Chapter 4

Implementation

**Collecting the data**

The Scratch Online Community has made the dataset for scratch platform publicly available on [15]. The dataset has the data for the first five years of the data from Scratch (approximately from 2007-2012). The data was collected from the MySQL database from the Scratch website. All datasets are provided in CSV file formats, which can be loaded into any compatible data analysis software. There are a total of 32 datasets which contain information about various aspects of the Scratch Online Community. These 32 datasets are divided into 3 main categories *Core Datasets, Text* and *Code Datasets* and *Project Analytics Datasets.*

Core dataset consists of major relationships and objects which are captured by the Scratch Online Community application.

Text and Code Datasets consists of very large tables consisting of texts submitted by the users. This is helpful for NLP and text-based analysis.

Project Analytics Datasets contains tables which have quantitative summaries of each project file. A project file is a .sb2 or .sb3 file that contains all of the content of a scratch project.

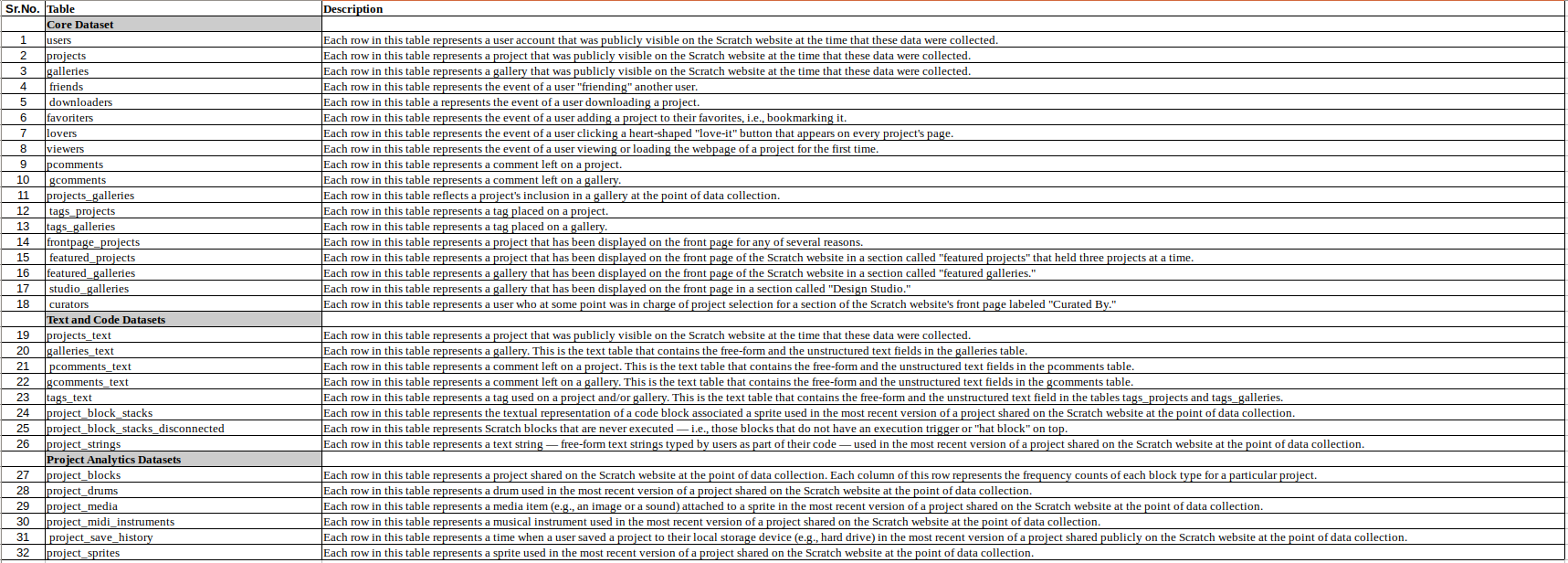


Table 1: Description of each dataset from the Scratch Online Community website

**Access to the Data**

The dataset that this research uses is the data from the Scratch Online Community. It has been supplied by the Lifelong Kindergarten Group at the MIT Media Lab under public domain. Although this data is available under public domain, a Scratch Research Data Sharing Agreement (SRDSA) [38] was signed to obtain access to the dataset files.

The users of the Scratch online platform are mostly young children many below 18 years of age, from all around the globe. Therefore an ethics approval was required by the LLK group from the MIT Committee on the Use of Humans as Experimental Subjects (COUHES). Which was granted to the team and the data was published under the specified protocol by (COUHES) MIT.

All the terms of the access to the data have been adhered to and therefore as the part of this dissertation, no ethics approval was required from the university.

**Identification of the relevant dataset**

Keeping the objective in mind of building a recommender system that would recommend “projects” to scratch users, first we need to identify the relevant data for our goal form this dataset.

For recommending projects based on user activity, we require the data related to projects and users on the Scratch platform. Looking at the list of the datasets we see that projects.csv, projects\_blocks.csv, projects\_sprite.csv and users.csv have most of the data related to users and projects.

* **Users** - holds the data about the user and which project the user has worked on.
* **Projects** - contains general information about the projects and the user associated with each public project.
* **Projects\_block** - holds the information about each block (the logical component of the code) used in a project, these are the extracts of the sb2 and sb3 files.
* **Projects\_sprite** - contains the information about sprite (scripts, sounds and images) used in each project.

Apart from these datasets, the following datasets were also used to obtain information required for user rating calculation.

* **Lovers -** which has information about the projects that the user has clicked “Love” for.
* **Friends -** contains each user and its friends ( the user who follow each other).
* **Favourites -** shows an event where a user has added a project to their Favourites.

**Data Analysis**

After identification of the data required, we apply some preliminary data analysis techniques to check for consistency and quality.

**Projects -** We see that the more of the projects are created in the latter half of the dataset time frame i.e. after 2010. This shows the popularity of the scatch in the initial phases.

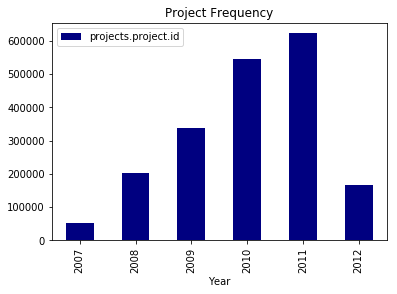
****

Fig 2: showing the distribution of projects in dataset over the period of 5 years.

**Project blocks -**

There are about 171 of these key block attributes associated with each project in this dataset.Table 2 shows each of this block. Figure 3 below shows the frequency of each block used in the projects. We see an even distribution of the most common operators in the projects, while the less common once are not used frequently. We can assume from this distribution that project dataset is good enough for finding the correlations between projects.

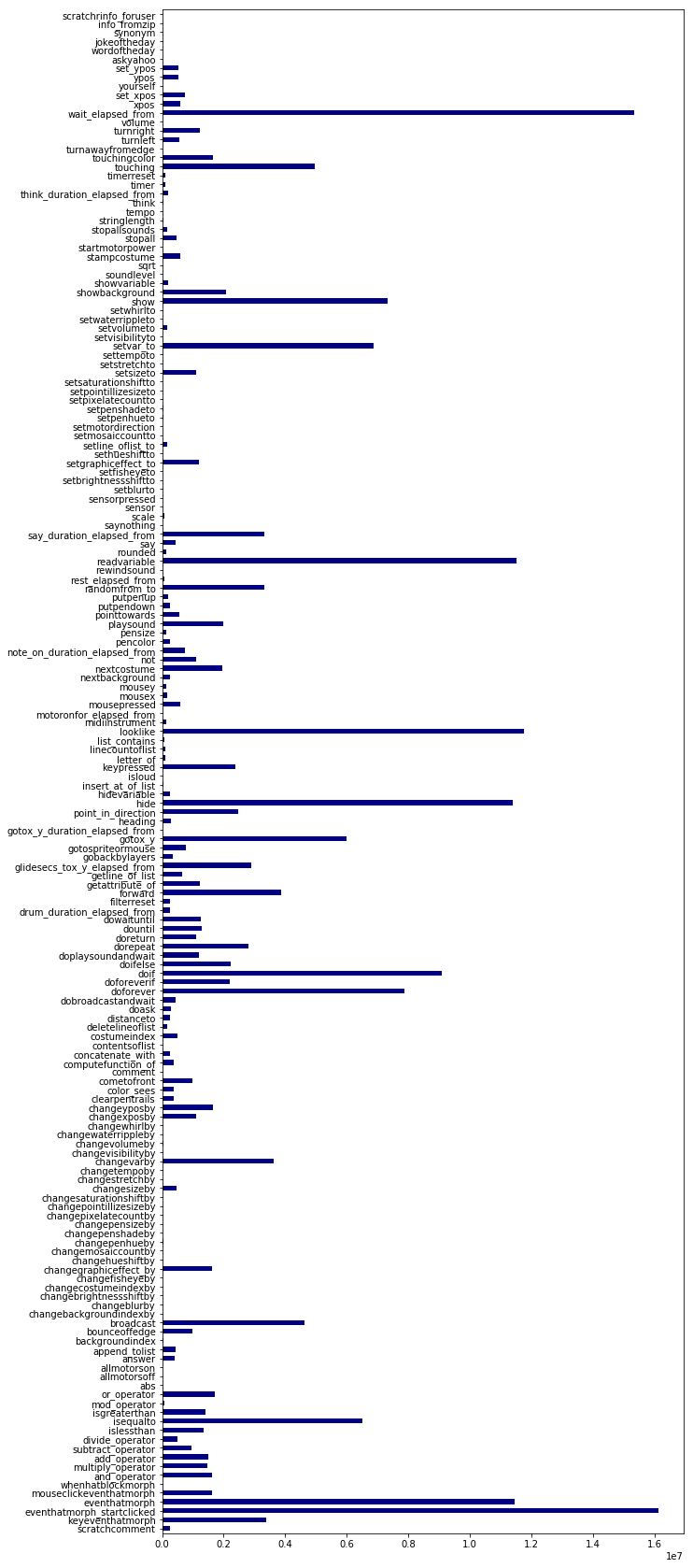


Fig 3 : Showing the frequency of the use of project blocks



**Users -** The following figure 5 shows the distribution for the accounts in scratch done by creation date in the period of 5 years.

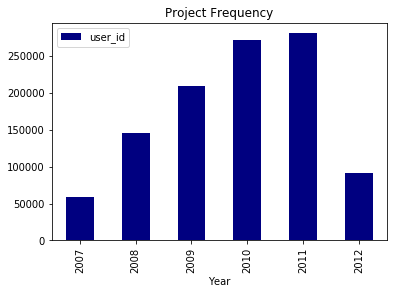


Figure 5; distribution of the user accounts created by creation date.

**Friends -** An event where a user adds another user as a friend on the Scratch community website is recorded in this dataset. The Fig 6 shows the distribution of this relationship over time.

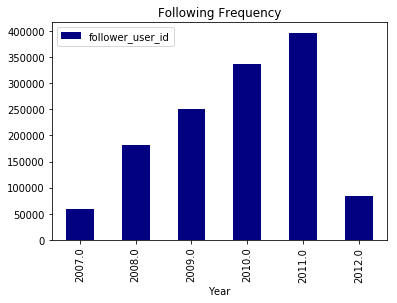


Figure 6; distribution of the friends over time.

**Calculating similarity matrix**

After the data analysis, we will apply the three similarity computation matrix discussed in chapter 3 on our projects data set.

All the programming and computation is done in python as the programming language. Various packages and libraries have been used to facilitate the easy application of various algorithms. Pythons *sci-kit learn (sklearn)*[39] library, which houses many machine learning packages has been used for applying the cosine- correlation and adjusted cosine correlation methods. Another library known as *Pandas* [40]has been used for computing the pearson correlation matrix.

**Preparing the data:**

Since our project\_sprite and project\_blocks both contains some information about the project components we can merge the two datasets to form single dataset.This has two benefits first it reduces on the process of applying computation on two different datasets and secondly it gives us 3 more parameters(sounds, images and scripts) from project\_sprite which will be beneficial in finding the correlation.

|  |
| --- |
| *#Load projects data from csv* projects\_df = pd.read\_csv('data/CSVs/project\_blocks.csv', sep=',',index\_col=0,)  *#Imputing the others data since it has mostly the null values.* projects\_df = projects\_df.drop(columns="other") *#Load projects\_sprites data from csv* project\_sprites\_df = pd.read\_csv('data/CSVs/project\_sprites.csv', sep=',',index\_col=0)  *#Merge both the panda dataframes on project\_id* main\_frame = pd.merge(projects\_df, galleries\_df, left\_index=True, right\_index=True) |

Fig: code snippets

**Imputing missing data**

As we have seen in our previous data analysis section, there are some missing values in our dataset. For our algorithm to work correctly we need to replace this missing data. For replacing the value we need to apply some data imputation techniques which estimates the best possible replacement by looking at neighbouring values.

**Mean imputation -**  In the mean imputation methods, the missing values are replaced by the mean of the observed variable. For example, if the variable of sound ( number of sounds used in the project) is missing for a particular project, we take the mean of all the projects who have sound in them and replace the missing value with it.

**KNN imputation -** In K nearest neighbour imputation the missing values are replaced by the approximate value based on the nearest points. The assumption being that the given K nearest neighbour point the missing value is an approximation of it ,k being a number of closest points to look for and is supplied by the user it is the most commonly used technique.

|  |
| --- |
| *#Impute the missing values using IterativeImputer from sklearn*  MAX\_ITER = 10 imp = SimpleImputer(missing\_values=np.nan, strategy='mean') imp = IterativeImputer(max\_iter=MAX\_ITER,initial\_strategy='knn', random\_state=0)  imputed\_DF = pd.DataFrame(imp.fit\_transform(main\_frame))  imputed\_DF.columns = required\_columns\_df.columns  imputed\_DF.index = required\_columns\_df.index |

For our purpose we use SimpleImputer and IterativeImputer from *sklearn* for imputing the data values.

**Correlation-based similarity**

For correlation-based similarity computation, pearson-r correlation method is applied. The corr*(method='pearson')* methods from the Pandas directly applies the selected correlation method on the data set.

After applying the correlation we get following out put

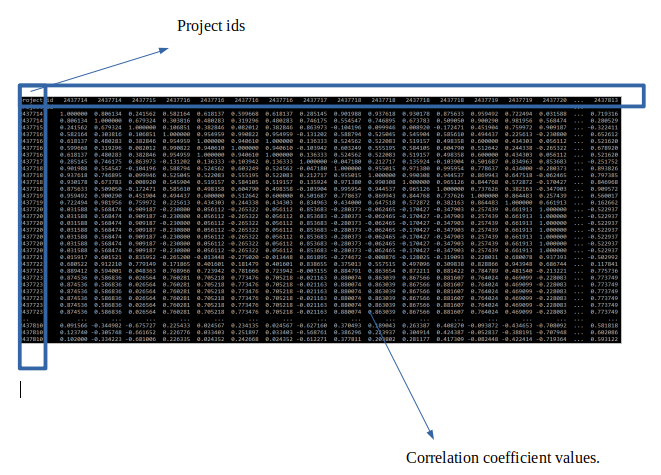


Fig: Pearson correlation output.

**Cosine-based similarity**

For cosine based similarity computation we apply the cosine\_correlation from the *sklearn.metrics.pairwise* module on the dataset.

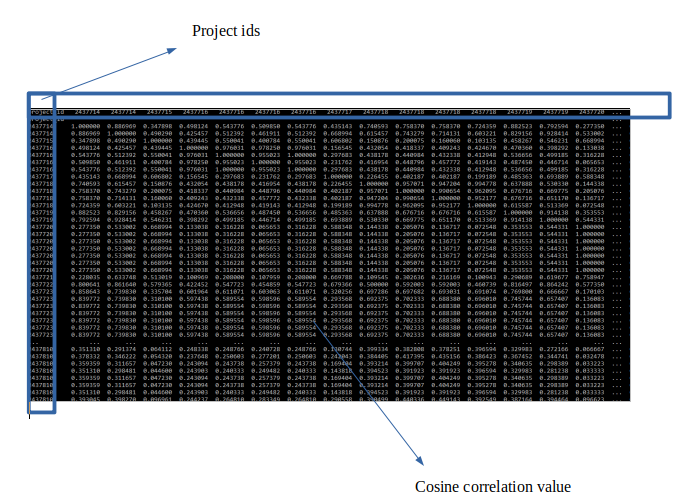


Fig: Cosine similarity computation output.

**Adjusted-cosine based similarity**

In the same way as above, we apply the adjusted cosine function from the sklearn library to obtain the output.

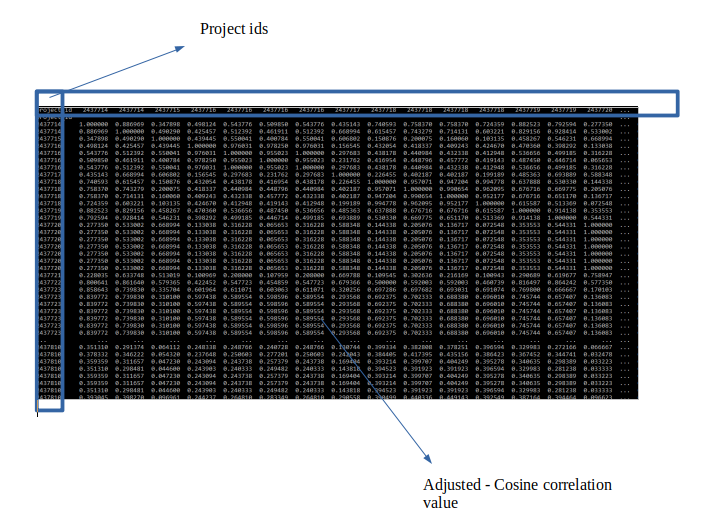


Fig: adjusted cosine similarity output

As we can observe from the results above the pearson-correlation gives us many negative correlations.

**Selecting top-N recommendations**

From the results of the similarity , for an item i we get the correlation score for each of the items from our dataset.

We select the top - N values from the matrix (N=5) to be presented to the user.

|  |
| --- |
| *#Get Top 5 from the similarity matrix* top5 = final\_df.loc[project\_id].nlargest(5) recommendation\_pairs = top5.to\_dict() |

**Prediction Calculation**

After we have the item-item correlation matrix ready, we proceed to calculate the prediction score for each of the item based on users activity.

**Assigning the rating score for user activities**

In a conventional personalised recommendation, a users feedback is captured through the likes and the rating given to an item. For example in case of netflix a user rates a movie on a scale of 1 to 5, 1 being the lowest and 5 being the highest. Some times decimal ratings are also allowed.

Since there is no rating system in scratch and a user can either Like, favorite or remix a project and these three details are publicly available we will use this information to develop a rating system. While viewing a project , a user has a choice of performing various actions on the project page. He/She can either click on Love/Like to show the appreciation or add the project to his favorites or click on remix to copy the project to his own profile and adapt on its code. User can perform one or two or all three of this activities.

For each of the user activity (love, favorite and remix) we assign a rating score of 1 if the user has performed the activity or 0 if not. We then take the sum of all three activities to find a user rating on a scale of 3. So for example, if a user has liked and favorited a project but not remixed the total user rating is **love (1) + favourite (1) + remixed (0) = 2**. And If a user has only remixed the project then the total rating for that project is **love (0) + favourite (0) + remixed (1) = 1.**

Thus in our weighted sum method we use this rating to find the most relevant projects for the user.

**Weighted Sum method**

Weighted sum method has been used for prediction calculation because of its simplicity to implement. For an item I for user U, this method calculates the prediction score by evaluating the sum of the ratings given by the user on other similar items to item I.Each rating by the user is assigned a weight si,j based on the similarity output of the item.

|  |
| --- |
| *#Calculates the prediction score for a user given recommendation as the list of similar recommended projects and target project\_id* def calculate\_prediction\_score(user\_id,project\_id,similar\_recommendation):  \_w = [get\_user\_ratings(user\_id)]  \_x = similar\_recommendation  score = sum( \_x \* \_w for \_x, \_w in zip( x, w ) )/ calculate\_score(\_x)  return score |

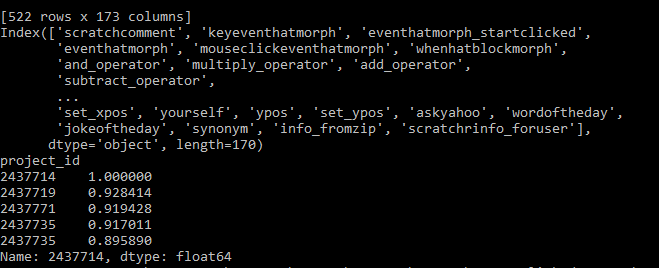


Fig:Top 5 project ids according to prediction

The figure shows the top 5 most similar items computed after applying the weighted sum method.

For the supplied user\_id: xxxxxxx and project\_id:2437714 all the other are the prediction scores for the top 5 similar items.

**Comparing the recommended items**

On comparing the results that we get from the recommendation method, we observe the following

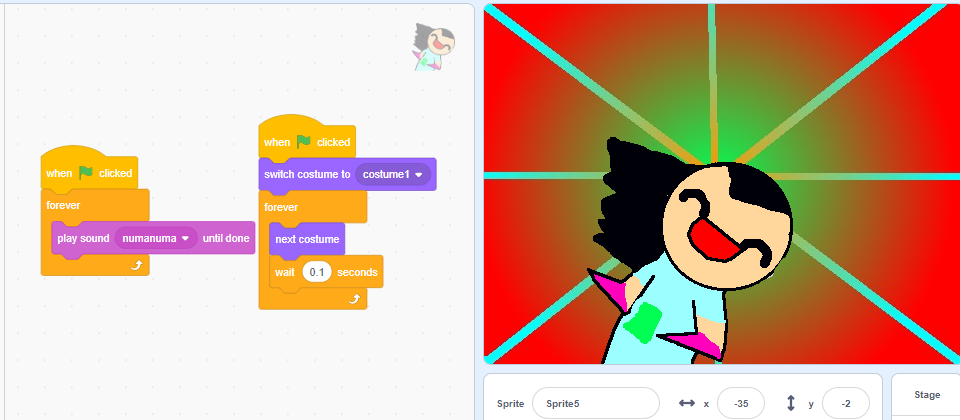
Every project comprises of a similar coding blocks as of the original user project.

Every project has one forever loop, one wait timer one costume switch.

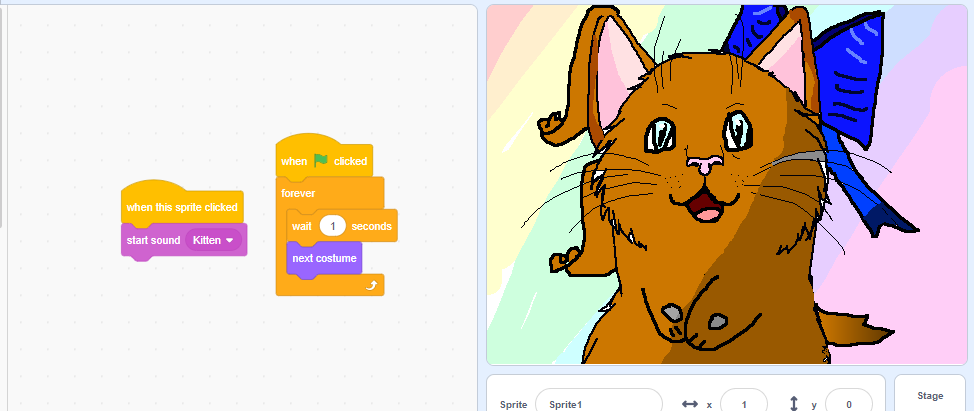
Also, note the the two most similar projects have the sound block explicitly specified and while the next two have an implicit sound build in requiring no sound block.

Also, note that second-most similar project is given priority because our test user has liked and remixed it.

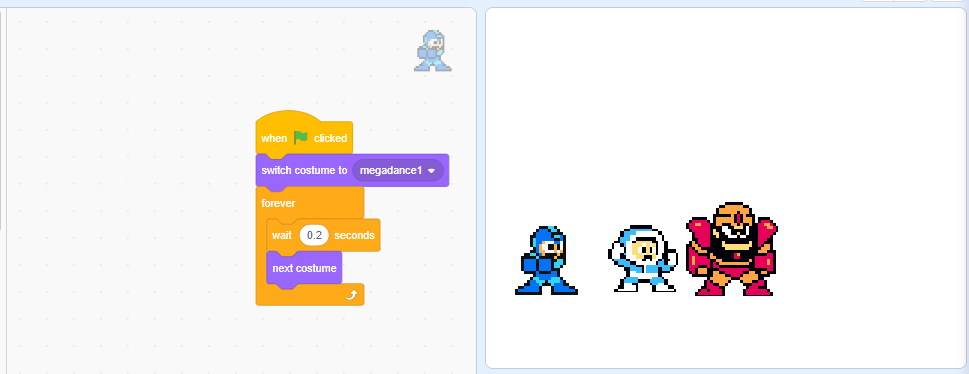
<https://scratch.mit.edu/projects/2437714/>



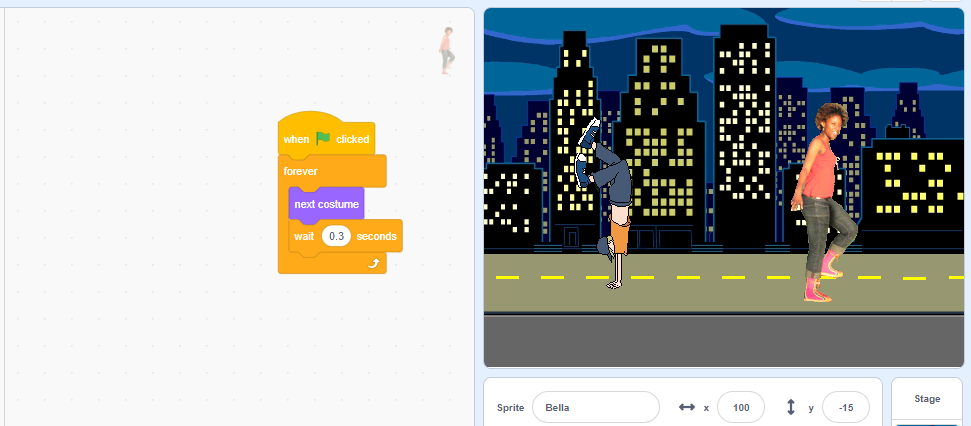
<https://scratch.mit.edu/projects/2437719/>



<https://scratch.mit.edu/projects/2437771/>



<https://scratch.mit.edu/projects/2437735/>



For evaluating the quality of this recommendations and prediction scores we will use more formal method.