Projet 2021 Data Science

Antoine Toullalan, Rosa Mendas

Descriptif du poster

Dans ce projet, nous nous plaçons du côté developpeur et tentons de résoudre 4 problèmes. Les deux premiers sont résolus grâce à un apprentissage supervisé, tandis que les deux derniers par un apprentissage non supervisé.

Problème 1: Supervisé

On souhaite déterminer la popularité dont fera preuve une nouvelle application mise sur le marché, autrement dit, le futur nombre d'installations et le rating. On tentera de résoudre cette problématique grâce à un apprentissage supervisé.

Modèle 1

Afin de résoudre ce problème, nous allons utiliser les perceptrons: ClassifierPerceptron, ClassifierPerceptronBiais, ClassifierPerceptronKernel, Classifier-ADALINE. Afin d'exploiter les données fournies, il est nécéssaire de les "filtrer" d'abord. On sépare les données en données d'entrainement et données à tester puis on utilise la méthode ClassifierMultiOAA avec les données d'entrainement. Après avoir appliqué chaque perceptron sur les données d'entrainement, on teste leur efficacité sur les données tests.

Résultats et Analyse 1

Les précisions des classifieurs sur le nombre d'installations sont assez faibles : environ 10% de prédictions correctes. En effet, le succès d'une application dépend d'autres paramètres plus subjectifs que la categorie, la taille, la version d'Android... de l'application. Mais les classifieurs prédisent à 60% quelles vont être les futures critiques.
On a donc vu que nos algorithmes marchent plutôt bien pour prédire les "Rating" plutôt que le nombre d'installations.

Problème 2: Supervisé

On souhaite déterminer quel succès une application aura (le nombre d'installations et ses critiques) avant qu'elle soit "lancée". On pourra d'une part, utiliser les perceptrons. D'autre part, on tentera de prédire le nombre de téléchargements grâce aux arbres de décision.

Modèle 2

Premièrement, on importe d'abord les données "nettoyées" de googleplaystoreCLEANpb1.csv, on considère que pour chaque description d'app, le label est dans la colonne "Installs" puis dansla colonne "Rating". On sépare les données en données d'entrainement et données à tester. On utilise la méthode ClassifierMultiOAA avec les données d'entrainement. Après avoir appliqué chaque perceptron sur les données d'entrainement, on teste leur efficacité sur les données tests. Pour le cas de l'arbre de décision, on crée un arbre de décision qui nous donne dans quelle catégorie se situe une app au niveau du nombre de téléchargements (les catégories sont : de 0 à 100, de 100 à 1000, de 1000 à 10000... jusqu'à 1 000 000 000), il y a donc 9 catégories.

Résultats et Analyse 2

On voit que les precisions des classifieurs sur le nombre d'installations sont assez faibles : environ 10% de prédictions correctes. Dans le 2eme arbre qu'on construit on arrive à avoir 63% de prédictions correctes pour prédire si une app aura plus ou moins d'un million de téléchargements car la valeur de coupure est 5 (les catégories sont 0 à 1 million (non inclus) et un million (inclus) à un milliard). Ce n'est pas très performant car comme on a 2 catégories, un classifieur aléatoire aurait une accuracy de 50%. Donc notre classifieur est un peu plus performant qu'un classifieur aléatoire.
On remarque que lorsqu'on rajoute la colonne "Rating", l'accuracy monte à 75% mais le developpeur n'a accès qu'aux données de l'application avant le lancement sur GooglePlayStore.
Donc avec un arbre de classification qui prédit si l'app aura plus ou moins de 1 000 000 de téléchargements, on arrive à avoir une performance légèrement supérieure à celle d'un classifieur aléatoire mais cela reste assez faible car le succès d'une application dépend aussi (et surtout) de données plus subjectives que celles que nous avons.

Projet 2021 Data Science

Antoine Toullalan, Rosa Mendas

Problème 1: Non supervisé

On se positionne du côté developpeur de l'application. On souhaite déterminer la note qu'une application pourrait obtenir. A partir des données fournies, nous allons tenter de déterminer le rating d'une application. Pour ce faire, nous allons baser notre analyse sur une classification non supervisée, notamment grâce à la méthode Kmoyennes.

Modèle 1

Afin de résoudre la problématique: On détermine les paramètres qui influent sur la note par rapport aux deux fichiers donnés. En réalité, au moment du lancement de l'application, le developper n'a aucun retour, on fait l'hypothèse d'avoir quelques retours avant le lancement.
On normalise l'ensemble des valeurs du df et on applique Kmoyennes au nouveau df normalisé. On construit un dictionnaire contenant l'ensemble des moyennes et écarts-types de chaque colonne pour chaque affectation. Enfin, on crée une fonction qui prend les caractéristiques de l'application en entrée, cherchera l'affectation qui lui convient le plus et retournera une notation suivie d'un écart-type.

Problème 2: Non supervisé

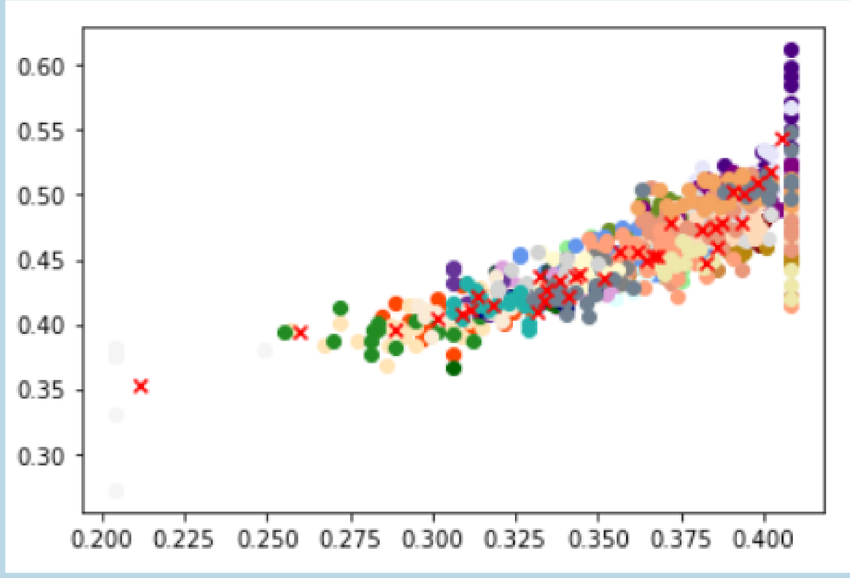
On souhaite déterminer le prix d'une application à mettre sur le marché. A partir des données fournies, nous allons tenter de déterminer à quel prix pourrait être lancée une application. Pour ce faire, nous allons baser notre analyse sur une classification non supervisée, notamment grâce à la méthode Kmoyennes.

Modèle 2

On doit d'abord déterminer les paramètres qui influent sur le prix. On normalise l'ensemble des valeurs du df et on applique Kmoyennes au nouveau df normalisé. On construit un dictionnaire contenant l'ensemble des moyennes et écarts-types de chaque colonne pour chaque affectation. Enfin, on crée une fonction qui prend les caractéristiques de l'application en entrée, cherchera l'affectation qui lui convient le plus et retournera une fourchette de prix.

Résultats et Analyse 1

Les résultats sont rendus en fonction du sentiment, sentiment subjectif et sentiment populaire. On remarque qu'ils sont compris entre l'ecart de base du cluster +- écart-type et sont cohérents comparé au df.
L'algorithme Kmoyennes nous a permis de déterminer de manière rapide et simple le Rating d'une application accompagné d'une marge. Le rating peut néanmoins être faussé par l'entrée de données non fiables: il faudrait donner en entrée la moyenne des sentiments et pas uniquement un des sentiments trouvé dans le df.



Résultats et Analyse 2

On remarque que les données sont réparties pour la plupart de manière cohérente au sein des clusters. On retrouve les prix recherchés initialement en fonction des arguments passés en paramètres.
Les résultats sont rendus en fonction de la catégorie, de la taille, mais aussi de la dernière version réalisée étant les critères les plus influents sur le prix. Les résultats sont compris entre prix de base du cluster +- écart-type.
Toutefois, Kmoyennes ne permet pas, de gérer les formes complexes. Quelques prix au sein d'une même catégorie ont des valeurs beaucoup trop éloignées de la moyenne et ne sont pas pris en compte dans le bon cluster.