

**Софийски университет ,,Св. Климент Охридски”, Факултет по математика и информатика**

Курсов проект

по Приложения на бази от данни в биоинформатиката

*Тема: Интегриране на данни от ракови заболявания – Breast Cancer Data*

Изготвено от: Елия Асенова, ФН 26131

Table of Contents

[Описание на задачата 3](#_Toc45231967)

[Избор на тема 4](#_Toc45231968)

[Решение 4](#_Toc45231969)

[1. Имплементиране на скрипт за популиране на данните в базата от данни 4](#_Toc45231970)

[2. Имплементиране на основните CRUD операции 5](#_Toc45231971)

[3. Предсказване на преживяемостта на пациенти с Breast cancer 6](#_Toc45231972)

[4. Допълнителен endpoint за предсказване на преживяемостта спрямо подадените данни 10](#_Toc45231973)

[Заключение 10](#_Toc45231974)

# Описание на задачата

Да се разгледат и интегрират данните за едно раково заболяване от [camda.info](http://camda2020.bioinf.jku.at/doku.php/contest_dataset#hi-res_cancer_data_integration_challenge) . Предизвикателството за интегриране на данни "Hi-Res" на рак предоставя клинични и съвпадащи молекулярни профили с данни за отделни геноми на три различни вида рак:

* Рак на гърдата (Breast cancer)
* Аденокарцином на белия дроб (Lung Adenocarcinoma)
* Бъбречно-бъбречно прозрачен карцином (Kidney Renal Clear Cell Carcinoma)

Данните в проекта са копие на публични данни, който са офускирани.

Има данни за RNA-Seq gene expression, microRNA expression, copy number data, and methylation в следните файлвое:

* cnv\_segments\_download\_xxx\*
* masked\_cnv\_segments\_download\_xxx\*
* methylation\_gdc\_download\_xxx\*\* - този тип данни не е задължително да бъде интегриран поради голмия си обем, който налага използването на много допълнително сървърно пространство.
* mirnas\_gdc\_download\_xxx
* clinical.cart.xxx

Задачата на проекта е да се създаде приложение (RESTFul API) за работа с интегрираните данни, което да позволява оснивните операции CRUD (create, read, update, delete) и възможност за изпълнение на **една** от следните задачи (Може да не се ограничавате само до тези две под - точки):

* Да се предостави възможност за предсказва преживяемостта на пациенти болни от избраното заболяване на база на интегрираните данни. Да се реализира допълнително API към CRUD операцийте, което да ползволява да се въвеждат избраните критерии за предсказване на преживяемостта на пациента.
* Да се намерят препокриващи се пациенти, който имат сходни мутации, нива на експресия и др.. Да се реализира допълнително API към CRUD операцийте, което да ползволява да се въвеждат избраните критерии за търсене на съвпадения.

За интегрирането на данните и изграждането на **подходящ модел на данните** е необходимо да се вземе предвид тяхната нехомогенност и обемности.

# Избор на тема

Избраните данни от изброените са тези за Breast Cancer. Като допълнителна задача се разглежда задачата за предсказване на преживяемостта на пациенти болни от рак на гърдата. Поради тази причина, всички данни са вкарани в една база от данни, но за предсказването са използвани само клиничните данни.

# Решение

## Имплементиране на скрипт за популиране на данните в базата от данни

Данните за Breast Cancer са разпределени по папки и файлове. Описанието на клиничните данни се намира във файл “clinical.tsv”, а описанието на структурата на останалите данни е във файл “gdc\_sample\_sheet.tsv”. Всеки пациент се асоциира с “caseId” и за съответното “caseId” е дадена информация за файловете свързани с него - “FileID” и “FileName”, които съдържат различни данни за този пациент. Имплементиран е скрипт “mongodb.py” за популиране на данните от файловете в базата от данни MongoDB. Данните се събират в една обща база от данни breast\_cancer и в нея се създават различните колекции за различните типове данни: Copy Number Variation, Masked Copy Number Variation, clinical, miRNA expression. Скриптът чете първо клиничните данни, като използва функцията за четене на tsv файлове и след това ги записва в БД. След това се обработва файлът, който описва останалите данни. За прочитане на този файл отново се използва функцията за четене на tsv файлове като за всеки ред от този файл се извличат съответните данни: File ID, File Name, Data Category, Data Type, Case ID. Спрямо прочетената категория на данните се използва функция, която съпоставя категорията със съответстващото име на папката, в която се намират данните. С извлечените данни се изгражда целият път до различните файлове по папките и данните се записват в базата. Като допълнение към всеки запис се добавя caseId-то на всеки пациент с цел по – лесно намиране на всички данни за един пациент по неговия идентификационен номер. Съответно при търсене на данни за определен пациент е достатъчно да подадем caseId-то му и ще получим всички документи от избраната колекция, които са свързани с този пациент.

Тъй като за решаване на задачата се използват клиничните данни, върху тях не са правени допълнителни промени освен това, че пациентите с липсващи данни за “age\_at\_diagnosis”, “tumor\_stage” или ”days\_to\_death” са изтрити, защото могат да доведат до грешни резултати генерирани от модела.

## Имплементиране на основните CRUD операции

Имплементацията на основните операции на REST API-то – Create, Read, Update и Delete е направена във скрипт “api.py”. Създадени са четири основни endpoint-a: за създаване на нов запис за пациент, за извличане на запис за пациент по подадено “caseId”, за обновяване на запис за пациент по “caseId” и за изтриване на всички записи на пациент с определено “caseId”.

* Създаване на нов запис за пациент – */patient/datatype/<datatype>*, POST рекуест

За създаване на запис за пациент се подава параметър datatype и body, което трябва да съдържа данни за пациента, които ще се запишат в базата. Параметърът datatype представлява типът на данните или по – точно може да приема 4 възможни стойности: “Copy Number Variation", "Masked Copy Number Variation", "clinical", "miRNA Expression". Ако е подадена различна стойност, в response-a се връща съобщение, че подаденият datatype е невалиден като се подават и валидните стойности на datatype-a. Ако не е подадено никакво body с информация за пациента се връща съобщение за невалиден рекуест поради празно body. При подадено body се прави проверка дали рекуеста съдържа задължителната информация, която се изисква за един пациент преди данните му да бъдат записани в базата от данни. В зависимост от типа на данните, задължителните полета са различни:

mandatoryFields = {"Copy Number Variation": "Case ID,Chromosome,Start,End,Num\_Probes","Masked Copy Number Variation": "Case ID,Chromosome,Start,End,Num\_Probes","clinical": "submitter\_id,tumor\_stage,age\_at\_diagnosis,vital\_status","miRNA Expression": "Case ID,miRNA\_ID,read\_count"}

Тези полета могат да се променят в зависимост от нуждите към API-то. Ако задължителните полета не са подадени в body-то на рекуеста се връща съобщение, че липсва съответното поле като се подават и всички задължителни полета.

При успешен рекуест, данните от него се записват в базата от данни в колекцията, която съответства на съответния datatype и се връща съобщение за потвърждение на успешния рекуест.

* Извличане на запис за пациент по подадено caseId – */patient/datatype/<datatype>/caseId/<caseId>,* GET рекуест

За извличане на запис за пациент се подава datatype (както при предишния endpoint) и caseId, което е id-то на пациента. При грешен datatype отново се връща съобщение за грешка. Ако подаденото id не съществува в базата, тоест няма данни за него се връща съобщение, че подаденото id не съществува. В зависимост от нуждите на клиентите, данните за пацинтите може да се извличат освен по caseId и по други налични характеристики в зависимост от datatype-a.

* Обновяване на запис за пациент по caseId - */patient/datatype/<datatype>/caseId/<caseId>,* PUT рекуест

За обновяване на запис за пациент по caseId се подават същите параметри както и при извличане на запис – datatype и caseId. Правят се същите проверки за валиден datatype и за подадено празно body. При подаване на коректни данни, ако данните са от тип “clinical” тогава заявката към mongoDB е с параметър “submitter\_id”. Това е същото id като caseId, но при клиничните данни е под различно име. Съответно ако данните са от някой от другите типове, то заявката към mongoDB е с caseId. По същия начин работи и когато изпращаме рекуест за извличане на данни.

* Изтриване на всички записи на пациент с определено caseId - */patient/datatype/<datatype>/caseId/<caseId>,* DELETE рекуест

За изтриване на всички записи се подават същите параметри – datatype и caseId и се прави същата проверка за валидност на datatype-a. Ако данните са клинични се търси по параметър “submitter\_id” иначе по “caseId” за останалите типове данни.

Отново спрямо нуждите на клиента може да се изтриват данни спрямо различни подадени критерии за пациента. Също така може да се изтрива само един запис с подаденото caseId, не всички както в този случай.

## Предсказване на преживяемостта на пациенти с Breast cancer

Решението на задачата за предсказване на преживяемостта се намира във файл “svm.py”. Приложени са два подхода, които се разглеждат подробно. Алгоритъмът, който се използва за решаване на задачата е Support Vector Machine. Той е supervised machine learning алгоритъм, който може да се използва както за класификация, така и за регресия. Следва техника, наречена “kernel trick” за преобразуване на данните и въз основа на тези трансформации, той намира оптимална граница между възможните изходи.

В клиничните данни има много на брой променливи, от които могат да се изберат предикторните променливи за решаване на проблема. В този случай са избрани две променливи “age\_at\_diagnosis” и “tumor\_stage”. Първата променлива се измерва в дни и представлява възрастта на индивида, когато е диагностициран. Тази възраст е от значение, тъй като може да има разлика в състоянието пациента ако той е на 20 или на 70 години, когато е поставена диагнозата. Втората променлива е стадий на тумора. Тя е от значение, тъй като в зависимост от стадия се прави заключение за големината на тумора и неговото разпространение. Колкото по – напреднал е стадият, толкова по – разпространен е той и толкова по – опасно е за живота на пациента. Това го прави и по – трудно лечимо. Тези два параметъра са от голямо значение за преживяемостта на пациента. Зависимата променлива или тази, която трябва да предскажем е “days\_to\_death”. Веднага се забелязва, че “days\_to\_death” е дадена само за вече починалите пациенти, тоест тези с “vital\_status=dead”, а тези данни са нужни, за да обучим алгоритъма да предсказва преживяемостта. Поради тази причина са избрани два подхода: първият използва само данните от пациентите, които са починали, а вторият се комбинира с още една колона към “days\_to\_death” – “days\_to\_last\_follow\_up”, за да се получат повече данни, които да използваме за обучаване за нашия алгоритъм.

* Подход 1: Две независими променливи “age\_at\_diagnosis” и “tumor\_stage” и една зависима “days\_to\_death” само за починалите пациенти.

Създаден е клас Model(), който съдържа в себе си създаването на модела, изчисляването на точността на модела и изчисляването на Root MSE – средната квадратична грешка.

Как се създава моделът? Променливата “tumor\_stage” е категорна променлива и стойностите й са във формата на стрингове. По този начин моделът няма да може да ги интерпретира за това трябва да се превърнат в числови данни. Поради тази причина създаваме един LabelEncoder, който задава поредни числа на туморните стадий, тоест цифрите съответстват по този начин:

'not reported' = 0

'stage i' = 1

'stage ia' = 2

'stage iia' = 3

'stage iib' = 4

'stage iii' = 5

'stage iiia' = 6

'stage iiib' = 7

'stage iiic' = 8

'stage iv' = 9

'stage x' = 10

Използваме този encoder, тъй като той ни степенува по сила стадиите, което е полезно за нашия модел, защото той трябва да разбира, че стадий 5 е по – напреднал от стадий 1 например. От това получаваме нова колона “tumor\_stage\_cat” вече с цифри вместо имена на стадии. След това разделяме променливите на features и target. Първият вид са предикторните променливи, а вторият е променливата, която ще предсказваме. Следва да разделим данните на train и test. Train данните се използват за обучаване на модела, той взима тях за пример, когато прави решения за нови подадени данни, по които трябва да върне survival time. Test данните са тези, които ще подадем след като обучим модела, за да изчислим колко добре се справя той. Много важна стъпка от изграждането на модела е нормализирането на данните. Нормализират се предикторните променливи, така че да се приведат към обща скала и съответно резултатите от модела да бъдат коректни. Има много начини за нормализация. Този, който е използван е StandardScaler(), който взима средното и стандартното отклонение на всяка характеристика и коригира скалата им, за да бъде между -1 и 1 и със средно 0. Важно е да се отбележи, че scaler-a го моделираме с данните за обучение, тъй като искаме спрямо средното и стандартното отклонение получено от train данните да трансформираме тестовата група. В противен случай ще направим две различни трансформации като вземем две различни средни и две различни стандартни отклонение за данни, които реално трябва да бъдат едни и същи.

Алгоритъмът, който използваме е SVC или Support Vector Classification. Има няколко на брой SVM алгоритъма, този даде най – добра точност с подадените данни. Той е класификационен алгоритъм и се използва, когато данните са по – малко от 10000. Параметрите, които сме подали са kernel=”poly” и C=”0.2”. Параметърът C показва до колко алгоритъмът да се адаптира към подадените му данни. Параметърът kernel определя дали SVM моделът използва линия или крива, за да отгатне класа или мярката на точката. Избраният kernel използва крива. След като подадем на вече създадения модел train данните на независимите и на зависимата променлива, можем да направим предсказвания по тестовите данни. За направените предсказвания се прави оценка на точността на модела и на Root MSE. Точността на модела не е много добра – 61.24 процента и Root MSE е около 1896, което е голямо число.

SVR алгоритъмът, който е за регресия също беше приложен, но с него резултатите не са толкова добри. Това вероятно се дължи на малкия брой данни (около 150), от което получаваме, че класификацията е по – добрият вариант тук.

* Подход 2: Две независими променливи “age\_at\_diagnosis” и “tumor\_stage” и една зависима “days\_to\_death\_combined” за всички пациенти.

Разликата тук е използването на новата колона “days\_to\_death\_combined”. Как я получаваме? За изграждане на модела освен “days\_to\_death” се използва и колоната “days\_to\_last\_follow\_up”. Тя описва кога е било последното посещение на пациента от времето на диагностицирането му в дни. За да използваме всички данни взимаме нея и “days\_to\_death” и ги обединяваме, така че да получим данни за survival\_time-a на всички пациенти и да можем да тренираме модела си с тях. Новата колона я получаваме при следните условия: Тъй като на места за даден пациент има данни и в “days\_to\_death” колоната и в “days\_to\_last\_follow\_up”, за всички починали пациенти взимаме само “days\_to\_death”. Това решение е на базата, че “days\_to\_death” е реалното число, което е измерено след като човекът е починал, докато “days\_to\_last\_follow\_up” е времето от последният му преглед, а не survival time-a на човека, тоест в случая е по – добре да изберем реалните данни. За това и предпочитаме “days\_to\_death”, когато ги имаме налични. За останалите пациенти взимаме “days\_to\_last\_follow\_up”.

Начинът на моделиране е същият както при първия подход, като тук се използва SVR, който е регресионен алгоритъм, с подадени параметри kernel=”rbf”, C=1000, gamma=5, epsilon=10. Root MSE на алгоритъма е около 1151, което е голямо число, но по – малко от това на предишния алгоритъм. Тук данните са много повече и SVR алгоритъма се справя много по – добре от SVC. Грешката е възможно да идва от неправилно построен модел или от това, че не можем да разчитаме изцяло на “days\_to\_last\_follow\_up”, тъй като тези данни не е сигурно, че представят точния survival time-a на един пациент, което вероятно води до объркване на алгоритъма и съответно на предсказаните данни.

## Допълнителен endpoint за предсказване на преживяемостта спрямо подадените данни

За предсказване на преживяемостта е създаден нов endpoint, който по подаден модел – 1(SVC) или 2(SVR), ageAtDiagnosis – възраст, когато е поставена диагнозата и tumorStage – стадий на тумора връща предсказаната стойност на преживяемост в дни, както и Root MSE на използвания модел. Тъй като се работи в дни, ageAtDiagnosis се умножава по броя дни за една година, за да се превърнат годините в дни. При подаване на модела се валидира, че стойността е 1 или 2 и при подаване на tumorStage-a се валидира, че стойността е между 0 и 10. Преди връщане на резултата подадените данни се нормализират, за да се трансформират по същия начин както данните, с които е обучен алгоритъма. Накрая се връща предсказаният survival time.

# Заключение

Получената преживяемост и от двата модела не е добра и двата алгоритъма не връщат надеждни резултати. Сравнени един с друг обаче, вторият алгоритъм се справя доста по – добре, което е повлияно от повечето данни, с които работи. Въпреки това и при него се получава голяма грешка, която може да се дължи на неправилно построен модел или неточни или недостатъчни данни.