Heidy Stewart

Prof. Koehler

ECON-UB 232

12 December 2024

Final Project: "Minecraft Gameplay Decision Dataset with Screenshots" [2024]

The dataset I chose for this project comes from Kaggle created by Artyom Tashyan, Pavel Tin, and Saidjon Khaydar-Zade. This dataset is composed of screenshots from Minecraft gameplay along with a csv file with the features found within each screenshot. The features listed in the table of the csv file include: screenshot title, activity, hearts, light level, in hand item, target mob, decision activity, decision hearts, decision light, and decision mob. All of these features are categorical except for hearts. The activity column categorizes what the player is doing in the screenshot into one of six categories: archery, building, fighting, mining, swimming, and walking. The hearts column lists the number of hearts the player has in each screenshot. The light level column indicates if there is low, mid, or high lighting in the screenshot. The in hand item column lists what is in the player's hand in the screenshot; there are seven possible values for this: pickaxe, sword, axe, bow, crossbow, block, miscellaneous. The target mob feature describes which mob the player is interacting with in the screenshot, which includes the categories of: zombie, spider, skeleton, creeper, ender, and other. This dataset was created to develop and train AI models which can make decisions for the player based on the previously explained features, which are the contextual data of the game, along with the visual data. Because of this, the features which begin with 'decision' are tied to their respective features, and the decision is based on certain rules outlined by the creators. The decision activity column is based on the activity listed. If the activity is archery, then the decision is to give a resistance potion. If the activity is building, then the decision is to give a jump boost potion. If the activity is fighting, then the decision is to give a strength potion. If the activity is mining, then the decision is to give a haste potion. If the activity is swimming, then the decision is to give a water breathing potion. If the activity is walking, then the decision is to give a speed potion. For the decision hearts column, the decision is based on the number of hearts the player has in the screenshot. If the player has between 0 and 5 hearts, then the decision is to give a regeneration level 4 potion. If the player has 5 to 10 hearts, then the decision is to give a regeneration level 3 potion. If the player has 10 to 15 hearts, then the decision is to give a regeneration level 2 potion. If the player has 15 to 20 hearts, then the decision is to give a regeneration level 1 potion. The decision light column is based on the light level column. If the light level is high, then there is no decision. If the light level is mid or low, then the decision is to place a light source. The decision mob column is based on the target mob. If the target mob is a creeper, then the decision is to go back. If the target mob is a skeleton, then the decision is to take a bow if it is not in hand. If the target mob is a zombie, spider, or ender, then the decision is to take a sword if it is not in hand or if an axe is not in hand. It does seem, though, that sometimes even if there is an axe in hand, the decision is still to take a sword. I won't be using this column, which I will explain the reason why later on, so this discrepancy will not affect my model. If the target mob is labeled as other, then there is no decision. There is more information on the Kaggle page (https://www.kaggle.com/datasets/sqdartemy/minecraft-screenshots-dataset-with-features) which outlines the creators' rationale behind the rules and more. However, what I wish to achieve in the project is to make a model that can predict the activity based on the data given.

The data that I downloaded from Kaggle included 6,205 images, a mixture of both png's and jpg's, and one csv file with a table of the features from the screenshots. Before starting, I ensured that the csv file loaded into the notebook correctly by checking the table and that there was no missing data by checking the info. Everything was fine, so I started off building some models based on the table of features. Because there is co-correlation between columns, I decided not to use the columns that begin with 'decision' in the column title; this means that I did not use the decision activity, the decision hearts, the decision light, nor the decision mob columns. I also thought that the screenshot title column would be irrelevant as a predictor since it mainly serves as an identifier. My target, or y value, is 'activity,' so I was left with 'hearts,' 'light lvl,' 'in hand item,' and 'target mob' as my predictors, or x values. Before I moved on, I decided to check if the in hand item would automatically give away the activity, meaning if the in hand item was correlated to the activity. Multiple items were associated with multiple activities rather than just one, so I decided to keep the variable in my model. I did a train test split and created an encoder for the categorical variables. I decided to test out a random forest model and a logistic regression model to see which one achieves better accuracy. I used a pipeline to put the instantiated random forest classifier and logistic regression and encoder together. The pipelines were then fitted to the X train and y train data. I also established a baseline as predicting the most frequently occurring activity, archery, for every data point. Based on accuracy, the best performing model was the random forest. The baseline had an accuracy of 28.76%, the random forest had an accuracy of 93.96%, and the logistic regression had an accuracy of 91%. Both of my models were able to do better than the baseline. I wanted to see where my models stumbled by creating a confusion matrix. Both the random forest and the logistic regression models struggled with distinguishing between swimming and walking, though the logistic regression struggled much more than the random forest. The two models also struggled a bit with distinguishing between archery and fighting, but it wasn't as detrimental as the previously mentioned issue. Because the random forest model was the better performing one, I opted to do a grid search to see if I can achieve an even better random forest model with different parameters. I was able to achieve a slightly better model with the parameters of having a max depth of 'none' and n-estimators of 10. This model had an accuracy of 94.02%. Based on this new model, I was able to determine that the most important feature seemed to be the categories of the in hand item column.

After playing around with the data on the features, the table, I wanted to try working with the screenshots. I first made sure to figure out how to retrieve the images to see if they were accessible. After verifying that I could access them, I wanted to try doing some image classification using a neural network. For this part, I did rely on the Gemini AI for some help, as this was a bit different to the image classification we did in class. I started off by assigning the unique activity labels to a variable as these would be the categories the model would use to classify the screenshots. I had to make sure that the screenshots were organized into the activity category they belong to, so I created a directory with subfolders to store and organize the screenshots; this was followed by loading the images. Once all the organizing was done, it was time to build the model. I created a train and test dataset that contained 80% and 20% of the screenshots, respectively. I then created data loaders for the train and test data that would load the data in batches. Because images are being used, they need to be flattened and moved to the GPU. To make up the neural network, a loss function and optimizer need to be defined. After doing all this, it was time to put them all together and train the model. This model performed better than the baseline but worse than the random forest and logistic regression models. Even so, I believe that this model performed very well going off of just the screenshots. It is a very useful model that can avoid time spent on manually extracting what the features are of each screenshot. Because this model didn't perform as well as the other two, I decided to take a look at where it made mistakes using a confusion matrix. Since the model spits back the labels as numbers, I encoded the unique values of the activity column to see which activity aligned with which number. Based on the confusion matrix, most of the errors came from classifying an image as fighting when it actually belonged to a different category. Unlike the other two models, this one did not struggle with distinguishing between swimming and walking.

Based on these results, the random forest model was the best one at predicting the activity. This model didn't use all parts of the data provided. When I tried using all of the features, the models were pretty much perfect, but that is because the categories in some of the columns are tied to each other, including the target column. The neural network used the screenshots which were not used by the random forest model. I think if the neural network is combined with the random forest model, a better model can be created. The random forest model struggled distinguishing between two different categories which the neural network was able to overcome; however, the neural network struggled with one category which the random forest didn't struggle with. Because of this, I think that these models can help each other do better if combined. The neural network took a long time to run, so I was discouraged from testing out different optimizers which may have an impact on the accuracy of the neural network. Maybe using different parameters could achieve a better performing neural network. The original use for this data was to be able to develop an AI model which can perform actions based on other features. My models predicted the activity of the player based on the screenshots or features of the screenshots. By predicting the activity, the decision activity can be determined based on the rules outlined by the creators. I do not think that the random forest model I created would be as useful if I switched one of the predictors with the original target. Since the in hand item column was the most significant feature of the random forest model, the model may still do well if it is predicting the in hand item; however, I don't think that this would be the case for the other columns. I think the neural network would still be useful even if the target changed; it might even be better at classifying other columns as the target compared to trying to classify the activity. For example, if the target were changed to be the hearts column, I don't think the predictors of my random forest model would help but I do think that the neural network would be able to classify how many hearts there are since it is found within the screenshot. Therefore, I think optimizing the performance of the neural network model would be much more effective as a next step, even though the random forest model did better in this case of predicting the activity.

Source

"Minecraft Gameplay Decision Dataset with Screenshots," [2024]. Available at https://www.kaggle.com/datasets/sqdartemy/minecraft-screenshots-dataset-with-features