Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Иншаков Константин Андреевич

Интерпретация нейросетевых моделей в задаче обнаружения автоматически сгенерированных новостных текстов

Выпускная квалификационная работа

Научный руководитель:
к.ф.-м.н., доцент
В. Г. Абрамов

Москва, 2025

Аннотация

В данной работе исследуются подходы к созданию моделей машинного обучения, поддающихся интерпретации на примере задачи выявления автоматически сгенерированных новостных текстов.

В работе предложена комбинированная модель, основанная на построении деревьев решений с использованием предсказаний более сложной базовой модели в качестве обучающих данных. На основе этой модели реализована программная система, проведены эксперименты с различными вариантами реализации модели.

Результаты исследования показывают, что при использовании определенных вариантов базовой модели, комбинированная модель позволяет повысить точность классификации по сравнению с применением дерева решений к необработанным данным.

Ключевые слова: деревья решений, алгоритм CART, нейронные сети, градиентный бустинг, генерация текста.

Содержание

1 Введение

Стремительное развитие больших языковых моделей, таких как GPT [1], в последние несколько лет привело к резкому росту числа правдоподобных текстов, сгенерированных нейронными сетями. Распространение заведомо ложной или неполной информации с целью влияния на пользователей также становится всё более серьёзной проблемой, а использование инструментов генерации текста позволяет значительно сократить время и ресурсы, необходимые для её подготовки.

Данная ситуация делает задачу обнаружения текста, сгенерированного нейросетями, крайне важной и актуальной, её решение практически необходимо для корректной модерации любого контента, опубликованного в интернете. В этой работе фокус сделан именно на новостных статьях, поскольку они представляют наибольшую опасность - при автоматической генерации и публикации зачастую не происходит проверки описанных в тексте фактов на достоверность, а современные языковые модели нередко предоставляют текст, никак не связанный с действительностью, что может привести к распространению недостоверной информации, если факт генерации текста не будет вовремя выявлен.

Важной проблемой в области выявления сгенерированного текста является отсутствие у значительной части пользователей каких-либо знаний о его характерных признаках, в отличие от других форм контента, например, изображений. Эта проблема обостряется тем, что для большинства задач в области обработки естественного языка, включая выявление фальшивого текста, в настоящее время активно применяются трансформеры и другие сложные модели машинного обучения, которые позволяют добиться высокой точности классификации, но не дают новой информации о зависимостях в данных в понятном для человека виде.

В данной работе предлагается скомбинировать несколько методов решения задачи классификации текста для получения модели, показывающей достаточно хорошую точность, но поддающуюся интерпретации лучше, чем сложные модели "чёрные ящики". На её основе могут быть получены более чёткие выводы о характерных свойствах сгенерированных текстов, которые могут быть использованы в дальнейшем, в том числе для выявления генерации без помощи модели. В качестве основы для такой комбинированный модели будут использованы нейросетевые классификаторы из-за их высокой точности и модели на основе правил (деревья решений), поскольку они лучше всего подходят для получения выводов в человекочитаемом виде, не слишком сильно жертвуя качеством классификации.

2 Постановка задачи

Цель работы - разработать комбинированную модель мягкой классификации, обнаруживающую сгенерированные новостные статьи и позволяющую получить выводы о работе нейронной сети. Модель должна удовлетворять обозначенным далее критериям интерпретируемости.

Под интерпретируемостью здесь и далее понимается следующее: интерпретируемая модель - модель, внутреннюю структуру и процесс работы которой способен объяснить эксперт, в частности объяснимы должны быть:

- смысл всех используемых признаков;
- функциональная взаимосвязь значений признаков и результата;
- влияние гиперпараметров на модель;

Нейросетевые модели зачастую не удовлетворяют ни одному из этих критериев, но показывают высокую точность классификации, в том время как модели, поддающиеся интерпретации, не могут достичь высокого качества классификации. В связи с этим в данной работе предлагается использовать объединение двух различных моделей.

Реализация комбинации моделей происходит следующим образом: на первом этапе происходит обучение нейронной сети и её валидация на отдельном наборе данных. После получения достаточного качества классификации нейронная сеть используется для предсказания меток классов документов в новом наборе данных. На втором этапе интерпретируемая модель (решающее дерево) обучается на этом наборе данных, используя в качестве эталонных меток не реальные метки классов, а метки, предсказанные нейронной сетью. После обучения новая модель тестируется на финальном проверочном наборе. Все четыре набора должны быть различны для избежания переобучения.

Обоснование данного процесса может быть проведено двумя способами: с одной стороны, происходит приближение работы нейронной сети другой более простой моделью, что позволяет интерпретировать выявленные сетью закономерности. С другой стороны, решающее дерево обучается на прогнозах нейронной сети, то есть на более простом наборе данных с более гладкой разделяющей поверхностью между классами и практически полным отсутствием выбросов. Такой подход часто помогает избежать переобучения, особенно при использовании алгоритма CART[1][3] для построения дерева, что будет оговорено далее.

В работе следует:

1. исследовать проблему интерпретации моделей машинного обучения;
2. разработать гибридную модель машинного обучения на основе нейронной сети и дерева решений, решающую задачу выявления автоматически сгенерированных новостных текстов;
3. реализовать программную систему на основе этой модели, получить выводы о её работоспособности.

[1] Classification And Regression Tree

3 Обзор существующих походов

3.1 Пространство признаков

Для того, чтобы модели машинного обучения могли обрабатывать текстовые данные, их необходимо привести к некоторому числовому виду фиксированной размерности. Поскольку все составляющие используемой модели должны обрабатывать одно и то же пространство признаков, на него накладываются ограничения: все признаки должны поддаваться интерпретации, не имея при этом негативного влияния на работу используемых алгоритмов. Рассмотрим несколько возможных подходов для конвертации текста в вектор числовых признаков.

3.1.1 Классические методы

Простейшим способом представления текста в виде набора чисел является переход от самого текста к его косвенным свойствам, таким как средняя длина слов и предложений, доля различных частей речи в тексте и максимальная частота употребляемых лемм. Такой подход позволяет получить небольшое число простых для понимания признаков с фиксированным диапазоном значений, что упрощает получение выводов из готовой модели и хорошо подходит для интерпретируемых моделей, используемых в данной работе. С другой стороны, данный подход уступает по точности более современным вариантам и может привести к медленному обучению нейросетей.

В данной работе в качестве одного из вариантов пространства признаков будет рассмотрено пространство, содержащее:

- среднюю длину слов;
- среднюю длину предложений;
- максимальную частоту слова;
- частоты частей речи: имён собственных, имён существительных, глаголов, прилагательных и наречий;
- число вхождений слов заголовка в основной текст.

Модель "мешок слов" [7] - один из первых подходов к векторизации слов текста. В рамках данной модели кодирование происходит в два этапа - на первом этапе все слова (или леммы), встреченные в некотором обучающем наборе, записываются в словарь, каждому из них присваивается индекс. На втором этапе каждое слово в словаре кодируется вектором размерности n, где n - размер имеющегося словаря. Все компоненты этого вектора равны 0, кроме компоненты, индекс которой равен индексу данного слова в словаре, эта компонента равна 1. Для кодирования полного текста все слова, находящиеся в нём и попавшие в словарь, кодируются векторами по описанному выше правилу, после чего эти векторы суммируются. Таким образом, весь текст представляется вектором фиксированной размерности n, в котором каждая компонента равна числу вхождений соответствующего слова в текст.

Главным плюсом данного метода является его простота, значения признаков для любого текста являются предельно понятными и могут быть получены вручную. Однако примитивность этого метода также приводит к значительному числу недостатков, главный из них - в полученных векторах учитывается только число вхождений слов, но не их порядок, поэтому при работе с такими векторами нет возможности учитывать контекст. Ещё одним важным минусом данного подхода является большая размерность полученных векторов, при этом большая часть их компонент всегда будет равна 0. Эти ограничения ведут к потере точности классификации по сравнению с более сложными методами, однако "мешок слов" показывает достаточно высокие результаты в задаче выявления сгенерированного текста и позволяет легко проводить интерпретацию, что делает его наиболее предпочтительным вариантом.

Модель TF-IDF[1] [7] также позволяет кодировать слова и тексты векторами фиксированной размерности и является усложнённой версией модели "мешок слов". На первом этапе применения TF-IDF генерируется словарь, аналогичный словарю, получаемому для предыдущего метода. На этапе векторизации конкретного текста первым шагом является получение аналогичного предыдущей модели вектора вхождений, после чего число вхождений каждого слова t умножается на весовой коэффициент w_t, полученный по формуле $w_t = \log \frac{1+n}{1+df(t)} + 1$, где n - общее число документов в обучающем наборе, $df(t)$ - число документов, в которых содержится слово t. Взвешивание слов происходит для увеличения влияния более редких слов и уменьшения влияния слов, встречающихся в большинстве текстов, так как во многих задачах такие слова несут меньше информации.

TF-IDF обладает многими недостатками предыдущей модели - полученные векторы также не зависят от порядка слов, а их размерность по-прежнему равна числу слов в словаре. Несмотря на это, в ряде случаев использование TF-IDF позволяет повысить точность классификации по сравнению с предыдущей моделью, однако в данной задаче использование мешка слов является более предпочтительным, поскольку он лучше поддаётся интерпретации и показывает практически аналогичное качество классификации.

3.1.2 Статистические эмбеддинги

Одним из вариантов устранения недостатков предыдущих моделей являются статистические эмбеддинги. Цель таких подходов - смоделировать семантическую близость слов - чем ближе два слова по смыслу, тем меньше должно быть расстояние между их векторами. Важно отметить, что в качестве расстояния в данном случае используется косинусная близость:

$$d_{cos}(\vec{x}, \vec{y}) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

Где $\vec{x}$ и $\vec{y}$ - произвольные векторы (в данном случае представляющие слова).

Одним из наиболее распространённых методов в данной категории является модель word2vec [8]. Данная модель основана на предположении, что слова, часто встре-

[1] Term Frequency–Inverse Document Frequency

чающиеся на небольшом расстоянии друг от друга в обучающих текстах, имеют некоторую смысловую связь. Ниже приведен принцип её работы.

В основе модели лежит нейронная сеть, предсказывающая скрытое слово обучающего текста по окружающим его словам - это модель CBOW[1], или окружающие слова по центральному слову - это модель skip-gram.

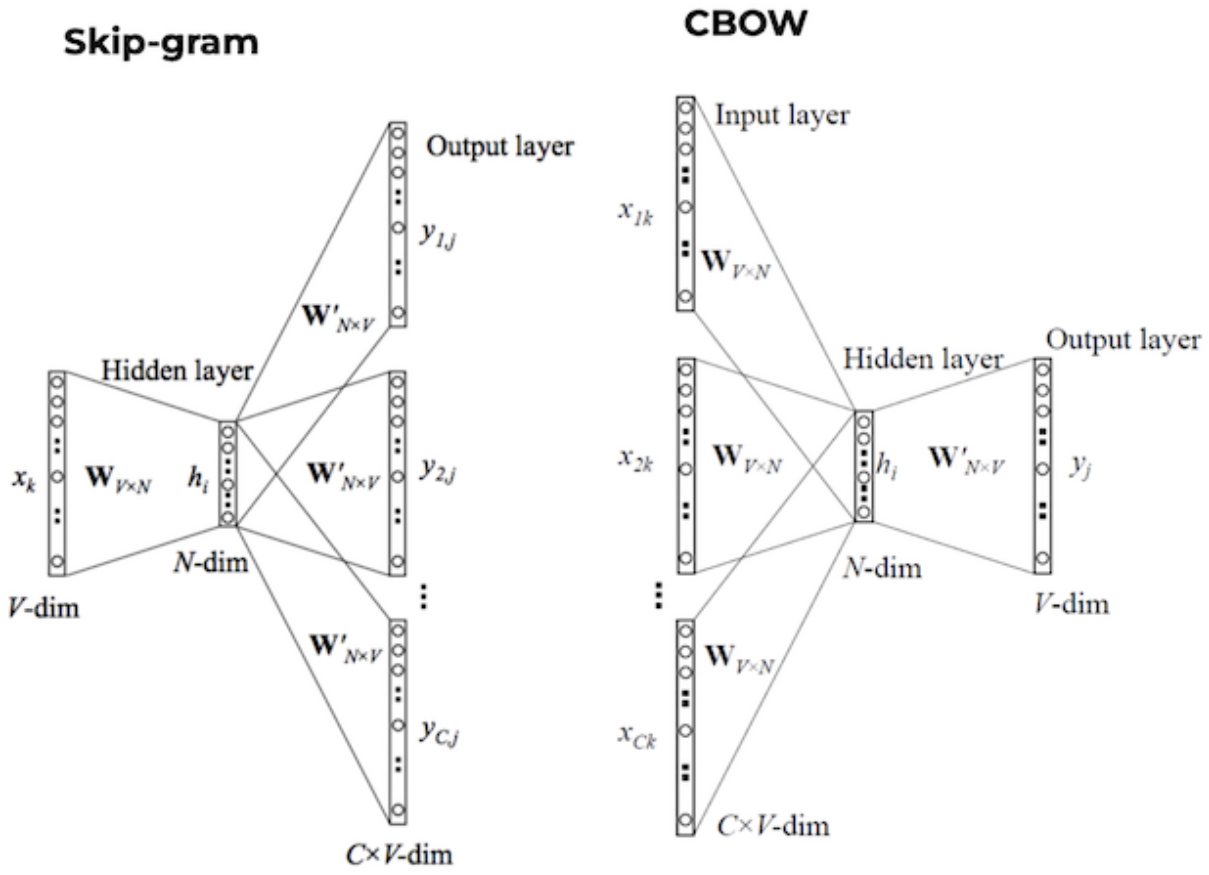


Рис. 1: Схема моделей CBOW и skip-gram

На вход сети подаются векторы, полученные с помощью модели "мешок слов", а прогноз определяется как слово, индекс которого в словаре совпадает с номером максимальной компоненты выходного вектора после применения к нему функции softmax. Сама сеть состоит из двух полносвязных слоёв без функций активации, то есть параметризуется двумя матрицами весов размера VxN и NxV соответственно, где V - размер словаря и входных векторов, N - произвольный размер скрытого состояния.

[1]Continuous Bag Of Words

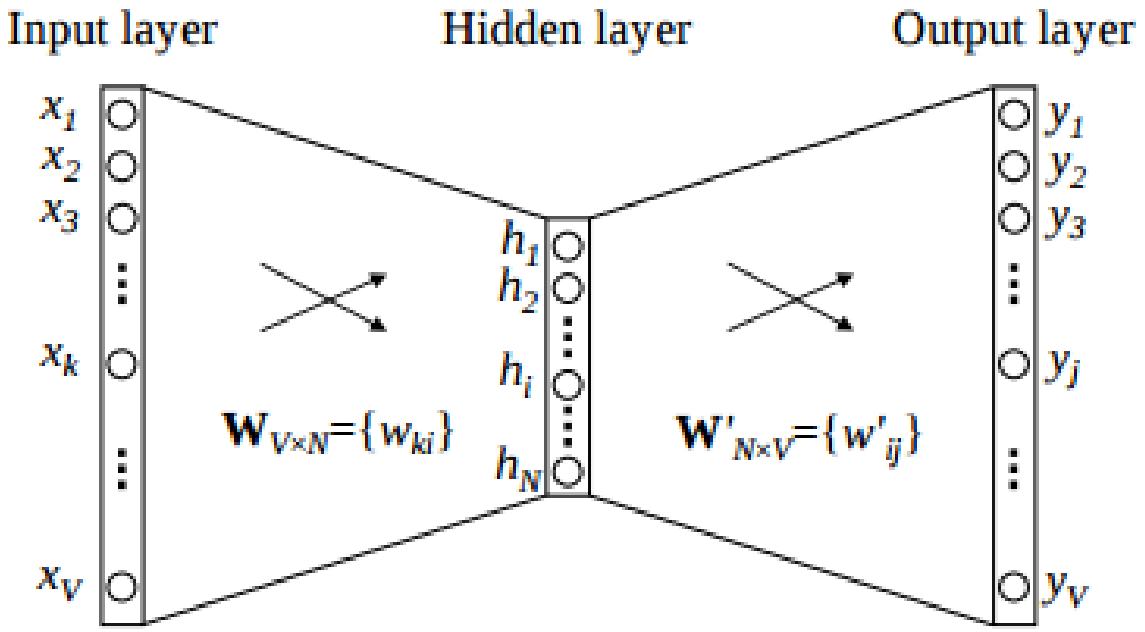


Рис. 2: Устройство нейросети в модели word2vec

Ниже приведены соответствующие формулы:

$$softmax(x_i, \vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{V} e^{x_j}}$$

$$\vec{h} = \vec{x}^T W$$

$$\vec{z} = \vec{h} W'$$

$$y_i = softmax(z_i, \vec{z})$$

Сами предсказания нейросети при этом не имеют большого значения, поскольку в качестве нового вектора i-го слова словаря используется i-я строка матрицы W или i-ый столбец матрицы W', содержащие веса нейронной сети после обучения модели. Обосновать использование этих векторов для представления слов можно следующим образом: элементы выходного вектора y_j после применения softmax можно расценивать как вероятности того, что j-ое слово словаря находится в том же контексте, что и входное слово, при этом нетрудно заметить, что y_j в точности равен скалярному произведению строки, соответствующей входному слову, на столбец, соответствующий j-му слову. Тогда во время обучения сети скалярное произведение между векторами часто встречающихся рядом слов будет максимизироваться, а косинусное расстояние между ними - минимизироваться, что и является целью построения данного векторного представления.

Word2vec и другие модели на её основе позволяют исправить ряд недостатков классических методов - полученные с её помощью векторы могут иметь произвольную размерность и являются плотными. Это позволяет получить качество, превышающее классические модели, используя векторы на несколько порядков меньшей размерности. Полученное пространство также обладает рядом других полезных свойств, связанных с малым расстоянием между близкими по смыслу словами. Благодаря этим факторам статистические модели крайне эффективны в ряде задач, однако они имеют один ключевой недостаток - значения компонент полученных векторов невозможно интерпретировать, поскольку они являются весами нейронной сети. Это делает применение подобных методов невозможных в задачах, где важна интерпретируемость ответа, включая задачу, поставленную в этой работе, поскольку получение выводов о работе модели требует понятности используемых ею признаков.

3.2 Методы классификации текста

На сегодняшний день существует колоссальное число моделей машинного обучения, применяемых для классификации. Рассмотрим те из них, которые наиболее актуальны для текстовых данных и поставленной задачи.

3.2.1 Алгоритм CART

Решающее дерево/дерево решений - связный ориентированный граф без циклов, каждая вершина которого, не являющаяся листом, помечена некоторым условием, дуги, выходящие из таких вершин, соответствуют значениям этого условия (истина/ложь), то есть дерево зачастую является двоичным, однако возможны и реализации с большим числом вариантов. Всем листовым вершинам соответствуют метки прогнозируемых классов или значения целевой функции. Процесс прогнозирования для нового объекта заключается в проходе от корня дерева до листа с проверкой находящихся в проходимых вершинах условий и переходами по соответствующим дугам. Значение, находящееся в достигнутом листе, является результатом.

CART - Алгоритм построения деревьев решений на основе обучающей выборки объектов. В основе алгоритма лежит разбиение пространства признаков на n-мерные параллелепипеды с помощью гиперплоскостей, параллельных осям координат. На первом шаге происходит подбор оптимальной разбивающей плоскости путём перебора для каждого признака - для категориальных признаков выбираются все возможные разбиения множества категорий, для числовых происходит поиск точки разделения по всему множеству значений признака с заранее определённым шагом. Оптимальным считается разбиение, на котором достигается минимум некоторой функции потерь, принимающей на вход полученные в результате множества. Ниже приведены формулы двух самых распространённых функций потерь.

Эмпирическая вероятность объекта из класса k попасть в подпространство R_m:

$$p_{mk} = \frac{1}{|R_m|} \sum_{i:x_i \in R_m} I(y_i = k)$$

Критерий Джини:

$$Q_{gini} = \sum_{m=1}^{M} \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

Энтропия:

$$H = -\sum_{m=1}^{M} \sum_{k=1}^{K} p_{mk} \log_2(p_{mk})$$

Где x_i - объекты выборки, y_i - их метки, R_m - подпространства, полученные после потенциального разбиения, k - номер класса, K - общее число классов, M - число подпространств, полученных при разбиении, I - индикаторная функция.

Эти функции достигают своего минимума в случае, если в полученных подпространствах встречаются объекты только одного класса, то есть при их минимизации строятся наиболее "чистые" подпространства.

После первого шага процесс повторяется рекурсивно для каждого из полученных подпространств, при этом на каждом из этапов учитывается только текущее подпространство, это необходимо для ускорения вычислений. Процесс разбиения прекращается при выполнении условия остановки - обычно это достижение максимально допустимой глубины и/или минимального числа наблюдений в одном подпространстве.

После получения последовательности разбиений, на её основе строится бинарное дерево решений - в корень помещается условие вида $x_i < a$, где x_i - выбранный на первом шаге признак, a - точка первого разбиения. Каждое из двух поддеревьев соответствует подпространству, полученному в результате этого разбиения, для них процесс построения повторяется рекурсивно.

Плюсы данного метода:

- простота интерпретации;
- отсутствие необходимости дополнительной обработки признаков;

Минусы:

- склонность к переобучению;
- затруднённая работа со сложными зависимостями;
- нестабильность - полученные деревья резко отличаются даже при небольших изменениях в обучающей выборке;

Деревья решений являются разновидностью моделей машинного обучения, наиболее подходящей для интерпретации. Процесс работы построенного дерева максимально понятен - он состоит в проверке набора простых условий. Благодаря этому свойству данная модель была выбрана для объяснения предсказаний базовой модели в этой работе.

3.2.2 Ансамблевые модели

Одним из способов повышения точности классификации деревьев решений является объединение нескольких деревьев в общую модель, показывающую большую точность, чем каждое из них в отдельности. Такие модели обычно хуже поддаются интерпретации, но лучше обрабатывают сложные наборы данных.

Самым простым вариантом такого объединения является случайный лес [4]. Принцип данной модели заключается в использовании нестабильности алгоритма CART - вместо одного дерева обученного на всей выборке модель предполагает обучение нескольких деревьев, каждое из которых имеет доступ только к случайно выбранному подмножеству выборки и случайно выбранному подпространству признаков. Такой метод позволяет получить набор сильно отличных друг от друга деревьев, что в теории должно повысить точность классификации при принятии решения о классе нового объекта с помощью голосования, поскольку их прогнозы можно считать независимыми. На практике используются модели, состоящие из тысяч независимых деревьев, что делает интерпретацию практически невозможной.

Плюсы данного метода:

- увеличенная точность;
- устойчивость к переобучению;

Минусы:

- долгое обучение;
- невозможность интерпретации;

Более совершенным вариантом ансамблевой модели является градиентный бустинг [5]. Данная модель также состоит из набора простых решающих деревьев, однако их ошибки компенсируются не с помощью голосования, а с помощью прогнозирования. Каждая модель предсказывает ошибку предыдущей, что позволяет скорректировать результат при совместном использовании их прогнозов. В общем случае формула предсказания данной модели выглядит следующим образом:

$$a(x) = D(\sum_{i=1}^{n} c_i b_i(x))$$

Где $a(x)$ - отклик модели, $b_i(x)$ - прогнозы составляющих моделей, c_i - веса, присвоенные им на основе уверенности прогноза, D - некоторое решающее правило.

В ряде задач градиентный бустинг показывает результаты, сравнимые по качеству с нейросетевыми моделями, однако он обладает теми же недостатками, что и случайный лес - полная невозможность интерпретации и долгое обучение с большим числом гиперпараметров.

Сложность ансамблей не позволяет использовать их в роли приближающей модели, однако они могут выступить в качестве базовой высокоточной модели для сравнения их влияния на точность решающего дерева с нейронной сетью.

3.2.3 Полносвязные нейронные сети

Нейронные сети были впервые предложены в качестве математической модели нейронов человеческого мозга. Полносвязные сети - простейшая вариант её реализации, представляющий собой последовательность слоёв нейронов фиксированного размера, каждый нейрон на очередном слое принимает на вход вектор чисел, полученных от всех нейронов предыдущего слоя. Первый слой получает на вход некоторый вектор фиксированного размера, выходы последнего слоя объединяются в вектор, являющийся итоговым результатом.

Работу очередного слоя полносвязной нейронной сети можно выразить формулой:

$$\vec{y} = F(A\vec{x} + b)$$

Где $\vec{x}$ - входной вектор, $\vec{y}$ - выход, A - матрица весов, b - вектор смещения, F - функция активации нейронов на этом слое, применяемая поэлементно.

В случае классификации выход модели должен быть целым числом, соответствующим номеру наиболее вероятного класса. Самым распространённым способом получения этого числа является применение функции softmax к выходному вектору, имеющему размерность, равную числу классов.

$$softmax(x_i, \vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

После применения функции softmax к вектору гарантирует, что все его элементы - положительные числа в интервале (0, 1), при этом их сумма всегда равна 1. Это позволяет рассматривать элементы вектора как вероятности принадлежности прогнозируемого объекта каждому из классов. Таким образом, прогнозируемый номер класса - индекс максимального элемента в выходном векторе.

Описанный выше подход применим для любого числа классов, однако в случае бинарной классификации зачастую используется другой метод - вместо двух нейронов последний слой состоит из единственного нейрона с функцией активации σ:

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

Применение данной функции также гарантирует принадлежность ответа интервалу (0, 1), в связи с чем значение можно рассматривать как вероятность принадлежности поданного на вход объекта второму классу, то есть модель относит объект к первому классу, если значение функции меньше чем 0.5, и ко второму в противном случае.

Настройка нейронной сети заключается в подборе оптимальных весов для каждого слоя, для этого применяется метод обратного распространения ошибки: на последнем слое нейросети вычисляется некоторая функция потерь, зависящая от эталонного значения предсказываемого вектора и значения, полученного нейронной сетью, после чего на каждом слое вычисляются производные данной функции по его параметрам и происходит оптимизация весов методом градиентного спуска. В задачах классификации с двумя классами в качестве функции потерь используются бинарная кросс-энтропия:

$$L_{BCE} = -y \log \hat{y} + (1 - y) \log (1 - \hat{y})$$

y - эталонное значение, $\hat{y}$ - предсказанное значение.

Плюсы нейросетевых моделей:

- высокая точность;
- возможность работы с многими типами данных без предобработки (изображения, звук).

Минусы:

- требуют большого объема обучающих примеров;
- долгий и вычислительно сложный процесс настройки;
- нет возможности интерпретации из-за сложной структуры.

Полносвязная нейронная сеть была выбрана в данной работе в качестве базовой модели по двум причинам. Во-первых, нейросетевые модели такой архитектуры показывают очень высокое качество во многих задачах классификации. Во-вторых, возможность работать с любым входным вектором позволяет использовать подходящее для интерпретации пространство признаков, в отличие от других более сложных архитектур, таких как сверточные нейронные сети.

3.2.4 Рекуррентные нейронные сети

Рекуррентные нейронные сети были предложены для решения задач анализа последовательностей, что делает их эффективным подходом при обработке текста, поскольку они позволяют рассматривать его как последовательность токенов (слов), заданных векторами чисел, а не как один монолитный вектор. Благодаря этому свойству при их использовании учитываются порядок слов и их контекст, что приводит к значительному повышению качества в задачах, где контекст представляет особую важность, таких как суммаризация и машинный перевод.

Выходом рекуррентной сети также является последовательность векторов, соответствующих каждому элементу входной последовательности, при этом каждый из них зависит как от текущего элемента последовательности, так и от внутреннего состояния сети, зависящего всех предыдущих элементов.

В задаче классификации требуется получить не набор векторов, а единственный числовой ответ - номер класса входной последовательности. Чтобы привести выход сети к нужному формату зачастую применяется агрегация полученных векторов с помощью суммирования, усреднения или конкатенации. Затем к агрегированному вектору применяется более традиционный классификатор (обычно это полносвязная нейронная сеть небольшого размера) для получения итогового результата.

Ниже приведена схема типичной архитектуры рекуррентных нейронных сетей, применяемой для решения задач с фиксированным размером выхода, в том числе классификации.

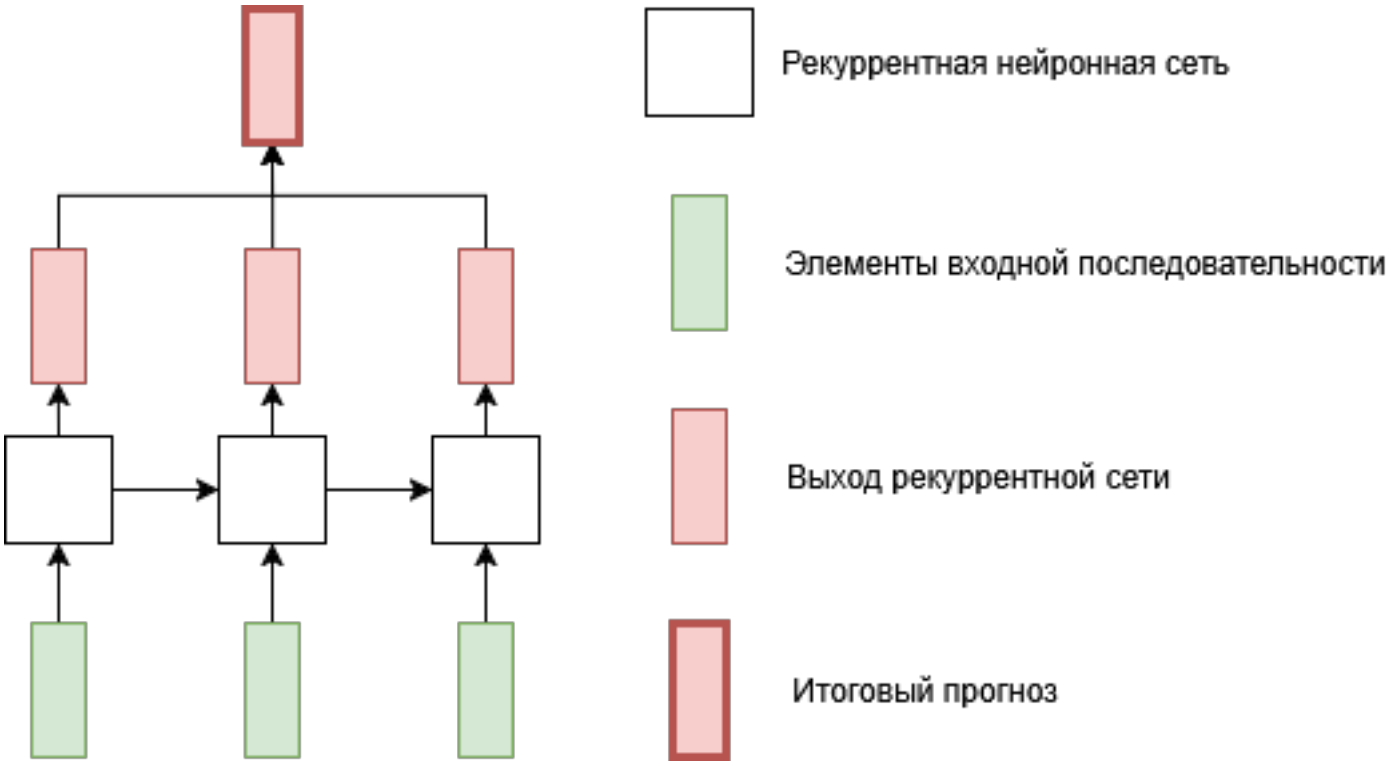


Рис. 3: Схема рекуррентного классификатора

Таким образом, в задачах классификации текста рекуррентная сеть служит для генерации внутреннего представления, учитывающего порядок слов, которое затем используется для классификации всего текста. Данный подход позволяет достичь крайне высокой точности, однако полученное внутреннее представление невозможно интерпретировать, что делает невозможным и интерпретацию всей модели классификации. Использование только рекуррентной части в качестве базовой модели также невозможно, поскольку базовая модель обязана обрабатывать то же пространство признаков, что и дерево решений, которое не способно обрабатывать последовательности.

4 Разработка комбинированной модели

В данном разделе будут рассмотрены основные этапы реализации предложенной модели, а также её главные характеристики. Главные составляющие разрабатываемой программной системы - нейронная сеть и решающее дерево. Для завершения комбинированной модели необходимо:

1. обучить базовую модель (нейронную сеть);
2. протестировать её на отдельном наборе данных с целью проверки качества классификации;
3. обучить дерево решений на откликах базовой модели;
4. протестировать полученное дерево решений.

4.1 Получение и подготовка данных

Первым этапом обучения любой модели является получение достаточного объёма данных, соответствующих поставленной задаче. В данном случае необходим большой объем новостных текстов, как написанных человеком, так и сгенерированных автоматически.

В качестве источника данных был использован датасет NeuralNews [6], созданный для решения задачи выявления новостных статей, сгенерированных нейронными сетями. Данный датасет состоит из 64000 текстовых документов, 32000 из которых - статьи New York Times, остальные документы были получены с помощью модели GROVER [13].

Первым шагом обработки данных является разбиение исходного набора текстов на четыре фрагмента, используемые для обучения и тестирования составляющих комбинированной модели. Этот шаг обусловлен тем, что используемые модели машинного обучения должны быть обучены и протестированы на непересекающихся наборах данных, чтобы избежать переобучения.

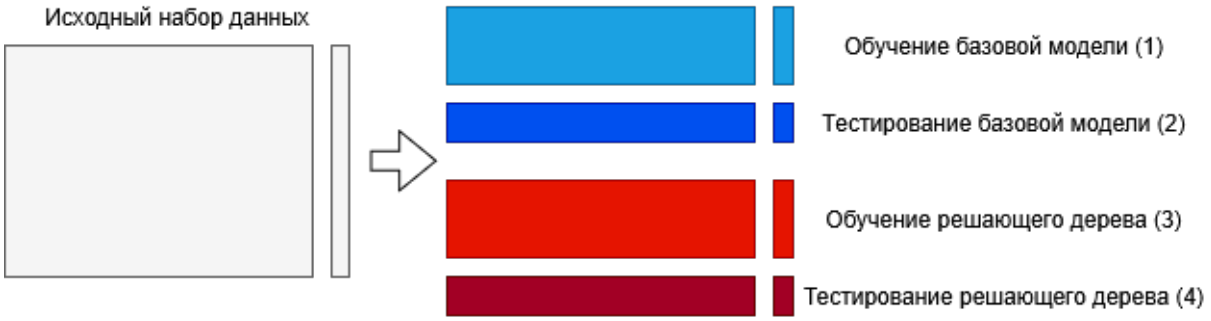


Рис. 4: Схема разделения данных данных

Обозначим полученные наборы данных:

- обучающая выборка базовой модели - набор (1);

- тестовая выборка базовой модели - набор (2);
- обучающая выборка решающего дерева - набор (3);
- тестовая выборка решающего дерева - набор (4).

Для применения используемых моделей машинного обучения, необходимо привести текстовые данные к векторам фиксированной размерности согласно описанным ранее моделям. Ниже описаны использованные для этого алгоритмы.

4.1.1 Пространство "мешок слов"

В рамках модели "мешок слов", составляется словарь, основанный только на текстах обучающего набора данных. Каждое слово в словаре кодируется вектором размерности n, где n - размер имеющегося словаря. Все компоненты этого вектора равны 0, кроме компоненты, индекс которой равен индексу данного слова в словаре, эта компонента равна 1. Для кодирования полного текста все слова, находящиеся в нём и попавшие в словарь, кодируются векторами по описанному выше правилу, после чего эти векторы суммируются. Таким образом, весь текст представляется вектором фиксированной размерности n, в котором каждая компонента равна числу вхождений соответствующего слова в текст.

На практике для получения пространства признаков модели "мешок слов" текст каждого файла заносится в csv-файл без дополнительной обработки, поскольку хранение векторов большой размерности в памяти не эффективно. Конвертация текстов в векторы происходит только при их использовании. Словарь составляется на основе текстов набора (1), удаления стоп-слов и другой очистки текста (за исключением приведения к нижнему регистру) при этом не проводится, чтобы избежать удаления полезной информации. Для эффективной реализации хранения словаря и процесса конвертации используется класс CountVectorizer библиотеки scikit-learn[15].

4.1.2 Пространство на основе текстовых характеристик

В рамках данной более простой модели каждый текст кодируется набором числовых характеристик:

- средняя длина слова;
- средняя длина предложения;
- частота самого популярного слова;
- частоты различных частей речи: имён собственных, имён существительных, глаголов, прилагательных и наречий;
- число слов заголовка (первого предложения) в остальном тексте;
- число ключевых слов в тексте.

Ключевые слова - набор слов, которые имеют большой разрыв между их частотами в корпусе сгенерированных статей и корпусе реальных статей. В данном случае это слова said, he, like, one, new, would, time, but, could и it.

При подготовке данного пространства признаков производится анализ каждого новостного текста в исходном наборе данных. Основные этапы этого анализа:

1. разделение текста - получение набора слов и предложений;
2. удаление небуквенных конструкций;
3. анализ заголовка, получение числа слов заголовка в остальном тексте;
4. получение средней длины слова и предложения на основе полученных на первом шаге наборов;
5. анализ части речи каждого слова с помощью инструмента NLTK[1][16], получение частот частей речи;
6. подсчет числа ключевых слов;
7. поиск самого популярного слова, получение его относительной частоты.

Данный процесс повторяется для каждой новостной статьи, полученные числовые признаки сохраняются в формате csv и используются всеми моделями машинного обучения без дополнительной обработки. Нормализация и другие преобразования не проводятся для сохранения интерпретируемости.

4.2 Обучение нейронный сетей

После получения данных необходимо определить архитектуру нейронной сети и обучить модель на её основе. Как было обозначено ранее, используемая нейронная сеть обязана обрабатывать то же пространства признаков, что и решающее дерево, в связи с чем она может принимать на вход только векторы фиксированной размерности, поскольку это единственный тип объектов, к которому применим алгоритм CART. Это делает невозможным использование более сложных архитектур, обрабатывающих последовательности, таких как рекуррентные сети и трансформеры. В связи с этим, в данной работе использована полносвязная нейронная сеть. Для её реализации была выбрана библиотека PyTorch языка Python.

Параметры сети, такие как число слоев, число нейронов в каждом слое, функции активации и наличие дополнительных нелинейных слоев были получены в результате экспериментов и сравнения работы моделей на тестовой части выборки. Итоговой вид использованных нейронных сетей и формулы их слоев приведены ниже.

[1]Natural Language Toolkit

```
DenseNetwork(
  (layers): Sequential(
    (linear0): Linear(in_features=128768, out_features=64, bias=True)
    (relu0): ReLU()
    (linear1): Linear(in_features=64, out_features=64, bias=True)
    (relu1): ReLU()
    (dropout1): Dropout(p=0.1, inplace=False)
    (linear2): Linear(in_features=64, out_features=64, bias=True)
    (relu2): ReLU()
    (dropout2): Dropout(p=0.1, inplace=False)
    (linear_final): Linear(in_features=64, out_features=1, bias=True)
    (sigmoid): Sigmoid()
  )
)
```

Рис. 5: Структура нейронной сети для пространства "мешок слов"

```
DenseNetwork(
  (layers): Sequential(
    (linear0): Linear(in_features=10, out_features=32, bias=True)
    (relu0): ReLU()
    (linear1): Linear(in_features=32, out_features=32, bias=True)
    (relu1): ReLU()
    (dropout1): Dropout(p=0.1, inplace=False)
    (linear2): Linear(in_features=32, out_features=32, bias=True)
    (relu2): ReLU()
    (dropout2): Dropout(p=0.1, inplace=False)
    (linear3): Linear(in_features=32, out_features=32, bias=True)
    (relu3): ReLU()
    (dropout3): Dropout(p=0.1, inplace=False)
    (linear_final): Linear(in_features=32, out_features=1, bias=True)
    (sigmoid): Sigmoid()
  )
)
```

Рис. 6: Структура нейронной сети для пространства низкой размерности

$$Linear(\vec{x}, A, b) = A\vec{x} + b$$

$$ReLU(x_i) = max(0, x_i)$$

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

$Dropout(x_i, p) = 0$ с вероятностью p, x_i с вероятностью 1 - p

Где A - матрица настраиваемых весов, b - настраиваемый вектор смещения.

Слой dropout используется при обучении модели для обнуления некоторых компонент промежуточных векторов, что соответствует случайному удалению связей между слоями. После обучения случайного выбора не происходит, удаляемые компоненты фиксируются. Эта процедура помогает уменьшить степень переобучения модели и повысить точность на тестовой выборке и реальных данных.

Поскольку ответ нейронной сети является единственным числом - 1 если полученный вектор соответствует сгенерированной новостной статье и 0 в противном случае, необходимо привести её выход к бинарному виду. Для этого на последнем слое сети применяется функция активации сигма, формула которой приведена выше.

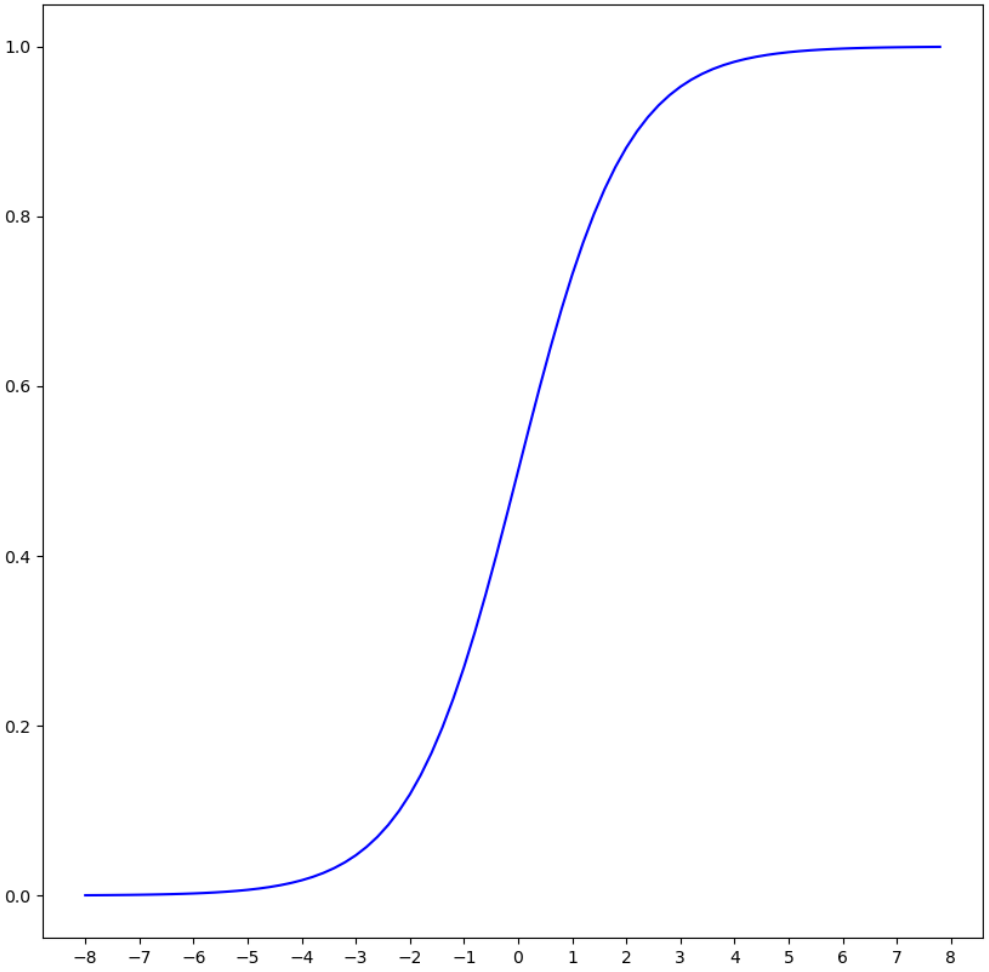


Рис. 7: График сигма функции

Важные свойства сигма-функции:

- $\sigma(0) = 0.5$
- $\lim_{x\to\infty} \sigma(x) = 1$
- $\lim_{x\to-\infty} \sigma(x) = 0$

Применение сигма-функции гарантирует, что предсказание сети будет находиться на интервале (0, 1). Благодаря свойствам данной функции, её значение можно расценивать как вероятность принадлежности одному из классов - при достаточно больших по модулю значениях аргумента уверенность прогноза максимальна, нулевое значение аргумента означает, что класс объекта установить не удалось. Таким образом, объект принадлежит классу 1 если предсказание нейронной сети больше 0.5 и классу 0 в противном случае.

Для обучения моделей был выбран алгоритм оптимизации Adam[14], являющийся усовершенствованной версией градиентного спуска. В качестве функции потерь была выбрана среднеквадратичная ошибка (MSE[1]) в случае пространства "мешок слов" и бинарная кросс энтропия (BCE[2]) в случае пространства низкой размерности:

$$MSE(\vec{x}, \vec{y}) = \frac{\sum_{i=0}^{n}(x_i - y_i)^2}{n}$$

$$BCE(\vec{x}, \vec{y}) = -\frac{\sum_{i=0}^{n}(y_i log(x_i) + (1 - y_i) log(1 - x_i))}{n}$$

Где $\vec{x}$ - вектор предсказанных нейросетью значений, $\vec{y}$ - вектор реальных меток классов, n - размерность этих векторов.

На графиках ниже представлено изменение точности нейронной сети на наборах данных (1) и (2) при увеличении числа эпох обучения (полных проходов по обучающей выборке).

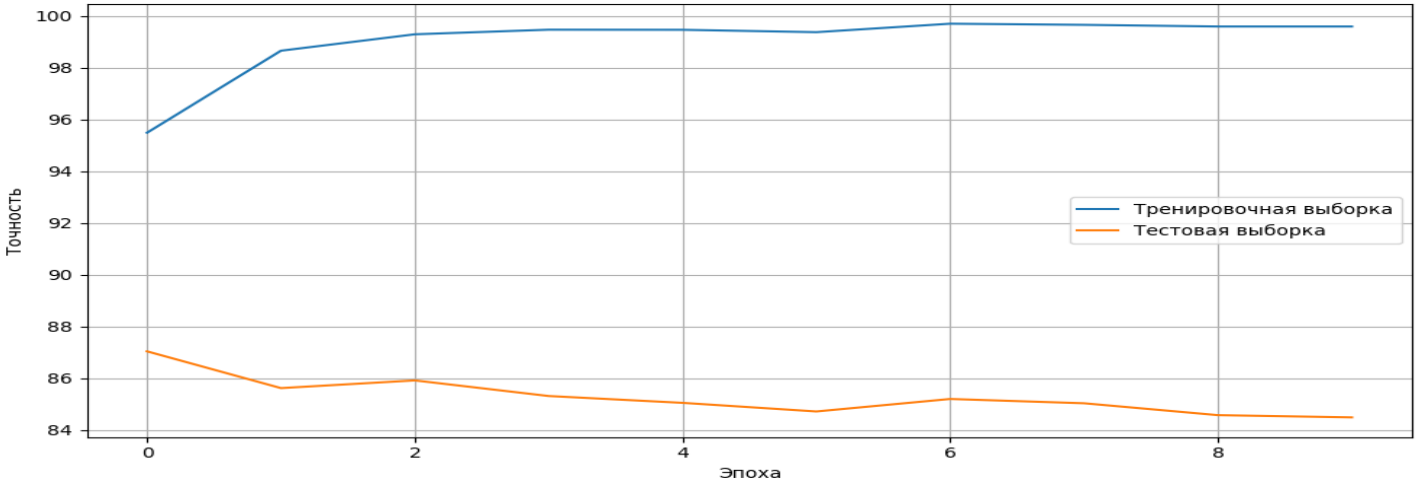
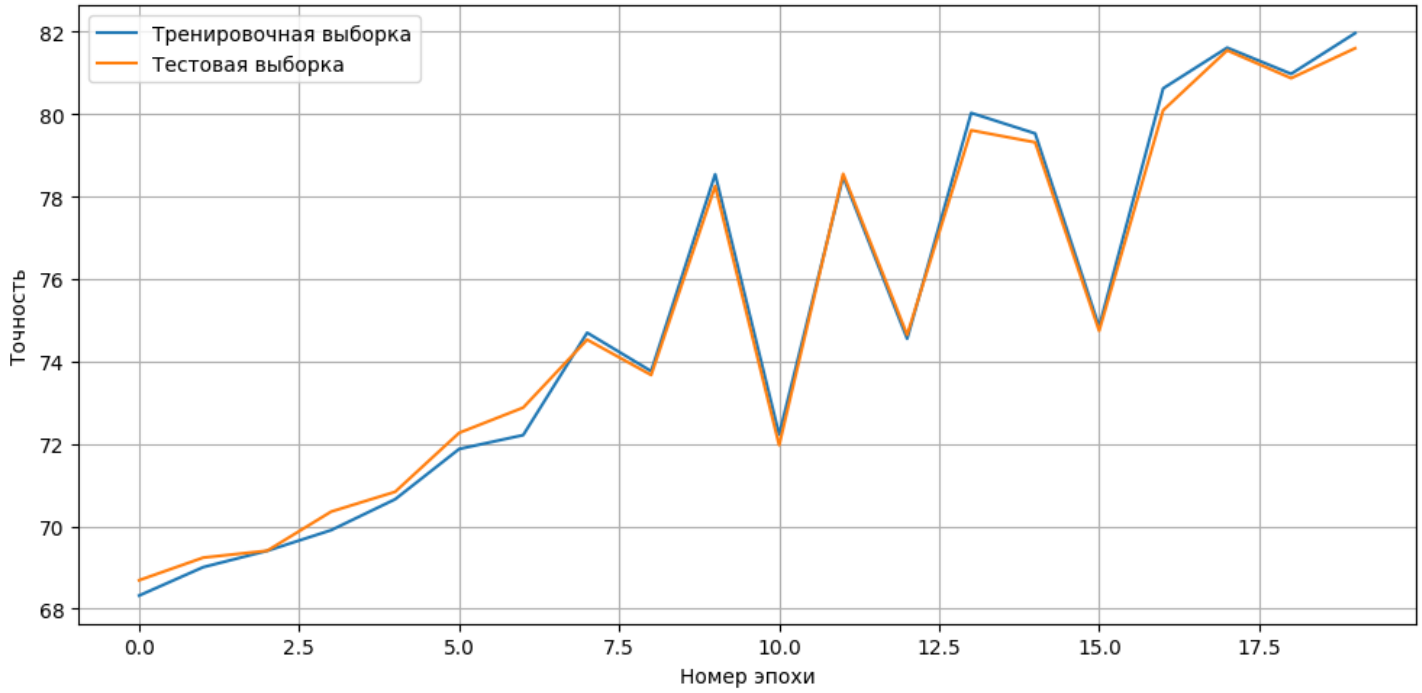


Рис. 8: Обучение нейронной сети для пространства "мешок слов"

[1]Mean Squared Error
[2]Binary Cross Entropy

Данная нейронная сеть показывает практически идеальную точность на обучающих данных даже при небольшом числе эпох, однако наблюдается большой разрыв между точностью на обучающей и тестовой выборках, что свидетельствует о сильном переобучении модели. Для снижения переобучения было уменьшено число слоев сети и применен метод dropout, описанный ранее. Несмотря на недостатки модели, её точность на тестовой выборке по-прежнему достаточно высока, что позволяет использовать её в качестве базовой модели.



2025-03-29 12:42:03.628816

Рис. 9: Обучение нейронной сети для пространства низкой размерности

Данная нейронная сеть имеет менее высокую точность и требует значительно большего числа эпох для обучения. Это связано с тем, что она обрабатывает небольшое число признаков различной природы, что затрудняет процесс оптимизации. В отличие от предыдущей нейросети, эта модель не проявляет признаков переобучения, в связи с чем её точность на тестовой выборке сравнима с точностью предыдущей модели.

4.3 Переход к дереву решений

Для реализации комбинированной модели необходимо объединить базовую модель и дерево решений. Реализация данного процесса заключается в получении прогнозов базовой модели (в том числе полученной на предыдущем шаге нейронной сети), обученной на наборе данных (1) для всех объектов набора (3). Спрогнозированные метки классов используются вместе с объектами набора (3) для обучения дерева решений.

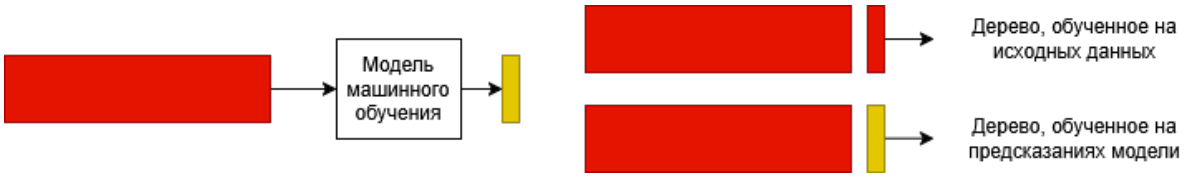


Рис. 10: Объединение моделей

Таким образом, решающее дерево приближает не реальную зависимость между данными и классами, а зависимость, обнаруженную базовой моделью. Точность классификации такого дерева по очевидным причинам не может превышать точность исходной модели, однако она может быть выше точности дерева, обученного на исходных данных. Это связано с рассмотренными ранее свойствами алгоритма CART, особенно склонностью к переобучению. Данные, полученные после применения базовой модели, будут иметь более простую структуру и меньшее число выбросов, что значительно снизит степень переобученности модели и позволит достигнуть более высокой точности на реальных данных.

4.4 Обучение деревьев решений

После получения вектора предсказаний базовой модели для набора (3), он используется вместо реальных меток классов для обучения деревьев с использованием описанного ранее алгоритма CART. В качестве реализации алгоритма CART был использован класс DecisionTreeClassifier библиотеки scikit-learn.

Основными параметрами алгоритма являются максимальная глубина дерева и функция потерь. Другие гиперпараметры, такие как минимальное число объектов в подпространстве, оказывают минимальное влияние на полученное дерево в данной задаче. Модели будут протестированы для различных значений максимальной глубины с целью получения дополнительной информации. Варианты функции потерь приведены ниже:

Эмпирическая вероятность объекта из класса k попасть в подпространство R_m:

$$p_{mk} = \frac{1}{|R_m|} \sum_{i:x_i \in R_m} I(y_i = k)$$

Критерий Джини:

$$Q_{gini} = \sum_{m=1}^{M} \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

Энтропия:

$$H = -\sum_{m=1}^{M} \sum_{k=1}^{K} p_{mk} \log_2(p_{mk})$$

Где x_i - объекты выборки, y_i - их метки, R_m - подпространства, полученные после потенциального разбиения, k - номер класса, K - общее число классов, M - число подпространств, полученных при разбиении, I - индикаторная функция.

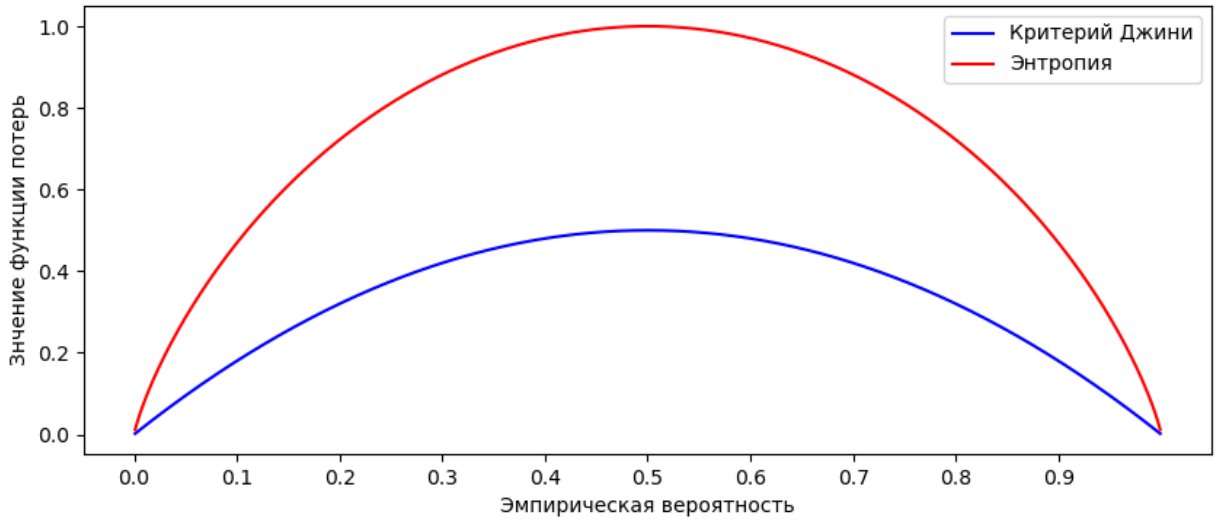


Рис. 11: Графики функций потерь для двух классов

В качестве функции потерь был выбран критерий Джини. Сравнение доступных критериев показывает крайне небольшое различие между использованными функциями потерь, однако критерий Джини показывает более высокую точность для деревьев небольшой глубины.

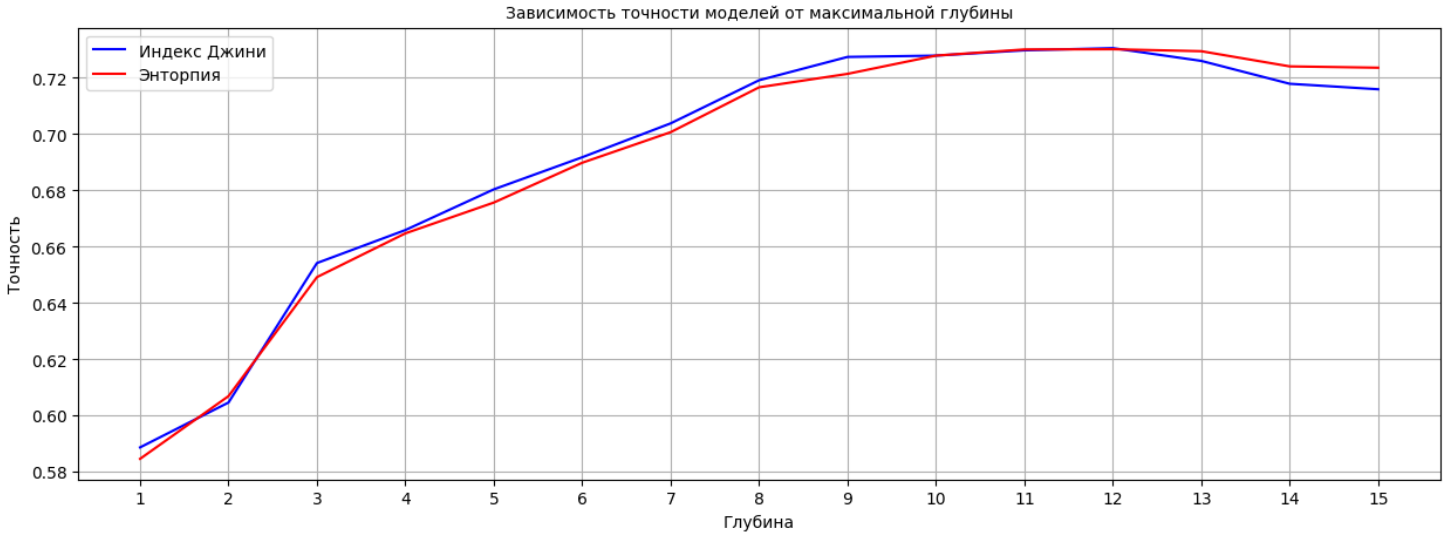


Рис. 12: Сравнений функций потерь алгоритма CART

После обучения происходит проверка качества полученных моделей на тестовой части выборки деревьев решений, эти данные не были использованы ни в одном из предыдущих шагов, благодаря чему они позволяют получить объективные результаты. На графиках ниже представлено сравнение точности деревьев решений, обученных на прогнозах различных моделей, с моделью, обученной на исходных данных.

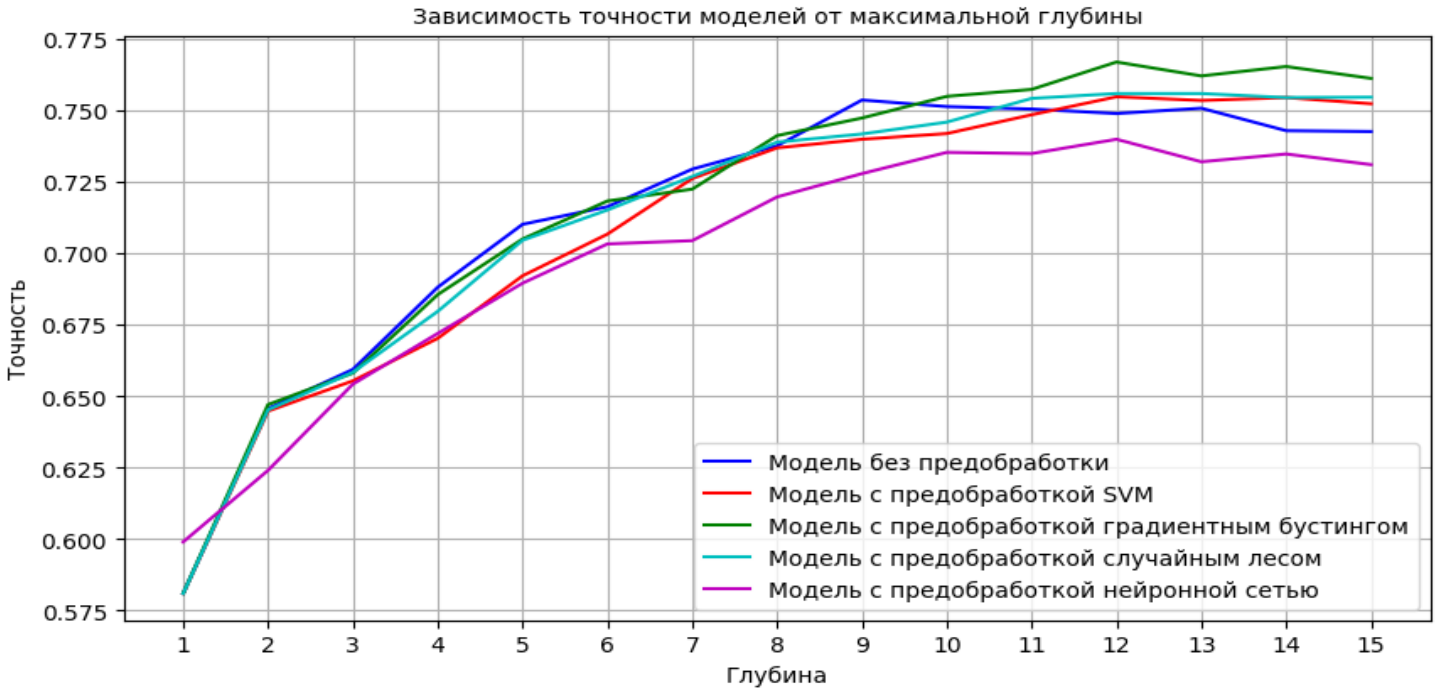


Рис. 13: Сравнение точности всех полученных моделей для пространства низкой размерности

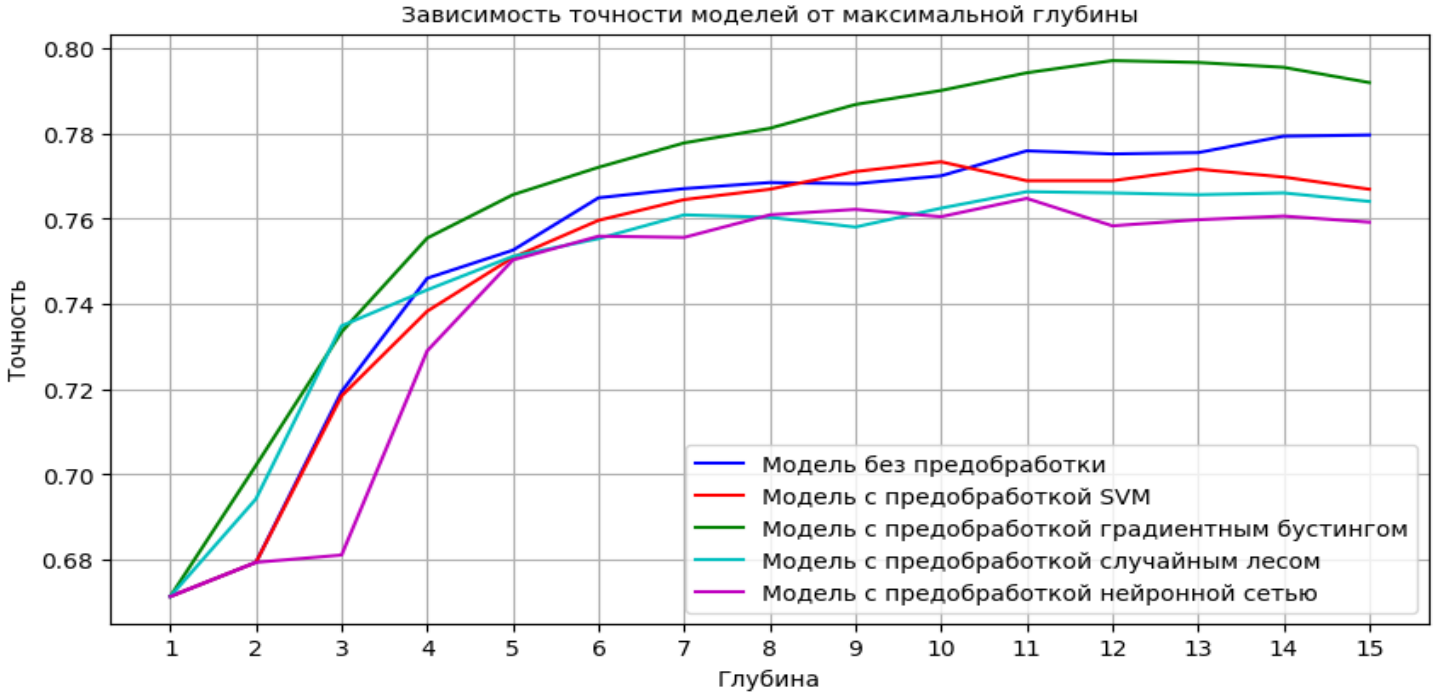


Рис. 14: Сравнение точности всех полученных моделей для пространства "мешок слов"

Как видно из полученных графиков, решающие деревья, обученные на прогнозах нейронной сети, имеют самую низкую точность, в то время как модели, обученные на прогнозах градиентного бустинга, значительно превосходят все остальные варианты.

Важно отметить, что все используемые базовые модели показывали схожую точность при их обучении и тестировании на наборах данных (1) и (2), но оказали кардинально разное влияние на точность дерева решений. Это означает, что качество предсказаний модели не даёт достаточно информации, чтобы определить её влияние на прогнозы решающего дерева.

4.5 Применение итоговой модели

4.5.1 Интерпретация результата

После обучения решающего дерева комбинированная модель считается завершенной, для её применения к новой новостной статье необходимо:

1. получить вектор, соответствующий данному тексту с помощью имеющегося словаря или программы;

2. совершить проход от корня полученного решающего дерева к одному из листьев;

3. присвоить статье класс, соответствующий полученному листу.

Важно отметить, что использование нейронной сети или другой базовой модели после обучения не требуется, базовая модель необходима только для построения более точного дерева решений.

Процесс интерпретации модели заключается в условиях, проверенных при проходе от корня к листу. Каждый шаг этого прохода объясняется значением очередного условия. Пример интерпретации для небольшого фрагмента полученного дерева решений приведен ниже. Интерпретация полного прохода от корня не приведена, поскольку полное дерево решений невозможно графически отобразить на одной странице из-за его размера (порядка 1900 вершин). В данном примере был совершен переход по отмеченному маршруту, поскольку доля имен собственных в рассматриваемом тексте принадлежит отрезку [0,123; 0,191], а доля имен прилагательных меньше 0,059.

Таким образом, каждой новостной статье соответствует не только прогноз модели, но и набор условий, позволяющий получить дополнительную информацию об исследуемом тексте. Такая прозрачность механизма выбора прогноза позволяет интерпретировать построенное дерево решений и делать выводы об исходных данных на его основе.

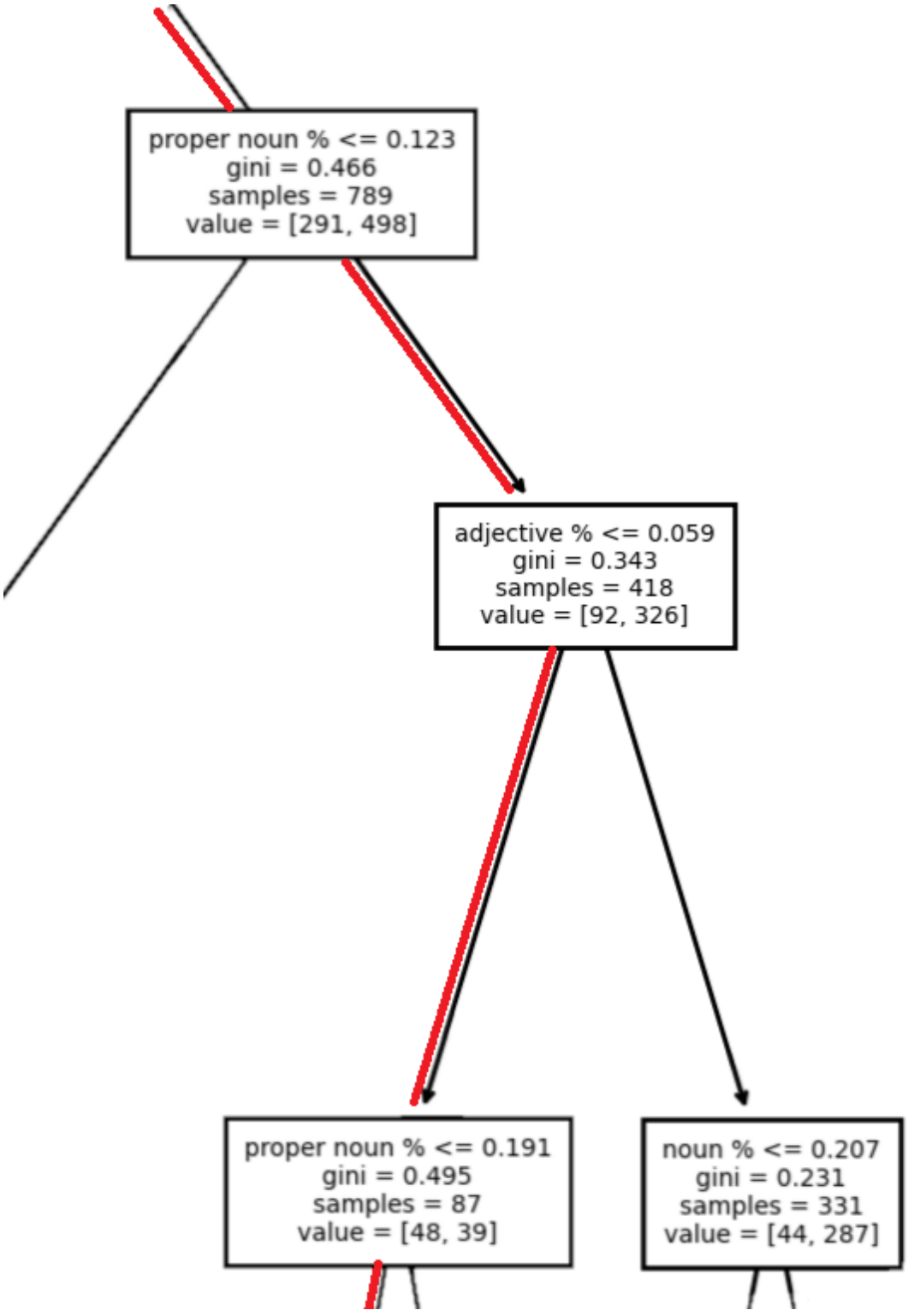


Рис. 15: Пример прохода по дереву

4.5.2 Результаты экспериментов

В таблицах ниже приведено полное сравнение моделей, обученных в рамках работы, при их тестировании на наборе данных (4). Комбинированная модель в данном случае - дерево решений, обученное на прогнозах градиентного бустинга, то есть самый успешный вариант её реализации. Максимальная глубина деревьев решений во всех таблицах - 12. Класс 0 обозначает новостные статьи, написанные человеком, класс 1 - сгенерированные.

	Нейронная сеть	Дерево решений	Комбинированная модель
Класс 0, предсказан 0	3036	2586	2671
Класс 0, предсказан 1	499	949	864
Класс 1, предсказан 0	758	792	768
Класс 1, предсказан 1	2700	2666	2690

Таблица 1: Результаты тестирования моделей для пространства низкой размерности

Применение комбинированного подхода в данном случае позволяет улучшить все параметры дерева решений, при это наибольший прирост точности происходит для объектов класса 0, то есть подлинных новостных статей.

	Нейронная сеть	Дерево решений	Комбинированная модель
Класс 0, предсказан 0	2951	2778	2676
Класс 0, предсказан 1	496	669	771
Класс 1, предсказан 0	549	903	661
Класс 1, предсказан 1	2997	2643	2885

Таблица 2: Результаты тестирования моделей для пространства ”мешок слов”

Для этого пространства признаков комбинированная модель показывает более низкую точность, чем одиночное решающее дерево, при обработке объектов класса 0, однако она позволяет получить большой прирост точности при обработке сгенерированных текстов (класс 1). Это свойство влияет на общую точность модели намного сильнее, чем небольшое ухудшение на реальных текстах, что дает ей значительное преимущество.

5 Заключение

Базовые модели, основанные на деревьях решений (случайный лес и градиентный бустинг) показали лучшие результаты для пространства низкой размерности, что может свидетельствовать о том, что решающие деревья получают наибольший прирост точности при обучении на предсказаниях моделей, также основанных на деревьях решений.

Выводы: обучение на прогнозах нейронной сети ведет к потере точности. Сравнение моделей показывает, что ансамбли решающих деревьев имеют наилучшее влияние на точность объясняющей модели, эту гипотезу необходимо проверить на более разнообразных данных.

В выпускной квалификационной работе достигнуты следующие результаты:

- Исследованы методы интерпретации моделей машинного обучения.
- Предложена комбинированная модель на основе дерева решений и нейронной сети, решающая задачу выявления сгенерированных новостных текстов.
- Разработана и реализована программная система на основе комбинированной модели, проведено сравнение с другими методами решения задачи.

Список литературы

[1] Radford, A. Improving language understanding by generative pre-training, 2018.

[2] Brown T. et al., Language Models are Few-Shot Learners, 2020.

[3] Breiman L., Friedman J. H., Olshen R. A., & Stone C. J. Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[4] Чистяков С. П. Случайные леса: обзор Труды КарНЦ РАН. 2013. №1.

[5] Friedman, Jerome H. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics - 2001. 29(5), С. 1189–1232.

[6] Reuben Tan, Bryan A. Plumme, Kate Saenko, Detecting Cross-Modal Inconsistency to Defend Against Neural Fake News. Empirical Methods in Natural Language Processing (EMNLP), 2020.

[7] C.P. Manning, P. Raghavan, and H. Schütze, Introduction to InformationRetrieval, Cambridge Univ. Press, 2008.

[8] Mikolov T. Efficient estimation of word representations in vector space, 2013.

[9] Elman J. L. Finding Structure in Time // Cognitive Science. — 1990. — Т. 14, № 2. — С. 179—211.

[10] Jordan, M. I. Serial Order: A Parallel Distributed Processing Approach // Advances in Psychology. - 1997. Т. 121. - С. 471—495.

[11] Sepp Hochreiter, Jürgen Schmidhuber. Long short-term memory // Neural Computation - 1997. Т. 9. №8. - С. 1735—1780

[12] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

[13] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. // Advances in Neural Information Process-ing Systems - 2019, С. 9051–9062,

[14] Diederik Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 2014.

[15] scikit-learn [Электронный ресурс]. – Электрон. дан. – URL: https://scikit-learn.org/stable.

[16] NLTK :: Natural Language Toolkit [Электронный ресурс]. – Электрон. дан. – URL: https://www.nltk.org.