

# Stage Orange

Garance Malnoë

Mai 2023

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Horaire et programme des journées</b>	<b>2</b>
<b>3</b>	<b>Prévision échecs de production</b>	<b>3</b>
3.1	Informations du projet . . . . .	3
3.2	Analyse des commentaires . . . . .	4
3.2.1	Première préparation des données . . . . .	4
3.2.2	Statistiques et nouvelles données . . . . .	6
3.2.3	Normalisation des données . . . . .	7
3.2.4	Polarité et subjectivité des commentaires . . . . .	8
3.3	Choix des estimateurs . . . . .	8
3.4	Modélisation . . . . .	9
3.4.1	Modélisation avec un arbre de décision . . . . .	9
3.4.2	Associations des mots à une réussite ou à un échec . . . . .	12
3.4.3	Autres méthodes similaires aux arbres de décision . . . . .	13
3.4.4	Modélisation K-nearest neighbors . . . . .	13
3.4.5	Modélisation régression logistique . . . . .	13
3.4.6	SVM . . . . .	13
3.4.7	Naive Bayes . . . . .	13

# 1 Introduction

Ce document a pour but de suivre le déroulé de mon stage chez Orange de Mai 2023 à Aout 2023. Je vais essayer de documenter mes activités quotidiennes, d'y mettre les notes des livres que je lis ainsi que mon travail sur le sujet de mon stage.

Quel est d'ailleurs ce sujet de stage ?

La création d'un modèle à l'aide de Machine Learning permettant de déterminer les dossiers (en cours) qui ont le plus grand risque de terminer en échec notamment avec de l'analyse de texte et l'analyse de la saturation du réseau.

Code : Programmation en ligne sur Jupyter avec ses notebooks en ligne et le code est sauvegardé sur ma page GitHub.

# 2 Horaire et programme des journées

Semaine 1 : Découverte des locaux pendant les premiers jours, présentation du sujet et travail sur le livre Machine Learning. Travail sur le livre Implémentation en Python avec Scikit-Learn pendant tout le reste de la semaine. Mon travail sur ce livre consistait à lire le livre, faire des recherches supplémentaires pour chaque questionnement qui me venait, chercher les mathématiques derrière etc et de synthétiser sur LaTeX.

Semaine 2 : J'ai continué à travailler sur le livre en prenant des notes sur LaTeX pour avoir une trace écrite de mon apprentissage et pouvoir m'y référer par le futur lorsque je travaillerai sur le sujet de stage. J'ai fini le livre le Mardi soir. Le Mercredi j'ai installé Jupyter et des package, cherché des livres sur le NLP et découvert le sujet avec une vidéo exemple de 2h. Les jours suivants, j'ai retranscrit le contenu de la vidéo sur ce document. Enfin, le Vendredi je me suis formée sur la bibliothèque Pandas avec Orange Learning.

Semaine 3 :

J'ai continuer à me former à la bibliothèque Pandas. J'ai appris à me servir de gitHub pour pouvoir sauvegarder mes fichiers de stage. Avec Jérôme et d'autres personnes (Alan et Bernard) nous avons fait des mises aux points pour bien délimiter le sujet final de stage, nous avons notamment mis de côté l'analyse de texte qui semble périlleuse au vu des textes, nous avons garder le sujet principal qui est de prévenir les échecs de production mais en s'appuyant sur les données des lignes en faisant des croisement de tableaux Excel. Mercredi et Jeudi, je me suis entraînée à travailler sur des cas d'arbre de décision et de régression linéaire.

Semaine 4 : Stage à Amiens.

Semaine 5 : Changement de sujet pour le retour à l'analyse de texte après

conseil de Jérôme. Lundi : nettoyage du fichier ETI31 pour obtenir quelque chose de propre. Mardi : suite du nettoyage du fichier ETI31 et début de la modélisation sur les commentaires avec l'analyse de sentiment et remise en place du GitHub au propre. Mercredi : Suite de la modélisation des l'analyse de commentaires et écriture du rapport sur lateX. Jeudi : Comme Mercredi + réunion avec Sandrine.

### 3 Prévision échecs de production

Dans cette section je vais essayer de retracer mon travail sur la modélisation des échecs : la préparation des données, la modélisation, la mise en place... En prenant soin de garder une trace de ce qui n'a pas marché et des choix effectués.

#### 3.1 Informations du projet

Pour ce projet j'ai suivi la méthode CRISP-DM présentée dans le livre Machine Learning (voir fichier ressources). Je vais donc essayer d'appliquer les étapes de Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation et Déploiement .

Business Understanding : La demande est de créer un modèle pour pouvoir prévoir les échecs de productions sur le réseau fibre afin d'intervenir plus tôt.

Les données à notre disposition :

Il y avait 6 jeux de données au format CSV :

- ETI31 : contient notamment les code d'échec ou de réussite des datasets, les commentaires faits et d'autres informations sur les interventions.
- ETI2 : contient les mêmes informations que ETI31 mais avant que les interventions aient lieu.
- Pilpro : contient des informations sur les dossiers en cours avec des commentaires des interventions.
- IPON\_UIO\_routes\_agregees (routes\_UIO): contient les routes des fibres optiques de l'ouest.
- raw\_dim\_ref\_edr\_ipo\_saturation\_pb\_t (saturation): contient les informations sur le nombre de lignes connectées à tel ou tel point et permet notamment de savoir s'il y a saturation.
- raw\_dim\_ref\_ipo\_adresses\_pf\_oi\_t (adresses) : contient des informations sur les lignes comme l'adresse, code INSEE, rivières sur toute la France.
- raw\_dim\_ref\_route\_agregee\_oi\_t (routes\_FR). contient les routes des fibres optiques de France.

Après un premier coup d’oeil les adresses ne se prêtent pas au Machine Learning, nous allons garder le fichier des routes optiques de France plutôt que celles de l’ouest puisque les données d’erreur sont sur toute la France, et ETI2 et Pilpro ne comporte pas d’information sur les échecs. Nous allons donc travailler avec les tables suivantes :

- ETI31
- routes.FR
- saturation

Et une fois que le modèle sera terminé, il sera à appliquer sur les tables Pilpro et ETI2.

Les bibliothèques utilisées :

- numpy (np)
- pandas (pd)
- seaborn (sns)
- sklearn (sk)
- matplotlib.pyplot (plt)
- Textblob

Pour éviter de faire toutes les étapes sur un seul notebook, on "pickle" les dataframe au fur et à mesure pour pouvoir les récupérer. Ils sont enregistrés dans le dossier "PICKLE".

## 3.2 Analyse des commentaires

### 3.2.1 Première préparation des données

La table ETI31 étant très volumineuse (32695 lignes et 175 colonnes), un premier traitement a été effectué sur Excel directement plutôt qu’avec Jupyter et Pandas car la tableau est beaucoup trop lourd pour être chargé en un temps acceptable. J’ai gardé uniquement les lignes concernant les production sur de la fibre (IQ\*FTH) et j’ai gardé les colonnes suivantes:

- No DESIGNATION (ex:299568158)
- CODE RELEVÉ (ex : RRC, indique l’erreur ou la réussite)
- COMMENTAIRE DE SIG
- COMMENTAIRE
- BLOC NOTE

- COMMLITIGEPIDI

La suite du traitement se fait ensuite sur le notebook Jupyter avec la bibliothèque pandas et consiste à enlever les lignes qui ne sont pas exploitables : pas de code de relève ou aucun commentaire.

L'étape suivante est la labélisation des données dans une colonne "Réussite" : passer des codes de relève à l'indication d'un échec ou d'une réussite et on peut enlever la colonne CODE DE RELEVÉ qui ne nous servira plus.

Réussite : RRC, DMS, TVC,

Échec: ANC, ANN, ETU, MAJ, ORT, PAD, PBC, REO, RMC, RMF, RRC

En suite, on nettoie les textes :

- Enlever les codes de relève.
- Enlever la ponctuation.
- Remplacement des abréviations et des acronymes par le.s mot.s correspondant.s.
- Tout mettre en minuscule.
- Enlever les nombres.

Puis on termine la préparation avec la transformation des colonnes de commentaire en une seule colonne contenant une liste de commentaires. Sur une seule colonne (par exemple : COMMENTAIRE) il peut y avoir plusieurs commentaires qui sont séparés par un anti-slash. On utilise la fonction split de python pour séparer les textes en fonction de ce séparateur et on l'applique à toutes les colonnes contenant du texte. Il est important de noter que pour certaines interventions, il n'y a pas de commentaires ou seulement des slashes.

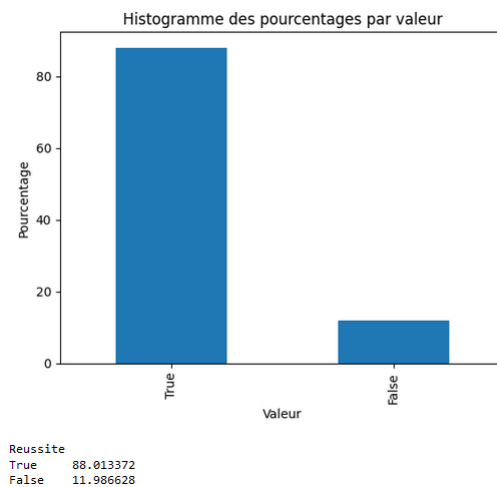
Après cette première phase de nettoyage on se retrouve donc avec un dataframe composé de 3 colonnes : le numéro de désignation, le label de réussite et la liste des commentaires.

	No DESIGNATION	Reussite	COMMENTAIRES
0	299568158	False	[ obtenu marchand christophe le a le clien...
1	297892193	True	[ point de terminaison optique existante malf...
2	297897767	True	[ point de terminaison optique non existante ...
3	223370510	True	[ point de terminaison optique non existante ...
4	299556603	True	[ point de terminaison optique existante malf...
...	...	...	...
4183	223275005	True	[merci de clrer le rendez vous du client car i...
4184	296611510	True	[ point de terminaison optique existante malf...
4185	223309738	True	[ point de terminaison optique non existante ...
4186	298170718	True	[ point de terminaison optique non existante ...
4187	298170734	True	[ point de terminaison optique non existante ...

4188 rows x 3 columns

### 3.2.2 Statistiques et nouvelles données

Cette partie a pour objectif d'obtenir des statistiques sur les commentaires d'ETI31 et de pouvoir les ajouter comme données dans le dataframe.



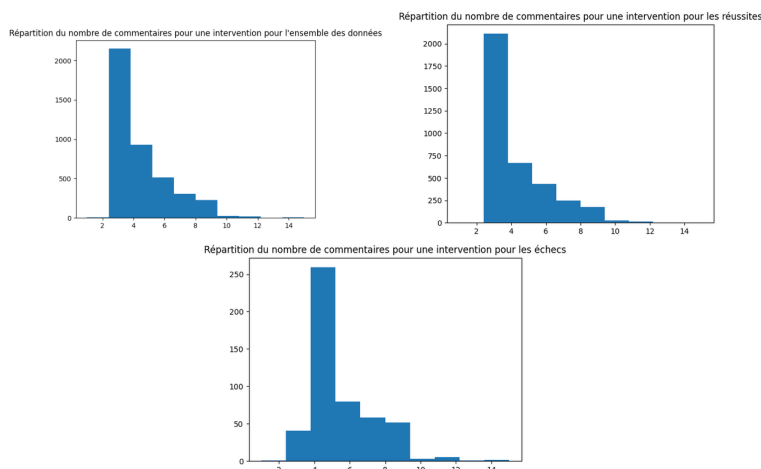
Dans cette partie j'ai fait la distinction entre le dataframe de l'ensemble, le dataframe des réussites et le dataframe des échecs.

On regarder 2 statistiques différentes : le nombre de commentaires et la longueur moyenne de chaque commentaire pour une intervention.

#### Nombre de commentaires

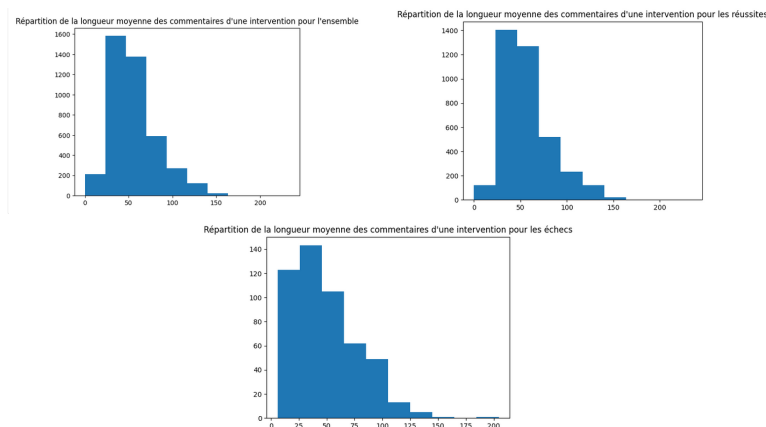
Pour l'ensemble la moyenne est de 4.38, pour les réussites elle est de 5.38 et pour

les réussites elle est de 4.25.



### Longueur moyenne

Pour l'ensemble la longueur moyenne d'un commentaire est de 56.1 caractères, contre 49.6 pour les échecs et 57 pour les réussites.



On peut observer quelques différences entre les échecs et les réussites. Les échecs ont souvent plus de commentaires mais ils sont plus courts. Ces 2 paramètres pourront sans doute être utile pour la création d'un modèle : on ajoute une colonne longueur moyenne commentaire et une colonne nombre\_commentaire.

### 3.2.3 Normalisation des données

A fin d'éviter que la longueur moyenne (réel autour de 56 caractères) n'ait artificiellement plus d'influence, on applique une normalisation aux données.

	nombre_commentaires	Longueur moyenne commentaire	No DESIGNATION	Reussite	COMMENTAIRES
0	-0.755824	-0.328797	299568158	False	[ obtenu marchand christophe le a le clien...
1	0.333062	-0.468411	297892193	True	[ point de terminaison optique existante mal...
2	1.966390	1.752941	297897767	True	[ point de terminaison optique non existante ...
3	-0.755824	-0.627969	223370510	True	[ point de terminaison optique non existante ...
4	-0.755824	-0.877279	299556603	True	[ point de terminaison optique existante mal...
...	...	...	...	...	...
4183	1.966390	-0.359961	223275005	True	[merci de clrer le rendez vous du client car l...
4184	-0.755824	-0.665366	296611510	True	[ point de terminaison optique existante mal...
4185	-0.755824	-0.627969	223309738	True	[ point de terminaison optique non existante ...
4186	0.333062	-0.722707	298170718	True	[ point de terminaison optique non existante ...
4187	-0.755824	-0.927141	298170734	True	[ point de terminaison optique non existante ...

### 3.2.4 Polarité et subjectivité des commentaires

Cette partie permet de déterminer si les mots utilisé dans le commentaires ont une polarité négative ou positive et s'ils sont subjectifs. On peut supposer que les échecs seront plus polarisés vers le négatif que les réussites et que cela permettra de le différencier pour le modèle. On crée donc 2 nouvelles colonnes : polarite et subjectivite. On les remplit en appliquant les fonction polarity et subjectivity de la bibliothèque TextBlob qui est spécialisée dans l'analyse de texte.

On trouve finalement les résultats suivants :

	nombre_commentaires	Longueur moyenne commentaire	Reussite	polarite	subjectivite
0	-0.755824	-0.328797	False	0.000000	0.000000
1	0.333062	-0.468411	True	0.100000	0.100000
2	1.966390	1.752941	True	0.062500	0.062500
3	-0.755824	-0.627969	True	0.133333	0.158333
4	-0.755824	-0.877279	True	0.166667	0.166667
...	...	...	...	...	...
4183	1.966390	-0.359961	True	0.000000	0.000000
4184	-0.755824	-0.665366	True	0.166667	0.166667
4185	-0.755824	-0.627969	True	0.166667	0.166667
4186	0.333062	-0.722707	True	0.100000	0.100000
4187	-0.755824	-0.927141	True	0.166667	0.166667

Dataframe	Polarité	Subjectivité
Ensemble	0,1	0,13
Réussites	0,12	0,14
Echecs	-0,002	0,04

Maintenant que nos données sont mise au propre et normalisée, on a tout ce qu'il faut pour pouvoir passer à la modélisation.

### 3.3 Choix des estimateurs

Les principaux estimateurs pour le machine learning et la création de modèles sont les suivants :

- L'accuracy : proportion de prédictions correctes  $(VP + VN) / (VP + VN + FN + FP)$ .



- Le rappel : proportion de réussites correctement prédites sur l'ensemble des réussites ( $VP/(VP+FN)$ )
- La précision : proportion de véritables réussites parmi les réussites prédites ( $VP/(VP+FP)$ )
- La spécificité : proportion d'échecs correctement prédits sur l'ensemble des échecs ( $VN/(VN+FP)$ )
- La valeur prédictive négative : proportion de véritables échecs parmi les échecs prédits ( $VN/(VN+FN)$ )

Ici, on cherche à prédire des échecs de production qui sont en plus faible proportion donc il faut surtout vérifier que la spécificité et surtout la valeur prédictive négative sont proche de 1.

Si la valeur prédictive négative n'est pas bonne alors trop de données seraient considérées comme des échecs.

Si la spécificité n'est pas bonne alors à l'inverse pas assez d'échecs seraient prédits comme tels.

Il faudra voir avec les équipes ce qu'il est le plus pertinent de connaître pour faire le meilleur modèle possible pour eux.

## 3.4 Modélisation

### 3.4.1 Modélisation avec un arbre de décision

Pour une première approche de modélisation, on fait une modélisation par un arbre de décision. On utilise le module `DecisionTreeClassifier` du module `scikit-learn`.

On a commencé par une premier arbre de profondeur maximale 5 et voici les résultats que l'on obtient :

Accuracy : proportion de prédictions correctes ( $VP + VN / (VP+VN+FN+FP)$ ) : 92.124%

Rappel : proportion de réussites correctement prédites sur l'ensemble des réussites ( $VP/(VP+FN)$ ) : 96.685%

Précision : proportion de véritables réussites parmi les réussites prédites ( $VP/(VP+FP)$ ) : 94.34%

Spécificité : proportion d'échecs correctement prédits sur l'ensemble des échecs ( $VN/(VN+FP)$ ) : 63.158%

Valeur prédictive négative: proportion de véritables échecs parmi les échecs prédits ( $VN/(VN+FN)$ ) : 75.0%

Ce qui n'est pas si mauvais pour un premier essai mais la spécificité et la VPN sont assez bas. On peut probablement faire mieux.

Pour essayer d'améliorer l'arbre de décision, on applique un algorithme qui permet de trouver les meilleurs hyperparamètres selon la VPN. Il consiste à créer x fois un nouveau découpage du dataframe (entraînement et test) et à chaque découpage trouver les meilleurs paramètres parmi une liste donnée en argument (en gardant ceux donnant la meilleur VPN). Finalement après les x itérations, on retient les hyper-paramètres qui apparaissent le plus souvent.

---

```
def BestHyperparameters_vpn
    (trainX,trainY,testX,testY,min_example,depth):
    best_vpn = 0
    best_min_example = 0
    best_depth = 0
    best_criterion = ""
    #On va tester toutes les combinaisons possibles :
    for criterion in ["gini","entropy"]:
        for example in min_example:
            for profondeur in depth:
                #On creer l'arbre
                arbre = DecisionTreeClassifier(criterion=criterion,
                    min_samples_leaf=example,
                    max_depth=profondeur)
                arbre.fit(trainX, trainY)
                #On calcule la prediction et la vpn associee
                predictionY = arbre.predict(testX)
                vpn =
                    round(sk.metrics.precision_score(testY,predictionY,pos_label=0),5)*100
                #On compare
                if vpn>best_vpn :
                    best_vpn = vpn
                    best_min_example = example
                    best_depth = profondeur
                    best_criterion= criterion

    return (best_vpn,best_min_example,best_depth,best_criterion)

def Finetune(nombre_essai,min_example,depth):
    l_vpn = []
    l_best_min_example = []
    l_best_depth = []
    l_best_criterion = []

    for i in range (0,nombre_essai) :
        if i%10==0 :
            print(i)
            trainX,trainY,testX,testY = creation_test_train(df2,"Reussite")
            best_vpn,best_min_example,best_depth,best_criterion =
                BestHyperparameters_vpn(trainX,trainY,testX,testY,min_example,depth)
```

```

l_vpn.append(best_vpn)
l_best_min_example.append(best_min_example)
l_best_depth.append(best_depth)
l_best_criterion.append(best_criterion)
return mean(l_vpn),
Counter(l_best_min_example).most_common(1)[0][0],
Counter(l_best_depth).most_common(1)[0][0],
Counter(l_best_criterion).most_common(1)[0][0]

```

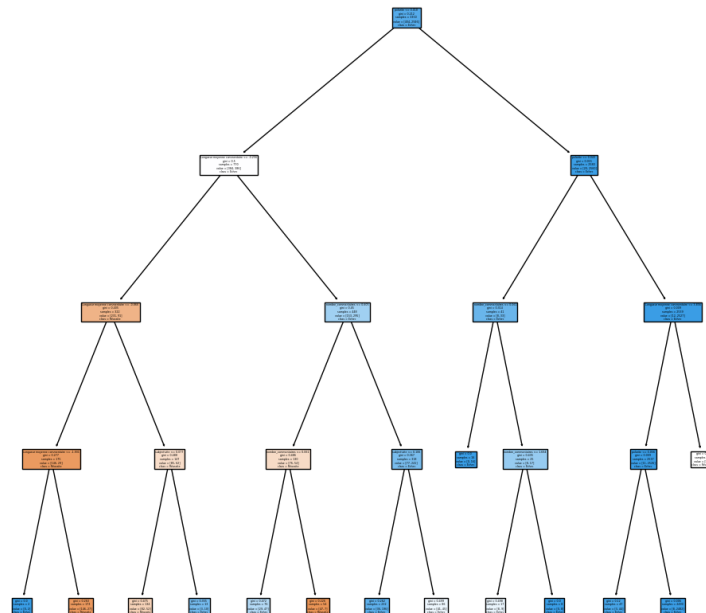
Finetune(500,[2,3,4,5],[2,3,4,5,6])

---

En choisissant d'itérer 500 fois comme sur la dernière ligne on trouve finalement une valeur prédictive négative moyenne de 72.6 et que les meilleurs paramètres sont :

- min\_example = 2
- max\_depth = 4
- criterion = "gini"

En appliquant le même algorithme mais cette fois la spécificité comme moyen de comparaison on trouve les mêmes paramètres !



Voici les mesures pour cet arbre :

Accuracy : proportion de prédictions correctes  $(VP + VN / (VP + VN + FN + FP))$   
: 93.3%

Rappel : proportion de réussites correctement prédites sur l'ensemble des réussites  
 $(VP / (VP + FN))$  : 96.429%

Précision : proportion de véritables réussites parmi les réussites prédites  $(VP / (VP + FP))$   
: 95.902%

Spécificité : proportion d'échecs correctement prédits sur l'ensemble des échecs  
 $(VN / (VN + FP))$  : 72.727%

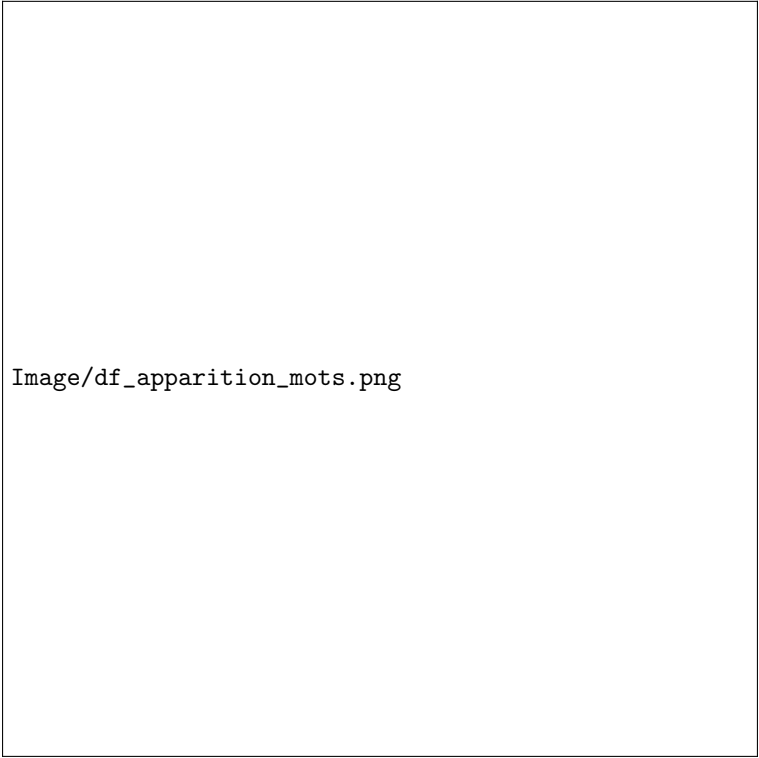
Valeur prédictive négative: proportion de véritables échecs parmi les échecs  
prédits  $(VN / (VN + FN))$  : 75.472%

On a déjà amélioré les hyper-paramètres mais il est sans doute possible d'encore améliorer les résultats du modèle. Pour cela nous appliquons des méthodes ensemblistes de Machine Learning aux arbres de décisions.

### 3.4.2 Associations des mots à une réussite ou à un échec

Dans cette partie, on cherche à déterminer si certains mots plus souvent utilisés dans les échecs ou les réussites mais sans se baser sur la polarité de la bibliothèque textBlob. Pour cela on crée une base de données avec tous les mots qui apparaissent dans l'ensemble des commentaires et le nombre de fois qu'ils apparaissent dans des réussites et le nombre de fois où ils apparaissent dans des échecs. Cependant, vu que les réussites sont bien plus nombreuses que les échecs, on s'est plutôt intéressé à la proportion de réussites ou d'échecs contenant ce mot parmi l'ensemble des réussites ou des échecs.

On a fait le choix de ne garder que ceux où la différence d'apparition est supérieure à 10% mais peut-être qu'un autre seuil serait plus pertinent. Voilà une partie du tableau que l'on obtient :



Image/df\_apparition\_mots.png

### **3.4.3 Autres méthodes similaires aux arbres de décision**

#### **3.4.4 Modélisation K-nearest neighbors**

#### **3.4.5 Modélisation régression logistique**

#### **3.4.6 SVM**

#### **3.4.7 Naive Bayes**