* Add smokenow
* Try Lime  (XGBoost explainer package)
* SVM with partial dependence plot
* Any potential for variable products ?
* Highest degree as classifier vs numeric

# Introduction et contexte

“Critical Illness” est un produit d’assurance populaire dans beaucoup de marchés qui couvre l’assuré contre les conséquences financières d’une maladie redoutée, comme le cancer ou la crise cardiaque.

Pendant l’achat d’une police de Critical Illness (CI), l’appliquant doit passer une sélection médicale, qui revoit toute condition préexistante ayant un impact potentiel sur les risques couverts, avec en conséquence soit une surprime ou un rejet de l’application. Le diabète fait partie des conditions qui en générale entraine un rejet à cause du risque important de complications, surtout cardio-vasculaires.

En même temps la prévalence de diabète a fortement augmenté les dernières 10-20 ans dans beaucoup de pays et atteint aujourd’hui xx % dans xxx. Il y a donc un fort intérêt de créer de produits adaptés aux diabètes, aidé du fait qu’un diabète est plus sensibilisé à son risque de maladie grave et donc à l’intérêt d’une assurance.

La manque d’expérience sur les diabètes et l’incertitude sur leur risque additionnel a jusqu’à présent limité le développement de produit par les assureurs.

Les assureurs ont suffisamment d’expérience sur le risque de CI pour les assurés non-diabètes, catégorisé par les dimensions de risque classique, le sexe, l’âge et souvent le statut fumeur. Pour la tarification d’un produit diabète idéalement on profiterait donc de cette expérience et on modéliserait le risque d’un diabète via son risque relatif à un non-diabète :

* : prime de risque pure pour un diabète
* : condition de CI couvert (cancer, crise cardiaque, attaque cérébrale etc.)
* : taux d’incidence de la condition j pour un risque standard (non-diabète)
* : risque relative pour la condition j d’un diabète comparé à un risque standard

Le but de ce projet est donc de modéliser , le risque relatif lié seulement au fait d’avoir du diabète, en normalisant pour tout autre dimension de risque, comme le sexe et l’age.

Dans la modélisation je me suis limité aux conditions CI les plus importantes et avec à priori le plus grand impact du diabète, la crise cardiaque et l’attaque cérébrale.

# Exploration de données

## Source de données

Aux Etats-Unis l’agence du gouvernement « Agency for Healthcare Research and Quality » (AHRQ) conduit des sondages annuels à grande échelle sur les ménages et leurs consommations de services médicaux « Medical Expenditure Panel Survey » (MEPS). Entre autres le sondage couvre les aspects suivants

* Caractéristiques démographiques
* Conditions médicales
* Consommation de services santé
* Emploie et revenues
* Couverture d’assurance

Les données du MEPS sont mis à disposition par le service IPUMS[[1]](#footnote-1), une base de données publique spécialisé sur les données au niveau individuel.

Parmi les variables très nombreuses (plusieurs centaines) une sélection de xx variables pertinentes a été effectués et un extrait récupéré couvrant les périodes 2010 à 2016. (Liste des variables en Annex A)

Les données étant basées sur des sondages, on fait face à les contraintes typiques de sondages :

* Analyse rétrospective : Idéalement on analyserait l’impact du diabète sur les incidences de critical illness futures. Avec une analyse rétrospective on ne peut pas etre sure de la direction de corrélation. Théoriquement une corrélation observée entre le diabète et une critical illness pourrait aussi être le résultat du critical illness entrainant le diabète. Une discussion avec des collègues médecin confirme que ce scenario est peu probable et qu’une corrélation devrait être le résultat du diabète influençant le risque de critical illness.
* Les réponses sont autodéclarés et peuvent donc etre susceptibles de biais, volontaire ou involontaire.

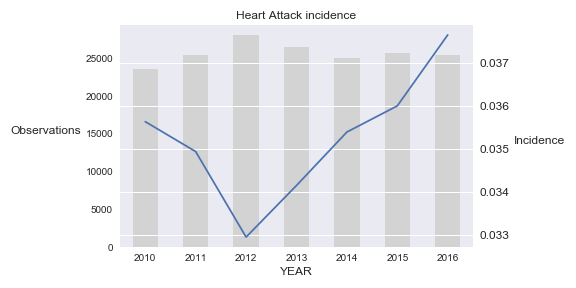
## Description des données

Au total l’extrait contient presque 250'000 observations repartit uniformément sur les 7 années.

Les variables targets sont :

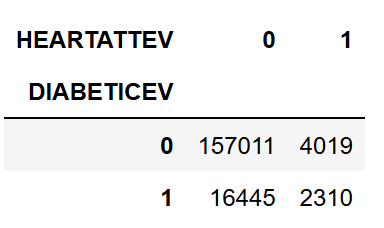
* HEARTATTEV : Existence d’une crise cardiaque dans la vie de l’individu
* STROKEV : Existence d’une attaque cérébrale dans la vie de l’individu.

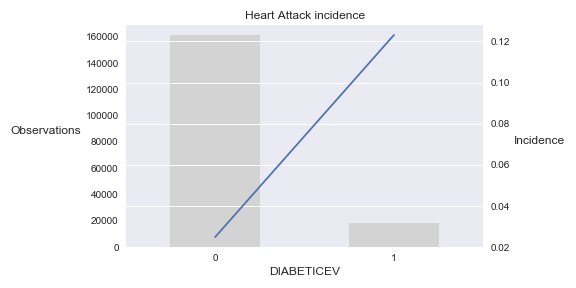
L’univers de variables de la base étant très large, pas toutes les variables sont incluses dans chaque questionnaire. Nos variables targets ne sont pas inclus dans environ 69'000 observations. Parmi les autres observations seulement pour environ 500 la réponse d’au moins une de variables target est classifié comme inconnue.   
Les observations inconnues et non-inclus sont donc enlevés, ce qui laisse environ 180'000 observations.



La moyenne générale pour HEARTATTACKEV est 3.5% avec 6'300 observations.

Notre feature principale est DIABETICEV, correspondant à l’existence d’une diagnose de diabète dans la vie de l’individu.  
Parmi les observations retenus, DIABETICEV est quasiment complète avec que très peu d’observations catégorisés comme inconnue, qui sont enlevés.





On observe un effet brute significative avec un taux d’incidence de 2.5% des non-diabètes et un taux de 12.3% des diabètes. Cependant cet effet est certainement biaisé par une population diabète plus âgée, plus défavorisée etc.

Toutes les features ont été analysés avec au moins leur distribution et leur impact sur les variables target. En générale, la qualité des données semble très haute avec que peu de variables aberrantes. Juste pour une variable (FTOTVAL) il a été décidé de borner quelques valeurs extrêmes négatives.

Autres observations :

* Tous les features sont fournies en numérique. Chaque feature qui n’est pas ordinal est classifié comme catégorique : MARSTAT, REGIONMEPS, RACEA, USBORN, INTERVLANG, FILESTATUS, WORKEV, HIDEG
* La base de données définit plusieurs catégoriques de valeur manquante, comme « inconnu », « réponse refusé », en plus des cas ou la variable n’était pas inclut dans le sondage. A cause de leur prévalences petites toutes ces catégories ont été combiné et classifié comme « unknown » pour les variables catégoriques et comme valeur manquante pour les variables numériques.
* Pour le feature « age » tous les âges au-dessus de 85 sont regroupés avec âge 85. Le feature a été laissé tel que
* Statistiques descriptives

# Preprocessing et feature engineering

## Valeurs manquantes

Comme les algorithmes ne peuvent pas gérer les valeurs manquantes il faut traiter chaque variable avec valeurs manquantes. Dans le dataset les valeurs manquantes sont déjà tous classifiés dans des catégories correspondantes (inconnu, refusé etc.) et il n’y a pas d’autres champs vides. Pour les features catégoriques les valeurs manquantes sont traités comme une catégorie spécifique, il faut donc juste traiter les valeurs manquantes pour les features numériques.

|  |  |  |
| --- | --- | --- |
| feature | valeurs manquantes | % |
| AGE | 1 395 | 1% |
| EDUCYR | 79 035 | 44% |

Seulement deux variables ont des valeurs manquantes lesquelles on impute avec le médian du feature.

## One hot encoding

Pour pouvoir traiter des features catégoriques il faut les transformer en variable numériques. On se sert de la fonction pandas Get\_Dummies(), permettant de garder les données en format DataFrame.

## Scaling numérique

Pour standardiser les features numériques et éviter que des features avec des grandes valeurs, comme par exemple FTOTVAL reçoivent un poids disproportionnel dans certain modèles, toute variable numérique est normalisée. Pour les modèles à base d’arbre ce n’est pas nécessaire, mais pour simplifier le proces on l’applique à tous les modèles. La méthode StandardScaler() de scikit-learn est utilisée.

## Déséquilibre

Dans le cas de données déséquilibrées il y a plusieures méthodes qu’on peut considérer pour équilibrer les données et assurer que chaque classe est bien modélisée, comme le sous-échantillonnage ou le sur-échantillonnage. Dans notre cas il est important que le modèle soit bien calibré, au moins pour les différentes classes du target et de DIABETICEV. Le besoin d’équilibrer les données sera revu pour chaque modèle.

# Métriques d’évaluation

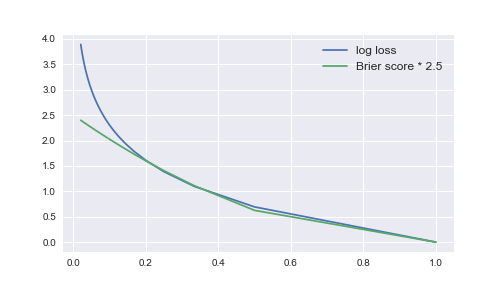
Dans la sélection de modèle plusieurs aspects sont considérés importants :

1. Précision de prédiction : La précision de prédire le risque de CI, sachant que le but est moins de pouvoir classifier les observations dans deux groupes, mais plutôt d’avoir des probabilités exactes pour application dans la tarification. De toute façon comme on modélise des probabilités assez basses c’est peu probable de pouvoir identifier individuellement les personnes avec une condition CI.
2. Calibration : En plus de l’objective d’avoir une bonne prédiction sur l’ensemble de données, on souhaite d’avoir un modèle bien calibré à travers des sous-groupes pertinentes, par exemple par sexe ou par tranche d’âge
3. Interprétabilité : Le but principale du modèle n’est pas de prédire la condition de CI mais de modéliser l’impact du diabète sur la condition de CI. On nécessite donc un modèle qui permet d’identifier les features importants et mesurer leur impact.

## Précision de prédiction

Vu l’importance des probabilités on considère seulement des métriques qui prennent compte de la probabilité prédite :

* Log-loss : Dans sa mesure de précision le log-loss tient compte de la confiance de prédiction. Un faux positif est donc pénalisé plus si la probabilité de prédiction est élevée. Dans notre situation avec une classification binaire le log-loss est définit comme :  
  log-loss(y,p) =   
  pour un ensemble de prédictions probabilités pi et vraie valeurs yi. On utilise le logarithme naturel.
* Brier score : Similairement au log-loss le Brier score tient aussi compte de la différence entre la probabilité prédite et la vraie erreur. Le Brier score pénalise l’erreur au carré :   
   BS(y,p) = .   
  Le Brier score prend que des valeurs entre 0 et 1 à la différence du log-loss qui peut tendre jusqu’à l’infini. Une autre variante du Brier score, le Brier skill score permet d’évaluer la performance relative à un modèle de référence.

Dans la comparaison on voit que la pente du Brier score est très similaire au log-loss à part pour les probabilités très petites. Pour faciliter la comparaison le Brier score a été multiplié par 2,5.

Si on souhaite utiliser le résultat de la métrique pour juger la performance du modèle il faut faire attention dans le cas de données déséquilibrés, car les métriques sont symétriques et ne prennent pas compte d’un déséquilibre. Du coup la prédiction naïve (constant 0) peut entrainer un « bon » score. Dans notre situation on s’en sert de la métrique surtout pour comparer la performance de modèles entre-elles et on évalue la bonne calibration avec d’autres critères.

## Calibration

Etant donné un problème de classification binaire, le focus des méthodes est la séparation des observations dans deux groupes, sans forcement bien représenter la structure de probabilité sous-jacente. Dans la pratique cependant il ne suffit pas d’être correcte en moyenne, mais c’est aussi important que les sous-groupes soient bien calibrés pour éviter des biais ou des subventions croisées.

* Calibration plot : Le graph de calibration, parfois aussi appelée « reliability curve » permet de visualiser si les probabilités sont bien calibrées. Par exemple pour un groupe d’échantillons avec une probabilité prédite autour de 0,6 on s’attend qu’environ 60% de l’échantillon appartienne à la classe prédite.   
  Le graph de calibration regroupe les observations suivant leur probabilité prédite et oppose le taux moyen du groupe au vrai taux moyen observé.
* Biais de sous-groupes : Le graph de calibration analyse le biais du modèle, regroupant les observations par quantile. De même on peut analyser le biais à travers d’autres axes. Dans notre situation c’est surtout important que les features qui contribuent le plus à la performance et qui sont retenus pour la tarification soient bien calibrés.

## Interprétabilité

Le but du projet est de modéliser l’impact du diabète sur les conditions CI ; le ou les modèles choisis doivent donc permettre de mesurer la contribution d’un feature individuel sur le résultat final du modèle.

En règle générale plus la complexité du modèle machine learning augmente plus l’interprétabilité réduit. Il y a que peu de modèles qui fournissent directement l’impact d’un feature, comme le modèle linéaire, et on est souvent amené à des méthodes indirectes pour quantifier l’impact :

* Feature importance: Une méthode pour connaitre l’importance que chaque feature apporte à la performance prédictive du modèle est la méthode de « feature importance ». L’idée est de mesurer pour un feature donné la dégradation du modèle si le feature est ignoré ou remplacé par des permutations arbitraires. Par ex. pour les méthodes basés sur les arbres on somme les gains de performance à chaque nœud qui est basé sur le feature donné, pondéré par le nombre d’observations dans le nœud.  
  La méthode permet d’identifier facilement l’ordre d’importance des features et de sélectionner les features les plus pertinents.
* Partial dependence plot: Comme l’indique son nom, le but d’un graph de dépendance partielle est de visualiser la structure de dépendance marginale du modèle sur un (ou plusieurs) feature(s) donné(s). Dans le cas d’un feature, on fixe le feature avec un nombre limité de valeurs et pour chaque valeur prend la moyenne de la fonction de prédiction à travers les autres features.  
  En plus générale, supposant que est l’ensemble de features évalué et que est l’ensemble des autres features, alors la fonction de dépendance partielle est définie par[[2]](#footnote-2) :

Comme c’est difficile de visualiser plus que deux dimensions à la fois, en générale le graph de dépendance partielle est utilisé pour analyser les features les plus importants. Il est surtout utile pour visualiser la structure de dépendance (monotone, linéaire etc.), mais en même temps il correspond aussi au risque relatif cherché :

avec x = DIABETICEV

L’inconvenance de la dépendance partielle est que son calcul suppose l’indépendance entre et , car on prend la moyenne sur indépendamment de la valeur de donnée, qui peut donc être mal pondérée. Dans notre situation l’hypothèse d’indépendance n’est pas satisfaite et il faut donc faire attention avec l’interprétation des résultats.

* Individual Conditional Expectation
* Accumulated Local Effects Plot

# Modélisation

Dans les étapes de modélisation suivantes le package de Python scikit-learn **sklearn** est utilisé, sauf si mentionné différemment.

## Calibration et évaluation

Pour tester la performance des modèles et les comparer entre eux, les données sont séparées en deux avec un tirage aléatoire, dans un set de données train/validation pour l’entrainement et la calibration du modèle et un set de données test pour tester la performance des modèles finaux avec des données vierges. Un ratio standard de 75% (train) / 25% (test) est appliqué.

La calibration de chaque type de modèle est effectué avec une validation croisée k-fold. Ceci permet d’avoir une meilleure estimation de la performance et alors de la qualité de calibration du modèle. Pour un compromis entre précision et temps de calcul de la modélisation on utilise 5 folds.

Comme la variable à prédire dans notre dataset n’est pas bien équilibrée on applique la méthode de validation croisée stratifiée **StratifiedKFold()** qui assure que le même ratio des deux classes est présente dans les données de train et de test.

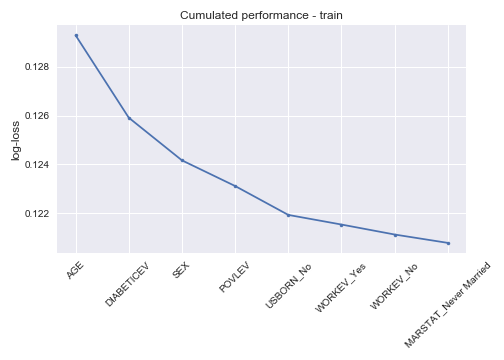
## Logistic Regression

La régression logistique est un modèle de classification linéaire qui modélise les probabilités des résultats avec une fonction logistique.  
Pour réduire le risque de sur-apprentissage des techniques de régularisation sont adoptés, ajoutant un cout à la fonction minimisé basé sur une norme sur les coefficients utilisés. Dans ce projet on se limite aux normes L1 et L2.

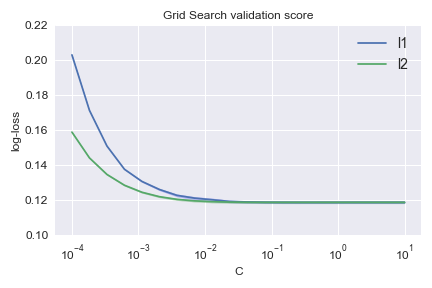
Pour une première analyse une régression séquentielle est effectuée. A chaque itération le feature qui améliore le plus le log-loss de la prédiction du modèle est ajouté. Les features les plus performants sont :

1. AGE
2. DIABETICEV
3. SEX
4. FTOTVAL
5. USBORN\_No
6. FILESTATS\_Single
7. WORKEV\_Yes
8. WORKEV\_No

Les trois premiers sont les facteurs de risque classiques.



On continue avec une recherche grid pour le modèle sur tous les features en optimisant sur les paramètres principaux, la norme de pénalisation (l1 ou l2) et l’inverse de la force de régularisation (C). Le log loss négative est utilisé comme métrique de validation.



La performance stabilise à partir d’un C autour de 0,1 et ne dégrade pas avec des pénalisations plus faibles, ce qui semble indiquer un faible risque de sur-apprentissage. Le choix de la norme a aussi qu’un faible impact, du coup l’approche ElasticNet qui permet de combiner les deux normes n’est pas exploré.

L’écart-type des résultats est très faible au point que ce n’est pas visible dans le graphique.

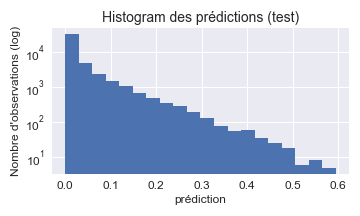
### Précision

La précision du modèle avec le paramétrage optimale (norme l1, C = 0.15) est :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modèle | Cross-validation  Log loss | Cross-validation  Brier score | Test  Log loss | Test  Brier score |
| LogReg(l1, C=0.15) | 0.1185 | 0.0305 | 0.1248 | 0.0325 |

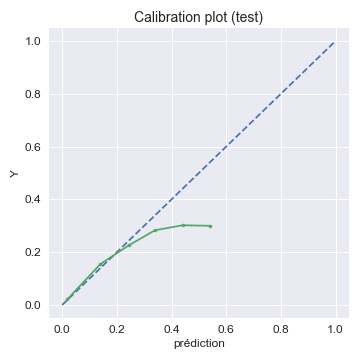
### Calibration

L’histogramme des probabilités prédites montre bien la nature du problème, probabiliste et déséquilibrée. La majorité des prédictions a une probabilité en dessous de 0.05 et il y a que très peu d’observations avec une probabilité au-dessus de 0.5



La comparaison des prédictions avec la réalité sur les données de test montre que le modèle est raisonnablement bien calibré sur les probabilités basses en-dessous de 0.25 avec une déviation autours de 10%, mais reflète moins bien les observations plus risquées. Il faut cependant tenir compte du fait que les classes avec probabilité élevée contiennent que peu d’observations.

Add second axis with ratio (or histogram)



### Interprétabilité

L’avantage de la régression est qu’elle fournit l’impact logistique de chaque feature sur la variable prédite et donc permet une interprétation facile du modèle. Ci-dessous en exemplaire les coefficients des features de la régression séquentielle :

|  |  |
| --- | --- |
| Feature | Coefficient |
| AGE | 2.35 |
| DIABETICEV | 2.55 |
| SEX | 0.44 |
| FTOTVAL | 0.88 |
| USBORN\_No | 0.65 |
| FILESTATS\_Single | 0.73 |
| WORKEV\_Yes | 0.99 |

Le risque relatif du diabète est donc 2.55.

Add graph with coefficient by regularization-strength

Impact of penalisation on coefficients?

## Gradient Boosting

Gradient (Tree) Boosting est une méthode de modélisation du type « ensemble », basé sur le principe d’améliorer la performance en combinant plusieurs instances d’un modèle simple, des apprenants faibles (« weak learner »).   
Pour un modèle de Gradient Boosting (GBM), en général on utilise comme modèle de base un arbre de décision peu profond. A chaque itération le modèle est amélioré en ajoutant un autre arbre calibré sur les résidus du modèle précédent.   
La complexité de l’arbre de décision utilisé pour chaque apprenant est contrôlée via plusieurs paramètres, comme la profondeur de l’arbre ou le nombre minimale d’observations dans chaque nœud.  
Le modèle GBM est à risque de sur-apprentissage et une technique de régularisation est adopté (« shrinkage »), limitant la contribution de chaque nouvel apprenant avec un poids faible (« learning rate »).

Les paramètres optimaux sont trouvés via une recherche grid sur les paramètres principaux, basé sur la métrique de performance du log loss. On retient un arbre de profondeur (« max\_depth ») 6, avec un minimum de 20 observations par nœud (« min\_samples\_leaf ») et une learning\_rate de 0.1 avec 100 itérations.   
La technique de « bagging » qui augmente la diversité des apprenants en les limitant à des sous-échantillons de données, n’a pas eu d’impact significatif sur la performance du modèle.

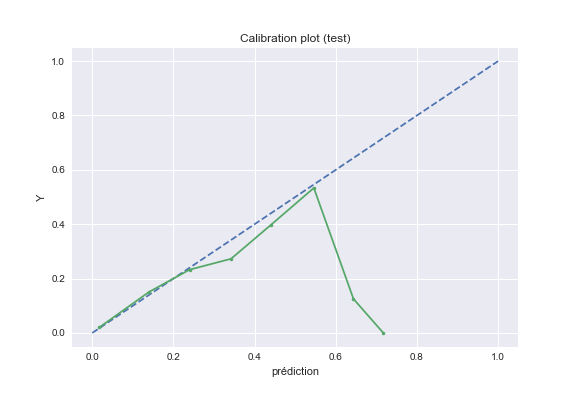
### Précision

La précision du modèle avec l paramétrage optimale est :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modèle | Cross-validation  Log loss | Cross-validation  Brier score | Test  Log loss | Test  Brier score |
| GBM(max\_depth = 6, min\_samples\_leaf=20, learning\_rate=0.1, n\_estimators=100) | 0.1162 | 0.0302 | 0.1219 | 0.0321 |

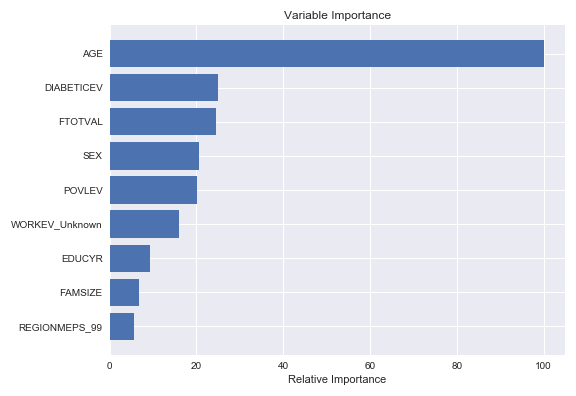
### Calibration

Le « calibration curve » montre une calibration raisonnable à travers tous les classes de risque et semble capturer mieux les risques élevés que le modèle linéaire. Les classes au-dessus de 0.6 contiennent que 10 observations.



### Interprétabilité

Scikit-learn fournit les « feature importance » du modèle. Les dix premiers donnés ci-dessous montrent que l’âge est de loin le feature le plus important est que la plupart des variables importantes sont les mêmes que pour le modèle linéaire, entre autres DIABETICEV. Des différences sont l’apparition du degré de pauvreté (POVLEV) et la disparaisson de FILESTATUS et USBORN.

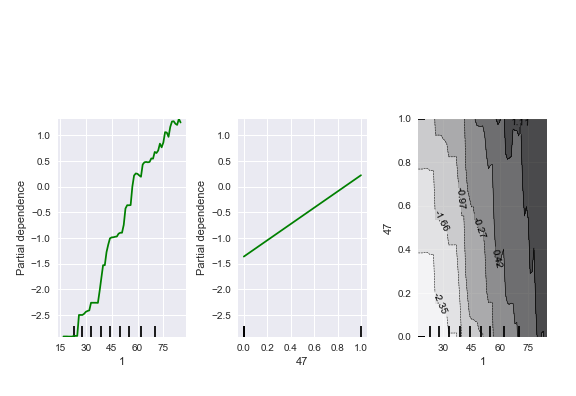


Pour les modèles basés sur des arbres de décision le package scikit-learn inclut la fonctionnalité de produire un partial dépendance plot, qui réutilise les arbres déjà construites et permet donc un calcul rapide.   
Le résultat produit montre l’impact de changer les valeurs pour le feature en question, relative à la prédiction initiale (correspondant au model naive de moyenne) :

partial dependence = sum(tree\_predictions \* learning\_rate)

Ca permet d’analyser la structure de dépendance du feature, mais ne permet pas de revenir aux prédictions mêmes, dont on aurait besoin pour en déduire le risque relative.

Les graphs ci-dessous montrent le « partial dependence » individuel des deux features AGE et DIABETICEV et leur « partial dependence » jointe. Comme on a un problème de classification, les résultats sont transformés par la fonction logit, ce qui explique les résultats négatifs :



On ne constate pas de surprises avec un effet d’âge monotone et aussi une certaine interaction entre l’âge et le diabète avec un risque plus élevé si les deux features sont élevés.

Les résultats pour le statut diabète ne permettent malheureusement pas de conclusion sur le risque relative.

## XGBoost

XGBoost est un package indépendant, séparé de scikit-learn, qui contient plusieurs optimisations de la modélisation « gradient boosting ». La méthodologie est similaire à GBM, mais on l’effectue quand-même pour revoir la stabilité des résultats du « gradient boosting » et explorer d’autres méthodes d’interprétation des résultats.

Similaire que pour le modèle GBM on effectue une recherche grid pour trouver les hyperparamètres optimaux, qui sont assez similaire aux paramètres du GBM, avec une profondeur d’arbre (« max\_depth ») de 5, un minimum d’observations par nœud (« min\_child\_weight ») de 10 et un learning\_rate de 0.1 avec 100 itérations. Encore une fois les paramètres d’échantillonnage (« bagging ») n’ont pas d’impact significatives.

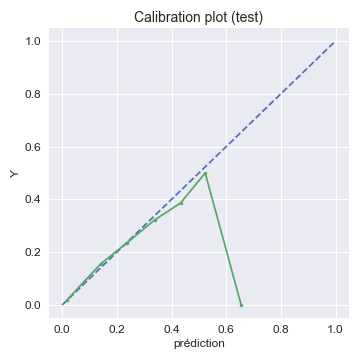
### Précision

La précision du modèle avec l paramétrage optimale est :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modèle | Cross-validation  Log loss | Cross-validation  Brier score | Test  Log loss | Test  Brier score |
| XGB(max\_depth = 5, min\_child\_weight=10, learning\_rate=0.1, n\_estimators=100) | 0.1161 | 0.03015 | 0.1220 | 0.03211 |

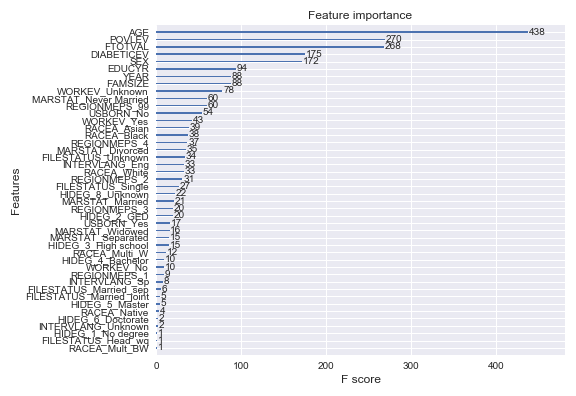
### Calibration

Le « calibration curve » montre une bonne calibration à travers tous les classes de risque, plus consistent encore que le modèle GBM :



### Interprétabilité

XGBoost fournit aussi une analyse de « feature importance »



Dans la représentation de XGBoot les variables numérique POVLEV et FTOTVAL ont une importance plus élévée que pour les algorithmes précedents.

Check scaled / unscaled !

## Random Forest

La méthode des forêts aléatoires ( « Random Forest ») se base aussi sur l’idée de combiner plusieurs apprenants faibles du même type pour une prédiction agrégée améliorée. Au lieu d’ajuster le modèle itérativement comme le « gradient boosting », les arbres aléatoires sont construits indépendamment et après combiné via un vote moyen. Pour réduire la corrélation des apprenants entre-eux et avoir une meilleure dispersion, l’arbre de décision n’est pas construit sur tous les features. Pour chaque arbre, à chaque coupage de nœud, un sous-ensemble de features est sélectionné par hasard et le découpage optimale est défini parmi ce sous-ensemble.

Comme pour les autres algorithmes on effectue une recherche grid sur les paramètres principaux. Ainsi la profondeur d’arbre (« max\_depth ») est définie à 12 et le nombre minimal d’observations par nœud (« min\_samples\_leaf ») à 10. La taille du sous-ensemble de features considéré à chaque coupage de nœud suit la règle « racine carré du nombre de features », souvent conseillé pour les modèles de classification.

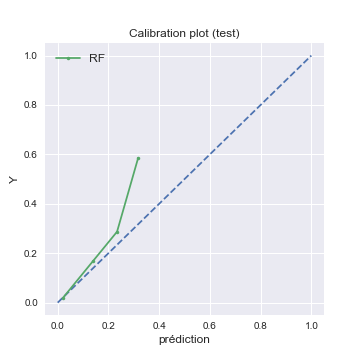
### Précision

La précision du modèle avec l paramétrage optimale est :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modèle | Cross-validation  Log loss | Cross-validation  Brier score | Test  Log loss | Test  Brier score |
| RF(max\_depth = 12, min\_samples\_leaf=10, max\_features=auto, n\_estimators=300) | 0.1160 | 0.03009 | 0.1215 | 0.03195 |

### Calibration

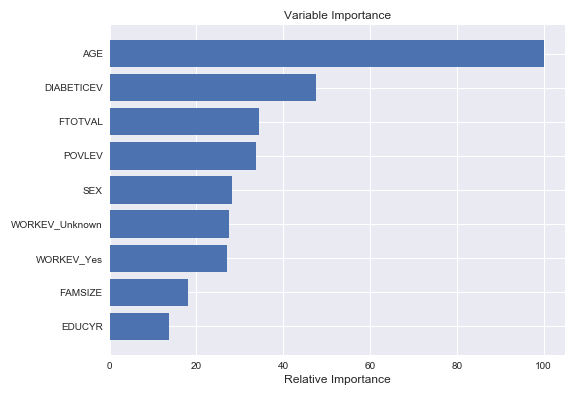
Le « calibration curve » montre que le modèle des forêts aléatoires modélise mal les risques élevés. Il prédit très peu d’observations avec un risque au-dessus de 0.3 et sous-estime tous les risques au-dessus de 0.1



### Interprétabilité

Sklearn pour l’instant supporte la fonctionnalité d’analyse « partial dependence » que pour l’algorithme de « gradient boosting » et pas pour les forets aléatoires.

Sklearn supporte les forets aléatoires pour l’analyse de « feature importance » qui, par défaut, pour les problèmes de classification est basé sur la réduction de diversité mesuré par l’indice Gini. Pour chaque « feature », on prend la moyenne à travers tous les nœuds et arbres.   
La liste et l’ordre des features pertinents est très similaire à celle du GBM.



## Evaluation

### Stabilité

Pour évaluer le potentiel d’améliorer la performance des modèles avec plus de donnés on prépare le graph « learning rate » qui montre l’évolution de la performance d’un modèle avec l’augmentation de nombre d’observations, partant d’un sous-échantillon des observations train données. Ci-dessous comme exemple le « learning rate » du modèle des forets aléatoires. On observe que la performance (log-loss) se stabilise et on s’attend pas à une amélioration importante si on ajouterait plus d’observations.



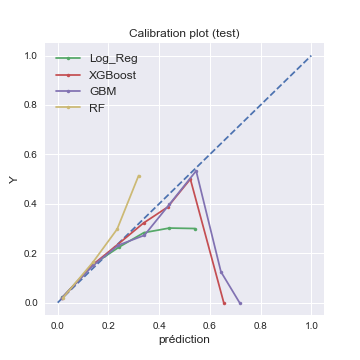
### Comparaison

Pour la comparaison de performance des modèles on ajoute un modèle naive, qui prend comme prédiction constante la moyenne du target. On ajoute le Brier skill score avec le modèle naive comme modèle de référence.

On observe un léger avantage des forets aléatoires. La performance du GBM est très cohérent avec celui de XGBoost, étant basé sur le memes principes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ordre | Modèle | CV  Log loss | CV  Brier score | Test  Log loss | Test  Brier score | Test  BSS |
| 1 | **RF**(max\_depth = 12, min\_samples\_leaf=10, max\_features=auto, n\_estimators=300) | 0.1160 | 0.03009 | 0.1215 | 0.03195 | 10.1% |
| 2 | **GBM**(max\_depth = 6, min\_samples\_leaf=20, learning\_rate=0.1, n\_estimators=100) | 0.1162 | 0.03023 | 0.1219 | 0.03205 | 9.8% |
| 3 | **XGB**(max\_depth = 5, min\_child\_weight=10, learning\_rate=0.1, n\_estimators=100) | 0.1161 | 0.03015 | 0.1220 | 0.03211 | 9.7% |
| 4 | **LogReg**(l1, C=0.15) | 0.1185 | 0.03058 | 0.1248 | 0.03246 | 8.7% |
| 5 | **Average** |  |  | 0.1581 | 0.03555 |  |

En termes de calibration, l’algorithme des forêts aléatoire est le moins performant et les algorithmes de « gradient boosting » réussissent le mieux à représenter les différents niveaux de risque.



# Conclusion et application

q

Annex A – Data extract variables

Variable / description / type / codes / #missing

Annex B – Bibliographie

“Interpretable Machine Learning - A Guide for Making Black Box Models Explainable” Christoph Molnar, 2018

“Elements of Statistical Learning”

1. <https://meps.ipums.org/meps/> [↑](#footnote-ref-1)
2. (1) Reference [↑](#footnote-ref-2)