# HIB VISA DATA PROCESSING

## Introduction

The H1B visa is an employment-based, non-immigrant visa category for temporary workers. For such a visa, an employer must offer a job and apply for your H1B visa petition with the US Immigration Department. This approved petition is a work permit which allows you to obtain a visa stamp and work in the U.S. for that employer. The H1B visa is issued for a specialty occupation, requires theoretical and practical application of a body of specialized knowledge and requires the visa holder to have at least a Bachelor’s degree or its equivalent[[1]](#footnote-1).

Every year, number of people that apply for H1B visas increasing with the numbers for 2016 alone, estimated to be 236,000[[2]](#footnote-2). The U.S. Citizenship and Immigration Services grants just 85,000 H-1B visas (20,000 of which are reserved for master's degree holders), which are selected on, via a lottery process.

Before application applying for the H1B visa, applicants have to apply for a Labor Condition Application, which is a mandatory document that the H1B Sponsor / employer needs to file with US Department of Labor before they file the H1B petition with USCIS for any non-immigrant worker.  The status of the LCA application determines if a new employee is eligible to apply for a H1B visa or not. Applying for the LCA is an expensive process both in terms of costs (attorney fees) and time. An important aspect of this project focused using LCA data to predict if a particular employee’s application will either be certified or denied. By being able to determine whether an application will be successful or not even before its filed would save potential employers a lot of time and money.

## Project Dataset

The dataset was obtained from Kaggle[[3]](#footnote-3) and contains over 3 million records of H1B petitions from the period 2011 to 2016. The original dataset size was over 700MB.

The columns in the dataset include:

1. CASE\_STATUS: Status associated with the last significant event or decision. Valid values include “Certified,” “Certified-Withdrawn,” Denied,” and “Withdrawn”.
2. EMPLOYER\_NAME: Name of employer submitting labor condition application.
3. SOC\_NAME: Occupational name associated with the SOC\_CODE. SOC\_CODE is the occupational code associated with the job being requested for temporary labor condition, as classified by the Standard Occupational Classification (SOC) System.
4. JOB\_TITLE: Title of the job
5. FULL\_TIME\_POSITION: Y = Full Time Position; N = Part Time Position
6. PREVAILING\_WAGE: Prevailing Wage for the job being requested for temporary labor condition. The wage is listed at annual scale in USD.
7. YEAR: Year in which the H-1B visa petition was filed
8. WORKSITE: City and State information of the foreign worker's intended area of employment
9. lon: longitude of the Worksite
10. lat: latitude of the Worksite

The CASE\_STATUS field denotes the status of the application after LCA processing. Certified applications are filed with USCIS for H-1B approval. CASE\_STATUS: CERTIFIED does not mean the applicant got his/her H-1B visa approved, it just means that he/she is eligible to file an H-1B.

## Project Objectives

The two main objectives of the project were:

* To use machine learning techniques to design a model aimed at predicting the case status of any given applicant, given the input data.
* To use big data tools to explore the dataset to determine which employers had the highest certification rates.

# Project Technologies

## Introduction

## We explored multiple big data technologies in order to be able to achieve the project objectives. The technologies listed below were the ones utilized.

## Spark

Apache Spark is a fast and general-purpose cluster computing system[[4]](#footnote-4). It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including [Spark SQL](http://spark.apache.org/docs/latest/sql-programming-guide.html) for SQL and structured data processing, [MLlib](http://spark.apache.org/docs/latest/ml-guide.html) for machine learning, [GraphX](http://spark.apache.org/docs/latest/graphx-programming-guide.html) for graph processing, and [Spark Streaming](http://spark.apache.org/docs/latest/streaming-programming-guide.html).

## MapReduce

MapReduce is a [programming model](https://en.wikipedia.org/wiki/Programming_model) and an associated implementation for processing and generating [big data](https://en.wikipedia.org/wiki/Big_data) sets with a [parallel](https://en.wikipedia.org/wiki/Parallel_computing), [distributed](https://en.wikipedia.org/wiki/Distributed_computing) algorithm on a [cluster](https://en.wikipedia.org/wiki/Cluster_(computing))[[5]](#footnote-5).

A MapReduce program is composed of a [Map()](https://en.wikipedia.org/wiki/Map_(parallel_pattern)) [procedure](https://en.wikipedia.org/wiki/Procedure_(computing)) (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() method that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System orchestrates the processing by [marshalling](https://en.wikipedia.org/wiki/Marshalling_(computer_science)) the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for [redundancy](https://en.wikipedia.org/wiki/Redundancy_(engineering)) and [fault tolerance](https://en.wikipedia.org/wiki/Fault-tolerant_computer_system).

## DeepLearning4j Framework

Deeplearning4j is the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala. Integrated with Hadoop and Spark, DL4J is designed to be used in business environments on distributed GPUs and CPUs[[6]](#footnote-6). Deep neural nets are capable of [record-breaking accuracy](https://deeplearning4j.org/accuracy.html). Deeplearning4j lets you compose deep neural nets from various shallow nets, each of which form a so-called `layer`. This flexibility lets you combine restricted Boltzmann machines, other autoencoders, convolutional nets or recurrent nets as needed in a distributed, production-grade framework that works with Spark and Hadoop on top of distributed CPUs or GPUs.

## Amazon Web Services

Amazon Web Services[[7]](#footnote-7) (AWS) is a secure [cloud](https://aws.amazon.com/what-is-cloud-computing/) services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow. Explore how millions of [customers](https://aws.amazon.com/solutions/case-studies/) are currently leveraging AWS cloud [products](https://aws.amazon.com/products/) and [solutions](https://aws.amazon.com/solutions/) to build sophisticated applications with increased flexibility, scalability and reliability.

## 40px-spaceS3

Amazon Simple Storage Service[[8]](#footnote-8) (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web.

# Methodology

## Introduction

In order to achieve the project objectives stated above the following approach was followed;

We processed the original dataset using java to produce an input file. This included the transformation of all the character content into numerical values; this was because for ANN training, the dataset can only contain numerical values, so we needed to transform the character values into indices and create the dictionaries for each character columns.

Using java code, we generated different hyperparameter combinations. The hyperparameters for the project are defined below;

* + Learning Rate:  the size of the adjustments made to the weights with each iteration.
  + Batch size: A batch is a bundle of examples, or instances from your dataset. With mini-batches you can get more updates to your model in a single epoch.
  + Epoch size: how many full passes of the data set (epochs) should be used.
  + Momentum: Momentum is an additional factor in determining how fast an optimization algorithm converges on the optimum point.

Using the DeepLearning4j framework, we designed a deep learning Artificial Neural Network to predict the case\_status, of any given applicant in the petition dataset. The processed input file was divided into two parts namely the training dataset and the testing dataset. The designed ANN was trained using the training dataset and was tested with data from the test dataset to determine the model’s accuracy.

Using Hadoop, Spark, DeepLearning4j along with hyper parameters generated from step 2 above, we created parallel clusters on Amazon Web Services EMR to train the ANN, using different hyperparameter combinations.

Using MapReduce, we collected output files from the multiple clusters, which had been stored on S3, and ranked them in order to determine which set of hyperparameters produced a model with the highest accuracy.

Also, using MapReduce and the original input file, we were able to rank which companies had the highest certification rates throughout the whole dataset.

# Practical Work Done

## Dataset Transformation:

The original dataset was transformed, by changing the character content into numerical values: because for ANN training, the dataset can only contain numerical values, so we needed to transform the character values into indices and create the dictionaries for each character columns.

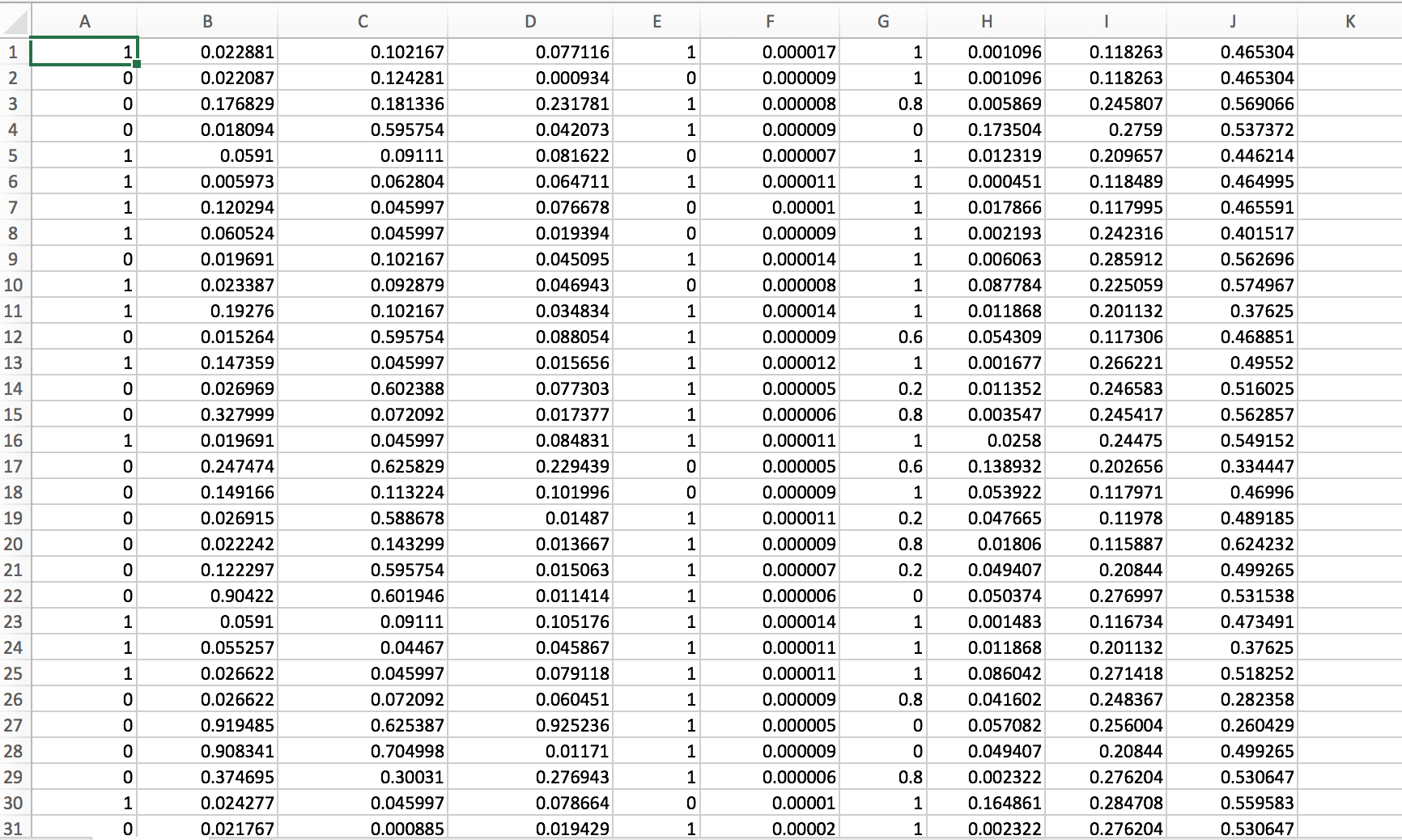


Figure 1: Snapshot of Processed Input File

The following steps were used to generate the input file;

1. We normalized the dataset: Neural networks work best when the data they are using is constrained to a range between -1 and 1, so we need to normalize the dataset before training.
2. Generated balanced data: our algorithm can just deal with balanced data, so we need to filtrate our data and make the dataset balance.
3. Shuffle the data: shuffling the data helps to improve the training efficiency. In our project, we shuffled the data 100 times before dealing with it.

## Deep Learning Network Design

Local Training

We locally built a neural network using DL4J framework. We trained the ANN on a sample of the original dataset, and viewed the training processes using UI.



## Parallel Training Using the Cloud

We used MapReduce to automatically parallel create clusters. For each cluster, it created a Spark job to train ANN, using the DeepLearning4j framework, the hyperparameters used for each cluster were automatically generated using java.

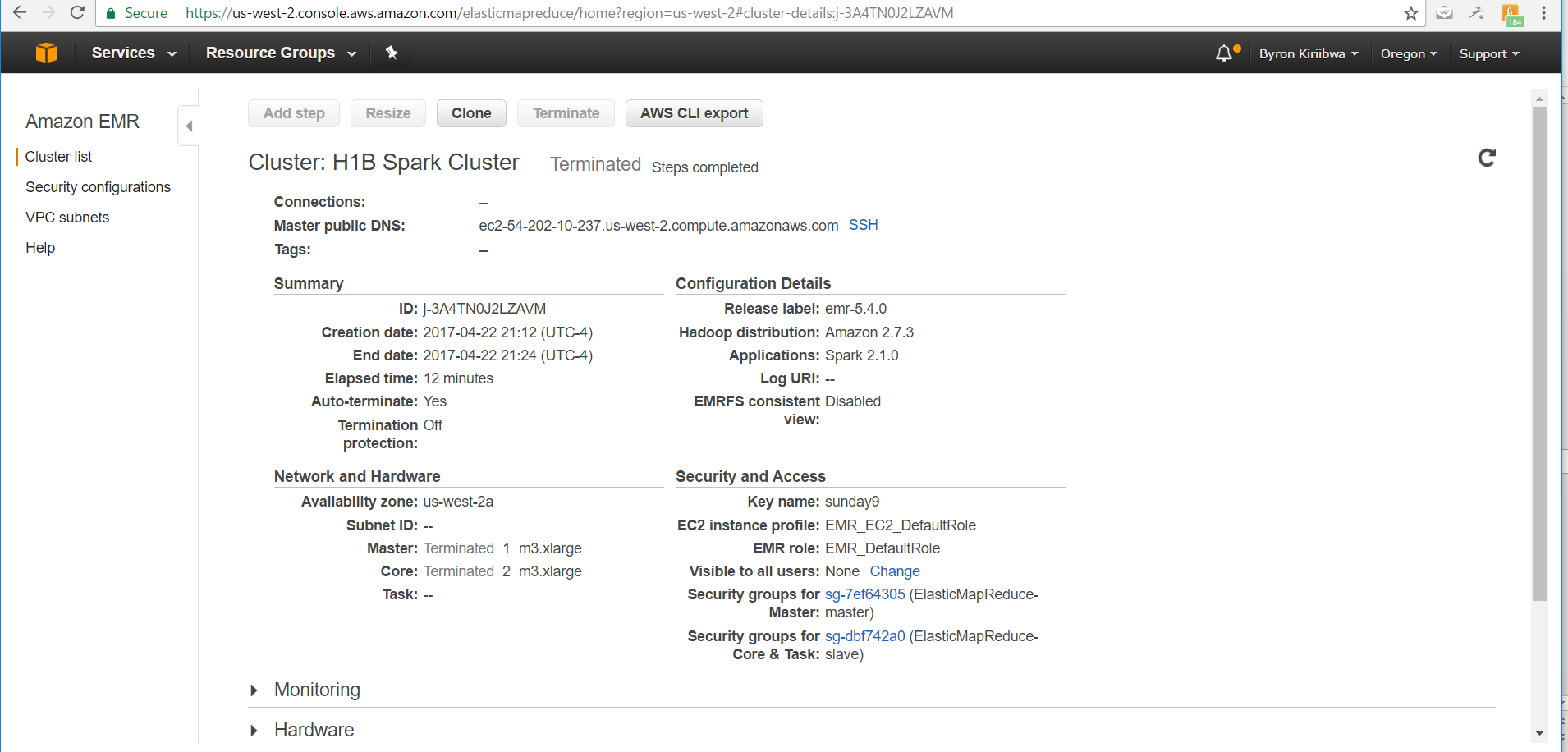


Figure 2: Showing Spark Cluster Configuration

When the training terminated, all clusters uploaded output into S3, and automatically terminated. The picture below shows an example of the results produced after the training, using a given set of hyperparameters

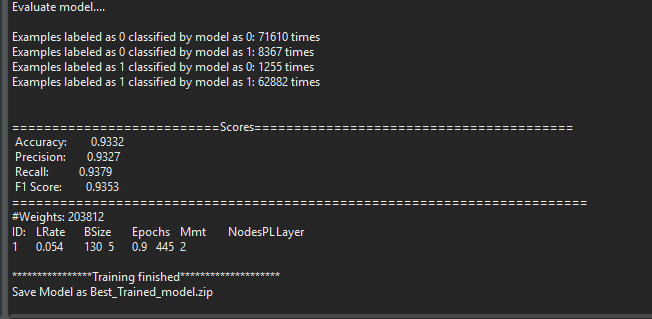


Figure 3: Showing a Spark Cluster Output

## MapReduce

We used MapReduce to generate groups of parallel clusters for ANN training. This was mainly done to determine the best hyperparameter combinations for training the dataset. Optimizing hyperparameters usually takes a long time. In order to speed up this process, we use MapReduce to create clusters and execute them in parallel.

Using MapReduce to collect results of the Spark job execution; we used MapReduce to rank the tested hyperparameter combinations based on their accuracies, and chose the top hyperparameter combination to train our model. The picture below shows the rankings of different training outputs based on model accuracy.

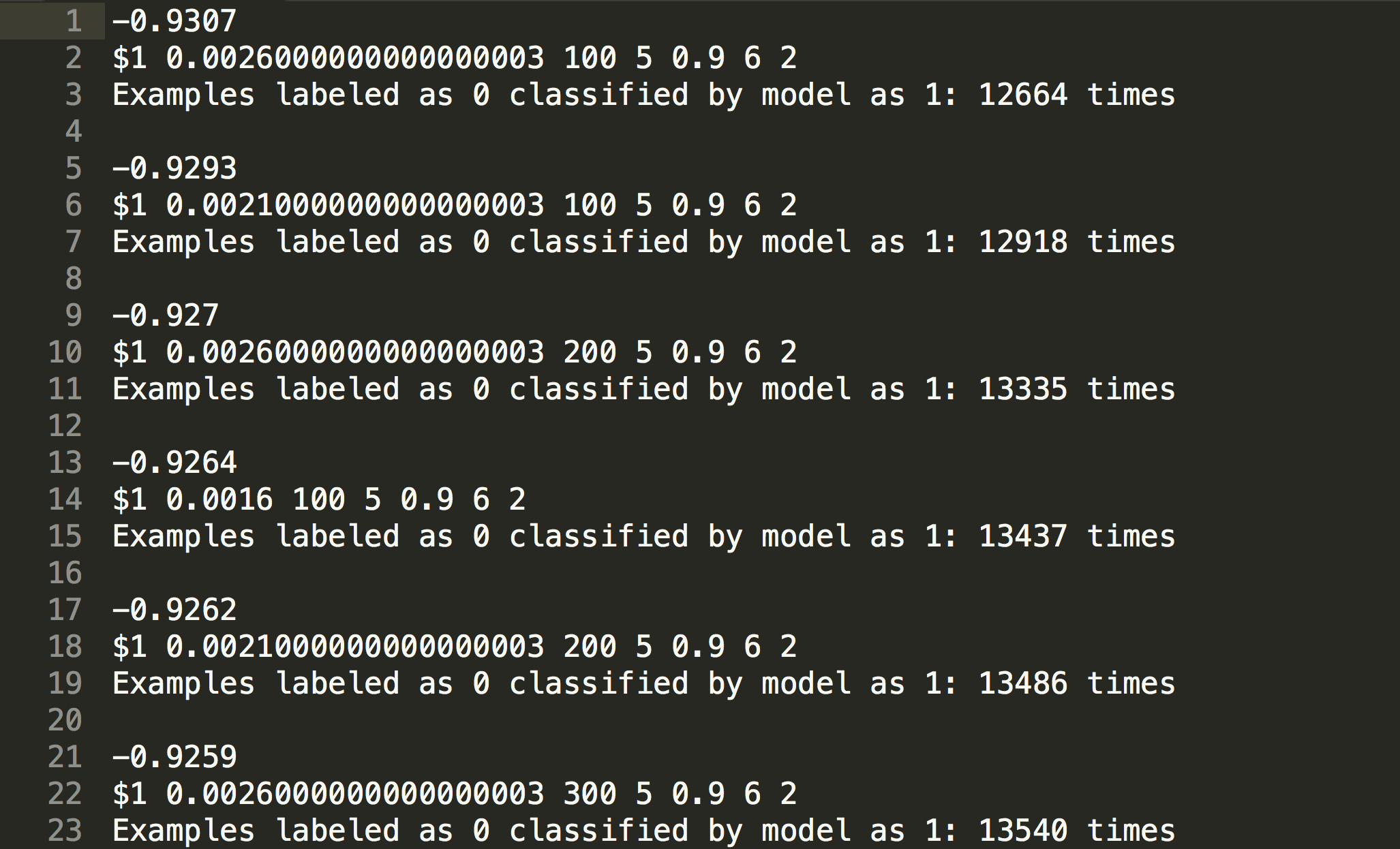


Figure 4: Showing Results of Training Hyperparameter Combinations Ranked by Accuracy

## Employers Rankings

The last part of the analysis involved ranking of employers based on the number of their applicants that got certified as the case status. This would help us determine which employer had the highest chances of their LCA applications certified. The results of this MapReduce process are shown in the figure below;

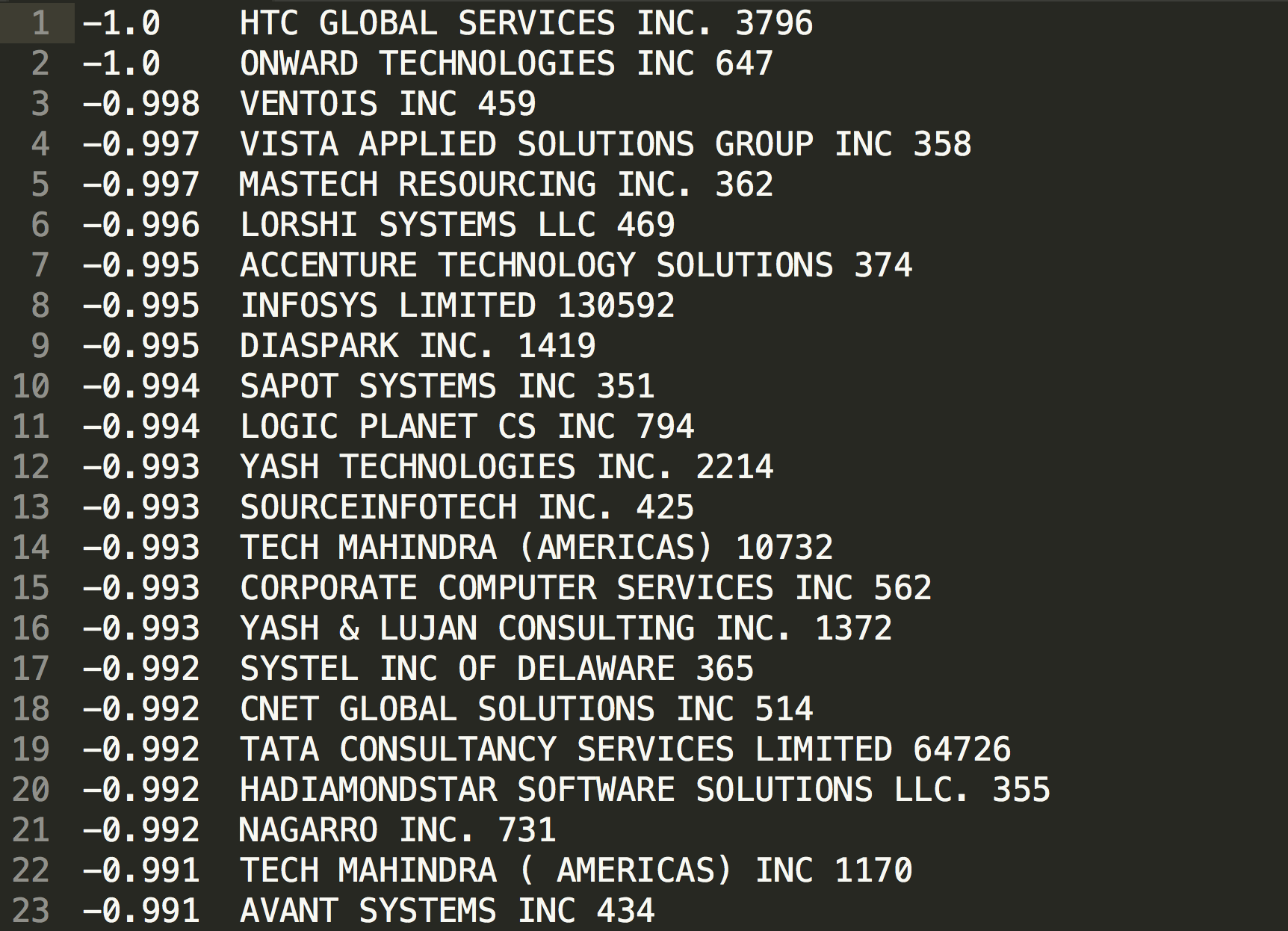


Figure 5: Employers Ranked based on Applicants Certified

# Challenges

## Introduction

During the course of the project, we faced the following challenges;

## Parallel training

We thought it was possible to use MapReduce to parallel train ANNs using different hyperparameter combinations, but later we found that this was impractical. This was due to the fact that only one Spark job can be executed on spark cluster at any given time, In order to execute parallel training, we had to commission multiple spark clusters at the same time, which of course had cost implications. Therefore, we created EMR Spark clusters using AWS SDK for Java rather than in AWS console. Then we tried to solve the storage problem because outputs were stored in different clusters. We uploaded the project jar to S3, and set Spark Steps for all clusters to use the jar in cloud, and to upload outputs to S3 after job finished. Finally, we downloaded all outputs from S3, and use MapReduce to analyze all outputs.

## Unbalanced Dataset

The dataset was too unbalanced to be directly used in any analysis that we needed to do. As an example, the dataset had 3 million petitions for LCA certification, with four possible outcomes of case status which included certified, denied, certified\_withdrawn and withdrawn. However, over 2.8 million of the applicants had been certified, leaving only 280,000 for the other three categories. We used

## Tuning Hyperparameters

It’s really hard to tune them although we used a semi-automatic approach. All parameters will influence the accuracy, and there is no general guide to set them. We think we can use some optimized algorithms (GA, PSO, etc. even ANNs) to deal with hyperparameter combinations. We believe it is a good way to use advanced algorithms to predict accuracy, and get a better model.

## Future Scope

Dataset:

* + 1. Unbalanced dataset: SMOTE could be a choice.
    2. Misleading date: We cannot deal with this part. In real life, there would be a lot of fake information that have the same features but are divided into different categories. We need to find a way to filter them to improve model accuracy.
    3. Other researchers could use our approach to tune their networks, especially when they are looking for the optimal hyperparameter combinations for their datasets.
    4. Other researchers could also use our most accurate model to design a web application so that people interested in applying for H1B visas can find their potential LCA certification before spending time and money in applying for LCA.
    5. Due to the limited time frame for this project, we were not able to get an optimal set of weights for the input data combinations, other researchers could focus on finding what these weights are.

1. https://www.path2usa.com/what-is-h1b-visa [↑](#footnote-ref-1)
2. http://money.cnn.com/2016/04/12/technology/h1b-cap-visa-fy-2017/?iid=EL [↑](#footnote-ref-2)
3. https://www.kaggle.com/nsharan/h-1b-visa [↑](#footnote-ref-3)
4. http://spark.apache.org/docs/latest/ [↑](#footnote-ref-4)
5. https://en.wikipedia.org/wiki/MapReduce [↑](#footnote-ref-5)
6. https://deeplearning4j.org/ [↑](#footnote-ref-6)
7. https://aws.amazon.com/ [↑](#footnote-ref-7)
8. https://aws.amazon.com/s3/ [↑](#footnote-ref-8)