Universitat Autònoma de Barcelona
Facultat de Ciències

Pràctica 2

Autors:

Andrea González & Gerard Lahuerta & Ona Sánchez

1603921 — 1601350 — 1601181

4 de Novembre del 2022

Índex

1 Introducció

L'objectiu d'aquesta pràctica és, mitjançant la interfície proporcionada per Jupyter Notebook, estudiar i classificar un valor en funció d'un conjunt de paràmetres que es calcularan mitjançant un conjunt de dades.
Les dades han sigut proporcionades per la web de Kaggle, concretament, la base de dades de telèfons mòbils.

El propòsit de la present pràctica és trobar models que descriguin les dades i permetin generar noves conclusions. Així doncs, després d'un estudi de les dades s'ha decidit intentar classificar el *price_range* en funció de les altres variables.

El dataset que s'utilitza es pot trobar al següent enllaç:
https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification.

Als comentaris del dataset s'esmenta que aquest ha estat generat aleatòriament, pel que part de la informació que hi conté pot ser incongruent amb d'altres.

A més, el fitxer amb les dades de test no conté els valors reals per a poder testejar el model amb aquestes dades, fet que obliga a utilitzar només una part del train dataset per a entrenar el model en qüestió.

Tot i així, s'ha procurat millorar la precisió tan com és possible i trobar una bona relació precisió-temps d'execució per així poder obtenir resultats de forma eficient.

2 Presentació de les funcions

2.1 Llibreries i importacions

Per tal de poder dur a terme aquesta tasca és imprescindible tenir intal·lades les següents llibreries, ja que s'utilitzen les funcions següents (d'entre altres).

Llibreria	**Funció utilitzada**
sklearn.datasets	make_regression
pandas (as pd)	read_csv DataFrame
matplotlib pyplot (as plt)	figure plot hist scatter
seaborn (as sns)	heatmap
sklearn.linear_model	LogisticRegression LogisticRegressionCV
sklearn.tree	DecisionTreeClassifier
sklearn.metrics	confusion_matrix ConfusionMatrixDisplay precision_recall_curve average_precision_score roc_curve auc precision_score
sklearn.model_selection	cross_val_score cross_validate train_test_split
sklearn.ensemble	RandomForestClassifier AdaBoostClassifier
sklearn.neighbors	KNeighborsClassifier
sklearn.pipeline	make_pipeline
sklearn.preprocessing	StandardScaler
sklearn.svm	SVC LinearSVC
sklearn.cluster	KMeans
sklearn.mixture	GaussianMixture
warnings	filterwarnings
numpy (as np)	meshgrid concatenate

Taula 1: Llibreries i funcions utilitzades

2.2 Funcions programades

2.2.1 standarize

- Entrada:
  - **np.array** X
- Sortida: **np.array** x
- Funcionament: Per cada atribut, calcula la mitjana i la desviació estàndar, posteriorment normalitza cada dada restant la mitjana i dividint per la desviació estàndar.
- Informació rellevant: Funció utilitzada exclusivament en casos on els rangs de valors son excesivament grans i provoquen problemes de compilació.

2.2.2 make_meshgrid

- Entrada:
  - **np.array** x
  - **np.array** y
  - **float** h
- Sortida:
  - **np.array** xx
  - **np.array** yy
- Funcionament: Crea dos llistes de valors entre $S - 0.25$ i $M + 0.25$ amb pas h (una per cada llista introduïda). S i M son el mínim i el màxim (respectivament) de cada una de les llistes.
- Informació rellevant: Funció utilitzada exclusivament com a pas intermedi per a representar la *zona de decisió* de cada mètode.
  El valor h és opcional i en cas de no introduïr-se cap s'inicialitza a 0.02

2.2.3 plot_contours

- Entrada:
  - **matplotlib axes object** ax
  - **object classifier** clf
  - **np.array** xx
  - **np.array** yy
  - **dictionary** **params
- Sortida:
  - **plot** out
- Funcionament: Grafica la zona de cada classe mitjançant la funció *contourf*.
- Informació rellevant: Funció utilitzada exclusivament com a pas intermedi per a representar la *zona de decisió* de cada mètode.

3 Gestió i estudi del Dataset

3.1 Explicació del Dataset

El dataset tracta sobre una empresa de mòbils fictícia. Aquesta companyia disposa de mòbils amb diferents especificacions però només amb 4 rangs de preus.
L'empresa ha disposat una mostra dels telèfons, amb l'objectiu de classificar els preus dels nous models que desenvolupin, de manera proporcional als que ja tenen categoritzats.
El dataset en qüestió té una mida de 2000 x 21 (files x columnes).
Els 21 atributs recollits dels telèfons són els següents:

Atribut	**Explicació**	**Tipus de dada**
battery_power	Potència de la bateria	int [500-1999]
blue	Bluetooth	binari
clock_speed	Velocitat del microprocessador	float [0.5-3]
dual_sim	Suport dual sim	binari
fc	MegaPixels de la càmera frontal	int [0-19]
four_g	4G	binari
int_memory	Interval de Memòria	int [2-64]
m_dep	Gruix del mòbil	float [0.1-1]
mobile_wt	Alçada del mòbil	int [80-200]
n_cores	Nuclis del processador	int [1-8]
pc	Píxels de la càmera primària	int [0-20]
px_height	Alçada dels píxels	int [0-1907]
px_width	Amplada dels píxels	int [501-1998]
ram	Memòria en megabytes	int [263-3989]
sc_h	Alçada de la pantalla	int [5-19]
sc_w	Amplada de la pantalla	int [0-18]
talk_time	Temps de la bateria	int [2-20]
three_g	3G	binari
touch_screen	Té pantalla tàctil o no	binari
wifi	Wifi	binari
price_range	Preus	int [0-3]

Taula 2: Explicació dels atributs i el seu rang de valors que assoleixen

Destacar que, en haver estat generat de manera aleatòria, el dataset presenta algunes incongruències. Alguns exemples són:

- Els atributs *Sc_ W*[1] i *px_ height*[2], assoleixen 0 que no tenen sentit físic.

No s'han tractat aquestes dades degut a que no són rellevants per a la classificació dels mòbils; explicat a 3.3.

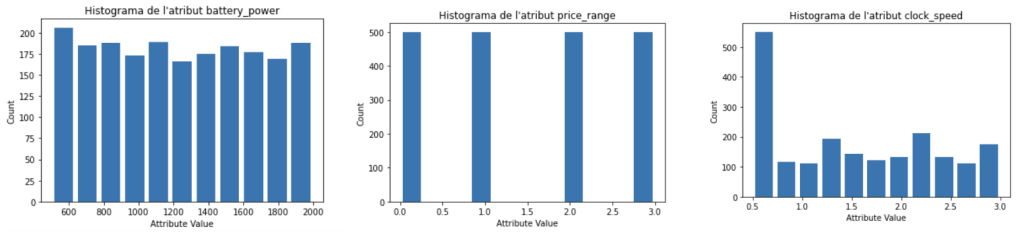
Altrament, existeixen atributs que assoleixen valor 0 únicament per representar que no tenen algun atribut, com és el cas de pc.

[1]Concretament hi existeixen 180 valors incongruents dels 2000 valors del dataset
[2]Concretament hi existeixen 2 valors incongruents dels 2000 valors del dataset

3.2 Distribució de les dades

S'iniciarà l'estudi del dataset observant la distribució de les dades per intuïr relacions senzilles des d'on començar a plantejar els primers models, així com crivar els atributs rellevants per a fer la classificació.

Es mostren ara alguns dels histogrames generats, així com *scatter-plots* del *price_ range* respecte les variables.



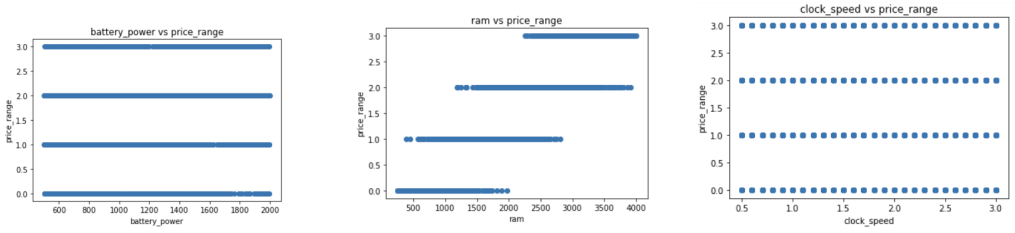
(a) *Battery Power* (b) *Price Range* (c) *Clock Speed*

Figura 1: Mostra dels histogrames generats per l'estudi inicial

S'observen les següents característiques dels histogrames[3]:

- L'atribut *price_ range* té el mateix nombre de mostres per a cada valor del seu rang.
- Existeixen atributs que tendeixen a tenir distribucions uniformes.
- Els atributs restants tenen valors molt comuns en comparació de la resta (pel que poden dificultar la classificació)

Per a obtenir millors conclusions, es decideix veure les relacions dels atributs representant *price_ range* respecte la resta d'atributs.



(a) *Battery Power* (b) *RAM* (c) *Clock Speed*

Figura 2: Mostra dels *scatter-plots* generats per l'estudi inicial

S'obté dels *scatter-plots* [4] la conclusió que l'únic atribut que sembla tenir bones propietats per a classificar el *price_ range* és el *RAM* ja que mostra una tendencia logística.

[3]El conjunt sencer d'histogrames són a 7.1
[4]El conjunt sencer d'*scatter-plots* són a 7.2

3.3 Correlació de les variables

S'ha decidit estudiar la correlació entre els atributs que conté la base de dades per tal d'analitzar la importància entre ells per poder trobar els millors paràmetres per classificar i decidir com tractar les incongruències de les dades exposades anteriorment a l'apartat 3.1.

Atribut	**# afins**	**Correlació**
Battery_power	1	0.2007
Blue	0	0.0205
Clock_speed	0	-0.0066
Dual_sim	0	0.0174
Fc	1	0.0219
Four_g	1	0.0147
Int_memory	0	0.0444
M_dep	0	0.0008
Mobile_wt	0	-0.0303
N_cores	0	0.0043
Pc	1	0.0335
Px_height	1	0.1488
Px_width	1	0.1658
RAM	1	0.9170
Sc_h	1	0.0229
Sc_w	1	0.0387
Talk_time	0	0.0218
Three_g	1	0.0236
Touch_screen	0	-0.0304
Wifi	0	0.0187
Price_range	2	1.0000

Taula 3: Taula d'atributs afins

A partir dels resultats obtinguts es pot deduïr que hi existeixen molt poques correlacions entre els atributs, concretament només hi ha dos atributs rellevants que tenen bona correlació amb l'atribut a classificar *prince_range*.

A la última columna s'exposen les correlacions dels atributs amb *prince_range*. Es poden extreure com a variables rellevants: *battery_power* i *RAM*.

Figura 3: Correlacions més altes entre els atributs del dataset

Es conclou que els atributs que seran utilitzats per a classificar el *prince_range* són els únics rellevants [5] *battery_power* i *RAM* i, per tant, la resta d'atributs no són importants per a la classificació; pel que no seran tinguts en compte.

Per aquest motiu, les incongruències trobades a 3.1 no cal tenir-les en compte ni tractar-les ja que no hi afecten per a la classificació.

[5] Definim rellevants els atributs que tenen una correlació superior o igual a 0.2

3.4 Anàlisi dels atributs rellevants

Analitzant en més detall les distribucions dels atributs rellevants mencionats anteriorment[6] s'observa com els dos atributs estan distribuits de forma uniforme entre els diversos rangs de valors.
Per altre banda s'intueix com l'atribut *RAM* té una tendència logística (ja explicada i analitzada a 3.1) per contra de l'atribut *battery_power*, al que no s'intueix cap relació.

El fet que ambdues variables tinguin distribucions uniformes és una qualitat molt útil a l'hora de classificar ja que permet diferenciar millor les categories en certs rangs (ja que les classes *price_range* són equiprobables).
Tot i així, al no veure una relació directa en la distribució de *battery_power* respecte a *price_range* i *RAM*, es va decidir representar les 3 variables per trobar noves relacions que permetessin començar a plantejar models de classificació.

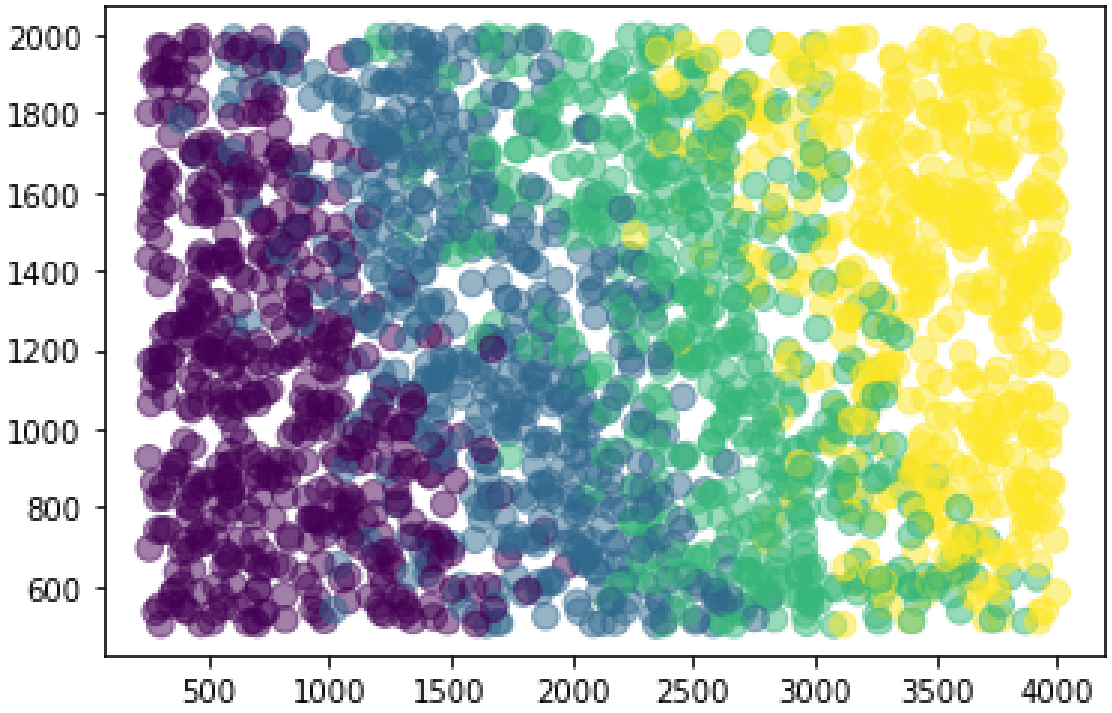


Figura 4: *Price_range* en funció dels paràmetres *RAM* i *battery_power*.

A partir de la imatge generada[7] es pot estimar que existeix una distribució de les dades fàcilment visible.
Es mostra també que existeix una certa divisió lineal entre les diferents classes, tot i que aquesta divisió no és completament lineal ja que és bastant difusa, tractant-se així d'una divisió progresiva/continua més que d'una discontinua.

Malgrat no ser una divisió discreta, s'observa com un model de regressió logística podria classificar de manera bastant precisa el dataset, pel que es començarà estudiant aquest model.

[6] Els histogrames dels atributs és troben a 7.1, així com els *scatter-plots* són a 7.2

[7] A la imatge es representa el *price_range* com a color (essent la gamma de color de més econòmic (lila) a més car (groc)), l'eix x representa la *RAM* i l'eix y representa la *battery_power*

4 Classificació del dataset

4.1 Logistic Regressor

Primerament es provarà un classificador logístic a causa de les observacions fetes anteriorment a 3.4.

4.1.1 Estudi de la precisió

S'inicia l'estudi observant la precisió del model logístic amb paràmetres estàndard.
El resultat d'aplicar un regressor logístic per a classificar el dataset de train sencer (sense utilitzar part del mateix per a validar) és:

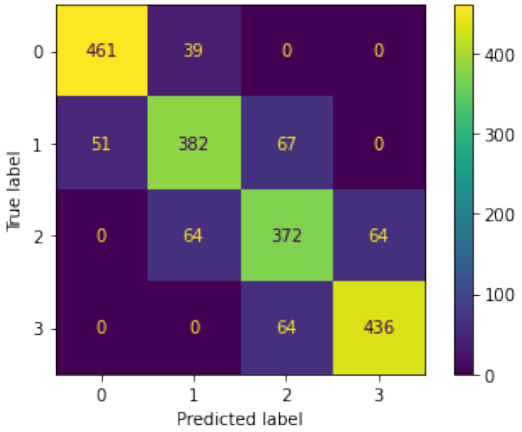


Figura 5: *Confusion matrix* del model logístic estàndard

Es representen en una *confusion matrix* els resultats obtinguts pel classificador Logístic. S'observa com el model no tendeix a confondre classes amb valors molt diferents, si no que, al contrari, tendeix a confondre classes properes entre elles.

Es dedueix que aquest error és degut a valors propers o pertanyents a les fronteres entre les classes (que són molt difuses ja que es sobreposen).

Recalcar com en les classes econòmiques (tipus 0) i les classes més cares (tipus 3) tendeix a encertar-ne casi la totalitat de les dades introduïdes que els hi corresponen.

Calculant l'*accuracy* del model s'obté una precisió del 82.55%. Com és una bona predicció s'opta per seguir millorant el model mitjançant un *Cross Validation*.

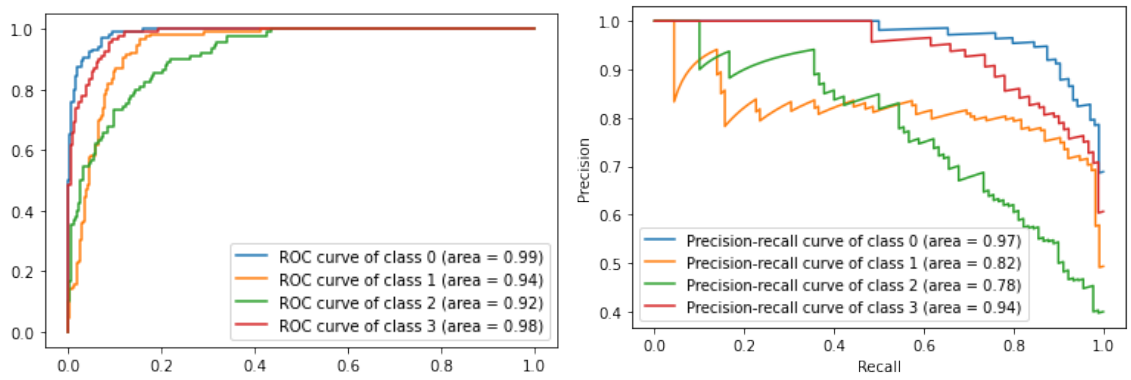
4.1.2 Cross Validation

Al aplicar un *cross validation* (amb 5 subdivisions del dataset) s'obté un resultat *accuracy* de 82.55%, és a dir, el mateix resultat que sense aplicar el *cross validation* amb tot el dataset.

D'aquest resultat es dedueïx que el regessor que havia singut obtingut inicialment no estava sent *overfitted* i que classifica les classes de telèfons mòbils amb una precisió bastant bona.

Se segueix l'anàlisis del classificador estudiant la *ROC curve* i la *precission-recall curve* del model.

4.1.3 Anàlisi de la ROC i Precission-Recall Curve



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 6: Corba *ROC* i *Precission-Recall* del model Logístic

S'observa de la corba *ROC* que l'*accuracy* del model és bona, ja que classifica força bé les diferents classes.
Per altra banda, de la corba *Precission-Recall* s'extreu que. a l'hora de classificar les dades, les classes 2 i 3 tendeixen a classificar-se de manera errònia, sobretot la clase 2 que tendeix a confondre's amb la classe 1 i 3.

Per tal de millorar la precisió del model, es cercarà la millor combinació d'hiperparàmetres del model.

4.1.4 Cerca dels millors hiperparàmetres

Paràmetres	Valors
solver	newton-cg
	lbfgs
	liblinear
	sag
	saga
	elasticnet
	none
penalty	l1
	l2
	elasticnet
	none
intercept_scaling	0.5
	1
	⋮
	3.5
	4}
weight	{1, 1, 1, 1}
	{2, 0.5, 0.5, 2}
	{0.5, 2, 2, 0.5}
	balanced
	None

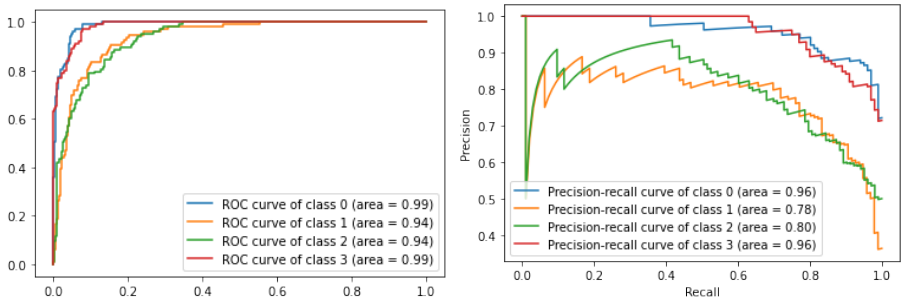
Figura 7: Hiperparàmetres model Logístic

Per a obtenir la millor precisió possible amb el logistic regression s'ha proposat trobar la combinació de paràmetres que maximitzin la *accuracy*.
Els paràmetres i valors testejats es mostren a la taula del costat.

Els resultat de la cerca dels millors hiperparàmetres són els valors remarcats a la taula. Aquests paràmetres permeten obtenir un *accuracy* del 82.75% (tot i que aquest valor pot avegades ser incrementat fins a 83% en alguns casos per fenòmens aleatoris a l'hora de generar el model).

S'observa una milloria no suficientement gran però, degut al poc esforç computacional del programa, és una opció a tenir en compte (ja que dona valors molt correctes i triga poc al executar-se tot i no poder-se millorar molt més el model).

Seguidament, es procedeix a analitzar les corbes *ROC* i *Precision-Recall* del model amb els millors hiperparàmetres:



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 8: Corba *ROC* i *Precission-Recall* del millor model Logístic

S'observa com les corbes han cambiat respecte a les obtingudes a 4.1.3, més concretament es pot apreciar que la corba *ROC* ha millorat en totes les classes.
D'altra banda *Precission-Recall* al principi les classes 1 i 2 empitjoren però millora breument al final. Tot i així, el valor que obté la classe 2 no és millor que anteriorment.

Al millorar les dades obtingudes (de l'*accuracy* i les corbes *ROC* i *Precision-Recall*), es conclou que aquest model és eficaç i que s'emprarà per comparar amb altres models.

4.1.5 Resultats del classificador

Finalment, s'obté la *zona de decisió* del classificador logístic amb els millors hiperparàmetres[8] obtinguts:

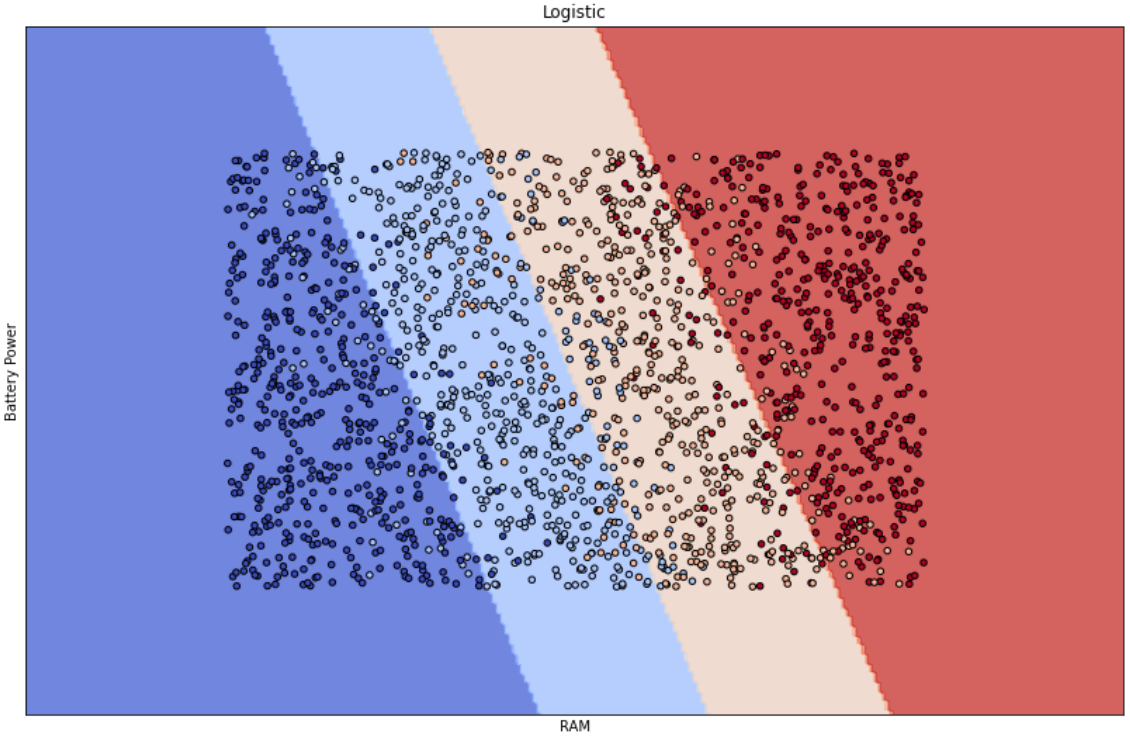


Figura 9: Zona de decisió del millor model logístic trobat

S'observa com el model senyala de manera intuïtiva les divisions entre les classes, tot i així, algunes dades que hi són a les fronteres es troben mal etiquetades.

No obstant, aquests casos són escassos i, per tant, són insuficients com per a alterar el classificador.
Es dedueïx que la precisió del model no podrà superar un cert umbral degut a aquest valor i, intuïm, que aquest serà inferior o igual a 90%.

Es provaran altres models de classificació per a intentar arribar aquest umbral (si existeix o superar-lo en cas contrari).

[8]Els hiperparàmetres que milloren el classificador són explicats a 4.1.4

4.2 K-Nearest Neighbors

4.2.1 Estudi de la precisió

Es continua l'estudi observant la precisió del model KNN amb paràmetres estàndards, ja que al tractar-se d'un dataset amb dades molt compactes podria obtenir bons resultats.
El resultat d'aplicar aquest model per a classificar el dataset de train sencer (sense utilitzar part del mateix per a validar) és:

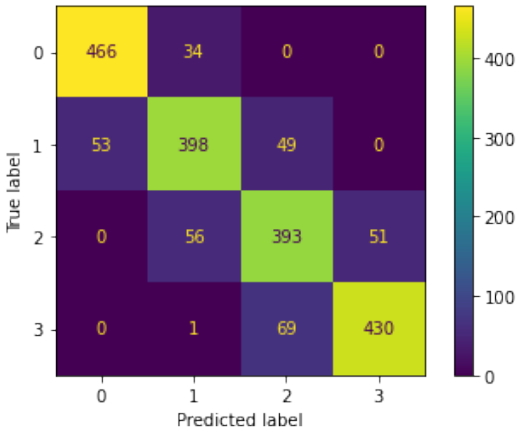


Figura 10: *Confusion matrix* del K-Nearest Neighbors

A la figura s'identifiquen en una *confusion matrix* els resultats obtinguts pel classificador. S'observa com el model tendeix a confondre classes properes entre elles, mentre que amb classes molt separades (com és el cas de la 0 i la 3) no té cap tipus de confusió.

Es dedueix que els errors es deuen a valors propers o pertanyents a les fronteres entre les classes (que són molt difuses ja que es sobreposen).

Cal recalcar que el comportament del model és molt similar al model logístic ja estudiat abans, pel que es pot calcular que la tendència serà similar en els següents models que seràn provats.

Calculant l'*accuracy* del model s'obté una precisió del 84.35%. Com és una bona predicció s'opta per seguir millorant el model mitjançant un *Cross Validation*.

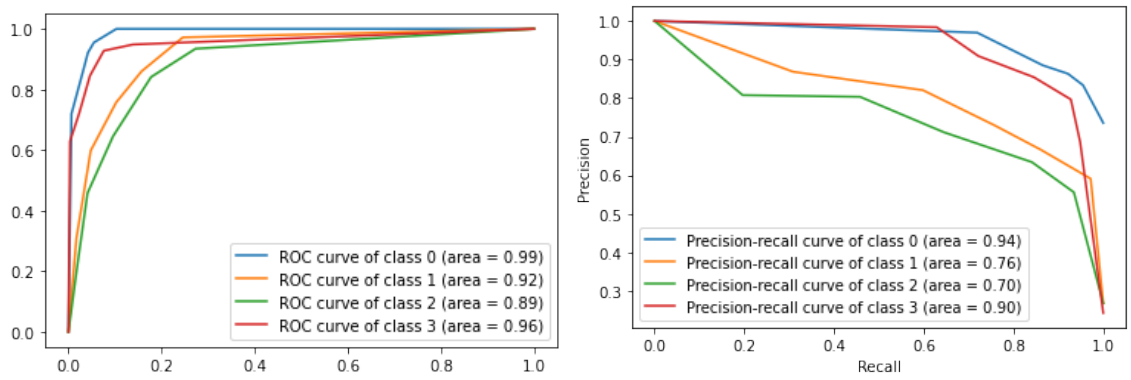
4.2.2 Cross Validation

Al aplicar un cross validation (amb 5 subdivisions del dataset) s'obté un resultat *accuracy* de 78.5%. Si bé és inferior al valor resultant inicialment, aquest comportament es pot explicar de dues maneres:

- El model està *overfitted*.
- El model requeria de més dades per a millorar la predicció.

No es pot assegurar que el model hagi estat *overfitted*, ja que al tractar-se d'un model KNN, requereix de moltes dades per a funcionar i, al reduïr el *train dataset* per a obtenir un *test dataset*, es podria donar una disminució de la precisió.
Es procedeix doncs a estudiar la precisió del model mitjançant les curves *ROC* i *Precision-Recall*.

4.2.3 Anàlisi de la ROC i Precission-Recall Curve



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 11: Corba *ROC* i *Precission-Recall* del model KNN

Es pot apreciar de la corba *ROC* que l'*accuracy* del model és bona, ja que classifica força bé les diferents classes.
Per altra banda, de la corba *Precission-Recall* s'extreu que, a l'hora de classificar les dades, les classes 0 i 3 es classifiquen de manera correcta en la majoria dels casos. En canvi, les classes 1 i 2 tendeixen a classificar-se de manera errònia, sobretot la classe 2, que tendeix a confondre's amb la classe 1.

Destacar que les corbes no són tan bones com al model Logístic (4.1.3). Tot i així, el rang de millora és bastant més elevat que amb el logístic, pel que es cercaran els millors hiperparàmetres per tal de millorar el model el màxim possible.

4.2.4 Cerca dels millors hiperparàmetres

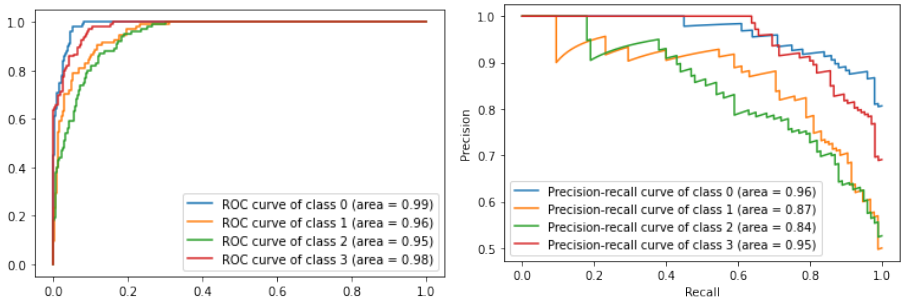
Paràmetres	Valors
weights	uniform
	distance
algorithm	auto
	ball_tree
	kd_tree
	brute
p	1
	⋮
	10
n_neighbors	1
	⋮
	11
	⋮
	30

Per a obtenir la millor precisió possible amb el KNN s'ha proposat trobar la combinació de paràmetres que maximitzin la *accuracy*.
Els paràmetres i valors testejats es mostren a la taula del costat.

Els resultat de la cerca dels millors hiperparàmetres són els valors remarcats a la taula. Aquest paràmetres permeten obtenir un *accuracy* del 81.5% (tot i que aquest valor, en alguns casos, es pot incrementar fins a 83% en alguns casos per fenòmens aleatoris a l'hora de generar el model).

Figura 12: Hiperparàmetres model KNN

S'observa una milloria però no n'és suficient en comparació al regressor logístic (4.1.4. Com el model té un cost computacional baix i un percentatge d'encert semblant al logístic, es manté el model com a vàlid i s'analitzen les seves corbes *ROC* i *Precision-Recall*.
S'analitzen ara les corbes *ROC* i *Precision-Recall* del model amb els millors hiperparàmetres:



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 13: Corba *ROC* i *Precission-Recall* del millor model KNN

S'observa com les corbes han cambiat respecte a les obtingudes a 4.1.3, concretament com les dues corbes ha millorat en totes les classes. No obstant, s'observa el següent:

- La *Recall* i *Precission* han millorat en valor en totes
- Les classes 1 i 2 inicialment tenen bastants errors, peró després milloren substancialment per a terminar oferint millors resultats
- Els valors son força semblants al Logístic, però no el suficient com per a classificar igual, pel que surgeix la idea de combinar els models en un ensemble.

4.2.5 Resultats del classificador

Finalment, s'obté la *zona de decisió* del classificador KNN amb els millors hiperparàmetres[9] obtinguts:

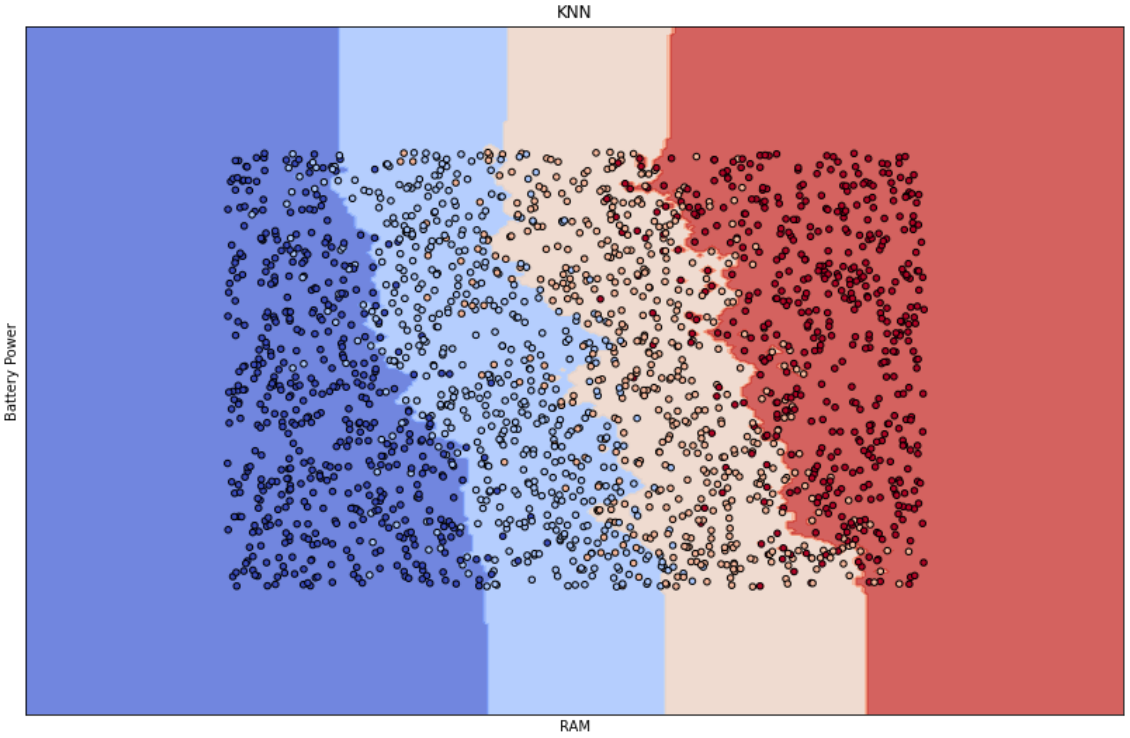


Figura 14: Zona de decisió del millor model KNN trobat

S'observa com el model senyala de manera difussa les divisions entre les classes, tot i així, algunes dades es troben mal etiquetades i les fronteres de decissió no son clares i obtenen formes peculiars.

Aquest comportament (com el de crear regions de tipus de classes on clarament predomina un tipus diferent) fa que aquest mètode no pugui ser molt util en clasificar dades noves entrants de manera eficaç.

Com el mètode obte prediccions semblants al Logístic i té una precisió decent, es manté el model KNN per a implementar-lo en clasificador de tipus ensemble.

[9]Els hiperparàmetres que milloren el classificador són explicats a 4.2.4

4.3 Support Vector Classification

4.3.1 Estudi de la precisió

Es continua l'estudi observant la precisió del model SVC amb paràmetres estàndards, ja que es considera que (al dividir zones per fronteres rectes amb toleràncies) podria donar bons resultats.
El resultat d'aplicar aquest model per a classificar el dataset de train sencer (sense utilitzar part del mateix per a validar) és:

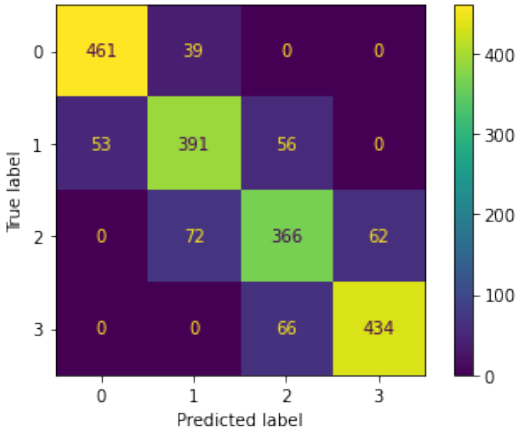


Figura 15: *Confusion matrix* del SVC

A la figura s'identifiquen en una *confusion matrix* els resultats obtinguts pel classificador. S'observa com el model obté resultats molt similars als del model logístic i del model KNN (4.1.5 i 4.2.5 respectivament).

La única diferencia significativa del model es que tendeix a infravalorar el tipus de classe (tendint a catalogar un tipus inferior més que superior en cas de classificar-lo de manera erronia).

Si bé no es un comportament desitjat pel classificador (ja que ha de ser el més precís possible), torna a proporcionar la idea de crear un ensemble de model.

Calculant l'*accuracy* del model s'obté una precisió del 82.6%. Com és una bona predicció s'opta per seguir millorant el model mitjançant un *Cross Validation*.

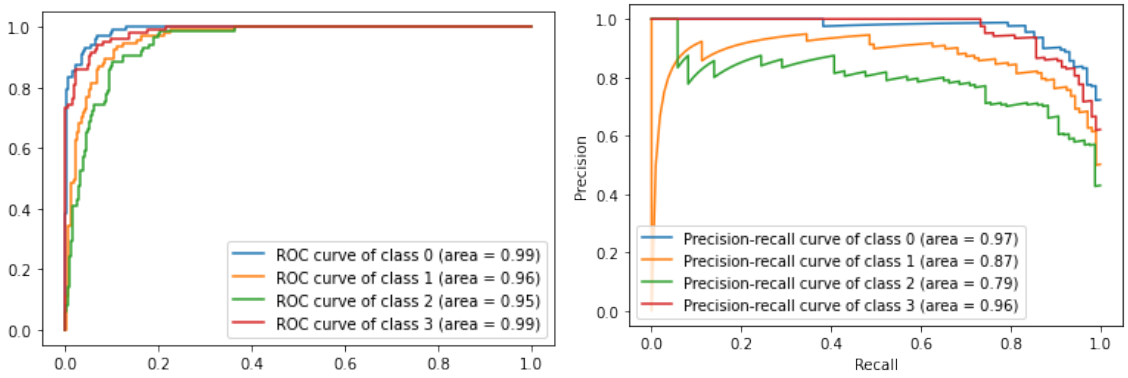
4.3.2 Cross Validation

Al aplicar un cross validation (amb 5 subdivisions del dataset) s'obté un resultat *accuracy* de 82.4%. Si bé és inferior al valor resultant inicial, aquest és molt proper.

No es pot assegurar que el model inicial hagi estat *overfitted*, peró al obtenir una precisió tan similar (i alt) es pot intuïr que realment pot tractar-se d'un model eficient a l'hora de classificar les dades.

Es procedeix doncs a estudiar la precisió del model mitjançant les curves *ROC* i *Precision-Recall*.

4.3.3 Anàlisi de la ROC i Precission-Recall Curve



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 16: Corba *ROC* i *Precission-Recall* del model SVC

Es pot apreciar de la corba *ROC* que l'*accuracy* del model és bona, ja que classifica força bé les diferents classes.
Per altra banda, de la corba *Precission-Recall* s'extreu un comportament similar al del model logístic (4.1.3).

Recalcar que aquest model té una pitjor precisió inicial (respecte a la classe 1) que la resta de models, pel que la cerca de hiperparàmetres no només es centrarà en la millora de la precisió, sino que també s'enfocarà en millorar la precisió de la classificació de la clase de tipus 3.

4.3.4 Cerca dels millors hiperparàmetres

Paràmetres	Valors
kernel	linear
	poly
	rbf
	sigmoid
gamma	scale
	auto

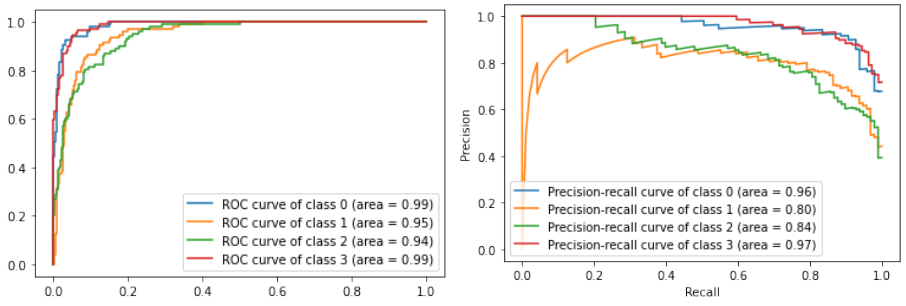
Per a obtenir la millor precisió possible amb el SVC s'ha proposat trobar la combinació de paràmetres que maximitzin la *accuracy*.

Els paràmetres i valors testejats es mostren a la taula del costat.

Figura 17: Hiperparàmetres model KNN

Els resultats de la cerca dels millors hiperparàmetres són els valors remarcats a la taula. Aquests paràmetres permeten obtenir un *accuracy* del 84% (tot i que aquest valor pot oscil·lar entre 82% i 85% per fenòmens aleatoris).

S'observa una milloria substancial en comparació al regressor logístic (4.1.4) i al anterior model KNN (4.2.4).
Com el model té un cost computacional baix i un percentatge d'encert semblant al logístic, es manté el model com a vàlid i s'analitzen les seves corbes *ROC* i *Precision-Recall*.
S'analitzen ara les corbes *ROC* i *Precision-Recall* del model amb els millors hiperparàmetres:



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 18: Corba *ROC* i *Precission-Recall* del millor model SVC

S'observa com les corbes no han cambiat respecte a les obtingudes a 4.3.3, No obstant, s'observa el següent:

- La *Recall* i *Precission* ha empitjorat en 3 de les 4 classes
- La classe 1 conté inicialment bastants errors peró després millora substancialment com per a obtenir valors similars a la gràfica del model sense cerca d'hiperparàmetres.
- Els valors segueixen sent força semblants al Logìstic, però no el suficient com per a classificar igual, pel que es manté la idea de combinar els models en un ensemble.

4.3.5 Resultats del classificador

Finalment, s'obté la *zona de decisió* del classificador SVC amb els millors hiperparàmetres[10] obtinguts:

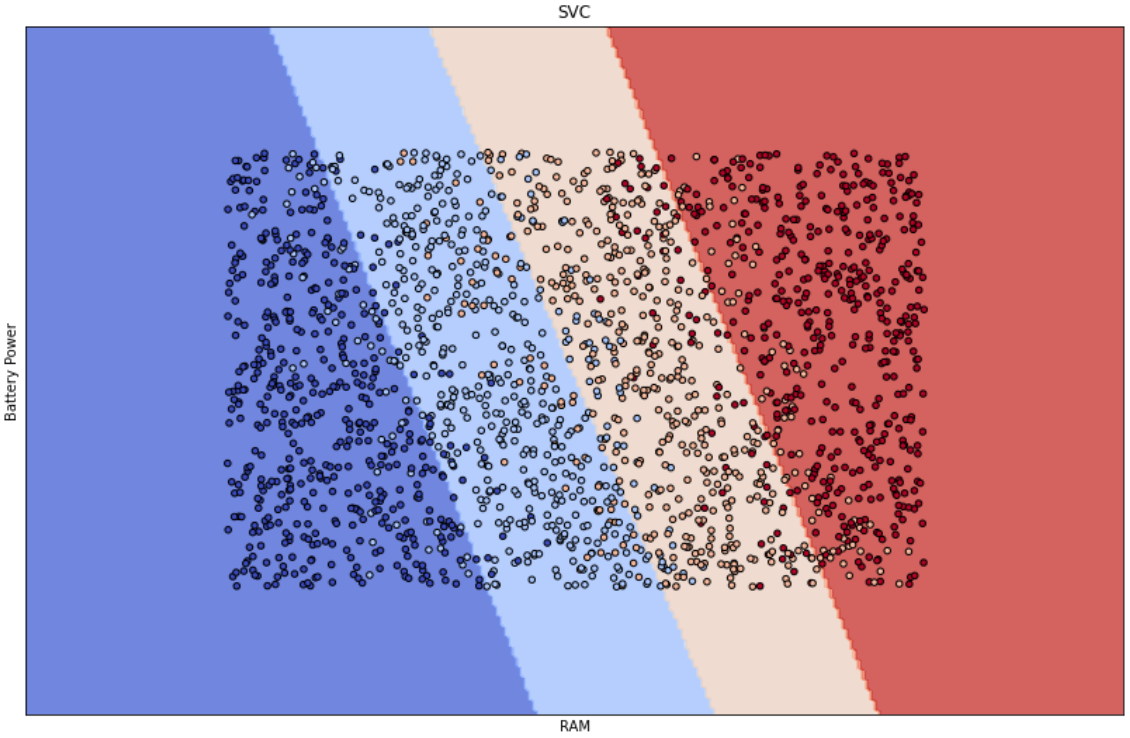


Figura 19: Zona de decisió del millor model SVC trobat

S'observa com el model senyala de manera similar al regressor logístic les divisions entre les classes, tot i així, algunes dades es troben mal etiquetades degut a les fronteres tan difuses de les dades.

Com el mètode obté prediccions semblants al Logístic i té una precisió eficaç (depenen d'un cert factor aleatori), es manté el model SVC com a un molt bon candidat i també per a implementar-lo en classificador de tipus ensemble.

[10]Els hiperparàmetres que milloren el classificador són explicats a 4.3.4

4.3.6 Model LinearSVC

Al haver observat que un dels hiperparàmetres del model SVC trobat que fa millorar la predicció, és el paràmetre de *Linear*[11], s'ha decidit estudiar el comportament del model LinearSVC de manera exprés (estudiant directament els hiperparàmetres[12] i la zona de decisió) amb el propòsit d'obtenir un millor resultat.
A continuació es presenta la zona de decisió:

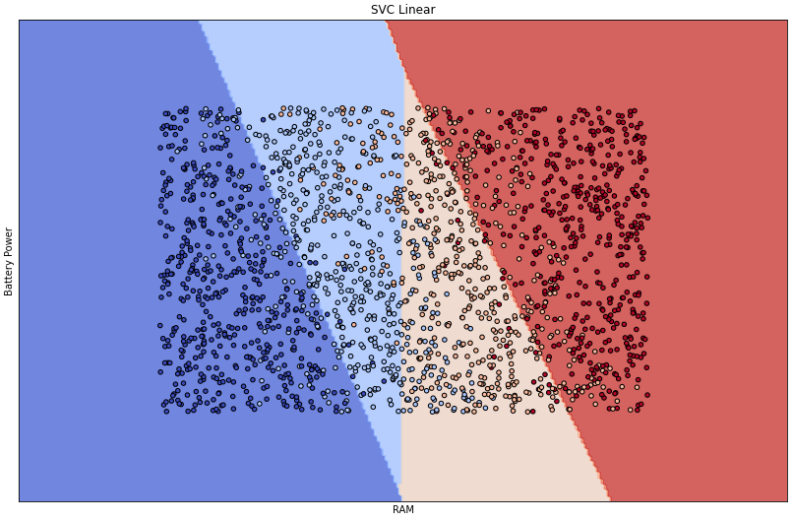


Figura 20: Zona de decisió del millor model LinearSVC

Executant el model, obtenim una *accuracy* del 76.25% que, en comparació amb la obtinguda amb el millor model SVC, no resulta prou bona.

Aquest resultat s'explica per com es divideixen les zones de decisió a aquest model, ja que divideix les classes 1 i 2 d'una manera poc intuïtiva i de forma que moltes dades s'etiqueten de forma errònia.

A causa de tot això, es conclou que es tracta d'un mal model.

[11] Hiperparàmetres explicats a 4.3.4
[12] En cas de volver consultar-los, revisar el codi entregat conjuntament amb aquesta memòria

4.4 Decision Tree Classificator

4.4.1 Estudi de la precisió

Es continua l'estudi observant la precisió del model *DecisionTreeClassifier* amb paràmetres estàndard, aquesta decisió es pren per intentar tornar a dividir el model en línies rectes, però ara mitjançant escalonaments.
El resultat d'aplicar aquest model per a classificar el dataset de train sencer (sense utilitzar part del mateix per a validar) és:

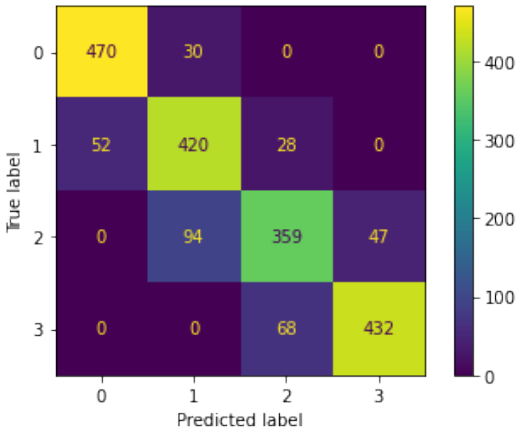


Figura 21: *Confusion matrix* del *DecisionTreeClassifier*

A la figura s'observen en una *confusion matrix* els resultats obtinguts pel classificador. S'observa com el model tendeix a un comportament similar al del model SVC (4.3), ja que tendeix a infraestimar les classes.

Cal recalcar que la classe de tipus 2 té bastants problemes a l'hora d'etiquetar-la, ja que és el model que menys etiquetes encerta de les de tipus 2.

Aquest comportament n'indica que el model podria no ser suficientment adien;, tot i així, s'opta per millorar-lo, amb la intenció d'esbrinar algun nou model que pugui oferir millors resultats

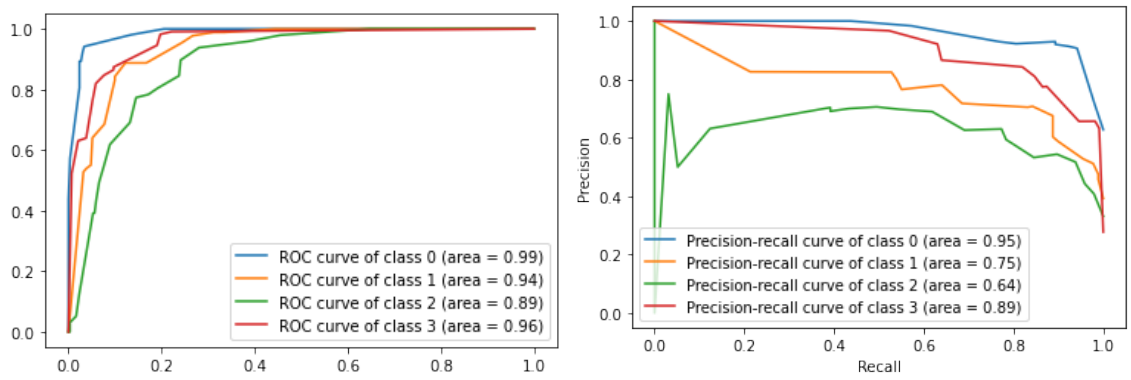
Calculant l'*accuracy* del model s'obté una precisió del 84.05%. Com és una bona predicció s'opta per seguir millorant el model mitjançant un *Cross Validation*.

4.4.2 Cross Validation

Al aplicar un cross validation[13] s'obté un resultat accuracy de 79.7%, el qual és un bon resultat, i, com en va passar al model KNN (4.2) la mateixa precisió que sense el *Cross Validation*.
Per tant, s'arriba a la mateixa conclusió i es continua amb l'anàlisi del model mitjançant un estudi de les corbes *ROC* i *Precision-Recall*.

[13]El *K-Fold* del *Cross Validation* és igual que als anterior $K = 5$, pel que s'omet i s'ometrà en els futurs anàlisis dels models

4.4.3 Anàlisi de la ROC i Precission-Recall Curve



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 22: Corba *ROC* i *Precission-Recall* del model DecisionTree

S'observa de la corba *ROC* que l'*accuracy* del model és bona, ja que classifica força bé les diferents classes. Tot i així, els valors que s'obtenen no són del tot adients en comparació amb altres models com el Logístic o el SVC (4.1 i 4.3 respectivament).

Per altra banda, de la corba *Precission-Recall* s'extreu que, a l'hora de classificar les dades, les classes 1 i 2 es classifiquen de manera errònia. Aquesta situació és especialment visible a la classe 2, que obté un valor de *Precission-Recall* molt baix per comparació a la resta de models provats. Es buscarà millorar el classificador mitjançant una cerca dels hiperparàmetres que permetin millorar la precisió del model així com la classificació de les dades amb etiqueta de valor 2.

4.4.4 Cerca dels millors hiperparàmetres

Paràmetres	Valors
criterion	gini
	entropy
splitter	best
	random
max_features	sqrt
	log2
	random
	None
max_depth	2
	⋮
	5
	⋮
	30
min_samples_split	2
	⋮
	10

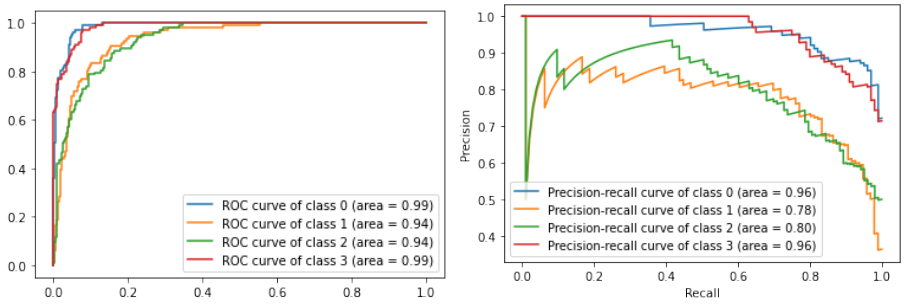
Figura 23: Hiperparàmetres model Decision Tree

Per a obtenir la millor precisió possible amb el DecisionTree s'ha proposat trobar la combinació de paràmetres que maximitzin la *accuracy*.
Els paràmetres i valors testejats es mostren a la taula del costat.

Els resultats de la cerca dels millors hiperparàmetres són els valors remarcats a la taula. Aquests paràmetres permeten obtenir un *accuracy* del 81.5%.

S'observa una milloria substancialment gran, però en tot cas insuficient com per a competir amb el model logistic i SVC (4.1 i 4.3 respectivament).

Precedim a estudiar les corbes *ROC* i *Precision-Recall*.



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 24: Corba *ROC* i *Precission-Recall* del millor model Logístic

S'observa com les corbes han cambiat dràsticament respecte a les obtingudes.

Tot i així, les característiques de les corbes són força similars al models ja observat del KNN (4.2), pel que no es creu necessari utilitzar aquest mètode degut a la ineficiència a l'hora de classificar respecte dels mètodes que hem utilitzar al comparar-lo.

Per altra banda, al millorar les dades obtingudes (de l'*accuracy* i les corbes *ROC* i *Precision-Recall*), es conclou que aquest model pot servir amb mètodes d'*ensemble* per a millorar la seva predicció.

4.4.5 Resultats del classificador

Finalment, s'obté la *zona de decisió* del classificador amb els millors hiperparàmetres[14] obtinguts:

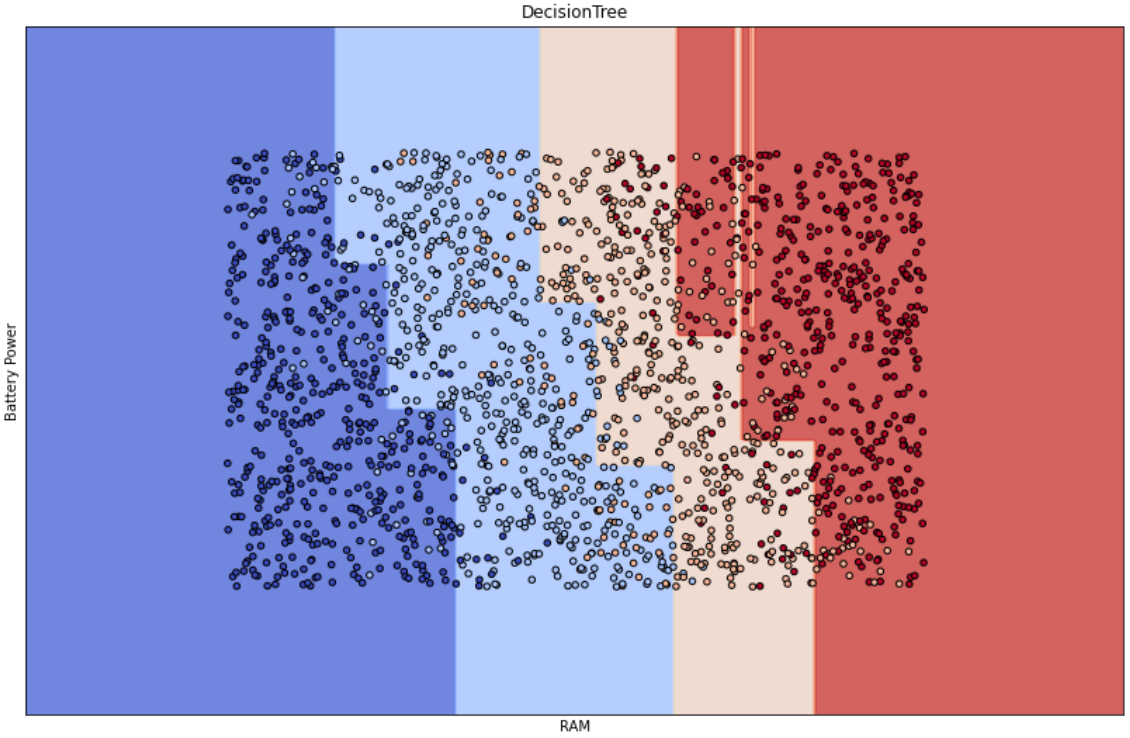


Figura 25: Zona de decisió del millor model *Decision Tree* trobat

S'observa com les fronteres de decisió no son intuïtives i que moltes dades no s'han etiquetat correctament.

Confirmem la conclusió de desestimar el mètode i és proposa estudiar-ne un ensemble de DecisionTree.

[14]Els hiperparàmetres que milloren el classificador són explicats a 7.3.5

4.5 Random Forest Classificator

4.5.1 Estudi de la precisió

Es continua l'estudi observant la precisió del *RandomForest* amb paràmetres estàndard, ja que amb els resultats anterior s'intueix que es poden obtenir bons resultats.
El resultat d'aplicar aquest model per a classificar el dataset de train sencer (sense utilitzar part del mateix per a validar) és:

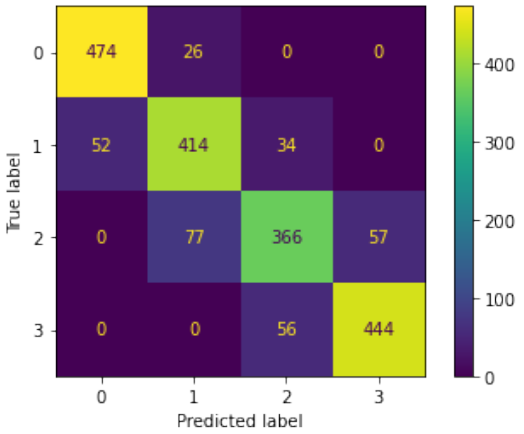


Figura 26: *Confusion matrix* del RandomForestClassifier

A la figura s'observen en una *confusion matrix* els resultats obtinguts pel classificador. S'observa com el model tendeix a obtenir els mateixos resultats que el DecisionTree (4.4), efecte que té sentit al tractar-se d'un ensemble de DecisionTrees.

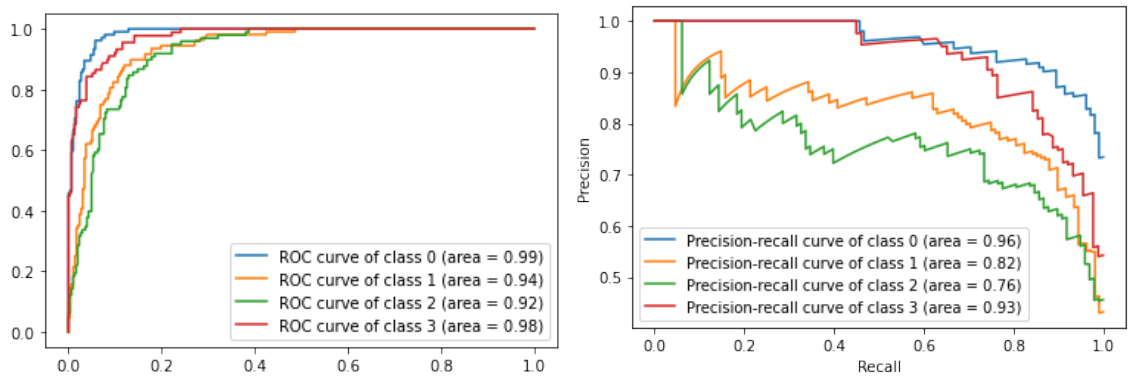
Com que els resultats són similars als Decision tree, però al executar-se s'obté una *accuracy* del model 84.9%, és a dir, una molt bona precisió. Per tant, s'opta per seguir millorant el model mitjançant un *Cross Validation*.

4.5.2 Cross Validation

Al aplicar un cross validation s'obté un resultat accuracy de 80.75%. Com que la precisió ha disminuït sustancialment, es conclou que el model estaba *overfitted* pel que caldra tenir en cura l'introducció de les dades i la gestió dels parametres per evitar aquesta situació en els futus analisis.

Se segueix l'anàlisi del classificador estudiant la ROC curve i la precission-recall curve del model.

4.5.3 Anàlisi de la ROC i Precission-Recall Curve



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 27: Corba *ROC* i *Precission-Recall* del model Logístic

S'observa de la corba *ROC* que l'*accuracy* del model és bona, ja que classifica força bé les diferents classes.

Per altra banda, de la corba *Precission-Recall* s'extreu una conclusió similar a les ja trobades anteriorment (problemes en classificar la clase 1 i 2).

Per a la resta de valors s'obtenen mesures decents pel que es seguirà l'estudi del model mitjançant la recerca de la millor combinació d'hiperparàmetres possible.

4.5.4 Cerca dels millors hiperparàmetres

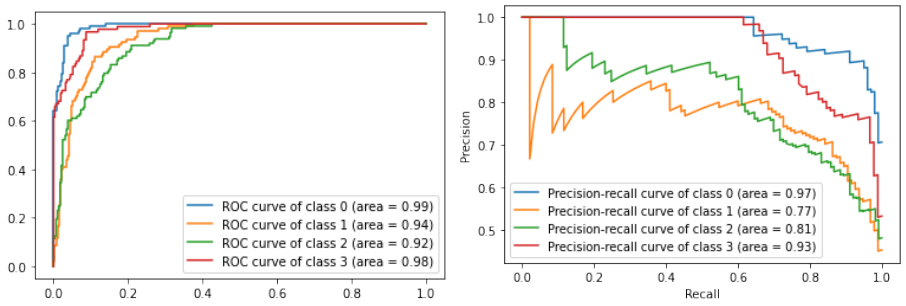
Paràmetres	Valors
criterion	gini entropy
max$_{f}eatures$	sqrt log2 None
class_weight	balanced balanced_subsample None
bootstrap	true false
max_samples	0.1 ⋮ 25
n_estimators	10 ⋮ 160 ⋮ 210
max_depth	2 ⋮ 11 ⋮ 30
min_samples_split	2 3 ⋮ 10

Figura 28: Hiperparàmetres model KNN

Per a obtenir la millor precisió possible amb el RandomForest s'ha proposat trobar la combinació de paràmetres que maximitzin la *accuracy*.
Els paràmetres i valors testejats es mostren a la taula del costat.

Els resultats de la cerca dels millors hiperparàmetres són els valors remarcats a la taula. Aquest paràmetres permeten obtenir un *accuracy* del 81.5% (tot i que aquest valor, en alguns casos, es pot incrementar fins a 23% per fenòmens aleatoris a l'hora de generar el model).

S'observa una milloria però no n'és suficient en comparació al regressor logístic (4.1.4.
Com el model té un cost computacional baix i un percentatge d'encert semblant al logístic, es manté el model com a vàlid i s'analitzen les seves corbes *ROC* i *Precision-Recall*.
S'analitzen ara les corbes *ROC* i *Precision-Recall* del model amb els millors hiperparàmetres:



(a) *ROC curve* (b) *Precission-Recall curve*

Figura 29: Corba *ROC* i *Precission-Recall* del millor model *Random Forest*

S'observa com les corbes han cambiat respecte a les obtingudes a 4.5.3, concretament com les dues corbes ha millorat en la majoria de les classes. No obstant, s'observa el següent:

- La *Recall* i *Precission* ha millorat en la classe 2 pero empitjorat en la classe 1
- Els valors son força semblants al Logístic, però no el suficient com per a classificar igual, pel que surgeix la idea de combinar els models en un ensemble.

4.5.5 Resultats del classificador

Finalment, s'obté la *zona de decisió* del classificador amb els millors hiperparàmetres[15] obtinguts:

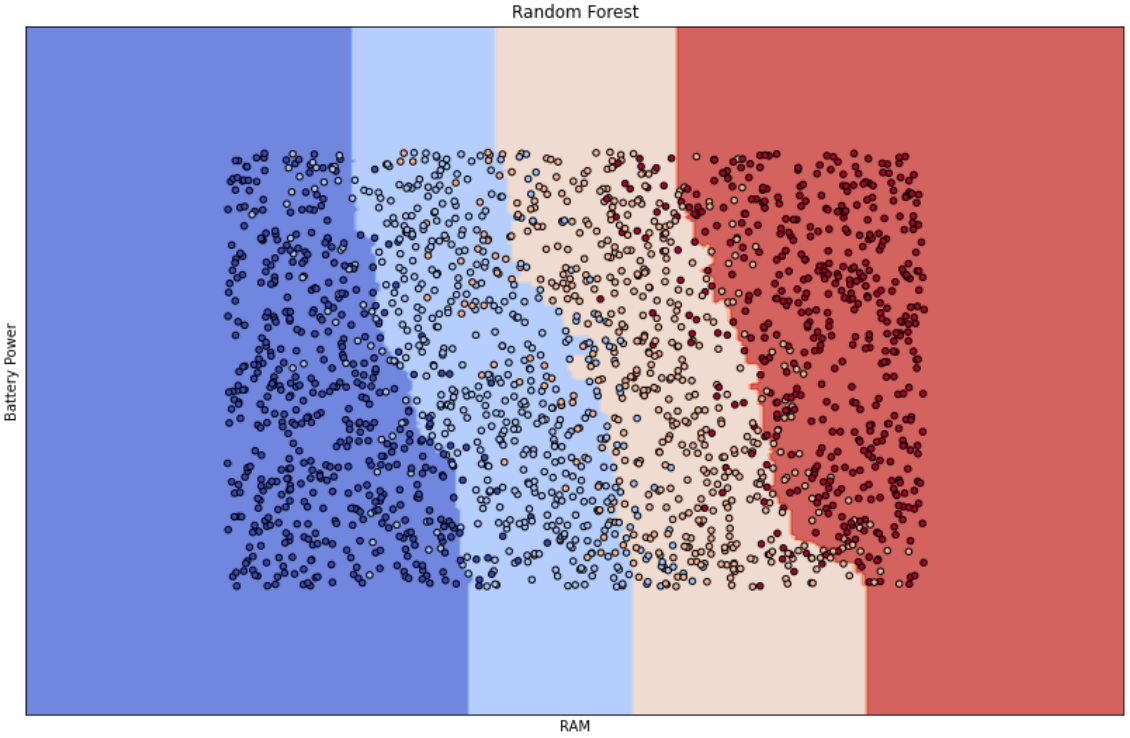


Figura 30: Zona de decisió del millor model *Random Forest* trobat

S'observa com la zona de decisió del millor model del Random Forest és molt similar al del model KNN (4.2) pel que arribem a la mateixa conclusió.

[15]Els hiperparàmetres que milloren el classificador són explicats a 4.5.4

4.6 Altres metodes

En el codi del programa s'observa com s'ha intentat millorar la predicció mitjançant ensemble methods, K-means i mètode EM, tot i així, la precisió no era prou bona pel que es van desentimar. Si es vol consultar els mètodes es recomana consultar el codi.

5 Resolució de les preguntes

5.1 Apartat B.1

1. **Quants atributs té la vostra base de dades?** 21 atributs.
2. **Quin tipus d'atributs tens? (Números, temporals, categorics, binaris...)** Binaris i numèrics.
3. **Com es el target, quantes categories diferents existeixen?** 4
4. **Podeu veure alguna correlació entre X i y?** Com s'ha comentat en apartats anteriors 3.3, observem que hi ha una correlació entre els atributs *price_ range i RAM.*
5. **Estan balancejades les etiquetes (distribució similar entre categories)? Creus que pot afectar a la classificació la seva distribució?** Si que existeix un cert balanç, ja que la majoria són binàries o uniformes, i si que afecta a la classificació ja que si una classe és propensa a tenir un cert valor d'atribut i les altres no, ajuda a classificar aquella classe.

5.2 Apartat B.2

1. **Estàn les dades normalitzades? Caldria fer-ho?** Veiem que les dades no estàn normalitzades, el nostre dataset no conté dades duplicades ni anomialies del tipus *NaN*. Tot i així, s'ha decidit normalitzar el dataset en casos específics, com els que s'expliquen a 4.4.5, 4.2.5, 4.3.6, 4.1.5, 4.5.5, 4.3.5.

2. **Teniu gaires dades sense informació? Els NaNs a pandas? Tingueu en compte que hi ha metodes que no els toleren durant el aprenentatge. Com afecta a la classificació si les filtrem? I si les reompliu? Com ho farieu?** Ens trobem en el cas, en que el nostre dataset no conté valors *NaN*, però si que hi ha alguns valors amb incongruències, expliquem el tractament d'aquestes incongruències a 3.1.

3. **Teniu dades categòriques? Quina seria la codificació amb més sentit?** El nostre dataset no conté dades categòriques.

4. **Caldria aplicar *sklearn.decomposition.PCA*? Quins beneficis o inconvenients trobarieu?** Hem considerat que no caldria aplicar una PCA perquè aquest s'usa quan comptem amb un gran nombre de variables amb altes correlacions, qualitat de la que no disposa el nostre dataset; exposat a 3.3.

5. **Es poden aplicar *Polynomial Features* per millorar la classificació? En quins casos té sentit fer-ho?** No s'ha aplicat cap *Polynomial Features* ja que amb les dues variables significatives *RAM* i *battery_power* no té sentit aplicar combinacions lineals o polinòmiques per a obtenir atributs nous.

5.3 Apartat B.3

1. **Quins models heu considerat?** Els classificadors considerats han estat exposats anteriorment a l'apartat 4: KNN, Logistic Regressor, Decision Tree, Random Forest, K-means, EM classifier, SVC, Linear SVC, AdaBoost Classifier, Ensemble Classifier, GridSearchCV i BayesSearchCV.

2. **Quin creieu que serà el més precís?** El GridSearchCV, ja que té en compte tots els models estudiats anteriorment per tal de combinar-los i donar la major precisió possible, tot i que al tenir un temps de computació molt elevat que fa inviable esperar a veure els resultats, pel que cal destacar que sense tenir-lo en compte, el Logístic o SVC serien els més precisos.

3. **Quin serà el més ràpid?** El model logístic.

4. **Seria una bona idea fer un *ensemble*? Quins inconvenients creieu que pot haver-hi?** Si; ja que les tècniques ensemble fan servir diversos models per obtenir un millor resultat del que ens donaria un sol model, millorant així la precisió. L'inconvenient principal és l'alt cost computacional que implica aquesta tasca.

5.4 Apartat B.4

1. **Per què és important cross-validar els resultats?** Per així no tenir overfitting en les dades de testeig i, per tant, millorar la precisió general del classificador.

2. **Separa la base de dades en el conjunt de train-test. Com de fiables serán els resultats obtinguts? En quins casos serà més fiable, si tenim moltes dades d'entrenament o poques?** Les dades seran menys fiables que si disposesim de dades especialment preparades per testejar i entrenar; tot i així, la fiabilitat és suficientment alta com per a utilitzar aquest recurs. El classificador obtindrà millors prediccions com més dades d'entrenament tingui, sempre i quant en disposi de suficients per al testeig.
   En el nostre cas s'ha dividit el dataset en un 80% i 20% en train i test.

3. **Quin tipus de K-fold heu escollit? Quants conjunts heu seleccionat (quina k)? Com afecta els diferents valors de k?** S'ha utilitzat el mètode *CrossValidation* de la llibreria *sklearn* amb un valor de $k = 5$. S'ha observat experimentalment que en cas d'augmentar el valor de la k la precisió del model únicament s'estabilitza; experimentalment s'ha trobat que el valor utilitzat, $k = 5$ és prou bo per a fer els càlculs i triga poc en obtenir-los.

4. **És viable o convenient aplicar *LeaveOneOut*?** No és viable ja que el nostre dataset no és binari i existeixen moltes similituds entre les diverses classes, pel que si només es testeja amb una dada el classificador tendirà a fer *overfitting* del train i no podriem tampoc assegurar una bona classificació de les dades.

5.5 Apartat B.5

1. **A teoria, hem vist el resultat d'aplicar el *accuracy_ score* sobre dades no balancejades. Podríeu explicar i justificar quina de les següents mètriques será la més adient pel vostre problema? *accuracy_ score*, *f1_ score* o *average_ precision_ score***. S'ha utilitzat *accuracy_ score* per els següents motius:

   (a) Les classes són equiprobables pel que no cal tenir en compte la distribució de les mateixes en el dataset.

   (b) L'objectiu del classificador és etiquetar el més bé possible totes les classes, pel que hem de contar només els casos correctes respecte al total (definició *accuracy_ score*).

2. **Mostreu la Precisió-Recall Curve i la ROC Curve. Quina és més rellevant pel vostre dataset? Expliqueu amb les vostres paraules, la diferència entre una i altre.** Les gràfiques tant de la Precisió-Recall Curve com de la ROC Curve de cada model es troben als apartats explicats anteriorment 4.4.3, 4.2.3, 4.1.3, 4.5.3, 4.3.3.
   En el nostre cas, és més important la corva ROC, degut a que el nostre dataset està balancejat i a que és més util quan es volen classificar diverses classes en representar de millor manera la precisió del mètode sense haver de binaritzar l'output.

3. **Què mostra *classification_ report*? Quina métrica us fixareu per tal de optimitzar-ne la classificació pel vostre cas?** Per fer la comparativa dels mètodes mitjançant la *classification_ report* utilitzarem la mètrica *f1_ score*, ja que relaciona la *recall* amb la *precision*, que són els dos paràmetres que s'utilitzen per a la representació de la *ROC curve*.

5.6 Apartat B.6

1. **Quines formes de buscar el millor paràmetre heu trobat? Són costoses computacionalment parlant?** Hem utilitzat un algoritme de tipus *Backtracking* que és força costós computacionalment, aquesta decisió ha sigut presa degut a que altres mètodes ja implementats en la llibreria *sklearn* que fan aquesta cerca dels hiperparàmetres, com pot ser el *GridSerachCV*, triguen massa en obtenir la millor combinació de paràmetres possible. De la nostra manera, si bé es més tediós, obtenim mica en mica la millor combinació i podem anant estudiant els paràmetres que obtenim per veure si té sentit o si podem utilitzar algún altre algoritme que s'ajusti millor.

2. **Si disposem de recursos limitats (per exemple, un PC durant 1 hora) quin dels dos mètodes creieu que obtindrà millor resultat final?** Seria bastant més eficaç utilitzar el *GridSearchCV* ja que la implementació propia de l'algoritme de cerca es bastant ineficient.

3. **Existeixen altres mètodes de búsqueda més eficients (*scikit-optimize*)?** Existeixen el *BayessianSearchCV* i el *Tunning* de *skopt* entre altres que són més eficients. S'ha testejat el *GridSearchCV* i el *BayesianSearchCV* en el codi entregat tot i que no s'han utilitzat els seus resultats a l'hora de trobar el millor classificador (ja que amb el nostre propi algoritme obteníem precisions semblants amb un temps d'execució relativament baix) i no s'ha inclòs l'ouput del programa quan s'executava el fragment de codi mencionat degut a la redundància dels resultats obtinguts amb el nostre propi mètode.

6 Conclusions

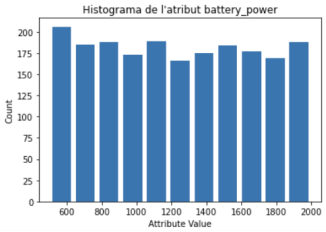
Concluïm el proces mostrant una taula de eficiencies (model i precisió) amb els millors models obtinguts.

model	**precisió**
KNN	82.5%
SVC	84%
RandomForest 82.75%	81.5% Logistic
SVC$_l inear$	76.25%
DecisionTree	81.5%

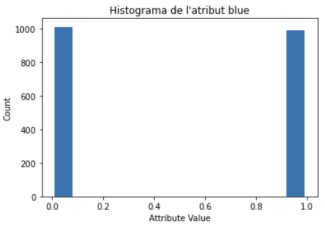
Concluïm doncs que el millor model és el SVC amb els hiperparàmetres explicats a 4.3.4. Tot i així, també es recomana utilitzar el Logistic en ja que no hem obtingut resultats amb tanta variança.

7 Annex

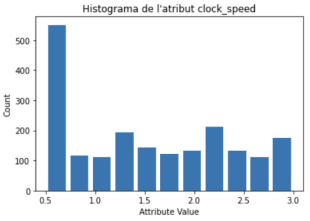
7.1 Histogrames



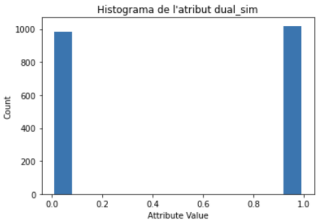
(a) *Battery Power*



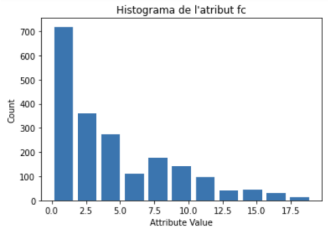
(b) *Blue*



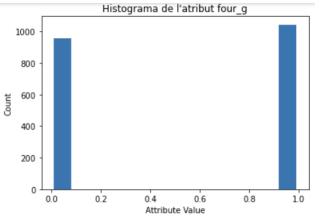
(c) *Clock Speed*



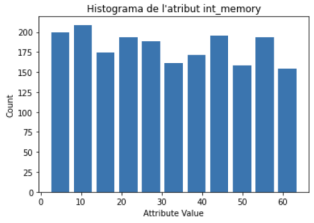
(a) *Dual Sim*



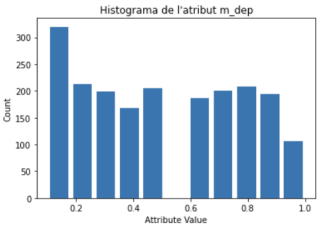
(b) *Frontal Camera*



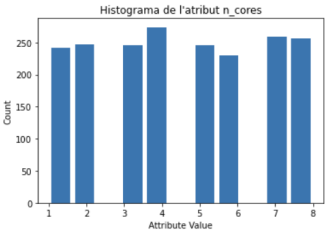
(c) *4G*



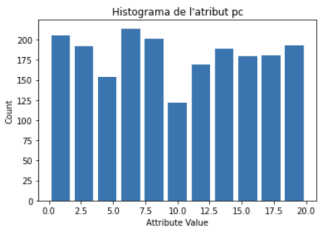
(a) *Intern Memory*



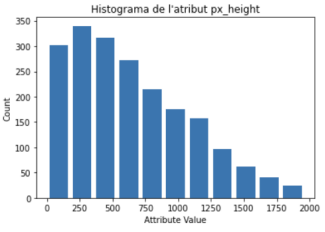
(b) *Mobile Depth*



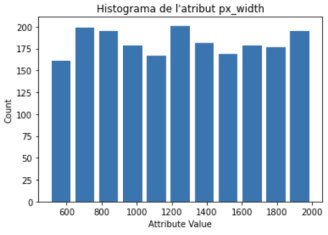
(c) *Number Cores*



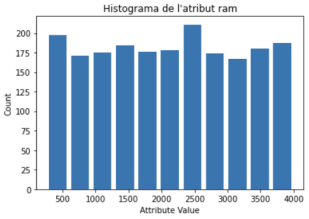
(a) *PC*



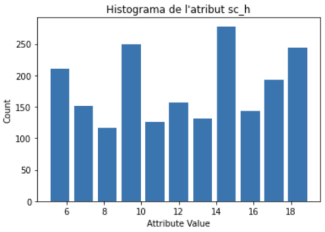
(b) *Px Height*



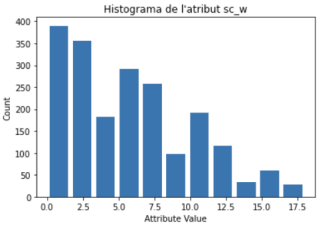
(c) *Px Width*



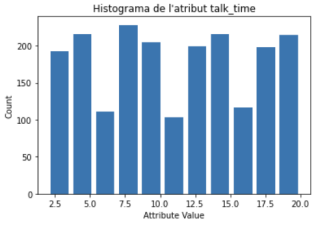
(a) *RAM*



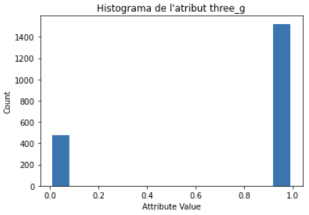
(b) *Screen Height*



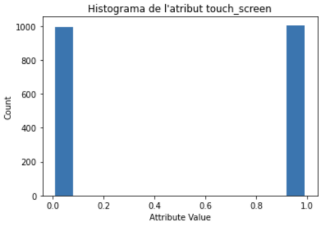
(c) *Screen Width*



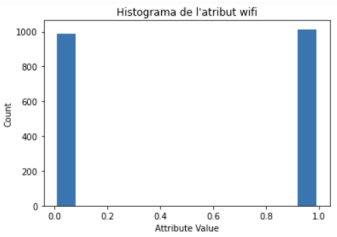
(a) *Talk Time*



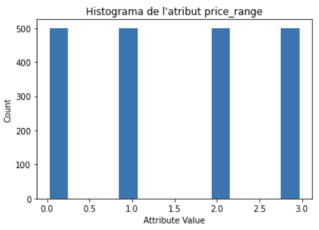
(b) *3G*



(c) *Touch Screen*

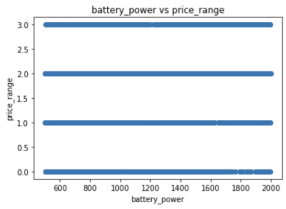


(a) *Wifi*

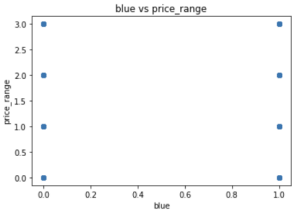


(b) *Price Range*

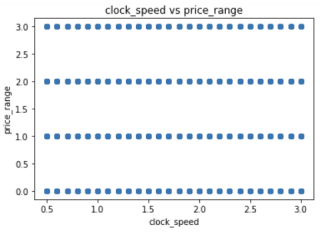
7.2 Correlacions amb *price_range*



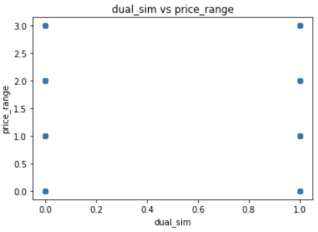
(a) *Battery Power*



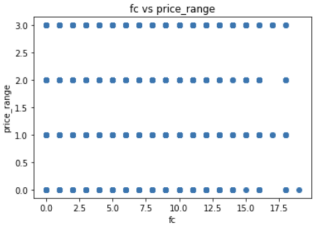
(b) *Blue*



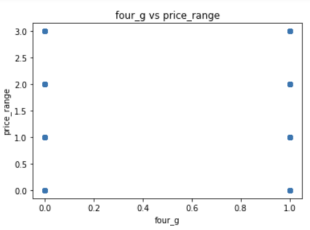
(c) *Clock Speed*



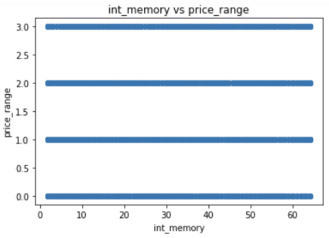
(a) *Dual Sim*



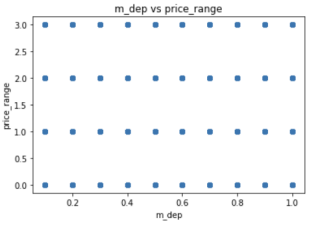
(b) *Frontal Camera*



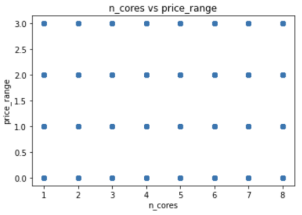
(c) *4G*



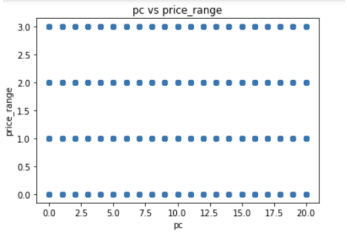
(a) *Intern Memory*



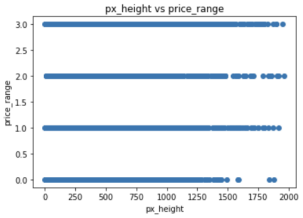
(b) *Mobile Depth*



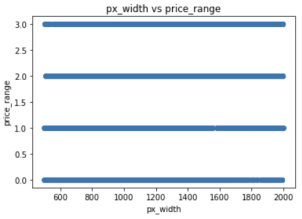
(c) *Number Cores*



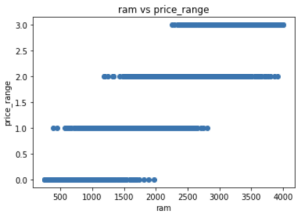
(a) *PC*



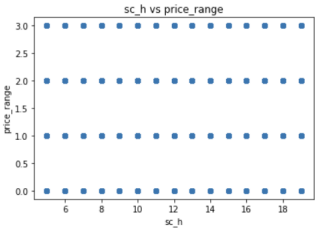
(b) *Px Height*



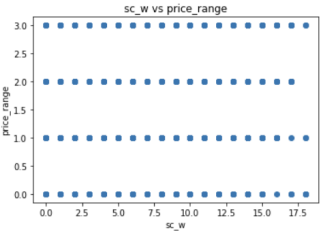
(c) *Px Width*



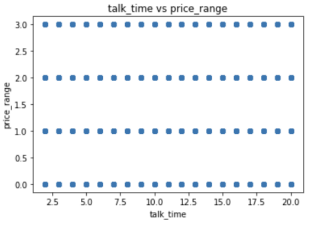
(a) *RAM*



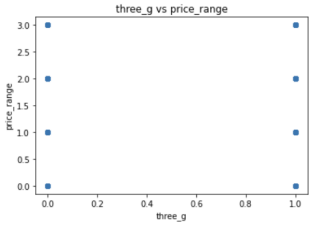
(b) *Screen Height*



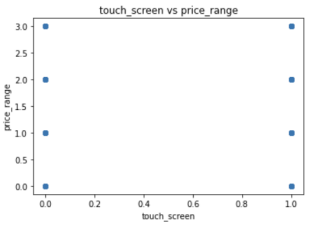
(c) *Screen Width*



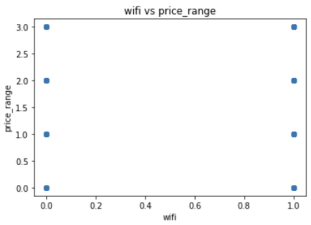
(a) *Talk Time*



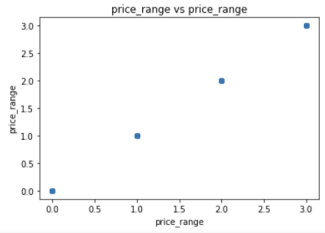
(b) *3G*



(c) *Touch Screen*



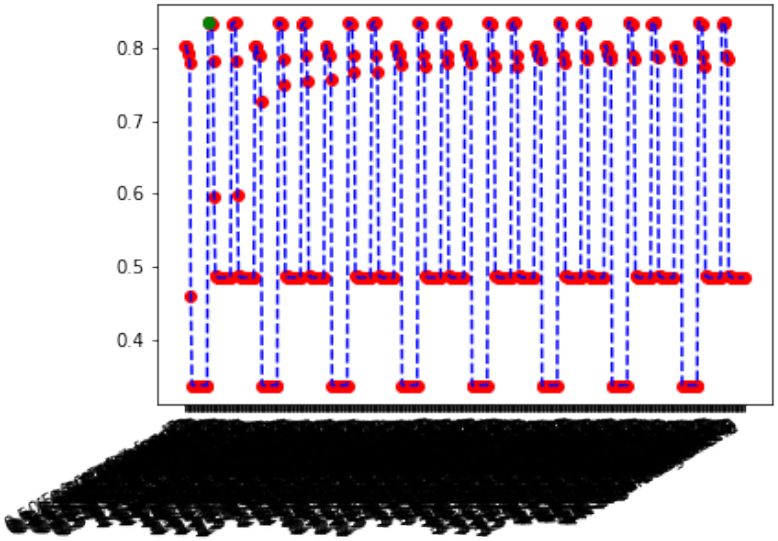
(a) *Wifi*



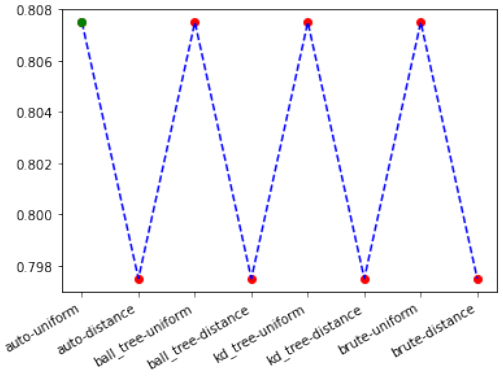
(b) *Price Range*

7.3 Cerca hiperparàmetres

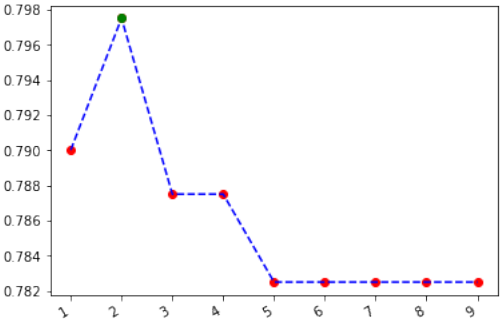
7.3.1 Logistic Regressor



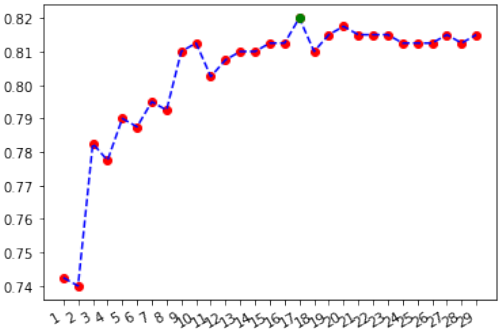
7.3.2 K-Nearest Neighbors



Búsqueda dels millors *weights* i *algorithm*

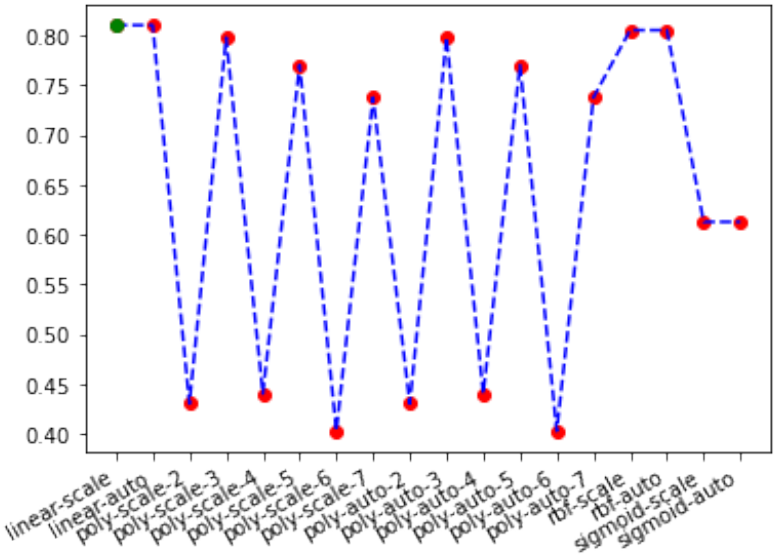


Búsqueda de la millor p

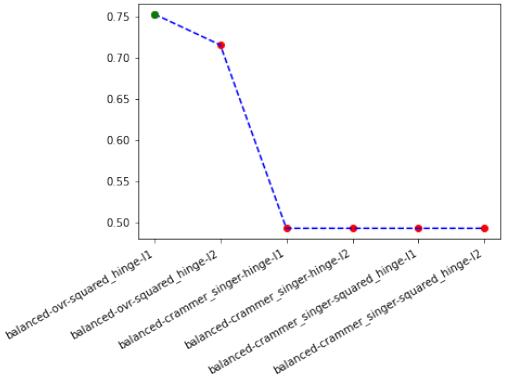


Búsqueda del millor $n\_neighbors$

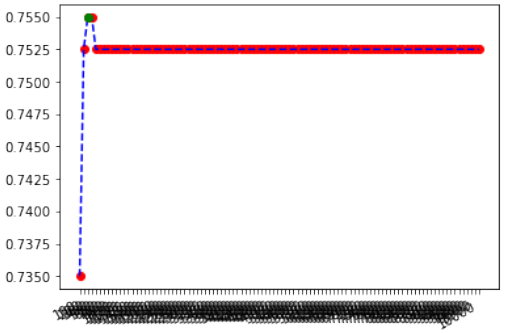
7.3.3 Support Vector Classification



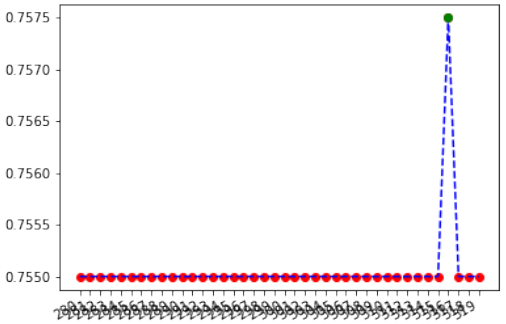
7.3.4 Linear Support Vector Classification



Búsqueda dels millors *class_ weight*, *multi_ class*, *loss* i *penalty*

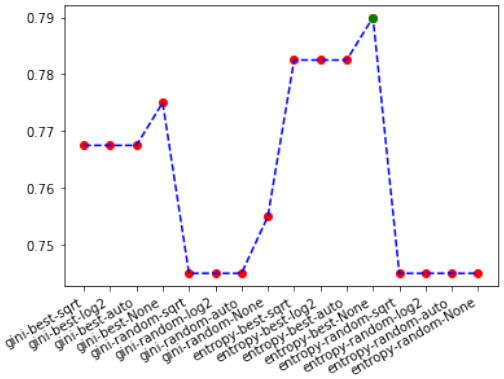


Búsqueda del millor *max_ iter*

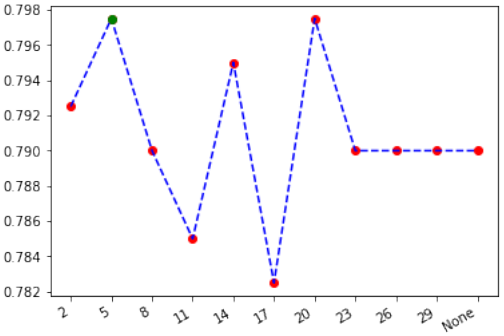


Búsqueda de *max_ iter* acotada

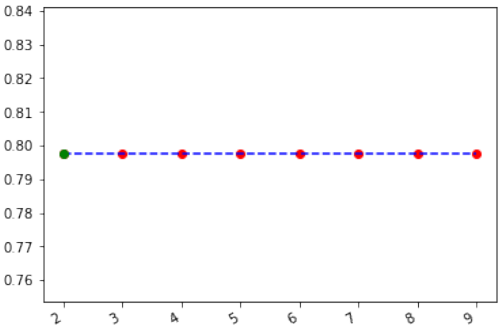
7.3.5 Decision Tree Classificator



Búsqueda dels *criterion*, *max_features* i *splitter*

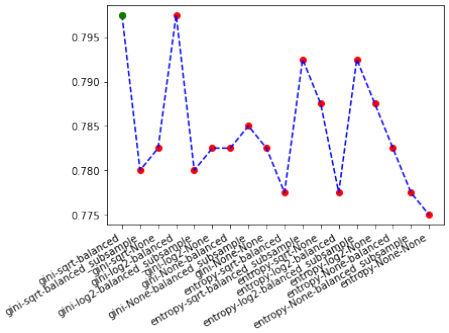


Búsqueda del millor *max_depth*

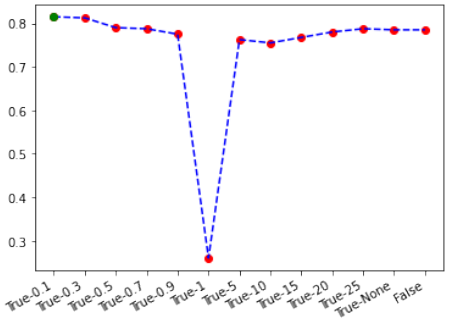


Búsqueda del millor *min_samples_split*

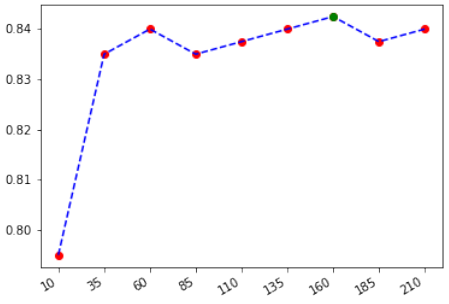
7.3.6 Random Forest Classificator



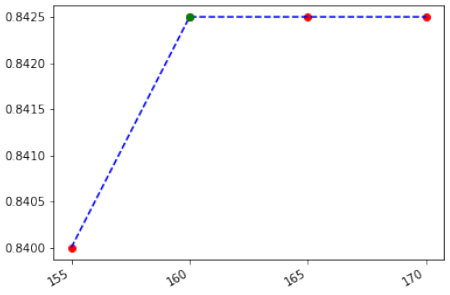
Búsqueda dels *criterion*, *max_features* i *class_weight*



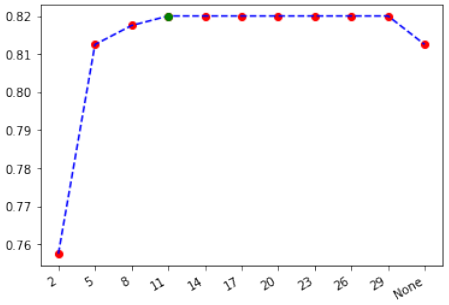
Búsqueda del millor *max_samples*



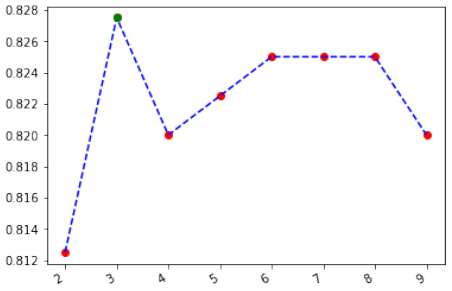
Búsqueda del millor *n_estimators*



Búsqueda del millor *n_estimators* acotada

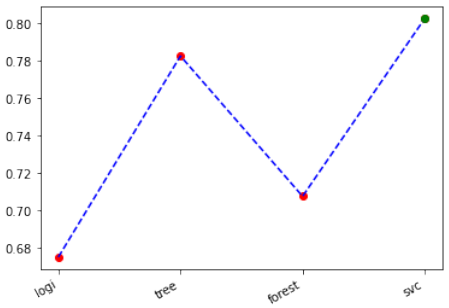


Búsqueda de la millor *max_depth*



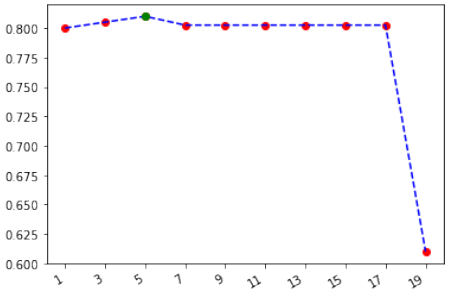
Búsqueda del millor *min_samples_split*

7.3.7 AdaBoost Classificator

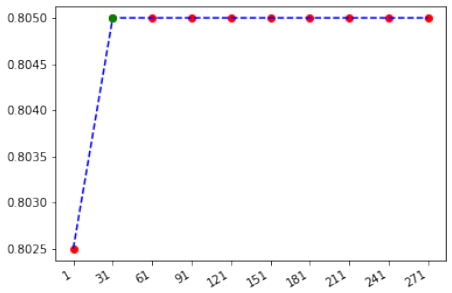


Búsqueda del millor *base_ estimator*

Búsqueda del millor *algorithm*



Búsqueda del millor *learning_ rate*



Búsqueda del millor *n_ estimators*