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))
```

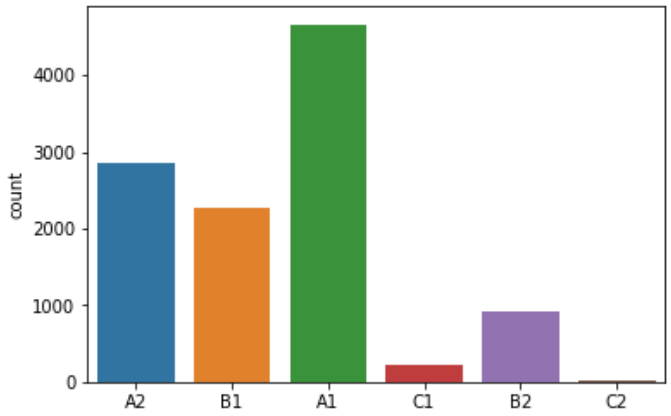
```
16386/16386 [==============================] - 0s 16us/step

loss: 5.04%

acc: 98.29%
```

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

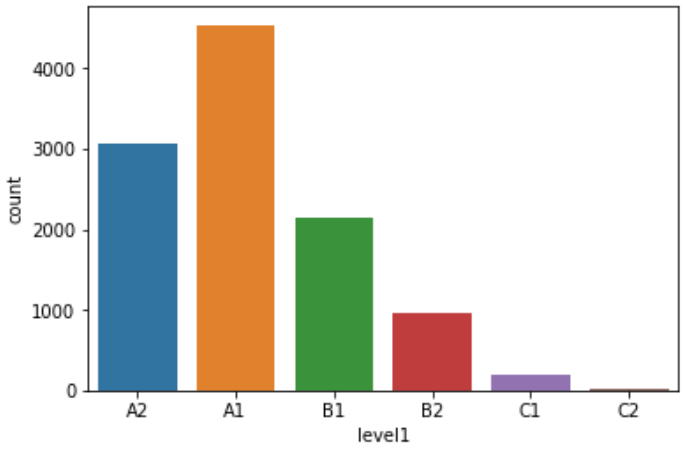
Figure 2: Prédictions sur X_test




```
{'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 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 charger tout en mémoire à chaque lancement, ce dernier est bien plus lent que les autres.

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

Table 1: Précision par classifieur

Regression Logistique	**Arbre de décision**	**KNN**	**LDA**	**Bayes Naif**	**Random Forest**
0.72%	0.72%	0.77%	0.73%	0.66%	0.78%

V 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

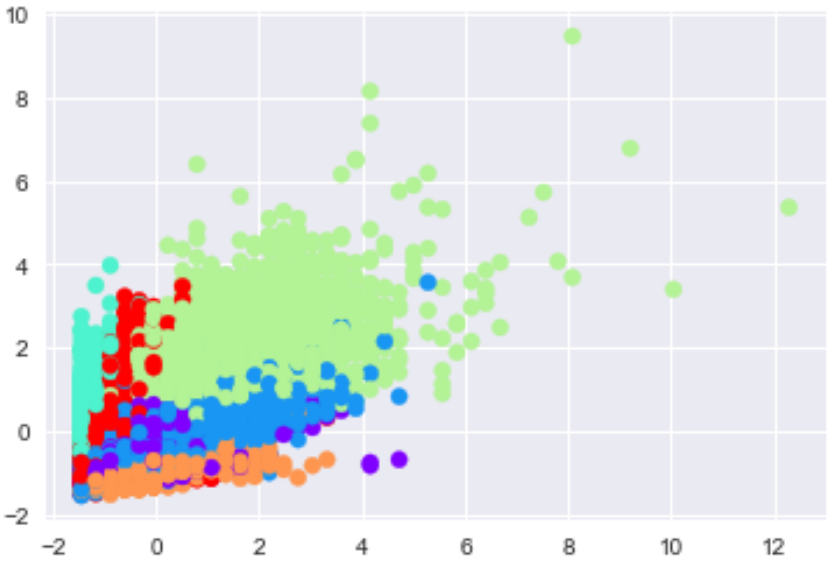
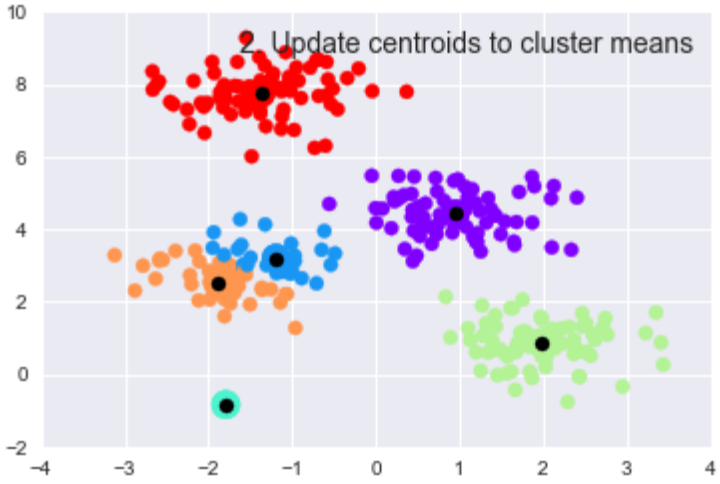


Figure 5: Nuage de points après 50 étapes



VI Seconde approche

VI.I Introduction

Dans cette partie nous allons analyser uniquement la partie texte, soit la colonne « fulltext » ainsi que la classification qui leurs sont associées. La classification de texte est utilisée dans différent domaine, que cela soit pour définir les sentiments émis par un texte, filtrer les « spams » etc... Notre but ici est donc d'assigner un texte à un des 6 classifications, nous ferons ici l'hypothèse que chaque texte sera associé à une et une seule catégorie.

VI.II Prétraitement

Dans notre cas, nous aurons en entré les textes en anglais « *fulltext* » et en sortie le niveau correspondant « *level1* ».
Nous avons aussi créé une colonne nommée « *category_id_df* » codant les niveaux d'anglais par un entier (de type Integer) car les variables sont souvent mieux représentées par des entiers que par des chaines de caractère.

Figure 6: Tableau représentant les textes selon leurs niveaux et catégories

	fulltext	level1	category_id
0	\r\r\n The Eiffel Tower The Eiffel Tower ...	C2	0
1	\r\r\n The Court Green burglar arrested A...	C2	0
2	\r\r\n Thank you for giving us the opport...	C2	0
3	\r\r\n The international AI conference ca...	C2	0
4	\r\r\n I believe that the creative writin...	C2	0

Il faut savoir que les classifieurs ainsi que les algorithmes d'apprentissage n'arrivent pas à traiter directement du texte brut, ils s'attendent à recevoir des vecteurs de caractéristiques numériques, on convertit donc durant cette phase les textes en entiers.

Une approche simple pour essayer d'extraire les caractéristiques d'un texte est tout simplement d'utiliser ce qu'on appelle communément « *a bag of*

words model », soit un modèle pour chaque texte où la fréquence d'apparition des mots est prise en compte et l'ordre des mots n'importe peu et donc nous obtenons pour chacun des 27 310 textes un total de 12 633 caractéristiques représentant le score pour différent unigram et bigram.
En utilisant la bibliothéque « *Sklearn* » cela nous donne accès à des fonctions pouvant faire un lien entre des termes et des niveaux d'anglais, ici nous avons choisi d'afficher deux unigrams et deux bigrams qui sont le plus corrélés à chaque niveau, nous obtenons :

Figure 7: Tableau représentant des unigrams et des bigrams selon le niveau

A1	A2	B1	B2	C1	C2
Live Busy	Feed Dog	Bowling Bottles	Discrimination Apply	Vote Council	Pettigrew Robot
Busy good Good evening	Talk phone Feed dog	Shots run Plastic bottles	Absolutely amazing Amazing job	Green business Student council	French revolution Emotionnal-intelligence

VI.III Construction du modèle

Après quelques recherches, nous nous sommes penchés sur le modèle « *LinearSVC* » donnant des résultats extrêmement satisfaisant, même meilleur que l'algorithme de « *RandomForest* », beaucoup de données sont affichés dans le programme *Etude-de-Texte.ipynb* , notamment la matrice de confusion ainsi que des tableaux montrant des erreurs de prédictions soit comment certains textes sont catégorisés dans un niveau qui n'est pas le leur.

Figure 8: Nous obtenons donc comme résultat

	precision	recall	f1-score	support
C2	1.00	0.28	0.43	18
C1	0.98	0.63	0.77	165
B2	0.92	0.91	0.92	795
B1	0.92	0.93	0.92	1744
A2	0.94	0.94	0.94	2537
A1	0.96	0.98	0.97	3754
avg / total	0.95	0.95	0.95	9013

VII Webographie

- https://medium.com/@pushkarmandot/build-your-first-deep-learning-neural-network-model-using-keras-in-python-a90b5864116d
- https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
- https://towardsdatascience.com/deep-learning-gender-from-name-lstm-recurrent-neural-networks-448d64553044
- https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
- https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/
- https://towardsdatascience.com/solving-a-simple-classification-problem-with-python-fruits-lovers-edition-d20ab6b071d2
- https://www.kaggle.com/mjamilmoughal/k-nearest-neighbor-classifier-to-predict-fruits
- https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f
- http://blog.aylien.com/first-text-mining-project-python-3-steps/
- https://www.digitalvidya.com/blog/an-introduction-to-text-analysis-in-python/
- https://fr.wikipedia.org/wiki/Encodage_one-hot/