Projet détection de langue.

Sciences des données 2.

Keroudine BELLADJO & Ameline FAVRAIS

Sommaire.

En premier lieu, afin que les fonctions puissent fonctionner, il faut importer différentes bibliothèques de python, à savoir:

- **Matplotlib**; qui va nous permettre de visualiser les données (tracer des graphiques)
- **Scikit-learn**; qui propose une implémentation de l'algorithme des k plus proches voisins.
- **Pandas**; qui va nous permettre de traiter nos données
- **Numpy** ; permet d'effectuer des calculs numériques et introduit une gestion facilitée des tableaux de nombres
- **Requests**; Gestion domaines et URLS internationales
- **Re**; permet d'utiliser des expressions régulières avec Python
- **BeautifulSoup**; est une bibliothèque Python d'analyse syntaxique de documents HTML et XML

Partie 1:

1) Création d'un ensemble de données en 5 langues.

Pour pouvoir établir une étude il faut d'abord:

- Choisir un texte suffisamment grand pour être étudié; ici nous avons choisi un texte sur Martin Luther King provenant de wikipédia.
- Le transcrire en plusieurs langues; ici nous avons décidé de choisir 5 langues (Français, Allemand, Anglais, Espagnol et italien).
- Prendre ensuite les 5 URL et extraire les textes.
- Une fois les textes extraits, il faut leur enlever les caractères spéciaux (parenthèses, point, chiffres, espaces etc) en créant une fonction qui va les filtrer, afin de pouvoir exploiter de meilleures données.
- Limiter tous les textes au même nombre de caractères, afin d'avoir le même nombre de fragments n pour tous; ici nous les avons limité à 40 000 caractères.

'vouslisezunarticledequalitpourlesarticleshomonymesvoirmartinlutherhomonymieetkingnedoitpastreconfonduavecmartinluthermodifie
rmodifierlecodemodifierwikidatamartinlutherkingjrpluscourammentappelmartinlutherkingnatlantaengorgielejanvieretmortassassinle
avrilmemphisdansletennesseeestunpasteurbaptisteetmilitantnonviolentafroamricainpourlemouvementamricaindesdroitscivіquesferven
tmilitantpourlapaixetcontrelapauvretilorganiseetdirigedesactionstellesqueleboycottdesbusdemontgomerypourdfendreledroitdevotel
adsgrgationetlemploidesminoritsethniquesilprononceundiscoursclbreleaotdevantlelincolnmemorialwashingtondcdurantlamarchepourle
mploietlalibertilsintituleihaveadreamcediscoursestsoutenuparjohnfitzgeraldkennedydanslalluttecontrelasgrgationracialeauxtatsun
isleprsidentlyndonbjohnsonparuneplaidoirieinfatigableauprsdesmembresducongrsarriverafairevoterdiffrentesloisfdralescommeleciv
ilrightsactdelevotingrightsactdeetlecivilrightsactdemettantjuridiquementfintouteslesformesdesgrgationracialesurlensembledesta
tsunismartinlutherkingdevientleplusjeunelauratduprixnobeldelapaixenpoursaluttenonviolentecontrelasgrgationracialeetpourlapaix
ilcommencealorsunecampagnecontrelaguerreduvitnametlapauvretquiprendfinenavecsonassassinatofficiellementattribujamesearlraydon
tlaculpabilitetlaparticipationuncomplotsonttoujoursdbattuesilsevoitdcernertitreposthumelamdailleprsidentielledelalibertparjim
mycarterenleprixdesdroitsdelhommedesnationsuniesenlamdailledorducongrsenetestconsidrcommelundesplusgrandsorateursamricainsdep
uislemartinlutherkingdayestunjourfriauxtatsunisdeuxcentresmartinlutherkingpourlactionnonviolenteexistentlunensuisselausanneet
lautreatlantadenombreuxautresmonumentsmusescolessontrpertorissouslenomdemartinlutherkingpartoutdanslemondemartinlutherkingest
lefilsdupasteurbaptistemartinlutherkingsretdalbertawilliamskingorganistedgliseilaunesuranechristinekingfarrisetunplusjeunefrr
ealbertdanielwilliamskingilnatauaubrunavenueatlantadansunemaisonquiatconserveettransformeenmusenationalquelquespasdelglisebap
tisteebenezeroprchesonpreilgranditauseindelamriquesgrgationnistedansunmilieuprivilgipourlpoquesapremireexpriencedelasgrgation
racialedatedesessixansquanddeuxcamaradesdejeublancsluidisentquilsnesontplusautorissjoueravecluisamreluiexpliquealorsquecestpa
rcequilssontmaintenantdansdescolessgrgationnistesblanchesaprsavoirsautdeuxannesdehighschoollyceetsansavoirofficiellementobten

Exemple d’un morceau du texte en français après toutes ces étapes:

2) Exploration des données.

Hypothèse: La distribution des caractères varie d'une langue à l'autre.

Pour pouvoir exploiter les données établies, il est préférable de créer un Dataframe du nombre de chaque caractère présent dans chacune des langues et de remplacer les valeurs « Nan » par « 0 ».

Voici un extrait de notre data obtenu par langue et par caractères.

	Italien	Espagnol	Français	Anglais	Allemand
a	4304	4728	3169	3276	2396
b	430	621	341	569	727
c	1651	1816	1374	1414	1087
d	1520	2086	1555	1669	1898
e	4523	5004	5618	4719	6604
f	425	345	376	791	676
g	893	610	559	1063	1368
h	454	521	435	2117	1640
i	4578	2885	3212	3239	3219
j	64	200	212	121	115
k	177	156	144	444	623
l	2580	2366	2406	1572	1378
m	1059	1094	1274	988	1191
n	3183	3082	3170	3102	4102
o	3401	3332	2239	2834	1262
p	1020	911	1075	689	362
q	116	261	447	26	5

Partie 2 : Représentation des données.

- Construction d'un diagramme en bâton pour avoir la dispersion des caractères pour chaque langue afin de déterminer ceux qui sont les plus fréquents et les plus discriminants d'une langue à l'autre.

- Il nous permettra par la suite de déterminer les caractères de notre ensemble d'entrainement.

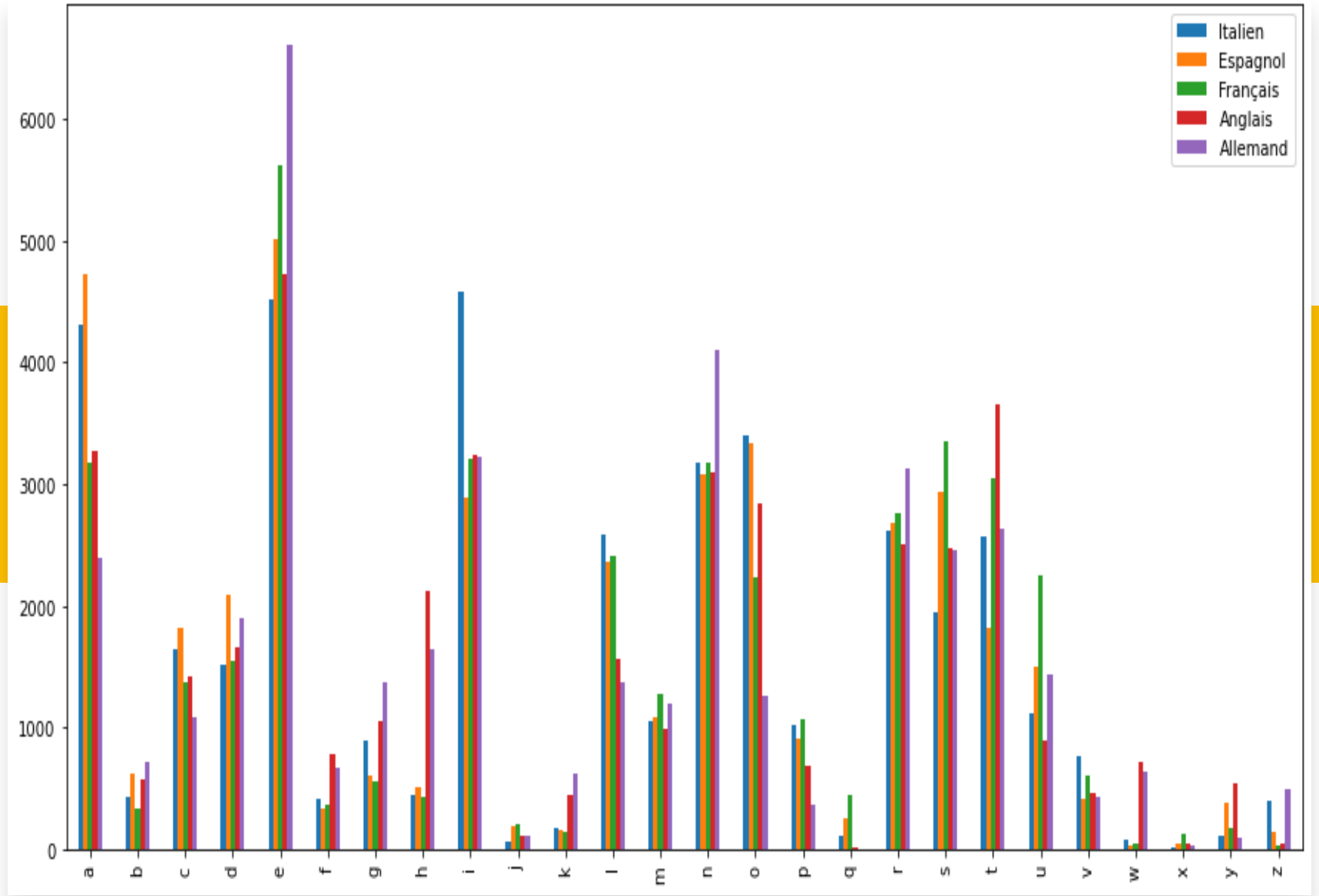


Diagramme en bâton du nombre de chaque caractère de chaque langue:

➢ On peut observer à travers ce graphique que la distribution des caractères varie d'une langue à l'autre. L'hypothèse est donc vérifiée,

➢ On pourra donc s'en servir pour faire de la détection de langues.

➢ On va donc faire de l'exploration de données pour tester cette hypothèse:

- ❖ quels sont les différents caractères utilisés dans les différentes langues?
- ❖ quelle est la fréquence d'apparition de chacun des caractères?

➢ Il nous faut donc choisir un d caractères parmi les plus fréquents et les plus discriminants.

➢ Ici, au vu de notre diagramme on prendra les 4 caractères les plus présents dans chacune des langues.

C’est-à-dire :

d=['a', 'e', 'n', 'u', 'p', 'i', 's', 'w','o', 'l', 'h','y','b','r','j','t']

➢ Ce qui répondra à notre première question.

Constitution d'un ensemble d'entraînement.

1) Fragmentation des textes.

➢ On dispose du même texte en 5 langues (Français,Anglais,Allemand,Italien,Espagnol.) avec 40 000 caractères (N).

➢Pour chaque texte, afin d'avoir une meilleure analyse, on va donc les découper en n fragments contenant tous l_min caractères.

➢Afin d'avoir un résultat cohérent, on a donc établi la racine carré de 40 000, ce qui donne donc l_min = 200. Cette manière de faire permet d'avoir une même taille de fragment pour tout le texte, avec un même nombre de caractères (200) par fragment.

➢ En résumé, les 5 textes sont alors découper en 200 fragments contenant chacun 200 caractères.

2) Définition des instance d'apprentissage.

➢On crée une fonction qui va calculer le pourcentage d'occurence de chaque caractère.

➢ Puis créer 5 dataframes contenant les fréquences de chaque caractère de notre d sur les 200 fragments pour chacune des 5 langues avec les classes des 5 langues

➢ Et enfin les concaténer dans un même dataframe.

Dataframe contenant la fréquence de chaque caractère de d des 200 fragments en français :

	a	e	n	u	p	i	s	w	o	l	h	y	b	r	j	t	Classe
0	0.065	0.105	0.065	0.055	0.025	0.105	0.035	0.005	0.075	0.060	0.025	0.010	0.000	0.085	0.005	0.075	FR
1	0.095	0.130	0.100	0.030	0.030	0.100	0.070	0.000	0.050	0.050	0.010	0.000	0.005	0.070	0.005	0.105	FR
2	0.050	0.145	0.070	0.035	0.020	0.080	0.065	0.005	0.090	0.080	0.010	0.010	0.015	0.065	0.000	0.080	FR
3	0.105	0.100	0.090	0.050	0.030	0.085	0.055	0.000	0.060	0.075	0.020	0.010	0.015	0.070	0.010	0.080	FR
4	0.065	0.150	0.040	0.030	0.005	0.085	0.085	0.000	0.040	0.060	0.015	0.000	0.010	0.090	0.005	0.095	FR
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	0.055	0.185	0.070	0.040	0.040	0.065	0.110	0.000	0.060	0.070	0.005	0.005	0.010	0.055	0.005	0.080	FR
196	0.080	0.150	0.100	0.055	0.040	0.075	0.055	0.000	0.065	0.070	0.015	0.005	0.010	0.055	0.005	0.085	FR
197	0.080	0.175	0.080	0.030	0.025	0.070	0.065	0.000	0.060	0.050	0.005	0.005	0.005	0.085	0.005	0.095	FR
198	0.040	0.165	0.115	0.100	0.015	0.100	0.080	0.000	0.055	0.040	0.030	0.000	0.005	0.040	0.025	0.060	FR
199	0.100	0.145	0.085	0.070	0.040	0.080	0.055	0.000	0.065	0.065	0.010	0.005	0.000	0.045	0.000	0.065	FR

200 rows × 17 columns

Dataframe contenant les 5 dataframe (concaténation) et leur classe:

	a	e	n	u	p	i	s	w	o	l	h	y	b	r	j	t	Classe
1	0.135	0.090	0.075	0.025	0.030	0.120	0.030	0.000	0.085	0.075	0.015	0.000	0.000	0.060	0.000	0.075	IT
137	0.110	0.100	0.060	0.025	0.055	0.130	0.025	0.000	0.080	0.055	0.010	0.000	0.010	0.070	0.000	0.075	IT
127	0.080	0.100	0.045	0.015	0.025	0.165	0.070	0.000	0.080	0.095	0.005	0.000	0.005	0.055	0.000	0.085	IT
128	0.095	0.105	0.065	0.025	0.035	0.110	0.070	0.000	0.085	0.090	0.010	0.005	0.015	0.040	0.005	0.090	IT
129	0.100	0.095	0.040	0.035	0.040	0.125	0.030	0.010	0.090	0.075	0.010	0.005	0.005	0.070	0.000	0.065	IT
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
68	0.085	0.135	0.090	0.040	0.005	0.105	0.050	0.025	0.020	0.015	0.025	0.005	0.040	0.080	0.000	0.060	DE
69	0.050	0.160	0.120	0.060	0.015	0.095	0.040	0.010	0.015	0.020	0.040	0.000	0.010	0.060	0.005	0.070	DE
70	0.050	0.180	0.105	0.025	0.015	0.095	0.050	0.030	0.030	0.045	0.015	0.000	0.010	0.070	0.000	0.090	DE
71	0.055	0.170	0.090	0.030	0.015	0.100	0.060	0.005	0.015	0.040	0.050	0.005	0.020	0.060	0.005	0.095	DE
100	0.075	0.185	0.095	0.030	0.010	0.070	0.060	0.010	0.040	0.020	0.035	0.005	0.020	0.095	0.000	0.080	DE

1000 rows × 17 columns

Erreur empirique et en généralisation:
Mesure la précision du modèle

- Le module d'apprentissage scikit-learn propose une implémentation de l'algorithme des k plus proches voisins.
- Commençons donc par départager notre échantillon en ensemble d'apprentissage et de test
- X_train, X_test, y_train, y_test = train_test_split(data[d],data[« Classe"],test_size=0.4,random_state=11)
- Les paramètres de la fonction train_test_split sont :
  - les données : variables prédictives ici data[d] (les fragments de texte)
  - les données : variable à prédire ici data["Classe"] (les langues) ou encore la cible
  - test_size : proportion de l'échantillon consacré au test
  - random_state : graine du générateur aléatoire utilisé pour le découpage

l’algorithme K-NN

- Nous allons utiliser l’algorithme k-plus proche voisin pour entraîner le modèle et prédire la langue
- Cet algorithme est utilisé pour résoudre les problèmes du modèle de classification. K-plus proche voisin ou algorithme K-NN crée essentiellement une frontière imaginaire pour classer les données. Lorsque de nouveaux points de données arrivent, l’algorithme essaie de le prédire au plus proche de la ligne de démarcation.
- Il est très important d’avoir la bonne valeur k lors de l’analyse de l’ensemble de données pour éviter le sur-apprentissage et le sous-apprentissage de l’ensemble de données.

- Nous avons vu comment nous pouvons utiliser l’algorithme K-NN pour résoudre le problème d’apprentissage supervisé. Mais comment mesurer la précision du modèle?

Mesure de la précision

- Cette mesure va nous permettre de savoir de combien on se trompe en faisant notre prédiction
- Nous avons maintenant définir la fonction Erreur qui nous permettra de calculer l'erreur empirique et en généralisation pour un k, l_min et d donnés en utilisant l'algo K-nn. Plus ces erreurs sont faibles plus on aura une meilleur précision pour la prédiction d'ou l'importance de cette fonction
- La fonction Entrainement que nous avons nommé par la suite f a le même fonctionnement que la fonction précédente sa seule particularité, il propose plusieurs k. Il nous permet d'avoir une liste des k pour en fin choisir celui qui minimise les erreurs (empirique et en généralisation) pour un l_min et d donné.

Ce que retourne notre fonction f

Pour l_min=200

d=['a', 'e', 'n', 'u', 'p', 'i', 's', 'w','o', 'l', 'h','y','b','r','j','t']

Et en considérant toutes les 5 langues , on a ce tableau

Ce sont nos valeurs par défaut

]:

	k	empirique	generalization
0	1	0.000000	0.1650
1	2	0.063333	0.1775
2	3	0.063333	0.1375
3	4	0.066667	0.1275
4	5	0.076667	0.1200
5	6	0.071667	0.1225
6	7	0.075000	0.1100
7	8	0.075000	0.1075
8	9	0.075000	0.1000
9	10	0.071667	0.1000
10	11	0.080000	0.0925
11	12	0.085000	0.0975
12	13	0.081667	0.0975
13	14	0.085000	0.0975
14	15	0.086667	0.1000
15	16	0.088333	0.0900
16	17	0.078333	0.0900
17	18	0.083333	0.0925
18	19	0.080000	0.1000

Ici, dans cet exemple , nous avons crée un graphique pour voir la valeur k pour laquelle nous avons une grande précision.

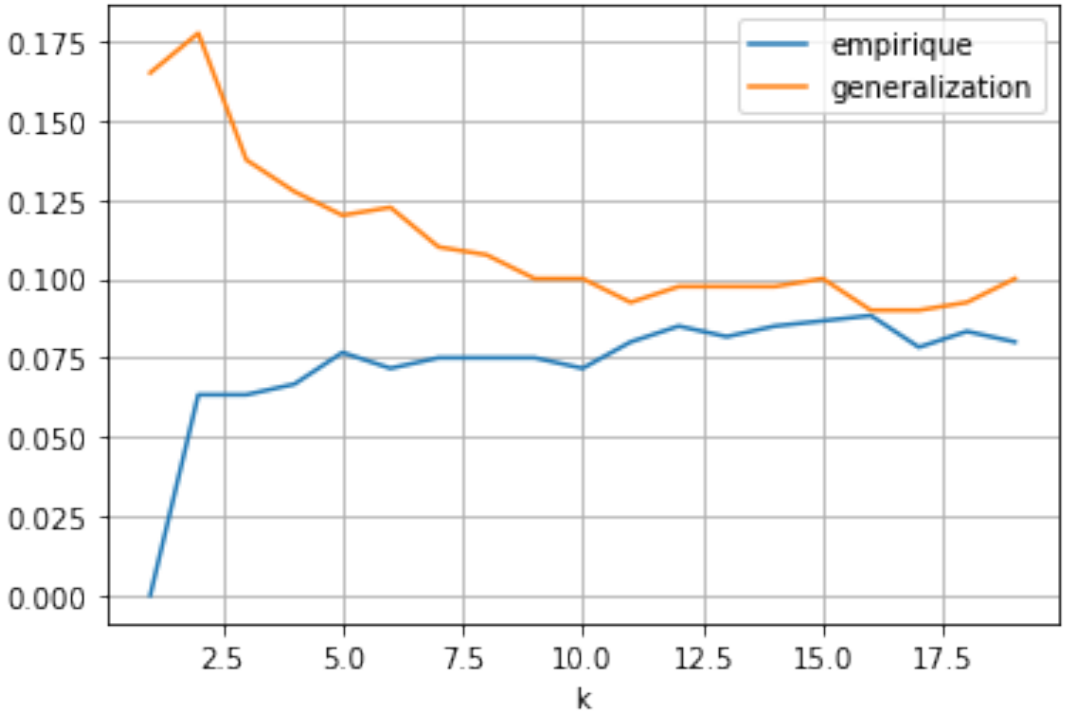
- Nous avons tracer à partir de la fonction f pour les même valeurs de base (d , l_min=200,5 langues)

- On choisit donc le k qui minimise l'erreur.

- Un classifier k-plus proches voisins avec k=16 semble donner le meilleur résultat en terme de généralisation.

```
erreur(d,l_min,16)
```

```
Erreur empirique: 0.08833333333333337
Erreur en généralisation: 0.08999999999999997
```



Approfondissement:

Nous allons effectuer un réglage des hyperparamètres (l_min ,d ,les langues) pour choisir la valeur qui donne les meilleures performances.

1 er cas étudié: prenons les mêmes instances mais avec des l_min et d différents :

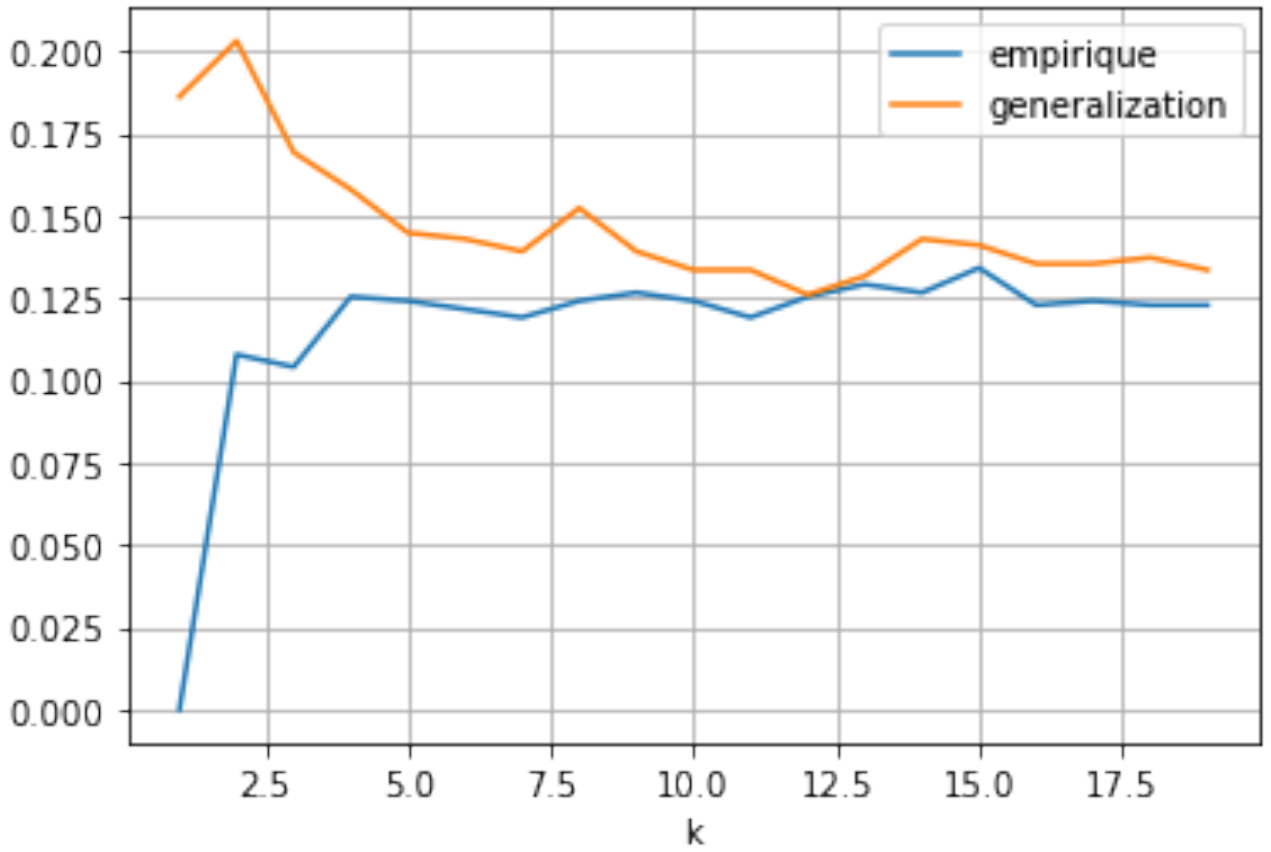
Nous gardons les mêmes instances, le même d, mais nous changeons notre l_min. Prenons l_min=150

- On choisit donc le k qui minimise l'erreur.
- Un classifier k-plus proches voisins avec k=12 semble donner le meilleur résultat en terme de généralisation.

```
erreur(d,150,12)
```

```
Erreur empirique: 0.12531328320802004
Erreur en généralisation: 0.12593984962406013
```

- le minimum des k pour l_min=150 est plus elevé que quand l_min=200
- l_min=200 est donc meilleurs car il a le bon résultat



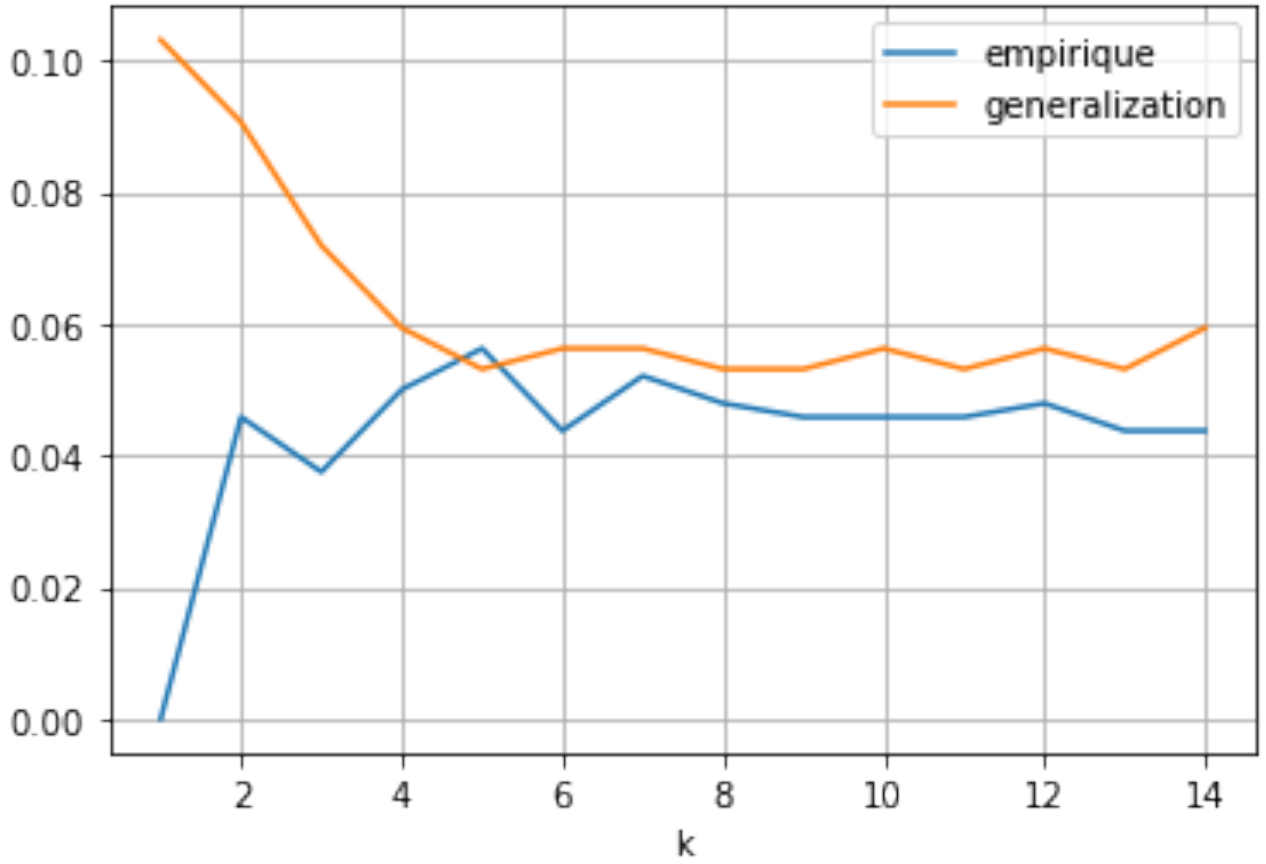
Maintenant Gardons les mêmes instances, le même d, mais prenons l_min=250

- Un classifier k-plus proches voisins avec k=5 semble donner le meilleur résultat en terme de généralisation.

```
erreur(d,250,5)
```

```
Erreur empirique: 0.03749999999999998
Erreur en généralisation: 0.07187500000000002
```

- Pour l_min=250 on se trompe moins par raport l_min=200
- l_min=250 donc est meilleurs car il donne le meilleur résultat
- On observe donc que selon le l_min l'erreur est plus ou moins importante.



Maintenant changeons simplement d et gardons notre l_min de base (l_min=200).

- Prenons d=[‘p','i','t','s']
- Un classifier k-plus proches voisins avec k=18 semble donner le meilleur résultat en terme de généralisation.

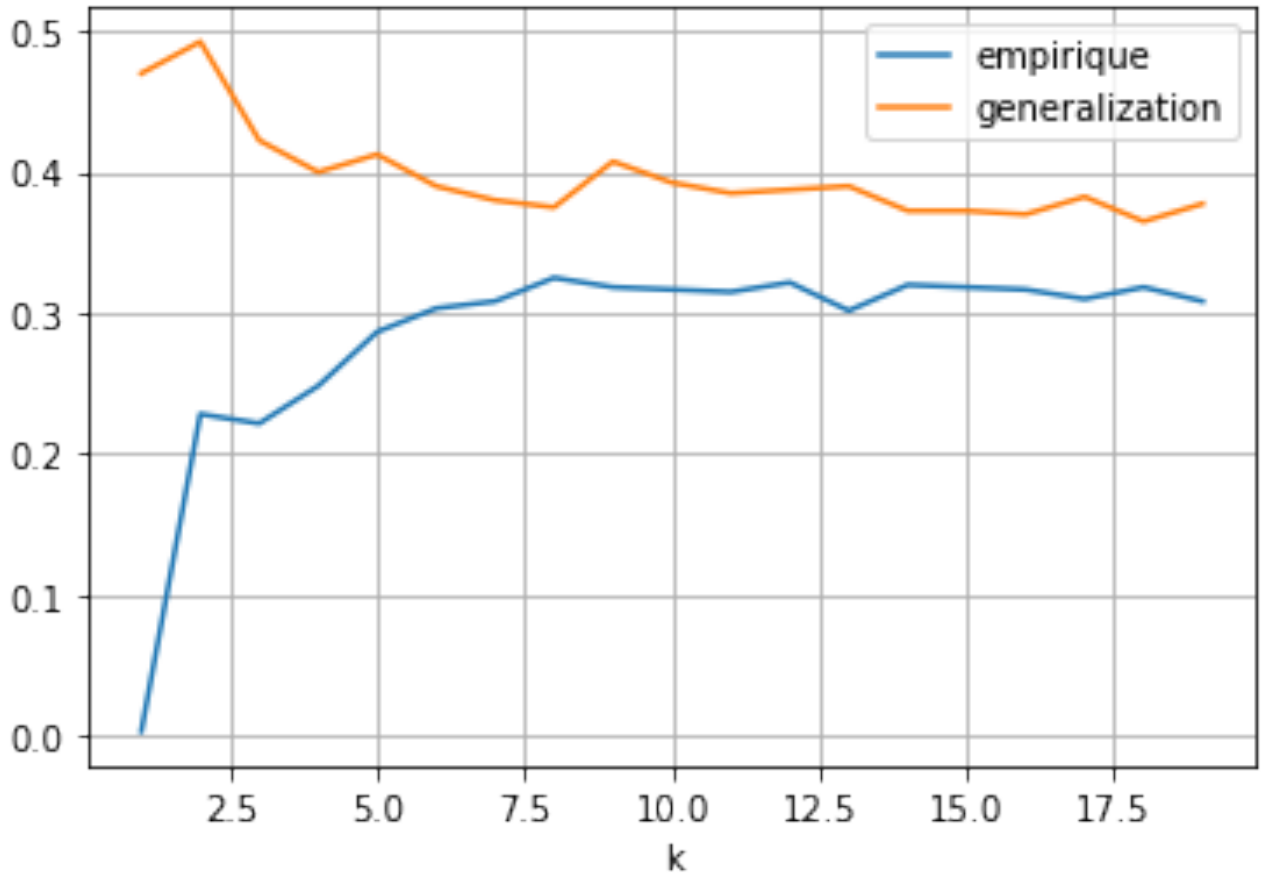
```
erreur(d2,l_min,18)
```

```
Erreur empirique: 0.31833333333333336
Erreur en généralisation: 0.365
```

- On observe une erreur plus importante pour ce d choisi.

Conclusion 1 :

- Avec les 5 instances on observe que l'erreur devient plus ou moins importante lorsque d ou l_min change.



2 ème cas:

On mesure la précision pour 2 langues proches et 2 langues qui ne le sont pas :

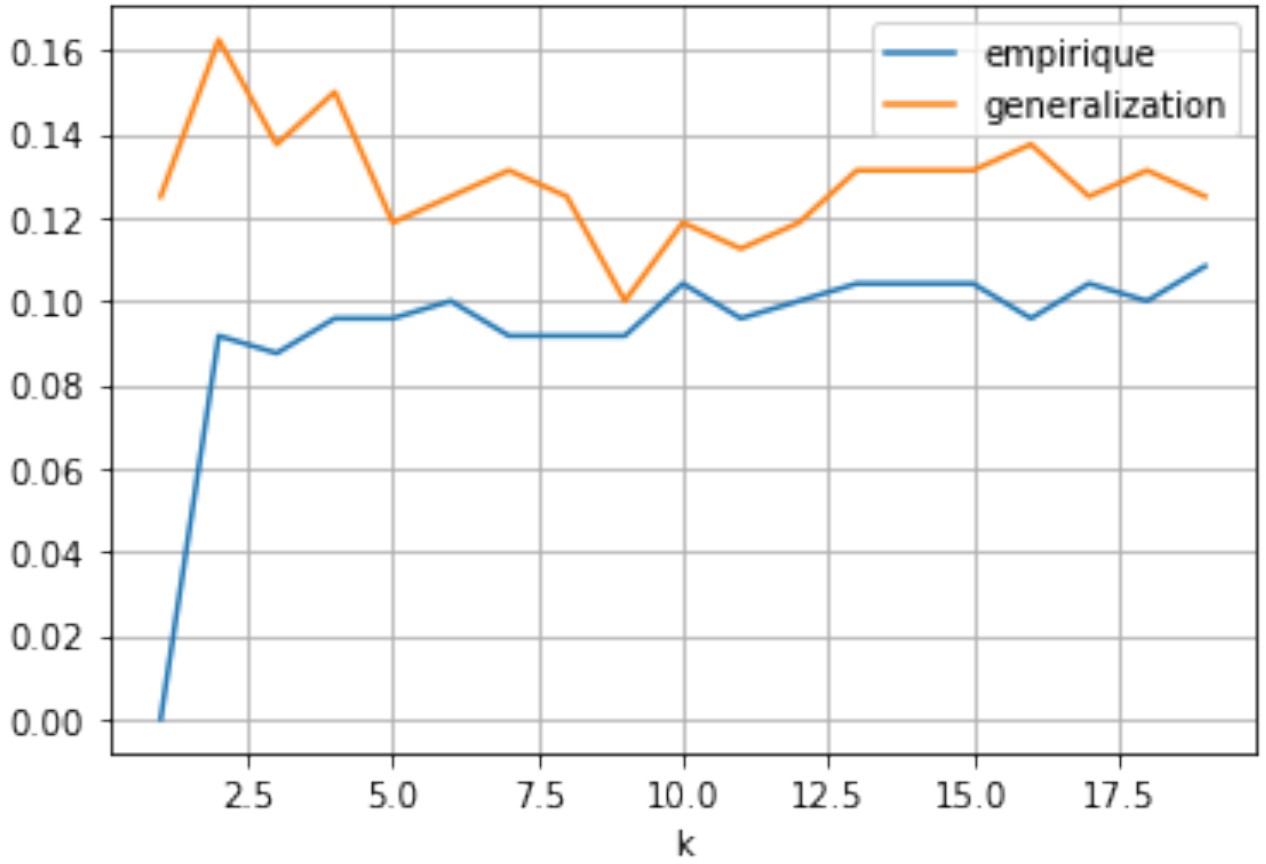
Maintenant changeons les instances (on garde uniquement espagnol et italien, langues qui se rapprochent) en gardant le meme l_min et le meme d :

- Un classifier k-plus proches voisins avec k=9 semble donner le meilleur résultat en terme de généralisation.

```
erreur2(d,l_min,9)
```

```
Erreur empirique: 0.09166666666666667
Erreur en généralisation: 0.09999999999999998
```

-



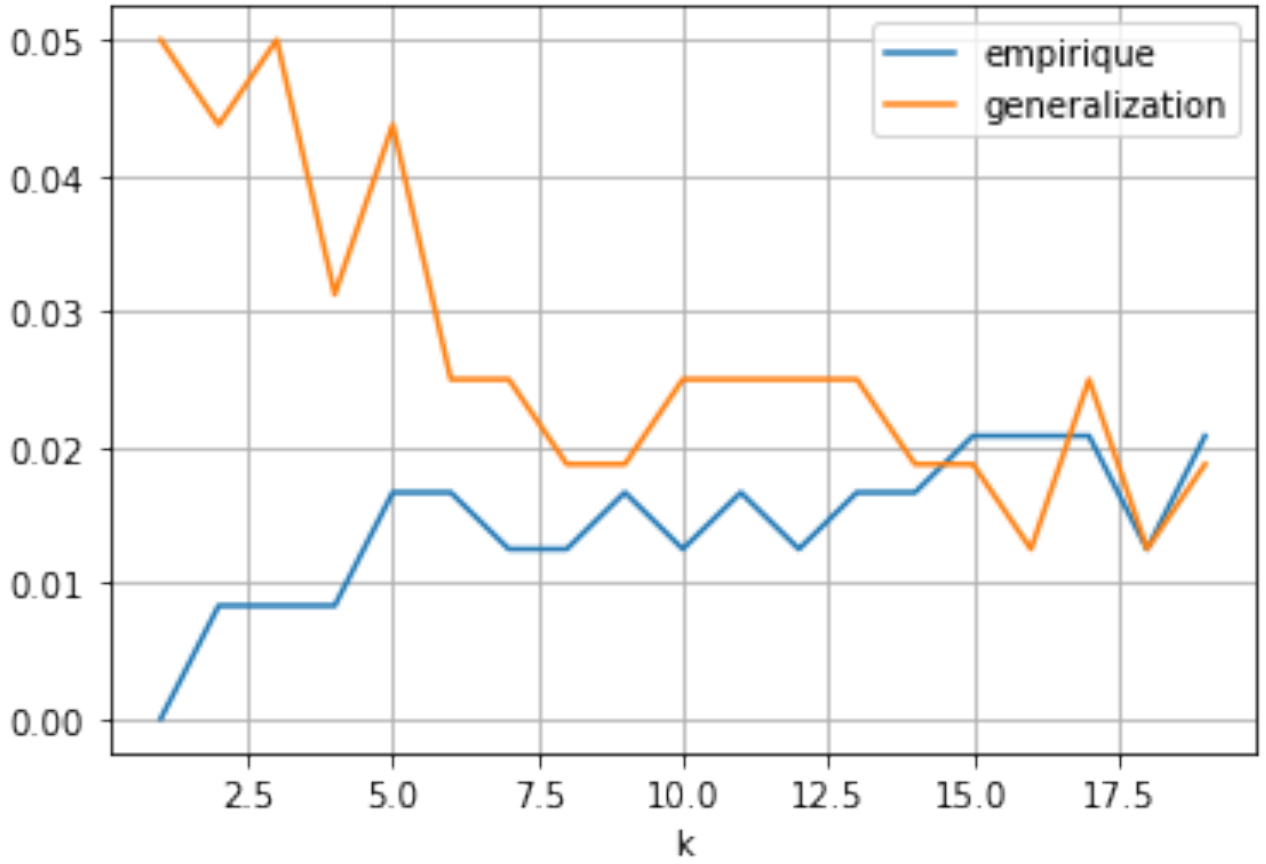
Maintenant prenons uniquement francais et allemand (2 langues pas proches) et comparons avec notre exemple précédent.

- Un classifier k-plus proches voisins avec k=16 semble donner le meilleur résultat en terme de généralisation.

```
erreur3(d,l_min,9)
```

```
Erreur empirique: 0.01666666666666672
Erreur en généralisation: 0.018750000000000044
```

- Conclusion :
  - Lorsque les langues se rapprochent il y a plus de risque d'erreur, en effet cela s'explique par le fait qu'elles auraient presque la même fréquence de caractère donc l'algorithme aura à du mal a les distinguer a cause de leurs similitudes.



Conclusion

- Il faut donc choisir un l_min et un d qui minimise les risques; en effet certains cas comme on a pu étudié, comme par exemple en modifiant le domaine des instances on remarque des risques d'erreurs plus ou moins importants.
- les langues les plus proches semble donnée des risques d'erreurs k plus élevés que celles qui ne le sont pas, la prédiction pour ces langues est donc moins précises.
- Parmi tous les cas qu'on a pu étudier, le cas où l_min=250 avec d=['a', 'e', 'n', 'u', 'p', 'i', 's', 'w','o', 'l', 'h','y','b','r','j','t'] et n=160 semble donne un meilleurs résultat (le risque d'erreur est beaucoup plus faible)