Estimation du churn dans un entreprise de Télécom

Yohan Foucade & Jean Zagdoun

March 7, 2018

Ce document constitue le rapport écris du projet du cours sciences des données de l'entreprise du master M2M0.

Introduction

D'après le site : `https://www.definitions-marketing.com/definition/taux-d-attrition/`
Le taux d'attrition, ou churn rate en anglais, est un indicateur qui permet de mesurer le phénomène de perte de clientèle ou d'abonnés. Le taux d'attrition est le ratio (nombre de clients perdus / nombre de clients total) mesuré sur une période donnée qui est le plus souvent l'année. Au vu de l'impact de la fidélité des clients sur la rentabilité de l'entreprise, le taux d'attrition est un indicateur dont le suivi est particulièrement important. Il s'agit notamment d'un indicateur marketing clé dans les domaines d'activité qui ont recours à l'abonnement (presse, TV satellite, téléphonie, etc..).
Ainsi c'est un indicateur précieux et il est normal qu'une entreprise veuille le prédire avant de lancer de nouvelles offres sur le marché ou à l'apparition d'offres de ces concurrents.
Le jeu de données que nous avons trouvé ,et qui est disponible ici : `https://www.kaggle.com/becksddf/churn-in-telecoms-dataset`, sera présenté dans la section une puis nous décrirons plus en détail à l'aide d'outils statistiques ses caractéristiques dans la section deux, ensuite nous entrerons dans le vif du sujet et essayerons de prédire le taux d'attrition à l'aide notamment de méthodes vues en cours.

1 Le jeu de données

Les entrées du jeu de données que nous nous sommes procure sont des lignes téléphoniques ouvertes auprès d'un opérateur de télécommunications. Pour chacune des 3 333 observations, nous disposons de 20 features et une variable cible de type binaire : churn ou non. Comme beaucoup de jeu de données relatifs a l'attrition, le notre est déséquilibré (14% de positifs). Voici la liste des features:

1. state : Code de l'état (ex : FL pour la Floride)
2. account length : il s'agit du nombre de transactions totales du client
3. area code : Code en fonction de la localisation (ex : +33 pour la France)
4. phone number : Numéro de téléphone
5. international plan : est-ce que le client a un forfait 'international'
6. voice mail plan : est-ce que le client possède ou non une messagerie vocale
7. number vmail messages : nombre de messages vocaux
8. total day minutes : temps d'appels en journée
9. total day calls : nombre d'appels en journée
10. total day charge : facturation des appels émis en journée
11. total eve minutes : soir
12. total eve calls
13. total eve charge
14. total night minutes : nuit
15. total night calls
16. total night charge
17. total intl minutes : a l'étranger
18. total intl calls
19. total intl charge
20. customer service calls : nombre d'appels au service client.

Lors de la phase de preprocessing, nous avons décidé de nous débarrasser des variables *area code* et *phone number*, qui correspondent au numéro de téléphone du client et ne nous apportent aucune information sur son comportement. Nous avons par la suite constate que la variable *state*, était elle aussi peu utile pour notre modèle.
Enfin, on peut aussi remarquer une relation linéaire entre les durées d'appels et les charges (Figure 1). On supprime donc les features correspondant aux charges (pour day, eve, night et intl).

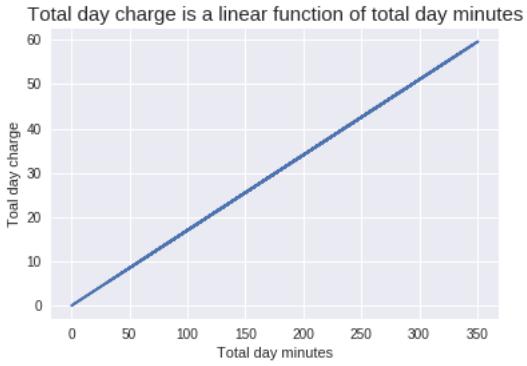


Figure 1: total day minutes vs total day charge shows a linear relation

Nous obtenons finalement un tableau avec 13 features et un label. Pour tous les algorithmes, nous avons séparé nos données en un jeu d'entraînement (70%) et un jeu de test (30%).

Le premier algorithme que nous ayons implémenté est la régression logistique.

Pour ce qui est des statistiques descriptives, nous évoquerons ici la corrélation entre le nombre d'appels au service client et le taux d'attrition (Figure 2).

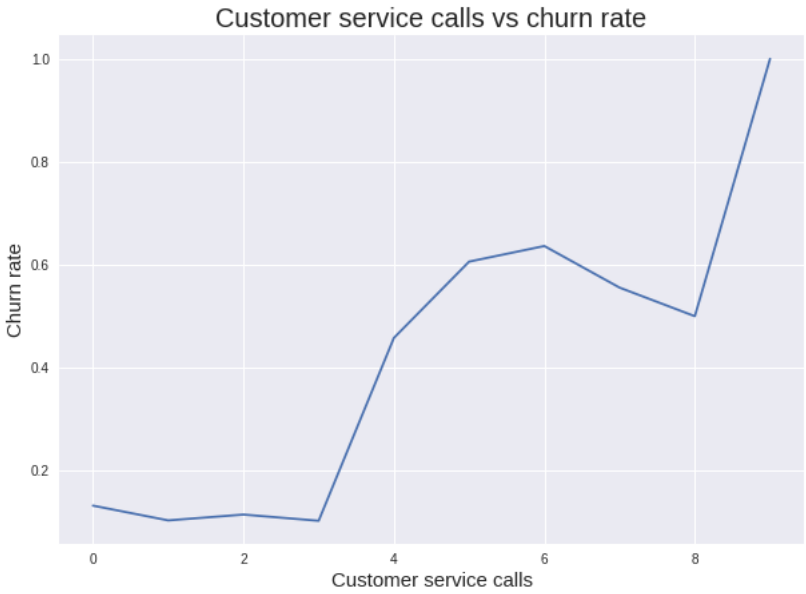


Figure 2: Au dela de 3 appels au service clients, le taux d'attrition augmente considérablement

2 Régression logistique et maximisation du profit

Lors de notre implémentation de l'algorithme, nous avons choisi le méta-paramètre C, qui pénalise la complexité du modèle, par validation croisée. Nous ne reviendrons pas ici sur le fonctionnement de l'algorithme mais donnons immédiatement nos résultats:

2.1 Résultats

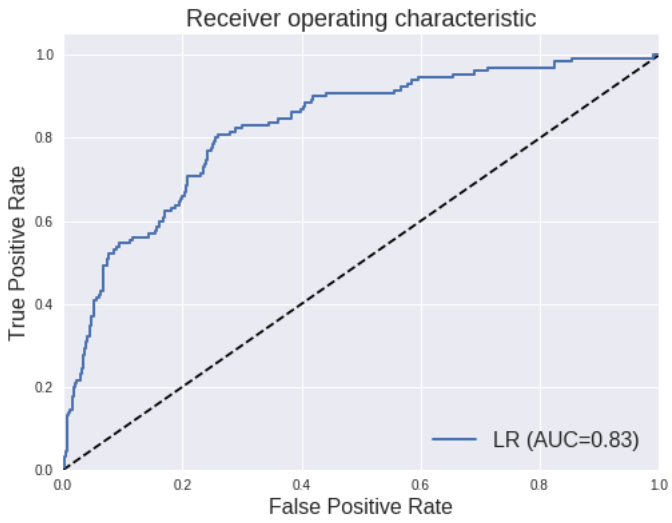


Figure 3: Courbe ROC

2.2 Maximisation du profit

La plupart des algorithme de machine learning se concentrent sur la précision de la prédiction. Cependant, depuis quelques années, des méthodes alternatives, basées sur la maximisation de profit, ont été proposées. En effet, le profit d'une entreprise désirant cibler les clients qu'elle est susceptible de perdre dépend non seulement de la précision de son modèle, mais aussi de la répartition entre clients "fidèles" ou non, de la probabilité que le client soit sensible a une éventuelle action commerciale visant à le retenir, et du coût d'une telle action pour l'entreprise. Un modèle prenant en compte tous ces paramètres est proposé par Lemmens et Gupta dans *Managing Churn to Maximize Profits*, 2013. Nous proposons ici le modèle simplifié suivant:

- l'entreprise fait une prédiction en appliquant son modèle
- pour les individus classés en "positif", l'entreprise fera une "action de rétention" sous forme d'offre commerciale.

On suppose qu'un client qui avait l'intention de partir et qui se voit proposer une offre, change d'avis avec probabilité p. La figure 4 montre, après évaluation par le modèle et proposition commerciale de l'entreprise, les 4 cas de figures possibles.

La variation du profit de l'entreprise, sur 1 mois peut alors s'écrire :

$$\Delta\Pi(t) = p * C_1 * F_1(t) - \tilde{\text{p}} * (C_0 - C_1) * F_0(t) \tag{1}$$

Ou p est la probabilité que l'action commerciale fonctionne sur un client partant. C_1 est le prix propose a la place du prix initiale C_0. $F_1(t)$ est la probabilité qu'un client souhaitant partir soit correctement classifie par le modèle. $\tilde{\text{p}}$ est la probabilité qu'un client qui ne souhaitait pas partir, accepte tout de même C_1. $F_0(t)$ est la probabilité qu'un tel client soit mal classifié par le modèle. F_0 et F_1 sont en fait les taux de faux et de vrais positifs.

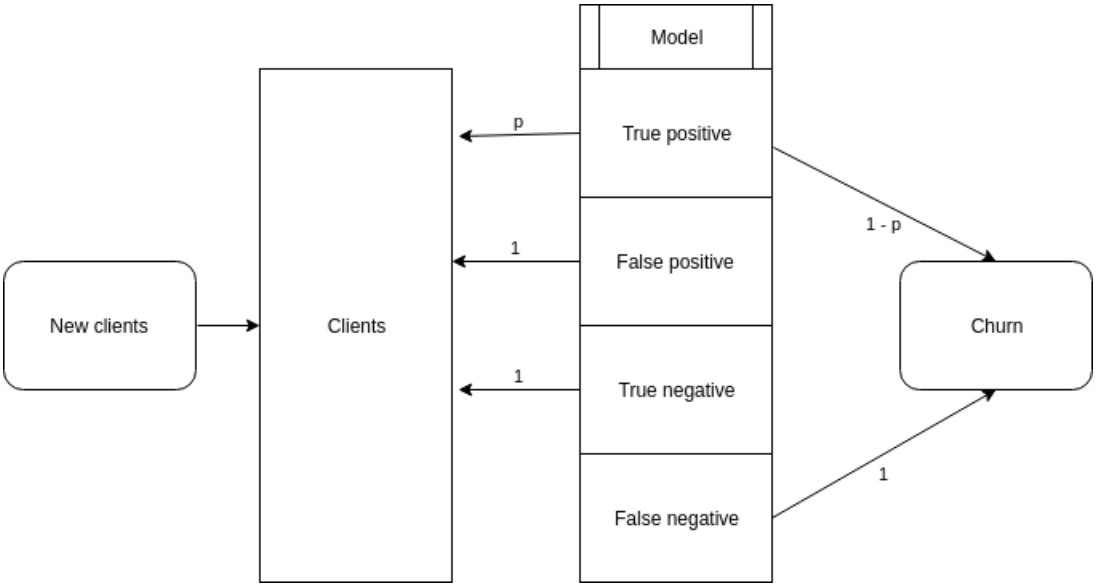


Figure 4: schéma des possibilités pour le client

Notre problème s'écrit alors :

$$\underset{F_1(t),F_0(t)}{\text{argmax}}\ p * C_1 * F_1(t) - \tilde{\text{p}} * (C_0 - C_1) * F_0(t) \tag{2}$$

Mais toutes les paires (F_1, F_0) ne sont pas réalisables. Si tel était le cas, il faudrait choisir $F_1 = 1$ et $F_0 = 0$ pour maximiser le profit. Dans ce cas, il serait impossible de faire mieux. Ici, comme le montre la courbe ROC, on peut écrire F_1 en fonction de F_0 :

$$F_1(t) = g(F_0(t)) \tag{3}$$

En injectant dans (2), on obtient :

$$\underset{F_0(t)}{\text{argmax}}\ p * C_1 * g(F_0(t)) - \tilde{\text{p}} * (C_0 - C_1) * F_0(t) \tag{4}$$

Il y a au mois 3 façons de résoudre (4) :

- "Brute force" : tester chaque paire (F_1, F_0) utilisées pour tracer la courbe ROC et choisir celle qui maximise le profit. Cette méthode est en $O(n_{test})$ où n_{test} est le nombre d'observations dans le jeu de test.

- "Approximation de g" : (par exemple modèle paramétrique) et résolution analytique du problème. L'avantage de cette méthode, est qu'une fois le problème résolut, l'obtention de la valeur de $F_0^*(t)$ pour des paramètres (p, p, C_0, C_1) donnés se fait en temps constant. Puis trouver le seuil $t*$ qui approche le mieux $F_0^*(t)$ se fait en $O(log_2(n))$ si nos données sont dans un arbre binaire de recherche.

- on approxime aussi $F_o(t)$, la résolution pour des paramètres donnés est alors en temps constant.

Il y a donc un arbitrage entre le temps de calcul et la rapidité du modèle. Cependant, dans certaine situation, comme des enchères en temps réel, il peut être utile de donner une réponse très rapidement.

Nous avons choisit de tester les deux premières méthodes,avec la grille de paramètres suivante : $p = 0.5$, $\tilde{p} = 0.5$, $C_0 = 60$, $C_1 = 35$, afin de les comparer. Le résultat est montré dans la figure 5 :

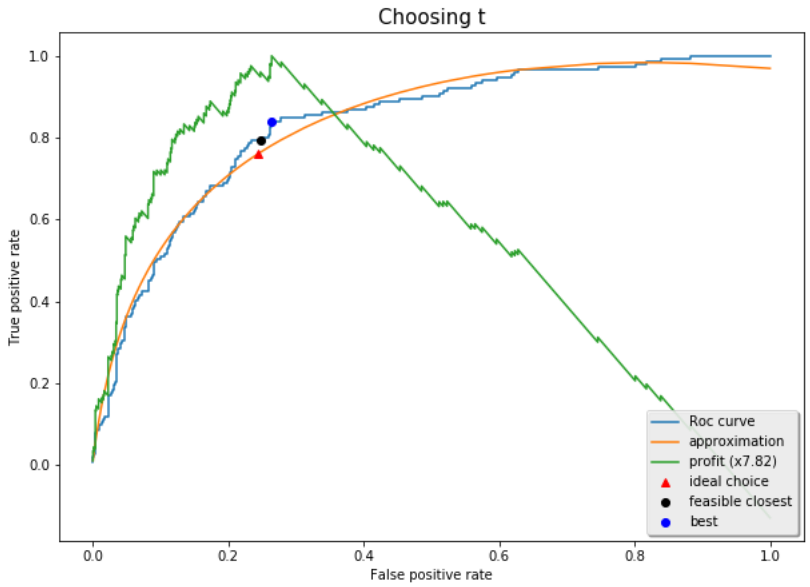
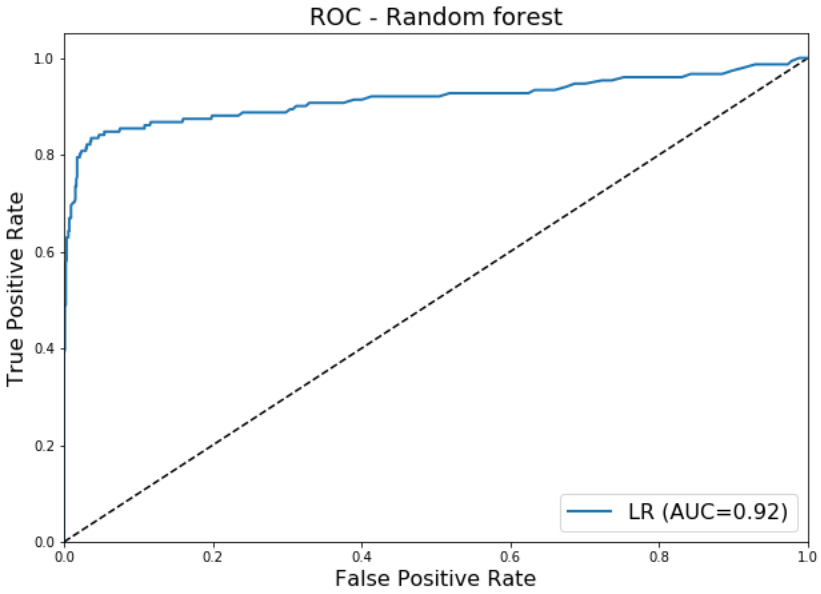


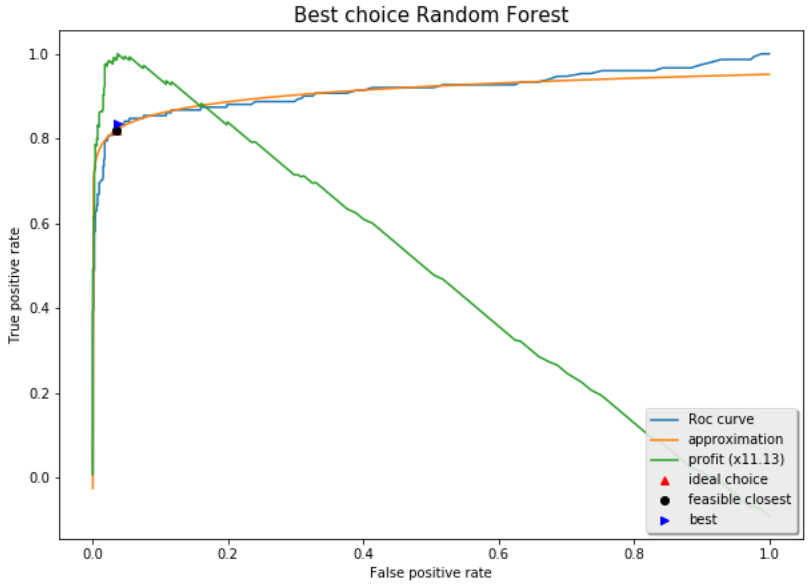
Figure 5: résumé des différentes méthode pour la régression logistique

3 Forêts aléatoire

Nous implémentons maintenant une méthode bien connue de classification: les forêts aléatoire.Cette méthodes à de très nombreuse qualités notamment sur l'interprétation possible des données comme vue en cours. Ici notre cette méthode ce montre particulièrement efficace comme le montre sa courbe ROC:

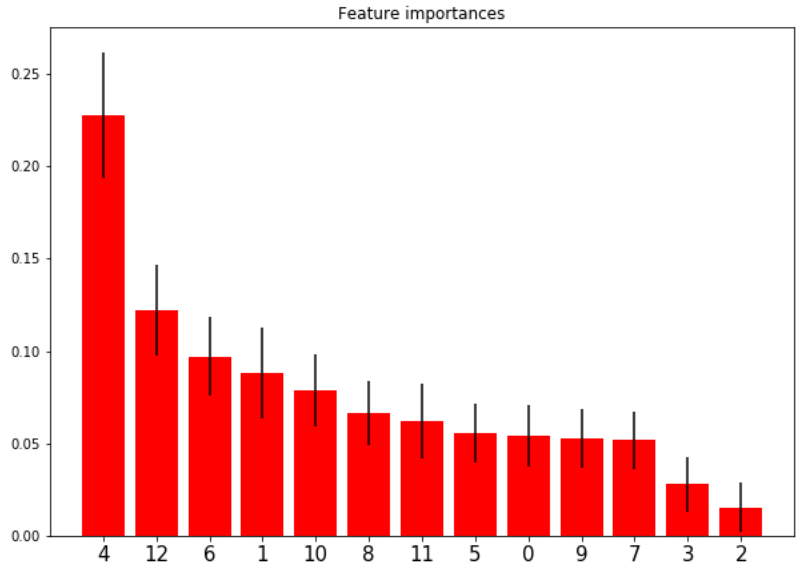


Une AUC de 0.92 est un résultat excellent,et avec la méthode développée précédemment cela nous permet d'aboutir à un profit maximum de $11,13$ ce qui est beaucoup plus qu'avec la régression logistique et de plus en approximant cette courbe par une fonction différentiable: $f(a,b,c,x) = a + xb + x^{\frac{1}{23}}c$,en l'optimisant en a, b, c et en cherchant à maximiser le profit on trouve :



Soit un résultat très proche du vrai maximum.Mais en plus d'être très précise cette méth-

ode à le bon gout de nous donnée une idée des caractéristiques les plus importantes de nos individues.Graphiquement:



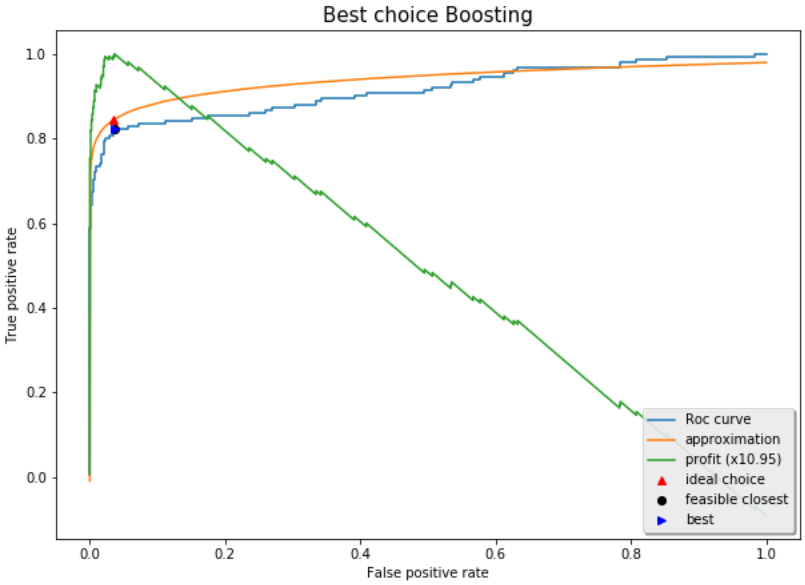
Où les ceux premières sont :

- 1. feature **total day minutes** (0.227784)
- 2. feature **customer service calls** (0.122050)

Ceci confirme donc notre intuition de la section de description du jeu de donnée.

4 Boosting

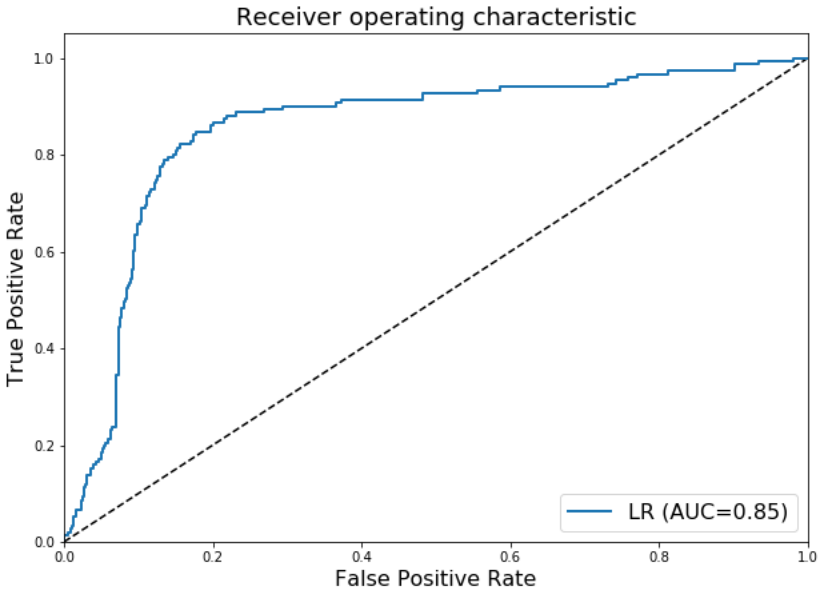
En faisant un recherche quadrillée classique sur l'échantillon d'entraînement du jeu de donnée pour l'algorihtme de boosting XGBClassifier() de la librairie python: xgboost nous trouvons des parmètres optimaux suivants : gamma: 0.5, max_delta_step: 3, max_depth: 3, min_child_weight: 1, n_estimators: 500, scale_pos_weight: 1.5, subsample: 1
Ce qui nous amène à une courbe ROC:
On remarque que cette courbe ROC à a peu près le même profil que celle des forêts aléatoire de la section précédente.Il est alors logique de l'estimer par la même fonction :
$f(a,b,c,x) = a + xb + x^{\frac{1}{23}}c$
après une optimisationet une recherche de maximum on trouve le graphique correspondant à une valeur de profit de $10,95$:



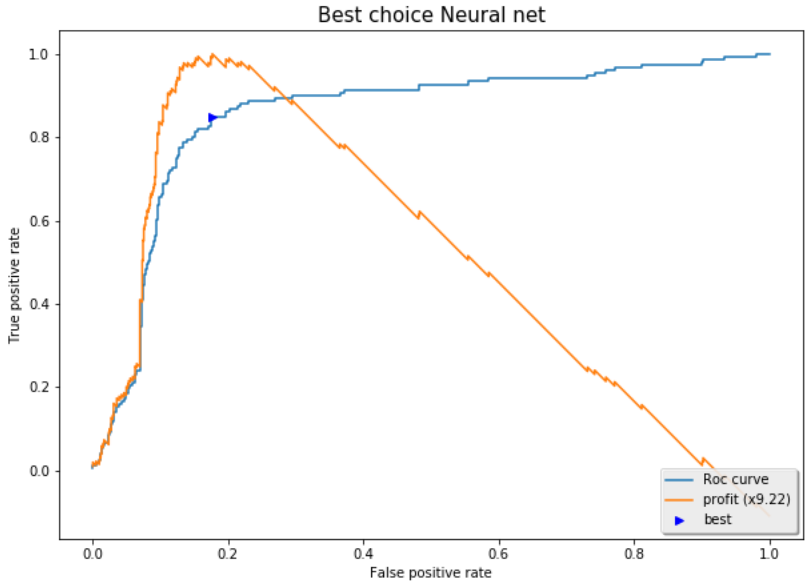
5 Réseaux de neurones

Dans cette section nous allons essayer de construire un modèle neuronale pour estimer le taux de churn.Nous étudions tout d'abord le churn sur des résultats classiques de de précisions et avec un modèle MLP (multi-layer perceptron) à 3 layers, respectivement : 10,5 et 2 neurones par layer et avec une fonction d'activation de type sigmoïd et pour finir un optimiseur de type adagrad.
Un des principal défaut des méthodes neuronales avec ce jeu de donnée est le faible nombre de donnée au totale et le fait qu'elles soient déséquilibrées.
Pour palier au dernier point nous mettons une fonction de poids sur la fonction de perte du MLP avec un poids de 1 si la classe est 0 et un poids de 6 si la classe est 1.
Pour parer à la faible quantité d'observation nous avons pris un nombre "d'epochs"de 1000 pour une taille de "batch" de 20 nous observons l'évolution d'une classification correcte suivante :

On voit de manière évidente un apprentissage régulier et une forte augmentation à partir de l'épcoh numéro 900 environ ceci nous encouragerais à rajouter plus d'époch encore si nos machines nous le permettait.
En ce qui concerne la courbe ROC on trouve:



Avec les paramètres de la section précédente et en essayant de trouver le meilleur choix possible qui maximiserai notre profit :



On voit tout de suite qu'un profit de 9,22 est nettement moins élever que celui trouver par les méthodes de forêts aléatoire et de gradient boosting.

Conclusion

Dans la résolution de se problème l'algorithme de forêt aléatoire nous à donner les meilleures performance et nous à permis de relever des variables importantes qui peuvent, en plus du modèle de maximisation de profit être utiliser sur les clients à qui l'entreprise.(dans leurs nouvelles offres par exemple ou pour extraire des groupes de client).

References

[1] Kevin Murphy. *Machine learning, a probabilistic perspetive*. MIT Press, 2012.

[2] Aurélie Lemmens and Sunil Gupta. *Managing Churn to Maximize Profits*. Harvard Business School, 2013.

[3] Glady, N., Baesens, B., Croux, C. *Modeling churn using customer lifetime value*. European Journal of Operational Research 197 (1), 402–411.