Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных и машинного обучения**

Пояснительная записка к курсовой работе

по дисциплине “Технологии анализа данных и машинного

обучения”

на тему:

«Выявление вредоносного программного обеспечения по заголовкам файлов»

Выполнила:

студентка группы ПИ20-3 факультета информационных технологий и анализа больших данных

Кириллова Е. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

доцент, к.ф.-м.н. Попов В.Г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2023 г

Оглавление

[Введение 3](#_Toc134642590)

[Основные понятия 4](#_Toc134642591)

[Машинное и глубокое обучения 4](#_Toc134642592)

[Вредоносное программное обеспечение 5](#_Toc134642593)

[Особенности обработки текстовых данных 6](#_Toc134642594)

[Метрики качества 7](#_Toc134642595)

[Выбор и описание датасета 9](#_Toc134642596)

[Анализ и предобработка датасета 11](#_Toc134642597)

[Создание и обучение моделей 16](#_Toc134642598)

[Алгоритмы машинного обучения 18](#_Toc134642599)

[Алгоритмы глубокого обучения 24](#_Toc134642600)

[Сравнение моделей 26](#_Toc134642601)

[Заключение 28](#_Toc134642602)

[Источники 29](#_Toc134642603)

# Введение

Развитие современного мира информационных технологий сопровождается увеличением числа кибератак, которые могут привести к серьезным последствиям для пользователей. С каждым днем вредоносные программы становятся все более разнообразными и опасными. Главная их опасность в том, что они распространяются незаметно и быстро, из-за чего за ними тяжело уследить. Так как вредоносные программы являются одной из основных угроз информационной безопасности, необходимо как можно быстрее определить их и обезвредить [1]. Одним из наиболее распространенных видов киберугроз является вредоносное программное обеспечение, которое может нанести значительный ущерб компьютерной системе. Чтобы заранее определить вредоносный файл или нет, нужно преждевременно его проверить на угрозу безопасности для компьютерной системы наиболее безопасным путем. В данной работе я буду рассматривать метод борьбы с вредоносным программным обеспечением с помощью выявления его по заголовкам файлов. Данный метод подразумевает анализ названия файлов и определения по ним: является файл вредоносным или нет.

Целью данной работы является определение наиболее эффективного метода выявления вредоносного программного обеспечения с помощью сравнения методов машинного обучения и глубокого обучения. Обучение моделей я буду производить как на основе хешированных названий файлов, переведенных в числовые векторы, так и на основе числовых данных файла (размер, значение энтропии, адрес памяти первого байта раздела), чтобы понять, насколько эффективно или неэффективно определение опасности файла на основе его названия.

Работа выполнена на языке Python в среде Jupyter Notebook. При обучении моделей использовались библиотеки scikit-learn и Keras.

# Основные понятия

## Машинное и глубокое обучения

Глубокое обучение и машинное обучение являются двумя подмножествами искусственного интеллекта.

Машинное обучение — это подход к искусственному интеллекту, который позволяет компьютерным системам самостоятельно извлекать знания из данных и использовать их для принятия решений без явного программирования. В машинном обучении используются различные алгоритмы, которые позволяют компьютеру "обучаться" на основе предоставленных ему данных и опыта.

Машинное обучение находит широкое применение в различных областях, таких как финансы, медицина, маркетинг, производство и т. д. Оно используется для прогнозирования трендов, оптимизации процессов, улучшения качества продукции и многого другого [2].

Глубокое обучение — это подмножество машинного обучения, которое использует нейронные сети с большим количеством слоев для обработки и анализа данных. Основное отличие глубокого обучения от машинного заключается в том, что глубокое обучение способно автоматически извлекать признаки из данных, в то время как в машинном обучении признаки должны быть явно заданы. Глубокое обучение также может использоваться для решения более сложных задач, таких как распознавание изображений, обработка естественного языка и автономная навигация.

В глубоком обучение, пока создаются все более крупные нейронные сети, используя для обучения большие объемы данных, их производительность показывает рост. При этом, в машинном обучение, если усложнять и использовать большие объемы данных, то скорость работы не увеличивается. Так глубокое обучение становится первым классом, который допускает масштабирование [3].

## Вредоносное программное обеспечение

Вредоносное программное обеспечение (англ. malware) — это любое программное обеспечение, которое разработано с целью нанесения вреда компьютерной системе, сети или данных пользователя. Вредоносное ПО может проникать на устройства пользователя через электронную почту, веб-сайты, загрузки файлов и другие источники.

Вредоносные программы классифицируются по их поведению или цели. Различаются такие типы вредоносных программ, как: вирусы, бэкдоры, программы-вымогатели, трояны и так далее. В зависимости от их вида они наносят разный ущерб компьютерной системе.

Чтобы не стать жертвой атаки вредоносного программного обеспечения, обычно люди устанавливают на устройства антивирусные программы, создают автономные резервные копии, внедряют расширенную защиту почты и т. д. [4].

В нашем случае мы обучим модели с помощью методов машинного и глубокого обучения, чтобы выявить опасное ПО.

## Особенности обработки текстовых данных

Обработка текстовых данных имеет свои особенности, так как мы не можем использовать текст в чистом виде для обучения модели, нам нужно перевести его в числовой формат.

Ключевыми особенностями являются:

1. Предобработка данных. Перед работай с данными нам нужно их обработать: удалить стоп-слова, заняться лемматизацией и токенизацией.
2. Векторизация. Нужно преобразовать текстовые данные в числовые векторы.
3. Размерность. В текстовых данных может содержаться большое количество признаков, что может вызвать затруднения. Поэтому иногда нужно уменьшать размерность данных для более корректной работы.
4. Выбор подходящей модели. Различные модели работают хорошо только с определенными типами данных. Наиболее распространенными моделями для обработки текстовых данных считаются Naive Bayes, SVM, нейронные сети [5].

## Метрики качества

Метрики качества — это числовые показатели, которые используются для оценки производительности модели машинного обучения. Они позволяют оценить, насколько хорошо модель работает на тестовых данных и как она справляется с поставленной задачей. По метрикам качества можно определить, насколько правильно обучена модель.

Матрица ошибок — это таблица, которая показывает, сколько объектов было классифицировано верно и сколько ошибочно в каждом из классов. Она является основным инструментом для оценки производительности модели в задачах классификации

В данной работе классификация будет проводиться по двум классам, поэтому матрица ошибок будет размером 2х2. Ее структура будет выглядеть таким образом:

|  |  |  |
| --- | --- | --- |
|  | 1 | 0 |
| 1 | True Positive (TP) | False Positive (FP) |
| 0 | False Negative (FN) | True Negative (TN) |

Таблица 1 – Матрица ошибок

Описание показателей:

- True Positive (TP) - количество объектов, которые были правильно классифицированы как положительные;

- False Positive (FP) - количество объектов, которые были неправильно классифицированы как положительные;

- True Negative (TN) - количество объектов, которые были правильно классифицированы как отрицательные;

- False Negative (FN) - количество объектов, которые были неправильно классифицированы как отрицательные.

Эти показатели берутся за основу для вычисления метрик качества. Основными метриками качества, которые мы будем использовать, называются accuracy, precision, f1-score и recall.

Описание метрик:

* Accuracy(точность) — это доля правильно классифицированных объектов от общего числа объектов в выборке. Она показывает, насколько хорошо модель справляется с задачей классификации в целом.
* Precision(точность) — это доля правильно классифицированных положительных объектов от общего числа объектов, которые модель отнесла к положительному классу. Она показывает, насколько точно модель определяет положительные объекты.
* Recall (полнота) — это доля правильно классифицированных положительных объектов от общего числа положительных объектов в выборке. Она показывает, насколько хорошо модель находит все положительные объекты.
* F1-score (F1-мера) — это гармоническое среднее между точностью и полнотой. Она показывает баланс между точностью и полнотой и является хорошей метрикой для оценки производительности модели в задачах с несбалансированными классами [6].

# Выбор и описание датасета

Для выполнения курсовой работы я буду работать с датасетом под названием «Malware Analysis Datasets: PE Section Headers» [7]. Датасет содержит данные статистического анализа (заголовки разделов PE секций .text, .code и CODE). Примеры вредоносных программ были взяты с сайта virusshare.com, а примеры программного обеспечения с portableapps.com и из каталогов Windows 7 x86.

Изначально датасет выглядит таким образом:

Изображение выглядит как стол

Автоматически созданное описание

Рис. 1 – Датасет и его размер

В датасете у нас есть 43293 измерений и 6 атрибутов под названиями: hash, size\_of\_data, virtual\_address, entropy, virtual\_size, malware.

Посмотрим, что содержится в каждой из колонок. В hash находятся названия файлов, хешированные с помощью 128-битного алгоритма под названием MD5, представлены в строковом формате. size\_of\_data целочисленного формата, содержит данные размере файла на диске. virtual\_address тоже является атрибутом с целыми числами, содержит данные об адресе памяти первого байта раздела относительно базы изображения. entropy содержит числа с плавающей точкой (float), тут находятся данные о расчетной энтропии раздела. virtual\_size формата integer, тут находится информация о первоначальном размере файла при загрузке. И последний, но не менее важный, атрибут malware. В данном столбце можно узнать является файл вредоносным (1) или нет (0). Это категориальный признак, позже буду проводить классификацию по нему.

# Анализ и предобработка датасета

Проведем анализ и предобработку датасета, сначала проверю на наличие пропусков:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 – Количество пропусков и информация о датасете

Как можно заметить, пропусков нет. В случае их наличия пришлось бы заняться их удалением или заполнением значениями. Так как их нет, перейдем к следующему шагу.

Проведем проверку на наличие дубликатов, для этого используем функцию duplicated(). Было найдено 24 повторяющейся строки.

Изображение выглядит как стол

Автоматически созданное описание

Рис. 3 - Дубликаты

Хоть их и немного, но лучше удалить, чтобы не иметь повторяющихся данных в таблице. Большое количество дубликатов может неправильно отразиться на результате.

После удаления еще раз проверяем количество дубликатов, теперь их стало 0.

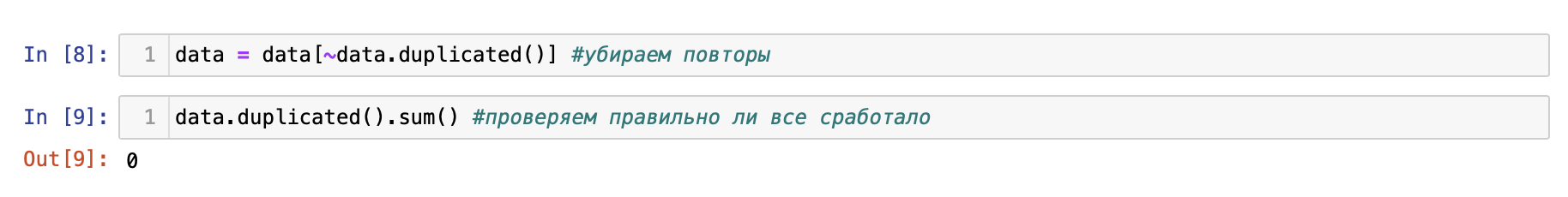


Рис. 4 – Устранение дубликатов

Проведу анализ колонок virtual\_size и size\_of\_data по признаку malware, для этого построю диаграмму отношения relplot с помощью библиотеки seaborn.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 5 – Диаграмма отношения virtual\_size и size\_of\_data

Из этой диаграммы можно сделать вывод, что в основном virtual\_size и size\_of\_data имеют размеры 1:1 или очень близки к этому соотношению. Но стоит обратить внимание на то, что при size\_of\_data = 0 существует множество значений virtual\_size > 0, все из которых являются вредоносными. На основе этого могу сделать предположение, что часть вредоносного ПО имеет схожее поведение: размер при загрузке файла больше, чем итоговый размер файла на диске.

Проанализируем категориальный признак, для этого рассмотрим распределение данных по двум классам с помощью диаграммы.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 6 - Диаграмма соотношения категорий признака malware

На рисунке пять отображена диаграмма, которую я построила с помощью библиотек matplotlib и seaborn, используя функции countplot(), которая строит столбчатую диаграмму с помощью полученных значений [8], и стандартную функцию show(), чтобы отобразить результат.

На диаграмме мы видим, что в датасете в основном присутствуют вредоносные программы. Можно сделать вывод, что в датасете дисбаланс классов. Это может плохо повлиять на качество обучения, поэтому уменьшу выборку примерно до 9 тысяч, в таком случае безопасное программное обеспечение будет иметь большую долю в данных и такого количества будет достаточно, чтобы качественно обучить модель. Для этого использую функцию resample из sklearn и concat из pandas для создания нового поднабора данных по классу 1 и последующего объединения с классом 0.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 7 – Устранение дисбаланса датасета

Еще раз рисуем диаграмму, только теперь на меньшем количестве данных.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 8 – Обновленная диаграммы соотношения категорий признака malware

На этом заканчиваю предобработку данных, часть с обработкой текста и разделением выборки на обучающую и тестовую описана в следующей главе

# Создание и обучение моделей

Для сравнения в данной работе я собираюсь использовать алгоритмы машинного и глубокого обучений, таким образом можно будет выявить наиболее результативный алгоритм для решения задачи по выявлению вредоносного программного обеспечения. Обучение моделей я буду производить на текстовых данных (название файла) и числовых данных (размер файла, значение энтропии, адрес памяти первого байта раздела) отдельно, таким образом смогу понять, является ли выявление вредоносного ПО по названию файла эффективным.

Для начала разберемся с текстовыми данными. Отделим категориальный признак от остальных данных и удаляем из основной таблицы.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 9 – Отделяем целевую переменную от остальных признаков

Так как мы работаем со строковыми данными (заголовки файлов), то в X оставим только столбец hash, а в Y у нас будет категориальный признак malware.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 10 – Разделение на набор признаков и целевую переменную

Теперь нужно разделеть на обучающую и тестовую выборку, для этого из библиотеки sklearn.model\_selection импортируем функцию train\_test\_split. Как видно из названия, она делит полученные данные на обучающую и тестовую выборку в соответствие с указанными параметрами [9].

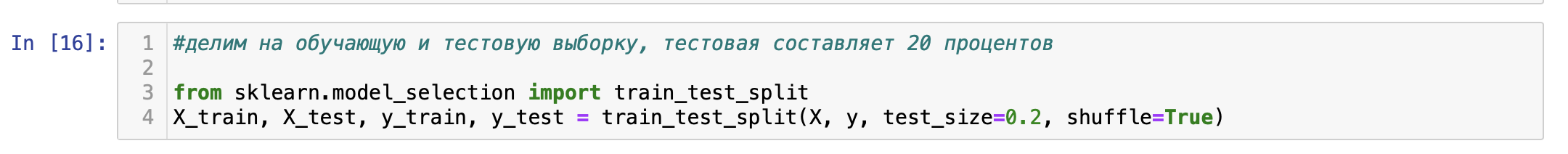


Рис. 11 – Деление на обучающую и тестовую выборку

Так как мы работаем со строковыми данными, то необходимо произвести последнюю подготовку перед началом обучения. Мы не можем использовать строковые данные в обучение, элементарно выпадет ошибка, так как компилятор не сможет их обработать. Поэтому преобразуем строки в векторы, в числовой формат. Для этого буду использовать CountVectorizer из библиотеки sklearn [10]. Данная функция используется для преобразования текстовых документов в матрицы частоты встречаемых слов. Сделаем это таким образом:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 12 – Преобразование текста в набор численных векторов

Теперь будем использовать обработанные данные для обучения моделей.

## Алгоритмы машинного обучения

Когда работаешь с числовыми данными в машинном обучение, то первый алгоритм, который приходит в голову – это мультиномиальный наивный Байес.

MultinomialNB (Multinomial Naive Bayes) — это алгоритм классификации текстовых данных, который использует метод наивного Байеса. Он основан на предположении о независимости каждого признака от других признаков в наборе данных. Используется для классификации текстовых документов, таких как электронные письма, статьи, отзывы и т.д. Алгоритм работает с множеством признаков, которые представляют собой слова или термины, встречающиеся в документе [11].

Данный алгоритм является эффективным и быстрым решением для классификации текстовых данных, который может использоваться для решения различных задач, таких как фильтрация спама, анализ тональности текстов и т.д.

Создаем и обучаем модель на основе уже переведенного текста в числовые векторы:

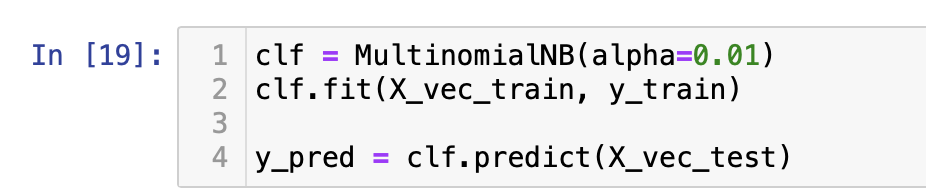


Рис. 13 – Модель с применением алгоритма MultinomialNB с обучением и предсказанием на текстовых данных

Для обучения модели используем функцию fit, прогнозирование целевой переменной делаем на значениях обученной модели с помощью функции predict.

Следующим шагом будет сравнить эти значения с действительными, чтобы понять, насколько хорошо обучена модель.

Посмотрим на матрице ошибок результат и проанализируем его:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 14 - Матрица ошибок модели с применением алгоритма MultinomialNB с обучением на текстовых данных

На рисунке №14 изображена матрица ошибок при обучении с помощью алгоритма MultinomialNB на текстовых данных. На матрице видно, что модель имеет большое количество правильно предсказанных вредоносных программ, но при этом очень мало предсказывает верно не вредоносного программного обеспечения. На самом деле, это не критично. Так как выявление вредоносных файлов важнее, чем безопасных. Большинство вредоносного ПО было предсказано верно, таким образом модель выполняет свою основную функцию.

Если опираться на значение метрик, то модель имеет хорошие показатели – большинство значений предсказано верно.

Изображение выглядит как стол

Автоматически созданное описание

Рис. 15 - Значения метрик модели с применением алгоритма MultinomialNB с обучением на текстовых данных

Для сравнения, опробовала этот алгоритм и на обучение числовых данных. В данном случае я проводила обучение на всех столбцах, кроме hash, который изначально содержал числовые данные.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 16 – Матрица ошибок модели с применением алгоритма MultinomialNB с обучением на числовых данных

Результат сильно отличается от того, что мы рассматривали до этого. Если судить визуально, только по матрице ошибок, то результат выглядит лучше, чем на предыдущей модели, но если взглянуть на значения метрик, то становится понятно, что данная модель обучена намного хуже.

Изображение выглядит как стол

Автоматически созданное описание

Рис. 17 - Значения метрик модели с применением алгоритма MultinomialNB с обучением на числовых данных

Точность модели составляет 57%, что очень мало. Также множество вредоносного ПО предсказано неверно. Из этого можно сделать вывод, что использование алгоритма MultinomialNB для обучения на числовых данных не является результативным. Этот алгоритм больше подходит для обучения на текстовых данных (числовых векторах).

Дальше я рассмотрю алгоритм RandomForestClassifier. Это алгоритм классификации, основанный на методе случайного леса. Случайный лес — это ансамбль решающих деревьев, который строит множество деревьев и использует их для принятия решений. Каждое дерево строится на основе случайной подвыборки данных и случайного подмножества признаков. Каждое дерево в лесу голосует за класс, который был выбран большинством деревьев.

RandomForestClassifier имеет несколько гиперпараметров, которые могут быть настроены для оптимальной производительности, включая количество деревьев, глубину деревьев и количество признаков, которые следует использовать при построении каждого дерева. Алгоритм может использоваться для решения задач классификации, таких как определение категории электронной почты как спам или не спам, классификация изображений и т.д. Он также может использоваться для решения задач регрессии, таких как прогнозирование цен на недвижимость или оценка рисков в финансовой отрасли [12].

При обучении модели на текстовых данных, результат не сильно отличается от того, когда мы использовали алгоритм MultinomialNB.

Изображение выглядит как стол

Автоматически созданное описание

Рис. 18 - Значения метрик модели с применением алгоритма RandomForestClassifier с обучением на текстовых данных

На основе метрик можно сказать, что модель плохо предсказывает значения безопасного ПО. Для сравнения попробуем применить этот алгоритм обучения на модели, которая использует числовые данные (вссе параметры кроме hash). Обучим и спрогнозируем целевую переменную malware. Результат на рисунке ниже:

Изображение выглядит как стол

Автоматически созданное описание

Рис. 19 – Значения метрик модели с применением алгоритма RandomForestClassifier с обучением на числовых данных

В итоге мы получаем 100% точность предсказания целевой переменной. Таким образом вредоносные и безопасные ПО были предсказаны верно.

## Алгоритмы глубокого обучения

Переходим к глубокому обучению, в случае с обучением на основе текстовых данных я буду использовать нейронную сеть для классификации текстовых данных. В этой части работы я буду использовать keras из библиотеки tensorflow [13].

Изображение выглядит как текст

Автоматически созданное описание

Рис. 20 – Создание модели нейронной сети для классификации текстовых данных

Описание кода: на вход нейронной сети подаются текстовые данные, которые проходят через слой Embedding, который преобразует каждое слово в вектор фиксированной длины. Затем данные проходят через слой GlobalAveragePooling1D, который усредняет векторы слов и создает единый вектор представления всего текста. Далее, данные проходят через полносвязный слой (Dense) с функцией активации sigmoid, который выдает вероятность принадлежности текста к определенному классу.

После компиляции и обучения модели accuracy составляет 80,5%, что является хорошим результатом, но при этом достаточно большое значение потерь, поэтому модель не является идеальной.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 21 – Оценка модели нейронной сети для классификации текстовых данных

Также обучим модель для работы с числовыми данными. Создам базовую модель нейронной сети с двумя слоями Dense. Первый слой имеет 5 нейронов, входной размер 5 и функцию активации ReLU. Второй слой состоит из одного нейрона, использует функцию активации sigmoid и является выходным слоем модели.

Для компиляции модели используется оптимизатор Adam, функция потерь binary\_crossentropy (для бинарной классификации) и метрики accuracy (для оценки точности классификации).

Потом создаю экземпляр класса KerasClassifier, который будет использоваться для оценки модели и экземпляр класса StratifiedKFold, который предназначен для разбиения данных на части при кросс-валидации. В самом конце оцениваю полученную модель.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 22 – Создание базовой модели нейронной сети для оценки числовых данных

Таким образом вышла хорошо обученная модель, результат чуть ниже, чем при обучении на текстовых данных.

# Сравнение моделей

Сначала рассмотрим результаты, которые вышли при работе с текстовыми данными.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 23 – Сравнение результатов моделей обученных на текстовых данных

Точность всех моделей составляет больше 80 процентов, что является хорошим результатом. При этом, как уже было рассмотрено раньше, модели плохо определяют безопасное ПО. Поэтому, несмотря на хорошую точность, эти модели все еще не идеальны.

Посмотрим на результаты обучения моделей при использование числовых данных.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 24 - Сравнение результатов моделей обученных на числовых данных

В данном случае результат более разношерстный. Модель, обученная с помощью алгоритма MultinomialNB имеет результат, при котором только около половины значений предсказывается верно. Что намного хуже, если сравнивать с той же моделью, обученной на текстовых данных.

Лучше всего себя показала RandomForestClassifier, получив идеальный результат. Для проверки точности модели я провела еще одно обучение этой модели, но уже на полной выборке (40000+ строк данных), результат остался прежним – 100%. Даже при дисбалансе данная модель обучается хорошо.

Нейронная сеть тоже проявила себя хорошо и показала стабильный результат.

Таким образом я бы подвела итог, что при обучение текстовых данных лучше всего использовать MultinominalNB, а для обучения числовых данных отлично подойдет RandomForestClassifier. Нейронные сети показали хорошие результаты, но были немного хуже, чем при обучении моделей с помощью алгоритмов машинного обучения.

# Заключение

При выполнение этой работы мной была разобрана тема выявления вредоносного программного обеспечения по заголовкам файлов. После того, как я разобралась в этой теме с теоретической точки зрения, я перешла к проверке этого метода на практике с помощью алгоритмов машинного и глубокого обучения. По средству обучения нескольких моделей была выявлена наиболее подходящая модель, обученная на алгоритме машинного обучения – MultinominalNB. В ходе выполнения работы я поняла, что при обучении модели на основе текстовых данных есть недостаток: она плохо определяет безопасное ПО, даже если уменьшать выборку данных. Таким образом я решила провести обучение на оставшихся (числовых) данных в датасете, чтобы понять, насколько эффективно обучение по заголовкам файлов. Результат при обучении с помощью алгоритма RandomForestClassifier показал отличный результат, но обучение с помощью других алгоритмов оказалось хуже, чем при обучении на текстовых данных.

Могу сделать вывод, что в зависимости от алгоритма обучения, обучение на текстовых и числовых данных показывают разные результаты, поэтому для получения лучшего результата надо правильно выбрать модель и данные, на основе которых будет проводиться обучение модели.

# Список использованных источников и интернет-ресурсов

1. Ле Д.Ч., Фам М.Х., Динь Ч.З., До Х.Ф. ПРИМЕНЕНИЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ОБНАРУЖЕНИЯ ВРЕДОНОСНЫХ ПРОГРАММ В ОПЕРАЦИОННОЙ СИСТЕМЕ WINDOWS С ПОМОЩЬЮ PE-ЗАГОЛОВКА // Информационно-управляющие системы. 2022. №4 (119). URL: https://cyberleninka.ru/article/n/primenenie-algoritmov-mashinnogo-obucheniya-dlya-obnaruzheniya-vredonosnyh-programm-v-operatsionnoy-sisteme-windows-s-pomoschyu-pe
2. Черкасов Денис Юрьевич, Иванов Вадим Вадимович Машинное обучение // Наука, техника и образование. 2018. №5 (46). URL: https://cyberleninka.ru/article/n/mashinnoe-obuchenie
3. What is Deep Learning? [Электронный ресурс]. URL: <https://machinelearningmastery.com/what-is-deep-learning>
4. Общие сведения о вредоносных программах [Электронный ресурс]. URL: <https://www.microsoft.com/ru-ru/security/business/security-101/what-is-malware>
5. Machine learning – text processing [Электронный ресурс]. URL: <https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958>
6. Основные метрики классификации и регрессии [Электронный ресурс]. URL: <https://webiomed.ru/blog/osnovnye-metriki-zadach-klassifikatsii-v-mashinnom-obuchenii/?ysclid=lhi18kyqdv37244928>
7. Malware Analysis Datasets: PE Section Headers [Электронный ресурс]. URL: <https://www.kaggle.com/datasets/ang3loliveira/malware-analysis-datasets-pe-section-headers?select=pe_section_headers.csv>
8. Seaborn.countplot [Электронный ресурс]. URL: <https://seaborn.pydata.org/generated/seaborn.countplot.html>
9. sklearn.model\_selection.train\_test\_split [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>
10. sklearn.feature\_extraction.text.CountVectorizer [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>
11. sklearn.naive\_bayes.MultinomialNB [Электронный ресурс]. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.MultinomialNB.html#sklearn-naive-bayes-multinomialnb](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html" \l "sklearn-naive-bayes-multinomialnb)
12. sklearn.ensemble.RandomForestClassifier [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
13. Keras [Электронный ресурс]. URL: https://keras.io