# CSE 487/587 Assignment 3: Predictive Analytics with Spark


**Team Members:**

1. **Amit Upadhyay: 50336909** amitupad@buffalo.edu
2. **Namrata Bakre: 50336849** nbakre@buffalo.edu
3. **Nitin Kulkarni: 50337029** nitinvis@buffalo.edu

## Aim:

- The objective of this assignment is to get started with predictive Analytics with Apache Spark.
- The goals of the assignment are to use Spark Libraries to implement an end to end Predictive Analytics Pipeline and get introduced to the data science competition platform Kaggle.

## Steps Taken:

Apache Spark is a unified analytics engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing.

1. Created an account on Kaggle for our team.
2. Downloaded train.csv, test.csv, sample.csv and mapping.csv from the Kaggle website.
3. Analyzed the data and preprocessed it.
4. We have used Logistic Regression in Spark to use the information provided in train set to predict the genres associated with a movie.
5. We have generated predictions for test data set and uploaded to Kaggle website.

We have used Google colab to run our code. For each run, we have uploaded Train.csv, Test.csv and Mapping.csv file using upload tab on the left.

## Implemented Logic:

## PART 1 – Basic Model

The aim of this part is to design a basic machine learning model that predicts movie-genre using term matrix of their plots as feature vectors to train and predict results from model. The model will be trained using the data available in train.csv file and will be predicting results for movies in test.csv. The model will be created, trained and tested on spark which is an open system distributed framework. Its preferred over Hadoop as its faster and easier to use. A special python API named pyspark will be used to implement this project.
The data from train.csv was 1-hot coded and it was done to create a new data frame which displays movies and all the genres they belong to. It's a matrix form representation with every index being "1" if movie belong to that particular genre, otherwise "0".
Then this data frame is cleaned for any form of abnormalities and stop-words were removed (as they are most common words and may lower the overall accuracy). Later, a term document matrix was created from the "cleaned "plot of data frame, (which is called feature in program)

Now, as there are 20 different genres, 20 different data frames which consists of movie_id, feature and one signal genre. These data frames will be later used to train the machine learning models.
The machine learning models used here are logistic regression based binary classifiers. There are twenty of them, each of which classifies the movie based on a particular genre. Each of these takes feature as input and returns "1" if the given movie belongs to particular genre, else "0'. The twenty data frames created earlier will be used to train each of these models.
Now, as the models are trained and ready, the test.csv file is uploaded, 1-hot coded, preprocessed. A term matrix from the plot of every movie id is created. Finally, its feature is given as input to all twenty classifiers. The classifiers classify each of the movies based of their belonging to that particular genre. Finally, a new data frame with movie id and their genre(s)("1" if it belongs and" "0" if it doesn't belong) is created, converted to csv file and is uploaded on Kaggle website and the "F1 score" is obtained. The obtained accuracy is **0.97380.**

## PART 2 - Use TF-IDF to improve the model

We have implemented Term Frequency-Inverse Document Frequency (TF-IDF) based feature engineering technique to improve the performance of the model. Term frequency-inverse document frequency (TF-IDF) is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus. Denote a term by tt, a document by dd, and the corpus by DD. Term frequency TF(t,d)TF(t,d) is the number of times that term tt appears in document dd, while document frequency DF(t,D)DF(t,D) is the number of documents that contains term tt. Our implementation of term frequency utilizes the hashing trick. A raw feature is mapped into an index (term) by applying a hash function. Then term frequencies are calculated based on the mapped indices. This approach avoids the need to compute a global term-to-index map, which can be expensive for a large corpus, but it suffers from potential hash collisions, where different raw features may become the same term after hashing.
We have generated predictions similar to Part 1 and have uploaded to Kaggle website and the F1 score was obtained. The obtained accuracy is **0.98839**.

## PART 3 – Custom Feature Engineering

We have implemented Word2vec text-based feature engineering method to improve the performance of the model. Word2Vec computes distributed vector representation of words. The main advantage of the distributed representations is that similar words are close in the vector space, which makes generalization to novel patterns easier and model estimation more robust. Distributed vector representation is showed to be useful in many natural language processing applications such as named entity recognition, disambiguation, parsing, tagging and machine translation. For our implementation, we have constructed a Word2Vec instance and then fit a Word2VecModel with the input data frame.

We have generated predictions similar to Part 1 and have uploaded to Kaggle website and the F1 score was obtained. The obtained accuracy is **0.99977.**