Documentație proiect

“Misogynistic Italian Tweets Detector”

Scopul proiectului este acela de a realiza un algoritm de machine learning pentru detectarea tweeturilor misogine in italiana. Etapele realizării acestui proiect sunt descrise în paşii următori.

1.Setul de date:

- Setul de date este format din 5000 de tweet-uri în italiană care reprezintă partea de antrenare a modelului şi 1000 de tweet-uri pentru partea de testare a modelului antrenat.
- Datele de antrenare sunt transmise în format CSV.
- Aspecte ale datelor ce trebuie menţionate sunt următoarele:
    - Tweeturile conţin “**twitter handle**” sub forma “@ + text”, **hyperlinkuri** de forma “https:// + text” şi **emoji-uri în forma hexazecimală**.
    - Datele de antrenare sunt aşezate într-un formate necorespunzător întrucât acestea au numai texte cu label-ul “1” la început, urmate de un număr foarte mare de texte cu label-ul “0” după care urmează o alternanţă.

2.Organizarea proiectului:

- Proiectul este împărţit în patru scripturi cu roluri distincte:
    - “**text_preprocessing.py**” cu scopul de a aplica toate modificările de preprocesare.
    - “**data_partition.py**” cu scopul de a procesa datele din corpus train
    - “**prediction_to_csv.py**” cu scopul de a transforma predicţiile modelelor într-un format potrivit pentru încărcare pe Kaggle.
    - “**main.py**” este scriptul principal care se foloseşte de restul scripturilor pentru a face predicţii.

3.Preprocesarea datelor:

- Pentru preprocesarea datelor ne folosim de librăriile nltk.corpus, nltk.stem.snowball, nltk.tokenize, re, preprocessor şi string.
- Pentru a putea avea un text cât mai relevant trebuie ca acesta să fie curăţat de numele de utilizatori, de linkuri, de cuvintele de legătură şi de punctuaţie.
- Ne creem variabilele **tokenizer (TweetTokenizer)**, s**temmer (SnowballStemmer("italian"))**, **stop_words** pentru a prelucra textul.
- Folosind **funcţia "clean"** din libraria **preprocessor**, pe fiecare propozitie din corpus o să eliminăm **emojiurile** în forma hexazecimală. În aceeaşi manieră folosim "***re.sub(r"http\S+", "", sentence)***" pentru a elimina link-urile de forma "**http**" urmate de un string (cu ajutorul wild cardului **\S+**).
- Cu ajutorul TweetTokenizer-ului o să transformăm tweet-urile în tokenuri şi scăpăm si de numele de utilizatori specificând "***strip_handles=True***".
- Parcurgând lista de tokenuri putem elimina cuvintele comune, şi semnele de punctuaţie folosind o listă nouă de tokenuri la care adăugăm cuvintele care nu se află în lista de semne de punctuaţie cât nici în lista de stop words.
- Înainte de a le adăuga la lista de tokenuri procesate o să le trecem şi prin stemmer pentru a le reduce la forma cea mai simplă.
- La finalul acestei proceduri propoziţiile sunt în cea mai simplă formă astfel obţinând date cât mai relevante.

4.Procesarea datelor:

- Procesarea datelor se face cu ajutorul scriptului "**data_partition.py".**
- Am definit două funcţii "to_TFIDF_bow()". Această funcţie este folosită pentru a transforma cuvintele în bag of words folosind algoritmul **TFIDF** format din **Term Frequency + Inverse Data Frequency**".
- **Term Frequency** reprezintă de câte ori a apărut un cuvânt în cadrul unei propoziţii împărţit la numărul total de cuvinte dintr-o propoziţie. Fiecare document(propoziţie) are un **TF** propriu.
- **Inverse Data Frequency** reprezintă logaritm din numărul total de documente împărţit la numărul de documente care conţin un anumit cuvânt. Logaritmul se foloseşte pentru a ameliora importanţa cuvintelor cu o frecvenţă foarte mare.
- **TFIDF** reprezintă produsul dintre **TF** şi **IDF.**
- Am ales ca parametrul "**max_features**" să fie egal cu 1000 prin observaţii din aplicarea mai multor valori

5.Modelele folosite:

a. **Multinomial Naive Bayes**:

i. Am importat din librăria **sklearn.naive_bayes** modelul MultinomialNB pentru a antrena datele. Am observat că atunci când există un număr mai mic de "**max_features**"(1000) în vectorizatorul **TFIDF**, modelul MultinomialNB() funcţionează mai bine.

max_features	acuratetea medie
2500	0,8122
2000	0,8110
1500	0,8290
1000	0,8372

ii. Pentru acest model hiperparametrii au fost cei default.

iii. În toate cazurile datele antrenate şi cele de test au fost prelucrate cu metodele prezentate în secţiunea "**Preprocesarea datelor**" ⇔ scoatem stop words, punctuaţie, numele de utilizatori, linkurile web, emojiurile şi le trecem printr-un stemmer.

iv. Am antrenat şi testat datele în manieră **10 fold cross validation,** datele au fost împărţite în 10 intervale egale. Folosind această metodă împărţim datele în 10 intervale şi acestea sunt pe rând atât date de antrenare cât şi date de validare. La final toate rezultatele sunt adunate pentru a obţine matricea de confuzie

```
Matricea de confuzie este:
 [[2347  472]
 [ 316 1865]]
```

v. Folosind clasificatorul de mai sus cu parametrii de la 10 fold cross validation obţinem următoarele scoruri şi acurateţe medie.

```
Iteratia numarul  1
Acuratetea modelului este:  0.836

Iteratia numarul  2
Acuratetea modelului este:  0.85

Iteratia numarul  3
Acuratetea modelului este:  0.836

Iteratia numarul  4
Acuratetea modelului este:  0.848

Iteratia numarul  5
Acuratetea modelului este:  0.85
```

```
Iteratia numarul  6
Acuratetea modelului este:  0.858

Iteratia numarul  7
Acuratetea modelului este:  0.83

Iteratia numarul  8
Acuratetea modelului este:  0.852

Iteratia numarul  9
Acuratetea modelului este:  0.808

Iteratia numarul  10
Acuratetea modelului este:  0.856

Media acuratetii pe model:  0.8424
```

vi. Deşi pe datele de test local acurateţea medie este **0.8424** pe Kaggle pe leaderboard-ul public acurateţea este **0.76842** şi pe cel privat este **0.77935.**

vii. Antrenarea datelor a durat **2.5 secunde.** Din libraria time am folosit funcţia time pentru a afla timpul de antrenare.

b. SVM/SVC:

i. Importăm din librăria **sklearn.svc** modelul de antrenare de date **SVM**. La fel ca la modelul de mai devreme un număr mic de "**max_features**"(1500) rezultă într-o acurateţe mai mare de prezicere a modelului pe datele de validare.

max_features	acuratetea medie
2500	0,8596
2000	0,8528
1500	0,8606
1000	0,8538

ii. Datele pe acest model au fost prelucrate precum cele de dinainte.

iii. Hiperparametrii modelului sunt cei default.

iv. Matricea de confuzie pentru datele de validare ale acestui model este:

```
Matricea de confuzie este:
 [[2264  298]
 [ 399 2039]]
```

v. Datele de validare au da o **acurateţe medie** de ~**0.8606** însă pe Kaggle pe **leaderboard-ul privat** acurateţea este **0.74446** şi pe cel **public** este de **0.74042**

vi. Antrenarea datelor a durat **0.3 secunde.**