**Text Based Analysis using Movie descriptions and genres**

**Team Name:**

**‘Allo ‘Allo World**

**Team members:**

**Christopher Ewart – W18012997**

**Matthew Woolley – W17016042**

**Christian Swan – W18018793**

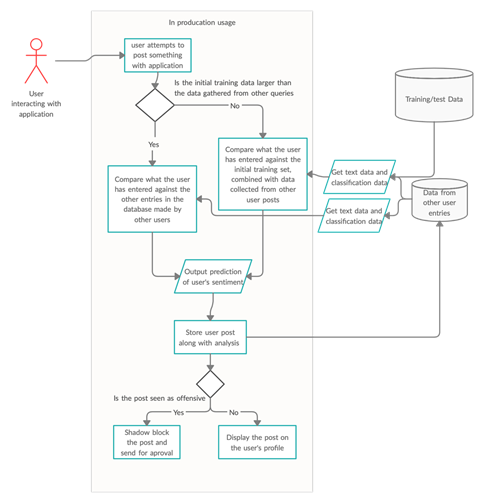
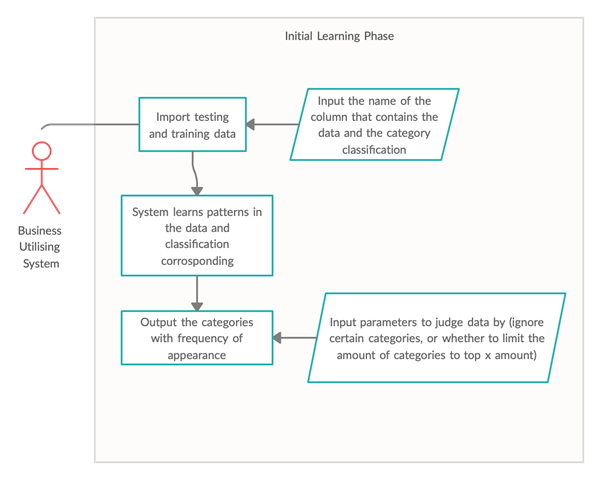
# Project Ideation - Mission owner: Chris S –

During several pre-covid19 discussions we mulled over what project would be within our capabilities and realistic. Prior to our team being a three, we had another member who was championing the idea of a medical system for diagnosis, however this route was ruled out due to none of us having any experience in that field and the complexities it could entail. Our team managed to settle on a solution which will address classification of text. Knowing that consolidating search results on keywords or filters is a widely adopted function of websites, there should be an abundance of material to review on the subject, to see where text classification has been used and what some of the issues were and to understand why it is a beneficial thing to do.

## Supporting Docs –

Project Ideation

# Solution Design – Mission owner: Matt W –

For the solution design documentation, the focus was primarily on the two main use cases that were decided on being the main consumer areas for the project. The first diagram shows a company who would be using the system and initially setting it up. This could be a company such as a sales company that wishes to filter their reviews. This shows an initial learning phase as well as some settings that a business could alter in order to customise the system to their liking. The second diagram shows how a business, after the initial learning phase, may use the system to filter comments or reviews on their website and contains some building blocks that were thought to be the most efficient way to improve the way the system categorised as well as making the data categorisation more relevant to the data the business realistically receives.

In order for the system to be used in a production environment, there is a small “learning” phase that is planned to happen. During this phase, the business who would be using the potential system, would first have to import their own training and testing data. During this import, the user will then be asked which column contains the text data (the data that will be analysed) and the classification/label data. Once this has been extracted, and the required algorithms have analysed the data, the user will then be shown a list of the categories as well as how often they appear in the data and asked if they would like to remove any of them from the training (for example, if reviewing movie genres, and there were only 2 of one specific category); the user will also be asked if they want to limit the categories to the top X amount, which would then only return the top X most frequent categories for the system to be trained on (for example, this may help the system be more accurate on things that are currently trending, possibly improving efficiency)

In the second example, a system that could be used for social media is shown. Initially, to start the system, a user will attempt to post something with the application; the system is only planned to analyse the text of a post so it would be limited to text based posts. Once the user has submitted their post, the system will take a tally of the training data and the collected user data so that it can decide which dataset should be used for predicting the response. If there is more test data than user data, the system will default to using the test data to analyse the text, however, we also plan on adding actual user data as well so that it has more examples. If there is more user data than test data, the system should start moving completely to the collected user data. Once the data has been pulled from the database and a model is created, the system will then perform a prediction based on what the user has input. This point would be standard for any program; however, the next portion is more specific to a social media website; the prediction would return the category that shows the most similarities to the text submitted by the user. This example is just showing how the system would operate if it were attempting to detect something that may be seen as offensive. If the system determines that the content posted is offensive, the application may place a shadow ban on the post (not technically a ban, but not showing the content either) and sends it for review; reviewing content that has been flagged allows the system to remove false positives, while allowing users to flag content that has gone through the filter helps the system learn false negatives too. If the content posted does not trigger the filter, it will be posted to the application.

## Supporting Docs

Solution Design

# Baseline Implementation – Mission owner: Chris E –

## Evaluation

The Base line is a proof of concept. It deals with the main aspects of a machine learning model. It has the following:-

**Dataset** - The dataset was manually cleansed first in excel (something I use a lot at work as an analyst, so used it for ease) to take out any of the genes that had more that one genre in it as well as take out any obscure genes (out of the thousands of lines there were some genres that only appeared a few times). This was to ensure the baseline could work as a foundation for the project and not have basic data cleansing as an issue.

**Data pre-processing** - it extracts the columns that are needed (x and Y, or data and labels). It also removes any blanks in the data

**Separates training and test data** – the baseline takes a random 80% of data for training and the rest (20%) for testing

**Sets up the classifier** – IT sets up a text pipeline

**Fits the model** – It trains the model using the pre set up pipeline

**Output accuracy** – accuracy for the baseline is 47.8%

The baseline works well as a baseline. It has a basic functionally of all of the above features. There is plenty of room to improve the model. This could be done by wrapping up some of the functionality in functions. The pipeline can be fine tuned by tweaking the hyperparameters within the pipeline. At first glance the accuracy felt very low (sub 50%) however, as there are 14 classes, if the model gets it correct about half the time I felt this is good but there was still room for improvement. The completed Baseline left room for the following to be improved upon and developed as part of the iteration mission:-

**Wrapping up functionality into functions -**  with a view to using the passed in parameters to allow flexibility and different data to be used

**Better data pre-processing** – Adding functions that allow easier pre processing and different date to be added to the model (i.e. data that has the X and Y columns in different locations to the baseline data)

**Improving the accuracy score** – This could be done by changing the hyper parameters or even using a different pipeline

Adding a predictor – The baseline only has an accuracy score; it does not have any functionality to input some unseen text and then output the predated class.

As well as the above the baseline provided an ideal platform for us as a team to develop and improve in a number of ways.

## Supporting Docs –

Baseline code in folder (and also on our GIT hub account)

# Solution Testing – Mission owner: Matt W –

In order to complete the solution testing, a list was compiled of all the requirements set out for the program that were intended to be completed in the “Run 1” phase; this list contained 6 elements as well as the initial baseline implementation. These requirements were then split in 2 sections, the first being focussed on the implementation of the method and the second focussing on how the method performed. In the first stage of testing, the methods were judged based on how they went about solving issues, for example, data loading and column fetching were linked by a single method which made sense as they both were required to run once at the start, however, this was judged to be less modular, and as such a recommendation to split the methods was placed forward. After the first stage was complete and all methods were judged according to their implementation a testing method was then created.

Some methods were altered in order to fit with the testing method, for example, instead of asking for input, the program just assumed the input was a certain value; this allowed the testing to run without any further user input. For the second phase of testing, the implementations were then tested based on their output and how they handle errors or incorrect input. These methods were intended to supplement the original phase of methods, as the initial phase may discover inefficiency or poorly implemented code, whereas this set of tests was intended to find the code that contained poorly implemented logic.

These tests were run both after the first run and the second run in order to find any issues that may have persisted or been introduced after each development cycle. The first phase of testing helped the team decide on making the code more modular and more open, and the second phase of testing helped uncover issues with the out of bound exceptions faced when incorrect information was provided that caused the program to crash.

## Supporting Docs

Testing Run 1 (2 versions)

Testing Run 2

methods.py (starting at the section marked “TESTING”)

# Project management – Mission owner: Chris E –

For the Project management mission the follow documents were used to keep track of tasks, progress and the assignment as a whole.

## Documents

**Code of conduct** – This ensured all team members understood what was expected of them throughout. There are three versions as amendments were made to the meeting times and methods of communication due to the university closing because of the COVID-19 pandemic

**Risk analysi**s – This document details possible risks and the actions we will take as a team to first mitigate the risk of it happening. It also identifies actions we will take if the identified risk does happen. The analysis also summarises the risk probability and severity. The document also has a second tab that has a second tab called “Risk Matrix”. This tab defines what each risk probability and servery level means.

**Gantt Chart** – This was used to map out what our overall plan was with regards what parts of the assignment would be done when.

**Meeting minutes** – Minutes were kept for each meeting we have (mainly on a Friday). This covers what was discussed in each meeting. Any actions from the meetings were logged on the actions log

**Action Log** - The action log is a list of actions with target dates, owners, task status etc. It is an

basic ideas sheet, missions. It is a spreadsheet with some auto functionality that auto updates the status when overdue or if there is no due date etc.

## Communication

This was done inline with the code of conduct documents. We used “WhatsApp” at first then moved to mainly “Discord” as a member of the team had issues with “WhatsApp” part way through the project. See Appendix 1 for screen shots of communication evidence.

## Supporting Docs

Code of conduct (3 versions)

Risk Analysis

Gantt Chart

Meeting Minutes (7 word docs in folder called “Meeting Minutes”)

Action Log

# Additional Data – Mission owner: Chris S -

Chris’ documentation

Maybe a walk through

# Iterative Development

For the Iterative development mission we advanced our baseline and made improvements to it. We planned in two runs with each run involving a requirements gathering, time to make the amendment on the program and time to test/review. In reality the second run was a slimmed down version focused on fixing some of the recommendations from the testing missions. The reason the second runs was a slimmed down version was due to timings and a few delays in getting some of the task done. As these delays were close to the end of the project it made more sense to do this as there was not more time to adjust the Gantt chart.

Below is a summary of the changes made in each run

## Run 1

**Error handle on data loading** – This is a function that handles any errors that might be caused when loading in the data.

**Output chosen columns based on INTs**– A function that takes in a matrix and two integers and returns two vector variables based on the Column numbers of the matrix passed into it.

**Output chosen columns based on STRINGs**– A function that takes in a matrix and two integers and returns two vector variables based on the Column strings of the matrix passed into it.

**GridSearchCV function** – Introduced a GridSearchCV function to cycle through the hyper parameters and improve accuracy.

**Predict function** – Introduced a function that took in a string of a movie description and returned a genre as a prediction.

**Print out categories and count them** – A function that takes in a string and a dataframe and returns the categories and number of them.

## Run 2

**Separate functions** – 3 of the function form run 1 were put into the same function we separated them

**Add a “please wait” message** – The Grid search took a few minutes to run so we added a “please wait” message to it.

## Supporting Docs

Requirements

Run 1 Code (and also on our GitHub account)

# Appendix 1 – Communication evidence

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated