**- INTELIGENTA ARTIFICIALA -**

**" Documentatie proiect "**

**Tema** : Construirea unui model de clasificare pentru a discrima intre dialetul romanesc si cel moldovenesc. ( Clasificare binara )

In realizarea temei, am folosit modelul BagOfWords, prezentat in laboratorul 5. Acest model reprezinta un algorithm de extragere a datelor din text pentru a le utiliza in modelare.

Pasul 1: Citirea datelor.

trainSamples = np.genfromtxt('train\_samples.npy', dtype=None, delimiter='\t', encoding='utf-8', names=('id','samples'), comments=None)

Citirea datelor se realizeaza prin functia np.genfromtxt pentru fiecare din datele de intrare necesare. Se citesc intr-un ndarray pe 2 coloane ( names=('id','samples') ) : id, sample. Delimitatorul intre cele 2 este "tab".

Pasul 2: Extragerea datelor din text.

Modelul BagOfWords :

Acest model se bazeaza pe numarul de aparitii al cuvintelor in text si implica 2 lucruri :

1. Vocabularul -> vocabularul este reprezentat de o lista de cuvinte. Crearea acestuia se realizeaza prin functia buildVocabulary care primeste ca parametru textul ( trainSamples['samples'] ) unde pe fiecare linie avem un string. Functia parcurge fiecare linie, extrage din fiecare linie(string) cuvintele si le adauga in lista daca nu se mai gasesc. Fiecare cuvant primeste un index unic.

extragere cuvinte in vocabular :

bagOfWords = BagOfWords()  
 bagOfWords.buildVocabulary(trainSamples['samples'])

2. Matricea a aparitiilor pentru fiecare cuvant. Fiecare linie a matricei reprezinta cate o linie din fisierele (trainSamples/testSamples), iar coloanele sunt reprezentate de cuvintele din vocabular. Valorea de la pozitia [x][y] este >0 (daca cuvantul y se gaseste in stringul de la linia x), sau 0 daca nu se gaseste.

creare matrici necesare pentru pregatirea modelului :

trainFeatures = bagOfWords.getFeatures(trainSamples['samples'])  
 testFeatures = bagOfWords.getFeatures(testSamples['samples'])

Dupa extragerea datelor in trainFeatures/testFeatures modelul este gata pentru clasificare.

**Pasul 3: Normalizarea datelor.**

Prin normalizare rescalam elementele din matrice in intervalul [0,1] . Am folosit normalizarea minMax pentru a scala elementele extrase in matricea aparitiilor. La alogritmul masini cu vectori suport datele normalizate sunt convertite mai rapid decat cele care nu sunt normalizate.

Pasul 4: Definirirea algoritmului de clasificare. ( SVM -> Masini cu vectori suport )

classifier = svm.SVC(C = 2\*\*5, kernel='rbf', gamma = 0.001)

SVM - ul este un algoritm de clasificare ajutandu-ne sa separam datele in mai multe categorii .( in cazul nostru in 2 categorii -> 0 ( Moldovenesc ) si 1 ( Romanesc ). Technica de separare a datelor este optimizata prin parametrii masinii. Eu am utilizat parametrii (C, kernel, gamma).

C -> cat de mult putem tolera clasificarea. Daca este mare, clasificatorul aseaza corect mai multe puncte. Pot aparea 2 probleme : overfitting cand este prea mare, underfitting cand este prea mic.

Kernel = 'rbf' -> rbf-ul de obicei da o acuratete mai buna decat linear.

Gamma = '0.001' -> paramaetrul gamma mic este utilizat pentru rbf ( curbura axei ) .

!!! Am incercat sa scalez datele fara while, dar imi da eroare de memorie.

Dupa definirea clasificatorului incepem sa facem predictiile

-> scalam datele

scaledTrainSamples, scaledTestSamples=minMaxScaler(trainFeatures[start:finish], testFeatures[start:finish])

-> facem fit pe date pentru a le utiliza mai tarziu ( necesar )

classifier.fit(scaledTrainSamples, trainLabels['labels'][start:finish])

-> calculam predictiile

predictions[start:finish] = classifier.predict(scaledTestSamples)

Calculam din 300 in 300 de linii si punem predictiile intr-o lista de predictii incepand cu liniile 0 - 300, 300 - 600 etc pana la 2600. De la 2600 punem urm predictii tot la fel doar pe alte pozitii 2600 - len(testSamples).

!!! Am facut aceasta metoda cu while, intrucat imi da eroare de memorie daca incerc sa scalez datele in dimensiunile (2623, 32874).

Citirea val\_labels, calculare acuratete si scorul f1 prin functiile din sklearn.metrics.

valLabels = np.genfromtxt('validation\_labels.npy', encoding='utf-8', dtype=None , delimiter='\t', names=('id','labels'))  
print('accuracy', accuracy\_score(valLabels['labels'][:2623], predictions))  
print('f1 score', f1\_score(valLabels['labels'][:2623], predictions))print('confusion matrix')  
print(confusion\_matrix(valLabels['labels'][:2623],predictions))

f1 scor : 0.5203310543360922

confusion matrix :

[[567 720]

[613 723]]