Aprenentatge Computacional

Pràctica 2. Classificació

Water Quality

Sergi Diaz Lopez
Oscar Moreno Ramos
Rubén Ramos Segarra

Universitat Autònoma de Barcelona,
Escola d'Enginyeria

Introducció

En aquesta pràctica aplicarem els coneixements d'anàlisi de dades i de models d'aprenentatge computacional per a resoldre un problema de classificació publicat a Kaggle. Volem desenvolupar un model per a, donades unes mesures d'anàlisi d'una mostra d'aigua com la quantitat de certs minerals o la duresa, predir si és apta per a consum humà o no.

Eines de treball

Per a realitzar aquesta pràctica hem utilitzat els següents programes i llibreries:

- **Jupyter Notebook,** com a entorn de treball.
- **Numpy,** la utilitzem principalment en conjunt amb les altres llibreries per a crear arrays de dades de forma eficient.
- **Pandas**, per a importar de manera senzilla les dades del dataset en format .csv.
- **Seaborn i matplotlib**, per a generar gràfiques (histogrames, matrius de correlació, etc.).
- **Sklearn** és una biblioteca per aprenentatge automàtic.

Apartat (B): Classificació Numèrica

EDA (exploratory data analysis)

La base de dades sobre la qual realitzarem el model conte 10 atributs i més de 3000 entrades amb dades resultants de l'anàlisi de mostres d'aigua.

```
In [14]: dataset.head()
```

Out[14]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

Context:
L'accés a l'aigua potable és essencial per a la salut, un dret humà bàsic i un component d'una política efectiva de protecció de la salut. Això és important com a qüestió de salut i desenvolupament en l'àmbit nacional, regional i local. En algunes regions, s'ha demostrat que les inversions en abastament d'aigua i sanejament poden generar un benefici econòmic net, ja que les reduccions dels efectes adversos per a la salut i els costos sanitaris superen els costos de dur a terme les intervencions.

El propòsit del model serà predir si l'aigua és potable o no, per tant, fixarem l'atribut "Potability" com el nostre target. Aquesta variable pren el valor 1 si és apta per al consum o el valor 0 si no és apta.

1. **pH value**: El valor de ph recomanable és entre 6,5 i 8,5 segons l'OMS, per tant, tot el que disti d'això farà que l'aigua no sigui potable. A la base de dades és del tipus float.

2. **Hardness**: és la duresa de l'aigua, no s'especifica quins valors són òptims pel consum. A la base de dades és del tipus float.

3. **Solids** (Total dissolved solids - TDS): Els valors desitjables són entre els 500 mg/l i 1000 mg/l. A la base de dades és del tipus float.

4. **Chloramines**: Són segurs fins als 4 mg/l. A la base de dades és del tipus float.

5. **Sulfate**: Es consideren valors segurs per aigua potable entre 3 i 30 mg/l. A la base de dades és del tipus float.

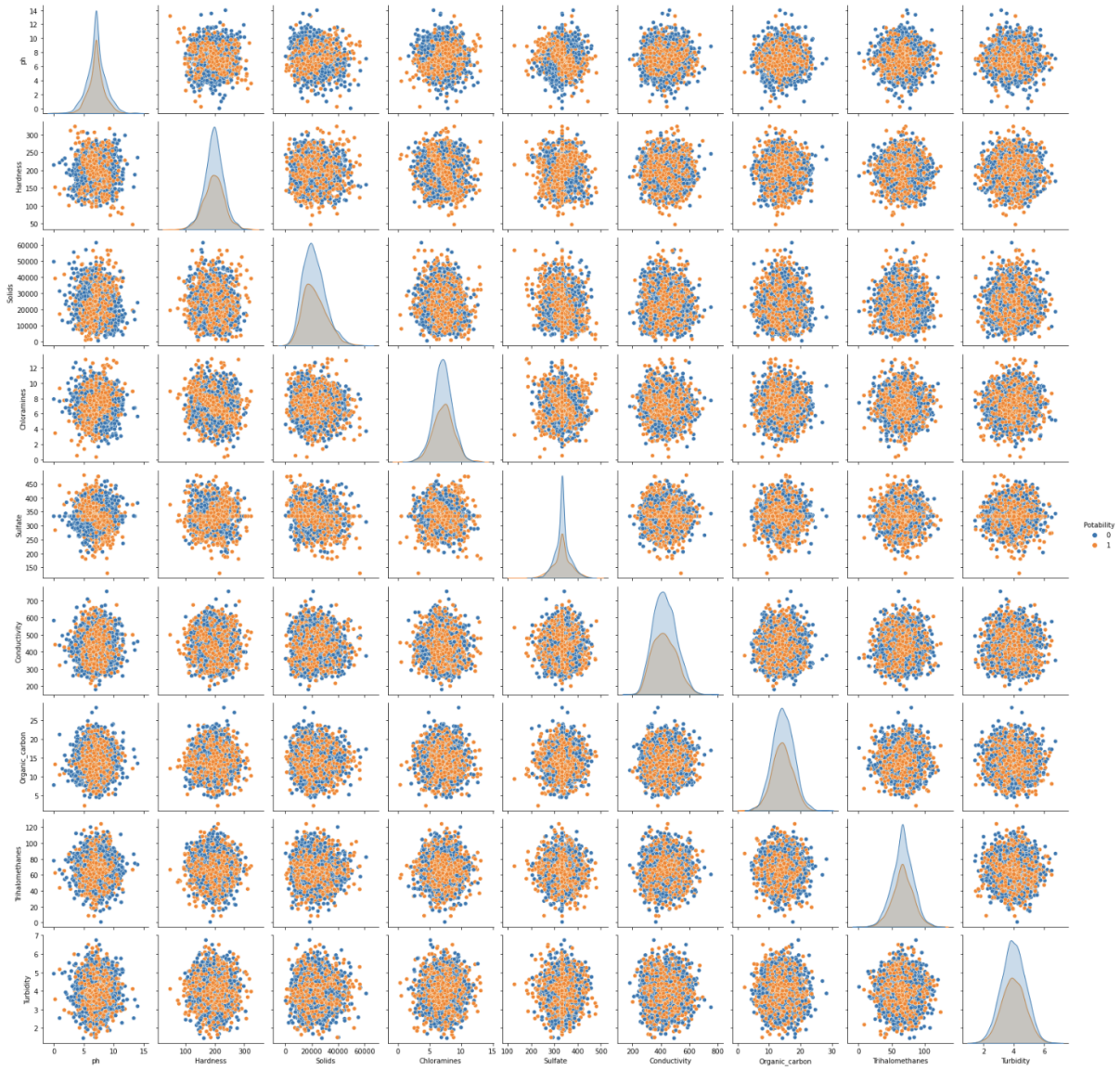
6. **Conductivity**: no pot superar els 400 µS/cm segons l'OMS. A la base de dades és del tipus float.

7. **Organic_carbon**: Ha de ser menor a 2 mg/L en aigua potable. A la base de dades és del tipus float.

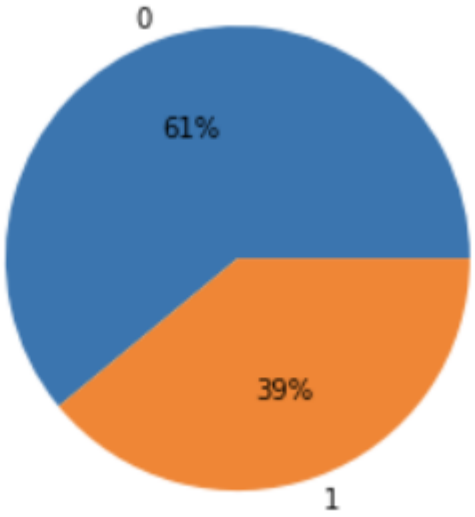
8. **Trihalomethanes**: Per a valors menors a 80 ppm serà potable. A la base de dades és del tipus float.

9. **Turbidity**:L'OMS recomana valors inferiors a 5,00 NTU per a l'aigua potable. A la base de dades és del tipus float.

10. **Potability**: Serà 1 en cas que sigui potable i 0 si no ho és. A la base de dades és del tipus int, però es comportarà com a binari, ja que només tindrà com a valors el 0 i l'1. Categòric de 2 categories.

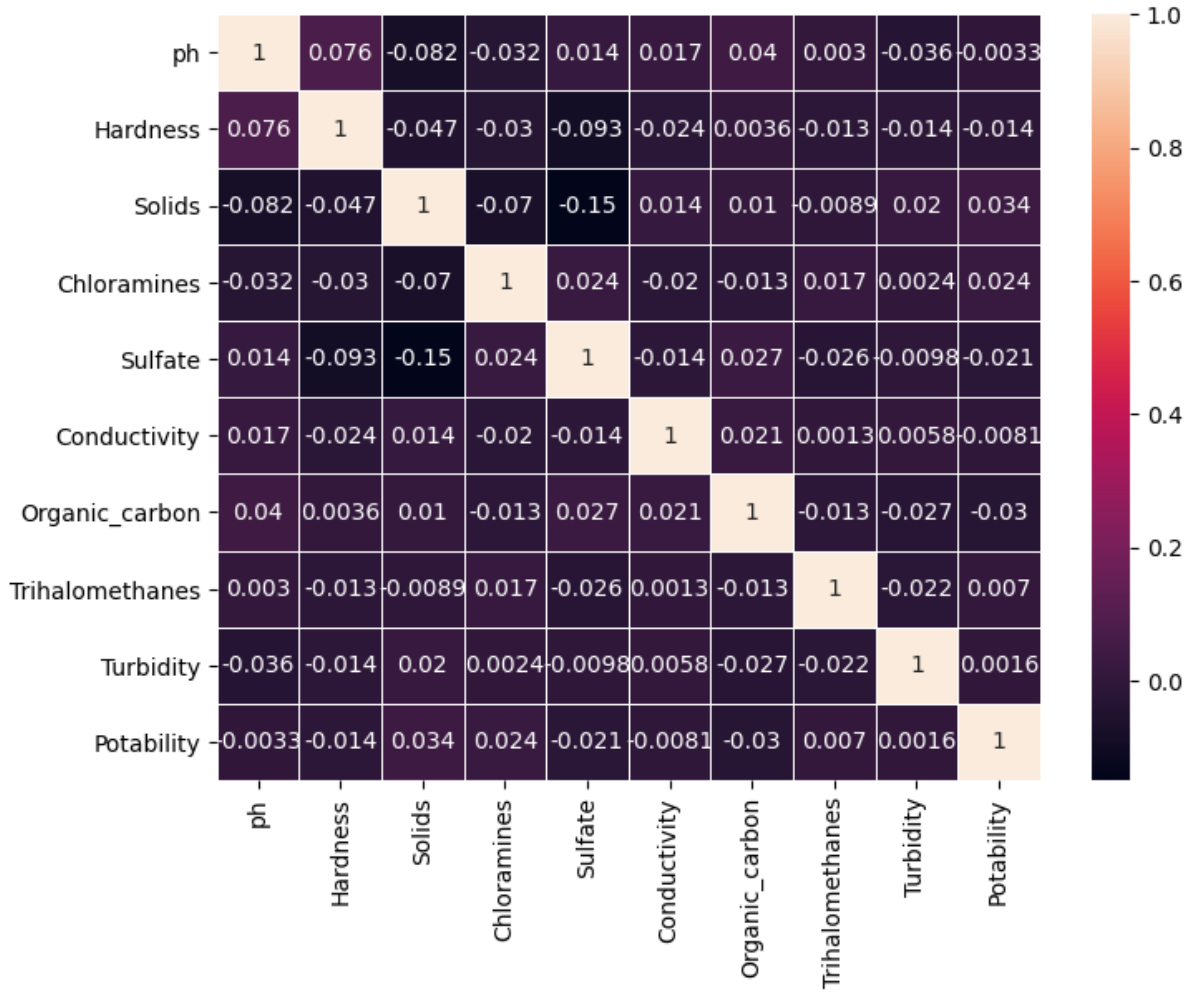


Realitzant la funció pairplot, observem com es comporta cada atribut en funció de si l'aigua és potable o no, i així de forma molt més visual podem veure a partir de quins valors l'aigua serà o no potable depenent de cada atribut.



Les dades no estan massa balancejades, trobem que el 61% dels casos seran on ens trobem amb aigua no potable 0 i en un 39% l'aigua sí que serà potable 1, com més balancejades estiguin millor a l'hora de fer qualsevol mena de classificació, ja que haurà de classificar els mateixos cops si és 0 o 1, a més, pot succeir que com que ens trobem amb una classe amb un percentatge major a l'altre, només dient que tot és aigua no potable tinguem dades d'un 61% d'encert i, per tant, sense poder fer una predicció, ja que tot ho classificarà com a aigua no potable.

Correlació



Les correlacions resultants entre la resta d'atributs i l'atribut objectiu (Potability) mostren que hi han valors molt baixos en la majoria de casos (a prop del 0), i indiquen que no hi ha cap correlació amb l'atribut objectiu, essent la de menor valor la de l'atribut Sulfate. És evident que donada aquesta observació haurem de fer servir altres mètodes per trobar un punt d'afluència entre atributs.

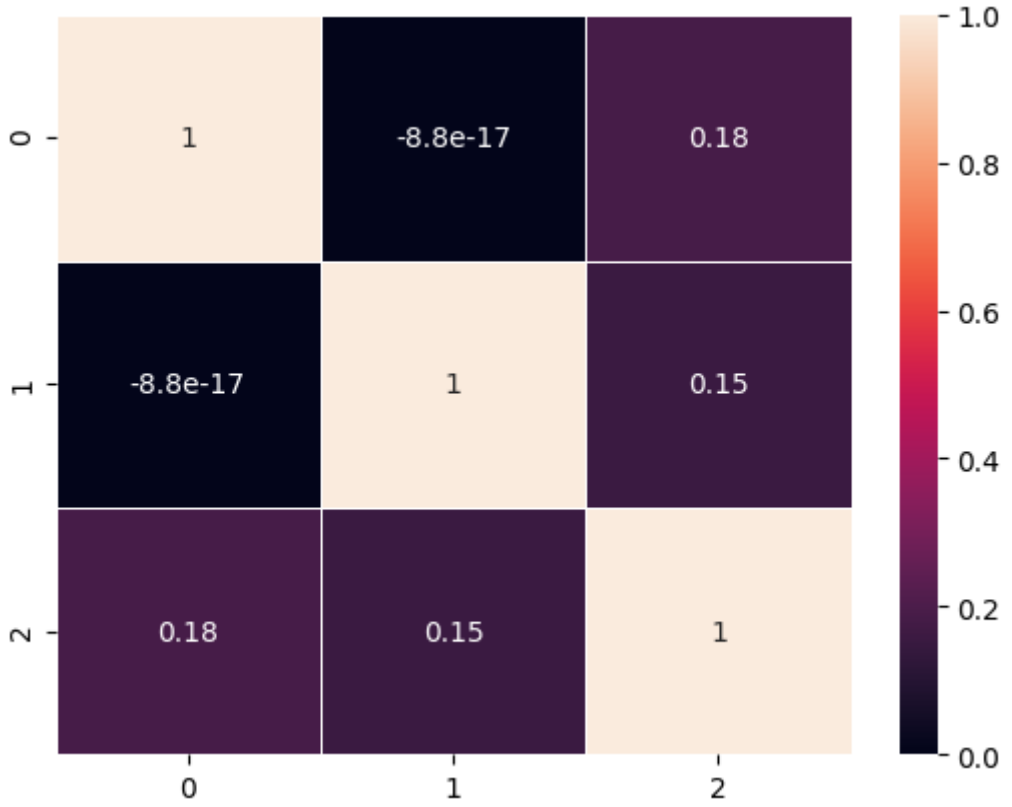
Preprocessing

Normalització

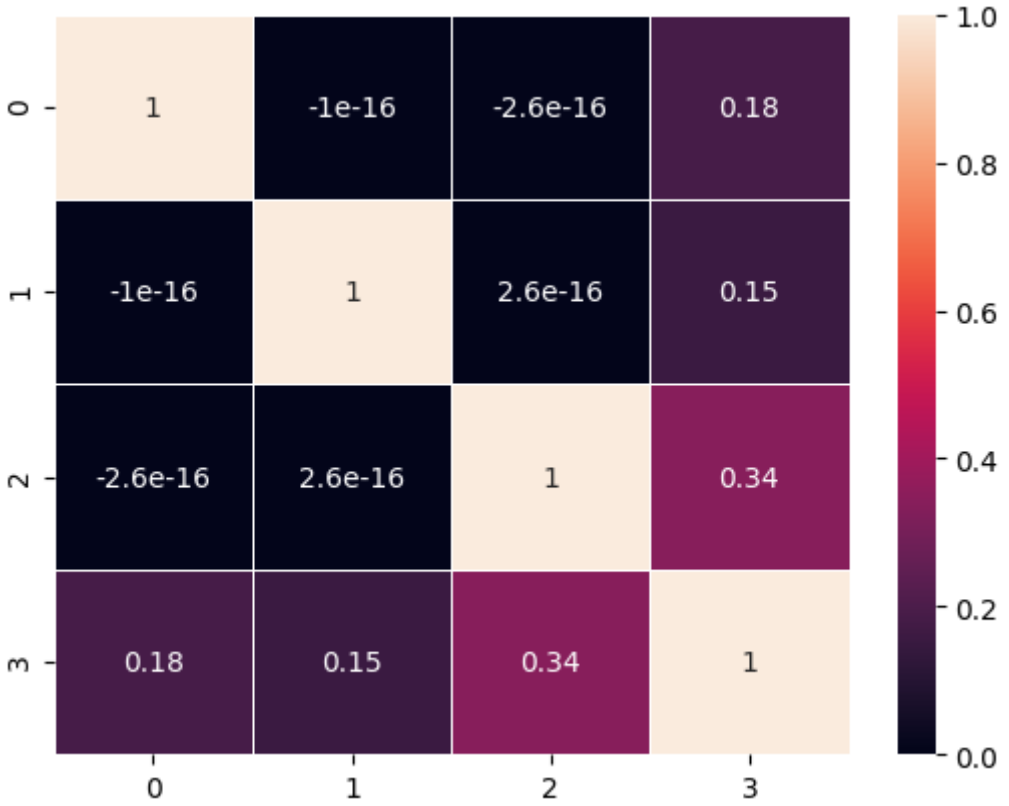
Tot seguit procedim a normalitzar les dades. En aquesta ocasió s'han trobat valors nuls incompatibles (NANs) en diverses mostres d'atributs de la base de dades, específicament en les columnes ph, Sulfate i Trihalomethanes, que representen un 14.99%, 23.84% i 4.94% de les respectives mostres. Ja que aquestes en conjunt representen més d'un 5% del total de mostres, s'ha decidit tractar totes aquestes calculant la mitjana aritmètica sobre la mateixa columna d'atributs on es troben i desar aquest valor a les cel·les afectades.

Donat que els valors obtinguts prèviament a la matriu de confusió no són gaire representatius, s'ha decidit aplicar un PCA (Principal Component Analysis). Amb aquest

mètode estadístic se simplifica el conjunt de dades que tenim abans sense alterar el nivell de representació de les mateixes agafant X característiques d'entrada i retornant Y components. Abans, però, cal estandarditzar les dades perquè cap dada tingui més dominància sobre la resta (mitjana a 0 i desviació estàndard a 1). Per a la nostra base de dades assignada s'ha reduït el nombre d'atributs a 2 i 3 (nombre de components recomanat). Un cop aplicada l'estandardització, aconseguim les matrius de correlació pertinents.



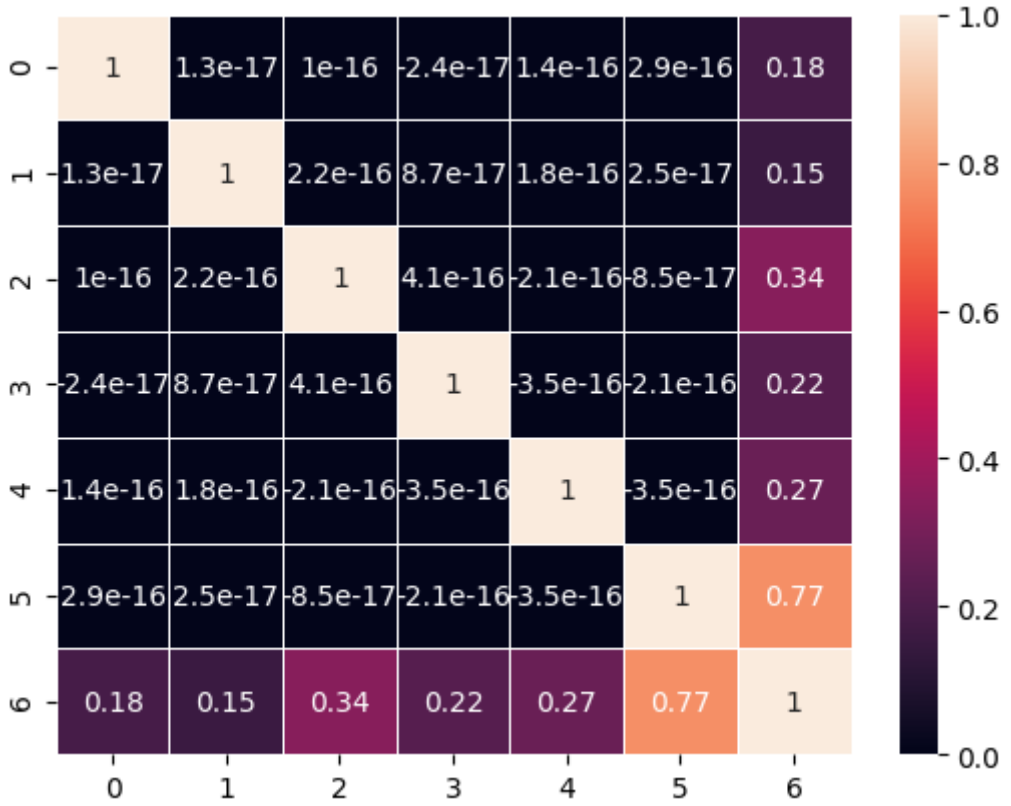
En el cas on el nombre de components és igual a 2, la matriu de correlació mostra uns valors que encara no són prou representatius, tal com passava a la matriu de correlació de les dades sense normalitzar, però que es troben per sobre del 0 (és a dir, tenen correlació positiva).



Si s'incrementa el nombre de components a 3, la matriu de correlació mostra valors que encara es troben per sobre del 0. Només hi ha un atribut que podria tenir relació amb l'atribut objectiu (4 és equivalent de Potability) que és l'atribut 2. Per corroborar això s'ha decidit provar els classificadors Logistic i SVM en aquest últim cas. A continuació trobem els següents resultats:

Correct classification Logistic 50 % of the data: 0.6910866910866911
Correct classification SVM 50 % of the data: 0.6758241758241759
Correct classification Logistic 70 % of the data: 0.7080366225839267
Correct classification SVM 70 % of the data: 0.6683621566632757
Correct classification Logistic 80 % of the data: 0.6870229007633588
Correct classification SVM 80 % of the data: 0.6732824427480916

Ara veurem què passa si incrementem el nombre de components a 6. La matriu de correlació resultant és la següent.



Excepcionalment, trobem que un dels atributs generats, el 6, arriba a un valor de correlació positiva elevat de 0.77, mentre que la resta són més petits i a prop del 0. També es manté invariant el valor de l'atribut 2, mantenint aquesta possible relació. Veiem ara què passa amb els resultats trobats amb els classificadors Logistic i SVM:

Correct classification Logistic 50 % of the data: 0.9975579975579976
Correct classification SVM 50 % of the data: 0.9926739926739927
Correct classification Logistic 70 % of the data: 0.9989827060020345
Correct classification SVM 70 % of the data: 0.987792472024415
Correct classification Logistic 80 % of the data: 0.9984732824427481
Correct classification SVM 80 % of the data: 0.9969465648854962

Es fa una classificació correcta del 99% amb el 0.5% de les dades. Això significa una millora considerable respecte l'anterior classificació amb menor nombre de components, però amb la hipòtesi que s'estigui produint overfitting sobre les dades, ja que són valors massa propers a l'1 dels quals anteriorment no tenien cap relació directa amb l'atribut objectiu.

Model Selection

Considerem com a models a seleccionar la regressió logística, ja que és possiblement el model més interessant pel nostre cas, aquest model està principalment pensat per a problemes binaris on volem trobar seguint unes dades si la classificació haurà de ser true o false, justament el que ens demana el problema.
També farem servir SVM, ens podrà ser útil, ja que busca la millor manera de classificar dades que es troben entre dues classes, ideal per poder decidir en quins casos s'haurà de classificar com a potable o no potable.

Per últim, utilitzarem el K-NearestNeighbour ideal per a trobar a quina classe pertany en funció de la distància entre punts.

Amb un % de dades mòbils i SVM amb kernel rbf i knn amb 2 neighbours trobem:

Correct classification Logistic 50 % of the data: 0.612942612942613
Correct classification SVM 50 % of the data: 0.6123321123321124
Correct classification KNN 50 % of the data: 0.575091575091575
Correct classification Logistic 70 % of the data: 0.6317395727365208
Correct classification SVM 70 % of the data: 0.6307222787385555
Correct classification KNN 70 % of the data: 0.5981688708036622
Correct classification Logistic 80 % of the data: 0.6051829268292683
Correct classification SVM 80 % of the data: 0.6067073170731707
Correct classification KNN 80 % of the data: 0.5838414634146342

Amb SVM amb kernel linear:
Correct classification Logistic 50 % of the data: 0.6257631257631258
Correct classification SVM 50 % of the data: 0.6208791208791209

Només hem pogut calcular amb el 50% de les dades, ja que donava errors constantment, com es pot veure no dista molt del kernel rbf. Podem observar que és molt lent, de fet és el més lent de tots.

Amb SVM amb kernel poli:
Correct classification Logistic 50 % of the data: 0.6172161172161172
Correct classification SVM 50 % of the data: 0.6172161172161172
Correct classification Logistic 70 % of the data: 0.6103763987792472
Correct classification SVM 70 % of the data: 0.6103763987792472
Correct classification Logistic 80 % of the data: 0.5929878048780488
Correct classification SVM 80 % of the data: 0.5929878048780488

Podem veure que aconseguim els mateixos resultats que amb Logístic, i per tant, possiblement els millors, ja que com es pot veure el classificador logístic obté els millors resultats de tots.

Amb SVM kernel sigmoid:
Correct classification Logistic 50 % of the data: 0.605006105006105
Correct classification SVM 50 % of the data: 0.5128205128205128
Correct classification Logistic 70 % of the data: 0.6225839267548321
Correct classification SVM 70 % of the data: 0.5361139369277721
Correct classification Logistic 80 % of the data: 0.5838414634146342
Correct classification SVM 80 % of the data: 0.5076219512195121

L’algorisme que millors resultats dona i més ràpid és el clasificador logístic

L'ús d'ensamble no ens resulta molt millor, de fet si fem bagging amb KNN, el resultat serà bastant baix:

Correct classification Logistic 50 % of the data: 0.608058608058608

Correct classification SVM 50 % of the data: 0.6074481074481074
Correct classification KNN 50 % of the data: 0.5726495726495726
Correct classification Ensemble 50 % of the data: 0.5848595848595849
Correct classification Logistic 70 % of the data: 0.5971515768056969
Correct classification SVM 70 % of the data: 0.5971515768056969
Correct classification KNN 70 % of the data: 0.5930824008138352
Correct classification Ensemble 70 % of the data: 0.5625635808748728
Correct classification Logistic 80 % of the data: 0.6051829268292683
Correct classification SVM 80 % of the data: 0.6067073170731707
Correct classification KNN 80 % of the data: 0.5960365853658537
Correct classification Ensemble 80 % of the data: 0.5777439024390244

El que fa és intentar realitzar una major generalització per al problema, per tant, la solució serà més robusta, però en casos més específics, no serà el més convenient.

Crossvalidation

Per què és important cross-validar els resultats?

La cross-validation consisteix en dividir les dades en dades de train i test. Si no fem això, utilitzarem les mateixes dades per a fer el model i per a testejar-lo, i tindrem un model pobre que només pot predir correctament les mateixes dades que hem usat per a fer el train (overfitting)

Provant el cross-validation i el K-fold obtenim aquests resultats:
Cross Validation Scores: [0.5929878 0.54198473 0.56946565 0.53282443 0.56793893]
Average CV Score: 0.5610403090672128
Number of CV Scores used in Average: 5

Cross Validation Scores: [0.55853659 0.54878049 0.60487805 0.57560976 0.55745721
0.50855746
0.63569682 0.55256724]
Average CV Score: 0.5677604508318922
Number of CV Scores used in Average: 8

Hem dut a terme proves amb split de 2 fins a 10 i trobem com a millor resultat el 8 i després el 5, són els resultats que en fer la mitjana surten més alts. En funció del valor de K trobarem un valor de Score millor o pitjor.

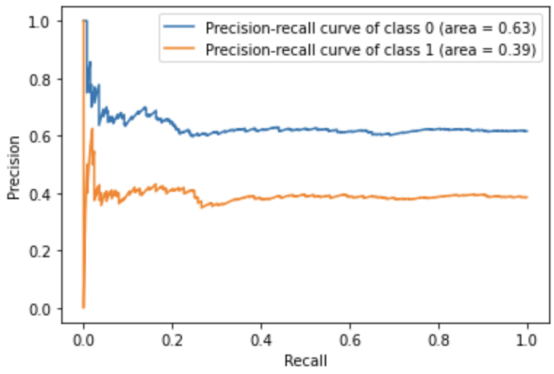
És viable o convenient aplicar LeaveOneOut?

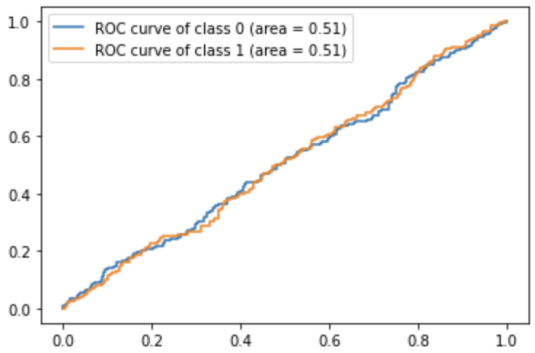
És viable però no massa convenient, ja que trigarà molt més que un K-fold per exemple.

Cross Validation Scores: [0. 0. 0. ... 0. 0. 1.]
Average CV Score: 0.590964590964591
Number of CV Scores used in Average: 3276

Podem observar com els resultats són millors als de K-fold. L'avantatge seria aquest, l'inconvenient és que ho fem 3276 cops.

Metric Analysis





En el nostre cas podem observar que tant la Precisió-Recall com la corba ROC varien bastant de l'objectiu. La millor pel nostre dataset seria la de precision recall, ja que podem veure amb claredat el tipus de dataset que tenim i que es troba desbalancellat.

Hyperparameter Search

Per trobar el millor paràmetre possible trobem diferents opcions:
Exhaustive Grid Search: genera candidats a partir d'un diccionari de paràmetres amb el seguent estil param_grid = [{'C': [1, 10, 100, 1000], 'kernel': ['linear']}, {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},]
Amb això realitza una "lluita" on provarà totes les combinacions per trobar els millors hiperparàmetres de cerca.

Randomized Parameter Optimization, implementa una cerca aleatòria de paràmetres, això genera dos avantatges: escollir el pressupost independentment del nombre de paràmetres i possibles valors, agregar paràmetres que no influeixin en el rendiment i no disminueix l'eficiencia. És més escalable, ja que en el cas de Grid Search la complexitat creix exponencialment a mesura que afegim nous paràmetres. Per tant, és més útil per a un temps concret.

També hi hauria l'opció de la força bruta sent interessant per a un rang molt petit de variables però molt tardat per a fer-ho amb un gran nombre de paràmetres.

Una opció a tenir en compte de scikit-optimize és la implementació de BayesSearchCV, és un model basat en el Grid Search, però amb una optimització bayesiana on s'utilitza un model predictiu per modelar els espais de cerca.

Github

https://github.com/sergidiazlopez/Practica2-Classificacio-GEI