ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(ФИНАНСОВЫЙ УНИВЕРСИТЕТ)**

Департамент анализа данных

и машинного обучения

**Дисциплина: «Технологии анализа данных и машинного обучения»**

Направление подготовки: «Прикладная математика и информатика»

Профиль: «Анализ данных и принятие решений в экономике и финансах»

Факультет информационных технологий и анализа больших данных

Форма обучения очная

Учебный 2022/2023 год, 7 семестр

**Курсовая работа на тему:**

«Классификация паролей пользователей по стойкости к взлому»

Выполнил:

студент группы ПИ20-5

Кравченко Д. Д.

Научный руководитель:

Попов Владимир Геннадьевич,

кандидат физико-математических наук, доцент

Москва 2023

[I. ВВЕДЕНИЕ 3](#_Toc134440231)

[II. МАШИННОЕ И ГЛУБОКОЕ ОБУЧЕНИЕ 4](#_Toc134440232)

[III. ОСНОВНЫЕ ПОНЯТИЯ ПРИ КЛАССИФИКАЦИИ ТЕКСТА 5](#_Toc134440233)

[Особенности обработки текста 5](#_Toc134440234)

[Метрики качества 6](#_Toc134440235)

[IV. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ 9](#_Toc134440236)

[Анализ данных 9](#_Toc134440237)

[Определение влияющих факторов 9](#_Toc134440238)

[Описательная статистика 11](#_Toc134440239)

[Групповая описательная статистика 12](#_Toc134440240)

[Количественный анализ 13](#_Toc134440241)

[Итог анализа 16](#_Toc134440242)

[Преобразование данных 18](#_Toc134440243)

[Преобразование датафрэйма 18](#_Toc134440244)

[Разделение выборок 18](#_Toc134440245)

[Преобразование паролей в тензоры 19](#_Toc134440246)

[V. СОЗДАНИЕ АЛГОРИТМОВ И ОБУЧЕНИЕ МОДЕЛЕЙ 21](#_Toc134440247)

[Модели глубокого обучения 21](#_Toc134440248)

[Свёрточная нейронная сеть 21](#_Toc134440249)

[Сеть долгой краткосрочной памяти 25](#_Toc134440250)

[Управляемый рекуррентный блок 26](#_Toc134440251)

[VI. СРАВНЕНИЕ МОДЕЛЕЙ 27](#_Toc134440252)

[VII. ЗАКЛЮЧЕНИЕ 28](#_Toc134440253)

[VIII. СПИСОК ЛИТЕРАТУРЫ 29](#_Toc134440254)

# ВВЕДЕНИЕ

Машинное обучение — это подход к созданию и обучению компьютерных систем, которые могут самостоятельно учиться и делать выводы на основе большого количества данных. Такие системы используют алгоритмы, которые помогают им находить закономерности в данных, делать прогнозы и принимать решения.

В современном мире машинное обучение находится повсюду. Оно применяется в таких разных сферах, как медицина, финансы, сельское хозяйство, образование и многие другие. Но применение машинного обучения только увеличивается, и в будущем мы увидим еще больше инноваций, которые возможны благодаря этой технологии.

Целью данной работы является создать классификатор паролей пользователей по стойкости к взлому, используя несколько моделей глубокого обучения. Работа основана на приведенном датасете - “Password Strength Classifier Dataset” (<https://www.kaggle.com/datasets/bhavikbb/password-strength-classifier-dataset>).

В качестве командного интерпертатора используется Python с интерактивными ноутбуками Jupyter Notebook. При работе с алгоритмами глубокого обучения применяется библиотека PyTorch.

# МАШИННОЕ И ГЛУБОКОЕ ОБУЧЕНИЕ

Часто термины машинное обучение и глубокое обучение путают и используют неправильно.

Машинное обучение (англ. machine learning) является одним из направления искусственного интеллекта. Основной принцип заключается в том, что машины обучаются на данных, которые они получили.

Глубокое обучение является разделом машинного обучения. Алгоритмам машинного обучения необходимы данные и учитель, алгоритмам глубокого обучения учитель не нужен, только подготовленные данные. В глубоком обучении используются некоторые методы машинного обучения.

Алгоритмами глубокого обучения являются нейронные сети, смоделированные по образцу человеческого мозга, они используются в глубоком обучении, но не в машинном. Одним из классических алгоритмов глубокого обучения является – сверточная нейронная сеть (CNN).

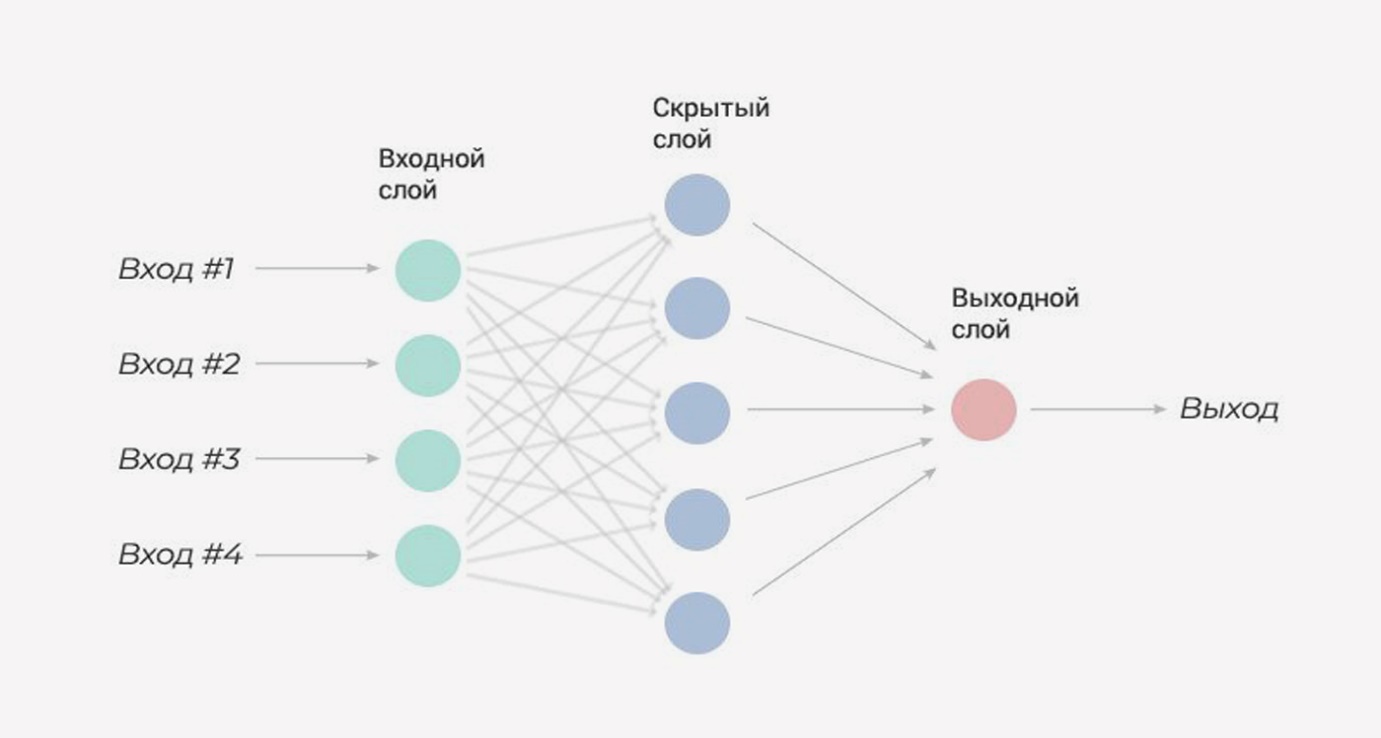


Рис. 1 – Структура нейронной сети

# ОСНОВНЫЕ ПОНЯТИЯ ПРИ КЛАССИФИКАЦИИ ТЕКСТА

## Особенности обработки текста

Любое машинное обучение начинается с подачи входных данных. В данном случаем входными данными будет текст, ведь пароли сами по себе являются текстовым типом данных. Данные считываются компьютером и после обработки алгоритмами выводит эти же данные, в виде полезной для анализа, иллюстрации, принятия решении, информации.

Главное отличие в обработке текста является интеллектуальный анализ текстовых данных. Для человека текстовый документ это набор слов, который несет смысл, а для машины это лишь битовые данные. И задача состоит в том, чтобы машина научилась понимать смысл набора слов. Для этого необходимо применить методы обработки текстовых данных, перед использованием алгоритмов машинного обучения. Сама обработка текста является сложной задачей в машинном обучении.

Обработка естественного языка (NLP) относится к области программного обеспечения, связанной с обработкой и анализом текстовой информации для извлечения смысла и знаний. Рассмотрим основные алгоритмы NLP, которые будут использоваться в данной работе:

* Токенизация ‒ это разделение входных данных на последовательность значимых частей, таких как слово, фрагмент изображения, предложение документа, которые могут быть отображены в векторное пространство.
* Векторизация с помощью пакета слов - набор слов (сокращенно BOW) и векторизация являются полезной техникой NLP для преобразования текстовых документов в числовую форму. Основная проблема с моделями машинного обучения и текстом заключается в том, что нейронные модели нуждаются в числовых данных для работы.

## Метрики качества

Для подсчета точности классификации данных на пароли в нашем классификаторе необходимо ввести ряд метрик. Все метрики будут зависеть от значения в матрице ошибок. Для упрощения рассмотрим матрицу ошибок для задачи бинарной классификации, где есть только два класса, в таком случае матрица путаницы представляет собой матрицу размерностью с целыми числами.

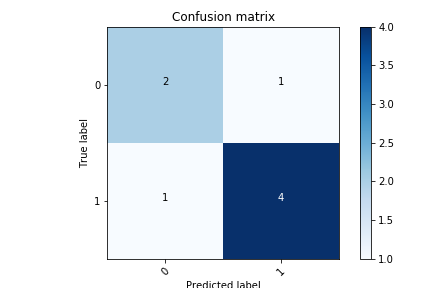


Рис. 2 – Матрица ошибок

Рассмотрим каждый элемент подробнее:

TP (True Positive) – модель правильно утверждает, что объект принадлежит к рассматриваемому классу.

FN (False Negative) - модель неправильно утверждает, что объект не принадлежит к рассматриваемому классу.

FP (False Positive) – модель неправильно утверждает, что объект принадлежит к рассматриваемому классу.

TN (True Negative) - модель правильно утверждает, что объект не принадлежит к рассматриваемому классу.

Наиболее популярной метрикой является accuracy, которая выражает долю ответов, которые были правильно определены системой. Вычисляется как отношение суммы правильно классифицированных объектов (TP и TN) ко всем объектам исходной выборки:

(1)

Однако ориентироваться на данную метку можно только в случае, когда классы сбалансированы и имеют равный размер. Если же классы не сбалансированы, то применяется метрика точности precision.

Точность вычисляется как отношение количества объектов, правильно отнесённых к корректному классу на сумму элементов, которое модель отнесла к этому классу.

Т.е. для положительной точности:

(2)

Для отрицательной точности:

(3)

Альтернативой метрике precision можно считать метрику полноты recall. Полнота – это отношение количества объектов, правильно отнесённых к корректному классу к сумме объектов, относящихся к этому классу.

Т.е. для положительной полноты:

(4)

Для отрицательной полноты:

(5)

Для обобщения результатов полноты и точности обычно используется метрика F-меры. Данная метрика вводит коэффициент β, который определяет вес точности в метрике в целях агрегации и балансировки значений точности и полноты.

(6)

Исходя из формулы, если один из коэффициентов близок к нулю, то общее значение F-меры также будет приближено к нулю.

# ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

## Анализ данных

Так как данная работа на основе приведенного [датасета](https://www.kaggle.com/datasets/bhavikbb/password-strength-classifier-dataset), мне не нужно собирать и фильтровать данные.

Датасет содержит 669 639 строк с паролем и их классом сложности. Пароли распределены на 3 класса: 1-Weak, 2-Medium, 3-Strong.

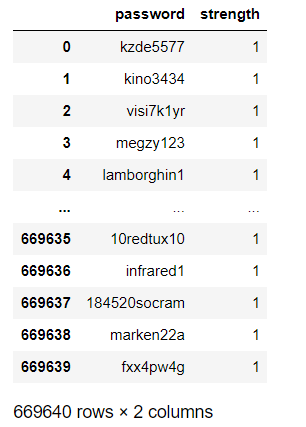


Рис. 3 – Данные датасета

### Определение влияющих факторов

Для начала я найду какие факторы влияют на надежность пароля. Я выявил несколько основных:

* Длина пароля
* Частота использования символов нижнего регистра
* Частота использования символов верхнего регистра
* Частота использования цифровых символов
* Частота использования специальных символов

Для отображения этих факторов, добавлю в датасет колонки с соответствующими факторами для каждого пароля:

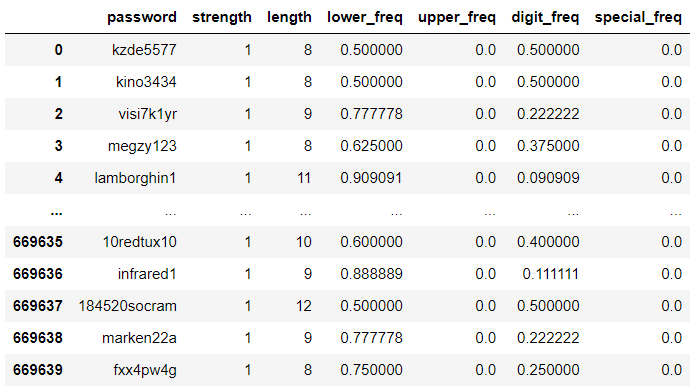


Рис 4. – Результат добавление новых колонок

Немного изучив влияющие факторы, можно заметить, что есть 2 функции, которые влияют на надежность пароля. Первая это закономерная последовательность, например “abc” или “123”. Вторая это последовательность с повторяющимися символами, например “aaa”. Поэтому добавлю столбец, который будет указывать есть ли идентичная последовательность в пароле или нет.

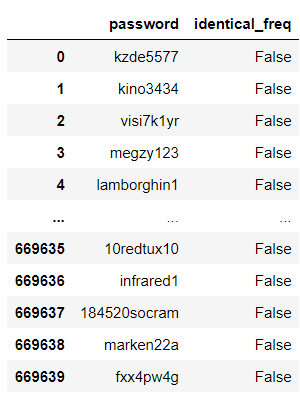


Рис 5. – Результат добавления столбца последовательности

После изучения вышеуказанных 2 функций, можно подумать, что пароль зависит не только от символов в нем, но и от порядка, поэтому я подумал, что, возможно, это зависит от количества двух групп, например, от того, насколько перепутаны эти группы символов. Например, a1b2c надежнее чем abc12. Поэтому я добавил столбец с номером группы последовательности.

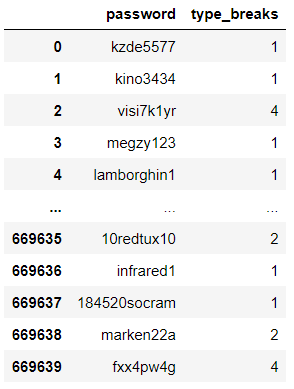


Рис 6. – Результат добавления столбца с группой последовательности

### Описательная статис тика

Чтобы получить смутное представление о распределении данных выведу описательную статистику.

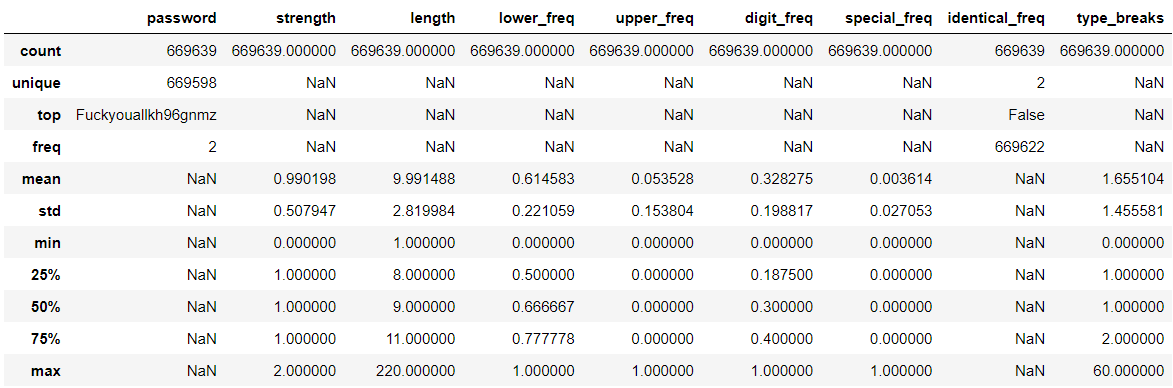


Рис 7. – Цифровая описательная статистика

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис 8. – Графики описательной статистики

Из приведенного выше графиках и таблицы можно получить общий обзор распределения данных по каждому объекту. Выбросы и квартильный диапазон есть, но таблица в значительной степени показала все.

### Групповая описательная статистика

Выведу групповую описательную статистику для понимания распределения данных по каждому классу. Это может помочь понять, как значения функции распределяются между целевыми классами.

Из следующих данных можно сказать, что:

* Чем больше длина, тем выше прочность
* В случае с алфавитом, чем частота выше – не значит лучше. Вероятно, потому, что это не будет надежным паролем, если максимальная часть занята только буквами.
* Пароль обладает большей надежностью, если типы символов распределены в приличных пропорциях.

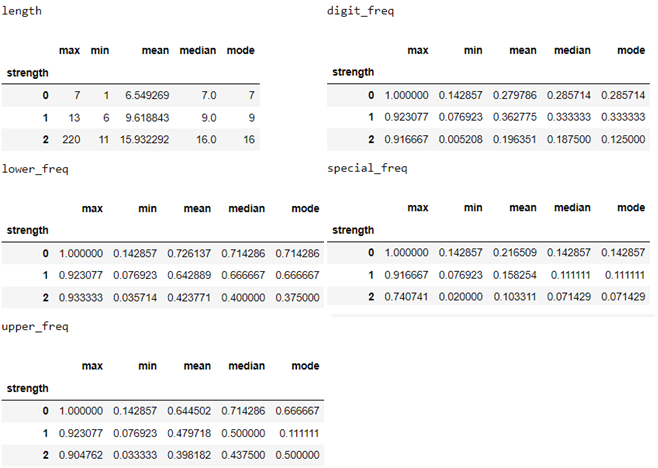


Рис 9. – Групповая статистика

### Количественный анализ

Далее выведу количественный анализ для понимания распределения и количества объектов в классах.

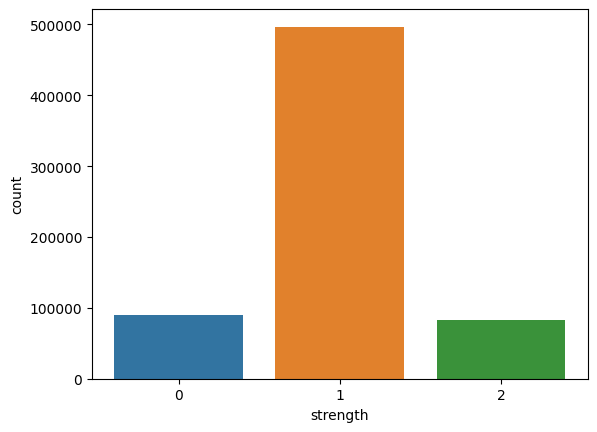


Рис. 10 – Распределение количества элементов по классам

Как можно заметить, класс 1 (Medium) имеет в несколько раз больше элементов чем остальные. Это довольно большой дисбаланс между целевыми классами. Дисбаланс классов довольно сильно влияет на производительность модели, поэтому, если обучить модель, не решая эту проблему, в конечном итоге можно получить модель, которая смещена в сторону класса большинства, поэтому в дальнейшем я удалю случайную выборку из класса 1, чтобы он не доминировал над классами меньшинства в наборе данных.

Далее я проведу двумерный анализ, для понимания отношения объектов друг к другу.

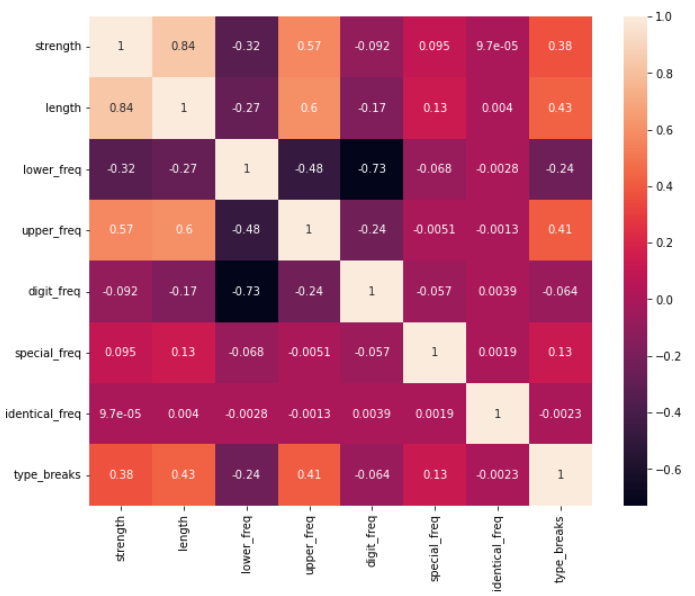


Рис 11. – Двумерный анализ, тепловая карта

Исходя из графика можно сказать следующие:

* Длина и сила демонстрируют высокую корреляцию, что неудивительно.
* identitical\_freq наименее коррелирует с силой, вероятно, будет удален.
* lower\_freq, upper\_freq и type\_breaks немного коррелируют с силой.

Чтобы убедится в том, что identitical\_freq следует удалить, стоит посмотреть на его распределение:

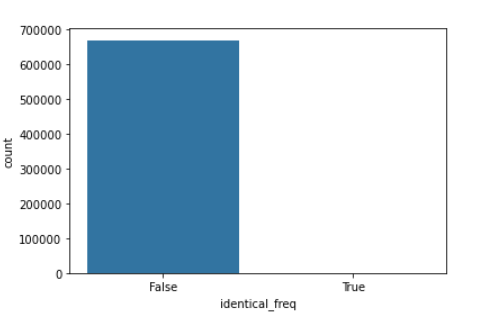


Рис. 12 – Распределение идентичной последовательности

### Итог анализа

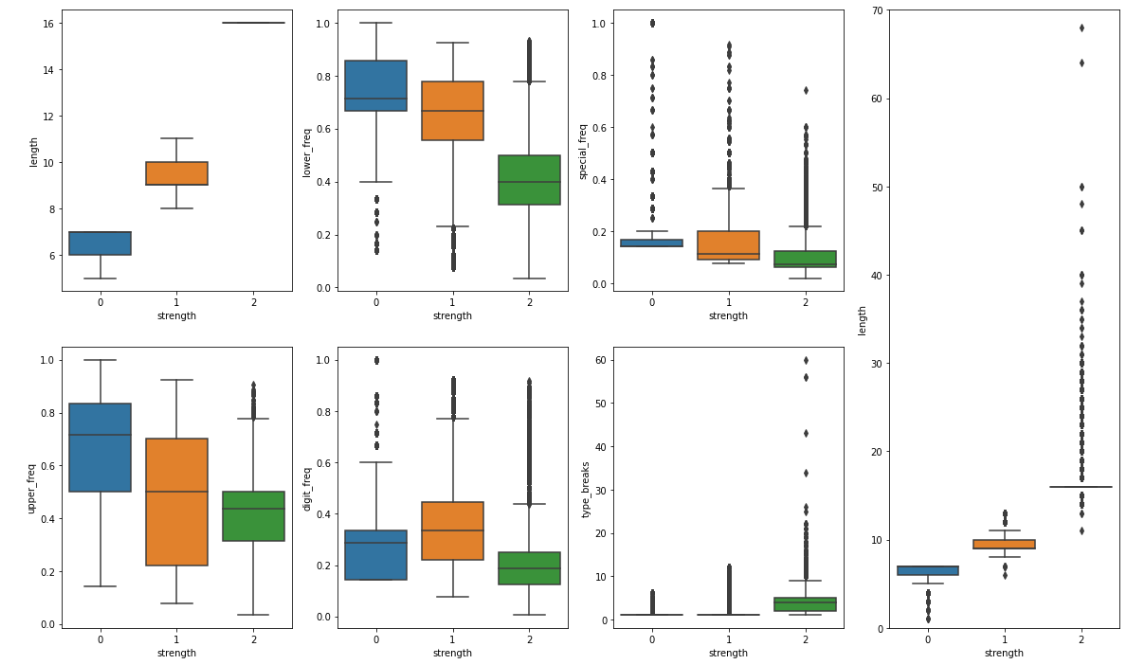


Рис. 13 – Двумерный анализ, ящик с усами

Подытожив, можно сказать, что прочность может быть определена исключительно на основе длины, поскольку она обеспечивает идеальный раскол. Тогда это была бы лучшая функция, но логически это звучит некорректно, потому что, когда я запустил 2 пароля одинаковой длины на сайте проверки пароля, они выдали 2 разных уровня. Локигу, которую они используют, отличается, и для этого длина данных является наилучшей, но в более крупном масштабе все может быть сомнительно. Но да, более длинные пароли обладают большей надежностью, поэтому, учитывая обстоятельства, мы могли бы даже просто использовать классификацию на основе длины или использовать ее в качестве основной функции.

Более высокая частота использования строчных букв наблюдается в паролях с низкой надежностью. Для паролей с более высокой надежностью частота тоже может быть высокой, но это, вероятно, эффект длины.

В upper\_freq есть тенденция, но не такая сильная, как в length или lower\_freq.

В digit\_freq есть разделение большинства значений классов 1 и 2, но для классов 0 и 1 существует перекрытие, так что там особо нечего сказать.

Пароли с более высокой надежностью содержат больше групп последовательностей.

## Преобразование данных

### Преобразование датафрэйма

Для начала, конечно, необходимо удалить все пустые ячейки в таблице. Далее я привел столбец с паролями в типу str и удалил все пробелы из пароля с помощью функции strip(), так как на сегодняшний день, принято не использовать пробелы в паролях.



Рис. 14 – Преобразование датафрэйма

Как я упоминал выше, следует удалить выборку из класса 1 до минимального количества объектов в других классах, тем самым решая проблему дисбаланса классов. В итоге получается датасет размером 255876, по 80000 объектов в каждом классе.

Изображение выглядит как текст

Автоматически созданное описание

Рис 15. – Удаление выборки из класса 1

### Разделение выборок

Необходимо разделить датасет на обучающую и тестовую выборку. Выборки делим в соотношении 80% на 20% с помощью метода scikit-learn train\_test\_split.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

В аргументе X передаём numpy.array с паролями, в Y – номера классов, которые характеризуют надежность пароля, в test\_size – размер проверочной выборки, random\_state=1 - то псевдослучайные величины при генерации будут иметь одни и те же значения при каждом вызове. При указании аргумента stratify метод производит разбиение таким образом, что доля значений в полученной выборке будет такой же, как и доля значений, предоставленных параметру stratify.

После обработки данного метода интерпретатором, получим 4 выборки:

* X\_train – пароли обучающей выборки
* X\_test – пароли тестовой выборки
* y\_train – классы, характеризующие надежность пароля обучающей выборки
* y\_test – классы, характеризующие надежность пароля тестовой выборки

### Преобразование паролей в тензоры

Для начала работы с обучением необходимо конвертировать все пароли в тензоры PyTorch. В нашем случае нужно каждый пароль преобразовать в тензор, с длиной равной максимальной длине пароля в датасете, который будет иметь вид последовательности индексов символов в словаре и 0 до конца тензора.

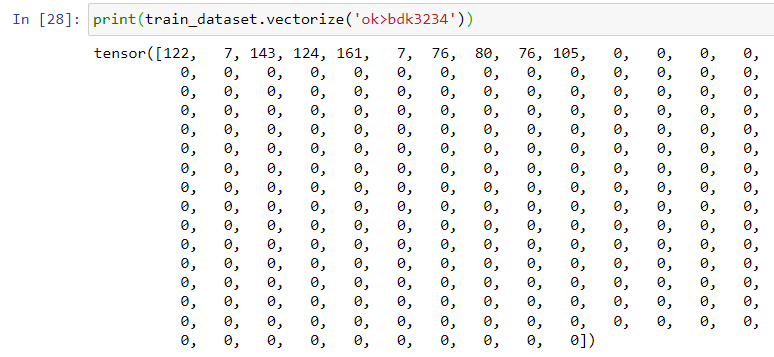


Рис. 16 – Пароль в виде тензора

# СОЗДАНИЕ АЛГОРИТМОВ И ОБУЧЕНИЕ МОДЕЛЕЙ

## Модели глубокого обучения

### Свёрточная нейронная сеть

Сверточные нейронные сети (convolutional neural networks, CNN) — класс искусственных нейронных сетей, основная идея которых состоит в том, чтобы переиспользовать одни и те же части нейронной сети для работы с разными локальными участками входов. Идея сверточных сетей основана на результатах исследований зрительной коры головного мозга.

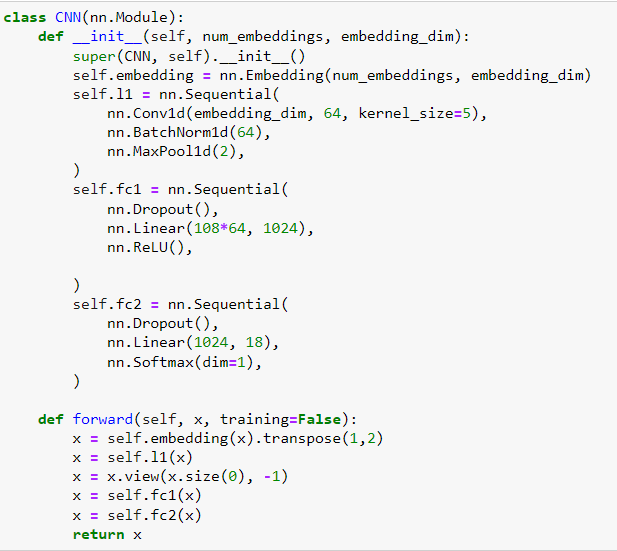
CNN достигли высочайшей производительности в широком спектре задач распознавания изображений, включая классификацию объектов, обнаружение объектов и сегментацию изображений. Они широко используются в компьютерном зрении, обработке изображений и других смежных областях и были применены в широком спектре приложений, включая беспилотные автомобили, медицинскую визуализацию и системы безопасности.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 17 – Пример работы свертки в CNN

Реализация выглядит следующим образом:

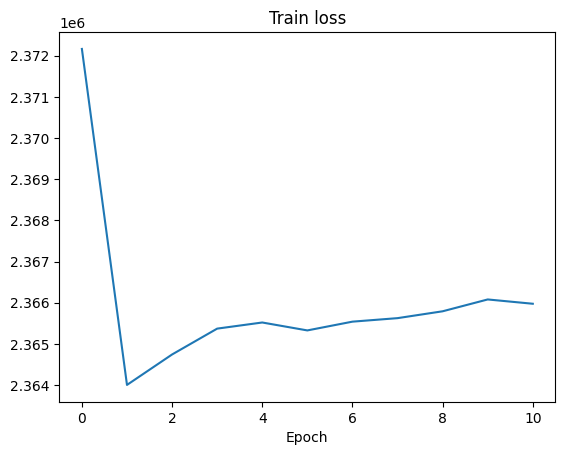


На вход модели мы подаем num\_embeddings=vocab.vocab\_len - размер словаря вложений, в нашем случае это длина словаря символов (178), и embedding\_dim=100 - размер каждого вектора вложения. Данные параметры служат для метода torch.nn.Embedding - простая таблица поиска, в которой хранятся вложения с фиксированным словарем и размером, входными данными для модуля является список индексов, а выходными - соответствующие вложения слов. Этот метод может оказать значительно влияние на производительность модели, насколько быстро или легко сеть сходится к хорошему решению.

Изображение выглядит как текст, стол

Автоматически созданное описание

Рис. 18 – Процесс обучения CNN



Изображение выглядит как текст, График, снимок экрана, линия

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, График, линия

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Рис. 19 – Результаты обучения CNN

Точность классификации составляет 100%, это значит модель идеально обучилась. Время обучения составило 136 минут.

Рекуррентная нейронная сеть

Рекуррентная нейронная сеть (RNN) — это класс искусственных нейронных сетей, в которых узел может получать входы не только от других узлов и текущих входных данных но и выходы узлов, полученные при рассмотрении предыдущих входных данных последовательности.

Если схематично, слой RNN использует цикл для итерации по упорядоченной по времени последовательности, сохраняя при этом во внутреннем состоянии, информацию о шагах, которые он уже прошел.

Перед обучением данной модели я создал подбор гиперпараметров и обучил модель с каждым из следующих гиперпараметров:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

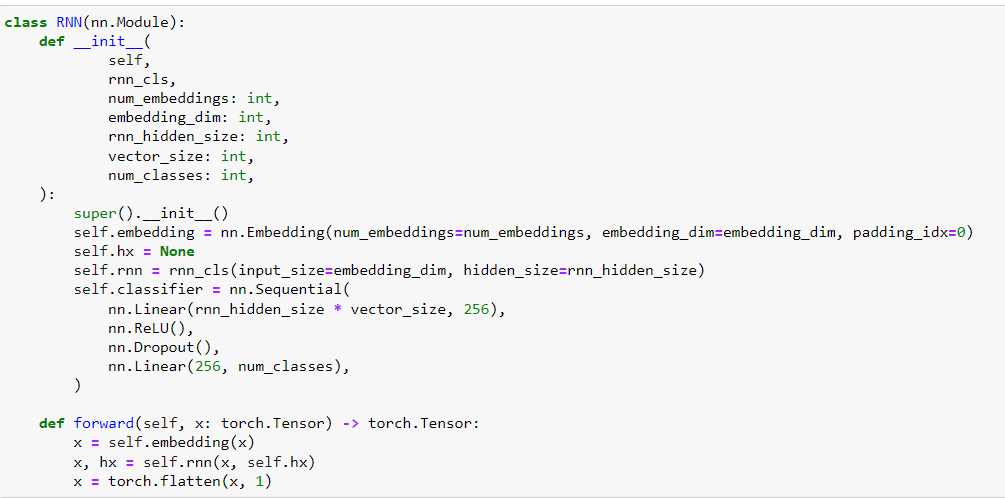
Автоматически созданное описание

Результаты выглядят так:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Реализация выглядит так:



Результат:

Изображение выглядит как текст, стол

Автоматически созданное описание

### Сеть долгой краткосрочной памяти

LSTM (Long Short-Term Memory) — это особый вид рекуррентных нейронных сетей, которые способны изучать долгосрочные зависимости. Обычные рекуррентные сети очень плохо справляются с ситуациями, когда нужно что-то «запомнить» надолго: влияние скрытого состояния или входа с шага t на последующие состояния рекуррентной сети экспоненциально затухает.

Вместо одного-единственного числа, на которое влияют все последующие состояния, используется специального вида ячейка моделирующая "долгую память" LSTM моделирует процессы записи и чтения из этой "ячейки памяти" у ячейки не один набор весов, как у обычного нейрона, а сразу несколько

Так как это вид рекуррентной нейронной сети, я буду использовать тот же класс RNN, изменив параметр rnn\_cls=nn.LSTM, что создаст слой LSTM в модели.

Изображение выглядит как текст

Автоматически созданное описание

Результат:

Изображение выглядит как стол

Автоматически созданное описание

### Управляемый рекуррентный блок

GRU (Gated Recurrent Unit) – это также вид рекуррентных нейронных сетей, направленный на решение проблемы исчезающего градиента, которая поставляется со стандартной рекуррентной нейронной сетью. Чтобы решить эту проблему GRU использует, так называемые, элементы обновления и сброса. По сути, это два вектора, которые решают, какую информацию следует передавать на вывод. Особенность их в том, что их можно обучить хранить информацию из давних времен, не стирая ее с течением времени или удаляя информацию, которая не имеет отношения к прогнозированию.

Реализация точно такая же как у модели LSTM:

Изображение выглядит как текст

Автоматически созданное описание

Результат:

Изображение выглядит как стол

Автоматически созданное описание

# СРАВНЕНИЕ МОДЕЛЕЙ

|  |  |  |
| --- | --- | --- |
|  | **Точность** | **Время (мин)** |
| CNN | 100% | 135.9 |
| RNN | 100% | 76.5 |
| LSTM | 100% | 136.8 |
| GRU | 100% | 128.8 |

Все модели показали максимальный результат, но по времени обучения лучше всего модель RNN, почти в 1.5 раза меньше. Поэтому это будет лучшей моделью.

Изображение выглядит как текст

Автоматически созданное описание

Рис.

# ЗАКЛЮЧЕНИЕ

В заключение хотелось бы сказать, что в процессе работы датасет был отфильтрован, в целях повышения точности классификации данных, описаны 4 модели глубокого обучения для классификации текстовых данных. Все модели максимально обучились и показали идеальный результат точности.

Наиболее предпочтительной моделью является .., так как точность у всех моделей одинаковая, а время обучения у данной модели меньше чем у остальных.

Если говорить об улучшении модели, то могу отметить, что для применения в реальной жизни, можно добавить учитывание паролей, которые состоят из электронной почты, сайта или других известных названий, как слабые пароли, ведь в реально жизни злоумышленники чаще всего взламывают именно такие пароли.

# СПИСОК ЛИТЕРАТУРЫ

1. М.В. Коротеев. Учебное пособие по дисциплине “Анализ данных и машинное обучение” - 2018.
2. Маккини У. Python и анализ данных / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 540 с.: ил.
3. Вейдман Сет. Глубокое обучение: легкая разработка проектов на Python. — СПб.: Питер, 2021. — 272 с.: ил. — (Серия «Бестселлеры O’Reilly»).
4. Плас Дж. Вандер Python для сложных задач: наука о данных и машинное обучение. – СПб.: Питер, 2018. – 576 с.
5. Password Strength Classifier Dataset - [Электронный ресурс]. - <https://www.kaggle.com/datasets/bhavikbb/password-strength-classifier-dataset>
6. Кластеризация и классификация больших Текстовых данных с помощью машинного обучения на Java. Статья #1 — Теория - [Электронный ресурс]. - <https://habr.com/ru/articles/526984/>
7. Как улучшить точность ML-модели используя разведочный анализ - [Электронный ресурс]. - <https://habr.com/ru/articles/719206/>
8. Репозиторий курсового проекта на GitHub - [Электронный ресурс]. - <https://github.com/Kravchenko-Daniil/University/tree/main/CoursePaper-course%203>