Rapport final projet 2 NF26

Emmanuelle Lejeail, Capucine Prudhomme

22 juin 2018

1 Modélisation du problème

Durant ce projet, nous souhaitons étudier les résultats des votes aux élections législatives en France, que ce soit concernant le taux d'abstention ou bien les partis élus. Nous souhaitons de plus corréler ces résultats avec certains facteurs (certains socio-économiques), tels que la géographie, les revenus moyens des habitants, le taux de chômage, le degré d'éducation, ou bien la "couleur" de la commune, c'est à dire le parti politique du maire.

Afin de modéliser cette problématique, nous avons créé un modèle UML pour représenter les liens entre les entités de notre problème selon les informations dont nous disposons. Nous avons identifié 3 dimensions, qui sont le temps (année, tour), la géographie (bureau de vote, commune, département, région) et candidat (candidat, parti) avec au centre le résultat d'un candidat à une élection pour un bureau de vote donné. Cependant, les critères socio-économiques dépendant à la fois de la géographie et du temps, ces informations n'appartiennent pas directement à une dimension. Nous nous retrouvons avec le schéma en étoile cyclique suivant, résumant la situation :

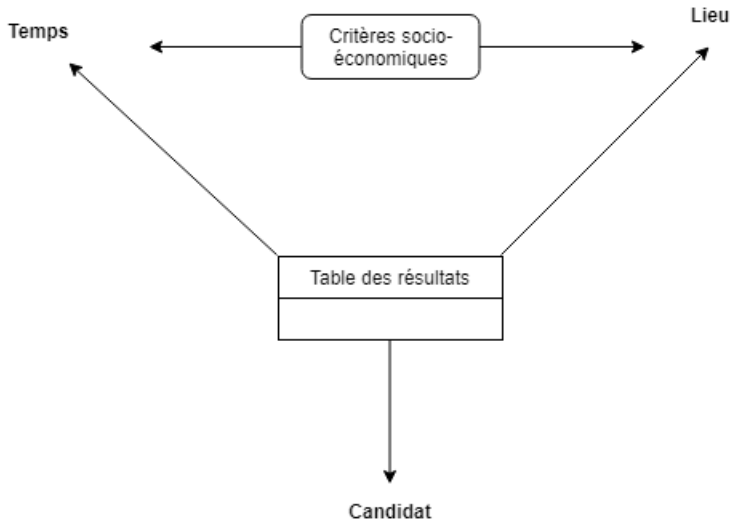

FIGURE 1 – Schéma en étoile de la situation

La modélisation UML complète est visible plus bas dans la figure 2.

Nous devons pour cela prendre en compte à la fois les résultats du premier et du second tour des élections, entre 1990 et aujourd'hui.

Nous disposons pour cela de données fournies par le ministère de l'intérieur, sous forme de fichiers csv ou txt, un fichier correspondant à un tour ou deux d'une élection selon les années. Ces fichiers ne sont pas tous organisés de la même manière, ils ne contiennent pas exactement les mêmes informations, nous devons donc organiser et adapter nos tables et notre récupération de données pour pouvoir traiter tout cet ensemble de données et arriver à des résultats nous permettant de comparer l'ensemble des élections. Cependant, pour l'ensemble de ces fichiers, les résultats sont répartis par bureau de vote, ce qui nous permet de prendre le bureau de vote comme granularité la plus petite. De ces fichiers nous pouvons retrouver des informations sur le nombre d'inscrits par bureau de vote, le nombre de votes blancs, d'abstentions, les candidats, leur sexe ainsi que leur nombre de votes.

Nous disposons aussi de données sur les caractéristiques sociales des habitants par commune (fichier venant de l'INSEE), les informations étant regroupées dans le même fichier : Revenu médian par commune, nombre de cadres, agriculteurs, (etc.), nombre de chômeurs. Puisque ces données sont par commune, la donnée sera redondante sur les bureaux de vote d'une même commune si nous les ajoutons à la même table qui à pour granularité la plus petite le bureau de vote, puisque nous répéterons ces informations sur chaque ligne.

Il nous faudra en plus récupérer un fichier de l'INSEE nous permettant de lier les départements à une région afin de pouvoir réaliser des statistiques à cette échelle géographique. Nous allons organiser et recouper l'ensemble de ces données pour pouvoir les analyser et en tirer des conclusions.

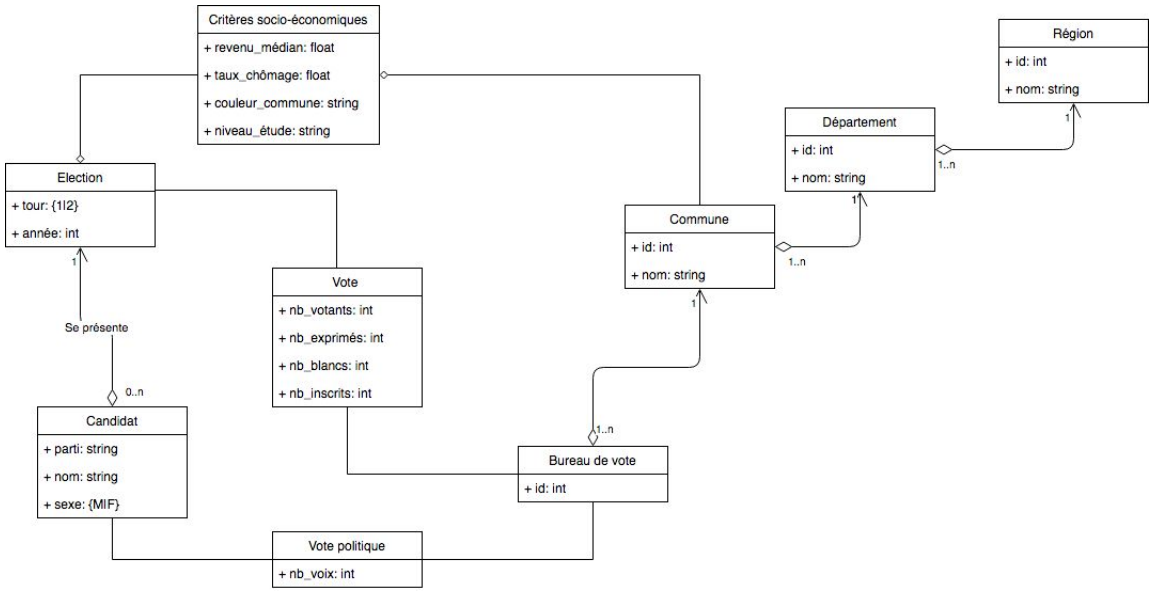

FIGURE 2 – Modélisation des entités de notre problème.

2 Matérialisation des données

La matérialisation consiste en la création des tables qui accueilleront nos données. Nous utilisons un modèle non relationnel, nous ne prêtons donc pas trop attention à la redondance des données dans le sens où cette redondance peut être là dûe à des choix stratégiques nous amenant vers plus d'efficacité pour répondre aux questions que nous nous sommes posées. Nous avons une base de données en colonnes, certaines colonnes seront parfois vides pour certaines entrées, il conviendra lors de tri sur les partitions concernées d'éviter les analyses se servant de ces colonnes comme clé de tri.

Nous avons fait le choix d'élaborer 4 tables, chacune permettant de répondre à des besoins différents. Les 4 tables ont pour clé de partitionnement l'ensemble `(année, tour)` afin de pouvoir avoir des résultats pour chaque élection et ensuite pour chaque tour. Le tour ici correspond à 1 ou 2 selon que l'élection était du premier ou deuxième tour. Tandis que l'année correspond bien entendu à l'année lors de laquelle a eu lieu l'élection.

Pour la première table, la **clé de tri** est `(Code Région, Code Dpt, Code commune, Code bureau de vote)` afin de procéder à un tri géographique, et de pouvoir remonter à tous les niveaux de granularité de cette géographie. L'objectif de cette table est de permettre un calcul facilité du taux d'abstention (au taux de vote blanc) à tous les niveaux de granularité géographique. Ces calculs sont faciles puisque dans cette table, il n'existe qu'une seule ligne par bureau de vote d'une commmune, d'une région donnée. Nous récupérons des données relatives au nombre d'inscrits, de votants et de votes blancs (ou votes exprimés). Grâce à cette table nous pouvons donc en pratique calculer le taux d'abstention dans un département particulier ou une région donnée, ou la proportion de votes blancs dans une commune, etc. Cette matérialisation est visible dans le schéma ci-dessous :

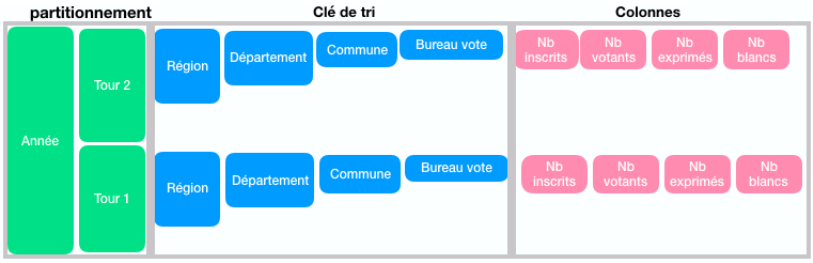

FIGURE 3 – Matérialisation d'une table de notre datawarehouse.

Pour la seconde, la **clé de tri** est `(parti, (Code Région, Code Dpt, Code Commune, Code bureau de vote))`. Nous voulons d'abord trier les informations par parti politique, puis ensuite permettre un tri géographique selon le niveau de granularité auquel nous désirons nous intéresser. Nous récupérons les colonnes suivantes en plus des informations situées dans les clés de tris et partitionnement : nombre de votants d'un bureau de vote, nombre de votes pour le candidat du parti situé dans la clé de tri et sexe du candidat. Ceci nous permet d'obtenir des résultats par parti politique, et donc par exemple de déterminer les régions (ou départements, communes) où un parti est le plus représenté.

Pour la troisième, la **clé de tri** de la table est `(parti, (Code Région, Code Dpt, Code Commune))`. Pour cette table, nous arrêtons notre clé de tri à l'échelle de la commune puisque son objectif est de pouvoir répondre à des questions sur le vote politique selon des critères socio-économiques. Ces critères existant uniquement à l'échelle de la commune et non du bureau de vote, nous avons choisi de les inclure dans une table différente de la seconde pour marquer une différence sémantique bien qu'il aurait été possible de les stocker dans la même table. Nous procédons toujours par un partitionnement par

tour et année, et par un tri par parti puis géographique. Cette nouvelle table nous permet de gagner en cohérence et en "séparation métier". Les colonnes de cette table sont les suivantes (en plus des colonnes déjà mentionnées dans la clé de tri et de partitionnement) : bureau de vote (si il y a), le nombre de votants (d'un bureau de vote ou de la commune directement), le nombre de voix pour un candidat, le sexe du candidat, le taux de chômage de la commune, le revenu médian de la commune, le parti du maire de la commune et le niveau d'étude moyen de la commune.

Enfin, pour la dernière table, la **clé de tri** est la suivante : `(Code Région, Code Dpt, Code Commune)`. Cette table, tout comme la précédente nous permet de faire des analyses sur les facteurs socio-économiques mais en les liants cette fois-ci au taux d'abstention ou de vote blanc d'une commune, d'un département ou d'une région. Ainsi cette table possède les colonnes suivantes : code du bureau de vote (éventuellement), nombre d'inscrits (dans la commune ou le bureau de vote), nombre de votants (dans la commune ou le bureau de vote), nombre d'exprimés(dans la commune ou le bureau de vote), nombre de votes blancs (dans la commune ou le bureau de vote), taux de chômage de la commune, revenu médian de la commune, niveau d'étude moyen de la commune et le parti du maire.

Une représentation graphique des tables est disponible en annexe A.

Pour conserver la scalabilité horizontale, nous avons décidé de faire des partitions dont la taille n'augmente pas avec le temps, de taille bornée. Ainsi, nous avons choisi année, tour comme clé de partitionnement. Dans tous les cas, même si le nombre de tours par an augmentait, la taille de la partition reste fixe.

Nous préférons récupérer le nombre d'inscrits, nombre de votants, nombre d'exprimés plutôt que directement les taux d'abstentions ou taux de blanc, parfois directement disponibles dans les fichiers de données, pour une question de maintenance. En effet, tous les fichiers, quel que soit le tour ou l'année, disposent des informations sur ces nombres, mais tous ne disposent pas directement du taux précalculés.

Nous avons donc décidé de ne pas stocker celui ci, pour ne pas avoir de nombreux champs vides alors que nous avons déjà l'information (mais non calculée) dans la table grâce aux nombres bruts. Nous décidons de plus de calculer les taux par MapReduce lors de la phase d'analyse, plutôt qu'au moment de l'alimentation de la base, afin de limiter les opérations en amont. Nous n'effectuerons les opérations que lorsque cela sera nécessaire, sur les données voulues. Il est à noter que les colonnes "exprimés" ou "blancs" seront parfois vides dans nos tables car certains fichiers possèdent soient le nombre de personnes ayant votés blancs ou bien le nombre de votants s'étant exprimés (c'est à dire n'ayant pas voté blanc). Il nous appartiendra durant la phase d'analyse d'être vigilant pour bien calculer le nombre de vote blanc dans le cas où seule la colonne "exprimés" serait présente. Ceci complique quelque peu notre algorithme de calcul mais il résulte d'un désir de limiter les calculs à faire avant l'insertion dans le datawarehouse.

De plus, certaines données comme celles des années 1997 et 1993 n'existent pas au niveau de granularité bureau de vote. Nous avons pour cela penser remplacer cette colonne par la valeur 1 ou bien la laisser vide. Avec la représentation que nous avons choisi, cela ne pose pas de problème puisque la colonne bureau de vote peut rester vide tandis que les nombre de votants pour un candidat, de blancs, d'exprimés, seront stockés dans la colonne habituelle mais concerneront directement la commune. Cette commune sera en quelque sort considérée comme si elle n'avait qu'un seul bureau de vote. Lors de l'agrégation par commune, comme il n'existera qu'une seule ligne pour un candidat pour une commune, cela ne posera pas de problème. Cependant, les recherches à l'échelle du bureau de vote ne pourront pas prendre en compte ces élections bien évidemment.

Etant donné notre choix de matérialisation des données, il nous faudra joindre les données de l'INSEE avec les données du Ministère de l'Intérieur durant la phase d'alimentation. Cela signifie qu'à chaque nouvelle insertion il nous faudra re-ouvrir le fichier de données de l'INSEE pour croiser les données et récupérer les codes régions et les informations socio-économiques de chaque commune. Ceci est d'une efficacité qui pourrait être contestée, cependant de par la nature non-relationnelle de notre matérialisation de données, c'est la méthode qui reste la plus cohérente (non ne voulions pas avoir à effectuer des jointures lors des requêtes).

Pour ce qui est du parti du maire d'une com-

mune, ces informations sont aussi disponibles dans des documents de l'INSEE. Cependant, il est possible que nous ne puissions pas avoir accès aux fichiers de toutes les années. Dans l'hypothèse où nous serions capables de trouver un fichier ayant moins de 2 ans d'écart avec les données de l'élection que nous souhaitons insérer, nous pourrions effectuer la jointure bien que les informations pourraient être faussée dans l'hypothèse ou des élections communales auraient pu avoir lieu. Nous considérons tout de même que ces informations pourraient nous aider à nous faire une idée de la tendance politique à l'échelle de la commune à cette époque là c'est pourquoi nous souhaiterions conserver les données. Au delà de cette période nous laisserons la colonne vide. Par exemple, nous avons trouvé un document relatant des informations de 2008, ce qui pourrait tout à fait convenir pour croiser avec les données des élections législatives de 2007.

3 Analyses

Nous pouvons effectuer divers algorithmes de MapReduce pour analyser les données et en tirer des conclusions. Il est intéressant de noter qu'il est possible d'obtenir la population d'une commune, d'un département ou région en sommant le nombre d'inscrits aux bureaux de votes appartenant à la zone géographique dont il est question. Pour connaître le nombre d'inscrits pour une commune, il suffit de sommer le nombre d'inscrits pour chaque bureau de vote différent.

Nous avons préféré garder cette information, bien que parfois pas parfaitement pertinente, plutôt que de nous en passer. Il aurait aussi pu être possible de garder cette information seulement pour les élections ayant eu lieu sous le mandat des maires déjà présents en 2008, et d'avoir une colonne vide pour l'ensemble des autres élections. Cependant, même si l'information n'est pas forcément à jour pour certaines élections, elle peut tout de même donner de l'information sur le passé de la commune.

Listing 1 – Algorithme taux d'abstention par commune :

```
data = get_data(X, N)

fonction groupe_commune (d):
        commune, bureau_vote, inscrits, votants = d
        retourner(commune, (votants, inscrits))

somme_commune = data.map(groupe_commune).reduce(lambda x,y: x+y)

fonction calcul_abstention (d):
        commune, somme_votants, somme_inscrits = d
        retourner(commune, (somme_inscrits-somme_votants)/somme_inscrits)

taux_abstention = somme_commune.map(calcul_abstention)
```

Listing 2 – Algorithme taux des partis par département pour un tour X sur une année N :

```
data = get_data(X,N)

fonction groupe_departement (d):
        departement, commune, bureau_vote, inscrits, votants = d
        retourner((parti, departement), (votants, vote_candidat))

sommme_departement_parti = data.map(groupe_departement).reduce(lambda: x,y: x+y)
```

```
fonction calcul_vote_politique(d):
        parti, departement, somme_votants, somme_vote_parti = d
        retourner((parti, departement), somme_vote_parti/somme_votants)

taux_vote_politque = somme_departement_parti.map(calcul_vote_politique)
```

Listing 3 – Algorithme taux d'abstention et revenu moyen par departement :

```
data = get_data(X,N)

fonction groupe_departement (d):
        departement, commune, bureau_vote,
                inscrits, votants, revenu_median = d
        retourner((departement,commune),
                (1, inscrits, votants, revenu_median))

somme_commune = data.map(groupe_departement).reduce(lambda x,y: x+y)

fonction divise_revenu_median_commune(d):
        departement, commune, somme1, somme_inscrits,
                somme_votants, somme_revenu = d
        retourner(departement, (1, somme_inscrits,
                somme_votants, somme_revenu/somme1)

somme_departement = somme_commune.map(divise_revenu_median_commune)
                                 .reduce(lambda x,y: x+y)

fonction calcul_abstention(d):
        departement, somme1, somme_inscrits, somme_votants,
                somme_revenus = d
        retourner(departement, (somme_inscrits-somme_votants)/somme_inscrits,
                somme_revenus/somme1)

taux_abstention_revenus = somme_departement.map(calcul_abstention)
```

Pour compléter des analyses comme celles du taux d'abstention et du revenu moyen par département, nous pourrions utiliser Spark qui fournit de nombreux outils pour le machine learning et l'analyse statistique comme la régression linéaire. En effet, nous pourrions, à partir des données obtenues sur les départements, créer un graphe contenant les points `(revenu_moyen, taux_abstention)` et réaliser une régression linéaire pour imaginer prévoir le taux d'abstention à une élection sachant le revenu_moyen d'un département.

4 Implémentation

En ce qui concerne l'implémentation de notre datawarehouse, nous avions pensé utiliser Cassandra pour le stockage de nos données. En créant 4 tables différentes que nous alimentons séparément et de manière personnalisée selon la configuration des fichiers dans lequels se trouvent les données. Le script de création de notre première table (permettant l'étude du taux d'abstention et de vote blanc selon la géographie) est disponible en Annexe B

Pour ce qui est des calculs distribués, nous avons décidé d'utiliser PySpark, particulièrement efficace et facile à prendre en main avec son système de "maps" (fixer une clé et les valeurs associées) et "reduce" par clé (application de calcul par clé identique).

Appendices

A Matérialisation des tables de notre Datawarehouse.

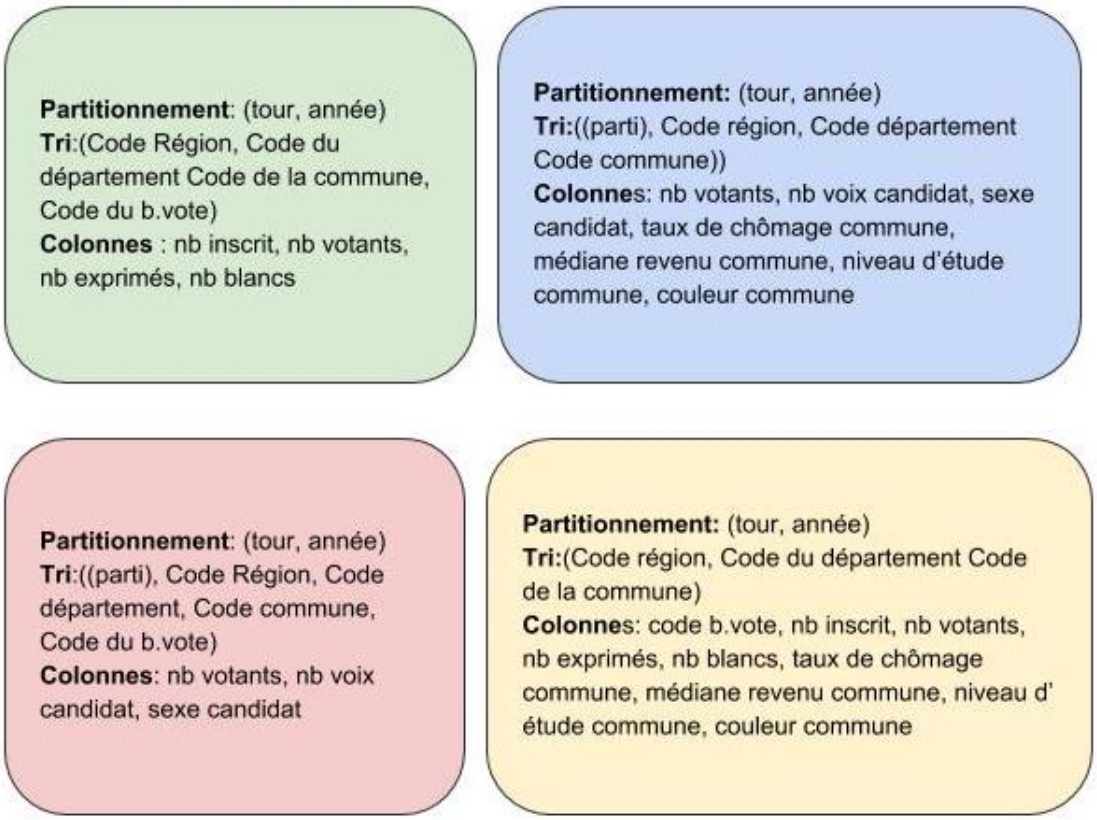

FIGURE 4 – Matérialisation des tables de notre datawarehouse

B Création d'une table de notre Datawarehouse avec Cassandra

Listing 4 – Création d'une table avec Cassandra :

```
CREATE TABLE abstention_blancs_geo
(
    annee int,
    tour int,
    region int,
    dept int,
    commune int,
    b_vote int,
    nb_inscrits int,
    nb_votants int,
    nb_exprimes int,
    nb_blancs int,
    PRIMARY KEY ((annee, tour), region, departement, commune, bureau_vote)
);

CREATE TABLE vote_politique_geo
(
    annee int,
    tour int,
    region int,
    dept int,
    commune int,
    b_vote int,
    parti string,
    nb_votants int,
    nb_voix int,
    sexe_candidat string,
    PRIMARY KEY ((annee, tour), parti, region, departement, commune, bureau_vote)
);
```

C Exemple de résultats possibles

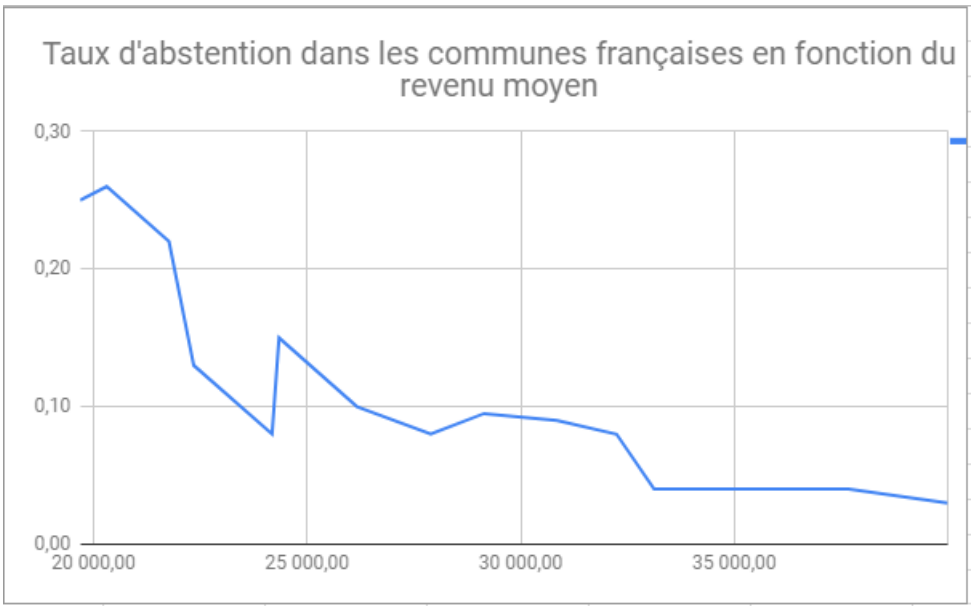

FIGURE 5 – Exemple de graphique obtenable avec notre modèle.

Ce graphique est un exemple de ce que nous pouvons obtenir en récupérant les données de nos map reduce. Ainsi, il peut être possible d'effectuer une régression linéaire pour prédire le taux d'abstention dans une commune en ayant le revenu moyen de celle ci.