:

$$\text{ранг}(c_j, d) = \sum_{d' \in S_k} I_{c_j}(d') \text{sim}(\bar{d}', \bar{d}) , \quad (13)$$

где $I_{c_j}(d') = 1$, если $d' \in c_j$, иначе $I_{c_j}(d') = 0$; $\text{sim}(\bar{d}', \bar{d})$ – косинусная мера близости.

В результате работы алгоритма документ присваивается классу с наибольшим рангом. В случае, если одинаковое число соседей принадлежит разным классам, выбирается класс с наиболее близкими соседями.

Несмотря на свою относительную алгоритмическую и семантическую простоту, метод показывает хорошие результаты при удачно выбранной функции расстояния. Главными его недостатками являются высокая вычислительная сложность и тот факт, что метод применим к выпуклой форме классов, но плохо подходит к невыпуклой форме.

1.9 Алгоритм опорных векторов

Задача алгоритма машины опорных векторов – найти гиперплоскость в N-мерном пространстве, которая четко классифицирует точки данных, где N – количество признаков. В алгоритме опорных векторов определение решающей поверхности происходит за счёт небольшого подмножества документов, элементы которого называются опорными векторами. На рисунке 3 изображены опорные вектора и разделяющая поверхность.

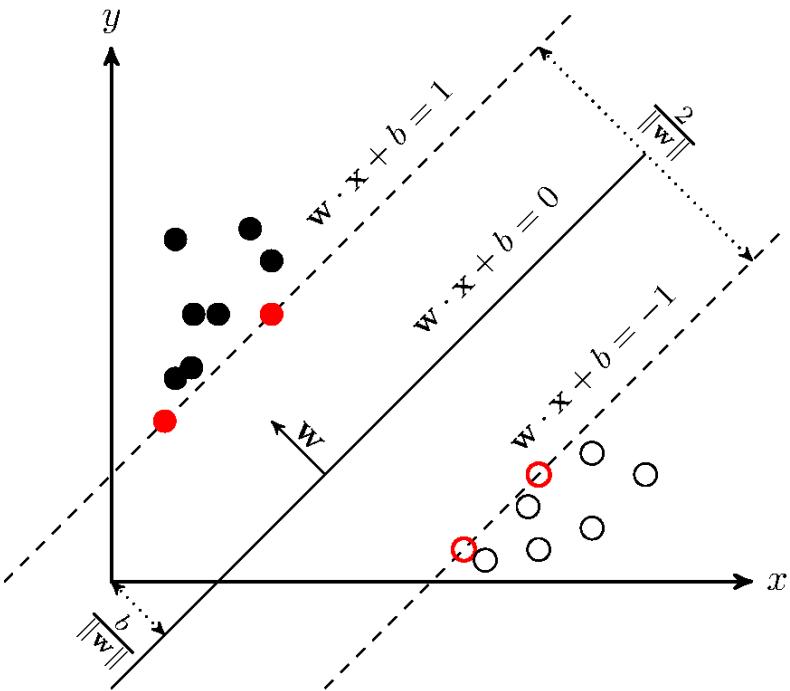


Рис.3 – Опорные вектора и разделяющая поверхность

Разделяющая поверхность задаётся уравнением:

$$\vec{w}^T \vec{x} = -b, \quad (14)$$

где \vec{w} – вектор нормали к разделяющей плоскости, b – параметр сдвига, \vec{x} – текстовый документ.

Линейный классификатор задаётся следующим образом:

$$f(\vec{x}) = \text{sign}(\vec{w}^T \vec{x} + b) \quad (15)$$

При соответствии одному классу результат классификатора равен -1 , при соответствии другому равен $+1$.

Полоса, проводимая между опорными векторами двух классов, имеет максимальную ширину, которая называется геометрическим зазором. Также существует зазор классификации – расстояние между поверхностью и ближайшей точкой данных, причем поверхность ищется с учётом максимизации этого зазора. Величина геометрического зазора должна вдвое превышать значение r :

$$r = \frac{\vec{w}^T \vec{x} + b}{\|\vec{w}\|} \quad (16)$$

Необходимо, чтобы для всех точек $(\vec{x}_i, y_i) \in \Omega$ соблюдалось неравенство:

$$y_i (\vec{w}^T \vec{x}_i + b) \geq 1 \quad (17)$$

Следовательно, данный алгоритм сводится к поиску наиболее оптимальных \vec{w} и b , которые бы удовлетворяли условиям.

1. Величина $\frac{\|\vec{w}\|}{2} = \frac{\vec{w}^T \vec{w}}{2}$ достигает минимальных значений.
2. При всех $(\vec{x}_i, y_i) \in \Omega$ выполняется вышеприведенное неравенство (17).

В результате получается задача минимизации квадратичной функции.

Алгоритм опорных векторов наиболее предпочтителен в пространствах большой размерности, а также в случаях, когда существует четкое разделение между классами [11].

1.10 Алгоритм деревьев принятия решений

Алгоритм деревьев принятия решений – машинное обучение, при котором данные непрерывно разделяются в соответствии с определенным параметром.

Дерево состоит из следующих элементов:

- узлы – термины документов;
- листья – метки классов;
- рёбра – веса терминов.

Пример дерева принятия решения представлен на рисунке 4.

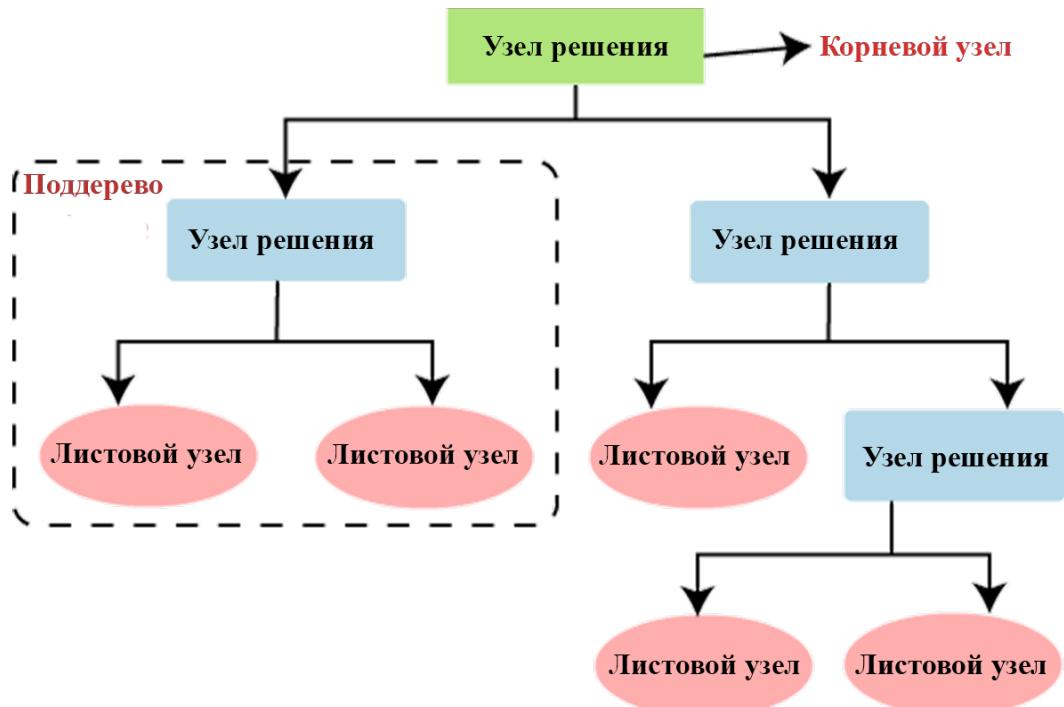


Рис.4 – Дерево принятия решений

При обучении данного алгоритма придерживаются следующей стратегии.

1. Рассматривается множество документов и проверяется, все ли документы имеют одну и ту же метку.
2. В случае, если не все документы имеют одинаковую метку, производится поиск термина с максимальной разделяющей способностью. Для выбора такого термина используется информационная энтропия:

$$E(A, Q) = - \sum_{i=1}^m \frac{m_i}{n} \times \log_2 \frac{m_i}{n}, \quad (18)$$

где A – множество из n элементов, Q – атрибут, которым обладают элементы из множества A , m_i – количество случаев, когда реализуется i -ое значение.

3. После нахождения разделяющего термина производится разделение документа на два подмножества и дальнейшее построение их поддеревьев.
4. Ранее приведенные действия повторяются до тех пор, пока в подмножестве не окажутся документы только одного класса. При

нахождении такого множества, его добавляют в лист с соответствующей меткой.

Одним из главных преимуществ такого метода классификации является легкость в интерпретации результата, а также меньший объем фильтрации. Однако деревья решений имеют неустойчивый характер предсказаний, то есть небольшое изменение входных данных влечет за собой значительное изменение структуры дерева.

1.11 Метод логистической регрессии

Логистическая регрессия является методом построения линейного классификатора, которая позволяет оценить апостериорные вероятности принадлежности объектов классам. Данный метод преобразовывает входные данные, используя логистическую сигмоидальную функцию, и возвращает значение вероятности принадлежности исследуемого объекта к одному из классов. Значение логистической регрессии всегда находится в промежутке от 0 до 1, так как данный алгоритм основан на концепции вероятности.

Логистическая регрессия бывает следующих типов:

- бинарная – определяется, к какому из двух классов принадлежат исследуемые данные;
- многолинейная – определяется принадлежность объекта к одному из нескольких классов, количество которых превышает 2.

Функцией, позволяющей сопоставить предсказанные моделью значения со значением вероятности, является сигмоидальная функция, представление которой приведено ниже на рисунке 5.

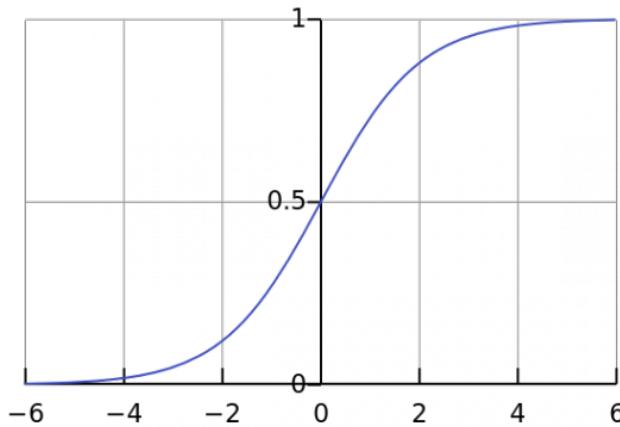


Рис.5 – Сигмоидальная функция

Формула сигмоидальной функции имеет вид:

$$f(x) = \frac{1}{1+e^{-x}} \quad (19)$$

Также в алгоритме логистической регрессии используется функция, называемая функцией потерь, целью использования которой является минимизация значения ошибки классификации, позволяющая разработать более точную итоговую модель. Функция потерь для логистической функции выглядит следующим образом:

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{если } y = 1 \\ -\log(1 - h_\theta(x)), & \text{если } y = 0 \end{cases}, \quad (20),$$

где h_θ – гипотеза логистической регрессии.

При этом приведенные выше две функции можно представить в следующем виде:

$$J(\theta) = -\frac{1}{m} \sum \left[y^{(i)} \log(h_\theta(x(i))) + (1 - y^{(i)}) \log(1 - h_\theta(x(i))) \right] \quad (21)$$

Для снижения значения функции потерь $J(\theta)$ используется градиентный спуск. Для достижения этого необходимо запустить функцию градиентного спуска для каждого параметра.

$$\theta_j := \theta_j - \alpha \frac{d(J(\theta))}{d(\theta_j)} \quad (22)$$

Вычисление будет производиться до тех пор, пока не будет достигнуто минимальное значение функции потерь.

В качестве достоинств логистической регрессии можно отметить простоту реализации, а также легкость в обновлении модели для отражения новых данных.

1.12 Вывод

В данном разделе были изучены принципы машинного обучения и изучены алгоритмы классификации текста с учителем. Для отбора терминов решено использовать метод документной частоты. В данной работе решено рассматривать исключительно методы нечёткой классификации, так как они позволяют рассчитать степень уверенности классификатора в присваиваемой метке класса. В результате сравнительного анализа был выбран алгоритм наивной Байесовской классификации, так как данный метод подходит для решения задач мультиклассового прогнозирования, а также отличается высокой скоростью проведения классификации. В качестве дополнительного метода классификации выбор пал на логистическую регрессию по причине устойчивости характера предсказаний.

2 Конструкторский раздел

2.1 Декомпозиция разрабатываемого метода

Разрабатываемый метод классификации платежей состоит из четырёх этапов.

1. Формирование входных данных.
2. Отбор терминов.
3. Обучение алгоритма классификации.
4. Тестирование обученной модели.

Ниже представлена IDEF0-диаграмма разрабатываемого метода на рисунке 6.

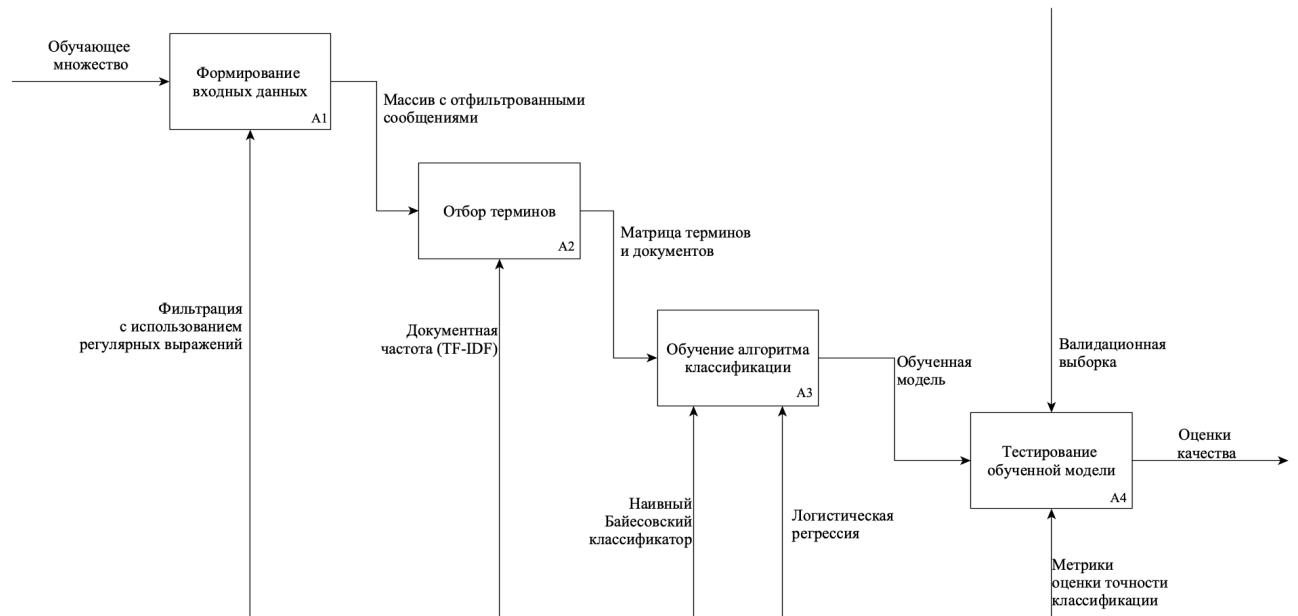


Рис.6 – IDEF0-диаграмма

Ниже рассмотрены и подробно описаны все разрабатываемые этапы.

2.2 Формирование входных данных

Данный этап предназначен для подготовки входных обучающих данных. На вход поступают предложения из Excel-таблицы, их вид представлен в таблице 1.

Таблица 1 – Пример входных данных

Код операции	Наименование	Назначение платежа
1800	Перевод средств на р/с	Оплата страхового взноса по Договору страхования № 0122130464484 от 05.09.2018, ФИО страхователя Ермошкина Екатерина Ивановна, сумма цифрами 550,00.
14	Перевод средств пенсионных накоплений в 153	Доход от инвестирования (срочная пенсионная выплата) по договору № 22-03У008 от 08.10.2003 г.

Для того, чтобы впоследствии корректно обучить алгоритм классификации, необходимо удалить термины, которые не несут смысловой нагрузки.

Данный этап состоит из следующих стадий:

- удаление всех слов, содержащих цифры;
- удаление названий компаний;
- удаление дат;
- удаление инициалов;
- удаление символов, мешающих правильному распознаванию слов (таких как точки, запятые, скобки и т.д.).

Также для правильного учета количества слов в документах необходимо привести в единому виду сокращения, такие как «з/п», «з/пл», «з/плат» и «зар. плата».

Далее все оставшиеся слова приводятся к начальной форме. После этого каждое слово проверяется на наличие в словаре стоп-слов, где содержатся предлоги, союзы, числительные, местоимения, имена, отчества, названия компаний без кавычек, отдельно стоящие символы (“%”, “/”, “№”, “-”).

После выполнения этапа будет получен массив предложений, каждое из которых содержит те слова в начальной форме, которые несут смысловую нагрузку для данного вида задачи.

2.3 Отбор терминов

Для реализации данного модуля используется терм-документная частота (TF-IDF) в качестве метода отбора терминов. Вычисление терм-документной частоты состоит из трёх этапов.

1. Вычисление TF.
2. Вычисление IDF.
3. Перемножение полученных значений TF и IDF и получение меры TF-IDF.

Вычисление TF (term frequency) позволяет оценить важность термина в пределах каждого отдельно взятого документа. Для вычисления TF необходимо воспользоваться формулой:

$$TF(t, d) = \frac{n_t}{\sum_k n_k} , \quad (23)$$

где n_t – количество вхождений термина t в документ, $\sum_k n_k$ – общее количество слов в документе.

IDF (inverse document frequency) – это инверсная документная частота, которая необходима для уменьшения веса широко употребляемых терминов.

$$IDF(t_i, D) = \log\left(\frac{|D|}{|D_i|}\right) , \quad (24)$$

где $|D|$ – общее количество документов, а $|D_i \in D|$ – число документов, где t_i встретилось хотя бы один раз.

Следовательно, мера TF-IDF вычисляется следующим образом:

$$TF - IDF(t, D) = TF(t, d) \times IDF(t_i, D) \quad (25)$$

Благодаря использованию такой комбинированной меры наибольший вес получат документы, которые часто встречаются в пределах одного документа, но редко употребляются во всём документном корпусе.

В результате выполнения этапа получится матрица, строки которой соответствуют документам корпуса, а столбцы – всем терминам, которые встречаются в документах. Матрица содержит получившиеся TF-IDF для каждого конкретного термина.

2.4 Обучение алгоритма классификации

В качестве алгоритмов классификации были выбраны наивный Байесовский алгоритм и логистическая регрессия. Каждый из них будет реализован отдельно, а затем проведен сравнительный анализ на основе экспериментальных данных. Рассмотрим их по порядку.

2.4.1 Алгоритм наивной Байесовской классификации

Алгоритм наивной Байесовской классификации осуществляет поиск максимальной апостериорной вероятности, которая будет означать наилучший класс для документа. Апостериорная вероятность вычисляется по формуле:

$$P(c_i|d_j) = \frac{P(c_i)P(d_i|c_j)}{P(d_i)} \approx P(c_j)P(d_i|c_j), \quad (26)$$

где $P(c_j)$ – априорная вероятность, что документ принадлежит c_j , $P(d_i|c_j)$ – вероятность встретить документ типа d_i среди документов, класса c_j .

В формуле (26) можно не учитывать знаменатель, так как он не повлияет на выбор класса. Тогда $P(d_i|c_j)$ можно вычислить как:

$$P(d_i|c_j) = \prod_{k=1}^{|\tau_{d_i}|} P(t_k|c_j), \quad (27)$$

где τ_k – множество терминов документа d_i , $P(t_k|c_j)$ – оценка вклада термина в $d_i \in c_j$.

Следовательно, решающее правило имеет вид:

$$c^* = \arg_{c_j \in \mathcal{C}} \max P(c_j) \prod_{k=1}^{|\tau_{d_i}|} P(t_k|c_j) \quad (28)$$

Однако может быть потеря значащих рядов при умножении условных вероятностей [6]. В таком случае используют логарифм вероятностей, так как логарифм является монотонно возрастающей функцией. Из-за этого c_j с максимальным значением логарифма останется наиболее вероятным. Следовательно, уравнение принимает вид:

$$c^* = \arg_{c_j \in \mathcal{C}} \max \left[\log P(c_j) + \sum_{k=1}^{|\tau_{d_i}|} \log P(t_k | c_j) \right] \quad (29)$$

Также на практике применяют сглаживание, так как в новых документах могут встретиться термины, не задействованные в обучающем множестве:

$$P(t_k | c_j) = \frac{tf(t_k, c_j) + 1}{\sum_{i=1}^{|\tau|} (tf(t_i, c_j) + 1)}, \quad (30)$$

где $tf(t_i, c_j)$ – частота встречаемости термина.

Схема работы данного алгоритма представлена на рисунке 7.

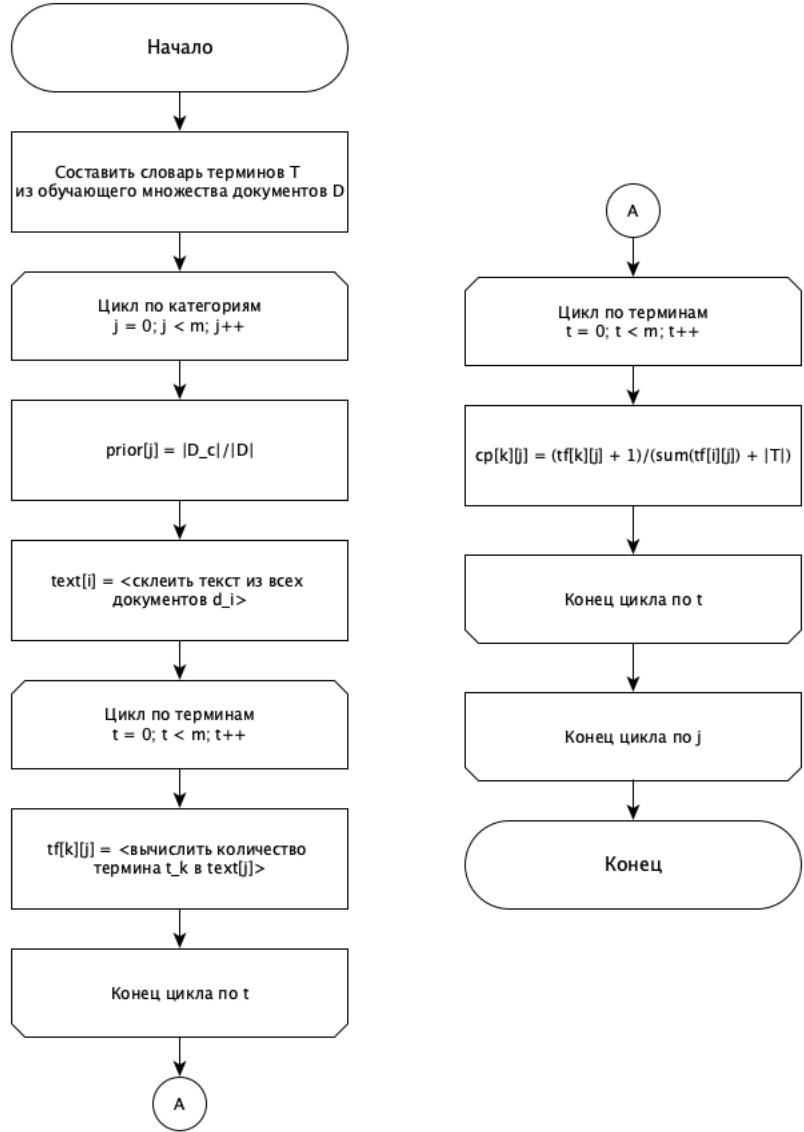


Рис.7 – Схема алгоритма наивной Байесовской классификации

2.4.2 Логистическая регрессия

Логистическая регрессия является одним из фундаментальных методов классификации и базируется на концепции вероятности. Так как результатами логистической регрессии являются вероятности принадлежностей документов к тем или иным классам, их значения остаются в интервале от 0 до 1, то есть:

$$0 \leq h_{\theta}(X) \leq 1, \quad (31)$$

где h_{θ} – гипотеза.

В случае линейной регрессии гипотеза вычисляется по следующей формуле:

$$h_{\theta}(X) = \beta_0 + \beta_1 X, \quad (32)$$

где β_0 и β_1 – коэффициенты линейного уравнения, определяющие положение прямой в пространстве.

Однако, в логистической регрессии формула принимает вид:

$$h_{\theta}(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \quad (33)$$

Также необходимо выбрать пороговое значение, которое позволит сопоставить предсказанные моделью значения со значением вероятности. Для этого понадобится сигмоидальная функция (рисунок 5), уравнение которой принимает вид:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (34)$$

Для того, чтобы оптимизировать классификацию, необходимо создать функцию потерь и минимизировать её, таким образом разработав наиболее точную модель с минимальной ошибкой. Функция потерь для логистической функции выглядит следующим образом:

$$Cost(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)), & \text{если } y = 1 \\ -\log(1 - h_{\theta}(x)), & \text{если } y = 0 \end{cases}, \quad (35)$$

где h_{θ} – гипотеза логистической регрессии.

Указанные выше функции можно привести к виду:

$$J(\theta) = -\frac{1}{m} \sum \left[y^{(i)} \log(h_{\theta}(x(i))) + (1 - y^{(i)}) \log(1 - h_{\theta}(x(i))) \right] \quad (36)$$

Минимизация функции потерь происходит с помощью градиентного спуска. Градиентный спуск – метод нахождения локального экстремума функции с помощью движения вдоль градиента.

$$\theta_j := \theta_j - \alpha \frac{d(J(\theta))}{d(\theta_j)} \quad (37)$$

Необходимо подобрать такое значение α , чтобы при каждом шаге величина θ_j уменьшалась (то есть α не должно быть слишком большим). Однако нужно учитывать, что при очень маленьких значениях α обучение будет происходить очень долго. Графическое представление градиентного спуска приведено на рисунке 8.

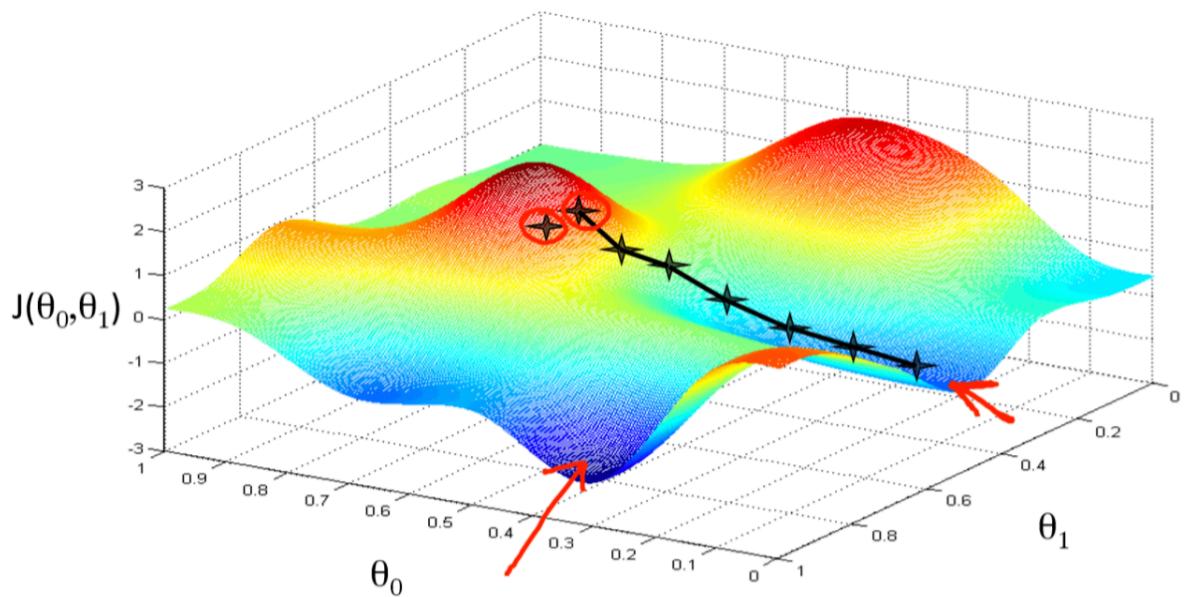


Рис.8 – Градиентный спуск

Схема работы данного алгоритма представлена на рисунке 9.



Рис.9 – Схема алгоритма логистической регрессии

2.5 Тестирование обученной модели

Выбор правильной метрики имеет решающее значение при оценке моделей машинного обучения. Существуют различные способы оценки модели классификации:

- аккуратность (accuracy);
- точность (precision);
- полнота (recall);
- F-мера (f1-score);
- макро-усреднение (macro-averaging) и взвешенное усреднение (weighted averaging).

Ниже рассмотрим каждую из них.

2.5.1 Аккуратность

Является наиболее простой метрикой, которая определяет долю правильных ответов результатов модели. Рассчитывается следующим образом:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (38)$$

где TP – истинно положительные ответы, TN – истинно отрицательные ответы, FP – ложно положительные ответы, FN – ложно отрицательные ответы.

Однако данная метрика не подходит в случае, если объем данных между классами распределен неравномерно [12].

2.5.2 Точность

Метрика точности показывает долю правильных ответов относительно всех положительных ответов в пределах одного класса. Формула вычисления точности приведена ниже:

$$precision = \frac{TP}{TP + FP} \quad (39)$$

Учитывая данную метрику, можно сказать о количестве ложно положительных предсказаний и сделать вывод о достаточности объема данных других классов.

2.5.3 Полнота

Полнота – это доля истинно положительных классификаций. С её помощью можно сказать какую долю объектов, которые действительно относятся к данному классу, модель предсказала верно.

$$recall = \frac{TP}{TP + FN} \quad (40)$$

Полнота показывает способность алгоритма обнаруживать данный класс.

2.5.4 F-мера

F-мера – это гармоническое среднее между точностью и полнотой. Данная метрика позволяет найти баланс между этими метриками, так как на практике зачастую невозможно достичь максимальной точности и полноты одновременно.

$$f1 = \frac{2 \times precision \times recall}{precision + recall} \quad (41)$$

Метрика достигает своего максимума при наибольших значениях полноты и точности, и стремится к нулю, если один из аргументов близок к нулю.

2.5.5 Макро-усреднение и взвешенное усреднение

Для подсчёта макро-усреднения таких метрик как точность, полнота и F-мера необходимо воспользоваться следующими формулами:

$$macro - averaged precision = \frac{\sum_{i=1}^n precision_i}{n}, \quad (42)$$

$$macro - averaged recall = \frac{\sum_{i=1}^n recall_i}{n}, \quad (43)$$

$$macro - averaged f1 = \frac{\sum_{i=1}^n f1_i}{n}, \quad (44)$$

где n – количество классов.

Найти взвешенное усреднение можно следующим образом:

$$\text{weighted averaged precision} = \frac{\sum_{i=1}^n \text{precision}_i * n_i}{\sum_{i=1}^n n_i}, \quad (45)$$

$$\text{weighted averaged recall} = \frac{\sum_{i=1}^n \text{recall}_i * n_i}{\sum_{i=1}^n n_i}, \quad (46)$$

$$\text{weighted averaged f1} = \frac{\sum_{i=1}^n f1_i * n_i}{\sum_{i=1}^n n_i}, \quad (47)$$

где n – количество экземпляров каждого класса.

Данные метрики позволяют найти усреднённое значение по точности, полноте и F-мере, что даёт представление о корректности предсказаний классификатора.

2.6 Вывод

В конструкторском разделе был описан метод классификации платежных документов. Продемонстрирована IDEF0-диаграмма разрабатываемого метода, описаны входные и выходные данные для каждого этапа. Также дано подробное описание метода отбора терминов и алгоритмов классификации с выделением основных этапов работы.

3 Технологический раздел

В технологическом разделе описываются средства реализации, которые используются для разработки программного продукта, а также преимущества выбранных инструментов. Приведены минимальные системные требования, необходимые для правильной работы системы. Описывается структура разработанного ПО. Представлено руководство для запуска программы.

3.1 Выбор средств разработки

В данном разделе рассматриваются такие инструменты разработки как язык программирования, среда разработки и используемые библиотеки.

3.1.1 Язык программирования

В качестве языка программирования для разработки программного обеспечения было решено использовать язык Python v3.8. К преимуществам данного языка программирования можно отнести совмещение нескольких парадигм программирования, таких как функциональное, объектно-ориентированное и структурное программирование, а также большое разнообразие представленных библиотек.

3.1.2 Среда разработки

Для разработки ПО в соответствии с выбранным языком программирования в качестве среды разработки использовалась кроссплатформенная IDE PyCharm v2020.3.2. Полную версию данной среды разработки можно получить бесплатно с помощью студенческой лицензии. PyCharm позволяет быстро перемещаться по модулям программного обеспечения, а также легко устранять ошибки с помощью автоматической подсветки синтаксиса и режима debug. Также процесс разработки облегчает автоматическое создание виртуального пространства и быстрая установка новых библиотек и модулей.

3.1.3 Используемые библиотеки

При разработке программного обеспечения использовались следующие библиотеки:

- NumPy – библиотека, поддерживающая работу с большими многомерными массивами и матрицами, а также предоставляющая набор математических функций для работы с этими данными [13];
- Scikit-learn – данная библиотека включает в себя различные алгоритмы машинного обучения и подготовку данных для последующей классификации, поддерживает взаимодействие с NumPy и SciPy [14];
- SciPy – содержит модули для оптимизации, линейной алгебры, интерполяции, специальных функций, БПФ, обработки сигналов и изображений, решателей ODE и других задач, распространенных в науке и технике [15];
- PyMorphy – морфологический анализатор, который позволяет приводить слова на русском языке к нормальной форме (например, к единственному числу, именительному падежу и т.д.). Возвращает грамматическую информацию о слове, при работе использует словарь OpenCorpora, а для незнакомых слов строятся гипотезы [16];
- PyQt – набор библиотек для создания графического интерфейса [17].

3.2 Минимальные требования к вычислительной системе

Для того, чтобы воспользоваться программным продуктом, необходимо ЭВМ с предустановленным интерпретатором Python3. Также необходимо скачать и установить все использованные в проекте библиотеки.

Минимальные требования к вычислительной системе:

- Операционная система – Windows 8, Ubuntu 16.04, MacOS 10.14 Mojave;
- Оперативная память – 4 Гб.

3.3 Структура разработанного ПО

Структура разработанного ПО изображена на рисунке 10.

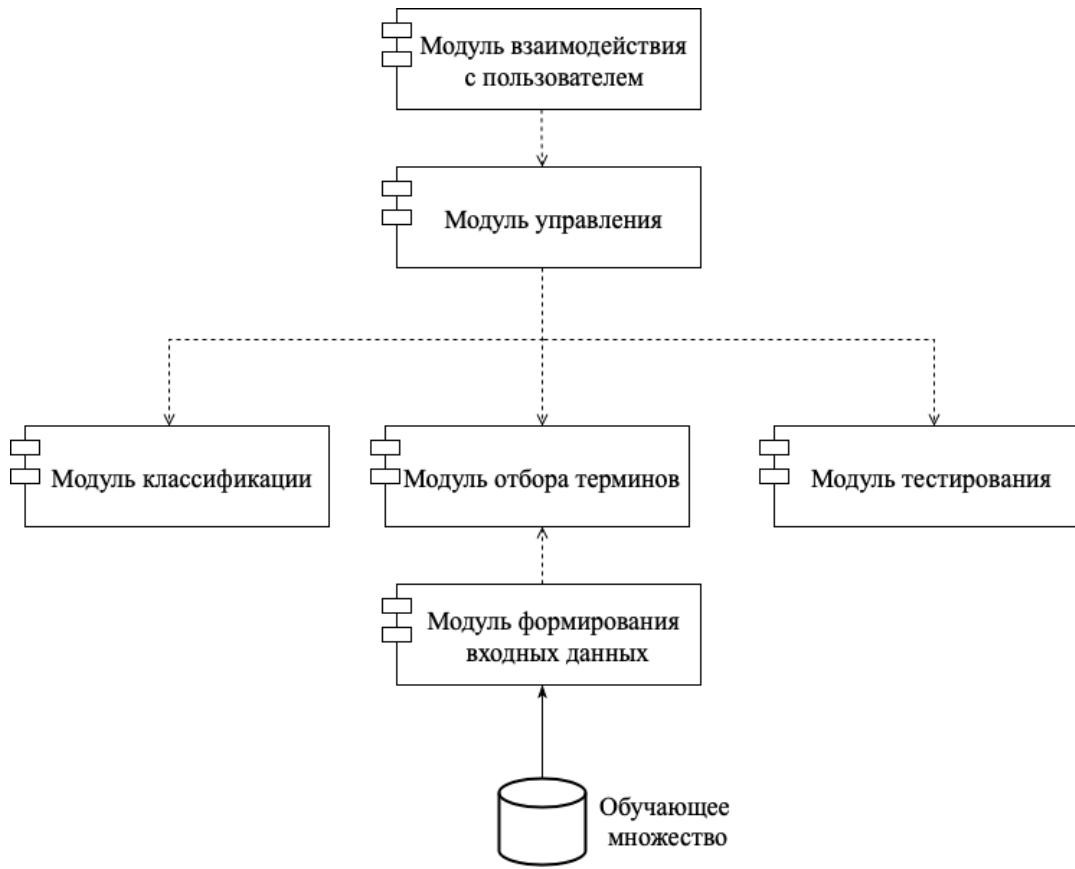


Рис.10 – Диаграмма структуры ПО

Каждый модуль, который изображен на диаграмме, содержит в себе сгруппированные по функциональному значению соответствующие классы. Модуль взаимодействия с пользователем отвечает за пользовательский интерфейс. Модуль управления связывает модули отбора терминов, классификации, тестирования, а также отвечает за координацию их работы.

3.4 Описание классов

Ниже представлены таблицы классов, где приведено описание каждого метода:

Таблица 2 – методы класса FileWorker

Метод	Описание метода
<code>__init__()</code>	Конструктор класса FileWorker

read_data()	Считывает обучающие и тестовые данные из Excel-таблицы
get_names()	Считывает стоп-слова (имена и отчества) из txt-файла для последующей предобработки данных

Таблица 3 – методы класса SelectionOfTerms

Метод	Описание метода
__init__()	Конструктор класса SelectionOfTerms
filter_with_masks()	Фильтрация документов с помощью регулярных выражений
filter_with_numbers()	Фильтрация слов, содержащих цифры
word_in_stop_words()	Обнаружение слов, содержащихся в словаре стоп-слов (названия, имена, отчества, местоимения, союзы и т.д.)
word_in_reductions()	Приведение часто встречающихся сокращений одного слова к единому виду
filter_with_stop_words()	Фильтрация слов, попадающихся в словарях стоп-слов
filter()	Разделение документов на слова и дальнейший вызов методов, фильтрующих данные

Таблица 4 – методы класса TfIdf

Метод	Описание метода
__init__()	Конструктор класса TfIdf
tf_idf()	Вычисление метрики TF-IDF для каждого слова и преобразование в

	матрицу, строка которой соответствует номеру документа, а столбец – термину
--	-----------------------------------------------------------------------------

Таблица 5 – методы класса ClassifierTraining

Метод	Описание метода
<code>__init__()</code>	Конструктор класса ClassifierTraining
<code>naive_bayes()</code>	Обучение наивного Байесовского классификатора
<code>logistic_regression()</code>	Обучение алгоритма логистической регрессии

Таблица 6 – методы класса ClassifierTesting

Метод	Описание метода
<code>__init__()</code>	Конструктор класса ClassifierTesting
<code>clf_testing()</code>	Тестирование обученного классификатора и подсчёт метрик классификации

Таблица 7 – методы класса ClassifierSaver

Метод	Описание метода
<code>__init__()</code>	Конструктор класса ClassifierSaver
<code>save_clf()</code>	Сохранение обученной модели классификатора
<code>load_clf()</code>	Загрузка обученной модели классификатора

Таблица 8 – методы класса Predict

Метод	Описание метода
<code>__init__()</code>	Конструктор класса Predict

predict()	Предсказание результата с использованием обученной модели классификатора
-----------	--------------------------------------------------------------------------

3.5 Установка ПО

Для работы созданного программного обеспечения необходимо установить интерпретатор Python3. Рекомендуется воспользоваться IDE PyCharm. Затем следует создать новое виртуальное окружение. Это можно сделать двумя способами – с помощью терминальных команд или с использованием возможностей IDE PyCharm. Для того, чтобы установить виртуальное окружение через терминал необходимо ввести команды, указанные в листинге 1.

Листинг 1 – Команды для установки виртуального окружения

```
python3 -m pip install --user virtualenv
python3 -m venv env
source env/bin/activate
```

Создать новое виртуальное окружение также можно с помощью инструментов IDE PyCharm, пример представлен на рисунке 11.

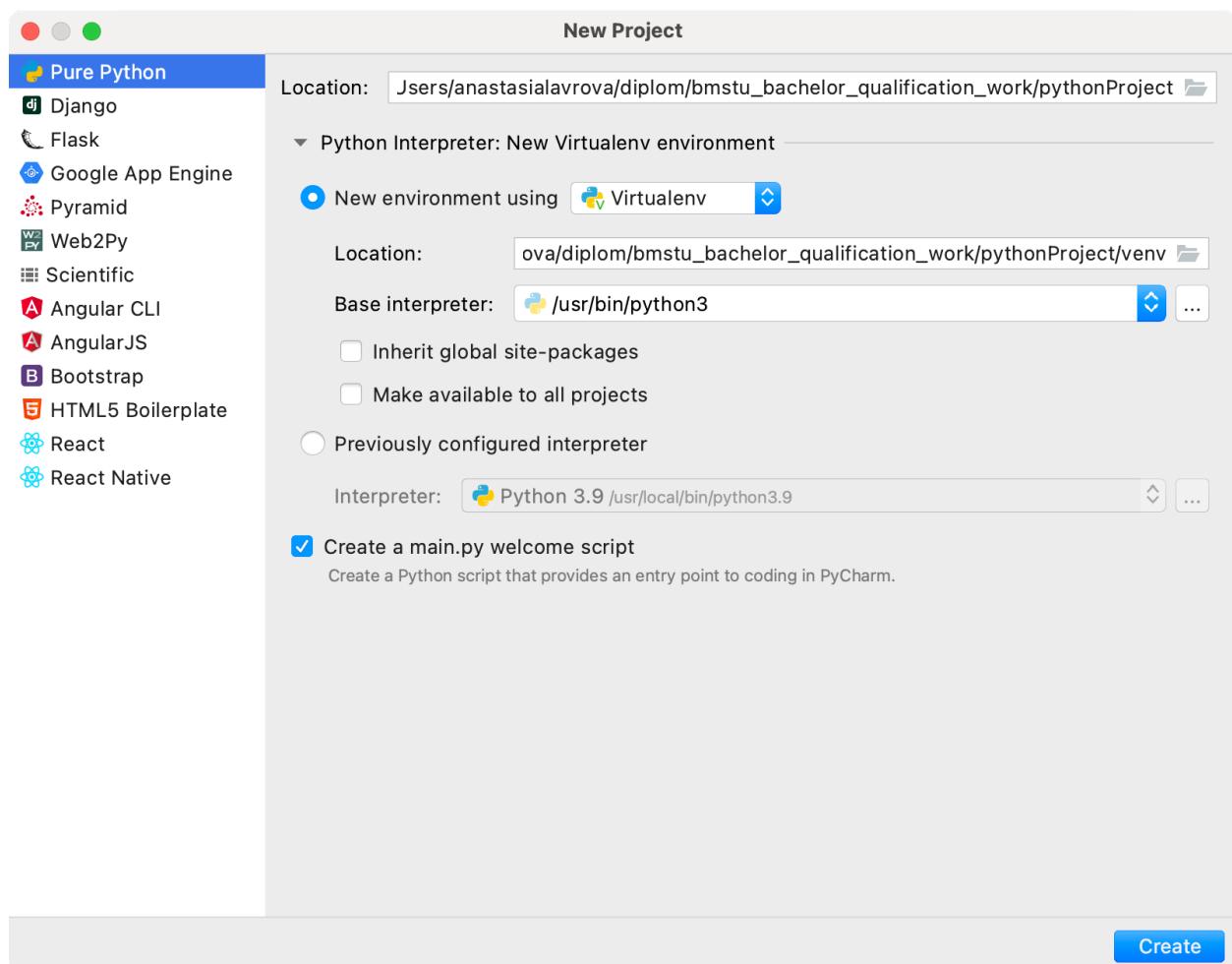


Рис.11 – Создание нового виртуального окружения в PyCharm

Также для запуска программного обеспечения понадобится установить все библиотеки, описанные выше. Это можно сделать с помощью программных средств IDE PyCharm, перейдя во вкладку по пути «File» – «New Projects Settings» – «Preferences for New Projects» и воспользовавшись встроенным обозревателем библиотек. Другой способ установки библиотек заключается в вводе следующих команд, см. листинг 2.

Листинг 2 – Команды для установки библиотек

```
pip install numpy  
pip install -U scikit-learn  
pip install scipy  
pip install pymorphy2  
pip3 install PyQt5
```

3.6 Пример работы программы

На рисунке 12 представлен графический интерфейс пользователя.

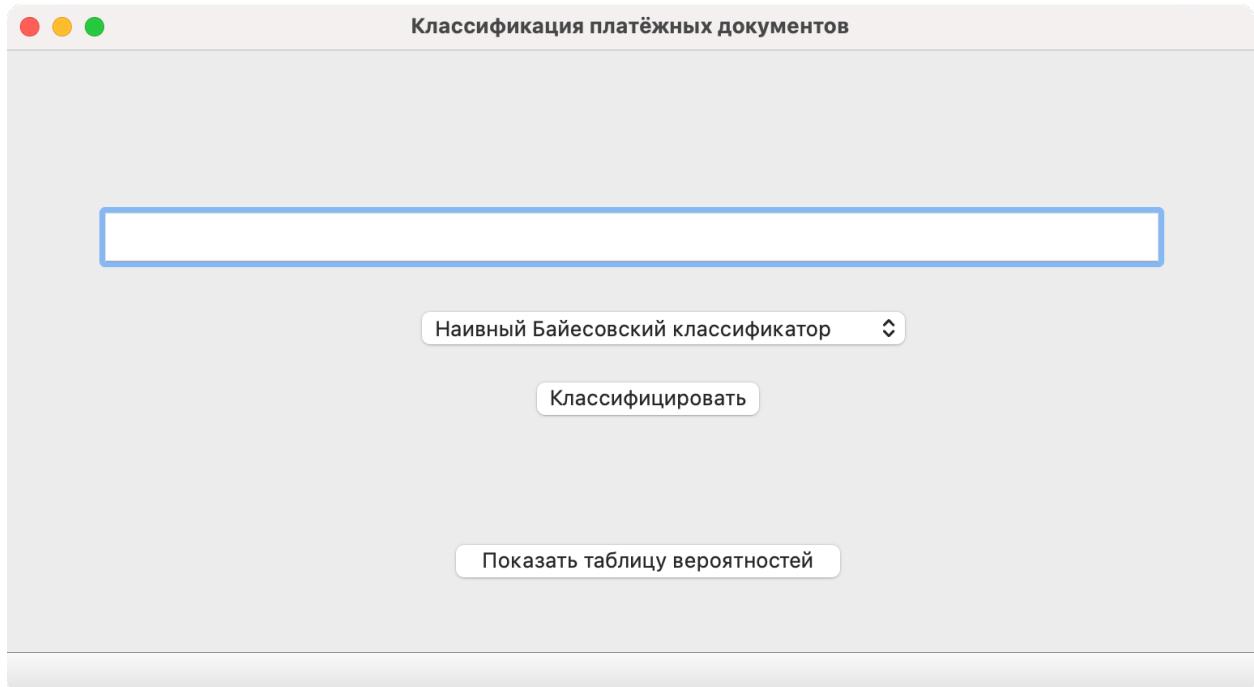


Рис.12 – Графический интерфейс пользователя

Интерфейс состоит из следующих элементов:

- строка ввода предложения, подлежащего классификации;
- переключатель между наивным Байесовским классификатором и логистической регрессией;
- кнопка «Классифицировать», по нажатию на которую будет выполнена классификация данной строки;
- кнопка «Показать таблицу вероятностей», по нажатию на которую будет открыто новое окно, показывающее в формате таблицы класс и вероятность принадлежности строки к нему (пример такого окна представлен ниже на рисунке 13).

	Код операции	Вероятность
1	1386	0.108023
2	1457	0.106212
3	178	0.064058
4	1394	0.044335
5	2295	0.02615
6	1387	0.024423
7	1397	0.021632
8	615	0.018539
9	1396	0.017312
10	490	0.016436
11	489	0.01581
12	927	0.014339
13	1621	0.013602
14	1812	0.012824
15	1385	0.01129
16	1180	0.011094

Рис.13 – Таблица вероятностей

3.7 Тестирование ПО

Было проведено модульное тестирование. Модульные тесты методов, обозначения которых приведены в табл.2-7, приведены в таблице 8.

Таблица 8 – Модульные тесты методов

Название метода	Описание теста	Ожидаемый результат
filter_with_masks()	На вход подаётся строка, из которой должны быть удалены слова или сокращения по регулярным выражениям	На выходе ожидается редуцированная строка
filter_with_numbers()	На вход подаётся строка, из которой должны быть удалены все слова, содержащие цифры	На выходе ожидается редуцированная строка

word_in_stop_words()	На вход подаётся слово	На выходе ожидается флаг 0 или 1, который показывает, присутствие данного слова в стоп-словарях
word_in_reductions()	На вход подаётся слово	На выходе ожидается унифицированный вид сокращения в случае, если слово оказалось таковым
filter_with_stop_words()	На вход подаётся строка, из которой должны быть удалены все стоп-слова	На выходе ожидается редуцированная строка
naive_bayes()	На вход подаётся матрица терминов с весами TF-IDF	На выходе ожидается модель с обученными весами
logistic_regression()	На вход подаётся матрица терминов с весами TF-IDF	На выходе ожидается модель с обученными весами
predict()	На вход подаётся строка	На выходе ожидается массив вероятностей принадлежности данной строки классам

Все вышеперечисленные тесты были успешно пройдены.

3.8 Вывод

В данном разделе для реализации сконструированного метода в качестве средств разработки был выбран Python v3.8 с использованием среды разработки IDE PyCharm. Определены минимальные требования к вычислительной системе.

Описана структура программного обеспечения и классы. Приведена инструкция для установки ПО и пример использования.

4 Исследовательский раздел

В данном разделе проводится параметризация выбранных алгоритмов классификации и отбора терминов. Также описывается ряд экспериментов, на основе которых будет сделан выбор наиболее оптимального метода для данной практической задачи.

4.1 Выборка данных

Для обучения и тестирования классификатора на вход подаётся 111805 документов, каждый из которых представляет собой строку на русском языке.

Данные, полученные из ПАО «ВТБ», хранятся в Excel-таблице. Также они поступают в первоначальном виде, то есть написаны в свободной форме естественным языком.

Для обучения классификатора была создана обучающая выборка, составляющая 80% от общего числа документов. Оставшиеся 20% необходимы для тестирования обученной модели и получения оценок качества классификатора.

Однако количество документов каждого класса неоднородно, поэтому точность определения на некоторых классах слишком низкая, чтобы рассматривать их. В приложении А на рисунках А.1 и А.2 представлены гистограммы наивного Байесовского классификатора и логистической регрессии, где по оси Y отображена F-мера, а по оси X – классы.

Таким образом, можно сделать вывод о том, что существуют классы, точность определения которых по F-мере ниже 60 %, следовательно, они не будут учитываться при дальнейшей работе.

4.2 Параметризация алгоритмов

Проведем параметризацию реализованного метода, для этого рассмотрим зависимость точности метода от изменения параметров метода отбора терминов на основании TF-IDF, наивного Байесовского классификатора и логистической регрессии.

Процесс параметризации метода отбора терминов TF-IDF заключается в варьировании максимальной величины параметра DF (документная частота, то есть максимальное допустимое число документов, в которых встретился термин t). Термины с документной частотой выше порогового значения будут исключены из рассмотрения при классификации.

В первом случае рассмотрим параметризацию наивного Байесовского классификатора. По оси X изменяется максимальная величина параметра $maxDF$, по оси Y – F-мера. На рисунке 14 изображены 4 ломаные, каждая из которых является зависимостью точности результата наивного Байесовского классификатора от параметра α (сглаживание). На анализируемой выборке данных метод слабо чувствителен к параметру α .

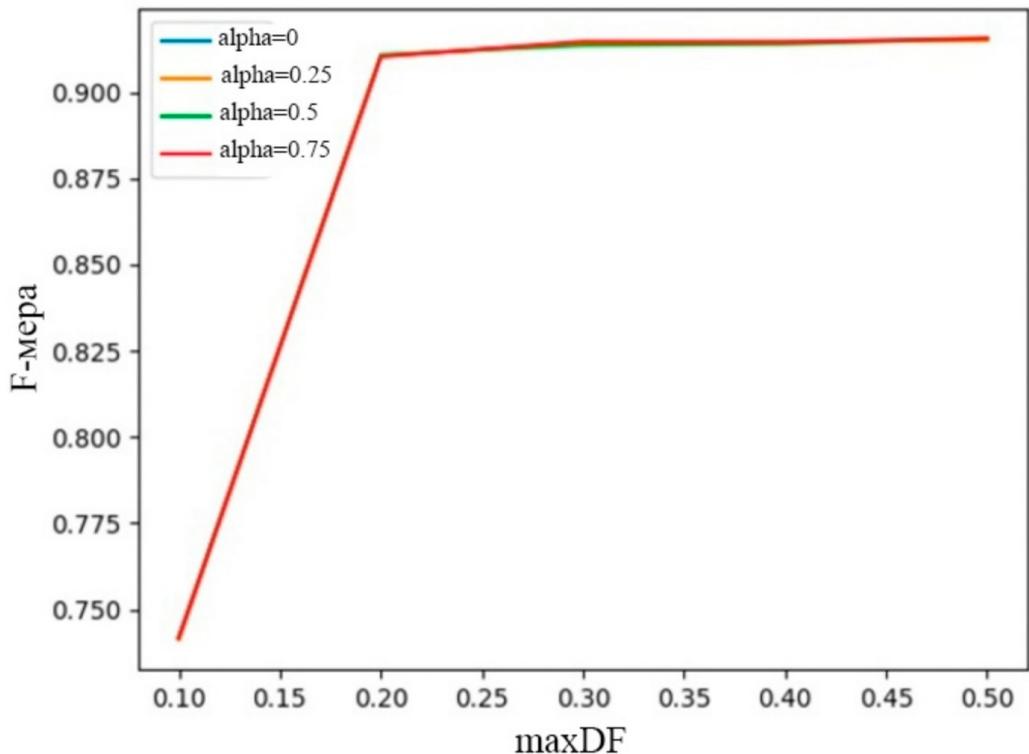


Рис.14 – Параметризация наивного Байесовского классификатора.

Во втором случае рассмотрим параметризацию логистической регрессии. На рисунке 15 изображены 4 зависимости точности данного метода классификации от параметра С (относительная сила регуляризации). На оси Y

отображена точность классификатора по F-мере, а по оси X меняется максимальная величина параметра $maxDF$. На анализируемой выборке данных метод слабо чувствителен к параметру C.

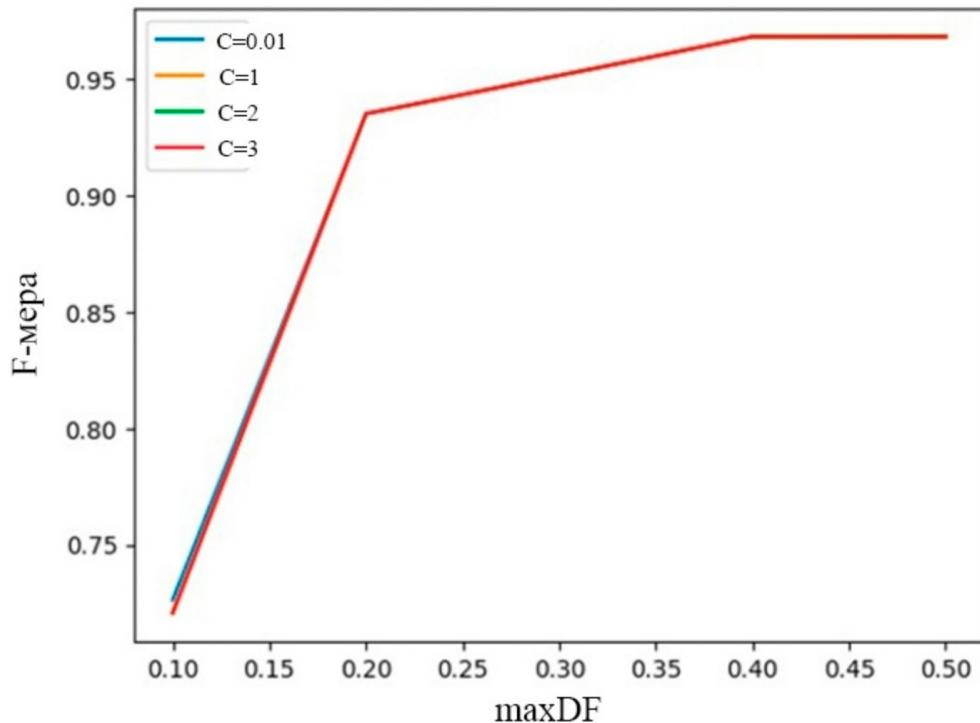


Рис.15 – Параметризация логистической регрессии

По результатам исследования видно, что точность по F-мере наивного Байесовского классификатора и логистической регрессии не зависит от изменения параметров α и C.

В таком случае сравним наивный Байесовский классификатор и логистическую регрессию. Зависимость классификаторов от величины $maxDF$ представлен на рисунке 16.

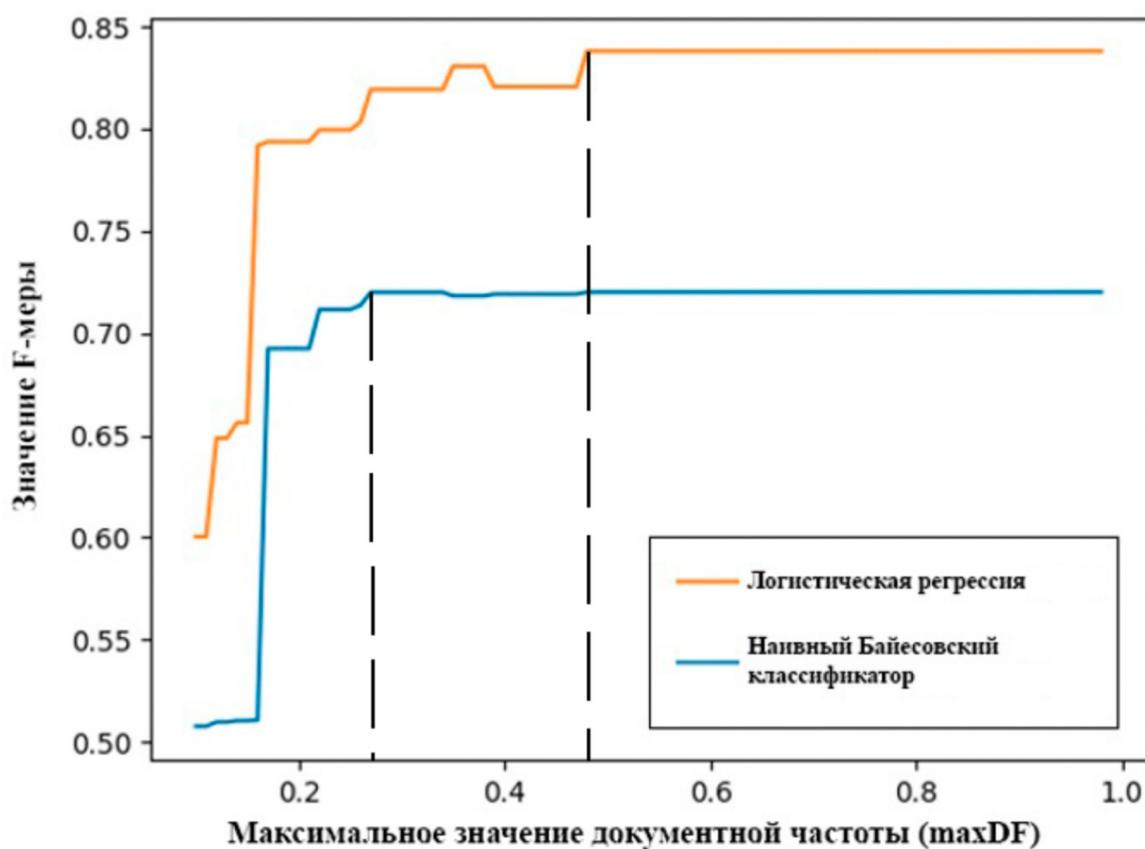


Рис.16 – Зависимость точности классификаторов от параметра $maxDF$

В результате наиболее оптимальным значением параметра $maxDF$ для последующего использования наивного Байесовского классификатора является $maxDF = 0.3$. В случае применения логистической регрессии $maxDF = 0.5$.

4.3 Сравнительный анализ методов классификации

Задача исследования – определить, какой из методов классификации показывает наилучшие результаты по выбранным мерам для решения задачи классификации платежей по полнотекстовому описанию на русском языке.

Для этого сравним классификаторы на материале данных рисунка 17.

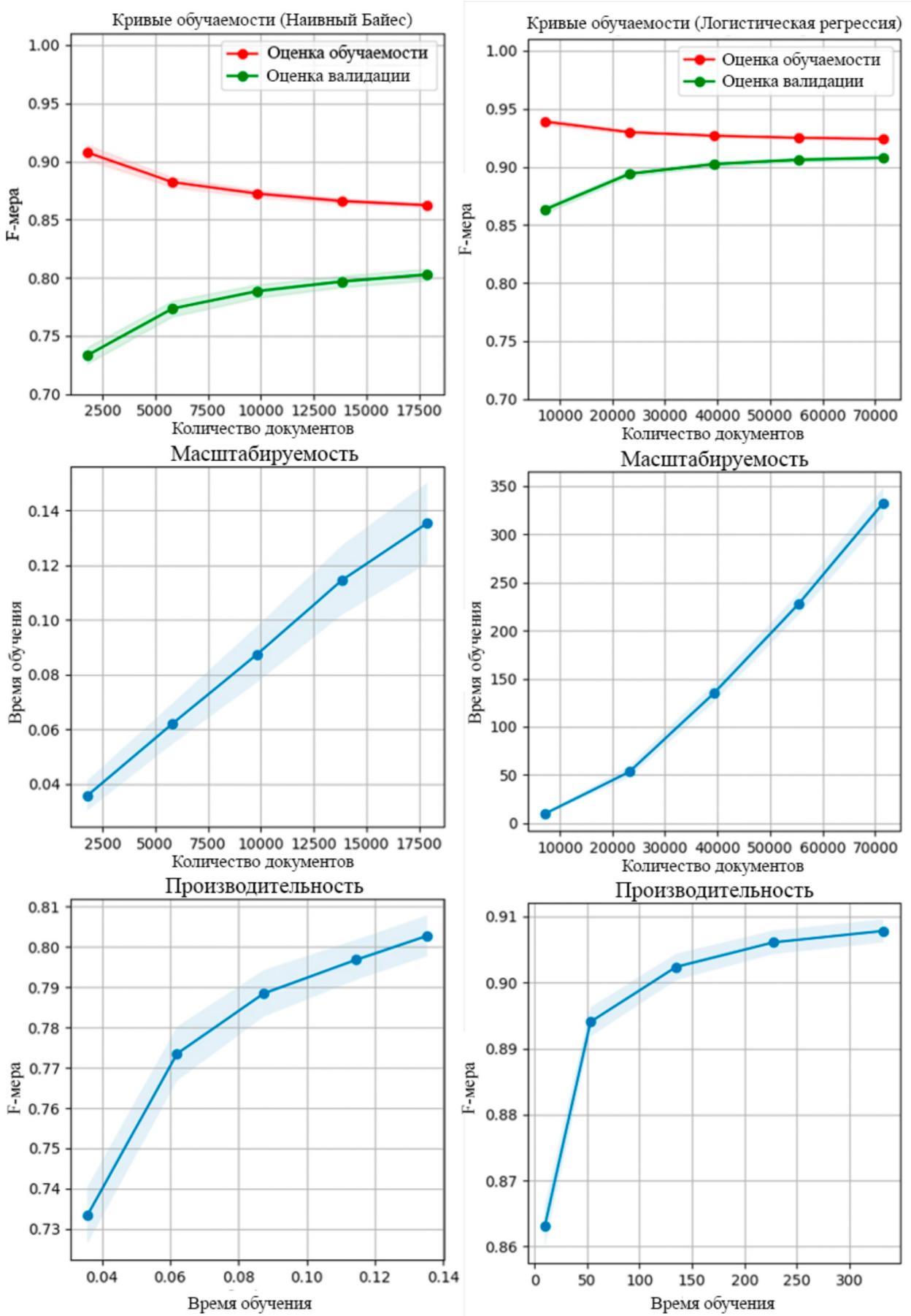


Рис.17 – Сравнительный анализ наивного Байесовского классификатора и логистической регрессии

Первый график отражает зависимость F-меры валидационной и тренировочной выборок от количества входных обучающих данных. В случае, если кривые сошлись в одной точке, добавление новых данных не увеличит точность классификатора. Несмотря на то, что возможно повысить точность наивного Байесовского классификатора в результате добавления новых данных, логистическая регрессия показывает наилучший результат по точности предсказания на предоставленной обучающей выборке.

С помощью второго графика можно оценить масштабируемость модели, то есть время обучения модели в зависимости от объема входных данных. Исходя из приведенных выше графиков можно сказать, что логистическая регрессия требует значительно больше времени для обучения на выборках разных размеров.

Третий график указывает на зависимость точности предсказаний от времени, потраченного на обучение модели. Обобщая вышеприведенные данные, сделан вывод, что в отличие от наивного Байесовского классификатора, логистическая регрессия показывает гораздо более высокие результаты по части точности, однако тратит на порядок больше времени на обучение модели.

Сравнение оценок классификаторов по метрикам аккуратности, точности, F-меры, а также макро- и взвешенного усреднения представлены в приложении Б. По результатам сравнительного анализа по всем метрикам наиболее точным классификатором является логистическая регрессия.

4.4 Вывод

На основе полученных экспериментальных данных сделано заключение о том, что логистическая регрессия показывает более высокую точность в сравнении с наивным Байесовским классификатором, следовательно, является наиболее подходящей среди методов нечёткой классификации для работы с платежными документами на русском языке. Для получения наилучших результатов для отбора терминов на основании TF-IDF следует выбрать пороговое максимальное значение параметра DF = 0.5.

Заключение

В рамках данной выпускной квалификационной работы был разработан и реализован метод нечёткой классификации платежей по полнотекстовому описанию на русском языке.

В результате проделанной работы были выполнены все поставленные задачи:

- рассмотрены существующие типы алгоритмов машинного обучения;
- проведен анализ существующих методов классификации;
- разработан метод нечёткой классификации платежей по полнотекстовому описанию на русском языке;
- сконструировано и разработано программное обеспечение, демонстрирующее работу метода;
- проведен сравнительный анализ двух подходящих нечётких методов классификации;
- разработанный метод исследован на предмет применимости.

В качестве направлений дальнейшей работы над методом можно выделить следующие:

- применение других стандартных методов классификации с модификацией, например, метода k-ближайших соседей с вычислением вероятности принадлежности документа к классу в зависимости от расстояний между объектом и ближайшими соседями;
- учёт не только отдельных терминов, но и словосочетаний.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. What is Machine Learning? [Электронный ресурс]. – Режим доступа: <https://www.ibm.com/cloud/learn/machine-learning> – Дата обращения: 01.05.2021
2. Supervised Learning [Электронный ресурс]. – Режим доступа: <https://www.sciencedirect.com/topics/computer-science/supervised-learning> – Дата обращения: 01.05.2021
3. Прикладная статистика: классификация и снижение размерности. / С.А. Айвазян, В.М. Бухштабер, И.С. Енуков, Л.Д. Мешалкин — М.: Финансы и статистика, 1989.
4. What is Text Classification? [Электронный ресурс]. – Режим доступа: <https://monkeylearn.com/what-is-text-classification/> – Дата обращения: 02.05.2021
5. Классификация текстов и анализ тональности [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=Классификация_текстов_и_анализ_тональности – Дата обращения: 02.05.2021
6. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика / Большикова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В, - М.: МИЭМ, 2011
7. Applied Predictive Modeling / Max Kuhn, Kjell Johnson - 2013
8. A Comparative Study of Feature Selection Methods for Dialectal Arabic Sentiment Classification Using Support Vector Machine / Omar Al-Harbil – Jazan University, 2019
9. Rocchio classification [Электронный ресурс]. – Режим доступа: <https://nlp.stanford.edu/IR-book/html/htmledition/rocchio-classification-1.html> – Дата обращения: 10.05.2021
10. KNN classification using Scikit-learn [Электронный ресурс]. – Режим доступа: <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn> – Дата обращения: 11.05.2021

11. Top 4 advantages and disadvantages of Support Vector Machine or SVM [Электронный ресурс]. – Режим доступа: <https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107> – Дата обращения: 12.05.2021
12. Оценка качества в задачах классификации и регрессии [Электронный ресурс]. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Оценка_качества_в_задачах_классификации_и_регрессии/ – Дата обращения: 15.05.2021
13. NumPy [Электронный ресурс]. – Режим доступа: <https://numpy.org> – Дата обращения: 17.05.2021
14. Scikit-learn [Электронный ресурс]. – Режим доступа: <https://www.sklearn.org> – Дата обращения: 17.05.2021
15. SciPy [Электронный ресурс]. – Режим доступа: <https://www.scipy.org> – Дата обращения: 17.05.2021
16. PyMorph2 [Электронный ресурс]. – Режим доступа: <https://pymorph2.readthedocs.io/en/stable/> – Дата обращения: 17.05.2021
17. PyQt [Электронный ресурс]. – Режим доступа: <https://doc.qt.io/qtforpython/> – Дата обращения: 17.05.2021

ПРИЛОЖЕНИЕ А

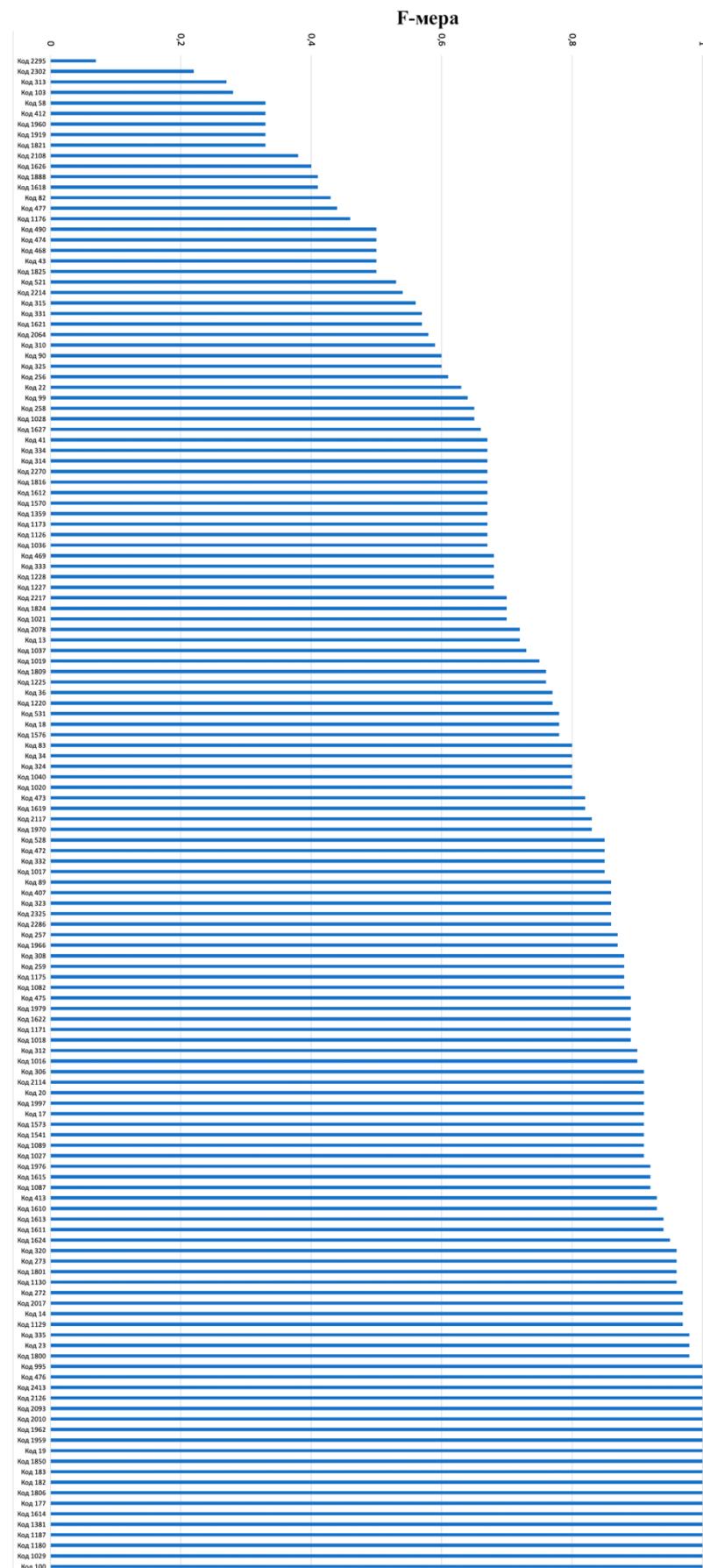


Рис. А.1 – Гистограмма точности распознавания классов логистической регрессией

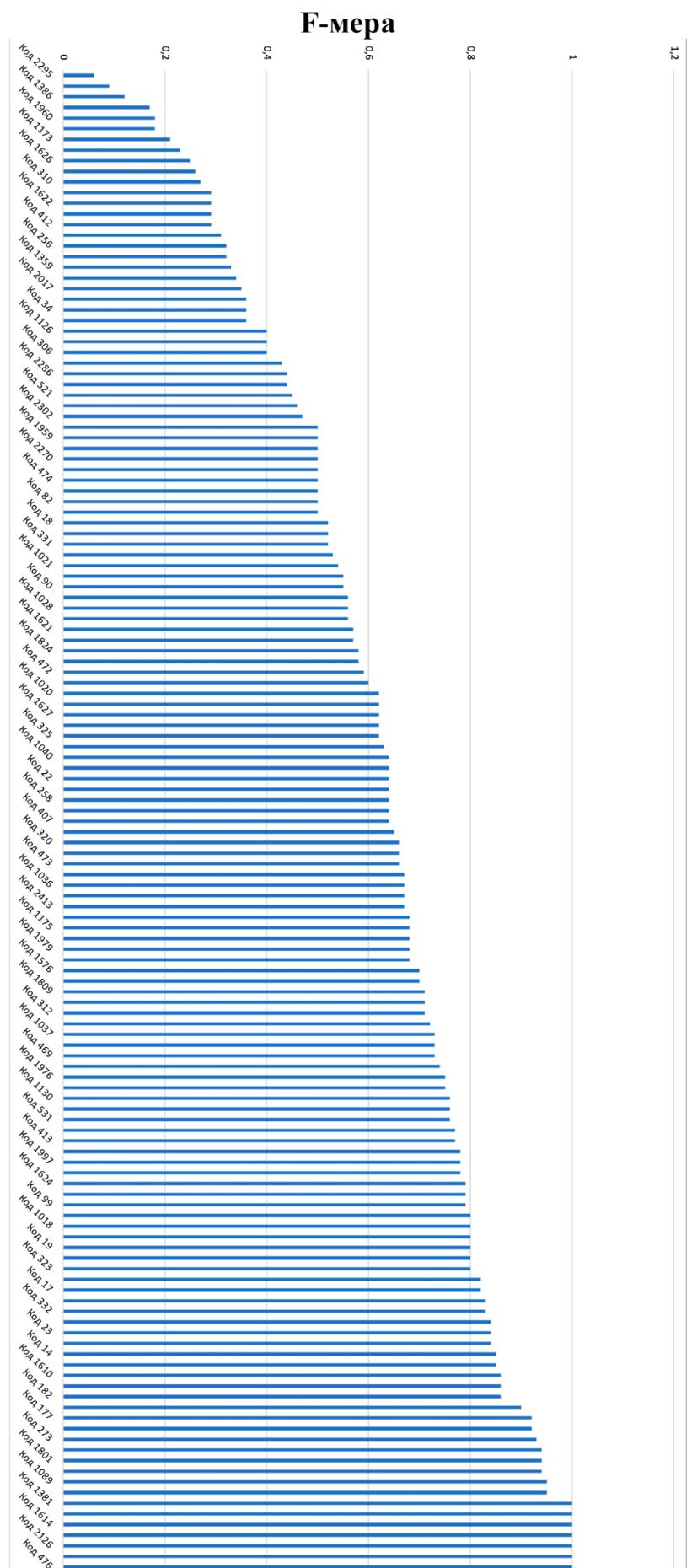


Рис. A.2 – Гистограмма точности распознавания классов наивного Байесовского классификатора

ПРИЛОЖЕНИЕ Б

	precision	recall	f1-score	support
100	0.77	0.77	0.77	30
1016	0.71	0.91	0.80	11
1017	0.82	0.68	0.74	135
1018	0.76	0.84	0.80	103
1019	0.44	0.79	0.56	263
1020	0.45	1.00	0.62	14
1021	0.62	0.47	0.54	202
1024	0.00	0.00	0.00	2
1027	0.74	0.63	0.68	621
1028	0.55	0.57	0.56	28
1029	0.50	1.00	0.67	1
103	0.44	0.22	0.29	167
1036	0.54	0.87	0.67	15
1037	0.60	0.92	0.73	13
1040	0.59	0.70	0.64	63
1082	0.74	0.97	0.84	77
1087	0.88	0.73	0.80	409
1089	0.91	1.00	0.95	21
1090	0.00	0.00	0.00	2
110	0.00	0.00	0.00	0
1101	0.00	0.00	0.00	1
1126	0.33	0.50	0.40	2
1129	0.83	0.99	0.90	100
1130	0.66	0.91	0.76	98
1171	0.60	0.69	0.64	13
1172	0.00	0.00	0.00	3
1173	0.13	0.57	0.21	7
1174	0.00	0.00	0.00	0
1175	0.58	0.81	0.68	32
1176	0.75	0.30	0.43	10
1177	0.00	0.00	0.00	1
1178	0.00	0.00	0.00	1
1180	0.00	0.00	0.00	3
1181	0.00	0.00	0.00	8
1187	1.00	0.50	0.67	4
1219	0.00	0.00	0.00	2
1220	0.37	0.77	0.50	30
1225	0.92	0.61	0.73	18
1227	0.86	0.50	0.63	60
1228	0.46	0.95	0.62	19
13	0.87	0.57	0.68	46
1359	0.20	1.00	0.33	1
1381	1.00	1.00	1.00	1
1385	0.00	0.00	0.00	4
1386	0.07	1.00	0.12	41
1387	0.00	0.00	0.00	5
1394	0.00	0.00	0.00	10
1396	0.00	0.00	0.00	9
1397	0.00	0.00	0.00	6

Рис. Б.1 – Оценка точности наивного Байесовского классификатора (часть 1)

14	0.98	0.76	0.85	166
1400	0.00	0.00	0.00	0
1402	0.00	0.00	0.00	2
1457	0.00	0.00	0.00	42
1541	0.81	0.84	0.82	106
1558	0.00	0.00	0.00	1
1568	0.00	0.00	0.00	0
1570	1.00	1.00	1.00	1
1573	0.47	0.70	0.56	10
1576	0.67	0.73	0.70	11
1610	1.00	0.75	0.86	8
1611	0.85	0.88	0.86	32
1612	0.25	1.00	0.40	1
1613	0.79	0.88	0.83	17
1614	1.00	1.00	1.00	17
1615	0.96	0.93	0.95	29
1616	0.00	0.00	0.00	1
1618	0.48	0.64	0.55	42
1619	0.64	1.00	0.78	32
1620	0.00	0.00	0.00	7
1621	0.67	0.50	0.57	4
1622	0.50	0.20	0.29	15
1624	0.77	0.81	0.79	21
1626	0.20	0.33	0.25	3
1627	0.70	0.56	0.62	68
1696	0.00	0.00	0.00	0
17	0.80	0.85	0.82	47
177	0.84	1.00	0.92	38
178	0.00	0.00	0.00	20
1798	0.00	0.00	0.00	2
1799	0.00	0.00	0.00	3
18	0.50	0.55	0.52	33
1800	0.98	0.91	0.94	5886
1801	0.98	0.91	0.94	8804
1804	0.00	0.00	0.00	0
1806	0.00	0.00	0.00	3
1808	0.00	0.00	0.00	1
1809	0.58	0.92	0.71	12
1810	0.00	0.00	0.00	2
1811	0.00	0.00	0.00	2
1812	0.00	0.00	0.00	4
1814	0.00	0.00	0.00	5
1816	0.20	0.67	0.31	3
1817	0.00	0.00	0.00	0
1819	0.00	0.00	0.00	0
182	0.75	1.00	0.86	3
1821	0.00	0.00	0.00	5
1824	0.52	0.65	0.58	20
1825	0.38	0.60	0.46	20
1826	0.22	0.06	0.09	142
1827	0.00	0.00	0.00	0

Рис. Б.2 – Оценка точности наивного Байесовского классификатора (часть 2)

183	0.40	1.00	0.57	4
1831	0.00	0.00	0.00	4
1849	0.00	0.00	0.00	1
1850	0.00	0.00	0.00	1
1873	0.00	0.00	0.00	2
1888	0.28	0.44	0.34	18
19	0.89	0.73	0.80	11
1919	0.29	0.50	0.36	4
1959	0.33	1.00	0.50	4
1960	0.11	0.50	0.18	2
1962	0.50	0.80	0.62	5
1966	0.79	0.66	0.72	29
1967	0.00	0.00	0.00	2
1970	0.36	0.84	0.50	25
1976	0.70	0.81	0.75	37
1979	0.72	0.65	0.68	20
1997	0.70	0.88	0.78	24
20	0.67	1.00	0.80	6
2010	1.00	1.00	1.00	3
2017	0.21	0.93	0.35	14
2028	0.00	0.00	0.00	1
2043	0.00	0.00	0.00	4
2056	0.00	0.00	0.00	0
2064	0.54	0.63	0.58	95
2072	0.00	0.00	0.00	30
2078	0.73	0.69	0.71	16
2093	0.89	1.00	0.94	8
2098	0.00	0.00	0.00	10
2108	0.88	0.15	0.26	153
2114	0.74	1.00	0.85	26
2117	0.77	0.73	0.75	37
2124	0.00	0.00	0.00	1
2126	1.00	1.00	1.00	1
2177	0.00	0.00	0.00	1
22	0.48	0.96	0.64	24
2214	0.88	0.37	0.52	19
2215	0.00	0.00	0.00	1
2217	0.48	0.93	0.64	15
2235	0.00	0.00	0.00	13
2270	0.50	0.50	0.50	2
2286	0.40	0.50	0.44	4
2295	0.08	0.05	0.06	22
23	0.81	0.88	0.84	33
2302	0.33	0.80	0.47	5
2317	0.40	0.11	0.17	19
2318	0.00	0.00	0.00	1
2323	0.00	0.00	0.00	0
2325	0.75	0.50	0.60	18
2372	0.00	0.00	0.00	1
2413	0.50	1.00	0.67	1
256	0.28	0.39	0.32	31

Рис. Б.3 – Оценка точности наивного Байесовского классификатора (часть 3)

257	0.72	0.79	0.76	43
258	0.54	0.78	0.64	27
259	0.76	0.83	0.79	30
272	0.90	0.94	0.92	505
273	0.93	0.92	0.93	211
306	0.25	1.00	0.40	10
308	0.88	0.38	0.53	161
310	0.60	0.17	0.27	35
312	0.85	0.61	0.71	18
313	0.26	0.42	0.32	24
314	0.88	0.52	0.65	27
315	1.00	0.29	0.44	14
320	0.51	0.92	0.66	25
323	0.88	0.73	0.80	30
324	0.53	0.87	0.66	30
325	0.48	0.86	0.62	14
33	0.00	0.00	0.00	0
331	0.46	0.59	0.52	136
332	0.83	0.84	0.83	320
333	0.54	0.81	0.64	309
334	0.17	1.00	0.29	1
335	0.52	0.97	0.68	224
34	0.25	0.67	0.36	6
36	0.70	0.88	0.78	8
407	0.52	0.83	0.64	18
41	1.00	1.00	1.00	1
412	0.17	1.00	0.29	1
413	0.65	0.95	0.77	21
43	1.00	0.33	0.50	3
468	0.47	0.15	0.23	52
469	0.67	0.79	0.73	105
471	0.00	0.00	0.00	0
472	0.45	0.87	0.59	117
473	0.58	0.75	0.66	56
474	0.40	0.67	0.50	3
475	0.33	1.00	0.50	4
476	1.00	1.00	1.00	1
477	0.50	0.29	0.36	7
489	0.00	0.00	0.00	4
490	0.00	0.00	0.00	3
499	0.00	0.00	0.00	0
521	0.31	0.83	0.45	52
528	0.77	0.93	0.84	106
531	0.77	0.75	0.76	264
58	0.33	0.12	0.18	16
615	0.00	0.00	0.00	8
630	0.00	0.00	0.00	2
82	0.33	1.00	0.50	7
83	0.40	0.67	0.50	6
89	0.00	0.00	0.00	3
90	0.71	0.44	0.55	34

Рис. Б.4 – Оценка точности наивного Байесовского классификатора (часть 4)

90	0.71	0.44	0.55	34
927	0.00	0.00	0.00	4
99	0.65	1.00	0.79	33
995	0.78	0.64	0.70	11
accuracy			0.83	22361
macro avg	0.42	0.50	0.43	22361
weighted avg	0.87	0.83	0.84	22361

Рис. Б.5 – Оценка точности наивного Байесовского классификатора (часть 5)

	precision	recall	f1-score	support
100	1.00	1.00	1.00	30
1016	1.00	0.82	0.90	11
1017	0.81	0.90	0.85	135
1018	0.89	0.89	0.89	103
1019	0.73	0.77	0.75	263
1020	0.75	0.86	0.80	14
1021	0.73	0.68	0.70	202
1024	0.00	0.00	0.00	2
1027	0.90	0.92	0.91	621
1028	0.76	0.57	0.65	28
1029	1.00	1.00	1.00	1
103	0.55	0.19	0.28	167
1036	0.67	0.67	0.67	15
1037	0.65	0.85	0.73	13
1040	0.88	0.73	0.80	63
1082	0.78	1.00	0.88	77
1087	0.89	0.94	0.92	409
1089	0.87	0.95	0.91	21
1090	0.00	0.00	0.00	2
1101	0.00	0.00	0.00	1
1126	1.00	0.50	0.67	2
1129	0.93	1.00	0.97	100
1130	0.96	0.96	0.96	98
1171	0.86	0.92	0.89	13
1172	0.00	0.00	0.00	3
1173	0.62	0.71	0.67	7
1175	0.96	0.81	0.88	32
1176	1.00	0.30	0.46	10
1177	0.00	0.00	0.00	1
1178	0.00	0.00	0.00	1
1180	1.00	1.00	1.00	3
1181	0.00	0.00	0.00	8
1187	1.00	1.00	1.00	4
1219	0.00	0.00	0.00	2
1220	0.71	0.83	0.77	30
1225	1.00	0.61	0.76	18
1227	1.00	0.52	0.68	60
1228	0.55	0.89	0.68	19

Рис. Б.6 – Оценка точности логистической регрессии (часть 1)

13	1.00	0.57	0.72	46
1359	0.50	1.00	0.67	1
1381	1.00	1.00	1.00	1
1385	0.00	0.00	0.00	4
1386	0.00	0.00	0.00	41
1387	0.00	0.00	0.00	5
1394	0.00	0.00	0.00	10
1396	0.00	0.00	0.00	9
1397	0.00	0.00	0.00	6
14	0.94	1.00	0.97	166
1402	0.00	0.00	0.00	2
1457	0.00	0.00	0.00	42
1541	0.93	0.90	0.91	106
1558	0.00	0.00	0.00	1
1570	0.50	1.00	0.67	1
1573	0.83	1.00	0.91	10
1576	1.00	0.64	0.78	11
1610	1.00	0.88	0.93	8
1611	0.94	0.94	0.94	32
1612	0.50	1.00	0.67	1
1613	0.94	0.94	0.94	17
1614	1.00	1.00	1.00	17
1615	0.90	0.93	0.92	29
1616	0.00	0.00	0.00	1
1618	0.59	0.31	0.41	42
1619	0.74	0.91	0.82	32
1620	0.00	0.00	0.00	7
1621	0.67	0.50	0.57	4
1622	1.00	0.80	0.89	15
1624	1.00	0.90	0.95	21
1626	0.50	0.33	0.40	3
1627	0.62	0.71	0.66	68
17	0.84	1.00	0.91	47
177	1.00	1.00	1.00	38
178	0.00	0.00	0.00	20
1798	0.00	0.00	0.00	2
1799	0.00	0.00	0.00	3
18	0.88	0.70	0.78	33
1800	0.98	0.97	0.98	5886
1801	0.94	0.99	0.96	8804
1806	1.00	1.00	1.00	3
1808	0.00	0.00	0.00	1
1809	0.65	0.92	0.76	12
1810	0.00	0.00	0.00	2
1811	0.00	0.00	0.00	2
1812	0.00	0.00	0.00	4
1814	0.00	0.00	0.00	5
1816	0.67	0.67	0.67	3
1819	0.00	0.00	0.00	0
182	1.00	1.00	1.00	3

Рис. Б.7 – Оценка точности логистической регрессии (часть 2)

1821	1.00	0.20	0.33	5
1824	0.70	0.70	0.70	20
1825	0.41	0.65	0.50	20
1826	0.00	0.00	0.00	142
183	1.00	1.00	1.00	4
1831	0.00	0.00	0.00	4
1849	0.00	0.00	0.00	1
1850	1.00	1.00	1.00	1
1873	0.00	0.00	0.00	2
1888	0.55	0.33	0.41	18
19	1.00	1.00	1.00	11
1919	0.50	0.25	0.33	4
1959	1.00	1.00	1.00	4
1960	0.25	0.50	0.33	2
1962	1.00	1.00	1.00	5
1966	0.92	0.83	0.87	29
1967	0.00	0.00	0.00	2
1970	0.87	0.80	0.83	25
1976	0.94	0.89	0.92	37
1979	0.94	0.85	0.89	20
1997	0.95	0.88	0.91	24
20	1.00	0.83	0.91	6
2010	1.00	1.00	1.00	3
2017	0.93	1.00	0.97	14
2028	0.00	0.00	0.00	1
2043	0.00	0.00	0.00	4
2057	0.00	0.00	0.00	0
2064	0.54	0.63	0.58	95
2072	0.00	0.00	0.00	30
2078	1.00	0.56	0.72	16
2093	1.00	1.00	1.00	8
2098	0.00	0.00	0.00	10
2108	0.70	0.26	0.38	153
2114	0.86	0.96	0.91	26
2117	0.96	0.73	0.83	37
2124	0.00	0.00	0.00	1
2126	1.00	1.00	1.00	1
2177	0.00	0.00	0.00	1
22	0.51	0.83	0.63	24
2214	1.00	0.37	0.54	19
2215	0.00	0.00	0.00	1
2217	0.56	0.93	0.70	15
2235	0.00	0.00	0.00	13
2270	1.00	0.50	0.67	2
2286	1.00	0.75	0.86	4
2295	0.17	0.05	0.07	22
23	1.00	0.97	0.98	33
2302	0.25	0.20	0.22	5
2317	0.00	0.00	0.00	19
2318	0.00	0.00	0.00	1

Рис. Б.8 – Оценка точности логистической регрессии (часть 3)

2325	0.84	0.89	0.86	18
2372	0.00	0.00	0.00	1
2413	1.00	1.00	1.00	1
256	0.61	0.61	0.61	31
257	0.82	0.93	0.87	43
258	0.68	0.63	0.65	27
259	0.93	0.83	0.88	30
272	0.97	0.97	0.97	505
273	0.95	0.96	0.96	211
306	0.83	1.00	0.91	10
308	0.88	0.88	0.88	161
310	0.66	0.54	0.59	35
312	0.82	1.00	0.90	18
313	0.67	0.17	0.27	24
314	0.93	0.52	0.67	27
315	0.64	0.50	0.56	14
320	0.96	0.96	0.96	25
323	0.89	0.83	0.86	30
324	0.74	0.87	0.80	30
325	0.45	0.93	0.60	14
331	0.90	0.41	0.57	136
332	0.86	0.84	0.85	320
333	0.53	0.94	0.68	309
334	0.50	1.00	0.67	1
335	0.97	0.99	0.98	224
34	1.00	0.67	0.80	6
36	1.00	0.62	0.77	8
407	0.84	0.89	0.86	18
41	0.50	1.00	0.67	1
412	0.20	1.00	0.33	1
413	0.91	0.95	0.93	21
43	1.00	0.33	0.50	3
468	0.44	0.58	0.50	52
469	0.70	0.66	0.68	105
472	0.88	0.82	0.85	117
473	0.75	0.91	0.82	56
474	1.00	0.33	0.50	3
475	0.80	1.00	0.89	4
476	1.00	1.00	1.00	1
477	1.00	0.29	0.44	7
489	0.00	0.00	0.00	4
490	0.33	1.00	0.50	3
499	0.00	0.00	0.00	0
521	0.39	0.81	0.53	52
528	0.77	0.94	0.85	106
531	0.81	0.75	0.78	264
58	0.50	0.25	0.33	16
615	0.00	0.00	0.00	8
630	0.00	0.00	0.00	2
82	0.43	0.43	0.43	7

Рис. Б.9 – Оценка точности логистической регрессии (часть 4)

83	1.00	0.67	0.80	6
89	0.75	1.00	0.86	3
90	0.84	0.47	0.60	34
927	0.00	0.00	0.00	4
99	0.94	0.48	0.64	33
995	1.00	1.00	1.00	11
accuracy			0.91	22361
macro avg	0.60	0.58	0.57	22361
weighted avg	0.90	0.91	0.90	22361

Рис. Б.10 – Оценка точности логистической регрессии (часть 5)