Projet de traitement de données massives

Samuel Bouffard, Étienne Chabot, Réjean Drolet
Université Laval, 4 avril 2022

Abstract

Le présent rapport fait suite à l'analyse et au prétraitement des données du sondage électoral canadien 2019 visant à déterminer les intentions de vote d'un sous-ensemble d'individus. Il vise à détailler le processus de développement des algorithmes ainsi qu'à présenter et discuter des résultats d'inférence obtenus. Un classificateur naïf de Bayes et divers types de classificateurs basés sur les arbres de décision ont été comparés, après ajustement des paramètres, sur différents ensembles d'attributs.

Mot-clés: Étude électorale canadienne, Données massives, Classification, Arbres décisionnels, Classificateur naïf de Bayes

1. Introduction

En politique démocratique, particulièrement lors d'une campagne électorale, la connaissance de l'opinion publique est une information clé pour un parti politique. En effet, la plupart des partis politiques ont recours à des stratégies visant à cibler directement un auditoire qui sera plus réceptif au message et aux idées véhiculées par le parti ou à l'inverse, à identifier les idéologies devant être mises de l'avant face à un certain public. En ce sens, des études de différentes envergures sont régulièrement réalisées sur la population d'électeurs afin de déterminer ses caractéristiques démographiques et les enjeux importants pour celle-ci, cela en vue de mieux connaître les différentes opinions des électeurs. L'une de ces études est l'Étude électorale canadienne [1]. Celle-ci est utilisée dans le cadre de ce présent projet afin de prédire l'intention de vote d'un sous-ensemble d'individus.

Ce rapport couvre la deuxième partie du projet et présente les différents tests effectués ainsi que les résultats obtenus à partir de différents classificateurs de type boîte blanche développés de manière itérative. Ces derniers ont été préconisés car ils permettent d'obtenir une meilleure rétroaction sur les règles de décision apprises en entraînement dans un contexte où ce gain d'information est tout autant utile sinon plus qu'un gain important en classification. Il est donc naturel de débuter par des classificateurs transparents permettant de choisir plus judicieusement les attributs servant à l'inférence. Des modèles plus complexes de type boîte noire pourraient être étudiés ultérieurement dans une autre projet sur le même jeu de données. Premièrement, un bref retour sur les informations pertinentes du rapport précédent sera fait suivi de la présentation de la démarche itérative entreprise. Deuxièmement, la stratégie de gestion des valeurs manquantes ainsi que les différents classificateurs et leurs résultats en inférence seront présentés. Enfin, le classificateur final et quelques études de cas décrivant son comportement en classification seront conclus par une rétrospective sur le projet en général.

2. Retour sur les informations importantes du rapport précédent et présentation de la démarche itérative

Lors de la première partie du projet, les données d'entraînement ont été séparées des données de test et des étiquettes cibles ont été établies. L'analyse des corrélations entre chaque attribut et les classes d'étiquette a permis une première sélection des attributs ayant un plus fort potentiel prédictif. Ensuite, une sélection avant des variables avec l'aide d'un

arbre de décision a été effectuée afin de finaliser le choix des attributs qui seront utilisés comme points de départ pour l'entraînement des classificateurs présentés dans le présent rapport. La liste des attributs de départ peut être consultée à la première ligne de la figure 4.

L'approche générale se veut de débuter le prototypage par des classificateurs simples et ensuite de complexifier selon les résultats obtenus. Trois axes principaux ont été explorés: la complexité du classificateur, l'ajout d'attributs et l'assemblage de classificateurs. Tous les classificateurs ont été testés par validation croisée à cinq plis et leurs résultats en inférence ont été comparés. Les ajustements à chaque itération ont été faites en fonction des résultats de l'itération précédente. En ce sens, la première itération a été réalisée avec des classificateurs de complexité croissante, permettant ainsi d'évaluer le compromis entre les résultats et la complexité. Les classificateurs retenus ont ensuite été utilisés pour la deuxième itération, dans laquelle des attributs ont été ajoutés à partir d'une nouvelle sélection de variables. Finalement, pour la troisième itération, un assemblage des classificateurs les plus performants a été envisagée.

3. **Gestion des valeurs manquantes**

Lors du premier rapport, les principaux attributs présentant des valeurs manquantes pour lesquels l'absence de valeurs est une information en soi ont été transformés en attributs binaires. Pour les valeurs manquantes restantes, elles ont été remplacées par une constante globale afin d'être traitable par les classificateurs de scikit-learn [2]. Ce remplacement ne devrait pas avoir un impact trop significatif puisque, tel que déterminé dans la première phase du projet, les valeurs manquantes sont distribuées uniformément à travers les différentes classes d'étiquette dans le jeu d'entraînement et le jeu de test pour la quasi-totalité des attributs (notion de "missing at random").

Afin de confirmer cette hypothèse, un classificateur naïf de Bayes a été entraîné avec deux jeux de données différents, le premier où les valeurs manquantes ont été remplacées par la constante globale et le deuxième où les instances présentant des valeurs manquantes ont été retirées. L'analyse des résultats présentée dans la table 1 a permis de constater qu'un classificateur entraîné seulement sur les données sans valeurs manquantes a une exactitude de 75,4% contre 73,8% pour celui sur toutes les données lorsqu'il s'agit de déterminer la performance sur leur jeu d'entraînement respectif. On réalise cependant que le classificateur entraîné sur toutes les données atteint une exactitude de 75,7% lorsque testé seulement sur le jeu de données n'ayant aucune valeur manquante. On constate donc que l'entraînement sur plus de données, malgré la présence d'une constante globale, semble améliorer les prédictions. Les instances que le classificateur a davantage de difficulté à classifier sont les enregistrements comportant des attributs dont la valeur manquante a été remplacée par la constante globale. Il est donc peu probable que la création de multiples classificateurs selon les différentes combinaisons de valeurs manquantes possibles permette d'augmenter significativement les performances.

			Tester sur enregistrements		
			Sans valeurs manquantes	avec des valeurs manquantes	Toutes les données
		Nb enregistrements	23668	9269	32937
Entraîner sur	Sans valeurs manquantes	23668	75,4%	N/A	N/A
	Toutes les données	32937	75,7%	69,0%	73,8%

Table 1. Comparaison de l'efficacité de classificateurs selon la présence ou non de valeurs manquantes dans les données.

4. Première itération

4.1. Classificateur naïf de Bayes

Un classificateur naïf de Bayes semble intéressant pour le jeu de données étudié car plusieurs des attributs retenus sont nominaux. Pour les attributs où la réponse est une valeur numérique, elles ont été catégorisées en utilisant des conteneurs (bins) pour éliminer les valeurs aberrantes et rendre les attributs nominaux. Les attributs binaires et ordinaux quant à eux, ont été traités de la même façon que les attributs nominaux. Une fois tous les attributs rendus nominaux, ceux-ci ont étés remplacés par un entier de 0 à $n-1$ (où n représente le nombre de valeurs différentes que prend l'attribut). Ce remplacement n'impacte en rien les probabilités apprises par le classificateur bayésien, mais c'est une condition nécessaire à l'utilisation du module implémenté par scikit-learn.

Ce type de classificateur est très efficace avec des données massives puisque les données sont traitées linéairement. Étant donné la sélection de variables effectuées (voir section 2) et le nombre d'objets qui n'est pas si imposant, ce classificateur s'entraîne en moins d'une seconde, ne causant donc aucun souci pour ce qui est du temps de calcul.

Pour l'ajustement du classificateur, un seul hyperparamètre est à tester. Il s'agit d'un paramètre de "smoothing" permettant de mieux gérer les situations où l'on doit prédire une classe à partir d'une valeur d'attribut inconnu au modèle (évite d'avoir une probabilité à exactement 0). Il est attendu que ce paramètre ait une très faible influence sur le résultat tant que la valeur assignée demeure faible. Afin de confirmer cette supposition, une recherche en grille fut effectuée pour des valeurs fluctuantes de 0.1 à 1000. Une différence de seulement 0.1% d'exactitude fut observée pour les variations de 0.1 à 16 alors que les performances chutes assez dramatiquement lors de l'utilisation de valeurs trop élevées. Puisque l'influence de cet hyperparamètre est négligeable, la valeur par défaut de 1 fut conservée pour le reste des travaux. Au final, une exactitude de 73.8% a été obtenue pour ce classificateur.

4.2. Arbre de décision simple

L'arbre de décision simple a été implémenté à l'aide de la librairie scikit-learn de Python. Cette librairie implémente l'arbre dans sa version binaire, selon la méthodologie CART, en utilisant le critère de Gini. Pour cette technique le jeu de données a été préparé en accordance avec la méthodologie vue dans le cours : les attributs nominaux ont été transformés en variables « dummy », les attributs ordinaux ont été répartis de façon ordonnée dans l'intervalle 0-1 et les attributs numériques ont été réparties dans des conteneurs (bins).

Un des paramètres principal à sélectionner pour cet algorithme est la profondeur maximale de l'arbre. Une recherche d'hyperparamètres a donc été effectué avec l'exactitude comme critère pour des valeurs de 1 à 40 noeuds de profondeur. Ultimement, c'est une profondeur maximale de 10 qui a été retenue, car c'est à cette profondeur que les meilleurs résultats et meilleurs compromis entre métriques étaient répertoriés.

Les résultats préliminaires du classificateur sont plutôt bons (74% d'exactitude), mais aux dépens d'une classification plus fine et nuancée. En effet, en observant les résultats de rappel très bas pour certaines classes (de l'ordre de 0% à 40%), il est conclu que l'arbre a beaucoup de difficulté à reconnaître certaines classes minoritaires et que la majorité des gains en exactitude sont faits en prédisant les partis les plus populaires uniquement. Il est aussi remarqué que les attributs utilisés pour faire les coupures dans l'arbre sont, sans surprise et tel qu'attendu, liés aux partis principaux.

4.3. Forêt aléatoire

La forêt aléatoire est un algorithme stable fonctionnant bien avec des attributs nominaux. Bien que cet algorithme souffre d'une faible perte en interprétabilité par rapport à un arbre

décisionnel simple, il gagne en performances en revanche par l'utilisation d'une méthode ensembliste afin de laisser voter un groupe d'arbres plutôt qu'un seul. De plus, une des faiblesses principales d'un arbre décisionnel étant sa sensibilité aux bruits et aux valeurs aberrantes, bien que ceux-ci soient peu nombreux dans le contexte d'un sondage à choix de réponses, la forêt aléatoire s'en trouve moins impactée quant à elle.

D'abord, une forêt aléatoire d'arbres décisionnels a été entraînée avec ajustement des paramètres sur le même jeu d'entraînement mentionné à la sous-section sur l'arbre simple. Les meilleurs résultats sans perte d'exactitude (74.6%) et en temps de calcul optimal moyen de 1.1 secondes par pli (entraînement + évaluation) ont été obtenus avec le paramétrage suivant:

- Un nombre maximal de 50 estimateurs
- Pour la fonction mesurant la qualité de la division à chaque niveau d'arbre, le critère de Gini offre sensiblement les mêmes performances que le critère d'entropie, le premier a été retenu.
- Une faible augmentation de l'exactitude a été enregistrée avec un nombre de 15 instances requises pour effectuer la division dans un arbre, le modèle tend à sur-apprendre en dessous et sous-apprendre au-dessus de ça.

Il est possible de rentabiliser l'interprétabilité de ce type de classificateur afin d'obtenir plus d'informations sur le poids de chaque attribut lors de l'inférence. Afin de classifier les attributs en ordre de priorité, la diminution moyenne des impuretés (DMI) a été utilisée. Cette mesure, également connu sous le nom d'importance de Gini, compte le nombre de fois qu'un attribut est utilisé dans la forêt, pondéré par le gain en information de ce noeud (calculé par le delta de l'index de Gini). À première vue, afin d'utiliser l'implémentation de la forêt aléatoire de scikit-learn qui utilise des bornes pour le filtrage des nombres, les attributs nominaux doivent être traités sous format binaire. Par exemple, une règle de décision dictant que si la langue d'un électeur est plus grande que français, il a l'intention de voter pour le Parti libéral ne ferait pas trop de sens. Un constat est rapidement survenu, il est bien difficile d'étudier les attributs sous leur format binaire. Ceux-ci sont assez nombreux et sont moins parlants lorsque les attributs sont éclatés en plusieurs variables binaires. Non seulement ça, mais un gain d'exactitude de 0.2% a été enregistré en conservant les attributs sous leur format ordinal comparativement à leur format binaire. En étudiant davantage la DMI de chaque attribut pour l'entraînement sur le jeu dont les attributs n'ont pas été binarisés, voir figure 1, on remarque que les deux attributs les plus importants (cps19_fed_id et cps19_imp_iss_party) sont justement des attributs nominaux. Leurs valeurs correspondent aux différents partis politiques et il ne fait pas vraiment de sens de les traiter comme des attributs ordinaux. Pourtant c'est bien ce que l'algorithme implémenté fait sans perte en inférence.

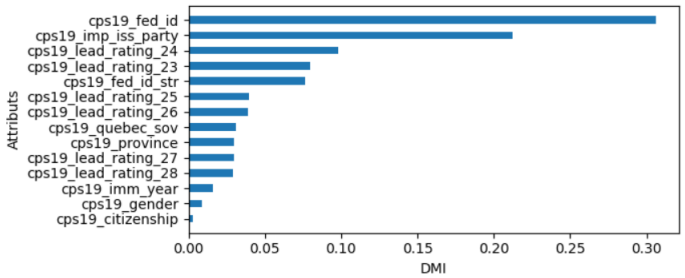

Figure 1. Importance des attributs selon la diminution moyenne des impuretés (DMI) sur le jeu d'entraînement non binarisé

Pour une même ligne hiérarchique d'un arbre dans une forêt aléatoire, un attribut ne peut être utilisé qu'une seule fois. Cependant, il peut se retrouver à plusieurs reprises dans le même arbre sous différents filtres s'il n'est pas dans la même ligne hiérarchique. De plus, l'attribut peut également se retrouver dans les autres arbres de la forêt. Une hypothèse est que le modèle parvient relativement bien, avec l'aide du filtrage par borne au travers des différents estimateurs, à séparer les attributs présentés comme ordinaux afin de les traiter comme des nominaux au point de surpasser la perte reliée à une plus grande dimensionnalité du jeu d'entraînement binarisé. Dans le contexte d'un autre jeu de données, ce genre de filtrage naïf pourrait causer beaucoup plus de tord que de bien, mais ça ne semble pas être le cas ici.

En résumé, l'ajustement des paramètres autour des valeurs par défaut et la sélection du format de données non binarisé ont permis d'augmenter l'exactitude des prédictions de 69.5% à 74.6%.

4.4. **Adaboost pour arbre de décision**

Suite aux résultats obtenus jusqu'à présent, la méthode d'apprentissage par ensemble AdaBoost a été essayée afin de déterminer si une méthode de boosting plutôt que de bagging pourrait augmenter l'exactitude moyenne des cinq plis. Peut-être que les relations ordonnancées apprises entre les souches d'arbres, des classificateurs très simplistes, permettront de faire ressortir des relations entre plusieurs attributs. Après ajustement des hyperparamètres, un taux d'apprentissage $\alpha = 0.85$ et un nombre maximal de 30 arbres ont permis d'obtenir une exactitude moyenne optimale de 72.9% sur les cinq plis en 1.3 secondes/plis (entraînement + évaluation), soit 1.7% de moins que le modèle de type forêt aléatoire. L'apprentissage séquentiel des règles de décision n'est donc pas une amélioration par rapport à l'apprentissage en parallèle pour ce jeu de données et ce modèle n'a donc pas été exploré davantage.

4.5. **XGBoost pour arbre de décision**

Afin de comparer les résultats obtenus jusqu'à présent, un modèle habituellement très performant a été testé. Le XGBoost est un modèle qui, un peu à la manière d'AdaBoost, prend des « weak learners » (dans le cas présent des arbres) pour les jumelés ensemble et en faire un classificateur très performant. L'algorithme ajoute progressivement des « weak learners » pour minimiser l'erreur. D'autres éléments comme des termes de régularisations, qui aident avec le sur-apprentissage, sont également ajoutés à l'algorithme pour finalement, constituer un classifieur ayant, depuis un certain temps, d'excellents résultats dans une multitude de situations. L'implémentation fournie dans la librairie XGBoost de Python a été utilisée avec la majorité des paramètres par défaut. Par contre, l'hyperparamètre lié à la profondeur maximale de l'arbre a été optimisé par recherche en grille. L'idée ici n'était pas d'aller chercher les performances maximales, mais bien de voir le gain obtenu en complexifiant de façon significative le modèle. L'entraînement prend quelques secondes et l'exactitude de 76.5% obtenue pour XGBoost est supérieure à celle des autres classificateurs (par 1.9% et plus).

4.6. **Analyse de la première itération**

Comme les mêmes observations ont été faites pour tous les classificateurs testés, cette section présente les conclusions de façon généralisée.

Une première observation est que la répartition des prédictions suit relativement bien la vraie répartition des classes dans les données, tel que présenté à la figure 2, ce qui amène à

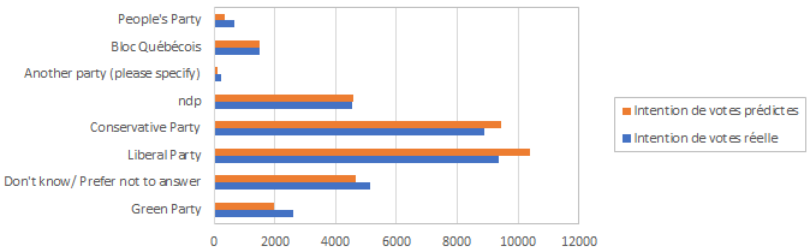

Figure 2. Comparaison de la répartition des intentions de vote prédites et réelles du jeu de données d'entraînement

penser que les classificateurs étudiés comprennent relativement bien les tendances dans les données.

La matrice de confusion des résultats en classification pour un des classificateurs (voir la figure 3), permet de constater que la majorité des erreurs sont naturellement au niveau des quatre classes les plus populaires que nous appellerons "classes primaires" et les autres "classes secondaires".

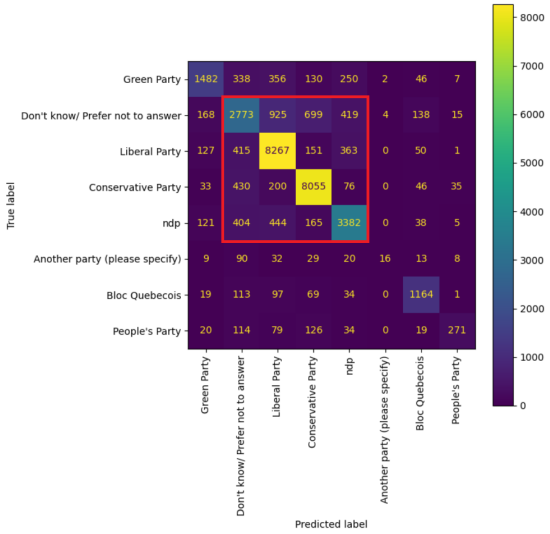

Figure 3. Matrice de confusion de la somme des k=5 plis, obtenue avec la forêt aléatoire

Par contre, à l'exception des classes représentées par les électeurs incertains et/ou ne souhaitant pas répondre ainsi que ceux ayant l'intention de voter pour le Bloc Québecois, le tableau 2 démontre que les classes primaires (en rouge) ont un rappel significativement plus élevé que les classes secondaires, ce qui indique que les classificateurs sont très bons pour reconnaître ces classes et ont de la difficulté à repérer celles moins populaires. Une raison expliquant la meilleure performance en lien avec le Bloc Québécois est que plusieurs questions distinctes et uniques sont fortement liées avec ce parti (souveraineté, lieu de résidence étant le Québec, etc.).

En étudiant davantage la matrice de confusion, on observe, tel qu'attendu, que la classe « Don't Know/Prefer not to Answer » est confondue avec toutes les autres classes, reflétant le fait que notre classificateur comprend l'appartenance à un parti sans nécessairement comprendre le désir de ne pas divulguer ses intentions. D'autres constats sont que le Npd est

Intention de vote	Rappel	Précision
Green Party	56.8%	74.9%
Don't know/Prefer not to answer	53.9%	59.3%
Liberal Party	88.2%	79.5%
Conservative Party	90.8%	85.5%
ndp	74.2%	73.9%
Another party	7.4%	72.7%
Bloc Quebecois	77.8%	76.9%
People's Party	40.9%	79.0%

Table 2. Rappel et précision moyen par classe pour les cinq plis, obtenus avec la forêt aléatoire

souvent confondu avec le Parti libéral ainsi que le Parti populaire avec le Parti conservateur. Les instances étiquetées « Another Party » ne sont presque jamais bien classifiées.

Bien que l'instinct dicte de s'attarder aux classes secondaires ayant un rappel plus faible et de tenter de l'améliorer, l'exactitude globale pourrait être largement atteinte par une faible diminution de la précision des classes primaires, celles-ci comportant un nombre beaucoup plus grand d'intentions de vote. En d'autres mots, il est beaucoup plus payant d'associer un répondant au Parti libéral qu'au Parti populaire en cas de doute étant donné le faible nombre d'électeurs ayant l'intention de voter pour le Parti populaire par rapport au Parti libéral.

5. **Deuxième itération**

Pour la deuxième itération, tous les classificateurs ont été utilisés et optimisés de la même façon qu'à l'itération précédente, mais avec des nouveaux jeux de données bonifiés par l'ajout de nouveaux attributs. Également, afin de tenter d'améliorer la reconnaissance des classes secondaires, une méthode par ensemble de classificateurs binaires pour chaque parti a été évaluée.

5.1. **Ajouts et modifications d'attributs**

Puisque le classifieur naïf de Bayes suppose l'indépendance conditionnelle des attributs, il semblait opportun de vérifier l'intérêt de fusionner les attributs qui vont ensemble. Les attributs concernés sont cps19_fed_id_str qui vient indiquer à quel point on se sent proche de notre parti politique entré en cps19_fed_id. Une série de tests fut effectuées avec différentes combinaisons d'attributs et la version combinant les deux attributs en un seul obtenait toujours des exactitudes supérieures (de l'ordre de 0.1% à 0.7%) au classificateur identique, mais avec les attributs non combinés. Il a donc été statué de conserver les attributs fusionnés pour le classificateur naïf de Bayes, mais de laisser les attributs séparés pour les autres types de classificateurs puisque ceux-ci, par leurs structures internes, sont d'avantages en mesure de travailler avec une certaine dépendance entre les attributs.

De plus, étant donné que la sélection de variables de la phase précédente du projet avait été effectuée par une combinaison d'analyse de données (sans prouver l'efficacité par un classificateur) et une sélection avant de variable à partir d'un arbre de décision sur un sous-ensemble des données, il semblait pertinent de confirmer que les attributs utilisés donnaient effectivement les meilleures performances possibles avec les classificateurs réellement utilisés. Il fut surprenant de constater qu'il était possible d'aller chercher environ 4% de gain en exactitude en sélectionnant mieux nos attributs. En effet, le gain est surtout attribuable à l'attribut cps19_outcome_most qui avait été ignoré lors de la première sélection ainsi qu'à l'utilisation des cps19_party_rating_X plutôt que les cps19_lead_rating_X. Les détails des différences entre les sélections de variables peut être consultés dans la figure 4.

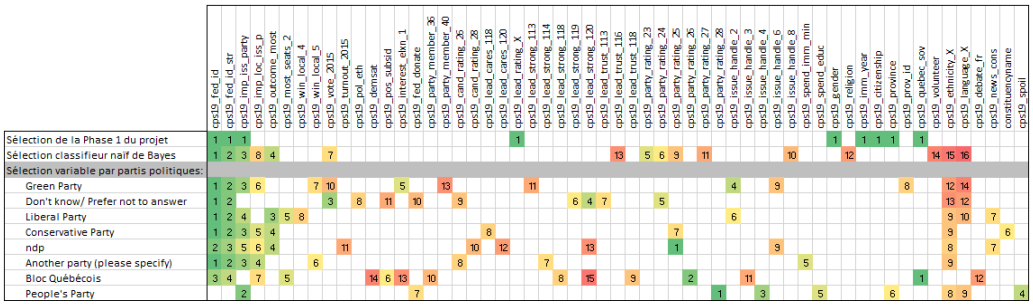

Figure 4. Ordre de priorité des variables obtenu suite à sélection avant de celles-ci avec le classificateur naïf de Bayes.

En utilisant ces nouveaux attributs, la plupart des classificateurs ont vu leur exactitude s'améliorer de quelques pourcents. De plus, une légère amélioration du rappel est aussi remarquée, signifiant que les nouveaux attributs ont aidé à identifier les classes, même celles les moins populaires. Les performances des meilleurs classificateurs sont présentées dans la table 3 de la section 6.

5.2. **Classificateurs par partis politiques**

Afin de raffiner les prédictions pour les partis politiques plus difficiles à prédire, une tentative a été faite afin de créer un classificateur par parti politique. Pour ce faire, une sélection avant de variable a été faite pour chacun des partis en présentant 50% de données du parti politique désiré et 50% de données des autres partis politiques. L'étiquette cible est donc booléenne et indique si l'enregistrement correspond au parti testé ou non. Il est clair que cette méthode ne permet pas d'entraîner des classificateurs réellement efficaces pour la tâche finale, mais l'idée ici est seulement de déterminer les attributs qui permettent de différencier le parti politique en question des autres partis. Les attributs sélectionnés sont présentés dans la figure 4. On peut constater que certains attributs reviennent dans presque tous les classificateurs alors que d'autres sont plus utiles pour certains partis politiques seulement.

Une fois les principaux attributs pour chaque parti trouvés, les classificateurs furent entraînés à nouveau, mais cette fois sur l'ensemble des données en modifiant le poids des enregistrements correspondant au parti politique du classificateur. L'idée ici en augmentant les poids est de réduire l'impact du dé-balancement des classes en simulant une plus grande fréquence des enregistrements correspondant au parti politique désiré. Les poids optimaux ont étés trouvés pour chaque classificateur selon les métriques F-Score, F0.5-Score et F2-Score. Puis les meilleurs classificateurs par parti politique ont étés combinés en brisant les égalités avec le classificateurs ayant la plus forte précision. Si aucun classificateur par parti ne prédit son parti politique, le parti du classificateur généraliste déjà développé fut utilisé. De façon unanime, les classificateurs optimisés selon la métrique du F0.5-Score étaient les meilleurs. Ceci n'est pas surprenant puisque le classificateur généraliste était déjà performant alors on veut seulement y ajouter les prédictions qu'on est vraiment certain et le F0.5-Score donne un peu plus de poids à la précision qu'au rappel.

Les résultats des meilleurs classificateurs par parti, ainsi que l'ensemble décrit ci-haut peuvent être consultés dans la figure 5. On y observe un fait surprenant, les poids utilisés ont, pour la plupart, réduit le poids des enregistrements du parti concerné plutôt que de l'augmenter tel que prévu. Ceci étant causé par le désir d'augmenter la précision au détriment du rappel, mais a aussi pour effet de seulement prédire les cas les plus certains. Puisque ces cas les plus certains sont généralement bien prédits par le classificateur généraliste, la

performance globale ne fut changée que d'un maigre 0.1%, ne permettant pas réellement de conclure à une augmentation de la performance.

Ces essais nous ont fait réalisé qu'il y a beaucoup d'éléments (attributs choisis, poids des enregistrements, métriques utilisées, etc.) qui entrent en ligne de compte pour ce type de classificateur par parti et il est difficile et long de déterminer la meilleure combinaison. Avec les résultats obtenus, il serait surprenant que cette avenue permette des gains réellement substantiels alors il a été statué de rester à l'idée de base prévue lors de la phase 1, c'est-à-dire, effectuer un ensemble de plusieurs types de classificateurs entraînés pour tous les partis politiques.

	Classificateur naif de Bayes										
			Classificateurs pour un parti spécifique								
	Sélection attributs de la Phase 1 du projet	Sélection attributs classifieur naif de Bayes	Green Party	Don't know/ Prefer not to answer	Liberal Party	Conservative Party	ndp	Another party (please specify)	Bloc Québécois	People's Party	Ensemble des classificateurs par parti politique
Exactitude	73,8%	77,7%	94,8%	87,6%	89,6%	92,6%	92,6%	98,9%	97,8%	98,6%	77,8%
Précision	N/A		76,0%	69,1%	88,4%	91,8%	84,0%	24,2%	85,6%	74,2%	N/A
Rappel			49,4%	37,3%	72,9%	79,6%	57,1%	31,8%	62,3%	45,6%	
F-Score			59,9%	48,5%	79,9%	85,3%	68,0%	27,5%	72,1%	56,4%	
F2-Score			53,1%	41,1%	75,5%	81,8%	61,0%	29,9%	65,8%	49,4%	
F0.5-Score			68,6%	59,0%	84,8%	89,1%	76,8%	25,4%	79,6%	65,9%	
Poids des enregistrements du partis à l'entraînement:			0,05	0,25	0,1	0,05	0,05	3	0,1	0,1	

Figure 5. Comparaison des performances des classificateurs par partis politiques.

6. **Classificateur final (troisième itération)**

Suite aux multiples expérimentations faites au cours de ce projet, l'attribut cps19_fed_id s'est avérer le prédicteur le plus fort autant lors des corrélations avec la cible (premier rapport), qu'avec la sélection avant de variables (figure 4) et la diminution moyenne des impuretés (figure 1). Ceci n'est pas surprenant puisque la question demande directement à quel parti politique le répondant s'associe, il est intuitif de penser qu'il votera aussi pour ce parti. L'attribut cps19_fed_id_str est aussi utile pour venir pondérer l'importance à accorder à ce premier prédicteur.

Les autres prédicteurs les plus forts sont cps19_imp_iss_party, cps19_imp_loc_iss_p, cps19_outcome_most et cps19_vote_2015. Il s'agit de questions qui amènent vraiment le répondant à préciser sa vision des partis politiques, par exemple en demandant quel parti politique adresse le mieux l'enjeu qu'il considère le plus important ou pour quel parti politique ils ont voté en 2015. Encore une fois, il n'est pas surprenant de constater que ces prédicteurs sont très près de l'intention de vote du répondant. De plus, il s'agit tous de champs où la réponse est directement un nom de parti politique ou presque (ex : NDP minority). Il est donc assez simple pour les classificateurs de retirer toute l'information nécessaire de ces champs.

Les champs cps19_party_rating_X performent mieux que les autres champs du même style (cps19_lead_rating_X, cps19_lead_trust_X, etc.) pour finaliser le départage des partis. On constate que les algorithmes ont tendance à surtout prendre ceux des plus gros partis puisqu'il est plus important de faire peu d'erreurs sur ceux-ci étant donné qu'ils sont sur-représentés dans le jeu de données.

Finalement, d'autres attributs ayant un pouvoir prédictif moins important ont été testés et offrent tout de moins de faibles gains en classification. Les détails peuvent être consultés à la figure 4. Le classificateur de Bayes final utilise sa propre sélection de variables (deuxième ligne dans cette figure) alors que les autres classificateurs utilisent tous les attributs des deux

sélections de variables (deux premières lignes de la figure) puisqu'ils sont en mesure, par leur structure interne, de choisir eux-mêmes les bons attributs à utiliser selon la situation.

Une fois les trois principaux classificateurs (Naïf de Bayes, forêt aléatoire, XGBoost) développés et optimisés, un ensemble fut développé dans la perspective que les classificateurs ne feraient pas nécessairement les erreurs aux mêmes endroits et que la combinaison de ceux-ci amènerait potentiellement un léger gain en performances. Le détails des performances de ces classificateurs, ainsi qu'un ensemble par votes de ceux-ci, est donné dans la table 3.

Classificateur	Exactitude
Classificateur naïf de Bayes	77.7%
Forêt aléatoire	77.3%
XGBoost	79.4%
Ensemble	78.7%

Table 3. Performance des classificateurs finaux

L'analyse des résultats démontre que le classificateur XGBoost est tellement meilleur que les autres classificateurs que l'ensemble des trois classificateurs performe moins bien que celui-ci. Cela veut dire que lorsque les deux autres classificateurs sont d'accord sur un choix différent de celui de XGBoost, il est plus fréquent que le choix de XGBoost soit meilleur que l'accord des deux autres classificateurs. Suite à ce constat, XGBoost seul fut utilisé pour la prédiction finale.

Une fois la prédiction finale sortie, une vérification a été faite afin de s'assurer que le Bloc Québécois n'était prédit que pour la province de Québec. Aucune prédiction ne présentait ce problème, indiquant ainsi que le classificateur a appris par lui-même cette règle. Si ce n'avait pas été le cas, il aurait été possible de remplacer spécifiquement ces prédictions par celles d'un autre classificateur et même, au besoin, utilisé le second choix du classificateur naïf de Bayes. En effet, ce dernier permet de récupérer les probabilités estimées pour toutes les classes.

7. Étude de cas

7.1. Cas particuliers et problématiques

Certaines erreurs faites par le classificateur final sont relativement justifiables puisque les réponses sont très surprenantes, pour ne pas dire aberrantes. En effet, par exemple, dans une instance, le répondant a répondu à la question "cps19_fed_id" comme quoi il s'identifie généralement au parti populaire, alors qu'il a voté pour le parti vert. Cela est très surprenant, puisque lors du premier rapport, dans la phase d'exploration des données, il avait été trouvé que les voteurs de ces deux partis ont peu de similarité dans leurs réponses. Ce répondant s'identifie probablement à différents partis selon l'enjeu, est indécis ou il n'a pas répondu sérieusement au questionnaire et représente donc un cas aberrant. Il est à noter qu'un cas comme celui-ci, où les réponses sont peu cohérentes, reste difficile à classifier, mais l'est d'autant plus pour notre classificateur qui utilise fortement l'attribut "cps19_fed_id".

Un autre type d'erreur fréquente du classificateur final est remarquable lorsque les répondant semblent indécis, ou n'exprime pas d'opinion tranchée, ce qui mène habituellement à une prédiction vers "Don't know / Prefer not to answer". Un exemple de ceci est une répondante qui a répondu ne s'identifier à aucun des partis (attribut "cps19_fed_id") et a coté les partis ou les chefs de façon relativement neutre (entre n'aime pas et aime) pour les attributs comme "cps19_lead_rating" ou "cps19_party_rating". Le classificateur a prédit "Don't know / Prefer not to answer" dans ce cas alors que la vraie réponse était le parti Libéral. Tel que mentionné à la section 4.6, la classe "Don't know / Prefer not to answer"

est confondu avec toutes les classes, il n'est donc pas surprenant de constater que ces cas peuvent être difficiles à discerner.

Une autre situation peut se produire lorsque le répondant semble penché légèrement vers un parti, menant à une prédiction en ce sens, alors qu'il ne sait pas ou ne veut pas déclarer ses intentions de vote. C'est notamment le cas d'un répondant qui a dit s'identifier au parti conservateur avec beaucoup de conviction, tout en ayant confiance en son chef, mais a refusé de divulguer ses intentions de votes. Ce type de cas est problématique pour le classificateur proposé puisqu'il base son analyse sur des attributs qui sont souvent fortement corrélés avec des partis politiques. On considère donc que l'erreur du classificateur est justifiée puisqu'elle ne représente pas un manque de compréhension de la situation, mais plutôt un désir de confidentialité du répondant.

7.2. **Cas bien classifiés**

Bien que l'attribut "cps19_fed_id" soit habituellement très utile au classificateur, d'autres attributs permettent de nuancer la prédiction en utilisant d'autres attributs clés. Par exemple, une répondante dans les données d'entraînement a répondu parti Libéral pour l'attribut "cps19_fed_id". Par contre, cette même répondante a répondu que le parti adressant le mieux son enjeu le plus important (cps19_imp_iss_party) est le NPD et qu'elle aimait autant Justin Trudeau que Jagmeet Singh. Cet exemple montre les performances du classificateur qui va chercher les nuances, soit que les gens favorisent probablement le parti qui s'attaquera à leur enjeu le plus important lors du vote, afin de faire une prédiction correcte, malgré les informations qui semblent contradictoires ou qui pourraient amener de la confusion dans le questionnaire.

Également, le classificateur réussi environ 59% du temps à bien classer des répondants ayant répondu qu'ils ne s'identifient à aucun parti. Ce qui aide le classificateur est que ces répondants ont presque toujours l'intention de voter pour l'un des trois partis principaux, plus faciles à reconnaître, ou de ne pas divulguer leurs intentions. Par exemple, un répondant a répondu qu'il ne s'identifie à aucun parti, mais a répondu le parti Conservateur à l'attribut "cps19_imp_iss_party" et qu'il aime Andrew Sheer. Ses intentions de vote étaient en faveur du Parti conservateur et ce cas a été bien classifié. D'autres cas similaires sont répertoriés pour le Parti libéral et le NPD. Un des répondants dans les données n'a pas donné ses intentions de votes et à répondu que le NPD répondait le mieux à son enjeu principal, mais toutes les autres questions avaient comme réponse "Don't know / Prefer not to answer". Comme illustré dans notre premier rapport, les questionnaires ayant beaucoup de "Don't know / Prefer not to answer" avait très souvent comme réponse aux intentions "Don't know / Prefer not to answer" également, ce que le classificateur semble avoir appris.

Finalement, le classificateur performe très bien pour les cas où un parti était fourni aux attributs "cps19_fed_id" et "cps19_vote_2015" sans autre information précise aux autres questions.

8. **Retrospective sur le projet**

Ce projet est un exemple assez réaliste d'analyse et de développement d'un classificateur. Il a permis de constater que l'intuition est parfois bonne et d'autres fois moins bonne et qu'il est important de valider chaque supposition. Par exemple, en début de projet, étant donné le grand nombre de valeurs manquantes dans certains attributs, il était tentant de se concentrer seulement sur les attributs n'ayant pas ou peu de valeurs manquantes. Cependant, en comprenant mieux les données, on constate que certains attributs sont très importants lorsqu'ils sont présents et qu'une valeur manquante ne signifie pas nécessairement que le répondant n'a pas répondu, mais parfois simplement qu'il n'a pas coché une réponse dans une question à choix multiples. La gestion finale effectuée pour les valeurs manquantes

(remplacement par une constante globale) s'est avérée très simple et efficace alors qu'on anticipait une grosse problématique par rapport à ces valeurs manquantes.

Une autre intuition de début de projet était que le champ cps19_fed_id était presque un classificateur à lui seul puisque la question se rapproche de celle demandant au répondant son intention de vote. En fait, selon les étiquettes de classes déterminées, cet attribut a une exactitude de 69% à lui seul, laissant présager que le classificateur final aurait une exactitude un peu plus élevé, mais qu'il serait difficile d'avoir un gros écart avec celui-ci. L'exactitude finale de 79.4% confirme cette intuition de départ. Il fut cependant surprenant que la création de classificateurs par partis politiques n'ait pas permis d'améliorer les prédictions. En effet, c'était une supposition très forte au début du projet, mais ces classificateurs n'ont pas donné les résultats escomptés.

Finalement, l'approche de développement utilisée visant à bien comprendre l'ensemble des attributs avant de commencer le développement d'un algorithme de sélection de variables n'était pas optimale. Ce dernier pouvant être assez long à mettre en place, il aurait été plus efficace qu'il soit développé en parallèle pendant l'analyse l'étude des attributs. Cela aurait permis d'avoir plus de temps pour le développer et de le rouler sur l'ensemble des données plutôt que de petits sous-ensembles augmentant ainsi la précision de cette sélection de variables lors du premier rapport et limitant la nécessité de le faire à nouveau pour le second rapport. De plus, il fut constaté que la sélection de variables par le classificateur naïf de Bayes mène à une progression plus régulière que celle par un arbre de décision tel que le démontre la figure 6. Ceci étant probablement dû au fait que l'arbre de décision peut changer complètement de forme lors de l'ajout d'un attribut et le phénomène fut possiblement amplifié par le petit sous-ensemble de données utilisé lors du premier rapport. Il est aussi beaucoup plus simple de généraliser une sélection de variables sur l'ensemble des variables par un classificateur naïf de Bayes qu'un arbre de décision pour les variables nominales puisqu'il est nécessaire d'exploser ces variables en variables binaires (dummy) pour l'arbre de décision, rendant ainsi l'interprétation des résultats plus complexes. Le classificateur naïf de Bayes aurait probablement été un meilleur estimateur de base pour la sélection de variable dès le premier rapport.

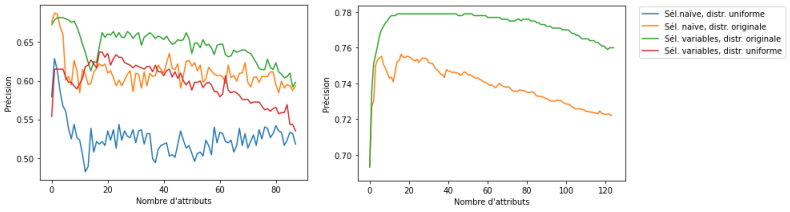

Figure 6. Évolution de la précision du classificateur en fonction du nombre d'attributs selon la distribution des classes et le type de sélection de variables effectuée. Le graphique de gauche représente la sélection de variable effectuée pour le premier rapport avec un arbre de décision sur environ 5% du jeu de données alors que le graphique de droite représente la sélection effectuée pour le second rapport avec le classificateur naïf de Bayes sur l'ensemble des données d'entraînement.

References

[1] *Étude électorale canadienne*. 2019. URL: http://www.ces-eec.ca/.

[2] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.