Group Member 1

Name: **Kunal Chand**
UB IT Name: **kchand**
UB Email: **kchand@buffalo.edu**

Group Member 2

Name: **Koushani Chakrabarty**
UB IT Name: **koushani**
UB Email: **koushani@buffalo.edu**

CSE589 DIC Project Phase 2

During this stage, we'll conduct statistical modeling on our pre-processed dataset. Our focus will be on constructing the subsequent models:

1. K-means clustering
2. K-nearest neighbors (KNN)
3. Logistic regression
4. Convolutional neural network (CNN) (*Algorithm NOT discussed in lecture*)
5. Recurrent neural network (RNN) (*Algorithm NOT discussed in lecture*)

Sources for algorithms taken from outside the class:

1. CNN:
    a. What Is a Convolutional Neural Network? A Beginner's Tutorial for Machine Learning and Deep Learning (freecodecamp.org)
    b. "Convolutional Neural Network (CNN)" by Google Colab

2. RNN:
    a. The Ultimate Guide to Recurrent Neural Networks in Python (freecodecamp.org)
    b. RNN (Recurrent Neural Network) Tutorial: TensorFlow Example (guru99.com)
    c. Recurrent Neural Network tutorial for Beginners - ThinkInfi

For each of the 5 above algorithms our goal is:

1. DESCRIPTION: Provide a brief description of the algorithm
2. JUSTIFICATION: Provide justification for why we chose the particular algorithm for our particular problem
3. TRAINING: Work we had to do to tune/train the model
4. EFFECTIVENESS: Discuss the effectiveness of the algorithm when applied to our data to answer questions related to our problem statement. This should include discussion of

any relevant metrics for demonstrating model effectiveness, as well as any intelligence we were able to gain from application of the algorithm to our data.

1) K-means clustering

DESCRIPTION:
K-means clustering is an unsupervised machine learning algorithm used for clustering or grouping similar data points together in a dataset. The goal of the algorithm is to partition a given dataset into K clusters, where K is a predefined number of clusters chosen by the user.

The algorithm works by randomly selecting K initial centroids (representative points) from the dataset. Then, it assigns each data point to the nearest centroid and forms K clusters. After that, the algorithm computes the new centroid for each cluster by taking the mean of all the data points belonging to that cluster. The algorithm iteratively repeats the assignment and centroid update steps until convergence, which occurs when the assignment of data points to clusters no longer changes or the maximum number of iterations is reached.

JUSTIFICATION:
We chose K-Means Clustering for our particular problem as it is particularly useful for identifying patterns and groupings in large datasets, especially for text data analysis. With the increasing amount of text data, it has become more difficult to manually analyze and classify this data. K-Means clustering can automatically group similar reviews together, providing a more efficient way to analyze the data. K-means clustering can help identify distinct clusters of reviews based on their underlying sentiment, without the need for labeled data. By grouping reviews into clusters, we can get a better understanding of the common themes or issues mentioned in positive and negative reviews and identify patterns in customer satisfaction levels across different video game genres and platforms.

TRAINING:
The text data is first preprocessed using TfidfVectorizer to tokenize and vectorize the text data, followed by dimensionality reduction using TruncatedSVD to reduce the dimensionality of the vectorized data to 50 dimensions. The resulting data is then normalized using the Normalizer function. The KMeans algorithm is then used to cluster the data into 10 clusters, and the cluster labels are assigned to each review in the merged_df_text DataFrame. The algorithm is parallelized using the Parallel function from the joblib library to speed up the processing time.

EFFECTIVENESS:
The count of reviews across each cluster is plotted using the countplot function from the seaborn library. This plot can be used to gain insights into the distribution of reviews across the clusters and identify any patterns or trends.

We can use the clusters to gain insights into the sentiment expressed by customers towards video games on Amazon. We can analyze the sentiment scores for each cluster to identify the most positive and negative clusters. We can also use the clusters to identify the most common

themes or issues mentioned in positive and negative reviews. By comparing the sentiment scores across different platforms, we can identify any differences in customer satisfaction levels and gain insights into the platform preferences of customers.

2) K-nearest neighbors (KNN)

DESCRIPTION:
K-nearest neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. The goal of the algorithm is to predict the class or value of a new data point by looking at the K closest data points in the training set. KNN is a simple and intuitive algorithm that does not require any training. It can handle both binary and multiclass classification tasks, as well as continuous and discrete values for regression tasks.

The algorithm works by first storing the entire training dataset in memory. When given a new data point, it calculates the distance between the new point and all the points in the training set. It then selects the K nearest neighbors to the new point, based on the calculated distances. For classification tasks, the algorithm assigns the class label that is most common among the K nearest neighbors to the new data point. For regression tasks, the algorithm calculates the mean or median of the K nearest neighbors' values and assigns that value to the new data point.

JUSTIFICATION:
KNN is a good choice for this problem because it is a simple and effective algorithm for text classification tasks. KNN is able to identify similarities in text data and classify them based on their similarities. It is particularly well-suited for the task of text classification because it works by finding the k nearest neighbors to a given data point based on some similarity measure. In the case of text data, this similarity measure is often based on the frequency of words in the text. Additionally, KNN is computationally efficient and does not require a training phase, which makes it a good choice for large datasets.

TRAINING:
The first step in training the KNN algorithm is to preprocess the text data by tokenizing and vectorizing it using a CountVectorizer. This converts the text data into a numerical representation that can be used by the algorithm. After vectorizing the data, the dataset is split into a training set and a testing set using the train_test_split function from sklearn. The value of k is set to 5, which means that the algorithm will classify a test point based on the majority class of its five nearest neighbors.

EFFECTIVENESS:
The effectiveness of the KNN algorithm is measured using various performance metrics such as accuracy, precision, recall, and f1-score. The accuracy of the KNN algorithm for the subset of the data is printed, which tells us the percentage of correctly classified instances out of all instances. Additionally, the precision, recall, and f1-score are printed for the positive class,

which tells us how well the algorithm is able to correctly identify positive reviews. These metrics can be used to evaluate the effectiveness of the algorithm for sentiment analysis of Amazon reviews for video games.

3) Logistic regression

DESCRIPTION:
Logistic regression is a supervised machine learning algorithm used for binary classification tasks, where the goal is to predict a binary output variable based on one or more input variables.

The logistic regression algorithm works by first calculating the probability of the binary output variable being 1 for each input variable using the logistic function. Then, it compares the calculated probabilities with a threshold value and assigns the binary output variable to 1 if the probability is greater than or equal to the threshold value, and 0 otherwise. The algorithm is trained using a labeled training dataset. During training, the algorithm adjusts the weights of the input variables to minimize the difference between the predicted probabilities and the actual binary output values in the training data.

JUSTIFICATION:
The logistic regression algorithm was chosen for the text analysis of Amazon reviews based on its effectiveness in predicting binary outcomes. In this case, the binary outcome is whether a review is positive or negative. Furthermore, the logistic regression algorithm is well suited for working with large datasets and is known for being robust against noise in the data. Since Amazon reviews are known to be noisy with a large volume of data, logistic regression provides a good solution for handling these characteristics of the data.

TRAINING:
The first step in training the model is to preprocess the text data. In this case, the review text is tokenized and padded to a fixed length of 100. The text is then converted into numerical vectors using the tokenizer, and the resulting matrix is standardized using the StandardScaler. The data is then split into training and testing sets using train_test_split. The logistic regression model is built using Keras Sequential API, with a single output layer using sigmoid activation function. The model is then compiled using the Adam optimizer and binary_crossentropy loss function, with accuracy as the metric. The model is trained using fit method with 5 epochs and a batch size of 1024.

EFFECTIVENESS:
The model's effectiveness is evaluated using several metrics such as accuracy, confusion matrix, ROC curve, and precision-recall curve. The accuracy score is calculated as the ratio of correctly classified instances to the total number of instances. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. The ROC curve and the area under the curve (AUC) is used to evaluate the performance of the model at various thresholds. A good model should have a higher AUC, which indicates that the model is capable

of distinguishing between the positive and negative classes.

4) Convolutional neural network (CNN)

DESCRIPTION:
Convolutional neural networks (CNNs) can be used to analyze Amazon customer reviews by processing and extracting features from the text data.

CNN algorithm consists of several layers, each with a specific purpose. At the heart of the CNN architecture are convolutional layers, which perform a series of convolutions over the input data using a set of learnable filters. These filters allow the network to learn spatially localized features from the input data, which are then passed through pooling layers to reduce the dimensionality of the output. The output of the final convolutional layer is then flattened and passed through a series of fully connected layers, which can be used for classification or regression tasks. During training, the network's weights and biases are updated using backpropagation.

JUSTIFICATION:
While CNNs are not typically used for text data analysis, CNNs have been found to perform well for text analysis tasks because of their ability to automatically learn useful features from text data. One of the key strengths of CNNs is their ability to identify local patterns within a sequence of input data, in our case, text.

CNN is a suitable algorithm for text analysis of Amazon reviews as it can extract meaningful features from the text data and classify the reviews based on their sentiments. Also, CNN has the ability to learn complex relationships between words and phrases makes it a suitable choice for analyzing reviews.

TRAINING:

Our CNN model was trained on a dataset of Amazon product reviews. The first step was to preprocess the text data by converting all the words to lowercase, removing punctuation, and lemmatizing the words. The dataset was split into training and testing sets, and the training data was tokenized and padded to a fixed length. The CNN model was then constructed using Keras with an embedding layer, followed by a convolutional layer, a max pooling layer, an LSTM layer, and a dense output layer. The model was trained using binary cross-entropy loss and the Adam optimizer.

EFFECTIVENESS:
The CNN algorithm was effective in classifying the sentiment of the Amazon reviews with an accuracy of approximately 88%. The confusion matrix shows that the model has high true positive and true negative rates, indicating that the model is able to accurately classify both positive and negative reviews. The area under the curve (AUC) for ROC curve for the model indicates that the model performs well at distinguishing between positive and negative reviews.

The precision-recall curve also shows that the model has a high average precision, indicating that the model is able to accurately predict positive reviews. Overall, the CNN algorithm is an effective algorithm for sentiment analysis of Amazon reviews.

5) Recurrent neural network (RNN)

DESCRIPTION:
RNN (Recurrent Neural Network) is a type of neural network that is specialized in processing sequential data. It has feedback connections that allow the network to store information about the previous input and use it to influence the current output. The specific implementation used here involves an embedding layer, a Long Short-Term Memory (LSTM) layer, and a dense layer with a sigmoid activation function. The embedding layer converts the text data into numerical vectors, which are then processed by the LSTM layer that learns to identify patterns in the data. Finally, the dense layer outputs a binary prediction for each review, indicating whether it is positive or negative.

The key feature of RNNs is that they have a "memory" of the previous inputs, which allows them to learn the temporal dependencies in the data. This memory is implemented using a hidden state that is updated at each time step using a set of learnable weights. The RNN architecture consists of three main components: an input layer, a hidden layer, and an output layer. The input layer receives the sequential input data and passes it through the hidden layer, which applies a set of weights to the input and the previous hidden state to produce a new hidden state. The output layer then takes the hidden state and produces a prediction or output based on the task at hand.

JUSTIFICATION:
RNNs are commonly used in natural language processing tasks, such as sentiment analysis, because they can handle variable-length inputs and capture long-term dependencies in the data. In this case, the goal is to classify Amazon reviews as positive or negative based on the text content. RNNs are well-suited for this task because they can process the reviews as sequences of words and capture the relationships between them. Additionally, the LSTM layer can help avoid the vanishing gradient problem commonly encountered in traditional RNNs.

TRAINING:
During the training process, the RNN model was optimized to classify Amazon reviews as either positive or negative. The model was trained using a training set of reviews, where binary cross-entropy loss was employed as the loss function, and the Adam optimizer was used to optimize the weights and biases of the model. The training was conducted for a single epoch, and the model's performance was evaluated on a validation set after each batch. The accuracy metric was used to monitor the model's performance during the training process. To fine-tune the model, hyperparameters such as the number of units in the LSTM layer, the learning rate, and the number of epochs were adjusted. By tuning these parameters, the model was optimized to achieve the best possible performance in classifying Amazon reviews.

EFFECTIVENESS:

The effectiveness of the RNN algorithm can be evaluated based on various metrics such as accuracy, precision, recall, F1 score, and the area under the ROC curve (AUC-ROC).

The classification report shows various metrics such as precision, recall, F1 score, and support for both the positive and negative classes. The precision metric measures the proportion of true positive instances out of all instances predicted as positive. The recall metric measures the proportion of true positive instances out of all actual positive instances. The F1 score is the harmonic mean of precision and recall, and support is the number of instances in each class.

The confusion matrix is a table that shows the number of true positives, false positives, true negatives, and false negatives for both the positive and negative classes. It provides a visual representation of how well the algorithm is performing.

The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. The AUC-ROC is a metric that measures the overall performance of the algorithm across all possible classification thresholds.

The precision-recall curve is a plot of precision against recall at different classification thresholds. The average precision score (AP) is the area under the precision-recall curve, and it provides a single metric to compare the performance of different classifiers.