Lexical Complexity Prediction

Subtask ID: Task 1

Group Details:

Name	Roll Number
Amrit Sahu	17EC10066
Ravi Ghadia	17EC10045
Shubham Maheshwari	17EC10055
Shivam Saxena	17EC10054
Ritik Kumar	17CS10044

Group ID: cs60075_team8
Codalab ID: IITKGP_CS60075_TEAM_8

1 Individual Contributions

Amrit Sahu: Came up with the idea of Bert-based ordinal classification. Code for the initial pipelining and ordinal classification experiments. Wrote some part of the architecture and experiment section of the report. Suggested some of the features like Relative Frequency, lemmatized length, hyponym and hypernym similarity.

Ravi Ghadia: Worked with Tree based ensemble regressors, and came up with the idea of a separate model of handcrafted features and coded the same. Explored the fact that word as a single entity without surrounding context may also be significant for predicting complexity. Also coded the final ensemble of both the models.

Shubham Maheshwari: Conducted dataset and model prediction analysis which brought out the fact that only regression is not suitable for the task. Experimented with classification approach using many of the hand-crafted features (dependency parsing features) and coded the same.

Shivam Saxena: Ran a few of the experiments considering some of the architectures mentioned in the experiment section. Was responsible for reading related literature and providing insights on the task. Contributed to report and presentation preparation.

Ritik Kumar: Experimented with PCA, analyzed correlation between features. Implemented many of the handmade features using relevant python packages. Gathered relevant corpus and trained word2vec. Contributed to report and presentation preparation.

2 Approach / Model Architectures

The complexity of a given word is useful in many readability applications. Accurately predicting lexical complexity will help a system better direct a user to the right text or customize a text to their needs.

Some of the crucial insights regarding the problem are stated below:

- Words as a single entity may vary in complexity, irrespective of content words surrounding it. For example, sermonise seems to be complicated to casual readers due to its infrequent use in everyday speech.
- We realised that the same tokens can have different definitions depending on the context, which may make it hard for the reader. As a result, we also look to include context in our complexity analysis.

As a result, we attempt to represent each word as a vector that contains both context and stand-alone complexity factors, and then process it using NNs, ensemble of tree-based models, treating the case as a regression problem later combining it with ordinal classification.

Tree-Based Ensemble Regressor

A Tree-based ensemble regression model to be trained on hand-made feature vectors for the target token. The description for the hand-made feature vector is given in the following section. The motive behind this is that the complexity of the target word can be inferred up to a considerable extent by just considering its structural information like its length, syllable count, frequency of occurrence in some reference corpus etc. Such information has proven to be highly correlated with the complexity of the word (David Alfter et. al, 2018) and thus has been taken into consideration.

Bert-based Ordinal Classifier

An Ordinal Classifier was trained using Bert word embeddings which capture the contextual information from the surrounding word of the target token. We have chosen the standard way of taking hidden states from the last 4 hidden layers of the Bert as the final embedding for the target token. Accounting to the fact that Bert uses a

WordPiece tokenizer, the target token may be broken down to smaller sub tokens by the Bert Tokenizer and thus it might not be present in its original form when being given as input to the Bert. To handle this, we took the average embedding of all the sub-tokens of the target token as its final Bert Embedding.

Feature Vector for Regressor

The hand-made feature vector for each target token has been constructed taking into account several structural, syntactic as well statistical aspects of the target token. We trained a Word2Vec model over the combined corpus from where the dataset has been derived i.e., Bible WEB corpus, Europarl Corpus, and CRAFT corpus. We then concatenated our hand-made features with the Word2Vec vector of the target token to get the final feature vector which was fed to the Regressor. Following hand-made features have been used:

- **Word Frequency**: The frequency of occurrence of the target token in the Google Big-Query corpus from the list of words occurring at least once every 100M words.
- **Relative Frequency**: The frequency of the target token as compared to the frequencies of its synonyms in the Google Big-Query corpus. Exponential Scaling was used to give a non-linear weightage to the targets.
- **Word Length**: Length of the target token.
- **Lemmatized Length**: It is the normalised difference between the lengths of the original token and its lemmatized form. The acquisition of derivational morphological information occurs progressively during the learning process, which motivated us to use this feature in our experiments.
- **Hypernym Similarity**: The average Wu-Palmer similarity of the target tokens with its hypernyms. This captures the info about how specific the target is.
- **Hyponym Similarity**: The average Wu-Palmer similarity of the target tokens with its hyponyms. This captures the info about how general the target is.
- **Num_of_Hypernyms**: Number of hypernyms of the target token, found using wordnet.
- **Num_of_Hyponyms:** Number of hyponyms of the target token, found using wordnet.
- **Num_of_dependencies**: Number of children in the dependency parse tree of the token found using SpaCy.
- **Num_of_synonyms:** Number of synonyms of the target token.
- **Avg_child_synonyms:** Average number of synonyms of each child of the target in its dependency parse tree.

We have used a dual model approach to model both contextual and non-contextual information about the target token to predict its lexical complexity. The final model used was an ensemble of Bert based ordinal classification (contextual) and hand-crafted feature based tree regressor (non-contextual).

Why Ordinal Classification?

The motivation of treating the problem as an ordinal classification problem came from the observation that the final complexity score provided in the CompLex dataset is derived from a classification task where annotaters were asked to label a sentence in one of the following categories of Very Easy (1), Easy (2), Neutral (3), Difficult (4), Very Difficult (5). Then the score from 1 to 5 for each annotator was scaled down to 0 to 1 by appropriate mapping. The final score obtained was the average of all the annotators. Hence the final scores don't completely span the real number space from 0 to 1 but are distributed in chunks spanning small areas as indicated in **Figure 2.**

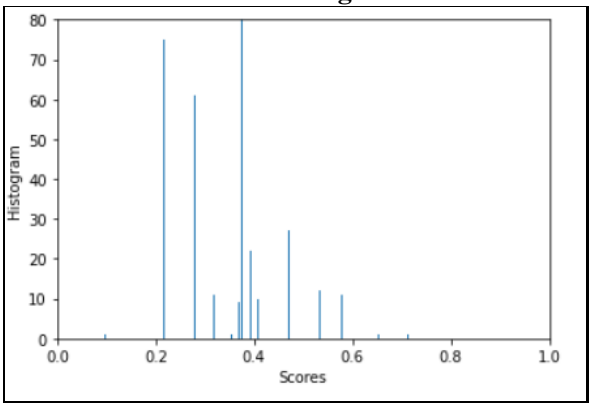


Figure 1. Histogram of complexity values

Ordinal classification considers ordering of classes, hence preferred over normal classification for our purpose. In ordinal classification we decided to break the space of real line 0 to 1 into 10 bins representing 10 classes. The complexity value range are as follows:
Class 0: [0,1/18), Class 1: [1/18,3/18) and so on until Class 9: [17/18,1).

While training each class is represented by special one-hot type vectors (of size number of bins-1) as shown below:

Class 0: [0,0,0,0,0,0,0,0,0]
Class 1: [1,0,0,0,0,0,0,0,0]
Class 2: [1,1,0,0,0,0,0,0,0]
Class 9: [1,1,1,1,1,1,1,1,1]

The model for ordinal classification is a simple multi-layer perceptron neural network which takes in BERT embeddings as input and has output dimension of number of bins -1, each dimension representing the probability of a particular dimension of the one hot type vector. The model is trained using binary cross entropy loss.

To map the output of the neural network (i.e., a vector of probabilities for each dimension of one hot type vector being triggered, there is no softmax layer but each of the probability is an output of sigmoid layer) to a complexity score we simply add the values of the output vector and divide it by (number of bins -1). Also, a threshold parameter was used during prediction to only consider the classes with probabilities above the threshold. This led to removal of less confident values and further improved score on dev-set.

For example if the model thinks the complexity falls in range of class 2 then a probable output vector generated is:
[0.9,0.99,0.01,0.01,0.01,0,0,0,0], which after thresholding becomes [0.9,0.99,0,0,0,0,0,0,0]
which will be mapped to: (0.9+0.99)/9 = 0.21, which is close to 4/18 which represents the centre of the bin of Class 2 [3/18,5/18] with slight displacement from center.

Ordinal Classification proved to be practically doing better than regression because it could predict very low complexity values as well due to its nature of prediction of a particular bin first along with a slight displacement from centre of the bin to give a real valued complexity score. The nature of regression lets it to predict more of average values to minimise mse, so it could never predict scores below 0.2 and was predicting scores in a range of 0.2-0.5.This flaw of regression can be understood from a thought experiment of trying to fit a straight line to samples derived from a step function.

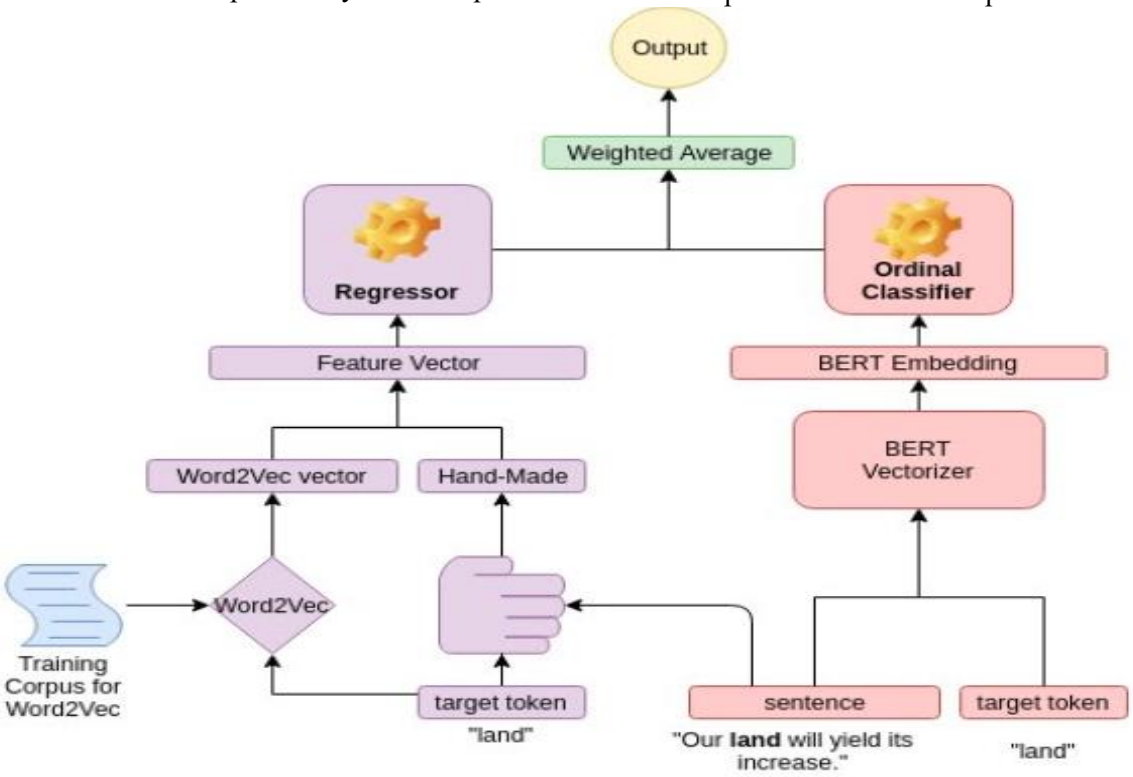


Figure 2. The dual Model Architecture

4 Experiments

For the model architecture mentioned in the last section, a number of combinations were tried to determine the best combination based on the Pearson correlation score for the development (trial) set. To verify the importance of the hand-crafted features as well as Word2Vec vector, experiments were conducted to see their effect on including them in the feature vector. Next we observe the performance of the ordinal classifier as compared to MLP regressor/classifier. We also conducted experiments to determine the regressor model for modelling the non-contextual vector. Apart from these, the hyperparameters were also finetuned on the development set, such as the weights of the final ensemble, threshold of the ordinal classifier, learning rate of MLP, number of bins for the ordinal classifier etc.

Model	Feature Vector Description	Pearson Corr. Coeff on Multi Trial Data	Pearson Corr. Coeff on Single Trial Data
MLP Regressor	Only BERT-embedding (last 4 average)	0.7011	0.6972
MLP Regressor	Only BERT-embedding (last 4 concat)	0.7241	0.712
MLP Regressor	BERT-embedding + Hand-crafted features	**0.7389**	**0.737**
MLP Classifier	BERT-embedding + Hand-crafted features	0.6412	0.6714
MLP Ordinal Classifier	BERT-embedding + Hand-crafted features	**0.7454**	**0.7433**
XGBoost Regressor	Hand-crafted features	-	0.7132
AdaBoost Regressor	Hand-crafted features	-	0.6935
GradientBoosting Regressor	Hand-crafted features	-	0.7436
XGBoost Regressor	Word2Vec + Hand-crafted features	0.7419	0.7275
AdaBoost Regressor	Word2Vec + Hand-crafted features	0.7174	0.6985
GradientBoosting Regressor	Word2Vec + Hand-crafted features	0.7501	**0.7678**
Stacking Regressor (XGBoost, AdaBoost, GradientBoosting)	Word2Vec + Hand-crafted features	**0.7587**	0.756
Ensemble of Stacking Regressor and MLP-Ordinal Classifier	BERT-Embedding + Word2Vec + Hand-crafted features	**0.7867**	0.7603
Ensemble of GradientBoosting Regressor and MLP-Ordinal Classifier	BERT-Embedding + Word2Vec + Hand-crafted features	0.7524	**0.796**

Table 1. Pearson's correlation scores for both sub tasks for different configurations, evaluated over the trial dataset.

5 Results and Discussions

The results of the experiments are present in Table 1. From the Pearson's correlation scores, we infer that inclusion of hand-crafted feature vector improves the score as compared to only Bert-Embeddings. Further, including Word2Vec with the hand-crafted feature vector increases the score. For the single-word prediction task, Gradient Boosting Regressor seemed to work better while for the multi-word, Stacking Regressor combining XGBoost, AdaBoost and Gradient Boosting seemed to perform better. Finally, the best overall performance was found using a weighted ensemble of the contextual and non-contextual model outputs. These results clearly show that both contextual and non-contextual features are strong contingents in determining the complexity of a word/phrase.

Further, work could be done in finetuning the hyperparameters, using threshold based training for the ordinal classifier, adding more features to the hand-crafted feature vector etc.

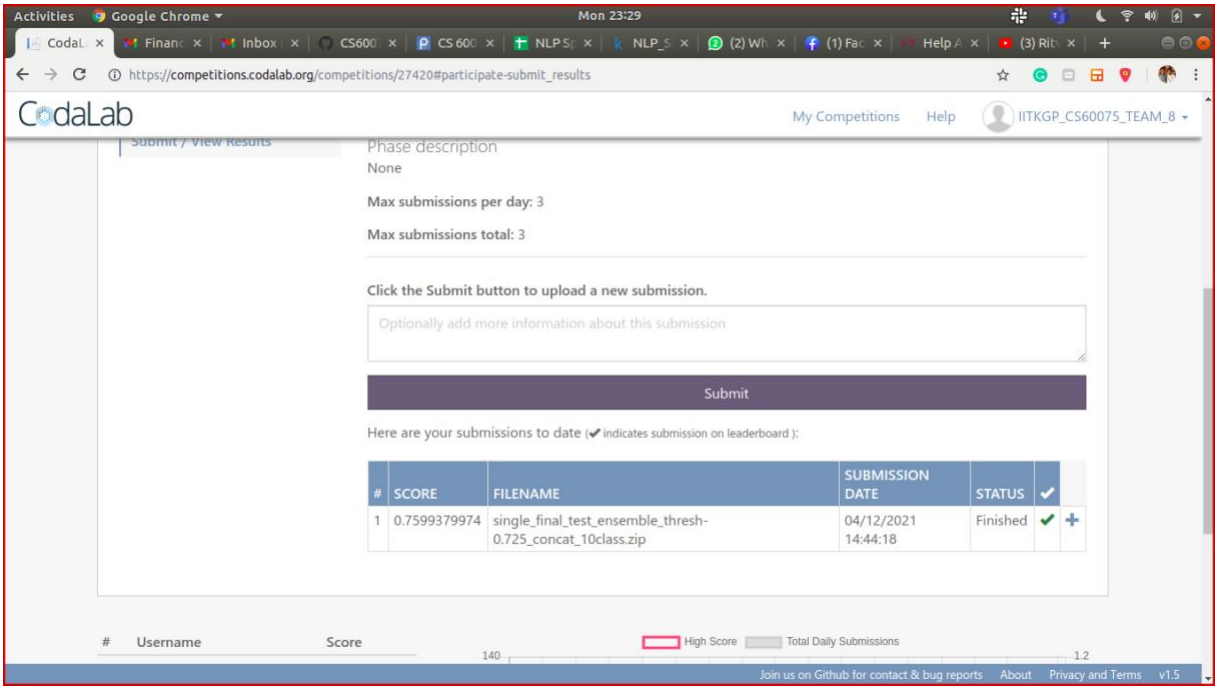
The link to the team's private GitHub repository:
https://github.com/shubham1280/CS60075-Team-8-Task-1

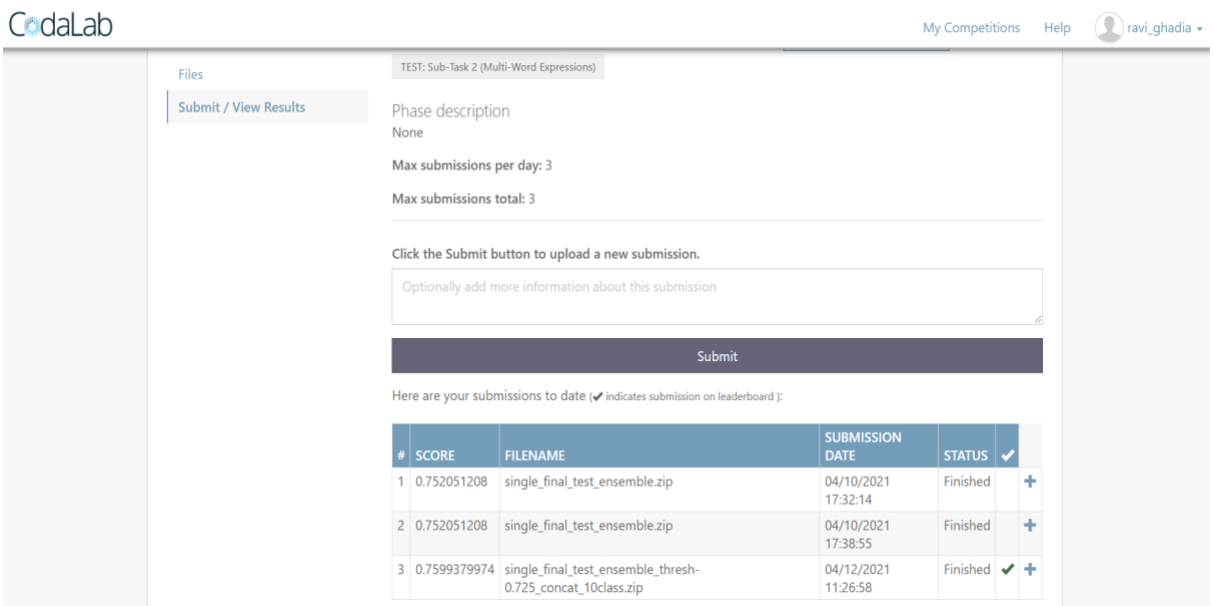
References

- Word2Vec Bible Corpus: World English Bible, https://www.kaggle.com/oswinrh/bible?select=t_web.csv
- Word2Vec Europarl English Corpus Small Version: https://www.kaggle.com/nltkdata/europarl
- Word2Vec Biomedical CRAFT Corpus: http://bionlp-corpora.sourceforge.net/CRAFT/
- CompLex Dataset M. Shardlow et.al, https://arxiv.org/abs/2003.07008
- Towards Single Word Lexical Complexity Prediction www.aclweb.org/anthology/W18-0508/
- BERT:- https://huggingface.co/transformers/model_doc/bert.html
- Word2Vec:- https://radimrehurek.com/gensim/models/word2vec.html
- Wordnet Word Sense Disambiguation: http://www.nltk.org/howto/wsd.html
- Dependency Parsing using SpaCy: https://spacy.io/usage/linguistic-features

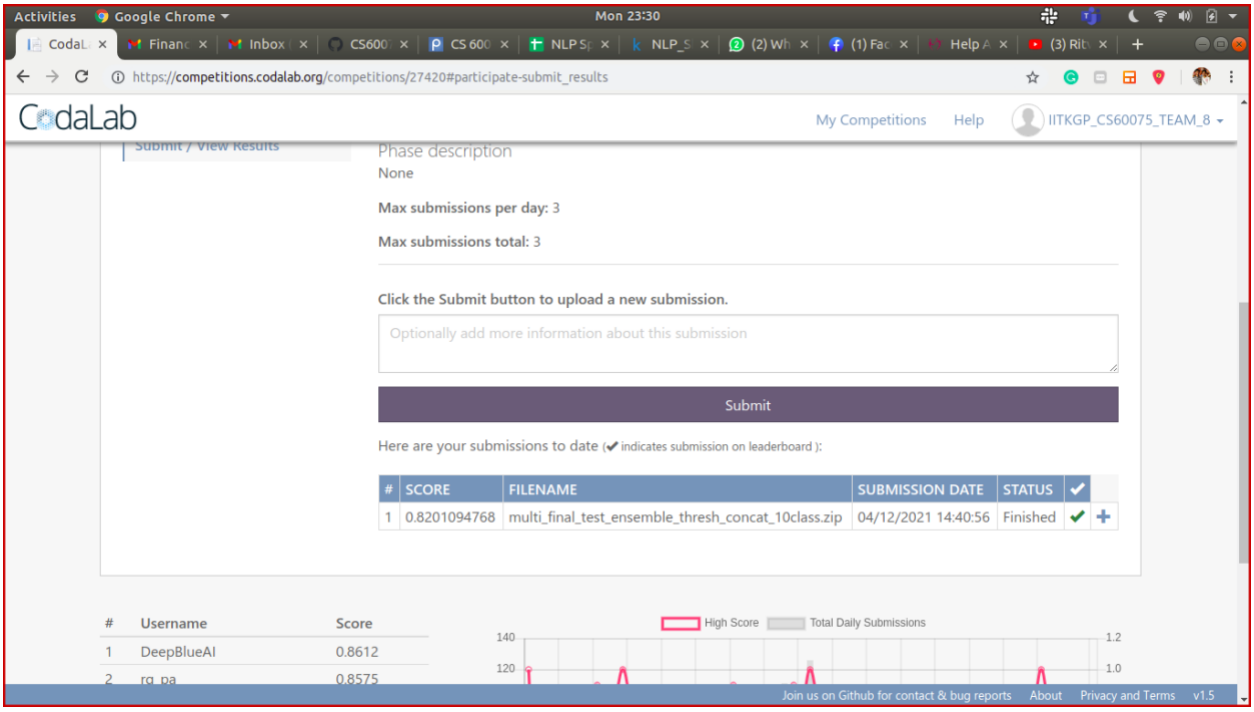
<u>NOTE:</u> FINAL SUBMISSIONS SHOULD ONLY BE CONSIDERED FROM THE USERNAME - IITKGP_CS60075_TEAM_8 (The username ravi_ghadia is to show our past experimentations since the test limit exceeded in that account, hence a new account was created with the username IITKGP_CS60075_TEAM_8.)

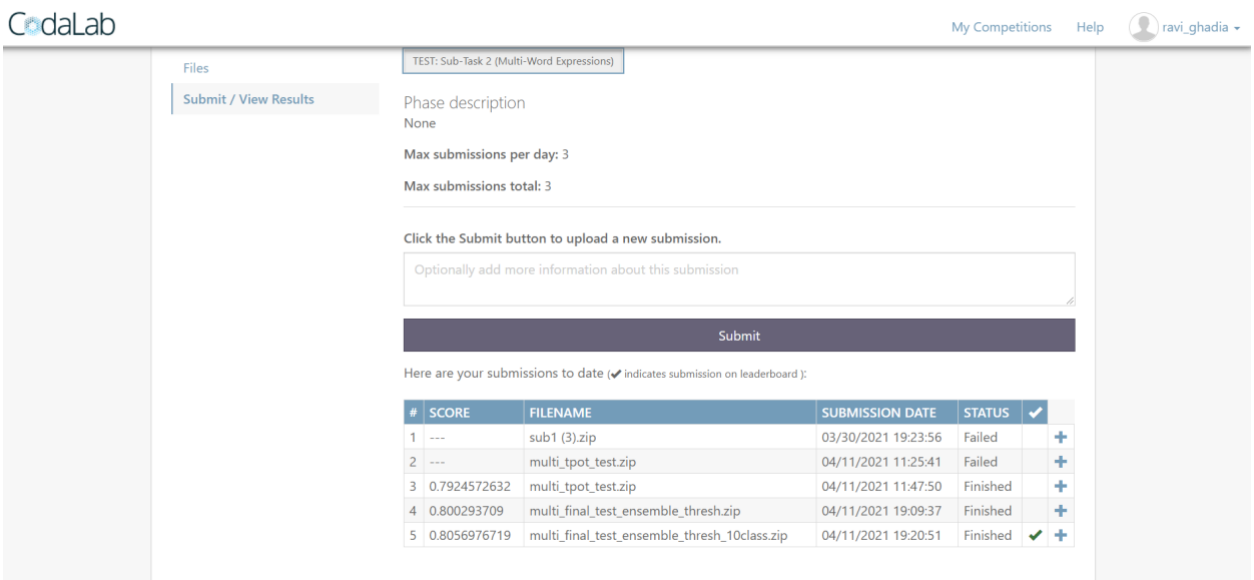
<u>Screenshot of Single Test</u>



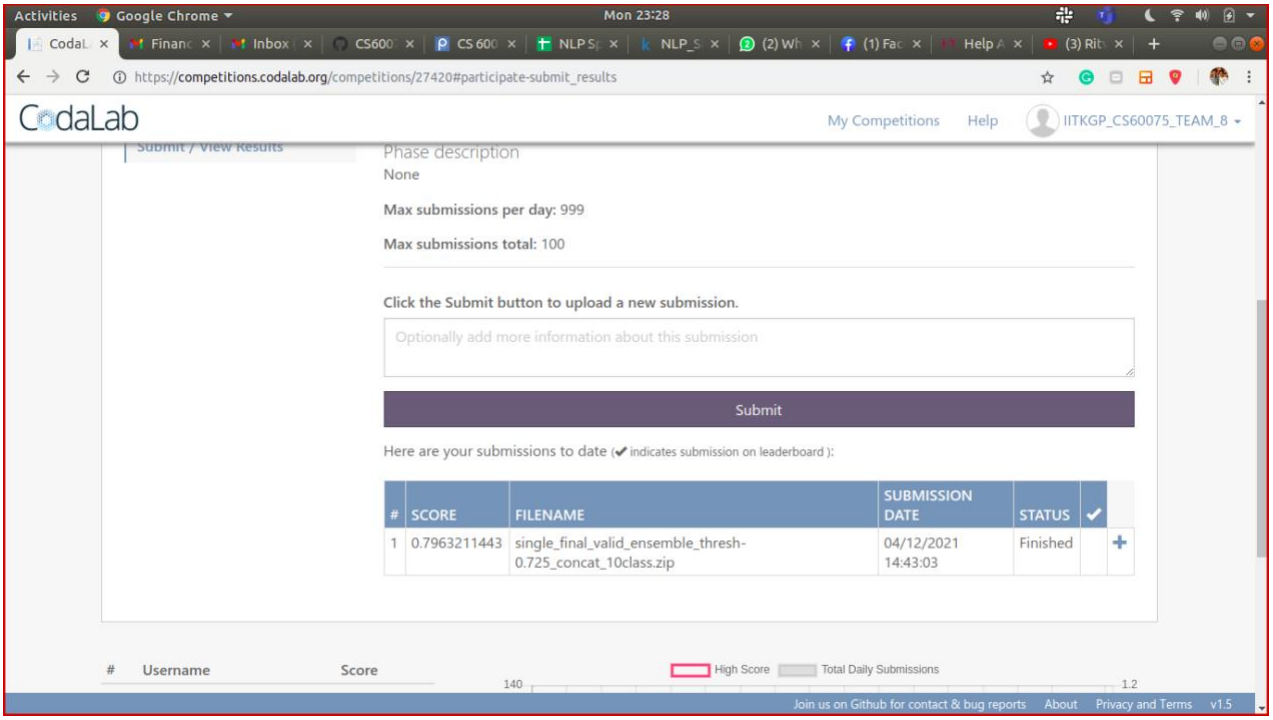


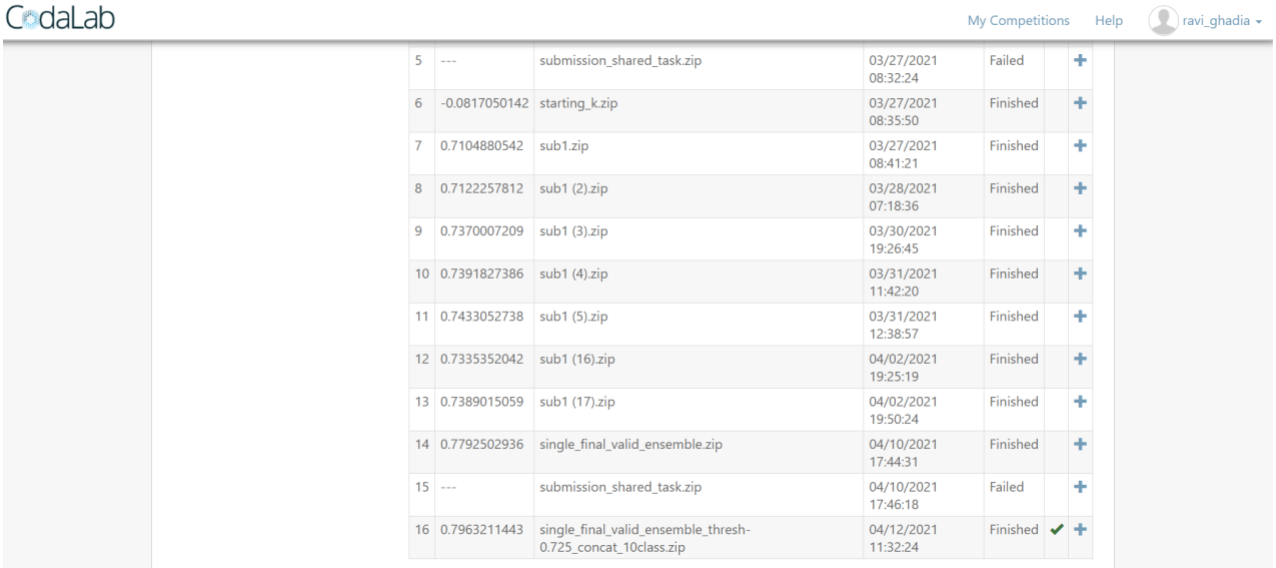
Screenshot of Multi Test





Screenshot of Single Trial





5	---	submission_shared_task.zip	03/27/2021 08:32:24	Failed	
6	-0.0817050142	starting_k.zip	03/27/2021 08:35:50	Finished	
7	0.7104880542	sub1.zip	03/27/2021 08:41:21	Finished	
8	0.7122257812	sub1 (2).zip	03/28/2021 07:18:36	Finished	
9	0.7370007209	sub1 (3).zip	03/30/2021 19:26:45	Finished	
10	0.7391827386	sub1 (4).zip	03/31/2021 11:42:20	Finished	
11	0.7433052738	sub1 (5).zip	03/31/2021 12:38:57	Finished	
12	0.7335352042	sub1 (16).zip	04/02/2021 19:25:19	Finished	
13	0.7389015059	sub1 (17).zip	04/02/2021 19:50:24	Finished	
14	0.7792502936	single_final_valid_ensemble.zip	04/10/2021 17:44:31	Finished	
15	---	submission_shared_task.zip	04/10/2021 17:46:18	Failed	
16	0.7963211443	single_final_valid_ensemble_thresh-0.725_concat_10class.zip	04/12/2021 11:32:24	Finished	✔

Screenshot of Multi Trial