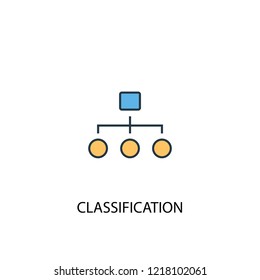
Practica 2.- Classificació



Grup: 304

Víctor Bosch Pueyo 1566583

Álvaro Caravaca Hernández 1566685

Juan Carlos Martínez Moreno 1566936

Índex

[Apartat B: Comparativa de models 3](#_Toc88211440)

[Introducció a la base de dades 3](#_Toc88211441)

[Comparació de models 3](#_Toc88211442)

[Apartat A: Classificació Numèrica 8](#_Toc88211443)

[EDA (exploratory data analysis) 10](#_Toc88211444)

[Preprocessing 10](#_Toc88211445)

[Model Selection 10](#_Toc88211446)

[Crossvalidation 10](#_Toc88211447)

[Metric Analysis 10](#_Toc88211448)

[Hyperparameter Search 10](#_Toc88211449)

# Apartat B: Comparativa de models

//bla bla bla, explicació apartat b

//alvaro

## Introducció a la base de dades

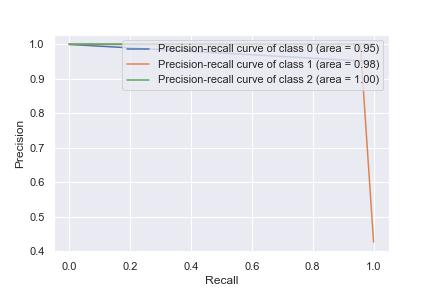
//alvaro

## Comparació de models

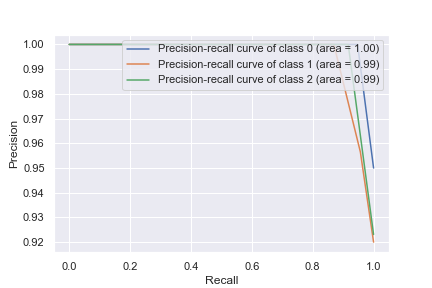
//alvaro i juankers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Logistic | SVM | KNN | Decision Tree |
| 50% train | 0.9775 | 0.9887 | 0.9775 | 0.8988 |
| 80% train | 0.9722 | 0.9722 | 0.9166 | 0.8888 |
| 70% train | 0.9814 | 0.9629 | 0.9814 | 0.9629 |

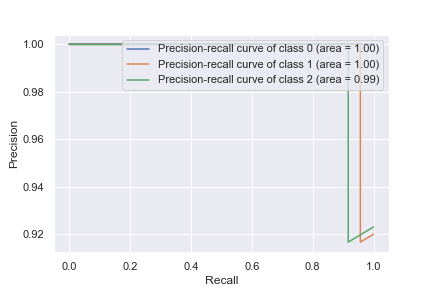
PR DECISION TREE



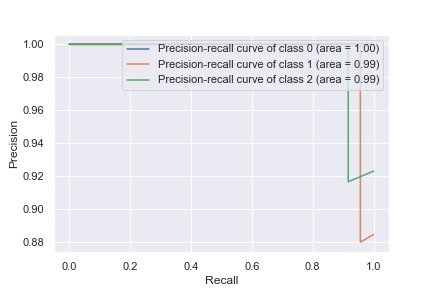
PR KNN



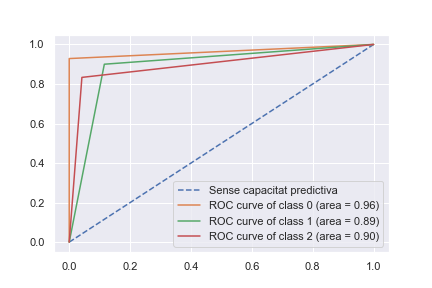
PR LOGISTIC REGRESSION



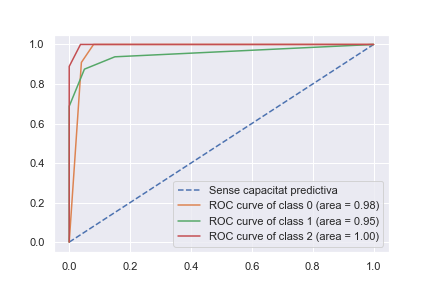
PR SVM



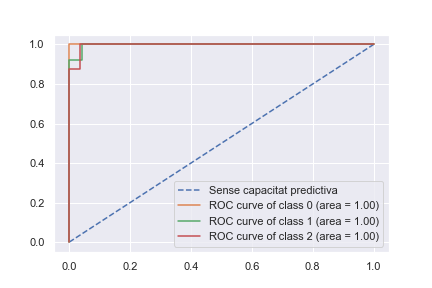
ROC DECISION TREE



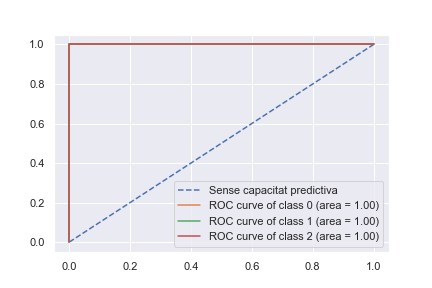
ROC KNN



ROC LOGISTIC REGRESSION



ROC SVM



Un dibujo de un perro

Descripción generada automáticamente con confianza baja

# Apartat A: Classificació Numèrica

En aquest apartat analitzarem una base de dades diferent a l’apartat A. Aquesta es una base de dades més rica i en la que analitzarem numèricament els resultats de diferents models.

Treballarem diferents aspectes de la classificació i tots giraran al voltant dels següents resultats obtinguts per cada model:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PERCEPTRÓ | Precisió | alpha | fit\_intercept | penalty | Tol |
| Bàsica |  |  |  |  |  |
| 50% train, 50% test | 0.706 | 0.0001 | True | None | 0.001 |
| 80% train, 20% test | 0.7175 | 0.0001 | True | None | 0.001 |
| 70% train, 30% test | 0.765 | 0.0001 | True | None | 0.001 |
|  |  |  |  |  |  |
| K-Fold |  |  |  |  |  |
| K = 2 | 0.713 | 0.0001 | True | None | 0.001 |
| K = 3 | 0.72101 | 0.0001 | True | None | 0.001 |
| K = 4 | 0.74 | 0.0001 | True | None | 0.001 |
| K = 5 | 0.702 | 0.0001 | True | None | 0.001 |
| K = 6 | 0.726 | 0.0001 | True | None | 0.001 |
|  |  |  |  |  |  |
| LOOCV | 0.727 | 0.0001 | True | None | 0.001 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KNN | Precisió | leaf\_size | n\_neighbors | metric | weights |
| Bàsica |  |  |  |  |  |
| 50% train, 50% test | 0.506 | 30 | 5 | minkowski | uniform |
| 80% train, 20% test | 0.52 | 30 | 5 | minkowski | uniform |
| 70% train, 30% test | 0.51167 | 30 | 5 | minkowski | uniform |
|  |  |  |  |  |  |
| K-Fold |  |  |  |  |  |
| K = 2 | 0.4985 | 30 | 5 | minkowski | uniform |
| K = 3 | 0.5085 | 30 | 5 | minkowski | uniform |
| K = 4 | 0.5 | 30 | 5 | minkowski | uniform |
| K = 5 | 0.503 | 30 | 5 | minkowski | uniform |
| K = 6 | 0.5015 | 30 | 5 | minkowski | uniform |
|  |  |  |  |  |  |
| LOOCV | 0.5145 | 30 | 5 | minkowski | uniform |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Decision Tree | Precisió | criterion | splitter | max\_leaf\_nodes | max\_depth |
| Bàsica |  |  |  |  |  |
| 50% train, 50% test | 0.809 | gini | best | None | None |
| 80% train, 20% test | 0.81 | gini | best | None | None |
| 70% train, 30% test | 0.8 | gini | best | None | None |
|  |  |  |  |  |  |
| K-Fold |  |  |  |  |  |
| K = 2 | 0.8295 | gini | best | None | None |
| K = 3 | 0.8245 | gini | best | None | None |
| K = 4 | 0.8295 | gini | best | None | None |
| K = 5 | 0.821 | gini | best | None | None |
| K = 6 | 0.8265 | gini | best | None | None |
|  |  |  |  |  |  |
| LOOCV | 0.839 | gini | best | None | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Random Forest | Precisió | criterion | bootstrap | max\_leaf\_nodes | max\_depth |
| Bàsica |  |  |  |  |  |
| 50% train, 50% test | 0.864 | gini | True | None | None |
| 80% train, 20% test | 0.88 | gini | True | None | None |
| 70% train, 30% test | 0.88 | gini | True | None | None |
|  |  |  |  |  |  |
| K-Fold |  |  |  |  |  |
| K = 2 | 0.8675 | gini | True | None | None |
| K = 3 | 0.873 | gini | True | None | None |
| K = 4 | 0.879 | gini | True | None | None |
| K = 5 | 0.877 | gini | True | None | None |
| K = 6 | 0.8865 | gini | True | None | None |
|  |  |  |  |  |  |
| LOOCV | 0.881 | gini | True | None | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Regressió logística | Precisió | C | fit\_intercept | penalty | tol |
| Bàsica |  |  |  |  |  |
| 50% train, 50% test | 0.948 | 1.0 | True | None | 0.0001 |
| 80% train, 20% test | 0.965 | 1.0 | True | None | 0.0001 |
| 70% train, 30% test | 0.96 | 1.0 | True | None | 0.0001 |
|  |  |  |  |  |  |
| K-Fold |  |  |  |  |  |
| K = 2 | 0.954 | 1.0 | True | None | 0.0001 |
| K = 3 | 0.956 | 1.0 | True | None | 0.0001 |
| K = 4 | 0.9605 | 1.0 | True | None | 0.0001 |
| K = 5 | 0.9625 | 1.0 | True | None | 0.0001 |
| K = 6 | 0.9635 | 1.0 | True | None | 0.0001 |
|  |  |  |  |  |  |
| LOOCV | 0.966 | 1.0 | True | None | 0.0001 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SVM | Precisió | C | kernel | decision\_function\_shape | tol |
| Bàsica |  |  |  |  |  |
| 50% train, 50% test | 0.854 | 1.0 | rbf | ovr | 0.001 |
| 80% train, 20% test | 0.855 | 1.0 | rbf | ovr | 0.001 |
| 70% train, 30% test | 0.85167 | 1.0 | rbf | ovr | 0.001 |
|  |  |  |  |  |  |
| K-Fold |  |  |  |  |  |
| K = 2 | 0.8635 | 1.0 | rbf | ovr | 0.001 |
| K = 3 | 0.868 | 1.0 | rbf | ovr | 0.001 |
| K = 4 | 0.886 | 1.0 | rbf | ovr | 0.001 |
| K = 5 | 0.885 | 1.0 | rbf | ovr | 0.001 |
| K = 6 | 0.885 | 1.0 | rbf | ovr | 0.001 |
|  |  |  |  |  |  |
| LOOCV | 0.888 | 1.0 | rbf | ovr | 0.001 |

## Introducció a la base de dades

En base de dades d’aquest segon apartat treballem amb diferents característiques dels telèfons per acabar fent la predicció del seu preu de mercat. Els atributs amb els que treballarem son:

**Battery\_power**: bateria total que podem emmagatzemar d’un sol cop mesurat en mAh.

**Blue**: si te o no Bluetooth.

**Clock\_speed**: velocitat a la que el microprocessador executa instruccions.

**Dual\_sim**: si podem posar dues targes SIM o no.

**Fc**: mega píxels de la càmera frontal.

**Four\_g**: Si té o no 4G.

**Int\_memory**: Memòria interna en gigabytes.

**M\_deep**: profunditat del telèfon en cm.

**Mobile\_wt**: pes del telèfon.

**N\_cores**: Número de cores del processador.

**Pc**: Mega píxels de la càmera principal.

**Px\_height**: Resolució de l’altura.

**Px\_width**: Resolució de l’amplada.

**Ram**: RAM en mega bytes.

**Sc\_h**: Altura de la pantalla en cm.

**Sc\_w**: Amplada de la pantalla en cm.

**Talk\_time**: temps màxim durarà que una sola càrrega en trucada.

**Three\_g**: té 3G o no.

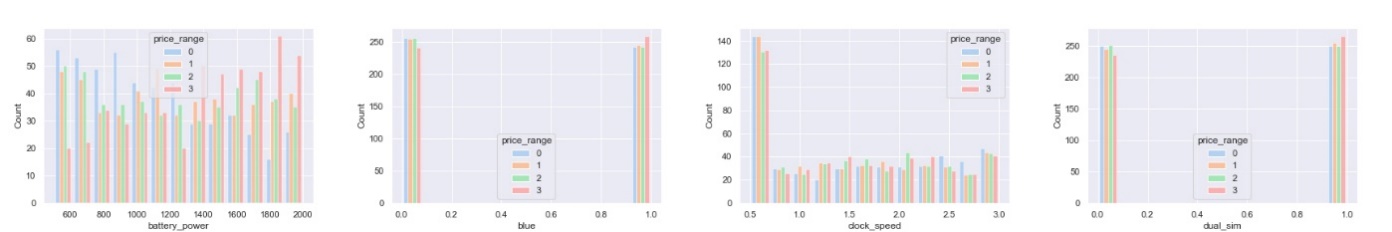
**Touch\_screen**: té pantalla tàctil o no.

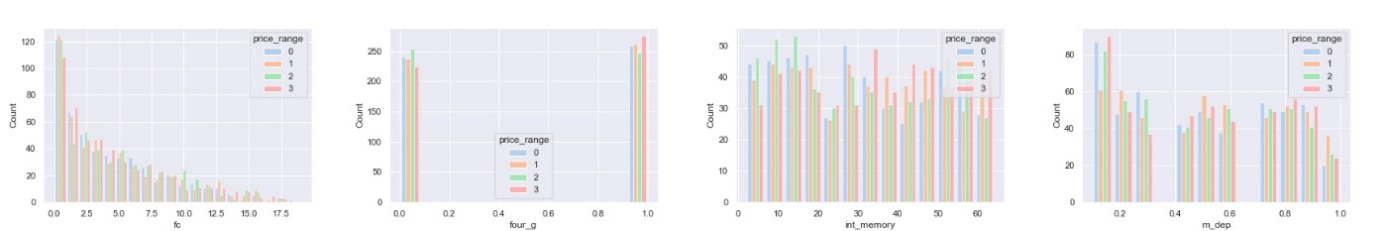
**Wifi**: té wifi o no.

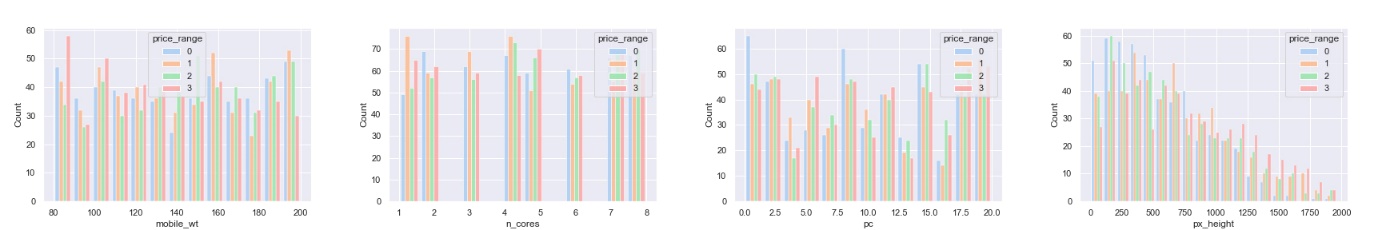
**Price\_range**: pot prendre 4 valors: 0 (low cost), 1 (medium cost), 2 (high cost) and 3 (very high cost).

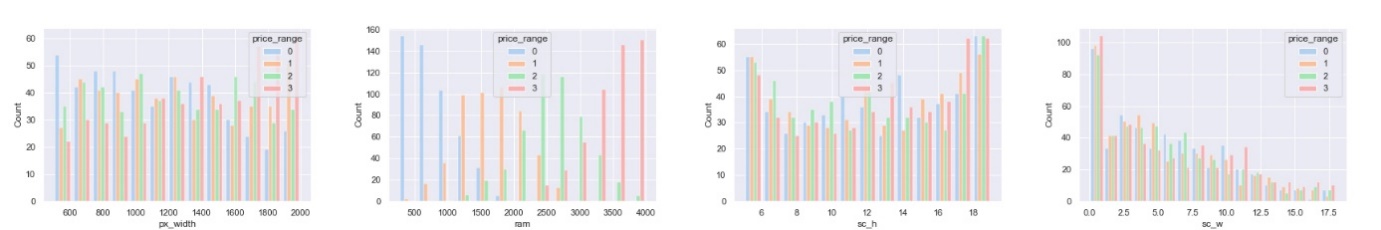
Per a poder fer un estudi adient de la nostra base de dades, fem una visualització dels histogrames i les gràfiques de dispersió de cada atribut. D’aquesta manera entendrem millor amb quines dades estem treballant. Les gràfiques són les següents:

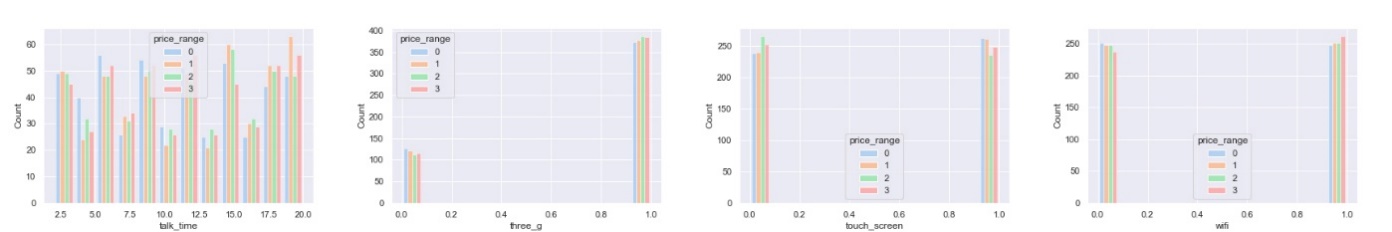
### HISTOGRAMES DELS ATRIBUTS

Histogrames dels atributs 1-4

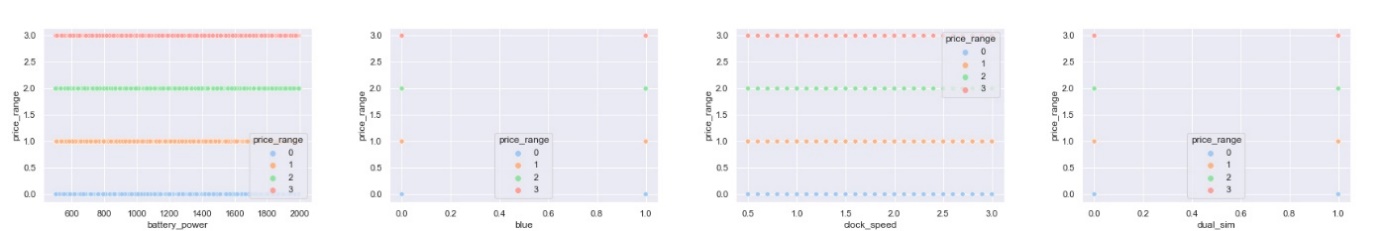
Histogrames dels atributs 5-8

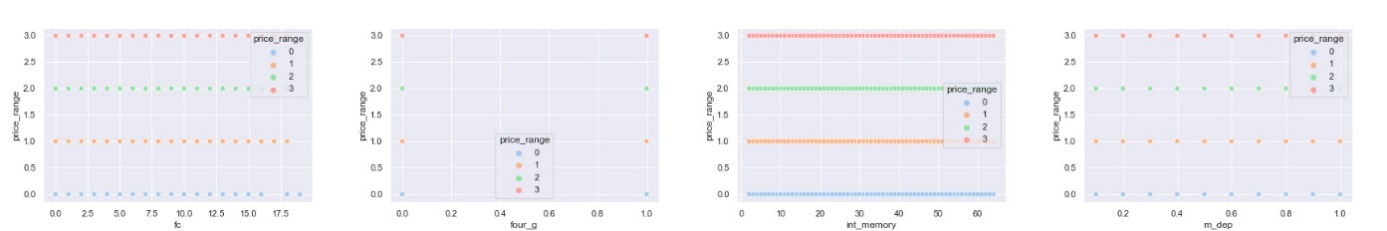
Histogrames dels atributs 9-12

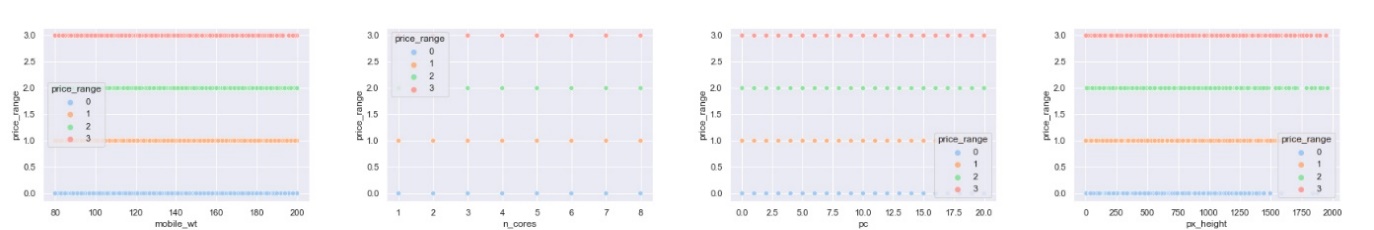
Histogrames dels atributs 13-16

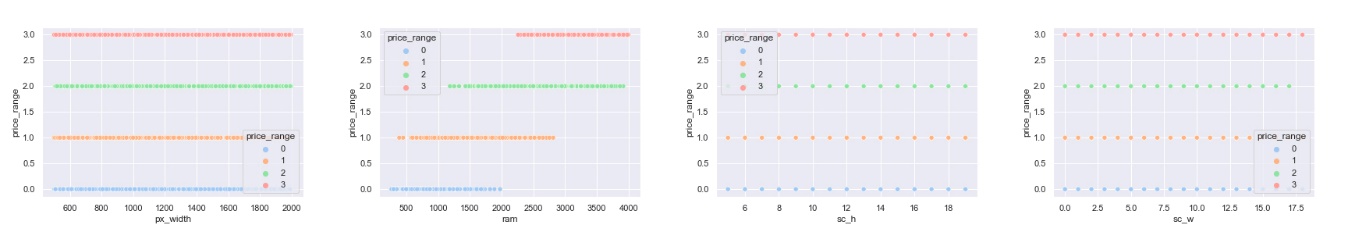
Histogrames dels atributs 17-20

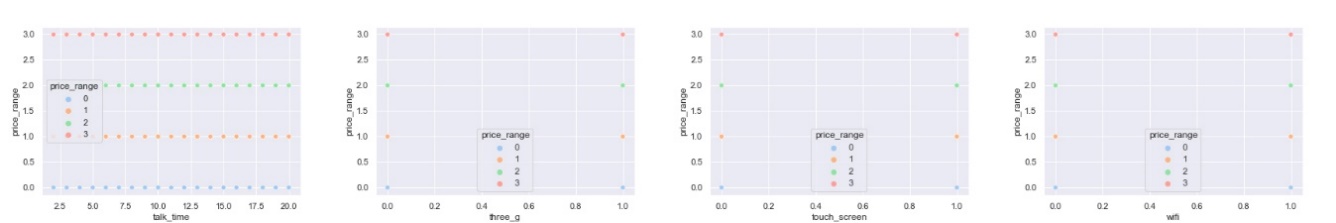
### GRÀFIQUES DE DISPERSIÓ

Dispersió dels atributs 1-4

Dispersió dels atributs 5-8

Dispersió dels atributs 9-12

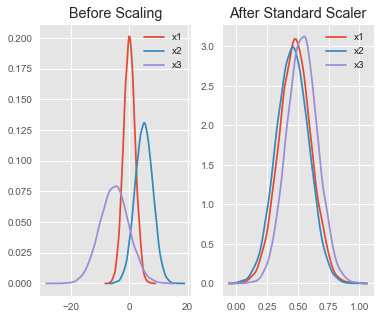
Dispersió dels atributs 13-16

Dispersió dels atributs 17-20

## Preprocessing

L’únic pre-processament que hem fet amb la nostra base de dades ha sigut escalar les dades. Ho hem fet ja que teníem dades numèriques amb rangs molt diferents i d’aquesta manera donarem la mateixa importància a tots els atributs.

Aquest canvi no faria falta per tots els models que hem implementat ja que, per exemple, els arbres de decisió no operen amb el valor exacte sinó que ramifica l’arbre per intervals de dades.



## Model Selection

En aquest subapartat analitzarem els resultats obtinguts en les diferents models que hem implementat. Aquests els trobem a les taules de l’inici de l’apartat A.

Com podem observar, els models que hem utilitzat són: Perceptró, k-nearest neighbors, desicion tree, Random Forest, Regressió logística i SVM. Apart d’això, per cada model, analitzem els seus resultats si dividim el data set amb diferents percentatges. Altres resultats obtinguts per a cada tipus de model que trobem a les taules els explicarem més endavant.

Executem tots els models amb els seus paràmetres per defecte.

Amb els resultats obtinguts podem afirmar que, per a la nostra base de dades el model que millors resultats ens proporciona es la regressió logística amb una precisió mitja de 95,77%. Tots els altres classificadors donen precisions al voltant de 82 – 83% menys el classificador k-Nearest neighbors, que ens dona uns resultats molt pobres (poc millors que tirar una moneda enlaire).

Havent fet la prova de dividir les dades en conjunts de mida diferent, observem que el que millors resultats ens proporciona es fer 80% de train i 20% de test. Es lògic aquest resultat ja que, com més dades donem per a entrenar el model, més podrem ajustar el nostre classificador. Igualment, no es recomanable utilitzar més del 80% de les dades totals per al conjunt de train perquè podem patir overfitting fàcilment.

Aplicar ensemble amb els mètodes utilitzat no tindria cap sentit ja que els ensemble methods s’utilitzen amb classificadors molt simples per a formar-ne un de complet. Els nostres classificadors ja són molt complexos i juntar-los seria molt costós.

## Crossvalidation

En comptes d’aplicar-ho només en certs models, hem aplicat crossvalidation a tots els implementats. Aquesta tècnica serveix per a analitzar el rendiment del model entrenat utilitzant un conjunt de validació. Aquest conjunt anirà variant en cada iteració ja que hem d’entrenar i validar el model per a cada subconjunt creat. Per exemple, si la nostra k es igual a 7, dividirem el model en 7 parts, 6 per a entrenar i una per a fer la validació. Aquest procés el farem 7 vegades fins que tots els subconjunts hagin estat conjunt de validació. Finalment retornem la mitjana del rendiment del model.

Igual que en els resultats anteriors, el que millor rendiment ens proporciona es la regressió logística amb un màxim de 96,35%. De la mateixa manera, tots els altres models han presentat resultats molt similars que en l’apartat anterior.

També, veiem que la score es millor com més dividim el conjunt. Té sentit ja que, com en el cas de dividir el data set en train i test, com més divisions en fem, més dades tindrem per a entrenar el model sobre el conjunt total. Si la k és igual a 2, estarem dividint en 50 – 50 en canvi, si es igual a 6, la nostra divisió serà de 83,33% d’entrenament contra 16,67%.

També hem aplicat la tècnica de leave one out però hem arribat a la conclusió que no és la més eficient en el nostre cas ja que obtenim un resultat similar a l’entrenament bàsic en tots els casos però tarda molt més en ser executat.

## Metric Analysis

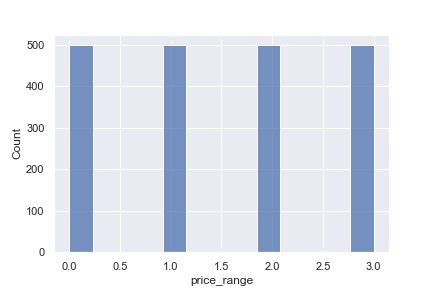
En aquest apartat ens centrarem en l’analisi dels 6 models que hem creat per implementar el nostre classificador de preus de mòbils. Aquest anàlisi, juntament amb altres tècniques de optimització de hiperparametres per a cada model, ens ajudaran a determinar quin dels 6 models es el millor a nivell de fiabilitat.

Aquest anàlisi el podem fer calculant unes mètriques de evaluació per cada model, i comparar-les juntament amb unes corves, anomenades ROC i PR, que ens determinaran el millor model. Les mètriques que podem utilitzar son moltes, com per exemple: accuracy, precision, recall, f1 score, etc...

A continuació deixem una taula per cada model amb les seves mètriques calculades i una petita explicació de cadascuna.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | Accuracy | Precision | F1 Score | Recall |
| Support Vector Machine | 0.855 | 0.8584 | 0.8563 | 0.855 |
| Perceptron | 0.725 | 0.7087 | 0.7082 | 0.725 |
| KNN | 0.4925 | 0.5546 | 0.5046 | 0.4925 |
| Decision Tree | 0.8325 | 0.8344 | 0.8331 | 0.8325 |
| Random Forest | 0.91 | 0.9109 | 0.9102 | 0.91 |
| Logistic Regression | 0.9725 | 0.9727 | 0.9724 | 0.9725 |

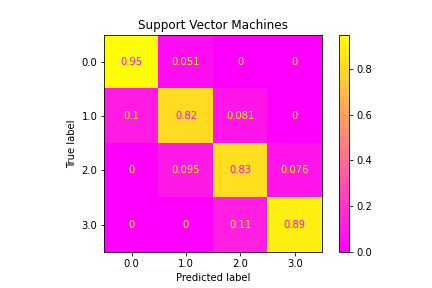
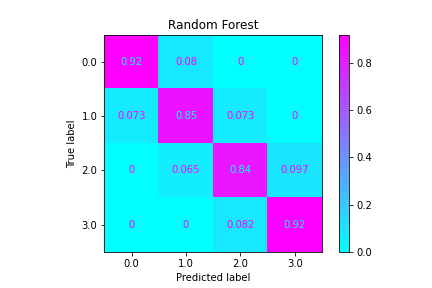
* Accuracy: mesura el ratio de true positive + true negative / total. En conjunts de dades on el numero de dades per classe no esta balancejat, aquesta mètrica pot donar problemes i no mostrar un indicatiu real de com treballa aquest model. En la nostra base de dades, les dades estan balancejades ja que tenim el mateix numero de dades classificat en les 4 classes (500 dades pertanyents a cada una de les 4 classes).



* Precision: mesura el ratio de true positive / false positive + true positive. La precisió és intuïtivament la capacitat del classificador de no etiquetar com a positiva una mostra que és negativa. Aquesta mesura, igual que les següents, te en compte el balanç de les dades i no hi ha cap problema en utilitzar-la en conjunts de dades desbalançejat.
* Recall: mesura el ratio de true positive / false negative + true positive. El recall és intuïtivament la capacitat del classificador per trobar totes les mostres positives.
* F1 Score: El F1 Score es pot interpretar com una barreja de precisió i recall. La formula es: 2 \* (precision \* recall) / (precision + recall)

Per tant, com es pot apreciar a la taula, la diferencia entre cada mètrica es pràcticament inapreciable, degut al balanceig de les dades a la nostra base de dades. Segons les mètriques, el millor classificador per ara, es el regressor logistic, amb un 97% d’accuracy, un classificador molt potent i casi perfecte.

Ara anem a observar les matrius de confusió de cada model, que ens donen informació de com s’ha classificat cada classe i quins errors han aparegut en cadascuna d’aquestes.



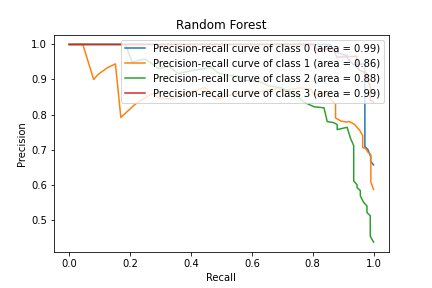
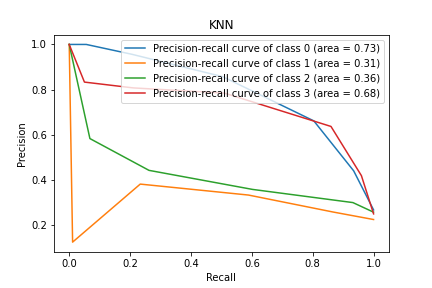
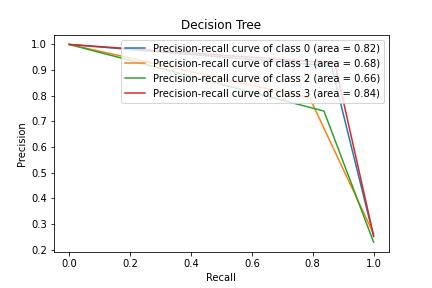
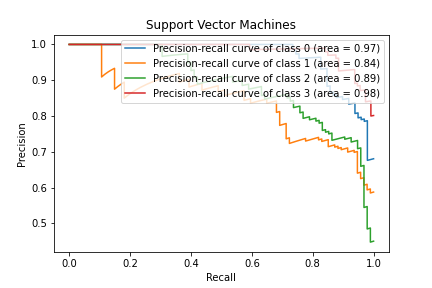
## Gráfico Descripción generada automáticamente

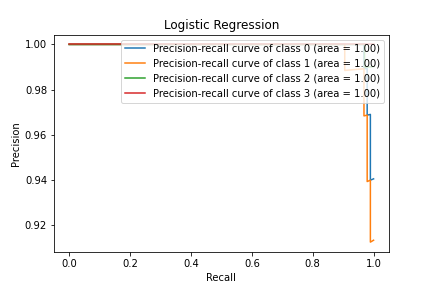
## 

## 

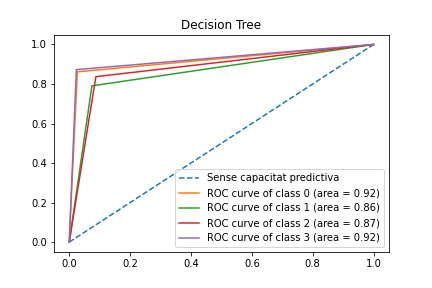
Les matrius de confusió ens mostren que la regressió logística fa una classificació de les 4 classes casi perfecte (els percentatges de la diagonal ronden el 98%), mentre que els altres models ho fan pitjor.

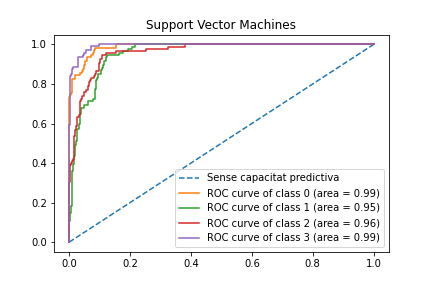
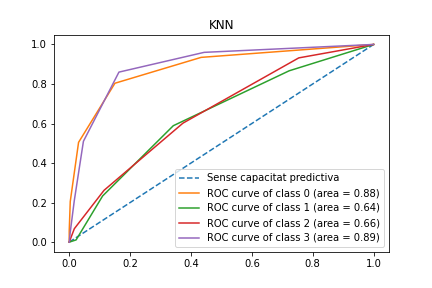
Per últim, farem la comparació de models utilitzant dos tipus de corbes que ens donen molta informació sobre com esta classificant cadascun dels models, i utilitzen les mètriques vistes anteriorment per dibuixar la corba. Aquestes son les corbes ROC (false positive-true positive) i PR (precision-recall).

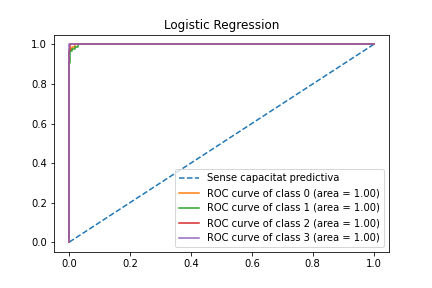
Començem per les curves PR:



Les corbes PR descriuen la relació que hi ha entre la precisió i el recall del model, i ens interessa que tant la precisió com el recall sigui de 1, i per tant, ens interesa que la punta de la corba estigui en el punt (1,1), es a dir, al costat superior de la dreta. La corba del model KNN es molt dolenta, les dels altres models son bones, especialment la del model de regressió logística, amb una corba perfecta.

Gráfico, Gráfico de líneas

Descripción generada automáticamenteAra mostrem les corbes ROC:



Les corbes ROC comparen el true positive ratio (eix y) i el false positive ratio (eix x). Ens interessa que la punta de la corba estigui en el punt (0,1), ja que en interessa que el false positive ratio sigui el mínim possible i el true positive ratio el màxim possible. La corba ROC de la regressió logística torna a ser perfecte, i les altres bastant bones i acceptables, ja que el model classifica millor que l’atzar, representat per la línia discontinua.

Per al nostre cas, les dues corbes ens son utils ja que les dades estan balancejades i no tenim la necessitat de utilitzar solament les corbes PR (que son especialment utils en bases de dades no balancejades), si no que podem utilitzar les dos i fer les comparacions.

## Hyperparameter Search

Ara que coneixem una mica més el rendiment que té cada model amb els paràmetres per defecte, ja es el moment de buscar els millors paràmetres que maximitzaran el rendiment del classificador escollit.

Per trobar aquests paràmetres, la llibreria sklearn implementa alguns mètodes que permeten escollir uns valors diferents per cada parametre d’un model, i es troba la combinació de valors pels paràmetres que maximitza l’score. En el nostre cas, el score escollit es el accuracy.

Els mètodes més importants per trobar els millors paràmetres son el **GridSearchCV** i el **RandomizedSearchCV**. El primer troba els millors paràmetres per un model a partir d’un diccionari amb els possibles valors que pot adoptar cada parametre, que aquest diccionari el defineix l’usuari, i el numero de cross-validations que es volen fer. El RandomizedSearchCV es molt semblant al GridSearchCV, amb la particularitat de que incorpora l’atzar per trobar millors resultats, a canvi d’un major cost computacional, i s’ha de definir el numero de iteracions que es fan, a més del numero de cross-validation.

Nosaltres hem decidit utilitzar el GridSearchCV perquè es més simple i com tenim 6 models, amb el randomized el temps d’execució s’eleva moltíssim. A més, si tinguessim els recursos limitats, es millor arriscar rendiment del model i millorar-lo una mica idea, utilitzant el GridSearchCV amb un numero de valors pels paràmetres reduït.

A continuació mostrem els **valors pels diferents paràmetres** que volem probar, i després el resultat dels millors paràmetres per cada model.

|  |  |
| --- | --- |
| Support Vector Machine | 'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid'] |
| KNN | 'n\_neighbors' : [5,7,9,11,13,15],  'weights' : ['uniform','distance'],  'metric' : ['minkowski','euclidean','manhattan'] |
| Decision Tree | 'max\_depth': [2, 3, 5, 10, 20],  'min\_samples\_leaf': [5, 10, 20, 50, 100],  'criterion': ["gini", "entropy"] |
| Random Forest | 'bootstrap': [True],  'max\_depth': [80, 90, 100, 110],  'max\_features': [2, 3],  'min\_samples\_leaf': [3, 4, 5],  'min\_samples\_split': [8, 10, 12],  'n\_estimators': [100, 200, 300, 1000] |
| Perceptron | 'penalty': ['l2','l1','elasticnet'], 'alpha': [0.0001, 0.001, 0.01, 0.1, 1], 'fit\_intercept': [True, False], 'shuffle': [True, False] |
| Logistic Regression | 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000] |

Ara mostrem el resultat de que ha donat el GridSearchCV per cada model:

|  |  |
| --- | --- |
| Support Vector Machine | 'C': 100, 'gamma': 0.01, 'kernel': 'sigmoid' |
| KNN | 'metric': 'manhattan', 'n\_neighbors': 15, 'weights': 'distance' |
| Decision Tree | 'criterion': 'gini', 'max\_depth': 20, 'min\_samples\_leaf': 10 |
| Random Forest | 'bootstrap': True, 'max\_depth': 110, 'max\_features': 3, 'min\_samples\_leaf': 3, 'min\_samples\_split': 8, 'n\_estimators': 1000 |
| Perceptron | 'alpha': 0.001, 'fit\_intercept': True, 'penalty': 'l1', 'shuffle': True |
| Logistic Regression | 'C': 10 |

Ara, comparem el score que té cada model amb els paràmetres per defecte i després d’aplicar la busqueda exhaustiva dels millors paràmetres:

|  |  |  |
| --- | --- | --- |
|  | Accuracy amb paràmetres per defecte | Accuracy amb els millors parametres |
| Support Vector Machine | 0.855 | 0.959 |
| KNN | 0.492 | 0.619 |
| Decision Tree | 0.832 | 0.853 |
| Random Forest | 0.91 | 0.87 |
| Perceptron | 0.725 | 0.791 |
| Logistic Regression | 0.972 | 0.973 |

Segons es pot veure en aquesta taula, en general tots els models han millorat el seu score, excepcionalment el SVM i el KNN amb una millora de mes del 10%, però també observem un empitjorament en el random forest, que ha baixat el rendiment un 4%. Aquesta baixada del score pot ser perquè els paràmetres que nosaltres hem especificat al diccionari no son els correctes o no hem inclòs els valors per defecte, que també s’han de provar per veure si els valors per defecte ja son els millors.

Amb la millora aplicada, podríem concluir en que la maquina de vectors de suport podria ser una fantàstica alternativa al regressor logístic per aquest increment tan gran del seu score.