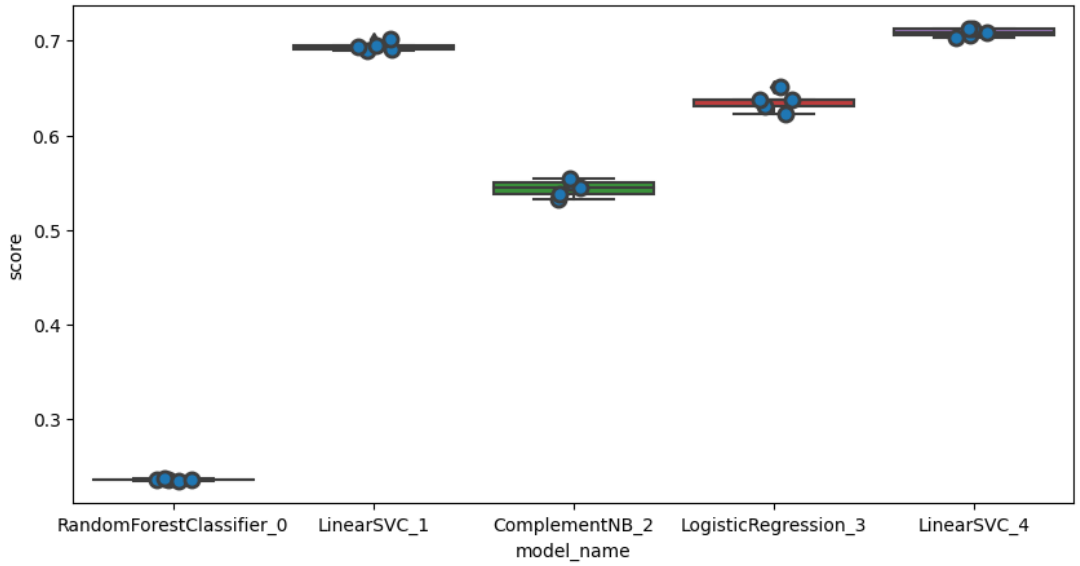
Licu Mihai-George CTI 362

Documentatie Submisii
Translation Source Dialect Identification

Sumar

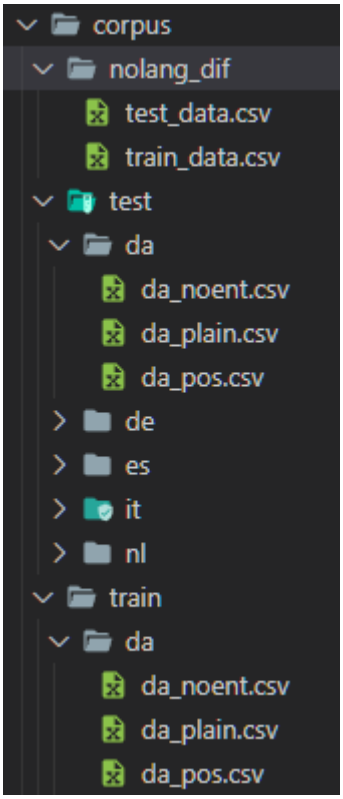
1. Model selection

Pentru inceput in scriptul `test_models.ipynb` am analizat datele si am efectuat un proces de model selection pentru a identifica modele promitatoare problemei.



2. Procesarea datelor

Pentru a nu face preprocesarea datelor in fiecare model la fiecare rulare a acestuia am decis sa fac preprocesarea externa si sa salvez o copie a datelor. Acest lucru se intampla in scriptul `preprocess.ipynb` . Pentru fiecare entry din seturile de date (train_data, test_data) s-au sanitizat datele si s-au pastrat doar randurile nenule.



Aditional am făcut mici preprocesari care erau necesare pentru alte modele decat cele finale din aceste doua submisii. Am separat datele pe baza limbii acestora, datele de train deja au limba textului ca coloana iar pentru datele de testare am folosit libraria din python `langdetect` (un port al librariei Google languagedetection din Java). Folosind libraria `spacy`, am facut copii ale seturilor de date in care am identificat partea de vorbire a cuvintelor si in care am inlocuit entitatile (substantive proprii, valori numerice) cu un token respectiv. In ambele cazuri am reprezentat finalul propozitiilor cu un token specific "SENSEP".

3. Primul model (CNB)

a. Prezentarea modelului

Modelele Naive Bayes folosesc teorema lui Bayes pentru a determina probabilitatea ca un element sa apartina unei anumite clase. Complement Naive Bayes este o adaptare asupra Multinomial Naive Bayes pentru a rezolva problemele in care clasele din setul de date sunt foarte imbalansate. In loc de a calcula probabilitatea ca un element sa apartina unei anumite clase se calculeaza probabilitatea ca elementul sa nu apartina tuturor celorlalte claselor, de aici si numele (complementul probabilitatii).

b. Caracteristici folosite

Caracteristicile au fost extrase din datele de antrenare folosind modelul CountVectorizer cu care am obinut numarul de aparitii a fiecarui token in toate textele, eliminand stopwords, convertind la litera mica.

c. Parametrii si hiperparametrii

Hiperparametrii alesi au fost cei default ai implementarii din `sklearn`, alpha=1, fit_prior=True, class_prior=None, norm=False.

d. Rezultate

5-fold Acc: 0.64 5-fold F1: 0.66 Public Kaggle: 0.66

4. Al doilea model (SVM)

a. Prezentarea modelului

Modelele Support Vector Machine construiesc un hiperplan care incearca sa separe datele de antrenare cat mai bine si se foloseste de acest hiperplan creat pentru a clasifica date noi.

b. Caracteristici folosite

Caracteristicile au fost extrase din datele de antrenare folosind modelul TF-IDF (Term Frequency-Inverse Document Frequency). Pentru fiecare propozitie la fiecare token a acesteia s-a memorat frecventa cu care acesta a aparut in propozitie inmultita cu o pondere logaritmica calculata in functie de cate ori token-ul respectiv apare in toate propozitiile din setul de date. De asemenea s-au eliminat stopwords, cuvintele au fost converite la litera si s-au folosit unigrame si bigrame.

c. Parametrii si hiperparametrii

In urma folosirii metodei GridSearchCV unde au fost alese valori rezonabile pentru fiecare hiperparametru au rezultat valorile cu cel mai bun scor 'C': 1, "class_weight': 'balanced', 'dual': False, 'loss': 'squared_hinge', 'multi_class': 'ovr', 'penalty': 'l1'.

d. Rezultate

5-fold Acc: 0.73 5-fold F1: 0.74 Public Kaggle: 0.70