PRÀCTICA 2: CLASSIFICACIÓ DE DADES
Implementació de models classificadors per a prediccions binàries i multiclasse

Marc Martín Ortega (1564274)
Marina Vázquez Guerrero (1563735)
Mireia Fernández Fernández (1562636)
Aprenentatge Computacional, III MatCAD

Desembre, 2021

Índex

1 Introducció

Des de temps immemorables, als humans ens han captivat les tasques de classificació i predicció, molt probablement degut a com funcionen els nostres cervells. Ja sigui pel pur oci de fer-ho o per poder prevenir possibles desastres, sempre hi ha hagut molt d'interès en predir com es desenvoluparà el temps.

L'objectiu base d'aquesta pràctica és aquest, a partir de la base de dades *Rain in Australia*[1], poder maximitzar la confiança de les prediccions (o ser capaços de classificar) de si plourà *demà*, donades una sèrie de dades sobre el temps del dia d'*avui*. Encara que totes les dades son d'Austràlia, com hi ha molta diversitat i les característiques del clima semblen estar incloses en els atributs de la base de dades, creiem que es podria utilitzar per qualsevol lloc.

Per tal de desenvolupar aquest treball, utilitzarem diferents models predictors amb diferents tècniques per validar-los i per poder aprofitar-los al màxim.

Però, per familiaritzar-nos amb algunes d'aquestes tasques -o de prèvies-, primer realitzarem un petit treball amb la famosa base de dades *IRIS*, on provarem diferents models de predicció.

El repositori de *GitHub* referent a aquesta pràctica amb tots els codis es troba aquí.

[1]Base de dades extreta de: Kaggle. "Rain in Australia". https://www.kaggle.com/jsphyg/weather-dataset-rattle-package (últim accés 13 de Desembre, 2021).

2 Base de dades Iris

La base de dades utilitzada per a la primera part del treball és el *Iris dataset* de la biblioteca *scikit-learn* [2]. Aquesta, composta per 150 mostres de tres espècies diferents (*Iris setosa*, *Iris versicolor* i *Iris virgínica*), es basa en la informació donada pels següents quatre atributs: la llargada del sèpal, l'amplada del sèpal, la llargada del pètal i l'amplada del pètal.

El nostre objectiu, per tant, serà classificar correctament de quina espècie de flor es tracta, donada una base de dades amb característiques de les tres espècies diferents d'aquest gènere. Per a poder fer aquestes prediccions, utilitzarem diferents models els quals es presentaran més endavant, a partir de 2.2.

2.1 Visualització de les dades

Primerament, visualitzem la base de dades de les característiques:

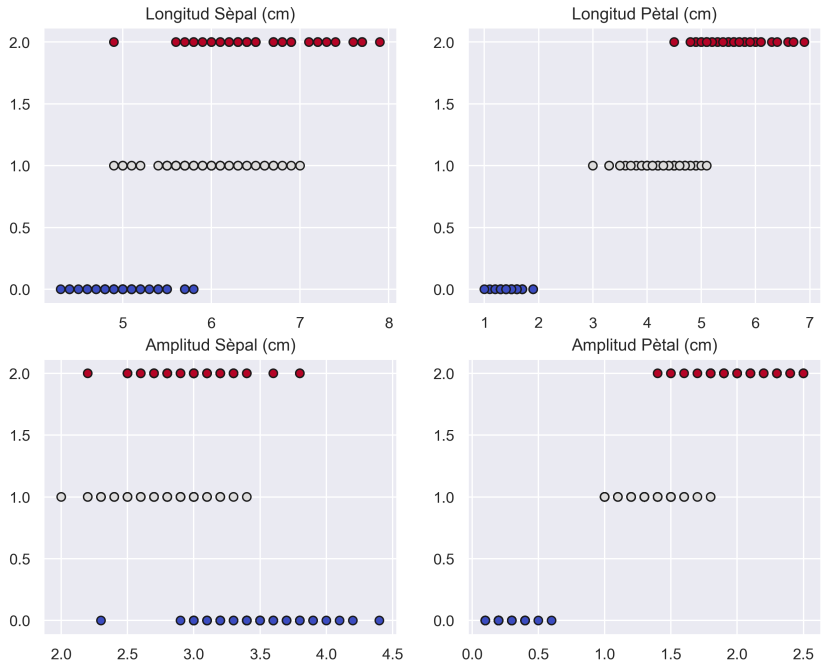


Figura 1: Representació gràfica de cada atribut segons la seva classe

[2] Base de dades extreta de: "sklearn.datasets.load_iris". https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html (últim accés 12 de Desembre, 2021).

Cada color en les figures representa a una de les tres espècies. Així, podem veure si hi ha certs atributs on la diferència de classe és més marcada que en uns altres, atès que aquests seran els més útils per a classificar. En aquest cas, observem que en l'amplitud del sèpal no sembla haver una diferència clara entre les tres espècies; mentre que, sorprenentment, la classe blava es desmarca completament de les altres dues en els atributs que defineixen el pètal de la flor.

Per senzillesa i comoditat alhora d'utilitzar i mostrar gràficament els diferents models de classificació que es presentaran en el següent punt, escollim utilitzar només dos atributs. Segons 1, els dos més indicats serien els referents al pètal, però abans cal tenir en compte la matriu de correlació per assegurar-se que les característiques escollides no siguin independents de la classe a predir.

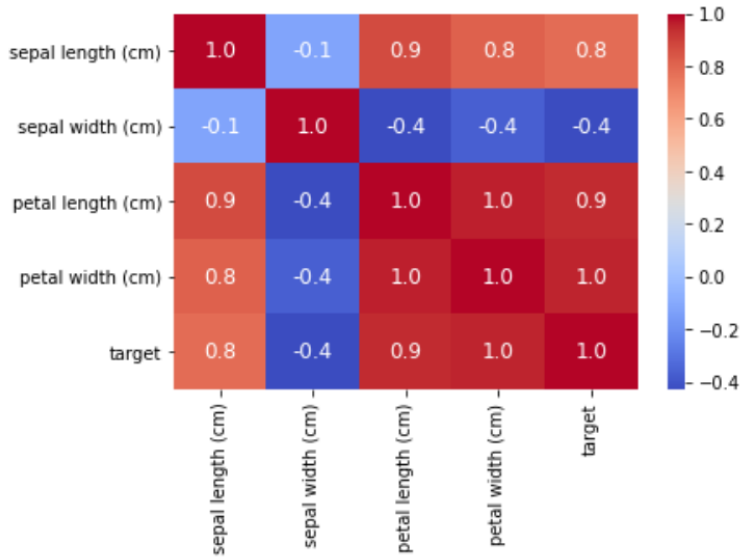


Figura 2: Matriu de correlació entre els atributs de les dades *Iris*

Podem observar a la matriu que, efectivament, la correlació entre la classe i l'amplitud del sèpal és molt menor a la correlació entre la classe i la resta d'atributs. A més, tal com ja s'ha esmentat abans, confirmem que les dues característiques referents al pètal es troben millor relacionades amb l'espècie de la flor que no pas la longitud del sèpal. Escollirem per tant aquestes dues per a fer les classificacions.

Seguidament, podem visualitzar les dades a través d'histogrames dels diferents atributs per a veure quantes mostres hi ha amb mesures semblants.

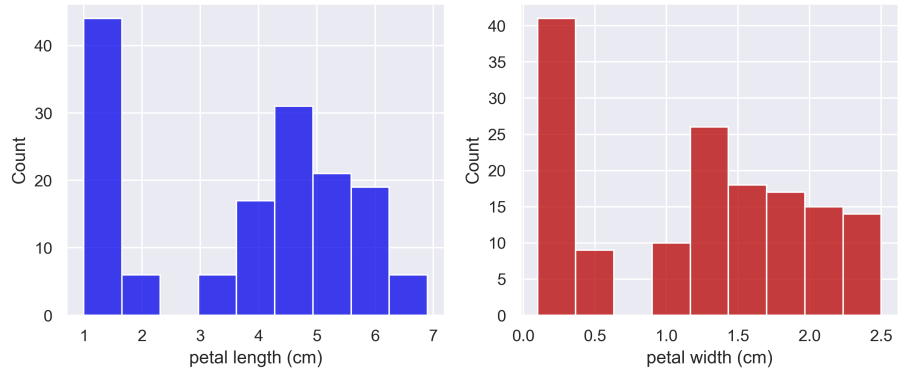


Figura 3: Histogrames dels dos atributs seleccionats per a les classificacions

D'aquesta manera, observem clarament la separació d'una de les classes que s'havia anticipat en la figura 1. Així com també es pot veure que allà on intersecten les dues altres espècies, el comptador retorna nombres majors.

Finalment, visualitzem la *target* de la base de dades, per tal de veure el balanceig de mostres de cada classe, que correspondran a les espècies *setosa*, *versicolor* i *virgínica* respectivament.

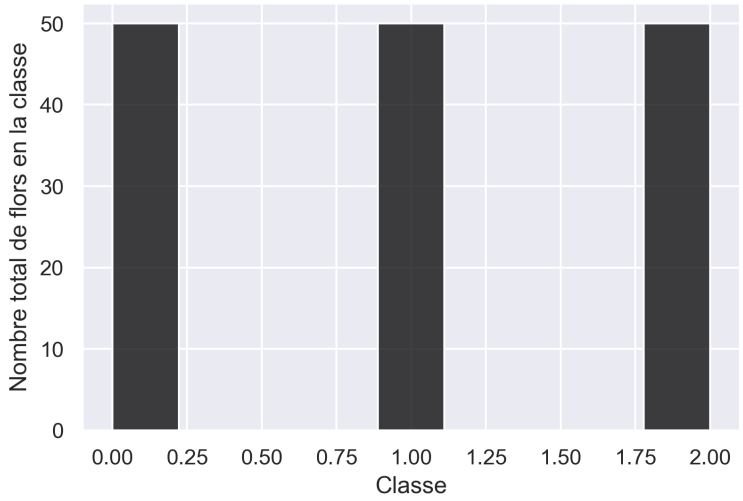


Figura 4: Histograma de les *targets* de la base de dades

Amb 7 veiem que el recompte de totes tres espècies dona la mateixa quantitat i que, per tant, la classificació serà equiprobable.

2.2 Models de classificació

A continuació utilitzarem el Regressor Logístic, el *Support Vector Machine* (SVM), el *Decision Tree*, el *Random Forest* i el *K-Nearest Neighbor* (KNN) per a fer prediccions i poder comparar els diferents models a partir dels seus resultats.

2.2.1 Regressor logístic

Aquesta és una tècnica d'aprenentatge computacional utilitzada quan es vol classificar un atribut categòric. S'anomena *regressió logística* per la funció que utilitza el mètode (funció logística).

Per a veure què tant bé funciona el model amb aquests dades, hem fet predit la classificació de unes dades *test* 10 vegades. Hem obtingut una precisió mínima de 0.933333 i, sorprenentment, una precisió màxima de 1.

Seguidament, hem calculat les corbes de *Precision-Recall* i de ROC, mostrades a 5.1. Per a aquest regressor en concret, veiem que en la primera la classe 0 és perfecta ja que la gràfica té forma totalment quadrada; en canvi, les altres dues classes no arriben a aquesta perfecció: els seus valors es queden just per sobre de 0.9. En general, però, podem veure que el rendiment del model és molt alt. Pel que fa al segon tipus de corba, les tres espècies semblen tenir un rendiment perfecte, atès que tenim l'àrea màxima i, per tant, un bon rendiment.

2.2.2 SVM

El *Suppot Vector Machine* és un conjunt d'algoritmes d'aprenentatge supervisat. La idea per a tots és la mateixa, utilitzar hiperplans per a separar les classes i anomentant les mostres més properes de les dues classes com a vectors de suport. Sempre tenint en compte d'evitar generar *overfitting*.

Si usem un *kernel* de tipus *Radial Basis Function*, obtenim precisions entre 0.933333 i 0.966667, els quals que ja són millors resultats que la *Regressió logística*. Amb els gràfics de les corbes de precisió i ROC, mostrades en 5.2. Tots tres resultats són molt bons. Del segon tipus de corbes, obtenim la perfecció en totes tres classes.

En general, en fer proves amb els diferents nuclis, veiem que el millor és el lineal. Els nucli polinomial i circular (*rbf*) mostren resultats lleugerament més petits, tot i ser superiors a 0.9. En canvi, el sigmoide funciona significativament pitjor, mostrant resultats inferiors al llindar del 0.5.

2.2.3 Arbre de decisió

Aquest algorisme és un model utilitzat per a la classificació que es basa en l'analogia d'un arbre amb branques que es van separant, fins arribar a les fulles, i en aquestes fulles es troba el valor que es prediu per a cert valor.

Les precisions obtingudes aquí també són excel·lents, això es deu al fet que hem utilitzat els dos atributs més correlacionats. Hem obtingut resultats entre 0.916667 i 0.983333. Respecte a les corbes mostrades a 5.3, hem obtingut altre cop àrees entre 0.9 i 1, per a ambdós casos.

2.2.4 Random Forest

El model de *Random Forest* utilitza un cert número d'arbres predictors de manera que la solució que dona és la classe que més arbres ha decidit.

En el nostre cas, hem fet que entri com a paràmetre un nombre per a que el nombre d'arbres que tingui el bosc sigui de 3 pel paràmetre. Aquesta decisió no té un argument gran sinó que ha estat un número que ens ha semblat bo a l'hora de fer el *boxplot* que en el punt 2.2.6 és veu.

Els resultats obtinguts segueixen la mateixa línia que els anteriors, estan a prop de la perfecció. En totes 5 proves, hem utilitzat 45 arbres i la precisió ha estat entre 0.9 i 0.983333. Parlant de les corbes, veiem en 5.4 que la classe 0 té 1, la classe 1 en té 0.98 i la 2, 0.96. En canvi, la corba ROC, dona resultats perfectes en totes tres classes.

2.2.5 KNN

El *K-Nearest Neighbor* és un model molt conegut i utilitzat en l'àmbit dels models de classificació supervisada. Aquí l'objectiu és classificar una etiqueta a partir de les etiquetes dels k veïns més propers.

Utilitzant 7 veïns, obtenim precisions d'entre 0.933333 i 1. Alhora, les corbes 5.5 també donen suport a l'excel·lencia d'aquests resultats.

Utilitzant altres ks, veiem que obtenim precisions diferents, tot i que amb diferents proves veiem que el resultat va oscil·lant i no sembla que la diferencia sigui suficientment significativa. A tall d'exemple: amb 10 veïns, hem obtingut una precisió de 0.933333; amb 15, obtenim una precisió de 0.950000; i, amb 20, la precisió augmentava a 0.983333. Encara que s'intueix una tendència de millora en tenir més veïns, la diferència és poc notable com per a ser concluent.

2.2.6 Comparativa de models

Com hem pogut observar, tots els models funcionen gairebé a la perfecció. Hem de destacar que com ja hem vist anteriorment, tant la longitud con l'amplitud del pètal tenen molt bona correlació amb la variable a classificar. En el següent *Boxplot*, es pot veure més visualment els resultats obtinguts:

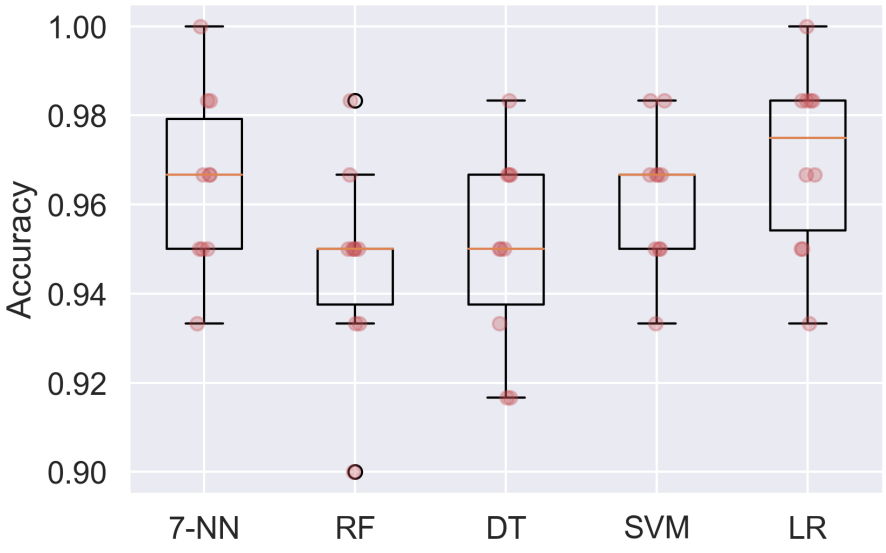


Figura 5: *Boxplot* de la comparació dels models a partir de les seves precisions.

Les mitjanes ens informen que tots cinc models presenten precisions d'entre 0.95 i 0.97 aproximadament. La precisió més baixa l'ha presentat l'algorisme *Random Forest* i la més alta, el Regressor logístic. En l'apartat 5.6 podem veure les diferents proves amb els diferents models en forma de taula, així com els temps d'execució.

3 Base de dades *Rain in Australia*

La segona base de dades utilitzada és *Rain in Australia*. Està composta per 145460 mostres, cadascuna amb 23 atributs. Aquests descriuen el panorama meteorològic donades una localització i una data. Utilitza dades com els màxims i mínims de les temperatures, la direcció del vent a diferents hores, la pressió, la humitat, etc. Tot amb objectiu de predir si l'endemà plourà o no (predir la variable *RainTomorrow*).

3.1 Anàlisi exploratori de les dades

La nostra base de dades ha hagut de passar per un procés de neteja i, en conseqüència, ha anat canviant de forma fins a acabar amb 78 atributs. Aquest canvi en la dimensionalitat de les dades ve donat per les addicions necessàries que han sorgit en tractar les característiques temporals i categòriques.

Comencem tenint atributs de tipus totalment diferents i que no els podem fer servir a l'hora de fer les prediccions. Com és el cas de les dates que són atributs temporals, categòrics com el *WindDir9am*, *WindGustDir* i *WindDir3pm* i binaris com és el *RainToday* i *RainTomorrow*. La resta dels atributs són de tipus numèrics i no comporten cap problema a l'hora de utilitzar els models.

Tot seguit, encara que sol ser útil visualitzar les dades, en aquest cas no ho podem fer tenint tants atributs. Per tant, hem escollit mostrar aquells que siguin més rellevants, guiant-nos per la correlació que tenen amb la nostra variable objectiu. Això ho calculem directament, sense mostrar la matriu de correlació, atès que el gràfic amb 78 característiques no resulta gaire entenedor.

Insòlitament, els valors obtinguts mostren correlacions molt baixes, la majoria inclús despreciables, però d'entre els pocs amb valors rellevants, hem escollit la quantitat de llum solar (*Sunshine*) i la humitat a les tres de la tarda (*Humidity3pm*), encara que les correlacions es trobin just per sota del 0.5 i el -0.5, respectivament.

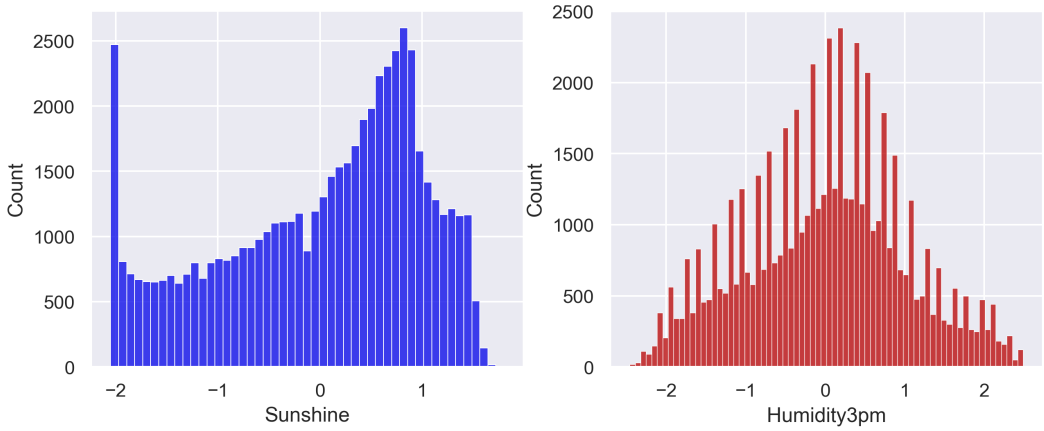


Figura 6: Histograma dels atributs més correlacionats amb *RainTomorrow*

Alhora hem fet un histograma mostrant el recompte de vegades que plou i que no plou per a observar si les dades estan balancejades o no.

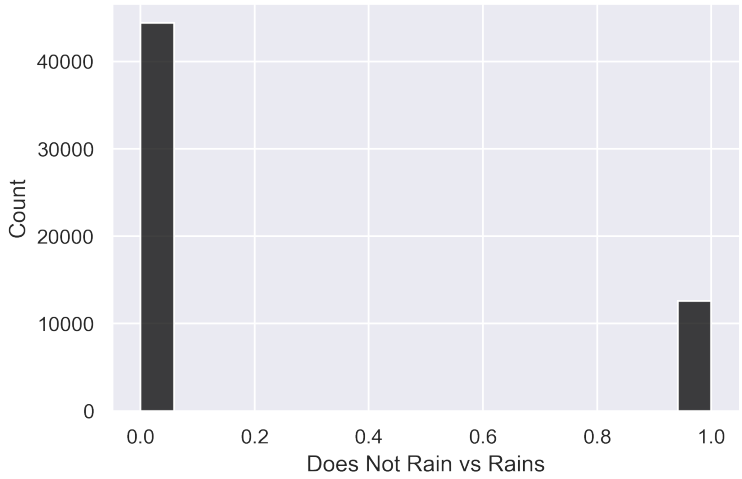


Figura 7: *Boxplot* de la comparació dels models a partir de les seves precisions.

És perfectament observable que les dades no estan balancejades. Concretament, hi ha 12606 *sí plou demà* i 44427 *no plou demà*, per tant, 3.5 vegades dies solejats que dies pluvosos. Això farà que els resultats de les prediccions siguin més pobres, especialment per a la classe minoritària, afectant negativament, encara que com que tenim bastantes dades d'ambdues classes, esperem que s'alleujari una mica aquest efecte.

3.2 Preprocessament de les dades

La base de dades escollida necessita una neteja de variables nul·les i transformacions diverses abans de ser-nos útil, ja que de primeres no la podem utilitzar per en els classificadors. Des de treballar en els atributs no numèrics fins a la seva estandardització.

Hi havia una quantitat preocupant de variables nul·les (*NaN*) en les característiques *RainTomorrow*, *Sunshine Cloud3pm*, *WindGustDir*, *Evaporation* i *WindDir9am*. En aquests casos, hem decidit eliminar les files que les contenen. Tenim una base de dades amb moltíssimes mostres i, eliminar una part, no suposa un problema. Pel que fa als dels altres atributs, els hem reomplert fent la mitjana.

Com ja hem comentat anteriorment, veiem que hi ha dades que no són numèriques, que hem de canviar, alhora que estandarditzem per a uns millors resultats. Per a aconseguir-ho, hem utilitzat les següents biblioteques de codi: *get dummies* de *pandas* per a *date*, *WindGustDir*, *WindDir9am* i *WindDir3pm*; i, *Label Encoder*, per a *RainToday* i *RainTomorrow*, atès que només necessitem que els *Sí*s i els *No*s passin a ser 0s i 1s.

En acabar, tenim 77 atributs diferents, ja que una gran quantitat d'atributs s'utilitzen per a explicar les direccions del vent, o els mesos de l'any. Per aquest motiu, seria faborable aplicar un *PCA* per a mirar si es possible reduir les dimensions de les dades sense molta pèrdua.

Si fem un *PCA* sense tenir en conta la variança retinguda i aplicant un regressor logístic, el resultat és una reducció de dimensió 0 i un R^2 de 0.211789. Això ens diu que no anem per bon camí i per tant intentarem buscar retenir la variancia de les dades originals fins a cert punt. En canvi, si utilitzem un model amb 99% de variança retinguda, obtenim que la millor reducció és la dimensió 47 amb un R^2 de 0.198847. I sent una mica menys estrictes, demanant una variança retinguda del 95%, obtenim que la reducció es limita a la dimensió 65, amb un R^2 de 0.202666.

Concluim, per tant, que utilitzar un analisis *PCA* per a reduir l'espai dimensional de la base de dades no és gaire útil en aquest cas. Consegüentment, procedirem a utilitzar tots els atributs en els models que es presentaran a continuació, excepte si s'utilitzen menys explícitament.

3.3 Models de classificació

Un cop hem obtingut una base de dades neta i preparada per a poder utilitzar, el següent pas és buscar aquell model que aconseguirà predir la classe de les nostres dades amb millors resultats. En el nostre cas buscarem en aquells que permetin fer una predicció binària, centrant-nos en la classificació fent servir el mètode de màquines de vectors de suport.

3.3.1 Precisió dels classificadors

Un primer enfocament, per a aconseguir una idea general de com es comporten les dades en els models és fer ús de diferents classificadors. A la següent taula mostrem els algorismes implementats i les *accuracy_score* retornades:

	Accuracy socre	**Temps d'execució (s)**
Regressor logístic	0.855308	0.490373
SVM (lineal)	0.829009	69.959064
KNN (7 veïns)	0.844262	193.476523
Random Forest (45 arbres)	0.853774	3.133465

Observem que els quatre models implementats obtenen precisions similars, per sobre del 80%, però amb temps d'execució molt variats. Curiosament, aquells amb menor *accuracy_score* han trigat magnituds senceres més de temps que els de les millors puntuacions. Això i tot, a continuació entrarem en detall en els diferents *kernels* de l'algorisme SVM per tal de buscar aquell més precís, i veure si supera els resultats obtinguts fins al moment.

3.3.2 Diferenciació de *kernels* en els SVM

Atès que l'algorisme SVM per a moltes mostres ha presentat grans temps d'execució, hem decidit fer proves utilitzant com a atributs per a la classificació de les classes els dos i, després, els tres més correlacionats amb la variable objectiu: *Sunshine*, *Humidity3pm* i *Cloud3pm*. En la taula següent es mostren els resultats obtinguts per als diferents tipus de *kernel*, a excepció del *precomputed* que només és possible usar quan la matriu d'atributs és quadrada.

	2 atributs		**3 atributs**	
	Accuracy socre	**T (s)**	**Accuracy socre**	**T (s)**
Lineal	0.829009	69.959064	0.831901	70.501107
Poly	0.828789	1770.248152	-	+7200.0
RBF	0.833041	390.444688	0.832252	506.565012
Sigmoid	0.683659	74.902673	0.683352	146.250606

Per una banda, observem que les precisions no varien gaire entre l'ús de diferents atributs: o bé s'han obtingut bons resultats en ambdós casos o bé s'ha aconseguit només resultats febles, com és el cas del *sigmoid kernel*. Només hi ha una variació, i és en el cas del *kernel* polinomial, ja que per a tres atributs ha estat executant més de dues hores sense retornar cap resultat. Per tant, hem decidit que és dispensable perquè hauria de ser extremadament perfecte com per a compensar tot el temps de còmput necessari.

Per l'altra, és interessant observar la diferència de temps entre els diferents algorismes, ja que és allò que més crida l'atenció. A tall d'exemple, el *kernel* lineal és unes 25 vegades més ràpid que el *kernel* polinomial. Tal és el fet, que només dos dels quatre es troben a la mateixa magnitud de temps, la més petita de totes.

Podem concloure que el *kernel* més precís és el *rbf*, mentre que el més ràpid és el lineal. Consegüentment, tenint en ment que el percentatge de millora de precisió del primer respecte al segon és només del 0,48%, per a una primera aproximació general coronem al SVM lineal com al *kernel* més eficient, mentre que si volem obtenir el millor resultat sense considerar els temps de còmput, llavors el més adient serà el *rbf*.

3.3.3 Optimitzacions usant *ensemble methods*

Un cop implementats els algorismes simples, així, ara voldrem intentar aconseguir millors resultats utilitzant tècniques més avançades com ho són els mètodes *ensemble*. D'aquests hi ha principalment dos tipus, un dels quals ja ha estat implementat: de la variant *bagging*, el algorisme *random forest*. Per tant, podem dir ja que aquest si funciona, veient que es troba entre els millors resultats.

De qualsevol manera, per a observar el seu funcionament hem implementat algorisme per a 500, 5000 i 10000 arbres. Això ens ha retornat precisions de 0.863242, 0.864338 i 0.860042, respectivament. L'increment en el temps d'execució ha estat quelcom lineal, la qual cosa és important perquè ha facilitat la possibilitat de fer proves ràpidament.

Del mateix estil, hi ha un altra tipus d'implementació que hem utilitzat: de la biblioteca de funcions *sklearn.ensemble*, el *BaggingClassifier*. Aquest ens permet aplicar computacionalment la tècnica *bagging* a qualsevol classificador dels que hem vist. Amb 10 estimadors, el KNN de la versió *bagging* ha arribat a una precisió de 0.843736 i el SVM lineal a una de 0.858902. Ambdós, aconseguint resultats més alts que els vists fins al moment.

Pel que fa a la tècnica *boosting* i els mètodes que apliquen, hem utilitzat l'algorisme *AdaBoost* que ens permet utilitzar els arbres de decisió, dels quals hem parlat anteriorment, però utilitzant pesos auxiliars que aconsegueixen millorar les prediccions. En implementar-ho, si usem 100 arbres aconseguim una precisió de 0.846914 i, utilitzant 500, 0.847071; i, tot, en menys de quatre minuts d'execució.

3.4 Validació creuada

La validació creuada és una tècnica que avalua un model en subconjunts de les dades entrades, o en conjunts reordenats. S'utilitza per detectar overfitting i per garantir la independència, per la qual cosa és important cross-validar sempre les dades. A més, ens permet tenir una idea de com de bé funcionarà el nostre model amb dades que encara no ha vist mai.

En el nostre cas, hem cross-validat les dades pels 5 models utilitzats. Podem observar en 8, tot i comparant-lo amb 5.7, que la variància és més alta per tots els models. Pensem que aquest fenomen passa per que al ser les dades una sèrie temporal i estar relacionades amb el clima, tenint en compte l'evolució d'aquest en els darrers anys, pot generar molta variància entrenar amb diferents trossos de la base de dades referint a aquests canvis del clima, cosa que no passa al boxplot anterior ja que entrenem amb el mateix tros.

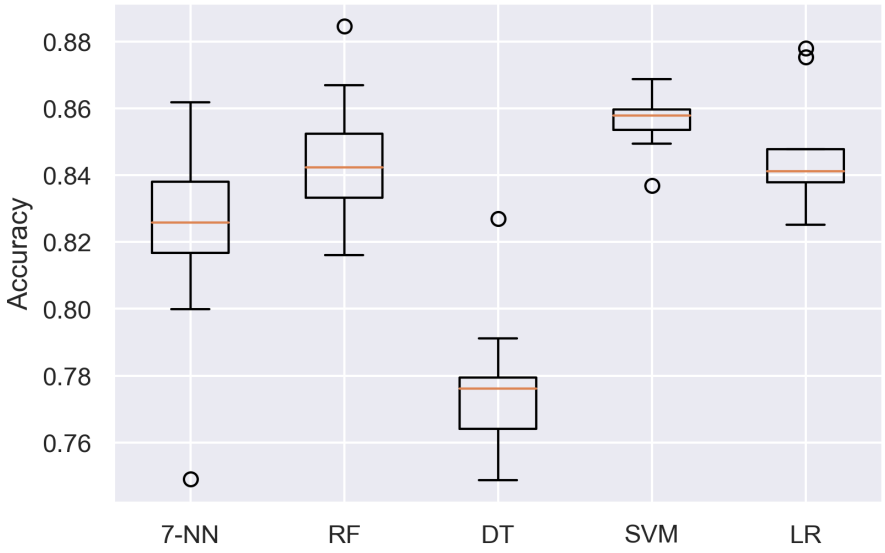


Figura 8: *Boxplot* de la comparació dels models *cross-validats* amb $k = 10$, a partir de les seves precisions.

En el nostre cas, hem realitzat la validació creuada per K-fold amb $k = 10$, tot i utilitzant el mètode *cross_val_score* ja implementat a sklearn, que s'encarrega de separar les dades, entrenar-les, predir, i calcular l'*score*. Hem escollit aquest valor de k per arbitrarietat, en general quasi no ens varien els resultats si la canviem.

Finalment comentar que per una base de dades amb aproximadament 57000 files, com és el cas de la nostra, no té molt sentit implementar una validació creuada amb *LeaveOneOut* ja que hauria de fer 57000 vegades tot l'entrenament, cosa que tardaria un temps extraordinari.

3.5 Mètriques

Quan parlem de mètriques per a avaluar classificadors, el més comú és mirar els resultats retornants per la puntuació de precisió (*accuracy_score*), però quan les dades estan no estan balancejades llavors una bona precisió pot no significar que un model sigui adient.

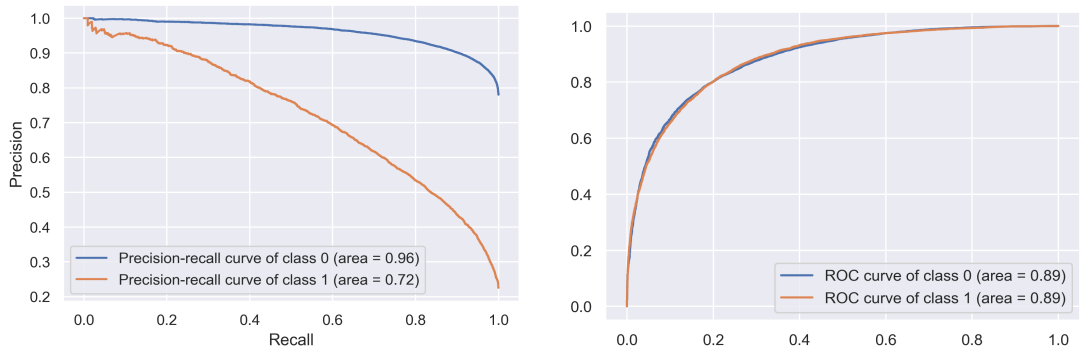
En el nostre cas, podriem usar dues mètriques alternatives a la utilitzada fins ara: la *f1_score* i l'*average_precision_score*. Aquesta última només funciona en els casos de predicció binària, com el nostre. De qualsevol manera, ens centrarem en la

f1_score perquè ens permetrà trobar aquells models on tant la precisió com el *recall* són bons, ajudant-nos també dels resultats obtinguts fins ara per a comparar.

Utilitzem doncs la funció *classification_report*, que retorna una taula amb informació sobre diferents mètriques donades unes prediccions, per a entendre com es comporta la puntuació *f1_score* per al nostre model, si utilitzem un regressor logístic.

	precision	**recall**	**f1-score**	**support**
Yes	0.88	0.95	0.91	17711
No	0.74	0.54	0.62	5103

Observem que, en aquesta taula simplificada, per als dies que sí plou, la puntuació *f1_score* és molt alta, per tant, significa que ho classifiquem bé i, alhora, no classifiquem malament els negatius. En canvi, per a l'altra classe (no plou), veiem que el *recall* dona baix i, en conseqüència, el *f1_score* també, ja que prediu bé quan no plourà però diu que no plourà per a molts dies que si ho farà (falsos positius).



(a) Corba Precisió-Recall (b) Corba ROC

Figura 9: Corbes per al regressor logístic presentat a 3.3.

A la figura anterior, en (a), correspon als resultats presentats en la taula. Per a la classe 1, amb una *f1_score* molt baixa, la corba és quasi una línia recta. En canvi, per a l'altra, l'àrea és molt major. Altre cop, aquesta diferència tan gran entre els resultats per a les diferents classes té a veure amb el no balanceig de les dades.

Oposadament, en (b) observem que les dues classes presenten el mateix resultat perquè, com ja s'ha dit abans, la sensibilitat del model respecte a les classes és molt similar per ambdues.

3.6 Búsqueda d'hiperparàmetres

Els models que hem seleccionat per a trobar els seus millors hiperparàmetres han estat el regressor logístic i el SVM amb *kernel* lineal, ja que son els més importants. Com només buscàvem un hiperparàmetre per a cadascun, hem utilitzat el mètode de cerca *GridSearchCV*, que ja ens permet aprofundir suficientment en els paràmetres. En cas que tinguéssim més, hagués pogut ser útil el mètode *RandomizedSearchCV*. De tota manera, pel nostre problema en un temps igual hagués donat millor resultat el Grid Search, atès que només tenim un paràmetre.

A més, cal aclarir que només hem buscat per al SVM amb *kernel* lineal perquè d'entrada, hem hagut de reduir l'espai mostral per tal de poder fer la cerca, ja que sinó trigava temps desmesurats i, a més, en les proves anteriors no observem quasi cap millora als models amb altres *kernels*.

Els resultats de les nostres cerques són els següents, amb $C = 0.838$ pel regressor logístic, i amb $C = 0.01$ pel SVM:

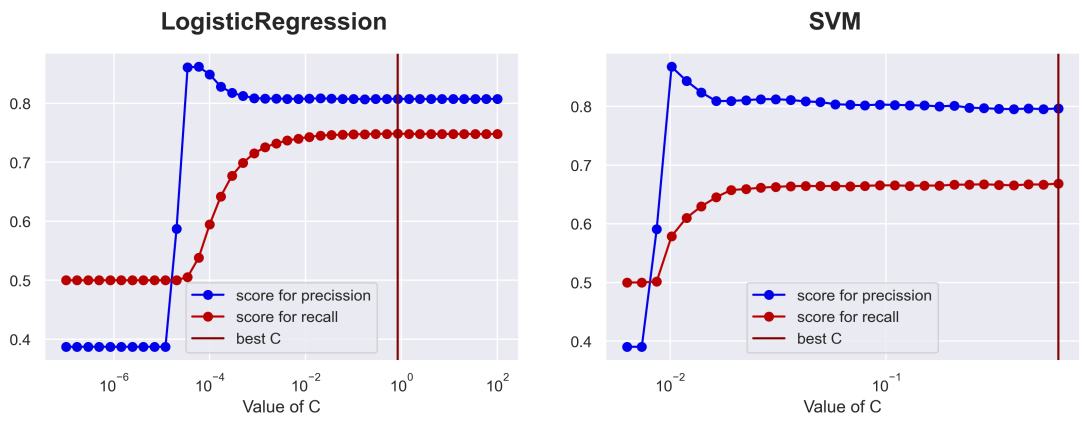


Figura 10: Puntuacions obtingudes de la precisió i el recall, per a ambdós models.

4 Conclusions

La base de dades amb la qual es va iniciar la pràctica portava amb si un munt de treball de preprocessament, on vam haver de prendre les primeres decisions que marcarien el recorregut de tot el projecte. En acabar, ens vam trobar amb unes dades poc correlacionades entre sí que, a més, estaven molt poc balancejades.

Arribats a aquest punt sabíem que no podríem arribar fàcilment a un model de regressió perfecte, però s'han utilitzat tota una sèrie de tècniques per tal de aconseguir un classificador potent.

Els algorismes utilitzats per a modelar les dades com a primer apropament a la solució final, van portar-nos cap a tres possibles camins: utilitzar un regressor logístic, un SVM o una de les tècniques de bagging, coneguda com a *Random forest*.

Malhauradament, en tenir tanta informació, el SVM necessitava un temps de còmput molt elevat; per això, en veure que els temps per als diferents *kernels* només feien que augmentar, sense millorar significativament la precisió, la única via lliure per a aquest algorisme, centrant-nos en la versió lineal, va ser buscar un hiperparàmetre el suficientment bo com per a alleugerar els càlculs i accelerar l'execució. Els resultats obtinguts mostrats a 10, però, deixen veure que el *recall* d'aquest model és el suficientment baix com per a poder descartar-lo quasi inmediatament.

Pel que fa a la tècnica dels boscs aleatoris, els resultats explicats en 3.3.3 reflecteixen com no es pot anar més enllà de cert valor en la puntuació de la precisió per molt que canviem el nombre d'arbres utilitzats. Això i tot, aquesta tècnica ha estat qui ha obtingut millors resultats, tant per sobre dels mètodes simples com de tots els altres *ensemble methods* implementats.

Finalment, per al regressor logístic, vam decidir que seria aquest amb qui comprovaríem si cercant el hiperparàmetre òptim aconseguíem un classificador amb un recall millor als obtinguts al moment. En la taula de 3.5, es veu com un regressor logístic sense cap canvi no obté un recall bo per a les prediccions de no pluja però obtenia resultats excel·lents per a l'altra classe. Això ho vam aprofundir en el apartat 3.6, on vam adonar-nos que aquest resultat venia de l'estabilització de les mètriques per al model i, conseqüentment, no podríem millorar-ho sense tornar a l'inici i recollir les dades, o preprocessar-les, d'una altra manera.

5 Appendix

5.1 Corbes del Regresor Logístic

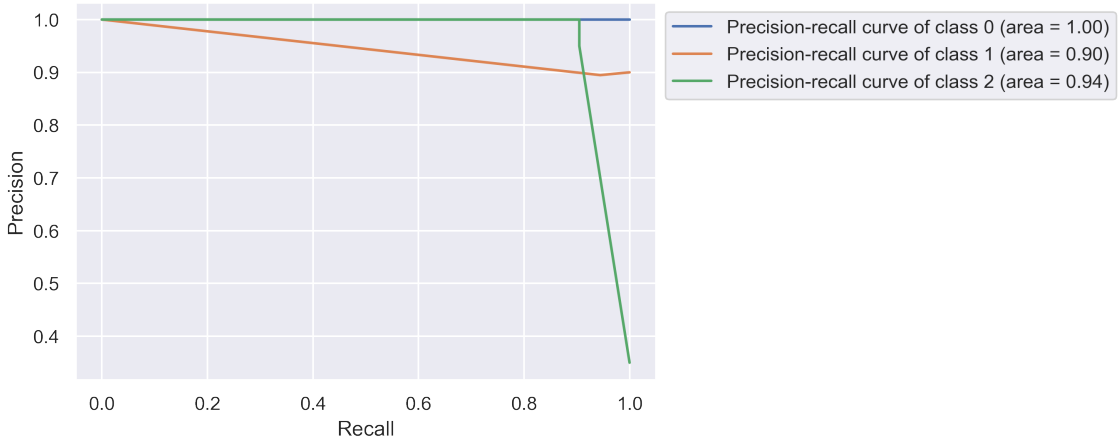


Figura 11: Corba de precisió del Regressor logístic de la base de dades Iris.

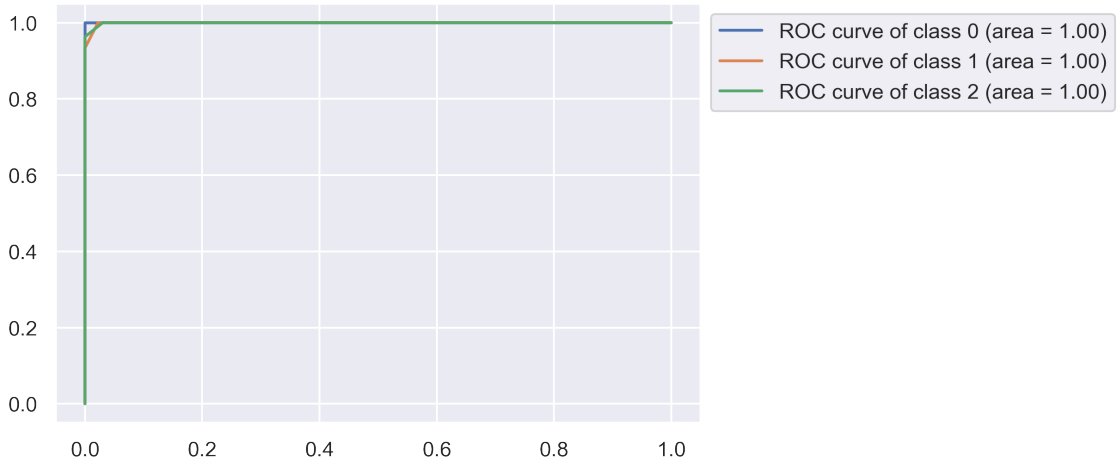


Figura 12: Corba ROC del Regressor logístic de la base de dades Iris.

5.2 Corbes del SVM

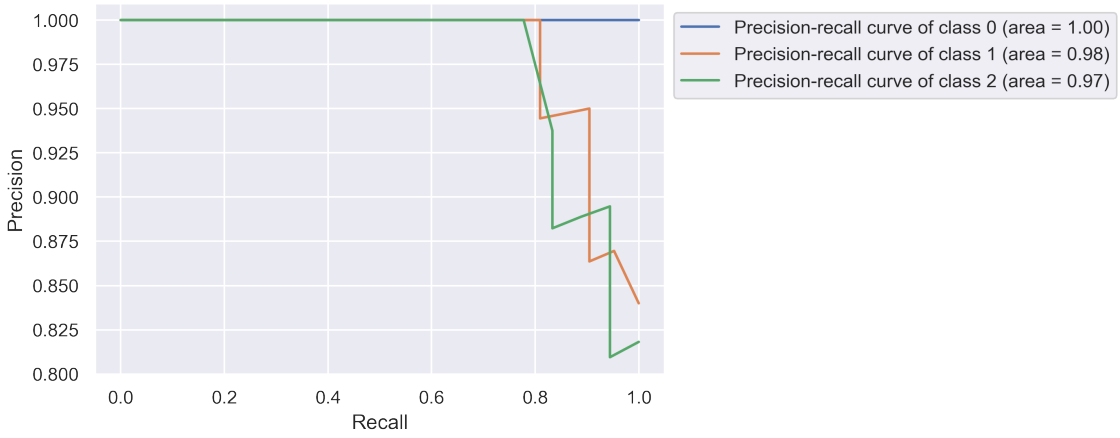


Figura 13: Corba de precisió del SVM de la base de dades Iris.

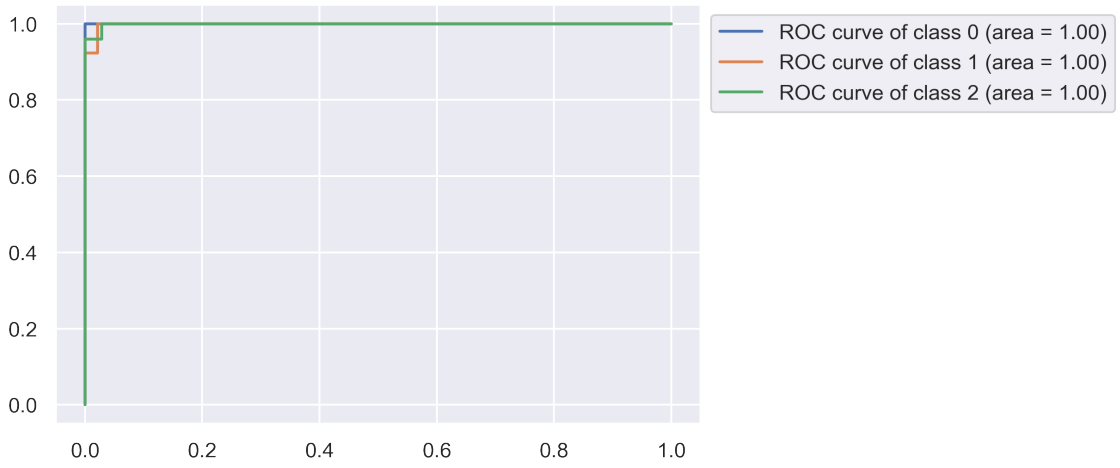


Figura 14: Corba ROC del SVM de la base de dades Iris.

5.3 Corbes del Decision Tree

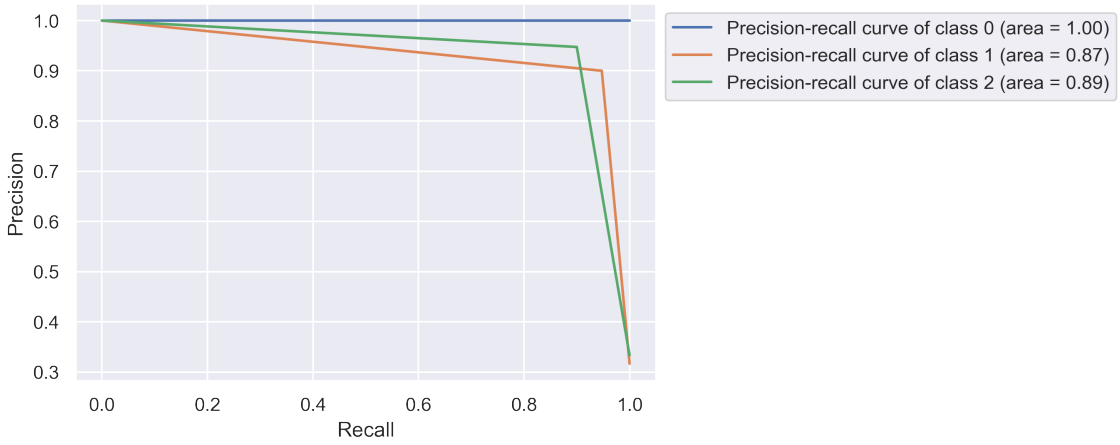


Figura 15: Corba de precisió del Decision Tree de la base de dades Iris.

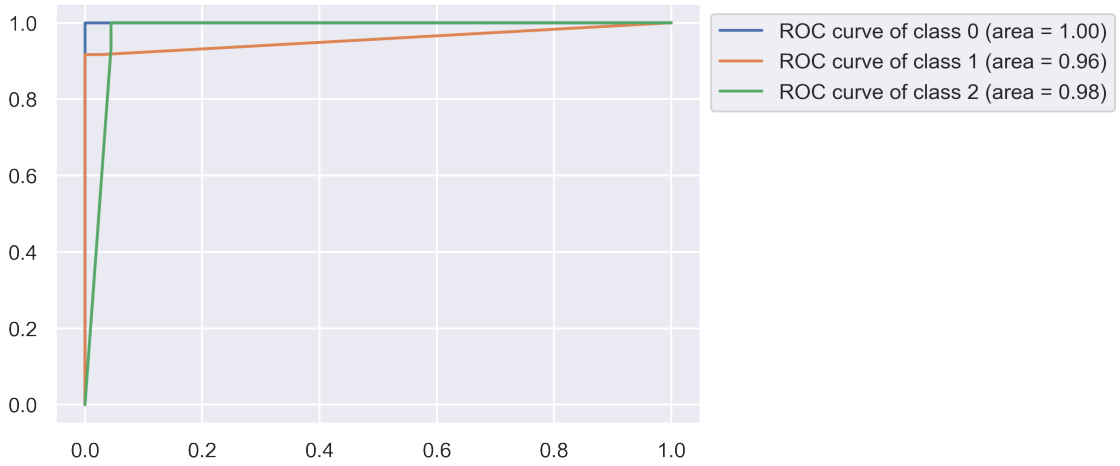


Figura 16: Corba ROC del Decision Tree de la base de dades Iris.

5.4 Corbes del Random Forest

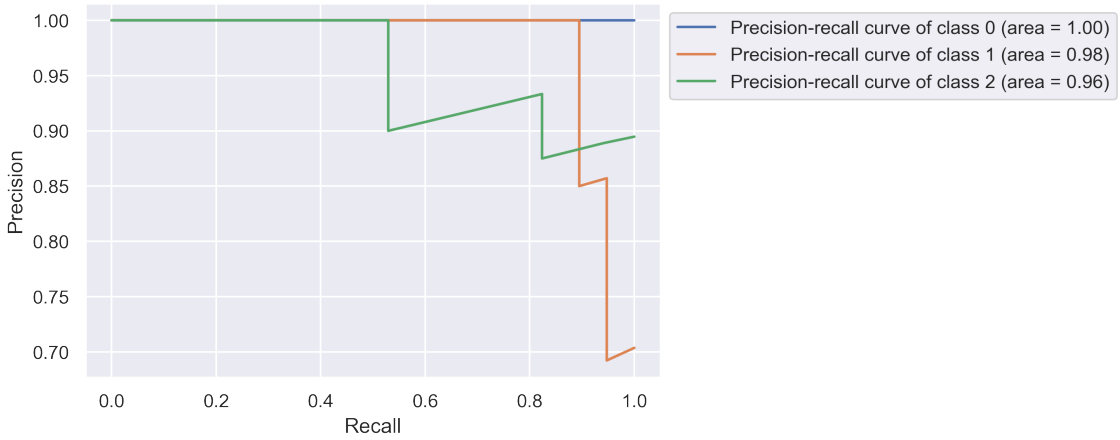


Figura 17: Corba de precisió del Random Forest de la base de dades Iris.

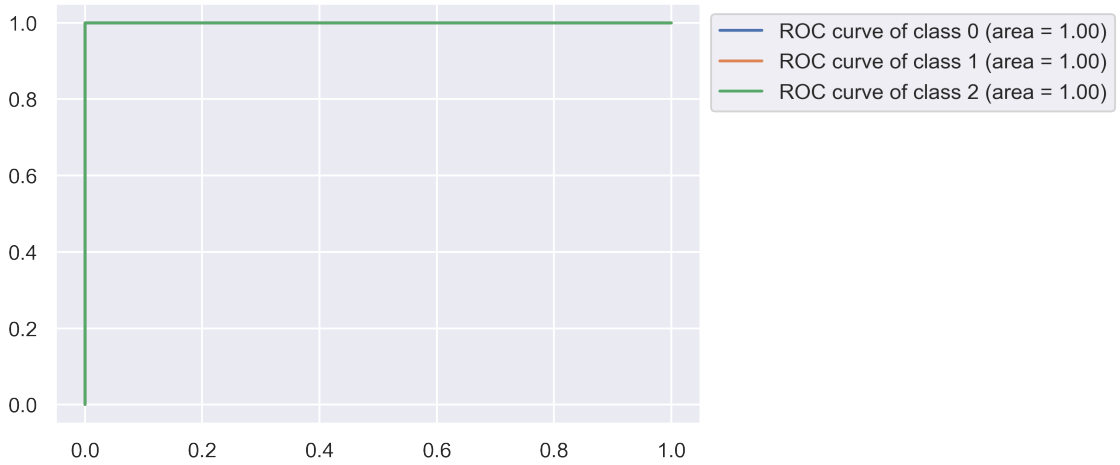


Figura 18: Corba ROC del Random Forest de la base de dades Iris.

5.5 Corbes del KNN

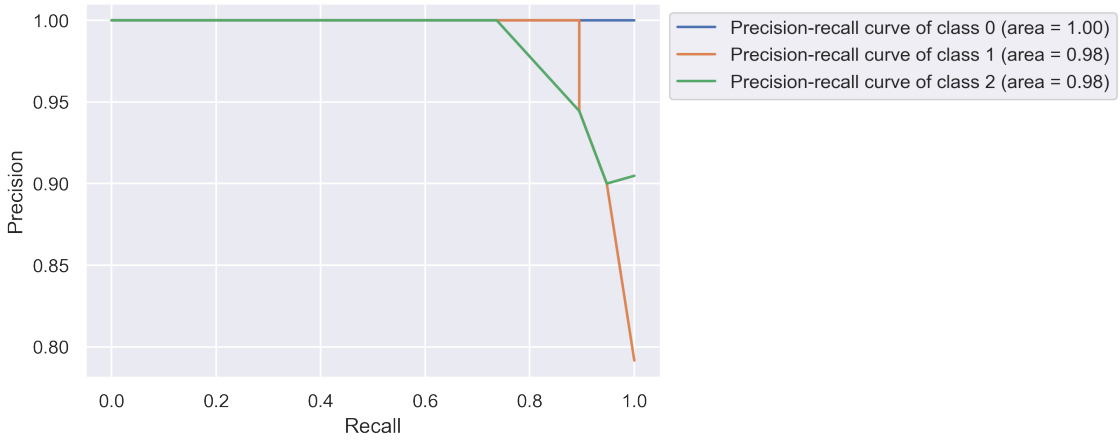


Figura 19: Corba de precisió del KNN de la base de dades Iris.

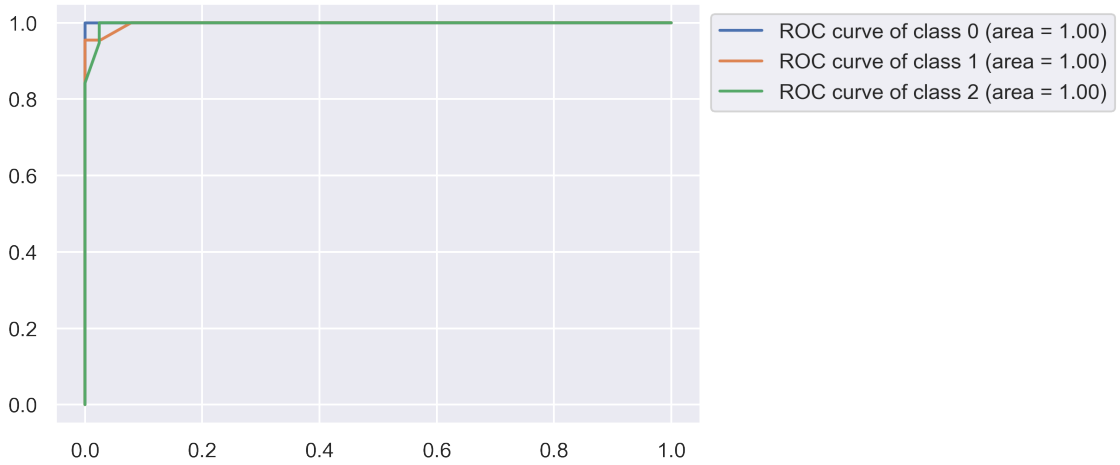


Figura 20: Corba ROC del KNN de la base de dades Iris.

5.6 Taules de resultats segons els models de 2.2

- Proves de precisió

KNN	Random Forest	Decision Tree	SVM	RL
0.950000	0.933333	0.950000	0.983333	0.95000
0.933333	0.966667	0.966667	0.966667	1.000000
0.983333	0.950000	0.950000	0.950000	0.983333
0.966667	0.933333	0.983333	0.950000	0.950000
1.000000	0.950000	0.950000	0.966667	0.966667
0.966667	0.950000	0.966667	0.966667	0.983333
0.966667	0.950000	0.916667	0.950000	0.983333
0.950000	0.950000	0.916667	0.983333	0.933333
0.950000	0.900000	0.933333	0.966667	0.983333
0.983333	0.983333	0.966667	0.933333	0.9666667

- Temps d'execució

KNN	Random Forest	Descision Tree	SVM	RL
0.015787	0.155543	0.004127	0.010959	0.030092

5.7 Boxplot 10 iteracions

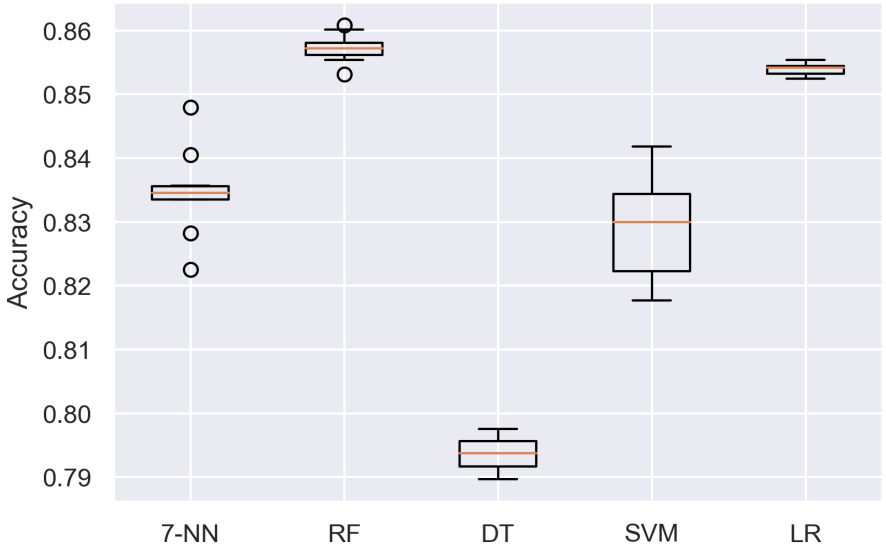


Figura 21: Boxplot amb les dades d'Australia de 10 iteracions per cada model.