Universitat Politècnica de Catalunya

GRAU EN CIÈNCIA I ENGINYERIA DE DADES

Projecte AA1

Aprenentatge Automàtic

Guillem Garrido Bonastre i Adrian Quirante González

Curs 2023-2024

3 de Juny del 2024

Índex

1 Descripció general del projecte

L'objectiu d'aquest treball és treballar amb un dataset per predir si una persona cobra més de 50k d'euros a l'any. Per tant, el nostre target és una variable binària (categòrica) amb dos classes: <=50K i >50K en funció de si cobra més o menys de 50K euros a l'any.

Tenint en compte que 'C' indica que és una variable categòrica i 'N' que és numèrica, els atributs a partir dels quals estudiarem i farem les prediccions del nostre target són els següents:

- **Age(N)**: Edat.
- **Workclass(C)**: Tipus de treballador (per exemple, si és autònom o pertany a una empresa privada, entre altres categories).
- **Fnlwgt(N)**: Valor que s'associa a les característiques d'una persona i el seu origen. Persones amb la mateixa descendència tenen valors similars d'aquest atribut.
- **Education(C)**: Nivell màxim d'educació assolit.
- **Education-num(C)**: Número corresponent al nivell màxim d'eduació assolit.
- **Marital-status(C)**: Estat civil.
- **Occupation(C)**: Sector en que treballa.
- **Relationship(C)**: Relació amb la seva família.
- **Race(C)**: Origen.
- **Sex(C)**: Sexe.
- **Capital-gain(N)**: Guany de capital d'una persona en relació amb les inversions que fa. No és el que cobra, sinò els benefecis que treu de les seves inversions (compres i vendes).
- **Capital-loss(N)**: Pèrdua de capital en les inversions
- **Hours-per-week(N)**: Hores de treball a la setmana (5 dies laborals).
- **Native-country(C)**: País d'origen.

Cada observació/fila del nostre dataset correspon a una persona. Al descarregar el dataset, ja ens ve separat una part per training i una part per fer la validació. Així doncs, comptem amb 32561 files del dataset d'entrenament i 16281 del dataset de test.

Per acabar, tot el codi utilitzat per realitzar el treball, obtenir resultats com mètriques, gràfiques... es troba en el document 'codiProjecteAAI_GarridoGuillem_QuiranteAdrian.ipynb'.

2 Preprocessament

A continuació [1], comentarem les transformacions pròpies del preprocessament que aplicarem sobre el nostre dataset abans de començar amb la construcció de models per fer prediccions. Cal remarcar, que en el porcessament hem treballat per separat amb el set de 'training' i el set de 'test', tot i que hem aplicat les mateixes transformacions.

[1] La descripció del prerocessament realitzat la farem a través del training set, però hem realitzat el mateix en anàlisi i transformacions pel test set (ho podeu trobar en el document amb el codi).

2.1 Visió general del dataset

Abans de començar amb les diferents transformacions, és necessari fer una inspecció inicial del dataset, per veure quins procediments haurem d'utilitzar per fer el preprocessing.

Fixant-nos en la Figura 1, el primer que ens crida l'atenció són els atributs **Capital-gain** i **Capital-loss**. Si ens fixem, fins al tercer quartil tenim valors igual a 0. De fet, tenim que 29849 de les 32561 observacions tenen 0 de Capital-gain i 31042 tenen 0 Capital-loss. A més, veiem un màxim extrany amb valor 99999, que ens fa sospitar que pugui ser un possible "missing value". Aquest fet, ens farà reconsiderar com treballarem amb aquestes variables.

	Age	Fnlwgt	Education-num	Capital-gain	Capital-loss	Hours-per-week
count	32561.000	3.256e+04	32561.000	32561.000	32561.000	32561.000
mean	38.582	1.898e+05	10.081	1077.649	87.304	40.437
std	13.640	1.055e+05	2.573	7385.292	402.960	12.347
min	17.000	1.228e+04	1.000	0.000	0.000	1.000
25%	28.000	1.178e+05	9.000	0.000	0.000	40.000
50%	37.000	1.784e+05	10.000	0.000	0.000	40.000
75%	48.000	2.371e+05	12.000	0.000	0.000	45.000
max	90.000	1.485e+06	16.000	99999.000	4356.000	99.000

Figura 1: Resum de les variables numèriques del dataset amb la funció 'describe'

D'altra banda, ens crida l'atenció els valors màxims i mínims de l'atribut **Hours-per-week**: el mínim és una persona que treballa 1 hora a la setmana i el màxim 99 hores a la setmana, que correspondria a 20 hores diaries en 5 dies laborals o 14 hores diàries si tenim en compte els caps de setmana. Aquests valors són molt extranys i a priori semblen valors atípics que haurem de tractar.

Per acabar la detecció de inchorències en variables concretes, hem vist, que tot i que a simple vista, la variable "Education-num"sembla una variable numèrica, és una variable categòrica que representa exactament el mateix que la variable "Education", simplement que els noms que reben els diferents nivells d'educació, venen representats per números.

L'últim apunt que podem treure de la figura 1 és l'escala de les variables. Si ens fixem l'atribut **Fnlwgt** té valors molt grans, de l'ordre de 10^5, mentre que **Education-num** pren valors fins al 16. Això últim ho haurem de normalitzar, perquè sinó ens pot donar problemes a l'hora de treballar amb alguns models.

	Workclass	Education	Marital-status	Occupation	Relationship	Race	Sex	Native-country	Income
count	32561	32561	32561	32561	32561	32561	32561	32561	32561
unique	9	16	7	15	6	5	2	42	2
top	Private	HS-grad	Married-civ-spouse	Prof-specialty	Husband	White	Male	United-States	<=50K
freq	22696	10501	14976	4140	13193	27816	21790	29170	24720

Figura 2: Resum de les variables categòriques del dataset'

En la figura 2, podem veure l'estadística de les variables categòriques. Primer de tot, i el més rellevant, és que estem treballant amb un dataset desbalancejat, ja que tenim moltes més observacions corresponents a la classe «=50k"(24720) que a la classe »50k"(7841). Aquest fet serà important tenir-lo en compte més endevant.

Un altre aspecte a destecar, però que no és gaire rellevant, és que el prototipus de persona més freqüent en les observacions del nostre set és un home de raça blanca d'origent Nord Americà.

Per acabar, hem volgut tenir una visualització general de la relació de cada variable amb el target. Les variables més destacades són les següents:

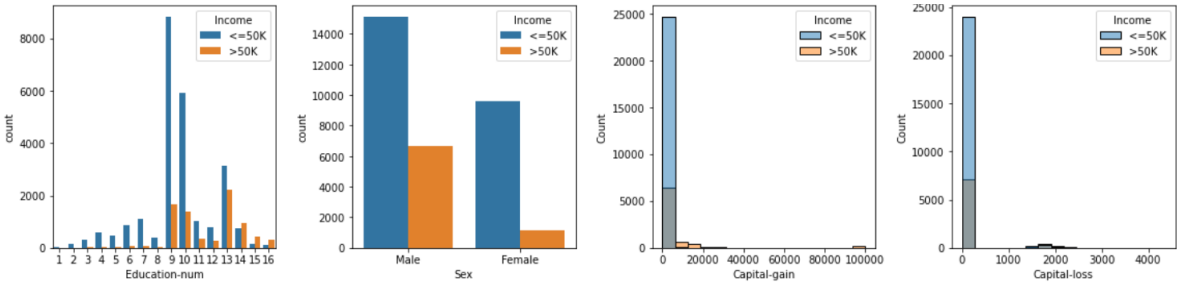


Figura 3: Relació entre variables i target

En la figura 3, podem veure que el nivell d'educació és molt important en el fet de cobrar més o menys de 50k, ja que com més estudis tinguis és molt més probable que guanyis més de 50k. D'altra banda, també podem veure, que la proporció d'homes que cobren més de 50k és molt més elevada que no pas la proporció de dones.

Per acabar, tot i que no es pugui visualitzar gaire bé, podem veure que quasi tothom que té un 'Capital-Gain' positiu cobra més de 50k, mentre que qui té un 'Capital-Loss' positiu cobra menys de 50k.

2.2 Feature extraction

Com hem vist anteriorment, el fet de tenir un 'Capital-gain' i/o 'Capital-loss' positiu està bastant relacionat en el resultat del target. Per aquest motiu, hem decidit crear una nova variable anomenada 'Capital-difference' que recolli la informació de la diferència de les dos variables anteriors, de manera que recollim la informació que ens donen 2 variables en una única variable i simplifiquem una mica l'estudi.

2.3 Feature selection

En la selecció de variables, hem decidit eliminar-ne 3. Primerament, hem eliminat les variables 'Capital-gain' i 'Capital-loss', ja que, com hem creat una nova variable que contè la informació d'aquestes dos, hem decidit que mantenir-les ens aportava informació redundant.

D'altra banda, també hem decidit eliminar la variable "Education-num"perquè explica exactament el mateix que l'atribut "Education", però en números, és a dir, cada nivell d'estudis reb un nom (atribut "Education") i a cada nivell d'estudis se li atribueix un número (atribut "Education-num"). Per tant, com ambdues variables aporten informació idèntica, hem decidit prescindir d'una d'elles.

2.4 Tractament de missing values

Aquesta secció, la dedicarem a la detecció de missing values. En el cas del nostre set, diferenciem 2 tipus de missing values: els explícits que els identifiquem amb el símbol '?' i els implícits que són valors extrems molt allunyats de la resta de dades.

Començant pels missing values explícits, si ens fixem en el valor que prenen totes les variables, hem pogut veure que les úniques que prenen el valor '?' són "Workclass", "Occupation i "Native-Country". El número de missing values trobats els podem veure en la taula 1:

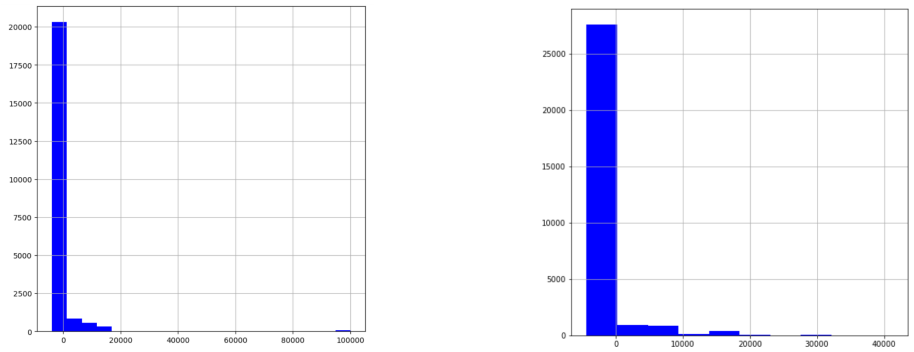
Atribut	Observacions amb MV
Wrokclass	1836
Occupation	1843
Native-Country	583

Taula 1: Observacions amb missing values

En total tenim 4262 missing values. Com aquests es troben en atributs categòrics i la imputació per aquest tipus d'atribut està fora dels continguts d'aquesta assignatura, hem decidit eliminar les files que contenen aquests missing values ja que suposen una petita proporció de les observacions totals.

D'altra banda, pel tractament de missing values explícits, hem cregut que l'única variable que en podia contenir és la variable 'Capital-difference'. En l'apartat 2.1 hem vist que el valor màxim de 'Capital-gain', que és una de les variables a partir de la qual hem extret 'Capital-difference', era 99999 i això ens ha fet dubtar de si era un missing value o no. A l'analitzar aquesta situació en profunditat hem vist que hi havia 86 files amb valor 99999 i, en canvi, 0 files amb valor entre els 40000 i els 90000. Això ha confirmat la nostra sospita inicial i tractarem aquests valors com missing values.

En aquest cas, si que podem imputar els missing values perquè es troben en una variable numèrica. Així doncs, utilitzem l'algorisme K-Nearest Neighbours amb 1 veí per tal de realitzar la imputació. Cal recalcar que els veïns es dessignaran en funció de les altres variables numèriques, sense tenir en compte les categòriques. En la següent figura podem veure el resultat abans de la imputació i després de la imputació de l'atribut 'Capital-difference':



(a) Abans d'imputar missing values

(b) Després d'imputar els missing values

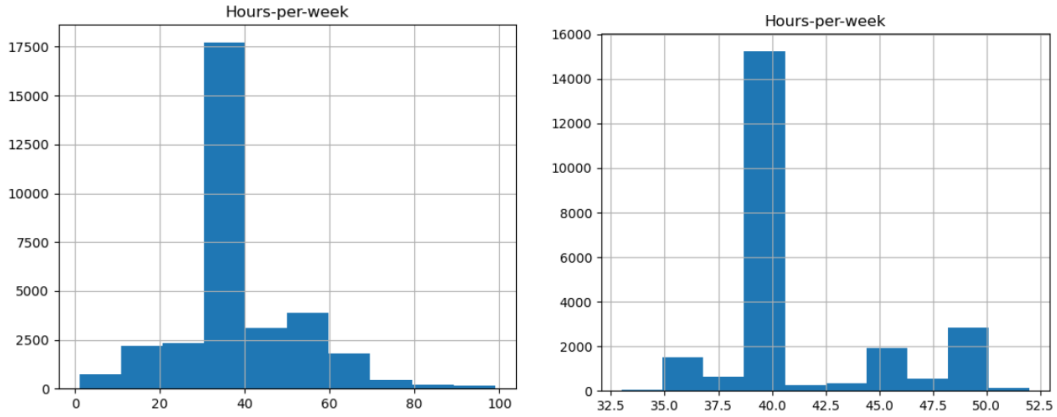
Figura 4: Comparació de l'histograma de l'atribut 'Capital-Difference' abans i després d'imputar

2.5 Tractament d'outliers

En la secció 2.1 ens hem adonat que l'atribut "Hours-per-week"presentava valors molt extranys, per tant, hem considerat que que la única variable en la que té sentit considerar outliers és en l'atribut "Hours-per-week". Si volguéssim buscar outliers en els atributs "Age"o "Fnlwgt"perdríem informació d'una part de la població que és important pel nostre estudi.

En l'atribut "Hours-per-week"hi ha persones que treballen 1 o 99 hores setmanals, fet que és molt improbable. Per això, utilitzarem el criteri del IQR per treure els valors extrems d'aquesta variable.

En la figura 5 podem observar com canvia la distribució de l'atribut "Hours-per-week"després d'eliminar els outliers. És important destacar com el nostre rang de valors passa de [0, 100] a [30, 50] aproximadament. Això ja té més sentit, doncs la jornada laboral estàndard és de 40 hores i considerar una variancia de 10 hores al voltant d'aquest valor és beneficiós pel nostre dataset i més realista.



(a) Abans de tractar els outliers (b) Després de tractar els outliers

Figura 5: Comparació de l'histograma de l'atribut Hours-per-week abans i després de tractar outliers

2.6 Normalitat

En aquesta part del preprocessament estem mirant si les nostres variables continues es distribueixen normalment o no. Això ens interessa mirar-ho perquè molts models assumeixen normalitat de les variables i una distribució allunyada d'aquesta normalitat podria ser perjudicial per l'estudi. Els dos atributs els quals podem corregir la seva normalitat utilitzant el logaritme són **Age** i **Fnlwgt**.

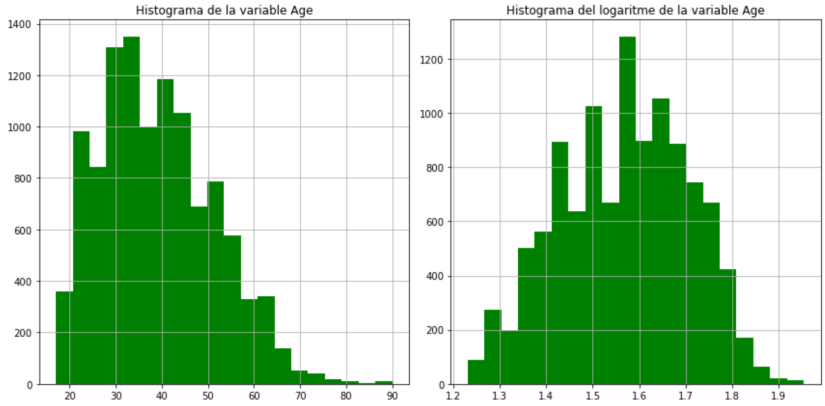


Figura 6: Comparació dels l'histogrames de l'atribut Age abans i després d'aplicar el logaritme

Podem observar que en la figura 6 i la figura 7, els histogrames abans d'aplicar el logaritme (histograma de l'esquerre) tenen una cua bastant més llarga que l'altra, provocant un allunyament de la distribució normal que busquem. En canvi, en aplicar el logaritme podem corregir la distribució perquè sigui més normal tal i com es veu en els histogrames de la dreta.

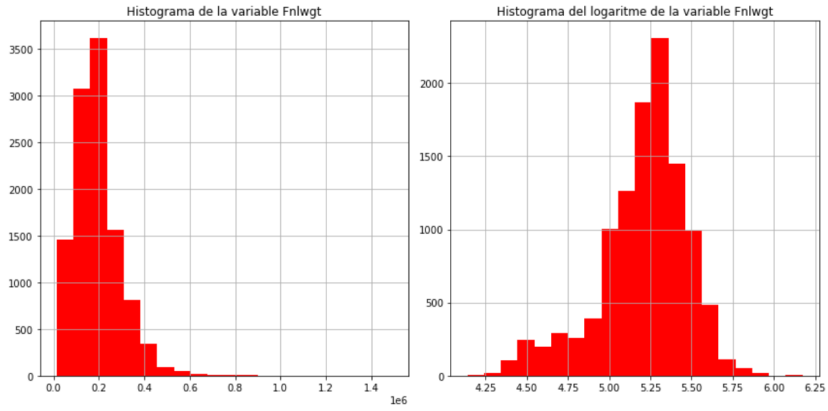


Figura 7: Comparació dels l'histogrames de l'atribut Fnlwgt abans i després d'aplicar el logaritme

2.7 Normalització

L'última transformació que hem de fer en el nostre dataset és normalitzar les variables numèriques, de manera que tinguin un rang similar. Hem decidit normalitzar perquè l'atribut "Fnlwgt"té valors de l'ordre de 10^5 i 'Capital-difference' valors d'ordre 10^3, mentre que "Age"té un rang de [0, 100] i "Hours-per-week"té un rang de [30, 50] aproximadament.

Hem deicidit escalar aquestes variabes utilitzant la transformació Mix-Max Scaling perquè acabem d'aplicar el logaritme en dos atributs i volem que es mantingui la seva dsitribució normal. A més, ens interessa escalar les variables en l'intèrval [0, 1] perquè els valors escalats no perdin el seu significat. Per exemple, valors de l'atribut "Age"petits es relacionen millor amb valors propers al 0, que amb valors negatius, que serien obtinguts si haguèssim aplicat standarization.

En aplicar la normalització de les variables contínues ens trobem amb un problema. La variable nova 'Capital-difference' ha perdut la seva estructura degut a la seva gran quantitat de valors 0 i la presència de valors negatius. En concret, els valors 0 s'han desplaçat a un altre valor i els negatius a els valors més propers a 0. Els nous valors que pren la variable 'Capital-difference' no són representatius de l'atribut i això no ens agrada.

2.8 Capital-difference categòric

Per solucionar l'anterior problema hem decidit transformar aquesta variable en una variable categòrica. Aquesta variable conté tres classes:

- 'Gain': Si la persona tenia 'Capital-difference' positiu.
- 'Same': Si la persona tenia 'Capital-difference' igual a 0.
- 'Loss': Si la persona tenia 'Capital-difference' negatiu.

D'aquesta manera, aconseguim mantenir l'estructura de l'atribut, preservant la seva informació impotant (si la persona guanya, perd o es queda igual). Això ens pot ser més útil per predir 'Income' que normalitzant els seus valors.

3 Modelització

Un cop realitzat tot el procés de preprocesament de les dades ja podem començar a treballar amb diferents models per a fer prediccions i quedar-nos amb aquell que ens doni millors resultats. Cal

recordar que estem treballant amb un dataset desbalancejat, aquest fet l'haurem de tenir en compte més endevant a l'hora de valorar la precisió dels models.

Primer de tot, a través del PCA visualitzarem el plot de les observacions d'entrenament:

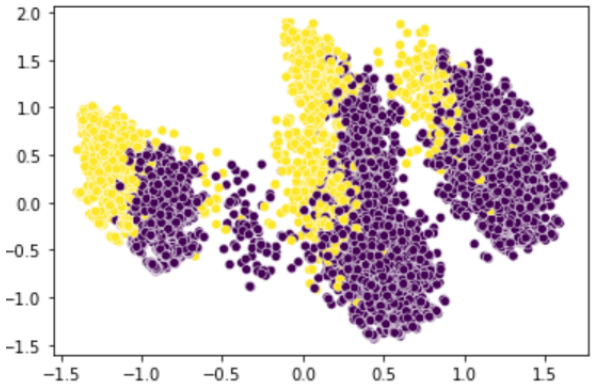


Figura 8: Plot de les observacions mitjançant la PCA

Com podem observar en la figura 8, podem veure com efectivament es tracta d'un dataset desabalancejat ja que hi ha moltes més observacions de color lila (corresponents a <=50k) que no pas de color groc.

Per altra banda, observem un patró peculiar a les dades. Podem diferenciar tres blocs clars que, si els analitzèssim per separat, podrien ser separats linealment. En canvi, en tenir en compte totes les dades, sembla que el problema de classificació no és lineal i ens donaran millors resultats models d'aquest tipus.

3.1 One-hot encoding

Abans de començar, però, hem creat dos nous sets de train i test utilizant la tècnica "One-hot encoding"que bàsicament s'encarrega de transformar les variables categòriques, de tal manera que puguin ser proporcionades als algorismes d'aprenantatge autòmatic que emprarem més endevant, ja que molts d'ells si no apliquem aquesta transformació no els podríem utilitzar.

Aquí entra en joc un nou ptoblema: la col·linearitat entre les variables. Com tenim variables amb moltes categories diferents (per exemple 'Native Country' té 42 categories diferents) això provoca generar moltes columnes per explicar una sola variable i que aquestes columnes estiguin molt relacionades entre elles (perquè la suma dels seus valors per una observació ha de ser 1). Això ho hem de tenir en compte quan interpretem els resultats perquè pot ser una limitació important.

3.2 Mètriques

Finalment, les mètriques que utilitzarem per evaluar el rendiment dels nostres models seran les següents:

Accuracy

Aquesta mètrica, bàsicament, representa el percentatge d'observacions classificades correctament. Aquesta mètrica no és gaire representativa per datasets desbalancejats, per tant, no li donarem gaire pes. Tot i així, creiem que és important tenir-la present.

Precision Macro

Aquesta mètrica representa la relació de vertaders postius entre la suma de vertaders positius i falsos positius de cada classe. Al ser "Macro"ens referim a la mitjana de la precisió de totes les classes. Ens serà útil per calcular posteriorment la mètrica en la que evaluarem el rendiment dels nostres models.

Recall Macro

Aquesta mètrica és similar a l'anterior, però en aquest cas mesura la relació de vertaders positius entre la suma de vertaders positius i falsos negatius per a cada classe. Al ser "Macro"ens referim a la mitjana de la sensibilitat de totes les classes. Ens serà útil per calcular posteriorment la mètrica en la que evaluarem el rendiment dels nostres models.

F1-score Macro Averages

La mètrica F1-score té en compte la sensibilitat i la precisió del model al fer prediccions. Com es tracta d'un dataset desbalancejat, aquesta serà la mesura que tindrem més present a l'hora d'evaluar el rendiment d'un model.

Hem decidit utilitzar la mètrica 'F1-score Macro' i no per una classe concreta perquè ens interessa tenir en compte ambdúes classes per igual. Cap de les dos és més important que l'altre quan classifiquem i no és més important tenir falsos positus que falsos negatius o al revés. A més, no utilitzem 'F1 Macro Weighted' perquè volem mantenir la distribució de les classes sense donar-li major importància a una, com acabem de comentar.

3.3 Protocol de validació i comparació de models

Per comparar els models hem decidit utilitzar cross-validation, doncs ens sembla el millor procediment per analitzar el rendiment de cada model. Hem utilitzat 5 folds per fer cross-validation perquè ens sembla un número de particions suficient i eficient en funció de la mida de les nostres dades i no creiem necessari augamentar el cost computacional afegint més. Sempre que hem utilitzat cross-validation hem utilitzat aquest número de folds per poder realitzar una comparació amb igualtat de condicions entre tots els models.

En quant a la selecció de paràmetres, en molts casos hem decidit utilitzar la funcionalitat 'GridSearch'. Aquesta funció permet comparar els resultats de predicció de cada model amb cross-validation en funció de les diferents combinacions de paràmetres per veure quins paràmetres ens donen els millors resultats.

Un cop hem obtingut els millors paràmetres per cada model, realitzem la comparació de models en funció de les mètriques obtingudes amb la validació del dataset d'entrenament (fent cross-validation), de manera que el dataset de test conté dades que no s'han vist mai en el procés d'entrenament. Així doncs, les taules que presentem a cada apartat corresponen a les mètriques de validació en l'apartat de train.

Posteriorment, d'entre tots els models ajustats, seleccionarem el millor respecte les mètriques de train i estimarem la 'performance' de generalització sobre el dataset de test, sobre unes dades que no hem vist encara.

3.4 Models

Abans d'ajustar models, cal comentar que hem hagut de treure una observació. Això es deu a que en el dataset de train i havia una única observació amb una categoria que no apareixia en cap observació del test. Això era un problema perquè, en fer "One-hot encoding", teniem una columna més en el dataset de train que en el de test, i no ens deixava realitzar prediccions amb un model fitejat amb un altre dataset amb diferents columnes. Concretament la categoria és 'Holand-Netherlands' de la variable 'Native-country'. Ens hem permés treure la observació perquè no afecta gens a l'estructura del dataset i si no la traièssim tindriem problemes per ajustar models.

Un cop solucionat això i havent realitzat la transformació dels sets amb "One-hot encoding"ja podem començar amb la creació de models.

Els models amb els que han treballat, com ens trobem en un problema de classificació, són models útils i que rendeixen correctament amb datasets de classificació.

A mesura que ens han anat presentant nous models en l'assignatura hem anat treballant amb ells, i sempre hem procurat que dins d'una família de models començar a treballar sempre amb el més simple i mica en mica anar pujant la complexitat per veure com evolucionava el rendiment dels models.

Per últim, hem decidit posar els models QDA i Clustering als Annexos 6.1 i 6.3 respectivament perquè ens han donat els pitjors resultats, bastant allunyats de la resta, i creiem que no són rellevants pel nostre estudi. Per altra banda, hem afegit Support Vector Machines a l'Annex 6.2 degut a que, tot i que els seus resultats són bons (però no dels millors), el seu cost computacional és massa elevat (tarda molt) i no es comparable amb la resta de models.

3.4.1 LDA i RDA

Per començar, els primers models que hem entrenat ha estat amb aquesta família de classificadors lineals generatius.

LDA

Primerament, hem començant amb LDA, és a dir suposant que totes les matrius de covariàncies són iguals ($\Sigma_k = \Sigma$). Els resultats obtinguts es poden observar en la taula 2:

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
LDA	0.834	0.77	0.79	0.755

Taula 2: Mètriques obtingudes amb LDA

Com podem veure, la mètrica "F1-Macro"al test és molt propera al 80%,. Per tant, podríem dir, que el model LDA fa bones prediccions per són molt millorables.

RDA

El paràmetre de regularització sobre QDA que ens ha proporcionat uns millors resultats d'entre els valors de l'Annex 6.4.1 ha estat $\lambda = 0.2$. Els resultats obtinguts en l'entrenament els podem observar en la taula següent:

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
RDA	0.798	0.758	0.779	0.785

Taula 3: Mètriques obtingudes amb RDA

Com podem veure, els resultats milloren clarament respecte QDA (Annex 6.1) i queden uns resultats molt propers als que hem obtingut amb LDA. Tot i així, els resultats obtinguts amb LDA són un pel millors, això es dona a que les variables estan correlacionades entre elles i és millor considerar que hi ha una úncia matriu de covariàncies igual per totes les distribucions de probabilitat.

3.4.2 Naive Bayes

El següent model que hem aplicat ha estat el classificador generatiu Naive Bayes. Naive Bayes assumeix que els atributs de les probabilitats condicionades de les classe són independents, això no sol ser cert però pot ser una bona aproximació.

Per trobar el millor model de Naive Bayes, hem probat diferents combinacions:

- Gaussian Naive Bayes (amb totes les variables).
- Categorical Naive Bayes (únicament amb variables categòriques).
- Naive Bayes only numerical (Gaussian Naive Bayes només amb variables numèriques).
- Combined Naive Bayes (Categorical + Only numerical).

Els resultats obtinguts es poden veure en la figura següent:

	Accuracy	F1 Macro	Precision Macro	Recall Macro
Gaussian Naive Bayes	0.497	0.497	0.643	0.645
Categorical Naive Bayes	0.755	0.722	0.714	0.767
Gaussian-NB-only-numerical	0.745	0.572	0.643	0.571
Combined-NB	0.801	0.755	0.744	0.771

Figura 9: Mètriques dels models de Naive Bayes

Com podem veure en la figura 9, hem obtingut uns resultats gens sorprenents. Si ens fixem exclusivanent en la mètrica F1-score Macro (que és la més rellevant), podem veure que els pitjors resultats s'obtenen aplicant Gaussian Naive Bayes amb totes les variables i només amb les numèriques, fet que no ens sorpren en absolut ja que la gran part de variables del nostre dataset són categòriques.

En canvi, aplicant categorical Naive Bayes podem veure com els resultats milloren considerablement pujant la mètrica F1-score per sobre del 70%. A més si combinem els resultats obtinguts usant Categorical Naive Bayes i Gaussian Naive Bayes (només amb variables numèriques), podem veure que el valor de F1-score és una mica més alt i que la precisió en les prediccions arriba a superar el 80%.

Per tant, podríem dir que el millor classificador de Naive Bayes és aquell que combina el Categorical Naive Bayes (només amb les variables categòriques) amb Gaussian Naive Bayes (només amb les variables numèriques), fet que té sentit ja que al combinar els dos models podem captar tota la informació que ens donen totes les variables del dataset per fer les prediccions.

3.4.3 K-Nearest Neighbours

Després de fer el 'GridSearch' amb els paràmetres que podeu veure a l'Annex 6.4.2, hem trobat que la combinació que obté millors mètriques utilitza 15 veïns i la mètrica distància 'minkowski'. Els resultats obtinguts es troben a la taula següent:

Model	**Accuracy**	**F1 Macro**	**Precision Macro**	**Recall Macro**
K-NN	0.827	0.757	0.782	0.741

Taula 4: Mètriques obtingudes amb K-Nearest Neighbours

Els resultat obtinguts amb K-Nearset Neighbours són interessants, perquè ens donen més 'accuracy' i menys 'F1 Macro' que el 'RDA'. En aquest cas, obtenim un millor percentatge en l'apartat Macro que en Recall, per tant la limitació d'aquest model és que té molts falsos negatius.

Aquesta presència alta de falsos negatius és degut a que K-NN és sensible a la dimensionalitat alta. Com tenim variables amb moltes categories, fer One-Hot encoding d'aquestes variables implica augmentar la dimensionalitat considerablement, i aquest augment implica una dificultat major per diferenciar les observacions.

A part d'això, no obtenim uns resultants dolents gràcies a que, en el proprocessament, hem normalitzat les variables numèriques i hem tret variables irrelevants en l'apartat 'feature selection'.

3.4.4 Logistic Regression

En el cas de Logistic Regression, el paràmetre de regularització C és la inversa de λ. Per tant, un valor més petit de lambda indica més regularització.

En calcular les mètriques sobre el dataset d'entrenament amb els diferents valors de C (Annex 6.4.3) observem que obtenim valors molt propers amb tots. Per aquest motiu, considerem que no val la pena regularitzar i afegir cost computacional. No cambiarem el paràmetre C i aquest es quedarà amb el seu valor 1.0 que té per defecte.

Els resultats sobre el test d'aquest model són els següents:

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Logistic Regression	0.835	0.771	0.792	0.756

Taula 5: Mètriques obtingudes amb Logistic Regression

'Logistic Regression' destaca, sobretot en l'apartat 'Precision', on obté els millor resultats vists fins ara. Això i el bon rendiment general en l'apartat 'Recall' (tot i que destaca més evitant falsos positius) fan que el seu rendiment sigui una mica millor que el model 'LDA'.

Aquest model ha donat prou bons resultats gràcies a que hem realitzat una neteja dels outliers, als quals el model és molt sensible.

3.4.5 Decision Tree, Random Forest and Extra Trees

Decision Tree

Primer ajustarem un Decision Tree sorbre les dades, que és un model senzill, i després augmentarem la complexitat ajustant Random Forests i Extra Trees.

La combinació guanyadora (d'entre tots els valors que es troben en l'Annex 6.4.4) en aquest cas és utilitzar el criteri d'entropia, sense profunditat màxima, l'arrel del nombre total de variables per realitzar la divisió de nodes, considerar 5 observacions com a mínim per dividir nodes i 4 per considerar-los com a fulla.

Els resultats obtinguts amb aquests paràmetres els podem visualitzar en la taula 6.

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Decision Tree	0.82	0.749	0.771	0.735

Taula 6: Mètriques obtingudes amb Decision Tree

Ajustant un Decision Tree hem obtingut un pitjor rendiment general respecte 'LDA' i 'Logistic Regression'.

A continuació ajustem diferents Random Forests per veure si milloren les prestacions.

Random Forest

Un Random Forest ens permet utilitzar un conjunt de Decision Trees per poder evitar l'overfitting que aquests generen. No osbtant això, en el nostre cas no hem sobreajustat les dades d'entrenament, així que volem veure si és capaç de millorar la predicció.

Hem deicidit no utilitzar el 'Out-of-Bag' error com a mètrica de validació per donar el mateix pes tant a la classes minoritària com a les classes majoritària del nostre dataset desbalancejat. Per aquest motiu, compararem els models d'acord a les mètriques que hem anat utilitzant fins ara.

En aquest cas mirarem dos paràmetres diferents que en el Decision Tree: 'n_estimators' indica el número d'arbres que utilitzem i 'class_weight' indica el grau de balanceig que introduïm. Com ens volem centrar en aquests, la cerca exhaustiva dels paràmetres contindrà els valors dels paràmetres que han maximitzat el rendiment del Decision Tree. D'aquesta manera, volem veure, sobretot, quin és el nombre òptim d'arbres i la classe de balanceig que utilitzem.

Després de fer la cerca amb els valors de l'Annex 6.4.5 veiem que, d'entre 4 combinacions diferents que ens donen el mateix resultat de 'f1 Macro', el millor Random Forest utilitza 50 Decision Trees, de qualsevol profunditat, amb fulles de 5 observacions mínim i necessitant 4 observacions per poder dividir els grups.

És interessant veure que el millor arbre ens ha sortit amb 'class_weigth' = balanced_subsample. Aquest paràmetre combat el desbalanceig del nostre dataset, per tant ens ha ajudat a obtenir millors mètriques.

Una altre observació és que, tret de 'n_estimators' i 'class_weight', hem obtingut els mateixos millors paràmetres pel Decision Tree i Random Forest. Això reafirma que són paràmetres positius pel nostre model i ens fan trobar bons resultats.

Les mètriques obtingudes pel millor Random Forest queden pal·leses a la taula següent:

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Random Forest	0.795	0.763	0.75	0.804

Taula 7: Mètriques obtingudes amb Random Forest

Aquest model ha obtingut els millors resultats vists fins ara en l'apartat 'Recall', per tant és bo evitant falsos negatius. Això fa que el poguem considerar millor model pel nostre problema que el 'Decision Tree', degut a que hem obtingut millors resultats de 'F1 Macro'. Per tant, tot i que el 'Decision Tree' ha obtingut millor valor de 'accuracy', el 'Random Forest' ha aconseguit millorar el rendiment combinant diferents arbres.

Extra Trees

Per últim, volem anar un pas enllà. Extra Trees és un model que utilitza un conjunt de Decision Trees, tal i com fa Random Forest, però introduïnt més aleatorietat en la construcció dels arbres.

En aquest cas mirarem exactament els mateixos paràmetres que amb el Random Forest per poder comparar bé el rendiment entre els dos. Els millors paràmetres obtinguts els podem observar en la Figura 10.

	param_n_estimators	param_class_weight	param_max_depth	param_min_samples_leaf	param_min_samples_split
19	100	balanced	None	5	2
32	200	balanced_subsample	None	5	2

Figura 10: Paràmetres d'Extra Trees que donen millors resultats

A la Figura 10 veiem que hem tingut un empat en termes de 'F1 Macro' amb les dos combinacions de paràmetres. La primera combinació surt a dalt perquè, davant l'empat, obté millors resultats de 'accuracy', però aquesta mètrica no ens interessa a nosaltres.

Per tant, fixant-nos en la segona combinació, podem veure que l'únic paràmetre que ha canviat és el mínim d'observacions per dividir un node. Per tant, els paràmteres obtinguts en afegir aleatorietat en la construcció dels arbres són pràcticament els mateixos. Les mètriques es troben en la següent taula:

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Extra Trees	0.79	0.759	0.747	0.806

Taula 8: Mètriques obtingudes amb Extra Trees

En afegir aleatorietat a la construcció dels arbres no hem aconseguit millorar les prestacions respecte el 'Random Forest'. Això es pot deure, un altre cop, a la col·linearitat entre les columnes del nostre dataset. El fet que els Extra Trees afegeixin aleatorietat no ajuda a que la creació dels arbres no sigui afectada per la relació entre les variables, ja que tenim moltes columnes col·lineals i això impedeix que millorem el rendiment.

La última observació és la similitud d'actuació entre el 'Random Forest' i els 'Extra Trees', doncs els dos han destacat en un bon percentatge de 'Recall' i no de 'Precision'.

3.4.6 Gradient Boosting

'Gradient Boosting' es conegut per donar un rendiment molt alt. Volem veure si també és el cas del nostre problema.

Entre els valors dels parèmetres de l'Annex 6.4.7, trobem que el millor cas utilitza 200 arbres de profunditat màxima 5 i divideix nodes amb un nombre mínim d'observacions equivalent a l'arrel del número d'observacions total. A més, la contribució de cada arbre és d'un 0.1. Aquestes propietats donen els resultats que es mostren a la Taula 9.

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Gradient Boosting	0.838	0.777	0.795	0.763

Taula 9: Mètriques obtingudes amb Gradient Boosting

'Gradient Boosting' ens dona els millores resultats que hem vist fins ara amb una diferència clara respecte 'Logistic Regression' i 'LDA' en la mètrica que ens interessa. Aquest model aconsegueix el millor equilibri entre 'Precision' i 'Recall' que hem vist, sense aconseguir el millor resultat en cap de les dos mètriques. En quant a la 'accuracy', la diferència respecte a la resta no és tan clara, i això és una senyal de que el mètode és especialment bo pel nostre problema de dades desbalancejades.

3.4.7 Ensambles

L'últim model que hem considerat és la unió de diferents models per realitzar la classificació en funció del vot promitjat entre aquests. Es tracta d'un vot promitjat i no vot majoritari perquè hem obtingut millors resultats amb la modalitat 'Soft Voting' que amb la 'Hard Voting'. Hem decidit incluir aquest model perquè els models que ens han donat millors resultats són diferents entre si i això pot afavorir a predir correctament.

Hem decidit realitzar tres models, combinant els dos, tres i quatre millors models respecte les mètriques de validació en el train:

- **Voting Best**: 'Gradient Boosting' i 'Logistic Regression'
- **Voting Best 3**: 'Gradient Boosting', 'Logistic Regression' i 'LDA'
- **Voting Best 4**: 'Gradient Boosting', 'Logistic Regression', 'LDA' i 'Random Forest'

A continuació mostrem la Taula 10, que conté els resultats dels tres tipus de models que hem ajustat.

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Voting Best	0.837	0.775	0.795	0.76
Voting Best 3	0.837	0.774	0.796	0.759
Voting Best 4	0.838	0.784	0.791	0.777

Taula 10: Mètriques obtingudes pels tres models de 'Ensambles'

En interpretar aquests resultats veiem que els resultats dels tres models són molt bons. Tots tres tenen un molt bon porcentatge de 'Precision', però lo important és que aconsegueixen trobar un bon balanç entre 'Precision' i 'Recall'. Els tres models aconsegueixen aprofitar bé les qualitats dels models que els composen per donar un bons resutats.

4 Model guanyador

Per decidir quin és el model guanyador, a continuació mostrarem una taula comparativa dels models que han donat millors resultats de cada grup. La taula està ordenada descendment pel valor de 'F1 Macro', que és la mètrica que ens interessa. No conté els models 'Best Extra Tree' o 'QDA', per exemple, perquè només hem afegit els millors de cada grup. Per això, en el nostre cas apareix 'Best Random Forest' i 'LDA', que són els millors del grup d'arbres i de 'Discriminant Analysis' respectivament.

La Figura 11 mostra aquesta comparativa de models.

	Accuracy	F1 Macro	Precision Macro	Recall Macro
Best Voting 4	0.838	0.784	0.791	0.777
Gradient Boosting	0.838	0.777	0.795	0.763
Logreg	0.835	0.771	0.792	0.756
LDA	0.834	0.77	0.79	0.755
Best Random Forest	0.795	0.763	0.75	0.804
KNN	0.827	0.757	0.782	0.741
Combined-NB	0.801	0.755	0.744	0.771
Clustering	0.457	0.426	0.462	0.451

Figura 11: Comparació de les mètriques dels millors models de cada apartat

Com es pot observar, el millor model que hem obtingut respecte 'F1 Macro' és 'Best Voting 4', que promitja les probabilitats dels quatre millors models: 'Gradient Boosting', 'Logistic Regression', 'LDA' i 'Random Forest'. Aquest model aconsegueix millorar el rendiment respecte els quatre models que l'integren degut a les diferències entre ells. 'Logistic Regression' i 'LDA' són models lineals, mentre que 'Gradient Boosting' i 'Random Forest' no ho són, per tant el ensamble aconsegueix abarcar les diferències entre els dos tipus per millorar una mica les prestacions.

A més, 'Best Voting 4' no tarda molt més que 'Gradient Boosting', així que l'augment de temps computacional el podem assumir per obtenir millors resultats.

Un cop escollit el millor model, anem a calcular l'estimació del rendiment de generalització amb el millor model que hem obtingut sobre unes dades que no hem vist mai (dataset de test).

L'estimació de la generalization performance del model guanyador, 'Best Voting 4', és 0.78.

5 Conclusions

Al llarg d'aquest treball hem probat diferents models lineals i no lineals. En total, hem utilitzat un total de 18 models diferentes (tenint en compte els diferents casos de Naive Bayes, Ensambles i clustering), fent la cerca dels paràmetres que donaven millors resultats pel nostre problema en aquells models que era possible. Hem vist en l'anterior apartat, que el model que ha donat millors resultats és el 'Enambles' que uneix 'Gradient Boosting', 'Logistic Regression', 'LDA' i 'Random Forest', amb un resultat de 0.78 en la mètrica 'F1 Score'.

Aquest resultat no és gens dolent tenint en compte les limitacions que ens hem trobat pel camí. Limitacions com, per exemple, l'alta col·linearitat entre certes variables en tenir moltes categories úniques i realitzar One-Hot encoding o el desbalanceig de la classe target. Aquestes limitacions reafirmen la nostra deicsió d'utilitzar 'F1 Macro' com a mètrica de referència en comptes de 'Accuracy', doncs el valor de 'Accuracy' és bastant més alt que el de 'F1 Macro', i no representa bé el rendiment tenint en compte les deficiències del nostre problema.

Per aquest motiu, una de les conclusións que extreiem del treball és que els procediments de validació i cerca dels millors paràmetres escollits han sigut exitosos, ja que hem obtingut bons resultats en la majoria de models. En concret, creiem que utilitzar cross-validation ens ha ajudat a obtenir unes métriques precises en el train i a evitar el sobreajust. Gràcies a això, estem segurs que els paràmteres que han donat millors resultats estan ben escollits. També volem recalcar la importancia de regularitzar, doncs creiem que ha sigut molt important per obtenir millors resultats, com és el cas del model RDA (passant de 0.435 amb QDA a 0.76 amb RDA en la mètrica 'F1 Macro').

Per altra banda, creiem que el preprocessing ens ha ajudat bastant a no obtenir resultats dolents. En aquesta part ens hem lliurat de problemes que podien ser crítics pels models que hem ajustat i per això hem obtingut bons resultats. Com acabem de comentar, el valor més gran d'estimació de performance obtingut és prou bo tenint en compte als problemes que hem hagut de fer front i una de les causes d'això són les transformacions que hem decidit aplicar abans d'ajustar models.

Una de les conclusions científiques que extraiem és la importància de les variables categòriques respecte a les numèriques. Això és normal, doncs teniem moltes més variables categòriques que numèriques. Els models on només hem utilitzat numèriques han donat resultats molt dolents en comparació als models amb categòriques (com és el cas de Naive Bayes). No obstant, cal remarcar que les variables numèriques també tenen un pes important en el problema. Proba d'això és que hem obtingut millors resultats amb 'Combined Naive Bayes' que amb 'Categorical Naive Bayes', és a dir, tenir en compte les numèriques ens ha millorat el resultat en aquest cas.

En aquest treball hem obtingut resultats bons tant amb models lineals com amb models no lineals. Si recapitulem, abans de modelar hem comentat a partir del PCA que el nostre problema semblava no lineal, però que tenia matissos lineals si separàvem els tres grups independentment). Aquesta interpretació és coherent amb el nostre resultat, ja que el millor model que hem obtingut uneix dos models lineals ('LDA' i 'Logistic Regression') amb dos models no lineals ('Gradient Boosting' i 'Random Forest'). Per tant, com a conclusió final podem dir que la millor froma d'afrontar el nostre problema ha estat juntar forces entre diferentes models per pal·liar les dificultats del nostre dataset i la seva complicada estructura.

Les limitacions de l'estudi les hem anat esmentant durant el treball. Les més importants són el desbalanceig de la classe target i la col·linearitat entre les columnes corresponents a variables que tenen moltes categories (en fer One-Hot encoding).

Aquest treball es podria extendre fent un preprocessing més profund, amb coneixements més avançats que no estan a l'abast de l'assignatura. També podriem mirar altres models més complexos que no hem fet a classe i utilitzar tècniques pròpies d'experts en la matèria per exprimir al màxim el nostre dataset i obtenir resultats més acurats.

6 Annex

6.1 QDA

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
QDA	0.484	0.481	0.578	0.589

Taula 11: Mètriques obtingudes amb QDA

Fent QDA hem obtingut uns resultats pitjors que un classificador random (per sota del 50%), , probablment per culpa de la col·linealitat entre variables.

6.2 Support Vector Machines

D'entrada, Support Vector Machines és el model que més ha tardat en ajustar les dades i trobar els millors paràmetres. De fet, va tardar tant en executar-se que en el document de codi la seva part està comentada (no es pot executar) per si decidiu executar-ho tot. Això és important considera-ho perquè a l'hora d'escollir el millor model, no volem estar esperant molt temps cada cop que volem trobar les mètriques.

Dit això, en el cas de Support Vector Machines, també hem obtingut uns paràmetres (d'entre els de l'Annex 6.4.6) que maximitxen el rendiment. El paràmetre C = 10 indica que assumim certs errors en la classificació de les classes a canvi de no sobreajustar les dades. Un valor més gran de C hagúes sobrejustat les dades. Per altra banda, transformem les dades d'entrada amb polinomis de segon grau ('kernel' = lineal i 'param_degre' = 2). Els paràmetres 'gamma' i 'coef0' ens han donat resultats que no són importants pel nostre cas de kernel lineal.

Els resultats obtinguts en la primera i única execució (degut al seu temps d'execució) amb aquest paràmetres es troben a la Taula 12.

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Support Vector Machines	0.833	0.767	0.791	0.752

Taula 12: Mètriques obtingudes amb Support Vector Machines

Aquest model, apart de ser molt lent, ha obtingut mètriques pitjors que LDA o Logreg. Per tant, no és un model molt útil en el nostre cas, tenint en compte la seva dificicultat computacional i el seu baix rendiment.

6.3 Clustering

En ajustar els models Kmeans ens hem trobat amb els pitjors resultats En quant a Kmeans, d'entre els valors de l'Annex 6.4.8, el número de clusters òptim pel nostre problema és k = 2, doncs ens ha donat el major valor de CH Score, el valor més proper a 0 en quant a DB score i el valor més proper a 1 en Silhouette score. Això té sentit, perquè el nostre problema és un problema de classificació binari, on només tenim dues categories a predir. Aquest número de cluster ens ha donat les mètriques següents:

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
kmeans	0.457	0.426	0.462	0.451

Taula 13: Mètriques obtingudes amb kmeans

Com es pot observar, els resultats són pitjors que un predictor random, com passava amb el QDA.

L'explicació d'aquest fet l'atribuïm, un altre cop, a la col·linealitat entre les columnes del nostre dataset, que impideixen realitzar una separació neta de les observacions en diferents clusters.

6.4 Cerca dels millors paràmetres

6.4.1 Paràmetres RDA

En aquest cas només hem hagut de buscar el paràmetre de regularització λ d'entre el set [0, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.5, 0.75, 1].

6.4.2 Paràmetres K-Nearest Neighbours

Els conjunts de paràmetres que hem tingut en compte en aquest cas són els següents:

- num_veïns: [1, 2, 3, 5, 7, 10, 15, 20, 30, 40, 50]
- distancia: ['euclidean', 'minkowski', 'manhattan']

6.4.3 Paràmetres Logistic Regression

La llista de valor del paràmetre C que hem mirat en aquest cas és [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10].

6.4.4 Paràmetres Decision Tree

Els paràmetres a tenir en compte quan ajustem un Decision Tree són 'criterion' 'max_depths', 'min_samples_split', 'min_samples_leaf' i 'max_features'. Aquests paràmetres regulen aspectes com la prfunditat de l'arbre o la mida mínima per partir un grup de manera que puguem controlar el sobreajust i l'aleatorietat de les dades. Els valors dels peràmetres que utilitzarem per fer 'GridSearch' en aquest cas són els següents:

- criterion: ['gini', 'entropy']
- max_depths: [None, 5, 10, 15, 20]
- min_samples_split: [1, 2, 3, 4, 5]
- min_samples_leaf: [1, 2, 3, 4, 5]
- max_features: ['auto', 'sqrt', 'log2', None]

6.4.5 Paràmetres Random Forest i Extra Trees

Els set de valors que mirem en la cerca dels millors paràmetres són els següents:

- ntrees: [50, 100, 200]
- max_depth : [20,None]
- min_samples_split: [2,5]
- min_samples_leaf: [4]
- balance: [None, 'balanced', 'balanced _subsample']

6.4.6 Paràmetres Support Vector Machines

Els paràmetres que hem mirat per ajustar aquest model són els següents:

- C: [0.1, 1, 10, 100]

- kernel: ['linear', 'poly', 'rbf', 'sigmoid']
- degree: [2, 3, 4],
- 'gamma': ['scale', 'auto'],
- coef0: [0.0, 0.1, 0.5]

6.4.7 Paràmetres Gradient Boosting

Els sets de paràmetres que hem mirat en aquest cas són els següents:

- n_estimators: [100, 200]
- learning_rate: [0.01, 0.1, 0.2],
- max_depth: [3, 4, 5],
- min_samples_split': [2, 5, 10],
- min_samples_leaf': [1, 2, 4],
- max_features: ['auto', 'sqrt', 'log2']

6.4.8 Paràmetres Clustering

Els valors dels paràmetres que hem mirat en aquest cas són els següents:

- k (num clusters en Kmeans): [2,3,5,15,20]
- covariance_type (MIxture of Gaussians) : ['diag', 'full', 'spherical']