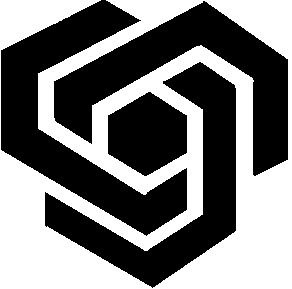
**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**Факултет Приложна математика и информатика**

**Курсов проект**

по

***Анализ на големи данни за прецизната медицина***

**Тема:** Рак на простатата

**Студенти:**

Есин Рашид, **Фак. №** 791322007

Кристина Герчева, **Фак. №** 791322010

Станислав Стоянов**, Фак. №** 791322012

**Специалност:** ИСН

Съдържание

**No table of contents entries found.**

# Обща информация за заболяването с фокус върху социалната му значимост в световен и национален мащаб.

Ракът на простатата е вид рак, който се развива в простатната жлеза, която е отговорна за производството на част от течността в спермата. Това може да бъде сериозно заболяване и докато някои случаи се развиват бавно и може да не изискват незабавно лечение, други видове могат да бъдат агресивни и да се разпространят бързо в други части на тялото.

Diagram

Description automatically generated

Фигура 1 - нормална простата и рак

Рисковите фактори за рак на простатата включват възраст, фамилна анамнеза, раса/етническа принадлежност и определени генетични мутации. Мъжете над 50-годишна възраст са изложени на по-висок риск от развитие на рак на простатата, като рискът нараства с възрастта. Мъжете с фамилна анамнеза за рак на простатата, особено ако техният баща или брат са имали заболяването, също са изложени на по-висок риск. Афро-американските мъже са по-склонни да развият рак на простатата, отколкото мъжете от други раси/етноси, и те също са по-склонни да развият агресивни форми на заболяването.

Симптомите на рак на простатата могат да включват затруднено уриниране, слаб поток на урина, кръв в урината или спермата, болка или дискомфорт в областта на таза и болка в костите. Въпреки това, в някои случаи ракът на простатата може да не причини никакви симптоми, докато не премине в по-късен стадий.

Диагнозата на рак на простатата обикновено включва комбинация от физически преглед, кръвни изследвания и образни тестове, като ултразвук или ЯМР. Обикновено е необходима биопсия за потвърждаване на диагнозата.

Възможностите за лечение на рак на простатата могат да включват операция, лъчева терапия, хормонална терапия и химиотерапия. Изборът на лечение зависи от различни фактори, включително стадия и степента на рака, възрастта и общото здравословно състояние на пациента и личните предпочитания на пациента. В някои случаи може да се препоръча подход на „бдително изчакване“, който включва внимателно наблюдение на рака, но не и незабавно лечение.

Въпреки наличието на ефективни възможности за лечение, ракът на простатата остава сериозен проблем за здравето, особено за по-възрастните мъже. В допълнение към физическото и емоционалното въздействие на болестта, ракът на простатата може да има и финансови последици за пациентите и техните семейства. Цената на лечението на рак на простатата може да бъде значителна и може да доведе до финансови затруднения за някои пациенти.

В национален и глобален мащаб ракът на простатата има значителни социални и икономически последици. В България тежестта на рака на простатата е висока, като всяка година се диагностицират голям брой нови случаи. Това натоварва здравната система и може да има значително въздействие върху икономиката на страната. Освен това ракът на простатата може да окаже значително влияние върху живота на пациентите и техните семейства, както и върху способността им да работят и да допринасят за обществото.

Цената на лечението на рак на простатата може да бъде значителна, като оценките варират от $10 000 до $100 000 или повече в зависимост от вида и стадия на рака и подхода на лечение. В Съединените щати ракът на простатата е един от най-скъпите за лечение ракови заболявания, като разходите за здравеопазване възлизат на около 12 милиарда долара годишно. (източник: Национален институт по рака)

Ракът на простатата е най-често диагностицираният рак при мъжете по света, представляващ 26% от всички случаи на рак при мъжете. През 2020 г. беше изчислено, че в световен мащаб са диагностицирани 1,4 милиона нови случая на рак на простатата и 375 000 смъртни случая от болестта. (източник: Световен фонд за изследване на рака)

Chart, pie chart

Description automatically generated

Фигура 2 - брой нови случаи за 2020г. в световен мащаб, всякаква възраст

Честотата на рак на простатата варира значително в зависимост от региона, като най-високите нива се наблюдават в Северна Америка, Западна Европа и Австралия, а най-ниските - в Азия и Африка. (източник: Световен фонд за изследване на рака)

Chart, pie chart

Description automatically generated

Фигура 3 - брой нови случаи за 2020г. по региони, всякаква възраст

Петгодишната преживяемост при рак на простатата е висока, като глобалната средна стойност е 84% сред мъжете, диагностицирани с болестта. Степента на преживяемост обаче варира в широки граници в зависимост от стадия на рака при диагностицирането и наличието и ефективността на лечението. (източник: American Cancer Society)

Chart, pie chart

Description automatically generated

Фигура 4 - брой преобладаващи случаи (5-годишни) за 2020г. в световен мащаб, всякаква възраст

В България ракът на простатата е вторият най-често диагностициран рак при мъжете след рака на белия дроб. През 2018 г. в България има 3101 нови диагностицирани случая на рак на простатата и 915 починали от заболяването. (източник: Национален раков регистър)

Chart, pie chart

Description automatically generated

Фигура 5 - брой нови случаи за 2020г., всякаква възраст

През последните години все повече се фокусира върху подобряването на превенцията, диагностиката и лечението на рак на простатата. Напредъкът в технологиите и медицинските изследвания доведоха до разработването на нови инструменти за скрининг и диагностика, както и до по-целенасочени и ефективни лечения. Освен това нараства осъзнаването на значението на ранното откриване и необходимостта от редовен скрининг за мъже, изложени на риск от рак на простатата.

Алгоритмите за машинно обучение имат потенциала да играят важна роля в подобряването на превенцията, диагностиката и лечението на рак на простатата. Чрез анализиране на големи набори от информация за пациенти тези алгоритми могат да идентифицират модели и да предскажат резултати, което може да помогне на доставчиците на здравни услуги да вземат по-информирани решения относно грижите за пациентите. Освен това, алгоритмите за машинно обучение могат да помогнат за идентифицирането на нови рискови фактори за рак на простатата и да подобрят разбирането ни за болестта.

В заключение, подобряването на превенцията, диагностиката и лечението на рак на простатата е критична цел на общественото здраве и алгоритмите за машинно обучение имат потенциала да играят важна роля за постигането на тази цел. Ракът на простатата е сериозен здравен проблем в глобален и национален мащаб, със значителни социални и икономически последици и може да има дълбоко въздействие върху живота на пациентите и техните семейства.

* 1. **Как се тества и открива ?**

Тестовете за скрининг на простатата могат да включват: дигитален ректален преглед (**DRE**[[1]](#footnote-1)).

Ако Вашият лекар открие някакви аномалии в текстурата, формата или размера на жлезата, може да се нуждаете от допълнителни изследвания. Тест за простатен специфичен антиген (**PSA**[[2]](#footnote-2)): кръвна проба се взема от вена на ръката ви и се анализира за **PSA**, вещество, което естествено се произвежда от вашата простатна жлеза. Нормално е в кръвта ви да има малко количество **PSA**. Въпреки това, ако се открие по-високо от нормалното ниво, това може да означава инфекция на простатата, възпаление, уголемяване или рак. Ако **DRE** или **PSA** тест открие аномалия, Вашият лекар може да препоръча допълнителни тестове, за да определи дали имате рак на простатата, като например: ултразвук. Ако други тестове предизвикват притеснения, Вашият лекар може да използва трансректален ултразвук за допълнителна оценка на простатата. Малка сонда, с размерите и формата на пура, се вкарва в ректума ви. Сондата използва звукови вълни, за да създаде картина на вашата простатна жлеза. Вземане на проба от простатна тъкан: ако първоначалните резултати от теста предполагат рак на простатата, Вашият лекар може да препоръча процедура за вземане на проба от клетки от простатата (**простатна биопсия**). Биопсията на простатата често се прави с помощта на тънка игла, която се вкарва в простатата за събиране на тъкан. Тъканната проба се анализира в лаборатория, за да се определи дали има ракови клетки.

**1.2. Каква е степента на ISUP сега ?**

Съгласно настоящите насоки на **Международното дружество по урологична патология** (ISUP), оценките на **Gleason**[[3]](#footnote-3) се обобщават в степен по **ISUP** по скала от 1 до 5 съгласно следното правило:

* Gleason резултат 6 = ISUP степен 1
* Gleason резултат 7 (3 + 4) = ISUP степен 2
* Gleason резултат 7 (4 + 3) = ISUP степен 3
* Gleason резултат 8 = ISUP степен 4
* Gleason резултат 9-10 = ISUP степен 5

Ако в пробата няма рак, използваме етикета ISUP степен 0.

Diagram

Description automatically generated

Фигура 6 - Gleason score и ISUP grade

**1.3. Как са генерирани резултатите от Gleason в набора от данни ?**

Всеки **WSI** в това предизвикателство (challenge) съдържа един или в някои случаи два тънки тъканни срези, изрязани от една проба за биопсия. Преди сканирането тъканта се оцветява с хематоксилин и еозин (H&E). Това е стандартен начин за оцветяване на първоначално прозрачната тъкан, за да се получи известен контраст. Пробите са съставени от жлезиста тъкан и съединителна тъкан. Жлезите са кухи структури, които могат да се видят като бели „дупки“ или разклонени кухини в **WSI**. Външният вид на жлезите формира основата на системата за класификация на **Gleason**. Жлезистата структура, характерна за здравата простатна тъкан, прогресивно се губи с увеличаване на степента. Системата за оценяване разпознава три категории: 3, 4 и 5.

* **[A]** Доброкачествени простатни жлези с нагънат епител: цитоплазмата е бледа, а ядрата малки и правилни. Жлезите са групирани заедно.
* **[B]** Аденокарцином на простатата: модел 3 на Gleason няма загуба на жлезиста диференциация. Между доброкачествените жлези се инфилтрират малки жлези. Цитоплазмата често е тъмна, а ядрата са разширени с тъмен хроматин и някои изпъкнали нуклеоли. Всяка епителна единица е отделна и има лумен.
* **[C]** Аденокарцином на простатата: модел 4 на Gleason има частична загуба на жлезиста диференциация. Има опит за образуване на лумина, но туморът не успява да образува пълни, добре развити жлези. Тази микроснимка показва неправилен крибриформен рак, т.е. епителни листове с множество лумини. Има и някои зле оформени малки жлези и някои слети жлези. Всички те са включени в Gleason Pattern 4.
* Map

  Description automatically generated**[D]** Аденокарцином на простатата: модел 5 на Gleason има почти пълна загуба на жлезиста диференциация. В стромата се виждат диспергирани единични ракови клетки. Gleason Pattern 5 може също да съдържа твърди листове или нишки от ракови клетки. Всички микроснимки показват петна от хематоксилин и еозин при 20x увеличение на лещата.

Фигура 7 - изображения на простатни жлези

**2. Описание на набора данни за диагностичния анализ**

В този курсов проект са разгледани два различни дейтасета. В първия дейтасет се прави анализ на предоставените данни, като цялото му съдържание е под формата на снимки във формат .tiff (411.9 GB) и се намира на посочения [competition](https://www.kaggle.com/competitions/prostate-cancer-grade-assessment/overview) в Kaggle[[4]](#footnote-4). Снимките съдържат микроскопични сканирания на проби от биопсия на рак на простатата. В проекта за този дейтасет е направен единствено преглед и анализ на данните чрез възможностите на **Python** библиотеките **numpy** и **pandas**. За визуализация на резултатите се използват **matplotlib.pyplot**, **seaborn**, **plotly**, **PIL** и **IPython.display**. Във втория дейтасет се прави преглед, анализ на данните и трениране на модел посредством класификационни алгоритми и невронна мрежа. Също така се разглежда точността на всеки от алгоритмите и се сравняват метрики за производителност/качество на бинарната класификация. Класификационните алгоритми, които са представени са: **Логистична регресия**, **KNN**, **Naive Bayes**, **Decision Tree** и **Random Forest**, а невронните мрежи са **ANN, RNN и MLP**. Съдържанието на дейтасета е текстово и таблично под формата на .csv файл. Този файл съдържа 100 записа на пациенти с направена проба за рак на простатата. В набора от данни се съдържат 10 променливи (9 числови и 1 категорийна – id). Отново може да бъде достъп на посочения [линк](https://www.kaggle.com/datasets/sajidsaifi/prostate-cancer?datasetId=66762&sortBy=voteCount) в Kaggle[[5]](#footnote-5).

**2.1.А. Описание на характеристиките за дейтасет 1**

### **Дефиниции на основните колони и етикетите от всеки институт**

**Dataset**: [train/test].csv

**image\_id**: Идентификационен номер на снимката.

**data\_provider**: Името на институцията, предоставила данните. Както институтът Каролинска, така и **Медицинският център на университета Radboud** допринасят с данни. Те са използвали различни скенери с малко по-различни максимални микроскопски разделителни способности и са работили с различни патолози за етикетиране на изображенията.

**isup\_grade**: Само за трениране, целева променлива. Тежест на рака по скала от 0-5.

**gleason\_score**: Само за трениране. Алтернативна система за оценка на тежестта на рака с повече нива от скалата на ISUP.

**[train/test]\_images**: Изображения. Всяка снимка представлява огромен **tiff** файл. Може да се очакват грубо 1000 снимки в скрития тестови сет. Обърнете внимание, че са въведени малко по-различни процедури за изображенията, използвани в тестовия набор от набора за обучение. Някои от изображенията от набора за обучение имат случайни следи от химикал, докато тези в тестовия сет са без такива следи.

**train\_label\_masks**: Маски за сегментиране, показващи кои части от изображението са довели до оценката на ISUP. Не всички тренировъчни изображения имат такива маски и може да има фалшиви положителни (false positives) или фалшиви негативи (false negatives) по различни причини. Тези маски са предоставени, за да помогнат при разработването на стратегии за избор на най-полезните подизвадки от изображенията. Стойностите на маската зависят от набора на данни:

**Radboud институт: Простатните жлези са индивидуално обозначени. Валидните стойности са**:

0: background (non tissue) or unknown

1: stroma (connective tissue, non-epithelium tissue)

2: healthy (benign) epithelium

3: cancerous epithelium (Gleason 3)

4: cancerous epithelium (Gleason 4)

5: cancerous epithelium (Gleason 5)

**Karolinska институт: Регионите са обозначени. Валидните стойности са**:

0: background (non tissue) or unknown

1: benign tissue (stroma and epithelium combined)

2: cancerous tissue (stroma and epithelium combined)

A screenshot of a computer

Description automatically generated with low confidence

Фигура 8 - извадка на данните в дейтасета

**2.1.Б. Описание на характеристиките за дейтасет 2**

**Дефиниции на основните колони**

**Dataset:** [Prostate\_Cancer.csv]

**id**: Уникален идентификационен номер за всяка проба в дейтасета.

**radius**: Средното разстояние между центъра и периметъра на тумора.

**texture**: Стандартното отклонение на стойностите на сивата скала в изображението на тумора.

**perimeter**: Периметърът или дължината на тумора.

**area**: Площта или общия брой пиксели на тумора.

**smoothness**: Вариация в дължината на радиуса.

**compactness**: Мярка за компактност на тумора, изчислена като (периметър^2) / (4 \* pi \* площта).

**diagnosis\_result**: Резултатът от диагнозата за всяка проба. Индикира дали пробата има етикет злокачествен **(M)** или доброкачествен **(B)**. Злокачествените се отнасят за ракови клетки, а доброкачествените за неракови.

**symmetry**: Мярка за симетрия представляваща приликата между двете половини на тумора.

**fractal dimension:** Мярка за сложността на тумора.

**2.2.A. Статистически анализ на характеристиките – визуализация, корелационна матрица (heat map), и др. (дейтасет 1)**

**Преглед на данните**

**A picture containing text, font, screenshot

Description automatically generated**

Фигура 9 - прочитане на данните

**A picture containing text, screenshot, font, document

Description automatically generated**

Фигура 10 - резултат от прочитането на данните

**A screenshot of a computer code

Description automatically generated with low confidence**

Фигура 11 - изследване на данните

**Визуализация**

**A screenshot of a computer

Description automatically generated with medium confidence**

Фигура 12 - статистически функции

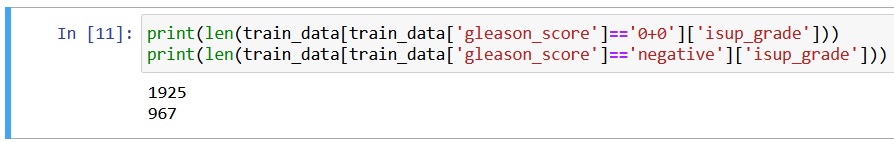
Функцията **unique** в **pandas** се използва за намиране на уникалните стойности от дадена серия. Поредицата е една колона от рамка от данни. Тази част от кода преброява броя на извадките в рамката с данни train\_data, които имат "gleason\_score" от "0+0" или "отрицателен", и показва броя на извадките, които имат съответното "isup\_grade". Това може да бъде полезно за разбиране на разпределението на стойностите на "gleason\_score" и "isup\_grade" в набора от данни за обучение.

A picture containing text, font, screenshot

Description automatically generated

Фигура 13 - резултат от изпълнението на функцията unique()

Ако оценката по Глийсън е изписана като 3+4=7, това означава, че по-голямата част от тумора е степен 3, а по-малката - степен 4, и те се сумират, за да се получи оценка по Глийсън 7. Други начини, по които тази оценка по Глийсън може да бъде изписана са: Глийсън 7/10, Глийсън 7 (3+4) или комбинирана оценка по Глийсън 7. Ако всички тумори са от една и съща степен (например степен 3), тогава резултатът по Глийсън се отчита като 3+3=6.



Фигура 14 - isup\_grade при gleason\_score = '0+0' и 'negative'

**A picture containing text, screenshot, software, operating system

Description automatically generated**

Фигура 15 - резултат от изпълнението на функцията groupby() за isup\_grade

Колкото по-висок е броят на "image\_id", толкова повече са жълтият и зеленият цвят, които помагат за визуалното идентифициране на групите с по-висок брой на "image\_id".

A picture containing text, screenshot, number

Description automatically generated

Фигура 16 - резултат от изпълнението на функцията groupby() за gleason\_score

Оценки по Глийсън от 5 или по-малко обикновено не се използват. Диагноза с оценка на Глийсън 6 се счита за диагноза от ниска степен. Ракът от средна степен е с оценка на Глийсън 7, а ракът от висока степен - с оценка 8, 9 или 10. Ракът от нисък клас се развива по-постепенно и има по-малък шанс за раково развитие, отколкото ракът от висок клас.

Създаване на фуниева диаграма с помощта на библиотеката **plotly** за визуализиране на разпределението на колоната "isup\_grade" в рамката с данни "train\_data". Диаграмата показва колко изображения принадлежат към всяка категория 'isup\_grade'. Цветовете на сегментите на диаграмата се задават въз основа на броя на идентификаторите на изображенията за всяка категория. Аргументите y и x се задават съответно на колоните "isup\_grade" и "image\_id" на данните. Аргументът за маркера се задава към колоната "image\_id" на данните, която задава цвета на всеки сегмент на фуниевидната диаграма въз основа на броя на изображенията за всеки "isup\_grade". Аргументите textposition и textfont се използват за показване на етикетите в диаграмата на фунията. Получената диаграма е озаглавена "**ISUP\_grade Distribution Funnel Chart**" (Диаграма на фунията на разпределението на ISUP\_grade), а осите x и y са съответно обозначени. Диаграмата се показва с помощта на метода show().

A screenshot of a computer code

Description automatically generated with low confidence

Фигура 17 - създаване на фуниева диаграма

A picture containing text, screenshot, line, design

Description automatically generated

Фигура 18 - ISUP\_grade фуниева диаграма

Системата за класификация **ISUP** е по-нов и по-прост метод за класификация на рака на простатата в сравнение с използваната преди това система, базирана на скалата на **Глийсън**. Тя използва петстепенна скала за класифициране на рака на простатата въз основа на моделите на развитие на клетките, видими под микроскоп, като варира от степен 1 (най-малко агресивна) до степен 5 (най-агресивна). Извадката от код създава стълбовидна диаграма, използвайки библиотеката **plotly**, за да визуализира разпределението на колоните "isup\_grade" и "image\_id" в dataFrame train\_data. По оста x са представени категориите ISUP\_grade, а по оста y - броят на изображенията. Всеки стълб в графиката е оцветен по различен начин в зависимост от категорията, която представлява. Диаграмата е озаглавена "**ISUP\_grade and image\_id Bar Chart**", а осите x и y са обозначени по съответния начин. Получената диаграма се показва с помощта на метода show().

A picture containing text, screenshot, rectangle, design

Description automatically generated

Фигура 19 - ISUP\_grade и image\_id стълбовидна диаграма

**A screenshot of a computer

Description automatically generated with medium confidence**

Фигура 20 - резултат от изпълнението на функцията head() за train\_data

**A screenshot of a computer

Description automatically generated with medium confidence**

Фигура 21 - създаванена хистограма за Gleason score

Групиране на данните по isup\_grade и data\_provider и преброяване на броя на появяванията с помощта на метода size(), след което завъртане на данните, за да се създаде матрица, подходяща за изчертаване на графики, с isup\_grade като индекс и data\_provider като колона. Накрая, данните се визуализират като стълбовидна диаграма, чрез използване на **plot(kind='bar', stacked=False)**. Добавя се заглавие и етикети на осите с помощта на title(), xlabel() и ylabel().

**A screenshot of a computer

Description automatically generated with low confidence**

Фигура 22 - създаване на стълбовидна диаграма за разпределението на ISUP Grade по източник на данни

**A picture containing text, screenshot, diagram, plot

Description automatically generated**

Фигура 23 - резултатна стълбовидна диаграма

По-голямата част от данните за категории 0 и 1 на класа isup са предоставени от **karolinska**. По-голямата част от данните за категориите 3, 4 и 5 на ISUP се предоставят от **radboud**.

A screenshot of a computer

Description automatically generated with low confidence

Фигура 24 - създаване на стълбовидна диаграма за разпределението на Gleason Score по източник на данни

**A picture containing text, screenshot, plot, diagram

Description automatically generated**

Фигура 25 - резултатна стълбовидна диаграма

Всяка информация за категорията на резултата на Глейсън (0+0) идва от **karolinska**. Цялата информация за категорията "отрицателен резултат" е предоставена от **radboud**. Karolinska е значителен източник на данни за категорията "оценка по Gleason" (3+3). Radbound обаче е значителен източник на данни за категориите (4+4), (4+3), (4+5), (5+4), (5+5), (5+3) и (3+5).

A close-up of a computer screen

Description automatically generated with low confidence

Фигура 26 - инициализиране на функция crosstab()

**A picture containing text, screenshot, diagram, rectangle

Description automatically generated**

Фигура 27 – heatmap за разпределението на ISUP Grade и Gleason Score

**2.2.Б. Статистически анализ на характеристиките – визуализация, корелационна матрица (heat map), и др. (дейтасет 2)**

**Преглед на данните**

**A screenshot of a computer

Description automatically generated with low confidence**

Фигура 28 - преглед на данните

A screenshot of a computer

Description automatically generated with medium confidence

Фигура 29 - резултат от изпълнението на кода на Фигура 28

A picture containing text, screenshot, number, font

Description automatically generated

Фигура 30 - резултат от изпълнението на кода на Фигура 28

A screenshot of a computer

Description automatically generated with low confidence

Фигура 31 - резултат от изпълнението на кода на Фигура 28

**Визуализация**

A picture containing text, screenshot, font, line

Description automatically generated

Фигура 32 - създаване на корелационната матрица

A picture containing text, font, number, screenshot

Description automatically generated

Фигура 33 - резултат от изпълнението на Фигура 32

A picture containing text, font, line, screenshot

Description automatically generated

Фигура 34 - създаване на кръгова диаграма за разпределението на злокачествени и доброкачествени случаи

A picture containing circle, diagram, graphics, logo

Description automatically generated

Фигура 35 - резултат от изпълнението на кода на Фигура 34

**2.2.В. Предварителна обработка на набора данни (preprocessing) – зашуменост, липсващи данни, редукция на размерност, и др. (дейтасет 2)**

**Обработка на данните (премахване на излишни колони)**

A picture containing text, screenshot, font

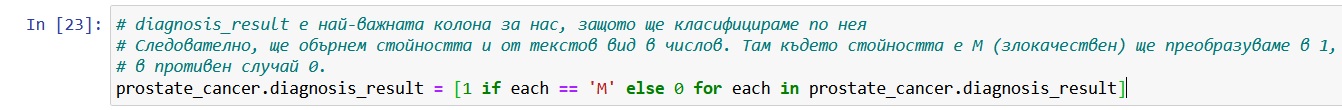
Description automatically generated

Фигура 36 – Принтиране на съществуващите колони в набора от данни

A screenshot of a computer

Description automatically generated with medium confidence

Фигура 37 - Премахване на излишни колони от данните



Фигура 38 - обработка на данните

**Създаване на x и y променливите**

A screenshot of a computer

Description automatically generated with medium confidence

Фигура - създаване на x и y променливите

**Нормализация**

A screenshot of a computer

Description automatically generated with medium confidence

Фигура 40 - Нормализация на данните

**Разделяне набора от данни на test и train (подготовка на данните за прилагане на ML класификационните алгоритми)**

**A screenshot of a computer

Description automatically generated with medium confidence**

Фигура 41 - разделяне на набора от данни на test и train

**3. Технологична работна рамка – програмна среда, език за програмиране, библиотеки, и др.**

**Програмна среда**

A picture containing graphics, font, circle, graphic design

Description automatically generated

[**Jupyter**](https://jupyter.org/) е проект за разработване на софтуер с отворен код, отворени стандарти и услуги за интерактивни изчисления в множество езици за програмиране. Той беше отделен от IPython през 2014 г. от Фернандо Перес и Браян Грейнджър. Името на Project Jupyter е препратка към трите основни езика за програмиране, поддържани от Jupyter, които са **Julia**, **Python** и **R**. Неговото име и лого са почит към откритието на Галилео за луните на Юпитер, както е документирано в бележници, приписвани на **Галилей**. Проектът Jupyter разработи и поддържа интерактивните изчислителни продукти **Jupyter Notebook**, [**JupyterHub**](https://jupyter.org/hub) и [**JupyterLab**](https://jupyterlab.readthedocs.io/en/latest/). **Jupyter** е финансово спонсориран от **NumFOCUS**.

**Jupyter Notebook**

**Jupyter Notebook** може да се отнася до две различни концепции, или към потребителско приложение за редактиране на код и текст, или към основния файлов формат, който е оперативно съвместим в много имплементации.

* Приложение за редактиране на код и текст

**Jupyter Notebook** (по-рано IPython Notebook) е уеб-базирана интерактивна изчислителна среда за създаване на бележни документи. **Jupyter Notebook** е създаден с помощта на няколко библиотеки с отворен код, включително IPython, ZeroMQ, Tornado, jQuery, Bootstrap и MathJax. Приложението **Jupyter Notebook** е базирано на браузър REPL, съдържащ подреден списък от клетки за вход/изход, които могат да съдържат код, текст (използвайки **Github Flavored Markdown**[[6]](#footnote-6)), математика, графики и медия.

* Файлов формат

Файловият формат на **Jupyter Notebook** е **JSON** файл, съдържащ схема с версии, обикновено завършващ с разширение ".ipynb". Основните части на **Jupyter Notebook** са: **метаданни**, **формат на бележника** и **списък с клетки**. Метаданните са речник на данни от дефиниции за настройка и показване на бележника. **Notebook Format** е номер на версия на софтуера. Списъкът с клетки са различни типове клетки за Markdown (показване), код (за изпълнение) и изход на клетките от типа код.

Въпреки че „.ipynb“ и JSON са най-често срещаният формат по подразбиране, възможно е да се откажете от някои функции (като съхраняване на изображения и метаданни) и да запазите бележника като документ за маркиране с разширение **JupyText**. **Jupytext** често е свързан с контрола на версиите (version control systems), за да направи разликата (diffing) и сливането (merging) на бележника по-просто.

**Език за програмиране**

**Python**

A picture containing logo, graphics, font, clipart

Description automatically generated

**Python** е език за програмиране от високо ниво, интерпретативен, интерактивен, обектно-ориентиран, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC **Monty Python’s Flying Circus**. Често бива сравняван с Tcl, Perl, Scheme, Java и Ruby.

Идеята за **Python** се заражда в края на 1980-те години, като реалното осъществяване започва през декември 1989 г. от Гуидо ван Росум в CWI (Centrum Wiskunde & Informatica – международно признат изследователски институт по математика и компютърни науки, локализиран в Амстердам, Холандия). **Python** имал за цел да се превърне в наследник на **ABC** (език за програмиране, от своя страна вдъхновен от SETL), който да бъде способен да обработва изключения и да е съвместим с операционната система **Amoeba**.

**Библиотеки**

* [Numpy](https://numpy.org/)
* [Pandas](https://pandas.pydata.org/)
* [Os](https://docs.python.org/3/library/os.html)
* [Math](https://docs.python.org/3/library/math.html)
* [matplotlib.pyplot](https://matplotlib.org/stable/tutorials/introductory/pyplot.html)
* [seaborn](https://seaborn.pydata.org/)
* [PIL](https://pillow.readthedocs.io/en/stable/)
* [IPython.display](https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html)
* [plotly](https://plotly.com/)
* [scikit-learn](https://scikit-learn.org/stable/)

**4. Описание на алгоритмите за класификация**

**Logistic Regression**

Логистичната регресия е статистически метод за прогнозиране на вероятността за настъпване на определен резултат въз основа на входни данни. Този метод се използва предимно за бинарна класификация, където целта е да се определи към кой от двата възможни класа принадлежи даден обект. Например, може да се използва логистична регресия за предсказване дали даден пациент има дадено заболяване или не, базирано на различни медицински характеристики.

Логистичната регресия използва логистична функция, наречена сигмоидна функция, за да моделира вероятността за настъпване на определен резултат. Тази функция има S-образен график и приема стойности между 0 и 1. Входните данни, които могат да бъдат числови или категорийни, се умножават със съответни параметри или коефициенти, и се сумират. След това се прилага логистичната функция върху резултата от тази сума, за да се получи вероятността за класификация.

Процесът на обучение на модела на логистичната регресия включва намирането на оптимални стойности на параметрите, които максимизират вероятността да се предсказва правилният класификационен резултат. Това се осъществява с помощта на метод на максимума на правдоподобието или други методи за оптимизация.

Логистичната регресия е широко използвана в машинното обучение и статистиката, особено при задачи на класификация, където целта е да се предскаже принадлежността на обектите към определени класове. Тя е една от основните техники за класификация и предоставя релативно прост начин за моделиране на вероятностите за класификация.

**KNN**

Алгоритъмът за k-най-близки съседи, известен също като KNN или k-NN, е непараметричен класификатор с контролирано обучение, който използва близостта, за да прави класификации или прогнози относно групирането на отделна точка от данни. Въпреки че може да се използва както за регресия, така и за проблеми с класификацията, обикновено се използва като алгоритъм за класификация, работещ на предположението, че подобни точки могат да бъдат намерени близо една до друга.

**Naive Bayes**

Наивният Бейсов класификатор (Naive Bayes classifier) е статистически алгоритъм за класификация, основан на приложението на теоремата на Бейс за вероятностите. Той е "наивен" в смисъл, че прави предположението, че характеристиките (признаците) на обектите са статистически независими една от друга.

Наивният Бейсов класификатор използва вероятности и байесов подход за класификация на нови обекти в определени класове. Той се базира на предположението, че вероятностите за принадлежност на обектите към определени класове са условни вероятности на срещането на характеристики, като се предполага независимостта им.

Стъпките за използване на наивния Бейсов класификатор са следните:

1. Събиране на обучаващия набор от данни, който съдържа обекти със свои характеристики и съответните им класове.
2. Изчисляване на вероятностите за принадлежност на обектите към всяка класа. Това се прави чрез изчисляване на априорните вероятности на класовете (вероятностите за срещане на всеки клас в обучаващия набор).
3. Изчисляване на условните вероятности за срещането на характеристиките във всяка класа. Това се прави чрез предположението за независимостта на характеристиките, което позволява разделянето на условните вероятности на срещане на отделните характеристики.
4. При получаване на нов обект, изчисляване на вероятностите за принадлежност към всяка класа, използвайки априорните вероятности и условните вероятности на срещането на характеристиките във всяка класа.
5. Класификация на новия обект, като се избира класът с най-висока вероятност.

Наивният Бейсов класификатор е лесен за разбиране и бърз за обучение и класификация. Въпреки че прави наивно предположение за независимостта на характеристиките, той често дава добри резултати, особено когато данните са достатъчно големи и независимостта не е силно нарушена. Той се използва в различни области, като текстова класификация, филтриране на спам и други.

**Decision Tree**

Дървото на решенията (decision tree) е графичен модел, който представлява последователност от въпроси или условия, базирани на характеристики на данните, за да се вземе решение или да се направи прогноза. Той се използва в задачи на класификация и регресия в машинното обучение.

Структурата на дървото на решенията прилича на обратен дървовиден граф, където всеки възел представлява въпрос или условие, а всяка връзка (ребро) от възела към други възли представлява отговор или резултат от условието. Всяка листова връзка в дървото отговаря на класификационен резултат или стойност.

Стъпките за построяване на дърво на решенията са следните:

1. Избор на коренов възел - това е въпросът или характеристиката, която най-добре разделя данните на различни класове или стойности.
2. Разделяне на данните според избраната характеристика и създаване на децата на кореновия възел - това са подмножества от данните, които отговарят на различните отговори или стойности на въпроса.
3. Рекурсивно повтаряне на стъпки 1 и 2 за всеки възел-дете, докато достигне условието за спиране, например когато достигне предварително зададена максимална дълбочина или когато вече не могат да бъдат разделени данните.
4. При прогнозиране на нови обекти, те се подават през дървото, като се следват последователността на въпросите, докато се достигне листовия възел, който предсказва класа или стойността на обекта.

Предимствата на дървото на решенията включват лесна интерпретируемост, способността да работи с различни типове данни и автоматично откриване на важността на характеристики. Въпреки това, дървото на решенията може да бъде склонно към прекомерно подгонване на данните и не винаги постига най-добри резултати в сравнение с други по-сложни модели. Затова са разработени различни техники за оптимизация и обобщаване на дърветата на решенията, като случайните гори и градиентното дърво на решенията.

**Random Forest**

Random Forest е алгоритъм за машинно обучение, който принадлежи към семейството на ансамбловото обучение. Основава се на концепцията за дърветата на решенията и комбинира множество дървета на решенията, за да създаде по-стабилен и точен модел. Използва техника, наречена "bagging", при която всяко дърво на решенията в гората се обучава върху произволно подмножество от данните за обучение и характеристиките. Окончателната прогноза се прави чрез обобщаване на прогнозите на всички отделни дървета чрез гласуване (за класификация) или осредняване (за регресия).

Random Forest е известна със способността си да се справя с данни с голяма размерност, оценка на важността на признаците и обработка на липсващи стойности. Той може да се използва както за задачи за класификация, така и за регресия и като цяло е по-малко склонен към свръхподготовка в сравнение с отделните дървета за вземане на решения.

* **Описание на алгоритъма използван в тази курсова работа**

1. Алгоритъмът започва с импортиране на необходимите библиотеки, включително `RandomForestClassifier` от `sklearn.ensemble`, както и други необходими библиотеки като `classification\_report`, `confusion\_matrix`, `plt` и `sns` за целите на визуализацията.

2. Създава се екземпляр на `RandomForestClassifier` с `n\_estimators=100`, който задава броя на дърветата на решенията в гората на 100. Параметърът `random\_state=42` осигурява възпроизводимост на резултатите.

3. Методът `fit` се извиква върху обекта `rand\_forest`, за да се обучи моделът на случайната гора, като се използват данните `x\_train` и `y\_train`.

4. Точността на теста на модела на случайната гора се извежда с помощта на метода `score`, който изчислява точността чрез сравняване на предсказанията върху данните `x\_test` със съответните етикети `y\_test`.

5. Името и резултатът на модела на случайната гора се добавят съответно към списъците `method\_names` и `method\_scores`.

6. Методът `predict` се използва, за да се получат предсказаните етикети (`pred\_rand\_forest`) за данните `x\_test`.

7. Извиква се функцията `classification\_report`, за да се генерира отчет (`class\_rep\_rand\_forest`), който включва precision (точност), recall (отзоваване), F1-скор и support (подкрепа) за всеки клас въз основа на предсказаните етикети (`pred\_rand\_forest`) и истинските етикети (`y\_test`).

8. Функцията `confusion\_matrix` се използва за изчисляване на матрицата на объркване (`conf\_mat`) чрез сравняване на предсказаните етикети (`y\_pred`) с истинските етикети (`y\_test`).

9. След това confusion matrix се визуализира с помощта на библиотеките `matplotlib` и `seaborn`, за да се покаже разпределението на предсказаните и истинските лейбъли.

Изборът на `n\_estimators=100` означава, че случайната гора се състои от 100 дървета на решенията. Тази стойност е хиперпараметър, който може да се регулира в зависимост от сложността на проблема и наличните изчислителни ресурси. Задаването на `random\_state=42` гарантира, че случайната гора дава едни и същи резултати всеки път, когато е обучавана, което е полезно за възпроизводимостта.

**ANN**

* **Изкуствени невронни мрежи (ANN):**

Изкуствените невронни мрежи са изчислителни модели, вдъхновени от структурата и функционирането на човешкия мозък. ANN се състоят от взаимосвързани възли, наречени неврони, които обработват и предават информация. Те се състоят от входен, скрит и изходен слой, където невроните във всеки слой са свързани с невроните в следващия слой. Връзките между невроните имат съответни тегла, които определят силата на връзката. ANN се учат, като коригират тези тегла въз основа на данни от обучението, за да правят точни прогнози или решения.

ANN се използват широко в машинното обучение за задачи като класификация, регресия и разпознаване на образи. Те могат да обработват сложни, нелинейни връзки и са способни да се обучават от големи количества данни.

* **Описание на алгоритъма използван в тази курсова работа**

1. Алгоритъмът започва с импортиране на необходимите библиотеки, включително KerasClassifier от keras.wrappers.scikit\_learn, cross\_val\_score от sklearn.model\_selection, Sequential от keras.models и Dense от keras.layers.

2. Дефинирана е функцията `build\_classifier`, която конструира архитектурата на модела на невронната мрежа. Тя инициализира последователен модел (`класификатор`) и добавя слоеве към него. Входният слой има 50 единици/неврони с функцията за активиране 'relu'. Този слой очаква входни данни с форма, определена от `x\_train.shape[1]`, където `x\_train` са данните за обучение. Скритият слой има 10 единици/неврони с активационна функция "relu", а изходният слой има 1 единица/неврон с активационна функция "sigmoid".

3. Моделът е съставен с помощта на оптимизатора 'adam', който е ефективен и широко използван оптимизационен алгоритъм. Използвана е функцията за загуби "binary\_crossentropy", тъй като разглежданият проблем е задача за бинарна класификация. Освен това за оценка на ефективността на модела е посочена метриката "accuracy".

4. Създава се екземпляр на `KerasClassifier` с помощта на функцията `build\_classifier`, като за обучението са зададени 200 епохи. Под `epochs` се разбира броят пъти, през които наборът от данни за обучение преминава през невронната мрежа по време на процеса на обучение.

5. Функцията `cross\_val\_score` се използва за извършване на кръстосано валидиране на класификатора. Тя приема обекта `classifier`, `x\_train` като входните характеристики, `y\_train` като съответните етикети и `cv = 3`, за да укаже, че трябва да се извърши 3-кратна кръстосана валидация.

6. Точностите, получени от кръстосаното валидиране, се съхраняват в масива `accuracies`.

7. Средната стойност и дисперсията на точността се изчисляват, като се използват съответно функциите `mean()` и `std()`.

8. Средната стойност и дисперсията на точността се изписват на конзолата.

Изборът на параметри в кода е както следва:

- `units`: Броят на единиците/невроните във всеки слой определя капацитета и сложността на модела на невронната мрежа. В този код се използват 50 единици във входния слой, 10 единици в скрития слой и 1 единица в изходния слой.

- `activation`: Функциите за активиране определят изхода на неврона. Във входния и скрития слой се използва 'relu' (Rectified Linear Unit), която е често използвана функция за активиране за улавяне на нелинейни връзки. 'Сигмоид' се използва в изходния слой, за да се получи изход, подобен на вероятност, за двоична класификация.

- `kernel\_initializer`: Схемата за инициализиране на теглото на невроните в слоевете е зададена на 'uniform', което инициализира теглото равномерно в определен диапазон. Този избор помага да се предотврати възможността някой конкретен неврон да доминира в процеса на обучение в началото.

- `optimizer`: Оптимизаторът "adam" е избран заради неговата ефективност при актуализиране на теглата по време на процеса на обучение. Той адаптира динамично скоростта на обучение и се справя добре с разредените градиенти.

- `loss`: Функцията за загуби "binary\_crossentropy" се използва за задачи за двоична класификация. Тя измерва различието между истинските етикети и предсказаните вероятности.

Като цяло кодът обучава ANN модел с помощта на Keras, извършва кръстосано потвърждение и оценява точността на модела.

**RNN**

* **Рекурентни невронни мрежи (RNN):**

Рекурентните невронни мрежи (RNN) са вид изкуствена невронна мрежа, специално проектирана за работа с последователни данни. RNN са подходящи за задачи, които включват обработка и изготвяне на прогнози въз основа на последователности, като например анализ на времеви редове, обработка на естествен език и разпознаване на реч. За разлика от захранващите невронни мрежи, RNN имат цикли в мрежовата си структура, които им позволяват да поддържат информация за минали входове.

RNN обработват последователни данни, като предават текущия вход заедно с информацията от предишните входове чрез рекурентни връзки. Този механизъм позволява на RNN да улавят времевите зависимости в данните, което ги прави ефективни при моделирането на последователни модели.

* **Описание на алгоритъма използван в тази курсова работа**

1. Алгоритъмът започва с импортиране на необходимите библиотеки, включително `Dense`, `SimpleRNN`, `Dropout` от `keras.layers`, `mean\_squared\_error` от `keras.metrics` и `Sequential` от `keras.models`.

2. Създава се последователен модел (`model`), като се използва `Sequential()`.

3. Архитектурата на RNN модела се определя чрез добавяне на слоеве към модела. Първият слой е слой `SimpleRNN` със 100 единици/неврони, активираща функция ` tanh` и `return\_sequences=True`. Параметърът `input\_shape` е зададен на `(trainX.shape[1],1)`, където `trainX` представлява данните за обучение. Конфигурацията `return\_sequences=True` гарантира, че изходът на този слой се подава като вход към следващия слой.

4. След първия слой на `SimpleRNN` се добавя слой `Dropout` със степен на отпадане 0,20. Отпадането е техника за регуларизация, която по време на обучението произволно задава на част от входните единици стойност 0, което помага да се предотврати прекомерното приспособяване.

5. Добавят се още два слоя с `SimpleRNN` и `Dropout`, като се следва подобен модел. Тези слоеве допринасят за дълбочината и капацитета на RNN модела.

6. Добавя се последният слой `Dense` с 1 единица, който служи като изходен слой.

7. Моделът се компилира, като се използва оптимизаторът `adam`, който е популярен избор за оптимизационни алгоритми, базирани на градиента. Използва се функцията за загуба "средна квадратна грешка" (`mean\_squared\_error`), тъй като решаваният проблем включва регресия.

8. Моделът се обучава чрез метода `fit`, където `trainX` представлява входните данни, а `y\_train` представлява съответните целеви стойности. Моделът се обучава за 200 епохи с размер на партидата от 32. По време на обучението моделът коригира теглата си, за да минимизира загубата на средната квадратна грешка.

9. Изчисляват се средната стойност и дисперсията на точността и се извеждат на конзолата. Важно е обаче да се отбележи, че масивът `accuracies` не е дефиниран в предоставения код, така че тези редове може да не са необходими или да се нуждаят от модификация.

Изборът на параметри в кода е както следва:

- `units`: Броят на единиците/невроните във всеки слой на `SimpleRNN` определя капацитета и сложността на модела RNN. По-голям брой единици може да позволи на модела да улавя по-сложни модели, но също така може да увеличи изчислителните разходи.

- `activation`: Функцията за активиране, използвана в слоевете `SimpleRNN`, е "tanh", която обикновено се използва в RNN, тъй като позволява да се уловят нелинейни връзки.

- `return\_sequences`: Задаването на `return\_sequences=True` в първия слой на `SimpleRNN` и следващите слоеве гарантира, че изходът на всеки слой се предава като вход на следващия слой, което позволява на модела да улавя дългосрочни зависимости в последователните данни.

- `Dropout`: Отпадането е техника за регуларизация, използвана за предотвратяване на прекомерното приспособяване в невронните мрежи. Слоевете `Dropout` в кода прилагат регуларизация с отпадане със степен на отпадане 0,20. Това означава, че по време на обучението 20% от единиците в съответните слоеве ще бъдат произволно задавани на 0 при всяко обновяване, което помага за намаляване на прекомерната зависимост от конкретни единици и насърчава мрежата да научи по-устойчиви и обобщени представяния.

- `optimizer`: Оптимизаторът "adam" се използва за актуализиране на теглата на модела RNN по време на процеса на обучение. Adam е оптимизационен алгоритъм с адаптивна скорост на обучение, който съчетава предимствата на два други популярни оптимизационни алгоритъма - AdaGrad и RMSProp. Той адаптира динамично скоростта на обучение за всяко тегло въз основа на минали градиенти, което го прави ефективен за обучение на дълбоки невронни мрежи.

- `loss`: Избраната функция на загубите е "mean\_squared\_error", която изчислява средната квадратична разлика между предсказаните стойности и истинските стойности. Тази функция на загубите обикновено се използва за регресионни задачи, при които целта е да се минимизира разликата между предсказаните и действителните стойности.

Като цяло кодът конструира RNN модел с помощта на Keras, състоящ се от множество слоеве `SimpleRNN` с регуларизация на отпадането. Моделът се обучава с помощта на оптимизатора Adam и функцията за загуба на средна квадратна грешка. Архитектурата на модела, с неговите рекурентни връзки и dropout регуларизация, има за цел да улови времевите зависимости в последователните данни и да предотврати прекомерното приспособяване.

**MLP**

* **Многослоен перцептрон (MLP):**

Многослоен перцептрон (MLP) е вид невронна мрежа, състояща се от множество слоеве от възли (перцептрони). Това е широко използвана архитектура за различни задачи за машинно обучение, включително класификация и регресия. MLP са известни със способността си да научават сложни нелинейни връзки в данните.

В MLP информацията тече в една посока, от входния слой към изходния слой, през поредица от скрити слоеве. Всеки възел в даден слой е свързан с възли в съседните слоеве и всяка връзка е свързана с тегло. Изходът на всеки възел се определя чрез прилагане на функция за активиране към претеглената сума на входовете.

* **Описание на алгоритъма използван в тази курсова работа**

1. Кодът започва с импортиране на необходимите библиотеки, включително `Sequential` и `Dense` от `keras.models`, `KerasClassifier` от `keras.wrappers.scikit\_learn` и `cross\_val\_score` от `sklearn.model\_selection`.

2. Дефинирана е функцията `build\_classifier`, която конструира архитектурата на MLP модела. Моделът се създава с помощта на `Sequential()`, което показва линейна подредба на слоевете.

3. Моделът MLP се дефинира чрез добавяне на слоеве `Dense`. Първият слой (`model.add`) има 50 единици/неврони, използва функцията за активиране `relu` и очаква входна форма `x\_train.shape[1]`, където `x\_train` представлява данните за обучение. Функцията за активиране "relu" обикновено се използва в скритите слоеве на MLP за въвеждане на нелинейност.

4. Добавят се два допълнителни слоя `Dense`, с 10 единици/неврони във втория слой и 1 единица/неврон в последния изходен слой. В изходния слой се използва функцията за активиране "сигмоид", за да се получи изход, подобен на вероятност.

5. Моделът е съставен с помощта на оптимизатора "adam", който е ефективен оптимизационен алгоритъм за обучение на невронни мрежи. Избраната функция на загубите е "binary\_crossentropy", която е подходяща за проблеми на двоична класификация.

6. Функцията `build\_classifier` връща конструирания модел.

7. Създава се инстанция на `KerasClassifier`, която обвива функцията `build\_classifier`. Тя приема `build\_fn`, броя на `epochs` и `batch\_size` като параметри. Функцията `KerasClassifier` е удобен начин за използване на модела Keras в рамката за кръстосано валидиране на scikit-learn.

8. Моделът MLP се обучава по метода `fit` с данните за обучение `x\_train` и `y\_train`.

9. Кръстосаното валидиране се извършва с помощта на `cross\_val\_score`, който оценява представянето на MLP модела върху данните за обучение. Параметърът `estimator` се задава на инстанцията `classifier` (KerasClassifier), а `X` и `y` се задават на данните за обучение.

10. Средната стойност и дисперсията на точността се изчисляват от резултатите от кръстосаното валидиране и се извеждат на конзолата.

Изборът на параметри в кода е както следва:

- `units`: Броят на единиците/невроните във всеки слой `Dense` определя капацитета и сложността на MLP модела. По-големият брой единици позволява на модела да изучава по-сложни представяния, но може да увеличи изчислителната сложност.

- `activation`: Функцията за активиране, използвана в скритите слоеве, е `relu`, която въвежда нелинейност и помага на модела да научи сложни връзки в данните. Функцията за активиране "relu" обикновено се използва в скритите слоеве на MLP.

- 'optimizer': Използва се оптимизаторът "adam", който е ефективен и широко използван оптимизационен алгоритъм за обучение на невронни мрежи. Той адаптира скоростта на обучение въз основа на градиентите и има вградени възможности за инерция и адаптивна скорост на обучение.

- `loss`: Избраната функция на загубите е 'binary\_crossentropy', която е подходяща за проблеми на двоична класификация. Тя измерва разликата между предсказаните вероятности и истинските етикети. Целта е да се минимизира тази разлика по време на обучението.

- `epochs`: Броят на епохите определя колко пъти MLP моделът се повтаря над цялото множество от данни за обучение. Всяка епоха се състои от преминаване напред, преминаване назад (обратно разпространение) и актуализации на теглото. Обучението за повече епохи позволява на модела потенциално да подобри представянето си, но твърде многото епохи могат да доведат до прекомерно приспособяване.

- `batch\_size`: Размерът на партидата определя броя на обучаващите извадки, които се обработват, преди да се актуализират теглата. Използването на по-малки партиди позволява по-чести актуализации на теглата, което потенциално води до по-бърза конвергенция и по-добро обобщение. Въпреки това по-малките размери на партидите могат да увеличат времето за обучение поради по-честите актуализации на теглата.

Като цяло кодът конструира MLP модел с помощта на Keras, състоящ се от множество `Dense` слоеве с 'relu' активация в скритите слоеве и 'sigmoid' активация в изходния слой за двоична класификация. Моделът е обучен с помощта на оптимизатора 'adam' и функцията за загуби 'binary\_crossentropy'. Изборът на броя на звената, функциите за активиране, оптимизатора и функцията за загуби зависи от конкретния проблем и характеристиките на данните, като се цели постигане на добър баланс между сложността на модела и ефективността на обобщението.

**5. Синтез на моделите за машинно обучение за диагностичен анализ - конфигуриране на REPO в GitHub със сорс кодовете на двата модела и линк към него. (дейтасет 2)**

[**GitHub Repo, съдържащо Jupyter Notebook**](https://github.com/stanislavstoyanov99/ProstateCancerML)

**Прилагане на класификационните алгоритми**

**Logistic Regression**

**A picture containing text, font, screenshot

Description automatically generated**

Фигура 42 - Логистична регресия

Logistic Regression Classification Test Accuracy: 0.8

**K-Neighbors (метод на съседите)**

**A picture containing text, screenshot, font

Description automatically generated**

Фигура 43 - KNN (метод на съседите)

Score for Number of Neighbors = 5: 0.8

**Naive Bayes**

**A picture containing text, font, screenshot

Description automatically generated**

Фигура 44 - Naive Bayes

Naive Bayes Test Accuracy: 0.8

**Decision Tree**

**A picture containing text, screenshot, font

Description automatically generated**

Фигура 45 - Decision Tree

Decision Tree Test Accuracy: 0.65

**Random Forest**

**A screenshot of a computer code

Description automatically generated with low confidence**

Фигура 46 - Random Forest

Random Forest Test Accuracy: 0.75

**ANN (невронна мрежа)**

A picture containing text, font, screenshot

Description automatically generated

Фигура 47 - ANN

A screenshot of a computer

Description automatically generated with low confidence

Фигура 48 - точност на невронната мрежа

**RNN (невронна мрежа)**

**A picture containing text, screenshot, font, document

Description automatically generated**

Фигура 49 - RNN

**A screenshot of a computer

Description automatically generated with low confidence**

Фигура 50 - точност на невронната мрежа

**MLP (невронна мрежа)**

**A picture containing text, screenshot, font, document

Description automatically generated**

Фигура 51 - MLP

**A picture containing text, font, number, line

Description automatically generated**

Фигура 52 - точност на невронната мрежа

**6. Експериментални резултати (дейтасет 2)**

**6А. Сравнителни диаграми за алгоритмите на оценките на параметрите PAS (Precision, Accuracy, Sensitivity) – дейтасет 2**

**Precision Score = TP / (FP + TP)**

**Sensitivity (Recall) = TP / (FN + TP)**

**Accuracy Score = (TP + TN)/ (TP + FN + TN + FP)**

**Logistic Regression – изчисление на precision, accuracy и sensitivity (recall)**

**A picture containing text, screenshot, font, receipt

Description automatically generated**

Фигура 53 - Logistic Regression – изчисление на precision, accuracy и sensitivity (recall)

**KNN – изчисление на precision, accuracy и sensitivity (recall)**

**A picture containing text, screenshot, font, receipt

Description automatically generated**

Фигура 54 - KNN – изчисление на precision, accuracy и sensitivity (recall)

**Naive Bayes – изчисление на precision, accuracy и sensitivity (recall)**

**A picture containing text, screenshot, font, receipt

Description automatically generated**

Фигура 55 - Naive Bayes – изчисление на precision, accuracy и sensitivity (recall)

**Decision Tree – изчисление на precision, accuracy и sensitivity (recall)**

**A picture containing text, screenshot, font, receipt

Description automatically generated**

Фигура 56 - Decision Tree – изчисление на precision, accuracy и sensitivity (recall)

**Random Forest – изчисление на precision, accuracy и sensitivity (recall)**

**A picture containing text, screenshot, font, number

Description automatically generated**

Фигура 57 - Random Forest – изчисление на precision, accuracy и sensitivity (recall)

**ANN – изчисление на precision, accuracy и sensitivity (recall)**

**A screenshot of a computer

Description automatically generated with medium confidence**

Фигура 58 - ANN – изчисление на precision, accuracy и sensitivity (recall)

**RNN – изчисление на precision, accuracy и sensitivity (recall)**

**A screenshot of a computer program

Description automatically generated with low confidence**

Фигура 59 - RNN – изчисление на precision, accuracy и sensitivity (recall)

**MLP – изчисление на precision, accuracy и sensitivity (recall)**

**A screenshot of a computer

Description automatically generated with medium confidence**

Фигура 60 - MLP – изчисление на precision, accuracy и sensitivity (recall)

**Извод**

От представените резултати за класификационните алгоритми: **ANN**, **RNN**, **MLP** като невронни мрежи се вижда, че всичките се представят сравнително добре с точност около 85%. При класификационните алгоритми логистичната регресия постига 80% точност, като с подобен резултат са KNN и Naive Bayes. С най-нисък резултат на точност е Decision Tree - 65%. Random Forest постига също добър резултат от 75%. Като цяло не бива да прави учудване, че именно невронните мрежи предоставят най-висок резултат поради сложността им на изпълнение.

**Изчертаване на сравнителна диаграма относно точностите на алгоритмите и невронните мрежи**

**A picture containing text, font, screenshot, line

Description automatically generated**

Фигура 61 - Изчертаване на сравнителна диаграма относно точностите на алгоритмите и невронните мрежи

A picture containing text, screenshot, rectangle, plot

Description automatically generated

Фигура 62 - резултат от изпълнението кода на фигура 61

**6Б. Сравнителни диаграми за алгоритмите на оценките на параметрите FP, FN, TP, TN – дейтасет 2**

**Logistic Regression**

**A picture containing text, screenshot, font

Description automatically generated**

Фигура 63 - създаване на heatmap за логистичната регресия

**A picture containing screenshot, text, rectangle, square

Description automatically generated**

Фигура 64 - резултат от изпълнението на кода на фигура 63

A screenshot of a computer program

Description automatically generated with low confidence

Фигура 65 - създаване на стълбовидна диаграма за Confusion Matrix

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

Фигура 66 - резултат от изпълнението на кода на фигура 65

**KNN**

A picture containing text, screenshot, font

Description automatically generated

Фигура 67 - създаване на heatmap за KNN

A picture containing screenshot, text, rectangle, design

Description automatically generated

Фигура 68 - резултат от изпълнението на кода на фигура 67

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

Фигура 69 - стълбовидна диаграма на Confusion Matrix за KNN

**Naive Bayes**

A picture containing text, screenshot, font

Description automatically generated

Фигура 70 - създаване на heatmap за Naive Bayes

A picture containing screenshot, rectangle, text, design

Description automatically generated

Фигура 71 - резултат от изпълнението на кода на фигура 70

A picture containing text, screenshot, rectangle, diagram

Description automatically generated

Фигура 72 - стълбовидна диаграма на Confusion Matrix за Naive Bayes

**Decision Tree**

A picture containing text, screenshot, font

Description automatically generated

Фигура 73 - създаване на heatmap за Decision Tree

A picture containing text, screenshot, rectangle, design

Description automatically generated

Фигура 74 - резултат от изпълнението на кода на фигура 73

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

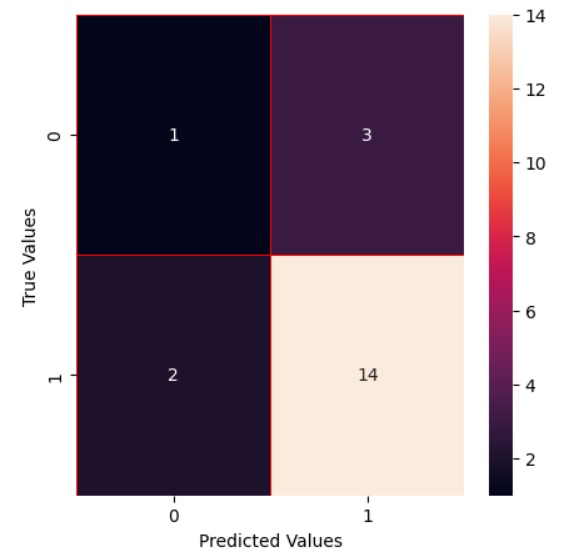
Фигура 75 - стълбовидна диаграма на Confusion Matrix за Decision Tree

**Random Forest**

A picture containing text, screenshot, font

Description automatically generated

Фигура 76 - създаване на heatmap за Random Forest



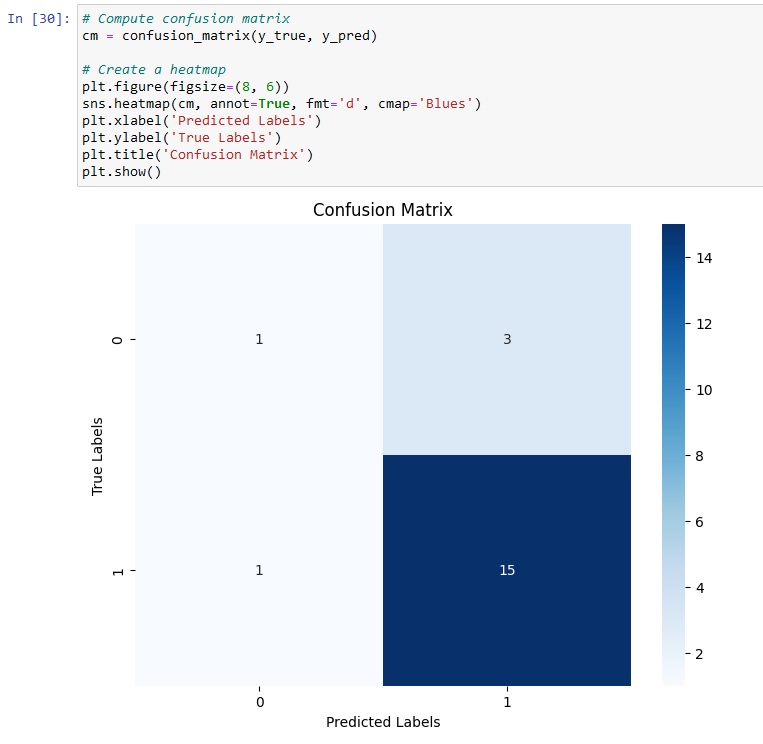
Фигура 77 - резултат от изпълнението на кода на фигура 76

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

Фигура 78 - стълбовидна диаграма на Confusion Matrix за Random Forest

**ANN**



Фигура 79 - създаване на heatmap за ANN

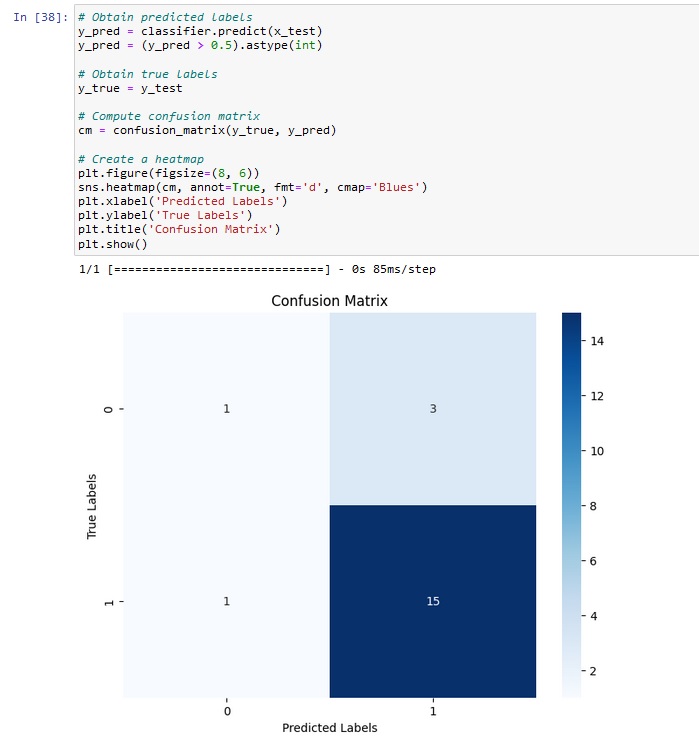
**RNN**

**A screenshot of a computer screen

Description automatically generated with low confidence**

Фигура 80 - създаване на heatmap за RNN

**MLP**



Фигура 81 - създаване на heatmap за MLP

**7. Информационни ресурси**

* <https://www.kaggle.com/competitions/prostate-cancer-grade-assessment/overview>
* <https://www.kaggle.com/datasets/sajidsaifi/prostate-cancer?datasetId=66762&sortBy=voteCount>
* <https://bg.wikipedia.org/wiki/Python>
* <https://en.wikipedia.org/wiki/Project_Jupyter>

1. <https://www.cancer.net/navigating-cancer-care/diagnosing-cancer/tests-and-procedures/digital-rectal-exam-dre> [↑](#footnote-ref-1)
2. <https://urology.bg/kakvo-e-psa-tumoren-marker-prostata/> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Gleason_grading_system> [↑](#footnote-ref-3)
4. <https://www.kaggle.com/competitions/prostate-cancer-grade-assessment/overview> [↑](#footnote-ref-4)
5. <https://www.kaggle.com/datasets/sajidsaifi/prostate-cancer?datasetId=66762&sortBy=voteCount> [↑](#footnote-ref-5)
6. <https://github.github.com/gfm/> [↑](#footnote-ref-6)