**Omer Khan**

**CS 461**

**Brian Hare**

**May 7, 2020**

**Program 4 Report**

To start off, the CSV file is reconfigured to match the necessary requirements to parse the data correctly. The python code creates a list of top 100 words by using counter function in the collection’s library. This list of words is then created into a dictionary based on the frequency and the number of times the word shows up. The whole CSV is then formatted to include 100 columns for variety; one for each top 100 keywords, and a floating decimal is written as a record of frequency. Next, the brand names are also similarly stored into a list of strings to make it easier to figure our how many times a brand name shows up and to mark single entries as ‘other’. Finally, 3 total new CSV are made based off the above process. One for each: training dataset, validation dataset, and testing dataset. The splitting ratio is 80/10/10 respectively.

The neural network part of the program starts by creating a pandas dataframe of each of the new CSVs. These dataframes are given ‘brand’, ‘style’, and ‘country’ as a categorical data and each is given a series of codes. Then keras is used to convert the categorical data into a vector class of binary matrix (this only includes the actual data and not the header row). Next, the training, validation, and testing datasets are put into tensor slices in tensorflow using their respective values and categorical data. These datasets are then shuffled and a batch of size 1 is created.

The main neural network consists of 5 total sequential layers. Four with rectified linear unit activation function of density 100, and a sigmoid activation function with density 1. The layers are compiled with the ‘adam’ optimizer and mean squared error for loss. The model is trained with accuracy metrics being measured. This model is fitted with the training and validation dataset with 10 total epochs. 1 epoch gave an accuracy of about 80% and after about 4 or 5 epochs, the accuracy would not increase or make a difference at all. After about 10 epochs, the accuracy started dropping. So, 10 epochs were best in order to not over-train or under-train the neural network. Maybe I should have stuck to around 5-8 epochs, but 10 seem sufficient.

The final outcome after running the model against the testing data results with a loss of 15.96% and an accuracy of 83.33%.

