Classifying Text by Time Period

Zachary Kelly, Sasha Casada,
& Nathan Le

GOALS & MOTIVATIONS

Goals:

- If texts can be dated by **content** ⟶ use ML to date **historical texts**?
- Can existing **machine learning** models help?

Motivations:

- Interest in **Natural Language Processing (NLP)**.
  - *Could we explore a problem that could utilize NLP & ML?*
- **Real-world application** of course content.

APPROACH

STAGE 1

DATA PREPROCESSING

Gather, *Tokenize, Label

STAGE 2

MODEL SELECTION AND TRAINING

Select Models, Build Pipeline

STAGE 3

EVALUATION AND OPTIMIZATION

Test Performance, Refining, & Validation

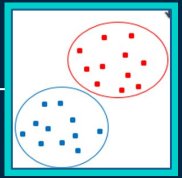
STAGE 4

ANALYSIS

Final Evaluation, Interpret Results

***Tokenization:** Systematically breaking down a text into units (often called words, or subwords), and in the process, removing irrelevant features like capitalization, whitespace, punctuation, etc.

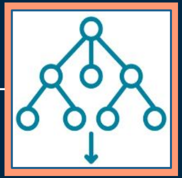
MODELS



01

NAIVE BAYES

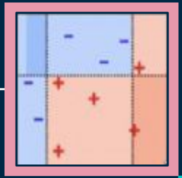
1. Classifies extremely **quickly**.
2. Views all features as being **independant**.



02

RANDOM FOREST

1. Can handle **high dimensional data** well.
2. Reduces **overfitting**.
3. Robust against **noise**.



03

ADABOOST

1. Reduces **overfitting**.
2. Robust against **noise**.
3. Handles **class imbalance** well

VECTORIZATION

Word Vector: A vector such that words are *mapped* to indices.

Vectorization Algorithms: Assign a numeric feature value per word *w* in the word vector.

Example using BoW:

- **Ex. 1** ("The dog sat, the cat sat too.")
- **Ex. 2** ("The dog and the cat and the other cat all sat.")

Text #	f_0 'the'	f_1 'dog'	f_2 'cat'	f_3 'sat'
Ex. 1	2	1	1	1
Ex. 2	3	1	2	1

VECTORIZATION ALGORITHMS

Bag-of-Words (BoW): "The frequency of occurrence of each word is used as a numeric feature for training a classifier."

Term-Frequency Inverse Document Frequency (TF-IDF):

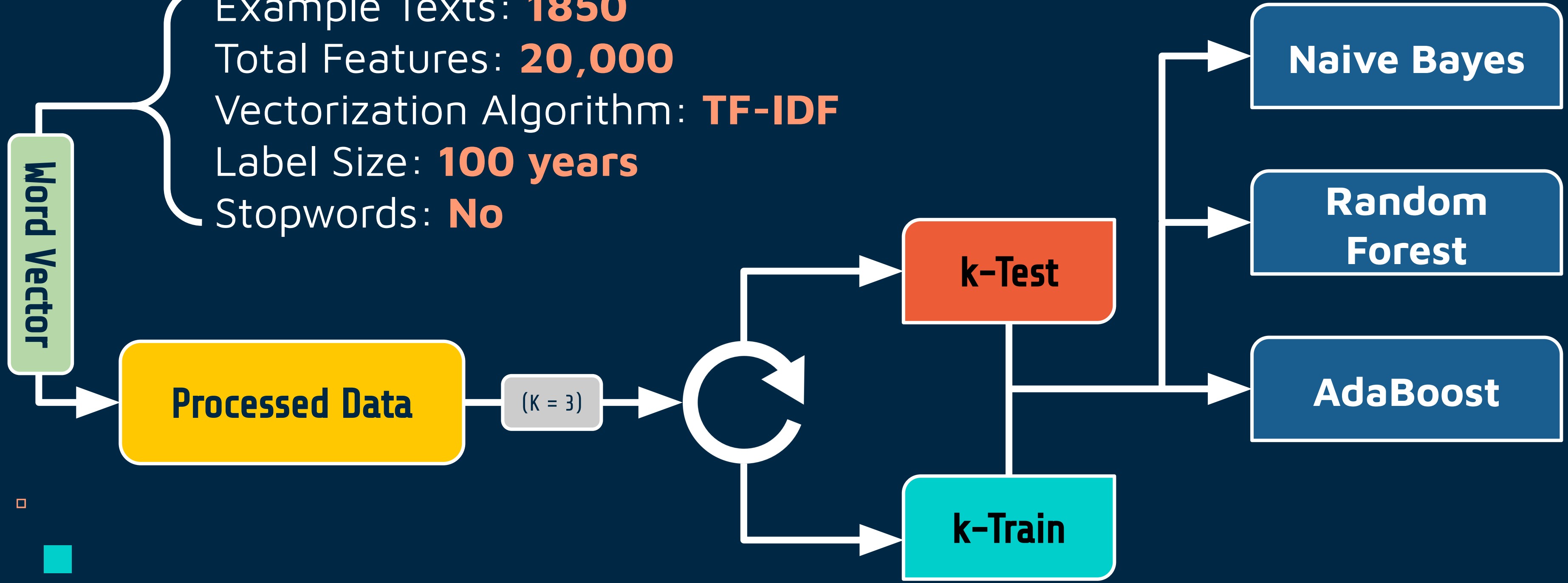
- "Numerically reflects how important a word is to a document."
- TF-IDF value increases proportionally to the number of times a word appears in the document, and is offset by the number of documents in the corpus that contain the word.

$$W_{i,j} = tf_{i,j} \times \log \frac{(N)}{df_i}$$

DATA GENERATION, TRAIN, & TEST PIPELINE

Example Data Overview:

- Example Texts: **1850**
- Total Features: **20,000**
- Vectorization Algorithm: **TF-IDF**
- Label Size: **100 years**
- Stopwords: **No**

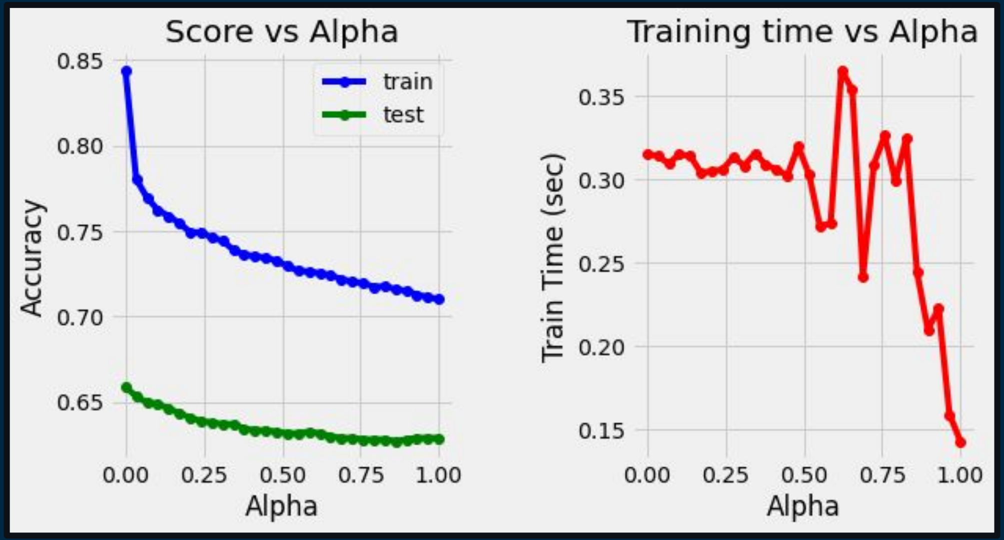


VALIDATION

Randomized Search with **cross validation** across a range of values for each of the values:

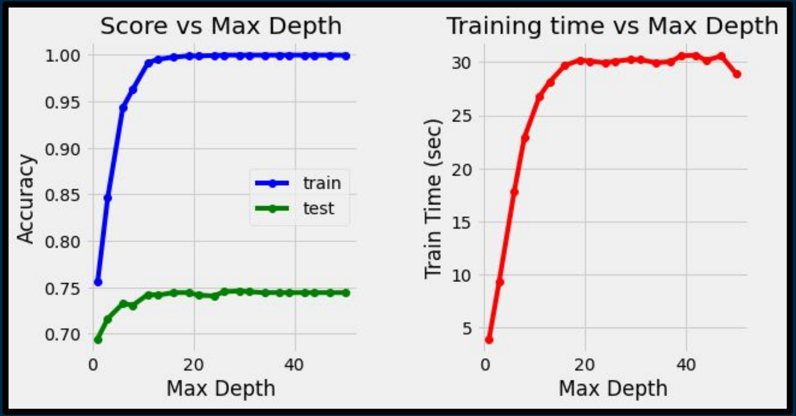
- **Naive Bayes (Hyperparameters):**
  - *alpha* (0, 1]
- **Random Forests (Hyperparameters):**
  - *n_estimators* [200, # unique words * 1/3]
  - *min_samples_split* [2, 10]
  - *min _samples_leaf* [1, 4]
  - *max_features* [‘auto’, ‘sqrt’]
  - *max_depth* `[10, 110]
  - *bootstrap* [true, false]
- **AdaBoost (Hyperparameters):**
  - *n_estimators* [50, 500]
  - *learning_rate* [0.001, 1.0]
  - *base_estimator* [1, 15]

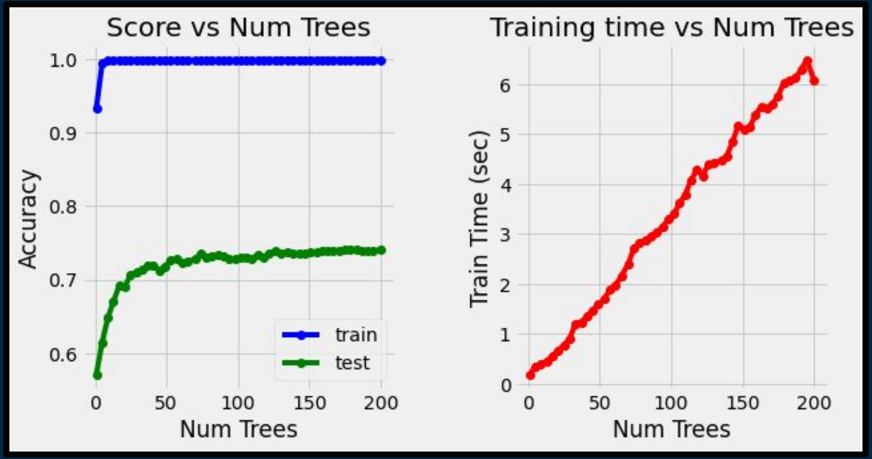
NAIVE BAYES



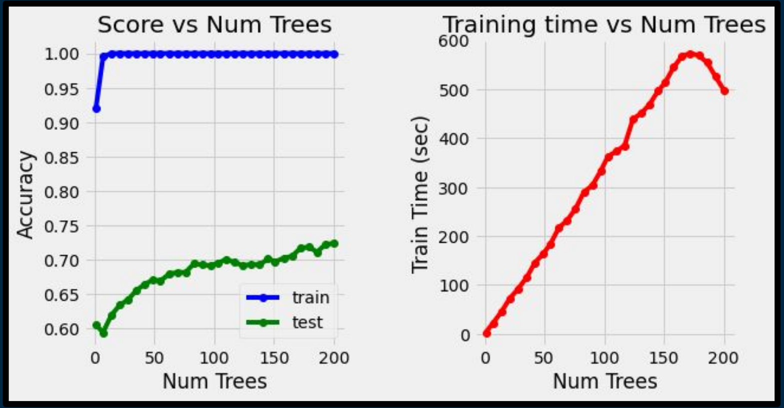
Score vs Alpha
Accuracy
0.85
0.80
0.75
0.70
0.65
train
test
0.00
0.25
0.50
0.75
1.00
Alpha
Training time vs Alpha
Train Time (sec)
0.35
0.30
0.25
0.20
0.15
0.00
0.25
0.50
0.75
1.00
Alpha

RANDOM FOREST

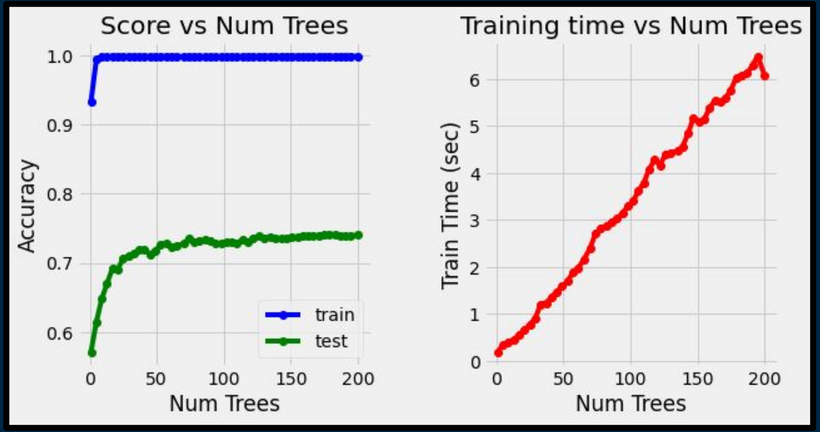




ADABOOST



RANDOM FOREST

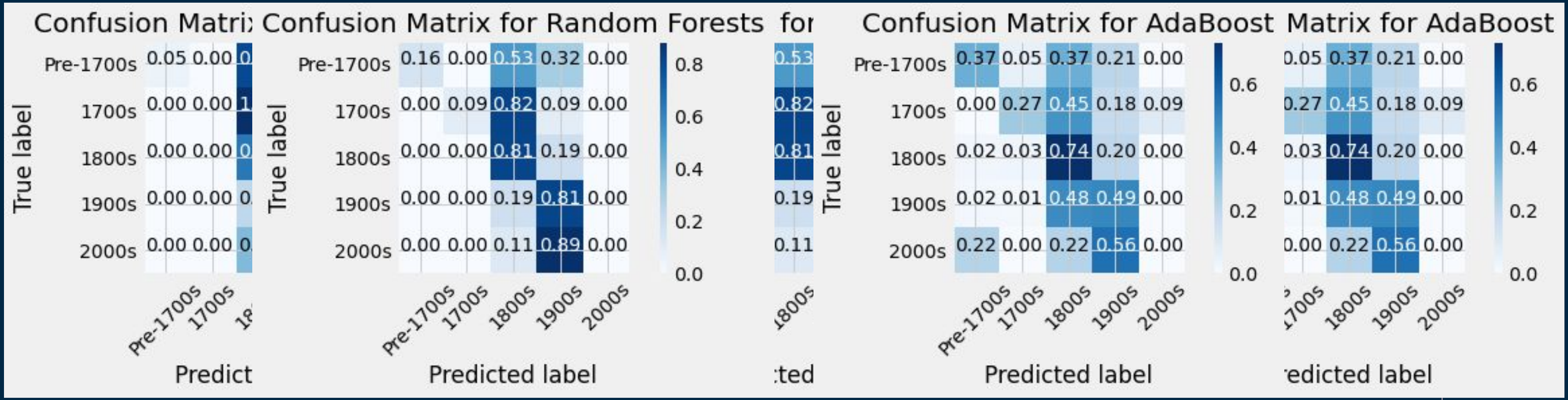


RESULTS

Model	Base	Tuned	Performance Delta
AdaBoost	0.5189	0.7270	0.2081
Naive Bayes	0.6527	0.7054	0.0527
Random Forest	0.7568	0.7773	0.0205

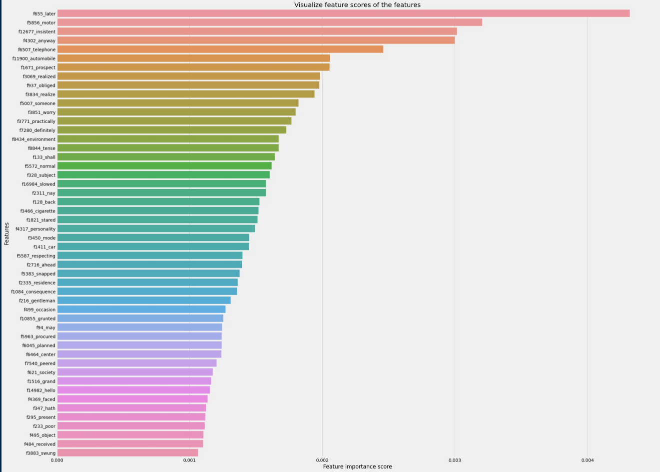
- Random Forest: Best Performing
  - We see the smallest gain in parameter tuning
- AdaBoost: Second Best (Most Potential)
  - Potentially more gains from more trees
- Naive Bayes: Fastest + Simplest
  - Simpler model (probably no further gains) but very fast.

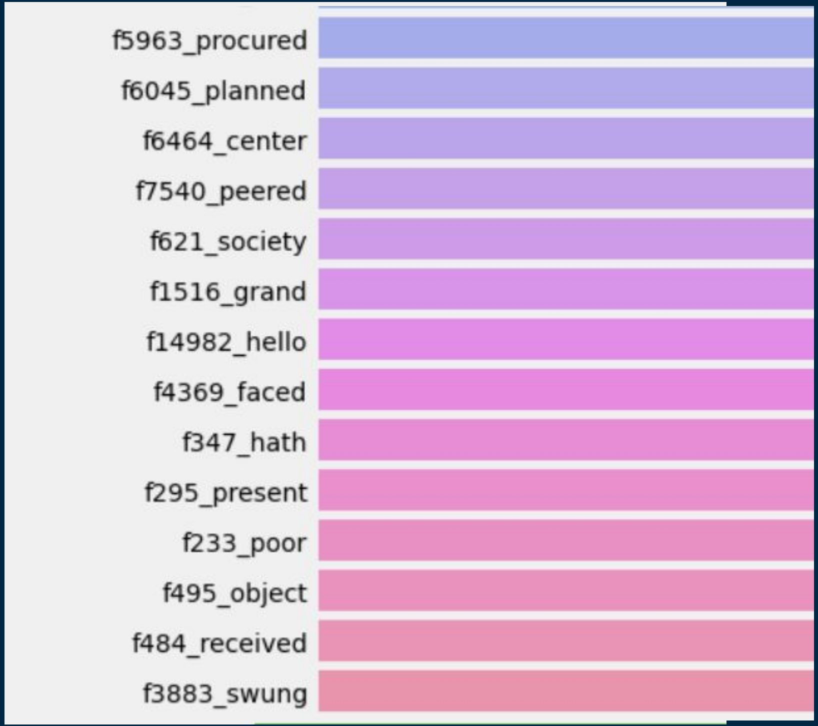
RESULTS (cont.)



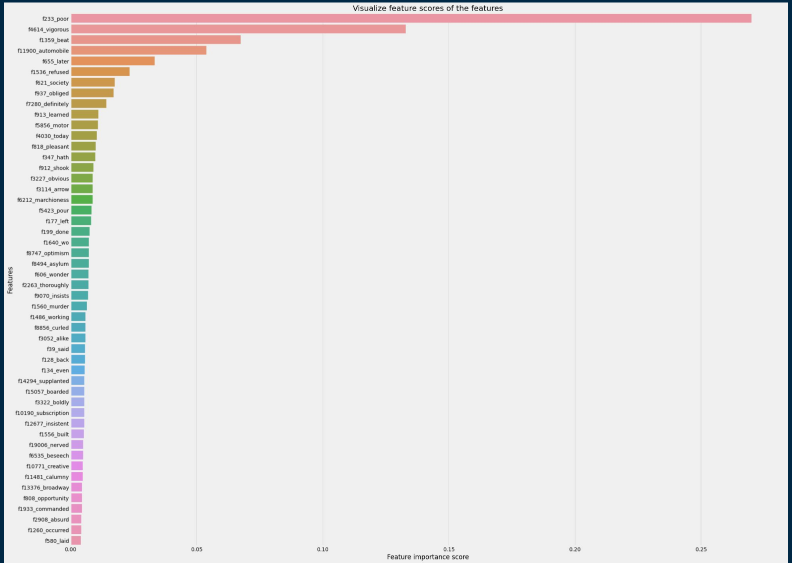
Features

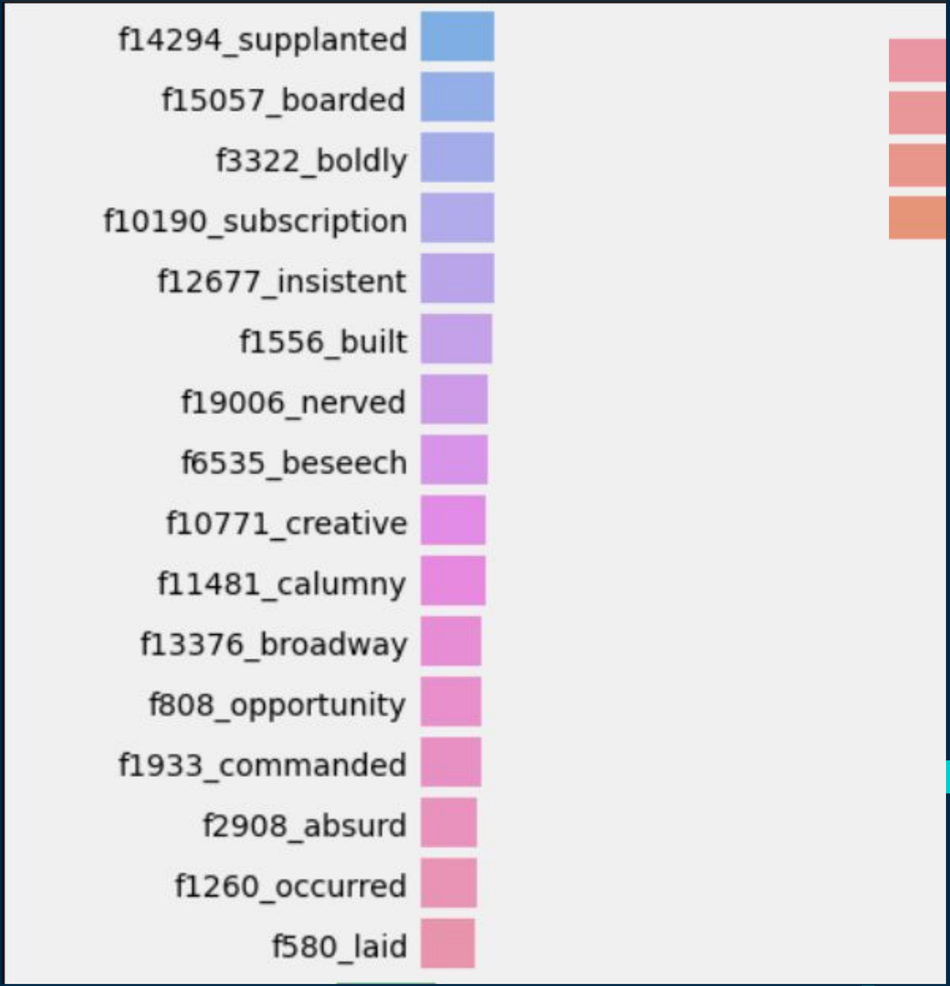
Random Forests





Features
AdaBoost





CRITIQUES, CHALLENGES & REFLECTIONS

- **Unbalanced Dataset (Classes)**
  - Date imbalance (severe class imbalance)
  - Unverifiable examples
- **Feature Standardization Approach**
  - ~100 texts arbitrary, could be missing important features
- **Boundary Classification**
  - Is there really a way to tell texts written in 1899/1900 apart?

SUMMARY & FURTHER WORK

- **Can Texts Be Dated Purely By Content?**
  - Yes!
- **Can Existing Machine Learning Models Help?**
  - Yes!
- **Are They Accurate?**
  - ~70%!
- **Could We Apply This To Date Historical Texts?**
  - A question for further applied research.

Further Work:

Feature Embedding; Sample Rebalancing; Deep Learning; Feature Semantic Analysis

REFERENCES

GitHub

1. *Project gutenberg*. Project Gutenberg. (1977). https://www.gutenberg.org/
2. Liebeskind, C., & Liebeskind, S. (2020). Deep learning for period classification of Historical Hebrew texts. *Journal of Data Mining & Digital Humanities, 2020.* https://doi.org/10.46298/jdmdh.5864
3. Gaudin, R. et al. (2014, July 6). *Openzim/Gutenberg: Scraper for downloading the entire ebooks repository of project gutenberg*. GitHub. https://github.com/openzim/gutenberg