Adam Kritz

DATS 6103 – Jafari

Group 2 Final Project: Individual Report

Our group's goal was to create a model that could predict the IMDb score of a movie. In order to achieve this goal, we broke up the work into easily digestible parts. We planned to have five general parts: data download, preprocessing, exploratory data analysis (EDA), modelling, and graphical user interface (GUI) creation. I worked primarily on the data download and GUI section. I also helped with some of the EDA.

Beginning with the data download, this section caused the most issues. Initially, I planned to download the data directly from Kaggle from within the Python script. However, I almost immediately ran into issues. Kaggle requires a username and password to download data, so it is difficult to do directly from Python. I first tried to download the data by creating a user login in the code and using that to login through Python. However, Kaggle often rejected this method, and it also involved including my username and password in the code. I then tried to use a key to download the data. Kaggle lets each user generate a key in a JSON file to download data. I attempted to generate the key in a JSON file on the user's computer in the code, but this again caused issues, as the key was supposed to be in a specific place on the computer, which could not always be accessed. Eventually, I gave up on downloading data directly from Kaggle.

I then turned to hosting the data on GitHub. Since the Kaggle data is public domain, there were no issues hosting it elsewhere. This initially seemed promising, however we had a massive amount of data to host, which was too large for GitHub even when compressed. Instead, I decided to host the data publicly on Google Drive. This worked the best out of any method. My code directly reads in all of our data through Pandas using the google drive share link. This method also allowed us to host any other large files we produced in our research on Google Drive.

The main portion of my work was done in the GUI. I spent my first several days of work going through PyQt5 tutorials online to learn how to create a GUI. I then made a couple of sample GUIs by altering code I had found online. After this, I wrote out a plan for creating the GUI, including how I wanted it to look.

As each of my group members finished their sections of work, I was able to add it to the GUI. In all the sections of the GUI, I used QGroupBox to display information, usually with a QVBoxLayout or QGridLayout. I also used QAction buttons from the main menu to access different sections.

I first added information on all the EDA and preprocessing we had done. I divided up the EDA and preprocessing section into numerical variables, categorical variables, and our target variable. For the numerical variables and categorical variables, I used QRadioButtons to allow users to switch between variables. I then used QLabel to include a short description for each variable. Lastly, I included one to two plots for each variable using the FigureCanvas function. These plots were primarily created with seaborn. For the target variable, I only included a description, QRadioButtons, a matplotlib Navigation Toolbar, as well as two plots.

The modelling section was a little less straightforward. In order to get information on these sections, I created a function to unzip information about our results stored on GitHub. The modelling part has five sections that cover the process our group went through while modelling. In these sections, I use QRadioButtons, QLabel, and FigureCanvas again to display relevant information again. I also used QPixmap to display images of our results, and QTableWidget to display data on our results.

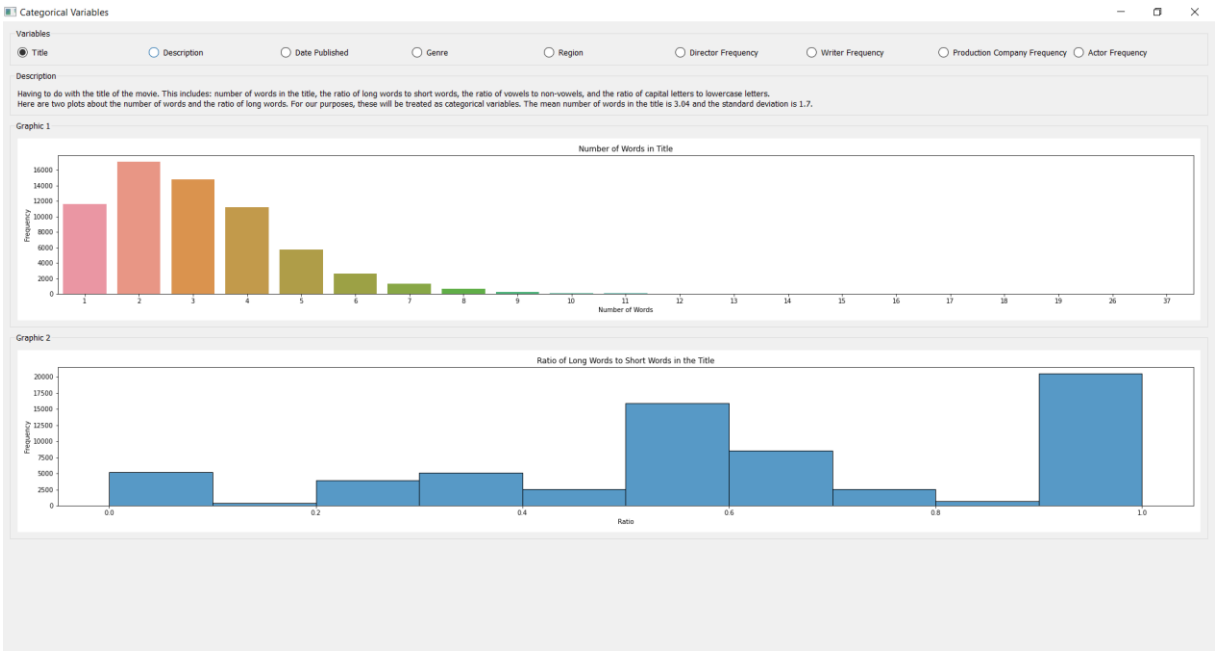
For fun, I also created a tool that allows users to randomly generate a movie and guess against our model. Again, I used QLabels and QTableWidgets to display information. I also used QPushButton to create the button to randomly generate a movie. The movie is randomly generated from a results dataset stored on Google Drive using Pandas's sample function. The user can then guess against what our model predicted, which is done through QLineEdit, and whoever is closest will win.

Lastly, I created a file section that has a link to our report and dataset using webbrowser, information about our project and the IMDb license using QMessageBox, and an exit button that closes the program. On the home page of our GUI I used, QLabel in tandem with QFont, as well as QPixmap to create a nice-looking layout when the GUI is opened.

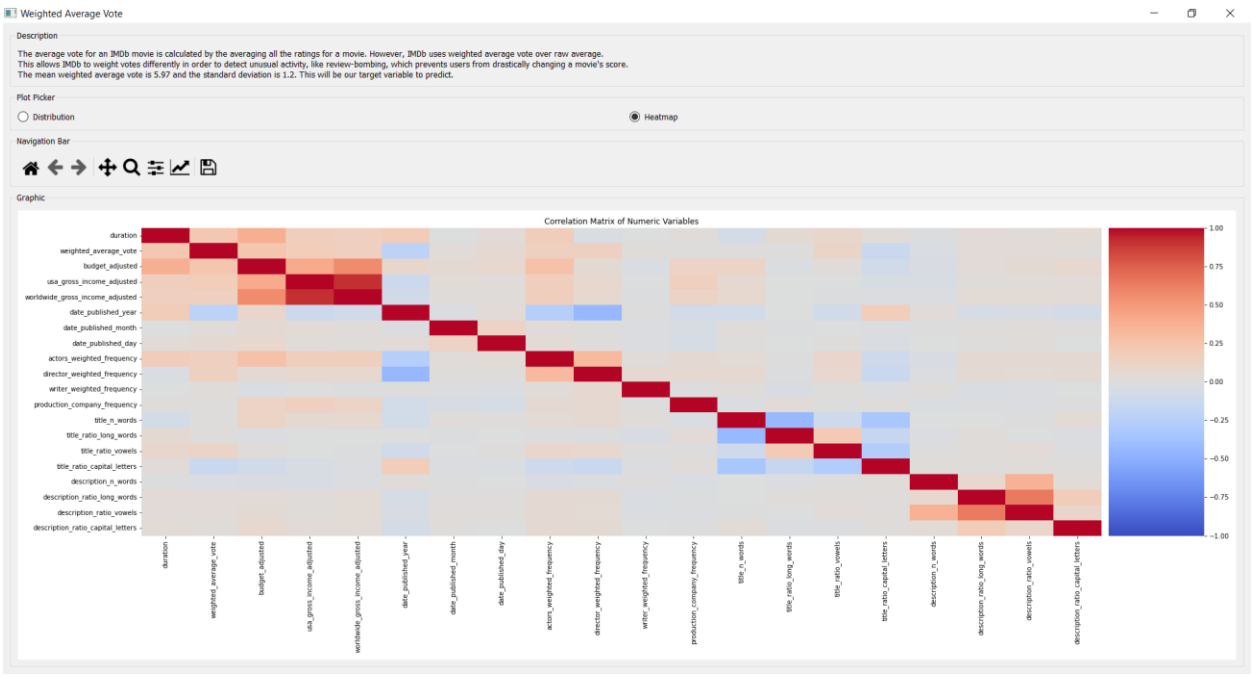
The last part of my work was on EDA. I helped with the EDA section by making plots for the categorical variables. I used Seaborn's histplot and countplot functions to create several plots which were then used in the GUI.

Using the tools I covered, I was successfully able to complete the data download, GUI, and portions of the EDA I did. The data download works nearly flawlessly, and only has issues if users repeatedly download data over and over, as Google does not allow users to do that. The GUI works extremely well as a method to show off our research. I included a few images of the GUI with captions so one can see how it works, though the easy way to see the results for this section is to simply use the GUI itself.

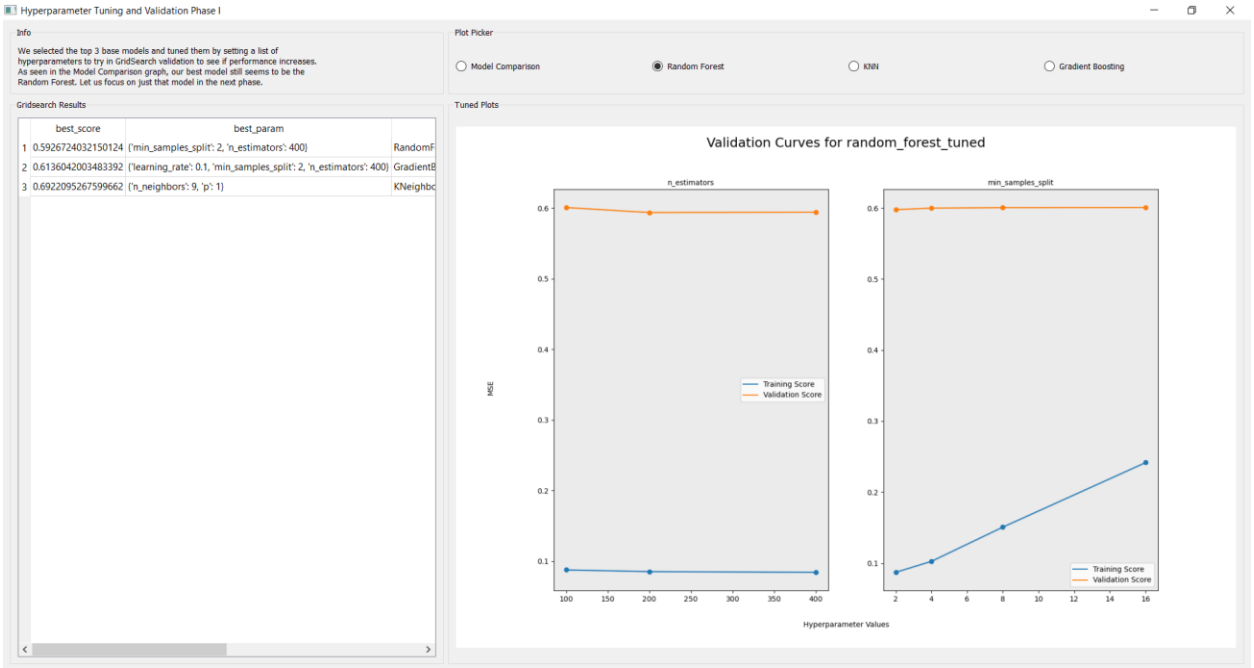
The GUI home page



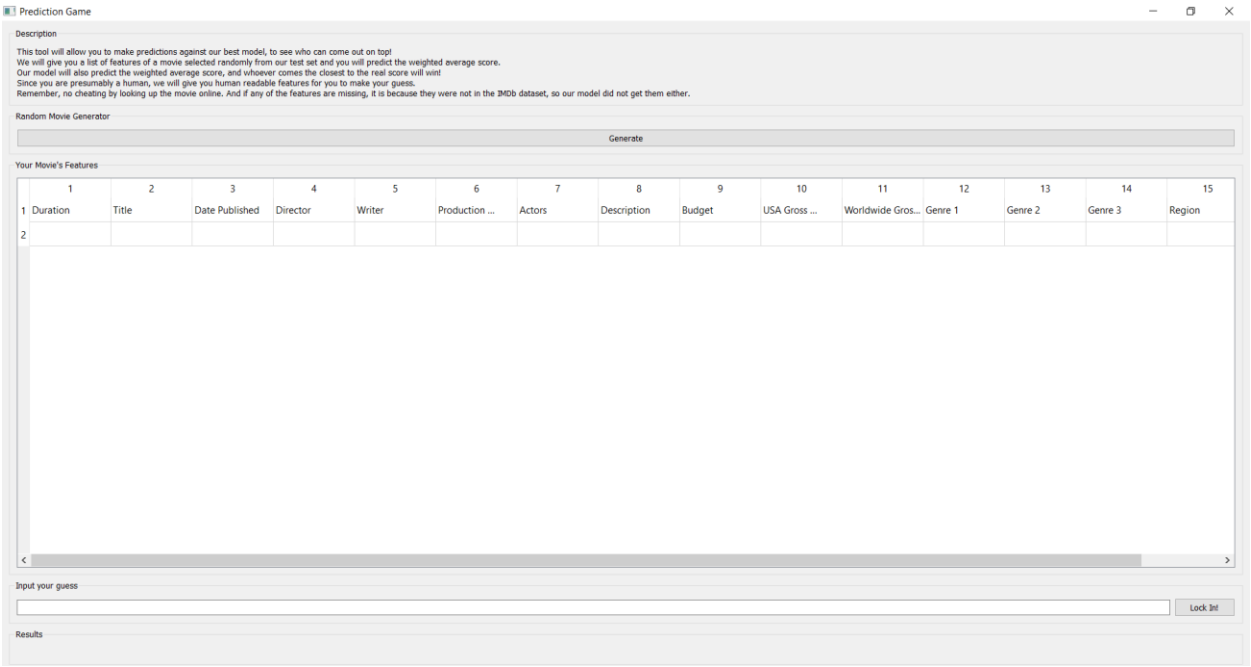
Window on “Title” in the categorical variable section



Window of our target variable



Window from the modelling section



The prediction game window

Results

Overall, I created a method to download data and display our research. I learned how to use PyQt5 extremely well at this point and can make GUIs in the future using it. I also learned good and bad methods for downloading data from the internet within Python. In the future, I would like to still find a better method of downloading data, specifically one that does not encounter so many issues, potentially by using webscraping.

Code percentage from the internet

~250 lines found on the internet

~175 of those lines modified

~1250 lines I wrote myself

((250-200)/(250+1250))x100 = 5% of code