Universitatea din Bucuresti
Facultatea de Matematica si Informatica
Specializare: *Calculatoare si Tehnologia Informatiei*
Grupa: *354*
Nume: *Dima Ana-Maria*

Documentatie Proiect I.A.

I. Etapele abordate in rezolvarea:
   1) Citirea datelor din fisiere .csv
   2) Prelucrarea textului
   3) Transformarea textului in „sac de cuvinte” (BoW)+ Prelucrari
   4) Clasificatorul folosit
   5) Ten fold cross validation
   6) Matricea de confuzie asociata

II. Observatii proprii (Concluzii+Dificultati intampinate)

Submisia 1-KNN

I.1) Citirea datelor din fisiere .csv:

Prima etapa abordata in rezolvarea acestui proiect a fost reprezentata de citirea informatiilor din fisierele tip csv. Cu ajutorul functiei `read_csv` din modulul `Pandas` importat in proiect, am citit datele din fisierele de antrenare si testare sub forma de DataFrame. In niste variabile separate (corpus si label) am memorat coloanale cu id text si label din fisierul de antrenare.

I.2) Prelucrarea textului:

Corpusul citit anterior l-am trimis ca parametru unei functii (`get_vocabulary_from_corpus`) care returneaza toate cuvintele din acel corpus impreuna cu numarul corespondent de aparitii (prin intermediul functiei `Counter()`, din modulul `Colections`). In interior acestei functii se face apelul catre o alta functie care se ocupa de prelucrarea si tokenizarea textului.

In interiorul functiei `tokenize(text)` se transforma toate literele in litere mici `(lower()`), iar cu ajutorul functiei `sub` din modulul `re` se elimina toate caracterele nonalfabetice. Functia in cauza returneaza textul prelucrat si tokenizat cu tokenizatorului `word_tokenize` din biblioteca `nltk`.

Se extrag cele mai „populare” n=100 cuvinte din dictionarul obtinut anterior si se returneaza 2 dictionare index catre cuvant si cuvant catre index, pentru a nu pierde legatura dintre acestea in timpul prelucrarilor ulterioare.

I.3) Transformarea textului in „sac de cuvinte” (BoW)+ Prelucrari

Cu ajutorul functiei `corpus_to_bow` se face conversia corpusului in Bag of Words. Fiecarei inregistrari de tip text ii corespunde un vector de aparitii raportat la vectorul de aparitii al celor mai frecvente 100 de cuvinte. Ulterior, datele de antrenare tip text sunt impartite astfel: 90% pentru antrenare, 10% pentru testare, nu inainte de a se face un shuffle al acestora cu ajutorul functiei `.shuffle()` din modulul `random`.

I.4) Clasificatorul folosit

In cadrul acestei submisii am folosit clasificator `KNeighborsClassifier`, implementat in modulul `sklearn.neighbors`, atribuindu-i acestui hiperparametrul `n_neighbors=3` (k=3). Acest clasificator nu are paramentrii.

Cu ajutorul functie `fit()` am realizat antrenarea modelului, aceasta durand 0:00:00.334368. Otimizarea acestui process am relaizat-o cu ajutorul `n_jobs=4`, pentru a desemna numarul de procese pe care CPU-ul le desfasoara in acelasi timp.

I.5) Ten Fold Cross Validation

Vectorul scors (de acurateti) obtinut in urma procesului ten fold cross validation este de forma: [0.82888889 0.85111111 0.82222222 0.81333333 0.83555556 0.82666667 0.83555556 0.84222222 0.82222222 0.85111111]

Valoarea medie a acuratetii este de 0.8328888888888889.

I.6) Matricea de confuzie asociata

Matricea de confuzie asociata acestui model este :

[[2121 542]

[285 2052]]

Normalizarea acesteia:

[[0.79626096 0.20373904]

[0.12160784 0.87839216]]

II. Observatii proprii (Concluzii)

Modelul in cauza a dat cele mai bune rezultate utilizand un K=3. Am incercat utilizarea mai multor valori ale parametrilor K (inclusiv 31= aprox. sqrt (1000)- numarul datelor de test sau 71=aprox sqrt (5000)-numarul datelor de antrenare), insa fara imbunatatiri sesizabile are performantei.

O contradictie pe care am intalnit-o in cadrul acestei abordari a fost diferenta considerabila intre media obtinuta prin procedeul ten fold cross validation si scorul obtinut pe Keggle, diferenta fiind de aprox 0.11-0.14. (valoarea mai mica a fost obtinuta pe Keggle).

Scorul obtinut pe Keggle este de 0.72509.

Submisia 2-SVMs

Subcapitolele :

1) Citirea datelor din fisiere .csv
2) Prelucrarea textului
3) Transformarea textului in „sac de cuvinte” (BoW)+ Prelucrari

din capitolul: I.Etapele abordate in rezolvarea, au ramas, in mare parte, neschimbate, asa ca nu le voi mai discuta in detaliu.

Singura diferenta fata de submisia precedenta o constituie utilizarea unui alt tokenizator: `TweetTokenizer`, care primeste ca atribut parametrii reduce_len si strip_handles.

I.4) Clasificatorul folosit

In cadrul acestei submisii am folosit un clasificator `SVMs – Support Vector Machines`, implementat in modulul `svm` din biblioteca `sklearn` atribuindu-i acestui un hiperparametru `C` si `o funtie kernel (nucleu)`. Am ales sa folosesc acest clasificator deoarece este un clasificator liniar, ce se ocupa cu clasificarea problemelor binare (aici: clasa 0 -nonmisogin sau clasa 1- misogin).

Valoarea pe care am ales-o pentru hiperparametrul `C` este `C=7`. Aceste hiperparametru este folosit pentru a controla tradeoff-ull dintre maximizarea marginii si acuratetea modelului. Am incercat diverse valori pentru acest hiperparametru, insa rezultatele cele mai bune le-am obtinut pentru valoarea 7.

Intrucat, in cazul problemei noastre nu este vorba de date liniar separabile, am utilizat un kernel trick `(kernel='linear'),` acesta constand in folosirea unei functii nucleu care mapeaza datele intr-un spatiu de dimensionalitate mult mai mare, pentru a putea aduce problema intr-o varianta simplificata, cu date liniar separabile.

Cu ajutorul functiei `fit()` se realizeaza antrenarea modelului. Aceasta antrenare a durat 0:00:04.569852

I.5) Ten Fold Cross Validation

Pentru a efectua Ten Fold Cross Validation am folosit functia `cross_val_score` importata din modulul `sklearn.model_selection.` Valoarea medie a acuratetii este de 0.845.

II. Observatii proprii (Concluzii)

In cazul acestui model, la fel ca in cel precedent, diferenta intre valoarea medie obtinuta local si cea de pe Keggle este una considerabila (aprox 0.11-0.12).

Scorul obtinut pe Keggle pentru acest model este 0.72941.