Projet Datascientest

Prédiction du succès d'une campagne de Marketing d'une banque

Réalisé par :

Jonathan BRULTEY - Seyfeddine CHEOUR - Carine TAN - Graziella DA SILVA

Introduction

L'analyse des données marketing d'une institution bancaire ou financière est l'une des applications les plus courantes de la Data Science et du Machine Learning.

Le jeu de données sur lequel nous allons travailler présente des données personnelles de clients d'une banque qui ont été "télémarkétés" afin de souscrire à un produit appelé "dépôt à terme".

Un dépôt à terme est un compte dans lequel le client ne touche plus aux fonds déposés pendant une période définie contre des intérêts qu'ils gagnent à la fin du terme. Ce serait un produit qui n'intéressait pas l'ensemble de la clientèle de la banque puisqu'il faut suffisamment d'épargne pour déposer dans le compte de dépôt à terme en question.

Le but de notre étude permettra de répondre à la question suivante : Quels clients sont susceptibles d'ouvrir des comptes à dépôts à terme ? Aussi, il sera fondamental d'être en mesure d'interpréter nos prédictions en expliquant pourquoi on considère qu'un client va souscrire ou non.

Pour répondre à cette question, nous allons dans un premier temps effectuer une analyse des données pour avoir les premiers éléments de réponses et effectuer le prétraitement nécessaire pour la suite de notre étude. Dans un second temps, nous allons tenter de définir un modèle prédictif nous permettant de prédire les clients qui ont souscrit à un dépôt à terme. Nous enchainerons avec l'interprétation de notre modèle. Et pour conclure, nous proposerons des recommandations pour améliorer les futures campagnes.

Le jeu de données est téléchargeable au lien suivant:

https://www.kaggle.com/janiobachmann/bank-marketing-dataset

I. Analyse exploratoire de la base de données

Cette première partie nous permet de prendre connaissance du jeu de données en l'explorant de façon globale puis de façon plus détaillée avec des statistiques descriptives. Par ailleurs, l'analyse et la visualisation graphique de nos données nous permettra de mettre en lumière les éléments pertinents de notre étude.

Suite à cela, le prétraitement de la base de données est nécessaire avant toute exploitation de données : en effet cette étape nous permettra de rectifier certaines données tout en mettant en évidence des phénomènes et obtenir de meilleurs résultats pour la partie modélisation.

A. Présentation du jeu de données et enjeux

a. Caractéristiques du jeu de données

Nous allons explorer et présenter le jeu de données. Le jeu de données est composé de 17 colonnes dont 10 qui sont catégorielles et 7 numériques avec un total de 11 162 lignes.
'deposit' est notre variable à prédire dans la partie suivante, et les potentiels features sont les autres variables.

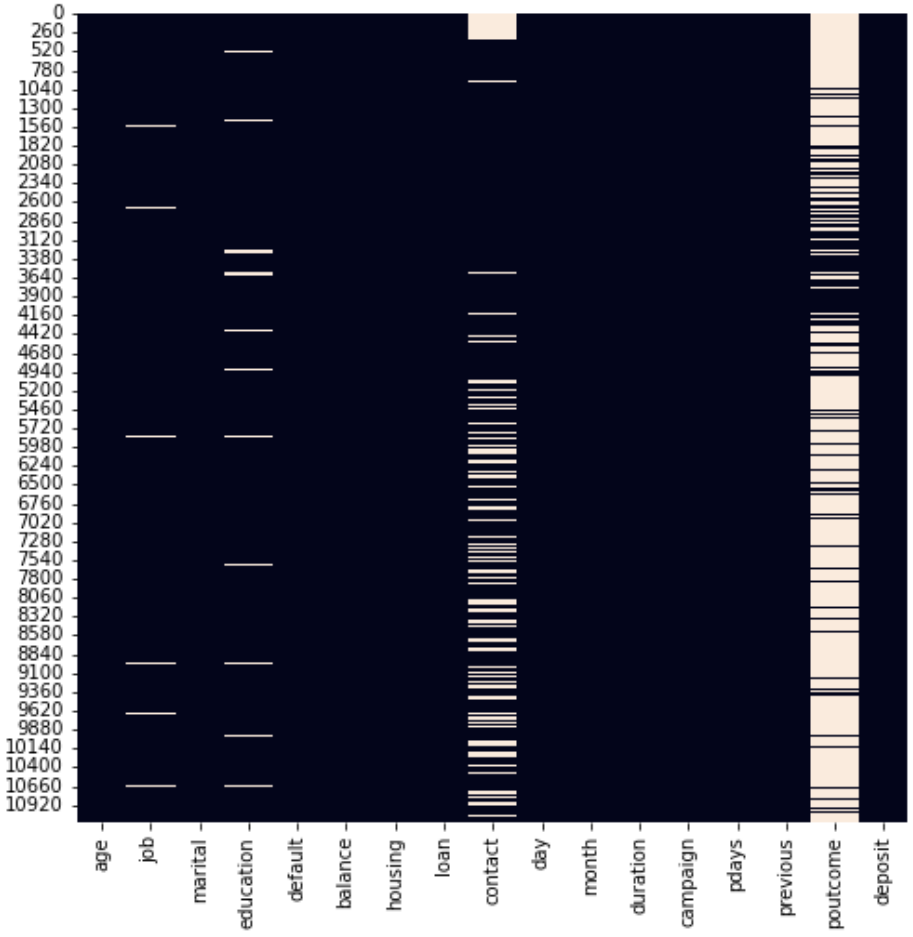
	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes
5	42	management	single	tertiary	no	0	yes	yes	unknown	5	may	562	2	-1	0	unknown	yes
6	56	management	married	tertiary	no	830	yes	yes	unknown	6	may	1201	1	-1	0	unknown	yes
7	60	retired	divorced	secondary	no	545	yes	no	unknown	6	may	1030	1	-1	0	unknown	yes
8	37	technician	married	secondary	no	1	yes	no	unknown	6	may	608	1	-1	0	unknown	yes
9	28	services	single	secondary	no	5090	yes	no	unknown	6	may	1297	3	-1	0	unknown	yes

Le tableau ci-dessous présente les différentes variables avec leur définition :

Variables	Type	Définition
age	numérique	Age du client
job	catégorielle	Métier du client
marital	catégorielle	Etat civil du client
education	catégorielle	Nivau d'éducation atteint par le client
default	catégorielle	Si le client a un crédit en défaut
balance	numérique	Solde bancaire du client
housing	catégorielle	Si le client a un prêt immobilier
loan	catégorielle	Si le client a un prêt personnel
contact	catégorielle	Type de communication du client
day	numérique	Jour de la semaine du dernier contact passé
month	numérique	Mois du dernier contact passé
duration	numérique	Durée du contact pour la campagne actuelle
campaign	numérique	Nombre de contacts effectués dans la campagne acutelle
pdays	numérique	Nombre de jours écoulés depuis la campagne passée
previous	numérique	Nombre de contacts effectués avant cette campagne actuelle
poutcome	catégorielle	Résultat de la dernière campagne du client
deposit	catégorielle	Résultat de la campagne actuelle

- **age** : (int) age
- **job** : (object) métier- ['admin.', 'technician', 'services', 'management', 'retired', 'blue-collar', 'unemployed', 'entrepreneur', 'housemaid', 'unknown', 'self-employed', 'student']
- **marital** : (object) statut matrimonial - ['married', 'single', 'divorced']
- **education** : (object) niveau d'éducation - ['secondary', 'tertiary', 'primary', 'unknown']
- **default** : (object) a un crédit en défaut de paiement - ['no', 'yes']
- **balance** : (int) solde du compte
- **housing** : (object) a un crédit immobilier - ['yes', 'no']
- **loan** : (object) a un autre crédit - ['no', 'yes']
- **contact** : (object) moyen utilisé pour contacté le client - ['unknown', 'cellular', 'telephone']
- **day** : (int) jour du dernier contact
- **month** : (object) mois du dernier contact - ['may', 'jun', 'jul', 'aug', 'oct', 'nov', 'dec', 'jan', 'feb', 'mar', 'apr', 'sep']
- **duration** : (int) durée du dernier contact
- **campaign** : (int) nombre de contact lors de cette campagne
- **pdays** : (int) nombre de jour depuis le dernier contact
- **previous** : (int) nombre de contact total lors des campagnes précédentes
- **poutcome** : (object) résultat de la précédente campagne - ['unknown', 'other', 'failure', 'success']
- **deposit** : (object) le client a t-il souscrit un dépot à terme - ['yes', 'no']

La campagne actuelle a sollicité 11 162 clients et ne présente aucune données vides.



Néanmoins, nous voyons que certaines variables présentent la modalité Unknown. Le graphique au-dessus représente les 11 162 lignes du dataset. Chaque donnée avec la mention Unknown est en blanc. Il y a quatre colonnes qui sont concernées : 'poutcome' avec 74.6% de 'unknown', 'contact' (21%), 'education' (4,5%) et 'job' (0.6%).

Nous gardons en tête cette information et nous y reviendrons plus tard.

```
poutcome     74.592367
contact      21.017739
education     4.452607
job           0.627128
```

b. Analyse descriptive du jeu de données

Nous allons faire une analyse descriptive des données pour prendre connaissance du jeu de données.

Lorsque nous regardons les statistiques élémentaires des données numériques, nous voyons qu'elles ne sont pas toutes à la même échelle. Il faudra nécessairement traiter ce point pour les mettre à l'échelle dans le pré-traitement concernant la partie modélisation.

	count	mean	std	min	25%	50%	75%	max
age	11162.0	41.231948	11.913369	18.0	32.0	39.0	49.00	95.0
balance	11162.0	1528.538524	3225.413326	-6847.0	122.0	550.0	1708.00	81204.0
day	11162.0	15.658036	8.420740	1.0	8.0	15.0	22.00	31.0
duration	11162.0	371.993818	347.128386	2.0	138.0	255.0	496.00	3881.0
campaign	11162.0	2.508421	2.722077	1.0	1.0	2.0	3.00	63.0
pdays	11162.0	51.330407	108.758282	-1.0	-1.0	-1.0	20.75	854.0
previous	11162.0	0.832557	2.292007	0.0	0.0	0.0	1.00	58.0

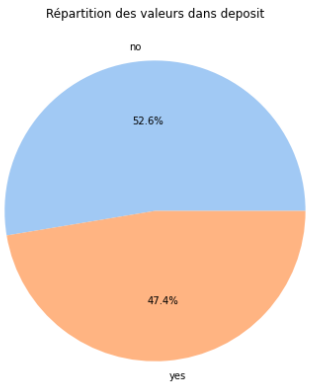
Si nous nous intéressons aux données catégorielles, nous voyons qu'il est aisément possible de recoder certaines d'entre elles qui n'ont que deux modalités en variable dichotomique. Pour les autres, nous verrons ce qu'elles peuvent nous apporter un peu plus tard.

```
job---------------------- ['admin.' 'technician' 'services' 'management' 'retired' 'blue-collar'
 'unemployed' 'entrepreneur' 'housemaid' 'unknown' 'self-employed'
 'student']
marital------------------ ['married' 'single' 'divorced']
education---------------- ['secondary' 'tertiary' 'primary' 'unknown']
default------------------ ['no' 'yes']
housing------------------ ['yes' 'no']
loan---------------------- ['no' 'yes']
contact------------------ ['unknown' 'cellular' 'telephone']
month--------------------- ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']
poutcome------------------ ['unknown' 'other' 'failure' 'success']
deposit------------------- ['yes' 'no']
```

B. Analyse de données

Nous pouvons dorénavant analyser de façon plus approfondie chacune de nos variables notamment par rapport à la variable cible deposit. Nous pourrons avoir les premiers éléments de réponse à notre problématique.

On notera tout d'abord que la target "deposit" est équilibrée, rendant ainsi la métrique accuracy pertinente.



a. Visualisation des données numériques

Nous faisons ici une simple description statistique pour chacune des variables numériques.

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

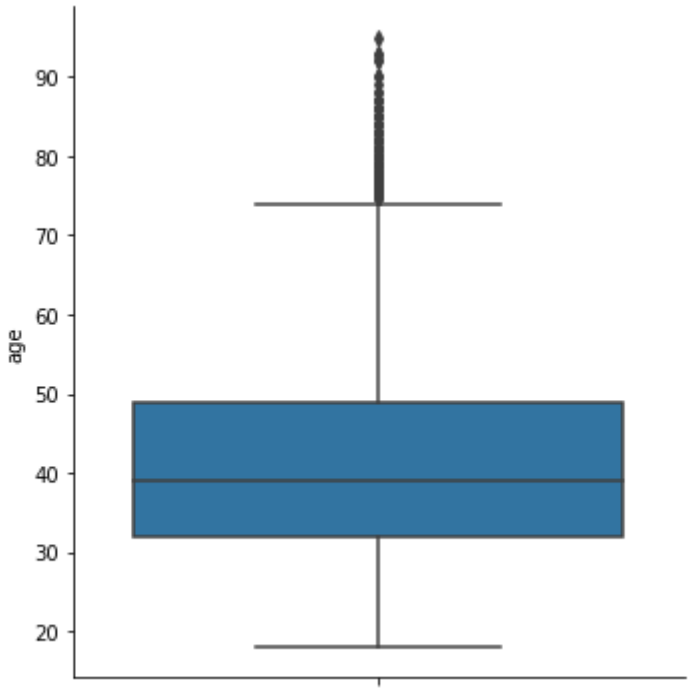
Il n'y a pas de valeur aberrante, toutes les données sont plausibles (âge 18-95 ans, day 1-31 etc). Il y a un écart significatif sur les échelles de données. Balance va de -6847 à 81204, tandis que l'âge va de 18 à 95. On devra donc normaliser les données pour nos modèles.

- *Analyse de la donnée 'age'*

Cette donnée représente l'âge du client. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target deposit.

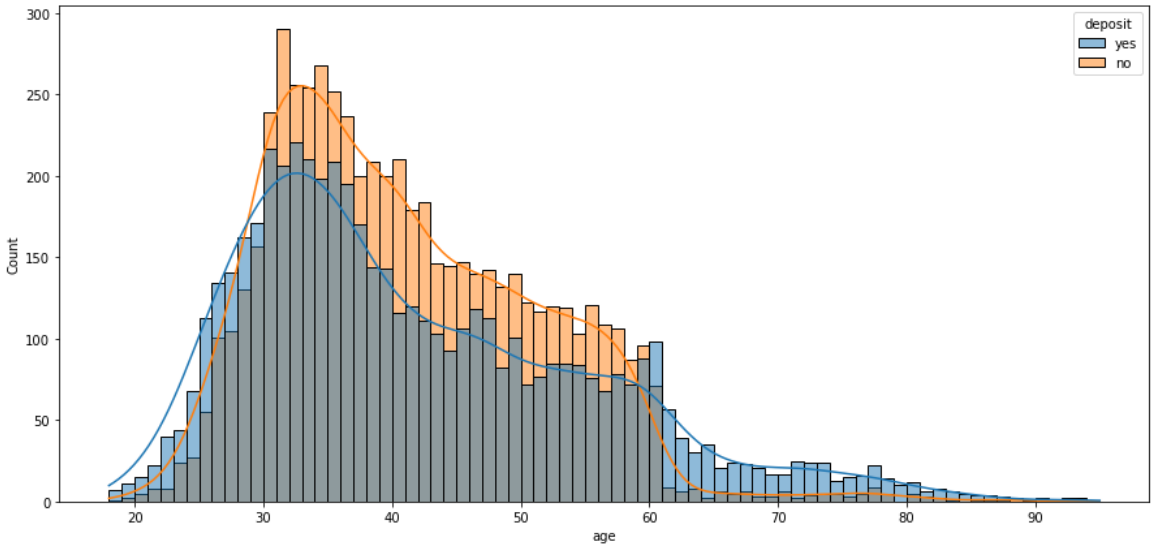
Analyse graphique :

La majorité des clients a entre 30 et 50 ans.



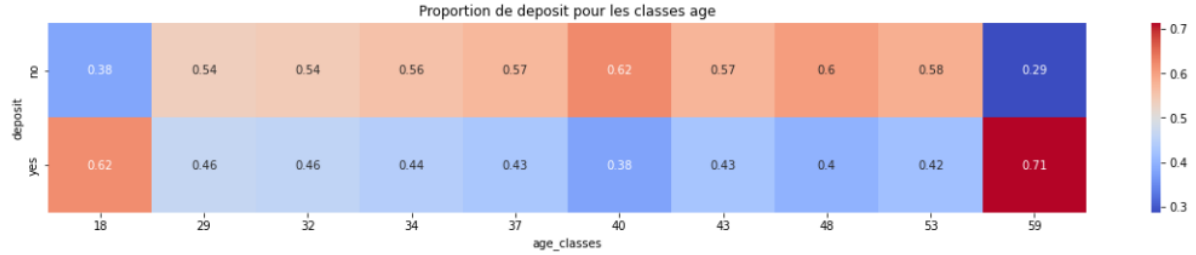
Analyse de l'âge en fonction de deposit :

Les clients de 19-29 ans et les 60+ ont plus souvent tendance à dire oui. En revanche pour les 30-59 ans, c'est le non qui l'emporte.



Si nous allons plus loin et si nous découpons l'âge en 10 classes, on observe également que ce sont les plus jeunes et les plus âgés qui souscrivent. Pour le reste des clients, c'est relativement similaire. On peut tout de même noter des scores un peu plus bas autour de 40-50 ans.

age_classes	count	mean	std	min	25%	50%	75%	max
18	1223.0	25.714636	2.190107	18.0	25.0	26.0	27.0	28.0
29	1280.0	30.131250	0.791842	29.0	29.0	30.0	31.0	31.0
32	941.0	32.493092	0.500218	32.0	32.0	32.0	33.0	33.0
34	1359.0	34.974982	0.812798	34.0	34.0	35.0	36.0	36.0
37	1066.0	37.974672	0.817827	37.0	37.0	38.0	39.0	39.0
40	920.0	40.966304	0.821339	40.0	40.0	41.0	42.0	42.0
43	1253.0	45.025539	1.416240	43.0	44.0	45.0	46.0	47.0
48	1048.0	49.937977	1.418586	48.0	49.0	50.0	51.0	52.0
53	1108.0	55.385379	1.695534	53.0	54.0	55.0	57.0	58.0
59	964.0	66.201245	7.763439	59.0	60.0	63.0	72.0	95.0

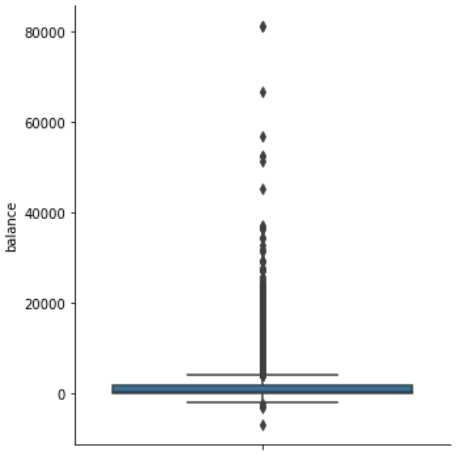


- *Analyse de la donnée 'balance'*

Cette donnée représente le solde du compte du client à un instant t. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

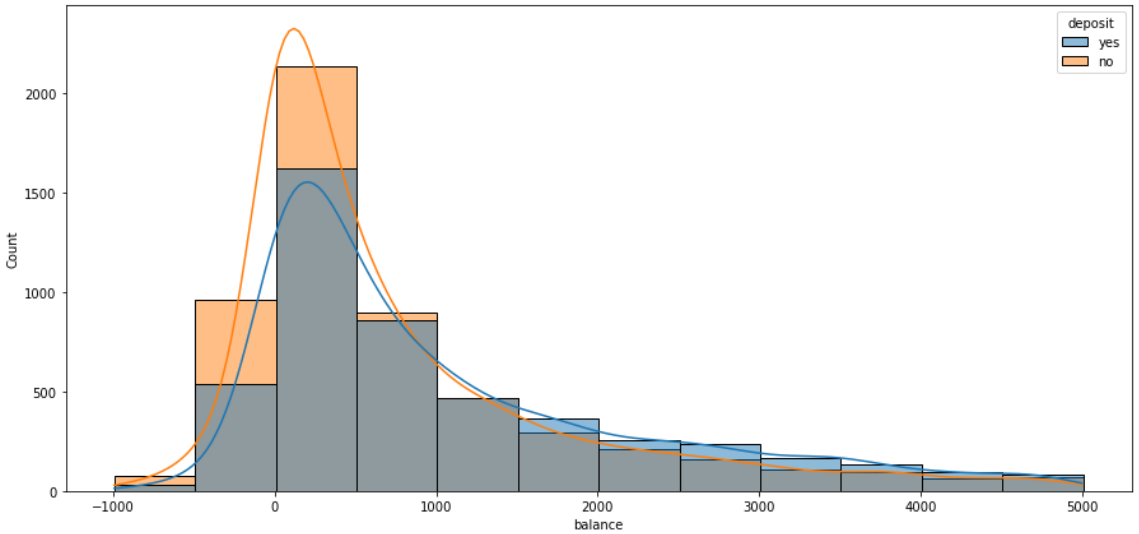
Analyse graphique :

Il y a des valeurs extrêmes qui écrase le graphique. Il faudra certainement retravailler ces données par la suite. On va également réaliser un focus sur les balances élevées.



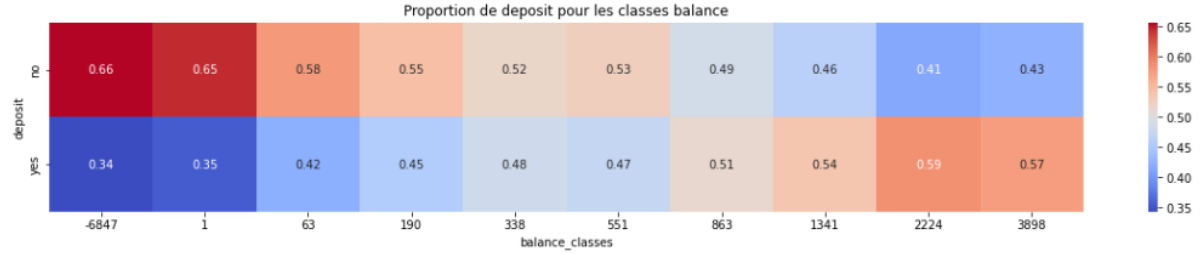
Analyse de 'balance' en fonction de 'deposit' :

En excluant les outliers de 'balance', on observe moins de souscriptions pour les plus faibles balances. Cela s'équilibre à partir de 1000 euros.



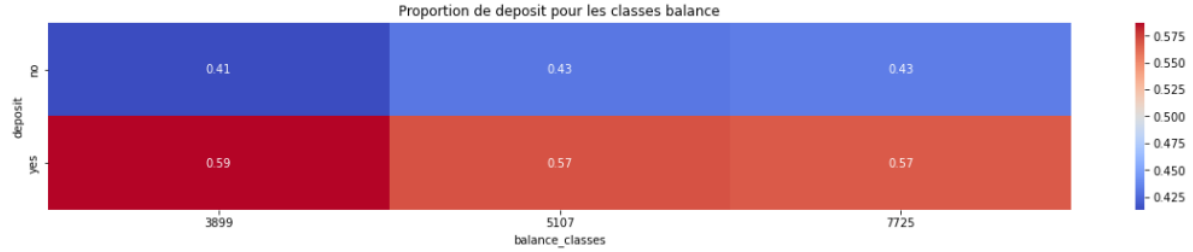
On observe une tendance nette qui est que plus la balance augmente, plus le client est susceptible de souscrire.

balance_classes	count	mean	std	min	25%	50%	75%	max
-6847	1462.0	-152.379617	329.087753	-6847.0	-210.75	0.0	0.00	0.0
1	778.0	26.041131	19.259880	1.0	8.00	23.0	43.00	62.0
63	1109.0	123.330027	35.932287	63.0	93.00	123.0	154.00	189.0
190	1119.0	261.521895	43.443682	190.0	223.00	261.0	300.00	337.0
338	1115.0	439.281614	60.617347	338.0	386.00	437.0	491.00	550.0
551	1114.0	689.955117	89.656536	551.0	608.25	681.5	767.00	862.0
863	1116.0	1079.656810	141.599446	863.0	951.00	1070.0	1204.25	1340.0
1341	1118.0	1732.496422	249.546578	1341.0	1521.00	1708.5	1943.75	2223.0
2224	1114.0	2944.773788	477.962356	2225.0	2544.00	2885.0	3342.75	3885.0
3898	1117.0	8195.107431	6790.608854	3899.0	4745.00	6027.0	8876.00	81204.0



Certaines valeurs sont surprenantes, on aurait tendance à croire qu'aucune souscription ne se fait avec un solde négatif. On constate bien que le taux de succès est plus bas pour les soldes négatifs, mais on obtient tout de même 1/3 de oui.

balance_classes	count	mean	std	min	25%	50%	75%	max
3899	373.0	4468.000000	347.801219	3899.0	4151.0	4464.0	4745.00	5106.0
5107	372.0	6183.244624	762.307019	5108.0	5475.0	6027.0	6810.75	7724.0
7725	372.0	13944.096774	9318.953612	7735.0	8876.0	10894.5	14524.75	81204.0



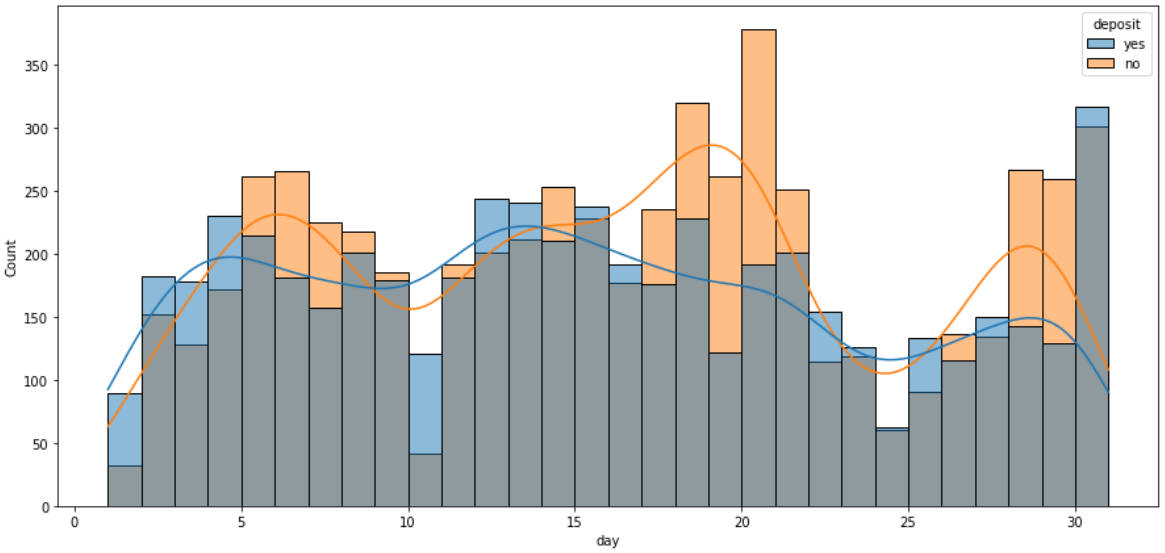
Lorsqu'on se concentre sur les balances les plus hautes, la tendance positive disparaît. Il semble donc qu'à partir d'un certain seuil, la balance n'est plus un facteur important.

- *Analyse de la donnée 'day'*

Cette donnée représente le jour du dernier contact. Nous croiserons celle-ci avec la valeur target deposit.

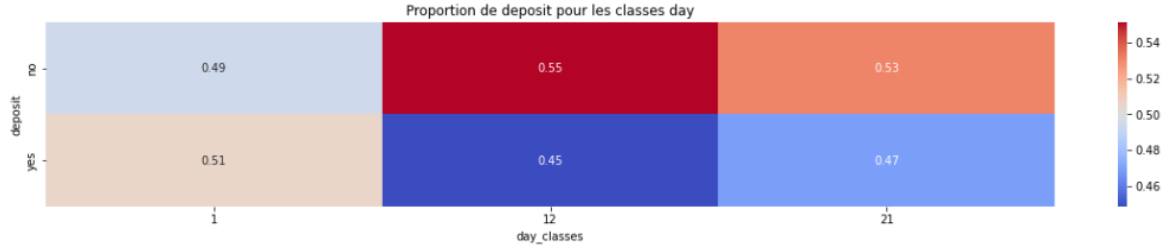
Analyse de 'day' en fonction de 'deposit' :

Il n'y a pas de tendance claire qui se dégage sur 'day'.



Le jour du contact à l'air d'avoir très peu d'incidence dans la décision finale du client. Cela semble logique, si je suis prêt à accepter de souscrire le 5 du mois, cela devrait vraisemblablement être également le cas le 20. Un facteur qui peut entrer en jeu est la date de paie ainsi que les factures des clients.

day_classes	count	mean	std	min	25%	50%	75%	max
1	3789.0	6.180523	2.831810	1.0	4.0	6.0	8.0	11.0
12	4109.0	16.099294	2.641031	12.0	14.0	16.0	18.0	20.0
21	3264.0	26.104473	3.277416	21.0	23.0	27.0	29.0	31.0

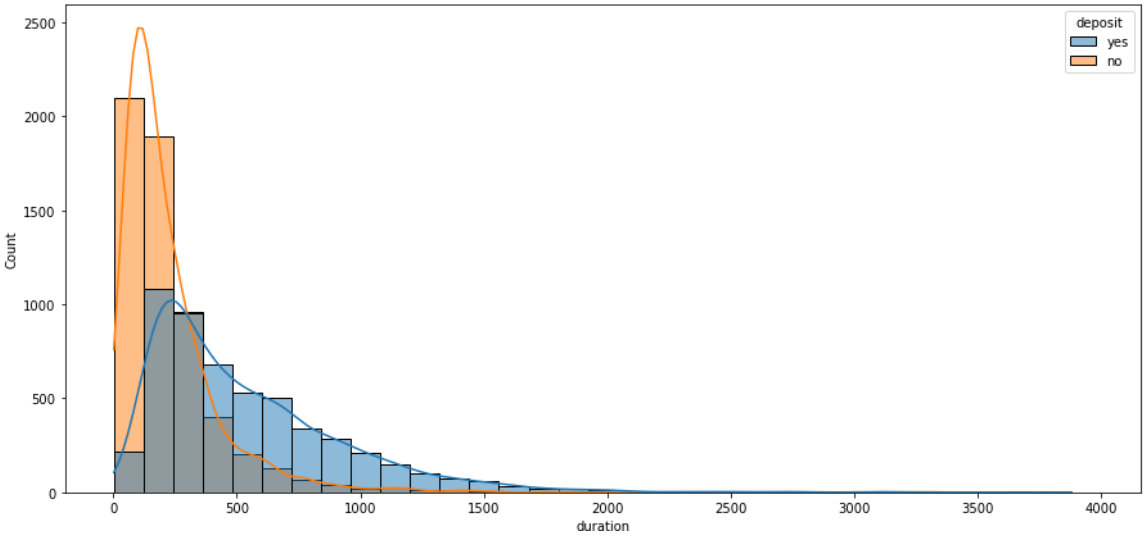


- *Analyse de la donnée 'duration'*

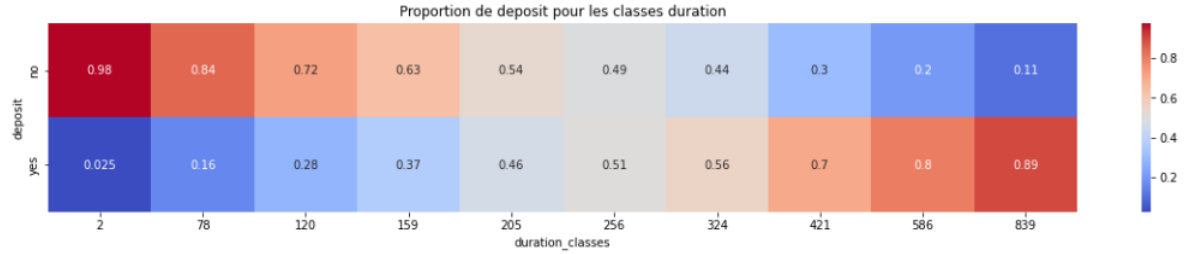
Cette donnée représente la durée du dernier contact. Nous croiserons celle-ci avec la valeur target deposit.

Analyse de duration en fonction de 'deposit' :

La durée s'exprime en seconde. Ici chaque barre représente 2 minutes. Un court échange (moins de 4 minutes) implique souvent un non.



duration_classes	count	mean	std	min	25%	50%	75%	max
2	1134.0	47.111111	21.602973	2.0	29.00	51.5	65.0	77.0
78	1128.0	98.993794	11.964565	78.0	89.00	99.0	109.0	119.0
120	1089.0	139.236915	11.314800	120.0	130.00	139.0	149.0	158.0
159	1127.0	180.057675	13.317054	159.0	168.00	179.0	192.0	204.0
205	1114.0	229.581688	14.857486	205.0	216.00	229.0	243.0	255.0
256	1107.0	286.779584	19.998106	256.0	268.00	285.0	304.0	323.0
324	1115.0	368.287892	28.172790	324.0	343.00	366.0	393.0	420.0
421	1119.0	498.922252	47.438994	421.0	458.00	496.0	538.5	585.0
586	1114.0	696.820467	71.373951	586.0	636.25	689.5	755.0	838.0
839	1115.0	1178.600897	352.172342	839.0	939.50	1080.0	1308.0	3881.0



On observe une tendance très nette: plus la durée de l'appel est élevée, plus le client a de chance de souscrire. La durée semble avoir un impact très fort sur 'deposit'. Cependant, nous ne pourrons pas utiliser cette donnée dans le cadre du modèle prédictif. En effet, la durée n'est connue qu'une fois que l'appel est effectué. Or, nous souhaitons justement prédire si c'est pertinent ou non de contacter le client.

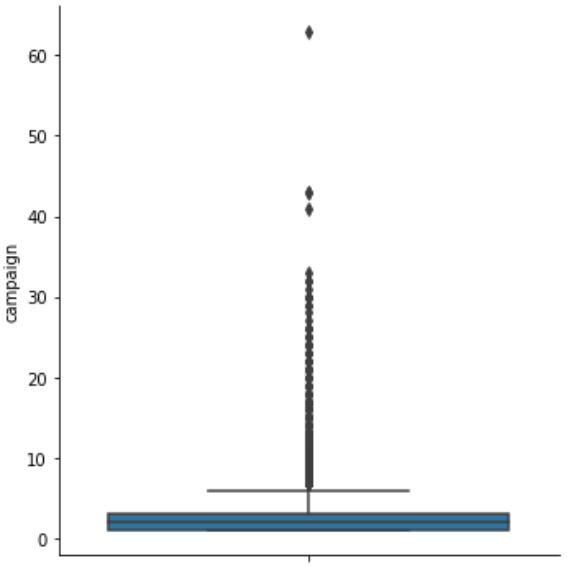
- *Analyse de la donnée 'campaign'*

Cette donnée représente le nombre de fois que l'on a contacté le client lors de la campagne. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target deposit.

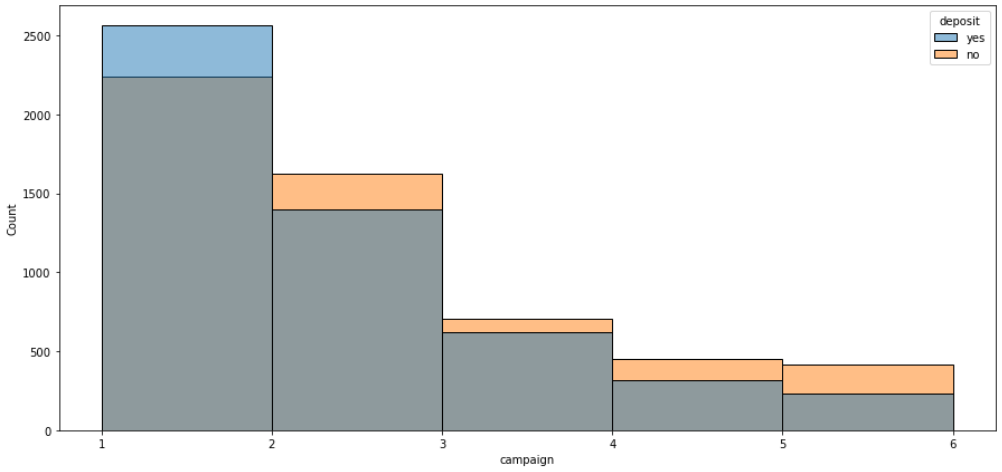
Analyse graphique :

Le graphique est écrasé à cause des valeurs extrêmes. La majorité des clients ont été contactés 3 fois ou moins.

```
count    11162.000000
mean         2.508421
std          2.722077
min          1.000000
25%          1.000000
50%          2.000000
75%          3.000000
max         63.000000
Name: campaign, dtype: float64
```



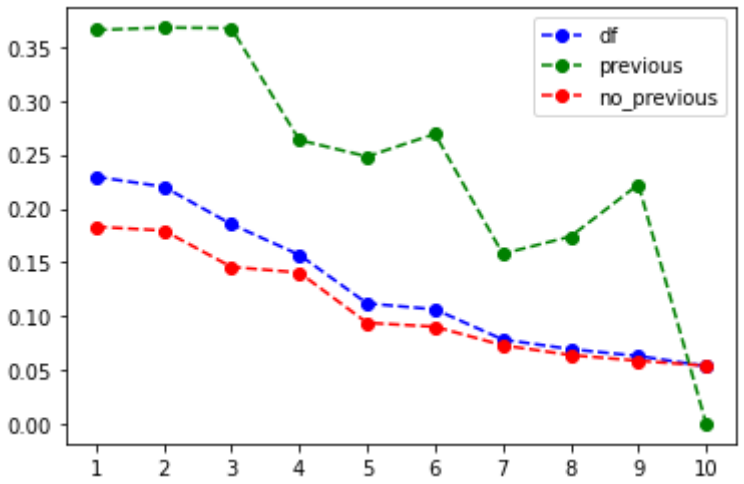
Analyse de distribution en fonction de 'deposit' :



De prime abord, on pourrait croire que plus on appelle, moins on est efficace. Cependant, 'campaign' = 1 montre uniquement les clients ayant fait un seul appel. Pourtant un client avec 'campaign' = 2 a refusé la proposition lors du premier appel. On va retravailler les données pour avoir une meilleure vision du taux d'acceptation.

Comme pour 'duration', ce champ ne sera pas utilisé pour le modèle prédictif, car nous ne connaissons pas cette donnée tant que la campagne n'est pas terminée. On pourra tout de même étudier le taux de retour en fonction du nombre d'appels et le prendre en compte lors des futures campagnes.

Le graphique ci-dessous montre le pourcentage de souscription en fonction du nombre d'appels. Nous avons également fait la distinction entre les clients ayant déjà participé à une campagne ou non.



Il y a un décrochage du taux de souscription après 3 appels sur les clients déjà connus. La diminution est plus régulière pour les autres.

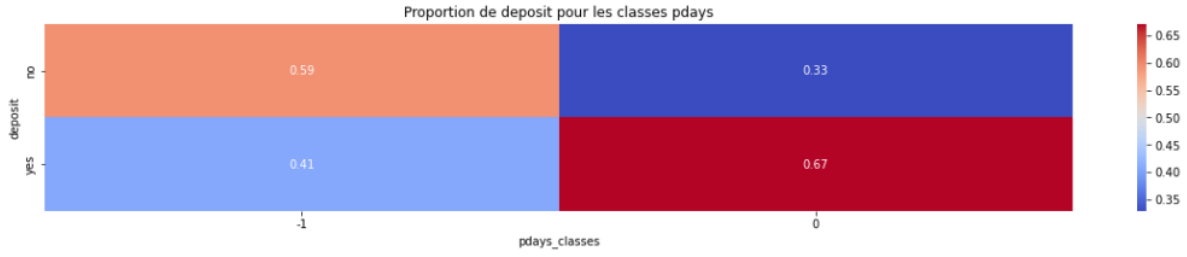
- *Analyse de la donnée 'pdays'*

Cette donnée représente le nombre de jours depuis le dernier contact. Nous croiserons celle-ci avec la valeur target deposit.

Analyse de 'pdays' en fonction de 'deposit' :

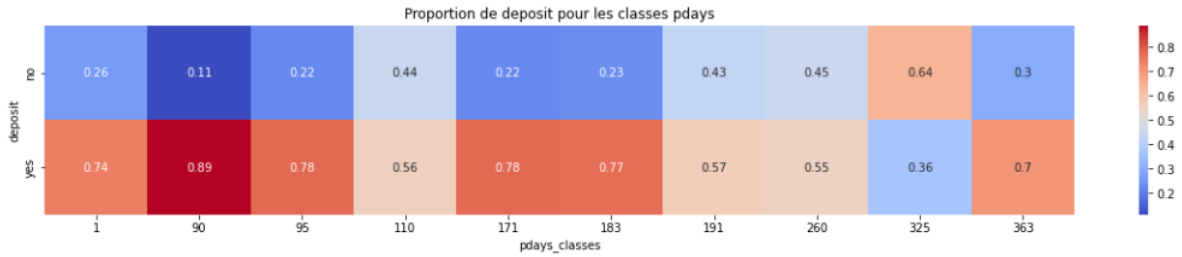
Il y a beaucoup de clients avec 'previous' = -1. Ce sont les clients qui n'ont jamais été contactés auparavant. On constate que les clients déjà connus de la banque sont plus susceptibles de souscrire le DAT.

pdays_classes	count	mean	std	min	25%	50%	75%	max
-1	8324.0	-1.000000	0.000000	-1.0	-1.0	-1.0	-1.0	-1.0
0	2838.0	204.818182	122.196901	1.0	98.0	182.0	286.0	854.0

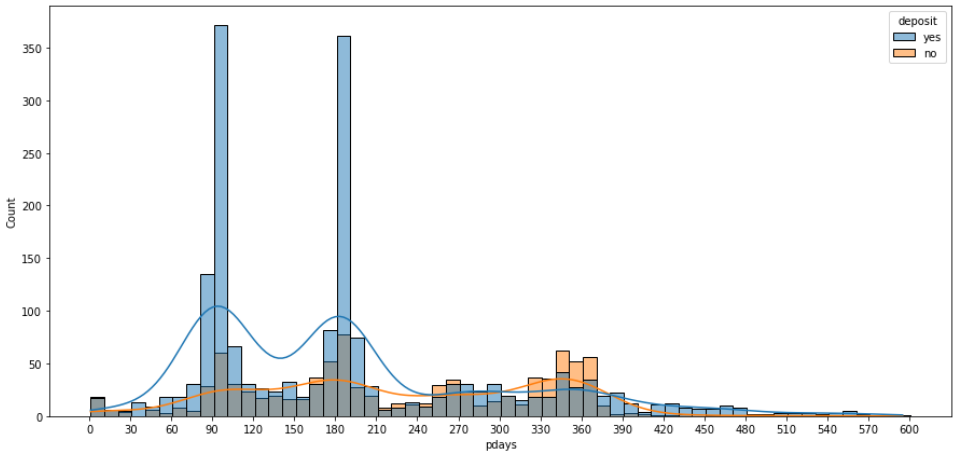


Nous allons donc exclure les clients n'ayant jamais été contactés ('pdays' = -1) puis nous découpons ce nouveau tableau en 10 classes de taille similaire.

pdays_classes	count	mean	std	min	25%	50%	75%	max
1	297.0	62.579125	29.186284	1.0	43.0	78.0	86.0	89.0
90	305.0	91.924590	1.174378	90.0	91.0	92.0	93.0	94.0
95	251.0	99.741036	3.929462	95.0	96.0	99.0	103.0	109.0
110	292.0	141.551370	18.769925	110.0	125.0	141.0	160.0	170.0
171	304.0	179.194079	3.253609	171.0	177.0	181.0	182.0	182.0
183	269.0	185.505576	2.331838	183.0	183.0	185.0	187.0	190.0
191	269.0	216.881041	23.114106	191.0	196.0	208.0	238.0	259.0
260	285.0	287.715789	19.193417	260.0	270.0	286.0	302.0	324.0
325	283.0	344.583039	10.006877	325.0	337.0	345.0	352.0	362.0
363	283.0	445.406360	114.232752	363.0	370.0	394.0	464.5	854.0



Quand on regarde les proportions de “oui” en fonction de ‘pdays’, on ne constate pas de tendance nette. Les scores montent et descendent sans vraiment de logique.

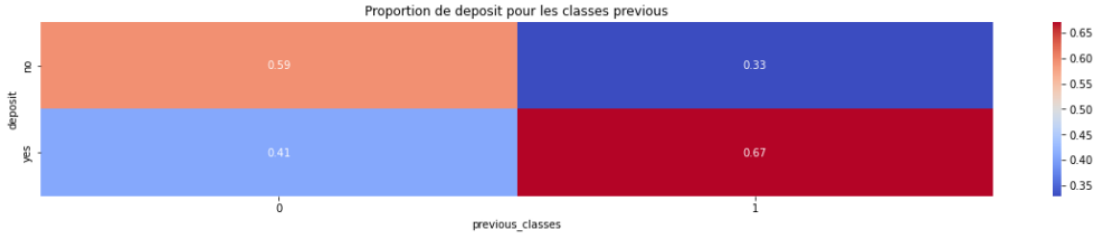


En revanche, ici, on comprend mieux les scores du graphique précédent. Il y a eu 2 grosses campagnes d'appels, autour de 90 et 180 jours qui ont un très bon succès. On a également deux campagnes d'appels plus discrètes, autour de 270 jours et de 360 jours. La banque semble faire des campagnes par trimestre car 90, 180, 270 et 360 jours représentent respectivement 3, 6, 9 et 12 mois.

- *Analyse de la donnée 'previous'*

Cette donnée représente le nombre de fois que l'on a contacté le client lors de la campagne. Les chiffres semblent similaires entre 'previous' = 0 & 'pdays' = -1, on vérifie cela.

previous_classes	count	mean	std	min	25%	50%	75%	max
0	8324.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
1	2838.0	3.274489	3.559216	1.0	1.0	2.0	4.0	58.0

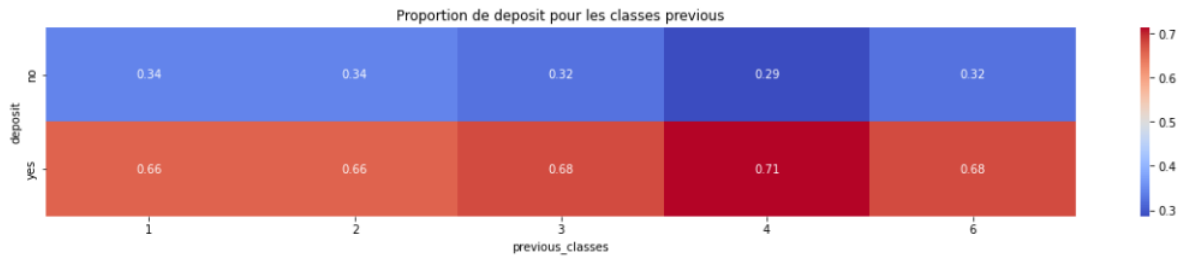


On constate que 'previous' = 0 et 'pdays' = -1 sont équivalents, cela correspond au fait que le client n'avait jamais été contacté auparavant.

previous \ pdays	False	True
False	2838	0
True	0	8324

Lorsque 'previous' est supérieur à 0, on constate que les valeurs sont similaires. Cela veut dire que le nombre de contacts auprès du client n'a pas d'importance. Ce qui compte, c'est de savoir si oui ou non il a été contacté au moins une fois. Lors de la création du modèle de machine learning, on pourra envisager de transformer previous en une variable binaire en regroupant ceux qui ont été contactés au moins une fois lors d'une ancienne campagne (previous >= 1) des autres clients qui n'ont jamais été contactés (previous = 0).

previous_classes	count	mean	std	min	25%	50%	75%	max
1	887.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0
2	693.0	2.000000	0.000000	2.0	2.0	2.0	2.0	2.0
3	435.0	3.000000	0.000000	3.0	3.0	3.0	3.0	3.0
4	409.0	4.403423	0.491185	4.0	4.0	4.0	5.0	5.0
6	414.0	9.454106	5.772656	6.0	6.0	8.0	10.0	58.0

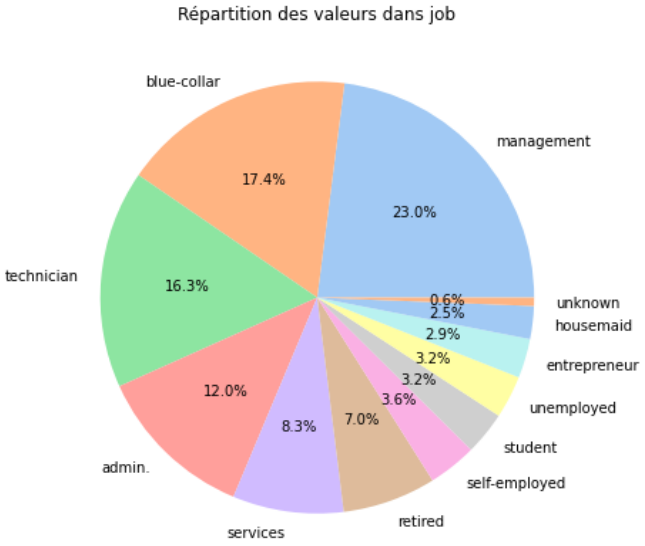


b. Visualisation des données catégoriques

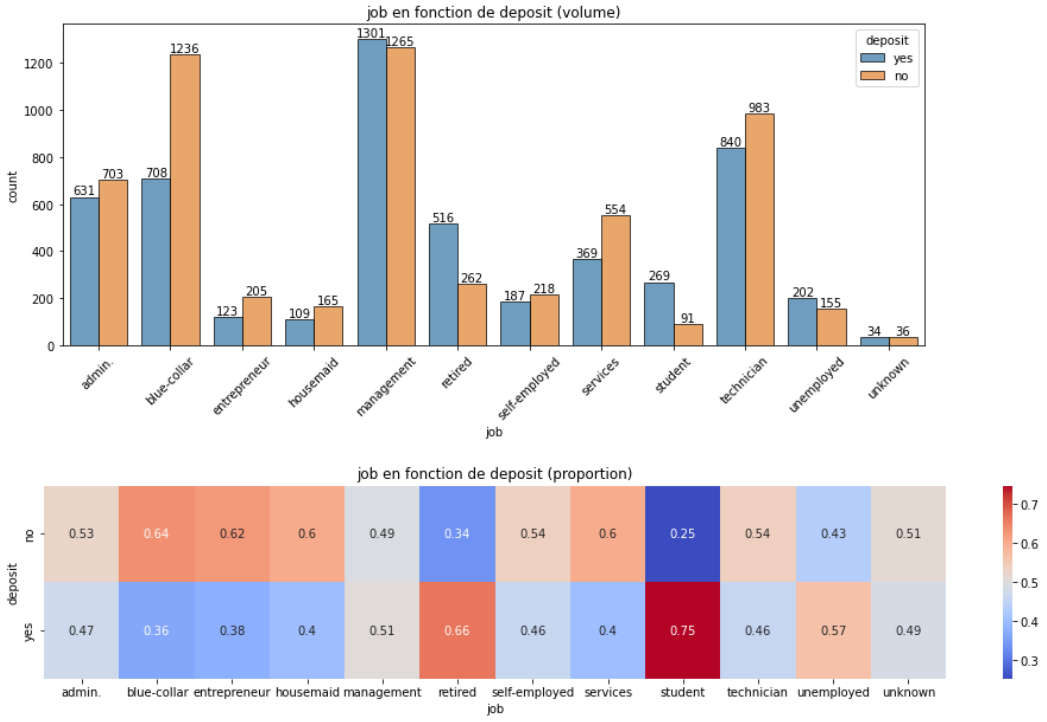
- *Analyse de la donnée 'job'*

Cette donnée représente le métier du client. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



Analyse de 'job' en fonction de 'deposit' :

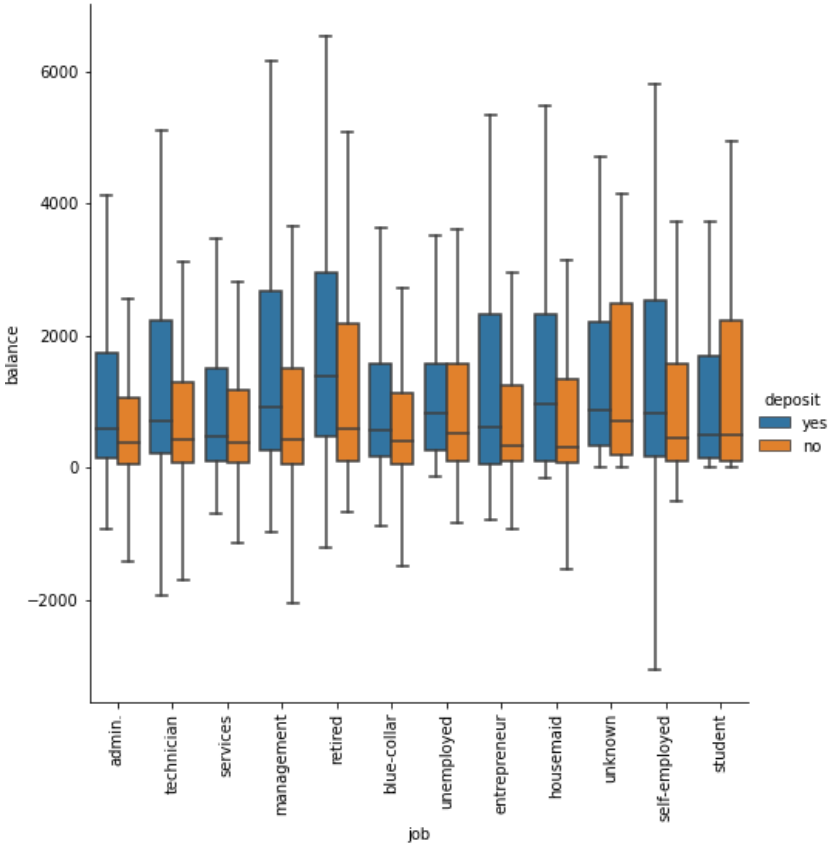


75% des étudiants souscrivent au DAT, suivi des retraitées avec 66%, il semble donc que les personnes qui ne font pas partie de la population active sont plus susceptibles de souscrire un DAT. On peut aussi noter que certains emplois comme col-bleu ou entrepreneur ont un résultat en deçà de la moyenne.

- *La balance en fonction du job*

Contrairement à ce qu'on pourrait penser, les personnes sans emploi ne sont pas forcément les plus pauvres. Les étudiants ne tombent pas à découvert.

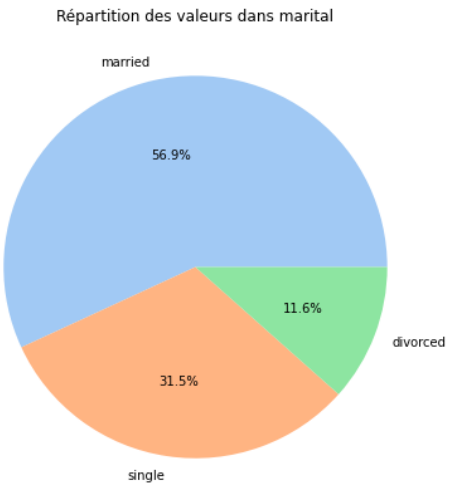
On voit bien que pour presque tous les métiers, le champ balance à un impact sur le deposit.



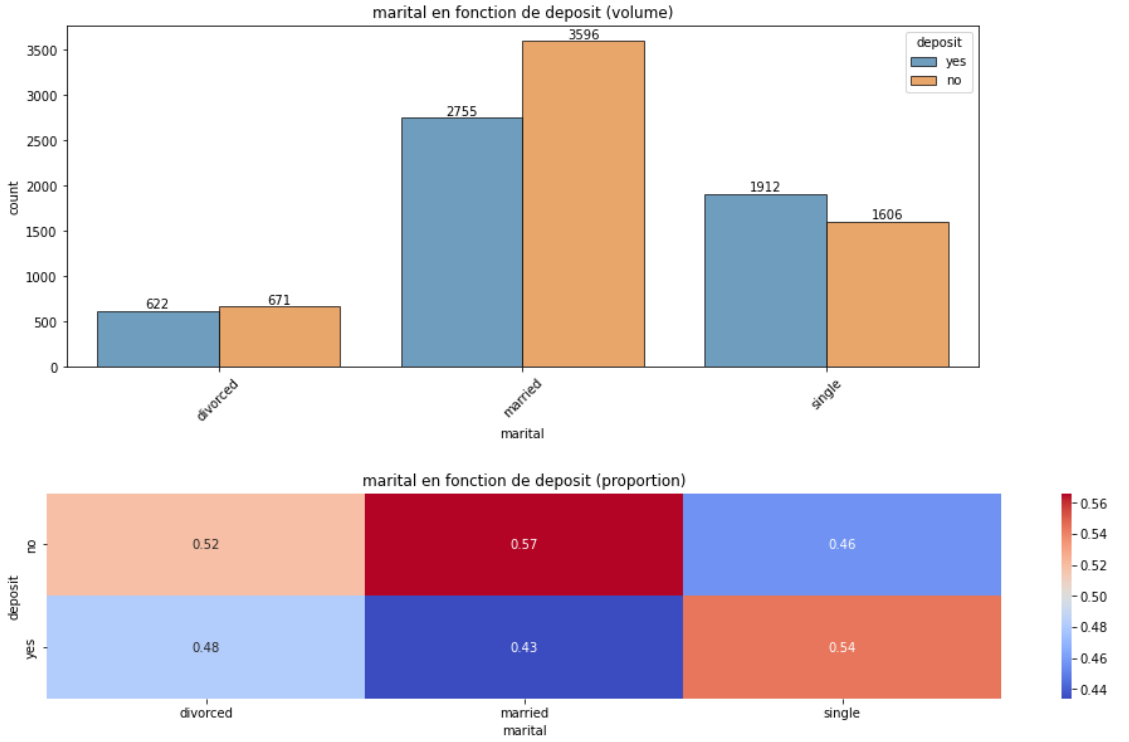
- *Analyse de la donnée 'marital'*

Cette donnée représente le statut marital du client (marié, célibataire, divorcé). Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target deposit.

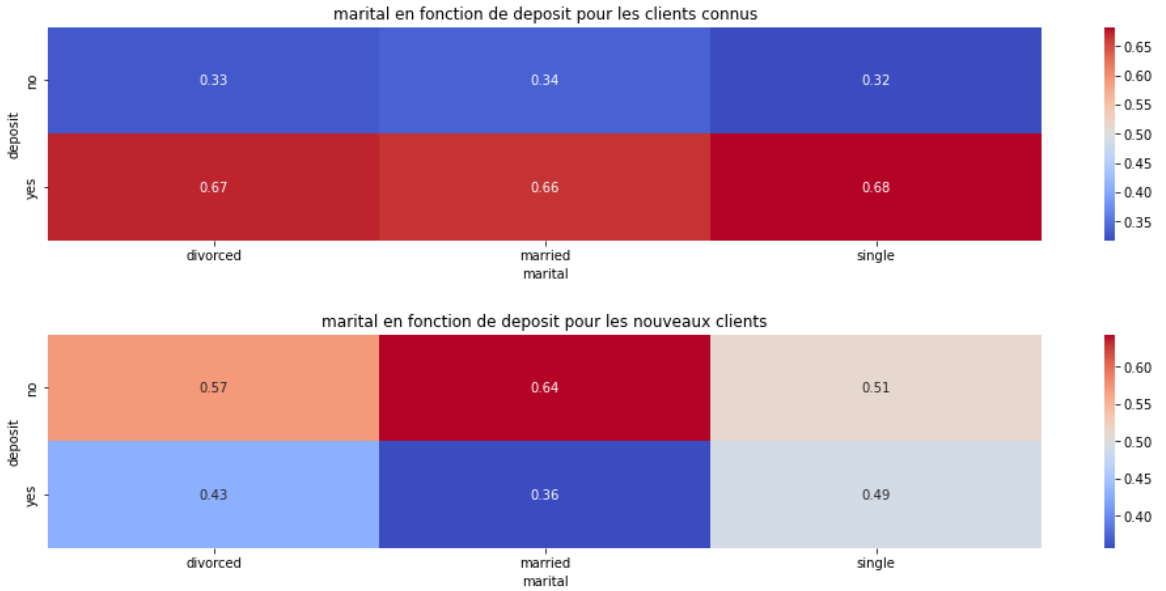
Analyse graphique :



Analyse de ‘marital’ en fonction de ‘deposit’ :



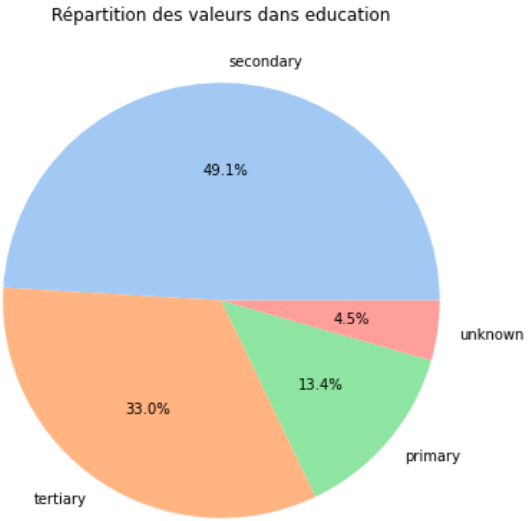
Le statut marital n'a pas d'impact pour les clients connus. Les scores sont similaires. En revanche, il y a des écarts lorsque ce sont de nouveaux clients. Les personnes mariées souscrivent moins. On peut imaginer que les personnes souscrivent moins lors d'un démarchage, ne souhaitant pas prendre une initiative sans l'avis du conjoint.



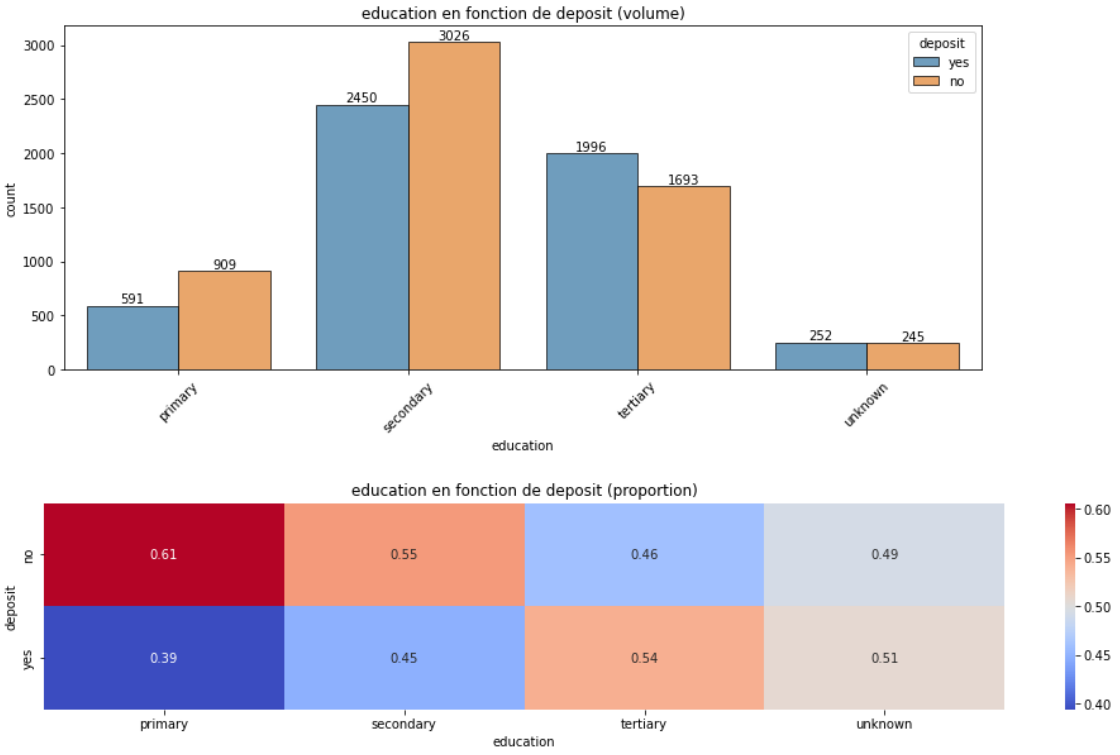
- *Analyse de la donnée 'education'*

Cette donnée représente le niveau d'étude du client. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



Analyse de 'education' en fonction de 'deposit':

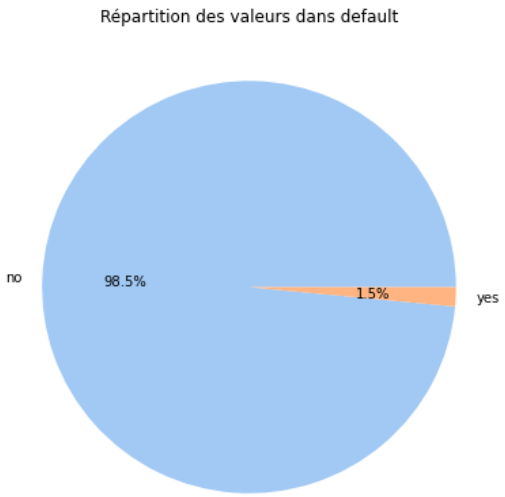


Nous constatons que plus le niveau d'éducation est élevé, plus on a de chance de souscrire un DAT.

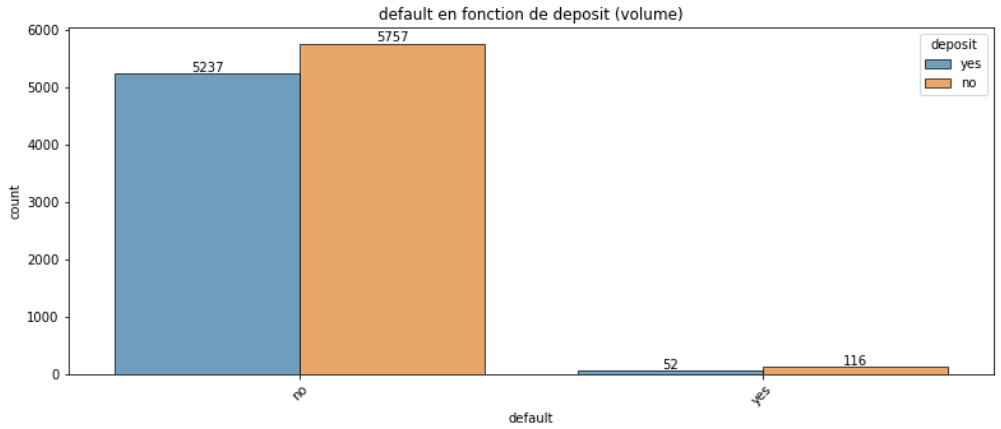
- *Analyse de la donnée 'default'*

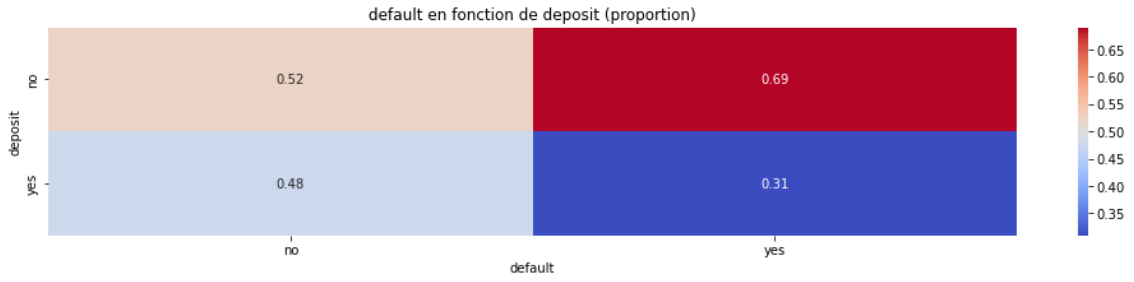
Cette donnée représente si le client a un crédit en défaut de paiement. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



Analyse de 'default' en fonction de 'deposit' :



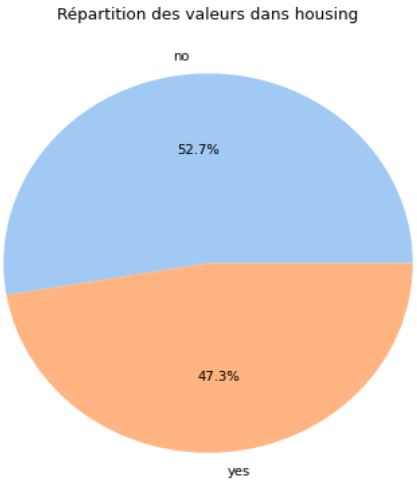


Les clients en défaut de paiement ont tendance à moins souscrire que les autres. En revanche, il y a très peu de ce type de client dans le jeu de données pour que cela soit pertinent.

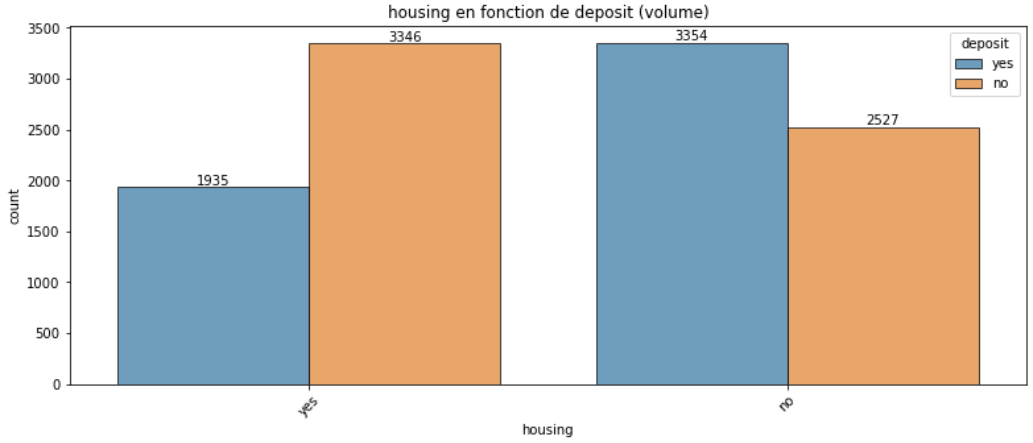
- *Analyse de la donnée 'housing'*

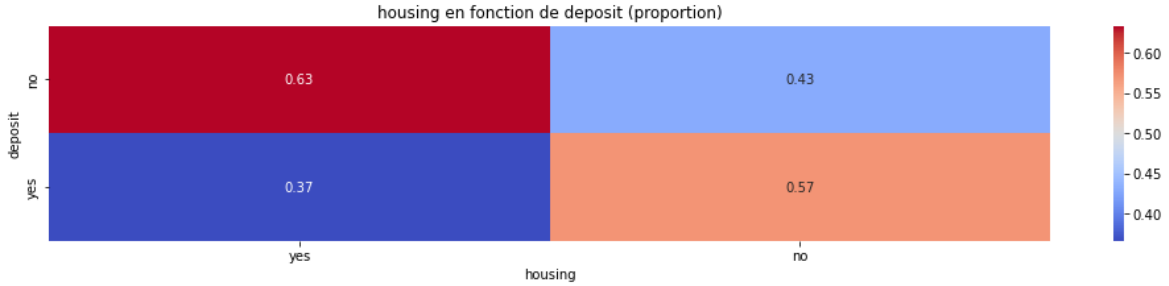
Cette donnée représente si le client a un crédit immobilier. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



Analyse de 'housing' en fonction de 'deposit' :



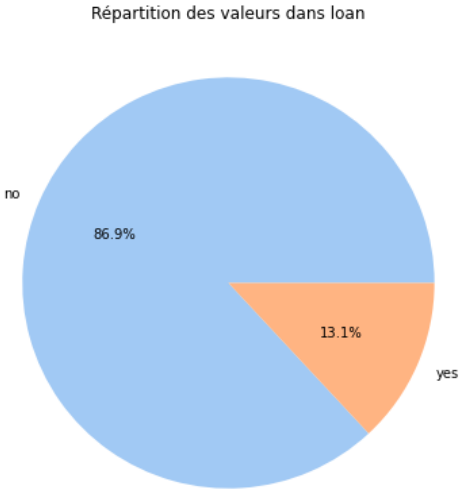


Les clients qui ont des obligations de remboursement de crédits immobiliers ont moins tendance à souscrire au dépôt à terme.

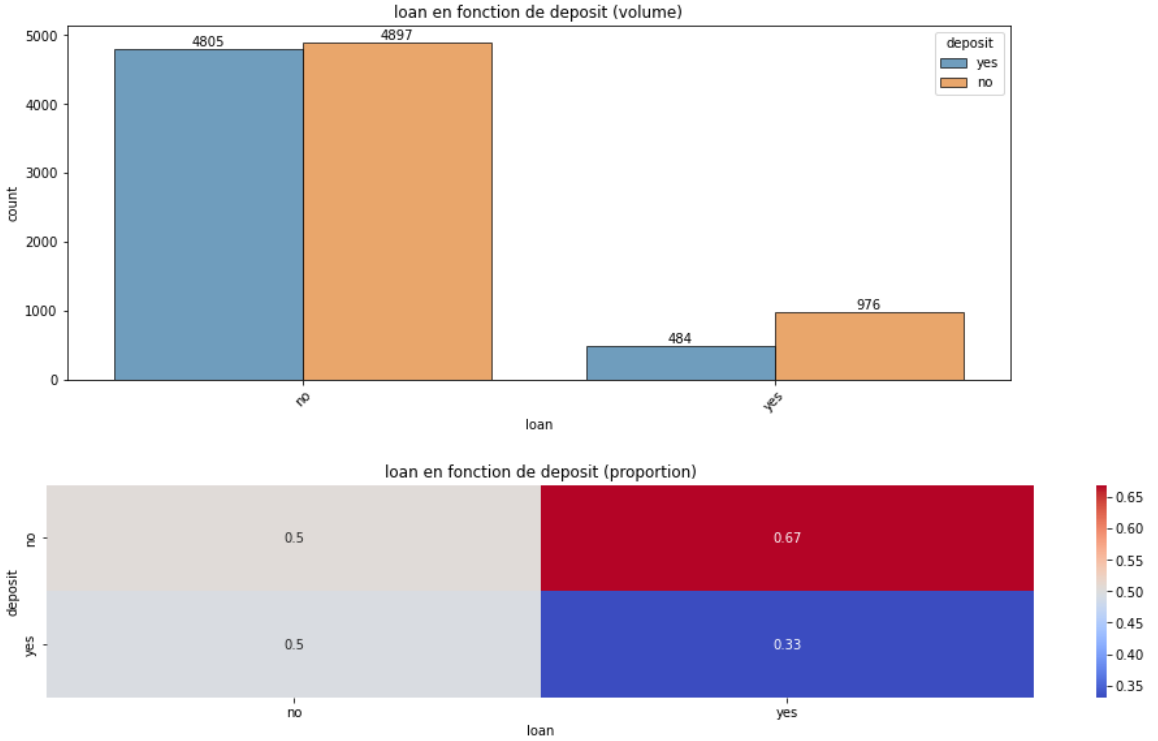
- *Analyse de la donnée 'loan'*

Cette donnée représente si le client a un crédit autre que immobilier en cours. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



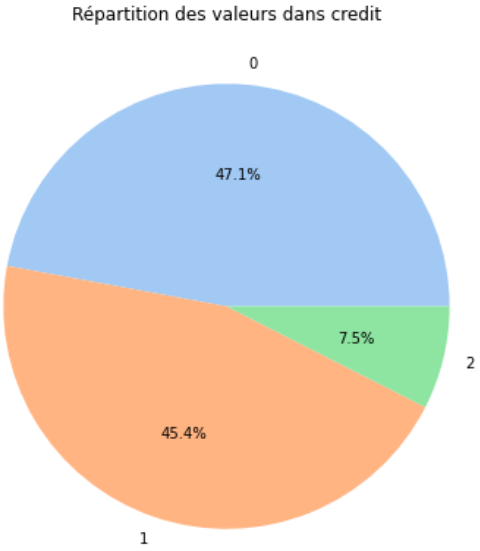
Analyse de 'loan' en fonction de 'deposit' :

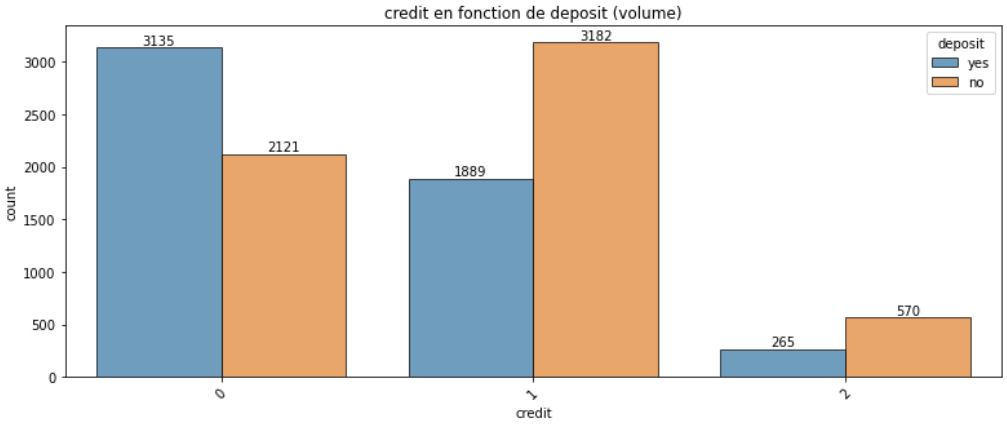


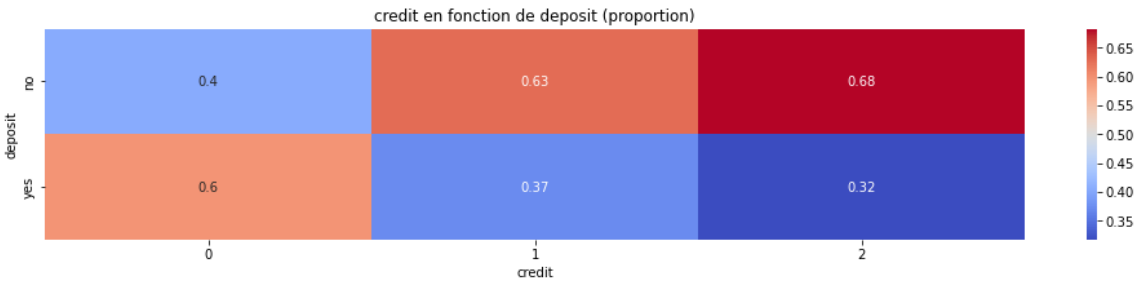
Les clients qui ont des obligations de remboursement de crédits ont moins tendance à souscrire au dépôt à terme.

- *Analyse des crédits ensemble*

Avoir deux crédits en cours réduit les chances de souscrire au DAT, mais l'écart n'est pas très significatif pour autant (avec un seul).







- *Nombre de crédit en fonction du job*

Nous nous demandons si confronter le nombre de crédits au métier du client pourrait nous donner des informations.

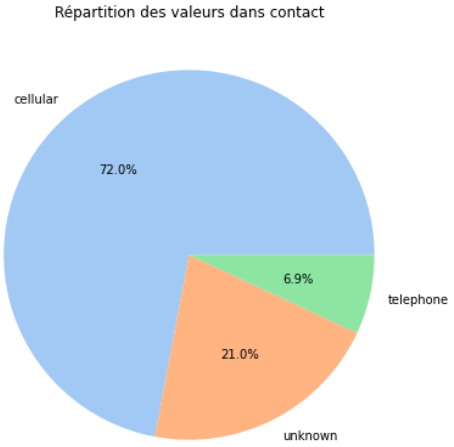
job credit	admin.	blue-collar	entrepreneur	housemaid	management	retired	self-employed	services	student	technician	unemployed
0	37.63	25.62	40.24	67.52	53.16	80.08	51.36	31.53	85.56	46.08	66.67
1	51.57	63.48	47.56	29.93	41.74	16.97	40.49	58.94	14.44	46.19	29.41
2	10.79	10.91	12.20	2.55	5.11	2.96	8.15	9.53	0.00	7.73	3.92

Il y a plus de 80% des retraités et des étudiants qui n'ont aucun crédit en cours. Cela explique en partie pourquoi c'est les deux job qui ont le meilleur taux de souscription.

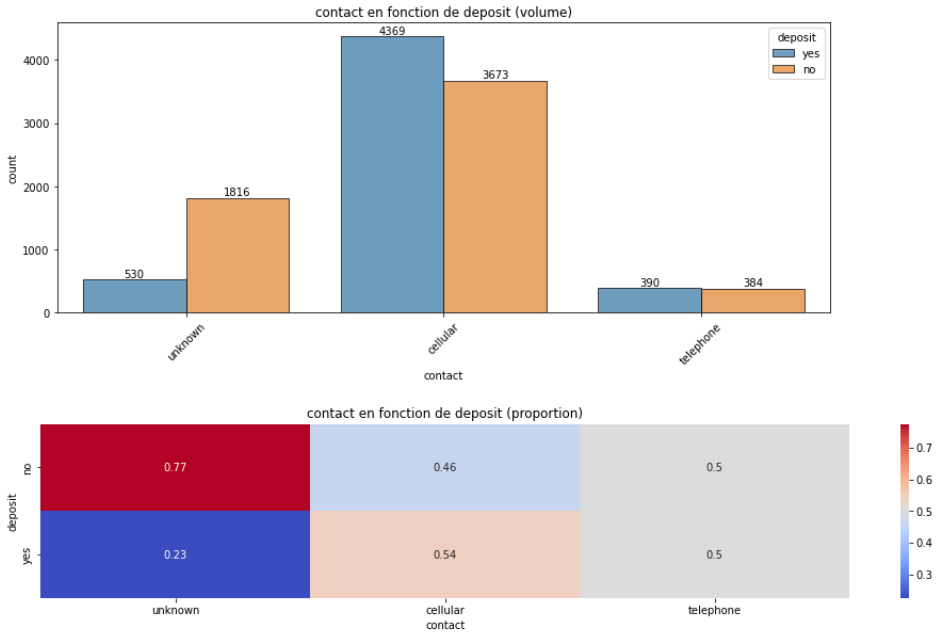
- *Analyse de la donnée 'contact'*

Cette donnée représente le moyen utilisé afin de contacter le client. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



Analyse de 'contact' en fonction de 'deposit' :



Les résultats entre cellular et téléphone sont très similaires. Il y a aussi peu de volume sur téléphone. Il pourrait donc être intéressant de combiner les deux valeurs.

Il y a beaucoup de valeurs “unknown” et nous ne savons pas vraiment à quoi cela peut correspondre. Lorsque l'on contacte un client, on sait si on a composé le numéro d'un portable ou d'un téléphone fixe. Il est donc assez étonnant de voir la valeur contact = ‘unknown’. Nous pourrions donc simplement l'ignorer.

Cependant, il y a un réel écart de résultat avec cellular et téléphone, indiquant ainsi que ce n'est pas anodin.

Dans un contexte professionnel, nous aurions contacté la banque pour comprendre le processus qui peut engendrer une valeur ‘unknown’ pour la colonne contact.

- *Analyse de la donnée ‘month’*

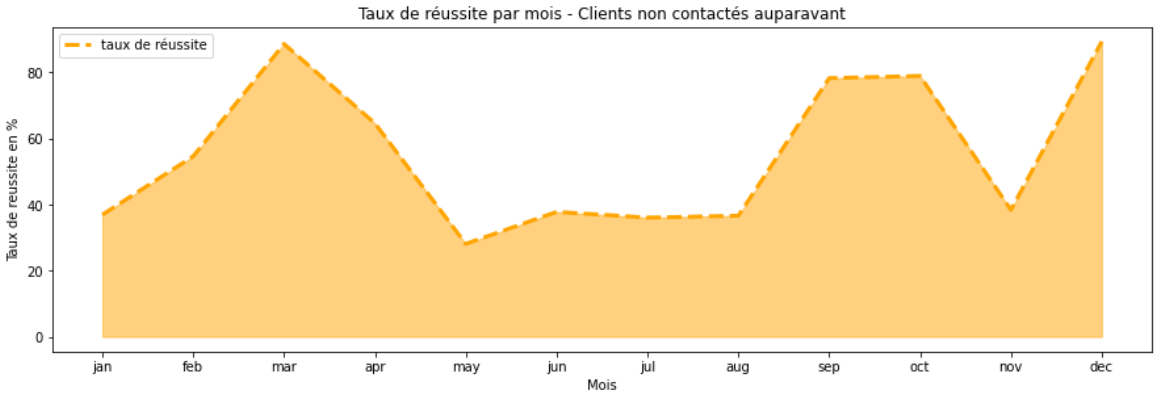
Cette donnée représente le mois auquel l’appel au client a été passé. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target ‘deposit’.

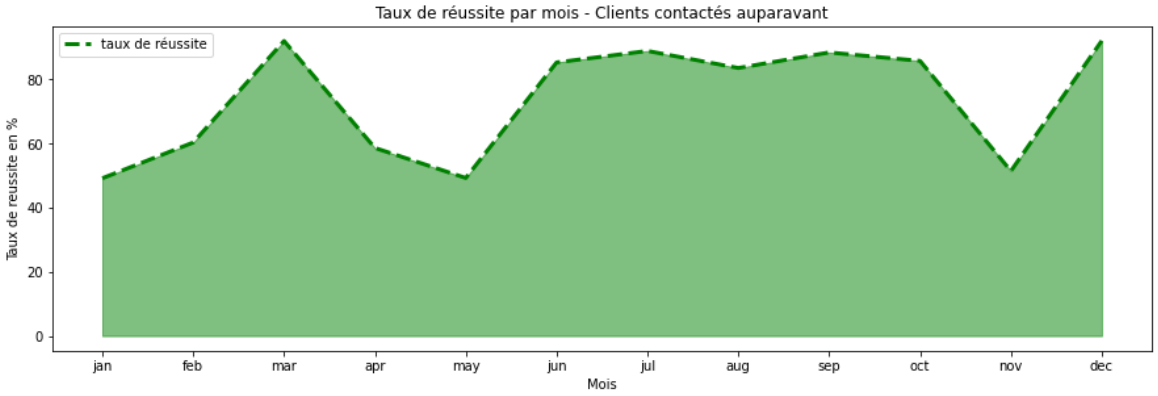
Analyse graphique :

deposit	no	yes	month	appel	tx_reussite_par_mois
0	202	142	jan	344	41.28
1	335	441	feb	776	56.83
2	28	248	mar	276	89.86
3	346	577	apr	923	62.51
4	1899	925	may	2824	32.75
5	676	546	jun	1222	44.68
6	887	627	jul	1514	41.41
7	831	688	aug	1519	45.29
8	50	269	sep	319	84.33
9	69	323	oct	392	82.40
10	540	403	nov	943	42.74
11	10	100	dec	110	90.91

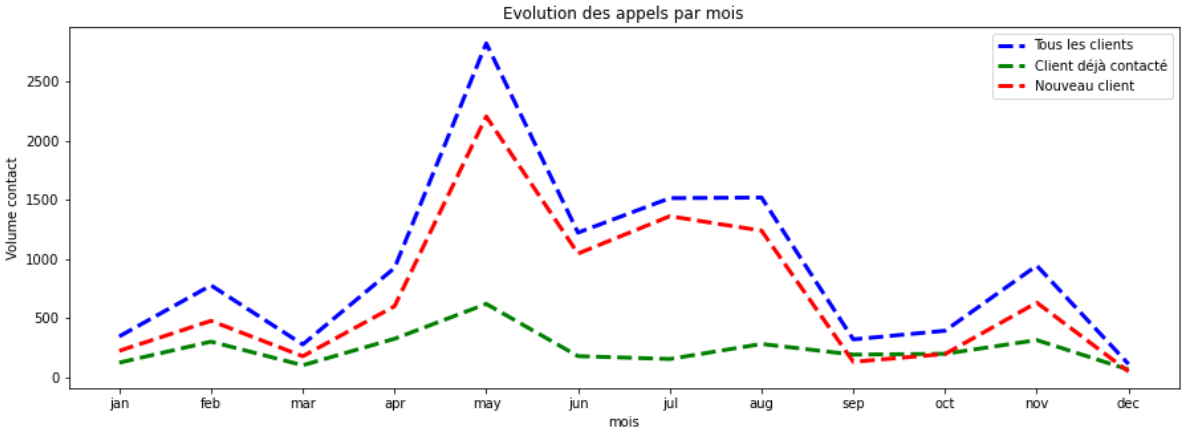
Analyse selon nombre contact

On voit que les mois les plus efficaces sont en mars, en septembre/octobre et décembre. La période estivale n'est pas la meilleure période pour appeler les nouveaux clients. Le mois de mai semble le pire. On va étudier les volumes d'appels par mois pour obtenir plus d'informations.





Evolution des appels par mois

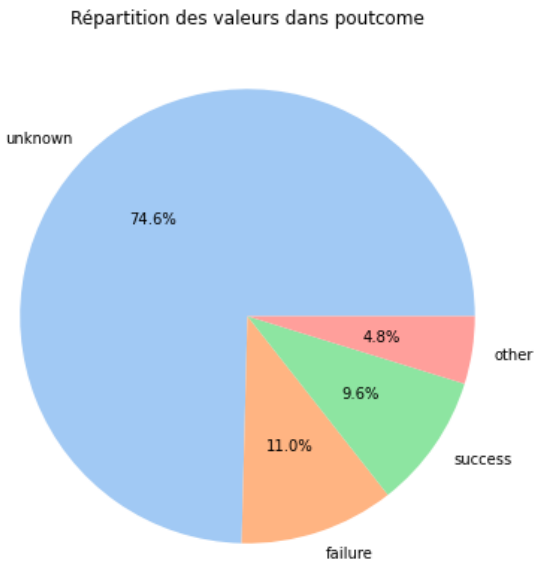


On note que le volume d'appels est inversement proportionnel aux résultats vu avant. Le mois de mai ainsi que la période estivale montrent un très fort volume d'appels, alors que nous avions un faible taux de succès. A l'inverse, nous avons peu d'appels sur les mois efficaces comme mars ou décembre. On peut imaginer qu'il y a eu des campagnes massives d'appels visant tous les clients lors des gros volumes, expliquant ainsi le faible taux de réussite. Pour les mois à faible volume, il n'y avait certainement pas de campagnes spécifiques, on a pu donc contacter des clients bien précis ayant de forte de chance de souscrire.

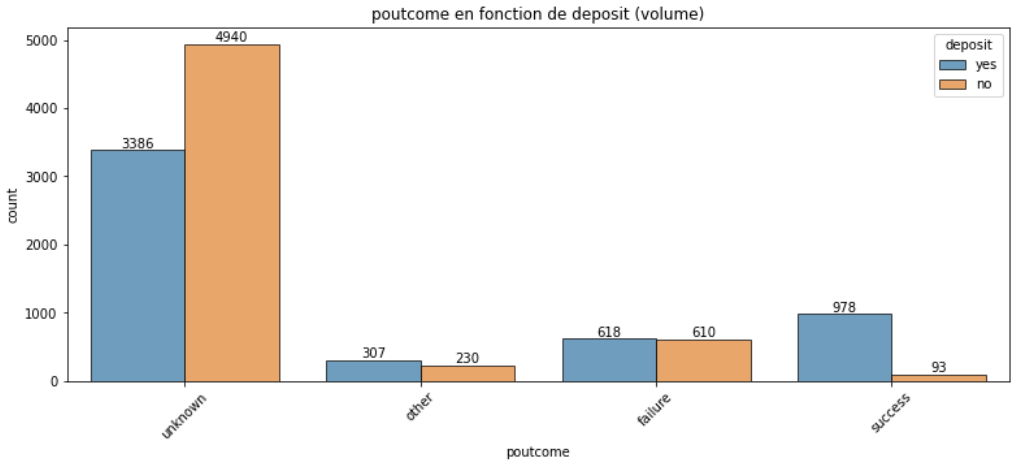
- *Analyse de la donnée 'poutcome'*

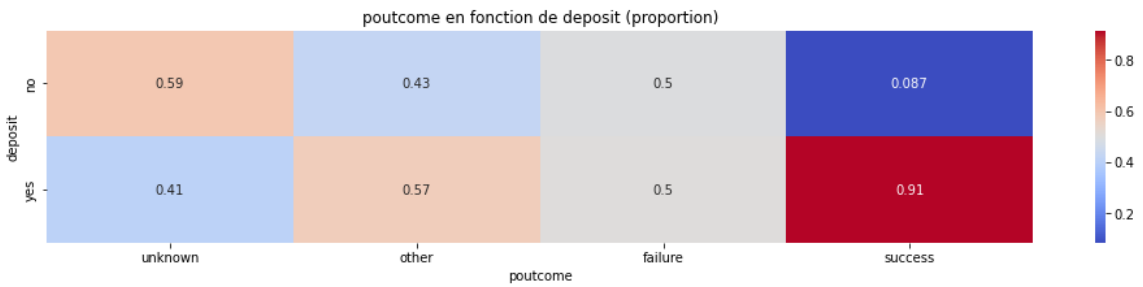
Cette donnée représente le résultat de la précédente campagne. Nous ferons dans un premier temps une analyse graphique de cette donnée puis nous croiserons celle-ci avec la valeur target 'deposit'.

Analyse graphique :



Analyse de contact en fonction de 'deposit' :





Il y a énormément d'unknown. Cela correspond à tous les nouveaux clients. C'est logique, nous ne pouvons pas avoir de résultat précédent, sur une personne que nous n'avons jamais appelée. Unknown correspond donc à previous = 0 ou pdays = -1

- “Success” correspond aux clients ayant dit oui lors d'une précédente campagne. La campagne actuelle connaît un franc succès pour ces clients (91%)

- “Failure” et “other” sont assez proches. “Failure” correspond à un refus du client lors de la précédente campagne. “Other” correspond à un client qui a fait un retour positif mais un autre service que le DAT. On pourra envisager de regrouper “other” et “faillure”, car les volumes sont très faibles.

Vérification de la logique entre client jamais contacté et les unknown de ‘poutcome’ :

poutcome	failure	other	success	unknown
has_previous				
False	0	0	0	8324
True	1228	537	1071	2

On constate que toutes les lignes sont cohérentes sauf 2 (unknown et previous > 0). Nous pourrons donc supprimer ces deux lignes lors de la phase de pré-processing. En effet, un client ayant déjà participé à une campagne précédente ne devrait pas avoir poutcome = unknown.

C. Conclusion data visualisation et hypothèses

En ce qui concerne la valeur cible 'deposit', nous retrouvons 47,4% de '**Yes**' et 52,6% de '**No**'.

La target est plutôt équilibrée. Nous ne constatons pas non plus de valeurs aberrantes.

Les retraités et les étudiants ont le taux d'acceptation le plus élevé. C'est certainement lié au fait qu'ils ont moins de crédit que les autres. On peut également souligner l'idée que les retraités préfèrent généralement les investissements sûrs sous forme de dépôts fixes plutôt que d'autres investissements risqués. Les étudiants quant à eux doivent souvent se préoccuper de leurs frais de subsistance et de scolarité.

En ce qui concerne les actifs, les clients dans le management sont les plus susceptibles d'accepter l'offre. En revanche les cols bleus, entrepreneurs, les employés de maisons ainsi que les employés dans le secteur du service semblent être moins réceptifs à cette offre. On peut supposer que ces emplois ne sont généralement pas très rémunérateurs et que ceux qui ont des emplois peu rémunérés sont moins susceptibles d'avoir des liquidités.

Nous voyons également qu'un niveau d'étude élevé vient augmenter la probabilité d'acceptation du contrat. Ceci pourrait s'expliquer par une meilleure compréhension des contrats financiers ainsi que par le fait qu'un meilleur niveau d'étude signifie souvent une plus haute rémunération.

En ce qui concerne le profil financier, la grande majorité des clients ayant accepté l'offre ne sont pas en défaut de paiement et ont une balance positive. On remarque également que plus la balance est élevée et plus on est enclin à accepter l'offre. Le fait d'avoir contracter un prêt immobilier ou une autre catégorie de prêt semble avoir un effet négatif sur le taux de signature du contrat.

Du côté du statut marital, lorsque le client est déjà un client de la banque le statut ne semble pas jouer un rôle important dans l'acceptation de l'offre. En revanche, lorsque nous nous focalisons sur les nouveaux clients nous pouvons observer que ce sont les personnes mariées qui acceptent moins facilement l'offre.

Méthode contact

Le jour du contact ne semble pas avoir un impact significatif sur l'acceptation du contrat.

En revanche, la durée de l'appel ainsi que le mois pourraient quant à eux influencer la réponse à l'offre. Les mois de décembre, mars, septembre et octobre ayant un taux de conversion plus élevé que les autres mois. De plus, plus l'appel sera long et plus le taux de réussite sera élevé.

Nous notons également que le nombre de relance ne devrait pas excéder 3 appels. Car au-delà de ce seuil le taux de signature diminue fortement.

Nous constatons également que les personnes qui n'ont pas été contactées précédemment ont le taux de signature le plus bas. A contrario, la réussite antérieure d'une campagne de marketing se traduit par un taux de conversion plus élevé.

II. Pré-traitement et feature engineering

A. Définition

La première étape dans le développement d'un modèle de Machine Learning prédictif est le Feature Engineering, il permet d'accroître l'exactitude du modèle sur les nouvelles données inconnues et a pour but l'extraction des caractéristiques des données brutes afin de résoudre des problèmes spécifiques à un domaine d'activité grâce au Machine Learning

Rappel : Qu'est qu'une feature ? :

Les données sont présentées en ligne dans des tableaux, tandis que leurs attributs et leurs variables sont présentés en colonnes. Un attribut peut être une caractéristique.

Les approches que l'on va utiliser pour sélectionner nos features seront :

- La "Feature Selection" ou sélection de caractéristiques est une autre méthode, permettant de supprimer les attributs des données inutiles ou redondantes dans le contexte du problème à résoudre.
- Coefficients de corrélation pour mesurer l'importance d'une caractéristique. Certains algorithmes de modélisation prédictive plus complexes effectuent cette sélection de manière interne parallèlement à la construction du modèle.

La stratégie afin d'optimiser nos modèles sera de faire des "allers-retours" entre le feature engineering et la partie "Machine Learning".

B. Feature Selection

Comme nous avons pu le voir lors de notre analyse exploratoire, les features 'duration' et 'campaign' ne seront pas utilisées pour nos modèles de "Machine Learning". En effet, pour la variable 'duration', la durée n'est connue qu'une fois que l'appel est effectué, or, nous souhaitons justement prédire si cela est pertinent ou non de le contacter. De la même façon, pour la variable 'campaign', ce champ ne sera pas utilisé, car nous ne connaissons pas cette donnée tant que la campagne n'est pas terminée.

```
# Suppresion de 'duration' & 'campaign'
df_1 = df_1.drop('duration', axis = 1)
df_1 = df_1.drop('campaign', axis = 1)

# Suppresion des 2 lignes incohérentes :
df_1 = df_1[(df_1['poutcome'] != 'unknown') | (df_1['previous'] == 0)]
```

C. Tests de corrélation

En Data science, il est primordial de découvrir et quantifier à quel point deux variables sont liées. Ces relations peuvent être complexes et ne sont pas forcément visibles. Or certaines de ces dépendances affaiblissent les performances d'algorithme de Machine Learning comme des régressions linéaires. Il devient alors impératif de mieux préparer nos données.

Un test statistique est une procédure de décision entre deux hypothèses. Il s'agit d'une démarche consistant à rejeter ou à ne pas rejeter une hypothèse statistique, appelée hypothèse nulle H0, en fonction d'un jeu de données.

La stratégie est simple, dans un premier temps il sera question d'étudier la corrélation entre les features (numériques et/ou catégoriques) avec la variable target 'deposit' et ensuite d'étudier les features entre elles.

- Corrélation des variables numériques avec la variable cible 'deposit'
- Corrélation des variables catégoriques avec la variable cible 'deposit'
- Corrélation des variables catégoriques entre elles
- Corrélation des variables numériques entre elles
- Corrélation entre variables numériques et catégoriques

Tests corrélation avec la valeur cible 'deposit'

a. Analyse des variables numériques avec la variable cible à l'aide du test ANOVA.

Une ANOVA (analyse de variance) est utilisée pour déterminer si oui ou non les moyennes des groupes indépendants sont égales.

Une ANOVA utilise les hypothèses nulles et alternatives suivantes :

- H0 : Toutes les moyennes de groupe sont égales.
- H1 : Au moins une moyenne de groupe est différente des autres.

Chaque fois que l'on effectue une ANOVA, on obtient un tableau récapitulatif qui ressemble à ceci :

La source	Somme des carrés (SS)	df	Carrés moyens (MS)	F	Valeur P
Traitement	192.2	2	96.1	2.358	0,1138
Erreur	1100.6	27	40,8		
Total	1292.8	29			

Les deux valeurs que nous analysons immédiatement dans le tableau sont la statistique F et la valeur p correspondante .

- *Comprendre la statistique F dans ANOVA*

La statistique F produite par l'ANOVA est le rapport entre la variabilité inter et intra-groupes. Elle permet de déterminer s'il existe une différence significative entre les groupes. Comme la variabilité inter-groupes est le numérateur de ce rapport, plus les moyennes sont éloignées les unes des autres, plus la valeur F est élevée.

- *Comprendre la valeur P dans ANOVA*

Si cette valeur de p est inférieure à $\alpha = 0,05$, nous rejetons l'hypothèse nulle de l'ANOVA et concluons qu'il existe une différence statistiquement significative entre les moyennes des groupes. Sinon, si la valeur de p n'est pas inférieure à $\alpha = 0,05$, nous ne rejetons pas l'hypothèse nulle et concluons que nous n'avons pas suffisamment de preuves pour dire qu'il existe une différence statistiquement significative entre les moyennes des trois groupes.

Dans notre cas, les p-values sont ici très largement inférieures à 0,05, on peut en conclure que chacune des variables numériques représente un poids sur la variable cible 'deposit', nous rejetons donc l'hypothèse H0 selon laquelle, il y a indépendance avec la target.

	F	PR(>F)
age	13.587852	2.290000e-04
balance	73.921610	9.212031e-18
days	35.584154	2.516763e-09
pdays	261.977062	2.925198e-58
previous	222.509262	7.740962e-50

b. Analyse des variables catégorielles à l'aide du test de khi-deux.

Pour savoir si deux variables catégorielles sont liées, on utilise le test du khi-deux. Dans le test qui nous intéresse l'hypothèse nulle H0 est simplement « les deux variables testées sont indépendantes ». Enfin le test s'accompagne d'une statistique de test qui participe à la décision de rejeter ou non H0. Cette statistique de la manière dont le test est construit, a le bon goût de suivre une loi de khi-deux avec un certain degré de liberté. Mais comment décider de rejeter ou non l'hypothèse nulle ?

Chaque test statistique dispose de ce qu'on appelle la p-value. On peut la voir comme une valeur référence pour décider du rejet ou non de l'hypothèse nulle. Si cette dernière est en-dessous de 5% alors on rejette l'hypothèse nulle.

Enfin on peut également mesurer le niveau de corrélation entre les deux variables en utilisant le V de Cramer. Il se calcule à l'aide de la statistique de test (Chi2), du degré de liberté (DoF) et des dimensions de la table de contingence. Il renvoie une valeur entre 0 et 1. Si la valeur renvoyée est supérieure à 0.9 on peut qualifier la relation de très forte. Si la valeur est inférieure à 0.10 on peut qualifier la relation de faible.
Le tableau ci-dessous nous donne les informations suivantes :

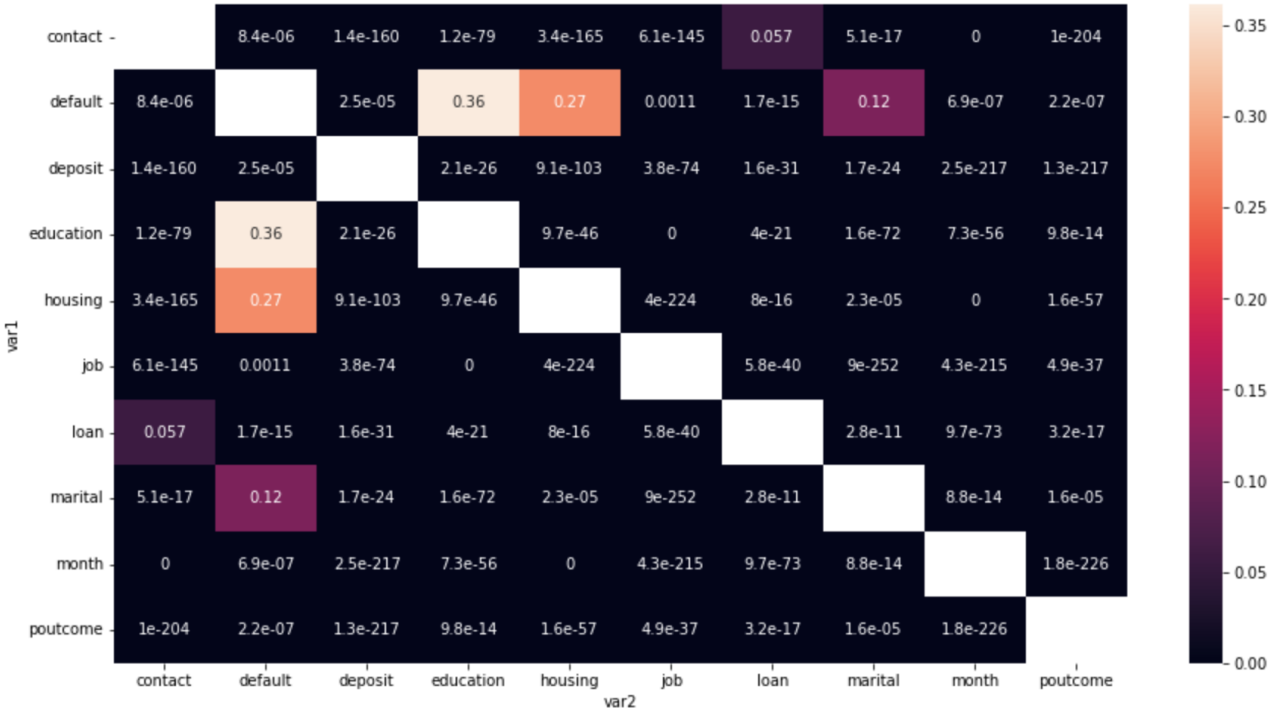
- statistique du test,
- p-value,
- degré de liberté,
- V de Cramer (coefficient de corrélation du $\chi 2$)

Il représente la corrélation entre les variables catégoriques et la valeur cible 'deposit'

	chi 2	p-value	DoF	V de Cramer
job	377.431711	3.753624e-74	11.0	0.181210
marital	109.486520	1.679981e-24	2.0	0.098144
education	122.621316	2.103014e-26	3.0	0.103536
default	17.788165	2.469144e-05	1.0	0.038787
housing	463.328156	9.070532e-103	1.0	0.203546
loan	136.427638	1.608706e-31	1.0	0.110164
contact	736.153834	1.400316e-160	2.0	0.256496
month	1045.899815	2.533382e-217	11.0	0.304534
poutcome	1005.259133	1.300958e-217	3.0	0.299693

Les p-values sont toutes sous la barre des 5%, on rejette donc l'hypothèse nulle H0. De même, le V de Cramer présente des valeurs insuffisantes pour la plupart des variables, on peut en conclure qu'il est difficile de mettre en évidence des features indispensables.

c. Corrélation entre les variables catégoriques à l'aide du test de khi-deux.



Ci-dessus, la heatmap qui nous décrit les corrélations entre variables catégoriques à travers la p-value. Pour rappel, si cette dernière est en-dessous de 5% alors on rejette l'hypothèse nulle (« les deux variables testées sont indépendantes »)

Ici p-value > 0,05 pour les couples de variables suivants :

- 'education' / 'default'
- 'housing' / 'default'
- 'loan' / 'contact'
- 'marital' / 'default'

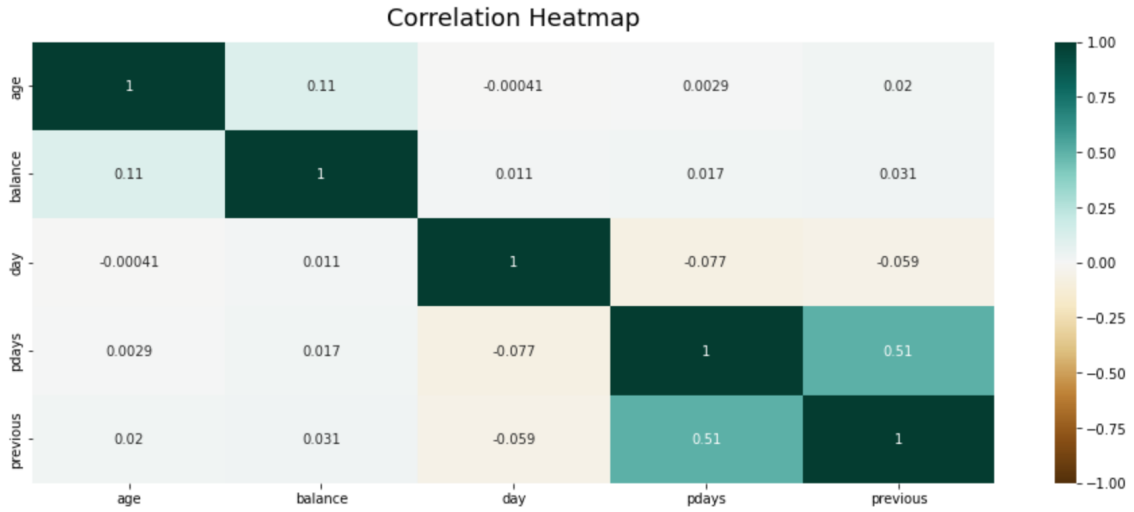
Ces variables sont donc indépendantes l'une de l'autre.

d. Corrélation entre les variables numériques à l'aide du test de Pearson

Le coefficient de Pearson est un indice reflétant une relation linéaire entre deux variables continues. Le coefficient de corrélation varie entre -1 et +1, 0 reflétant une relation nulle entre les deux variables, une valeur négative (corrélation négative) signifiant que lorsqu'une des variable augmente, l'autre diminue ; tandis qu'une valeur positive (corrélation positive) indique que les deux variables varient ensemble dans le même sens.

La valeur de r obtenue est une estimation de la corrélation entre deux variables continues dans la population. Dès lors, sa valeur fluctue d'un échantillon à l'autre. On veut donc savoir si, dans la population, ces deux variables sont réellement corrélées ou pas. On doit donc réaliser un test d'hypothèse.

H0: Pas de corrélation entre les deux variables : $\rho = 0$
HA: Corrélation entre les deux variables : $\rho \neq 0$
Faisons apparaître la heatmap des corrélations entre les valeurs numériques :



- ‘pdays’ et ‘previous’ présentent une corrélation avec un coefficient r = 0,51

Pour rappel, 'pdays' représente le nombre de jour depuis le dernier contact et previous le nombre de contact total lors des campagnes précédentes.

Cette corrélation avait déjà été mise en évidence graphiquement en effet lorsque ‘previous’ est supérieur à 0, on constate que les valeurs sont similaires. Cela veut dire que le nombre de contact auprès du client n'a pas d'importance. Ce qui compte, c'est de savoir si oui ou non il a été contacté au moins une fois.

e. Corrélation entre les variables numériques à l’aide du test de Kendall

Le tau de Kendall est une statistique qui mesure l’association entre deux variables. Plus spécifiquement, le tau de Kendall mesure la corrélation de rang entre deux variables. Le tau s'interprète de la même façon que les autres coefficients de corrélation. Sa valeur est comprise entre -1 et 1 : s’il s’approche de 1, on peut supposer l’existence d’une corrélation positive (variation dans le même sens), s’il tend vers -1, on peut dire qu’il existe une corrélation négative et si le tau est proche de 0, il est fort probable qu’il n’y ait aucune liaison entre les 2 variables.

- ‘job’ / ‘balance’

```
Entrée [217]: corr, _ = kendalltau(df_1['balance'], df_1['job'])
              print('Coefficient de corrélation Kendall : %.5f' % corr)
Coefficient de corrélation Kendall : 0.03460
```

- ‘day’/‘month’

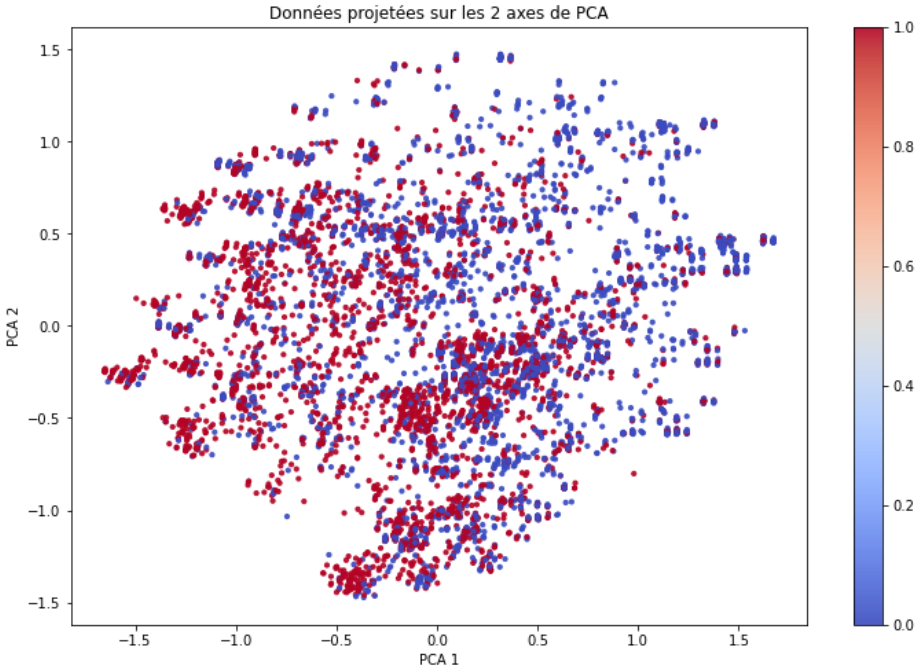
```
Entrée [220]: corr, _ = kendalltau(df_1['day'], df_1['month'])
              print('Coefficient de corrélation Kendall : %.5f' % corr)
Coefficient de corrélation Kendall : -0.00250
```

Les coefficients étant proches de 0, nous sommes tentés de conclure qu’il y a de fortes chances pour qu’il n’y ait aucune dépendance entre les couples de variables.

D. Analyse des composantes principales sur l'ensemble des données

Pour terminer notre première partie, nous souhaitons effectuer une analyse des composantes principales sur nos données pour avoir une meilleure idée de nos données. En effet, l'ACP permet de transformer des variables qui sont corrélées en variables dé-corrélées appelées Composantes principales. Cette analyse permet de réduire le nombre de variables du jeu de données pour simplifier les observations tout en conservant un maximum d'informations.

Pour cela, il est nécessaire de n'avoir que des variables numériques. Nous transformons donc les variables catégorielles en variables indicatrices. Nos données ne sont pas sur la même base, on met tous les champs à la même échelle en centrant et réduisant les données (MinMaxScaler).



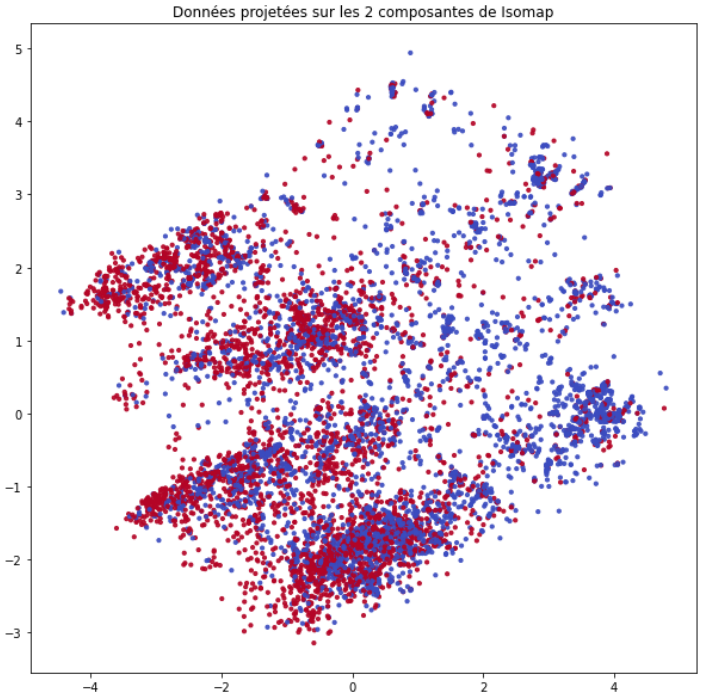

```
La part de variance expliquée est 0.24
```

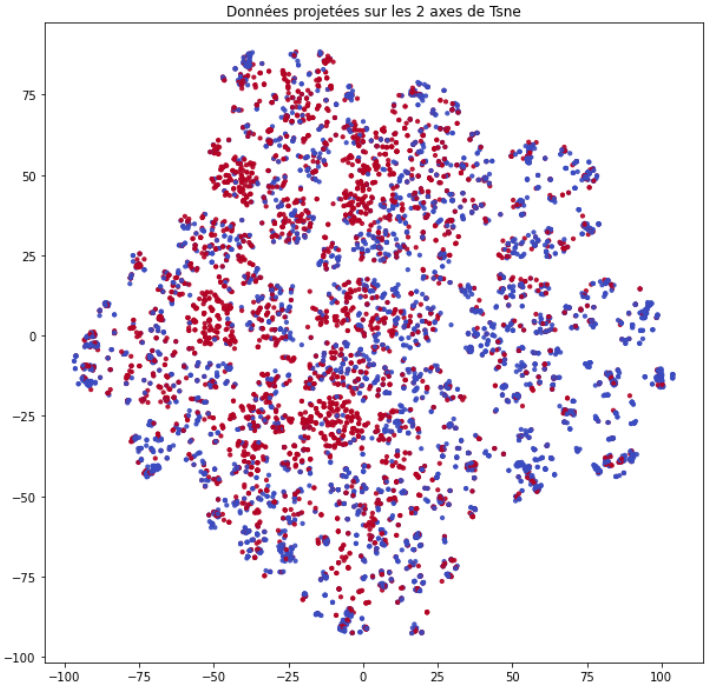
En projetant les 2 premières composantes, nous atteignons 24% d'inertie expliquée. Les données sont très mélangées et il est vraiment difficile de distinguer des zones départageant le deposit.

On peut tout de même voir que les points rouges sont majoritaires en bas à gauche et les points bleus le sont en haut à droite du graphique.

E. Autre méthode de réduction de dimension

Il existe beaucoup d'autres méthodes de réduction de dimension que l'ACP. Nous avons également explorer deux autres méthodes pour voir si nous obtenions des résultats qui partageraient mieux le dataset.





On a testé deux autres méthodes : Isomap (à gauche) et TNSE (à droite). Nous parvenons au même constat que pour le PCA. Il y a bien certaines zones regroupant uniquement ou majoritairement du rouge. Mais on a également énormément de zones où les points rouge et bleu sont mélangés.

Ce que montre ces 3 graphiques, c'est qu'il sera très difficile de départager l'ensemble des données. Nous ne sommes pas sur un sujet simple et nous ne pourrons donc pas espérer de résultat avoisinant les 100%.

III. Création de modèles “simples”

Maintenant que nous avons une bonne compréhension du dataset, on peut commencer la création d’un modèle de machine learning pour répondre à notre problématique. Il en existe une multitude ayant chacun des avantages et des inconvénients. Il est difficile de dire avec exactitude quel sera le modèle le plus efficace pour prédire les clients qui vont souscrire à un dépôt à terme.

La stratégie choisie ici est donc, dans un premier temps, de lancer plusieurs algorithmes et observer ceux qui ont les meilleurs résultats. Nous pourrons ainsi sélectionner le meilleur modèle et chercher ensuite à l’optimiser. Nous devons entamer avant tout une phase de pré-processing simple pour que les modèles puissent tourner correctement.

A. Pre-processing simple

a. Train test split

Jusqu’à maintenant, nous avons travaillé sur le dataset complet. Cependant, dans le cadre de la création d’un modèle, nous devons utiliser une partie des données et garder la seconde pour nous assurer que notre prédiction fonctionne.

On appelle cela le train test split. Nous allons découper le dataset de la façon suivante :

- 80% des données iront dans le trainset. Le trainset va servir à entraîner le modèle, le paramétrer. Lors de son entraînement, le modèle ne verra que les données du trainset.
- 20% des données iront dans le testset. Le testset sera mis complètement de côté dans ce projet. Comme notre modèle n’aura jamais vu les données du test set, on pourra donc vérifier si nos prédictions fonctionnent correctement sur des cas concrets.

Afin de pouvoir répliquer avec exactitude le découpage du dataset, il existe un paramètre random_state, qui permet d’avoir un résultat aléatoire tout en étant reproductible. Dans ce projet, nous utiliserons toujours le paramètre random_state = 0, que ce soit pour le train test split et pour les modèles de machine learning.

```
# on découpe le dataframe en 2, un trainset et un testset
trainset, testset = train_test_split(df, test_size=0.2, random_state=0)
```

Nous allons maintenant pouvoir commencer le pré-processing sur le trainset.

b. Encodage des valeurs binaires

L’encodage des valeurs binaires est simple. Il consiste à transformer chaque variable ayant uniquement 2 valeurs possibles, en 0 et 1. Dans ce dataset, ‘no’ sera égale à 0, tandis que ‘yes’ sera égal à 1. Il y a 4 colonnes concernées. Voici un avant/après :

	default	housing	loan	deposit
6978	no	no	no	no
1149	no	no	no	yes
6316	no	yes	yes	no
2457	no	yes	no	yes
6610	no	yes	no	no

	default	housing	loan	deposit
6978	0	0	0	0
1149	0	0	0	1
6316	0	1	1	0
2457	0	1	0	1
6610	0	1	0	0

c. Création de dummy variables pour les colonnes catégoriques

En ce qui concerne les variables catégoriques comprenant plus de deux sorties, c'est un peu plus compliqué que pour l'encodage de données binaire.

Ici, nous allons créer automatiquement une nouvelle variable binaire, pour chacune des valeurs disponibles. Voici un exemple avec la colonne poutcome (avant / après) ::

	poutcome
6978	failure
1149	unknown
6316	unknown
2457	failure
6610	unknown

	poutcome_failure	poutcome_other	poutcome_success	poutcome_unknown
6978	1	0	0	0
1149	0	0	0	1
6316	0	0	0	1
2457	1	0	0	0
6610	0	0	0	1

d. Features scaling des données numériques

Les données numériques peuvent déjà être lues par un modèle de machine learning sans être modifié. Cependant, les données ne sont pas toujours à la même échelle, et cela peut avoir un impact très négatif lors de l'entraînement du modèle. Typiquement, l'échelle de l'âge qui va de 18 à 95 ans, n'est pas du tout la même que celle de balance (-6847 jusqu'à 66653 sur le trainset).

Il existe plusieurs méthodes pour mettre les données numériques sur une même échelle. Nous utilisons le StandardScaler pour le moment.

- Avant :

	age	balance	day	pdays	previous
6978	48	430	14	371	8
1149	59	514	3	-1	0
6316	33	3071	21	-1	0
2457	42	3403	7	99	1
6610	36	191	25	-1	0

- Après :

	age	balance	day	pdays	previous
6978	0.562733	-0.356728	-0.196336	2.968931	2.992357
1149	1.484812	-0.328908	-1.504191	-0.479689	-0.355535
6316	-0.694649	0.517945	0.635936	-0.479689	-0.355535
2457	0.059780	0.627900	-1.028607	0.447359	0.062951
6610	-0.443172	-0.435882	1.111520	-0.479689	-0.355535

Les valeurs sont maintenant sur une échelle similaire.

e. Information du trainset

Voici les informations du trainset avant de faire du pre-processing :

```
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        8928 non-null   int64
 1   job        8928 non-null   object
 2   marital    8928 non-null   object
 3   education  8928 non-null   object
 4   default    8928 non-null   object
 5   balance    8928 non-null   int64
 6   housing    8928 non-null   object
 7   loan       8928 non-null   object
 8   contact    8928 non-null   object
 9   day        8928 non-null   int64
 10  month      8928 non-null   object
 11  pdays      8928 non-null   int64
 12  previous   8928 non-null   int64
 13  poutcome   8928 non-null   object
 14  deposit    8928 non-null   object
dtypes: int64(5), object(10)
```

Voici la liste des features après le preprocessing :

```
Data columns (total 46 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  8928 non-null   float64
 1   default              8928 non-null   uint8
 2   balance              8928 non-null   float64
 3   housing              8928 non-null   uint8
 4   loan                 8928 non-null   uint8
 5   day                  8928 non-null   float64
 6   pdays                8928 non-null   float64
 7   previous             8928 non-null   float64
 8   job_admin.           8928 non-null   uint8
 9   job_blue-collar      8928 non-null   uint8
 10  job_entrepreneur     8928 non-null   uint8
 11  job_housemaid        8928 non-null   uint8
 12  job_management       8928 non-null   uint8
 13  job_retired          8928 non-null   uint8
 14  job_self-employed    8928 non-null   uint8
 15  job_services         8928 non-null   uint8
 16  job_student          8928 non-null   uint8
 17  job_technician       8928 non-null   uint8
 18  job_unemployed       8928 non-null   uint8
 19  job_unknown          8928 non-null   uint8
 20  marital_divorced     8928 non-null   uint8
 21  marital_married      8928 non-null   uint8
 22  marital_single       8928 non-null   uint8
 23  education_primary    8928 non-null   uint8
 24  education_secondary  8928 non-null   uint8
 25  education_tertiary   8928 non-null   uint8
 26  education_unknown    8928 non-null   uint8
 27  contact_cellular     8928 non-null   uint8
 28  contact_telephone    8928 non-null   uint8
 29  contact_unknown      8928 non-null   uint8
 30  month_apr            8928 non-null   uint8
 31  month_aug            8928 non-null   uint8
 32  month_dec            8928 non-null   uint8
 33  month_feb            8928 non-null   uint8
 34  month_jan            8928 non-null   uint8
 35  month_jul            8928 non-null   uint8
 36  month_jun            8928 non-null   uint8
 37  month_mar            8928 non-null   uint8
 38  month_may            8928 non-null   uint8
 39  month_nov            8928 non-null   uint8
 40  month_oct            8928 non-null   uint8
 41  month_sep            8928 non-null   uint8
 42  poutcome_failure     8928 non-null   uint8
 43  poutcome_other       8928 non-null   uint8
 44  poutcome_success     8928 non-null   uint8
 45  poutcome_unknown     8928 non-null   uint8
dtypes: float64(5), uint8(41)
```

On est donc passé de 15 colonnes avec la target deposit, à 46 colonnes sans la target deposit.

Nous affichons plus deposit ici, car nous avons profité de ce pre-processing, pour découper notre jeu de données en deux. D'un côté les features qui sont dans X_train. De l'autre la target qui se trouve dans y_ train. Cela sera indispensable pour créer nos premiers modèles de machines learning.

B. Modèle simple

a. Le principe du Machine Learning

Dans notre situation, le but est de prédire la valeur de deposit en fonction des autres variables disponibles (46 dans X_train). Dans ce cas, on dit que les données sont étiquetées. On entend par là que nous connaissons la variable que nous essayons de prédire.

Le machine learning est une sous catégorie de l'intelligence artificielle, dont le but est d'établir/trouver des patterns faisant le lien entre les features et la target. Le machine learning va donc faire ce que nous avons dans la première partie, d'une manière infiniment plus efficace et précise. Le machine learning à beau être très efficace quand c'est bien employé, il faut que nous soyons capable de mesurer son efficacité. Il existe pour cela ce qu'on appelle des métriques.

b. Les métriques d'évaluation

Il existe plusieurs métriques différentes qui répondent à des types de projets différents. Nous sommes sur un problème de classification binaire: est-ce que oui ou non le client va souscrire à un dépôt à terme. Voici les métriques que nous allons suivre au cours de ce projet

- *Accuracy*

L'accuracy montre le pourcentage de bonnes réponses lors d'une prédiction. Par exemple, si sur 100 clients, je prédis avec justesse si le client va souscrire ou non à 75 reprises, on dira alors que j'ai une accuracy de 75%.

L'accuracy n'est pas toujours une métrique pertinente. Imaginons un instant que seulement 1% des clients souscrivent au DAT. Si je reprends l'exemple précédent d'un échantillon de 100 clients, on pourrait simplement dire à chaque fois que le client ne va pas souscrire. On aura raison 99% du temps, mais ça ne veut pas pour autant dire que nos prédictions étaient pertinentes.

Plus la target est équilibrée, à savoir le ratio de oui et non est proche du 50/50, plus l'accuracy est pertinente. Et au contraire, elle devient complètement inutile lors d'un très fort déséquilibre 1/99.

L'accuracy est pertinente pour notre trainset car on a une répartition 52,5/47,5 en faveur du non.

```
round(y_train.value_counts(normalize=True),3)
```

```
0    0.525
1    0.475
Name: deposit, dtype: float64
```

- *Precision*

La précision est une métrique qui se concentre sur l'évaluation d'une classe. A chaque fois que le modèle va dire que deposit = 1, on va vérifier si c'est effectivement le cas. On parle de vrai positif (TP pour True positive). On compare le nombre de vrais positifs et le nombre de faux positifs (client qui ne souscrit pas mais on a prédit qu'il allait souscrire, False negative FP).

Dans le cadre du projet, on va définir une liste de clients pour lesquels on pense qu'ils vont souscrire. Ça veut dire que la précision correspondra parmi cette liste, aux clients ayant effectivement dit oui.

- *Recall*

Le recall est une métrique qui se concentre sur la proportion trouvée d'une classe. On va donc regarder l'ensemble des True Positive parmi l'ensemble des personnes qui ont effectivement souscrit. Dans certains cas, on va prédire qu'un client ne va pas souscrire alors que c'est en réalité le cas. On parle d'un False Negative (FN).

Dans ce projet, le recall va donc s'apparenter à la proportion de clients que l'on a trouvé parmi tous ceux qui sont prêts à souscrire.

- *F1*

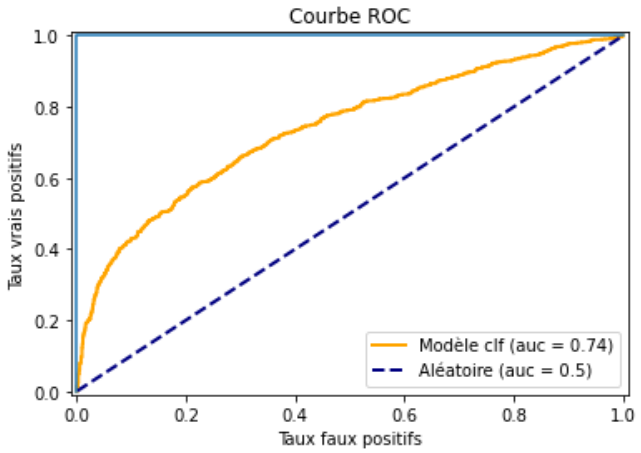
Le score F1 est un compromis entre le recall et la précision. C'est une métrique très utile pour remplacer l'accuracy dans un dataset déséquilibré.

- *ROC AUC*

La métrique ROC AUC est celle qui nous intéresse le plus, du moins dans le cadre de l'entraînement de nos modèles de machine learning. La courbe ROC (receiver operating characteristic) se base sur des probabilités que la target soit oui ou non, là où pour les autres métriques le modèle va prédire de manière catégorique un oui ou un non.

Quand on utilise des probabilités, le modèle peut plus ou moins être sûr de lui. Il va par exemple prédire un oui à 90%, ou alors un oui à 51% en cas de doute. La courbe ROC permet de classer toutes les prédictions de la plus forte probabilité à la plus faible dans un graphique en 2D. En ordonnées, on a le taux de True Positive trouvé, et en abscisse le taux de False Positive trouvé.

Voici à quoi ressemble une courbe ROC :



La ligne bleu clair dans le coin supérieur gauche montre le ROC parfait. On trouve d'abord tous les éléments de la classe que l'on souhaite, puis on finit par trouver les éléments de la classe que l'on ne souhaite pas.

La ligne en tiret bleu montre une prédiction aléatoire. En jouant à pile ou face, on a va trouver aussi rapidement la classe recherchée que la classe non recherchée. C'est pourquoi le taux de vrai et faux positif avance au même rythme. C'est le pire modèle possible.

La courbe orange est un exemple de prédiction. Elle doit normalement toujours se situer entre la prédiction parfaite et la prédiction aléatoire. Si ce n'est pas le cas, il y a une erreur dans le code.

AUC (Area under the curve) signifie l'aire sous la courbe. Le ROC AUC consiste donc à tracer une courbe, puis à mesurer la surface sous celle-ci. Un modèle parfait aura un aire de 100%, le modèle aléatoire de 50%. Pour maximiser ce score, il faut donc s'approcher au maximum des 100%.

Notre but étant de sélectionner un panel réduit de client mais qui ont une très grande chance de souscrire, cette métrique est très utile, car elle permet de classer du mieux possible l'ordre des clients susceptibles de souscrire à l'offre.

Maintenant que nous avons défini nos métriques utiles dans le cadre du projet, nous allons pouvoir entamer nos modèles de machines learning, à un détail près.

c. La Cross-Validation

Il nous reste une chose à faire avant de lancer nos modèles de machines learning. Nous devons faire une cross-validation. Il s'agit de couper le dataset en plusieurs parties (en 4 dans notre projet).

On va entraîner notre modèle 4 fois. A chaque fois, on va prendre 3 parties des données, et on va se servir pour essayer de prédire la dernière non utilisée. Cela permet de simuler des données de test. En effet lorsque X_train 1, X_train 2 et X_train 3 sont utilisés pour prédire les données de y_train 4, les données de X_train n'ont jamais été vues.

X_train & y_train 1	X_train & y_train 2	X_train 3 & y_train 3	X_train & y_train 4

On réalise donc cette opération 4 fois puis on fait la moyenne des résultats obtenus.

On intègre donc dans une fonction cross_validate, le modèle, les données, nos métriques.

```
cross_validate(clf, X_train, y_train,
               scoring=['accuracy', 'precision', 'recall', 'f1', 'roc_auc']
```

d. Création de différents modèles "simples"

Tout est prêt, il ne nous reste plus qu'à créer des modèles simples. On ne va pas détailler ici le principe de tous ces modèles, on le fera uniquement pour celui que l'on retient.

```
# Logistic Regression
LR_clf= LogisticRegression(random_state=0, max_iter=1000)

# SGD classifier
sgd_clf = SGDClassifier(random_state=0)

# LinearSVC
lsvc_clf = LinearSVC(random_state=0)

# Decision Tree
DTC_clf = DecisionTreeClassifier(random_state=0)

# Random Forest
RF_clf = RandomForestClassifier(random_state=0)

# Naive Bayes
nb_clf = GaussianNB()

# K-Nearest Neighbors - KNN
Knn_clf = KNeighborsClassifier()

# Adaboost Classifier
Ada_clf = AdaBoostClassifier(random_state=0)

# Bagging Classifier
bag_clf = BaggingClassifier(random_state=0)

# SVM classifier
svm_clf = SVC(random_state=0, probability=True)
```

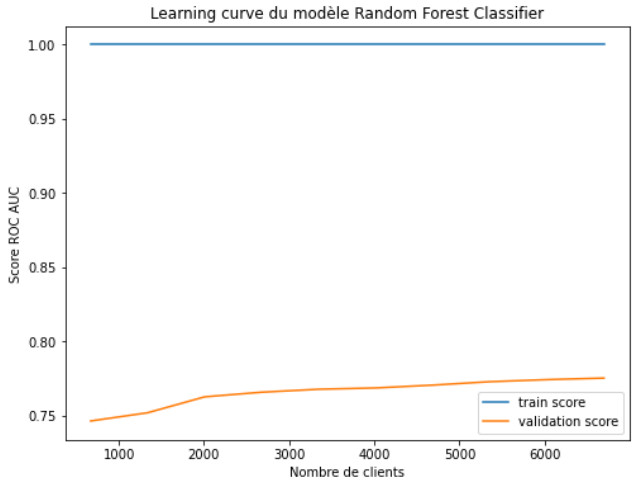
Ces 10 modèles ont été entraînés et voici les résultats :

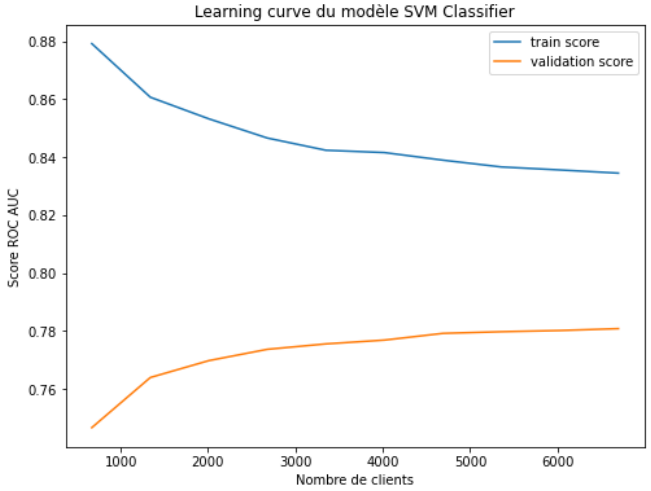
	Decision Tree Classifier	SGD Classifier	Linear SVC	Logistic Regression Classifier	Random Forest Classifier	AdaBoost Classifier	Bagging Classifier	KNN Classifier	Naive Bayes	SVM Classifier
Accuracy	0.641241	0.684364	0.698365	0.701165	0.718414	0.708109	0.698589	0.679323	0.673163	0.731855
Precision	0.623936	0.702104	0.735260	0.736106	0.729368	0.737446	0.719586	0.685596	0.755723	0.788232
Recall	0.615241	0.600832	0.570179	0.577965	0.647089	0.599433	0.598490	0.599906	0.460722	0.595424
F1	0.619487	0.641555	0.642214	0.647472	0.685718	0.661060	0.653466	0.639853	0.572411	0.678345
Roc_auc	0.639995	0.739381	0.757826	0.758460	0.775206	0.768568	0.751998	0.719771	0.732291	0.780861

Les 2 meilleurs modèles en se basant sur le ROC AUC sont Random Forest et SVM. C'est également ces 2 modèles qui obtiennent le meilleur score dans au moins l'une des métriques affichées.

Nous allons introduire le concept de learning curve. Cela permet deux choses :

- voir si nous disposons de suffisamment de données pour que notre modèle soit efficace.
- évaluer si le modèle fait de l'overfitting.





En ce qui concerne le nombre de données, il faut regarder le validation score en orange. Plus on a de données et plus le score va augmenter. Pour les deux modèles on constate que cela continue de monter mais que l'augmentation marque le pas à partir de 2000 clients. Les modèles pourraient donc augmenter en performance avec plus de données, mais cela ne serait pas très significatif car on a atteint un plateau.

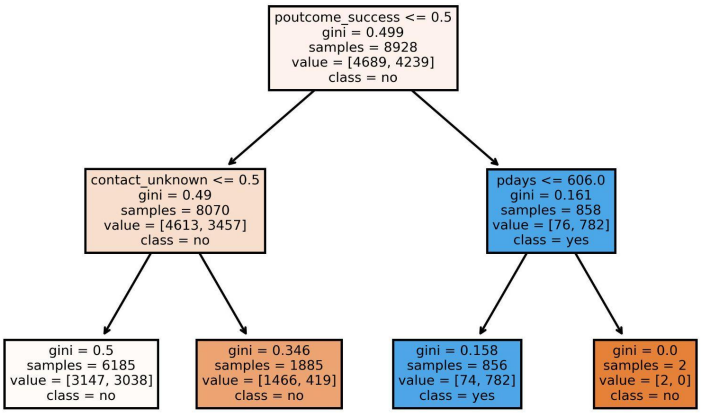
Concernant l'overfitting, c'est un phénomène dit de sur-apprentissage du modèle. Ce dernier apprend par cœur les données d'entraînement, mais n'est pas en mesure de généraliser sur des données qu'il n'a jamais vu. On peut le constater notamment lorsque l'écart entre train et validation set est élevé. C'est ici le cas du random forest, qui fait un 100% sur le trainset. Il est possible de supprimer en partie ou totalité overfitting d'un modèle en le paramétrant correctement.

Sur ce projet, l'interprétabilité du modèle est fondamentale. C'est pourquoi nous allons mettre de côté le SVM, qui offre moins de possibilités à ce sujet. C'est également un modèle plus lent que le Random Forest. Nous décidons donc d'optimiser le Random Forest.

C. Le modèle Random Forest

a. Principe du Decision Tree

Pour parler d'un Random Forest, il faut déjà évoquer le Decision Tree (Arbre de décision). Au sommet de l'arbre, on a toutes les données. Le Decision Tree va à chaque étape, découper les données en deux en se basant sur une feature afin de maximiser la séparation entre deposit = yes et deposit = no.



Dans cet exemple, on a 8928 données au départ répartie en 4689 / 4239 en faveur du 'no'. Si poutcome_success = 0, les données partent à gauche, sinon elles partent à droite. Plus un carré est bleu, plus la proportion de yes est élevée et vice-versa pour l'orange. C'est ici un exemple très simple, on pourrait ajouter de nouveaux embranchements en dessous. Le Decision Tree est un modèle facilement interprétable, il n'est en revanche pas très performant et est très sensible à l'overfitting que nous souhaitons éviter. C'est ici qu'intervient le Random Forest.

b. Principe du Random Forest

Le Random Forest (Forêt aléatoire) fait appel à plusieurs Decision Tree. On y ajoute également une composante aléatoire, en limitant le nombre de données pour chaque arbre. Le Random Forest, va ensuite faire une moyenne des différents Decision Tree. Le Random Forest, va donc gagner en performance et réduire l'overfitting. En revanche, il va perdre un peu de son interprétabilité.

- *Points forts du Random Forest*

- Bien qu'il soit moins bon en interprétabilité qu'un Decision Tree, le Random Forest reste un des meilleurs modèles dans ce domaine. Il est facile d'afficher un arbre de décision et de comprendre comment le résultat à été défini.
- Ce modèle ne nécessite aucune normalisation. Ce qui veut dire que l'on va pouvoir conserver l'âge, la balance…etc à leurs échelles respectives.
- De plus, il est très résistant aux outliers (valeurs extrêmes), et nécessite donc très peu d'intervention de ce côté là.

- *Points faibles du Random Forest*

- Le risque d'overfitting est moins fort que pour un Decision Tree notamment grâce à l'aléatoire, mais il reste très élevé par rapport à d'autres modèles.
- Il y a une méthode simple appelée feature importance qui permet comme son nom l'indique de classer les features par importance. Celle-ci est biaisée, elle va favoriser les features pour lesquelles il existe le plus de valeur unique. Il faut donc être vigilant avec l'interprétation de cette information.

Maintenant que nous avons défini et expliqué notre futur modèle, nous allons optimiser nos résultats.

Créer un modèle Random Forest performant

A. Pre-processing avancé

Nous avions fait un pre-processing simple pour faire tourner des premiers modèles. Nous allons maintenant chercher à optimiser cette phase afin de mettre le modèle dans les meilleures dispositions lors de son entraînement.

a. Méthode de feature selection

- *VarianceThreshold*

VarianceThreshold permet de conserver uniquement les features qui ont une variance minimum. Par exemple, si nous avions une constante dans notre dataset, VarianceThreshold la supprimerait directement car c'est inutile et même plutôt néfaste pour le modèle.

Comme nous avons pu le voir dans la première partie, la feature default varie très peu. On devrait donc envisager de la supprimer.

- *SelectKBest*

SelectKBest permet de déterminer les meilleures features en fonction de la target. Nous avons 46 colonnes pour le moment. Nous pouvons donc lancer une boucle pour connaître le nombre de features à conserver. La boucle va entraîner le modèle sur la meilleure feature, puis sur les deux meilleures etc.

Cela n'a pas donné des résultats très significatifs, mais certaines features comme job_admin. job_self-employed ont tendance à améliorer le résultat du modèle lorsqu'elles sont supprimées. On va les conserver pour le moment mais on note que job à des valeurs susceptibles d'être supprimées. Les valeurs unknown sont aussi problématiques, job_unknown et education_unknown font partie des pires features pour SelectKBest.

b. Imputation des données Unknown

Pour rappel, nous avions vu 4 colonnes avec des données 'unknown' :

```
poutcome     74.592367
contact      21.017739
education     4.452607
job           0.627128
```

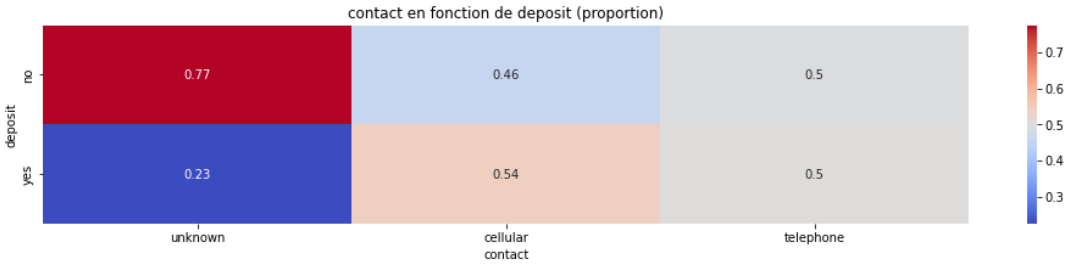
Pour 'poutcome', c'est simple à expliquer, cela concerne tous les clients dont c'est la première campagne. Il n'y a donc pas de résultat précédent. Nous pouvons donc laisser cette donnée en l'état.

Pour contact, nous n'avons aucune piste. Nous constatons juste que la grande majorité des cas se trouve en mai ou en juin.

Voici la proportion de contact par mois :

month	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
contact												
cellular	0.89	0.89	0.89	0.93	0.46	0.31	0.85	0.96	0.87	0.78	0.89	0.82
telephone	0.11	0.10	0.09	0.07	0.04	0.02	0.12	0.04	0.10	0.17	0.09	0.17
unknown	0.00	0.01	0.01	0.01	0.50	0.66	0.03	0.01	0.03	0.05	0.02	0.01

Nous sommes obligé de conserver cette information car on avait constaté un très fort écart de souscription pour cette valeur inconnue :



Pour 'job' et 'education', nous avons très peu de données manquantes. Nous avons deux possibilités:

- Supprimer les lignes concernées au risque de perdre de l'information sur les autres colonnes
- Remplacer les unknowns par une autre valeur au risque de perdre de l'information sur ces valeurs.

Nous avons décidé de remplacer la valeur unknown, pour compléter éducation, nous nous sommes basé sur le job du client :

```
# gestion des unknown pour education
primary = ['housemaid']
secondary = ['admin.', 'entrepreneur', 'retired', 'services', 'student', 'technician', 'unemployed', 'blue-collar', 'unknown']
tertiary = ['management', 'self-employed']
```

Inversement, pour compléter job, on s'est appuyé sur la colonne 'education' ainsi que 'age' avec l'algorithme suivant :

```
if df['age'] >= 60:
    return 'retired'
if df['age'] <= 25:
    return 'student'
if df['education'] == 'tertiary':
    return 'management'
if df['education'] == 'secondary':
    return 'technician'
if df['education'] == 'primary':
    return 'blue-collar'
if df['education'] == 'unknown':
    return 'technician' # job qui complétera le reste des unknown
```

c. Feature selection manuelle

Maintenant que nous n'avons plus d'unknown nous pouvons regarder les features qui ne sont pas utiles pour le modèle. On a tenté de supprimer chacune des colonnes une à une pour voir s' il y avait un impact positif, négatif ou neutre. Suite à cette étape, nous avons supprimé 'default' qui ne variait pas assez. Plus surprenant, nous avons constaté que supprimer complètement la colonne 'job' améliore le résultat du modèle. Nous avions vu des écarts entre les différents 'job', notamment pour student et retired, ils semblent que ces informations soient trop redondantes avec la feature 'age' et que cette dernière se suffit à elle-même. 'previous' est également supprimé car trop redondant. L'information utile était de savoir si oui ou non le client a déjà été contacté dans une campagne précédente, c'est une information disponible dans pdays ou encore poutcome = unknown.

Nous supprimons donc 'duration' et 'campaign' car nous ne pouvons pas avoir cette information avant l'appel. Puis nous supprimons 'previous', 'job' et 'default', car cela améliore la performance du Random Forest. Voici les colonnes que nous conservons (12 sur 17) :

```
keep_columns = ['age', 'marital', 'education', 'balance', 'housing', 'loan',
                'contact', 'day', 'month', 'pdays', 'poutcome', 'deposit']
```

d. Feature Engineering

Le feature engineering consiste à modifier ou créer de nouvelles variables à partir de l'existant. Nous avons testé énormément de possibilités. On va lister quelques tentatives (liste non exhaustive) qui n'ont pas fonctionné puis nous listerons celles qui ont donné un résultat positif.

- *Ce qui n'a pas fonctionné*

- Définir des classes :
  - pour balance (ex : Solde négatif, Solde de moins de 1000€ etc…)
  - pour âge (ex : 18-29 ans - 30-39 ans etc…)
  - pour day (1-15 et 16-31… découpé en 3, en 4….)
- Regrouper :
  - les actifs et inactifs (inactif = student + retired, actifs = les autres)
  - les grandes campagnes d'appel (pdays autour de 90 ; pdays autour de 180…)
  - pdays en mois, en trimestre…
  - le contact telephone et cellular ensemble par opposition à unknown
  - les prêts housing et loan, soit en considérant "au moins un crédit parmi les deux", soit en faisant la somme des crédits.
  - les mois directement de 1 à 12 comme pour day

- *Ce qui a fonctionné*

- Conserver uniquement la valeur success pour poutcome et supprimer le reste
- Classer l'éducation avec le format suivant : primary = 1, secondary = 2, tertiary = 3
- Classer le statut marital avec le format suivant : single = 1, divorced = 2, married = 3

e. Encodage

Pour l'encodage, nous reprenons simplement le code de la première phase de pre-processing. En gérant correctement les types de colonnes, nos modifications précédentes ont été prises en compte automatiquement.

f. Gestion des outliers

Le Random Forest est très résistant aux outliers, mais nous avons tout de même réussi à améliorer le modèle en imposant une limite sur la balance. Nous avions vu que plus la balance augmente, plus on a de chance d'avoir deposit avec yes, mais aussi, qu'il y avait une espèce de seuil, qui fait qu'à partir d'un moment, avoir plus d'argent n'augmente pas les chances de souscrire.

Nous avons utilisé la méthode des interquartiles. Cela consiste à prendre la valeur de balance à 25% et 75%. Puis d'appliquer le code suivant :

```
Q1, Q3 = np.percentile(df['balance'], [25,75])
IQR = Q3 - Q1
lower_range_balance = Q1 - (1.5 * IQR)
upper_range_balance = Q3 + (1.5 * IQR)
```

lower_range_balance	upper_range_balance
-2235.5	4056.5

Cela veut dire par exemple, que si un client à une balance de 15000, celle-ci sera considéré comme 4056,5.

g. Feature scaling

C'est l'un des avantages du Random Forest, il ne nécessite pas de Feature Scaling pour fonctionner de manière optimale. Nous pouvons donc sauter cette étape.

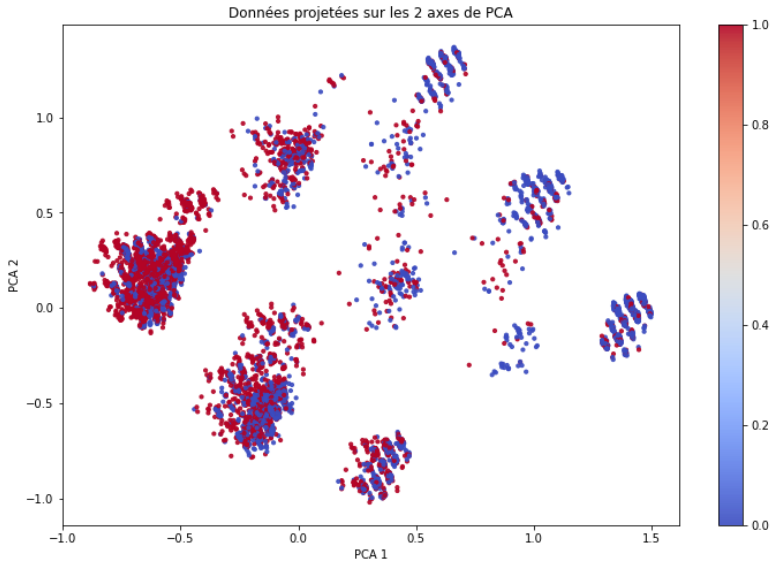
On peut maintenant lancer le résultat du pré-processing.

h. Information sur le trainset après le pre-processing.

Nous avions 46 colonnes après le premier pré-processing. Nous en avons maintenant 24.

```
Data columns (total 24 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   age                8928 non-null   int64
 1   marital            8928 non-null   int64
 2   education          8928 non-null   int64
 3   balance            8928 non-null   int64
 4   housing            8928 non-null   uint8
 5   loan               8928 non-null   uint8
 6   day                8928 non-null   int64
 7   pdays              8928 non-null   int64
 8   poutcome_success   8928 non-null   int64
 9   contact_cellular   8928 non-null   uint8
 10  contact_telephone  8928 non-null   uint8
 11  contact_unknown    8928 non-null   uint8
 12  month_apr          8928 non-null   uint8
 13  month_aug          8928 non-null   uint8
 14  month_dec          8928 non-null   uint8
 15  month_feb          8928 non-null   uint8
 16  month_jan          8928 non-null   uint8
 17  month_jul          8928 non-null   uint8
 18  month_jun          8928 non-null   uint8
 19  month_mar          8928 non-null   uint8
 20  month_may          8928 non-null   uint8
 21  month_nov          8928 non-null   uint8
 22  month_oct          8928 non-null   uint8
 23  month_sep          8928 non-null   uint8
dtypes: int64(7), uint8(17)
```

i. PCA après la seconde phase de pré-processing



Sur ce nouveau PCA, les données sont mieux regroupées en différents blocs car on a supprimé toutes les données qui polluent le dataset. Il est plus facile d'identifier le groupe rouge majoritaire qui est à gauche ainsi que les groupes bleus. Cependant, nous avons toujours énormément de points bleu et rouge mélangés. Cela confirme notre première analyse et montre que pour certains clients, il sera difficile de faire une bonne prédiction.

j. Résultat du pré-processing

	1st RF & 1st preprocessing	1st RF & 2nd preprocessing	Diff entre preprocessing
Accuracy	71.841398	72.323029	0.481631
Precision	72.936796	73.265322	0.328526
Recall	64.708852	65.675922	0.967070
F1	68.571827	69.261645	0.689818
Roc_auc	77.520618	78.644876	1.124258

Nous avons toujours le même modèle basique de Random Forest. Juste en améliorant la phase de pré-processing, on a amélioré toutes les métriques. On gagne 1,12 % sur le ROC AUC qui est la métrique principale pour le moment.

Nous pouvons maintenant passer à l'optimisation des hyperparamètres de notre modèle.

B. Optimisation des hyperparamètres du Random Forest

a. Définition de l'hyperparamètre

Un hyperparamètre est un paramètre lié au modèle qui permet de contrôler son processus d'apprentissage. En modifiant ces derniers, on peut fortement changer l'approche du modèle lors de l'entraînement et par conséquent, changer également la performance du modèle.

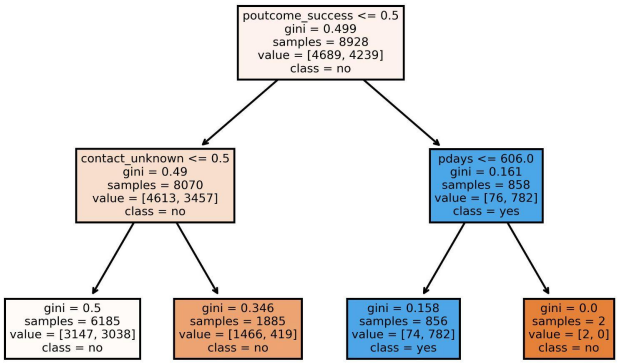
On va détailler les hyper paramètres utilisés.

b. Criterion

Dans un Decision Tree, lors de la séparation des données en deux branches, le modèle cherche à maximiser la séparation entre les deux classes recherchées. Criterion est la méthode de calcul appliquée pour choisir cette séparation.

c. Max_depth

Max_depth représente la profondeur maximale d'un arbre, dans l'exemple ci-dessous, max_depth est égale à 2. Plus la profondeur est élevée et plus le modèle fera de l'overfitting. Par défaut, la valeur est illimitée, ce qui explique pourquoi nous avons de l'overfitting sur le graphique de la learning curve.



d. Max_features

Max_features permet d'apporter de l'aléatoire au modèle. Pour chaque embranchement, on va prendre au hasard le nombre de features max défini préalablement. Parmi elles, le modèle va choisir la feature la plus utile. Cet hyperparamètre force le modèle à tester des features moins utiles. Plus max_features est élevé, plus le modèle fera de l'overfitting car on réduira la part d'aléatoire.

e. Min_Samples_split

min_samples_split est la limite qui permet de découper un nœud en deux nouvelles branches. Dans l'exemple au dessus, si min_samples_split était égal à 1000. La première case bleu a droite n'aurait pas pu se diviser en deux car il n'y a que 858 données. Plus on augmente la valeur, plus on réduit l'overfitting. En revanche, si on augmente trop cette valeur, on risque de fortement baisser les performances du modèle.

f. Min_samples_leaf

min_samples_leaf fonctionne un peu comme l'hyperparamètres précédent. La différence réside que la valeur vérifiée n'est plus celle de départ mais celle d'arrivée. Si min_samples_leaf était égal à 3, la case orange en bas à droite n'aurait pas pu être créée, car il n'y a que 2 données dedans. Augmenter min_samples_leaf réduit l'overffiting mais peut réduire la performance du modèle.

g. Class_weight

class_weight permet d'équilibrer le poids entre les classes 1 et 0 de deposit. Notre target n'est pas parfaitement équilibré avec 52,5/47,5 en faveur du no. Class_weight permet d'augmenter le poids d'une des deux classes, afin qu'il soit plus punitif pour le modèle de se tromper sur une classe plutôt que l'autre. Il y a un critère 'balanced', qui permet d'avoir un ratio inversement proportionnel à notre distribution, ce qui permet de simuler une distribution équilibrée.

h. N_estimators

C'est simplement le nombre d'arbres utilisés dans notre forêt aléatoire. Plus le chiffre est élevé et plus la performance augmente. Cependant, on atteint à partir d'un plateau un certain seuil. Ce n'est donc pas utile d'avoir une valeur trop élevée, d'autant que plus celle-ci est élevée, plus le modèle va prendre du temps à s'entraîner.

i. GridSearchCV

La fonction GridSearchCV, inclut directement une cross-validation qui est essentielle pour s'assurer de la performance du modèle sur des nouvelles données. Cela permet également de rentrer une liste complète d'hyperparamètres, afin que le modèle testé toutes les possibilités. C'est donc très efficace pour trouver la meilleure combinaison d'hyperparamètre. Le problème étant, que plus il y a de possibilité à tester, plus le résultat sera long à obtenir.

j. Résultats de l'optimisation des hyperparamètres

Voici les deux meilleurs résultats obtenus (n'affiche pas l'hyperparamètre en cas de valeur par défaut) :

```
Validation score : 0.8017583067133703
Best estimator : RandomForestClassifier(criterion='entropy', max_depth=17, max_features=3,
                       min_samples_split=10, n_estimators=425, n_jobs=-1,
                       random_state=0)
Train score 0.9308695606019739
```

Ce qui nous intéresse pour juger le modèle est le validation score. C'est le score qu'on espère avoir lorsque l'on aura de nouvelles données.

Le train score est très élevé, il est à 93% ce qui montre un fort overfitting. C'est souvent le cas pour des modèles Random Forest, mais nous avons essayé de réduire le train score sans impacter le validation score.

```
Validation score : 0.8015409621241476
Best estimator : RandomForestClassifier(criterion='entropy', max_depth=21, max_features=2,
                       min_samples_split=24, n_estimators=181, n_jobs=-1,
                       random_state=0)
Train score 0.899907534818079
```

On perd 0,02% sur le validation score ce qui est négligeable. On perd également un peu plus de 3% sur le train score ce qui est bon signe. Il y a encore de l'overfitting mais il a été considérablement réduit. De plus, si on baisse plus le train score, on a un impact significatif sur le validation score, ce qui indique alors que notre modèle serait moins bon sur de nouvelles données. C'est la meilleure configuration que nous ayons trouvé.

Nous pouvons maintenant analyser les performances de notre modèle.

IV. Résultats du modèle

A) Analyse de performance sur le training set

a. Écart de performance entre la dernière itération du modèle et les précédentes.

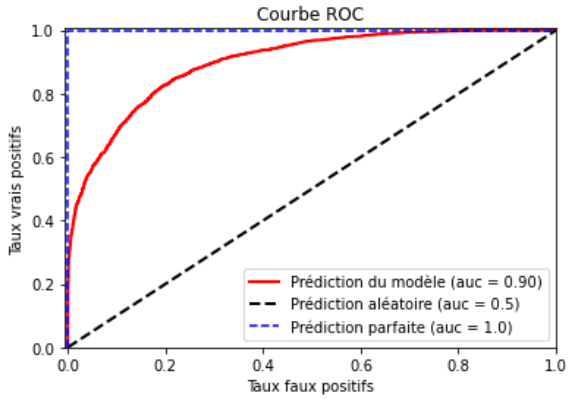
	1st RF & 1st preprocessing	1st RF & 2nd preprocessing	Random Forest Final	Final vs 1st	Final vs 2nd
Accuracy	71.841398	72.323029	74.025538	2.184140	1.702509
Precision	72.936796	73.265322	77.291688	4.354892	4.026366
Recall	64.708852	65.675922	64.142592	-0.566260	-1.533331
F1	68.571827	69.261645	70.101452	1.529625	0.839807
Roc_auc	77.520618	78.644876	80.154096	2.633479	1.509220

Nous avons encore amélioré le score du modèle. La seule métrique qui est moins bonne est le recall et ce n'est pas vraiment important. Cela veut juste dire que le précédent modèle préconise d'appeler plus de clients quitte à se tromper plus souvent.

Nous allons maintenant pouvoir regarder les résultats sur le train set en sachant qu'il va y avoir de l'overfitting. On va prendre cela en compte pour essayer d'anticiper les résultats du test set.

b. ROC AUC

Voici la courbe ROC lorsque l'on prédit les données du trainset.

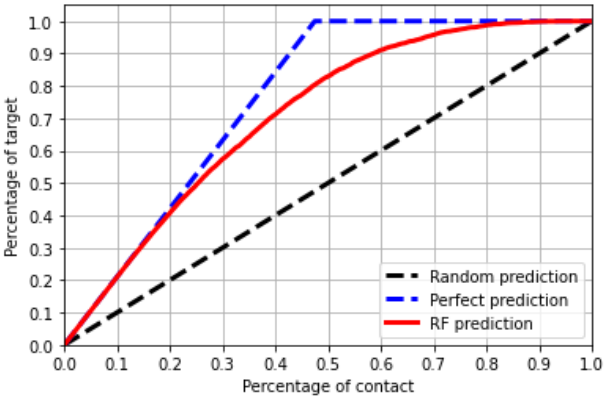


On s'attend, pour des données jamais vues, à un score de 0.8, soit 0.1 point de moins. On voit bien l'overffiting sur cette courbe rouge. Au départ, elle est complètement collée à la prédiction parfaite. C'est également le cas à la fin. Si il n'y avait pas d'overfitting, la courbe serait plus en dent de scie.

c. Cumulative Gain Curve

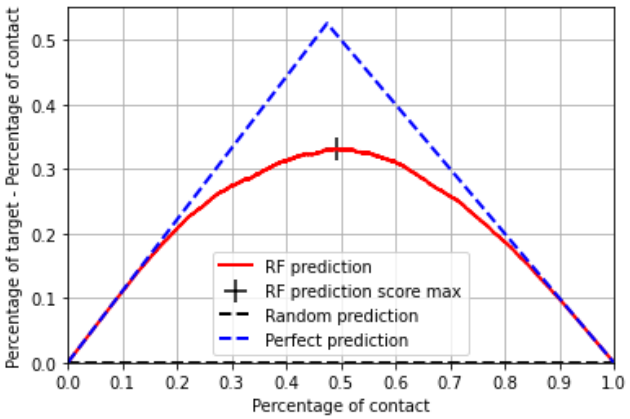
La métrique ROC AUC est très utile pour mesurer la performance du modèle. Cependant, c'est assez difficile de se projeter sur le nombre de clients que l'on va contacter et le nombre de clients qui vont souscrire.

Pour cela, nous avons la Cumulative Gain Curve. C'est en principe assez proche de la courbe ROC, mais beaucoup plus compréhensible. Le défaut de la Gain Curve, c'est qu'elle est dépendante de la distribution de données. La prédiction aléatoire sera toujours identique, en revanche, la prédiction parfaite dépend directement de la distribution.



On peut voir qu'en contactant 50% des clients, on arrive à trouver plus de 80% de la target. Il ne faut pas oublier qu'on regarde le trainset, et que ce dernier est en overfitting. Il faut minorer le résultat affiché.

Il est possible d'aplatir cette courbe pour faire en sorte que la prédiction aléatoire soit sur l'axe horizontal.



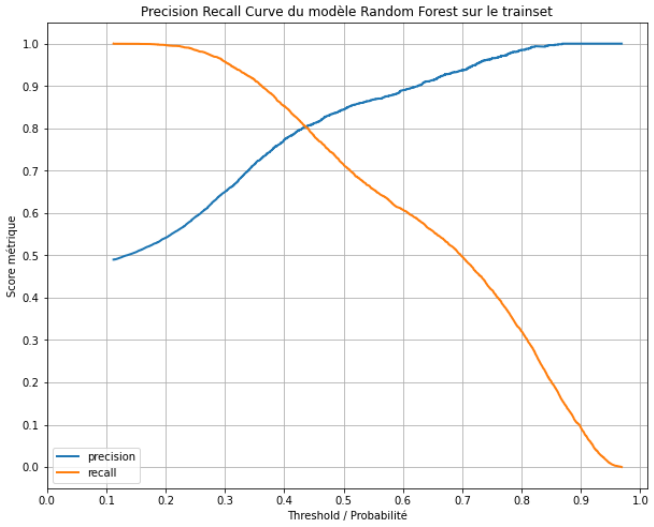
Lorsque la courbe rouge monte, cela veut dire que la prédiction du modèle est meilleure que l'aléatoire. C'est le contraire lorsqu'elle descend.

Notre modèle trouve beaucoup de clients qui vont souscrire au départ. A cause de celà, il reste moins de clients à trouver. C'est pour cela que plus on contact de client moins notre modèle est performant par rapport au hasard.

Le problème de ce graphique, c'est qu'il ne nous montre pas le seuil de probabilité à prendre pour considérer que le client va souscrire. Comme précisé auparavant, la cumulative gain curve est dépendante de la distribution de deposit. On est à 47,5% sur le trainset, mais dans le cadre d'une future campagne, nous pourrions avoir uniquement 20% de client prêt à souscrire par exemple. C'est pourquoi il est important de connaître le seuil de prédiction (Threshold).

d. Precision Recall Curve

La Precision Recall Curve permet d'afficher l'évolution de la Précision et de Recall en fonction du Threshold. Avec un Threshold = 0, on contacte tous les clients, c'est pourquoi le recall est à 100%. Au contraire, le Threshold = 1, signifie qu'on ne contacte personne. En ne contactant personne, on a forcément une precision de 100%.



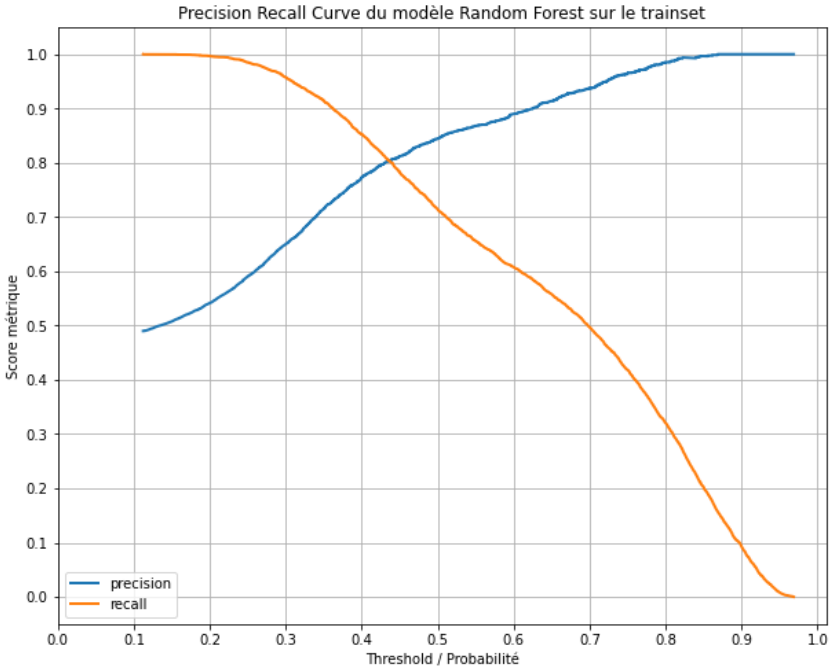
Sur le trainset, on voit par exemple qu'avec un seuil de 0.6, on a presque 90% de précision. Cela veut dire que 9 clients sur 10 souscrivent. On trouverait pour cet exemple 60% des clients prêts à souscrire.

e. Écart de performance entre Train et Validation score

Comme mentionné précédemment, les graphiques sont basés sur le Trainset dans lequel on a un overfitting prononcé. Nous allons comparer les différentes métriques entre le Train score et le Validation score, pour essayer d'anticiper la précision recall curve du testset.

	1st RF & 1st preprocessing	1st RF & 2nd preprocessing	Random Forest Final	Random Forest Train	Train - Validation
Accuracy	71.841398	72.323029	74.025538	80.174731	6.149194
Precision	72.936796	73.265322	77.291688	84.454368	7.162680
Recall	64.708852	65.675922	64.142592	71.384761	7.242169
F1	68.571827	69.261645	70.101452	77.371516	7.270064
Roc_auc	77.520618	78.644876	80.154096	89.990753	9.836657

Comme prévu, l'overfitting fait que les résultats du train score sont élevés. On peut utiliser la dernière colonne pour essayer d'estimer le résultat du testset. On voit que la précision et le recall sont environ 7% inférieur sur le score de validation.

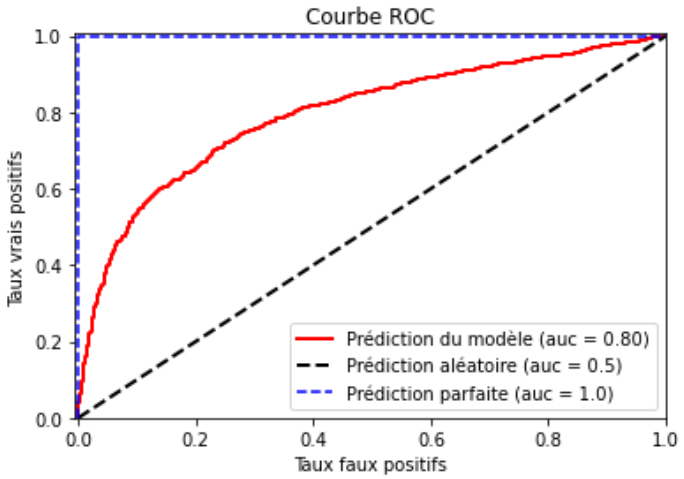


Si on reprend l'exemple du Threshold à 0,6. On peut imaginer que la précision serait aux alentours de 81% (88% - 7%) et le recall autour de 55% (62% - 7%). Cette estimation est possible uniquement parce que l'on sait que le testset est plus ou moins équilibré comme le trainset.

B) Performance sur le test set

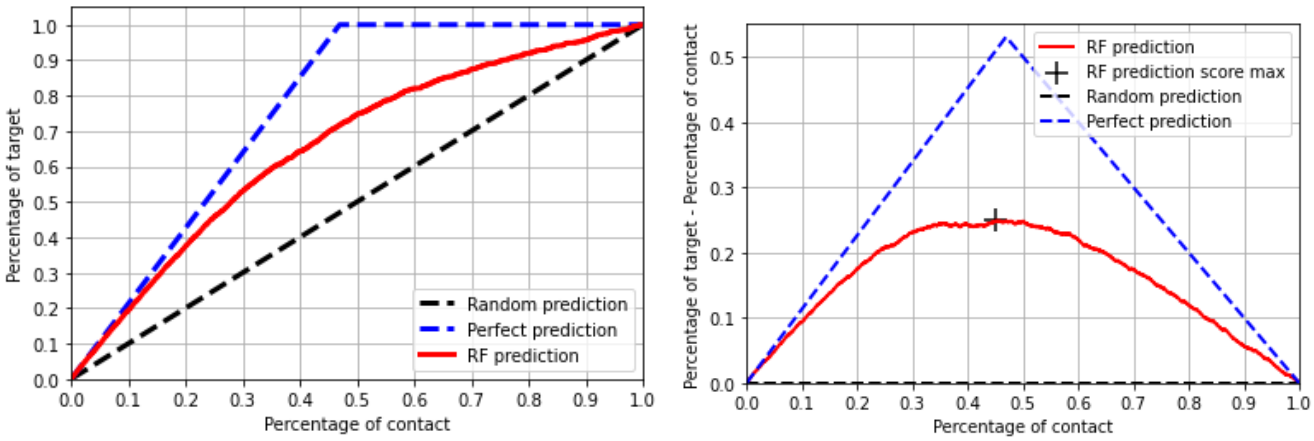
a. Courbe ROC

Le modèle obtient un score de 80% sur la prédiction du testset. Pour rappel, notre modèle n'a jamais vu ces données. Cela confirme notre score de validation, qui était lui aussi de 80%. Cela montre que notre modèle est capable de généraliser sur des nouvelles données.



On voit que l'overfitting à disparu. La courbe rouge est détachée de la prédiction parfaite.

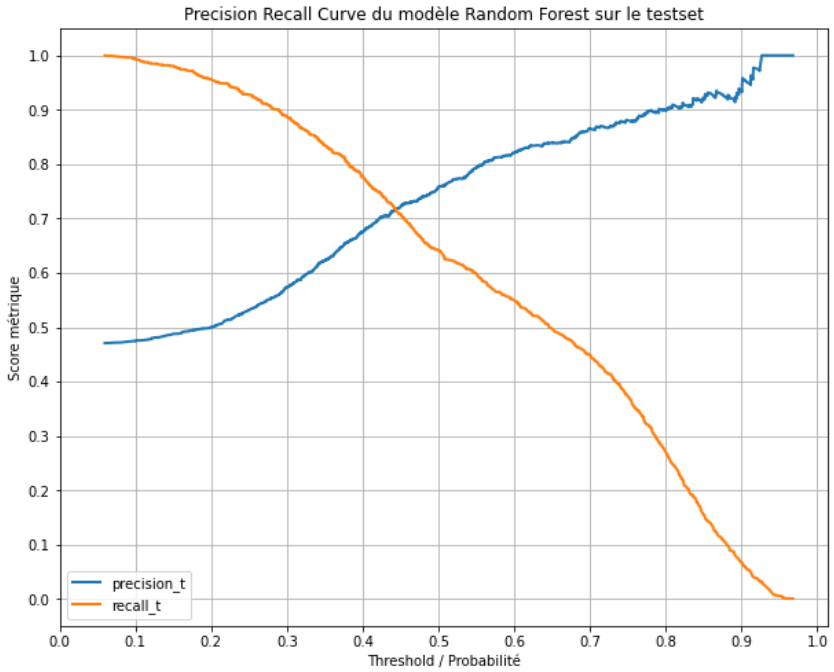
b. Cumulative Gain Curve



Le point maximum se trouve autour de 45% de contact. Mais la courbe ne monte plus vraiment à partir de 30%.

c. Précision Recall Curve

Le graphique ci-dessous confirme l'estimation de la precision recall curve du train score. Cela veut dire qu'à l'avenir, si la target à un répartition similaire, on peut prendre le graphique ci-dessous pour faire une estimation de la précision en fonction du Threshold.



d. Matrice de confusion et résultat

Avec un Threshold à 0.5, on obtient la matrice de confusion suivante :

Predicted	0	1
Reality		
0	969	215
1	376	672

Voici les résultats du testset :

	Random Forest Final	Random Forest Test	Test - Validation
Accuracy	74.025538	73.521505	-0.504032
Precision	77.291688	75.760992	-1.530696
Recall	64.142592	64.122137	-0.020454
F1	70.101452	69.457364	-0.644088
Roc_auc	80.154096	79.603121	-0.550975

Le score de validation est meilleur que celui du test. Cela n'a rien d'étonnant. On peut toujours avoir une légère perte sur de nouvelles données. Globalement le résultat est très proche de ce qu'on avait eu dans la validation. Le modèle conserve un bon Roc_Auc ainsi qu'une bonne precision. Il est possible d'augmenter la precision, en réduisant le recall (utilisation d'un Threshold plus élevé que 0.5).

Il est maintenant temps de passer au dernier point du projet, l'intéprétabilité du modèle.

C) Interprétabilité du Random Forest

Dans l'énoncé du projet, il est clairement stipulé que l'interprétabilité du modèle est indispensable. La banque souhaite comprendre les prédictions effectuées par le modèle.

a. Decision Tree basé sur le Random Forest

L'avantage d'un Random Forest, c'est qu'il est possible de faire un Decision Tree calqué sur notre système de prédiction. C'est aussi pour l'interprétabilité que nous avons choisi le modèle Random Forest.

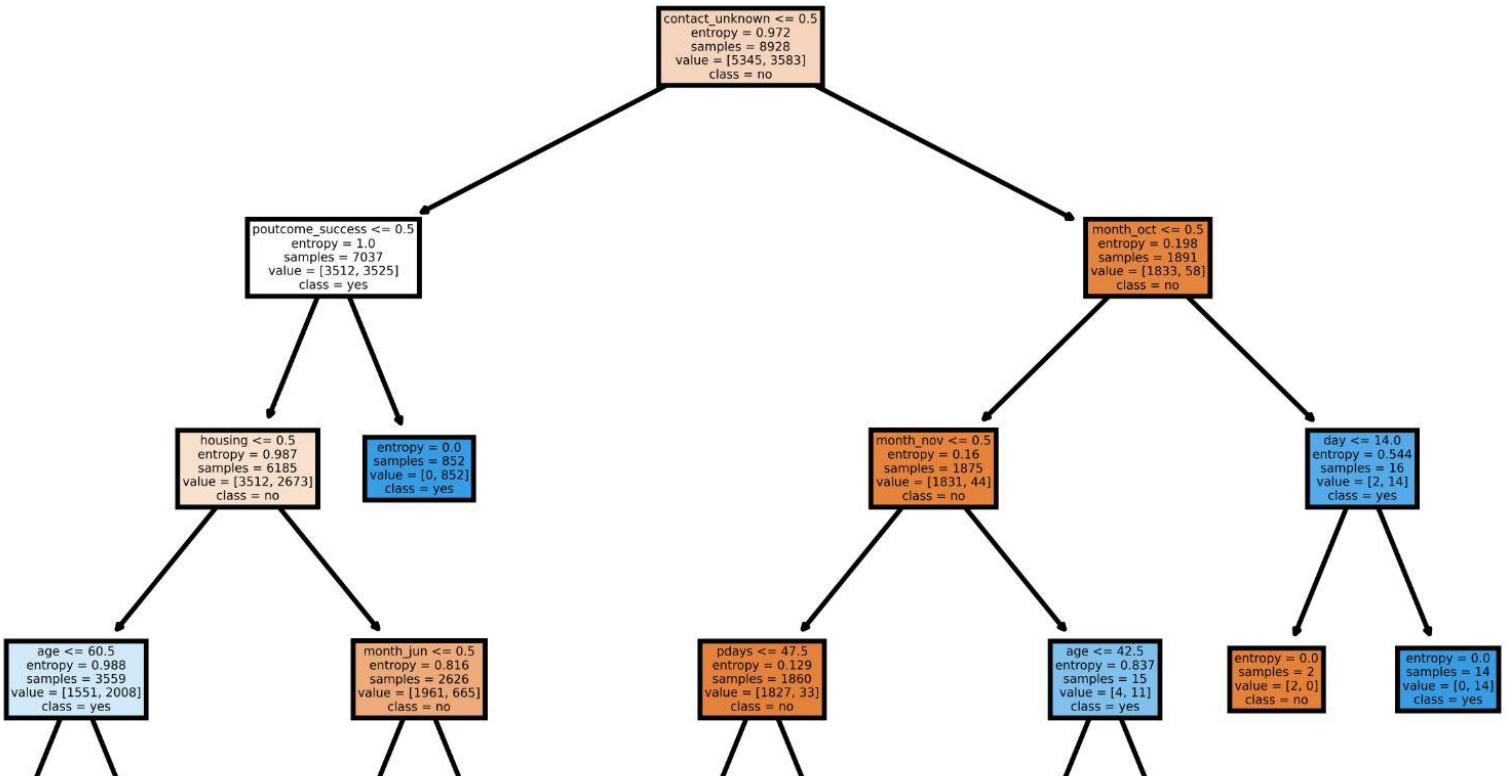
Nous allons entraîner une modèle Decision Tree, en se basant non pas sur le vrai deposit, mais sur les prédictions de notre Random Forest. On va également laisser le Decision Tree overfit volontairement. Le but n'est pas d'optimiser un modèle mais d'essayer de simuler celui existant.

On affiche ensuite le décalage de prédiction sur le trainset et le testset :

```
Différence de prédiction sur le trainset entre notre modèle et l'arbre de décision : 0
Différence de prédiction sur le testset entre notre modèle et l'arbre de décision : 0.07571684587813621
```

On a donc un Decision Tree qui prend exactement les même décisions que le Random Forest sur le trainset. En ce qui concerne le testset, la prédiction est identique dans 92,4% des cas. Notre Decision Tree représente donc avec une certaine fidélité la manière dont sont prises la décision du Random Forest.

- Affichage du decision Tree



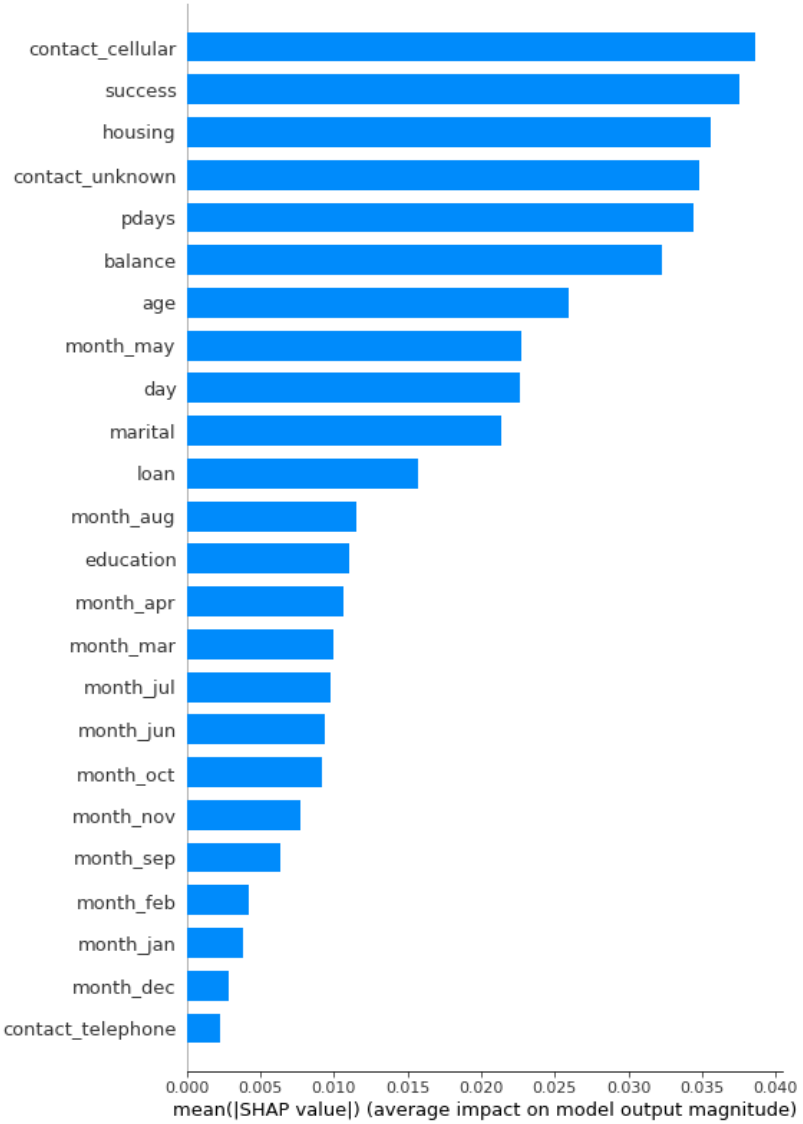
Si on prend l'exemple de cet arbre, on sait qu'un client qui n'a pas un contact unknown et qui a un poutcome = success, sera prédit comme allant souscrire au DAT. Pour une question d'affichage, on montre uniquement 3 étages, mais il est possible de publier l'arbre en entier. C'est plutôt utile pour comprendre la décision du modèle, c'est en revanche peu pratique de vérifier chaque client en déroulant l'arbre. Aussi, ça ne montre pas clairement qu'elles sont les features les plus impactantes.

On va éclaircir ces points tout de suite.

b. Interprétation globale avec SHAP

Shap est une librairie spécialisée dans l'interprétation des modèles. Elle permet de faire une analyse globale et locale de notre modèle.

L'interprétation globale est une vue générale sur comment se comporte le modèle. Nous pouvons afficher les features par ordre d'importance :

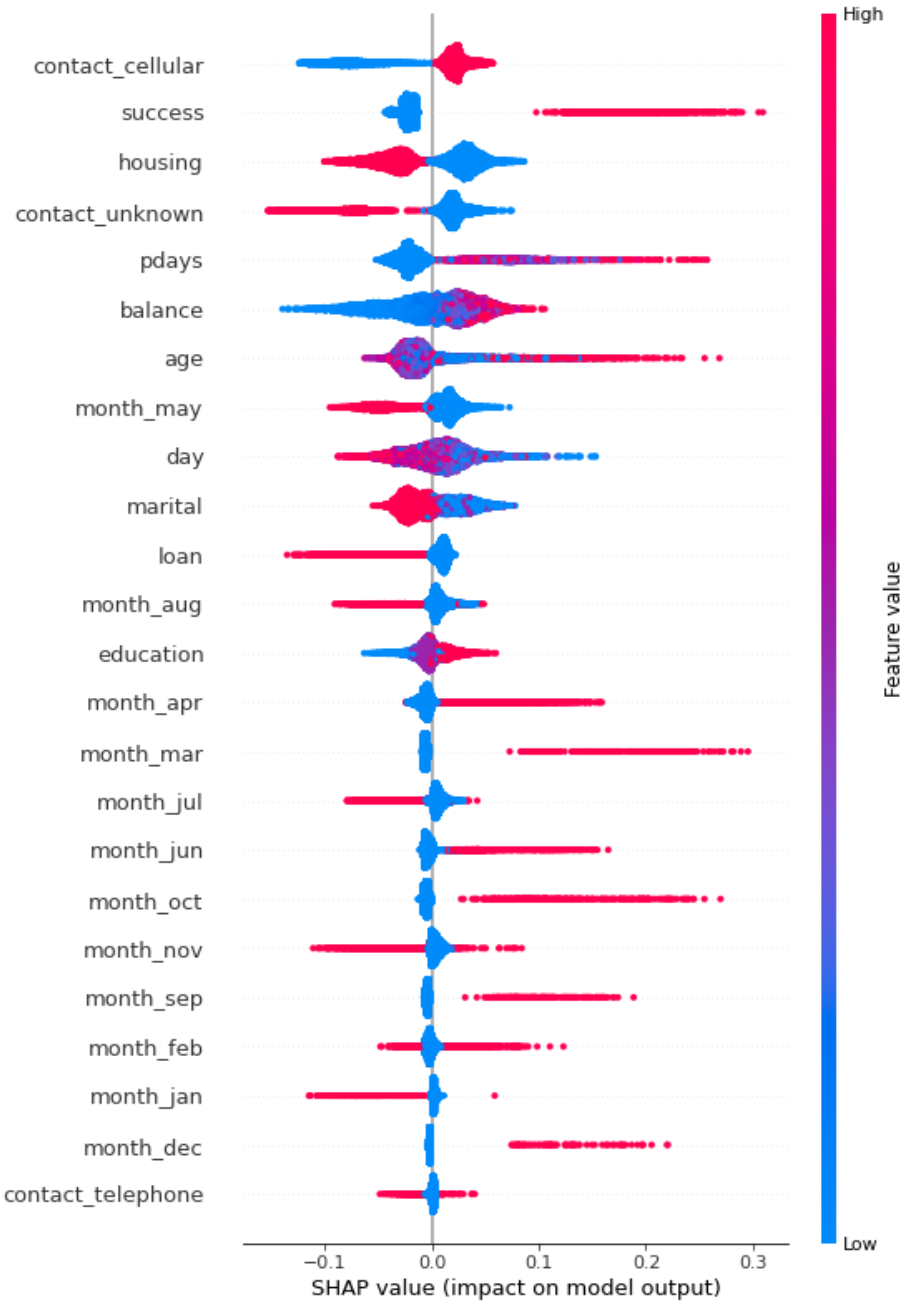


Ce graphique montre l'impact moyen de chaque feature dans le modèle. C'est la moyenne des valeurs absolue, ça veut dire que l'impact peut aussi bien être négatif que positif.

On se rend compte que dans le top, il y a contact_cellular et contact_unknown. C'est donc une information extrêmement importante. Nous ne sommes malheureusement pas en mesure de savoir ni de comprendre d'où vient le contact unknown. On a forcément le numéro du client pour l'appeler. Et en même temps, il se passe forcément quelque chose de marquant pour que l'écart de souscription soit aussi énorme.

On a ensuite le poutcome_success qui était forcément attendu dans le top.

Maintenant que nous avons une idée du classement des features, on peut rentrer dans le détail de celles-ci.



Ce second graphique affiche également les features dans le même ordre. Sauf que cette fois ci, on peut voir pour chacune d'entre elles, si l'impact est positif ou négatif en fonction de sa valeur.

Prenons l'exemple de success qui est flagrant. Lorsque success est égal à 0, il est en bleu. On peut voir qu'il tend vers la gauche. L'impact est donc légèrement négatif. Au contraire, lorsque success = 1 (en rouge), il y a un fort impact positif.

Grâce à ce graphique, on voit qu'avoir un crédit est négatif pour la prédiction ou que les meilleurs mois sont mars et octobre tandis que les pires mois sont mai et août.

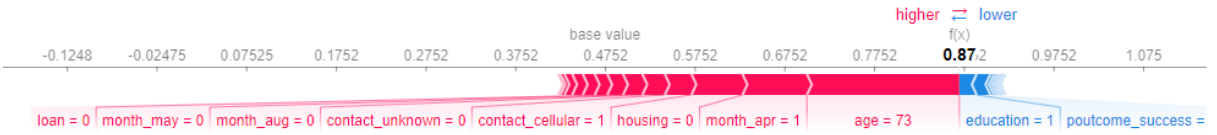
Pour certaines features, c'est beaucoup plus partagé, c'est le cas pour le mois de novembre et février qui partent des deux côtés. Cela veut certainement dire que cela peut être des bons ou mauvais mois en fonction du jour. Pour un dernier exemple, l'âge aussi est partagé. On avait vu que c'était les plus jeunes et les plus âgés qui souscrivaient le plus. C'est donc logique de retrouver ce résultat ici.

c. Interprétation locale avec SHAP

On va maintenant interpréter localement ce qui impacte la prédiction. Nous avons pris un échantillon de 5 clients avec des profils variés.

Client 1 :

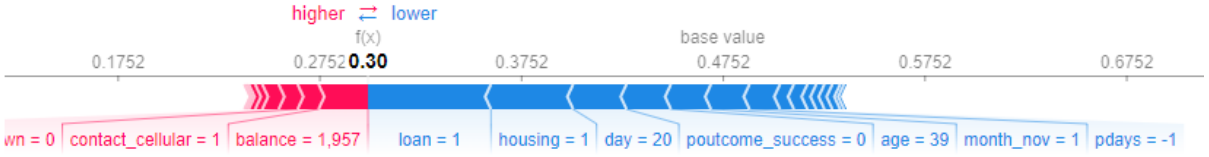
	age	job	marital	education	default	balance	housing	loan	contact	day	month	pdays	previous	poutcome	deposit
1567	73	retired	married	primary	no	253	no	no	cellular	20	apr	-1	0	unknown	yes



Les éléments les plus positifs pour ce client sont son âge, le fait de l'avoir contacté en avril et le fait qu'il n'a pas de prêt immobilier. Ce qui baisse le plus la probabilité est son niveau d'éducation primary et le fait de ne pas avoir accepté l'offre lors d'une campagne précédente. La prédiction est de 87%. C'est une bonne prédiction.

Client 2 :

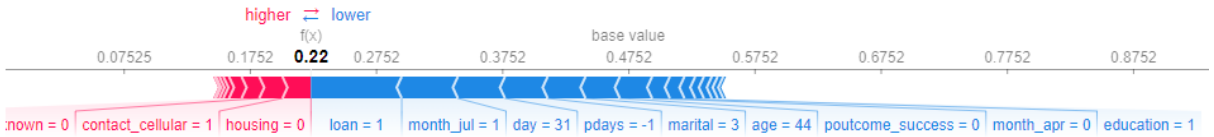
	age	job	marital	education	default	balance	housing	loan	contact	day	month	pdays	previous	poutcome	deposit
10606	39	services	divorced	secondary	no	1957	yes	yes	cellular	20	nov	-1	0	unknown	no



Le point le plus positif est la balance. En revanche il y a beaucoup de points négatifs, les deux prêts sont les plus impactants. On fait une prédiction de 30%. C'est une bonne prédiction.

Client 3 :

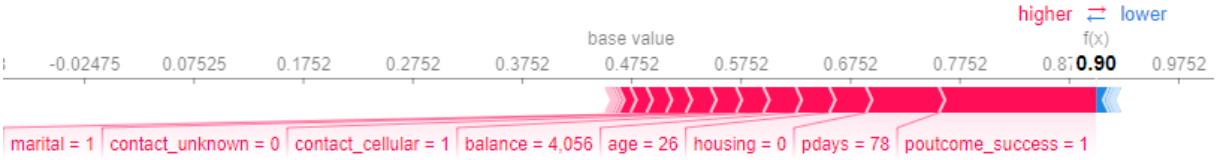
	age	job	marital	education	default	balance	housing	loan	contact	day	month	pdays	previous	poutcome	deposit
651	44	technician	married	primary	no	755	no	yes	cellular	31	jul	-1	0	unknown	yes



En point positif, il y a l'absence de crédit immobilier. On cumule pas mal de points négatifs ici, un prêt personnel, un appel lors du 31 juillet (fin de mois et mauvais mois), le client n'était pas connu avant (pdays = -1). Il y a aussi le fait d'avoir 44 ans ou d'être marié (marital = 3 ⟶ Married). A cause de tout cela, la prédiction est de 22%. C'est cependant une mauvaise prédiction.

Client 4 :

	age	job	marital	education	default	balance	housing	loan	contact	day	month	pdays	previous	poutcome	deposit
8582	26	student	single	tertiary	no	10086	no	no	cellular	8	jul	78	2	success	no



Pour ce client, il n'y a quasiment que des points positifs, il a souscrit lors de la précédente campagne, ce qui implique qu'il soit connu de la banque. Il n'a pas de crédit et est jeune. Il est au-dessus de la limite fixée de 4056 euros pour la balance. C'est vraiment le client type qui va souscrire et c'est pour cela qu'il a une prédiction à 90%. Cependant, ce client n'a pas souscrit. C'est le genre de cas, impossible à prédire correctement, c'est aussi pour ce genre de cas, que le modèle à une marge significative avec le 100%

Client 5 :

	age	job	marital	education	default	balance	housing	loan	contact	day	month	pdays	previous	poutcome	deposit
7679	40	housemaid	divorced	secondary	no	108	no	no	unknown	9	may	-1	0	unknown	no

Ce dernier client testé a très peu d'éléments en sa faveur. En revanche, il cumule les éléments négatifs. Contact unknown, contacté en mai qui est le pire mois, 108€ sur le compte etc. La prédiction est de 15% et c'est en effet une bonne prédiction car le client n'a pas souscrit à l'offre.

Même quand le modèle se trompe, les prédictions sont cohérentes. On a donc un modèle efficace au regard des cas clients que l'on peut rencontrer. En effet, il arrive que quand tout soit favorable pour une souscription, le client refuse et inversement.

Nous avons donc un modèle efficace et interprétable. Cependant, ce dernier est limité par le dataset d'origine. En améliorant le dataset, le modèle pourrait être encore plus efficace.

D) Limites du dataset et recommandation pour le futur

a. Manque de features

Le point fort et le point faible de ce dataset sont identiques. C'est le faible nombre de features. C'est un avantage, car ça rend le dataset beaucoup plus simple à préparer, il est facilement manipulable et compréhensible. En revanche, cela limite énormément les possibilités d'optimisation. C'est dommage car certaines données seraient assez simples à collecter.

- *Année*

Nous avons le jour et le mois, mais nous ne disposons pas de l'année. Cela fait une énorme différence, car il est difficile de comparer le 10 mai 1995 et le 10 mai 2020 par exemple. Il se trouve que dans ce dataset, il n'y a a priori pas un grand échantillon d'année, mais on est certains que ce dataset s'étend sur au minimum 2 ou 3 années.

- *Contexte économique*

L'année permet également de faire un lien sur le contexte économique. En effet, la capacité d'épargne des ménages est fortement impactée par ce contexte. On a des exemples récents avec la crise du Covid qui a fait augmenter l'épargne lors des couvre-feu, car les ménages ne pouvaient plus consommer autant. Au contraire, la crise inflationniste actuelle limite la capacité d'épargne des ménages. D'autant que par exemple, un DAT à 3% est rentable en cas d'inflation à 2%, mais ne l'est plus lors d'une inflation à 5%.

En plus du contexte général, on peut aussi regarder les indices en bourse, ou les taux des autres systèmes d'épargne. Les taux changent régulièrement. Le meilleur investissement d'hier n'est pas forcément celui de demain.

- *Duration de la campagne précédente*

On a remarqué que la duration était un facteur clé pour déterminer la souscription du client. On ne pouvait cependant pas l'utiliser dans le cadre de notre projet, car c'est une donnée que l'on connaît une fois que l'appel est terminé.

On pourrait par contre, stocker les durations des campagnes précédentes. Si un client nous "raccroche au nez" à chaque campagne, on peut stocker cette information avec une duration très courte. Il est inutile de l'appeler. Cela pourrait complètement changer nos prédictions et nous aider à mieux cibler certains clients.

- *Nombre de campagne précédente*

Nous avons la donnée previous. Elle nous indique le nombre total d'appels réalisés auprès d'un client. Elle ne nous indique pas le nombre de campagnes auxquelles le client à participer. Imaginons qu'on a deux client avec previous = 10. Le premier peut très bien avoir participé à une seule campagne, ce qui signifie qu'on l'a contacté 10 fois. Le second peut très bien avoir participé à 10 campagnes, ce fait que nous l'aurions contacté une seule fois dans le cadre de 10 campagnes différentes.

Notre modèle n'a pas utilisé la donnée previous car il n'a pas jugé cette information suffisamment pertinente. C'est très certainement lié à ce flou sur cette donnée. Avoir le nombre de campagnes, permettrait de connaître le ratio previous / campagne. Cette donnée serait plus pertinente.

D'autant que cela permettrait d'éviter les clients indécis. Ceux que l'on appelle plusieurs fois pour au final avoir un résultat négatif. Il est certainement plus tentant d'appeler un client qui a une chance de souscrire de 50% en un seul appel, plutôt qu'un client qui pourrait souscrire à 60% après 15 appels.

- *Taux d'endettement du client*

C'est la première donnée qui est certainement plus difficile à obtenir (si le client n'a pas son crédit dans la banque en question), mais qui pourrait avoir un fort impact sur notre modèle. Nous savons si les clients ont un crédit immobilier et un prêt personnel. Par contre, nous ne connaissons pas le taux d'endettement du client. Avoir un crédit qui représente 10% ou 30% de ses revenus n'est pas du tout la même chose.

Aussi, il pourrait être intéressant de connaître la mensualité exacte des crédits. Cela pourrait nous aider à faire la différence avec l'information suivante…

- *Revenu du client*

Nous avons la balance du client. Cela représente le solde du client à l'instant T (ce n'est d'ailleurs pas totalement vrai, nous développons ce point juste après). On peut avoir 2 clients avec une balance à 2000. Peut être que le premier gagne 2000€ par mois, tandis que l'autre en gagne 5000€. La balance n'est en rien un indicateur de revenu. Pourtant le revenu serait fortement utile pour estimer la capacité d'épargne du client. Surtout si on fait le différentiel avec le montant des crédits cité juste avant.

- *Épargne du client*

Toujours sur l'aspect financier, nous ne connaissons pas non plus le niveau d'épargne actuel du client. Que ce soit dans la banque dont il est question, voire dans des banques concurrentes (certains clients ont plusieurs banques). Cela pourrait contrebalancer le montant affiché dans balance. Un client avec une balance plus faible peut finalement avoir une épargne plus élevée.

- *Identification des clients uniques*

Il n'y a pas de doublon dans le dataset, nous avons tout de même compris que plusieurs lignes peuvent concerner un seul et unique client.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
3043	84	retired	married	secondary	no	81204	no	no	telephone	28	dec	679	1	313	2	other	yes
3380	84	retired	married	secondary	no	81204	no	no	telephone	1	apr	390	1	94	3	success	yes

On a deux clients, qui ont exactement le même travail, la même situation maritale et scolaire ainsi que la même balance. Sur la première ligne on voit que le client avait été contacté 2 fois avant la campagne. Il a été contacté une fois dans le cadre de la campagne. Sur la seconde ligne, il a été contacté 3 fois avant la campagne ce qui correspond bien au 2 + 1 précédent. Poutcome = Success sur la deuxième ligne, c'est qui correspond bien au deposit = yes de la première. Et enfin l'information qui lève complètement le doute :

Sur la première ligne, le client a été contacté le 28 décembre. Sur la deuxième ligne, pdays = 94. Si on ajoute 94 jours au 28 décembre, on tombe sur le 1er avril, date à laquelle le client a été contacté sur la seconde ligne. C'est beaucoup trop pour être une simple coïncidence d'autant que nous avons trouvé d'autres cas comme celui-ci. On a un donc plusieurs occurrences d'un même client. Il serait intéressant d'avoir un élément permettant de comprendre que l'on parle d'un même client (quelque chose qui ne permet pas pour autant d'identifier le client personnellement). On pourrait alors étudier l'évolution du comportement des clients et peut-être en tirer de nouvelles informations utiles.

Aussi, 94 jours sépare les deux lignes. Pour autant, le balance est identique pour le client, c'est d'ailleurs grâce à ça que nous avons pu l'identifier. C'est quand même étonnant d'avoir une balance identique avec 94 jours d'intervalle.

Voici un autre cas client (qui nous fait dire que le dataset comporte à minima 2 ou 3 ans. On notera aussi, que l'identifiant disponible dans le dataset, ne garantit pas l'ordre temporel. La 3ème ligne survient après la 4ème sur une échelle de temps.) :

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
2495	41	management	married	tertiary	no	27696	no	no	cellular	12	aug	147	2	104	1	failure	yes
3213	41	management	married	tertiary	no	27696	no	no	cellular	12	feb	138	2	184	3	success	yes
5092	42	management	married	tertiary	no	27696	no	no	cellular	12	may	176	2	89	5	success	yes
3870	42	management	married	tertiary	no	27696	no	no	cellular	11	aug	370	1	91	7	success	yes

Il y a exactement 364 jours qui séparent la 1ère et la 4ème ligne pour ce même client. La balance est identique sur les 4 lignes. Il est impossible que cela soit une simple coïncidence. C'est le cas pour tous les doublons que nous avons identifiés avec certitude. Cela veut dire que la balance n'affiche pas le montant à l'époque de l'appel, mais le montant actuel du client. Cela fausse énormément l'information.

Imaginons un cas volontairement extrême pour imager la situation. On a un client avec un solde négatif que l'on appelle. On le recontacte plus tard dans le cadre d'une autre campagne, sauf qu'entre-temps, il a gagné au Loto. Sur ce dataset, la balance serait un montant très élevé et identique pour les deux lignes. Même si à l'époque du premier appel, le client avait un solde négatif.

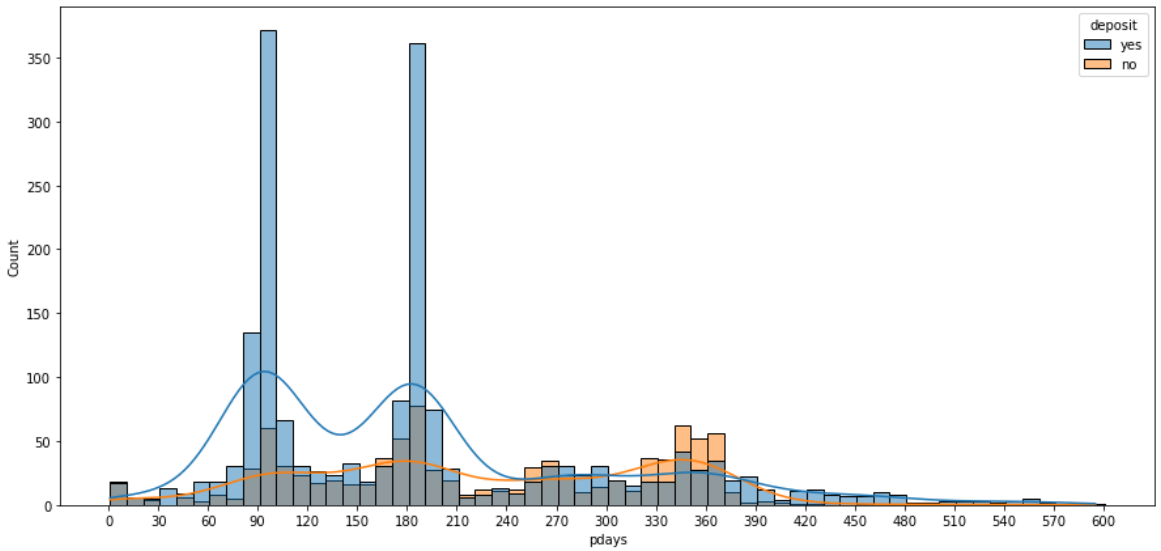
Il faut donc avoir pour chaque ligne, la balance exacte du jour de l'appel pour rendre cette feature plus pertinente et efficace.

b. Manque d'information sur certaines données

- *Contact = Unknown*

C'est la grande énigme de ce projet. Dans un véritable contexte professionnel, nous n'aurions eu qu'à contacter la banque et se renseigner sur le processus entourant de résultat. Pour qu'il y ait une différence aussi significative avec telephone/cellular, c'est qu'il se passe quelque chose de concret. En comprenant cette situation, nous pourrions peut-être changer d'angle d'attaque dans le traitement des données. Il y a aussi peut être des différenciations à faire entre les valeurs unknown.

- *Pic de contact sur pdays*



Si on reprend le graphique de pdays, on constate que le volume d'appels augmente énormément autour de 90, 180. On relève également des augmentations moins significatives autour de 270 et 360.

Ces chiffres ne sont pas anodins, chacun représente un trimestre. Depuis le début nous parlons de DAT. Parmi les facteurs clés d'un dépôt à terme, il y a la durée durant laquelle le revenu est bloqué. On peut supposer que ce sont des dépôts à terme qui durent 3 mois mais cela reste une supposition. Nous pourrions contacter la banque, pour obtenir un maximum d'informations sur le DAT.

- *Taux du DAT*

Nous ne savons pas si on parle du même produit sur toutes les campagnes. Un DAT est une offre qui est susceptible de changer régulièrement de taux. Un taux de 3% est plus intéressant qu'un taux de 2% à une même date.

Au contraire, même si le taux du DAT est toujours identique, il ne sera pas toujours aussi compétitif en fonction du contexte économique.

Le taux du DAT mis en relief avec le contexte économique, pourrait nous aider à mesurer la compétitivité de l'offre. Cette information serait très utile pour notre modèle.

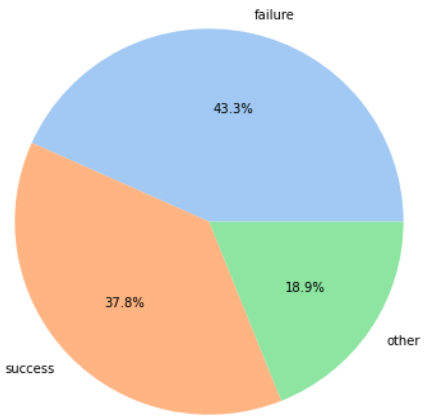
- *Politique de contact*

Le volume d'appels n'est pas également réparti entre les différents mois. Il semble qu'il y a des appels réguliers tout au long de l'année ainsi que certaines périodes de campagne durant lesquelles on a un gros volume d'appels sur un temps très resserré. Il serait intéressant de connaître la politique d'appel qui a été réalisée lors des précédentes campagnes.

- *Poutcome_faillure et poutcome_other*

Dans ce dataset, il y avait une majorité de nouveaux clients. Logiquement, plus les campagnes avancent, plus la part de clients ayant déjà participé à une campagne va augmenter. Cela va complètement changer la répartition de la variable poutcome et cela devrait faire réduire la part de unknown. Il serait donc utile d'apporter plus de précision dans les autres catégories en les découpant par exemple en nouvelle sous catégorie (motif de refus pour faillure, type de produit souscrit pour other….).

La répartition de poutcome sans les unknown.



c. Recommandation de contact

- *Contacter uniquement les clients avec une prédiction positive*

Lorsque la prédiction est supérieure à 0.5, le modèle considère que le client va souscrire. C'est pourquoi, nous recommandons de ne pas descendre en dessous.

Plus le Threshold est bas et plus la banque trouvera de clients. Cela se fera en sacrifiant la précision et donc le pourcentage de souscription lors des appels.

Si la banque souhaite se fixer un objectif de précision minimum, il est possible d'appeler les clients par "lot". On peut imaginer que la banque commence par appeler tous les clients avec une probabilité supérieure à 0.9 et note les résultats obtenus. On peut calculer la précision et continuer en baissant encore la probabilité.

A chaque nouveau lot, la précision devrait logiquement baisser. La banque pourra ainsi s'arrêter à partir du moment où elle considère que ce n'est plus pertinent de continuer.

- *Contacter les clients en début de mois*

Le jour est l'une des rares variables sur laquelle on peut avoir un impact direct. Nous pouvons choisir quand contacter le client. A ce titre, nous invitons la banque à effectuer ces campagnes d'appels plutôt en début de mois. La tendance n'est pas très significative mais elle existe. Il y a plus de chances de souscrire en début qu'en fin de mois. Cela pourrait faire basculer d'un côté ou de l'autre les indécis, ou les probabilité avoisinant les 50%. Cette tendance doit s'expliquer par la réception du salaire. Il doit être plus simple de souscrire un DAT quand le compte en banque vient d'être rempli plutôt qu'à la fin du mois.

- *Contacter les clients durant les mois favorables*

Comme le jour, on peut décider du mois du contact, même si il est évident que c'est plus simple d'attendre quelques jours avant d'appeler plutôt que quelques mois.

On a classer les mois en 3 catégories grâce à SHAP : Favorable - Neutre - Défavorable

Décembre	Janvier	Février
Mars	Avril	Mai
Juin	Juillet	Août
Septembre	Octobre	Novembre

Comme les souscriptions ont l'air efficace avec un pdays autour de 90 (soit 3 mois). Nous recommandons à la banque de diriger leurs appels sur les mois de Décembre, Mars, Juin et Septembre. Il y aura comme ça un cycle de 90 jours sur les meilleurs mois. Cette stratégie pourra évoluer en fonction des premiers résultats.

- *Se limiter à 3 appels par client*

Nous avions vu qu'il y avait un décrochage assez net dans le taux de souscription après 3 appels. Nous recommandons donc à la banque de se limiter à 3 appels par client. 1 ou 2 appels supplémentaires peuvent être fait à la marge si le commercial pense que le client va souscrire et qu'il juge donc pertinent de pousser un peu plus le nombre d'appels.

Conclusion

Le but de ce projet était de prédire les clients susceptibles d'ouvrir un dépôt à terme dans le cadre d'une campagne de télémarketing. Il fallait également être en mesure de justifier nos prédictions auprès de la banque.

Pour cela, nous avons commencé par réaliser une analyse exploratoire de notre jeu de données, nous permettant ainsi d'avoir une vision plus claire du type de client qui pourrait accepter de souscrire un contrat de dépôt à terme.

Nous avons ensuite réalisé une étude statistique sur nos données afin de déterminer si certaines d'entre elles sont indépendantes de notre target 'deposit'. Cette étude a révélé que toutes les features ont une p-value inférieure à 5% et sont donc dépendantes de 'deposit'.

Puis, nous avons observé le dataset en l'affichant dans un graphique 2D grâce aux techniques de réduction de dimension. Cela nous a permis de constater qu'il n'y avait pas de possibilité de distinguer facilement les clients qui souscrivent un dépôt à terme des autres.

Avec toutes ces informations en tête, nous avons pu entamer la création d'un modèle prédictif afin de répondre au besoin du projet. La stratégie que nous avions choisie était de lancer plusieurs algorithmes et d'observer ceux qui auraient les meilleurs résultats. Nous pouvions ainsi sélectionner le meilleur modèle et chercher ensuite à l'optimiser.

Avant cette étape, nous avons réalisé une phase de pré-processing simple afin que les modèles puissent tourner correctement. Pour cela, nous avons réalisé une cross-validation tout en choisissant les métriques qui nous permettraient d'évaluer les modèles (accuracy, precision, recall, f1, roc_auc).

Les meilleurs modèles qui en sont ressortis sont le Random Forest et SVM. L'interprétabilité du modèle étant fondamentale, nous avons décidé d'optimiser le Random Forest.

Afin d'optimiser le modèle, nous sommes ensuite repassés par une phase de pré-processing plus avancée comprenant entre autres de la feature selection, de l'imputation de valeur 'unknown' ou encore du feature engineering.

Cette phase de pré-processing nous a permis à elle seule d'améliorer toutes les métriques. Notamment le ROC AUC qui était la métrique principale à ce moment de l'analyse. Cela a également amélioré la visibilité du dataset en 2D via le PCA et permis d'identifier plus clairement des groupes de clients en fonction de 'deposit'.

Nous nous sommes ensuite penchés sur les hyperparamètres de notre modèle afin de l'optimiser. Pour trouver la meilleure combinaison d'hyperparamètres, nous sommes passés par un GridSearchCV qui inclut directement une cross-validation. Cette optimisation des hyperparamètres nous a permis d'améliorer encore une fois le score du modèle. Ce modèle est performant au vu de la difficulté à départager certains clients.

Nous devions enfin justifier nos prédictions. Pour cela, nous avons construit et affiché un arbre de décision qui se calque au maximum sur notre modèle permettant ainsi de comprendre les choix de notre modèle. Nous avons également utilisé la librairie Shap afin d'établir l'interprétation globale et locale de notre modèle. Nous sommes donc en mesure d'expliquer les facteurs qui jouent sur notre prédiction.

Enfin, nous avons émis une liste de recommandation en direction de la banque. Ces recommandations nécessitent peu d'effort pour être mises en place et permettraient d'améliorer encore les performances du modèle établi.