Matemàtica Computacional i Analítica de Dades

Aprenentatge Computacional

Pràctica 2: Classificació

Autors (NIU):

Àlex Correa Orri (1564967)

Júlia Pumares Benaiges (1566252)

Curs 2021-2022, 1r semestre

Índex

1 Repositori GitHub

https://github.com/AlexCorreaO/prac_2_APC

2 Apartat (B): Comparativa de models

2.1 Models

Hem provat diferents models per classsificar el datset IRIS. En cada execució ens sortien *scores* diferents. Els que veiem en la següent taula són els que ens han sortit en una certa execució. En cada execució ens sortien resultats diferents.

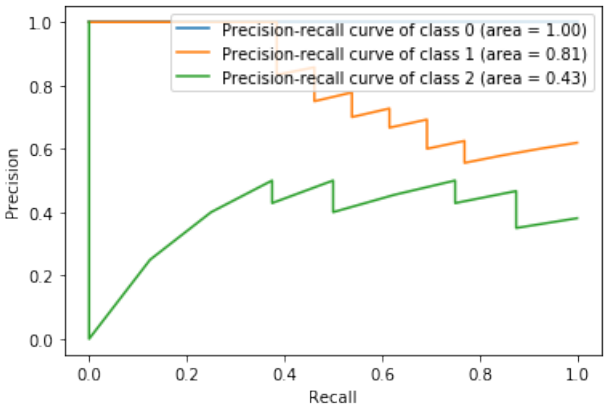
Mètode	**0.5**	**0.7**	**0.8**
SVM	0.85	0.76	0.77
KNeighbors	0.67	0.73	0.7
Decision Tree	0.63	0.8	0.7
Naive Bayes	0.85	0.78	0.73
Lineal	0.73	0.64	0.61
Logistic	0.67	0.78	0.5
Adaboost	0.76	0.38	0.7
Bagging	0.69	0.67	0.67
Perceptron	0.37	0.51	0.5

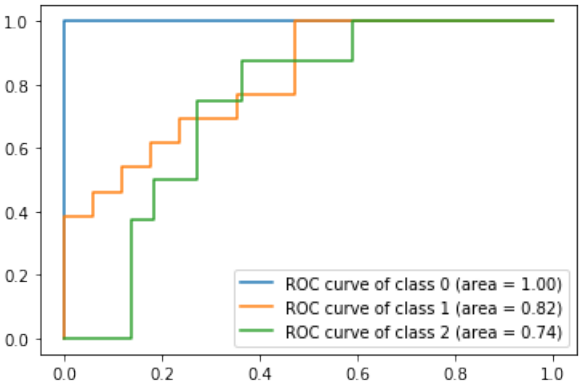
En la majoria de casos quan augmentem el nombre de dades train al 0.7% l'*accuracy* augmenta exepte en alguns casos com *SVM, Naive Bayes* i lineal respecte quan feiem servir 0.5%. Quan augmentem al 0.8% gaire bé sempre disminueix.

Això varia bastant en cada execució. No sempre ens surt el mateix mètode com a més precis. Ni el mateix model es comporta igual sempre al augmentar o disminur el nombre de dades train.

El mètode *Adaboost* és dels que va pitjor ja que en la majoria de les execucions ens surt per sota 0.4 en totes les particicions, exepte alguns casos que puja més. El *perceptron* també dona resultats bastant baixos.
Els que millor funcionarien són el *Naive Bayes, Decision Tree, SVM* i *KNeighbors.*

Posteriorment hem fet els gràfics de precision-recall per veure en quins casos funciona millor la classificació.





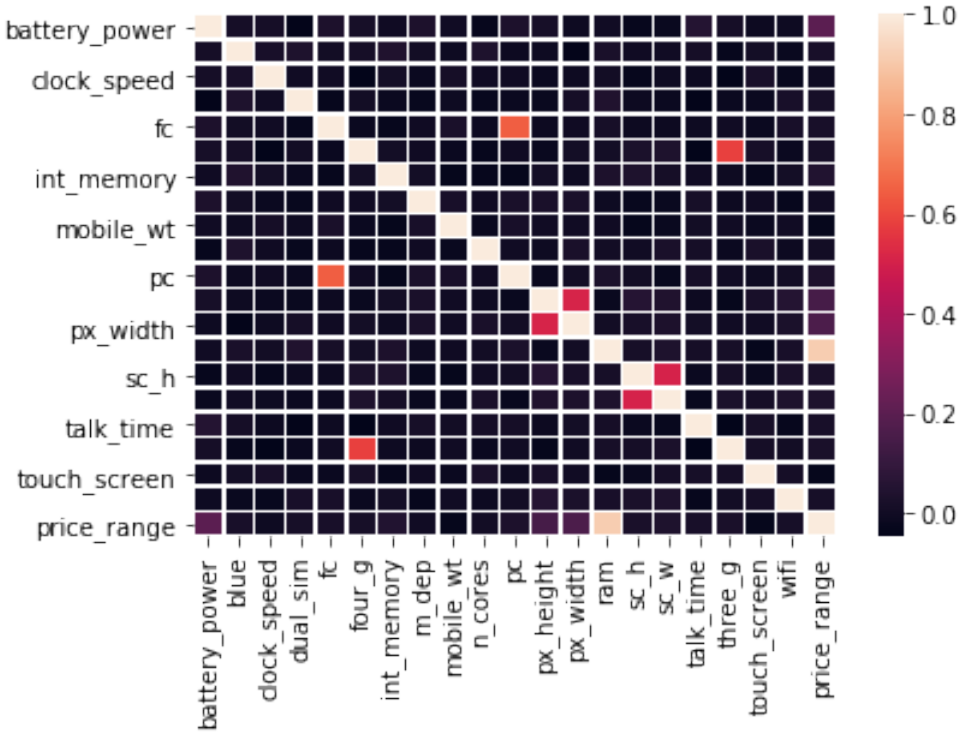
En funció del paràmetre c hem fet diferents proves amb el classificador SVC per veure com es comport millor i quines són les diferències. A l'arxiu notebook de python s'hi poden veure diferents proves.

3 Apartat (A): Classificació Numèrica

3.1 EDA

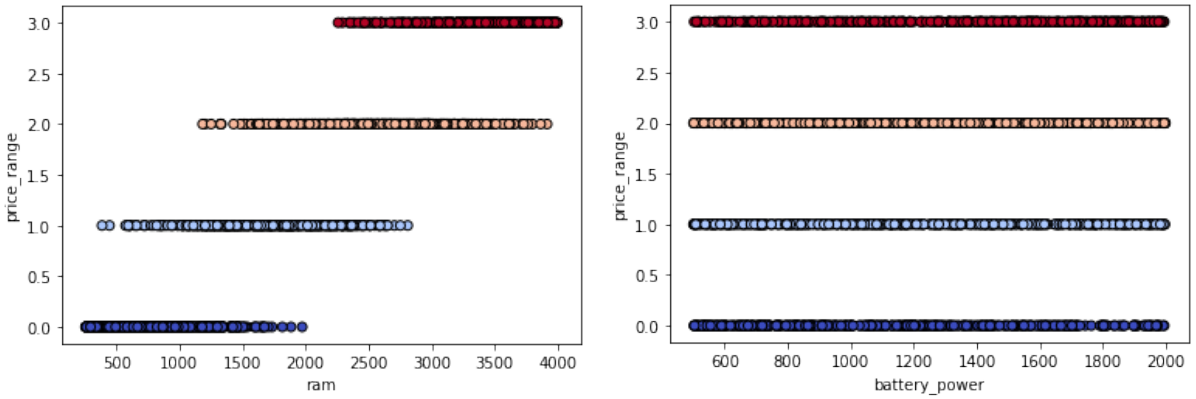
La nostra base de dades conté les dades de diferents mòbils i es vol classificar el mòbil en 4 grups diferents segons el rang en el que es troba el seu preu.

Tenim en total 20 atributs, dels quals 6 són binaris, i la resta són numèrics, n'hi ha de decimals i d'enters. El *price_range*, l'abtribut objectiu està balancejat.Tenim en total 2000 dades d'entrenamnent, 500 per cada classe.



Els atributs que estàn més correlacionats amb el *price_range* són els següents, amb les correlacions indicades (la resta d'atributs tenen una correlació més petita de 0.05):

- **px_height:** 0.149
- **px_width:** 0.166
- **battery_power:** 0.201
- **ram:** 0.917



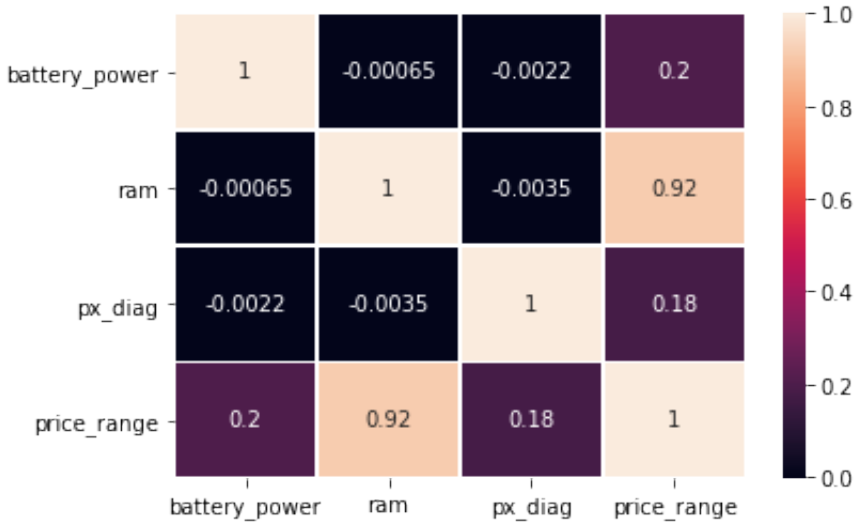
3.2 Preprocessing

No tenim cap dada sense informació (no tenim NaN).

No tenim les dades normalitzades així que hem considerat oportú estandaritzar les dades donat que hi havia dades de bateria (valors de l'escala dels centenars o milers) i dades de la quantitat de nuclis del processador (unitats o desenes). Hem definit una funció 'standarize' en que, per cada variable V, la recalculem per V = (V-mean(V))/S; On mean(V) és la mitjana de la variable V i S és la desviació estàndard. Considerem el conjunt de variables V el conjunt d'atributs del nostre dataset (sense incloure l'atribut objectiu). D'aquesta manera obtenim per a cada atribut estandaritat una mitjana igual (o molt propera; escala *10**(-16)) a 0 i una desviació estàndard igual (o molt propera) a 1.

Hem provat d'aplicar *polynomial features* i un *kernel pca* però no han fet que milloressin els resultats.

Tenim dos atributs que són l'amplada (*px_width*) i la llargada del mòbil (*px_height*). Amb aquest dos atributs podem calcular quan mesura la diagonal del mòbil (*px_diag*) . Així ens quedem amb aquest nou atribut i eliminem els dos anteriors, d'aquesta manera reduim la dimensionalitat de la base de dades. Finalment ens quedem només amb els atributs que tenen una major correlació amb el *price_range*. I la nostra X queda amb els següents atributs:



3.3 Model Selection

Hem provat diferents models fent servir les dades estandarditzades i sense estandarditzar. Ens sortien millors resultats estandarditzant.

També ens sortien una mica millors els resultats en alguns mètodes si agafàvem tots els atributs per la X (*DecisionTree* donava *score* 1), però això feia el programa bastant lent. Per això hem agafat el dataset

reduït a 3 atributs (explicat en l'apartat anterior) només per predir la y. Així ens surten resultats bons i ràpids.

En la següent taula podem veure els *scores* dels diferents mètodes que hem provat:

Mètode	**0.5**	**0.7**	**0.8**
SVM	0.91	0.93	0.94
KNeighbors	0.9	0.91	0.91
Logistic	0.85	0.84	0.84
Decision Tree	0.74	0.75	0.75
Perceptron	0.57	0.72	0.69
Bagging	0.72	0.71	0.33
Adaboost	0.84	0.7	0.78

En aquest cas en cada execució variaven una mica els valors però sempre ens més o menys sortien amb el mateix ordre.
Adaboost, Baggin i *Perceptron* són els que més variaven els resultats en cada execució i sengons el percentage de mostres train i per això els hem descartat.
La resta de mètodes mantenien la seva *accuracy* en les diferents execucions i particions.
Els mètodes que hem vist que funcionaven millor són el *SVM* i el *KNeighbors* amb un accuracy per sobre 0.9. Aquests seràn els models que farem servir en la resta d'apartats.

Comparant-ho amb els resultats que obteniem al fer servir tots els atribus hem observat que els models que ens han sortit com a millors amb només 3 atributs eren justament els que abans sortien com a pitjors. En canvi, amb el datset complet els millors mètodes eren el classificador *Decision Tree*, el regressor logístic i el *Bagging KNeighbors*.

3.4 Crossvalidation

Veient la inestabilitat dels mètodes que funcionen pitjor, per a trobar i confirmar els millors models utilitzem cross-validation en aquells que hem tingut millors resultats prèviament: SVM, KneighborsClassifier, LogisticRegression i BaggingClassifier.

A la següent taula podem veure com el Kneighbors i el SVM funcionen amb una precisió molt alta i molt constant (desviació estàndard baixa). Veiem com al augmentar el nombre de 'folds' obtenim una desviació estàndard major ja que augmenta la probabilitat d'obtenir valors diferents. Tot i això observem que segueixen sent precisions molt semblants i per tant el resultat és fiable. Hem utilitzat k = 5 (folds), i obtenim els resultats següents:

```
SVM:
Scores      :   [0.93   0.9225 0.9225 0.9225 0.9225]
Scores Mean :   0.924
Scores std  :   0.0030000000000000248

KNeighborsClassifier:
Scores      :   [0.905  0.91   0.9275 0.905  0.9075]
Scores Mean :   0.9109999999999999
Scores std  :   0.008455767262643871

LogisticRegression:
Scores      :   [0.845  0.8325 0.855  0.845  0.82  ]
Scores Mean :   0.8394999999999999
Scores std  :   0.012083045973594577

BaggingClassifier:
Scores      :   [0.6175 0.6525 0.7075 0.7225 0.5025]
Scores Mean :   0.6405000000000001
Scores std  :   0.07865112840894277
```

3.4.1 LeaveOneOut

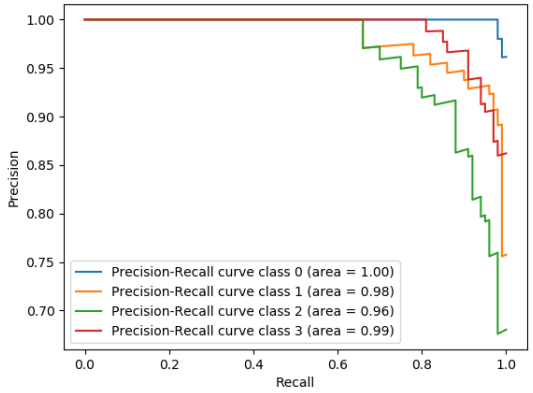
Hem intentat implementar el *LeaveOneOut* per no és viable ja que tarda molt temps. En cada partició tenim una dada de test i la resta d'entrenament, per tant haurem d'entranar tants models com dades tinguem. En el nostre cas hauríem d'entrenar per cada mètode 2000 models.

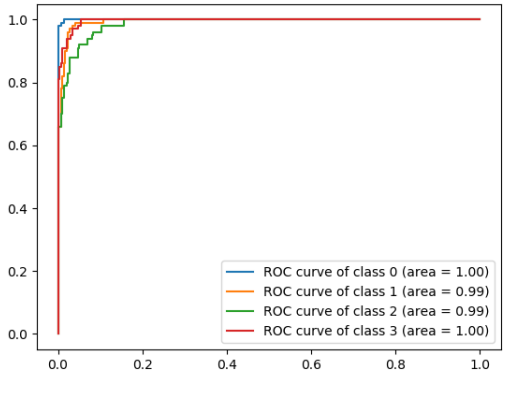
3.5 Metric Analysis

Els dos mètodes que ens han donat millors *scores* són el SVM i el *KneighborsClassifier*. Farem un gràfic de la *Precision-Recall* i de *ROC* i els *Classificatio Reports*.

3.5.1 SVM

En les imatges d'a-continuació veiem que tan l'area de la *Precision-Recall* com la de la corba *ROC*, és molt propera a 1 per algunes classes i per altres és igual a 1.



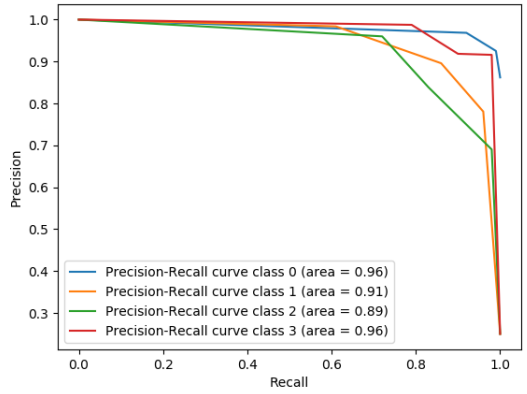


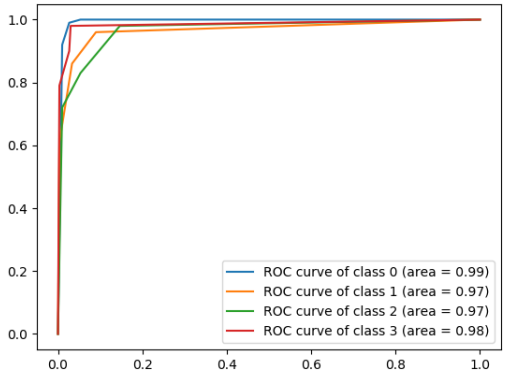
Correct classification SVM: 0.94

	precision	recall	f1-score	support
class 0	0.94	0.98	0.96	100
class 1	0.95	0.91	0.93	100
class 2	0.92	0.93	0.93	100
class 3	0.96	0.95	0.95	100
accuracy			0.94	400
macro avg	0.94	0.94	0.94	400
weighted avg	0.94	0.94	0.94	400

3.5.2 KneighborsClassifier

Amb aquest model també ens surten valors molts properas a 1.





Correct classification Kneighbors: 0.905

	precision	recall	f1-score	support
class 0	0.93	0.97	0.95	100
class 1	0.91	0.89	0.90	100
class 2	0.85	0.87	0.86	100
class 3	0.93	0.89	0.91	100
accuracy			0.91	400
macro avg	0.91	0.91	0.90	400
weighted avg	0.91	0.91	0.90	400

Podem observar que la classe 0 és classificada correctament al 100% amb SVM i amb l'altre model és també la que és classifica amb menys error de les 4. I que la que es classifica pitjor és la classe 3.

3.6 Hyperparameter Search

Per trobar el millor hyperparàmtre hem calculat els *scores* del mètode amb *Cross Validation* provant amb diferents valors i ens hem quedat amb el valor que feia que el *score* fos màxim.

3.6.1 SVM

Pel SVM tenim dos paràmetres a buscar la C i la gamma. Hem trobat que els millors valors són C=35 i gamma = 0.1. Això ens ha donat un *score* de 0.9425.

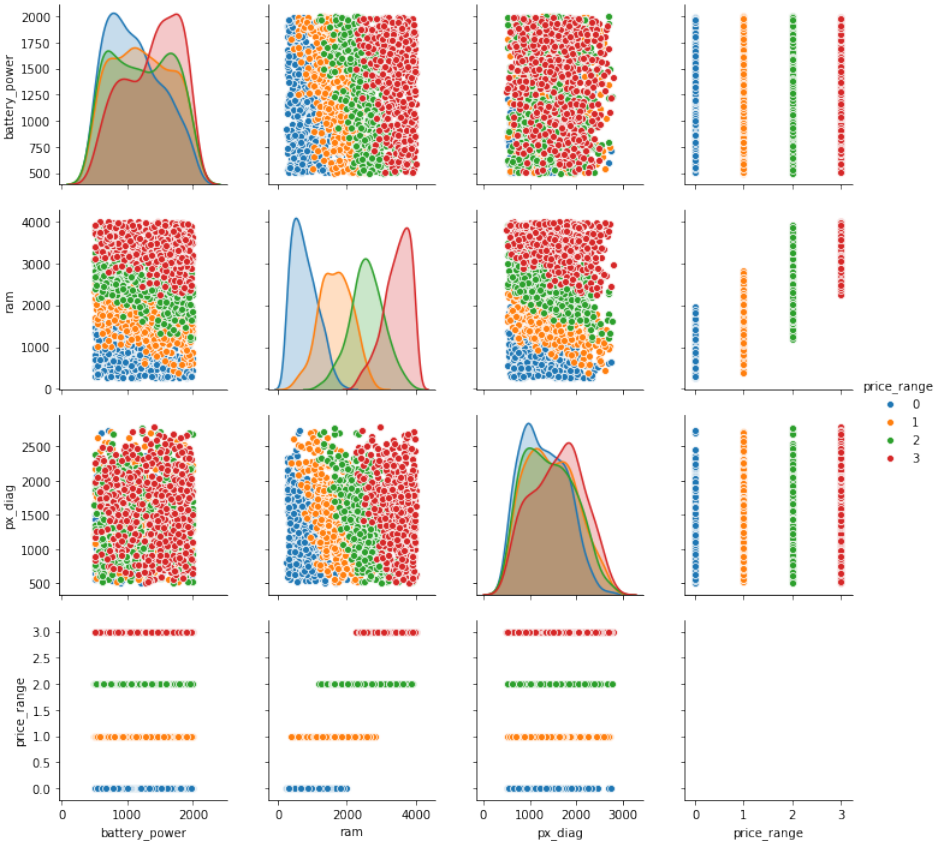
3.6.2 KneighborsClassifier

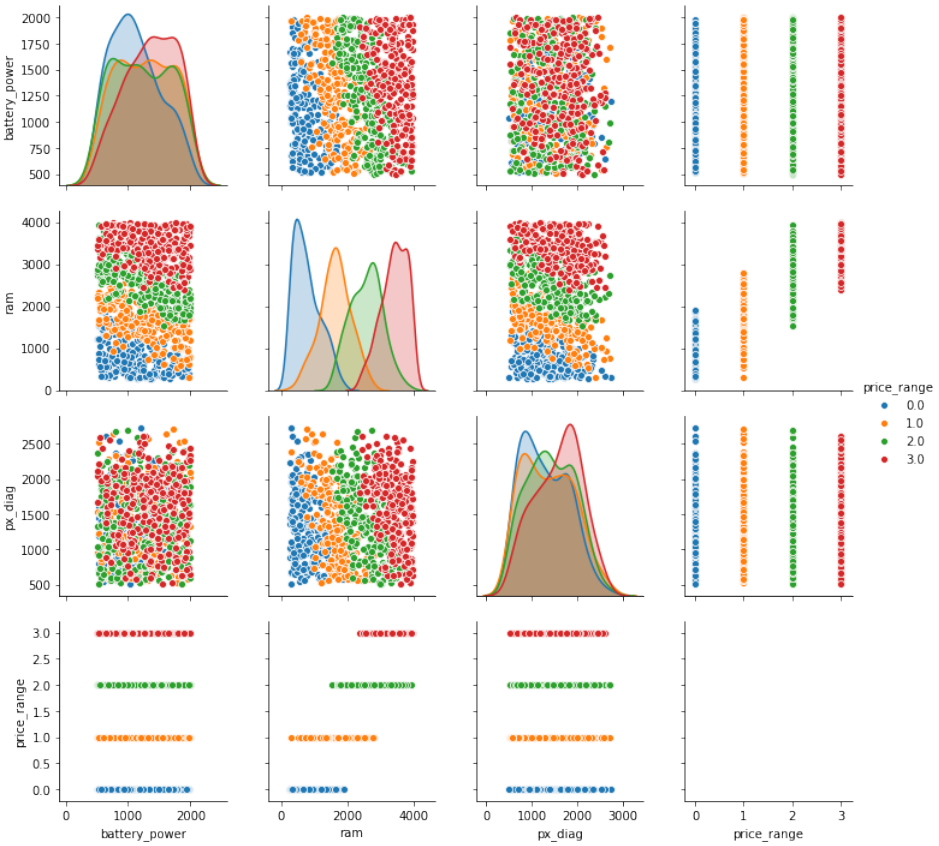
Pel *Kneighbors* hem trobat que el millor nombre de veïns és 7. Això ens ha donat un *score* de 0.919.

3.7 Predicció

Hem agafat el conjunt de test pel qual no sabem la classe de cada dada i la hem predit amb els nostres models. Amb SVM ens ha classificat 251 elements com a classe 0, 227 com a classe 1, 266 de la classe 2 i 256 de la 3. En el train teniem el mateix nombre de mostres per classe. En el test segurament també és així, per tant ens haurien de sortir 250 elements a cada classe, i ja és el que ens ha sortit tenint en compte l'**accuracy** esperada.
Amb el Kneighbors ens han sortit 251 de la 0, 232 de la 1, 255 de la 2 i 262 de la 3.

A continuació veiem dos *pairplots*, el primer està fet amb les dades d'entrenamnet i el segon amb les dades de test després d'haver predit la classe per cada mostra amb el model SVM. Podem observar distribucions similars i això ens fa pensar que hem predit bé les classes.





3.8 Conclusions

El model que ens ha donat una accuracy més gran ha sigut el SVM fent servir un Radial basis function kernel. I el 2n millor el Kneighbors. Això ha estat fent les proves amb el dataset reduït a 3 atributs i havent-los normalitzat. Per a altres models hauríem d'utilitzar una 'configuració' de dataset diferent, o amb les dades no normalitzades o amb tots els atributs o ambdues coses.