Projecte MIDA

Detecció de càncer de mama segons les característiques del tumor

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

Arnau Roca

Roberto Lupu

Desembre de 2023

## 0.- Dataset

Per començar, descriurem el dataset seleccionat i el motiu. Vam començar amb el procés de cerca per trobar algun dataset que ens sigui interessant, seguint datasets relacionats amb els nostres hobbies o aficions, com podrien ser la música, el futbol o els cotxes. Arran d’aquests datasets ens vam adonar que no trobariem cap que ens serveixi per a aquest projecte.

Finalment ens vam decantar pel *Breast Cancer Wisconsin (Diagnostic) Data Set,* accessible des de <https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data>.

Aquest dat set, tal i com el propi nom indica és un recull d’informació de diferents mostres de massa de mama. Per ser més concrets, les dades estan recollides d’imatges digitalitzades d’aquestes tumors. A partir de les quals s’han pogut extreure les següents característiques:

* Diagnosis: Descriu el tipus de tumor segons si es Benigne (B) o Maligne (M).
* Radius: Mitjana de radis entre el centre i diferents punts del perímetre.
* Textura: Mesurada per la desviació estàndard de valors en escala de grisos.
* Perímetre: Longitut que envolta el tumor.
* Àrea: Superficie que ocupa el tumor.
* Suavitat: Variabilitat en les longituds dels radis. Quant més s’assembli el tumor a un cercle més alta és la suavitat.
* Compacitat: Es calcula a través del perímetre al quadrat dividit entre l’àrea menys 1. Quant més alta sigui més irregular és la forma del tumor.
* Concavitat: Aquest camp mesura el grau de les corbes còncaves del tumor. Quant més alta sigui, més tancades són les corbes.
* Punts còncaus: Quantitat de parts còncaves del contorn del tumor.
* Simetria: Mesura el grau de simetria o balanç en la fora del tumor.
* Dimensió fractal: La dimensió fractal mesura la complexitat del contorn del tumor. Quant més irregular sigui i més complex sigui el contorn la dimensió fractal tendirà a ser major. En aquests contextos és una bona pràctica restar-li una unitat al valor calculat (la resta ja està feta per defecte en el dataset).

Per cadascun d’aquestes propietats en la imatge d’un tumor, podem observar que tenim la mitjana d’aquest valors, l’error estàndard i la mitjana dels 3 valors més alts.

Tots els valors estan arrodonits a quatre valors significatius. No fica enlloc les unitats, però a jutjar pels valors i per normalment es mesuren els tumors, tot apunta a que son milímetres.

## 

## 

## 1.- Preprocessing

Pel que fa al pre-processament de dades comencem per importar les dades del dataset. Per assegurar-nos que tot s’ha realitzat correctament, visualitzem les primeres files del dataset acompanyades d’unes petites estadístiques relacionades.

A continuació seguim amb els següents passos de pre-processament de dades.

### Valors nuls

Fent una cerca de valors nuls al dataset descobrim que aquest té d’entrada 569 valors nulls per l’última columna, la d’Unamed. Tenint en compte que aquesta columna no conté cap informació, i que ni tan sols està descrita a la documentació del dataset, ens decantem per esborrar la columna sencera.

### Numerització de dades no numèriques

Els següent pas que hem seguit ha estat escanejar en el dataset quines són les columnes que contenen valors no numèrics.

Ens trobem que l’únic camp que compleix aquesta condició és la “variable objectiu” o “variable a predir”, és a dir, la variable diagnosis.

Analitzant més detalladament els valors de la columna, veiem que els únics valors que aquesta conté són “M” i “B”. Segons la documentació, els tumors catalogats amb diagnosis “M” fan referència a tumors Malignes, mentre que els altres a tumors benignes.

El criteri de mapeig que decidim aplicar és assignar un 1 als tumors malignes i un 0 al benignes.

### Normalització de dades

La normalització de dades és un procés que pot afectar tant positiva com negativament el l’eficàcia d’un model. Tenint això en compte, per a la part del pre-processament guardarem un dataset amb les dades sense normalitzar i un altre amb les dades normalitzades. El primer d’ells guardat al fitxer anomenat dataset\_preprocessat.csv mentre que l’altre es guarda al fitxer anomenat dataset\_preprocessat\_norm.csv.

Al script Preprocessing imprimim per pantalla abans i després de la normalització els següents valors: la mitjana, la desviació estàndard, i els valors mínim i màxim.

Si comparem els resultats observem que totes les mitjanes passen de ser valors variats a ser valors molt propers a 0 (els considerem 0 pràcticament, ja que el que més s'allunya del 0 s'allunya per un nombre elevat a -17, insignificatiu en comparació amb la desviació estàndard, molt propera a 1).

De la mateixa manera, observem com els valors de les desviacions estàndard, són valors molt propers a 1. Tot i que poden no ser 1 exactament, són valors molt propers.

### Columnes irrellevants

Per últim, estudiarem més detalladament cada columna per separat. Tenint en compte els diagnòstic de cada tumor separem les dades en els dos possibles grups. Així podem observar la diferència de la distribució de dades per cada columna de cara a establir, a simple vista, la importància cada columna vers el diagnòstic.

La primera manera és creant, per cada columna, una gràfica amb les freqüències de cada valor. Com a comentari general, podem dir que la majoria de gràfiques segueixen una distribució normal.

### Eliminació d’outliers

Aquest apartat no està del tot finalitzat. La idea era esbrinar la distribució de cada columna per aplicar segons el tipus de distribució una forma diferent de trobar outliers. Malauradament, per falta d’organització de temps només hem aconseguit fer la funció normal.

Tot i així per les columnes que segueixen distribuició normal hem observat que la quantitat d’outliers no és molt elevada, ens sortien una mitja de 7 outliers per columna.

La idea que tenim no és eliminar tots les outliers trobats en totes les columnes, sinó simplement relacionar les entrades que eren outliers de diferents columnes i eliminar aquelles que estiguessin implicades en més ocasions.

## 2.- Evaluació dels criteris dels models de mineria de dades

De cara a l'avaluació utilitzem diversos criteris, segons el tipus de model utilitzat. Tot i així, hi ha un interès comú i prioritari.

Està clar que tots els models poden tenir els seus errors en la classificació, però hi ha una diferència entre ells si ens fixem de quina manera s’equivoquen. Pot ser que donats dos models, els dos acertin un 95% de les seves prediccions, no obstant, és possible que un sigui molt millor que l’altre en els nostres ulls.

Donem el cas que el primer model s’equivoca en el 5% dels casos diagnosticant amb càncer unes persones que no el tenen. L’altre model s’equivoca al revés, diagnostica com a sans uns pacients amb càncer.

El percentatge d’error és el mateix, però les repercussions de les prediccions dels models són molt diferents. És per això que insistim en entrenar models que diagnostiquen més aviat com a càncer aquells tumors en els que es dubti.

No obstant, s’ha de ser coherents, el model no es pot centrar només en pacients amb càncer. Si entrenem un model basat únicament en aquestes premises, i perdent de vista les altres, obtindriem un model molt poc útil. El model seria massa agressiu identificant massa pacients amb càncer. Un doctor que diagnostica a tots els seus pacients amb càncer és un doctor que mai s’equivocarà amb algú que ho tingui, però no és per a res un bon doctor.

Així doncs, entrenar un model basat en el recall dels tumors malignes no és gaire eficient. Per altra banda, entrenar un model basat en accuracies perdria de vista el problema esmentat anetriorment. És per això que proposem com a solució entrenar models basats en l’f1-score en els tumors malignes.

Aquest f1-score és simplement una puntuació calculada a partir dels dos tipus de puntuació mencionats. Per ser més precisos, aquesta és la fórmula que els relaciona:

## 3.- Execució de diferents mètodes de ML

### Naive Bayes

Abans d’entrenar el model de Naive Bayes, hem volgut calcular la puntuació del k-fold cross validation dels dos conjunts de dades que tenim (normalitzat i no normalitzat). Per fer-ho hem utilitzat k = 10 i hem emprat com a estimador la Gaussian Naive Bayes.

Després d’obtenir la puntuació del cross validation, la confusion matrix i accuracy score observem que sembla ser que les dades no normalitzades beneficien lleugerament el model.

Tot i així, la diferència no és gaire gran. De fet, si printem les puntuacions de cada corss validation dels 10 que es fan, tot i que en la mitjana guanyi un, l’altre obté millor puntuació en algun cas. A més, els resultats de la funció *classification\_report* mostren que amb les dades normalitzades, el model té millor precisió per detectar tumors benignes i més sensibilitat als tumors malignes.

El f1-score resulta ser el mateix per ambdós casos, tot i així, per a l’entrenament del model Naive Bayes ens decantem pel conjunt de dades no normalitzat.

A continuació, separem les dades deixant un 70% per a l’entrenament i un 30 per al testeig.

Tot seguit procedim a fer diversos càlculs per obtindre el llindar òptim. Hem calculat tres llindars, el primer d’ells es centra en escollir un llindar que aporta millor recall per als casos de tumor benigne. El segon escull el llindar que aporta major accuracy. I en tercer lloc, càlculem el llindar òptim per al f1-score.

Els resultats de les dades separades amb el primer llindar calculat son molt poc satisfactòries. Sí que es veritat que aconseguim el màxim recall, que es el que busquem, però alhora massa tumors benignes són classificats com a malignes. El cost d’obtindre màxim recall per als tumors malignes és massa gran, i tampoc ens interessa.

Aplicant aquest model un entorn real, és com si arribessis al metge tenint mal de panxa i et diu que tens càncer. Aquesta manera de diagnosticar trobarà tots els verdaders positius, però també molts falsos positius. Com podem veure en en els resultats, només un 40% d'aquesta gent que li han dit que era positiu realment ho és.

Per als altres dos llindars, veiem que els resultats son pràcticament similars, cosa que a priori sembla extrany. Per aquest motiu és que vam probar amb llindars aleatoris d’entre 40% i 60%, i els resultats seguien sent iguals. Jugant amb aquest llindar ens hem adonat que aquest és molt permissiu per a un determinat rang.

Fixant-nos millor en el vector de probabilitats que el nostre model proporciona hem descobert el perquè. I és que les dades en aquest vector estan molt allunyades. Gairebé per a la majoria de dades el model no dubta. En la majoria de casos té bastant clar si un tumor és benigne o maligne. Fins i tot impressiona que amb un llindar de 0.1% els resultats siguin tan bons! Això vol dir que cada cop que diu que diu que algun tumor és benigne és perquè està com a mínim 99,9% segur. Que obviament, també hi ha algún cas on s’equivoca, però el criteri que té a l’hora de decidir (amb un llindar tan baix) és el que defensem, més val equivocar-se en un fals positiu que en un fals negatiu.

### 

### KNN

Pel que fa al KNN, hem fet el cross validation de la mateixa manera que anteriorment, a excepció que ara usem l’instància KNeighboursClassifier en comptes de la Naive Bayes.

L’altra diferència amb l’anterior apartat està en el resultat. I és que en aquest cas les dades normalitzades desemparen una millor puntuació, i aquest cop la diferència ja comença a ser considerable.

Les dades a entrar en el KNN tenen un altre marge de millora. Amb la funció *SelectKBest* ens podem estalviar una quantitat important de dades, i inclús millorar l’eficàcia del model. Per exemple, repetint el cross validation utilitzant només les primeres 20 columnes de dades millorem la puntuació insignificativament (no és fins la 16a xifra significativa que s’observa la diferència).

De fet, on millor es veu l’eficàcia és amb les gràfiques. Arribats a aquest punt en tenim dues. La primera està entrenada amb les k millors columnes enfocades a precisió, mentre que en la segona ens centrem en la sensibilitat (recall).

No obstant, les dues gràfiques ens porten a la mateixa conclusió, les Kbest columnes amb les que decidim treballar són les 19 primeres. Més concretament, i ordenades per preferència de SelectKBest:

1. perimeter\_worst
2. area\_worst
3. radius\_worst
4. points\_mean
5. points\_worst
6. perimeter\_mean
7. concavity\_mean
8. radius\_mean
9. area\_mean
10. area\_se
11. concavity\_worst
12. perimeter\_se
13. radius\_se
14. compactness\_worst
15. compactness\_mean
16. points\_se
17. texture\_worst
18. concavity\_se
19. texture\_mean

Aquesta informació és molt útil per ser comparada amb les gràfiques i boxplots obtinguts al preprocessing. En l’apartat on segons el nostre parer definiem la importància d’algunes columnes sobre altres. Clarament es pot veure que les columnes seleccionades amb més antelació són les columnes on hi ha major diferència entre tumors benignes i malignes.

Un cop analitzades les gràfiques decidim que només ens quedem amb les 19 millors columnes. Només resta cercar el millor nombre de veïns i el millor tipus de pes (distance o uniform) Igual que anteriorment, cerquem els millors valors per a l’accuracy, el recall y l’F1-Score.

Els resultats coincideixen bastant en alguns aspectes. Tots coincideixen en que el millor pes ha de ser l’uniform. El dilema vé al parlar de nombre de veïns, ja que tots coincideixen en que ha de ser 5, excepte la cerca per accuracy, que opta per 13 veïns.

Així doncs ens decantem per provar entrenar dos models, un amb 5 i l’altre amb 13 veïns.

D’aquests resultats, el que més ens impacta és que amb els paràmetres proposats per aconseguir un f1-score òptim s’obté un f1-score menys satisfactori que amb els paràmetres per obtindre un accuracy òptim.

### Decision Trees

Avaluarem el model, com en els casos anteriors amb el conjunt normalitzat i amb el no normalitzat. Ens assegurem que els folds estiguin stratificats per tal de tindre una proporció realista del conjunt i realitzem una **cross validation score**, amb el **DecisionTreeClassifier()** per avaluar el rendiment del model en diferents subconjunts de dades. D’aquesta manera obtenim una estimació més robusta del seu rendiment general.

D’aquest primer apartat podem veure que el conjunt NO normalitzat ens dona un resultat lleugerament millor: 0,91224 respecte 0,9068. Per acabar de verificar que tenim millors resultats quan no està normalitzat, farem la confusion matrix.

En la confusion matrix del no normalitzat obtenim també un resultat lleugerament millor. Encara que hi ha una diferència més baixa, podem observar que és millor, ja que tenim menys número de falsos negatius, una millor accuracy i f1 score.

Podem dir que és coherent, ja que els decision trees no es veuen beneficiats per la normalització. Això és perquè els decision trees divideixen les dades en funció dels valors de les característiques, i si normalitzes les característiques, els valors canviaran, però les relacions relatives entre ells es quedaran iguals. També ens quedem en el no normalitzat ja que a l’hora de veure a partir de quin numero anirà cap un costat o un altre de l'arbre, en el cas del normalitzat sortirien valors molt petits i en el cas de la sense normalització surten valors més normals.

Vist això, ara agafem la mateixa separació de test i training que en els casos anteriors, un 30% de les dades a test i les altres al train.

Crearem un objecte DecisionTreeClassifier, amb l’ús del criteri “Entropy” i després ho farem per “Gini”. Com podem veure, ens surt un millor resultat amb Gini i per tant escollim aquest. Cal destacar que estem obtenint resultats molt bons, cosa que ens indica que aquest és un bon model pel nostre dataset.

Un cop tinguem el objecte, entrenarem el model amb les dades d’entrenament i farem proves amb les dades de test. D’aquí obtindrem el score, la confusion matrix, un classification report i el interval de confiança per f1.

D’aquests resultats podem extreure moltes conclusions, ja que com podem veure te un molt bon score, un 0.9239, i en la confusion matrix podem observar que treballa molt bé, ja que el cas crític que són els falsos negatius, els minimitza fins a 4. Pel que fa al report podem veure que tenim molt bona precisió en general (0.92 per 0 i 0.93 per 1) cosa que ens indica que treballa molt bé per aquest training set. Seguim amb el recall, on veiem que pel 0 és molt bo, però pel 1 és més baix. Realment nosaltres busquem maximitzar el recall de 1, per tal de trobar els màxims malignes possibles, però tampoc és un resultat alarmant, ja que 0,86 no és un mal resultat tampoc. Podem veure, que el F1 score, com és obvi, és més baix al 1 que al 0.

L’interval de confiança que ens ha donat ens indica la precisió en la estimació del f1 score. Aquest interval ens assegura que amb un 95% de confiança, la veritable puntuació de f1 estarà dins d’aquest rang. Podem veure que és un rang bastant ajustat i bastant alt, això ens indica que tenim un bon model.

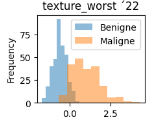
Seguidament el que fem és dibuixar aquest decision tree,

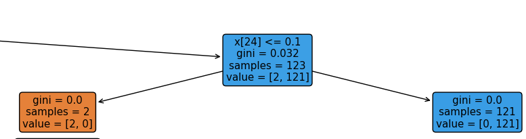
(Imatge al annex i format png adjuntat al zip)

Degut a que no es veu molt bé tenim també la representació en format txt.

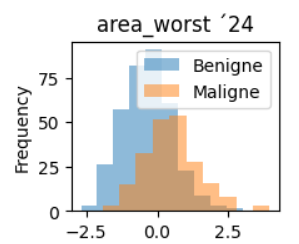
Seguidament comentarem el resultats d’aquest arbre, relacionant-ho amb els histogrames que hem fet al començament.

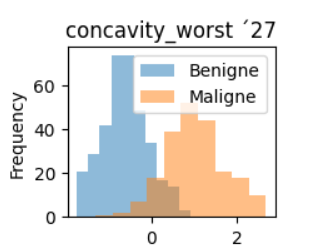
Com podem veure, aquest decision tree, comença amb la feature 22, que si ens fixem en el seu histograma:



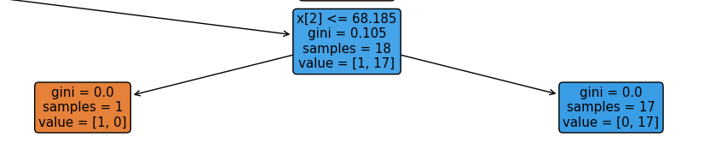
Ja podem intuir que podria ser aquesta. ja que ofereix una molt bona separació entre malignes i benignes, això farà que en aquesta primera volta ja classifiquem molts dels casos. Els casos que quedaran per classificar, serien els representats per la zona que s’ajunta el blau i taronja. Pel cas de l’esquerra, que respresentaria els positius, tenim després la feature 27. Que també ens ofereix molt bona separació. Per la dreta tenim la 24, que com podem veure ja està molt ben separada de el node anterior, només tracta algun cas molt concret. Aquest correspondria a l'últim quadrat taronja de l’esquerra.

En aquest extracte de la imatge del decision tree ho podem veure, com arriba a les fulles classificant tots els casos, i com hem dit classificant molt pocs en l'últim node.





Per seguir analitzant tot l’arbre el procediment a seguir seria anar mirant aquests histogrames i relacionar-ho com hem fet anteriorment, però degut a que és un procés molt mecànic i repetitiu, ho deixarem aquí i passarem al següent punt. Abans però cal comentar que la divisió de cada classe en cada un dels nodes ha estat molt bona, ja que no hem hagut de baixar molts nivells, això ens indica que tenim una primera feature que ens diu molta informació. També veiem que en algun cas com el següent, on s’afegeix una divisió per només un element (per aquesta divisió de dades) i podria indicar que aquest cas fos una dada no real.

  
Un cop analitzat el arbre, farem un grid search per trobar els millors paràmetres de min\_impurity\_decrease i min\_samples\_split. El min\_impurity\_decrease determina la quantitat mínima de reducció de la impuresa requerida per dividir un node de l'arbre. Un valor més gran d'aquest paràmetre pot conduir a arbres més petits, ja que només es realitzen divisions quan la reducció de la impuresa supera aquest llindar. Un valor òptim per a aquest paràmetre pot ajudar a prevenir sobreajustament (overfitting) i a obtenir un model més generalitzable. En min\_samples\_split estableix el nombre mínim de mostres requerides per dividir un node intern. Si el nombre de mostres en un node és inferior a aquest valor, la divisió no es realitzarà. Augmentar aquest paràmetre pot conduir a arbres més petits i menys complexos, també ajudant a prevenir sobre ajustament.

I després amb aquests paràmetres tornem a entrenar el DecisionTreeClassifier per obtenir els resultats finals. Veiem però que no millora, de fet empitjora el resultat (pel que estem buscant), ja que tenim un recall molt alt per 0, però realment busquem maximitzar-lo pel 1.

Finalment ens quedem amb la versió sense l’ajustament que ens ha donat millors resultats, amb un 0,93 de precision, 0,89 de recall i f1-score de 0,91 amb un interval de confiança de 0,86 i 0,94.

Pel que fa a la polaritat de les fulles, observem que aquestes poden ser o bé positives o bé negatives. Doncs bé, la quantitat de cada tipus i la distribució que es segueix en l’arbre dóna pistes sobre la fiabilitat d’aquest.

A continuació comentem alguns aspectes que caracteritzen un arbre de decisió robust:

* Proporció igualada de fulles positives i negatives.
* Coherència entre fulles adjacents, és a dir que dues fulles properes continguin majoritàriament la mateixa classe de dades.
* Tamany de la fulla (si aquesta és massa petita és molt possible que el model tingui overfitting)

### SVMs

En aquest apartat hem treballat aplicant l'algorisme de Support Vector Machines. Aquest alogirtme consisteix en trobar el hiperplà que millor separa les classes de diferents objectes en l’espai de característiques. Les característiques son els diferents atributs de cada exemple. El que busca SVM és trobar el hiperpla que maximitza el marge (la distància entre el hiperpla i els exemples més propers de cada classe.

Començarem amb el Kernel Lineal, provant com en totes les ocasions amb l’opció normalitzada i amb l'opció que no ho està. Ens surt millor resultat en la normalitzada, ja que en aquest algoritme sí que importen els pesos.

El que hem fet per descobrir que és millor normalitzada, és entrenar un model i fer prediccions sobre aquest. En el primer cas utilitzem un kernel lineal i obtenim un resultat en F1 de 0,935. És un bon primer resultat. El kernel lineal utilitza una funció de separació lineal, això ens indica que el dataset es pot separar bé linealment, no contempla les relacions que no son lineals.

Per millorar aquest resultat busquem el millor valor del paràmetre de regularització C amb un grid search amb validació creuada. El que fem és visualitzar com canvia la precisió mitjana amb els diferents C. Seguidament entrenarem una nova SVM amb el millor valor de C, que és 0,1. Això ens millora també el resultat dins a 0,94 en F1 score.

Seguidament, mirem com el rendiment del model evoluciona en una altra gràfica on veiem el training score i el cross validation score. D’aquesta manera tornem a verificar que és 0.1.

Continuem mirant amb kernel polinòmic. Aquest permet capturar estructures més complexes que no es poden fer en un lineal. Quan augmentem el número del polinomi estem augmentant la dimensió del hiperpla i per tant del espai de característiques. Això tot i que sembla que sempre ha de millorar el resultat, no és així, ja que pot afegir complexitat al model i risc de sobreajustament.

Primerament provem amb el de grau 2, repetirem el procés que hem fet abans i ens donarà un resultat de 0,625. També buscarem el millor valor de C i ens donarà 100. Buscarem el resultat per els polinomis de grau del 1 al 10 amb un bucle, però ens donem compte de que no millorarà significativament, apenes puja un 1,5%.

Per acabar realitzarem el RBF kernel, aquest transforma les dades a un espai de característiques infinits. Això és especialment útil quan les relacions no són lineals o quan les dades no son separables de manera lineal. Tornem a entrenar un nou model aquesta vegada en aquest RBF kernel i tenim un f1 score de 0,935.

També diu que tenim un C de 10 i un gamma de 0.01. Gamma és la influència dels punts d’entrenament en la construcció del límit de decisió.

Tornem a entrenar el SVM aplicant els resultats de C i gamma i obtenim un resultat de 0,94. També obtenim un molt bon classification report amb una precision de 0.98, un recall de 0.91, que ens dona un f1-score de 0.94.

Falta comentar com afecten el numero de supports i explicar quin és el millor kernel que s’ha aplicat.

### META

#### Voting scheme

El voting scheme és un model que barreja les decisions de diferents models, optimitzant i potenciant els seus resultats. Com que la nostre model voting scheme serà la combinació de Naive Bayes, KNN i Decision trees primer fem un cross validation per obtindre els resultats de cada model per separat. Seguidament, i amb les mateixes dades entrenem la voting scheme.

Per la voting scheme tenim dues formes d’entrenar-la:

* Hard voting: Aquesta premisa indica que cada model vota per un resultat, i el que tingui més vots és el resultat final.
* Soft voting: En aquest cas, cada model retorna les probabilitats de cada opció. D’aquesta manera es fa la mitjana de les probabilitats i s’acaba decidint el resultat final.

Pel que fa als resultats, ens hem fet proves buscant màxim accuracy, recall y f1-score. Dels tres cross validations, el que obté una puntuació més alta és l’accuracy, seguit de l’f1-score, i el recall.

Una de les coses importants que trobem és la influència del Naive Bayes sobre la voting scheme de soft voting. Tenint en compte el funcionament d’aquest tipus de votació, i lo que explicarem de les probabilitats que proporciona el Naive Bayes (molt proper a 0% o molt proper a 100%), el Naive Bayes té un pes molt important quan s’equivoca.

I explicat en un exemple numèric: Imaginem que per un tumor en concret s’obtenen les següents probabilitats per a un càncer maligne:

* Naive Bayes: 0.17%
* KNN: 64%
* Decision Tree: 75%

En el cas de hard voting el resultat seria tumor maligne, ja que hi ha ⅔ models ho assumeixen. Però que passaria en un soft voting? La mitjana dels tres valors seria 46,39%, i el resultat final seria càncer benigne.

El cas més extrem on notem aquest fet és per al recall, on la puntuació del soft voting scheme és inferior a la del decision tree degut a que el voting scheme tendeix més a obtenir un resultat més proper al del Naive Bayes, que en aquest cas és encara inferior.

#### Bagging

Per a aquest model el que fem és entrenar vàries instàncies de decision trees. I com que aquest model té una part aleatòria en l’entrenament del model, cada instància té molta probabilitat de ser diferent a la resta. Usem aquest algoritme amb 1, 2, 5, 10, 20, 50, 100, i 200 estimadors (en altres paraules, models entrenats),

Això ho fem primer de forma normal, i després ho tornem a repetir afegint un valor màxim de features de 35%, el que vol dir que el 35% del que es decideixi al model serà aleatori.

A més, hem fet càlculs de puntuació de cross validation per obtenir l'accuracy, el recall i l’f1-score.

#### Random forest

El Random forest, tal i com el propi nom indueix, tracta múltiples models d’arbres de decisió. En el nostre cas utilitzem dos models, el randomForestClassifier i l’extraTreesClassifier.

El primer d’ells selecciona les dades fent bootstrap per tindre diferents datasets per cada arbre. Després a partir del resultat de cada model es fa una votació i es retorna el que s’hagi aprovat per majoria.

El segon però, és encara més aleatori, ja que crea arbres més simples encara. Ja que escull les característiques de construcció de l’arbre aleatòriament en cada nivell de l’arbre. Un altra diferència és que escull només les prediccions de màxim valor de confiança de cada model.

Com a resultat, aquest últim és més ràpid per tindre major implicació de l’aleatorietat. Pel que fa el resultat, podem observar com l’ExtraTreesClassifier està una mica per sota quan es té un nombre de classificadors relativament petit. Però a mesura que aquest valor augmenta, no només que iguala el RandomForestClassifier sinó que en alguns casos el supera.

#### Boosting

Per aquest apartat entrenem tres models que utilitzen l’enfocament de boosting. És a dir que a partir de més models febles, es van millorant iterativament fins a aconseguir un model fort. Es caracteritza per donar més pes als tumors mal classificats en cada iteració.

El primer d’ells utilitza com a models inicials arbres de decisió, el segon igual però amb una limitació de profunditat de 5, mentre que el tercer utilitza arbres de decisió residuals. Els arbres de decisió residuals són arbres que estan entrenats a partir de les diferències entre les prediccions de la iteració anterior i el resultat original.

Una altra diferència està en l’ús dels models creats. En el primer i el segon, en cada iteració es creen nous arbres petits de decisió que el que fan és corregir els errors dels models de la iteració anterior donant més pes als tumors mal classificats. Mentre que en l’últim cas el que es busca més aviat és crear els nous models buscant reduir les diferències entre la predicció anterior i el resultat original.

Els resultats parlen per si mateixos, el primer model, amb arbres de decisió sense profunditat limitada són els que obtenen un millor resultat i que s’aconsegueixen entrenar més ràpid dels 3.

## 

## 4.- Comparacions i conclusions

Aquest projecte ha estat molt interessant, ens ha ajudat a entendre millor el món de les dades i el Machine Learning.

Coses tan poc obvies com la importància del dataset. I és que de bones a primeres sembla mentida que l’organització de les dades, simplificació d’aquestes o millora puguin afectar el rendiment d’un determinat model, si en teoria la informació és la mateixa. Tot i així, pel contacte amb altres datasets no comentats al treball i per la diferència de resultats entre models entrenats amb dades tractades i sense tractar podem concluir amb seguretat que el tractament de dades és un requeriment indispensable per a qualsevol ML.

Deixant de banda el processament de les dades, hem observat també una diferència no només entre models, sinó també entre les diferents formes d’entrenar un model utilitzant un mateixa funció. Tan sols cal observar la quantitat de resultats i puntuacions que n’obtenim per cada tipus de model…

Del Naive Bayes, ens queda clar l’agressivitat i firmesa que té a l’hora de donar un veredicte. I això ho hem descobert a cost de perdre el nostre temps intentant buscar l’error en el nostre codi de perquè obteniem un llindar tan petit per al recall. L’error no existeix, simplement la distribució tan exageradament separada de probabilitats és la que provoca que el llindar es situi tan aprop del 0.

I més que canvis dràstics en el llindar, s’han de considerar els canvis influents que aquest té en les Voting Machine de votació per mitjana (soft Voting Machines). És important conèixer el funcionament de cada model afegit en una Voting Machina abans d’implementar-la per evitar donar més prioritat involuntàriament a uns models sobre altres.

En KNN hem pogut comprovar de primera mà el que havíem vist a la teoria, l’importància de la normalització de dades per a un model que es basa en distàncies. En aquest apartat, però, hem après a no enfocar-nos massa en les dades de test. Sobre tot a l’hora d’escollir llindars, o paràmetres d’entrenament d’un model. Que sempre és indicat fer proves amb més valors diferents independentment dels resultats d’una prova de puntuacions específica.

Aquest tipus de funcions serveixen per fer una pinzellada només de quins podrien ser els valor òptims, molt útils per saber quins paràmetres probar. No obstant, barajar entre diferents opcions és el procediment més sensat i que aportarà millors resultats a l’entrenament del nostre model.

Els decision trees no ens han portat gaire problemes. Es tracta d’un algoritme eficient a l’hora de realitzar classificacions. I sobretot, es diferencien de la resta per proporcionar un resultat fàcil d'interpretar per humans. La mateixa fotografia inclosa en el zip del projecte en format png, serveix per classificar qualsevol tumor, de manera que aquest tipus de model no requereix ni tan sols de poder computacional. Simplement se li pot entregar aquesta fotografia a un doctor, i seguint les indicacions e spodria fer una idea del que prediu un model d’arbre de decisions.

De tots els models, els que tenen un cost temporal més elevat són les SVMs. En aquest apartat, ens ha sorprès a priori la manca de millora entre el kernel lineal i el polinòmic de 2n grau. Ens esperàvem al menys una petita millora.

Entenent una mica millor les bases de l’algoritme hem pogut concloure que el que veiem era totalment normal, i que tenia tot el sentit del món. De fetsi els resultats no milloren en un respecte l’altre és perquè l’hiperpla és pot representar linialment, de manera que representar-lo en forma de polinomi no aporta cap avantatge significatiu.

Per últim en l’apartat final, es veu la millora que s’obté al juntar diferents models i unir les seves decisions per obtenir un resultat mes satisfactori. Cadascun d’ells actua de manera diferent, però els seus pros i contres estan comentats cadascun al seu apartat.

En la taula final de resultats, mostrem els millors valors trobat d’f1-score per cada model entrenat:

| nom model | f1-score (%) |
| --- | --- |
| Naive Bayes | 94,5 |
| KNN | 92,5 |
| Decision Tree | 93,12 |
| Lineal Kernel | 93,44 |
| Degree 2 Kernel | 73,33 |
| RBF Kernel | 94,31 |
| Voting Machine | 93 |
| Baging | 95,5 |
| Random Forest | 96,1 |
| Boosting | 96,3 |

Tal i com es pot observar, en general la tendència és molt bona, els valors estàn compresos majoritàriament entre un rang del 92 i 97 per cent.

## 

## Annex

