CIS 4496 INDIVIDUAL LEARNINGS NLP DISASTER TWEETS

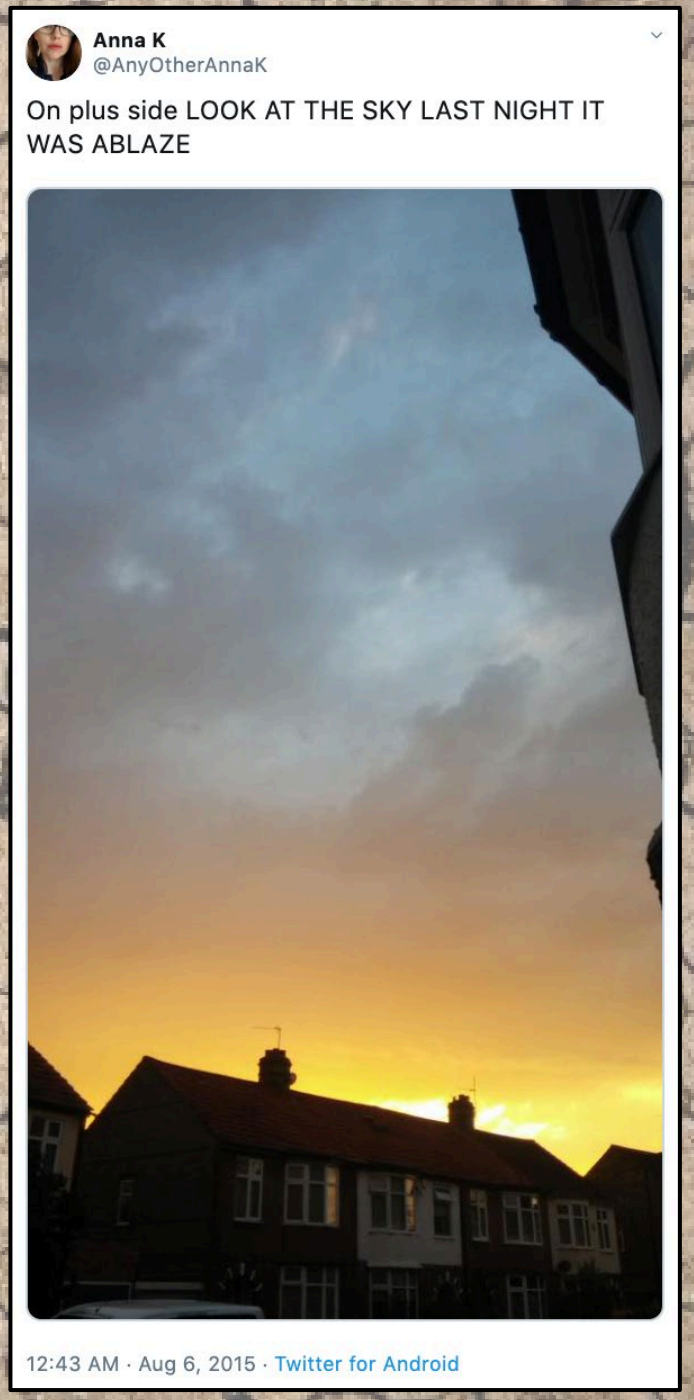
Arun Agarwal

Business Problem

- Twitter: important communication channel in times of emergency
- Ubiquitousness of smartphones → anyone to announce an emergency at any time
  - Disaster Relief Organizations and News Agencies need to programmatically monitor Twitter
- Problem: Twitter isn't all credible
  - Slang, metaphors, homonyms, source
    - A human may understand but what about a machine?

Competition Introduction

- “Getting Started” Competition
  - NLP and Preprocessing
- **Competition Goal**: build ML model to distinguish between real and fake disaster tweets
- **Personal Goal**: Learn NLP preprocessing techniques



Data

- Data gathered by Figure-Eight Website and uploaded to Appen as pre-labeled dataset
- 10,000 hand-classified tweets
- 3 files: train, test, sample_submission

`id`: identifier for each tweet

`text`: the tweet

`keyword`: keyword from tweet (many blank)

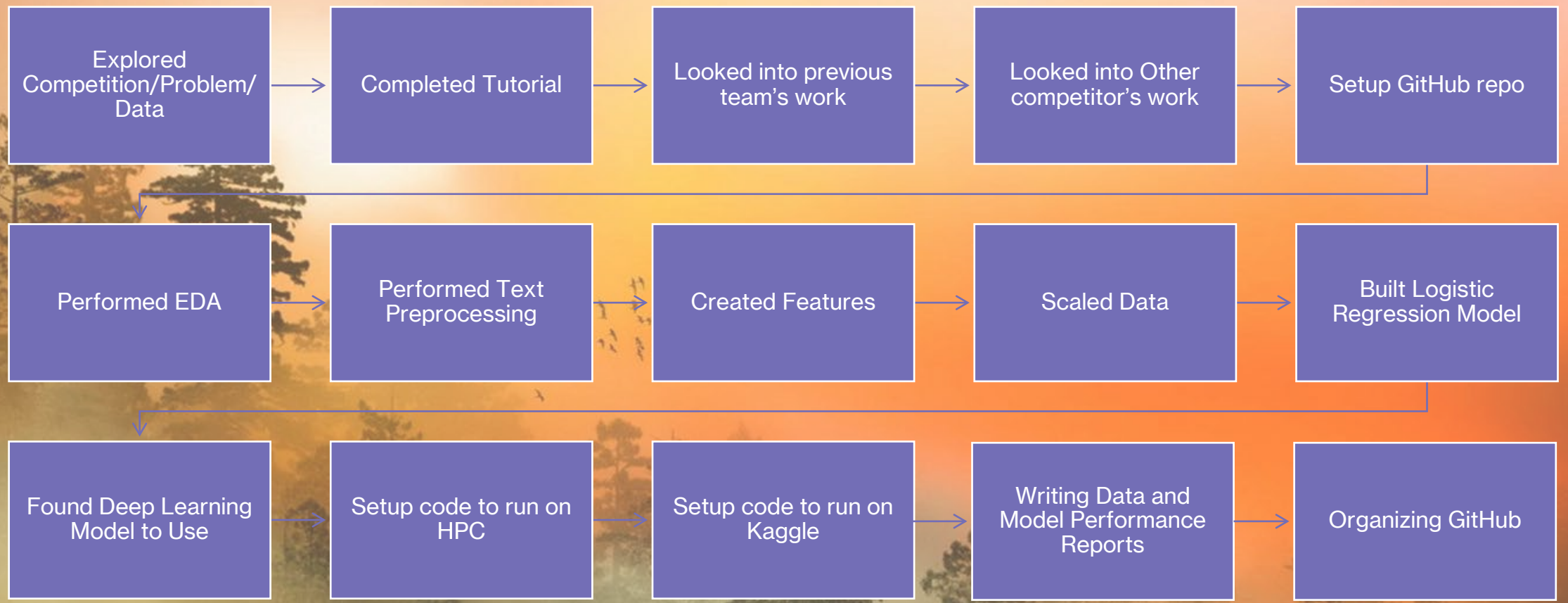
`location`: location sent from (many blank)

`target`: disaster (1) or not (0)

figure eight

appen

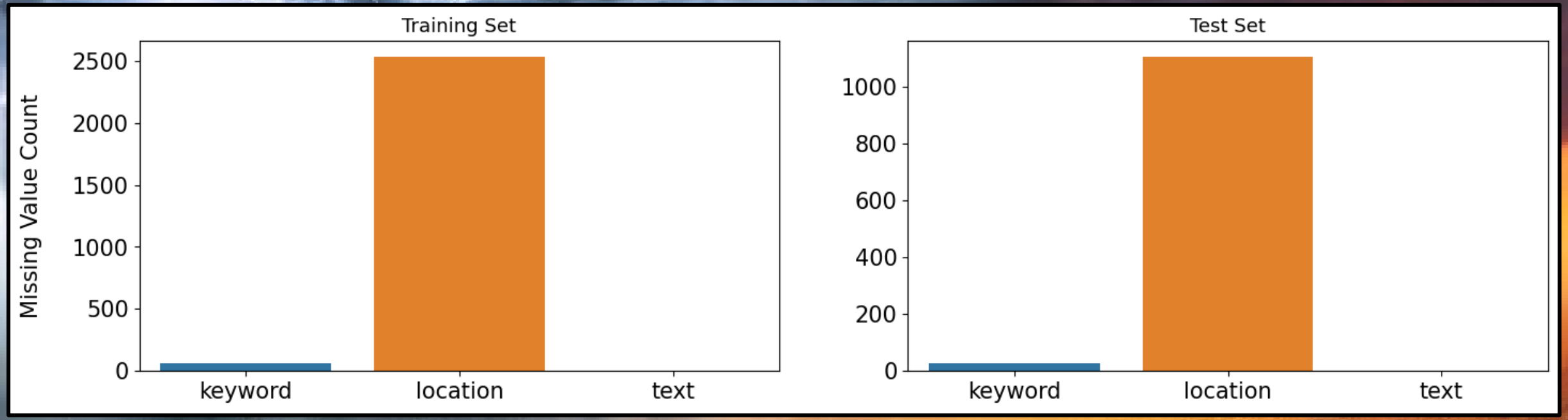
Workflow


Explored Competition/Problem/ Data
Completed Tutorial
Looked into previous team's work
Looked into Other competitor's work
Setup GitHub repo
Performed EDA
Performed Text Preprocessing
Created Features
Scaled Data
Built Logistic Regression Model
Found Deep Learning Model to Use
Setup code to run on HPC
Setup code to run on Kaggle
Writing Data and Model Performance Reports
Organizing GitHub

Preprocessing Steps

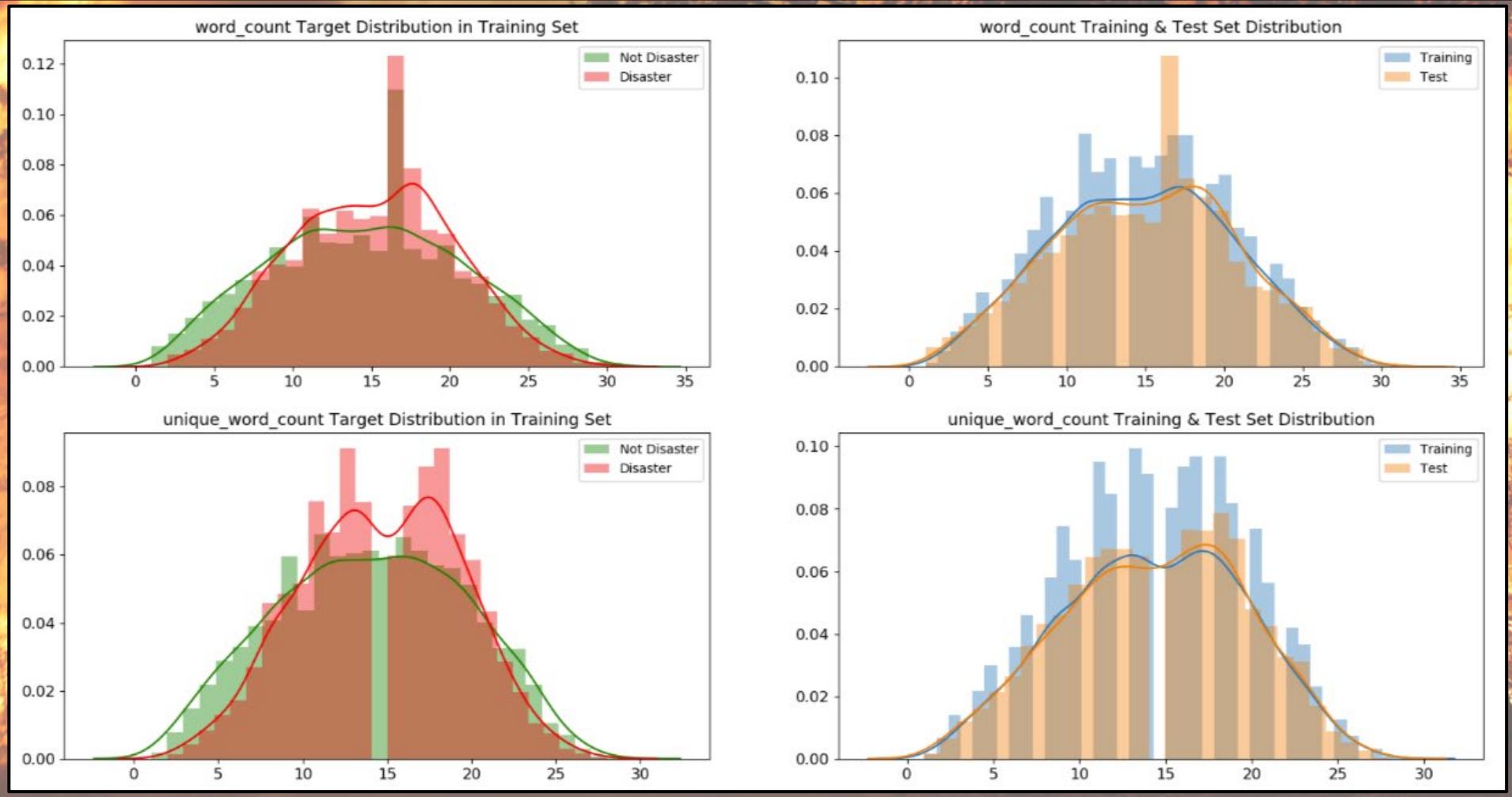
- Found list of abbreviations online
- Lowercase everything
- Remove emojis and special characters
- Remove punctuation (EDA decision)
- Expand contractions
- Remove URLs (EDA decision)
- Replace slang, informal abbreviations, and acronyms
- Manually cleaning (typos, hashtags, usernames)
- Find and manually deal with mislabeled samples
  - Remove duplicate instances
- Remove additional whitespace at end

EDA – Missing Values



Distributions of missing values for each column in training and test datasets

EDA – Target Distributions (Sample)



Distributions of two of created features, broken down by target variable and by training/test set

Baseline Model → Final Model

- **NOTE**: did not get score for test data for initial models because was running code locally while still in preprocessing phase
  - Since data came from same source, it is fine

Baseline Model (Tutorial):

- No preprocessing
- Count Vectorizer
- Linear Model (Ridge Classifier)
  - No Parameter Tuning
- 3-Fold Cross Validation

F1 Scores: 0.594, 0.565, 0.641 → **0.60**

Second Model (Logistic Regression):

- Some preprocessing
- Features Created as Input to Model
  - Instead of text
- Logistic Regression Model
  - No Parameter Tuning

F1 Score: **0.546**

Final Model (DL):

- All preprocessing
- Tokenization (FullTokenizer)
- 2-Fold StratifiedKFold
- Features Created not Inputted
- BERT Model Implementation from TensorFlow
  - L = 12 hidden layers
  - H = 768 hidden size
  - A = 12 attention heads
  - Uncased inputs

F1 Score: **0.812**

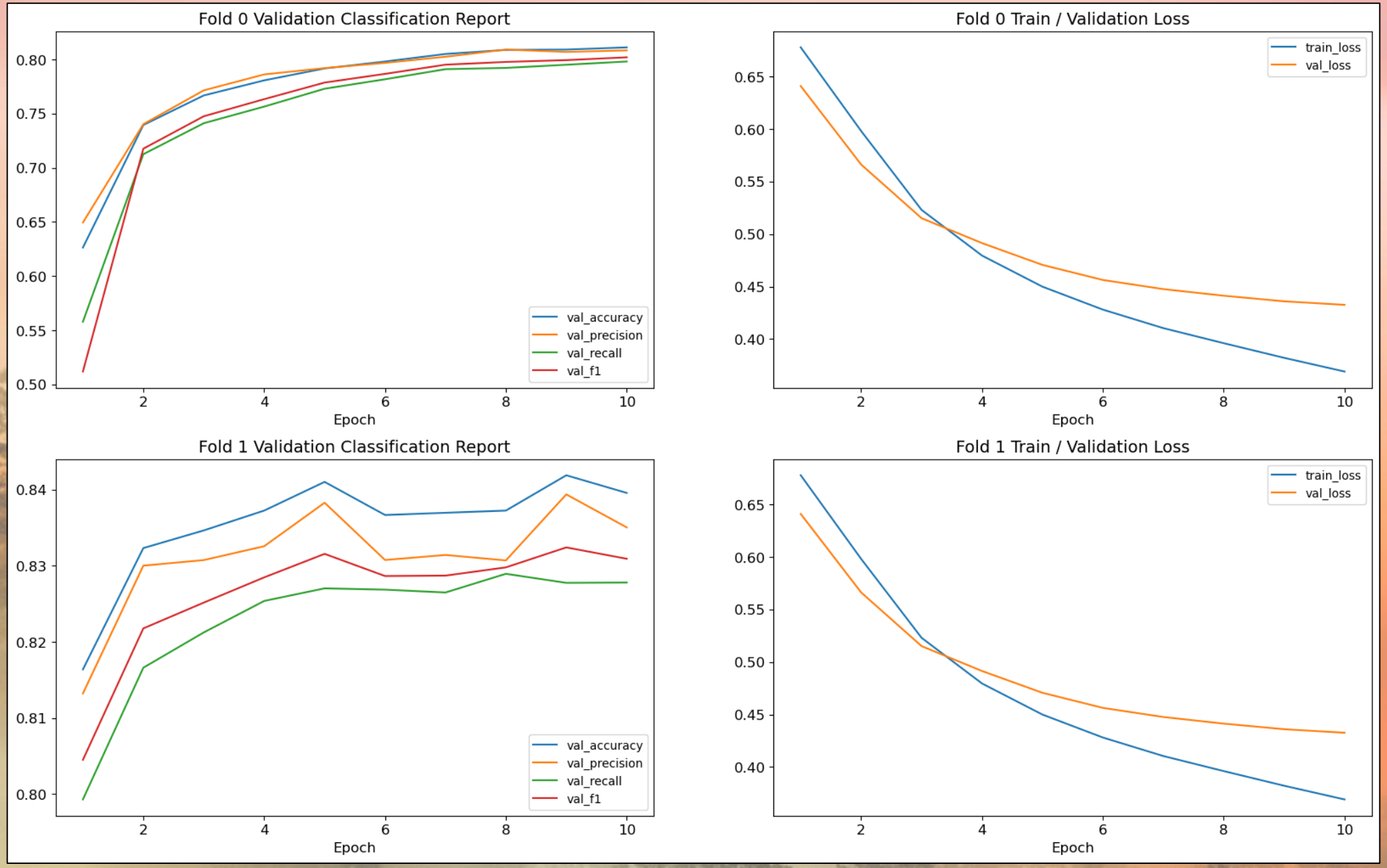
Performance Evaluation

- Performance measured using F1 Score
- Two month rolling window for submissions
- Best Scores in mid 80s
- 311th/1228 → ~25th percentile
- Test set labels publicly available, so perfect submissions on leaderboard
  - Doesn't really matter though because this competition has no prizes
- Only one leaderboard (no separation between private and public)

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$



Charts of Validation Accuracy, Precision, Recall, and F1 for 2 Folds Over 10 Epochs (Left) and Charts of Train and Validation Loss for 2 Folds Over 10 Epochs (Right)

Challenges

- Time
  - 2-3 hours/week
- Runtime Limits
  - Kaggle: TPU overflow
    - GPU worked?
  - CPU: running would have to be overnight and would sometime time-out
  - Hard to experiment with different approaches
    - Back Translation
    - Feature Creation
    - Other Dataset Versions
    - Different Deep Learning Models

Learnings

- Various ways to preprocess and think about text data
- New ideas on how EDA can help us learn the data and how to preprocess it
  - Checking missing value distributions for each column and target distributions in train and test → data from same source
  - Location column → realization that can't be auto-generated (user-input) → too unique/not helpful
  - Meta features distributions → realization that are real disasters are written with more formal language/longer words and less typos

Potential Future Steps

- Back Translation
- Input Features Created into Model
- Try Other Dataset Versions
- Try Different Deep Learning Models
- Organize GitHub and Code Further

References

- NLP Disaster Tweet Data Source
- NLP Disaster Tweet Kaggle Competition
- Gunes Evitan Kaggle Competitor Code
- Previous Group GitHub
- Transformers Modeling BERT
- How To Train a BERT Model - Towards Data Science
- GitHub NLP Tokenization Code
- My GitHub Repository