DT2D

- Collected data and documented the sources
- Created SVM and NB classifiers over the BBC dataset and optimized them by segmenting the text, removing duplicates, Tfidf, Bigrams and using the optimal hyperparameters,
- Created a neural network for the Parts of Speech Data
- Requested access to the PACE clusters.

NB experiments:

- Number of features: Tried reducing the features to 500, 2500 from 5000 but consistently got reduced performance. Increased it to 10000 but not a significant change in performance. So kept it at 5000. 10000 gave me slightly better accuracy ~ 1 % change. But since it consumes much more resources on my laptop, I left it at 5000.
- Training data size: Changed training data samples per class from 100, 500, 1000, 5000, and all samples per class. The metrics changed and are shown in the graph. TfIDF does not perform well consistently on all datasets. Bigrams yield the best results consistently.
- Removed duplicates in each sample and trained the model on different training sizes. The overall accuracy without duplicates was less than the original data. But the trend within the dedup data over different training sizes was similar to the original data over different training sizes. Need to generate graphs. But is it necessary? since the performance without duplicates was lower.
- Performed Grid Search cross-validation to find optimal parameters for the classifier. The optimal parameters were different for different training sizes and different feature extraction methods.
- Tfidf with Bigrams or Unigrams performs as good as just using Bigrams or Unigrams.

- Using unigrams and bigrams both in feature extraction works identical to just using Bigrams.
- Also played around with trigrams but no improvement in performance.

SVM experiments

- Training data size: Changed training data samples per class from 100, 500, 1000, 5000, and all samples per class. The metrics changed and are shown in the graph. TfIDF performed consistently well on all datasets.
- Performed Grid Search cross-validation to find optimal parameters for the classifier. The optimal parameters were different for different training sizes and different feature extraction methods.

How were the optimal parameters chosen?

- In ML, the generalization curve generally follow a U shaped curve where simple hyper parameters that are not able to capture all patterns in the training data lead to high generalization error (underfitting) and complex hyper parameters lead to overfitting the training data causing high generalization error. Hence, the generalization error is high on both ends of the U shaped curve because of either underfitting or overfitting. The goal was to find a point in between that results in the best accuracy. For SVM, I initially formed a range for each hyperparameter. If the optimal hyperparameter was in the middle of that range with the end values giving high generalization error, I can conclude the the middle value of the range yields the minima on the generalization curve. If the optimal value ended up being on one of the ends of the range, I would readjust the range until I find the optimal hyperparameters that lie in the middle of the range. This ensures that the hyperparameters I selected are indeed optimal.