Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Проект по предметот

Податочно рударство

Тим:

Марија Илиевска 171056

Елена Каранфиловска 171048

Магдалена Крежеска 171013

Октомври 2020, Скопје

Содржина

1. Апстракт

Музиката несомнено игра важна улога во нашите животи, инспирира, забавува, зближува… Секојдневно расте бројот на луѓе кои слушаат музика преку стриминг музичките сервиси кои што овозможуваат пристап до виртуелен, неограничен каталог на музика. Популарноста на овие сервиси во огромна мера се должи на опцијата за предлог на музика, базирано на преференциите на корисникот, т.е што слушал претходно и дали песната ја додал во омилени или не. Но, за да се предвиди дали на корисникот ќе му се допадне одредена песна сепак претставува комплексен проблем и треба да се земат во предвид различни карактеристики кои ги имаат песните кои веќе му се допаднале на корисникот или ги слуша постојано. На овој начин корисникот може да открие нови, дотогаш нечуени песни, а стриминг сервисите да си ја зголемат популарноста и да го задржат вниманието на корисникот.

2. Вовед

Spotify е еден од најпознатите стриминг сервиси. Еден од многуте интересни и корисни сервиси што ги нуди е предлог плејлиста секој понеделник со 30 песни за кои смета дека најдобро го рефлектираат вкусот на корисникот. Ваквата услуга овозможува откривање на одлична музика, која можеби инаку никогаш нема да ја сретнете и со тоа се зголемува задоволството кај корисникот. Тука влијаат многу фактори и се прават комплексни анализи на корисникот и неговата интеракција на апликацијата или како што тие го нарекуваат "taste profile". Токму ваквиот проблем беше инспирација за нашето истражување, иако во многу поедноставена верзија. Анализите на Spotify секако се далеку покомплексни и се земаат во предвид повеќе фактори, како претходно слушани песни, но и што слушаат луѓето со сличен вкус и што додаваат во нивните плејлисти.

Главната цел на овој проект е да конструираме модели кои ако корисникот издвои две плејлисти една со песни кои му се допаѓаат и една со песни кои не му се допаѓаат, моделот со помош на аудио карактеристиките на песните да може да

препознава шеми во музиката и успешно да предвиди во иднина дали одредена песна ќе му се допадне или не на корисникот.

За таа цел податоците ги собравме од Spotify API каде што се обезбедени податоци за акустичните карактеристики на секоја песна врз основа на анализа на аудио сигнал и дополнително со скрапирање на веб страната http://organizeyourmusic.playlistmachinery.com/ за да го добиеме жанрот на песните. Овој атрибут се покажа исклучително важен во класификацијата кои песни се “good”, а кои “bad”.

Изградивме повеќе модели и добивме релативно задоволителни резултати, особено ако се земе предвид дека вкусот за музика е многу лична работа и може да биде фрустрирачки тешко да се опише зошто сакаме или не сакаме одредена песна.

3. Собирање на податоците

Со цел да изградиме добар модел кој ќе предвидува дали одредена песна ќе му се допаѓа на даден корисник или не, најпрво треба да го изградиме податочното множество. За таа цел ќе се користат податоци кои ќе се извлечат со помош на Spotify Web API (spotipy). За таа цел потребно е да се има Spotify профил и листи со песни на тој профил. Исклучиво за потребите на овој проект е направен профил и две листи со песни, од кои едната ги содржи песните кои припаѓаат на позитивната класа(песни кои според корисникот се „убави“), а другата ги содржи песните кои припаѓаат на негативната класа(песни кои според корисникот се „лоши“). Со помош на ова API, најпрво се извлекуваат листите со песни на корисникот, за секоја листа се зема нејзината содржина - песните, и на крај за секоја песна се извлекуваат нејзините аудио катактеристики(audio features). По ова, за секоја песна во податочното множество се чуваат податоци за:

- ***Energy*** - *float* - Вредностите на оваа карактеристика се движат од 0.0 до 1.0 и претставува перцептивна мерка на интензитетот и активноста на песната. Обично, ако песната е брза и гласна ќе има повисока вредност на оваа карактеристика.

- **LIVENESS** - *float* - Оваа катактеристика го претставува количеството на публика на аудио снимката. Што повисока вредност на овој атрибут, значи дека е поголема веројатноста дека песната/аудио снимката е снимена во живо.
- ***TEMPO*** - *float* - Вкупно проценето темпо на песната, мерено во beats per minute(BPM), ја претствавува брзината на исполнување на песната.
- ***SPEECHINESS*** - *float* - Оваа карактериските го детектира присуството на говор во песната. Вредностите кои може да ги има се од 0.0 до 1.0. Вредности над 0.66 се добиваат за аудио снимки кои се најверојтно составени само од изговорени зборови, вредности помеѓу 0.33 и 0.66 се добиваат за аудио снимки кои се содржаи и од музика и од говор(на пример рап музика), а вредности помали од 0.33 се добиваат за снимки кои се состојат од музика.
- ***ACOUSTICNESS*** - *float* - Претставува која е веројатноста аудио снимката да е акустична(колку е повисока вредноста на оваа карактеристика, поголема е веројатноста аудио снимката да е акустична).
- ***INSTRUMENTALNESS*** - *float* - Предвидува дали аудио снимката не содржи вокали. „Ohh“ и „Ahh“ звуците во овој контект се сметаат како инструментален дел од снимката. Рапот и говорот се чисто „вокални“. Колку е поблиску вредноста до 1.0, толку е поголема веројатноста дека аудиото е чисто инструментално.
- ***TIME_SIGNATURE*** - *int* - Проценет вкупен временски потпис на снимката.
- ***DANCEABILITY*** - *float* - Опишува колку дадена песна/аудио снимка е погодна за играње врз основа на темпото, стабилноста на ритамот и јачината на битот. Може да има вредности од 0.0 - песна на која не може да се игра, до 1.0.
- ***KEY*** - *int* - Вкупна проценета скала на песната, мапирана во цел број со користење на Pitch Class notation стандардот. На пример: 0 = C, 1 = C♯/D♭, 2 = D итн.
- ***DURATION_MS*** - *int* - Времетраење на песната/аудио снимката во милисекунди.
- ***LOUDNESS*** - *float* - Ја мери јачината на звукот во децибели.

- ***Valence*** - *float* - Има вредности од 0.0 до 1.0, ја опишува позитивнота која ја пренесува песната. Колку поголема вредност, толку песната е попозитивна.
- ***Mode*** - *int* - Дава информација дали песната е minor - 0 или major - 1.
- ***Type*** - *string* - Има иста вредност, „audio_features“, за секоја песна.
- ***Popularity*** - *int* - Ја претставува популарноста на песната на скала од 0 до 100.
- ***Uri*** - *string* - Уникатно Spotify URI за секоја песна.
- ***Title*** - *string* - Наслов на песната.
- ***Artist*** - *string* - Артистот чијашто е песната.

По разгледувањето на овие карактеристики дојдено е до заклучок дека би било добро ако покрај сите нив има и жанр на песната и годината на нејзиното издавање. За таа цел, се користи веб сајтот http://organizeyourmusic.playlistmachinery.com/ каде песните се организираат по повеќе карактеристики, меѓу кои и жанрот и годината. За влечење на соодвените податоци се користи пакетот Selenium. По ова, во подарочното множество се додаваат и тие две карактеристики.

На крај, за секоја песна се додава уште еден атрибут, „target“ , кој за песните од листата на „убави“ песни е 1, а за песните од листата на „лоши“ песни е 0.

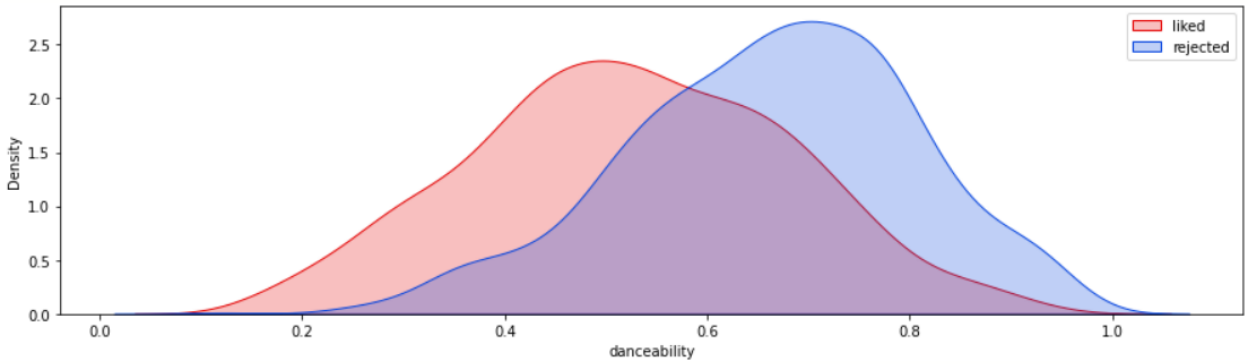
4. Визуелизација

Пред да почнеме со градењето на модели за класификација, најпрвин потребно е да се запознаееме со нашето податочно множество. Претходно ги разгледавме атрибутите од кои е составено нашето податочно множество, од каков тип се и кое е нивното значење.

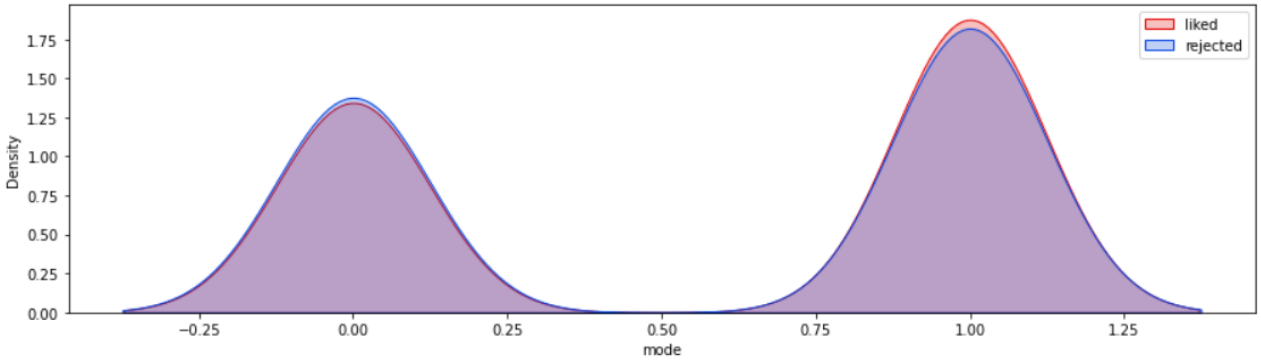
Во овој дел ќе се обидеме да добиеме уште појасна претстава за атрибутите и интервалот на вредности кои што можат да ги примаат.

За таа цел, истражувањето на податоците ќе го направиме преку визуелизација на распределбата на атрибутите кои што се присутни во нашето податочно множество.

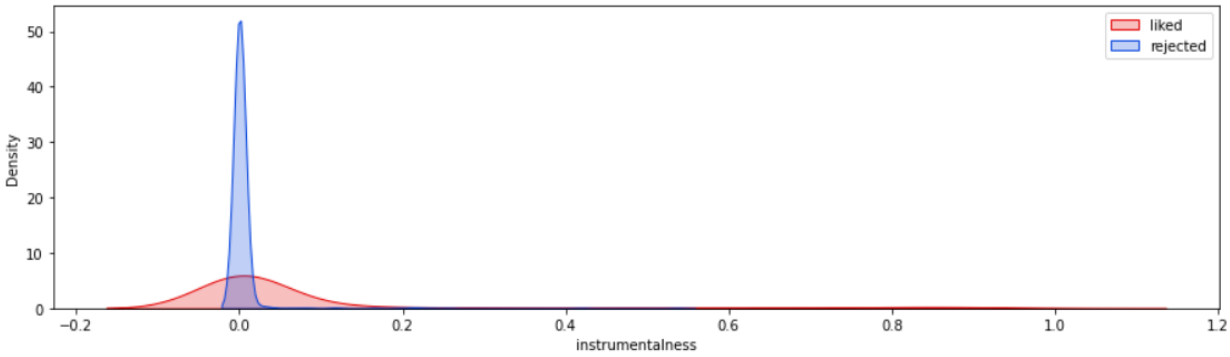
Најпрвин ќе ја разгледаме распределбата на атрибутите и на двете класи песни и со тоа можеме да добиеме некаква груба претстава за тоа како и колку влијаат атрибутите на тоа дали ни се допаѓа или не ни се допаѓа одредена песна.



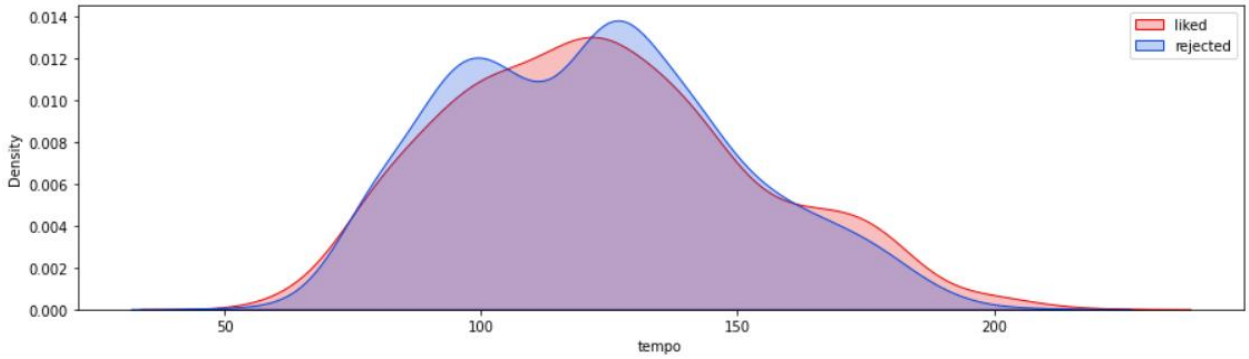
Можеме да забележеме да атрибутот danceability има пониски вредности кај песните означени како песни што ни се допаѓаат.



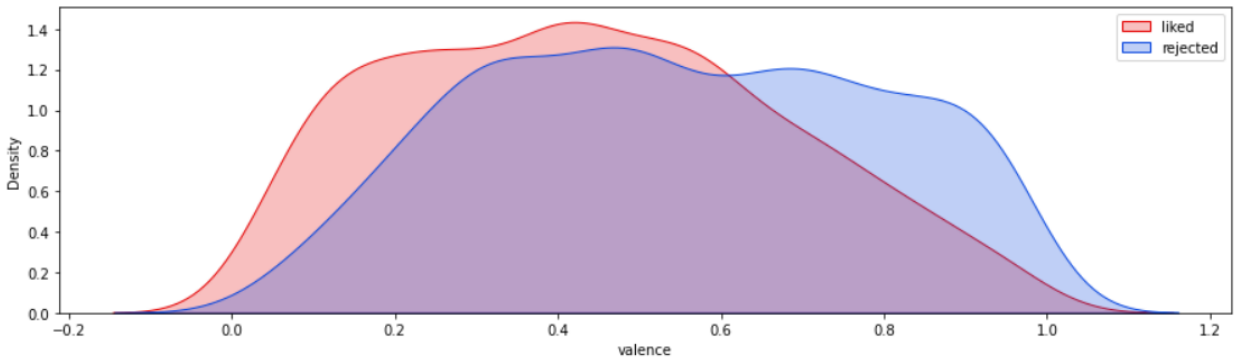
Атрибутот mode има прилично исти вредности кај двете класи песни.



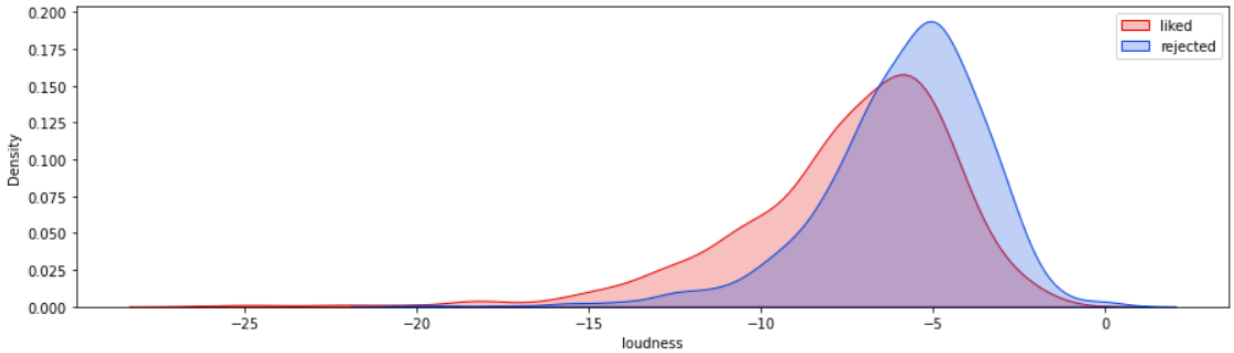
Во класата на песни кои не ни се допаѓаат најмногу се вклучени песни кои имаат ниско ниво инструменталност, т.е висока вокалност како рап и слично.



Темпото е приближно исто и за двете класи и е главно умерено.

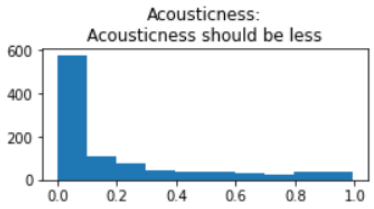


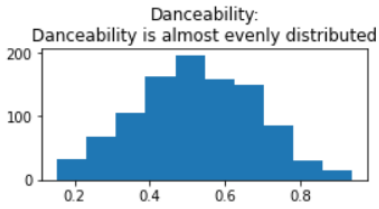
Атрибутот валентност е покажува различна распределба кај двете класи песни. Песните означени како негативна класа т.е песни кои не ни се допаѓаат имаат висока валентност односно се попозитивни.

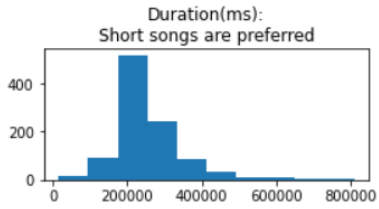


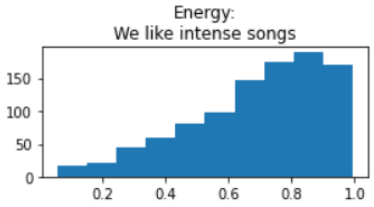
Атрибутот гласност има повисоки вредности за песните означени како песни што не ни се допаѓаат.

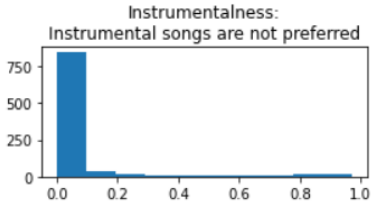
Следно, со помош на *хистограми* ќе ја разгледаме распределбата на секој од атрибутите во класата песни означени дека ни се допаѓаат.

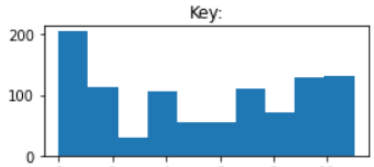


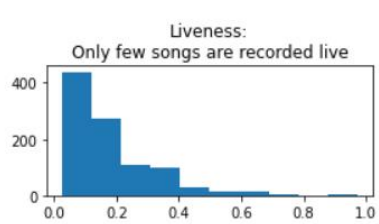


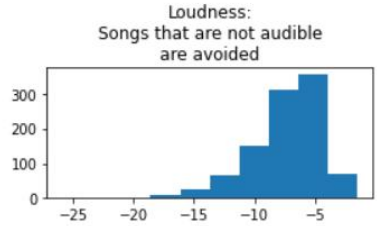


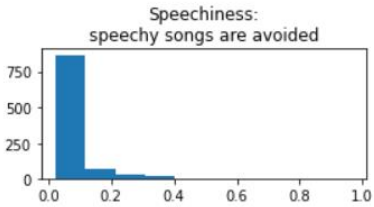


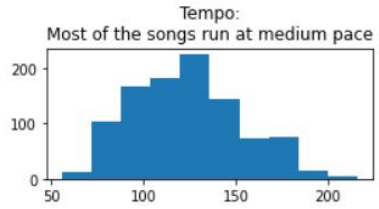


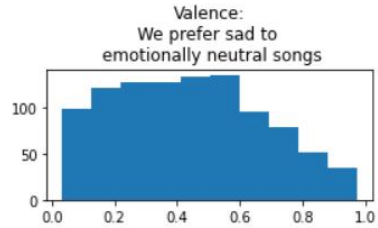












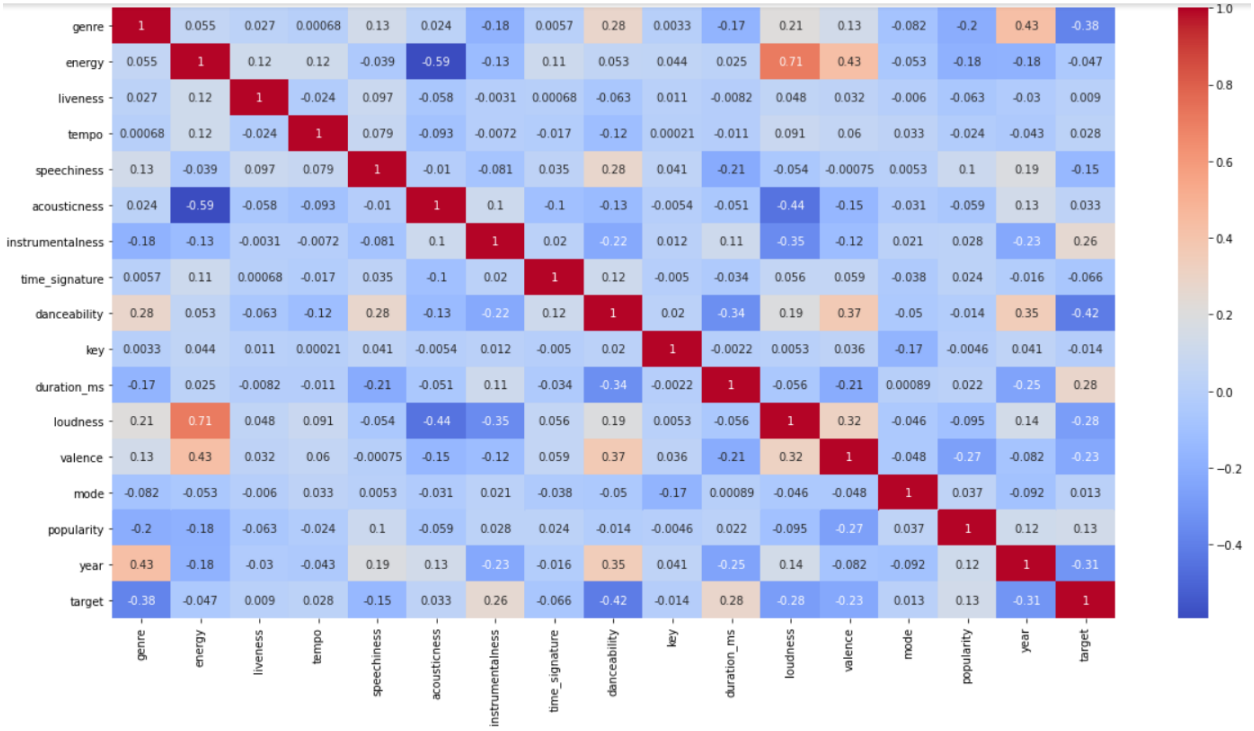
5. Претпроцесирање

5.1. Label Encoding

Како прв чекор на претпроцесирање, со цел компатибилност со повеќето алгоритми за класификација кои прифаќаат само нумерички атрибути, во помош на Label Encoder, правиме асоцијација на жанровите со цели броеви, за да може жанрот подоцна да се користи како нумерички атрибут.

5.2. Корелација

Како следен чекор ќе ја разгледаме корелираноста меѓу атрибутите, но и меѓу таргет атрибутот и останатите атрибути. *На следната слика се претставени*



Пирсоновите коефициенти на корелација меѓу секој пар атрибути.

Овие коефициенти ја покажуваат линеарната зависност меѓу два атрибута, при што може да примат вредност од -1 до 1, при што вредностите близу 0 покажуваат дека тие два атибути не се линеарно корелирани. Перфектна линеарна врска (корелацијата да е -1 или 1) би значело дека едниот од атрибутите може да биде објаснет преку линеарна функција од другиот атрибут.

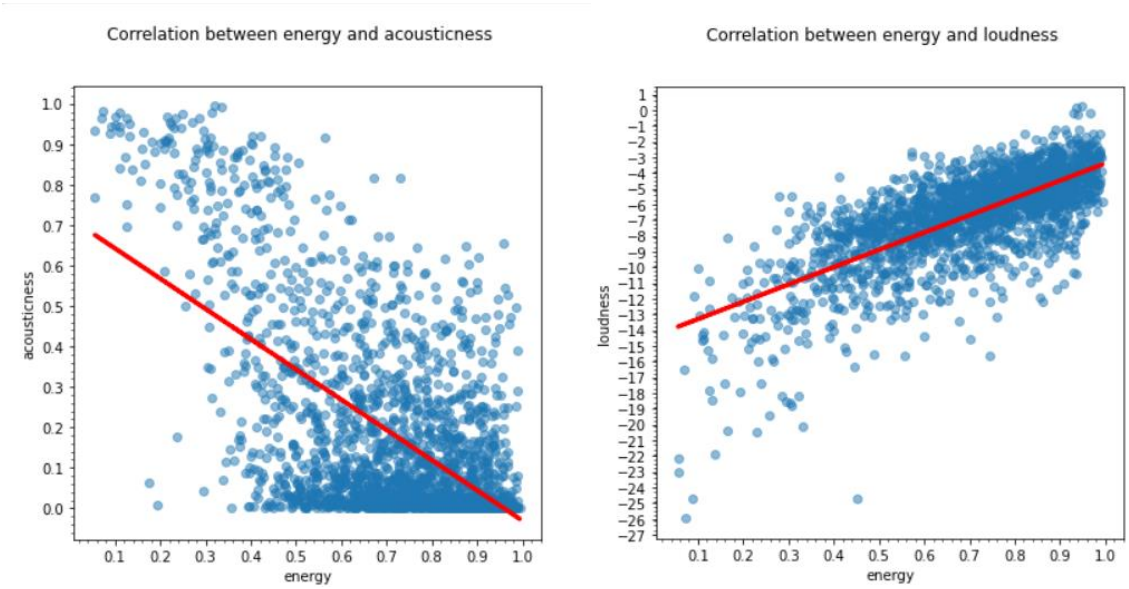
Може да се забележи дека таргет атрибутот со останатите атрибути не покажува големи вредности во однос на корелираноста, па тоа сугерира дека линеарен класификатор веројатно нема да биде соодветен.

Истовремено можеме да забележиме дека некои од останатие атрибути покажуваат силна зависност меѓусебе. Енергијата и гласноста се високо позитивно корелирани со вредност од 0.71, како и енергијата и акустичноста кои што се негативно корелирани со вреднот -0.59, и некои послаби корелации.

Вообичаено сакаме да избегнеме да работиме со атрибути кои што се силно корелирани едни со други и со тоа да немаме редунданција главно заради две причини:

- моделите се поедноставни и полесни за интерпретација,
- кога податочното множество е големо, намалувањето на димензионалноста може драстично да го намали времето на извршување на алгоритмите.

Дополнително, за атрибутите за кои видовме во минатиот чекор дека постои висока корелираност, нивната врска ќе ја моделираме и со помош на линеарна регресија.



Корелацијата и линеарната регресија се методи со кои можеме да одредиме дали два нумерички атрибути се линеарно зависни.

Корелацијата обезбедува информации за јачината и насоката на линеарната врска меѓу два атрибута, додека линеарната регресија ги проценува параметрите во линеарно равенство со кое се предвидуваат вредностите на едниот атрибут базирано на вредностите на другиот атрибут.

Како следен чекор ќе примениме Principal Component Analysis (PCA). PCA се користи за намалување на димензионалноста на податочното множество и ако имаме големо множество во кое атрибутите се високо корелирани, PCA го сведува на помало, помалку корелирано множество кое и понатаму ги содржи најголемиот дел од информациите од првичното множество.

5.3. Linear Discriminant Analysis

Идејата позади LDA е едноставна. Од математичка гледна точка, треба да се најде нов простор на карактеристики каде што ќе се пресликаат податоците со цел да се максимизира сепарабилноста на податоците. LDA може да се користи во два контексти: може да се користи за класификација, но и за намалување на димензионалноста. Тука, пробавме да ја намалиме димензионалноста со помошта на овој алгоритам, со што бројот на диманзи од оригиналниот број(16) се намали на C - 1, каде што C е бројот на класи(2). По ова, податочното множество има само една карактеристика, која според алгоритмот е „најважна“(најдобро ги разделува податоците) - жанрот.

Во креирањето на моделите не продолживме со користење на ова податочно множество, бидејќи сепак намалувањето на димензионалноста со LDA се користи за мулти-класни класификации.

Како што видовме, и PCA и LDA се користат за намалување на димензионалноста, но за разлика од LDA, PCA се опишува како „unsupervised“ алгоритам, бидејќи ги игнорира класните лабели(„target“) и целта е да најде насоки - „principal components“, кои максимизираат дисперзијата на податочното множество.

5.4. Standard Scaler

Стандардизацијата на податочното множество е чест дел од претпроцесирањето, бидејќи многу естиматори од машинско учење се однесуваат лошо доколку одредени карактеристики не личат, повеќе или помалку, како нормално распределени податоци. Затоа користиме стандардизација на карактеристиките со одземање на средната вредност и скалирање на варијансата $z = (x - m)/s$.

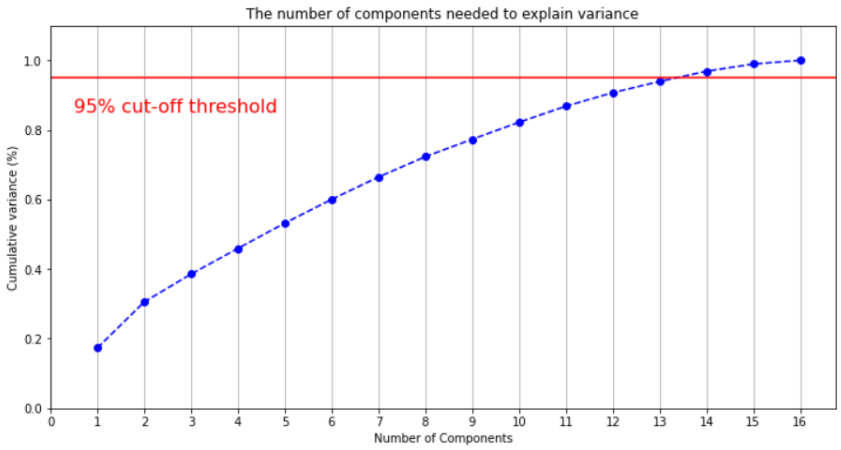
5.5. PCA - Principal Component Analysis

Како што споменавме порано, може да биде особено корисно да се поедностават нашите модели и да се користат што помалку атрибути за да се постигне најдобриот резултат. PCA е метод за намалување на димензионалноста на податочното множество, но притоа да се зачува што поголема варијација, односно статистички информации.

Тоа го прави со создавање на нови, некорелирани атрибути кои сукцесивно ја зголемуваат варијансата и овозможува сумирање на информациите од голем број карактеристики во ограничен број компоненти кои се линеарни комбинации од оригиналните карактеристики. Наоѓањето на вакви нови атрибути, PCA го сведува на проблем на наоѓање на сопствени вектори и тие го дефинираат новиот простор.

Меѓутоа, бидејќи PCA ја користи апсолутната варијанса на одреден атрибут за да ги ротира податоците, атрибутите со поширок опсег на вредности ќе го пристраснат алгоритмот во однос на другите атрибути. За да се избегне ова, прво ги нормализираме нашите податоци.

Како што се гледа на сликата подолу, занашето податочно множество добивме дека оптимален број атрибути е 13, а вкупниот беше 16.



Од графикот можеме да забележеме дека сината линија којашто ја претставува кумулативната варијанса, не е перфектно линеарна, но е прилично блиску до права линија. Тоа значи дека скоро сите карактеристики придонесуваат подеднакво за објаснување на варијансата на податоците.

Дополнително, знаеме дека PCA не ги зема во предвид класите па тоа би значело дека може да се отфрли атрибут кој многу придоесува за прецизна разделба на податоците во класи и понекогаш примената на PCA може да доведе до полоши резултати во проблемите со надгледувано учење т.е класификација. Тоа беше случај и со нашето податочно множество т.е ги констуиравме моделите со и без примена на PCA и резултатите во вториот случај беа значително подобри.

5.6. Делење на податочното множество

Податочното множество го поделивме на тренирачко, валидациско и тестирачко множество, така што првите 70% од секоја од класите се доделени во тренирачкото множество. Следните 10% од секоја од класите влегуваат во валидациско множество, а последните 20% од секоја од класите се дел од тестирачкото множество. На крај се прави shuffle на множествата (заедно со соодветните лабели).

Кај моделите каде што нема потреба од валидациско множество беа споени тренирачкото и валидациското множество, односно првите 80% од секоја од

класите беа доделени во тренирачкото, а последните 20% од секоја од класите беа доделени на тестирачкото множество.

6. Модели на класификација

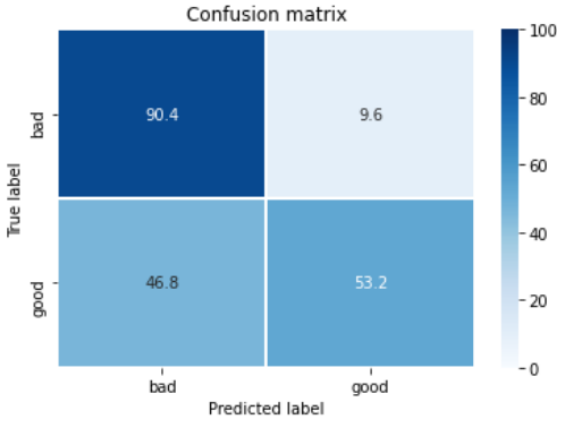
6.1. Наивен Баесов класификатор

Наивниот Баесов класификатор е статистички класификатор базиран на Баесовата теорема и е пример за надгледувано учење. Се користи за класифицирање на објектите во класи според најголемата веројатност за припаѓање на една класа. Тој сите атрибути, заедно со класниот атрибут ги смета како случајни променливи. Ако имаме еден запис со атрибути (X1, X2,...,Xn), целта е да ја предвидиме класата C(атрибутот target во нашиот случај), поточно треба да ја најдеме C која има максимална вредност за P(C|X1, X2,...,Xn). Овој класификатор се нарекува „наивен“ бидејќи претпоставува дека карактеристиките на објектот се независни една од друга. Оваа претпоставка скоро секогаш е неточна, но сепак методот се покажува како ефективен за голем број проблеми. Исто така, поради оваа претпоставка, пред да се пресмета на која класа и припаѓа влезниот објект, мора да се пресмета веројатноста со која секоја од карактеристиките припаѓа на одредена класа. По ова, веројатностите се множат, со што се добива веројатноста со која влезниот објект и припаѓа на таа класа. Процесот се повторува за секоја од класите со цел да се најде класата во која дадениот влезен податок е највероајтно да припаѓа. Откако ќе се разгледаат сите карактеристики и сите класи, онаа класа со најголема веројатност ќе се додели како класа на влезниот податок.

Бидејќи податочното множество кое се користи во овој проект има карактеристики кои имаат непрекинати вредности, се одлучивме да користиме Гаусов наивен Баесов класофокатор, кој претпоставува дека секоја карактеристика има нормална распределба.

Како што видовме претходно, поради корелацијата на карактеристиките, со овој модел се добиваат послаби резултати во споредба со оние кои ќе ги обработиме подоцна.

Точноста на моделот е 73%, со тоа што моделот добро ги класифицира песните кои припаѓаат на негативната класа(true negative) - 91%, но прилично лошо ги класифицира песните кои припаѓаат на позитивната класа - 55%, што значи дека има поголем дел објекти од тестирачкото множество кои припаѓаат на позитивната класа, но нашиот модел ги класифицирал во негативната класа(45% false negative rate).



Naive Bayes score on test:

	precision	recall	f1-score
0	0.66	0.90	0.76
1	0.85	0.53	0.65
accuracy			0.72
macro avg	0.75	0.72	0.71
weighted avg	0.75	0.72	0.71

6.2. Дрво на одлука

Учењето со дрвата на одлучување е една од најупотребуваните техники за класификација. За едноставни множества податоци, точноста на класификувањето е споредлива со останатите техники на класификување.

Класификаторот претставува дрво кое се гради рекурзивно со методот раздели и владеј, при што се конструираат многу линеарни хиперамнини.

Дрвото на одлучување или класификациското дрво ги има следниве својства:

- Секој внатрешен јазел означува тест според даден атрибут.
- Секоја гранка претставува исход од даден тест.
- Секој лист претставува припадност на одредена класа.

Дрва на одлучување се структури во облик на дрво кои претставуваат групи на одлуки. Овие одлуки генерираат правила за класификација на податоците.

Дрвата за одлучување користат „лакоми“ (greedy) алгоритми при градењето на стеблото. Податоците се делат врз основа на исходот од дадениот тест според одреден атрибут кој оптимизира одреден критериум.

Некои алгоритми на дрвата на одлучување кои се користат за класификација се Classification and Regression Trees (CART), Chi Square Automatic Interaction Detection (CHAID), ID3, C4.5. Тие обезбедуваат збир на правила кои може да се применуваат на нови (некласифицирани) податоци за да се предвиди кои податоци ќе го имаат дадениот исход.

Решавањето на проблемот на класификација со користење на дрва на одлучување е процес кој се изведува во две фази:

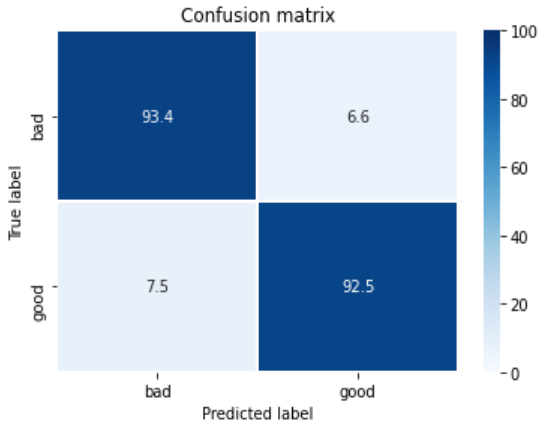
- Конструкција на дрво на одлучување со користење на множество податоци за обучување.
- За секој податок од множеството за тестирање, се применува дрвото за одлучување да се одреди припадноста на одредена класа.

Конструираното дрво ја претставува потребната логика за извршување на класификацијата.

Постојат многу предности за употреба на дрва на одлучување за класификација. Тие се ефикасни и се лесни за употреба. Може да се генерираат правила кои се лесни за интерпретација и разбирање.

Во нашиот модел како метрика врз база на која се избираат атрибутите за поделба беше користена Gini impurity.

Точноста на добиениот модел е 93%, притоа може да се забележи дека моделот многу добро ги предвидува и двете класи и тоа негативната класа (true negative) - дури 93,4%, а позитивната класа (true positive) – 92,5%.



Decision Tree score on test:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	198
1	0.93	0.93	0.93	201
accuracy			0.93	399
macro avg	0.93	0.93	0.93	399
weighted avg	0.93	0.93	0.93	399

6.3. Random Forest

Random Forest претставува алгоритам за машинско учење и е еден од многуте алгоритми за надгледувано учење кој може да се користи и за класификација и за регресија.

Поради тоа што прави предвидувања со комбинирање на резултатите од повеќе класификатори (слаби дрва на одлука т.н шума од дрва на одлуки), спаѓа во категоријата на учење со ансамбли.

Кај ваквиот тип на учење, податоците се независно класифицирани од страна на неколку класификатори, а потоа нивните резултати се комбинираат на некаков начин. Дури и ако некој од класификаторите понекогаш греши во класирањето, мнозинското гласање (што се користи кај Random Forest) или некој друг систем на комбинирање на резултатите во ансамблот веројатно ќе доведе до точен резултат.

Главната цел при градењето на ансамблот е да има ниска корелација меѓу класификаторите кои го сочинуваат.

За да се обезбеди ова алгоритамот Random Forest дрвата на одлука ги гради на таков начин што секое дрво се тренира со различно случајно подмножество од обучувачкото множество притоа за секој јазол во секое дрво, се зема само случајно подмножество на атрибутите..

Оваа метода на добивање различни множества за обука со иста големина како и оригиналното со употреба на статистичката техника *bootstrap* е позната како bagging (“**bagging**” = “**b**ootstrap” + “**agg**regat**ing**”).

Неколку од можните параметри кои можат да се подесат во Random Forest се бројот на дрва што треба да се комбинираат, максимална длабочина на дрвјата, максималниот број на опции што се разгледуваат при секое поделба, дали bagging/bootstrapping се врши со или без замена итн.

За нашиот проблем, како најдобри вредности за параметрите, со користење на RandomizedSearchCV, беа избрани следните:

Best parameters set:

bootstrap: True

criterion: 'entropy'

max_depth: None

max_features: None

max_samples: 0.5

min_samples_leaf: 1

min_samples_split: 5

n_estimators: 100

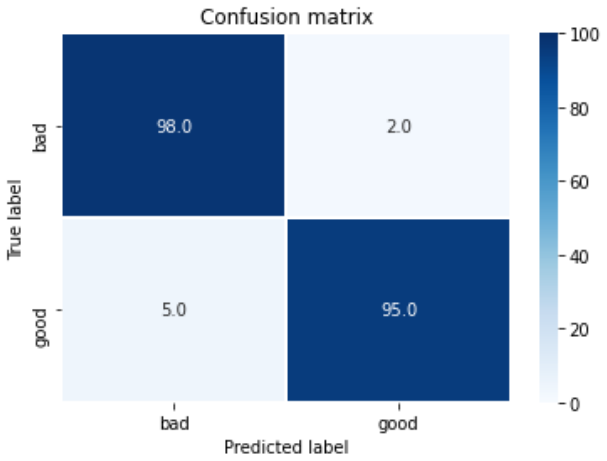
oob_score: False

Со вака подесени параметри добивме подобри резултати – точност од 96,4% во споредба со резултатите добиени од произволен Bagging класификатор – 88%. Од матрицата на конфузија, прикажана на сликата, може да се забележи дека моделот многу добро ги предвидува и двете класи и тоа негативната класа (true negative) - дури 98%, а позитивната класа (true positive) - 95%.

	precision	recall	f1-score	support
0	0.95	0.98	0.97	198
1	0.98	0.95	0.96	201
accuracy			0.96	399
macro avg	0.97	0.97	0.96	399
weighted avg	0.97	0.96	0.96	399

accuracy: 0.9649122807017544

[]



Друга важна особина на Random Forest е тоа што дава проценка за тоа кои атрибути се важни во класификацијата. Важноста на атрибутите може да се извлече од обучен модел која се добива како сума на намалувањата кај Gini метриката на сите темиња разгранети за тој атрибут во Random Forest.

Во нашиот случај атрибутот со најголема важност е ***genre***, што ја потврдува нашата претпоставка.

```
importances = list(rfc.feature_importances_)

most_important_feature = importances.index(max(importances))
least_important_feature = importances.index(min(importances))
print('Most important feature: '+ df.columns[most_important_feature])
print('Least important feature: '+ df.columns[least_important_feature])
```

```
Most important feature: genre
Least important feature: time_signature
```

6.4. XGBoost

Ансамблите се често користени механизми при изградбата на алгоритмите за машинско учење. Ансамблите можат да се имплементираат на различни начини и да се комбинираат различни алгоритми за машинско учење. Во овој случај основните класификатори претставуваат дрва на одлука, а алгоритамот кој што е имплементиран е алгоритамот наречен boosting, поточно gradient boosting.

Алгоритамот за boosting на дрвата на одлука се темели на учење од грешките на претходните дрва на одлука што значи дека дрвата на одлука не се градат засебно туку едно по друго. (Одлуките кои ги донесуваме ретко се изолирани и независни. Нашите моментални избори зависат од предходните, но и од антиципираните следни избори и нивните резултати.) Секое ново дрво на одлука учи од грешките кои што ги направиле претходните дрва на одлука. Поради тоа што учи од грешките на претходните дрва на одлука на алгоритамот му треба помало време (помал број на итерации) за да го предвиди резултатот.

Постојат повеќе типови на на boosting алгоритми и тоа AdaBoost, Gradient Boosting и e**X**treme **G**radient **Boost**ing односно XGBoost. Разликата е во начинот на учење на новите класификатори.

Кај алгоритмите кои се темелат на gradient boosting, како што кажува и самото име, наместо доделување на тежини на примероците за тестирање (како што прави AdaBoost), се користат градиентите во функциите на загуба.

Во алгоритамот Gradient Boosting дрвата се градат едно по друго. Во секоја итерација се гради по едно дрво на одлука при што предвидувањето направено со тоа дрво на долука се споредува со вистинскиот резултат. Таа разлика меѓу превидувањата на алгоритамот и вистинскиот резултат можат да се претстават со функција на загуба.

Во класификациските алгоритми функцијата на загуба претставува цена која што е платена за грешната класификација односно за грешката во предвидувањата, а главна цел е минимизирање на истата.

Постојат разни функции на загуба и се разликуваат меѓу класификациските и регресиските проблеми. Во овој случај бидејќи се работи за проблем на класификација како функција на загуба се користи cross entropy (позната и како логаритамска функција на загуба log loss) .

Идејата е да се поправи грешката направена од претходниот модел, да се научи од неа и со тоа во наредниот чекор да се подобрат перформансите. Ова се повторува се додека нема простор за дополнително подобрување.

За нашиот проблем, како најдобри вредности за параметрите, со користење на RandomizedSearchCV, беа избрани следните:

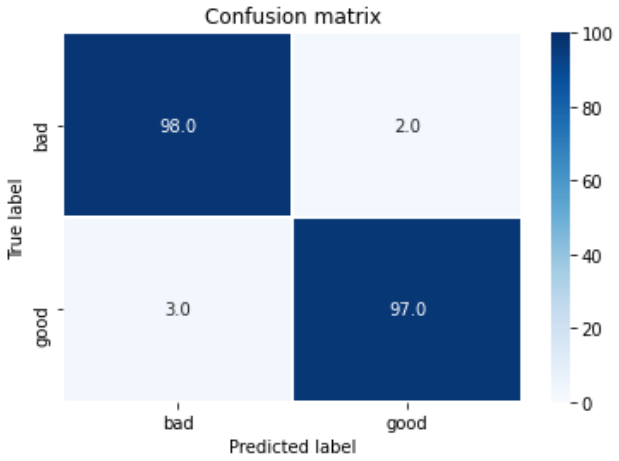
Best parameters set:

colsample_bytree: 0.9310027847448368
gamma: 2.4271218554791196
learning_rate: 0.2282336457043735
max_depth: 8
min_child_weight: 7.183432442939054
n_estimators: 35
reg_alpha: 0.6721642969744801
subsample: 0.9943412796064232

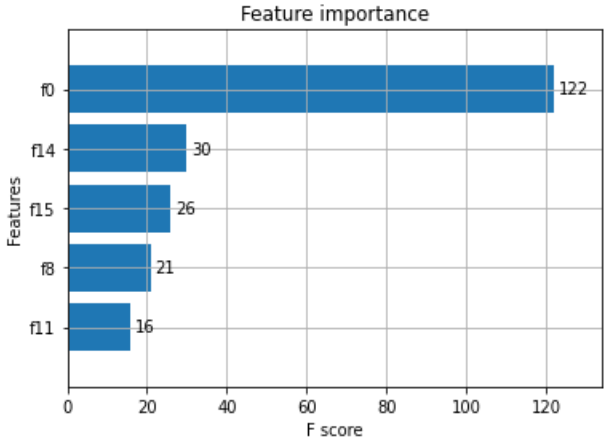
Со вака подесени параметри добивме најдобри резултати – точност од 97% и може да се забележи дека моделот многу добро ги предвидува и двете класи и тоа негативната класа (true negative) - дури 98%, а позитивната класа (true positive) - 97%.

	precision	recall	f1-score	support
0	0.97	0.98	0.97	198
1	0.98	0.97	0.97	201
accuracy			0.97	399
macro avg	0.97	0.97	0.97	399
weighted avg	0.97	0.97	0.97	399

accuracy: 0.974937343358396



Слично како Random Forest и XGBoost дава проценка за тоа кои атрибути се важни во класификацијата.



6.5. ANN

Вештачка невронска мрежа (Artificial Neural Networks – ANN) или поедноставно невронска мрежа е една од најуспешните компјутерски парадигми инспирирана од начинот на кој природните мозоци ја обработуваат информацијата, составена од густо меѓусебно поврзани паралелни структури.

Невронските мрежи имаат голема сличност со биолошките невронски мрежи, како што е мозокот, и поголемиот дел од користената терминологија е позајмена од областа на неврологијата.

Невроните во биолошкиот мозок меѓусебно се поврзани со врски кои се нарекуваат синапси. Секој неврон поседува илјадници врски со други неврони преку кои константно добива сигнали што треба да бидат стигнат до телото на клетката (невронот). Доколку резултантната сума од сигналите надмине одреден дефиниран праг (threshold), преку аксонот се испраќа одговор. Тоа значи дека биолошкиот неврон ги сумира сигналите кои пристигнуваат до него, добиената сума ја споредува со одреден праг и доколку таа го надмине прагот, невронот испраќа одреден излезен сигнал.

Најосновните компоненти на невронските мрежи се моделирани токму според структурата на човечкиот мозок. Вештачката невронската мрежа се состои од множество на вештачки неврони кои меѓусебно комуницираат со испраќање на сигнали едни на други преку голем број на тежински врски. Притоа невроните преку процесот на учење вршат нагодување на тежините на тие врските.

Кај невронските мрежи може да се разгледуваат три вида на неврони: влезни неврони кои ги примаат податоците надвор од невронската мрежа, излезни неврони кои ги праќаат излезните податоци надвор од невронската мрежа, и скриени неврони чии влезни и излезни сигнали остануваат внатре во мрежата.

Секој неврон извршува релативно едноставна работа, тој ги прима влезните сигнали од невроните од претходниот слој или надворешните извори и ги користи да го пресмета излезниот сигнал кој е проследен до невроните од следниот слој.

За нашиот проблем беше користена повеќеслојна невронска мрежа (feed-forward) која е надградба на еднослојниот перцептрон, првата топологија во ова област која се состои од едно ниво на неврони.

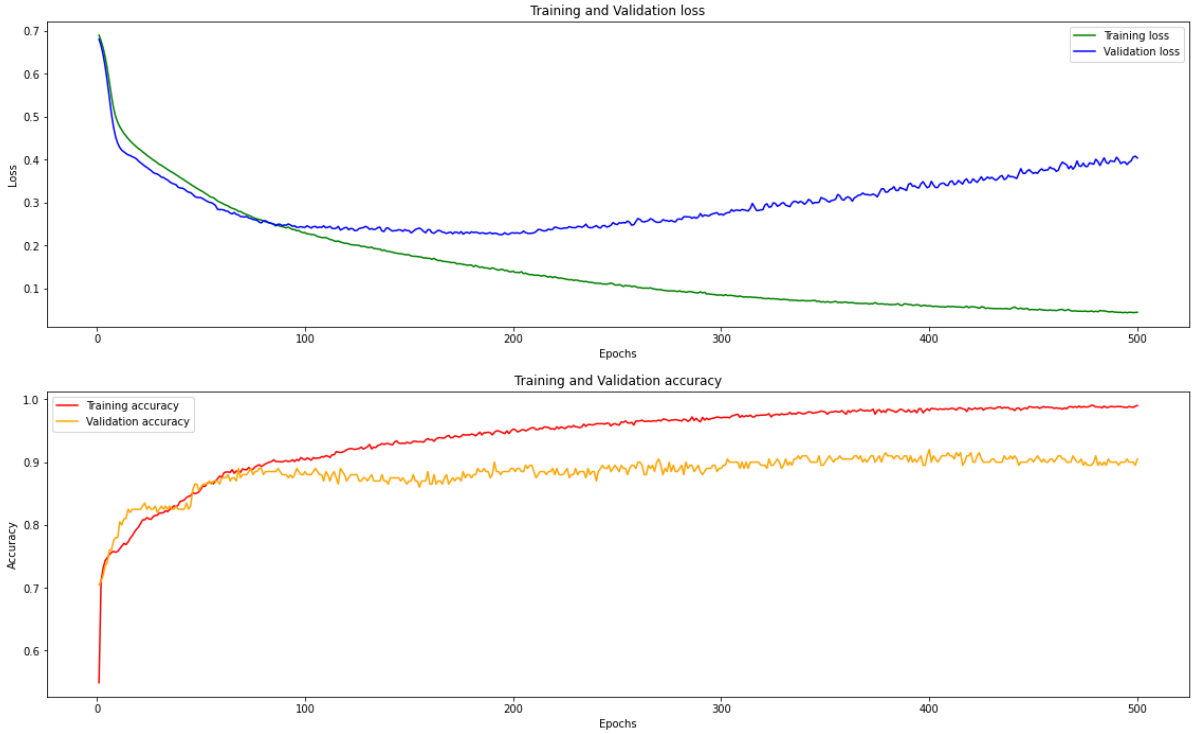
Нашиот модел беше креиран со Keras, Python библиотека за длабоко учење и претставува линеарен стек на слоеви кој се нарекува Sequential модел.

Невронската мрежа се состои од два fully-connected (целосно-поврзани) скриени слоеви со големина од 16 и 128 неврони. Овие слоеви се активираат со ReLU нелинеарната активациска функција. Последниот слој е излезниот слој составен од 1 неврон активиран со sigmoid функцијата (0 – позитивна класа, 1 – негативна класа).

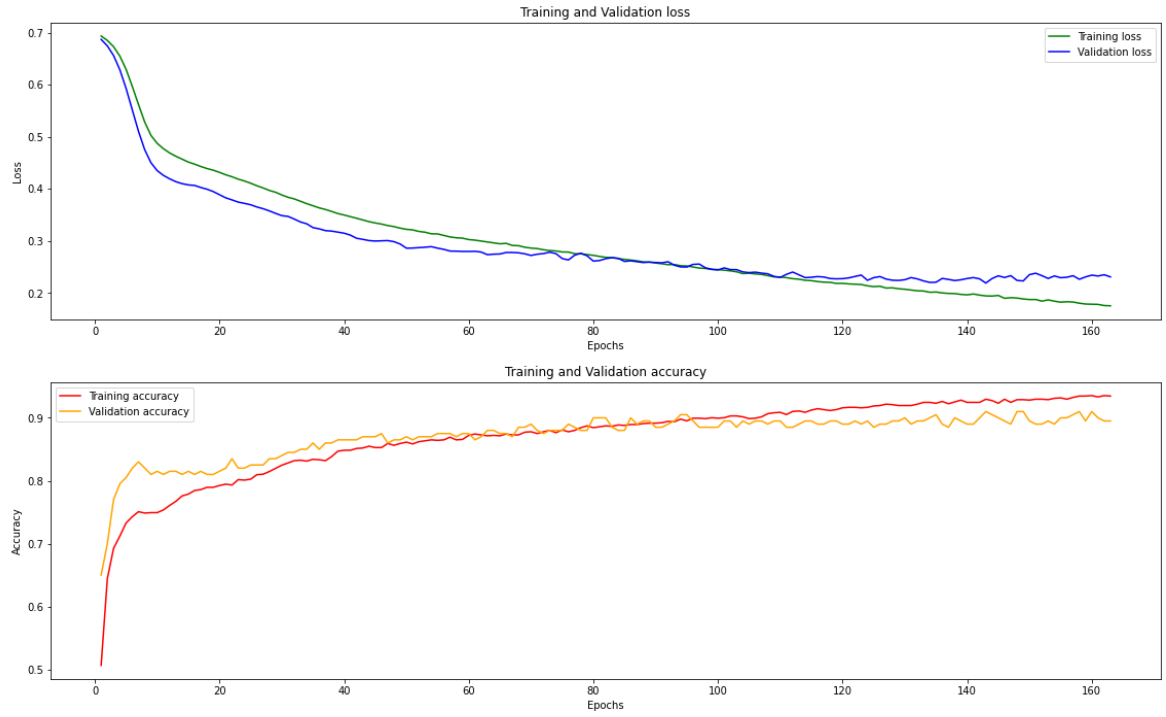
Откако беше дизајниран моделот, односно е креиран Model објект, беше потребно да се најдат соодветните параметри за иницијализација. Како функција на загуба беше користена binary_crossentropy затоа што се работи за проблем со бинарна класификација. Исто така беше користен оптимизаторот Adam со default-ните вредности за параметрите и ратата на учење, за минимизирање на вредноста на функцијата на загуба.

При тренирањето на моделот за валидација беа одделени 10% од секоја од класите. Притоа податоците од тренирачкото, валидациското и тестирачкото множество беа нормализирани со StandardScaler(). По повеќе обиди со пробување, како најсоодветни вредности за максималниот број на епохи (epochs) и batch_size беа одбрани 500 и 260 соодветно. За да се спречи потенцијален overfit за време на тренирањето беше користена Early Stopping методата за да се прекине тренирањето кога функцијата на загуба ќе го достигне својот минимум, при што параметарот patience (број на епохи пред да се прекине тренирањето откако ќе го достигне минимумот, односно откако ќе почне да расте) беше подесен на 5 за да се спречи underfitting.

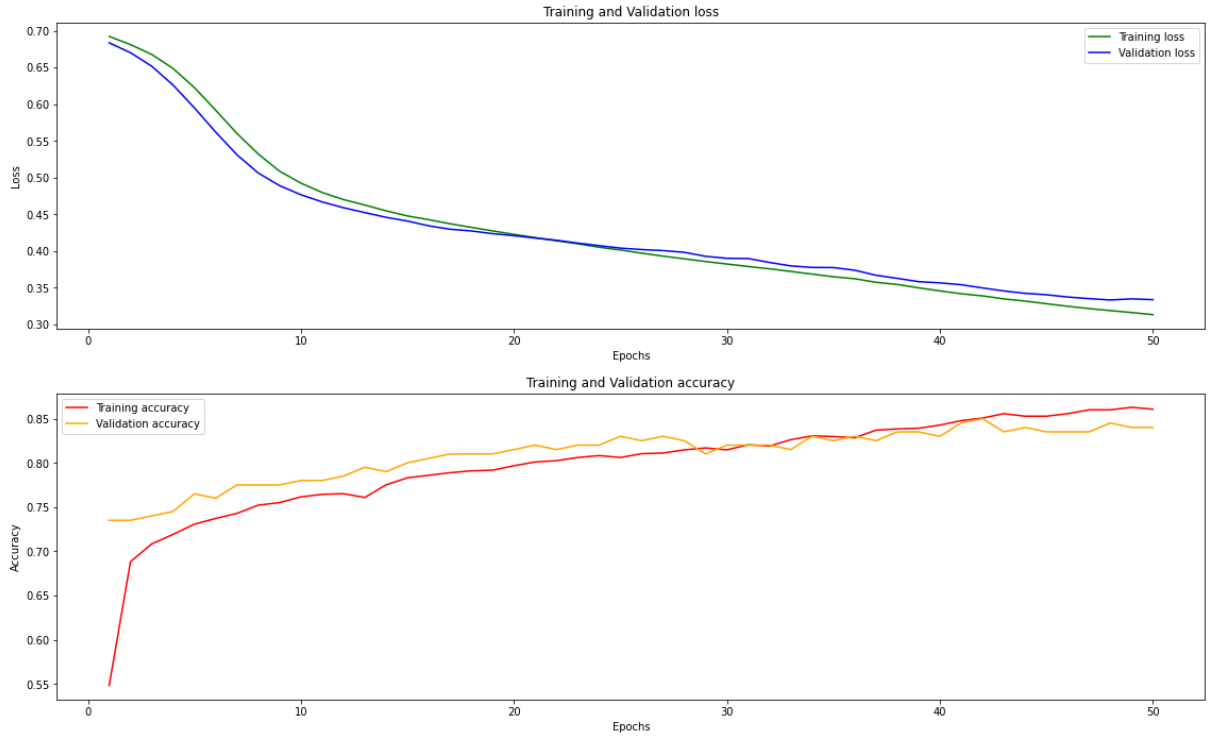
Со визуелизација на вредностите на функцијата на загуба при тренирање и валидација може да се заклучи дека моделот не прави overfit или underfit.



Визуелизација пред употреба на Early Stopping каде што може да се забележи дека се случува overfit.

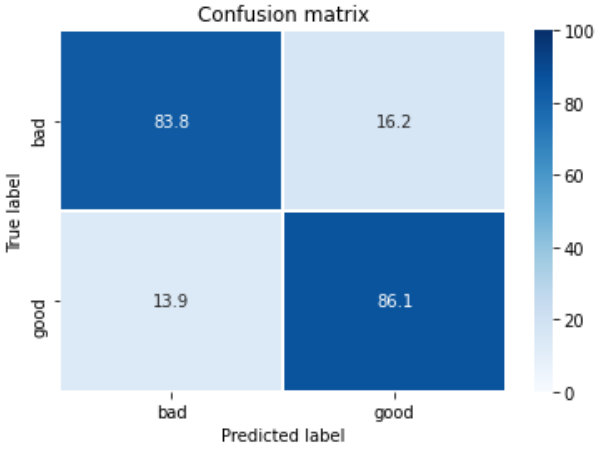


Визуелизација со употреба на Early Stopping со параметарот patience подесен на 20 каде што може да се забележи дека сеуште се случува overfit.



Визуелизација со употреба на Early Stopping со параметарот patience подесен на 5

Со овој модел добивме прилично задоволителни резултати точност од 85%, притоа може да се забележи дека моделот прилично добро ги предвидува и двете класи и тоа позитивната класа (true positive) - дури 86,1%, а негативната класа (true negative) – 83,8%.



Neural network score on test:

	precision	recall	f1-score
0	0.83	0.87	0.85
1	0.86	0.83	0.84
accuracy			0.85
macro avg	0.85	0.85	0.85
weighted avg	0.85	0.85	0.85

6.6. K nearest neighbors

Еден од наједноставните методи на машинско учење е методот на k-Најблиски Соседи. Методот може да се користи за класификација и за регресија. Доколку се користи за класификација, резултатот е класификација на објектот според резултатот од гласањето на неговите соседи, при што објектот се доделува на класата за која и припаѓаат мнозинство од гласовите. Иако спаѓа во алгоритмите од областа на машинското учење, сепак кај овој алгоритам процесот на учење не се извршува. Она што обично претставува фаза на тренирање кај паметните алгоритми, овде опфаќа само зачувување на векторите со влезни вредности и соодветните ознаки за класата на која и припаѓа примерокот (доколку алгоритмот се користи за класификација). Алгоритмот со k-Најблиски Соседи, за разлика од останатите паметни алгоритми, не прави претпоставка за распределеноста на податоците ниту пак донесува генерални заклучоци. За сметка на тоа мора да ги чува сите податоци кои инаку би се користеле за тренирање и кои после тренирањето кај најголем дел од паметните алгоритми повеќе не се потребни и не мора да се чуваат во меморија. Но, бидејќи кај овој метод нема фаза на тренирање, мора да ги има сите податоци, затоа што ќе му требаат за време на тестирањето (кога работи со реални податоци).

Класификацијата се прави така што се наоѓаат најблиските k соседи на влезниот објект и преку пресметување на просечна ознака за истите, ќе го добија резултатот. За да можеме да го направиме ова најпрвин треба да одговориме на две прашања:

1. Што ќе претставува растојанието? Односно според кои критериуми ќе одредуваме кои се најблиските соседи? Кои вредности ќе ги кориситме за да го пресметаме растојанието со цел да ги одредиме најблиските соседи?
2. Што ќе претставува просечната вредност? Одкако ќе ги издвоиме најблиските соседи, која е вредноста за која ќе бараме просек и како ќе го пресметуваме просекот? За да го одговориме првото прашање, најпрвин треба да знаеме што претставуваат објектите кои ги добиваме на влез.

Доколку станува збор за точки во картезијанов простор или други континуирани вредности можиме да користиме (и најчесто се користи) Евклидово растојание

доколку влезните вредности се од сличен тип (пример должина и ширина) или Менхетн растојание или попознато како такси геометрија доколку влезните податоци не се од сличен тип (висина, тежина, возраст итн.)

Вредноста k која цело време се спомнува е всушност бројот на најблиски соседи кои ќе помогнат во пресметување на резултатот за новиот објект. Бројот k секогаш ќе биде цел број поголем од 0. Вредноста на k е од особена важност кога алгоритмот се користи за класифицирање.

Доколку имаме две класи тогаш секогаш треба да користиме непарен број, бидејќи доколку бројот на соседи е непарен, тогаш една од класите секогаш ќе има барем еден објект повеќе од другата класа. Кога бројот на класи е поголем од 2, тогаш е потешко да се избере вредноста на k. Во најмала рака, бројот k се препорачува да биде за 1 поголем од бројот на класи.

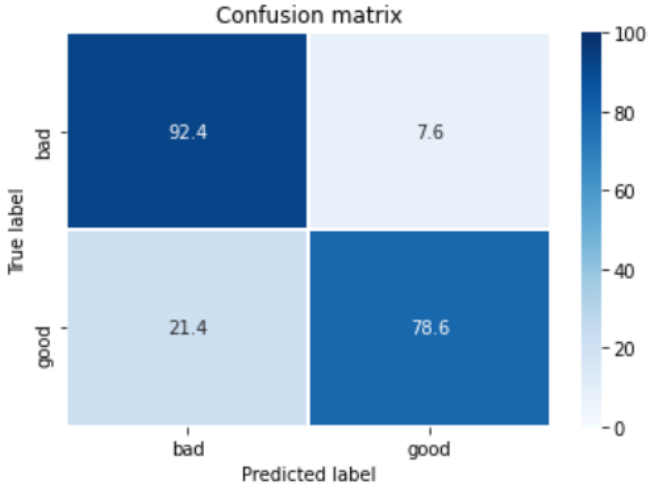
Методот со k-Најблиски Соседи има одредени слабости. Една од нив е што за да се одредат најблиските k соседи треба да се пребараат голем дел од податоците во базата, во најлош случај сите. Овој проблем може да се адресира на неколку начини, најчесто преку соодветен избор на алгоритам за пребарување. Друг проблем, е проблемот со димензионалноста. Овој метод функционира многу добро, доколку бројот на влезните променливи со кои е опишан еден објект е мал, но има големи потешкотии кога имаме поголем број на влезни променливи (најчесто повеќе од 10). Методот иако има одредени слабости, сепак се користи во голем број на полиња. Затоа и се одлучивме да го испробаме и овој начин за класификација на нашето податочно множество, а резултатие кои ги добивме и не беа толку лоши како оние на наивниот Баесов класификатор.

По повеќе проби со различни параметри, најдобар резултат добивме ако користиме 11 најблиски соседи и ако користиме Менхетен растојание - точност од 85%, а додека пак ако го користиме Евклидовото растојание добиваме за нијанса помала точност од 80%, сето тоа на скалирани податоци. Ако податоците не се скалирани моделот станува драстично полош и има околу 60% точност. На сликата е прикажана матрицата на конфузија добиена со скалирани податоци и Менхетен растојание, може да се забележи дека моделот многу добро ја

предвидува позитивната класа(true positive) - дури 91%, а малку полошо ја предвидува негативната класа(true negative) - 81%.

	precision	recall	f1-score	support
0	0.81	0.92	0.86	198
1	0.91	0.79	0.84	201
accuracy			0.85	399
macro avg	0.86	0.86	0.85	399
weighted avg	0.86	0.85	0.85	399

[]



6.7. SVM - Support Vector Machine

SVM претставува метод на машинско учење кој се користи за препознавање на облици, предвидување на вредности и естимација на функции. SVM врши класификација на тој начин што ја пронаоѓа онаа права, рамнина или хиперрамнина што ја максимизира маргината помеѓу две класи.

Алгоритамот на SVM работи на следниов начин

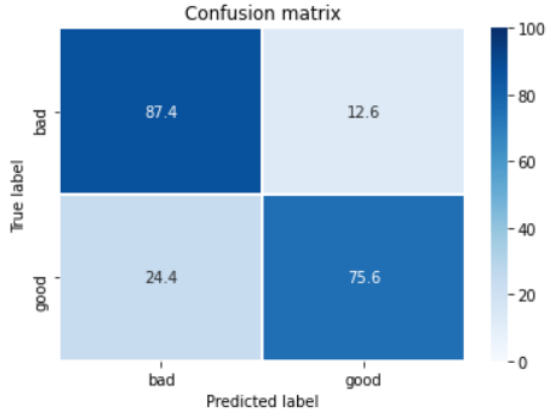
1. Дефинира оптимална хиперрамнина која ги разделува инстанциите во однос на класите на кои им припаѓаат, според нивните атрибути (својства). Оптималната хипер-рамнина се наоѓа на тој начин што се максимизира маргината помеѓу поддржувачките вектори.

2. Проширување на претходниот чекор со цел да се решат проблемите каде податоците не се линеарно сепарабилни – на тој начин што се даваат „пенали“, односно казни при погрешно класифицирање.
3. Доколку станува збор за податоци кои не се линеарно сепарабилни, можно е тие имплицитно да се мапираат во повеќедимензионален простор во кој би биле линеарно сепарабилни и полесно би се направила класификацијата.

Кога податоците не се линеарно сепарабилни, се користат специјални видови на функции наречени кернел функции и се прави пресликување од влезниот простор на особини во некој друг високо димензионален простор на особини каде што се зголемува моќноста на линеарните машини. Оптималната хиперрамнина во нелинеарен случај се добива со метод на пресметување скаларен прозивод во високодимензионалниот простор, нелинеарно поврзан со влезниот. Најчесто користени кернел функции се линеарни, полиномијална и Гаусова радијална базисна функција.

За нашиот проблем, пред да искористиме СВМ прво го нормализиравме нашето податочно множество. Станува збор за податоци кои не се линеарно сепарабилни, па како кернел функција се користи default- ната од библиотеката sklearn односно радијална базисна функција.

Добивме прилично задоволителни резултати со 87.4% точни предвидувања на bad класата и 75.6% на good класата.



	precision	recall	f1-score	support
0	0.78	0.87	0.82	198
1	0.86	0.76	0.80	201
accuracy			0.81	399
macro avg	0.82	0.81	0.81	399
weighted avg	0.82	0.81	0.81	399

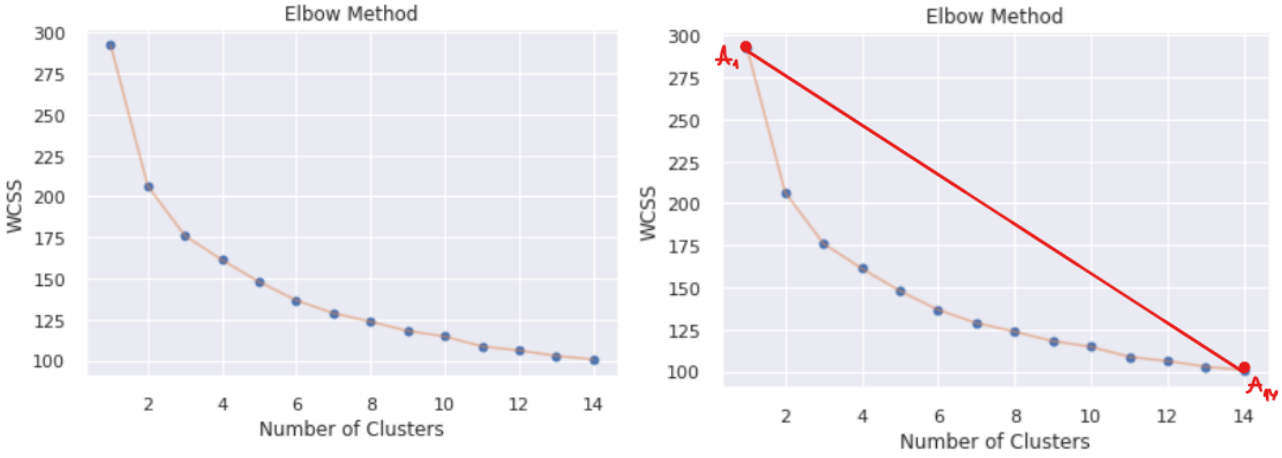
7. K-means алгоритам

K-means е едноставен „unsupervised“ алгоритам кој бара предефинирани K групи во дадено податочно множество. Со дефинирање на бројот K, дефинираме број на потребни центроиди во податочното множество - бројот на кластери на кои ќе се подели податочното множество. Центроиди се имагинарни или реални локации кои ги претставуваат центрите на кластерите. Кластерирањето се прави со минимизирање на квадрирано Евклидово растојание помеѓу податоците и центроидите.

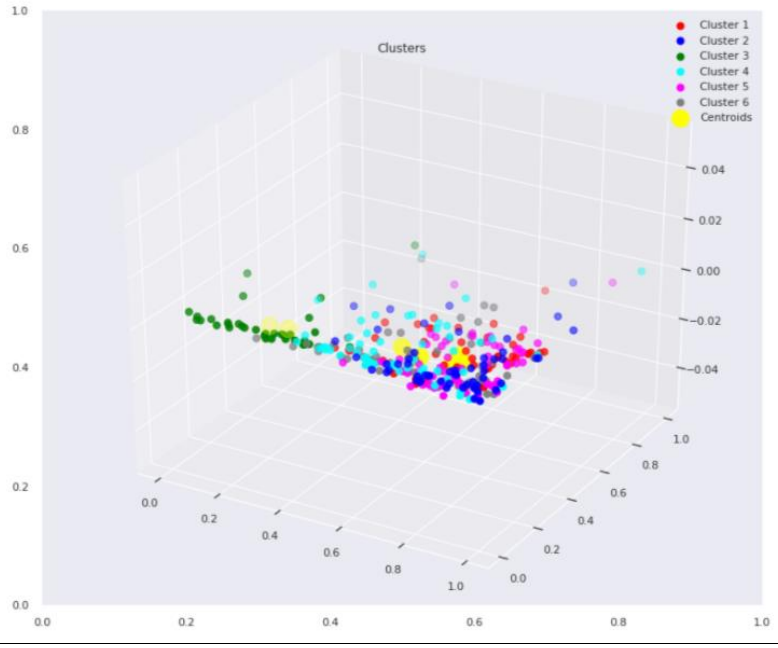
Пред да ги кластерираме податоците направен е MinMaxScaler – скалирање на катактеристиките со одреден дефиниран ранг на вредности со цел задоволителни резултати, бидејќи овој алгоритам е зависен од рангот во кој се движат податоците. Исто така, со цел поедноставување на приказот на крајниот резултат, земено е подмножество од околу 300 објекти од податочното множество.

Првиот проблем со кој се соочивме во овој случај е избирањето на бројот на кластери/центроиди/К. Постојат неколку методи со кои оптимално може да се одреди овој број меѓу кои и Elbow методот. Идејата е доста едноставна: алгоритмот се извршува повеќе пати се додека не се најде оптималниот број на кластери. Она што се случува најчесто е, како што го зголемуваме бројот на кластери разликите помеѓу кластерите стануваат се помали, а разликата помеѓу самите објекти во кластерите исто така се зголемува. Значи, треба да најдеме баланс, точка каде објектите во една група се похомогени колку што е возможно, а кластерите се најразлични еден од друг. Бидејќи k-means го пресметува растојанието помеѓу објектите и центроидот на кластерот на кој припаѓаат, идеално е ова растојание да е најмало можно. За секоја вредност на К, се извршува k-means алгоритмот и се пресметува сумата на квадрираните растојанија од објектите во податочното множество до најблискиот центроид. Како што К се зголемува, сумата на квадрираните растојанија(wcss) тежи кон 0. Ако К го одбереме да биде еднакво на бројот на објекти во податочното множество, тогаш

секој објект ќе креира свој кластер, што значи дека сумата на квадрираните растојанија ќе стане 0.



На *графикот* се прикажани сумите на квадрираните растојанија за различните броеви на кластери(2-15). Како да најдеме која е точката која го индицира балансот помеѓу поголема хомогеност во рамките на кластерот и поголема разлика помеѓу кластерите? Тоа е точката од кривата која се наоѓа на најголемо растојание од правата која минува низ точката А1 и точката А14. Со тоа, доаѓаме до заклучок дека оптималниот број на кластери е 7. *На сликата со посебни бои се прикажани кластерите и центроидите.*



Откога го најдовме оптималниот број на кластери, можеме да ги кластерираме податоците. Крајниот резултат е прикажан на сликата, со тоа што секој кластер е со посебна боја, а воедно се прикажани и центроидите.
На овој начин, се издвојуваат 7 кластери на песни со слични карактеристики. Како пример за сличноста на карактеристики може да послужи пресметката на просечната популарност на песните во секој кластер, од што можеме да заклучиме дека најпопуларните песни припаднале во 6-тиот кластер, а најнепопуларните во 4-тиот кластер.

8. Анализа и споредба на модели

8.1 Acccuracy

Acccuracy или точност е една од наједноставните методи за евалуација на даден модел. Се дефинира како сооднос од број на точни предвидувања и вкупниот број на сите предвидувања. Понекогаш точноста не е најсоодветна мерка за проценка на даден модел, особено кога станува збор за неизбалансирани класи.

Во нашиот случај имавме избалансирано податочно множество, па точноста ни даде добра слика за тоа колку се добри нашите модели. Вредностите се движат во интервалот од 0.71 кај Naive Bayes до високи 0.97 за XGBoost.

8.2 Precision и Recall

Како дополнителни и често користени метрики ги разгледавме и Recall и Precision. Веќе ги разгледавме матриците на конфузија добиени за секој модели и можевме да ги видиме за секој од моделите вредностите за TP (true positive), TN (true negative), FP (false positive) и FN (false negative).
Precision е бројот на точно класификувани позитивни примери поделени со вкупниот број на примери кои биле позитивно класификувани. Ваквата метрика ни дава слика за тоа колку се прецизни и корисни резултатите.
Recall е бројот на точно класификувани позитивни примери поделен со вкупниот број на позитивните примероци од тест-множеството и ни дава некава слика за тоа колку ни е комплетен моделот.

На следната слика се прикажани вредностите за Accuracy и Recall за секој од моделите и забележуваме дека генерално имаат одлични вредности.

	Model	Acc	Recall	ROC	TP	FN
0	XGBoost	0.974937	0.970149	0.974974	195	6
1	Tuned Random Forest	0.964912	0.950249	0.965023	191	10
2	Decision Tree	0.929825	0.925373	0.929858	186	15
3	Random Forest	0.924812	0.895522	0.925034	180	21
4	Bagging Classifier	0.877193	0.80597	0.877733	162	39
5	Tuned KNN	0.854637	0.78607	0.855156	158	43
6	ANN	0.849624	0.860697	0.84954	173	28
7	SVM	0.829574	0.800995	0.82979	161	40
8	KNN	0.824561	0.771144	0.824966	155	46

Досега наведените метрики ја оценуваат точноста на моделот во однос на позитивната класа т.е во нашиот случај песните кои што му се допаѓаат на корисникот.

Секако, не помалку важна во нашиот случај е и негативната класа т.е песните кои не му се допаѓаат на корисникот. Затоа е корисно да разгледаме мерка која во предвид ќе ја земе и негативната класа и со тоа ќе добиеме подобра слика за квалитетот на нашите модели.

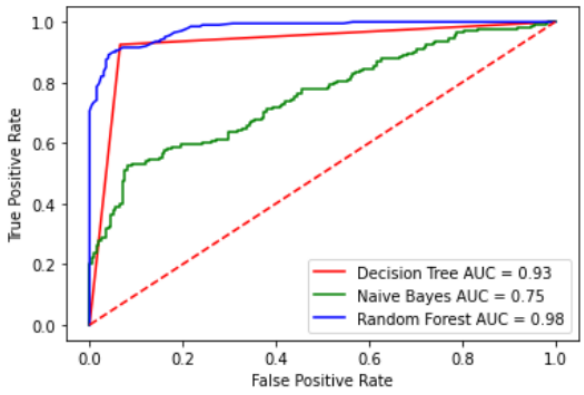
8.3. ROC (Receiver Operating Characteristic) - крива

ROC-кривата претставува графички приказ на успешноста на моделот за класификација, а може да се користи и за споредба помеѓу два или повеќе класификатори. На x-оската, таа ја прикува лажната позитивна стапка (false positive rate), т.е. односот меѓу негативните случаи кои погрешно се препознаени како позитивни и сите негативни случаи, додека на y-оската ја прикажува

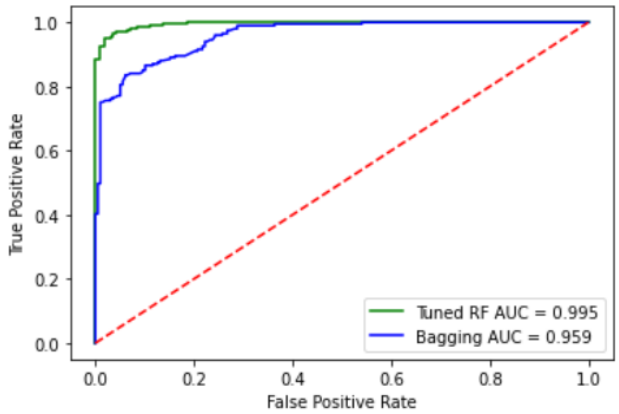
чувствителноста, т.е. делот од позитивните случаи кои се точно идентификувани како позитивни. Притоа, на графикот на ROC-кривата, најчесто се претставува и дијагонална линија која претставува случајно изгенериран модел, како најлошото можно сценарио, т.е. како најлош можен класификатор, додека пак кај совршениот класификатор ROC-кривата ќе го допре горниот лев агол од графикот.

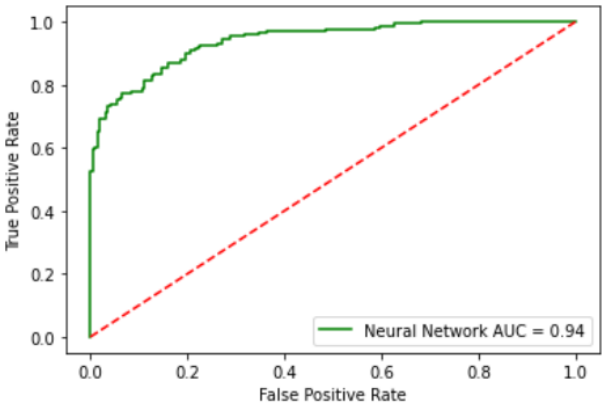
Со цел да се квантифицира успешноста на даден класификатор, во литературата се користи „површината под ROC-кривата“ (Area Under ROC Curve, Area Under the Curve, AUC) како мерка за квалитетот на моделот за класификација (класификаторот).

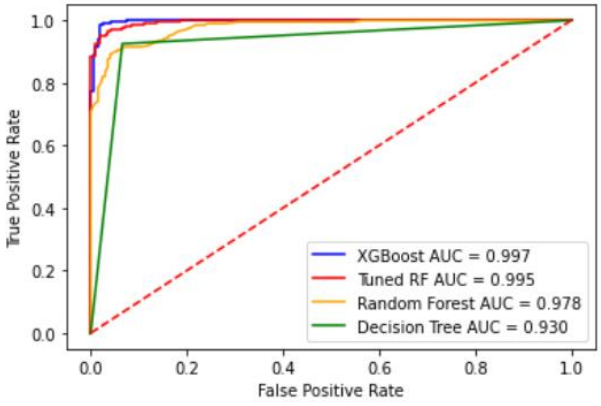
На сликата се прикажани ROC кривите на наивниот Баесов класификатор, дрвото на одлука и random forest класификаторот, заедно со вредностите за површината под кривите. Како што и претходно е нагласено, од тука може да се забележи дека дрвото на одлука и random forest класификаторот имаат доста задоволителни резултати, со тоа што нивната AUC вредност е блиску до 1, додека пак наивниот Баесов класификатор е има AUC=0.75 со што може да се заклучи дека солидно ги класифицира податоците.

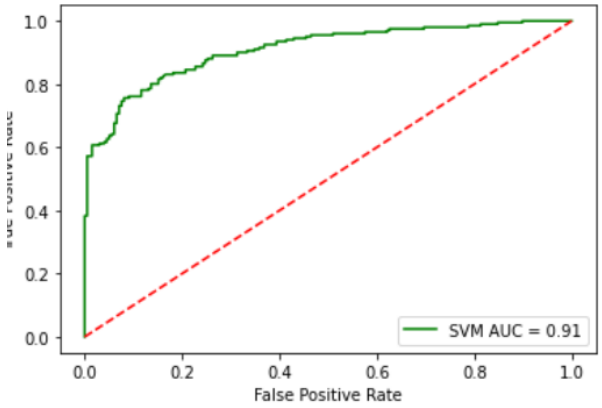


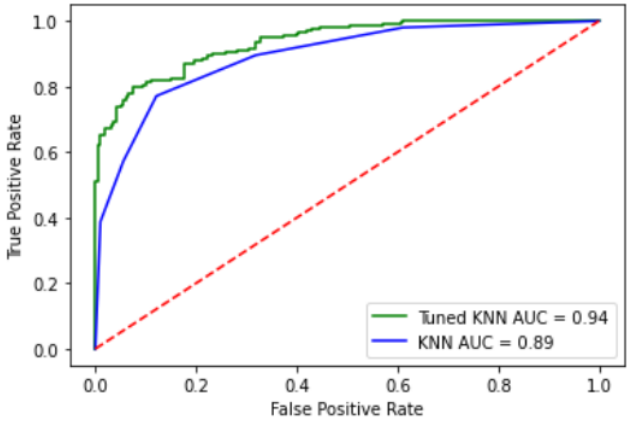
На *сликите* што следуваат се прикажани ROC кривите, заедно со AUC вредностите за повеќе од класификаторите кои ги имплеметиравме. Може да се забележи дека AUC вредностите варираат од 0.91(невронската мрежа) до 0.997(XGBoost класификаторот), што значи дека сме добиле одлични класификатори, кои предвидуваат за дадена песна дали ќе му се допаѓа на корисникот или не.


True Positive Rate
False Positive Rate
Tuned RF AUC = 0.995
Bagging AUC = 0.959


True Positive Rate
False Positive Rate
Neural Network AUC = 0.94


True Positive Rate
False Positive Rate
XGBoost AUC = 0.997
Tuned RF AUC = 0.995
Random Forest AUC = 0.978
Decision Tree AUC = 0.930


False Positive Rate
SVM AUC = 0.91


True Positive Rate
False Positive Rate
Tuned KNN AUC = 0.94
KNN AUC = 0.89

9. Заклучок

Музиката е наша секојдневност и се смета за уметност, но многу поважно е што ја прави да биде уметност. Дали тоа е жанрот, енергијата, акустичноста, инструменталноста...?

Со помош на овие карактеристики се обидовме да изградиме модели кои ги трениравме со податоци од две плејлисти, една со песни кои ни се допаѓаат нам како Spotify корисници и една со песни кои не ни се допаѓаат. Во тест фазата видовме колку успешно нашите модели прават разлика меѓу двете класи на песни и со помош на соодветните метрики видовме дека изградивме одлични модели.

Се обидовме да искористиме што повеќе модели и да ги подесеме нивните параметри за да добиеме што подобри резултати. На крај можеме да заклучиме дека добивме задоволителни резултати, особено ако се земе предвид дека вкусот за музика е многу лична работа и може да биде фрустрирачки тешко да се опише зошто сакаме или не сакаме одредена песна.

10. Користена литература

1. Spotify.com. (2020). *Get Audio Features for a Track | Spotify for Developers*. [online] Available at: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/ [Accessed 19 Oct. 2020].
2. Dinesh Yadav (2019). *Categorical encoding using Label-Encoding and One-Hot-Encoder*. [online] Medium. Available at: https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd [Accessed 19 Oct. 2020].
3. Machine Learning Mastery. (2020). *Linear Discriminant Analysis for Dimensionality Reduction in Python*. [online] Available at: https://machinelearningmastery.com/linear-discriminant-analysis-for-dimensionality-reduction-in-python/ [Accessed 19 Oct. 2020].
4. Meigarom Lopes (2017). *Is LDA a dimensionality reduction technique or a classifier algorithm?* [online] Medium. Available at: https://towardsdatascience.com/is-lda-a-dimensionality-reduction-technique-or-a-classifier-algorithm-eeed4de9953a [Accessed 19 Oct. 2020].
5. Bmj.com. (2019). *11. Correlation and regression | The BMJ*. [online] Available at: https://www.bmj.com/about-bmj/resources-readers/publications/statistics-square-one/11-correlation-and-regression [Accessed 19 Oct. 2020].
6. https://www.facebook.com/displayr (2018). *What is a Correlation Matrix? | Displayr*. [online] Displayr. Available at: https://www.displayr.com/what-is-a-correlation-matrix/ [Accessed 19 Oct. 2020].
7. Built In. (2019b). *A Step by Step Explanation of Principal Component Analysis*. [online] Available at: https://builtin.com/data-science/step-step-explanation-principal-component-analysis [Accessed 19 Oct. 2020].
8. Scikit-learn.org. (2020). *6.3. Preprocessing data — scikit-learn 0.23.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/preprocessing.html [Accessed 19 Oct. 2020].
9. Rohith Gandhi (2018a). *Naive Bayes Classifier - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c [Accessed 19 Oct. 2020].
10. Prashant Gupta (2017). *Decision Trees in Machine Learning - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 [Accessed 19 Oct. 2020].
11. Yiu, T. (2019). *Understanding Random Forest - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 [Accessed 19 Oct. 2020].
12. Built In. (2019a). *A complete guide to the random forest algorithm*. [online] Available at: https://builtin.com/data-science/random-forest-algorithm?fbclid=IwAR3n4148TbcQRnnGVo-tm3adNFz7zAUm_6mdHI6d_Q0BHh0o0q5pkZpL8Qg [Accessed 19 Oct. 2020].

13. Readthedocs.io. (2020). *XGBoost Parameters — xgboost 1.3.0-SNAPSHOT documentation*. [online] Available at: https://xgboost.readthedocs.io/en/latest//parameter.html?fbclid=IwAR2o1iRoAgXZyhrEm_gFQLSt5XB6sYJa2rNnfxu24BVG178DgB6uVViLhh8 [Accessed 19 Oct. 2020].
14. Machine Learning Mastery. (2016a). *A Gentle Introduction to XGBoost for Applied Machine Learning*. [online] Available at: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ [Accessed 19 Oct. 2020].
15. Brilliant.org. (2013). *Artificial Neural Network | Brilliant Math & Science Wiki*. [online] Available at: https://brilliant.org/wiki/artificial-neural-network/?fbclid=IwAR2o1iRoAgXZyhrEm_gFQLSt5XB6sYJa2rNnfxu24BVG178DgB6uVViLhh8 [Accessed 19 Oct. 2020].
16. Deep, A. (2016). *InnoArchiTech*. [online] InnoArchiTech. Available at: https://www.innoarchitech.com/blog/artificial-intelligence-deep-learning-neural-networks-explained?fbclid=IwAR143sH84RZIR9NfXhvCl6wv-l5CQs-TKS8H2fXLBvSQLp8NWlh_U8-neao [Accessed 19 Oct. 2020].
17. Analytics Vidhya (2018). *K Nearest Neighbor | KNN Algorithm | KNN in Python & R*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/ [Accessed 19 Oct. 2020].
18. Machine Learning Mastery. (2016b). *K-Nearest Neighbors for Machine Learning*. [online] Available at: https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ [Accessed 19 Oct. 2020].
19. Jessica Temporal. (2019). *How to define the optimal number of clusters for KMeans*. [online] Available at: https://jtemporal.com/kmeans-and-elbow-method/ [Accessed 19 Oct. 2020].
20. Tola Alade (2018). *Tutorial: How to determine the optimal number of clusters for k-means clustering*. [online] Medium. Available at: https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f [Accessed 19 Oct. 2020].
21. Rohith Gandhi (2018b). *Support Vector Machine — Introduction to Machine Learning Algorithms*. [online] Medium. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Accessed 19 Oct. 2020].
22. Markham, K. (2014). *Simple guide to confusion matrix terminology*. [online] Data School. Available at: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/#:~:text=A%20confusion%20matrix%20is%20a,related%20terminology%20can%20be%20confusing. [Accessed 19 Oct. 2020].
23. Završni, R. and 5927 (n.d.). *SVEUČILIŠTE U ZAGREBU FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA*. [online] Available at: https://bib.irb.hr/datoteka/1009326.Dominik_Hrastic_Zavrsni_final.pdf [Accessed 19 Oct. 2020].
24. Wikipedia Contributors (2020). *Receiver operating characteristic*. [online] Wikipedia. Available at:

https://en.wikipedia.org/wiki/Receiver_operating_characteristic#:~:text=A%20receiver%20operating%20characteristic%20curve,why%20it%20is%20so%20named. [Accessed 19 Oct. 2020].

25. Koo Ping Shung (2018). *Accuracy, Precision, Recall or F1? - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 [Accessed 19 Oct. 2020].