Projet Algo ML

Classification de partie du jeu "League Of Legends"

Cavalier Pierre

1 Introduction

League of Legends (LoL) est un jeu en ligne multijoueur très populaire qui implique deux équipes de cinq joueurs s'affrontant sur une carte virtuelle. Dans ce jeu, chaque joueur contrôle un personnage appelé « champion », avec des capacités uniques. L'objectif du jeu est de détruire la base ennemie située dans le coin opposé de la carte tout en défendant la sienne.

Dans ce contexte, l'analyse de données peut fournir des informations précieuses pour aider à comprendre les facteurs clés qui déterminent la victoire d'une équipe sur l'autre. Dans ce projet, nous allons utiliser des techniques de classification pour prédire quelle équipe a gagné la partie en fonction des caractéristiques des équipes telles que la composition de l'équipe, les objectifs de la carte. Les objectifs notables pris en compte dans cette analyse sont : deux monstres qui apparaissent à des intervalles réguliers, le héraut de la faille, le baron et le dragon, ainsi que les batiments de chaque équipe, les tourelles et les inhibiteurs.

En utilisant les données de parties antérieures, nous allons entraîner un modèle de classification capable de prédire quelle équipe est susceptible de gagner une partie donnée. Ce modèle peut être utilisé pour identifier les facteurs qui ont une incidence sur la victoire et pour aider les joueurs à améliorer leurs stratégies de jeu.

2 Jeu de données

Le jeu de données à ce lien est intitulé "League of Legends Ranked Games" et contient des données sur des matchs classés dans le jeu en ligne League of Legends.

Les données comprennent des informations sur 50 000 matchs classés datant de 2019 ainsi que les caractéristiques des différents champions et des sorts du jeu. Le jeu de données comprend également des informations sur les statistiques globales pour chaque match, comme la durée du match, le nombre total de kills, le nombre de tours détruites, le nombre de dragons tués, etc.

Ce jeu de données est utile pour les analystes de données et les chercheurs intéressés par l'étude des tendances et des habitudes de jeu dans League of Legends, ainsi que pour les développeurs de jeux vidéo qui cherchent à comprendre les préférences et les comportements des joueurs.
En outre, l'étude de ce jeu de données permet de mettre en avant les facteurs principaux de victoire lors d'une partie, ainsi que la prédiction du vainqueur d'une partie en cours.

3 Visualisation et Feature engineering

Dans un premier temps, il a fallu procéder à un tri de variable. En effet, pour chaque partie jouée, il y a 61 variables dans le dataset, donc 60 variables explicatives. On peut séparer les variables dans plusieurs groupes. Celles à titre informatif, l'ID de la game ou encore la date, qui n'ont pas spécialement d'importance dans le modèle. On distingue aussi les variables qui concernent le pré lancement de la game, à savoir les champions qui ont été bannis et sélectionnés par les participants, avec les sorts qui ont été pris. La dernière catégorie concerne les évènements qui ont lui dans la partie à proprement parler. Comme la prise des objectifs mentionnés dans l'introduction bien que cela ne référence pas tout les évènements de la partie.

Pour ce qui relève de la sélection de champion, on a regardé si certains avait un taux de victoire significativement plus élevé que la moyenne. Pour chaque champion dans l'équipe 1 nous avons regardé la moyenne des victoires de l'équipe un et le résultat est représenté sur la figure 3.1.

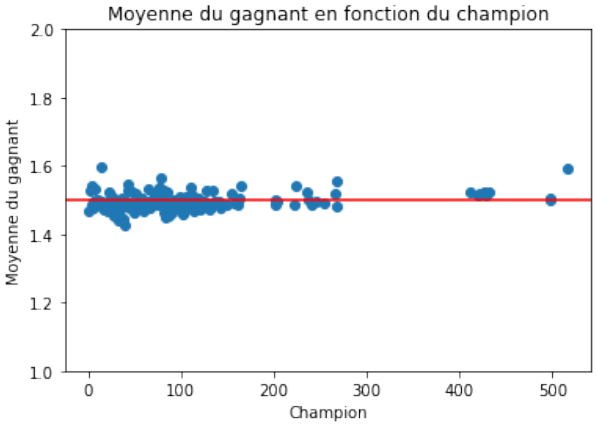


FIGURE 3.1 – Matrice de corrélation

On remarque que certains champions s'éloignent de la droite du taux de victoire moyen, bien que cette statistique n'est pas pondérée par la présence de chaque champion.

Par la suite, en traçant la matrice de corrélation sur la figure 3.2 entre l'équipe gagnante et la première prise de chacun des objectifs mentionnés précedemment. On remarque une relativement grosse corrélation entre le vainqueur et l'équipe qui prend le première inhibiteur. En outre les variables explicatives de la première tour, du premier baron et du premier inhibiteur semblent corrélées entre elles.

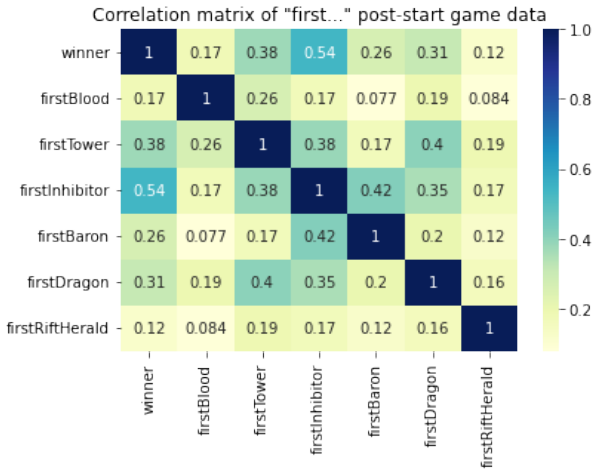


FIGURE 3.2 – Matrice de corrélation

Ainsi pour les trois variables les plus corrélées avec l'équipe gagnantes, la première tour, le premier inhibiteur et le premier dragon, on a tracé sur la figure 3.3 les boîtes à moustaches avec le nombre de fois où ces objectifs ont été pris dans la partie en fonction de l'équipe gagnante.

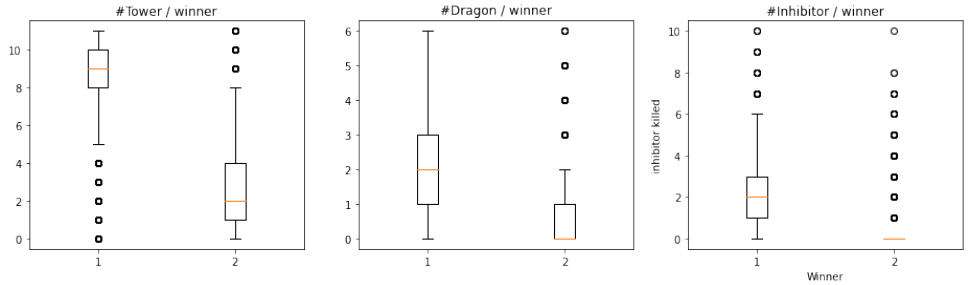


FIGURE 3.3 – Boîtes à moustaches

A partir de l'analyse de ces diagrammes en boite, on remarque que ces trois variables ont un très fort impact sur l'issue de la partie et qu'elles vont de se fait jouer un rôle prépondérant dans les différents modèles.

On a vérifié la répartition des données en vérifiant que les données sont bien répartis entre chaque équipe. Les victoires de la première équipe représentent 49.3% des victoires totales ce qui est convenable. Le nombre d'objectif récupéré par chaque équipe est en moyenne la même, à 0.1 près.

Pour la suite, on a discerné deux jeux de variables explicatives, un d'avant la partie et un de pendant la partie. Pour la sélection des champions pour chaque équipe on utilise un OneHotEncoder ce qui permet d'éviter la réprésentation du choix des champions par un entier qui n'a pas de signification numérique. En outre, les champions bannis ne sont pas pris en compte car il ne joue pas un rôle aussi prépondérant que ceux sélectionnés.

Pour le second jeu de de variables explicatives, on a pris toutes les premières fois où des objectifs ont été pris, ainsi que le nombre d'objectifs pris par chaque équipe. On a ensuite normalisé les données avec une normalisation min max pour éviter que certaines caractéristiques ne dominent les autres en raison de leur échelle et pour accélérer la convergence des algorithmes d'apprentissage automatique et améliorer la précision des modèles.

Pour l'entrainement des modèles, le jeu de données a été séparé en un jeu d'entrainement et un jeu de données test avec une répartition 70-30.

4 Benchmarks

4.1 Régression logistique

Notre problème étant un problème de classification en deux classe, le modèle le plus simple a essayer est celui de la regression logistique. Elle est basée sur la fonction logistique, qui transforme une valeur continue en une valeur binaire en utilisant une fonction de seuil. L'équation de la régression logistique est donnée par :

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_p x_p)}}$$

où y est la variable binaire à prédire, $x_1, x_2, ..., x_p$ sont les variables explicatives, $\beta_0, \beta_1, \beta_2, ..., \beta_p$ sont les coefficients de régression, la probabilité que la variable binaire soit égale à 1 pour une combinaison donnée de variables explicatives.

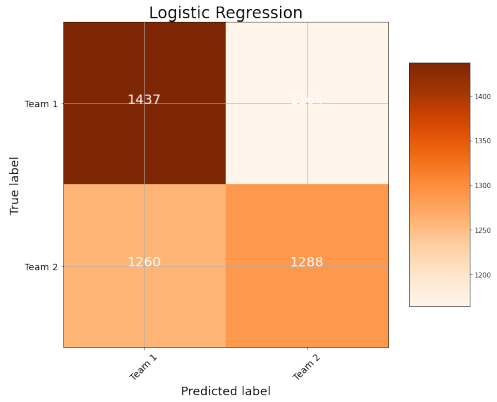


FIGURE 4.4 – Avant la partie

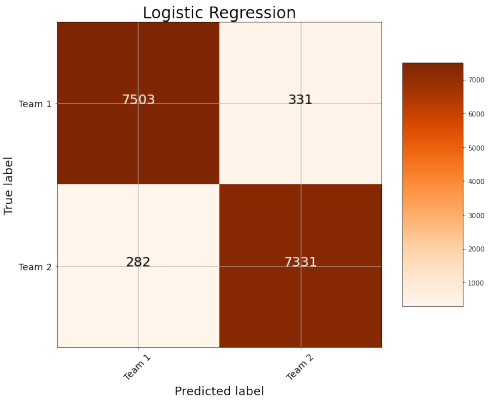


FIGURE 4.5 – Pendant la partie

Pour évaluer les modèles, on calcule dans un premier temps leur précision sur le jeu de données test. Pour le modèle s'entrainant uniquement sur les données de sélection de champion, on obtient une précision de 52,9% et pour le second modèle 96%. On regarde aussi les matrices de confusions pour voir si les performances du modèles fluctuent selon une catégorie.

4.2 Arbre de décision

On a utilisé ensuite un arbre de décision pour trouver l'équipe ganant. L'équation utilisée pour calculer l'homogénéité est basée sur la mesure de l'entropie qui est définie comme :

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

$$Gain(X, Y) = H(X) - \sum_{y \in Y} \frac{|X_y|}{|X|} H(X_y)$$

Où X est l'ensemble de données, Y est la variable cible, X_y est le sous-ensemble de X où la variable cible est égale à y, $|X_y|$ est le nombre d'éléments dans X_y, et $H(X)$ est la mesure d'entropie de X. Le but est de trouver la caractéristique qui maximise le gain d'information pour diviser l'ensemble de données.

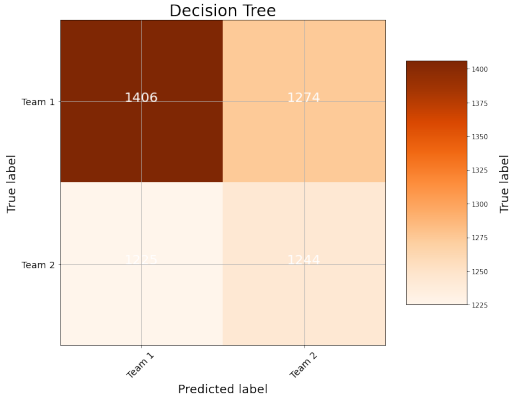


FIGURE 4.6 – Avant la partie

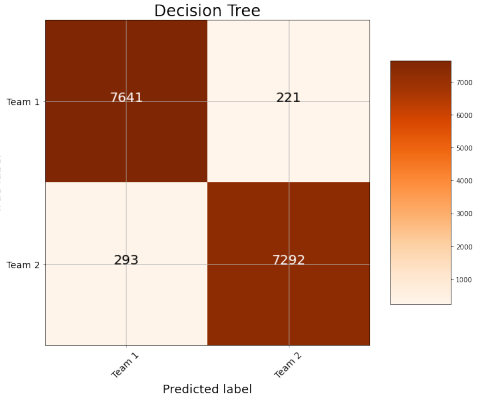


FIGURE 4.7 – Pendant la partie

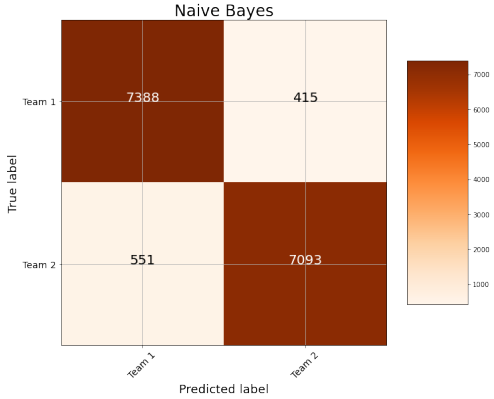
On obtient des précisions respectives de 51.4%, 96.6% ce qui correspond à une amélioration pour le modèle de pendant la partie mais ce qui correspond à une dégradation pour le modèle avant la partie. Pour les matrices de confusion, on remarque qu'il y a une centaine de prédiction de l'équipe 2 gagnante quand l'équipe 1 gagne en moins par rapport à la regression logistique ce qui peut être intéressant. On peut remarquer que le prédicteur pour les données d'avant partie est proche de 50% et se base uniquement sur les champions. Les champions évoluant au cours du temps ce prédicteur dépend de la force des champions à un instant t, nous allons donc cesser de tester pour les donner d'avant partie, faute de résultats convainquant.

4.3 Naive Bayes

Cette approche repose sur l'hypothèse que toutes les variables explicatives sont indépendantes les unes des autres, ce qui permet de simplifier le calcul de la probabilité conditionnelle d'une classe donnée étant donné un ensemble de caractéristiques. Cette probabilité est calculée à l'aide de l'équation de Bayes, qui est la suivante :

$$P(C_k|x_1, x_2, ..., x_n) = \frac{P(C_j) \prod_{i=1}^{n} P(x_i|C_k)}{\sum_{j=1}^{2} P(C_j) \prod_{i=1}^{n} P(x_i|C_j)}$$

où $P(C_k|x_1, x_2, ..., x_n)$ est la probabilité conditionnelle que l'équipe k gagne étant donné les caractéristiques $x_1, x_2, ..., x_n$, $P(C_k)$ est la probabilité a priori que l'équipe k, et $P(x_i|C_k)$ est la probabilité de la caractéristique x_i étant donné que l'équipe k gagne.



On obtient une précision de 93.7% ce qui est moindre par rapport aux précédents modèles ce qui revient à dire que l'hypothèse d'indépendances de chacune des variables explicatives est fausse.

4.4 Forêt aléatoire

La forêt aléatoire (ou random forest en anglais) est une méthode d'apprentissage supervisé qui combine plusieurs arbres de décision pour améliorer les performances de prédiction et réduire le surapprentissage. Chaque arbre est construit sur un sous-ensemble aléatoire des données d'entraînement et un sous-ensemble aléatoire des caractéristiques. L'ensemble des prédictions des arbres est ensuite agrégé pour obtenir une prédiction finale. La probabilité de prédiction pour une classe donnée est calculée comme suit :

$$P(C_k|x_1, x_2, ..., x_n) = \frac{1}{T}\sum_{i=1}^{T} I(f_i(x_1, x_2, ..., x_n) = C_k)$$

où $P(C_k|x_1, x_2, ..., x_n)$ est la probabilité de la classe C_k étant donné les caractéristiques $x_1, x_2, ..., x_n$, T est le nombre d'arbres dans la forêt, $f_i(x_1, x_2, ..., x_n)$ est la prédiction de l'arbre i pour les caractéristiques $x_1, x_2, ..., x_n$, et $I(.)$ est la fonction indicatrice qui prend la valeur 1 si l'expression entre parenthèses est vraie et 0 sinon.

On a généré deux forêts différentes, la première avec les paramètres de base de la fonction RandomForestClassifier de sklearn et la seconde où les différents paramètres ont été choisi par validation croisée, ces paramètres étant 126 arbres, 10 échantillons minimum par feuille, 40 feuilles par noeud et une profondeur maximale de 17.

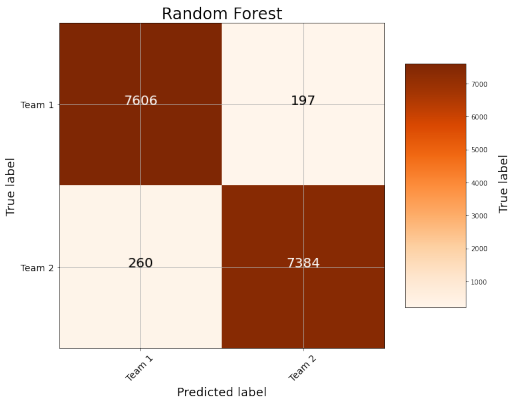


FIGURE 4.8 – Forêt de base

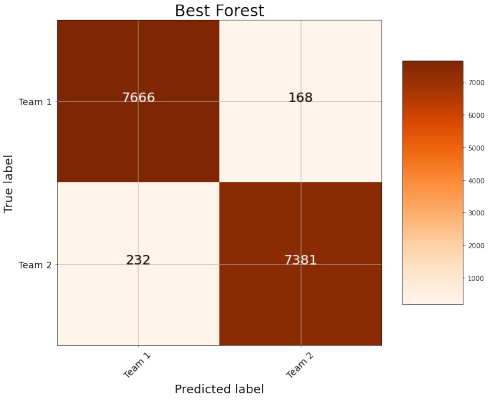


FIGURE 4.9 – Forêt obtenue par CV

On obtient des précisions de 97.1% pour la forêt avec les paramètres de base et 97.3% pour la forêt obtenue par validation croisée, l'amélioration est moindre mais au vu de la précision déjà élevée du premier modèles c'est un bon point, d'autant plus que la meilleure forêt réduit le taux de faux positifs dans les deux classes différentes.

4.5 Réseau de Neuronnes

Par la suite, on a essayé un réseau de neuronnes assez simple avec trois couches denses de respectivement 64, 32 et 1 couches, les deux premières couches ont pour fonction d'activation ReLU tandis que la dernière a pour fonction d'activation une sigmoid. On entraine ensuite le modèle sur cinquante epochs avec une taille de batch de 16.

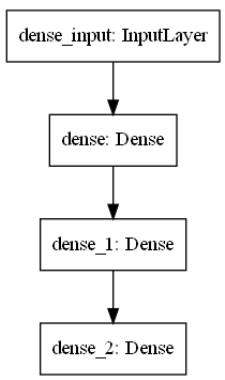


FIGURE 4.10 – Réseau de neurones

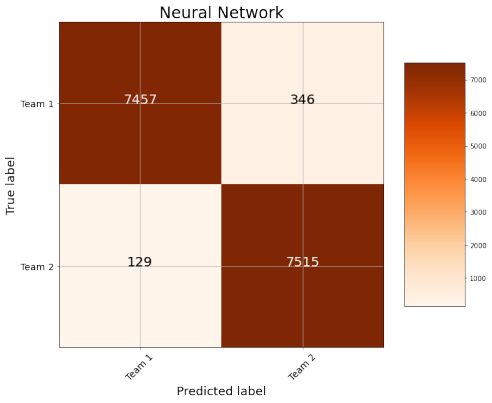


FIGURE 4.11 – Matrice de confusion

On obtient une précision de 96.9% ce qui reste de l'ordre de grandeur des meilleurs modèle mais qui est néanmoins moins bien que les forêts. On remarque néanmoins sur la matrice de confusion que le taux de mal classés est près de trois fois plus élevé pour la prédiction de l'équipe 2 qui gagne quand l'équipe 1 gagne que pour l'inverse.

4.6 XGBoosting

XGBoost est une méthode de Machine Learning qui se concentre sur l'utilisation de l'apprentissage par renforcement pour construire un ensemble de modèles prédictifs. Contrairement aux autres méthodes de Machine Learning, tels que les réseaux de neurones, les arbres de décision et les random forests, XGBoost utilise une approche de boosting qui combine plusieurs modèles plus simples pour former un modèle plus puissant et plus précis. De plus XGBoost permet de calculer l'importance de chaque variable ce qui permet de comprendre quelle variable influe le plus sur l'équipe gagnante.

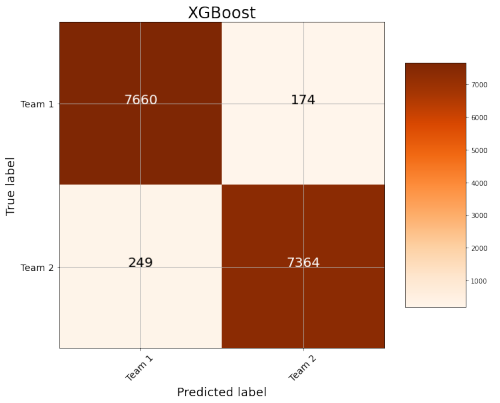


FIGURE 4.12 – Matrice de confusion

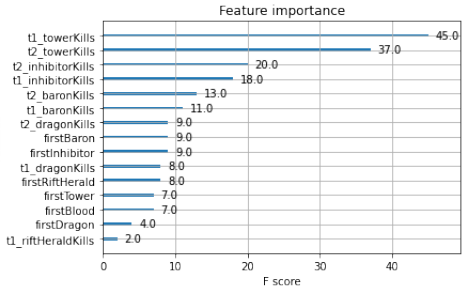


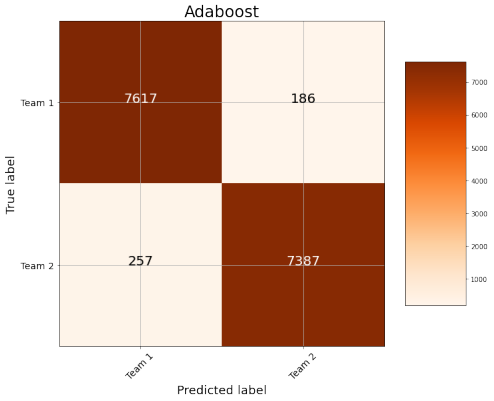
FIGURE 4.13 – Feature importance

On obtient une précision de 97.2% ce qui reste dans l'ordre de grandeur des meilleurs estimateurs mais qui se place en deuxième position cette fois-ci. En outre on remarque que la variable la plus importante est le nombre de tour tuées par équipe et ce de très loin.

4.7 Adaboost

Adaboost (Adaptive Boosting) est une technique de Machine Learning d'ensemble learning, similaire à XGBoost dans son objectif de combiner plusieurs modèles plus simples pour former un modèle plus puissant et plus précis.

Cependant, Adaboost utilise une approche différente de celle de XGBoost, en utilisant des poids pour donner plus d'importance aux exemples mal classés, tout en diminuant l'importance des exemples bien classés. Les modèles individuels sont créés en série, en ajustant les poids des exemples à chaque étape pour se concentrer davantage sur les exemples mal classés. Etant donné que le meilleur estimateur était la forêt aléatoire, on propose de faire un bosot adaptatif avec 200 fois la forêt en question.



On obtient au final une précision de 97.1% ce qui n'est pas spécialement plus convaincant tandis que la matrice de confusion est pire sur tout les points que pour le XGBoost.

4.8 Aggrégation d'expert

Etant donné le grand nombre de modèle déployés, nous avons décidé d'essayer une aggrégation d'expert en accordant le même poids à chaque modèle. Pour chaque partie, nous avons recueilli la prédiction de chacun des modèles et le vote majoritaire sera la prévision.

Due à la diversité des matrices de confusion et que certains modèle avait plus tendance à se tromper sur un certain type de faux positif que l'autre cela pouvait s'avérer être une bonne solution. On obtient une précision de 97.2% ce qui n'est pas autant qu'attendu, certainement parce que les modèles se trompent sur les mêmes parties.

5 Discussion des résultats

Après avoir essayé de nombreux modèles, le meilleur modèle qui en ressort est la forêt aléatoire avec les paramètres obtenus par validation croisée avec une précision de 97.3% ce qui est relativement élevé. La plupart des modèles ayant une précision relativement proche, on trace la courbe ROC qui représente la performance d'un modèle de classification binaire en fonction de ses seuils de classification. Elle permet de visualiser la capacité du modèle à discriminer les classes positives et négatives en fonction des faux positifs et des vrais positifs.

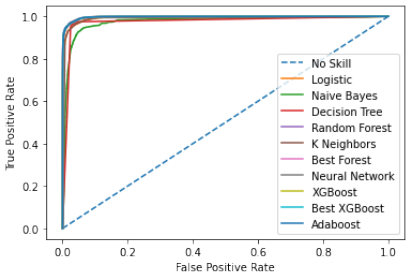


FIGURE 5.14 – Courbe ROC

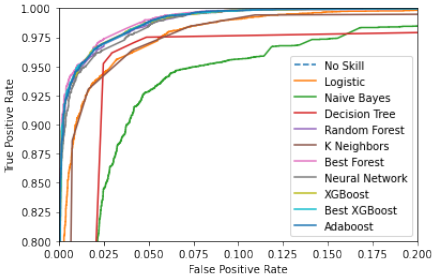


FIGURE 5.15 – Courbe ROC zoomé

Dans ce graphique nous avons affiché le courbe ROC de modèle donc nous n'avons pas parlé dans le rapport comme K-Neighbors. La forêt optimale est meilleures en chaque point que les autres estimateurs. Si les différents modèles ont des taux de précision similaires, cela peut indiquer que les données sont suffisamment claires pour que les modèles puissent facilement identifier des motifs. Cela peut également indiquer que les données sont suffisamment grandes pour que les différences entre les modèles ne soient pas significatives.

Pour conclure, les modèles avaient des performances similaires et dès lors que la complexité de ce celui-ci passait un certain seuil, il atteignait une précision de l'ordre de 97%. Cela provient du jeu de données qui était "trop facile" à apprendre, il pourrait être intéressant à l'avenir de créer un modèle qui prédit la gagnant d'une partie avec des informations obtenues seulement avant un certain instant t donné.