

APC: PRACTICA 2

CLASSIFICACIÓ



**Universitat Autònoma
de Barcelona**

Ivan Peñarando Martínez - 1599156
Joel Marco Quiroga Poma - 1504249
Ferran Martínez Reyes - 1565491

12 d'Octubre de 2022

GPA604-1530

GITHUB: [teolicht505/APC-Prac2 \(github.com\)](https://github.com/teolicht505/APC-Prac2)

INDEX

APARTAT B	3
1. EDA (exploratory data analysis)	3
2. Preprocessing (normalization, outlier removal, feature selection..)	4
3. Model Selection	6
4. Crossvalidation	7
5. Metric Analysis	8
6. Hyperparameter Search	10
APARTAT A	11

APARTAT B

1. EDA (exploratory data analysis)

Preguntes:

- **Quants atributs té la vostra base de dades?**

La base de dades té en total 21 atributs. On 1 d'ells correspon a un identificador per cada fila del dataset.

- **Quin tipus d'atributs tens? (Numèrics, temporals, categòrics, binaris...)**

Els nostres datasets tenen atributs enters i decimals, que els podem dividir en diversos tipus:

- Atributs categòrics, com podria ser el “price_range” que categoritza les gammes de preus on es troben els diferents telèfons.
- Atributs binaris: Són valors que indican si una cosa o bé està no, com podria ser el wifi (0 si no té, 1 si té), el 4G...

- **Com és el target, quantes categories diferents existeixen?**

L'únic atribut que pren valors que poden ser classificables com a categories és l'atribut price_range, a més a més, és un atribut prou interessant per fer la classificació ja donat un telèfon amb x característiques seria molt útil saber a quina gamma de telèfon pertany (gamma preu baix, gamma preu alt...)

Les categories de price_range són:

0 (low cost), 1 (medium cost), 2 (high cost) and 3 (very high cost)

- **Podeu veure alguna correlació entre X i Y?**

L'única correlació de price_range que trobem en fer un heatmap de les correlacions és amb l'atribut RAM (Amb una correlació del 92%). En canvi, les correlacions amb els altres atributs són gairebé inexistentes (20% o inferiors)

- **Estan balancejades les etiquetes (distribució similar entre categories)? Creus que pot afectar a la classificació la seva distribució?**

Amb la instrucció `dataset.mean()` veiem que la mitjana de price_range de totes les files és 1.5 (Molt proper a 2, que és la mitjana d'una distribució balancejada) tot i així, al no ser perfecte, pot passar que la classificació doni més importància a una categoria que a una altra.

2. Preprocessing (normalization, outlier removal, feature selection..)

Preguntes:

- **Estàn les dades normalitzades? Caldria fer-ho?**

No, ja que cada atribut té diferents variàncies (valors en rangs diferents). Caldria fer una normalització de les dades, ja que el SVM assumeix que les dades que li passarem estan estandarditzades en un rang de 0 a 1 (o de -1 a 1).

- **En cas que les normalitzeu, quin tipus de normalització serà més adient per les vostres dades?**

La normalització més adient serà la normalització via escalat a un rang (scaling) que deixarà els atributs escalats a un rang (0, 1), ja que serà un requisit indispensable si volem fer un classificador amb màquines de vectors de suport SVM.

- **Teniu gaires dades sense informació? Els NaNs a pandas? Tingueu en compte que hi ha mètodes que no els toleren durant l'aprenentatge. Com afecta la classificació si les filtrem? I si les reompliu? Com ho faríeu?**

No tenim cap atribut que tingui dades NaN, ni al train dataset ni al test dataset.

```
Dataset train:  
battery_power      0  
blue                0  
clock_speed        0  
dual_sim            0  
fc                  0  
four_g              0  
int_memory          0  
m_dep               0  
mobile_wt           0  
n_cores             0  
pc                  0  
px_height           0  
px_width            0  
ram                 0  
sc_h                0  
sc_w                0  
talk_time            0  
three_g              0  
touch_screen         0  
wifi                0  
price_range          0  
dtype: int64
```

```
Dataset test:  
id                  0  
battery_power        0  
blue                0  
clock_speed          0  
dual_sim            0  
fc                  0  
four_g              0  
int_memory           0  
m_dep               0  
mobile_wt            0  
n_cores             0  
pc                  0  
px_height           0  
px_width            0  
ram                 0  
sc_h                0  
sc_w                0  
talk_time            0  
three_g              0  
touch_screen         0  
wifi                0  
dtype: int64
```

- Teniu dades categòriques? Quina seria la codificació amb més sentit? (OrdinalEncoder, OneHotEncoder, d'altres?)

Sí, price_range és l'únic atribut de tipus categòric. La codificació amb més sentit seria un OrdinalEncoder, ja que en ser valors amb una relació ordinal seria la codificació més simple i suficient a aplicar.

- Caldria aplicar `sklearn.decomposition.PCA`? Quins beneficis o inconvenients trobarieu?

Caldria aplicar PCA en cas que el nostre model final tingui tantes dimensions que sigui impossible la seva visualització. Mentre que estaríem perdent algunes característiques amb la reducció de dimensions, aconseguiríem aquesta visualització de les nostres dades.

En el nostre cas, al tenir en compte solament l'atribut "ram" molt correlacionat (0.92) amb el target "price_range" no ens farà falta la PCA, ja que podrem visualitzar aquest espai sense cap problema. Els altres atributs no valdrien molt la pena per classificar amb correlacions extremadament baixes 0.04 - 0.2

- Es poden aplicar `PolyomialFeatures` per millorar la classificació? En quins casos té sentit fer-ho?

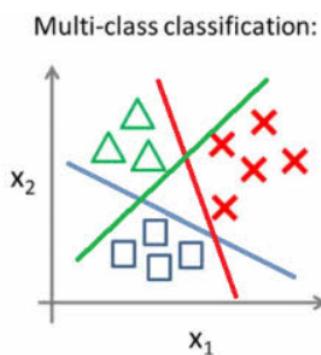
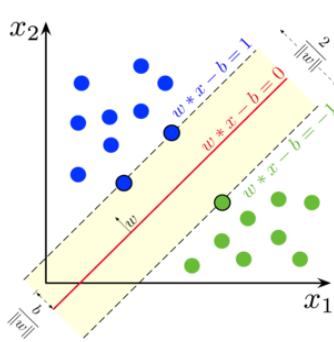
En principi podríem aplicar PolyomialFeatures en el cas que les nostres features no descriguin suficientment bé una bona classificació (es produueixi underfitting o overfitting) llavors en casos on això ocorri seria interessant provar un model on fem que els features (atributs predictors) passin a ser polinomials.

3. Model Selection

Preguntes:

- Quins models heu considerat?

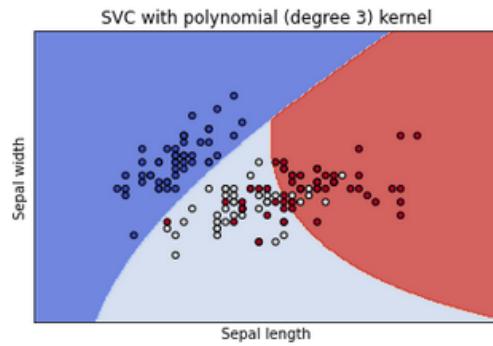
Per a fer una bona classificació hem considerat els models de les màquines de vectors de suport (SVM) i el descens de gradient aplicat als regressors logístics.



En tots dos models, farem ús de la multiclass classification, és a dir, One vs. Rest, ja que el nostre target “price_range” és multicategoría. Posarem cada categoria en contra de totes les altres per tal de diferenciar-les.

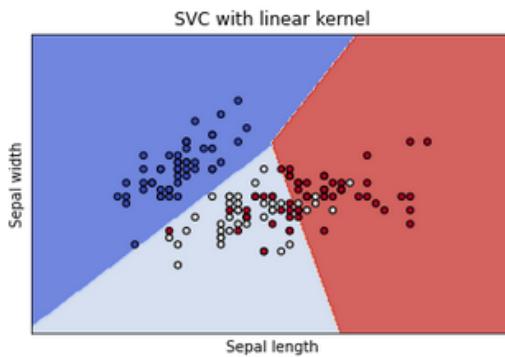
- **Considereu les SVM amb els diferents kernels implementats. Quin creieu que serà el més precís?**

El SVM de tipus polinomial serà el més precís, ja que no està restringit a definir un “límit” lineal entre classes (on es produiria underfitting), sinó que deixa marge per a més flexibilitat per tal de buscar el millor classificador, això si, sense passar-se, ja que sinó es produiria l’efecte contrari (overfitting)



- **Quin serà el més ràpid?**

Gràcies a la seva simplicitat, el SVC de tipus linear serà el més ràpid, ja que els límits entre classes són simples rectes i no polinomis



- **Seria una bona idea fer un ensemble? Quins inconvenients creieu que pot haver-hi?**

Un ensemble es tracta de la unió de molts predictors que per si sols no són molt bons classificadors, però que en ajuntar-los tots a la vegada es poden aconseguir molt bones classificacions.

Tot i així, de vegades no podria ser convenient l'ús d'aquests pel seu ús tan gran de memòria i temps. En fer tants classificadors, estarem ocupant molt espai i no solament això sinó que hem d'entrenar-los, cosa que allargaria molt el procés d'aconseguir una bona classificació en casos en els quals potser aconsegueixes un millor resultat solament amb un classificador simple com SVM o el regressor logístic.

4. Crossvalidation

Preguntes:

- **Per què és important cross-validar els resultats?**

La cross-validation és important per avaluar si el nostre model de classificació està fent bé la seva feina. Podem mirar el nombre de resultats que són encerts, els resultats que són falsos positius, i els que són falsos negatius, i a partir d'això treure'n les conclusions de si s'està classificant correctament o si, per contra, hi ha masses falsos positius o falsos negatius d'una classe.

- **Separar la base de dades en el conjunt de train-test. Com de fiables seran els resultats obtinguts? En quins casos serà més fiable, si tenim moltes dades d'entrenament o poques?**

Haurem de trobar una bona proporció per tal de fer la divisió train-test, seguint els següents passos:

- No podem tenir pocs valors del train set perquè en el moment en què vulguem validar el nostre classificador potser no fa la seva tasca bé al no haver tingut suficients valors.
- Tampoc podem assignar quasi tot a train, ja que quedaran pocs valors per tal de validar que els nostres resultats siguin realment correctes i fins i tot podríem ajustar massa bé les dades i que amb futures dades la classificació no es pugui fer de manera òptima.

En el cas general, una bona proporció podria ser la de 80% per les dades de train i 10% tant per les dades test com pel validation set. Tot i així, també es poden provar diverses proporcionalitats bones de classificació com un 70:30, 60:40...

- **Quin tipus de K-fold heu escollit? Quants conjunts heu seleccionat (quina K)? Com afecta els diferents valors de K?**

En el nostre cas, com la nostra base de dades és bastant gran, hem decidit no aplicar LeaveOneOut, que consistiria a fer que la K sigui igual al nombre d'elements n. Si no tinguéssim tants valors, podríem permetre's aplicar aquest mètode sense que el seu cost computacional en temps sigui massa gran. Per contra, hem decidit agafar K = 5.

Si tenim un valor de K més petit estarem apartant molts valors per entrenament que poden ser valuoses, mentre que per valors més grans estarem fent més precís el nostre classificador a costa de més temps per generar-ho.

- És viable o convenient aplicar LeaveOneOut?

Direm que tenim una cross-validation de tipus LeaveOneOut si el K-fold és n, on n és el nombre de mostres de la base de dades.

Aquest tipus de validació no ens aportaria un millor rendiment que K-fold en bases de dades amb un gran nombre d'elements, ja que haurem de fer N iteracions, on N serà un nombre massa gran.

En el nostre cas, com que tenim moltes mostres en la base de dades (2001 mostres) i no és viable aplicar el LeaveOneOut.

5. Metric Analysis

Preguntes:

- A teoria, hem vist el resultat d'aplicar l'accuracy_score sobre dades no balancejades. Podríeu explicar i justificar quina de les següents mètriques serà la més adient pel vostre problema? accuracy_score, f1_score o average_precision_score.

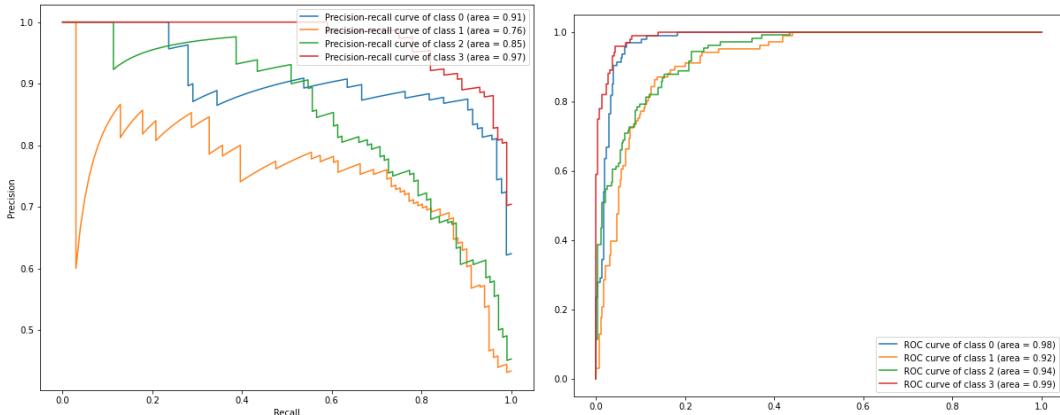
L'accuracy score representa el percentatge dels encerts totals que s'han fet, és a dir, representa la tassa de true positives més els true negatives entre tots els elements. Aquesta no és una bona mètrica per dades desbalancejades, ja que l'estimador tendirà a buscar classificar bé les dades amb menys elements per tal d'arribar a un accuracy més alt, perjudicant les altres.

Millors opcions pel nostre problema serien les mètriques f1 que té en compte el precision i el recall o bé les average precision, ja que tendirà a mostrar valors menys favorables que l'accuracy, ja que se centraran en els positius detectats i/o en les classificacions positives encertades.

RECOGNIZER OUTPUT		PEOPLE	NON-PEOPLE
CORRECT RECOGNITION	PEOPLE	True Positive	False Negative
	NON-PEOPLE	False Positive	True Negative

- 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 nostres Precision-Recall i ROC són les següents:



Precision-recall curve

ROC CURVE

La principal diferència entre aquestes dues corbes està en el fet que la precision-recall il·lustra una combinació de la ràtio de classificacions correctes encertades (precision) amb el ratio de tots els positius detectats (recall). Per altra part, el que la corba ROC té en compte són la combinació del recall o positius detectats amb el ratio de falsos negatius.

Pel nostre dataset, la més rellevant és sens dubte la del precision-recall. Això és perquè les nostres dades train estan desbalancejades. Cosa que es pot veure clarament amb la corba ROC que és massa bona, a diferència de l'altre corba que no surt tan bé. Serà més útil la funció ROC quan tinguem les dades més balancejades que no és el nostre cas, com vam respondre a la pregunta del subapartat 1 (EDA) les nostres dades ens donaven 1.5 de mitja quan hauria de donar 2 per ser perfectament balancejat.

- Què mostra [classification_report](#)? Quina mètrica us fixareu per tal d'optimitzar-ne la classificació pel vostre cas?

`Classification_report` mostra una sèrie de mostres rellevants per saber si el nostre classificador funciona de la manera correcta. Per a cada classe ens mostra mètriques com la “precision” (qualitat en reconèixer true positives), “recall” (eficiència reconeixent a tots els true positives), “f1-score” (mitja balancejada de “precision” i “recall”) i els pesos “support”.

A més també se'n calcula la mitjana amb “micro average” (true positives, false negatives i false positives) o “macro average” (la mitjana sense tenir en compte els pesos), a més del weighted average que inclourà els diferents pesos.

Per l'optimització del nostre classificador, caldrà tenir en compte la mètrica “f1-score”, ja que és la que ens engloba el “precision” i “recall” aportant una bona distinció entre una classificació bona i altre no tan bona.

6. Hyperparameter Search

Preguntes:

- Quines formes de buscar el millor paràmetre heu trobat? Són costoses computacionalment parlant?

Grid Search Exhaustiu (Exhaustive Grid Search): Es tracta d'una cerca exhaustiva en un subconjunt dels hiperparàmetres del algorisme d'aprenentatge. Per cada hiperparàmetre es té en compte el cross validation en el training set, que serveix d'indicador de rendiment per a guiar la cerca. La seva complexitat computacional és la més elevada dels mètodes de cerca, ja que es basa en la força bruta

Random Search (Randomized Parameter Optimization): Molt més eficient que el grid search exhaustiu, especialment quan un nombre petit d'hiperparàmetres afecta el rendiment de l'aprenentatge. Es basa en la selecció a l'atzar de diferents combinacions de valors d'hiperparametres. És el mètode més àmpliament utilitzat en la cerca d'hiperparametres.

Successive Halving: Consisteix en procés de selecció entre combinacions de valors possibles dels paràmetres. Primer se seleccionen a l'atzar, després s'avaluen i finalment rebutgem la meitat amb pitjor puntuació, repetint fins que sol ens quedi una sola configuració dels paràmetres. Aquests seran els candidats classificats amb més puntuació en totes les iteracions. L'objectiu de Successive Halving és acabar al més aviat possible per tal de no perdre temps en configuracions. Té una complexitat computacional molt menor a la de grid search exhaustiu o random search

- Si disposem de recursos limitats (per exemple, un PC durant 1 hora) quin dels dos mètodes creieu que obtindrà millor resultat final?

Si el que ens interessa és acabar la cerca l'abans possible el millor serà usar el mètode de successive halving, que és molt més ràpid que no pas l'exhaustive grid search ni el random search, a més de ser una solució que funciona molt bé a l'hora de tenir en compte coses com l'escalabilitat, ja que té una complexitat computacional menor.

- Existeixen altres mètodes de cerca més eficients?

L'optimització bayesiana és un mètode més eficient a l'hora de cercar el millor paràmetre. Es tracta d'un mètode que utilitza el teorema de Bayes per trobar el mínim o màxim d'una funció. Aquesta funció ve mapejada pels valors dels hiperparametres. Resulta molt més útil quan tenim funcions complexes, amb molt soroll o que siguin cares d'avaluar. Es tracta d'avaluar iterativament configuracions d'hiperparametres basant-se en observacions anteriors.

APARTAT A

Comparativa de models

L'objectiu d'aquest apartat és aprendre a comparar el rendiment obtingut amb els diferents models, utilitzant les mètriques adequades: precision-recall, accuracy, corba ROC, F1 score, etc.

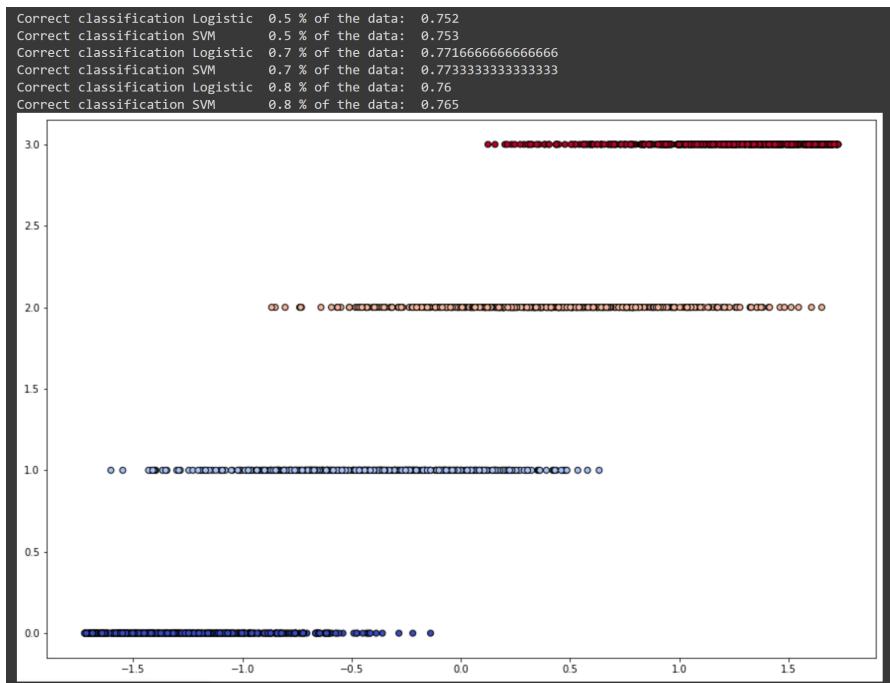
Es tracta d'ordenar per rendiment i descriure el perquè dels resultats obtinguts amb els mètodes utilitzats en el següent apartat B.

Amb l'heatmap hem vist que els atributs que tenen millor correlació amb el nostre target són “RAM” amb un 92% i “battery_power” amb un 20%.

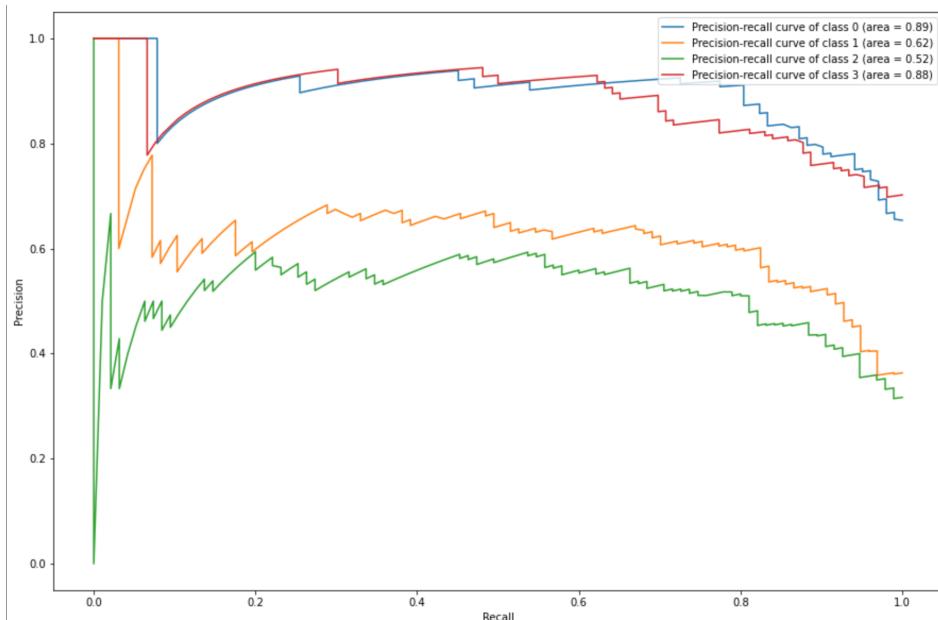
Classificació amb l'atribut “ram”:

Com a primer pas podem observar que l'atribut que té més correlació amb el target és “ram”.

En aquest cas farem els classificadors utilitzant només l'atribut “ram” i farem les corbes PR i ROC segons el model SVC.



Corba Precision-Recall:



Es pot veure uns valors de la corba PR una mica desbalancejades per cada classe. tenim uns valors de l'àrea per cada classe de:

classe 0: 0.89

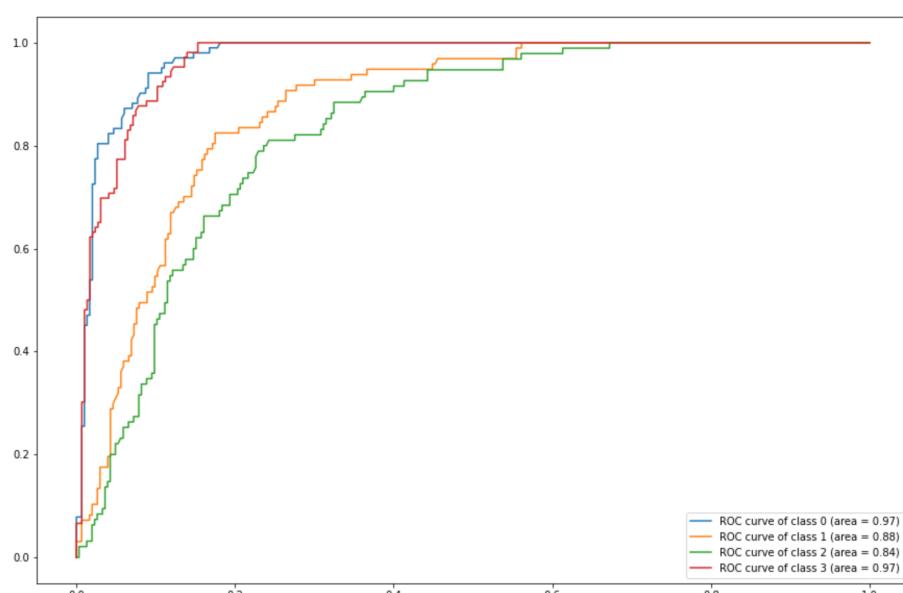
classe 1: 0.62

classe 2: 0.52

classe 3: 0.88

són valors bons obtinguts només per un atribut.

Corba ROC:



Els valors de la corba ROC per cada classe:

classe 0: 0.97

classe 1: 0.88

classe 2: 0.84

classe 3: 0.97

els valors són bons, això és degut al fet que hi ha pocs valors positius a la nostra base de dades.

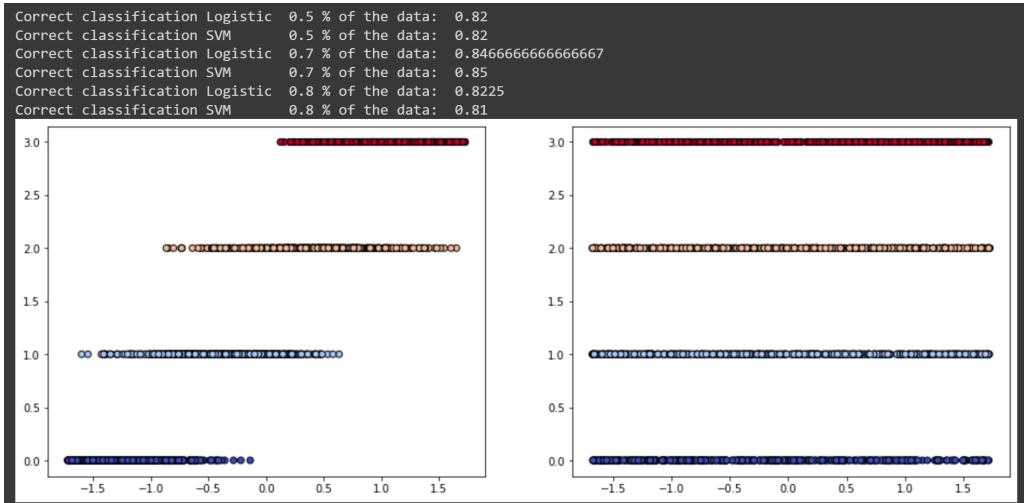
Scores dels models amb K-fold:

Amb un entrenament k-fold obtenim els següents scores:

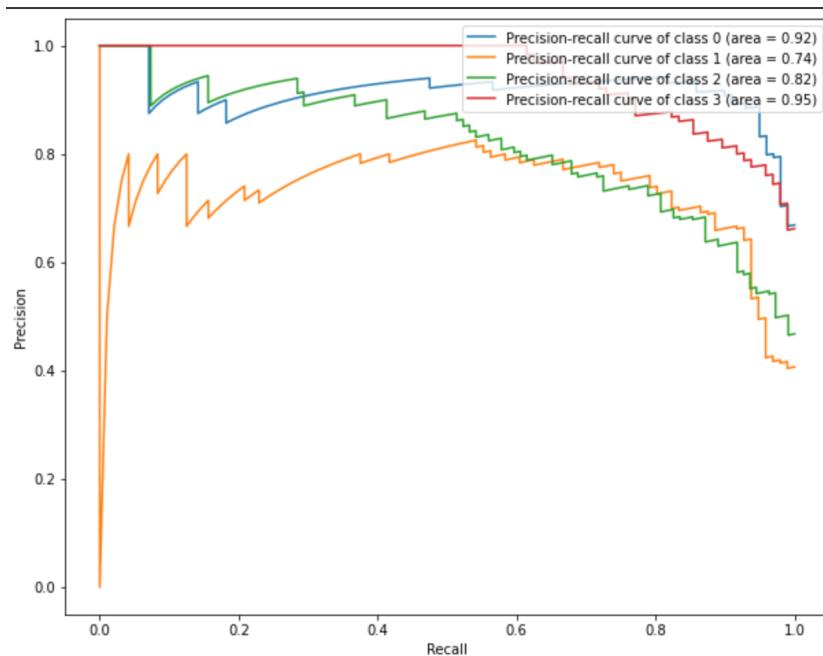
```
fold 0 : fit_time : 0.18
fold 0 : score_time : 0.02
fold 0 : test_accuracy : 0.76
fold 0 : test_precision : 0.76
fold 0 : test_recall : 0.76
fold 0 : test_f1_score : 0.76
fold 1 : fit_time : 0.18
fold 1 : score_time : 0.02
fold 1 : test_accuracy : 0.77
fold 1 : test_precision : 0.77
fold 1 : test_recall : 0.77
fold 1 : test_f1_score : 0.77
fold 2 : fit_time : 0.18
fold 2 : score_time : 0.02
fold 2 : test_accuracy : 0.76
fold 2 : test_precision : 0.76
fold 2 : test_recall : 0.76
fold 2 : test_f1_score : 0.76
fold 3 : fit_time : 0.19
fold 3 : score_time : 0.02
fold 3 : test_accuracy : 0.75
fold 3 : test_precision : 0.75
fold 3 : test_recall : 0.75
fold 3 : test_f1_score : 0.75
fold 4 : fit_time : 0.18
fold 4 : score_time : 0.02
fold 4 : test_accuracy : 0.74
fold 4 : test_precision : 0.75
fold 4 : test_recall : 0.74
fold 4 : test_f1_score : 0.74
```

Classificació amb l'atribut “ram” i l'atribut “battery_power”:

Encara que l'atribut “battery_power” tingui una correlació amb el nostre target d'un 20%, molt diferent de “ram” que té un 92% com es pot observar al gràfic, l'agafem també i fem els models amb aquests 2 atributs.



Corba Precision-Recall:



Tenim uns valors de l'àrea per cada classe de:

classe 0: 0.92

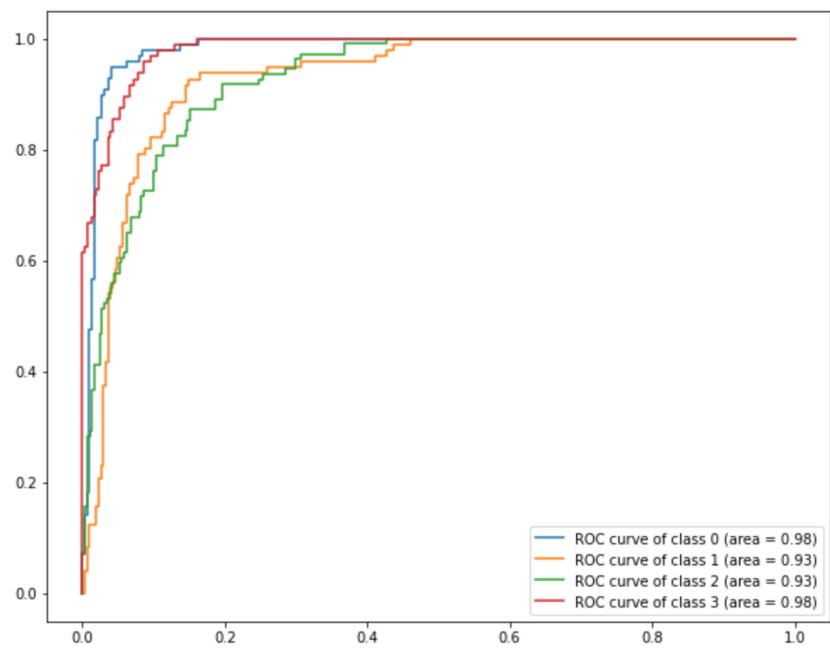
classe 1: 0.74

classe 2: 0.82

classe 3: 0.95

Els valors han millorat respecte a agafar només 1 atribut.

Corba ROC:



Els valors de la corba ROC per cada classe:

classe 0: 0.98

classe 1: 0.93

classe 2: 0.93

classe 3: 0.98

Els valors, al igual que la corba PR, també han millorat una mica.

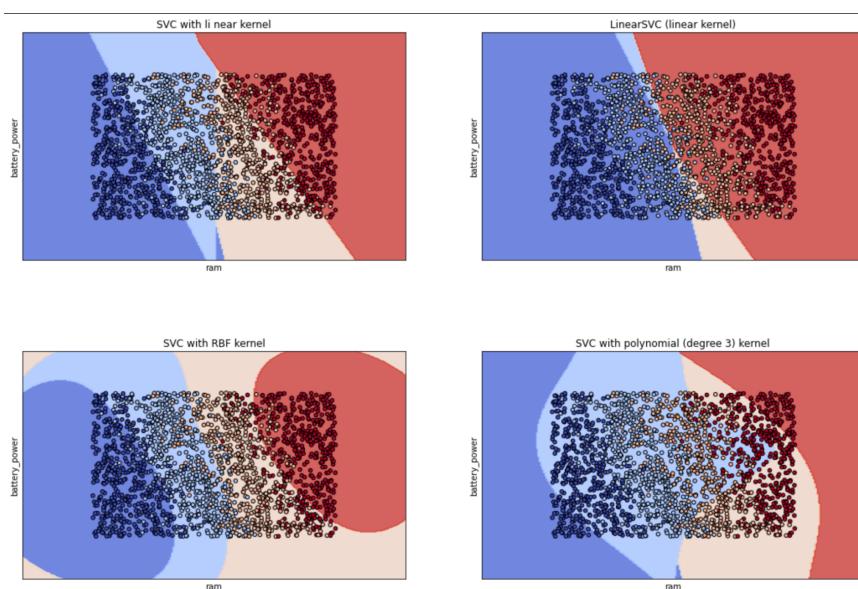
Scores dels models amb K-fold:

Amb un entrenament k-fold amb els 2 atributs obtenim els següents scores:

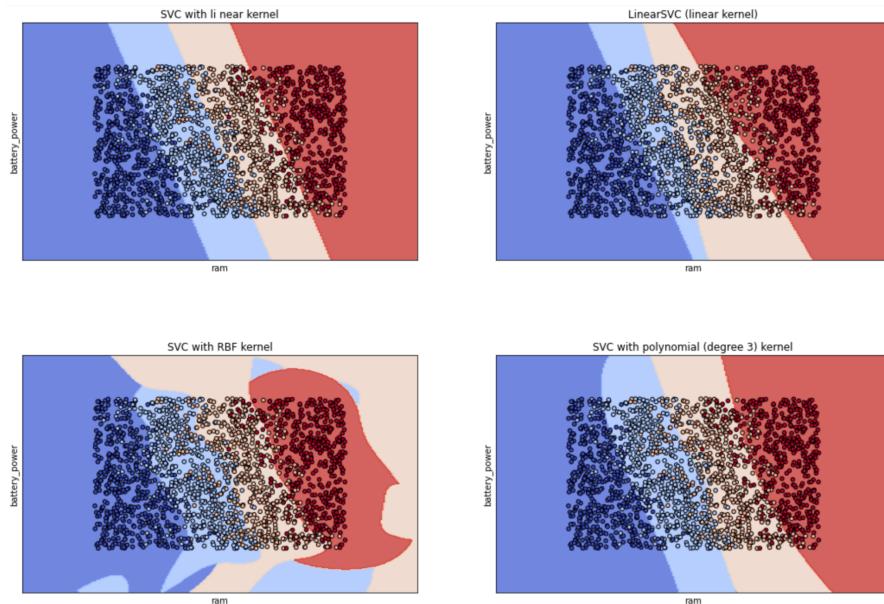
```
fold 0 : fit_time : 0.14
fold 0 : score_time : 0.01
fold 0 : test_accuracy : 0.81
fold 0 : test_precision : 0.82
fold 0 : test_recall : 0.81
fold 0 : test_f1_score : 0.82
fold 1 : fit_time : 0.13
fold 1 : score_time : 0.02
fold 1 : test_accuracy : 0.81
fold 1 : test_precision : 0.81
fold 1 : test_recall : 0.82
fold 1 : test_f1_score : 0.81
fold 2 : fit_time : 0.14
fold 2 : score_time : 0.01
fold 2 : test_accuracy : 0.83
fold 2 : test_precision : 0.83
fold 2 : test_recall : 0.84
fold 2 : test_f1_score : 0.83
fold 3 : fit_time : 0.14
fold 3 : score_time : 0.01
fold 3 : test_accuracy : 0.82
fold 3 : test_precision : 0.82
fold 3 : test_recall : 0.82
fold 3 : test_f1_score : 0.82
fold 4 : fit_time : 0.13
fold 4 : score_time : 0.01
fold 4 : test_accuracy : 0.82
fold 4 : test_precision : 0.82
fold 4 : test_recall : 0.82
fold 4 : test_f1_score : 0.82
```

Gràfics d'efecte segons el canvi de valors del paràmetre de regularització:

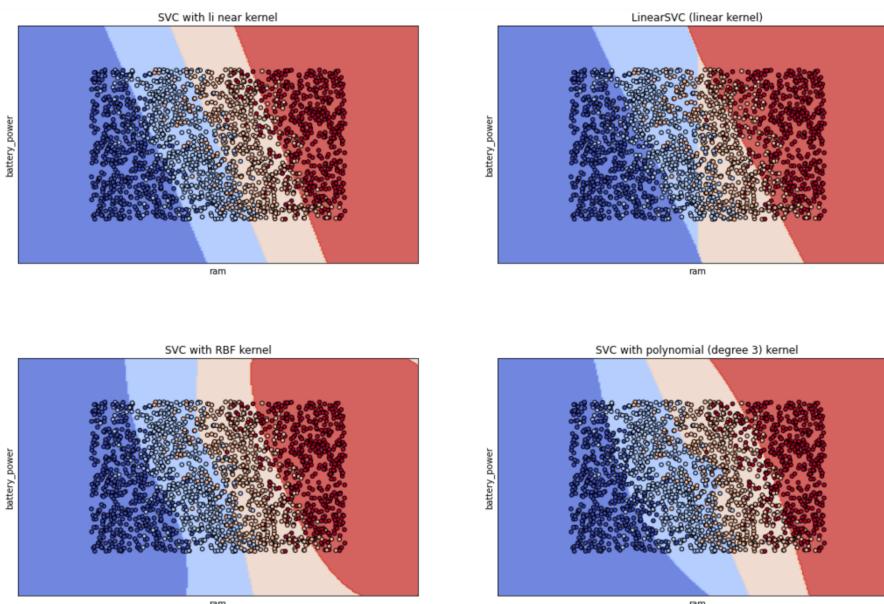
Gràfic amb paràmetre de regularització “C” petit (0.00001).



Gràfic amb paràmetre de regularització “C” gran (1000).



Gràfic amb paràmetre de regularització “C” amb valor 0.1.



Sabem que C és inversament proporcional a la força de regularització, i, per tant, amb un valor de C molt gran fem una penalització molt petita i en conseqüència el model no termina d'estar totalment ajustat, i amb un valor de C petita fem una força molt gran i ens arrisquem a què el model no serveixi per generalitzar (overfitting).