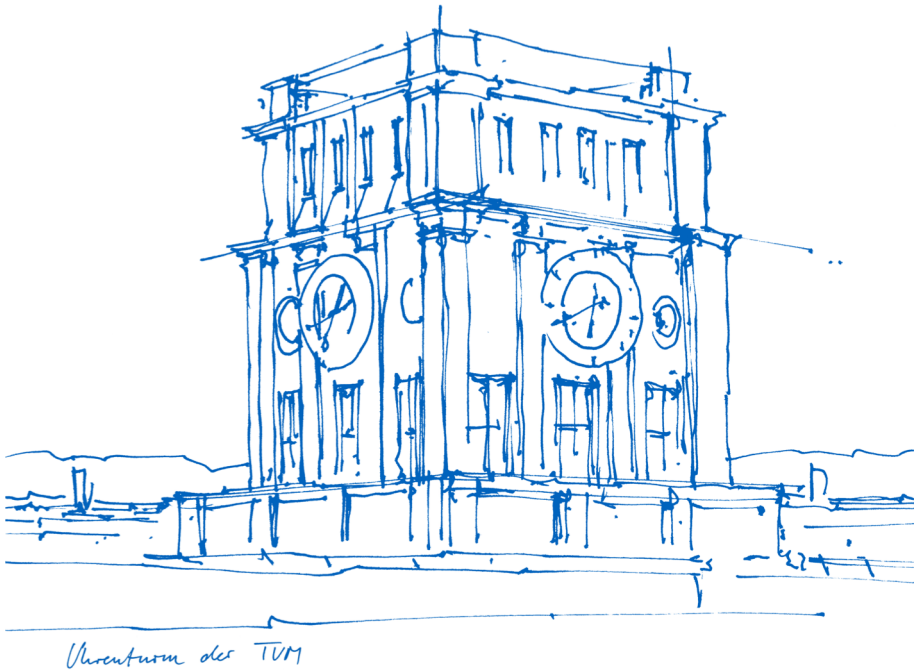
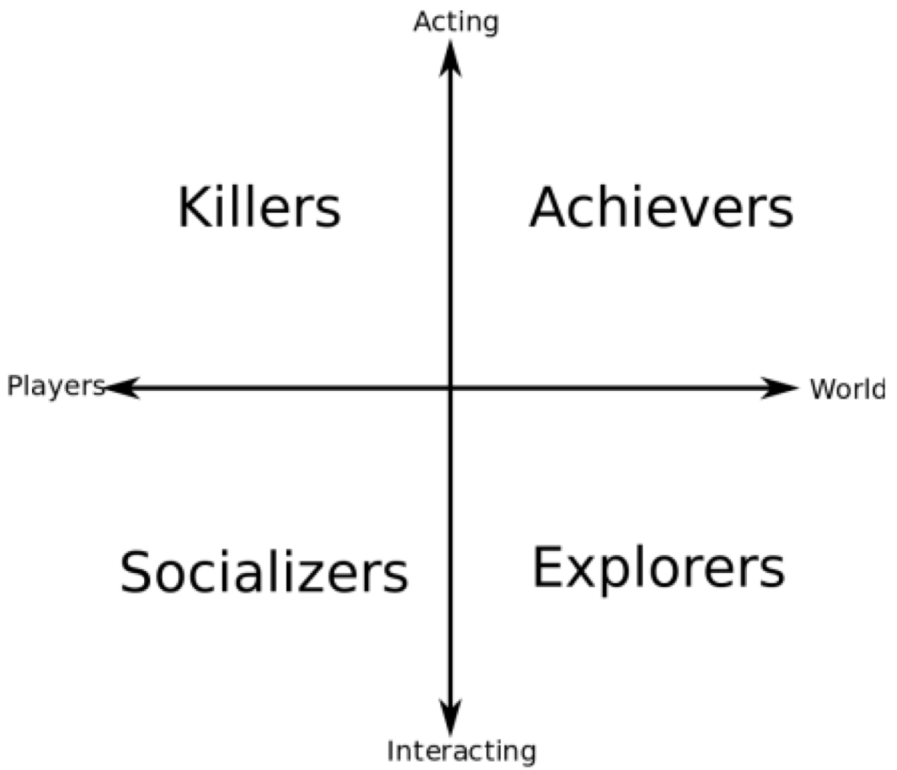
Exercises for
Social Gaming and
Social Computing
(IN2241 + IN0040) –
Exercise Sheet 3 -
Clustering with EVE



Exercise Sheet 3: Clustering with EVE

- goal: revise K-means clustering algorithm
- process data gathered from the sandbox MMO EVE Online
- cluster players into 4 groups (Radoff):
  - Sozializers
  - Achievers
  - Explorers
  - Killers



The Data: EvePlayerStats.csv

- *EVEPlayerStats.csv*:
  - *soloRatio*: nr. of kills without assitance of others
  - *secStatus*: measure of criminal activity
  - *shipsDestroyed*: nr. of total destroyed ships
  - *combatShipsLost*: nr. of lost combat ships
  - *miningShipsLost*: nr. of lost minings ships
  - *exploShipsLost*: nr. of lost explorations ships
  - *otherShipsLost*: nr. of lost ships other than the aforementioned

K-Means

- K-Means: objective function (see lecture):

$$J(\mu) = \sum_{k=1}^{K} \sum_{\{n|x_n \in C_k\}} ||x_n - \mu_k||^2$$

where $\mu_k \in \mathbb{R}^d$ and $x_n \in \mathbb{R}^d$

Task 3.1: Preparation

Now that you are armed with all the knowledge needed, let us begin.

a) First, **read** the .csv that you downloaded with this exercise into a dataframe.

b) Then **drop** the `Unnamed: 0` and `characterID` columns. We don't need them.

Task 3.2: Normalizing & Clustering

As you might have seen, the value ranges differ greatly across the metrics. Where the number of kills can reach up to 10.000, the security status rarely exceeds 5. This creates an imbalance, as the calculation of distance will obviously be impacted a lot more by kill counts than security status. To rectify this and let all metrics influence the result in an equal manner, we need to normalize the data.

a) Go through the dataframe and normalize all values to a [0,1] range

b) Convert the normalized absolute numbers into ratios. Divide the absolute number by the number of all ships lost by a player, this will give you the ratio.

c) Cluster the dataset with the k-Means algorithm and print out the centroids.

Hint: For the clustering we will use the k-means algorithm provided by the scikit-learn library. Import the algorithm and use the `fit()` function to let the algorithm do its work. Remember to set the amount of clusters to 4.

Task 3.3: Analyzing the results

a) Heatmap

Since we have 7 features for each player in total, our datapoints lie in a 7-dimensional space. It can be tricky to read a 7-dimensional graph, so we will first use a heatmap to analyze our data. A heatmap is a data visualization technique that shows the data as color in two dimensions.

1. Use the seaborn library to **generate a heatmap**. For readability purposes, **display the last 20 players** from the processed dataset **only**.
Hint: If you feel like the graph is too small, scale it up a bit.

2. From these 20 entries, **choose 4** that you think are the most representative for each of Bartle's player groups (one for each group) and **briefly explain** why you chose them based on the heatmap.

b) t-SNE

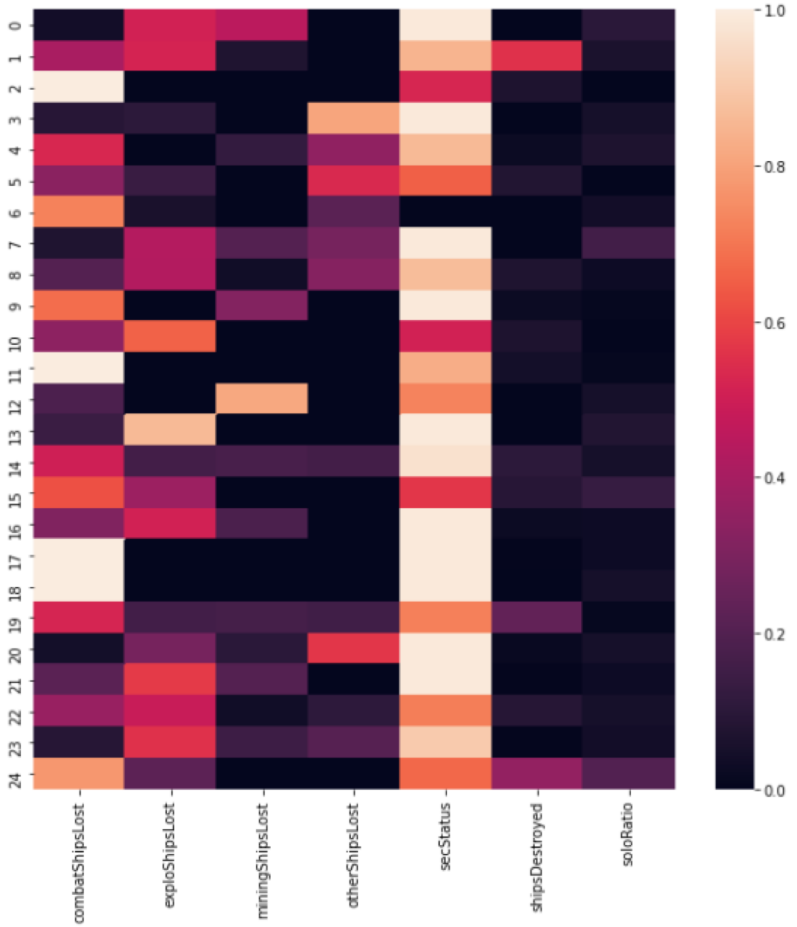
Heatmaps are nice, but if we want to display large amounts of data, they become unreadable. Therefore, we introduce an algorithm called t-SNE [3] that can transform a high dimensional dataset into a 2 dimensional plot. For more information you can check out the linked paper. For a simple but intuitive explanation have a look at this video [4].

1. Run the given code to generate a t-SNE graph. Look at the plot and choose **one** cluster which you want to analyze. From this cluster **choose 2-3 players**, analyze their stats and describe your observations. Can you tell what kind of player type the cluster represents in Bartle's model? Can you explain the meaning of the distance between the clusters?
Hint: You can see the assigned clusters for each player with the list `kmeans.labels_`

Note: If you get the impression that the clustering is not very accurate do not feel discouraged as the data set does not contain enough information about the other activities of the players besides ship killing.

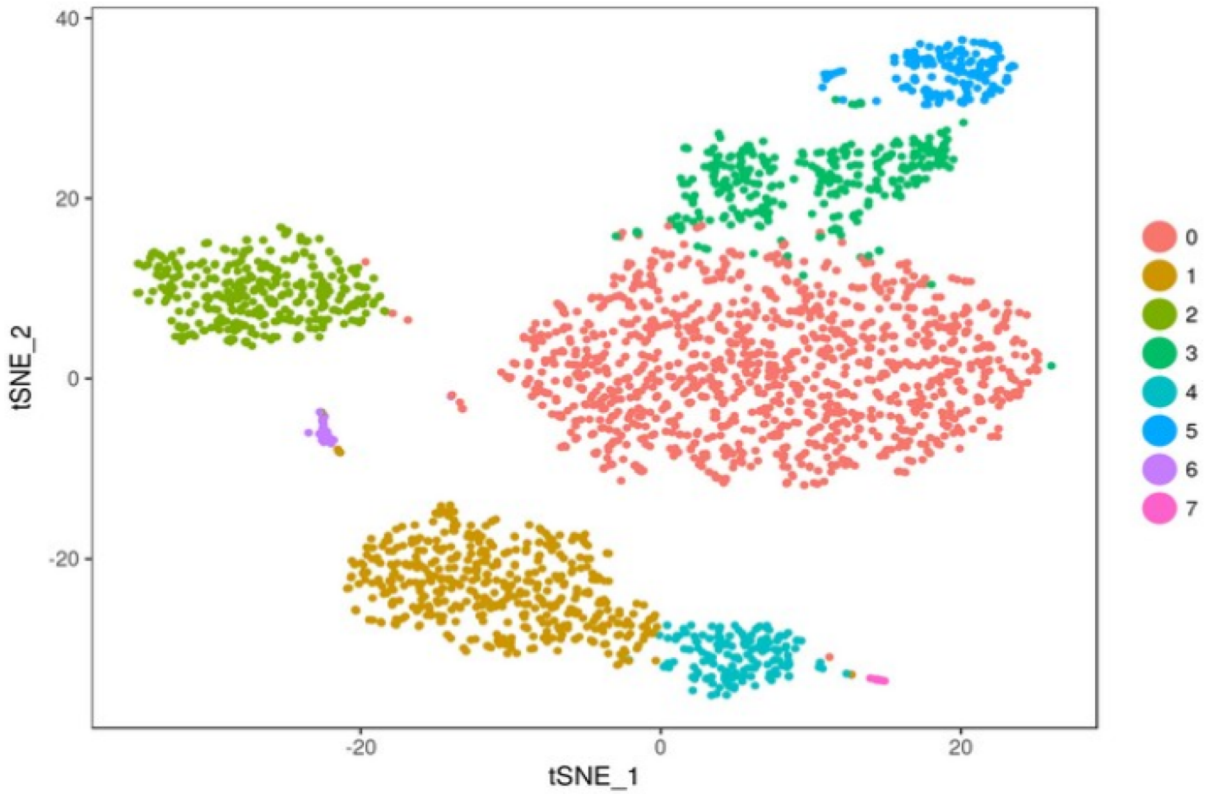
Heatmap

- heatmap = data visualization technique that reduced the dimensionality of the data
- displays the data as color in two dimensions



t-SNE[1]

- designed to plot multidimensional data in a 2 dimensional grid
- helps to visualize data and identify clusters and similarities between clusters



DBSCAN

- DBSCAN: (see lecture):

Rough idea: iterate:
visit previously unseen pattern x:
if in ε-neighborhood {x'} of x: |{x'}|≥ minPt then
start new cluster: include x and {x'} and those of their ε-neighborhoods {x''} that are dense enough (|{x''}|≥ minPt), etc.
else: x is noise

Gaussian Mixture Model (GMM)

- GMM: (see lecture):

  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

parameters to be estimated

b) DBSCAN and Gaussian Mixture

Next look at the Clustering with DBSCAN and Gaussian Mixture Model. Compare these two and Kmeans and rite down your Observations.

Tips and Tricks

- before you start coding, familiarize yourself with the data
- you don't have to code K-Means yourself, you can make use of libary functions
- the first code cell is only explanation for the data and thus commented out. Do not run this Code!
    - you can analyze it, but there is no reason to run it

Submitting your solution

- work by expanding the .ipynb iPython notebook for the exercise that you downloaded from Moodle
- save your expanded .ipynb iPython notebook in your working directory
- submit your .ipynb iPython notebook via Moodle (nothing else)
- remember: working in groups is not permitted. Each student must submit their own .ipynb notebook!
- we check for plagiarism. Each detected case will be graded with 5.0 for the whole exercise
- deadline: check Moodle

Citations

(1) Laurens van der Maaten, Geoffrey Hinton: Visualizing Data using t-SNE, 2008 (PDF)