FAKE NEWS DETECTOR

NATURAL LANGUAGE PROCESSING

DESIREE MAESTRI
JAN DIRK
JONATHAN SADA

Summary

PROCESS

1. Environment Setup

TOOLS AND LIBRARIES

- pandas,
- sklearn,
- xgboost,
- transformers,
- hugginface,
- python

CUSTOM UTILITY FUNCTIONS

- helpers

GLOBAL PARAMETERS

- warnings
- seed = 42

2. Data Loading and Preprocessing

2.1. INITIAL DATA INSPECTION
2.2. DATA CLEANING
2.3. DATA SPLITTING

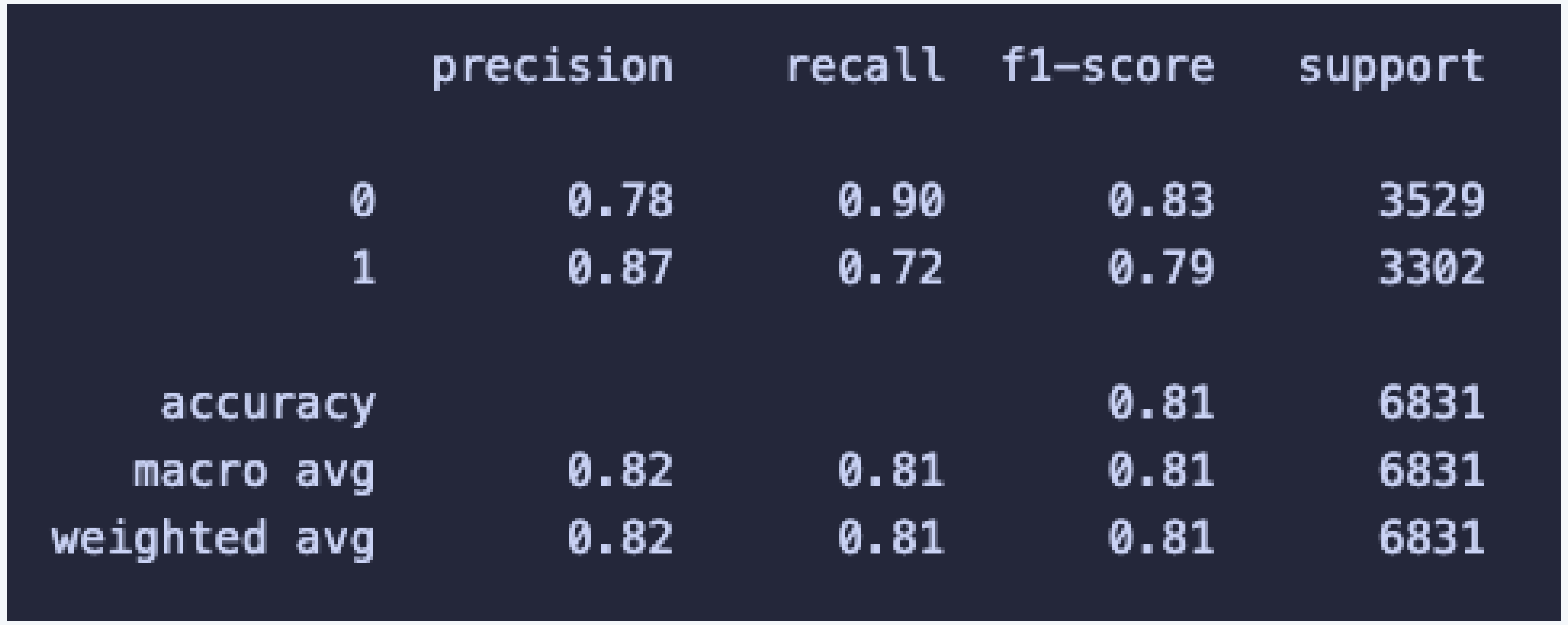
- No code : HMTL, CSS, JS
- Removed all special characters and numbers
- X = cleaned data
- Split 20% training and testing

```
Original: trump is so obsessed he even has obama,s name coded into his website (images)
Cleaned:  trump is so obsessed he even has obama name coded into his website images
```

3. BASELINE MODEL

LEARNED IN CLASS

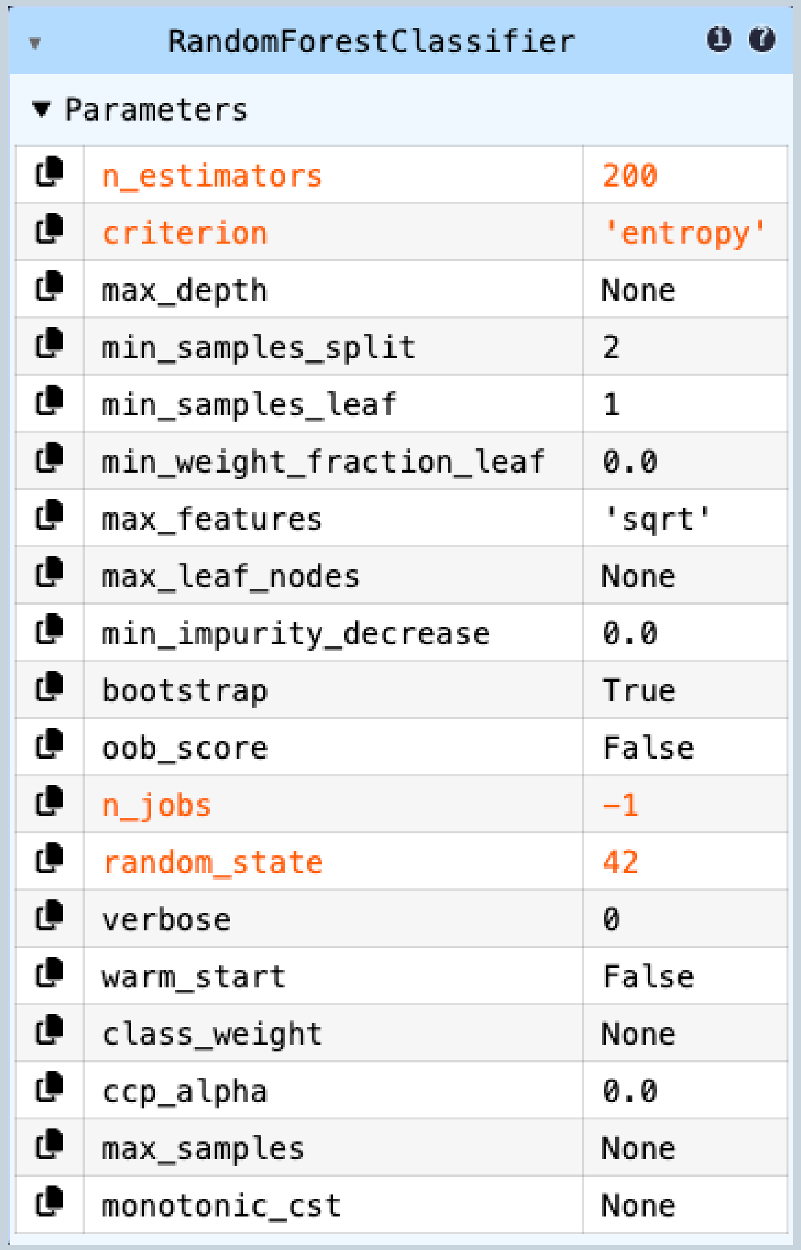
3. Baseline Model



```
              precision    recall  f1-score   support

           0       0.78      0.90      0.83      3529
           1       0.87      0.72      0.79      3302

    accuracy                           0.81      6831
   macro avg       0.82      0.81      0.81      6831
weighted avg       0.82      0.81      0.81      6831
```



RandomForestClassifier

▼ Parameters

n_estimators	200
criterion	'entropy'
max_depth	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	'sqrt'
max_leaf_nodes	None
min_impurity_decrease	0.0
bootstrap	True
oob_score	False
n_jobs	-1
random_state	42
verbose	0
warm_start	False
class_weight	None
ccp_alpha	0.0
max_samples	None
monotonic_cst	None

4. CLASSICAL ML MODEL EXPERIMENTATION

4.1. Classical ML Model Experimentation

- KNeighbors
- LogisticRegression
- DecisionTree
- RandomForest
- AdaBoost
- XGBoost
- BernoulliNB

GLOBAL SETUP

DATA SOURCE

cleaned X
split 80-20

4.2. Performance Summary

SETUP			TRAIN RESULTS				TEST RESULTS			
model	**vectorizer**	**fit_time**	**accuracy_train**	**precision_train**	**recall_train**	**f1_train**	**accuracy_test**	**precision_test**	**recall_test**	**f1_test**
KNeighborsClassifier	CountVectorizer	0.00095	0.60730	0.95502	0.20146	0.33273	0.57385	0.89655	0.13386	0.23294
LogisticRegression	**CountVectorizer**	**4.64458**	**0.98122**	**0.97620**	**0.98539**	**0.98077**	**0.94935**	**0.93962**	**0.95669**	**0.94808**
DecisionTreeClassifier	CountVectorizer	2.38226	1.0	1.0	1.0	1.0	0.87937	0.88170	0.86675	0.87416
RandomForestClassifier	CountVectorizer	1.13210	1.0	1.0	1.0	1.0	0.92973	0.93847	0.91460	0.92638
AdaBoostClassifier	CountVectorizer	0.54137	0.77351	0.69457	0.95308	0.80354	0.78232	0.70109	0.95821	0.80972
XGBClassifier	CountVectorizer	10.66694	0.91684	0.88067	0.95880	0.91808	0.90807	0.87451	0.94549	0.90861
BernoulliNB	CountVectorizer	0.00299	0.95271	0.93976	0.96453	0.95198	0.94481	0.93333	0.95397	0.94354
KNeighborsClassifier	TfidfVectorizer	0.00099	0.93038	0.91941	0.93907	0.92914	0.89431	0.87566	0.91066	0.89281
LogisticRegression	TfidfVectorizer	1.88468	0.96175	0.95297	0.96912	0.96098	0.94481	0.93180	0.95578	0.94364
DecisionTreeClassifier	TfidfVectorizer	2.22129	1.0	1.0	1.0	1.0	0.88333	0.87221	0.88886	0.88046
RandomForestClassifier	**TfidfVectorizer**	**1.00727**	**1.0**	**1.0**	**1.0**	**1.0**	**0.93442**	**0.92445**	**0.94125**	**0.93277**
AdaBoostClassifier	TfidfVectorizer	1.52704	0.79144	0.71554	0.94758	0.81537	0.79857	0.71996	0.95457	0.82083
XGBClassifier	TfidfVectorizer	15.62038	0.93247	0.90311	0.96453	0.93281	0.91802	0.88663	0.95215	0.91822
BernoulliNB	TfidfVectorizer	0.00306	0.95271	0.93976	0.96453	0.95198	0.94481	0.93333	0.95397	0.94354
KNeighborsClassifier	TfidfVectorizer	0.00087	0.93038	0.91941	0.93907	0.92914	0.89431	0.87566	0.91066	0.89281
LogisticRegression	TfidfVectorizer	1.98830	0.96175	0.95297	0.96912	0.96098	0.94481	0.93180	0.95578	0.94364
DecisionTreeClassifier	TfidfVectorizer	2.22913	1.0	1.0	1.0	1.0	0.88333	0.87221	0.88886	0.88046
RandomForestClassifier	TfidfVectorizer	0.96854	1.0	1.0	1.0	1.0	0.93442	0.92445	0.94125	0.93277
AdaBoostClassifier	TfidfVectorizer	1.53524	0.79144	0.71554	0.94758	0.81537	0.79857	0.71996	0.95457	0.82083
XGBClassifier	TfidfVectorizer	18.49969	0.93247	0.90311	0.96453	0.93281	0.91802	0.88663	0.95215	0.91822
BernoulliNB	TfidfVectorizer	0.00308	0.95271	0.93976	0.96453	0.95198	0.94481	0.93333	0.95397	0.94354

4.3. Top-Performing Models x N-Grams

SETUP				TRAIN RESULTS				TEST RESULTS			
model	vectorizer	ngram_range	fit_time	accuracy_train	precision_train	recall_train	f1_train	accuracy_test	precision_test	recall_test	f1_test
LogisticRegression	CountVectorizer	(1, 1)	4.63550	0.98122	0.97620	0.98539	0.98077	0.94935	0.93962	0.95669	0.94808
LogisticRegression	CountVectorizer	(1, 2)	4.35019	0.99824	0.99685	0.99955	0.99819	0.95286	0.94083	0.96305	0.95181
LogisticRegression	CountVectorizer	(2, 2)	4.24104	0.99535	0.99163	0.99887	0.99524	0.90119	0.86496	0.94276	0.90219
LogisticRegression	CountVectorizer	(1, 3)	4.29268	0.99938	0.99872	1.0	0.99936	0.95023	0.93739	0.96124	0.94916
LogisticRegression	CountVectorizer	(2, 3)	3.64245	0.99795	0.99580	1.0	0.99790	0.89709	0.85652	0.94549	0.89881
LogisticRegression	CountVectorizer	(3, 3)	4.90670	0.99521	0.99023	1.0	0.99509	0.76958	0.68693	0.96154	0.80136
RandomForestClassifier	TfidfVectorizer	(1, 1)	0.99577	1.0	1.0	1.0	1.0	0.93442	0.92445	0.94125	0.93277
RandomForestClassifier	TfidfVectorizer	(1, 2)	4.49988	1.0	1.0	1.0	1.0	0.91304	0.88953	0.93640	0.91236
RandomForestClassifier	TfidfVectorizer	(2, 2)	6.34077	0.99982	1.0	0.99962	0.99981	0.80530	0.85699	0.71684	0.78067
RandomForestClassifier	TfidfVectorizer	(1, 3)	9.69194	1.0	1.0	1.0	1.0	0.90470	0.88521	0.92247	0.90346
RandomForestClassifier	TfidfVectorizer	(2, 3)	30.27073	0.99978	1.0	0.99955	0.99977	0.79827	0.85420	0.70260	0.77102
RandomForestClassifier	TfidfVectorizer	(3, 3)	15.50311	0.99890	1.0	0.99774	0.99887	0.63636	0.90495	0.27680	0.42393

5. CURRENT TOP CLASSICAL MODEL

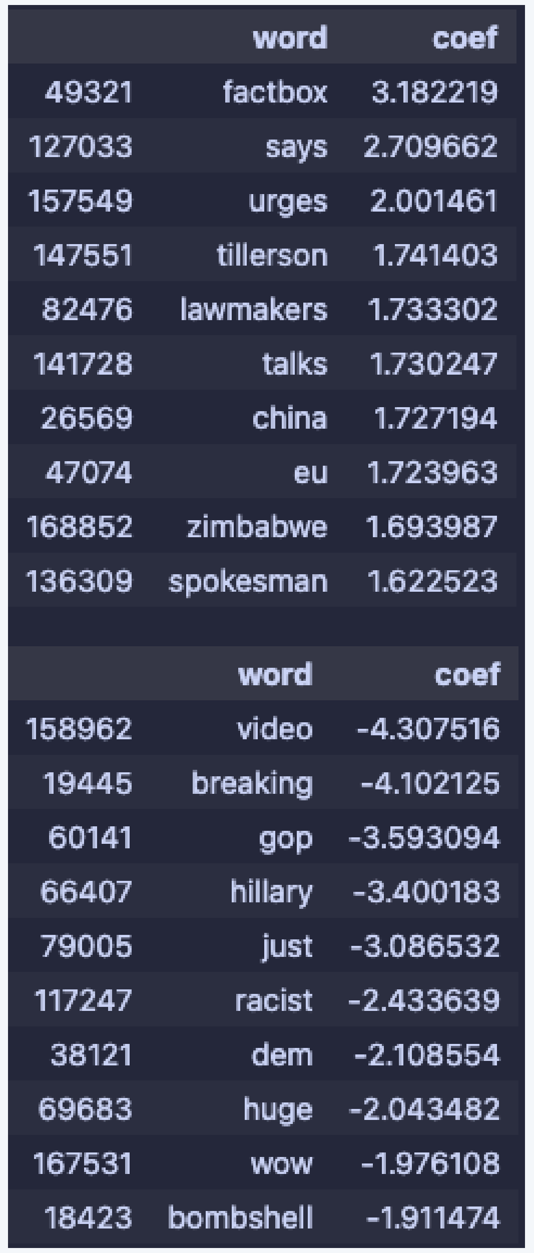
DATA SOURCE	Cleaned 80-20 split
MODEL	LogisticRegression
FEATURE VECTORIZATION	CountVectorizer
N-GRAM CONFIGURATION	1, 2
RESULTS	Accuracy (Train) 0.98 Recall (Train) 0.98 Accuracy (test) 0.94 Recall (Test) 0.95

6. TOP CLASSICAL MODEL OPTIMIZATION

6.1. Custom Stop Word Removal

- After removing the words iterativelly based on the coeficient of the words we see a decrease in accuracy (0.844 by keeping only words with coeficient inbetween -1 and 1)
- The recall stays relevant in between 0.937 and 0.963
- This test shows that the model is

SETUP				TRAIN RESULTS				TEST RESULTS			
model	vectorizer	ngram_	fit_ti	accuracy_	precision	recall_tr	f1_train	accuracy_	precision	recall_te	f1_test
LogisticRegres	CountVectoriz	(1, 1)	0.12015	0.98122	0.97620	0.98539	0.98077	0.94935	0.93962	0.95669	0.94808
LogisticRegres	CountVectoriz	(1, 2)	1.07524	0.99824	0.99685	0.99955	0.99819	0.95286	0.94083	0.96305	0.95181
LogisticRegres	CountVectoriz	(2, 2)	0.28462	0.99535	0.99163	0.99887	0.99524	0.90119	0.86496	0.94276	0.90219
LogisticRegres	CountVectoriz	(1, 3)	0.76832	0.99938	0.99872	1.0	0.99936	0.95023	0.93739	0.96124	0.94916
LogisticRegres	CountVectoriz	(2, 3)	0.53244	0.99795	0.99580	1.0	0.99790	0.89709	0.85652	0.94549	0.89881
LogisticRegres	CountVectoriz	(3, 3)	0.44946	0.99521	0.99023	1.0	0.99509	0.76958	0.68693	0.96154	0.80136



	word	coef
49321	factbox	3.182219
127033	says	2.709662
157549	urges	2.001461
147551	tillerson	1.741403
82476	lawmakers	1.733302
141728	talks	1.730247
26569	china	1.727194
47074	eu	1.723963
168852	zimbabwe	1.693987
136309	spokesman	1.622523

	word	coef
158962	video	-4.307516
19445	breaking	-4.102125
60141	gop	-3.593094
66407	hillary	-3.400183
79005	just	-3.086532
117247	racist	-2.433639
38121	dem	-2.108554
69683	huge	-2.043482
167531	wow	-1.976108
18423	bombshell	-1.911474

6.2. Alternative Text Cleaning

SETUP				TRAIN RESULTS				TEST RESULTS			
model	vectorizer	fit_time	cleaning	accuracy_train	precision_train	recall_train	f1_train	accuracy_test	precision_test	recall_test	f1_test
LogisticRegression	CountVectorizer	4.65329480171203	less_cleaning	0.9984261119	0.997219926	0.999548124	0.998382668	0.952715561	0.94028968371	0.963355542	0.951682872
LogisticRegression	CountVectorizer	4.53271770477294	clean_serie	0.998243109	0.996845425	0.999548124	0.998194945	0.952861952	0.9408284023	0.963052695	0.951810835
LogisticRegression	CountVectorizer	4.93542718887329	no_cleanning	0.9984261119	0.997219926	0.999548124	0.998382668	0.952569169	0.9397519196	0.963658388	0.951555023

7. TRANSFORMER-BASED MODELS EXPERIMENTATION

7.3. Performance summary

model	accuracy	precision	recall	f1
mrm8488/bert-tiny-finetuned-fake-news-detection	0.465214335910049	0.474819068126084	0.957599517490953	0.634851453476748
jy46604790/Fake-News-Bert-Detect	0.652319044272663	0.983162217659137	0.288781664656212	0.446433566433566
yasmine-11/distilbert_fake_news	0.484425899491233	0.485064695009242	0.988175930109956	0.650714144019043

8. TRANSFER LEARNING EVALUATION

8.1. Transfer learning configuration

DATA SOURCE	Cleaned + 80-20 split	
MODELS	**distilbert-base-uncased**	**bert-base-uncased**
EVALUATED ON	eval_recall	
RESULTS	{'eval_loss': 0.07611262053251266, 'eval_accuracy': 0.9817010686575904, 'eval_recall': 0.9766807995154452, 'eval_runtime': 49.1314, 'eval_samples_per_second': 139.035, 'eval_steps_per_second': 8.691, 'epoch': 3.0}	{'eval_loss': 0.09613175690174103, 'eval_accuracy': 0.9822866344605475, 'eval_recall': 0.9887946698970321, 'eval_runtime': 27.6307, 'eval_samples_per_second': 247.225, 'eval_steps_per_second': 15.454, 'epoch': 3.0}

9. FINAL MODEL APPLICATION

9.1. Champion Classical Model

Cleaned 80-20 split

LogisticRegression

CountVectorizer

N-gram: 1, 2

TRAIN

ACCURACY: **0.98**

RECALL: **0.98**

TEST

ACCURACY: **0.94**

RECALL: **0.95**

THANKYOU

QUESTIONS?