UNIVERSITÉ PARIS DAUPHINE

MASTER 1 MIAGE

Projet d'Analyse et Fouilles de Données

Réalisé par :
Lyes MEGHARA
Kamel SIDHOUM

Encadré par :
Mr Yann CHEVALEYRE

18 Mai 2017

I Introduction

La notion d'apprentissage supervisé a été évoquée depuis plus de cinquante ans. Cela a permis l'évolution concrète de l'intelligence artificielle au plus grand bonheur des informaticiens et étudiants. Le projet que nous allons exposer traite de la capacité de plusieurs algorithmes d'apprentissage supervisé à apprendre, puis à prédire.

Nous disposons pour cela d'une base de données fournie par LITISLAB dans le contexte d'une compétition nommée « *My Tailor is rich!* » composée de plusieurs milliers de textes en Anglais, chacun possédant des variables numériques prédictives, et la variable à prédire (i.e le niveau).

Notre objectif est d'établir un (voir plusieurs) classifieurs à partir de cette base de données. Nous allons répartir les données en deux ensembles, un sur lequel notre (nos) classifieur va « *apprendre* » et un autre ensemble sur lequel il va pouvoir « *prédire*» le niveaux d'Anglais.

II Outils de développement

Dans le cadre de ce projet, nous avons utilisé GitHub pour une meilleure collaboration, Jupyter Notebook avec les librairies Keras et Scikit-Learn
Pour finir ce rapport a été rédigé en utilisant LateX.

III Première approche

III.I Prétraitement

Une première approche a consisté a utiliser un réseaux de neuronnes artificiels et multiclasse pour procéder à l'apprentissage puis à la prédiction, mais avant cela un prétraitement (dite de preprocessing) des données était un passage obligatoire, en effet, les données ont plusieurs valeurs manquantes, tels que les NaN (Not a Number).
Après avoir remplacé ces dernières par des 0, il a fallu centrer-réduire les données c'est là que StandardScaler() nous a été utile.
Enfin, puisque les variables à prédire sont nominales (Niveaux allant de A1 à C2) il a fallu les encoder via L'Encodage one-hot, puis les transformer en valeurs numériques.

Il est à noter, que cette première approche ne tient pas compte du texte, mais uniquement des variables numériques.

III.II Construction du modèle

Une fois le prétraitement terminé, on découpe notre modèle en deux sous ensembles X_train et X_test avec une répartition de 60%, 40% .

Après plusieurs recherches et tests, nous avons opté pour le modèle *Sequential* de Keras, celui ci permet de paramétrer le réseau couche par couche sans trop de difficulté.

Comme citée lors du dernier cours, la fonction d'activation *Relu* a donné de bien meilleurs résultats comparé à la fonction *Sigmoid*.
Nous avons donc opté pour un réseau à 3 couches et c'est donc dans les deux premières couches (input et hidden) que notre choix s'est porté sur *Relu*.

En revanche, pour la dernière couche (output), d'après nos recherches puis, nos tests; il s'avère que la fonction d'activation*Softmax* ainsi que la fonction de perte *Categorical_crossentropy* étaient plus adaptés pour des classes multiples.

III.III Résultats des tests

Après plusieurs paramétrages, on constate qu'avec 100 itération, on arrive à des résultats très satisfaisants. Notre taux d'erreurs se situe entre 3% et 5%

Figure 1: Evaluation du modèle

```
# evaluation du modèle
scores = model.evaluate(X, dummy_y)
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]*100))
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

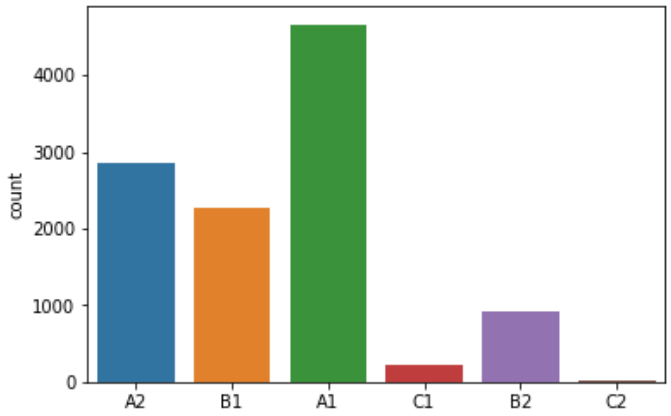
```
16386/16386 [==============================] - 0s 16us/step

loss: 5.04%

acc: 98.29%
```

Les prédictions sur X_test comparé aux valeurs réelles Y_test sont exposées sur les 2 figures suivantes :

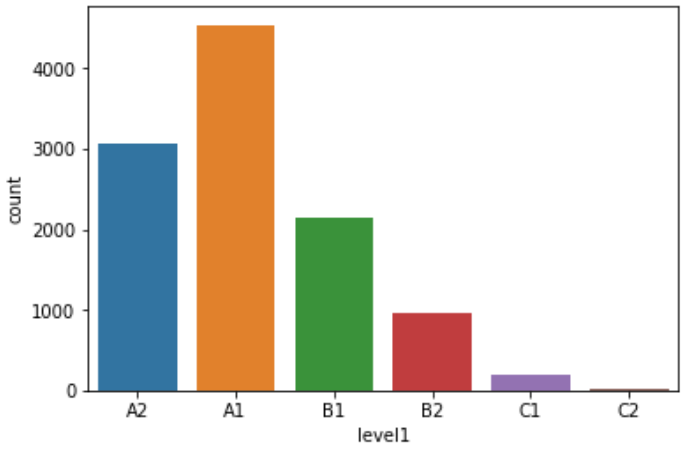
Figure 2: Prédictions sur X_test




```
{'A1': 4662, 'A2': 2844, 'B1': 2276, 'B2': 913, 'C1': 214, 'C2': 15}
```

Effectifs prédits par classe

Figure 3: Les valeurs réelles




```
{'A1': 4543, 'A2': 3072, 'B1': 2137, 'B2': 955, 'C1': 195, 'C2': 22}
```

Effectifs réels par classe

IV Seconde approche

V Autres Classifieurs

Cette fois-ci, en utilisant la librairie Scikit-Learn, on a pu tester et comparer divers classifieurs entre eux.
Les classifieurs testés sont : Régression Logisitque, Arbre de décision, KNN, LDA, Random Forest et d'autres encore.

Sans surprise, comme notre enseignant Mr Chevalyere nous l'avait recommandé, Le Random Forest est bien plus performant que les autres.

KNN aussi se démarque bien, mais comme il ne construit pas de modèle et est obligé de chargé tout en mémoire à chaque lancement, ce dernier est bien plus lent que les autres.

Tout les résultats détaillés sont consultables sur le fichier *ManyClassifiers.ipynb*

VI KMeans

Bien que le Clustering, et plus particulièrement KMeans ne soit pas une méthode d'apprentissage supervisée, mais plutot non supervisée, nous avons voulu tester et voir les résultats par nous meme, et ce en faisant abstraction de la variable à prédire (le niveau).

Nous nous basons sur k=6, (6 clusters, car 6 niveaux différents), les résultats obtenus sont présentés dans les figures suivantes :

Figure 4: Nuage de points au début

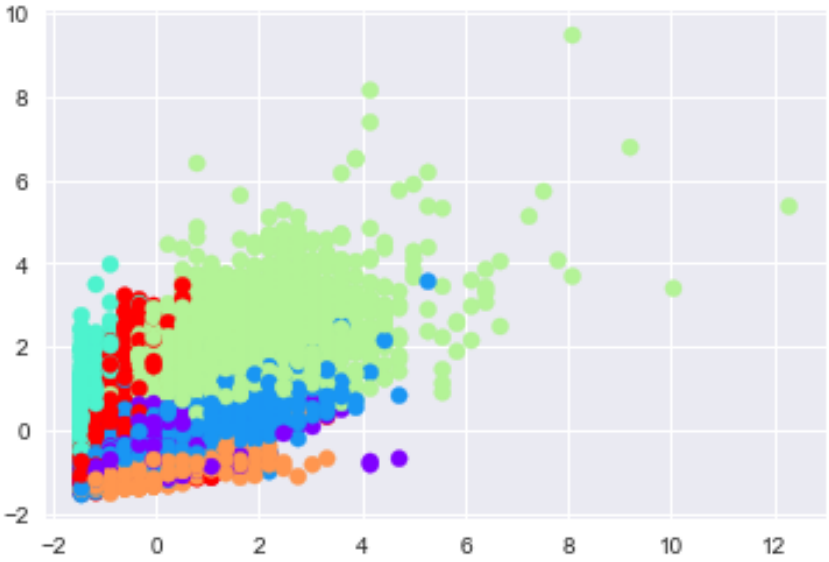
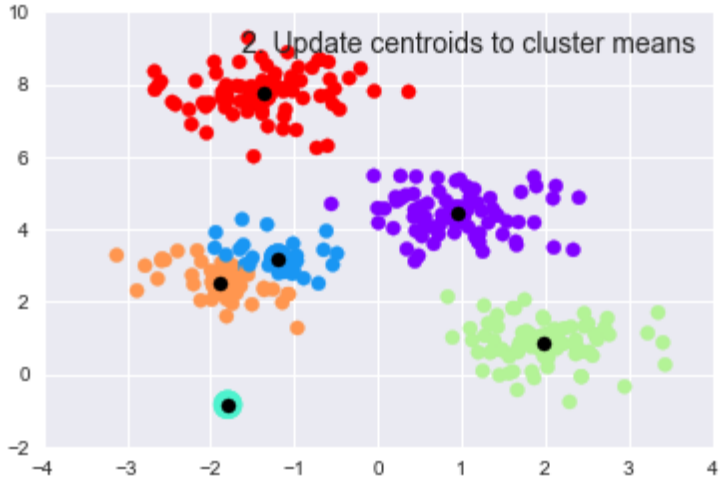


Figure 5: Nuage de points après 50 étapes



VII Webographie

- https://fr.wikipedia.org/wiki/Perceptron
- https://fr.wikipedia.org/wiki/Perceptron_multicouche
- alp.developpez.com/tutoriels/intelligence-artificielle/reseaux-de-neurones/
- https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-
- https://www.cs.cmu.edu/afs/cs/academic/class/15883-f15/slides/backprop.pdf
- http://www.dgp.toronto.edu/people/RMB/papers_old/p6.pdf
- https://www.labri.fr/perso/nrougier/downloads/Perceptron.pdf
- http://www.turingfinance.com/misconceptions-about-neural-networks/
- https://youtu.be/L7WNYvbvGBc