# PRACTICA 2

**Aprenentatge Computacional (ApC)**

Pol Espinasa Vilarrasa - 1566792

Marc Gonzalez Amores - 1564995

## APARTAT A: Comparativa de models (4pts)

En el primer apartat d'aquesta pràctica hem utilitzat una base de dades de la qualitat del aigua, la qual hem analitzat i mesurat els diferents models per aquesta. En primer lloc començarem introduint les dades de la nostra base de dades:

La nostra base de dades conté mètriques de qualitat de l'aigua per a 3276 masses d'aigua diferents. La explicarem amb més detall en el següent apartat de la pràctica.

### Comparació dels models

Per tal de comparar-los hem aplicat regressió logística, la màquina de vectors de suport (SVM), el KNN i Decision Tree.

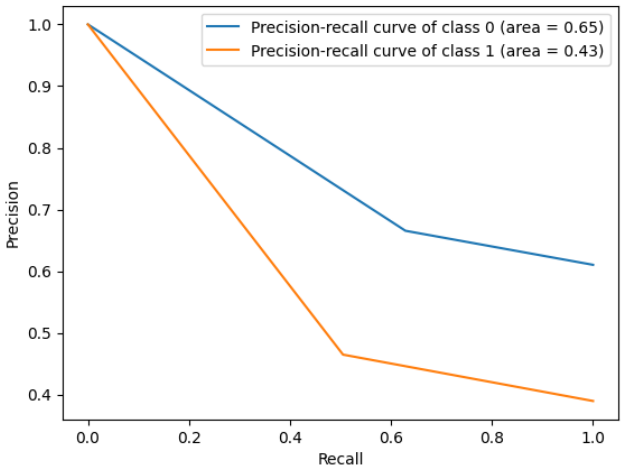
Per això hem hagut d'entrenar els models amb percentatges de dades d'entrenament i de test diferents, i així poder observar la influencia de la quantitat de dades de traint al model. Aquests són els resultats que hem obtingut:

| | Regressió | SVM | KNN | Decision Tree |
|---|---|---|---|---|
| 50 % train | 0.59 | 0.58 | 0.61 | 0.61 |
| 70 % train | 0.61 | 0.59 | 0.63 | 0.63 |
| 80 % train | 0.59 | 0.60 | 0.65 | 0.63 |

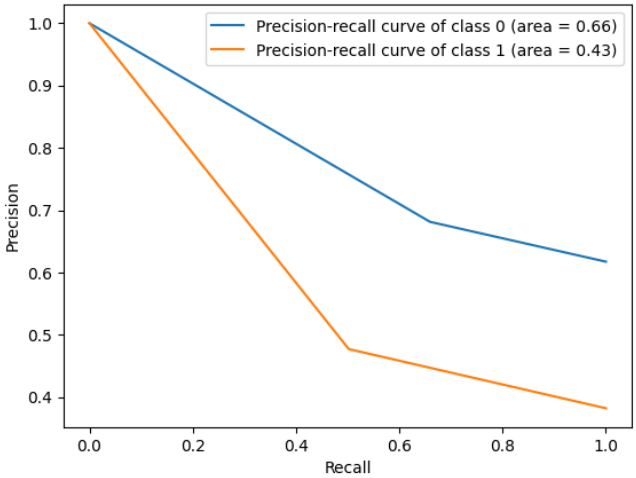
Com es pot observar tots els models ens donen valors al voltant del 60% sent el més alt un 65% d'acuracy al KNN amb un 80% de train..

A continuació hem creat les rectes ROC i PR en cada model i per un train del 50, 70 i 80%:

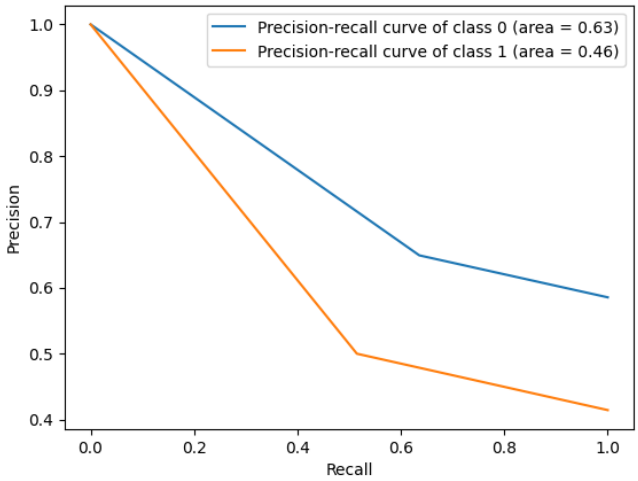
**PR Decision Tree**




50 %

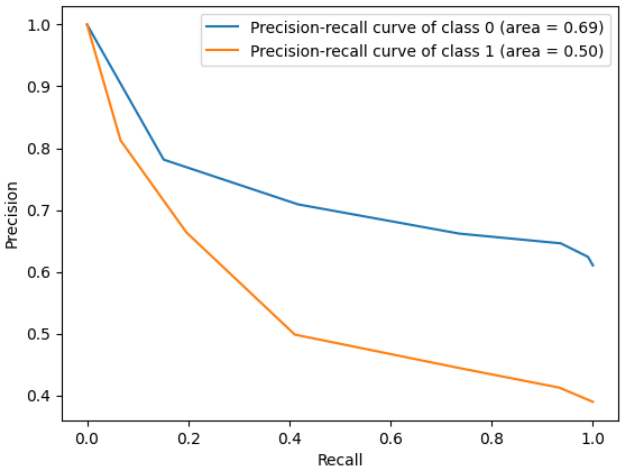



70%

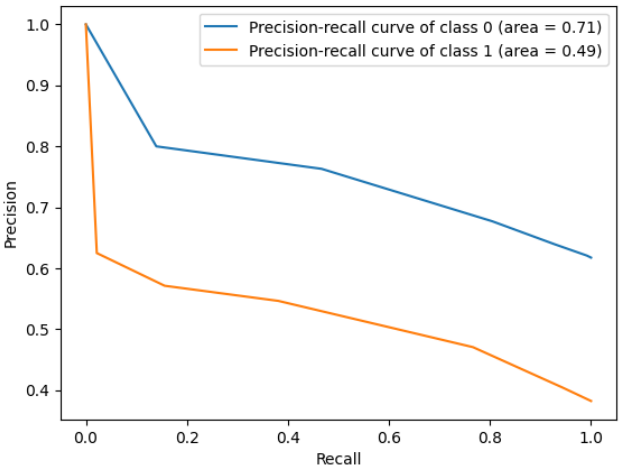



80 %

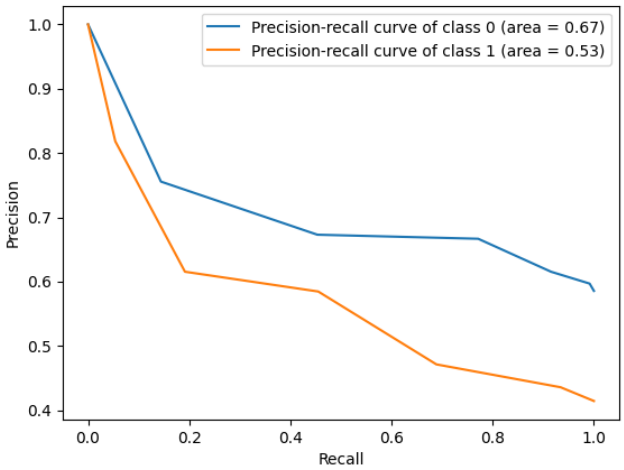
**PR KNN**




50 %

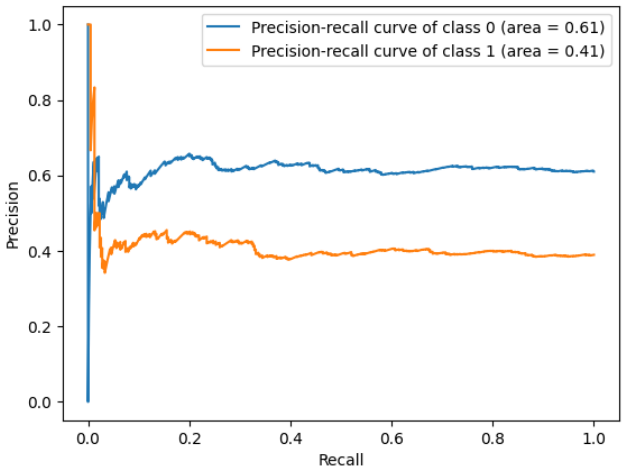



70%

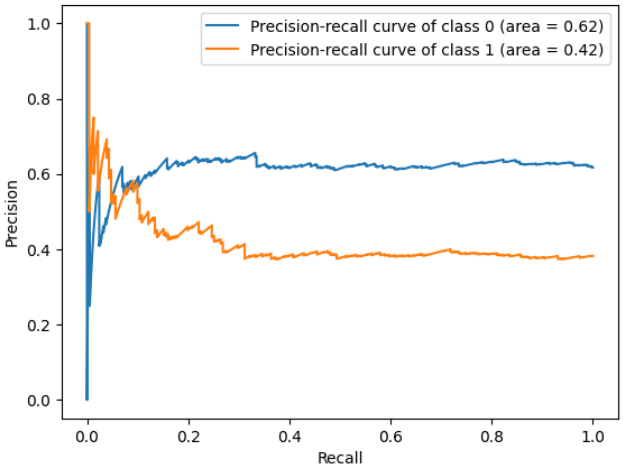



80 %

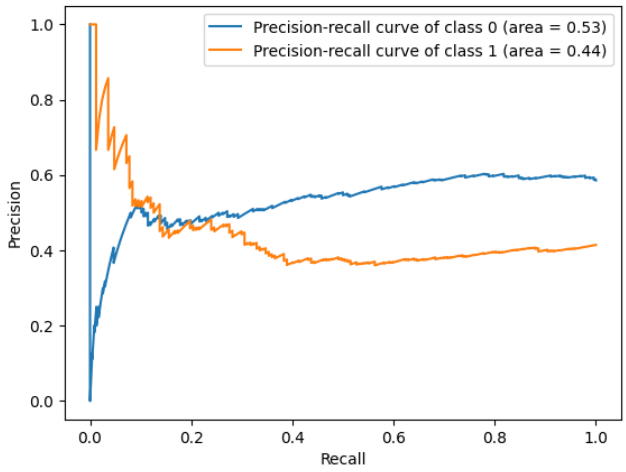
**PR Regressor Logístic**




50 %

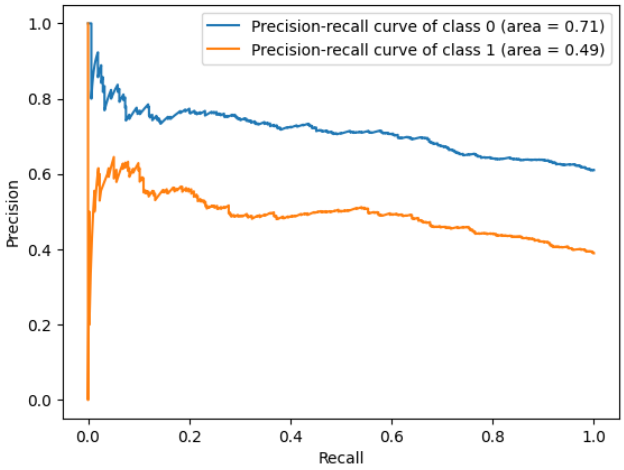



70 %

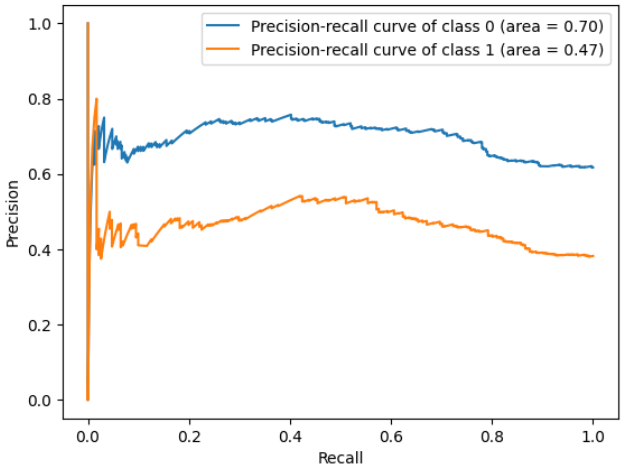



80%

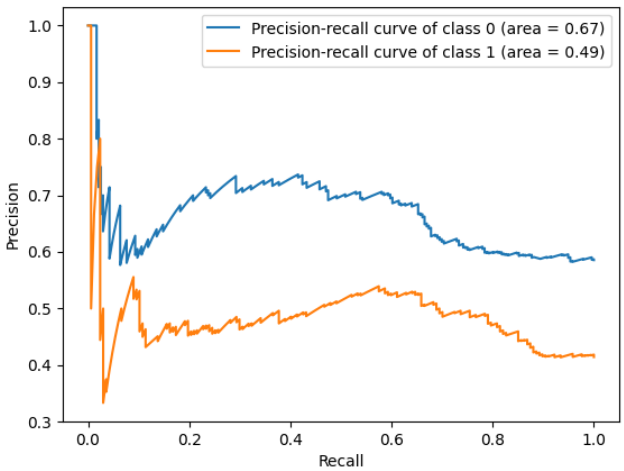
**PR SVM**




50 %

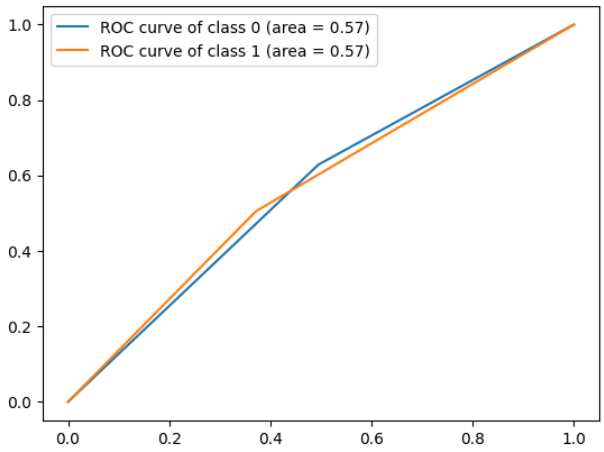



70 %

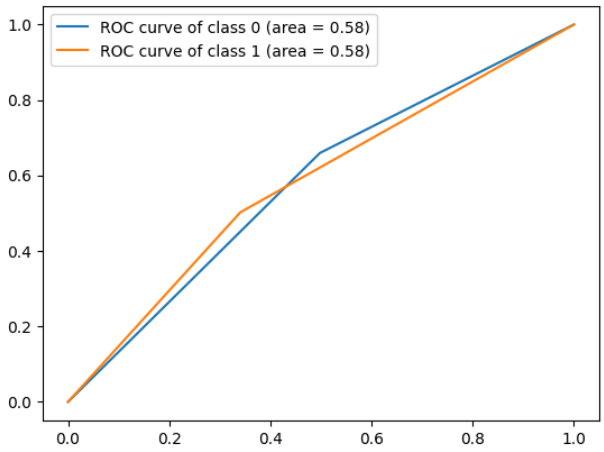



80 %

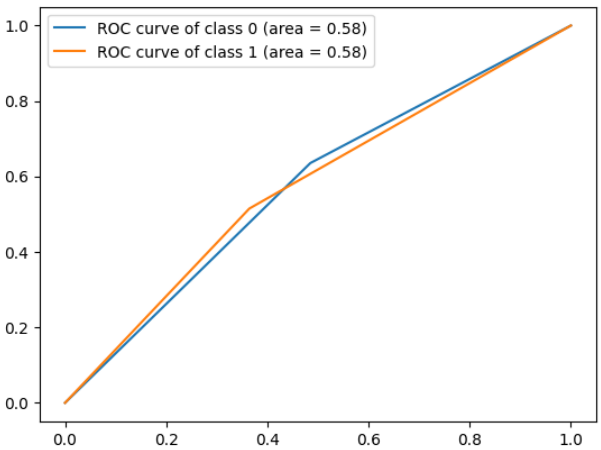
**ROC Decision Tree**




50 %

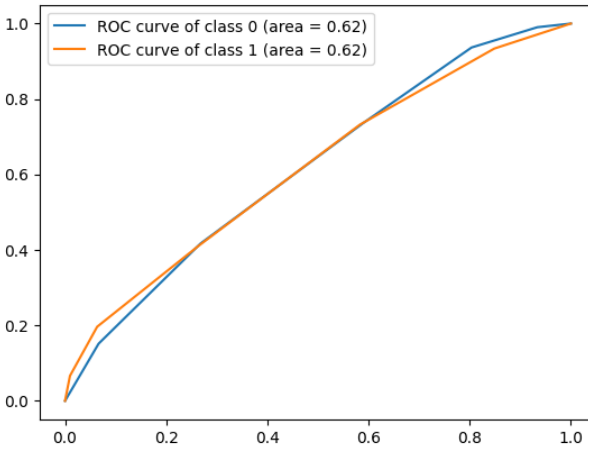



70 %

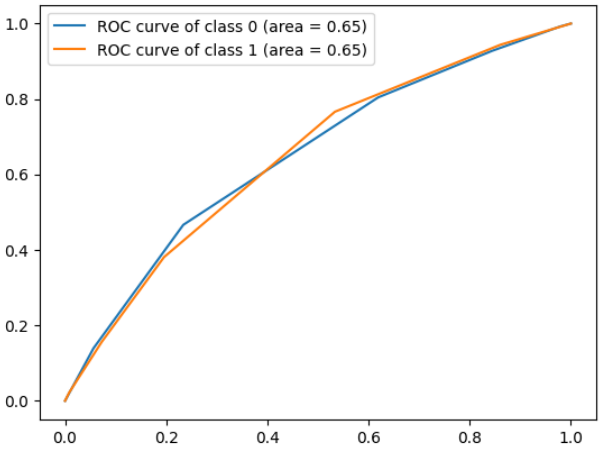



80 %

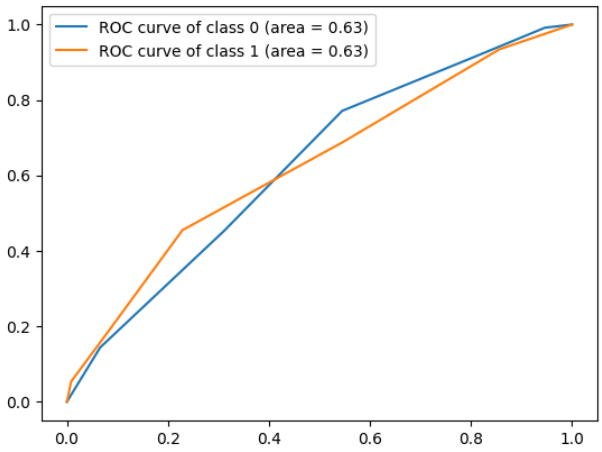
**ROC KNN**




50 %

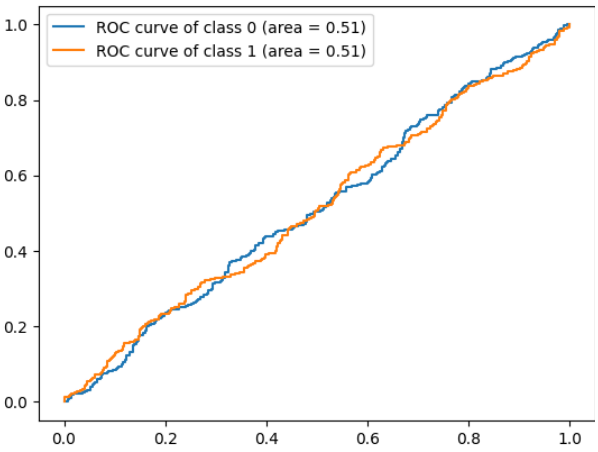



70 %

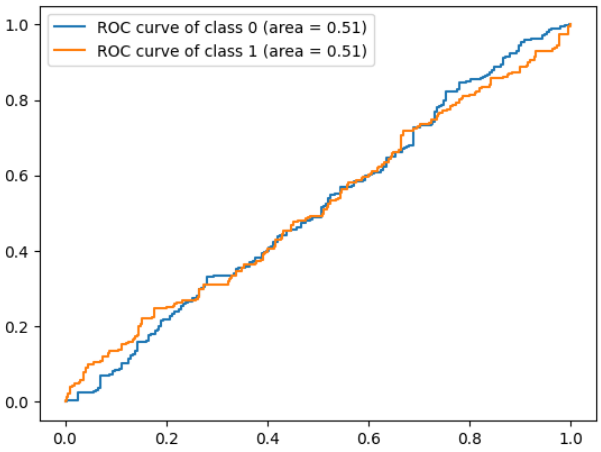



80 %

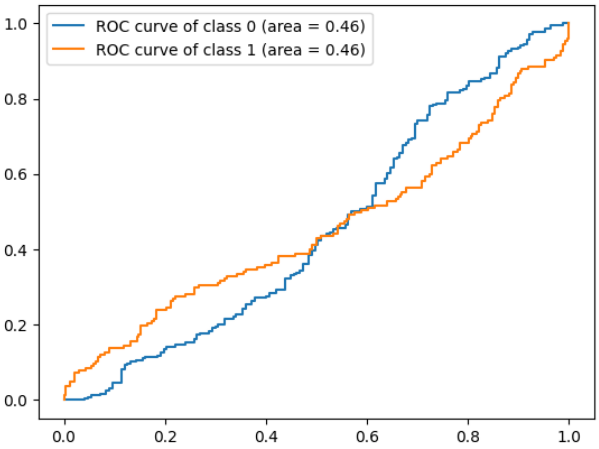
**ROC Regressor Logístic**




50 %

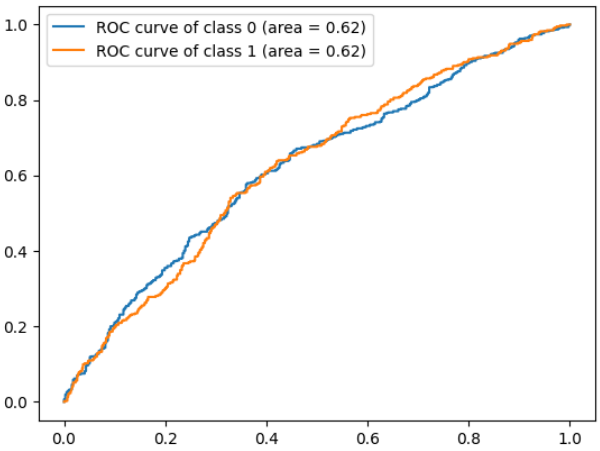



70 %

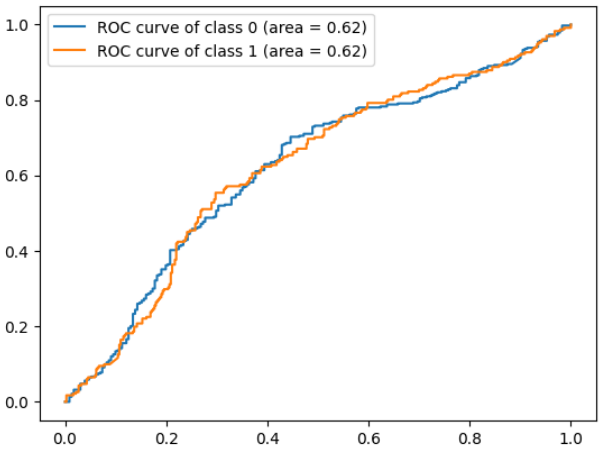



80 %

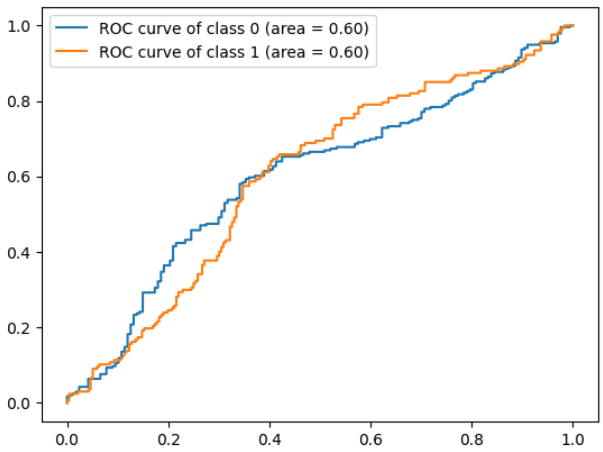
**ROC SVM**

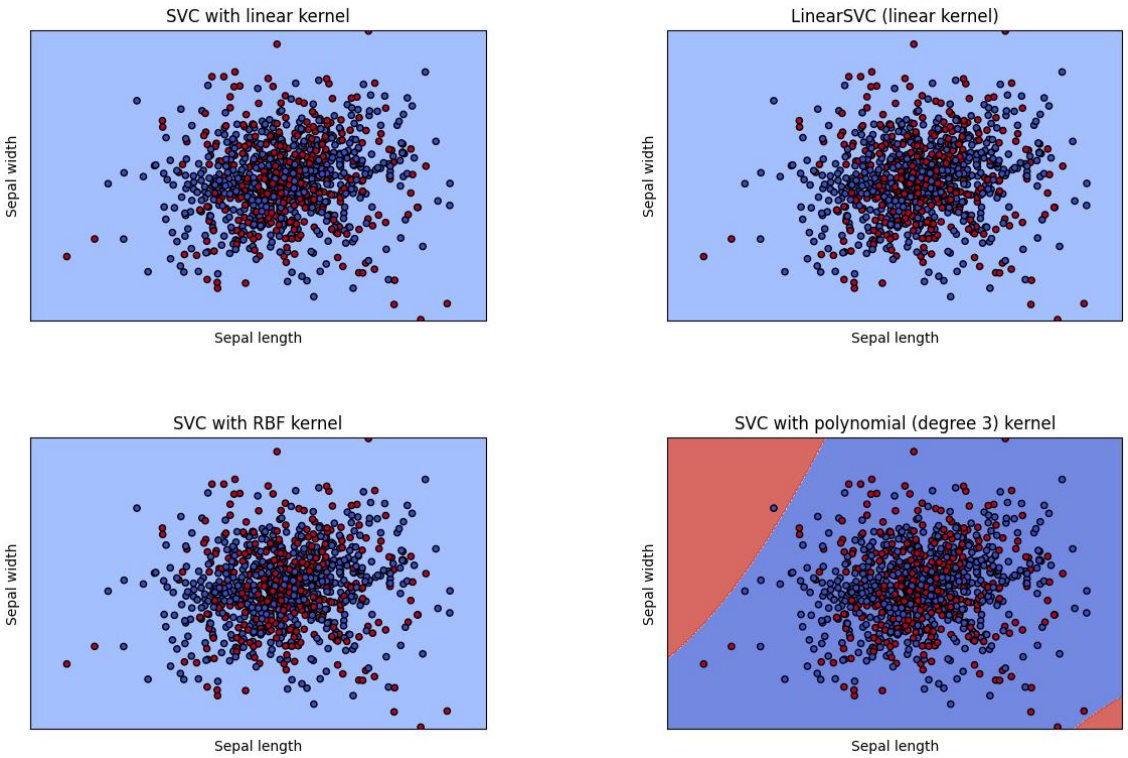

ROC curve of class 0 (area = 0.62)
ROC curve of class 1 (area = 0.62)
0.0
0.2
0.4
0.6
0.8
1.0

50 %


ROC curve of class 0 (area = 0.62)
ROC curve of class 1 (area = 0.62)
0.0
0.2
0.4
0.6
0.8
1.0

70 %


ROC curve of class 0 (area = 0.60)
ROC curve of class 1 (area = 0.60)
0.0
0.2
0.4
0.6
0.8
1.0




Per finalitzar, hem mostrat gràficament com el SVC classifica les classes amb els diferents kernels. Per defecte el kernel que s'utilitza és el RBF, però hem provat amb diferents kernels com el linear o el polinòmic.

## Apartat (B): Classificació Numèrica (6pts)

En aquest apartat analitzarem la nostre base de dades, aquesta tracta sobre la potabilitat de l'aigua depenent de diferents atributs i factors que posteriorment veurem.

Water Quality | Kaggle

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

| **SVM** | **Precisió** | **C** | **Fit_intercept** | **penalty** | **tol** |
|---|---|---|---|---|---|
| **Bàsica** | | | | | |
| **50% train, 50% test** | 0.588 | 1.0 | True | l2 | 0.0001 |
| **80% train, 20% test** | 0.607 | 1.0 | True | l2 | 0.0001 |
| **70% train, 30% test** | 0.586 | 1.0 | True | l2 | 0.0001 |
| | | | | | |
| **K-Fold** | | | | | |
| **K = 2** | 0.601 | 1.0 | True | l2 | 0.0001 |
| **K = 3** | 0.599 | 1.0 | True | l2 | 0.0001 |
| **K = 4** | 0.599 | 1.0 | True | l2 | 0.0001 |
| **K = 5** | 0.597 | 1.0 | True | l2 | 0.0001 |
| | | | | | |
| **LOOCV** | 0.597 | 1.0 | True | l2 | 0.0001 |

| **Perceptron** | **Precisió** | **alpha** | **Fit_intercept** | **penalty** | **tol** |
|---|---|---|---|---|---|
| **Bàsica** | | | | | |
| **50% train, 50% test** | 0.509 | 0.0001 | True | None | 0.001 |
| **80% train, 20% test** | 0.506 | 0.0001 | True | None | 0.001 |
| **70% train, 30% test** | 0.556 | 0.0001 | True | None | 0.001 |
| | | | | | |
| **K-Fold** | | | | | |
| **K = 2** | 0.494 | 0.0001 | True | None | 0.001 |
| **K = 3** | 0.514 | 0.0001 | True | None | 0.001 |
| **K = 4** | 0.517 | 0.0001 | True | None | 0.001 |
| **K = 5** | 0.510 | 0.0001 | True | None | 0.001 |

| | | | | | |
|---|---|---|---|---|---|
| **LOOCV** | 0.512 | 0.0001 | True | None | 0.001 |

| **KNN** | **Precisió** | **Leaf_size** | **N_neighbours** | **Mètric** | **weights** |
|---|---|---|---|---|---|
| **Bàsica** | | | | | |
| **50% train, 50% test** | 0.621 | 30 | 5 | Minkowski | uniform |
| **80% train, 20% test** | 0.617 | 30 | 5 | Minkowski | uniform |
| **70% train, 30% test** | 0.637 | 30 | 5 | Minkowski | uniform |
| | | | | | |
| **K-Fold** | | | | | |
| **K = 2** | 0.571 | 30 | 5 | Minkowski | uniform |
| **K = 3** | 0.589 | 30 | 5 | Minkowski | uniform |
| **K = 4** | 0.564 | 30 | 5 | Minkowski | uniform |
| **K = 5** | 0.585 | 30 | 5 | Minkowski | uniform |
| | | | | | |
| **LOOCV** | 0.512 | 30 | 5 | minkowski | uniform |

| **Decision Tree** | **Precisió** | **criterion** | **splitter** | **max_leaf_nodes** | **max_depth** |
|---|---|---|---|---|---|
| **Bàsica** | | | | | |
| **50% train, 50% test** | 0.559 | gini | best | None | None |
| **80% train, 20% test** | 0.615 | gini | Best | None | None |
| **70% train, 30% test** | 0.586 | gini | Best | None | None |
| | | | | | |
| **K-Fold** | | | | | |
| **K = 2** | 0.601 | gini | best | None | None |
| **K = 3** | 0.598 | gini | best | None | None |
| **K = 4** | 0.599 | gini | best | None | None |
| **K = 5** | 0.597 | gini | best | None | None |
| | | | | | |
| **LOOCV** | 0.597 | gini | best | None | None |

| **Logistic Regression** | **Precisió** | **C** | **Fit_intercept** | **penalty** | **tol** |
|---|---|---|---|---|---|
| **Bàsica** | | | | | |
| **50% train, 50% test** | 0.509 | 1.0 | True | l2 | 0.0001 |
| **80% train, 20% test** | 0.506 | 1.0 | True | l2 | 0.0001 |
| **70% train, 30% test** | 0.556 | 1.0 | True | l2 | 0.0001 |
| | | | | | |
| **K-Fold** | | | | | |
| **K = 2** | 0.601 | 1.0 | True | l2 | 0.0001 |
| **K = 3** | 0.598 | 1.0 | True | l2 | 0.0001 |
| **K = 4** | 0.599 | 1.0 | True | l2 | 0.0001 |
| **K = 5** | 0.597 | 1.0 | True | l2 | 0.0001 |
| | | | | | |
| **LOOCV** | 0.597 | 1.0 | True | l2 | 0.0001 |

## Exploratory Data Analysis

En aquesta base de dades tenim un total de 10 columnes, és a dir 10 atributs, treballarem amb aquests per acabar predint la potabilitat de l'aigua.

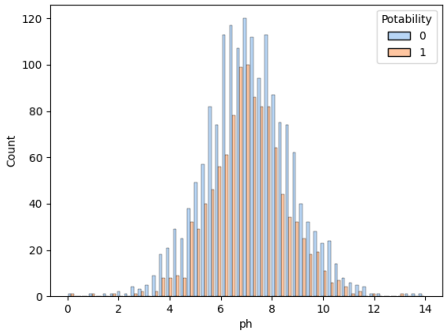
*ppm → part per milió. 1ppm = 1mg/L

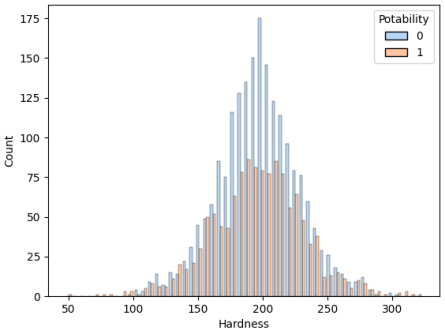
- Ph: pH de l'aigua.
- Hardness: Capacitat de l'aigua de precipitar sabó en mg/L.
- Solids: Total de sòlids dissolts en ppm.
- Chloramines: Quantitat de cloramines en ppm.
- Sulfate: Quantitat de sulfats dissolts en mg/L.
- Conductivity: Conductivitat de l'aigua en µS/cm.
- Organic_carbon: Quantitat de carbó orgànic en ppm.
- Trihalomethanes: Quantitat de Trihalometà en µg/L.
- Turbidity: Mesura de la propietat d'emissió de llum de l'aigua en NTU (Nephelometric Turbidity Units).
- Potability: Indicació si l'aigua és potable o no.

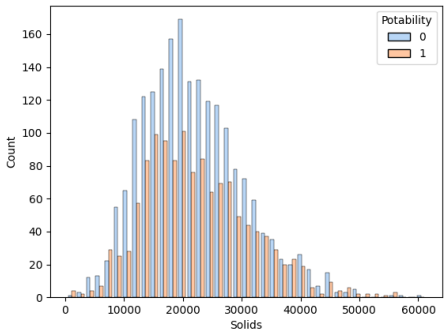
Tots els atributs són numèrics, menys la Potabilitat que és un valor binari.

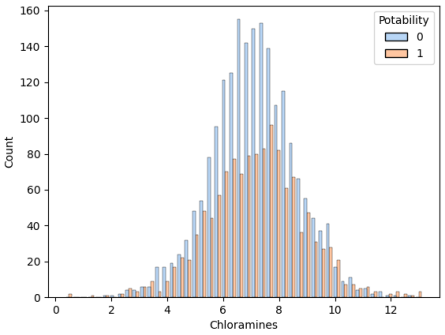
Per poder entendre millor els atributs i com influeixen a la potabilitat de l'aigua, generarem uns histogrames dels atributs i unes gràfiques de dispersió:

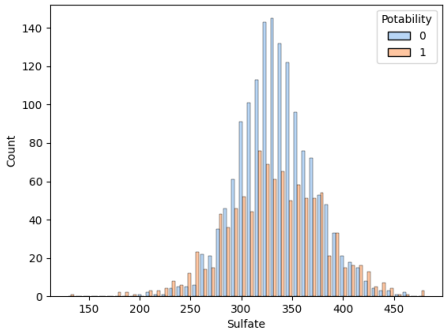
## HISTOGRAMES DELS ATRIBUTS

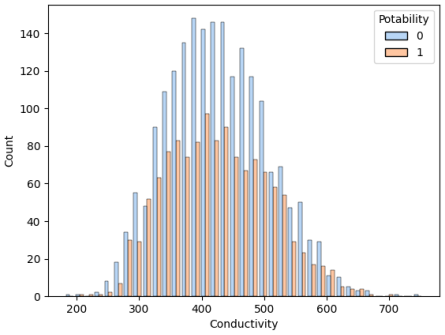


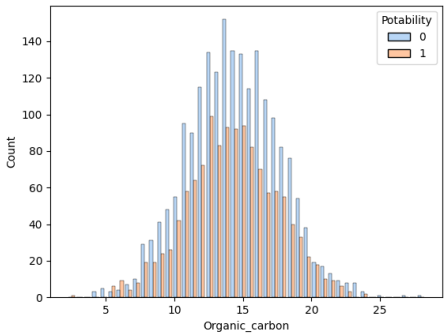



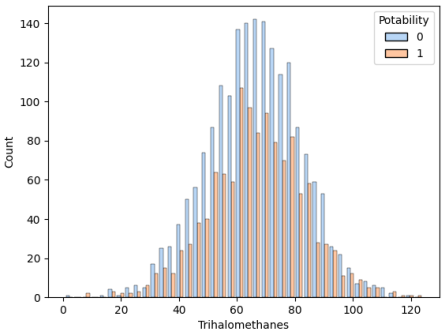



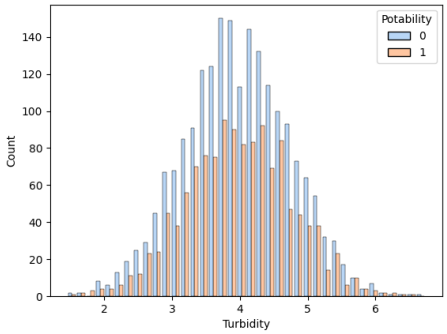



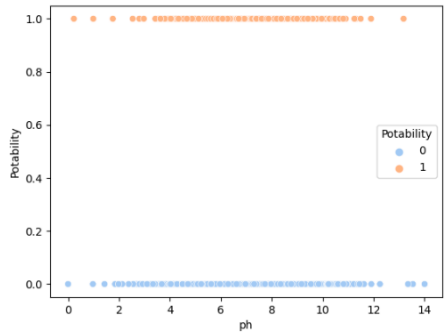



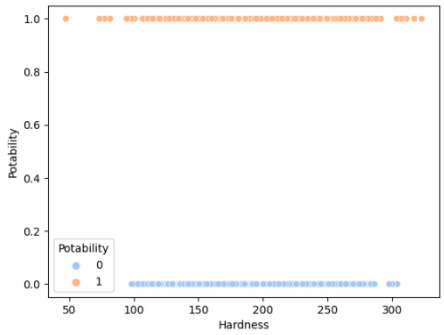


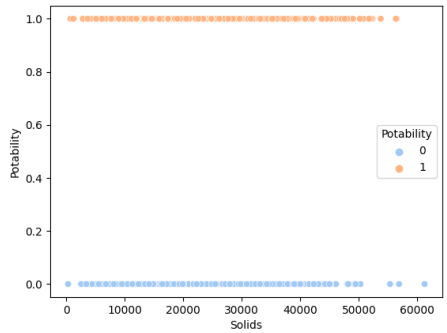



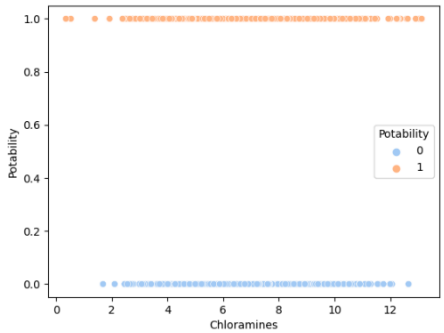



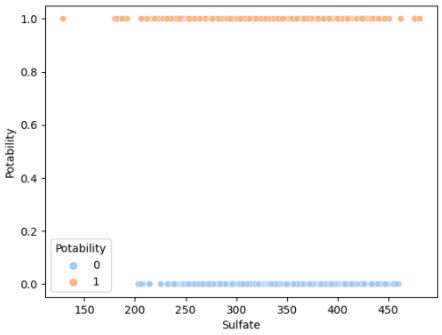

# GRÀFIQUES DE DISPERSIÓ

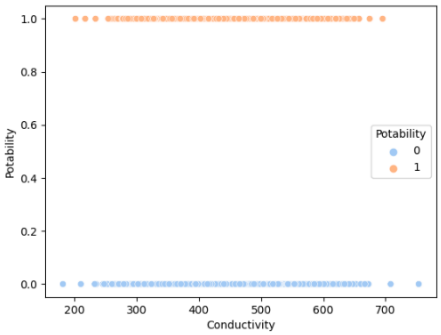


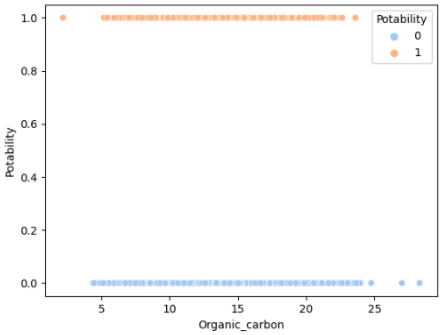



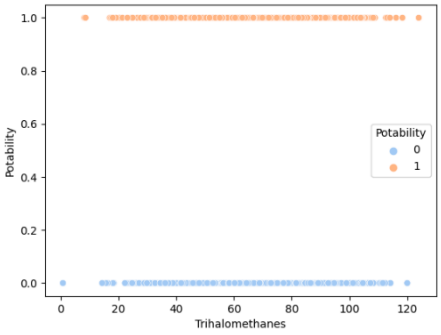



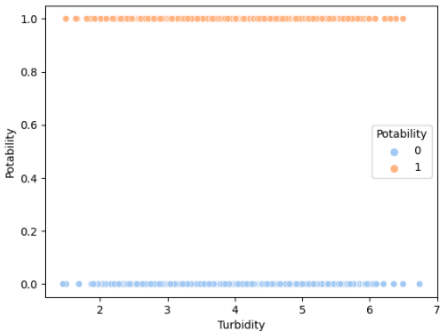






## Preprocessing

Podem veure és que els valors són molt dispersos per tant el que podríem fer és escalar les dades, per tenir rangs de valors més similars.

Amb això aconseguim donar un pes similar a tots els atributs, tot i que això no apliqui per a tots els models. Ja que alguns no es basen en els valors dels atributs sinó en rangs, etc.

A part hem eliminat les files que contenien valors NaN, per tenir més facilitat a la hora de treballar amb les dades.

## Model Selection

En aquest apartat, utilitzarem les taules del inici del apartat, on tenim els resultats obtinguts en els diferents models que hem utilitzat.

Aquests models són el Perceptró, KNN, SVM, Logistic Regression i el Decision Tree. Per cada model analitzem els resultats dividint el data set en diferents percentatges. Cal destacar que hem executat els models amb els paràmetres per defecte.

Observant els resultats, el model que millors resultats ens dona és el KNN utilitzant un 70% de les dades per training i un 30% per test, el resultat és d'un acuracy del 63,7%, els altres ens donen resultats una mica menors a aquest.

Utilitzat ensemble no tindria gaire sentit ja que aquest agafa diversos classificadors simples per formar-ne un de més complex. Els nostres ja són suficientment complexos i en alguns casos cars d'utilitzar com per a sobre ajuntar-los.

## Crossvalidation

El crossvalidation és una tècnica utilitzada per analitzar el rendiment d'un model entrenat utilitzant un conjunt de validació. Per cada iteració aquest conjunt va canviant ja que hem d'entrenar el model i validar-lo per cada subconjunt. És a dir en el cas k= 6, dividim el model en 6 parts, 5 per entrenar i l'altra per fer la validació, això es repetirà 6 cops fins que tots els subconjunts hagin estat conjunt de validació. Es retorna la mitjana de rendiment per cada model.

Els resultats que hem obtingut d'aplicar això a tots els nostres models ens mostren que tenim diversos models amb un percentatge d'acuracy idèntic, 60,1%, aquests serien: SVM, Decission Tree i Logic Regression. Obtenim millors resultats com més baixa es la K, cosa que no té gaire sentit ja que, com més dividim el data set més dades d'entrenament tindrem. Cal remarcar però que la diferencia entre els diferents valors de K és molt baixa i podríem fins i tot depreciar els resultats a l'hora de prendre decisions.

Hem aplicat la tècnica de leave on out, però hem vist que no era la més eficient, és més costosa computacionalment parlant i els resultats obtinguts són casi iguals als entrenaments previs.

## Mètriques de avaluació

En aquest apartat analitzarem els 6 models creats i juntament amb altres tècniques d'optimització d'hiperparàmetres per cada model, determinarem quins model és el més fiable.

Per fer aquest anàlisi calcularem unes mètriques d'avaluació per cada model, per després comparar-les amb les corves ROC i PR, per tal de determinar el millor model. Podem utilitzar mètriques com per exemple: precision , accuraccy, , recall, f1 score...

En aquesta taula podem veure cada model amb les mètriques utilitzades per l'avaluació:

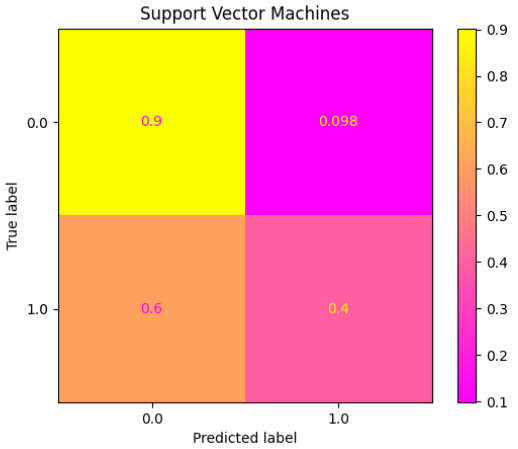
| **Models** | **Accuracy** | **Precision** | **F1 Score** | **Recall** |
|---|---|---|---|---|
| **Support Vector Machine** | 0.692 | 0.667 | 0.705 | 0.692 |
| **Perceptron** | 0.464 | 0.459 | 0.471 | 0.464 |
| **KNN** | 0.655 | 0.640 | 0.645 | 0.655 |
| **Decision Tree** | 0.600 | 0.602 | 0.604 | 0.600 |
| **Logistic Regression** | 0.627 | 0.491 | 0.767 | 0.627 |

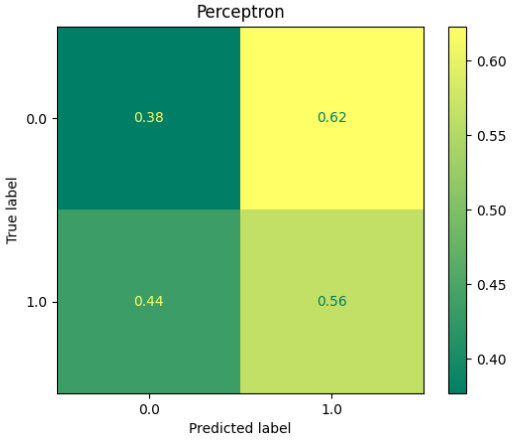
A continuació explicarem en detall cadascuna de les mètriques que hem utilitzat per realitzar l'anàlisi:

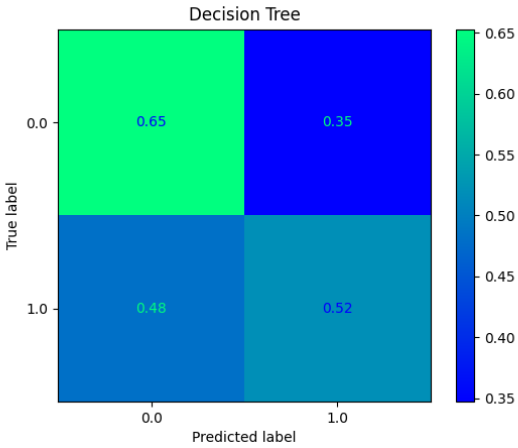
- Accuraccy: S'utilitza per mesurar el ratio de verdaders positius + verdaders negatius / total.
- Precision: S'utilitza per mesurar el ratio de verdaders positius / falsos positius + verdaders positius. És la capacitat del classificador en no etiquetar com a positiu una mostra que és negativa.
- Recall: Mesura el ratio de verdaders positius / falsos negatius + verdaders positius. És la capacitat del classificador en trobar totes les mostres positives.
- f1 score: Podríem dir que es una barreja de precisió i el recall.

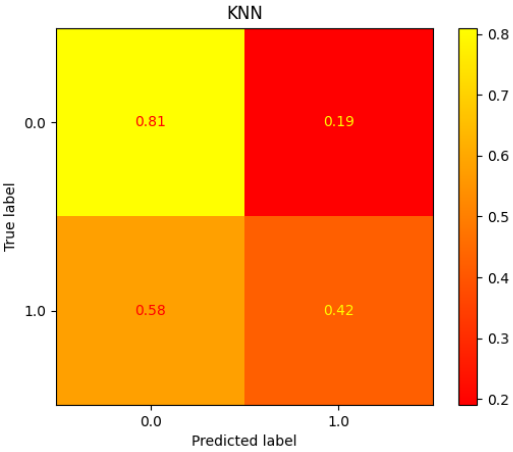
Podem observar a la taula obtinguda, que el millor classificador és Suport Vector Machine amb un 69,2% d'accuracy.

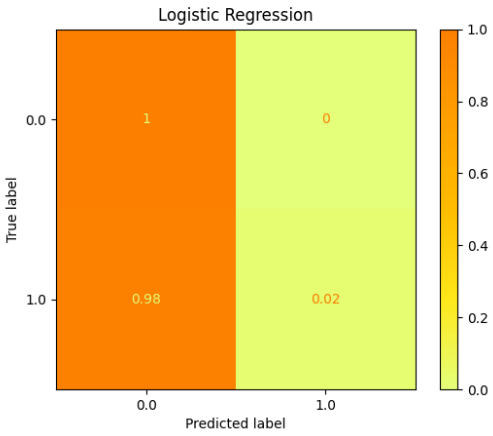
Les matrius de confusió son les següents:




En les matrius de confusió podem observar com en la Regressió logística per als atributs de la primera classe fa una predicció perfecte, mentre que alhora de predir la segona classe fa una molt dolenta.

Un altre anàlisi que podem fer seria el de les corbes ROC i PR per tal de fer la comparació dels models. En les corbes PR es descriu la relació entre el recall i la precisió del model. Volem que el recall i la precisió sigui de 1 per tant com més a dalt a la dreta es trobi millor. Podem observar que les millors serien la del KNN i el SVM amb un 70%. Per l'altra banda tenim les corbes ROC el rati de vertaders positius i el rati dels falsos positius. Ens interessa minimitzar el rati de falsos positius i que el rati de verdaders positius sigui el màxim, per tant com més a l'esquerra i a dalt estigui millor. Trobem que el KNN i el SVM tornen a ser els millors amb un 60%.

## Hyperparameter search

Un cop hem vist el rendiment dels models amb els paràmetres per defecte, prosseguim a buscar paràmetres que millorin el rendiment dels classificadors.

Per trobar aquests paràmetres la llibreria sklearn implementa alguns mètodes. Aquests són: GridSearchCV i RandomizedSearchCV. El primer troba els millors paràmetres per un model a partir d'un diccionari amb els possibles valors que poden prendre els paràmetres. En canvi el RandmizedSearchCV afegeix un component d'atzar per trobar els millors resultats, el cost és menor ja que es salta algunes combinacions, però tècnicament això pot implicar no trobar la millor combinació de paràmetres.

Hem utilitzat els dos i al revés del que ens pensàvem el RandomizedSearchCV ens ha donat millors resultats, així que ens hem quedat amb els resultats d'aquests.

A la taula següent es poden veure els diferents paràmetres dels diccionaris:

| | |
|---|---|
| Support Vector Machine | - 'C': [0.1,1, 10, 100]<br>- 'random_state': [0,10,30,90] |
| KNN | - 'n_neighbors' : [5,9,11,15],<br>- 'weights' : ['uniform','distance'],<br>- 'metric':['minkowski','euclidean','manhattan'] |
| Decision Tree | - 'max_depth': [2, 5, 10, 20],<br>- 'min_samples_leaf': [5, 10, 20, 50, 100],<br>- 'criterion': ["gini", "entropy"] |
| Logistic Regression | - 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000] |
| Perceptron | - 'alpha': [0.0001, 0.001, 0.01, 0.1, 1]<br>- 'fit_intercept': [True, False]<br>- 'shuffle': [True, False] |

Els resultats obtinguts són els següents:

| | |
|---|---|
| Support Vector Machine | - 'random_state': 90<br>- 'C': 10 |
| KNN | - 'weights': 'uniform'<br>- 'n_neighbors':15<br>- 'metric': 'minkowski' |
| Decision Tree | - 'min_samples_leaf': 10<br>- 'max_depth': 5<br>- 'criterion': 'gini' |
| Logistic Regression | - 'C': 0.1 |
| Perceptron | - 'shuffle': False<br>- 'fit_intercept': False<br>- 'alpha': 0.0001 |

Comparem ara els scores obtinguts amb els paràmetres per defecte i amb els paràmetres escollits per la llibreria de sklearn:

| | Paràmetres per defecte | Millors Paràmetres |
|---|---|---|
| SVM | 0.692 | 0.598 |
| KNN | 0.655 | 0.626 |
| Decision Tree | 0.600 | 0.620 |
| Logistic Regression | 0.627 | 0.597 |
| Perceptron | 0.464 | 0.514 |

Com podem veure en la taula només milloren dos models, Decision Tree i Perceptron, la resta empitjoren. Això pot ser degut a que no hem escollit els millors paràmetres perquè provi o simplement perquè els valors per defecte són millors i no estaven inclosos en el diccionari.

## Conclusions

Podem veure al acabar aquesta pràctica que els nostres models per aquesta base de dades no aconsegueixen en cap cas un accuracy remarcable, quedant-se la més alta amb un score de 0.692 lo qual no és gaire bon resultat. El que intentem veure en la nostra base de dades és si

l’aigua és potable, només donant un cop d’ull a la base de dades podem veure que els valors dels atributs son molt similars tant si l’aigua es potable com si no, això sumat a que la base de dades no era gaire gran, unes 3 mil mostres, pot conduir a aquests resultats tant baixos.