Moving on to applying models. To select the best model from all possible combinations of embedders and classifiers, we implemented stratified cross-validation. It splits our training set into four folds. Each fold will be used as a validation set to evaluate models that have been trained on the other three. The percentage of non-disaster and disaster tweets in the training set (57 and 43 respectively) also remains the same across all folds.

Now, before the Tweets are classified, they are to be vectorized by embedders. Tweets are embedded directly in Bag-of-Words and TF-IDF. The dimension of their vectors is the same as the vocabulary size. For Glove and Word2Vec embedders, a pre-trained model must be downloaded before words can be vectorized and then aggregated to encode the Tweets. Their vector dimensions are hard-coded, being 50 for Glove and 300 for Word2Vec. For Sentence BERT embedder, not only can we embed the Tweets directly with a fine-tuned model, but it is also possible to continue training it with pairs of Tweets, so that more disaster-relevant contexts can be captured in its 384-dimensional vectors.

The next step is to initialize the classifiers. For Logistic Regression we use Ridge regularization and set the inverse of regularization strength to 1. For multilayer perceptron we use relu activation, only one hidden layer with a hundred neurons, and set the learning rate to 0.001. For Random Forest we chose to build 100 Decision Trees with Bootstrapping and set Gini Impurity as their split criteria.

Now, let us look at the results. From the bar plot we can see that the highest cross-validation score is 0.975, which was delivered by the combination of Sentence-BERT and Logistic Regression. However, when we applied the chosen model to the test set, it achieves only a F0.5 Score of 0.654, indicating a huge performance drop. But when tested on Kaggle with their own labeled test set, our model achieved a F1-Score of 0.792. Despite F0.5 and F1 being different, this implies that our model is not necessarily of poor quality.

We conducted an analysis and found six problems regarding the low performance. First, there was human error. Since each of us hand classified one half of the test set, our different views on the disaster-relevance of a Tweet most likely have led to inconsistency in our test set. There was also no hyperparameter tuning for the Logistic Regression Classifier. In addition, the Sentence BERT embedder was trained only one time with inaccurate and incomplete samples. Other possible reasons include added potential indicators being insignificant, data being overfitted or Tweets being inadequately cleaned.

To solve these problems, first one may try using Random and Grid Search to optimize hyperparameters. Increasing the number of training epochs as well as preprocessing samples before training the Sentence-BERT embedder is also recommended. Other options include treating potential indicators as hyperparameters during fine-tuning, applying stronger regularization and performing stricter data cleaning.

That concludes our presentation. Thank you for listening. We welcome any questions.