Projet de département : "Toutes les comédies françaises sont-elles bleues ?"

Maxime Poli, Nicolas Schlegel, Alex Fauduet, Virginie Loison

6 Mai 2020

1 Description des données

Nous avons fait des manipulations sur la base de donnée pour éliminer les genres peu représentés.

La base de donnée résultante contient 11935 films, répartis dans 7 genres : Action, Animation, Comédie, Comédie dramatique, Documentaire, Drame, Thriller-Policier. La répartition par genre est la suivante :

- Action : 885 films
- Animation : 710 films
- Comédie : 2332 films
- Comédie dramatique : 1086 films
- Documentaire : 1401 films
- Drame : 4225 films
- Thriller-Policier : 1296 films.

De ces donnée, nous avons extrait un training set contenant 595 films de chaque genre, et un testing set contenant 105 films de chaque genre.

2 Méthodes utilisées et premières analyses

2.1 Posters bruts + kNN

2.2 PCA sur les features

En grande partie à cause de la très grande dimension des features après application de resnet sur les features, il nous a semblé pertinent d'appliquer une PCA, permettant de grandement réduire la taille des objets manipulés avec une parte minimale d'information. De plus, notre problème souffre très peu du problème habituel de la PCA qu'est la perte d'interprétabilité des features, comme nous avions déjà grandement abandonné cette idée en passant dans un réseau de neurones. De plus, le fait d'appliquer un kNN aus features et d'afficher les plus proches voisins de posters tests nous donne une idée qualitative des features extraits.

Conformément à ce que nous espérions, cette méthode est très efficace pour notre problème : sur les 32 000 features fournies par resnet, 700 suffisent à expliquer 80% de la variance de nos données et 2700 à en expliquer 95%.

De plus, on obtient des résultats analogues ou même légèrement meilleurs dans les différents algorithmes après une PCA, justifiant cette méthode (figure 1).

2.3 ResNet + kNN

Pour construire des features, nous utilisons le réseau ResNet 50v2. Nous appliquons ensuite un k-NN sur ces features. Pour l'instant, nous utilisons comme feature l'output de l'avant-dernière couche de ResNet. Chaque feature a donc une taille 32768.

Nous avons appliqué une PCA aux outputs de ResNet pour diminuer la taille des features. pour $n_{composantes} = 0.95$, les features ont ainsi une taille de 2796. Voici les matrices de confusion pour k=7 avec et sans PCA :

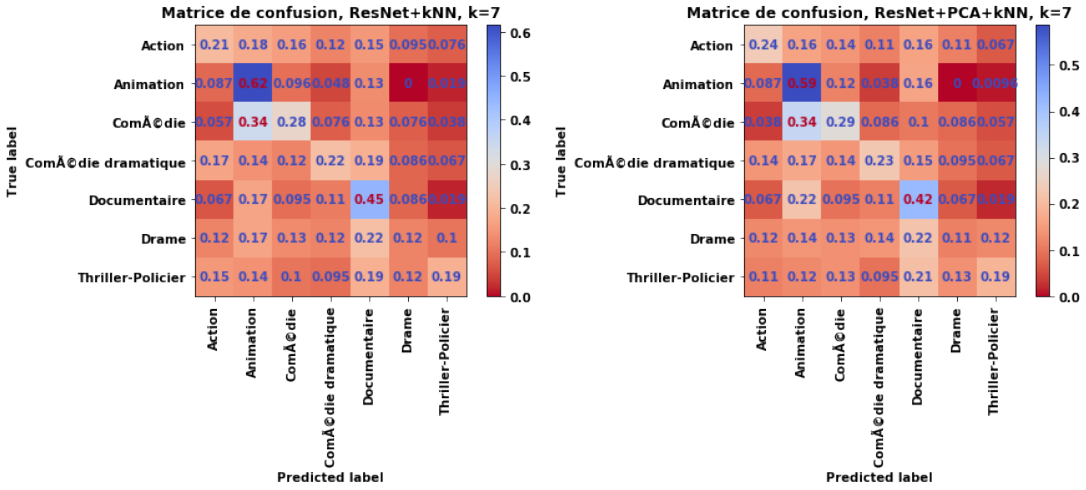
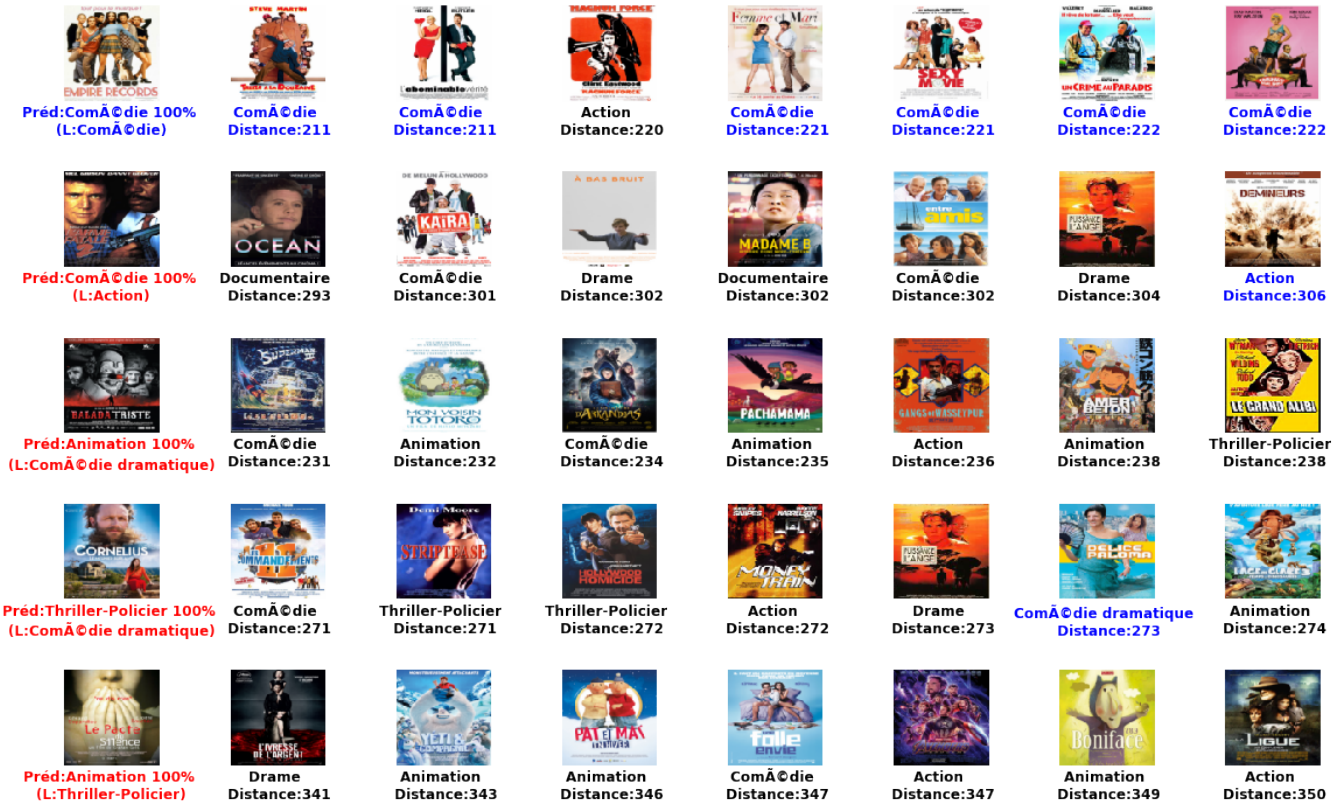


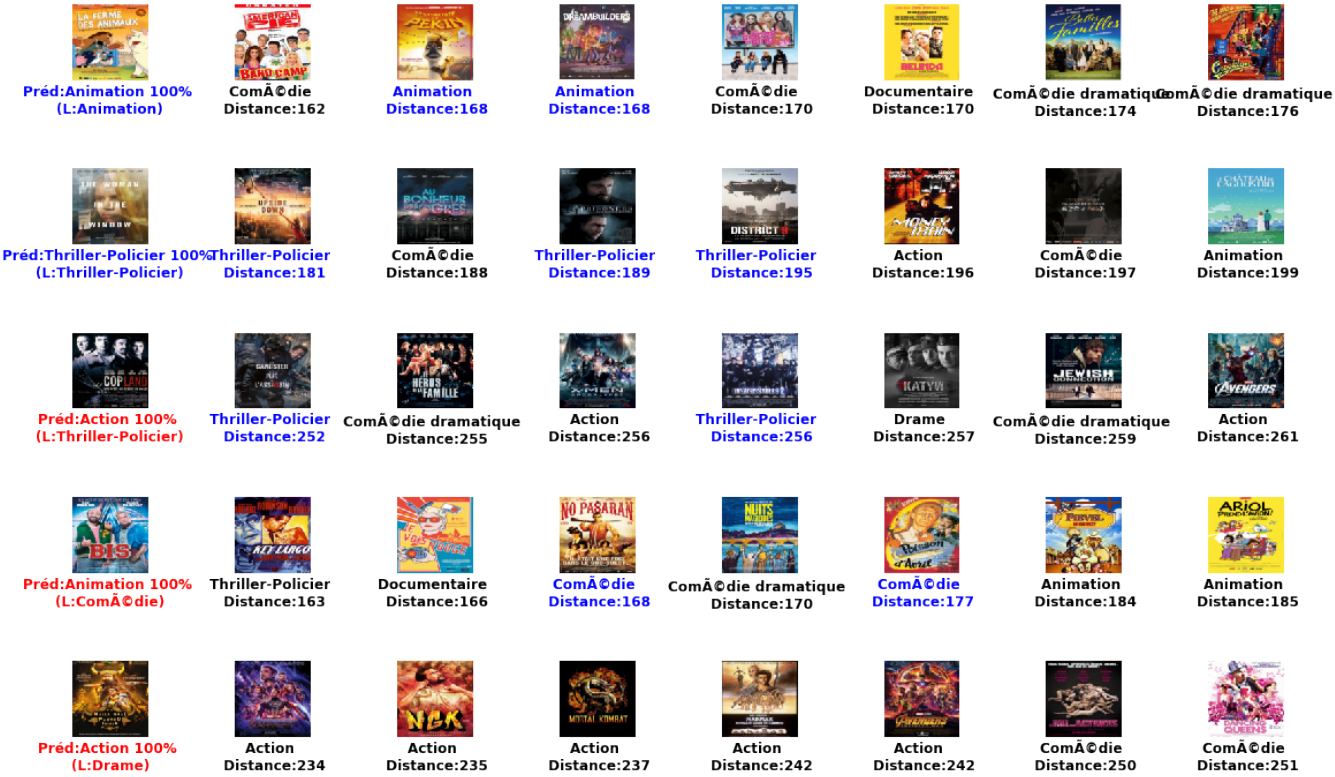
FIGURE 1: Matrices de confusion d'un k-NN sur l'output de ResNet, avec et sans PCA

Page suivante : on affiche les plus proches voisins de quelques posters tests, avec et sans prétraitement par PCA.

Questions On a remarqué que le knn fonctionnait mieux sur la PCA sur les posters non standardisés, bien que ça semble contre-intuitif.
Aurait-on des meilleurs résultats en prenant comme output d'autres couches de ResNet ?
Nous avons remarqué que certains posters se retrouvent plusieurs fois dans les voisins de posters . Par exemple, sur la figure (b) de la page suivante, le poster "Seuls ensemble" apparaît aux lignes 3 et 4. Y a-t-il une explication à cela ?



(a) Plus proches voisins, ResNet + kNN



(b) Plus proches voisins, ResNet + PCA kNN

2.4 Histogramme de couleurs + kNN

Nous avons voulu tester le k-NN sur d'autres features que l'output de ResNet, pour pouvoir comparer. Nous l'avons testé sur les histogrammes LAB et RBG. Les résultats pour ces deux histogrammes sont peu satisfaisants, comme on peut le voir à leurs matrices de confusion.

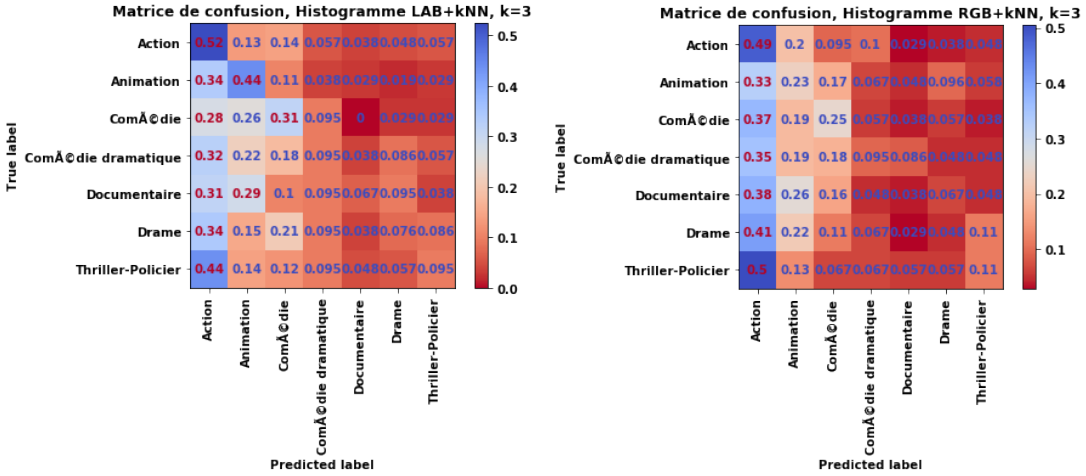


FIGURE 3: Matrices de confusion d'un k-NN avec des histogrammes de couleurs commes features

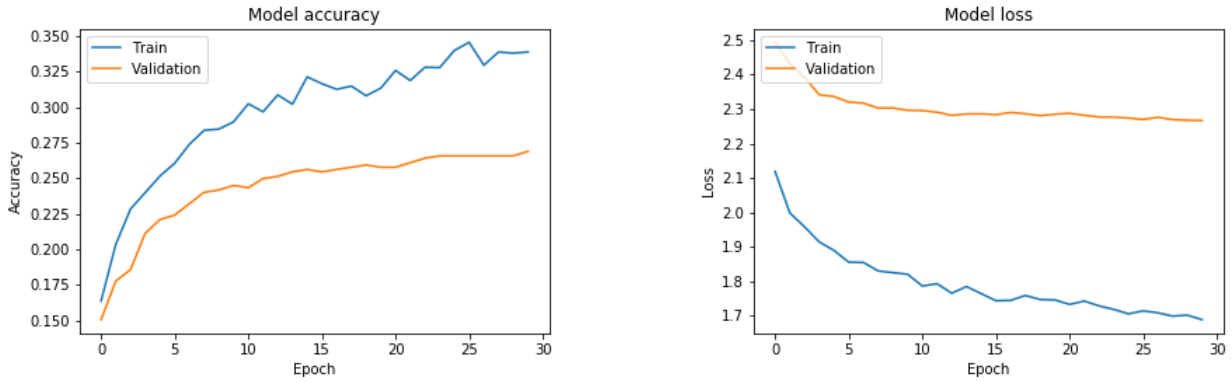
La prépondérance du genre "Animation" dans les prédictions nous fait penser à une indécision. Le k-NN, indécis pour sa prédiction, renvoierait le premier genre dans l'ordre alphabétique. Nous n'avons pas poursuivi l'étude sur les histogrammes. D'autres features marchent mieux, et nous souhaitons y consacrer le temps qu'il nous reste.

2.5 Transfer learning avec ResNet

Nous avons tout d'abord tenté une approche utilisant du *transfer learning* avec un ResNet. Le réseau a donc été implémenté de la manière suivante :

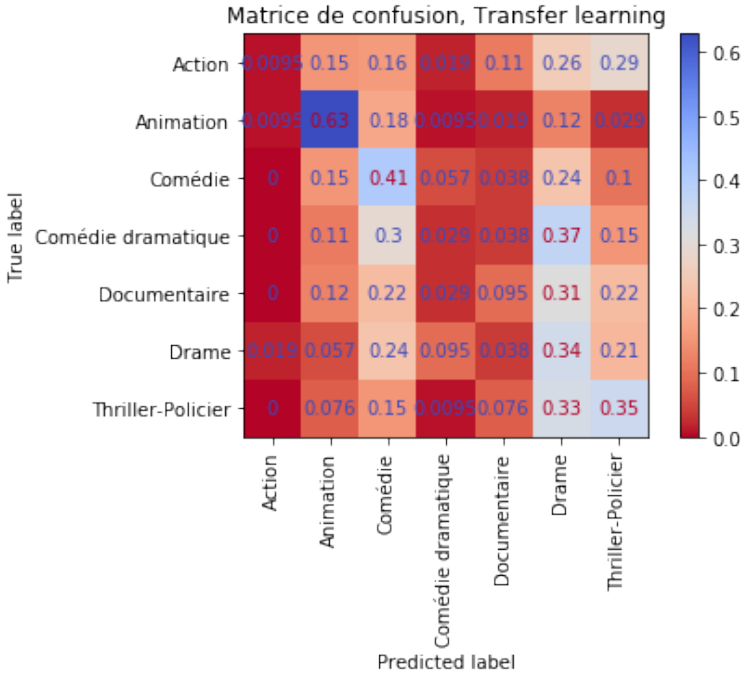
ResNet50V2 + GlobalAveragePooling2D + Dense(nb_genres) (avec une activation *softmax*).

La *loss* est la *categorical cross-entropy*, l'optimiseur est Adagrad. Les résultats qui suivent sont obtenus avec des *batchs* de taille 16, un ensemble d'entraînement de 3540 posters de taille 100×100 et un ensemble de validation de 625 posters. **Testing accuracy : 27%**



Validation loss décroît très peu. Ici le ResNet a été laissé figé, l'apprentissage ne s'est fait que sur nos couches rajoutées. Est-ce qu'on dégèle certaines couches du ResNet après avoir fait un certain nombre d'epochs ? (comme ce qui est fait ici : `https://www.tensorflow.org/tutorials/images/transfer_learning`).

Est-ce qu'on peut améliorer les performances en changeant de loss ? Ou bloqués par la faible taille du dataset ? Résultats très différents des autres approches : très mauvais sur Documentaire alors que c'est un genre bien prédit par



les autres méthodes, mais un peu meilleur sur Animation, Drame, Thriller-Policier. Prédictions très tranchées : un genre se distingue souvent ; que ce soit le bon ou un mauvais. Intérêt à tester d'autres réseaux ? (MobileNet, VGG16, ...). Si plusieurs sujets sur un même poster : ne correspond plus au cadre d'Imagenet. A quel point ça joue ?

Grosses disparités suivant les genres : très bons résultats sur Animation (0.54 de F1-score), pas du tout sur Comédie dramatique (0.05 de F1-score). F1-score micro de 0.27 (*Calculate metrics globally by counting the total true positives, false negatives and false positives*) et macro de 0.23 (*Calculate metrics for each label, and find their unweighted mean*). Discuter du F1-score et comment bien l'interpréter.

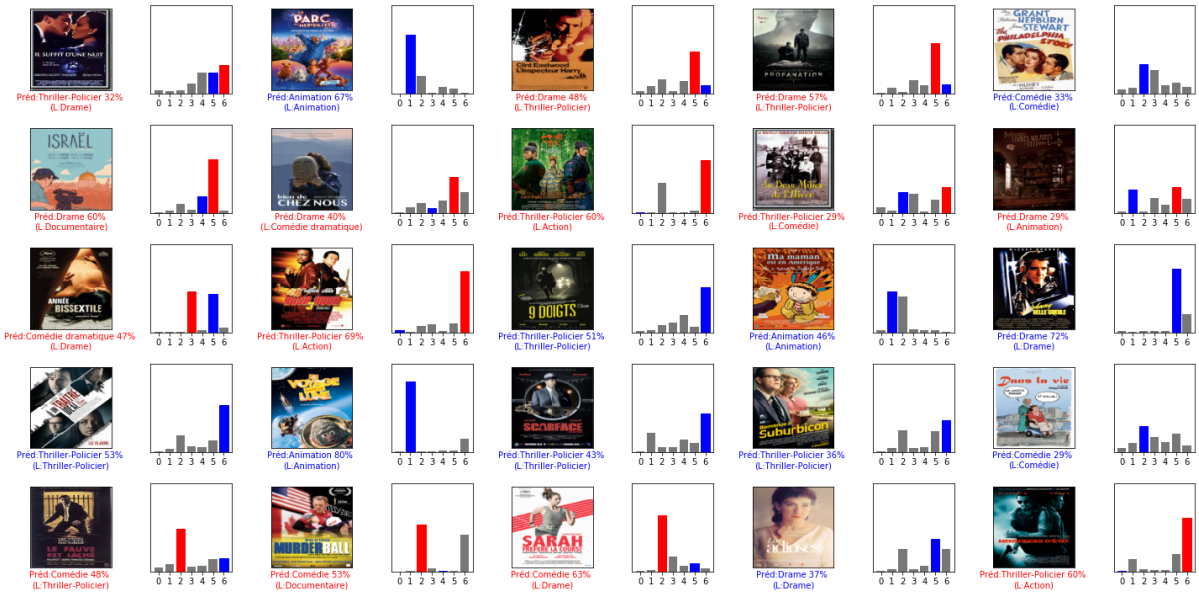
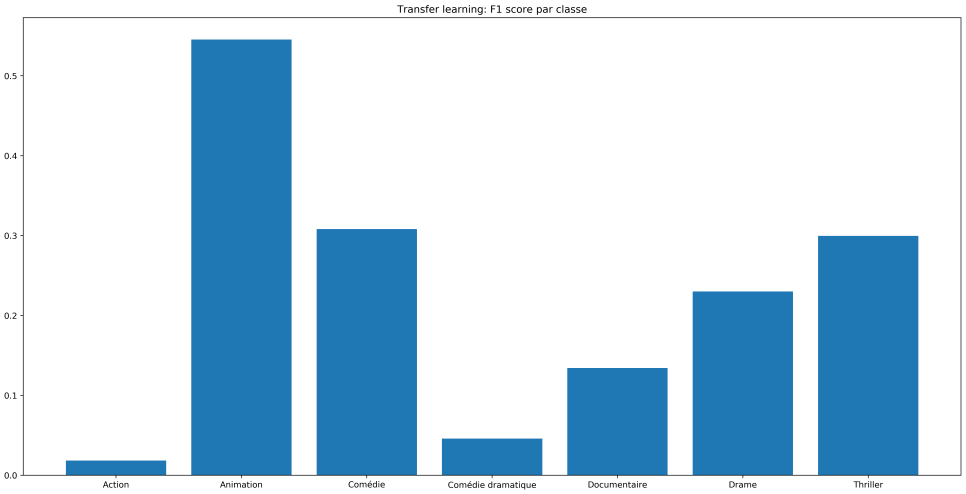


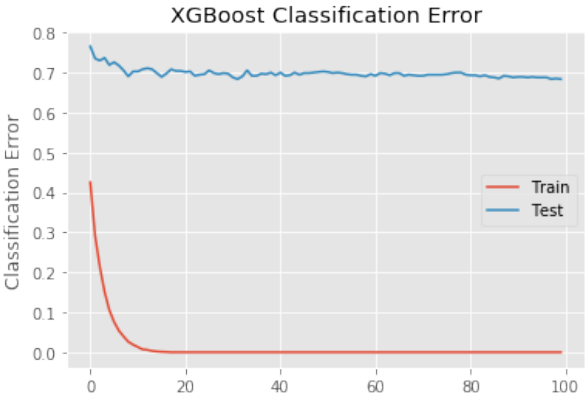
FIGURE 4: Quelques résultats du transfer learning sur le testing set

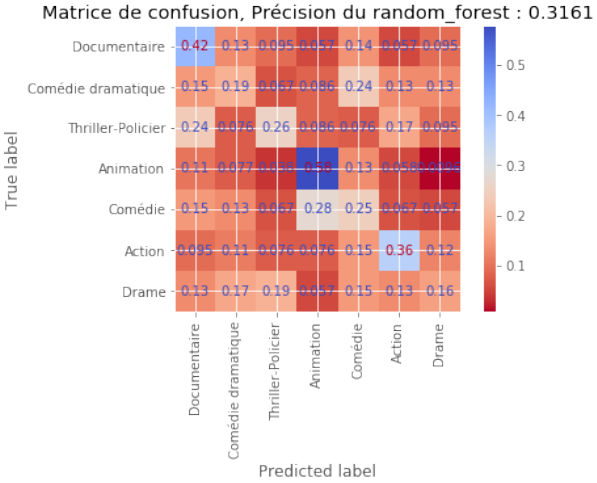


2.6 ResNet + Random Forest

Une autre approche a été d'appliquer une random forest aux features que nous renvoyait le resnet. Voici les résultats que l'on a obtenus :

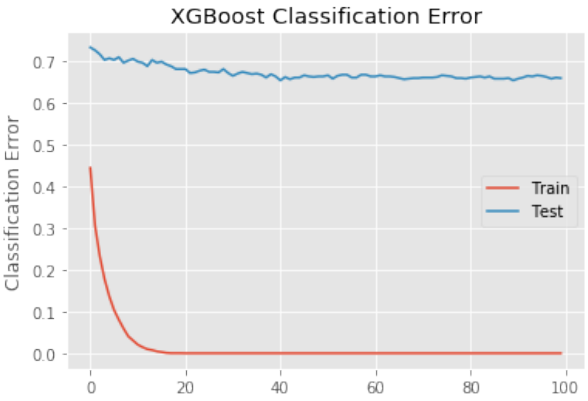
Random Forest sur la dernière couche en gardant 98% de la variance avec la PCA : Pour cette première méthode,

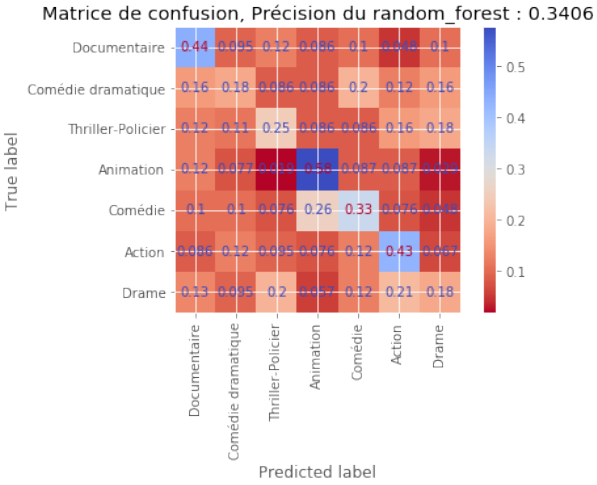




Bien que le modèle nous donne une des meilleures précisions parmis celles que nous avons obtenu jusqu'alors, nous avons remarqué que le modèle overfittait énormément. C'est pour cela que nous avons encore diminué la variance conservée avec la PCA.

Random Forest sur la dernière couche en gardant 80% de la variance avec la PCA : Ce modèle nous donne un encore meilleur résultat.





Nous avons essayé d'optimiser les hyperparamètres que nous utilisons avec la bibliothèque hyperopt, mais jusqu'alors nous nous sommes heurtés à des problèmes techniques avec la machine virtuelle.

L'optimisation des hyperparamètres (c'est à dire les paramètre de la random forest, mais aussi la couche du resnet que l'on prend ainsi que la variance que l'on conserve avec la PCA) constitue la prochaine étape en ce qui concerne la partie sur la random forest.

Ensuite, pour essayer de comprende le résultat que l'on obtiendra, nous disposons d'outil pour étudier l'importance des features et d'outils de visualisation d'arbres.

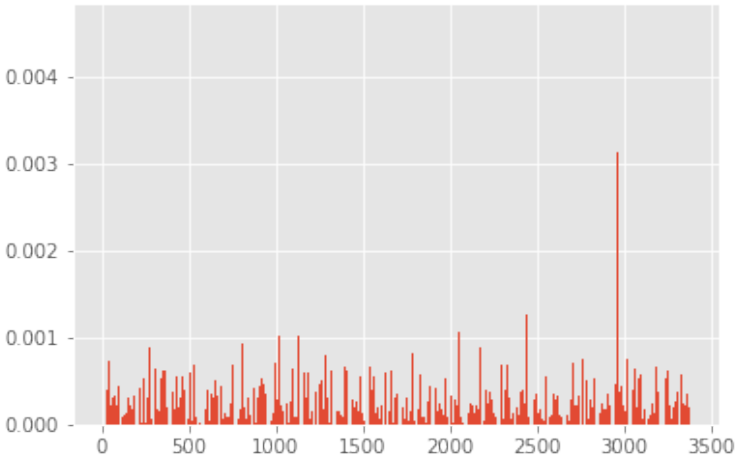


FIGURE 5: graphique qui trace l'importance des features pour le XGBoost, on constate qu'une feature se démarque, mais elle n'est pas pour autant si discriminante (elle vaut environ 0.03 alors que la somme des poids vaut 1

FIGURE 6: un exemple d'arbre