Veille sur la *Logistic Regression*

Julien ABOUTARD
Marwin LAUNAY

DevIA#2 – 24/10/2022

I. Qu'est-ce la *Logistic Regression*

1. Présentation de la *logistic regression*

a. Fondamentaux de la *Logistic Regression*

La *Logistic Regression* est un modèle statistique qui sert à déterminer la probabilité qu'un événement se produise. Elle montre la relation entre les caractéristiques, puis calcule la probabilité d'un certain résultat.

Le test de *Logistic Regression* simple est utilisé lorsqu'il s'agit d'expliquer une variable qualitative binaire (variable dépendante) par plusieurs variables indépendantes quantitatives.

C'est une technique prédictive dont l'objectif est de produire un modèle capable de prédire et/ou d'expliquer les valeurs de la variable qualitative.

Il existe deux grands types de *Logistic Regression*, La Binaire et La Multinomial puis nous avons l'Ordinale qui est un sous type de la Multinomial

Si cette dernière est binaire, ce qui arrive le plus souvent, on décrit le test comme étant une *Logistic Regression* binaire ou une classification binaire.

Au contraire si la variable qualitative possède plus de deux modalités qui, ne seront pas dans un ordre quelconque, le test est décrit comme une *Logistic Regression* multinomiale/polytonique ou une classification multi-classes.

La *Logistic Regression* ordinale comme la régression multinomiale peut contenir trois variables ou plus, cependant, les mesures doivent respecter un certain ordre.

Par exemple, la notation d'un hôtel sur une échelle de 1 à 5.

b. Secteur d'application fréquent

La *Logistic Regression* peut être appliquée à plusieurs domaines, mais elle en possède quelques-uns de prédilection comme la santé, la politique, le e-commerce, le marketing, les tests de produits ou bien la finance.

Santé - Exemple, le Trauma and Injury Severity Score (TRISS), il utilise des variables telles que le score révisé de traumatisme, le score de gravité des blessures et l'âge du patient pour prédire les résultats de santé des maladies comme le diabète de plus les maladies cardiaques peuvent être prédites à partir de variables telles que l'âge, le sexe, le poids et les facteurs génétiques.

E-commerce - Les régressions logistiques aident à maximiser le retour sur investissement (ROI) dans les campagnes de marketing, ce qui est avantageux pour les résultats de l'entreprise à long terme.

Commercial - Les chances qu'une demande de renseignements d'un client se transforme en vente.

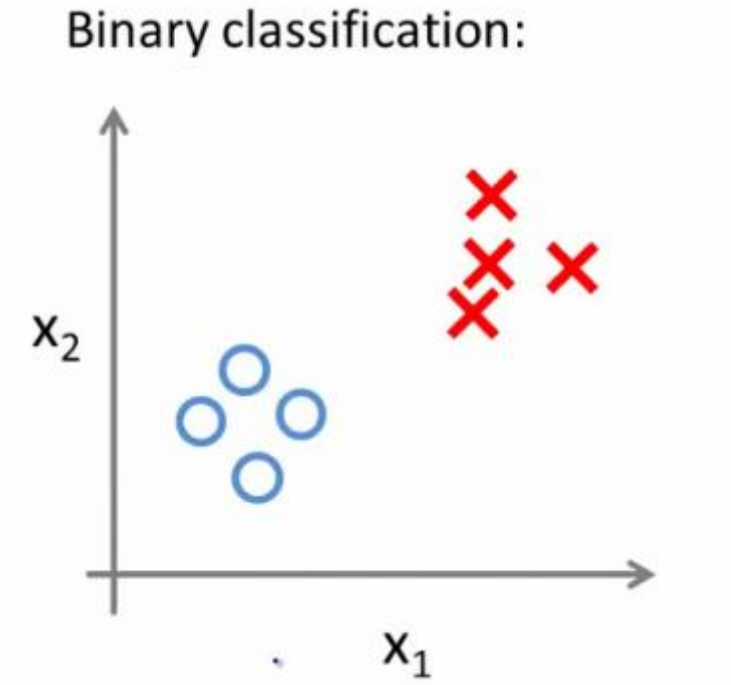
Finance - Une société de cartes de crédit qui l'utilise pour prédire la probabilité qu'un client ne respecte pas ses paiements.

Test de produit - Le succès ou l'échec d'un système en cours de test ou d'un prototype de produit.

Politique - Pour tenter de prédire les élections.

2. Un peu de math

Pour la classification binaire de la *Logistic Regression*, on cherche à classer un élément soit dans l'étiquette {1/Vrai/Chien/…} ou dans l'étiquette {0/Faux/Chat/…}.



On cherche donc, en machine learning, à trouver une frontière séparatrice entre ces deux étiquettes.

En mathématiques, la *Logistic regression* se base sur la régression linéaire où le résultat de la fonction linéaire à pour valeur 0 ou 1. Le modèle étant linéaire, on peut écrire la fonction hypothèse $S(X)$ pour un vecteur *features* X ayant n éléments comme ceci :

$$S(X\ ) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

- x_i : représente une variable qui sert dans le calcul du modèle prédictif c'est-à-dire pour le machine learning une valeur d'une colonne de *features.*
- θ_i : représente l'importance de la variable x_i dans le calcul du modèle prédictif. C'est cette valeur que l'on calcul lors de l'entrainement du modèle.
- θ_0 : constante particulière appelée le biais (*bias*).

Cette équation peut s'écrire aussi sous la forme suivante :

$$S(X\ ) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

Avec $x_0 = 1$, cela permet de factoriser cette équation sous forme de somme :

$$S(X) = \sum_{i=0}^{n} \theta_i \, x_i$$

Sous cette forme, on reconnait le produit de deux vecteurs ici le vecteur X contenant les éléments x_i et le vecteur Θ contenant les éléments θ_i d'où l'équation suivante :

$$S(X\ ) = \Theta X$$

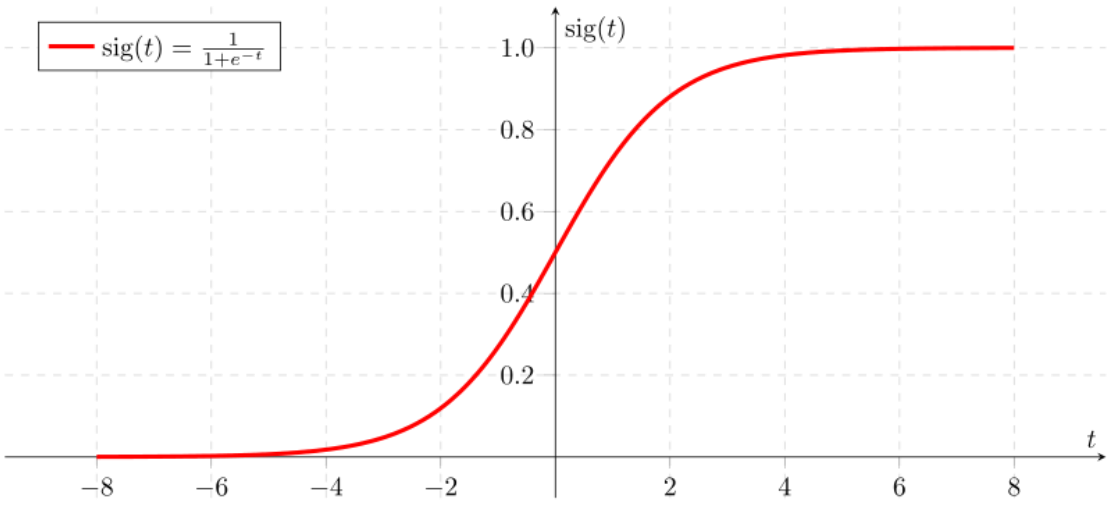
On peut appeler notre fonction hypothèse la **fonction score** où le but est de trouver la valeur du vecteur Θ en sachant que l'on cherche au final un résultat de 1 ou 0. Ainsi, on peut supposer que :

- Si $S(X) > 0$ alors la valeur prédite est 1
- Si $S(X) < 0$ alors la valeur prédite est 0

La **fonction Sigmoïde** (***Sigmoid function***) est une fonction qui produit des valeurs entre 0 et 1. Elle a pour formule mathématique :

$$Sigmoid(t) = \frac{1}{1 - e^{-t}}$$

Et pour représentation graphique :



Pour cette fonction :

- $Sigmoid(0) = 0{,}5$ et est le point de symétrie de cette courbe

- Elle asymptote en 0 et 1, c'est-à-dire la fonction sigmoïde se rapproche de 1 et 0 sans les atteindre
- Pour $x > 0$ on a $Sigmoid(x) > 0{,}5$ et pour $x < 0$ on $Sigmoid(x) < 0{,}5$

Ainsi, si on applique cette fonction sigmoïde à notre fonction score :

$$H(X) = Sigmoid\big(S(X)\big) = \frac{1}{1 - e^{-\Theta X}}$$

On obtient donc les propriétés de la fonction sigmoïde et on peut affirmer que :

- Si $S(X) > 0$ alors $H(X) > 0{,}5$ donc arrondie à l'unité $H(X) = 1$
- Si $S(X) < 0$ alors $H(X) < 0{,}5$ donc arrondie à l'unité $H(X) = 0$

On peut interpréter aussi le résultat de la fonction sigmoïde de notre fonction score sous forme de probabilité.

Donc, si on applique la fonction $H(X)$ pour déterminer le statut d'une tumeur. Si H(X) renvoie une valeur de 0,7 on peut l'interpréter ainsi, un patient à 70% d'avoir une tumeur maligne et donc 30% que cette tumeur soit bénigne.

Plus formellement, la probabilité que X soit de classe (étiquette) 1 avec les paramètres Θ :

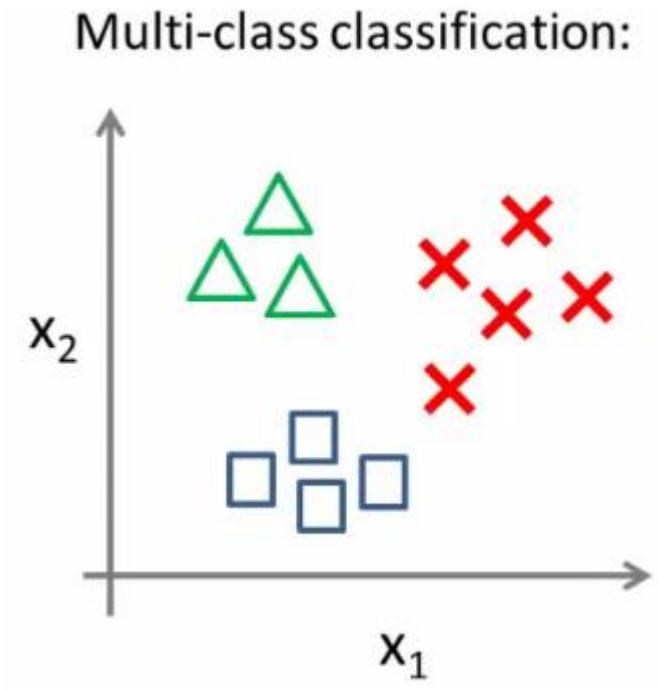
$$H(X) = P(y = 1 \parallel X; \Theta)$$

D'où la probabilité pour la classe (étiquette) 0 :

$$P(y = 0 \parallel X; \Theta) = 1 - P(y = 1 \parallel X; \Theta)$$

3. Multinomial *logistic regression*

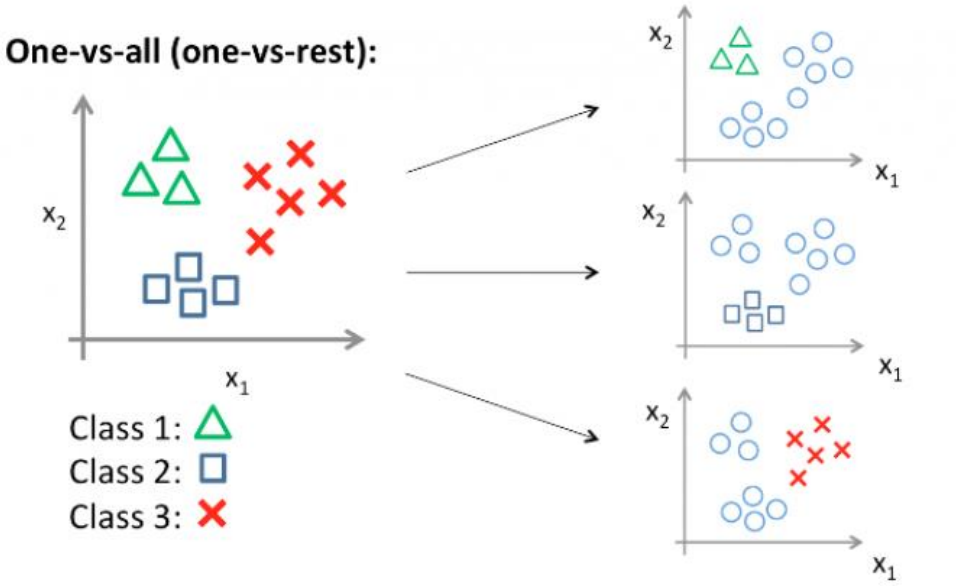
En machine learning, la *Logistic Regression* possède un 'hyperparamètre" **multi_class** qui permet de sélectionner la méthode de résolution pour les classifications multi-classes.



Il existe deux méthodes de résolution pour la classification multi-classes :

- OVR (One vs Rest)
- Multinomial

La méthode OVR consiste à découper la classification multi-classes en sous classification binaire. Pour illustrer voici un schéma :



- On considère que les triangles sont la classe positive (étiquette 1) et le reste comme la classe négative (étiquette 0), puis on entraîne la régression logistique sur cette configuration de données. Ce qui produira une fonction de prédiction $H^1(x)$
- Puis on considère les carrés comme la classe positive et le reste comme la classe négative, puis on entraîne la régression logistique pour obtenir une deuxième fonction de prédiction $H^2(x)$
- Pour finir, on considère les croix comme la classe positive et le reste comme la classe négative, puis on entraîne la régression logistique pour obtenir $H^3(x)$

Ainsi, la bonne classe (triangle, carrée, croix) pour l'élément x est celle où la probabilité entre $H^1(x)$, $H^2(x)$ et $H^3(x)$ est la plus grande.

En généralisant, la classe d'un élément x , parmi K classes, est :

$$\max_{i \in \{1,\dots,K\}} \quad H^i(x)$$

Pour la méthode multinomiale, la probabilité d'un élément x soit d'une classe, parmi K , ce calcule à partir de la fonction mathématique softmax :

$$soft\max(x) \;=\; \frac{e^x}{\sum_{i=1}^{K} e^x}$$

Sans plus rentrer dans les détails mathématiques, la méthode multinomiale permet de minimiser la perte dû à l'imprécision du raisonnement.

II. En Machine learning ça donne quoi ?

1. Présentation des variables du modèle

a. Les Variables de la *Logistic Regression*

La *Logistic Regression* propose de tester un modèle de régression dont la variable dépendante est dichotomique/binaire, par exemple : "Mourir ou Vivre", "Malin ou Bénin" etc.

Et variables indépendantes peuvent être quantitatives (continues ou discrète) ou qualitative (catégorielle).

Une variable indépendante est un paramètre qui varie sans être influencé par les autres paramètres mais à contrario, influe sur la variable dite dépendante, à titre d'exemple, lorsqu'on essaie de prédire si un étudiant va réussir ou échouer à un test, les heures étudiées ou les endroits d'étude sont les variables indépendantes et la variable dépendante serait la réussite ou l'échec à l'examen.

Ce lien entre une variable dépendante et une variable indépendante est appelé "relation".

Sur un graphique cartésien, la variable indépendante est associée à l'axe des abscisses (x) et la dépendante, celles des ordonnées (y)

b. Les principaux hyperparamètres

La *Logistic Regression* n'a pas vraiment d'hyperparamètres vraiment critiques à configurer mais parfois on peut constater de nette différence en fonction des solveurs.

Elle en possède cependant trois principaux : "C", "Solver" et "Penalty"

La *Logistic Regression* offre d'autres paramètres tels que : class_weight, dualbool (pour les ensembles de données clairsemés (lorsque n_samples > n_features), max_iter (peut améliorer la convergence avec des itérations plus élevées) et autres, cependant, ceux-ci ont moins d'impact.

C : Ou force de régularisation doit être un flottant positif (compris généralement entre 0.001 et 1). La force de régularisation fonctionne avec Penalty pour réguler le surajustement.

Des valeurs plus petites spécifient une régularisation plus forte et une valeur élevée indique au modèle d'accorder un poids élevé aux données d'apprentissage.

La régularisation fait généralement référence au concept, selon lequel il devrait y avoir une pénalité de complexité pour des paramètres plus extrêmes

Grosso modo un C élevé signifie "Faites énormément confiance à ces données d'entraînement", tandis qu'une valeur faible indique généralement "Ces données peuvent ne pas être entièrement représentatives des données du monde réel, donc si cela vous demande de faire des paramètres plus grands, ne l'écoutez pas"

Solver : L'hyperparamètre solver possèdes plusieurs paramètres en son sein, qui ont chacun des spécificités différents et situationnels :

-lbfgs : fonctionne relativement bien par rapport à d'autres méthodes et économise beaucoup de mémoire, cependant, il peut parfois y avoir des problèmes de convergence.

-saq : plus rapide que les autres solveurs pour les grands ensembles de données, lorsque le nombre d'échantillons et le nombre d'entités sont importants.

-saqa : le solveur de choix pour la *Logistic Regression* multinomiale clairsemée et il convient également aux très grands ensembles de données.

-newton-cq : coûteux en calcul à cause de la Hessian Matrix.

-liblinear : recommandé pour un ensemble de données de grandes dimensions - résout des problèmes de classification à grande échelle.

Penalty : Pénaliser un algorithme de Machine Learning signifie essentiellement que vous ne voulez pas que votre algorithme soit sur-adapté à vos données, donc sur l'apprentissage.

L’hyperparamètre Penalty possède deux paramètres : "L1", "L2" ou les deux (elasticnet), on peut aussi ne pas lui en assigner avec "None" par conséquent l'algorithme ne sera pas pénalisé

(Attention, certaine Penalty ne fonctionne pas avec certains solvers, documentation recommandée)

Avant de rentrer dans les détails de L1 et L2, abordons brièvement le concept de Weight ou poids en français.

Le weight est une terminologie utilisée pour les tâches de classification où chaque catégorie de l'ensemble de données reçois un certain poids en fonction de la fréquence d'occurrence de chaque catégorie, l’utilisation d’un weight déséquilibré peut être responsable de biais ou d’underfitting en général, d’où l’intérêt de le réguler quand cela est nécessaire.

Une distribution uniforme des poids peuvent servir à la précision de divers paramètres tels que : “Precision”, “Recall “ et “F1 score”, car les poids seraient équilibrés.

Par exemple si un étudiant est très bon en multiplication mais qu’il n’est pas bon sur ses divisions, l’idée serait de lui interdire de continuer à réviser les multiplications pendant ses révisions afin qu’il soit obligé d’améliorer ses connaissances en divisions pour atteindre son but qui serait ici, réussir son examen

Cette technique permettrait alors de généraliser son apprentissage, de telle sorte qu’il pourrait réussir son test aussi bien sûr les multiplications que les divisions.

En ML cela permet au modèle de mieux généraliser son apprentissage et donc améliorer ses performances sur les données non observées.

Maintenant rentrons dans les détails de L1 et L2 :

L1 - La régularisation L1, également appelée norme L1 ou Lasso (dans les problèmes de régression), combat l'overfitting en rendant les weight à 0. Cela rend donc certaines features obsolètes.

C'est une forme de sélection de features, car lorsque nous attribuons une feature avec un weight de 0, nous multiplions les valeurs de caractéristiques par 0, ce qui renvoie 0, éliminant ainsi l'importance de cette feature.

Si les features d'entrée de notre modèle avaient des poids plus proches de 0, notre norme L1 serait clairsemée.

Par exemple, imaginons que nous voulions prédire les prix des logements à l'aide de cette ML.

Nous avons en valeur indépendante : La Rue, Le Quartier, l’Accessibilité (accès aux transports), l’Année de construction, Les Chambres et la Cheminée et en valeur dépendante nous aurions Le prix du bien.

En analysant d'intuition ces caractéristiques, on peut supposer que l'emplacement de la maison (quartier) a énormément plus d'impact que le nombre de cheminée.

L1 va donc rendre la feature cheminée à 0 car elle n'a pas une influence significative sur le prix.

Essentiellement, lorsque nous utilisons L1, nous pénalisons les valeurs absolues du weight des paramètres.

L2 - La régularisation L2, ou la norme L2, ou Ridge (dans les problèmes de régression), combat l'overfitting en forçant le weight des paramètres à être minime, mais sans les rendre exactement 0.

Donc, si nous prédisons à nouveau les prix des maisons, cela signifie que les features les moins importantes pour prédire le prix ces maisons, auraient encore une certaine influence sur la prédiction finale, mais seulement à petite échelle.

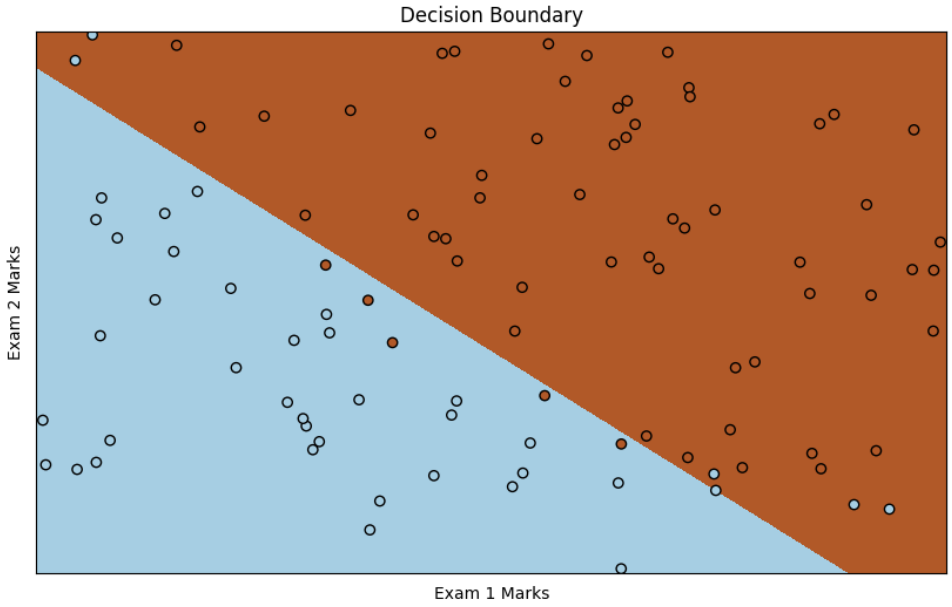
Ainsi, la régularisation L2 renvoie une solution non parcimonieuse puisque les weights seront non nuls (bien que certains puissent être très proche de 0).

2. Exemple sous forme de graphe

a. Classification binaire

Voici un exemple de classification binaire avec une *Logistic Regression*.
On cherche à classer en admis ou non-admis des élèves suivant leurs résultats sur deux examens.

Crédit pour le code : **Satish Gunjal**

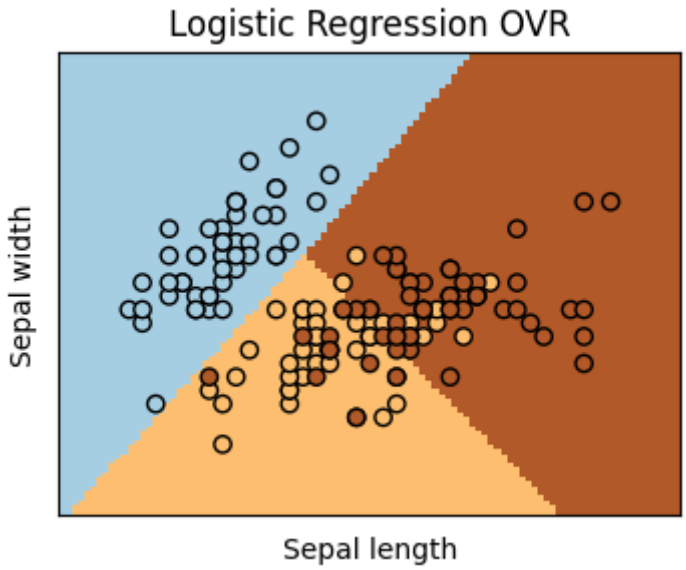


b. Classification multi-classe

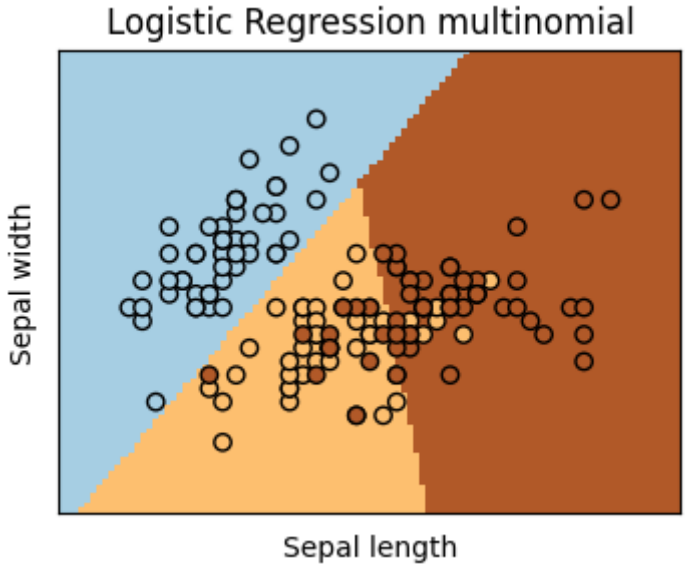
Voici un exemple de classification multi-classe avec la *Logistic Regression* appliquer à trois étiquettes.
Cet exemple concerne la base de données iris présente dans la librairie scikit-learn et permet de classifier entre 3trois types d'Iris (setosa, versicolor, virginica) en utilisant que deux des quatre *features* de la base de données.

Crédit pour le code à : **Gaël Varoquaux** et **Jaques Grobler.**

Pour la méthode OVR, on obtient le graphique suivant :



Pour la méthode multinomiale, on obtient le graphique suivant :



L'*accuracy* pour le modèle multinomial est : 0.783
L'*accuracy* pour le modèle OVR est : 0.75

Leur *accuracy* étant légèrement différente il est intéressant de trouver le modèle de résolution le plus adapté aux données pour la mise en place du machine learning avec ***GridSearch***.

3. Avantages/inconvénients

a. Avantages

La *Logistic Regression* est largement utilisée, car elle est extrêmement efficace et ne requiert pas d'énormes quantités de ressources informatiques.

Elle peut être interprétée facilement et ne nécessite pas de mise à l'échelle des caractéristiques d'entrée. Elle est simple à régulariser et les résultats qu'elle fournit sont des probabilités prédites bien calibrées.

Tout comme dans la régression linéaire, la *Logistic Regression* a tendance à fonctionner plus efficacement lorsque les attributs non liés à la variable de sortie et ceux qui sont corrélés, sont omis.

Le Feature Engineering joue un rôle important dans l'efficacité des performances de la *Logistic Regression* et linéaire.

La *Logistic Regression* est également simple pour former les utilisateurs et facile à mettre en œuvre, ce qui en fait une excellente référence pour aider à mesurer les performances d'autres algorithmes complexes.

b. Inconvénients

La *Logistic Regression* ne peut pas être utilisée pour résoudre des problèmes non linéaires et, malheureusement, de nombreux systèmes actuels sont non linéaires.

De plus, la *Logistic Regression* n'est pas l'algorithme le plus puissant disponible. Il existe plusieurs alternatives qui peuvent créer des prédictions bien meilleures et plus complexes.

La *Logistic Regression* repose également fortement sur la présentation des données.

Cela signifie que si vous n'avez pas identifié toutes les variables indépendantes nécessaires, le résultat n'a aucune valeur. Avec un résultat qui est discret, la *Logistic Regression* ne peut être utilisée que pour prédire un résultat catégorique.

Enfin, il s'agit d'un algorithme dont la vulnérabilité à l'ajustement excessif est connue.