Membership Inference Attacks on Sequence-to-Sequence Models: Is My Data In Your Machine Translation System?

Farhad Ali Irinel GUL | Eduard PETCU

https://github.com/FarhadGUL06/membership_seq2seq_attack.git

Cuprins

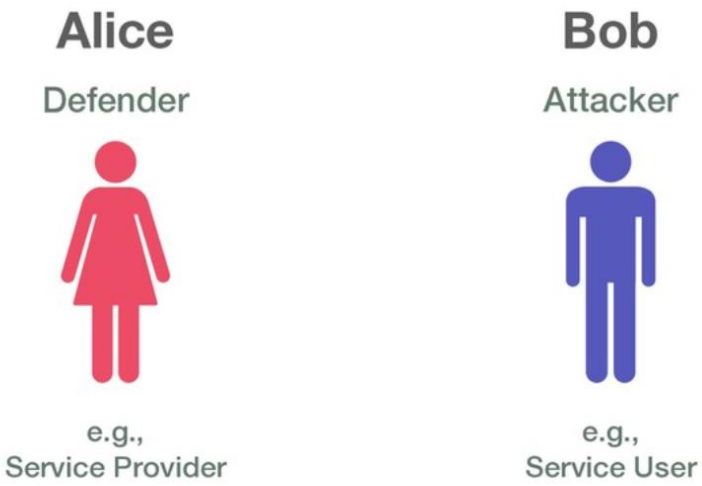
1. Atacul reprodus în cadrul lucrării

Atacul reprodus este unul de tip Membership inference și se axează pe modele de tip Sequence-to-Sequence (în cazul nostru, un model de traducere din limba germană în limba engleză) [1]. Avem un set de date și un model de tip black-box care se antrenează pe acesta, și vrem să aflăm dacă un atacator ce deține o parte din datele pe care s-a antrenat modelul, poate determina care sample face parte din setul de date de antrenare. Pentru a putea realiza această observație, avem nevoie de un third-party neutru (numit „Carol") care să dețină ambele perspective. Întrucât acesta are acces la anumite sample-uri folosite la antrenarea modelului de traducere, poate să încerce să diferențieze pe viitor sample-uri folosite de cele neutilizate la antrenare.

Dacă încercarea se materializează și acesta poate propune un model de clasificare binară care să separe sample-urile atunci putem spune că modelul de traducere nu este sigur, întrucât se pot face astfel de observații.

Fiind 2 părți implicate, „Alice" ca fiind modelul și „Bob" atacatorul, interesul lui Alice este acela ca Bob să nu poată face distincția între sample-uri folosite / nefolosite la antrenare prin inferențe repetate iar interesul lui Bob este acela de a realiza acest lucru.

În realitate, Bob nu este neapărat personajul negativ, întrucât intenția acestuia poate fi aceea de a limita folosirea datelor private de către Alice. Analiza efectuată va fi din perspectiva lui Carol, care deține informații despre Alice și despre Bob.



2. Ipoteza

Nu se poate realiza un model de clasificare binară care să facă perfect distincția între sample-uri folosite și nefolosite la antrenarea unui model Sequence to Sequence (utilizând LSTM în cazul nostru) pe baza observării nivelului de acuratețe în traducere (BLEU Score) [2]. În cele ce urmează, vom analiza această ipoteză, construind mai multe modele de clasificare binară în jurul rezultatelor obținute din cadrul modelului de traducere.

3. Relevanța practică

Această temă poate fi relevantă din perspectiva unui autor a cărui texte sunt / pot fi folosite în antrenarea unui model fără permisiunea acestuia, încălcându-se astfel drepturile de autor.

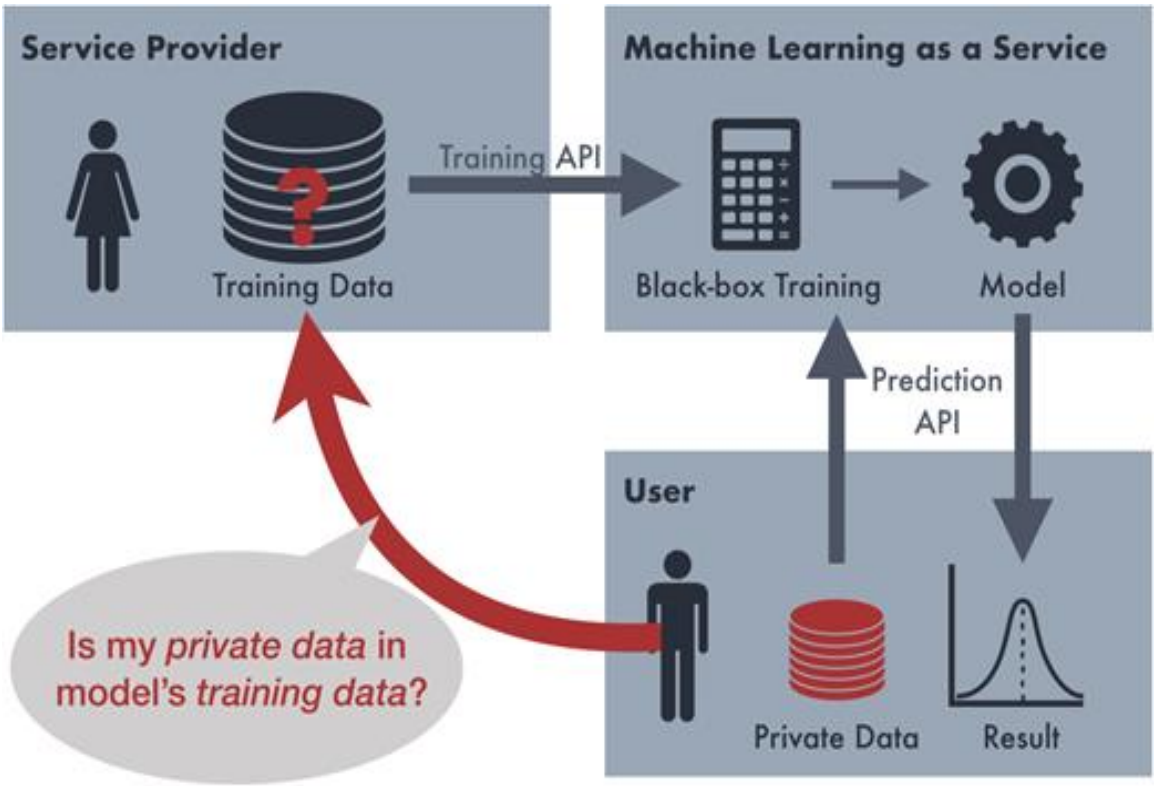
Un alt motiv, enunțat și în paperul dat spre analiză, este utilizarea datelor introduse în serviciul de machine learning cu alt scop decât cel pentru care a fost intenționat: clientul trebuie să aibă încredere în furnizor că acesta nu va folosi în scop negativ datele oferite, însă acest nivel de încredere nu poate fi verificat în niciun fel.

Astfel, o soluție pentru această problemă ar fi să observăm cum se comportă modelul atât pe date care fac parte din setul de antrenare cât și pe date provenite din alte surse.

În realitate, experimentul nu este chiar realizabil întrucât implică deținerea de colecții de date folosite în antrenarea modelului (care de obicei sunt private).

4. Verificarea corectitudinii datelor

Pentru modelul antrenat de Alice, am ales traducerea din limba germană în limba engleză (asemănător alegerii din paper) întrucât pe această pereche se găsesc cele mai multe dataseturi. Tot pe acest tuplu s-au demonstrat cele mai mari BLEU scoruri (Bilingual Evaluation Understudy Score) [3] în comparație cu toate celelalte perechi de limbi străine. Această metrică este folosită pentru clasificatorul binar pentru a determina dacă un sample face parte din setul de antrenare al lui Alice. Analiza datelor pe care am făcut-o imită interpretarea realizată în lucrarea lui Sorami Hisamoto.



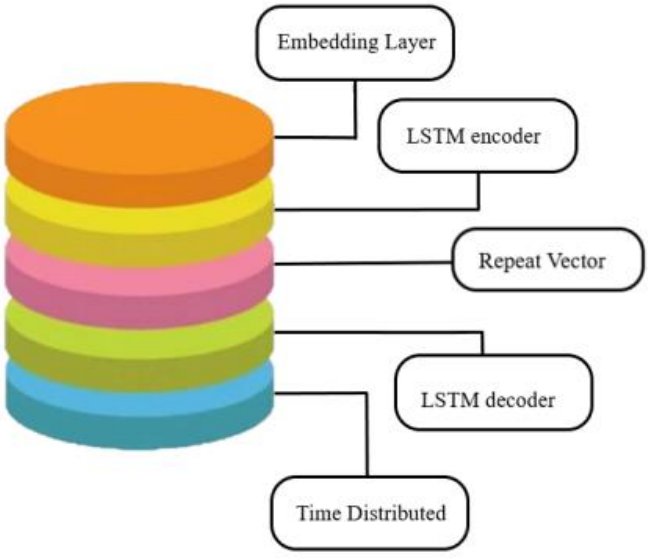
5. Reproducerea atacului

Din păcate, nu am avut acces la modelul folosit pentru realizarea paper-ului [4], de aceea am antrenat noi unul de la 0 cu un dataset public [5].

Am ales să antrenăm în diverse scenarii acest model:

```
English Vocabulary Size: 6707
English Max Length: 8
German Vocabulary Size: 11545
German Max Length: 17
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 17, 256)           2955520

 lstm (LSTM)                 (None, 256)               525312

 repeat_vector (RepeatVecto  (None, 8, 256)            0
 r)

 lstm_1 (LSTM)               (None, 8, 256)            525312

 time_distributed (TimeDist  (None, 8, 6707)           1723699
 ributed)

=================================================================
Total params: 5729843 (21.86 MB)
Trainable params: 5729843 (21.86 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Pe scurt, modelul este compus din următoarele layere:



- Embedding Layer: convertește din tokeni / cuvinte în vectori de dimensiune fixă
- LSTM [6] - encoder: procesează secvența de intrare, capturând informații secvențiale
- Repeat Vector: mărește dimensiunea datelor pentru a putea intra în următorul layer
- LSTM - decoder: generează secvența de ieșire
- Time Distributed: produce probabilități finale pentru fiecare cuvânt, layer wrapper pentru LSTM

Din cauza limitărilor hardware, a trebuit să ne structurăm analiza, alegând un subset din dataset pentru a obține un traducător cât de cât funcțional. Intenția noastră nu este aceea de a obține un model aproape perfect de traducere ci să facem analiză asupra lui.

Antrenarea modelului a fost făcută atât local cât și pe clusterul facultății, pe coada xl, utilizând următoarele comenzi:

```
srun -p xl --gres gpu:2 -A student --cpus-per-task=8 --mem-per-cpu=16G --time 0-23 --pty /bin/bash

singularity run --nv docker://tensorflow/tensorflow:latest-gpu
```

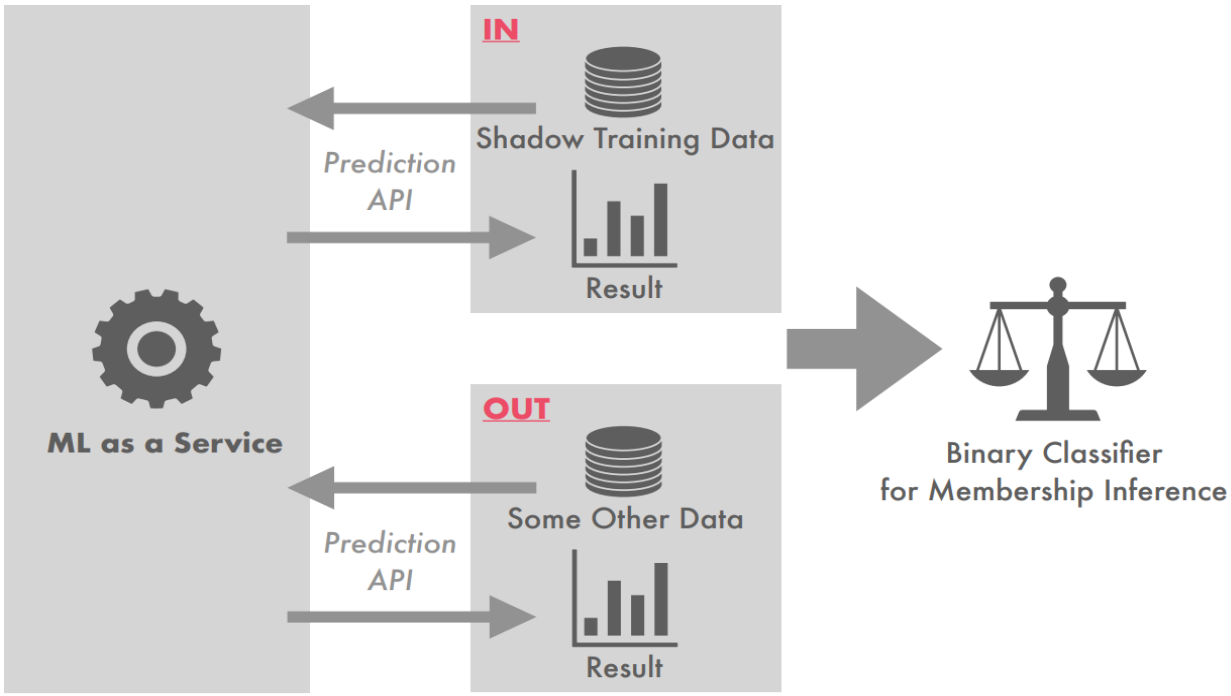
Antrenarea a durat mai puțin pe cluster decât local (antrenat pe CPU Ryzen 7 5800H, ca pe GPU nu aveam suficientă memorie).

Pentru a nu invoca „coincidența” în rezultatele noastre, am testat mai multe scenarii: model subantrenat (antrenat puțin pe un set de date limitat), antrenat suficient pe un subset de date dar și antrenat prea mult pe un subset mai mic de date (overfit). Am dorit să ilustrăm faptul că ipoteza noastră ține indiferent de gradul de antrenare al modelului.
Pentru a reproduce atacul, am folosit un model bazat pe LSTM pe care l-am antrenat pe mai multe traduceri din germană în engleză.

Pentru comparația mai multor outcome-uri, am split-uit diferit datasetul în rulări diferite. După care am folosit mai multe modele de clasificare binară (Naive Bayes, Random Forest, Nearest Neighbour, Perceptron, Multi-layer Perceptron (MLP), Decision Tree cu număr de sample-uri din dataset folosite la antrenare egal cu numărul de sample-uri nefolosite la antrenarea modelului de traducere. În “df” stocăm propoziția în germană, traducerea originală, traducerea obținută prin inferare, BLEU score generat în urma evaluării traducerii și label-ul final care este 0 dacă nu a fost folosit la antrenare și 1 dacă a fost folosit.

Pasul actual este să învățăm modelul de clasificare binară să facă o distincție clară între sample-urile folosite la antrenare față de cele nefolosite. Scopul acestui pas este acela de a încerca să combatem ipoteza acestei lucrări. În cazul în care reușim în unul din scenarii să obținem un model de clasificare binară care face clară distincția (acurateț̦e mare) atunci ipoteza este invalidată.

În caz contrar, ipoteza se menține mai departe și este aplicabilă și pe modelul antrenat de noi, deci acesta este protejat în fața atacurilor de membership inference.



Semnificația acurateții modelului de clasificare binară este următoarea: dacă acuratețea este în jur de 50%, clasificarea binară este echivalentă cu a fi aleatorie și atunci modelul de traducere este în siguranță din punct de vedere al atacurilor de membership inference, adică nu se poate trage o concluzie privind folosirea anumitor date în procesul de antrenare cu certitudine.

Dacă acuratețea este cu mult peste 50%, atunci putem determina dacă un anumit sample a fost sau nu folosit în cadrul procesului de antrenare și deci avem o breșă de securitate în modelul de traducere, în contradicție totală cu ipoteza noastră.

Mai este și cazul în care se obține o acuratețe de sub 50%, doar că acest caz duce modelul de clasificare binară într-o poziție în care nu are rata de predicție deloc relevantă, și astfel orice observație obținută de pe urma acestuia este nulă.

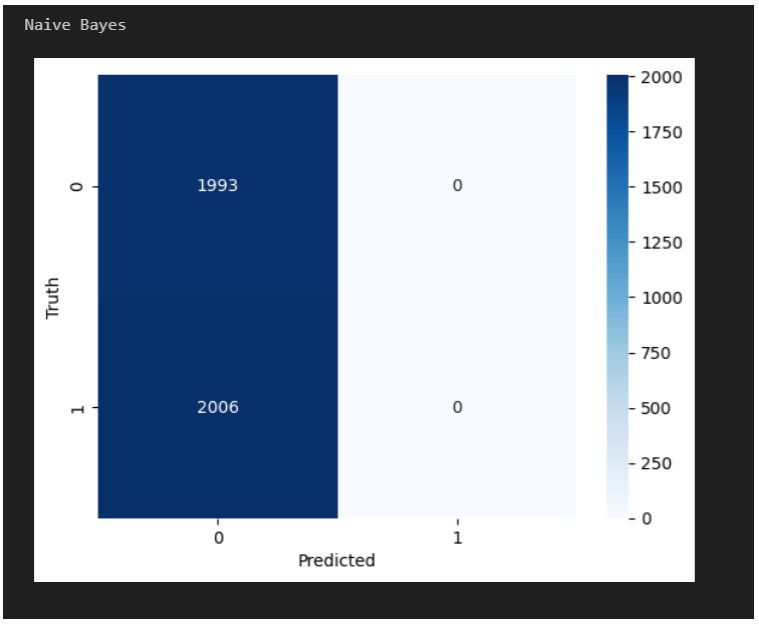
1. Model traducere: Training set: 38000, Test set: 12000; Model clasificare: Training set: 80 % din 20000 (dintre care 10000 folosite la antrenarea modelului de traducere și 10000 nu), Test set: 20%

Descrierea modelului: Mai puțin antrenat (repetări) pe un set de date mai mare (10 Epochs, 64 batch_size pe 38000 propoziții)

```
None
Epoch 1/10

Epoch 1: val_loss improved from inf to 3.27686, saving model to model.h5
C:\Users\farha\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2
  saving_api.save_model(
594/594 - 168s - loss: 3.6443 - val_loss: 3.2769 - 168s/epoch - 283ms/step
Epoch 2/10

Epoch 2: val_loss improved from 3.27686 to 3.06167, saving model to model.h5
594/594 - 158s - loss: 3.1410 - val_loss: 3.0617 - 158s/epoch - 265ms/step
Epoch 3/10

Epoch 3: val_loss improved from 3.06167 to 2.86876, saving model to model.h5
594/594 - 152s - loss: 2.9226 - val_loss: 2.8688 - 152s/epoch - 256ms/step
Epoch 4/10

Epoch 4: val_loss improved from 2.86876 to 2.67325, saving model to model.h5
594/594 - 160s - loss: 2.6939 - val_loss: 2.6732 - 160s/epoch - 269ms/step
Epoch 5/10

Epoch 5: val_loss improved from 2.67325 to 2.43788, saving model to model.h5
594/594 - 147s - loss: 2.4439 - val_loss: 2.4379 - 147s/epoch - 248ms/step
Epoch 6/10

Epoch 6: val_loss improved from 2.43788 to 2.23229, saving model to model.h5
594/594 - 147s - loss: 2.1774 - val_loss: 2.2323 - 147s/epoch - 247ms/step
Epoch 7/10

Epoch 7: val_loss improved from 2.23229 to 2.07888, saving model to model.h5
594/594 - 153s - loss: 1.9405 - val_loss: 2.0789 - 153s/epoch - 258ms/step
...
Epoch 10/10

Epoch 10: val_loss improved from 1.83029 to 1.75242, saving model to model.h5
594/594 - 156s - loss: 1.3682 - val_loss: 1.7524 - 156s/epoch - 263ms/step
```

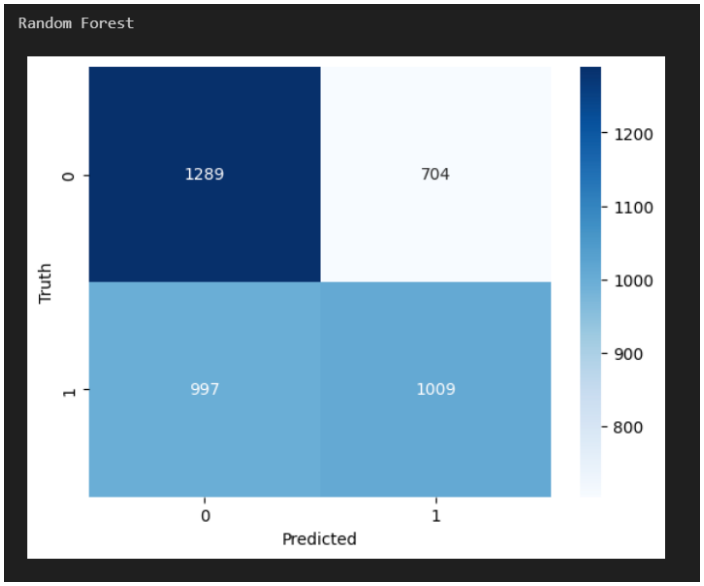
a) Naive Bayes

```
Accuracy: 0.4983745936484121
Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67      1993
           1       0.00      0.00      0.00      2006

    accuracy                           0.50      3999
   macro avg       0.25      0.50      0.33      3999
weighted avg       0.25      0.50      0.33      3999
```



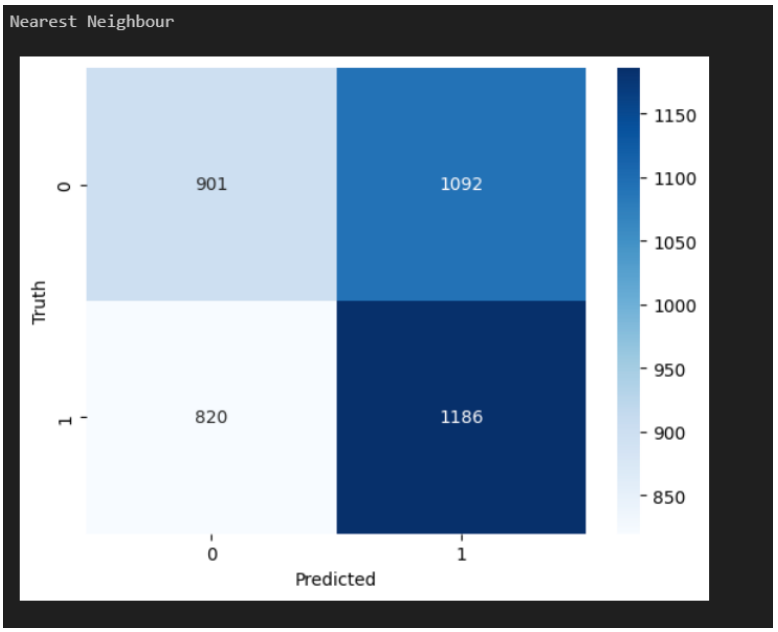
b) Random Forest

```
Accuracy: 0.5746436609152288
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.65      0.60      1993
           1       0.59      0.50      0.54      2006

    accuracy                           0.57      3999
   macro avg       0.58      0.57      0.57      3999
weighted avg       0.58      0.57      0.57      3999
```



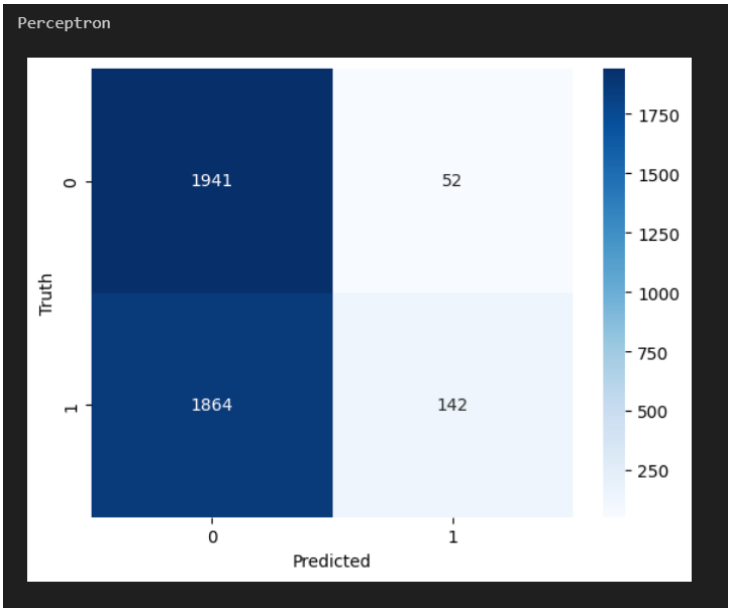
c) Nearest Neighbour

```
Accuracy: 0.5218804701175294
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.45      0.49      1993
           1       0.52      0.59      0.55      2006

    accuracy                           0.52      3999
   macro avg       0.52      0.52      0.52      3999
weighted avg       0.52      0.52      0.52      3999
```



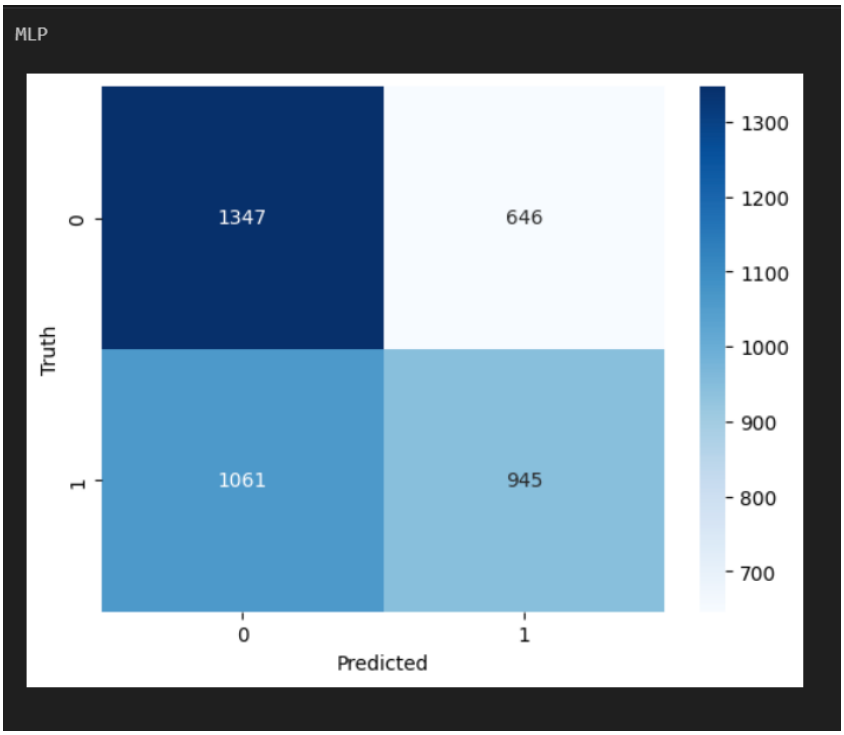
d) Perceptron

```
Accuracy: 0.5208802200550138
Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.97      0.67      1993
           1       0.73      0.07      0.13      2006

    accuracy                           0.52      3999
   macro avg       0.62      0.52      0.40      3999
weighted avg       0.62      0.52      0.40      3999
```



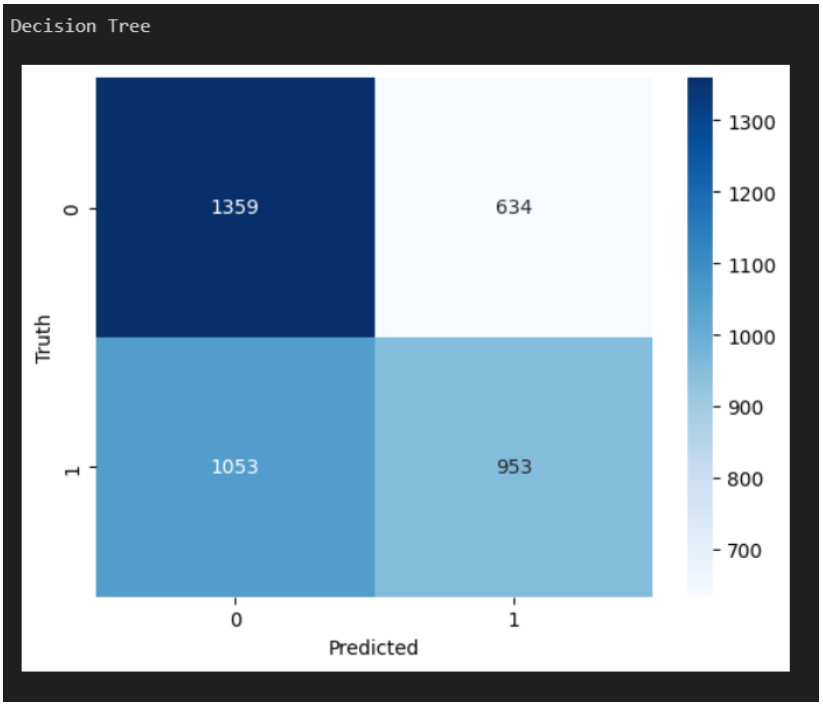
e) Multi-layer Perceptron (MLP)

```
Accuracy: 0.5731432858214554
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.68      0.61      1993
           1       0.59      0.47      0.53      2006

    accuracy                           0.57      3999
   macro avg       0.58      0.57      0.57      3999
weighted avg       0.58      0.57      0.57      3999
```

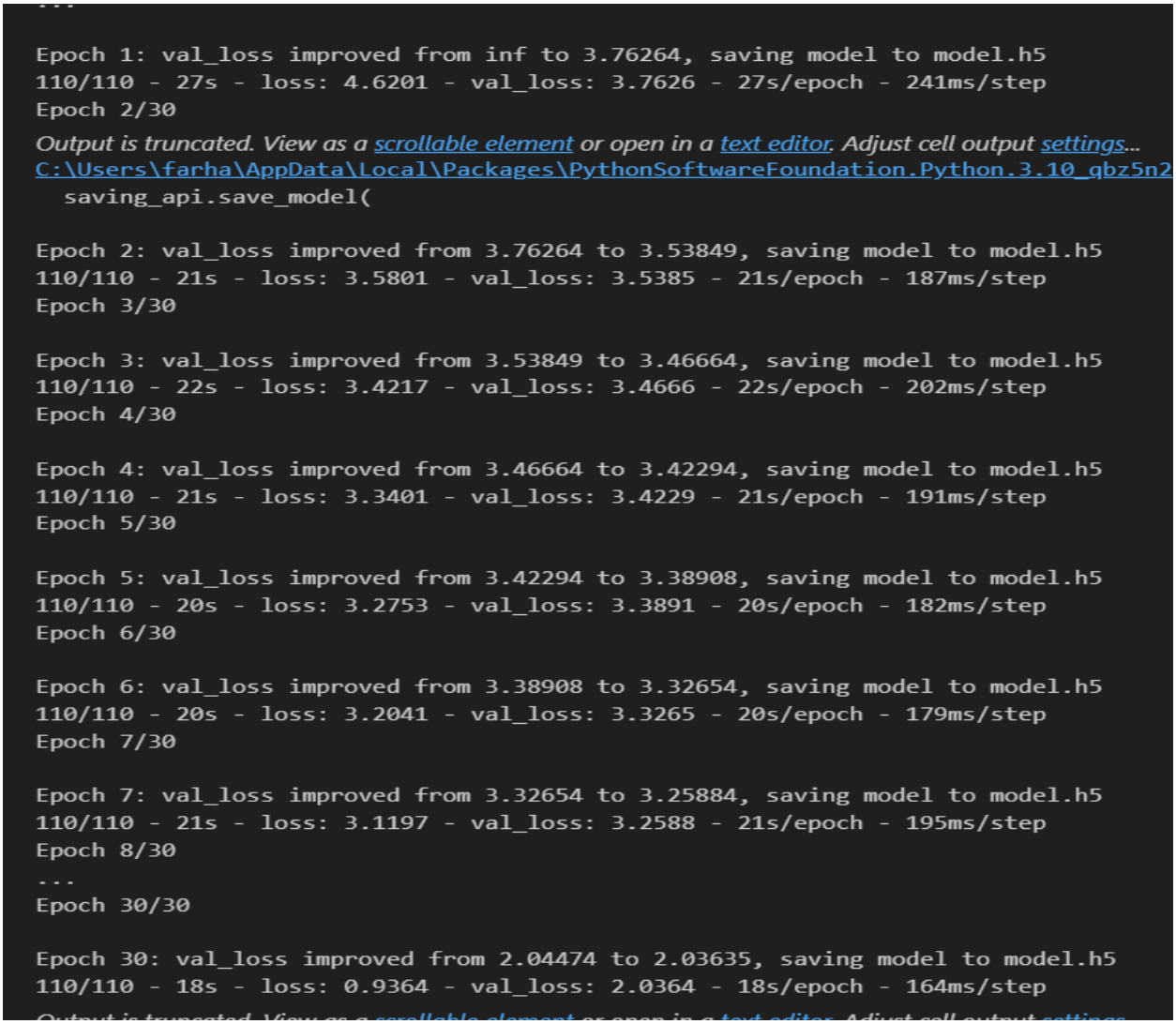


f) Decision Tree

```
Accuracy: 0.5781445361340335
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.68      0.62      1993
           1       0.60      0.48      0.53      2006

    accuracy                           0.58      3999
   macro avg       0.58      0.58      0.57      3999
weighted avg       0.58      0.58      0.57      3999
```

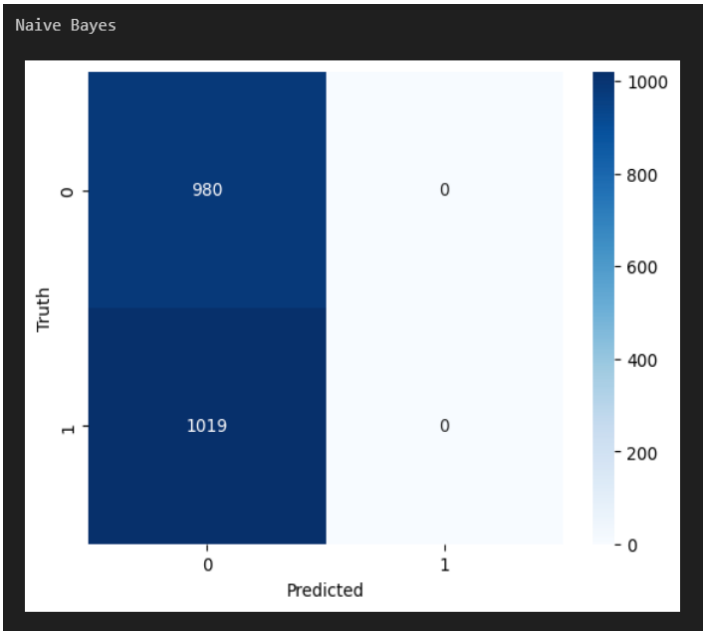


2. Model traducere: Training set: 14000, Test set: 6000; Model clasificare: Training set: 80 % din 10000 (dintre care 5000 folosite la antrenarea modelului de traducere și 5000 nu), Test set: 20%



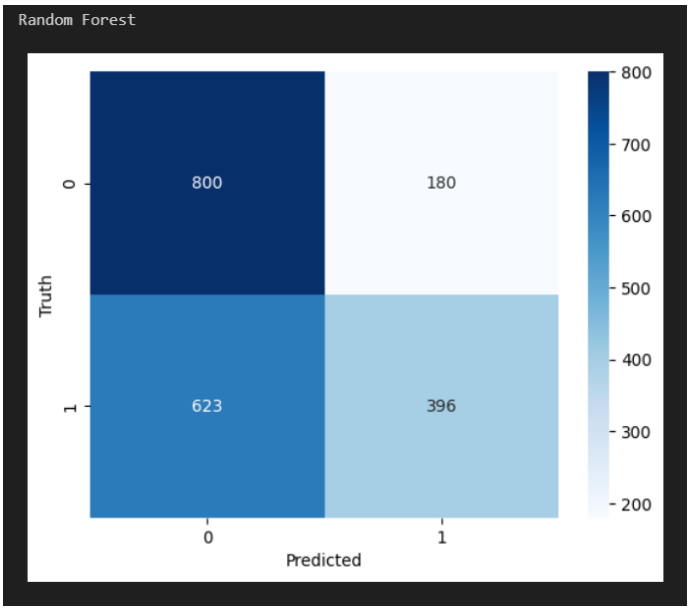
a) Naive Bayes

```
Accuracy: 0.49024512256128067
Classification Report:
              precision    recall  f1-score   support

           0       0.49      1.00      0.66       980
           1       0.00      0.00      0.00      1019

    accuracy                           0.49      1999
   macro avg       0.25      0.50      0.33      1999
weighted avg       0.24      0.49      0.32      1999
```



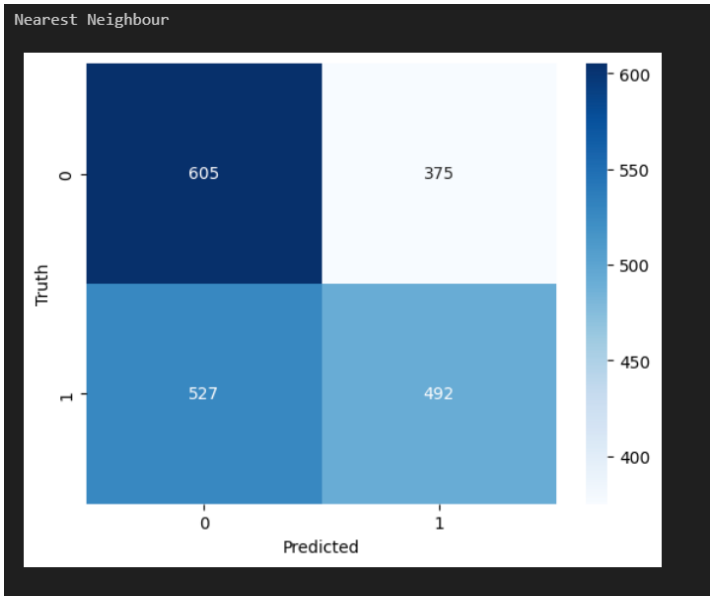
b) Random Forest

```
Accuracy: 0.5982991495747874
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.82      0.67       980
           1       0.69      0.39      0.50      1019

    accuracy                           0.60      1999
   macro avg       0.62      0.60      0.58      1999
weighted avg       0.63      0.60      0.58      1999
```



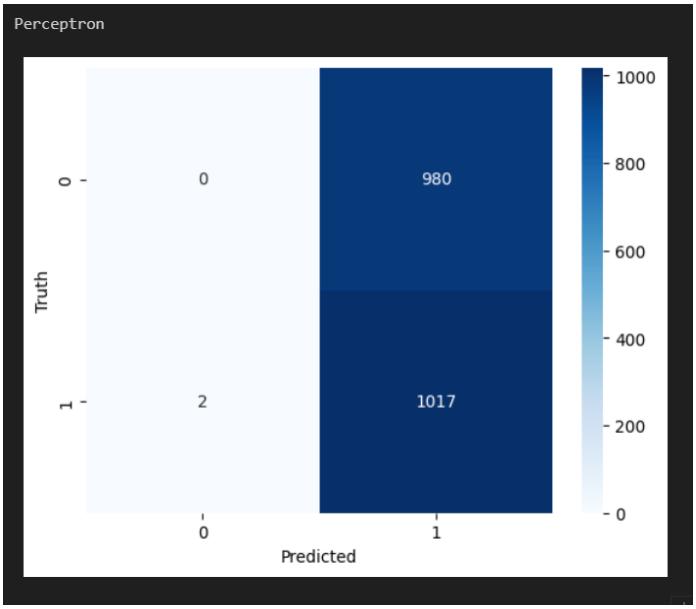
c) Nearest Neighbour

```
Accuracy: 0.5487743871935968
Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.62      0.57       980
           1       0.57      0.48      0.52      1019

    accuracy                           0.55      1999
   macro avg       0.55      0.55      0.55      1999
weighted avg       0.55      0.55      0.55      1999
```



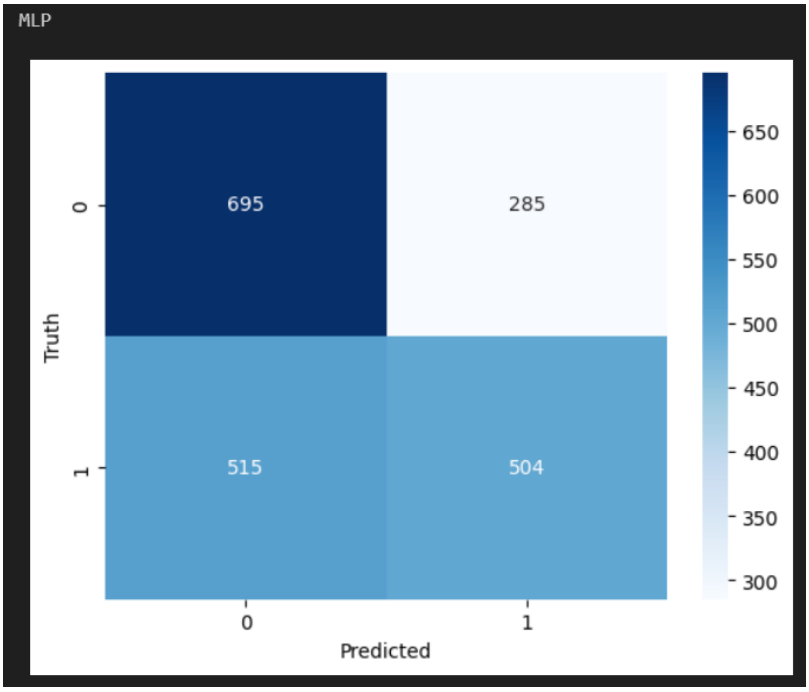
d) Perceptron

```
Accuracy: 0.5087543771885943
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       980
           1       0.51      1.00      0.67      1019

    accuracy                           0.51      1999
   macro avg       0.25      0.50      0.34      1999
weighted avg       0.26      0.51      0.34      1999
```



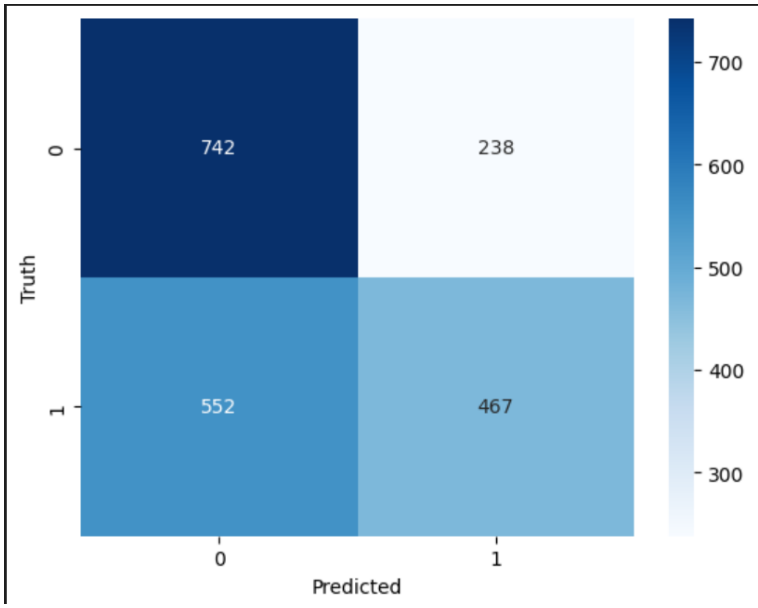
e) Multi-layer Perceptron (MLP)

```
Accuracy: 0.599799899949975
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.71      0.63       980
           1       0.64      0.49      0.56      1019

    accuracy                           0.60      1999
   macro avg       0.61      0.60      0.60      1999
weighted avg       0.61      0.60      0.60      1999
```



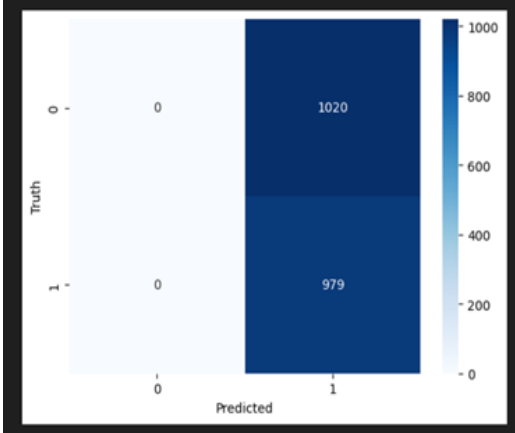
f) Decision Tree

```
Accuracy: 0.6048024012006002
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.76      0.65       980
           1       0.66      0.46      0.54      1019

    accuracy                           0.60      1999
   macro avg       0.62      0.61      0.60      1999
weighted avg       0.62      0.60      0.60      1999
```



3. Model traducere antrenat pe clusterul facultății cu aproximativ 40000 propoziții la antrenare

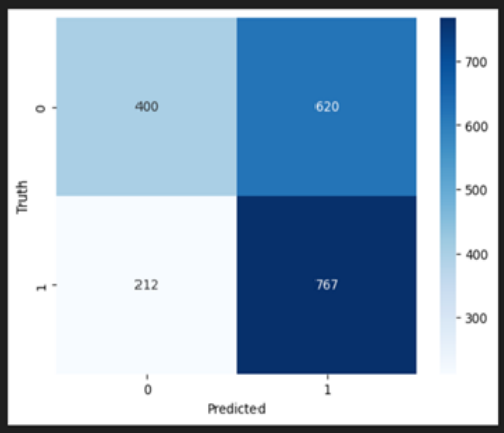
Naive Bayes



Accuracy: 0.4897448724362181
Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1020
1	0.49	1.00	0.66	979
accuracy			0.49	1999
macro avg	0.24	0.50	0.33	1999
weighted avg	0.24	0.49	0.32	1999

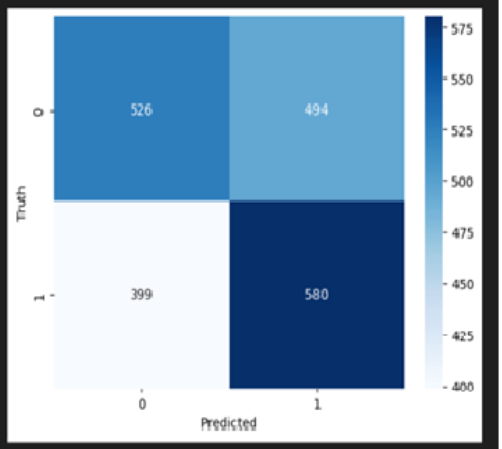
Random Forest



Accuracy: 0.583791895947974
Classification Report:

	precision	recall	f1-score	support
0	0.65	0.39	0.49	1020
1	0.55	0.78	0.65	979
accuracy			0.58	1999
macro avg	0.60	0.59	0.57	1999
weighted avg	0.60	0.58	0.57	1999

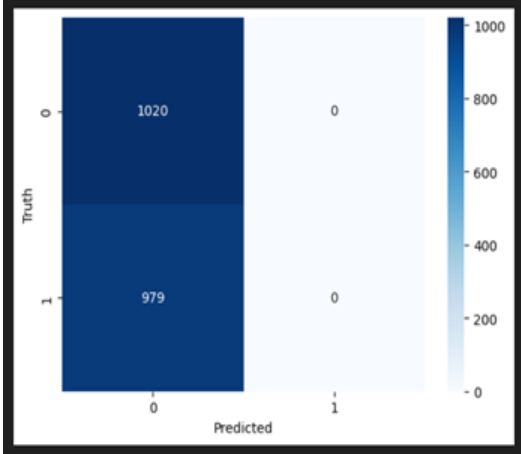
Nearest Neighbour



Accuracy: 0.5532766383191596
Classification Report:

	precision	recall	f1-score	support
0	0.57	0.52	0.54	1020
1	0.54	0.59	0.57	979
accuracy			0.55	1999
macro avg	0.55	0.55	0.55	1999
weighted avg	0.55	0.55	0.55	1999

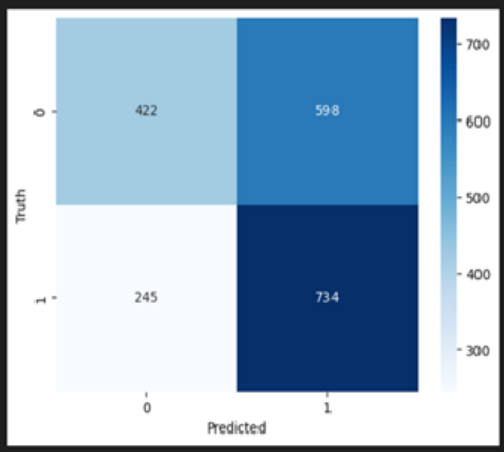
Perceptron



Accuracy: 0.5102551275637819
Classification Report:

	precision	recall	f1-score	support
0	0.51	1.00	0.68	1020
1	0.00	0.00	0.00	979
accuracy			0.51	1999
macro avg	0.26	0.50	0.34	1999
weighted avg	0.26	0.51	0.34	1999

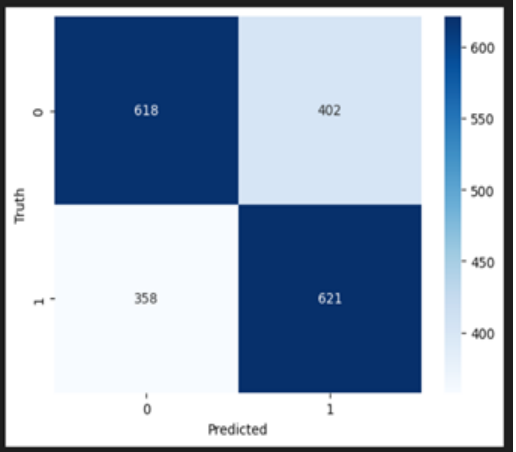
MLP



Accuracy: 0.5782891445722862
Classification Report:

	precision	recall	f1-score	support
0	0.63	0.41	0.50	1020
1	0.55	0.75	0.64	979
accuracy			0.58	1999
macro avg	0.59	0.58	0.57	1999
weighted avg	0.59	0.58	0.57	1999

Decision Tree



Accuracy: 0.6198099049524762
Classification Report:

	precision	recall	f1-score	support
0	0.63	0.61	0.62	1020
1	0.61	0.63	0.62	979
accuracy			0.62	1999
macro avg	0.62	0.62	0.62	1999
weighted avg	0.62	0.62	0.62	1999

6. Concluzii

În concluzie, conform datelor obținute din rularea modelelor de clasificare binară, putem afirma că în mai multe stadii ale nivelului de antrenare pentru modelul antrenat pe subsecțiuni din dataset, acesta nu este vulnerabil la atacurile de tip membership inference întrucât ipoteza lucrării se menține și modelele de clasificare binară au eșuat să ofere o certitudine cu privire la etichetarea unor propoziții ca fiind folosite sau nu la antrenare în modelul de traducere.

7. Bibliografie

[1] Sorami Hisamoto, Matt Post, Kevin Duh; Membership Inference Attacks on Sequence-to-Sequence Models: Is My Data In Your Machine Translation System?. *Transactions of the Association for Computational Linguistics* 2020; 8 49–63. doi: https://doi.org/10.1162/tacl_a_00299

[2] https://www.digitalocean.com/community/tutorials/bleu-score-in-python

[3] https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

[4] https://github.com/sorami/TACL-Membership

[5] https://valueml.com/german-to-english-translator-using-keras-and-tensorflow-using-lstm-model/

[6] https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/