## Team members and contributions

1. Anurag Anand

* Performed Exploratory Data Analysis
* Data Cleaning
* Created single word, bigram, and trigram vectors
* Developed benchmark logistic regression model and XGBoost models for sentiment classification

1. Sandip Sonawane

* Performed Exploratory Data Analysis
* Data Cleaning
* Created tf-idf and word2vec models
* Created myvocab list and Interpreted the algorithm

## 2. Introduction

We are provided with a dataset consisting of 50,000 IMDB movie reviews, where each review is labeled as positive or negative. The goal is to build a binary classification model to predict the sentiment of a movie review. We built least squares, tree-based classification models (Gradient Boosted Trees and Random Forest Models) to generate classifications.

3. Data Source

The data is taken from [Kaggle](https://www.kaggle.com/c/word2vec-nlp-tutorial/data). The labeled data set consists of 50K IMDB movies, specially selected for sentiment analysis. The sentiment of reviews is binary, meaning the IMDB rating < 5 results in a sentiment score of 0, and rating >=7 has a sentiment score of 1. No individual movie has more than 30 reviews. The 25,000 reviews labeled training set does not include any of the same movies as the 25,000 review test set. In addition, there are another 50,000 IMDB reviews provided without any rating labels.The description of each variable can be found at this [link](https://www.kaggle.com/c/word2vec-nlp-tutorial/data).

We have the below files to build and train models.

* **train.tsv(fold 1 to 5):** 3 columns (“id”, “sentiment”, “review”)
* **test.tsv(fold 1 to 5):** 2 columns (“id”, “review”), in the same format as the train.csv file on Kaggle
* **test\_y.tsv(fold 1 to 5):** 3 columns (“id”, “sentiment,“score”). The score column was not available during the training time so it was not used during the model development process

## 4. Evaluation metric and performance target

The goal was to build a binary classification model to predict the sentiment of a review with a vocabulary size less than or equal to 1000. We have used the same vocabulary for all five training/test datasets.

The evaluation metric is AUC on the test data. Our target was to produce AUC equal to or bigger than 0.96 overall for all the five folds

## 5. Data Preparation

For this project, we essentially have only one dependent variable i.e. review. Since the review column is raw text, we undertook specific transformations that gave us tokens, based on which we predicted sentiments. We undertook the following steps as part of the data preparation process:

1. Loaded the data and cleared the HTML tags
2. Removed stop words, converted all text to lowercase, and tokenized the text
3. Created a document term matrix using ngrams (maximum 4-grams)
4. Pruned the underlying vocabulary in the Document Term Matrix

## 6. Construction of customized vocabulary

This has been mentioned in the html markdown file. We selected 976 words in our vocabulary list based on lasso fit.

## 7. Modeling

### 7.1 Benchmark Logistic Regression Model

Our First Model was a vanilla logistic regression model without any ridge penalty. In order to arrive at the best model for each fold, we undertook 5 fold cross-validation to arrive at the optimal set of hyperparameters for each fold, then we utilized the list of best parameters to do the prediction for each fold. The parameter search space for each model was {*"penalty": ['l1', 'l2'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], "solver":['newton-cg', 'lbfgs']}.* For all other parameters default values were used

### 7.2 Extreme Gradient Boost (XGBoost)

Following the Logistic Regression, we went ahead with XGBoosted trees. We followed a similar approach and did a 5 fold cross-validation for each fold to arrive at the optimal set of hyperparameters. We used the list of best parameters to do the prediction for each fold. The parameter search space for each GBR tree was *{"min\_child\_weight":[4,5,6], "max\_depth":[3,4,5]}.* For all the other parameters we used default settings

### 7.3 Logistic regression with ridge penalty

As per prof. Liang’s post on what we have tried, we created the vocabulary list by selecting non-zero features that are close to but less than 1000. Using these 976 words, we create a document term matrix for train and test sets. We then fit a logistic regression with ridge penalty and cross validation using glmnet. We created the probability of classification using min. Lambda by glmnet.

## 8. Results

**8.1 Benchmark Logistic Regression Model (Foldwise) [ordered from fold 1 to fold 5]:**

[0.89630, 0.89283, 0.89553, 0.89508, 0.89484] (**Avg ROC**): 0.89416

Vocabulary Size = approximately 2000

**8.2 XGBoost classification Model (Foldwise) [ordered from fold 1 to fold 5]:**

[0.8597, 0.8586, 0.8631, 0.8593, 0.8606] (**Avg ROC**): 0.86026

Vocabulary Size = approximately 2000

**8.3 Linear Model (Foldwise) [ordered from fold 1 to fold 10]:**

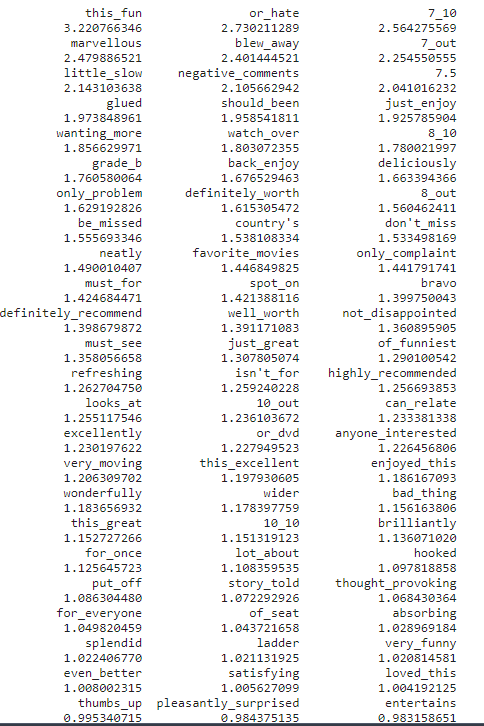
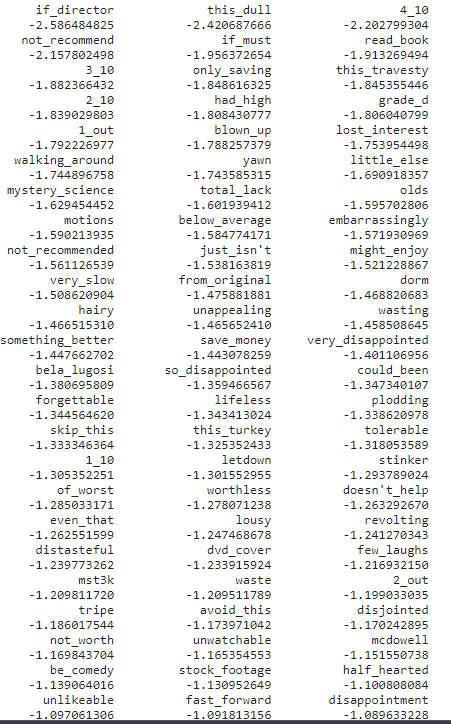
[ 0.9683942, 0.9681930, 0.9681220, 0.9690059, 0.9675730] (**Avg ROC**): 0.96824

Vocabulary Size = approximately 976

## 9. Running Time

* The system used for running a logistic regression model: Kaggle R notebook, 4 CPU cores, 16GB RAM. We could not find the exact processor used. It takes 41 +-2 seconds to train the model and generate predictions on test data for each split.
* The system used for running the Baseline Logistic Regression and XGBoost: Win 10, Intel Core i5 - 10600K CPU @4.10GHz (12CPUs). It takes 5 min to train 5 different models for both Random Forest and XGBoost

## 10. Interpretability of the chosen algorithm

* We have used ridge regression penalty with logistic regression. Since there are only 976 words used, we need to check the top words belonging to each sentiment (positive and negative)
* We have run the algorithm in an html markdown file and printed the coefficient values.
* The lower the coefficient value, the stronger the weight it has for a negative sentiment and vice-versa for the positive if the coefficient value is high (above zero)
* Top five words for positive sentiment are: 7\_10, should\_been, marvellous, only\_compliant, 7\_out. Top five words for negative sentiment are: scriptwriters, 4\_10, had\_high, if\_director, half\_hearted.
* These can be verified from the coefficient value (higher for positive sentiment, lower(negative) for negative sentiment). Left side image is for words(features) having high predictive power for positive sentiment. Right side image above is for words(features) having high predictive power for negative sentiment.

## 11 Error Analysis

The below review is having positive sentiment, but our model has predicted it as negative sentiment with prob. 0.3408.

The review: “Now, Throw Momma from the Train was not a great comedy, but it is a load of fun and makes you laugh. The title may seem a little strange, but the entire movie isn't literally about that, although it is about something just as sinister.<br /><br />Danny De Vito basically wants to kill his overbearing mother, and fast forward a little bit, some random and funny events take place. The premise is quite funny, and the things that Billy Crystal and Danny De Vito get into were great. Some of the scenes seemed to not fit in for me, but this didn't make it a bad movie.<br /><br />For what it is, a wacky comedy, it pulls it off well and should be seen once just to say you saw it.”

Words such as not great, isn’t literally, bad movie, not fit, are associated with negative sentiment, but the reviewer is actually using these to say the opposite. Because our model learns from all the reviews, this was misclassified. There are positive sentiment words too, but because of higher coefficient values for these negative sentences, the model predicts it as negative sentiment. We can also observe that reviews that got extreme negative probability of extreme positive probability are classified correctly.

## 12 Model Limitations and Future Actions

* We have fit a logistic regression model which has a hard decision boundary. Our algorithm is more interpretable but can suffer for such a boundary.
* Using a vocab list selected from all reviews gives higher auc. If we select the vocab list from the 1st split, it gives AUC around 0.955. But for other splits it gives auc higher than 0.96. Again because we already knew these words beforehand. We can perform further semantic analysis. Similar words can be grouped. This can further reduce the number of features we use in our document term matrix and for unseen words, it can identify the semantics.
* We can try some of the advanced NLP algorithms such as LSTM that can give us high auc.

## 13 Interesting Findings

* Word2vec models performed better than tf-idf models. The difference in auc was about 7%. Performance of count vectorized models (bag of words) is worse compared to tf-idf and word2vec models.
* We could identify words that have higher predictive power to classify a review as positive or negative sentiment.

## 14. Conclusion

The project made us realize the importance of building an interpretable classifier. The initial version of the classifier that we built with tf-idf vectorizer was giving us good results in terms of auc. When we changed it to the word2vec model, it gave us better results. We could get satisfying output results with an interpretable and fast logistic regression algorithm.

## 15. References

1. Bag of Words Meets Bags of Popcorn Kaggle https://www.kaggle.com/c/word2vec-nlp-tutorial/code?competitionId=3971&sortBy=voteCount
2. STAT 542 - Project 3. UIUC https://liangfgithub.github.io/F21/F21\_Project3.nb.html
3. STAT 542 - Campuswire posts on Project 3. UIUC <https://campuswire.com/c/G497EEF81>