

СОФИЙСКИ УНИВЕРСИТЕТ "СВ. КЛИМЕНТ ОХРИДСКИ"

ФАКУЛТЕТ МАТЕМАТИКА И ИНФОРМАТИКА

Катедра „Компютърна информатика“

ИЗПОЛЗВАНЕ НА МЕТОДИ ОТ ИЗКУСТВЕНИЯ ИНТЕЛЕКТ ЗА БЪРЗО  
ОПРЕДЕЛЯНЕ НА СХОДСТВОТО В ДАННИ ОТ СЕКВЕНИРАНЕ

ДИПЛОМНА РАБОТА

Дипломант: **Мария Влахова**

Специалност: Информатика

МП: Изкуствен интелект

Фак. № 25836

Дипломен ръководител:

**Доц. д-р Валерия Симеонова**

Катедра:

Информационни технологии

2021



## СЪДЪРЖАНИЕ

1	Увод .....	2
2	Преглед и характеристика на проблемната област .....	3
2.1	1. Намиране на подобност на последователността: .....	5
2.2	2. Клъстерен анализ на база намереното сходство. ....	7
3	Описание на използваните данни .....	8
4	Избор на библиотеки за работа с биологични данни .....	9
5	Сравнение между различните методи за клъстеризация .....	10
5.1	Mean-shift (средно отместване) .....	10
5.2	Генетичен алгоритъм .....	12
5.3	K-means .....	15
5.4	Affinity Propagation .....	16
5.5	Deep clustering .....	17
6	Анализ и избор на техники от машинно самообучение .....	19
6.1	Изследване с техники на дълбокото клъстеризиране .....	19
6.2	Изследване с техники на клъстеризиране чрез използване на инженеринг на характеристиките (clustering with feature engineering) .....	20
7	Избор и оптимизация на финален алгоритъм .....	34
8	Измерване на резултатите на база тестови данни .....	36
9	Анализ, визуализация, описание и документиране на получените резултати от гледна точка на информатиката и биомедицината. ....	40
10	Заключение .....	41
11	Списък на фигурите в текста .....	42
12	Използвани литературни източници в бележки в края на документа .....	42

Сравненията на генома и метагенома, базирани на големи количества данни от последователността, представляват значителни предизвикателства за съвременните генетични анализи. Един бърз алгоритъм за намиране на сходство, дори и не изключително точен, може да ни даде достатъчно знания на база на които да се вземе решение дали да извърши по обстойно проучване по-темата или не. Подходите да се случи това са:

- подходи базирани на подравняване
- подходи не базирани на подравняване

Подходите базирани на подравняване са много по точни, но време и ресурсоемки. Подходите които не са базирани на подравняване са по-бързи, но и по-неточни. Те са един от основните насоки в която се работи от дейта специалистите, занимаващи се със сравнение на генома.

Най-често използваните методи са:

- базирани на броене на думи
- базирани на най-дълъг подстринг
- базирани на изчисляване на дистанция
- базирани на трансформация на Фурие
- базирани на теорията за Хаоса

В този проект аз ще представя моя вариант на алгоритъм за сравнение без подравняване. Той се базира на методи за клъстеризация от машинното самообучение и обработка на ДНК данните с методи без подравняване(броене на думи, определяне на дистанция между отрязъци от ДНК-то и най-дълъг подстринг). За бейзлайн ще ползвам същите методи за клъстеризация от машинното самообучение, но ДНК данните ще бъдат обработени с методи за подравняване. Това изследване трябва да бъде извършено с използването на минимални технически ресурси, тъй като идеята е да представим бързо и евтино решение на проблема.

Изследването се състои от два ключови термина: клъстерен анализ и секвениране от следващо поколение. За да се направи пълноценен преглед и характеристика на проблемната област трябва първоначално да уточним понятията.

Клъстерният анализ<sup>i</sup> е изследователски анализ, който се опитва да идентифицира структури в данните. Клъстерният анализ се нарича още сегментационен анализ или анализ на таксономията. По-конкретно, той се опитва да идентифицира хомогенни групи случаи, ако групирането не е известно преди това. Тъй като е изследователски, той не прави никаква разлика между зависими и независими променливи. Клъстерният анализ често се използва заедно с други анализи (като дискриминантния анализ). Изследователят трябва да може да интерпретира клъстерния анализ въз основа на разбирането им за данните, за да определи дали резултатите, получени от анализа, всъщност са значими. Типичните изследователски въпроси, на които отговаря клъстерният анализ в Биологията са както следва:

- Каква е таксономията на видовете?
- Какви са различните атрибути, функции /характеристики и други на различните видове?

Клъстерният анализ може да групира тези наблюдения в поредица от клъстери и да помогне за изграждането на таксономия на групи и подгрупи на подобни растения.

Секвениране от следващо поколение<sup>ii</sup> (Next Generation Sequencing)

Масивната технология за паралелно секвениране, известна като секвениране от следващо поколение (NGS), революционизира биологичните науки. Със своята свръхвисока производителност, мащабируемост и скорост, NGS позволява на изследователите да изпълняват голямо разнообразие от приложения и да изучават биологични системи на ниво, което никога преди не е било възможно. Днешните сложни геномни изследователски въпроси изискват по-задълбочена информация отвъд възможностите на традиционните технологии за секвениране на ДНК. Последователността от следващо поколение запълни тази празнина и се превърна в ежедневен инструмент за изследване на тези въпроси.

Технологията NGS коренно промени видовете въпроси, които учените могат да задават и да отговорят. Иновативните опции за подготовка на проби и анализ на данни позволяват широк спектър от приложения. Например NGS позволява на изследователите:

- Бързо секвениране на цели геноми
- Дълбоко секвениране на целеви региони
- Използване на последователността на РНК (RNA-Seq), за да се открият нови варианти на РНК и места за снаждане или да се определи количествено mRNA за анализ на генната експресия

- Анализиране на епигенетични фактори като метилиране на ДНК в целия геном и взаимодействия между ДНК и протеини
- Проучване на човешкия микробиом
- Идентифициране на нови патогени

Съществуват редица различни NGS платформи<sup>iii</sup>, използващи различни технологии за последователност, подробно обсъждане на които е извън обхвата на тази статия. Всички NGS платформи обаче паралелно извършват секвениране на милиони малки фрагменти от ДНК. Използват се биоинформатични анализи за разделяне на тези фрагменти чрез картографиране на отделните показания към човешкия референтен геном. Всяка от трите милиарда бази в човешкия геном е секвенирана многократно, осигурявайки висока дълбочина за предоставяне на точни данни и прозрение за неочаквани ДНК вариации (фигура 1). NGS може да се използва за секвениране на цели геноми или да се ограничава до специфични области на интерес, включително всички 22 000 кодиращи гени (цял екзон) или малък брой отделни гени.



Фигура 1: Пример за следващо поколение секвениране (NGS) сурови данни-BRAF V600E мутация при меланом. Мутацията е открита през 2002 г. като част от няколко годишни усилия за дефиниране на соматични мутации при човешки рак, използвайки секвениране

Клъстерният анализ при генетични секвенции се използва за да се определят повтарящи се или сходни сектори от генома на 1 организъм или намиране на сходни сектори между различни организми.

В тази дипломна работа решаваме втората задача. За да го направим разделяме задачата на 2 подчасти:

## 2.1 1. НАМИРАНЕ НА ПОДОБНОСТ НА ПОСЛЕДОВАТЕЛНОСТТА:

Подобността на последователностите<sup>iv</sup> е концепция от изчислителната биология и компютърни науки. Подобността на последователността е число, което показва доколко две последователности са сходни. Подобността на последователността понякога, но не винаги, се определя чрез разстоянието на последователността: колкото по-малко е разстоянието, толкова по-сходни са последователностите 1. Например, следните две ДНК последователности са силно сходни (разликите са подчертани в червено) :

```
GTCCTCATAACTCTCTAG
GTCGTCATAAC CTCTCTAG
```

За тяхното сходство свидетелства ниското им разстояние на Левенщайн от 2 - необходими са само две операции за редактиране (една подмяна и едно заличаване), за да се трансформира първата последователност във втората 2. Но има и други показатели за разстоянието в последователностите и има и показатели за сходство, които не могат да се изразят като разстояния. Следователно, докато сходството на последователността винаги е число, определено въз основа на две последователности, спецификата на начина на изчисляване на това число може да варира. Понякога резултатът за сходство се изразява като процент, а именно „процент сходство“ или „процент идентичност“. Процентната идентичност обикновено се отнася до съотношението между броя на съответстващите остатъци към общата дължина на подравняването (виж по-долу), напр.  $18/20 = 90\%$  в горния пример. Вижте също Li, 2018. Процентът на сходство отчита „подобни“ остатъци (обикновено аминокиселини) в допълнение към идентичните. Сходството между аминокиселините може да се определи или чрез техните химични свойства, или въз основа на РАМ матрица.

Подравняването е друга концепция от изчислителната биология. Това е просто всеки начин да се подредят две последователности една под друга,. Вече беше показано примерно подравняване по-горе. Ето го отново, този път с пропуските (заличаванията), посочени с тире (-), както е обичайно за подравняването на последователността:

GTCCTCATAACTCTCTCTAG  
GTCGTCATAAC-CTCTCTAG

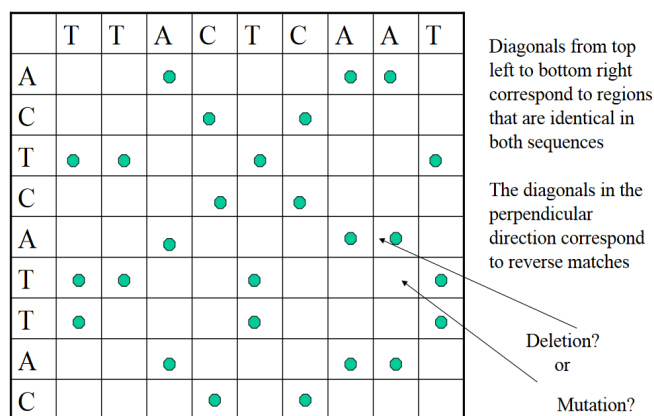
Бихте си помислили, че това е подравняването за тези два низа, но според дефиницията следното също е валидно подравняване:

GTCCTCAT-AACTCTCTCTAG  
GTCGTCATAA-C-CTCTCTAG

Алгоритъмът за подравняване обикновено има функция за оценяване, която присвоява на всяко подравняване цифров резултат, показващ колко добро е подравняването, и се опитва да намери най-доброто подравняване според неговата функция за оценяване.

Методите за подравняване са:

1) Dot Plot Matrix<sup>v</sup> - Най-простият метод е точковата графика. Едната последователност се изписва хоризонтално, а другата се изписва вертикално, по горната и страничната страна на мрежата  $m \times n$ , където  $m$  и  $n$  са дължините на двете последователности. Точка се поставя в клетка в мрежата навсякъде, където двете последователности съвпадат.



Фигура 2: Илюстрация на Dot Plot Matrix

Диагонална линия в мрежата визуално показва къде двете последователности имат идентичност на последователността. Сравненията на точки с последователност от нуклеинова киселина ще покажат много високо ниво на фона (25% шанс за случайно съвпадение), така че параметрите трябва да бъдат модифицирани, за да поставят точка само ако има почти перфектно съвпадение по плъзгащ се „прозорец“ от 10 или повече последователни нуклеотиди. Илюстрацията е взета от тук<sup>vi</sup>.

2) global alignment<sup>vii</sup> - Алгоритъмът, публикуван от Needleman и Wunsch през 1970 г. за подравняване на две протеинови последователности, е първото приложение на динамичното програмиране към биологичния анализ на последователността. Алгоритъмът на Needleman-Wunsch намира най-добре постигнатото глобално

подравняване между две последователности. Публикация в блог на Chetan има много ясно обяснение как става това. Глобалните подравнения са най-полезни, когато двете последователности, които се сравняват, са с еднаква дължина и не са твърде различни.

3) local alignment <sup>viii</sup>-Реалният живот често е сложен и ние наблюдаваме, че гените и протеините, които те кодират, са претърпели разместване на екзон, рекомбинация, вмъкване, делеции и дори сливания. Много протеини проявяват модулна архитектура. При търсене в бази данни за подобни последователности е полезно да се намерят последователности, които имат подобни домейни или функционални мотиви. Smith & Waterman (1981) публикува приложение на динамично програмиране за намиране на оптимални локални подравнения. Алгоритъмът е подобен на Needleman-Wunsch, но отрицателните стойности на клетките се нулират и процедурите за трасекбек започват от клетката с най-висок резултат, навсякъде в матрицата и завършват, когато пътят срещне клетка със стойност нула.

4) alignment-free -Това са методи без подравняване, повечето от които ще намерите подробно описани по долу в дипломната работа. Такива са:

- *phylogenetic analysis*
- *genomic signal processing (GSP)*
- *методи базирани на броене на думи*
- *методи базирани на изчисляване на дистанция.*
- *методи базирани на теорията за Хаоса*
- *методи базирани на Fuzzy integral similarity*
- *Center Star Method*

## 2.2 2. КЛЪСТЕРЕН АНАЛИЗ НА БАЗА НАМЕРЕНОТО СХОДСТВО.

Сравнението на последователностите се превърна в много важен инструмент в съвременната молекулярна биология. Всъщност в биомолекулните последователности високото сходство обикновено предполага значително функционално или структурно сходство. Традиционните подходи използват техники, които се базират на подравняване на последователността, способни да измерват разликите в нивото на символите. Въпреки това, неотдавнашното развитие на цялостната технология за секвениране поражда нужда от мерки за сходство, способни да уловят пренарежданията, включващи големи сегменти, съдържащи се в последователностите.

Клъстерният анализ може да играе ключова роля за постигането на тези цели. При него трябва да се обърне възможност и дали дадения алгоритъм има възможност за работа с големи масиви от данни и дали има възможност за скалиране.



### 3 ОПИСАНИЕ НА ИЗПОЛЗВАНИТЕ ДАННИ

Всички данни са извлечени от Националната база за биологични данни на САЩ NCBI, която се явява едно от трите световни хранилища за секвенционни данни, наред с европейската база EBI и японския архив, като трите бази са в непрекъснат синхрон.

Основните данни, с които е работено са, като в скоби е посочен референтният номер към NCBI:

- SARS (MW369393.1)
- InfluenzaA (EU478839.1)
- InfluenzaB ( NC\_002205.1)
- HIV (NC\_001802.1)
- Covid19 (NC\_045512.2)

В процеса на работа бяха генерирани и Синтетични данни въз основа на оригиналните. За да имаме 100 % съвпадение беше изчислена и добавена към синтетичните данни двойката HIV-HIV.

Синтетичните данни плюс стандартните данните бяха смятани на няколко машини:

	Машина 1	Машина 2	Машина 3
CPU	Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz 6 cores	Intel(R) Core(TM) i5-6600K CPU @ 3.50GHz 3.50 GHz	Intel(R) Core(TM) i5-3320K CPU @ 3.50GHz 3.50 GHz 2cores
RAM	16 GB	16 GB	8 GB
Drive	953 GB	953 GB	810 GB

Изчисляването на двойките (всеки със всеки) + тренирането на регресорите отне към 2 седмици паралелно на 3 те машини. За регресорите беше предложено трансферно обучение, тъй като беше невъзможно да се обработят данните на един компютър. Освен това с цел да се предпазя от евентуална загуба на данни (тъй като тези компютри се използваха и за други цели и стигаха до прегряване и изключване) данните се записваха в множество .csv файлове и общия им обем излезна към терабайт и половина (нека се има предвид, че се смяташе всеки чънк от едната секвенция с всеки чънк от другата секвенция и всички сметки се правеха паралелно).

Програмните средства, които ще бъдат използвани за имплементацията са:

Програмно средство	Предназначение
<b>Python</b>	Това е програмният език, който е избран за имплементация, защото той е най-предпочитания за ML проекти, поради множеството библиотеки и бързото изчисляване на математически задачи благодарение на NUMPY.
<a href="#"><u>Biopython</u></a>	В програмната имплементация е използван за подравнявания и четене на FASTA файлове. Това е стандартна библиотека за работа с генетични данни.
<a href="#"><u>Pyeasyga</u></a>	Това е библиотеката, която е използвана за генетичните алгоритми
<a href="#"><u>Sklearn</u></a>	Това е библиотеката, която се използва за моделите за машинно самообучение.
<a href="#"><u>ClusterOmega</u></a>	Това е софтуер за подравняване на множество последователности, която използва засадени дървета за ръководство и техники за HMM профил, за да генерира глобално подравняване между две или повече последователности. Чрез него ще бъдат намерени секторите на подобие, които ще използваме, за да оценим колко добре се справя нашата програма от гледна точка на биологията.
<a href="#"><u>Blast</u></a>	BLAST открива области на сходство между биологичните последователности. Програмата сравнява нуклеотидни или протеинови последователности с бази данни на секвенции и изчислява статистическата значимост.
<a href="#"><u>joblib</u></a>	Библиотека, която ще използваме за трансферирано обучение на регресорите.

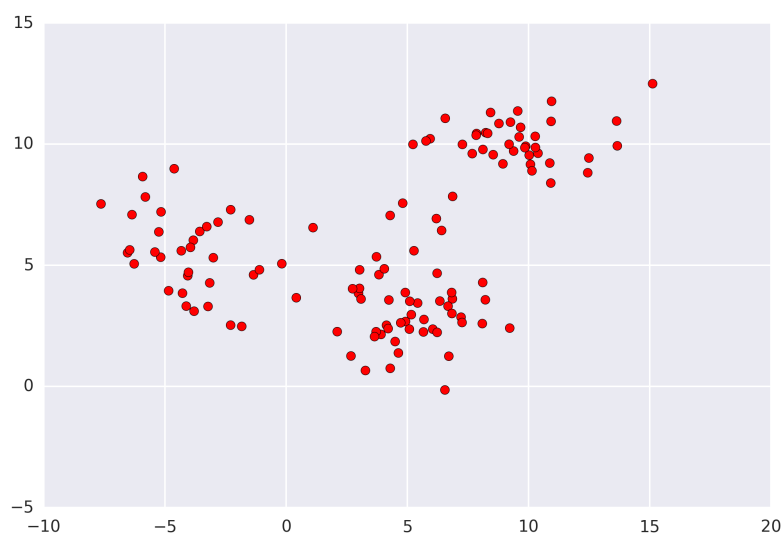
## 5 СРАВНЕНИЕ МЕЖДУ РАЗЛИЧНИТЕ МЕТОДИ ЗА КЛЪСТЕРИЗАЦИЯ

Моделите, които са използвани в програмната реализация са K-means, K-means++, Affinity Propagation и Mean-shift.

За оценка на моделите се използва метриката `silhouette_score`, както и за оценка на бързодействието на моделите.

### 5.1 MEAN-SHIFT<sup>ix</sup>(СРЕДНО ОТМЕСТВАНЕ)

Първата стъпка при прилагане на средно отместване (и всички алгоритми за клъстериране) представя данните по математически начин. За средно изместване, това означава представяне на вашите данни като точки, като дадените по-долу (виж Фигура 3).

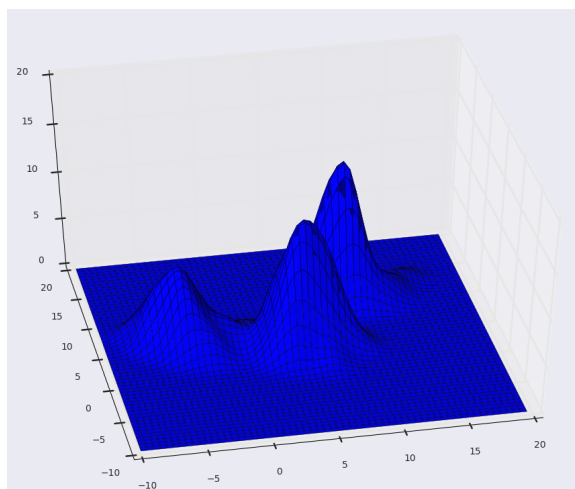


Фигура 3: mean\_shift\_points

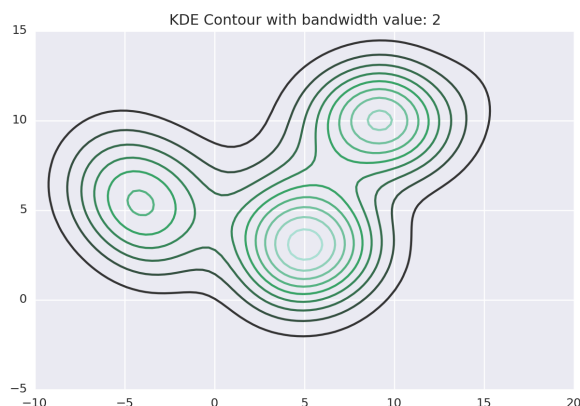
Средната промяна се основава на концепцията за оценка на плътността на ядрото (KDE). KDE е метод за оценка на основната дистрибуция (наричана също функция на вероятностната плътност) за набор от данни.

Той работи чрез поставяне на ядро във всяка точка от набора от данни. Ядрото е математическа дума за функцията за претегляне. Има много различни видове ядра, но най-популярната е ядрото на Гаус. Добавят се всички отделни ядра до генерира вероятностна повърхност (например функция на плътност). В зависимост от използвания параметър на честотната лента на ядрото, резултантната функция на плътността ще варира. По-долу е показана повърхността на KDE за точките по-горе, като се използва ядрото на Гаус с честотна лента на ядрото 2. Първото изображение е

повърхностно (виж Фигура 4), а второто е контурно изображение на повърхността (виж Фигура 5).



Фигура 4: KDE изображение на повърхнината



Фигура 5: KDE контурно изображение

И така, как се вписва mean shift (средното отместване) в картината? То използва идеята на KDE за постепенно изкачване до най-близкия връх на повърхността на KDE. Това става чрез итеративно преместване на всяка точка нагоре, докато достигне връх. В зависимост от използваната пропускателна способност на ядрото, повърхността на KDE (и крайната клъстеризация) ще бъде различна. Като краен случай си представете, че се използват изключително високи и малки ядра (например малка честотна лента на ядрото), тогава резултантната повърхност на KDE ще има пик за всяка точка. Това ще доведе до поставянето на всяка точка в свой собствен клъстер. От друга страна ако се използва ядро с голяма пропускателна способност, това ще доведе до широка гладка повърхност на KDE с един връх, към който ще се изкачат всички точки, което ще доведе до само един клъстер. Ядрата между тези две крайности ще доведат до по-хубави клъстери.

### *The Mean Shift Algorithm*

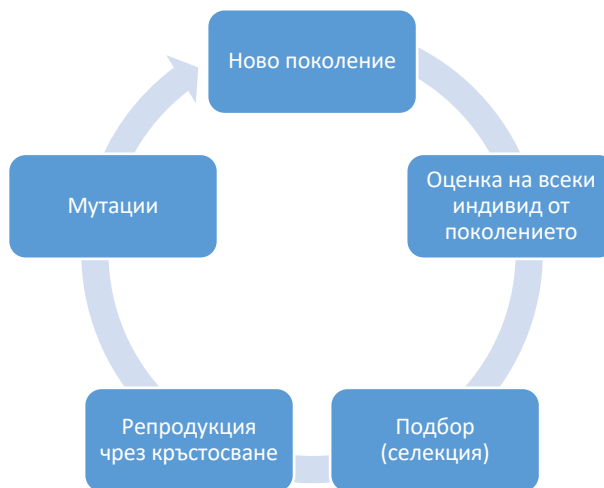
Както е описано по-горе, алгоритъмът за средно отместване (mean shift) минава итеративно през всяка точка в набора от данни се изкачи до най-близкия пик на KDE. Алгоритъмът започва с копиране на оригиналния набор от данни и фиксиране на оригиналните точки. Копираните точки се изместват спрямо оригиналните фиксирани точки.

## 5.2 ГЕНЕТИЧЕН АЛГОРИТЪМ

Генетичният алгоритъм<sup>x</sup> е евристика за търсене, която е вдъхновена от теорията на Чарлз Дарвин за естествената еволюция. Този алгоритъм отразява процеса на естествен подбор, при който най-силните индивиди са избрани за възпроизвеждане, за да произведат потомство на следващото поколение.

Процесът на естествен отбор започва с подбора на по-силни индивиди от населението. Те произвеждат потомство, което наследява характеристиките на родителите и ще бъде добавено към следващото поколение.

Ако родителите имат по-добра фитнес оценка, потомството им ще бъде по-добро от родителите и ще имат по-голям шанс да оцелеят. Този процес продължава да се повтаря докато накрая се намери поколение с най-силните индивиди. На Фигура 6 може да се видят основните стъпки, през които се минава за постигане на желанния резултат.



Фигура 6: Обща схема на работата на генетичния алгоритъм

*Оценка на всеки индивид от поколението (Evaluation of each individual)* - тази стъпка започва с набор от индивиди, които се наричат Population (население). Всеки индивид е възможно решение на проблема, който искате да решите и се характеризира с набор от параметри (променливи), известни като Гени. След това се оценява годността на всеки индивид да участва в създаването на новото поколение (или казано с други думи колко добро е решението). Това става с помощта на фитнес функция, която дава фитнес резултат на всеки индивид (вероятността индивидът да бъде избран за възпроизвеждане).

*Подбор (Selection)* - Идеята на фазата на подбор е да се подберат най-приспособените индивиди и да се оставят да предадат гените си на следващото поколение.

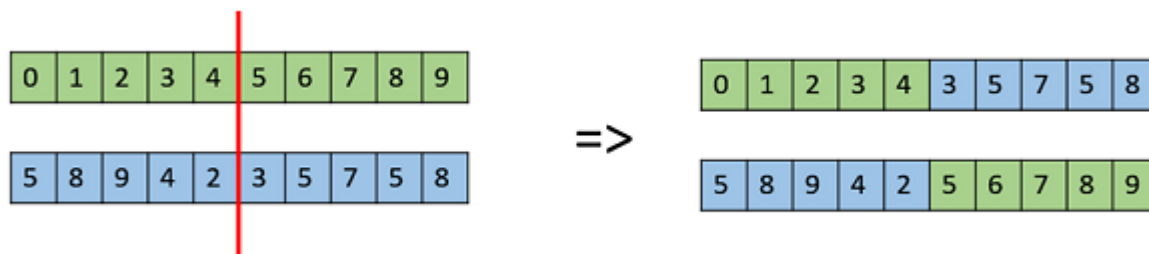
Две двойки индивиди (родители) се избират въз основа на техните фитнес резултати. Индивидите с висок фитнес имат повече шанс да бъдат избрани за възпроизвеждане.

*Репродукция чрез кръстосване (Reproduction(Crossover) <sup>xi</sup>)* :

Тази фаза е изключително важна тъй като при нея се генерира новото поколение. За всяка двойка родители, които трябва да бъдат чифтосани, се избира случайна точка на

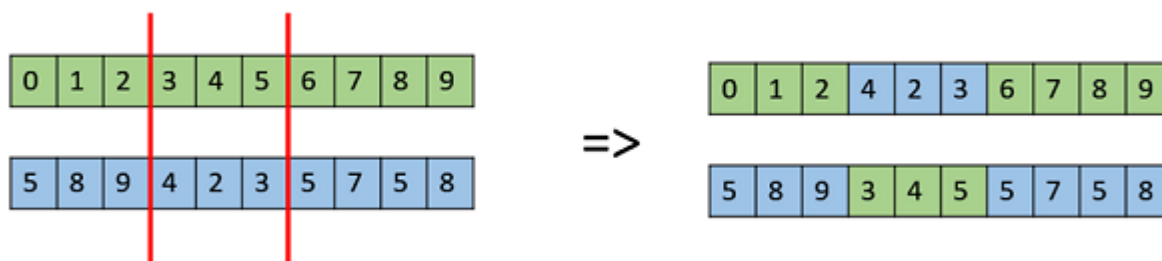
пресичане от гените. Потомството се създава чрез обмен на гените на родителите помежду си, като има няколко възможни подхода:

- Едноточкова кръстоска (One Point Crossover) - избира се случайна точка на кръстосване и опашките на двамата ѝ родители се разменят (виж Фигура 7)



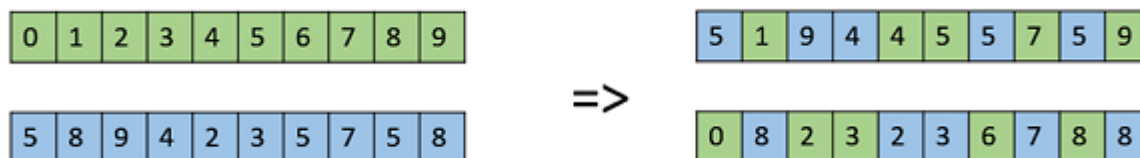
Фигура 7: One Point Crossover

- Многоточкова кръстоска (Multi Point Crossover) - избират се множество случайни точки на кръстосване и опашките на двамата родители се разменят (виж Фигура 8).



Фигура 8: Multi Point Crossover

- Uniform Crossover - не се разделя хромозомата на сегменти, а се третира всеки ген поотделно. В този случай по същество ще “хвърляме монета” за всяка хромозома, за да се реши дали ще бъде включена в гените на наследника. Също така може да се даде предимство на единия родител (виж Фигура 9).



Фигура 9: Uniform Crossover

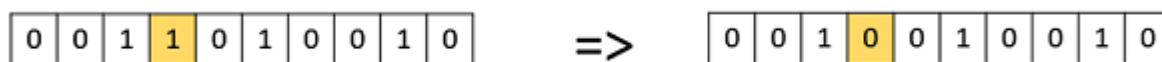
- Davis' Order Crossover (OX1)

OX1 се използва за превключвания на базата на пермутация с намерението да се предава информация за относително подреждане на отделните извори. Тя работи както следва:

1. Създават се две случайни точки на кръстосване в родителя и се копира сегмента между тях от първия родител до първото потомство.
2. Сега, започвайки от втората точка на кръстосване във втория родител, се копират оставащите неизползвани номера от втория родител към първото дете.
3. Повтаря се това за второто дете, като втория родител става първи.

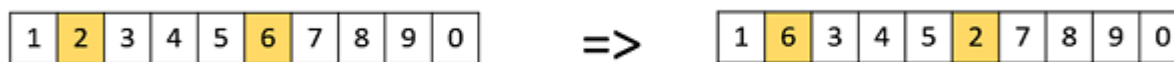
*Mutation*: Казано с прости думи<sup>xii</sup>, мутацията може да се определи като малка случайна настройка в хромозомата, за да се получи ново решение. Тя се използва за поддържане и въвеждане на разнообразие в генетичната популация. Видове мутация:

- Bit Flip Mutation (виж Фигура 10) -избира се един или повече случайни битове(гени) и ги се обръщат(от 0 на 1-ца и обратното). Това се използва за двоично кодирани GA.



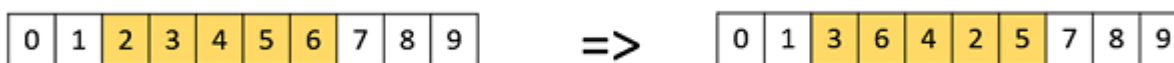
Фигура 10: Bit Flip Mutation

- Random Resetting-Случайното нулиране е разширение на Bit Flip Mutation. При него произволна стойност от множеството допустими стойности се присвоява на случайно избран ген.
- Swap Mutation(виж Фигура 11) - избират се две позиции на хромозомата на случаен принцип и се обменят стойностите. Това е често срещано при кодирането на базата на пермутация.



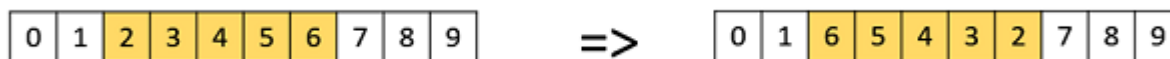
Фигура 11: Swap Mutation

- Scramble Mutation-Scramble мутация (виж Фигура 12) също е популярна с пермутационни представяния. При това, от цялата хромозома, се избира подмножество гени и техните стойности се разбъркват.



Фигура 12: Scramble Mutation

- Inversion Mutation(виж Фигура 13) -избира се подмножество гени като в Scramble мутацията, но вместо да се разбърква подмножеството, просто се инвертира целия низ



Фигура 13: Inversion Mutation

*Ново поколение (New Generation)* : Това е последната стъпка от алгоритъма и при нея се решава кои индивиди ще останат в популацията. Тук има 2 възможни подхода:

- заменя се цялата популация с наследниците
- заменя се някакъв процент от популацията с наследниците (тук най-често се заместват индивидите с най-ниска фитнес оценка с дъщерните или се заместват индивидите на случаен принцип)

### 5.3 K-MEANS

K-means алгоритмите са едни от най-често използваните алгоритми за клъстеризация, защото постигат добър баланс между сложност, бързодействие и резултати. За да се намери най-оптималното решение най-бързо, те в повечето случаи се използват съвместно с генетичен алгоритъм (подхода, който и аз съм избрала да следвам).

Този алгоритъм работи на базата на даден набор от данни, чрез предефиниран брой от клъстери. Изходът от него е K клъстера. Самият алгоритъм можете да видите описан по-долу:

1.  $k$  начални "среди"(kmeans) (в този случай  $k = 3$ ) са произволно генерирани -на случаен принцип
2.  $k$  клъстери се създават чрез свързване на всяко наблюдение с най-близката средна стойност. Определя се за всеки индивид към кой клъстер трябва да принадлежи.
3. Центроидът на всеки от  $k$  кластерите става новата средна стойност.
4. Стъпки 2 и 3 се повтарят до достигане на сходство.

Недостатък на алгоритъма е, че ако се пусне два пъти един след друг алгоритъма ще даде различни резултати. При K-means++ е променена само стъпка 1-центровете не са генерирани на случаен принцип, благодарение на което ако се пусне два пъти един след друг алгоритъма ще даде еднакви резултати.



Разпространението на афинитета е публикувано за първи път през 2007 г. от Brendan Frey и Delbert Dueck в Science. За разлика от други традиционни методи за клъстериране, Affinity Propagation не изисква да указвате броя на клъстерите. Казано по-неспециалистично, в Affinity Propagation всяка точка от данни изпраща съобщения до всички други точки, като информира своите цели за относителната привлекателност на всяка цел за подателя. След това всяка цел отговаря на всички податели с отговор, информиращ всеки подател за наличността му да се свърже с подателя, предвид привлекателността на съобщенията, които е получила от всички други податели. Изпращачите отговарят на целите със съобщения, информиращи всяка цел за преразгледаната относителна привлекателност на целта за подателя, предвид съобщенията за наличност, които е получил от всички цели. Процедурата за предаване на съобщения продължава до постигане на консенсус. След като подателят бъде свързан с една от целите си, тази цел се превръща в пример на точката. Всички точки с един и същ пример са поставени в един и същ клъстер.

Алгоритъм: На база на дейта сета трябва да изчислим:

1. Similarity Matrix -С изключение на тези по диагонала, всяка клетка в матрицата на подобие се изчислява чрез отрицание на сумата от квадратите на разликите между участниците.
2. Responsibility Matrix-Започваме с изграждането на матрица за наличност с всички елементи, зададени на нула. След това изчисляваме всяка клетка в матрицата на отговорността, като използваме следната формула:

$$r(i, k) \leftarrow s(i, k) - \max_{k' \text{ such that } k' \neq k} \{a(i, k') + s(i, k')\},$$

където  $i$  се отнася за индекса на реда, а  $k$  за индекса на колоната.

3. Availability Matrix: Използваме отделно уравнение за актуализиране на елементите по диагонала на матрицата за наличност, отколкото елементите извън диагонала на матрицата за наличност. Формулата на процедурата се използва за попълване на елементите по диагонала:

$$a(k, k) \leftarrow \sum_{i' \text{ such that } i' \neq k} \max\{0, r(i', k)\},$$

където  $i$  се отнася за индекса на реда, а  $k$  за индекса на колоната.

4. Criterion Matrix: Всяка клетка в критериалната матрица е просто сумата от матрицата за наличност и матрицата за отговорност на това място. Изчислява се чрез формулата:

$$c(i, k) \leftarrow r(i, k) + a(i, k).$$

Най-високата стойност на критерия за всеки ред е посочена като пример. Редове, които споделят един и същ пример, са в един и същ клъстер.

## 5.5 DEEP CLUSTERING

Deep Clustering е метод, който едновременно изучава представления за характеристики и клъстерни задания, използвайки дълбоки невронни мрежи. DEC научава mapping от пространството с данни към пространство с по-ниски измерения, в което итеративно оптимизира обединяващата цел. Чрез извличане на семантична информация от изображението, можем да получим по-добър изглед на сходството на екземплярите (в нашия случай това са четенията).

Примерна архитектура за този алгоритъм е:



Фигура 14: Примерна архитектура за deep clustering

Предимства на алгоритмите, използващи дълбоко обучение<sup>xiv</sup>:

- *Висока размерност:* Много алгоритми за клъстериране страдат от основния недостатък на проклятието за размерност. Повечето от алгоритмите разчитат до голяма степен на мерки за сходство, базирани на разстояния. Тези мерки работят сравнително добре при данни с ниски размери, но с нарастването на размерите те губят своята дискриминационна сила, силно засягайки качеството на клъстерирането. Дълбоките клъстеризиращи алгоритми използват дълбоки невронни мрежи, за да научат подходящо представяне на нискоразмерни данни, което до известна степен облекчава този проблем. Въпреки че класическите алгоритми като Spectral Clustering решават този проблем, като включват намаляване на размерите в своя дизайн, невронните мрежи са много успешни при създаването на подходящи представления от данни за широк спектър от задачи, когато са снабдени с подходящи целеви функции. Следователно, алгоритмите за дълбоко клъстериране блестят за способността им да научават изразителни, но нискоразмерни представления на данни, подходящи за групиране от сложни високоразмерни данни.
- *Рамка от край до край:* Дълбоките рамки за клъстериране комбинират извличане на характеристики, намаляване на размерността и клъстериране в модел от край

до край, позволявайки на дълбоките невронни мрежи да научат подходящи представяния, за да се адаптират към предположенията и критериите на модула за клъстериране, който се използва в модела. Това облекчава необходимостта да се извършва многостранно обучение или намаляване на размерността на големи набори от данни поотделно, вместо да се включи в обучението на модела.

- *Мащабируемост:* Чрез включването на дълбоки невронни мрежи, алгоритмите за дълбоко клъстериране могат да обработват големи високоразмерни набори от данни като изображения и текстове с разумна сложност във времето. Наученото пространство за представяне са пространства с ниски размери, позволяващи на други клъстериращи алгоритми ефективно да групират големи реални набори от данни и да извеждат информация за клъстера в реално време след първоначалното обучение.

Предизвикателства при използването на алгоритми за дълбоко обучение:

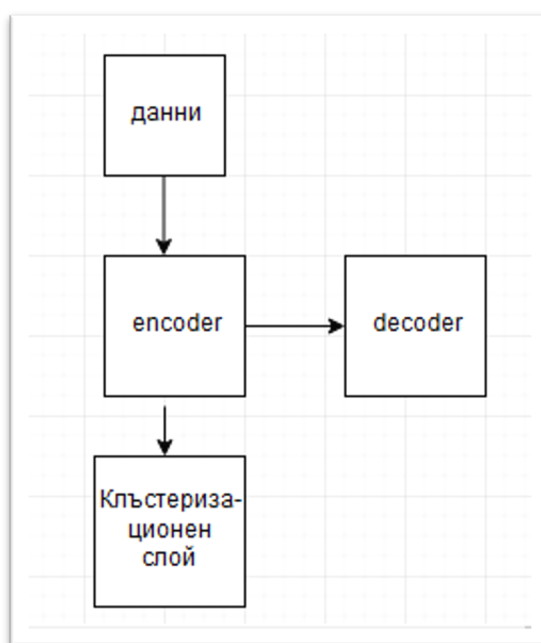
- *Хипер-параметри:* Моделите за дълбоко клъстериране имат няколко хиперпараметри, които не са тривиални за задаване. Основният недостатък на дълбокото клъстериране произтича от факта, че при клъстерирането, което е задача, която не се контролира, ние не разполагаме с лукса да проверяваме ефективността на реални данни. Трябва да разчитаме на референтни набори от данни, за да оценим хиперпараметрите и да се надяваме, че това се превръща в реалния свят, което сериозно поставя под съмнение правдоподобността на прилагането на дълбоки клъстерни модели в сценариите от реалния свят. Това е още по-притеснително, когато забележим, че всички модели, които обсъдихме по-горе, обикновено се представят много добре на MNIST, но тяхното представяне може да варира диво в други набори от данни като Reuters.
- *Липса на разбираемост:* Въпреки че интерпретативността е голям проблем с невронните мрежи като цяло, липсата на такава е особено по-значима при сценарии, при които валидирането е трудно. Представянията, научени от дълбоките невронни мрежи, не са лесно интерпретируеми и по този начин трябва да положим значително ниво на доверие на резултатите, получени от моделите. Следователно трябва да бъдат разработени дълбоки клъстерни модели с интерпретируеми или разчленени представления, които дават представа за това какви характеристики улавят представителствата и на какви атрибути на данните се базират клъстерите.
- *Липса на теоретична рамка:* По-голямата част от алгоритмите за дълбоко клъстериране, които обсъдихме по-горе, нямат силна теоретична основа. Моделът може да бъде изразителен и надежден без теоретично заземяване, но често е много трудно да се предскаже тяхното поведение и ефективност в ситуации извън извадката, които могат да представляват сериозно предизвикателство при ненаблюдавани настройки, като клъстериране.

Конструиране на общия алгоритъм на изследването е най-важната стъпка от цялостния проект. В тази фаза е основния рисърч. В това изследване ще изтествам два възможни подхода за решаването на тази задача.

Първият подход е стандартно клъстеризиране на база предварително обработени данни, а вторият е подхода на “deepclustering”.

### 6.1 ИЗСЛЕДВАНЕ С ТЕХНИКИ НА ДЪЛБОКОТО КЛЪСТЕРИЗИРАНЕ

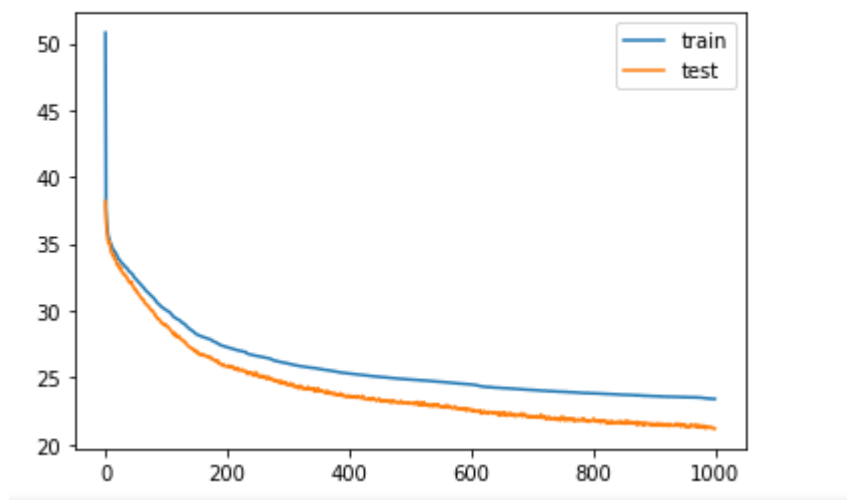
Дълбоките клъстериращите алгоритми комбинират извличане на характеристики, намаляване на размерността и клъстериране в модел от край до край, позволявайки на дълбоките невронни мрежи да научат подходящи представяния, за да се адаптират към предположенията и критериите на модула за клъстериране, който се използва в модела. Алгоритъма, който ще използваме за построяване на нашето изследване е предложението от Yuefeng Zhang в статията му “Deep Clustering for Financial Market Segmentation”. Алгоритъма е изграден на база на аутоенкодер и клъстеризационен слой, а самата структурата на модела, който ще бъде използван можете да видите на Фигура 15. За характеристики на модела ще бъдат използвани директно чънковете на секвенциите транслирани до числови стойности. Сходството ще бъде определено по това дали чънковете се намират в едни и същи клъстери. Аутоенкодера ще бъде обучен за 1000 епохи, а за клъстеризационния слой отново ще използваме генетичния алгоритъм от библиотеката “pyeasyga” за да определим правилния брой клъстери.



Фигура 15: Структура на модел за дълбоко клъстериране

Резултатите, които получих от обучението на аутоенкодера са:

Epoch 1000/1000  
4140/4140 - 24s - loss: 23.3983 - val\_loss: 21.1714



Фигура 16: Резултати получени при обучението и тестването на аутоенкодер

За жалост не можах да изчисля `silhouette_score`, заради липса на памет (**MemoryError**: Unable to allocate 1.00 GiB for an array with shape (6078, 22079) and data type float64), така че този алгоритъм се отхвърля като възможен за крайната имплементация на решението.

## 6.2 ИЗСЛЕДВАНЕ С ТЕХНИКИ НА КЛЪСТЕРИЗИРАНЕ ЧРЕЗ ИЗПОЛЗВАНЕ НА ИНЖИНЕРИНГ НА ХАРАКТЕРИСИТИКИТЕ (CLUSTERING WITH FEATURE ENGINEERING)

При него предварителната обработка на данните е основополагаща за бъдещия успех на клъстеризацията. Проучването е насочено към проучване на методи **не**базирани на подравняване, тъй като те са значително по-бързи. Такива методи са:

*-phylogenetic analysis<sup>xv</sup>*-Състои се от установяване на еволюционна връзка между последователностите от семейства нуклеинови киселини или протеини.

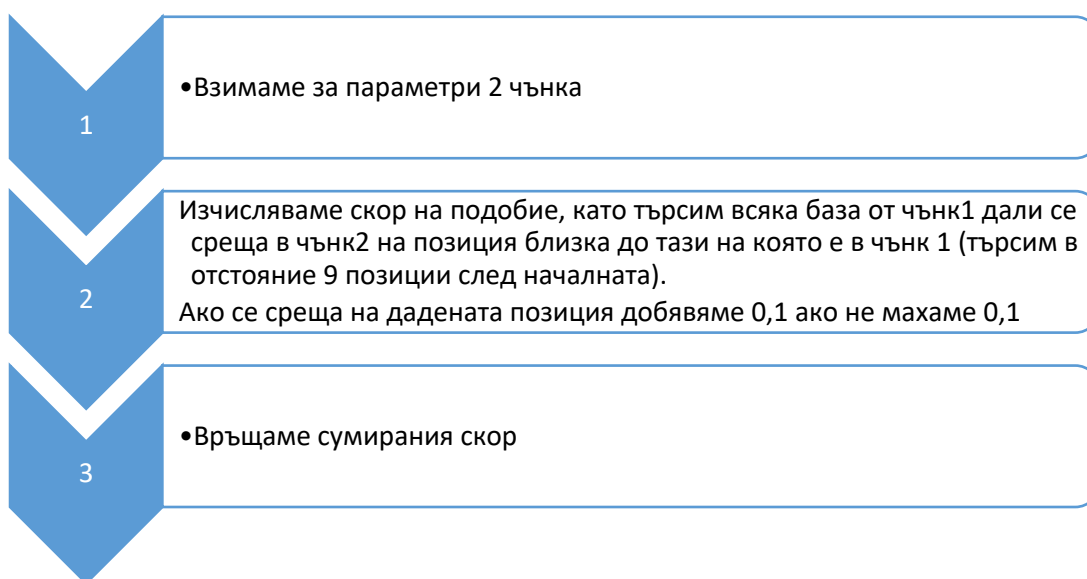
Обикновено се изобразява чрез използването на дихотомични дървета, за които клоните представляват отделяне на организма. Клонове, които са близо един до друг, предполагат подобен организъм. За съжаление този метод е много труден за имплементация и скъп откъм ресурси и време заради това беше отхвърлен като метод който да бъде приложен. Към този раздел влизат всички методи базирани на дървовидни структури.

*-genomic signal processing (GSP)<sup>xvi</sup>*-Обработката на геномни сигнали (GSP) е дефинирана като анализ, обработка и използване на геномни сигнали за придобиване на биологични знания и трансформиране на тези знания в системни приложения, където под геномни сигнали имаме предвид измеримите събития,

главно производството на иРНК и протеин, извършвани в клетката. Благодарение на определящата роля на ДНК в производството на иРНК, структурната характеристика на ДНК неизбежно е част от GSP и, интересно, методите за обработка на сигнала се използват за разбиране на структурата на ДНК. Този метод е сравнително нов. Базиран е на фурие трансформации и дава доста обещаващи ресурси според статиите които използвах за изследването, но за жалост моята имплементация се оказва твърде тежка за машината на която работя, така че не мога да приложа резултати от метода, но това е насока за доработка на този проект.

*-методи базирани на броене на думи* -Тези методи са доста използвани в различни публикации (Jun S-R, Sims GE, Wu GA, Kim S-H. Whole-proteome phylogeny of prokaryotes by feature frequency profiles: an alignment-free method with optimal feature resolution. Proc Natl Acad Sci U S A. 2010;107:133–8. ;Sims GE, Kim S-H. Whole-genome phylogeny of Escherichia coli/Shigella group by feature frequency profiles (FFPs) и други).

В проекта аз съм използвала моя версия на този алгоритъм, състоящ се от следните стъпки:



В него изчисляваме стойност (scoring) на подобие на двата откъса от последователности, като за всяка база изчисляваме стойността въз основа на това дали има съответствие между двата откъса и дали тези съответствия са част от последователността или не.

*-методи базирани на най-дълъг общ подстринг*-Също доста популярен метод, използван в редица успешни проучвания като Ulitsky I, Burstein D, Tuller T, Chor B. The average common substring approach to phylogenomic reconstruction. J Comput

Biol. 2006;13:336–50. и Leimeister C-A, Morgenstern B. Kmacs: the k-mismatch average common substring approach to alignment-free sequence comparison. Bioinformatics. 2014;30:2000–8. Този метод ще го видите приложен и в моята имплементация, чрез SequenceMatcher класа предоставен ни от Python.

-методи базирани на изчисляване на дистанция-Също много популярна група от методи, но този който аз съм избрала да имплементирам за проекта е “rank distance”<sup>xvii</sup>. Скорът, който връща този метод е сумата от абсолютните разлики между подрежданията на символите в чънк А и чънк Б. Най-добрият резултат ще е този близък до 0. Друг метод, който е използван е Левенщайн разстояние, което е минималният брой операции, необходими за трансформиране на един масив в друг, чрез операциите вмъкване, изтриване или заместване на единичен елемент (в нашия случай знак).

-методи базирани на теорията за Хаоса-Тази група методи, изискват по-голяма изчислителна мощност и не бяха включени като част от изследването

-методи базирани на Fuzzy integral similarity

-Center Star Method<sup>xviii</sup>-Методът на централната звезда е алгоритъм за сближаване с отношение 2. Той работи по следния начин: Идентифицираме централния низ и се използват подравняванията на двата подниза за да се създаде множествено подравняване. Решението на проблема е с линейна сложност, а резултатът е неоптимален. В проекта съм използвала имплементацията на techoverflow. net<sup>xix</sup>

Следващата стъпка е да бъде избран метод за клъстеризация, който ще бъде използван за финалната имплементация. За мястото ще се състезават всички клъстерни алгоритми описани в подбор на алгоритми за клъстерен анализ в областта на ИИ (K-means, K-means++, Affinity Propagation и Mean-shift). Мерките, по които ще бъдат сравнявани са бързодействие и Silhouette Coefficient.

Сравнението на алгоритмите използвани за препроцесинг е на база скорост и колко добре клъстеризират. Изследването е направено с използването на данните за ХИВ и ковид 19 разделени на под чънкове по 200 бази и обработени с алгоритъм без подравняване и са сравнени с резултатите от същото изследване направено с алгоритъм с глобално подравняване, а като метод за клъстеризиране е използван K-means

Алгоритми за препроцесинг	бързина	скоп(test set)
word count	14. 59 sec	0. 54
longest_substring	3. 08 sec	0. 97
rank_distance	8. 78 sec	0. 52
Center Star Method	1129. 80 sec	0. 55
Levenshtein Distance	200 sec	0. 81 (16 клъстера)

boost_similarity (взети са в предвид rank_distance, word_count и longest_substring)	23. 49 sec	0. 55
global align	774. 42 sec	0. 55

От резултатите се вижда, че boost\_similarity (където имаме няколко алгоритъма които взимат решение), не е значително по-добро от rank\_distance (който е най-добре представилия се алгоритъм -правилно разпознава 17% от случаите които метода със използване на глобално подравняване ще класифицира като сходни), заради това ще пробвам и двата алгоритъма при реалната имплементация и ще представя резултатите на по-добре представилия се. Освен това от тези данни стигаме до следните изводи:

- очаква се новия алгоритъм да е до 85 пъти по бърз от метод с подравняване.
- трябва да се изследват и други методи без подравняване за в бъдеще, тъй като rank\_distance е най-добре представилия се, а не е с впечатляващ скор.
- longest\_substring очаквано е най-слабия алгоритъм, но и най-бързия.

По отношение на алгоритмите за клъстеризация данните са следните (Изследването е направено с използването на данните за ХИВ и ковид 19 разделени на под чънкове по 200 бази и обработени с алгоритъм без подравняване-boost\_similarity) Броя на клъстерите беше избран с помощта на генетичен алгоритъм, изграден на база питонската библиотека “pyeasyga”.

	бързина	скор
Mean-shift	123. 15 sec	0. 22
K-means	34. 72 sec	0. 83
K-means++	30. 60 sec	0. 83
Affinity Propagation	456. 61 sec	0. 059

На базата на тези данни взех решение, че за реалната имплементация ще използвам K-means++, като алгоритъм за клъстеризация. Освен това за да се оптимизира алгоритъма беше използван генетичен алгоритъм, който да ни помогне да определим **k** на база на silhouette\_score. Неговите резултати са:

K-means стартов брой клъстери	Резултат:
3	{'clusterNum': 3, 'score': 0. 3314904954508576}
4	{'clusterNum': 4, 'score': 0. 33925889355778627}



5	{'clusterNum': 5, 'score': 0.3545574142634729}
6	{'clusterNum': 6, 'score': 0.2815897772011829}
7	{'clusterNum': 7, 'score': 0.29105539139355924}
8	{'clusterNum': 8, 'score': 0.2944827773467973}
9	{'clusterNum': 9, 'score': 0.295807824475671}

(5, [0, 0, 0, 0, 1, 0, 1])

Както се вижда най-добри резултати получаваме ако тренираме Kmeans за 5 клъстера.

Освен това за подобряване на оценката на клъстеризацията са използвани:

- **adjusted mutual info score<sup>xx</sup>**-Коригирана взаимна информация (AMI) е корекция на оценката за взаимна информация (MI), за да се отчете случайността. Той отчита факта, че MI обикновено е по-висок за две клъстери с по-голям брой клъстери, независимо дали всъщност има повече споделена информация. AMI връща стойност 1, когато двата дяла са идентични (т. е. идеално съвпадащи). Случайните дялове (независими етикети) имат очакван AMI около 0 средно, поради което могат да бъдат отрицателни.
- **completeness score**-Резултатът от клъстериране удовлетворява пълнотата, ако всички точки от данни, които са членове на даден клас, са елементи от един и същ клъстер. Резултат между 0, 0 и 1, 0. 1. 0 означава идеално пълно етикетирание
- **adjusted rand score**-Индексът Rand изчислява мярка за сходство между две клъстери, като разглежда всички двойки проби и броене на двойки, които са присвоени в един и същи или различни клъстери в предвидените и истински клъстери. Резултат за сходство между -1, 0 и 1, 0. Случайните етикети имат ARI близо до 0, 0. 1. 0 означава перфектно съвпадение.
- **calinski harabasz score**-Резултатът се определя като съотношение между дисперсията в рамките на клъстера и дисперсията между клъстерите.
- **davies bouldin score**-Резултатът се определя като средна мярка за сходство на всеки клъстер с най-сходния му клъстер, където сходството е съотношението на разстоянията в рамките на клъстера към разстоянията между клъстера. По този начин клъстерите, които са по-отдалечени и по-малко разпръснати, ще доведат до по-добър резултат. Минималният резултат е нула, като по-ниските стойности показват по-добро групиране.

- fowlkes mallows score-Индексът на Fowlkes-Mallows (FMI) се определя като средно геометрично между precision и recall
- homogeneity score-Резултатът от клъстериране удовлетворява хомогенността, ако всичките му клъстери съдържат само точки от данни, които са членове на един клас. Резултат между 0, 0 и 1, 0. 1. 0 означава идеално хомогенно етикетиране
- mutual info score-Взаимната информация е мярка за сходство между два етикета на едни и същи данни.
- silhouette score-Силуетният коефициент се изчислява, като се използва средното вътрешно-клъстерно разстояние (a) и средното разстояние на най-близкия клъстер (b) за всяка проба. Силуетният коефициент за проба е  $(b - a) / \max(a, b)$ . За да се изясни, b е разстоянието между проба и най-близкия клъстер, от която пробата не е част. Обърнете внимание, че силуетният коефициент се дефинира само ако броят на етикетите е  $2 \leq n\_labels \leq n\_samples - 1$ .

За имплементацията на тези метрики беше използван sklearn.

Тъй като претеглената оценка от гореспоменатите алгоритми (word\_count, levenshtein\_distance, central star и rank distance) не даде достатъчно добри резултати при клъстеризацията избрах един не толкова стандартен подход-да обуча регресор с който да предскажа скората базиран на метод с глобално подравняване. Ще създам регресор, който ще предсказва global alignment score, на база на word\_count, levenshtein\_distance, central star и rank distance алгоритмите. Моделите за регрешън, който ще бъдат тествани са MLPRegressor, LinearRegression, XGBRegressor и RandomForestRegressor. За да съм сигурна, че използвам алгоритъма оптимално всеки от тях беше оптимизиран с използването на "GridSearchCV". Освен това фичиите предварително бяха обработени с StandardScaler.

Получените резултати бяха:

RandomForestRegressor<sup>xxi</sup> - Случайната гора е вид контролиран учебен алгоритъм, който използва ансамблови методи (пакетиране) за решаване както на проблеми с регресията, така и на класификацията. Алгоритъмът работи чрез конструиране на множество дървета за вземане на решения по време на обучение и извеждане на средната стойност / режим на прогнозиране на отделните дървета. Основната концепция зад случайната гора е мъдростта на тълпите, при която голям брой некорелирани модели, действащи като комитет, ще надминат всеки от отделните съставни модели. Причината за това е фактът, че дърветата се предпазват взаимно от своите индивидуални грешки. В случайна гора няма взаимодействие между отделните дървета. Случайната гора действа като алгоритъм за оценка, който обобщава резултата от много дървета за вземане на решения и след това извежда най-оптималния резултат.

```
explained_variance: 0.6049
r2: 0.6049
MAE: 19.4098
MSE: 601.263
RMSE: 24.5207
```

LinearRegression<sup>xxii</sup> - LinearRegression се вписва в линеен модел с коефициенти  $w = (w_1, \dots, w_p)$ , за да минимизира остатъчната сума от квадратите между наблюдаваните цели в набора от данни и целите, предвидени от линейното приближение.

```
explained_variance: 0.6196
r2: 0.6196
MAE: 18.8112
MSE: 585.7251
RMSE: 24.2018
```

MLPRegressor<sup>xxiii</sup> - класът MLPRegressor реализира многослоен перцептрон (MLP), който тренира, използвайки обратно разпространение без функция за активиране в изходния слой, което също може да се разглежда като използване на функцията за идентичност като функция за активиране. Следователно, той използва квадратната грешка като функция на загубата, а изходът е набор от непрекъснати стойности. MLPRegressor също поддържа регресия с много изходи, при която извадката може да има повече от една цел.

```
explained_variance: 0.6298
r2: 0.6297
MAE: 18.6609
MSE: 570.1427
RMSE: 23.8777
```

XGBRegressor<sup>xxiv</sup> - XGBoost означава „Extreme Gradient Boosting“ и представлява реализация на алгоритъм за увеличаване на градиентните дървета. XGBoost е популярен контролиран модел за машинно обучение с характеристики като скорост на изчисление, паралелизация и производителност.

```
explained_variance: 0.6889
r2: 0.6889
MAE: 17.3129
MSE: 479.0267
RMSE: 21.8867
```

Както виждаме най-добре представилия се алгоритъм е XGBRegressor. Той беше обучен върху всички възможни данни, като разбира се 20 % от тях бяха оставени само за тестване.

На база на този регресор пробвах какви резултати ще получа от клъстеризирането(на база К ако за характеристика използвам само този скор или добавя mean,std и signal to noise метрики на база на движещ се прозорец с големина 3.

Резултатите, които получих са:

-без преобразуване:

```
ami_score: 0.27178627052362825
compl_score: 0.2717883179734934
adj_rand_score: 0.322670382157766
calinski_harabasz_score: 1520550.3048562098
davies_bouldin_score: 0.4371781851861802
fowlkes_mallows_score: 0.6616862302382961
homogeneity_score: 0.2740341103912846
mutual_info_score: 0.19521746492739153
silhouette_score: 0.5515149235725403
```

---

-с преобразуване:

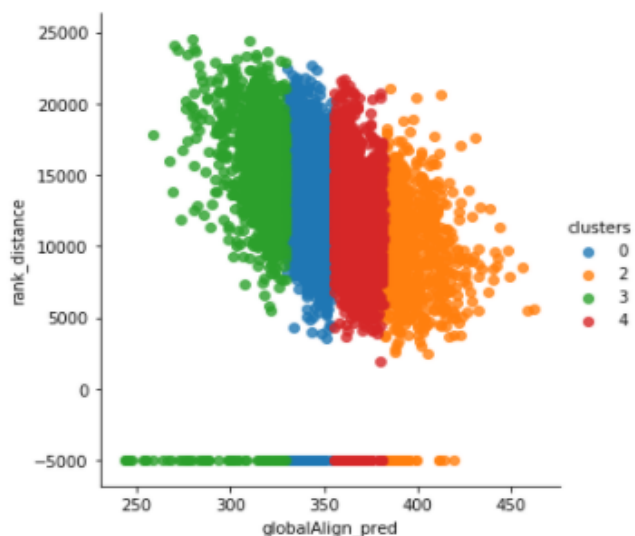
signalToNoise параметъра беше изчислен с формулата 'mean'/'STD' изчислени на база на плъзгащи се прозорци с големина 3. Идеята зад този подход е да вземем в предвид сходството не само на 2 чънка а и на целия регион в който се намират. Получените резултати са:

```
ami_score: 4.41397633868181e-06
compl_score: 8.664012883248568e-06
adj_rand_score: 6.146901212196874e-06
calinski_harabasz_score: 207826.8280578309
davies_bouldin_score: 0.9252134322080752
fowlkes_mallows_score: 0.7032650263409023
homogeneity_score: 0.01593939910659514
mutual_info_score: 6.378530308559058e-06
silhouette_score: 0.30189395806224983
```

Както виждаме подходът без преобразуване ни дава по-добри резултати.

Така, че ще измерим резултатите от биологична гледна точка базирани на него.

Самите данни бяха разделени на 5 клъстера, както може да се види на Фигура 18.



Фигура 17: Визуално разпределение на резултатите от клъстеризирането

По детайлно отделните клъстери съдържат следните характеристики:

-клъстер 0:

дължина на клъстер 0:2290 примера  
максимален скор в клъстер 0:355.18511962890625  
минимален скор в клъстер 0:329.8431091308594

-клъстер 1:

дължина на клъстер 1:0 примера  
максимален скор в клъстер 1:nan  
минимален скор в клъстер 1:nan

-клъстер 2:

дължина на клъстер 2:1078 примера  
максимален скор в клъстер 2:462.2823181152344  
минимален скор в клъстер 2:381.48712158203125

-клъстер 3:

дължина на клъстер 3:1165 примера  
максимален скор в клъстер 3:329.7847595214844  
минимален скор в клъстер 3:243.86570739746094

-клъстер 4:

---

дължина на клъстер 4:2367 примера  
максимален скор в клъстер 4:381.4415588378906  
минимален скор в клъстер 4:355.19586181640625

Трябва да се има предвид, че най-добрият скор, който може да бъде постигнат за глобално подравняване при използваните параметри е 1121. Ако секвенциите са абсолютно еднакви, то стойността ще бъде 0, докато най-лошият скор ще бъде със стойност 104. В тестовия сет най-добрият изчислен скор за глобално подравняване с който разполагаме е 440. Както виждате най-добър скор имаме в клъстер 2 и клъстер 4. Сравнявайки резултатите с тези получени от clustal omega установих че повечето сходни секвенции попадат в клъстер 4, а не в клъстер 2 както очаквах. Това може да се дължи на :

1. Алгоритъма по който работят 2-та алгоритъма. Clustal Omega прави глобал алаймънт на целите секвенции, докато при нашия алгоритъм разделяме секвенциите на части по 200 бази и след това изчисляваме сходството на принципа всеки срещу всеки (тоест може да намерим най-високо сходство между чънк 1 и чънк 18)
2. Начина по който са представени резултатите. Софтуера ни показва къде имаме съвпадение според глобал алаймънта, а горе представения алгоритъм показва сектори с най-добър скор за прилика.
3. Грешката на регресора

За да проверим дали правилно откриваме сходства ще използваме Clustal Omega (примерни резултати може да се видят на Фигура 19)

Фигура 18: Резултати взети от клъстер омега показващи сходство между Ковид19 и ХИВ

30

```

NC_045512.2      ACACGTCCTCAACTCAGTTTGCCTGTTTTACAGGTTTCGCGACGTGCTCGTACGTGGCTTTGG 360
NC_001802.1      TAAGCCTCAATAAAGCTTGCC---TTGAGTGCTTCAAGTAGTGTGTGCCCGTCTGTTGTG 121
                *   ***   **   *****   **   *   *   ***   *   ***   *   ***   **   *

```

```

testM=X_test_hc[X_test_hc['chunk_or1']].str.contains('TTGAGTGCTTCAAGTAGTGTGTGCCCGTCTGTTGTG', regex=False)]
testA=testM[testM['chunk_or2']].str.contains('TTTACAGGTTTCGCGACGTGCTCGTACGTGGCTTTGG', regex=False)]
testA[['clusters','chunk_or1','chunk_or2']]

```

clusters	chunk_or1	chunk_or2
1	2	GGTCTCTCTGGTTAGACCAGATCTGAGCCTGGGAGCTCTCTGGCTA... TTCGTCCGTGTTGCAGCCGATCATCAGCACATCTAGGTTTCGTCCG...

```

NC_045512.2      TCCTTATGAAGATTTTCAAGAAAACCTGGAACACTAAACATAGCAGTGGTGTACCCGTGA 780
NC_001802.1      TGAGTACGC-----CAAAAATTTTACTAGCGGAGGCTAGAAGGAGAGAGATGGGTGC 343
                *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *

```

```

testM=X_test_hc[X_test_hc['chunk_or1']].str.contains('AAAAATTTTACTAGCGGAGGCTAGAAGGAGAGAGATGGGTGC', regex=False)]
testA=testM[testM['chunk_or2']].str.contains('CAAGAAAACCTGGAACACTAAACATAGCAGTGGTGT', regex=False)]
testA[['clusters','chunk_or1','chunk_or2']]

```

clusters	chunk_or1	chunk_or2
154	4	TGAAAGCGAAAGGGAAACCAGAGGAGCTCTCTGACGCGAGGACTCG... CGAAATACCAGTGGCTTACCGCAAGGTTCTTCTCGTAAGAACGGT...

```

NC_045512.2      AAGGAAGGTGTAGAGTTTCTTAGAGACGGTTGGGAAATTGTTAAATTTATCTCAACCTGT 2230
NC_001802.1      ATAAAAGATGGATAA-TCCTGGGATTAAATAAAATAGTAAGAATGTATAGCCCTACCAGC 1178
                *   ***   *   *   *   *   *   *   *   *   *   *   *   *   *

```

```

testM=X_test_hc[X_test_hc['chunk_or1']].str.contains('TCCTGGGATTAAATAAAATAGTAAGAATGTATAGCCCTACCAGC', regex=False)]
testA=testM[testM['chunk_or2']].str.contains('GAAATTGTTAAATTTATCTCAACCTGTGCTTGTGAA', regex=False)]
testA[['clusters','chunk_or1','chunk_or2']]

```

clusters	chunk_or1	chunk_or2
759	4	TATTGCACCAGGCCAGATGAGAGAACCAAGGGGAAGTGACATAGCA... TGGGAAATTGTTAAATTTATCTCAACCTGTGCTTGTGAAATTGTG...

```

NC_045512.2      AGTTGAACTCGGTACAGAAGTAAATGAGTTCGCCTGTGTTGTGGCAGATGCTGTCTATAAA 2907
NC_001802.1      AGCAGGAGCCGATAGACAAGGAACGTATCCTTTAACTTCCCTCAGGTCACTCTTTGGCA 1818
                **   *   *   *   *   *   *   *   *   *   *   *   *   *   *

```

```

testM=X_test_hc[X_test_hc['chunk_or1']].str.contains('AGCAGGAGCCGATAGACAAGGAACGTATC', regex=False)]
testA=testM[testM['chunk_or2']].str.contains('AGTTGAACTCGGTACAGAAGTAAATGAGTT', regex=False)]
testA[['clusters','chunk_or1','chunk_or2']]

```

clusters	chunk_or1	chunk_or2
1216	4	AATGAAAGATTGTACTGAGAGACAGGCTAATTTTTAGGGAAGATCA... GATGAAAGGATTGATAAAGTACTTAATGAGAAGTGCTCTGCCTATA...

```

NC_045512.2      CTTTAAGAGTTTGTGTAGATACTGTTTCGCACAAATGTCTACTTAGCTGTCTTTGATAAAA 3806
NC_001802.1      CAGTAAAATTAAAGCCAGGAATGGATGGCCCAAAAGTTAAACAATGGCCATTGACAGAAG 2180
                *   ***   *   *   *   *   *   *   *   *   *   *   *   *

```



Както виждаме според Clustal Omega получаваме не лоши резултати, но този инструмент не връща конкретно дали намерените сходства в секвенциите имат биологическа значимост. За тази цел ще използваме Blast.

Резултатите от BLAST ни показват, че макар и да има дадено сходство между части от геномите, то не е биологически значимо. За жалост нашия алгоритъм не отразява това, защото не е обучен на база на данни, които имат биологическо

значение. За да поправим това трябва да се добавят синтетични данни които да покриват целия спектър (от данни без биологическо значение до абсолютно еднакви чънкове). Освен това за да подобрим точността на нашия алгоритъм трябва да намалим размера на чънковете и да добавим допълнителни характеристики към регресора. Тези които ще пробвам са:

- предсказана стойност за локално подравняване
- предсказана стойност на броя гапове които ще има при локално подравняване
- предсказана стойност на броя гапове които ще има при глобално подравняване


За да се види дали тези характеристики биха подобрили представянето на регресора ще ги изтествахме първо с малко данни и линейна регресия.

Резултати от регресора базирани на скората на регресора, изчислен с малко данни без допълнителни характеристики:

```
explained_variance: -0.0101
r2: -0.0209
MAE: 4.759
MSE: 57.9507
RMSE: 7.6125
```

Предсказана стойност на броя гапове които ще има при глобално подравняване

Job Title	NC_045512.2 Severe acute respiratory syndrome		
Query ID	23EKT6UJ11N	Search expires on 02-10 05:26 am	<a href="#">Download All</a> ▼
Program	Blast 2 sequences <a href="#">Citation</a> ▼		
Query ID	Ic Query_68835 (dna)		
Query Descr	NC_045512.2 Severe acute respiratory syndrome corona...		
Query Length	29903		
Subject ID	Ic Query_68837 (dna)		
Subject Descr	NC_001802.1 Human immunodeficiency virus 1, complete ...		
Subject Length	9181		

 No significant similarity found. For reasons why, [click here](#)

Фигура 19: Резултати получени при използването на Blast за определяне на сходство между ХИВ и КОВИД

Резултати от регресора създаден за предричане стойността на характеристика

```
explained_variance: -0.0101  
r2: -0.0209  
MAE: 4.759  
MSE: 57.9507  
RMSE: 7.6125
```

Резултати от регресора за предричане на скората изчислен от глобално подравняване :

```
explained_variance: 0.2257  
r2: 0.2182  
MAE: 4.6236  
MSE: 44.3788  
RMSE: 6.6617
```

#### Предсказана стойност на скората при локално подравняване

Резултати от регресора създаден за предричане стойността на характеристика

```
explained_variance: 0.7417  
r2: 0.7414  
MAE: 8.9039  
MSE: 129.6205  
RMSE: 11.3851
```

Резултати от регресора за предричане на скората изчислен от глобално подравняване :

```
explained_variance: 0.6649  
r2: 0.6649  
MAE: 9.0954  
MSE: 167.9902  
RMSE: 12.9611
```

#### Предсказана стойност на броя гапове които ще има при локално подравняване

Резултати от регресора създаден за предричане стойността на характеристика:

```
explained_variance: 0.3072  
r2: 0.3039  
MAE: 6.0089  
MSE: 61.7824  
RMSE: 7.8602
```

Резултати от регресора за предричане на скората изчислен от глобално подравняване :

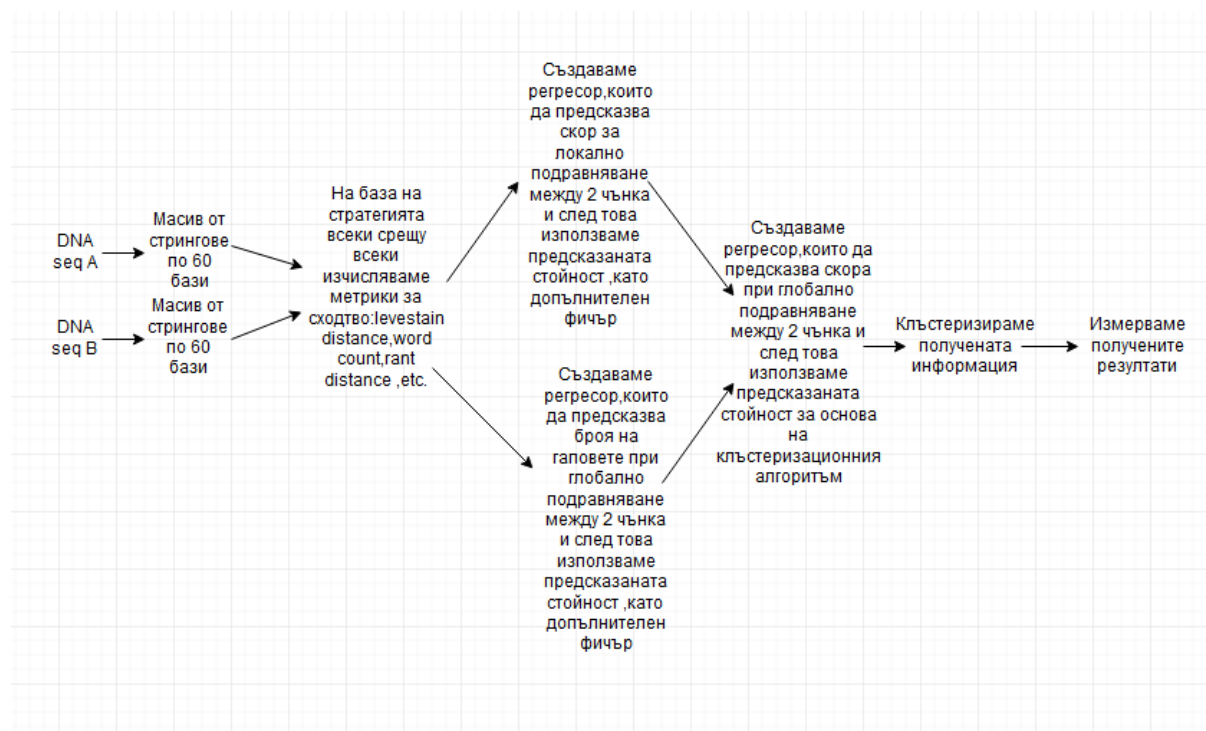
```
explained_variance: 0.2003
r2: 0.1944
MAE: 6.1796
MSE: 71.502
RMSE: 8.4559
```

Резултати от регресора за предричане на скор изчислен от глобално подравняване като използваме всички характеристики заедно

```
explained_variance: 0.6667
r2: 0.6667
MAE: 9.0942
MSE: 167.082
RMSE: 12.926
```

## 7 ИЗБОР И ОПТИМИЗАЦИЯ НА ФИНАЛЕН АЛГОРИТЪМ

В финалната имплементация на решението се опитам да постигна максимален баланс между бързина и точност. Тя включва:



Фигура 20: Схема на алгоритъма, който ще бъде използван за финалната имплементация

Нека разгледаме в детайли по-важните стъпки:

1. На база стратегията всеки срещу всеки се изчисляват метрики за сходство като word count,levestain distance, central start and rank distance algorithm. При

генерирането на метриците се използва паралелно програмиране(това включва мултитрединг и мултипроцесинг), за да може да се получи по добра производителност. Тъй като данните са разделени на чънкове по 60 бази крайния обем на данните, които са използване за обучение на алгоритъма е приблизително 1 терабайт.

2. Създаване на регресор, който ще се ползва за предсказване на скор за локално подравняване между 2 чънка. За характеристики ще бъдат използвани `word_count`, `rank_distance`, `levestain_distance` и `central_star distance`, а вече изчислената стойност за локално подравняване ще бъде „таргет“ променлива. Алгорътъма, който ще се използва е Линейна регресия. След тренирането му този регресор ще бъде използван като допълнителен характеристика към данните с името „`local_align_pred`“.
3. Създаване на регресор, който ще се използва за предсказване броя на гаповете при изчисляване на скор за глобално подравняване между 2 чънка. За характеристики ще бъдат използвани `word_count`, `rank_distance`, `levestain_distance` и `central_star distance`, а вече изчислената стойност за броя на гаповете ще бъде „таргет“ променливата. Алгорътъма, който ще се използва е Линейна регресия. След тренирането му този регресор ще бъде използван като допълнителен характеристика към данните с името „`global_align_gap_pred`“.
4. След като генерираме обучим гореспоменатите регресори и генерираме стойности за новите характеристики, ще бъде създаден основния регресор. Него ще използваме за предсказване на скор за глобално подравняване между 2 чънка и след това ще използваме предсказаната стойност, като основа за кластеризация. За характеристики ще бъдат използвани `word_count`, `rank_distance`, `levestain_distance`, `central_star distance`, `local_align_pred` и `global_align_gap_pred`, а вече изчислената стойност за глобално подравняване ще бъде „таргет“ променливата. Алгорътъма, който ще се използва е `XGBRegressor`.

*Всички регресори са обучени на база изчисленията за tuberculosis -Covid19, tuberculosis-HIV и SARS-Covid19, като се използват 80 процента от данните за обучение и 20 процента за тестване. Също за да осигурим примери в които имаме 100 процента сходство ще тренираме и върху HIV-HIV(за тази двойка има 100% сигурност, че сходствата имат биологическо значение). Отчитането на резултатите ще бъде извършено на база HIV-COVID19. Преди да бъдат обучени регресорите характеристиките бяха обработени със `StandardScaler`. Самите регресионни*

алгоритми бяха тюннати с помощта на *GridSearchCV*. Заради обема на данните с който се борави ще се наложи използването на трансферно обучение. За него ще се използва библиотеката *joblib*.

5. За клъстеризирането ще се използва метода *Kmeans++* с  $k=5$ ., като намирането на оптималната стойност за  $k$  става, чрез *pyeasyga*(генетичен алгоритъм)

## 8 ИЗМЕРВАНЕ НА РЕЗУЛТАТИТЕ НА БАЗА ТЕСТОВИ ДАННИ

Резултатите получени от обучението на регресора за предричане на скор за локално подравняване:

```
explained_variance: 0.4969
r2: 0.4969
MAE: 9.0527
MSE: 130.4205
RMSE: 11.4202
```

Резултатите получени от обучението на регресора за предричане на броя гапове при глобално подравняване:

```
explained_variance: 0.3894
r2: 0.3894
MAE: 4.3684
MSE: 31.0613
RMSE: 5.5733
```

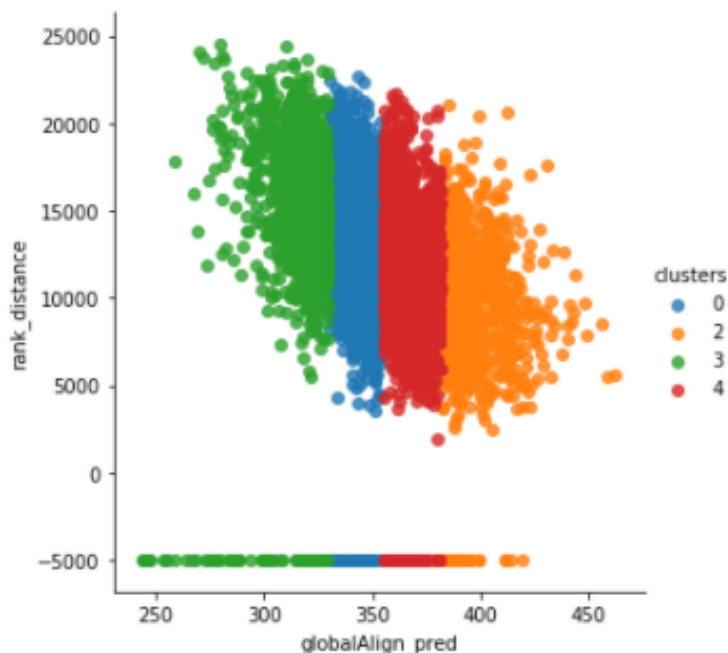
Резултатите получени от обучението на регресора за предричане на скор при глобално подравняване:

```
explained_variance: 0.6177
r2: 0.6177
MAE: 8.9864
MSE: 127.9803
RMSE: 11.3128
```

Резултати получени от клъстеризацията :

```
ami_score: 0.20445195918503672
compl_score: 0.23264538816868519
adj_rand_score: 0.1442579400245844
calinski_harabasz_score: 165305.73366573217
davies_bouldin_score: 0.4426093151848976
fowlkes_mallows_score: 0.44074599968757094
homogeneity_score: 0.20451743677368817
mutual_info_score: 0.25996734029650836
silhouette_score: 0.5039693117141724
```

Самите данни бяха разделени на 15 клъстера, както може да се види на Фигура 22.



Фигура 21: Визуално разпределение на резултатите от клъстеризирането на база на тестовия сет

По детайлно отделните клъстери съдържат следните характеристики:

-клъстер 0:

```
големина на клъстер 0:1932429 примера  
максимален скор в клъстер 0:72.23065948486328  
минимален скор в клъстер 0:65.3995132446289
```

-клъстер 1:

```
големина на клъстер 1:3343746 примера  
максимален скор в клъстер 1:99.68022918701172  
минимален скор в клъстер 1:94.48251342773438
```

-клъстер 2:

```
големина на клъстер 2:1755892 примера  
максимален скор в клъстер 2:118.6746826171875  
минимален скор в клъстер 2:112.0467758178711
```

-клъстер 3:

големина на клъстер 3:4011601 примера  
максимален скор в клъстер 3:89.27458190917969  
минимален скор в клъстер 3:83.58536529541016

-клъстер 4:

големина на клъстер 4:61619 примера  
максимален скор в клъстер 4:13.0  
минимален скор в клъстер 4:-46.0

-клъстер 5:

големина на клъстер 5:3475692 примера  
максимален скор в клъстер 5:105.59324645996094  
минимален скор в клъстер 5:99.68256378173828

-клъстер 6:

големина на клъстер 6:373480 примера  
максимален скор в клъстер 6:55.634788513183594  
минимален скор в клъстер 6:14.0

-клъстер 7:

големина на клъстер 7:3371512 примера  
максимален скор в клъстер 7:83.58480834960938  
минимален скор в клъстер 7:77.86998748779297

-клъстер 8:

големина на клъстер 8:800881 примера  
максимален скор в клъстер 8:138.29437255859375  
минимален скор в клъстер 8:126.65636444091797

-клъстер 9:

големина на клъстер 9:3052477 примера  
максимален скор в клъстер 9:112.04222869873047  
минимален скор в клъстер 9:105.59339141845703

-клъстер 10:

големина на клъстер 10:229407 примера  
максимален скор в клъстер 10:379.0  
минимален скор в клъстер 10:138.37582397460938

- клъстер 11:

големина на клъстер 11:2130939 примера  
максимален скор в клъстер 11:77.8692626953125  
минимален скор в клъстер 11:72.23479461669922

- клъстер 12:

големина на клъстер 12:3503658 примера  
максимален скор в клъстер 12:94.48140716552734  
минимален скор в клъстер 12:89.2755355834961

- клъстер 13:

големина на клъстер 13:1194494 примера  
максимален скор в клъстер 13:65.39480590820312  
минимален скор в клъстер 13:55.65174865722656

- клъстер 14:

големина на клъстер 14:1363776 примера  
максимален скор в клъстер 14:126.56180572509766  
минимален скор в клъстер 14:118.68519592285156

Трябва да се има предвид, че най-добрия скор, който може да бъде постигнат за глобално подравняване при използваните параметри е 379 (ако са абсолютно еднакви двете секвенции), а най-лошия -46. Както виждате най-добри резултати имаме в клъстер 10, а най-лоши в клъстер 4. Като цяло възможна грешка може да се получи, защото границите в които попадат примерите в клъстер 10 са много широки. Това е така тъй като имаме твърде малко примери с голямо сходство. Този проблем може да бъде решен, като се добавят повече положителни примери и /или разделим дейта сета на повече клъстери.

Втората компонента на която трябваше да се наблегне е времето за изпълнение. Времето което ни беше нужно за да изчислим нужните ни данни за дейта сета ХИВ -Ковид19 беше 651. 64 секунди, а това за клъстеризация-6. 71 секунди, тоест общо 11 минути.



Вече знаем, че Ковид и ХИВ секвенциите нямат значимо биологическо сходство от изследването ни с BLAST,но все пак имат някакво сходство -показано от Clustal Omega.

В такъв случай очакваме примерите, които ще проверим да попадат в някой от средните по скор клъстери. За да потвърдим това ще избира 5 примера на случаен принцип и ще видим в кой клъстер попадат при нас. Данните подчертани с жълто са извадките, на база на които ще търся, а под тях където е код сегмента (с него илюстрирам как точно намирам в кой клъстер са тези чънкове) с червено е отбелязано в кой клъстер попадат.

```
testM=X_test_hc_r[X_test_hc_r['chunk_or1'].str.contains('ACAGACGTTTTTCAGTGTAGCTTGTTCACATGGTTCA', regex=False)]
testA=testM[testM['chunk_or2'].str.contains('AGTGCTTCAAGTAGTGTGTGCCCCGTC', regex=False)]
testA
#testM
```

NC_045512.2	TGGACAATTCACCTAATTTAGCATGGCCTCTTATTGTAACAGCTTTAAGGGCCAATTCTG	12671
NC_001802.1	CATACAATACTCCAGTATTTGCCATAAAGAAAAAAGACAGTACTAAATGGAGAAAATTAG	2318
	. ***** . : * : * : * : * : * : * : * : * : * : * : * : * : *	

```
testM-X_test_hc r[X_test_hc r['chunk_or1']].str.contains('TGGACAATTACCTAATTAGCATGGCCCTTTATTGTAAACGCTTTAAG', reg
testA=testM[testM['chunk_or2']].str.contains('TTTGCCATAAAGAAAAAGACAGTACTAAATGGAGAAAAATTAG', regex=False)]
testA
#testM
```

NC_045512.2	GATGT-TGACACAGACTTTG-TGAATGAGTTTTACGCATAATTTCGGTAAACATTTCTCAA	15703
NC_001802.1	AAAAAGTGTGCTTTTCATTGC-CAAGTTTGTTTCATAACAAAAGCCTTAGGCATCTCCTAT	5517
	*.: *: *: *: ***** * *:***** * *:***** * *:***** *	

```
testM=X_test_hc_r[X_test_hc_r['chunk_or1'].str.contains('TGAATGAGTTTTACGCATATTTGCGTAACATTTCTCAA', regex=False)]
testA=testM[testM['chunk_or2'].str.contains('AAGTTTGTTTCATAACAAAACCTTAGGCATCTCCTAT', regex=False)]
testA
#testM
```

NC\_045512.2 TATTCTACACTCCAGGGACCACCTGGTACTGGTAAGAGTCATTTTGCTAT----TGGCCT 17120  
NC\_001802.1 TAAGCAATCCTCAGGAGGGGCCAGAAATTGTAAAGCACAGTTTTAATTGTGGAGGGGA 6912

\*\*\*:

```
testM=X_test_hc_r[X_test_hc_r['chunk_or1'].str.contains('TATTCTACACTCCAGGGACCACCTGGTACTGGTAA', regex=False)]
testA=testM[testM['chunk_or2'].str.contains('TAAGCAATCCTCAGGAGGGGACCCAGAAATTGTAA', regex=False)]
testA
```

clusters	chunk_or1	chunk_or2	global_align_pred
43850	12	T, A, T, C, A, A, A, A, G, G, T, T, G, G, T, ... (A, A, C, A, A, T, A, A, T, C, T, T, T, A, A, ...	91.39772

```
NC_045512.2      TCTGACAAATTACAGATGGTGTATGCCTATTTTGGAAATTGCAATGTCGATAGATATCCT      19218
NC_001802.1      GACATCGAGCTTGCTACAAGGGACITTTCCGCTGGGGACITTTCCAGGGAGG-CGTGGCCTG      9020
                  . .:*. * .*:...* *: * . * **.* *.* * .*: . *
```

```
testM=X_test_hc_r[X_test_hc_r['chunk_or1'].str.contains('TCTGACAAATTACAGATGGTGTATGCCTATTTTGGAA', regex=False)]
testA=testM[testM['chunk_or2'].str.contains('TCGAGCTTGCTACAAGGGACITTTCCGCTGGGGAC', regex=False)]
testA
```

clusters	chunk_or1	chunk_or2	global_align_pred
49275	1	(T, A, T, T, C, T, T, A, T, G, C, C, A, C, A, ... (C, G, G, A, G, T, A, C, T, T, C, A, A, G, A, ...	96.186363

Както очаквахме, данните попадат в среден по скор сегмент.

## 10 ЗАКЛЮЧЕНИЕ

Генетичните изследвания, поради комплексността си и обема на данните, са една от насоките в които изкуствения интелект може да разкрие потенциала си. Един бърз алгоритъм за намиране на сходство между генетични секвенции, дори и не изключително точен, може да ни даде достатъчно знания на база на които да се вземе решение дали да извърши по обстойно проучване по-темата или не, както и да ни даде поне първоначална (макар и непълна) информация при поява на нов вирус. Това беше причината поради, която избрах тази дипломна работа.

Смятам, че получих задоволителни резултати и по отношение на точност и по отношение на време. Това което смятам, че може да се подобри е съотношението на данните участващи в клъстерирането (тоест да имаме повече и по-различни, като предсказан скор за глобално подравняване, примери на високо сходство). Освен това би било хубаво да се пробва да се намери оптималното k за клъстериране в по-голям рейндж и то когато клъстерираме с пълния набор от данни. Освен това може да се работи по оптимизирането на времето, като се намали броя на характеристиките и/или се увеличи големината на чънковете, които използваме, ако разбира се това не намали точността.

## 11 СПИСЪК НА ФИГУРИТЕ В ТЕКСТА

Фигура 1: Пример за следващо поколение секвениране (NGS) сурови данни-BRAF V600E мутация при меланом. Мутацията е открита през 2002 г. като част от няколко годишни усилия за дефиниране на соматични мутации при човешки рак, използвайки секвениране .....	4
Фигура 2: Илюстрация на Dot Plot Matrix .....	6
Фигура 3: mean_shift_points .....	10
Фигура 4: KDE изображение на повърхнината .....	11
Фигура 5: KDE контурно изображение .....	11
Фигура 6: Обща схема на работата на генетичния алгоритъм .....	12
Фигура 7: One Point Crossover .....	13
Фигура 8: Multi Point Crossover .....	13
Фигура 9: Uniform Crossover .....	13
Фигура 10: Bit Flip Mutation .....	14
Фигура 11: Swap Mutation.....	14
Фигура 12: Scramble Mutation .....	14
Фигура 13: Inversion Mutation .....	15
Фигура 14: Примерна архитектура за deep clustering .....	17
Фигура 15: Структура на модел за дълбоко клъстериране.....	19
Фигура 16: Резултати получени при обучението и тестването на аутоенкодер .....	20
Фигура 18: Визуално разпределение на резултатите от клъстерирането .....	28
Фигура 19: Резултати взети от клъстер омега показващи сходство между Ковид19 и ХИВ.....	30
Фигура 20: Резултати получени при използването на Blast за определяне на сходство между ХИВ и КОВИД .....	32
Фигура 21: Схема на алгоритъма, който ще бъде използван за финалната имплементация .....	34
Фигура 22: Визуално разпределение на резултатите от клъстерирането на база на тестовия сет .....	37

## 12 ИЗПОЛЗВАНИ ЛИТЕРАТУРНИ ИЗТОЧНИЦИ В БЕЛЕЖКИ В КРАЯ НА ДОКУМЕНТА

<sup>i</sup> Уеб-страница:Conduct and Interpret a Cluster Analysis извлечено на 26/01/2018 г.

<https://www.statisticssolutions.com/cluster-analysis-2/>

<sup>ii</sup> Уеб-страница:Introduction to NGS извлечено на 26/01/2018 г.

<https://www.illumina.com/science/technology/next-generation-sequencing.html>

<sup>iii</sup> Статия: What is next generation sequencing? [Sam Behjati](#)<sup>1,2</sup> and [Patrick S Tarpey](#)<sup>1</sup> извлечено на 26/01/2018 г. от <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3841808/>

<sup>iv</sup> Статия:Homology, similarity, and alignment, Roman Cheplyaka от 01. 05. 2019 -<https://ro-che.info/articles/2019-01-05-homology-similarity-alignment>

- 
- <sup>v</sup> Веб-страница: <https://gtbinf.wordpress.com/biol-41506150/pairwise-sequence-alignment/> извлечено на 26. 01. 2021г.
- <sup>vi</sup> Фигура: Dot Plot Illustration взета от "Lecture 2 Pairwise sequence alignment. Principles Computational Biology by Teresa Przytycka, PhD" извлечена на 26. 01. 2021г.
- <sup>vii</sup> Веб-страница: <https://gtbinf.wordpress.com/biol-41506150/pairwise-sequence-alignment/> извлечено на 26. 01. 2021г.
- <sup>viii</sup> Веб-страница: <https://gtbinf.wordpress.com/biol-41506150/pairwise-sequence-alignment/> извлечено на 26. 01. 2021г.
- <sup>ix</sup> Веб-страница: Mean Shift Clustering MATT NEDRICH извлечено на 28/12/2018 г.  
<https://spin.atomicobject.com/2015/05/26/mean-shift-clustering/>
- <sup>x</sup> Веб-страница: Introduction to Genetic Algorithms [Vijini Mallawaarachchi](#) извлечено на 28/12/2018 г.  
<https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>
- <sup>xi</sup> Веб-страница: Genetic Algorithms - Crossover Tutorialspoint извлечено на 28/12/2018 г.  
[https://www.tutorialspoint.com/genetic\\_algorithms/genetic\\_algorithms\\_crossover.htm](https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm)
- <sup>xii</sup> Веб-страница: Genetic Algorithms - Mutation Tutorialspoint извлечено на 28/12/2018 г.  
[https://www.tutorialspoint.com/genetic\\_algorithms/genetic\\_algorithms\\_mutation.htm](https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm)
- <sup>xiii</sup> Веб-страница: [Affinity Propagation Algorithm Explained](#) извлечено на 31/10/2020г.
- <sup>xiv</sup> Веб-страница: <https://deepnotes.io/deep-clustering#direct-cluster-optimization> извлечена на 01. 30. 2021г.
- <sup>xv</sup> Genomic signal processing for DNA sequence clustering by Gerardo Mendizabal-Ruiz, Israel Román-Godínez, Sulema Torres-Ramos, Ricardo A. Salido-Ruiz, Hugo Vélez-Pérez and J. Alejandro Morales Departamento de Ciencias Computacionales, Universidad de Guadalajara, Guadalajara, Mexico published 24 January 2018
- <sup>xvi</sup> A diagnostic genomic signal processing (GSP) -based system for automatic feature analysis and detection of COVID-19 Safaa M. Naeem, Mai S. Mabrouk, Samir Y. Marzouk and Mohamed A. Eldosoky Corresponding author: Mai S. Mabrouk from Misr University for Science and Technology (MUST University) published 14 August 2020
- <sup>xvii</sup> статия: Clustering Methods based on Closest String via Rank Distance Liviu P. Dinu and Radu-Tudor Ionescu Department of Computer Science University of Bucharest
- <sup>xviii</sup> статия: 2012 International Conference on Medical Physics and Biomedical Engineering A Novel Center Star Multiple Sequence Alignment Algorithm Based on Affine Gap Penalty and K-Band автори: Quan Zou, Xiao Shan, Yi Jiang\* извлечена на 1/11/2020г.
- <sup>xix</sup> сайт:  
<https://techoverflow.net/2013/12/08/center-star-approximation-identifying-the-center-string-in-python/>  
извлечен на 1/11/2020г.
- <sup>xx</sup> извлечено от [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted\\_mutual\\_info\\_score.html#sklearn.metrics.adjusted\\_mutual\\_info\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html#sklearn.metrics.adjusted_mutual_info_score) на 24. 11. 2020
- <sup>xxi</sup> Статия [A Quick and Dirty Guide to Random Forest Regression](#) написана от [Ashwin Raj](#) извлечен на 06. 02. 2021г.
- <sup>xxii</sup> веб-страница [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) извлечена на 06. 02. 2021г.
- <sup>xxiii</sup> веб-страница [https://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](https://scikit-learn.org/stable/modules/neural_networks_supervised.html) извлечена на 06. 02. 2021г.
- <sup>xxiv</sup> веб-страница: <https://www.datatechnotes.com/2019/06/regression-example-with-xgbregressor-in.html> извлечено на 06. 02. 2021