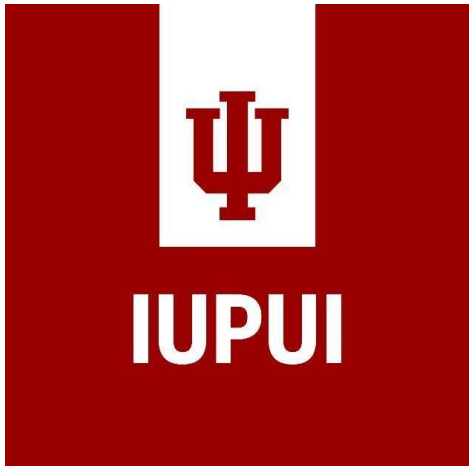


CSCI578-SML Project Presentation

Team Members (Group 11):

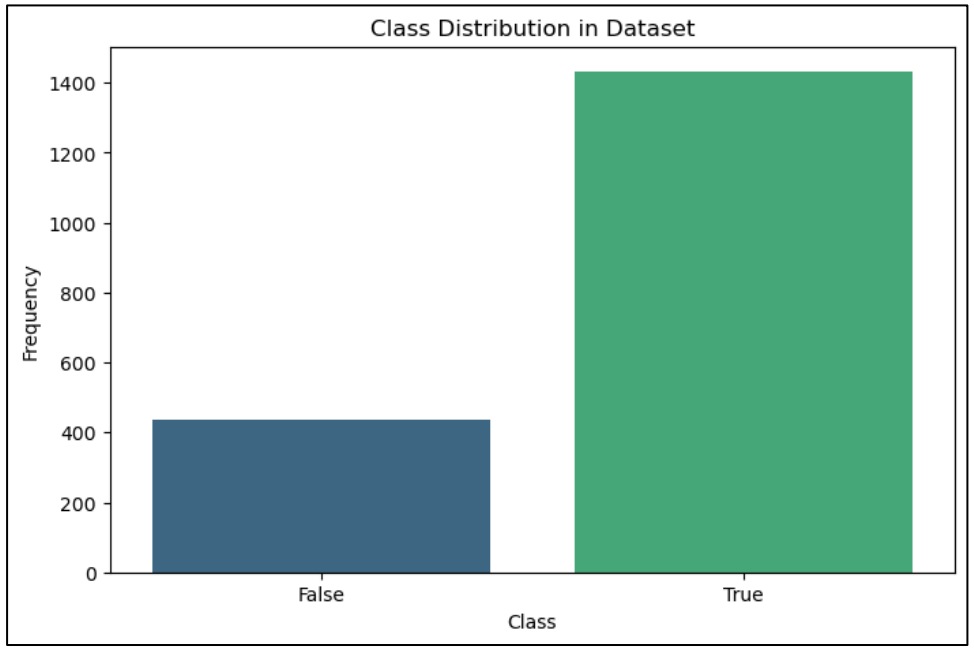
- *Ayush Manojkumar Lodha*
- *Sameer Hussain*
- *Aditya Gaitonde*
- *Ravi Teja Seera*

Dataset Description

```
df.head()
```
✓ 0.0s

	metaphorID	label_boolean	text
0	0	True	Hey , Karen !!!! I was told that on the day of...
1	2	False	Hi Ladies ... my last chemo was Feb 17/09 , ra...
2	2	False	I have just come form my consult with a lovely...
3	4	False	I also still question taking Tamox for stage 1...
4	2	False	Just checking in to say hello ladies . I had a...

First 5 samples of the provided 'train.csv' for metaphor detection



Class Distribution of Dataset which shows the imbalance

- There are **1870 data samples** (rows) in the data.
- There are 2 independent columns (X):
  - **'metaphorID':** Categorical variable representing different types of metaphors in the text. A unique integer denotes each metaphor type.
    - **metaphor_mapping** = { 0: 'road', 1: 'candle', 2: 'light', 3: 'spice', 4: 'ride', 5: 'train', 6: 'boat' }
  - **'text':** Text data column, which are sentences or paragraphs.
- There is 1 dependent column (y):
  - **'label_boolean':** Binary variable indicating whether the word is the metaphor or not.
- There are **1432 True** Samples and **438 False** Samples
- Since the data is imbalanced it is recommended that:
  - Sampling should be **stratified**. (keeping the ratio of test and train samples the same in training and validation) while the train–validation split
  - Another method used for imbalanced dataset is **SMOTE** (Synthetic Minority Over-sampling Technique)

Data Preprocessing

Tokenization Ways

TD-IDF (Term Frequency-Inverse Document Frequency)

>>

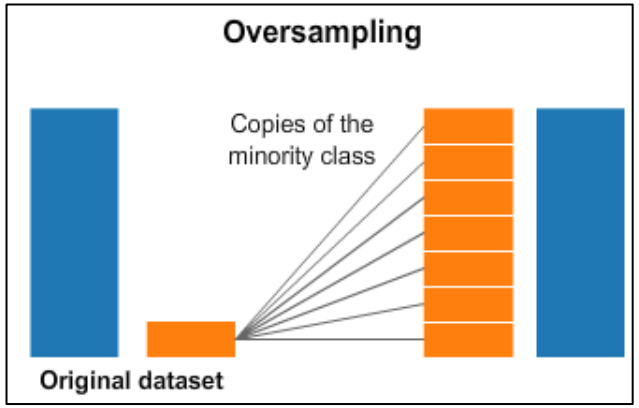
- Better approach than word2vec

Word2vec

- Relevance of word in document is taken.
- Caution!: TD-IDF should be applied separately to the train and validation set because it would violate the golden rule of machine learning.

- The meaning of the word is interpreted on basis of its usage.

Sampling of the Dataset



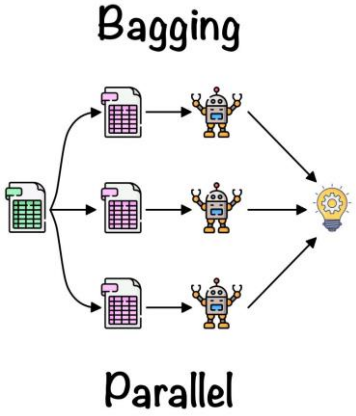

```
Original Class Distribution:
 label_boolean
1    1146
0     350
Name: count, dtype: int64
Resampled Class Distribution:
 label_boolean
1    1146
0    1146
Name: count, dtype: int64
Number of Samples Added to Each Class:
 label_boolean
1      0
0    796
Name: count, dtype: int64
```

- Since our data is imbalanced, we need to oversample our minority class of label 0.

Reference: https://medium.com/analytics-vidhya/undersampling-and-oversampling-an-old-and-a-new-approach-4f984a0e8392

Modelling

Variety of Machine Models Applied

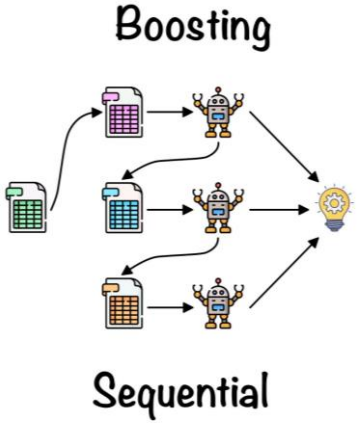


Bagging:
Combining multiple models to reduce variance.

- Random Forest
- ExtraTrees

Boosting:
Sequentially building models to reduce bias and improve accuracy.

- Gradient Boosting
- Adaboost
- XGBoost



- We implemented LSTM, but we did not explore it further because it was out of the scope of the class.

```
Epoch 1/5
36/36 [==============================] - 306s 8s/step - loss: 0.6537 - accuracy: 0.6283
Epoch 2/5
36/36 [==============================] - 308s 9s/step - loss: 0.4729 - accuracy: 0.7858
Epoch 3/5
36/36 [==============================] - 296s 8s/step - loss: 0.3724 - accuracy: 0.8534
Epoch 4/5
36/36 [==============================] - 322s 9s/step - loss: 0.3014 - accuracy: 0.8844
Epoch 5/5
36/36 [==============================] - 307s 9s/step - loss: 0.2580 - accuracy: 0.9036
12/12 [==============================] - 12s 923ms/step - loss: 0.5939 - accuracy: 0.7460
Loss: 0.593923807144165
Accuracy: 0.7459893226623535
```

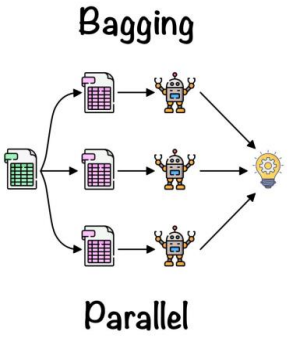
```
12/12 [==============================] - 15s 1s/step
              precision    recall  f1-score   support

     Class 0       0.46      0.48      0.47        88
     Class 1       0.84      0.83      0.83       286

    accuracy                           0.75       374
   macro avg       0.65      0.65      0.65       374
weighted avg       0.75      0.75      0.75       374
```

Reference: https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

Hyperparameter Tuning using GridSearch



Model: Random Forest

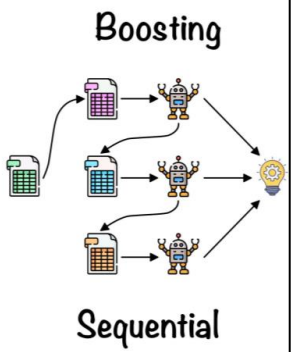
	precision	recall	f1-score	support
0	0.73	0.36	0.48	88
1	0.83	0.96	0.89	286
accuracy			0.82	374
macro avg	0.78	0.66	0.69	374
weighted avg	0.81	0.82	0.79	374

Accuracy: 0.8181818181818182

Model: Extra Trees

	precision	recall	f1-score	support
0	0.78	0.45	0.58	88
1	0.85	0.96	0.90	286
accuracy			0.84	374
macro avg	0.82	0.71	0.74	374
weighted avg	0.84	0.84	0.83	374

Accuracy: 0.8422459893048129



Model: Gradient Boosting

	precision	recall	f1-score	support
0	0.74	0.58	0.65	88
1	0.88	0.94	0.91	286
accuracy			0.85	374
macro avg	0.81	0.76	0.78	374
weighted avg	0.85	0.85	0.85	374

Accuracy: 0.8529411764705882

Model: XGBoost

	precision	recall	f1-score	support
0	0.66	0.57	0.61	88
1	0.87	0.91	0.89	286
accuracy			0.83	374
macro avg	0.77	0.74	0.75	374
weighted avg	0.82	0.83	0.82	374

Accuracy: 0.8288770053475936

Model: AdaBoost

	precision	recall	f1-score	support
0	0.49	0.39	0.43	88
1	0.82	0.87	0.85	286
accuracy			0.76	374
macro avg	0.65	0.63	0.64	374
weighted avg	0.74	0.76	0.75	374

Accuracy: 0.7593582887700535

Hyperparameter Tuning using GridSearch and Averaging

Random Forest

(n_estimators=300,
random_state=42,
class_weight='balanced')

ExtraTrees

(learning_rate=0.01, max_depth=6,
n_estimators=300, random_state=42,
use_label_encoder=False,
eval_metric='logloss')

Adaboost

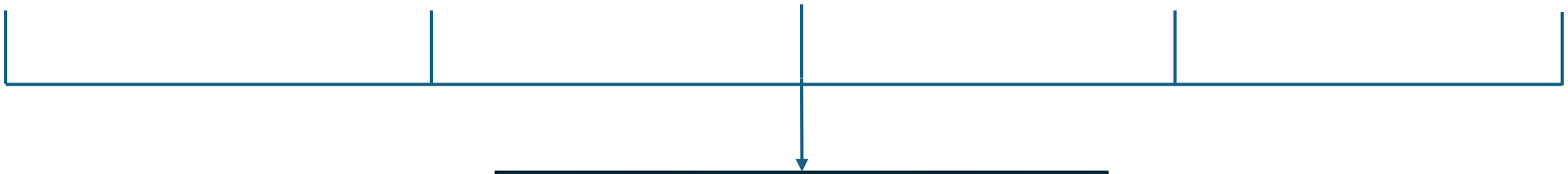
(learning_rate=0.01,
n_estimators=200,
random_state=42)

Gradient Boosting

(learning_rate=0.1,
max_depth=10,
n_estimators=200,
random_state=42)

XGBoost

(learning_rate=0.01, max_depth=6,
n_estimators=300, random_state=42,
use_label_encoder=False,
eval_metric='logloss')



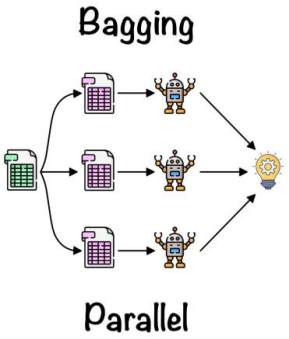
Averaging

Ensemble Model Performance:

	precision	recall	f1-score	support
0	0.74	0.52	0.61	88
1	0.87	0.94	0.90	286
accuracy			0.84	374
macro avg	0.80	0.73	0.76	374
weighted avg	0.84	0.84	0.83	374

Accuracy: 0.8449197860962567

Hyperparameter Tuning using RandomSearch and Averaging



Model: Random Forest

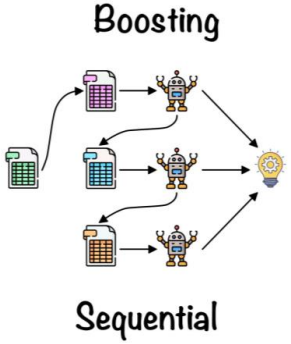
	precision	recall	f1-score	support
0	0.72	0.33	0.45	88
1	0.82	0.96	0.89	286
accuracy			0.81	374
macro avg	0.77	0.65	0.67	374
weighted avg	0.80	0.81	0.78	374

Accuracy: 0.8128342245989305

Model: Extra Trees

	precision	recall	f1-score	support
0	0.77	0.42	0.54	88
1	0.84	0.96	0.90	286
accuracy			0.83	374
macro avg	0.81	0.69	0.72	374
weighted avg	0.83	0.83	0.82	374

Accuracy: 0.8342245989304813



Model: Gradient Boosting

	precision	recall	f1-score	support
0	0.72	0.58	0.64	88
1	0.88	0.93	0.90	286
accuracy			0.85	374
macro avg	0.80	0.75	0.77	374
weighted avg	0.84	0.85	0.84	374

Accuracy: 0.8475935828877005

Model: XGBoost

	precision	recall	f1-score	support
0	0.73	0.56	0.63	88
1	0.87	0.94	0.90	286
accuracy			0.85	374
macro avg	0.80	0.75	0.77	374
weighted avg	0.84	0.85	0.84	374

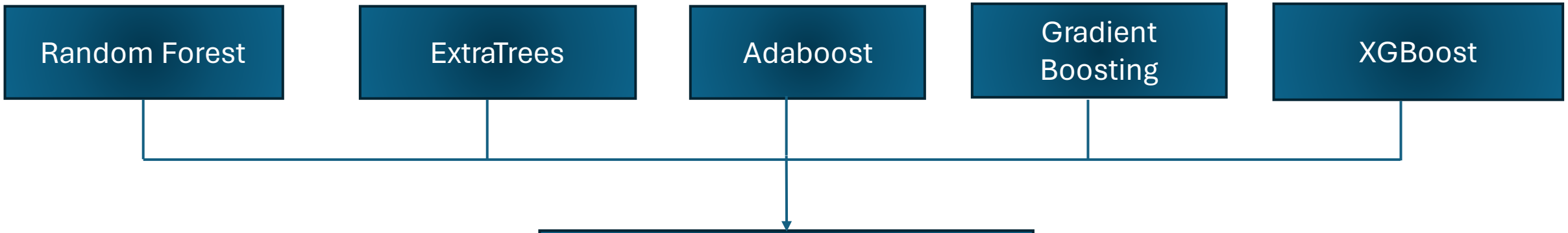
Accuracy: 0.8475935828877005

Model: AdaBoost

	precision	recall	f1-score	support
0	0.67	0.66	0.66	88
1	0.90	0.90	0.90	286
accuracy			0.84	374
macro avg	0.78	0.78	0.78	374
weighted avg	0.84	0.84	0.84	374

Accuracy: 0.8422459893048129

Hyperparameter Tuning using RandomSearch and Averaging



Ensemble Model Performance:

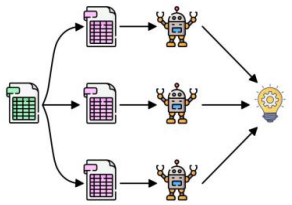
	precision	recall	f1-score	support
0	0.78	0.53	0.64	88
1	0.87	0.95	0.91	286
accuracy			0.86	374
macro avg	0.83	0.74	0.77	374
weighted avg	0.85	0.86	0.85	374

Accuracy: 0.8556149732620321

- Better Performance, but there is time constraint for RandomSearchCV.

Cross Validation (k=5)

Bagging

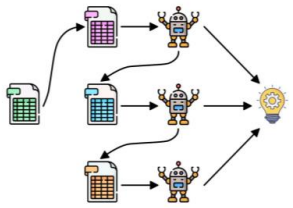


Parallel

```
Model: Random Forest
Mean Accuracy: 0.8914411940589952
Mean precision_0: 0.9418459092174694
Mean recall_0: 0.8293755025462343
Mean f1_0: 0.8651047722106664
Mean precision_1: 0.8794087043764863
Mean recall_1: 0.9532126410175191
Mean f1_1: 0.9073554411066436
```

```
Model: Extra Trees
Mean Accuracy: 0.9190117038284578
Mean precision_0: 0.9474680050959247
Mean recall_0: 0.8881118881118881
Mean f1_0: 0.9091897008400129
Mean precision_1: 0.9123817651660622
Mean recall_1: 0.9496990814063985
Mean f1_1: 0.9259692365729816
```

Boosting



Sequential

```
Model: Gradient Boosting
Mean Accuracy: 0.8952727028643259
Mean precision_0: 0.9238828955820029
Mean recall_0: 0.8615457713018688
Mean f1_0: 0.8797715806423685
Mean precision_1: 0.8952258525621115
Mean recall_1: 0.9287541726566116
Mean f1_1: 0.905241107284614
```

```
Model: XGBoost
Mean Accuracy: 0.8792150258118845
Mean precision_0: 0.9175036413564674
Mean recall_0: 0.8328890621573548
Mean f1_0: 0.8580570751452858
Mean precision_1: 0.8752045715478551
Mean recall_1: 0.9252601057479104
Mean f1_1: 0.892201330590843
```

```
Model: AdaBoost
Mean Accuracy: 0.8809559550397246
Mean precision_0: 0.918635363685517
Mean recall_0: 0.8336176019102848
Mean f1_0: 0.8636564538934218
Mean precision_1: 0.869059338828514
Mean recall_1: 0.9280597451329158
Mean f1_1: 0.8922683562459988
```

Thank you!
Any Questions