

Department of Computer Science

Individual Project – CS3IP16

Predictive Analytics for Film User Ratings

|  |  |
| --- | --- |
| **Student:** | Kane Small |
| **Student Number:** | 23013043 |
| **Supervisor:** | Jonathan Boyle |
| **Submission Date:** | 20th April 2018 |
| **Word Count:** |  |

Abstract

*Here you create a brief summary of the content of your report. Making clear the subject matter, the problem, solution and headline results. An abstract should not normally be longer than about 250 words, and in most cases should be finalised last when the main content of your report is complete.*

# Acknowledgements

First and foremost, I would like to express my gratitude toward Jonathan Boyle for his continued support throughout my project’s lifetime and helping to guide the project in the right direction.

Secondly, I would also like to thank my Mother for her wisdom, support and strength throughout this entire year.

Lastly, I would like to thank my friends for assisting me with carrying out testing and providing valuable feedback toward improving my project along the way.

Table of Contents

[Acknowledgements i](#_Toc511998577)

[Glossary of Terms and Abbreviations iv](#_Toc511998578)

[1 Introduction 0](#_Toc511998579)

[2 Problem Articulation and Technical Specification 2](#_Toc511998580)

[2.1 Problem statement 2](#_Toc511998581)

[2.2 Stakeholders 2](#_Toc511998582)

[2.3 Technical specification 2](#_Toc511998583)

[3 Literature Review 5](#_Toc511998584)

[3.1 Similar projects 5](#_Toc511998585)

[3.2 Rating platforms 6](#_Toc511998586)

[3.3 Data sources 7](#_Toc511998587)

[3.4 Machine learning technologies 9](#_Toc511998588)

[3.5 Feature selection 10](#_Toc511998589)

[3.6 Back-end technologies 12](#_Toc511998590)

[3.7 Front-end technologies 12](#_Toc511998591)

[4 Solution Approach 14](#_Toc511998592)

[4.1 Similar project takeaways 14](#_Toc511998593)

[4.2 Data sources 14](#_Toc511998594)

[4.3 Machine learning implementation 14](#_Toc511998595)

[4.4 Coding languages and frameworks 14](#_Toc511998596)

[4.5 Chosen solution 14](#_Toc511998597)

[5 Design and Implementation 15](#_Toc511998598)

[5.1 Environment configuration 15](#_Toc511998599)

[5.2 Data acquisition 15](#_Toc511998600)

[5.3 Data cleansing/pre-processing 16](#_Toc511998601)

[5.4 Regression algorithm and predictions 16](#_Toc511998602)

[5.5 API routing 16](#_Toc511998603)

[5.6 Front-end design and user experience 16](#_Toc511998604)

[6 Testing Verification and Validation 20](#_Toc511998605)

[6.1 Unit testing 20](#_Toc511998606)

[6.2 Compatibility testing 26](#_Toc511998607)

[6.3 Usability testing 27](#_Toc511998608)

[6.4 Regression metrics 27](#_Toc511998609)

[6.5 Testing limitations 28](#_Toc511998610)

[7 Discussion: Contribution and Reflection 29](#_Toc511998611)

[7.1 Discussion 29](#_Toc511998612)

[7.2 Reflection 33](#_Toc511998613)

[8 Social, Legal, Health and Safety and Ethical Issues 35](#_Toc511998614)

[8.1 Social, legal and ethical issues 35](#_Toc511998615)

[8.2 Health and safety risks 35](#_Toc511998616)

[9 Conclusion and Future Improvements 36](#_Toc511998617)

[9.1 Conclusion 36](#_Toc511998618)

[9.2 Future Improvements 37](#_Toc511998619)

[10 References 39](#_Toc511998620)

[11 Appendices 41](#_Toc511998621)

[11.1 Appendix 1: Project Initiation Document 41](#_Toc511998622)

[11.2 Appendix 2: Logbook 41](#_Toc511998623)

[11.3 Appendix 3: Questionnaire 41](#_Toc511998624)

[11.4 Appendix 4: Scikit-learn algorithm cheat sheet 44](#_Toc511998625)

# Glossary of Terms and Abbreviations

API – Application Programming Interface

CA – Certificate Authority

Clustering – A machine learning problem whereby the outcome is to create groups of values based on their similarities.

Classification – A machine learning problem whereby the outcome is to predict a discrete output, i.e. a class or category.

CSS – Cascading Style Sheets

Data mining – Is the process of finding patterns within data.

DBMS – Database Management System

FTP – File Transfer Protocol

HTML – Hypertext Markup Language

JS – JavaScript

JSON – JavaScript Object Notation

Machine learning – Is the process of building models using data mining techniques, in order to predict future outcomes.

MAE – Mean Absolute Error

Model – In machine learning, a model is the output generated after training a machine learning algorithm with a training dataset. It takes in a predefined set of inputs and produces an output based on these variables.

One-hot encoding – Converts a set of categorical values into separate features, that are all represented by a binary value (0, 1).

ORM – Object-relational Mapping

PCA – Principal Component Analysis

PID – Project Initiation Document

RDP – Remote Desktop Protocol

Regression – A machine learning problem whereby the outcome is to predict a continuous value.

SSH – Secure Shell

Supervised learning – A machine learning category that relies on humans to act as the teacher whereby, we feed in a training dataset that includes the features and target values and from this data an algorithm can learn the relationships between the features and the target values in order to predict target values for previously unseen data.

TMDb – The Movie Database

UoR – University of Reading

UPS – Uninterruptable Power Supply

VFX – Visual Effects

VPN – Virtual Private Network

# Introduction

Machine learning is still in a very primitive state at this point in time but is being utilised a lot more frequently across different industries, from entertainment, to finance, to transportation services and many more (Sas.com, 2018). The process for machine learning models is to formulate patterns and understand the inter-linking features within the dataset being analysed, that can then be used against previously unseen datasets in order to predict values to a certain degree of accuracy. But ultimately, the end goal for using this new technology is to make better decisions without the need for human intervention.

This is where this project comes into play. The application uses machine learning regression algorithm to train a model that can successfully predict rating values for films that have yet to be released, based on the most important key attributes (the features) that influence viewer’s decisions. The main purpose behind building this application is to essentially make it easier for anyone to decide whether to see a film, based solely on a single value. This singular rating percentage is the result of, finding the inter-linking patterns and relationships of the most important film attributes and is ultimately as unbiased as possible by using only factual information.

The project commences by identifying the problem that is trying to be solved, listing the stakeholders that will form the main target audience of the application and more importantly, the main objectives of the project. In order to accomplish these objectives, it is necessary for the project to be split into five separate parts, data acquisition, data cleansing/pre-processing, regression algorithm implementation, API routing and front-end design/user experience. The first phase focuses on acquiring data from a viable source that meets the requirements (i.e. relevant metadata), the next phase transforms the acquired data into a correct format for the input to the algorithm and stores it into the database, the thirds phase implements the regression algorithm and train a model; the fourth phase revolves around building an API that will interface with the database and display data to the front-end and the last phase will concentrate on the user-experience and ease-of-use of the application.

Furthermore, the motivation and reasoning behind undertaking this project is brought to light, followed by the constraints that illustrate potential challenges for the project.

The application must then look next at any existing applications that carry out either the same or similar tasks and see how it can be improved upon, from both a machine learning accuracy and a user-experience perspective. This is accomplished by reviewing existing literature and from the findings, justify an informed solution approach on how best to meet the objectives outlined. The solution approach itself will also aim to cover any tools and technologies utilised throughout the development process in brief.

The design and implementation of the application will be explored in detail, illustrating exactly how the technologies and any libraries aided in the development of the product. The implementation will build all the way to the result of the model, which will be examined in detail as part of the discussion. This will also address the success of the solution, as well as any limitations that have been identified along the way. A reflective piece will lead on from here, bringing to light what the challenges were, what has been carried out well and how the project would be approached differently if attempted a second time.

Any potential social, legal, ethical and health and safety risks of the project are reviewed in short at this point, which will support the continued development and improvement of the application in the future.

Lastly, the report comes to an with a conclusion that quickly restates the main project objectives and discusses whether the solution that has been implemented has in fact met them. Nevertheless, this outcome leads into a list of future improvements that could potentially elevate both the project’s effectiveness and use.

# Problem Articulation and Technical Specification

## Problem statement

The number of variables that determine whether a film will be a high user-rated success has increased dramatically over the years, in-part due to the advancements made within the film industry allowing production companies to offer a wider variety of films to the viewers, specifically in the Visual Effects (VFX) category. So, with this comes the pressure of leaving it to the viewer to determine whether a film is going to be enjoyable based on these attributes. But, a viewer can only use what they already know about an upcoming film such as the genre of the film, or the actors cast, or the trailer and so on and these attributes will simply *influence* their decisions; but will not tell them whether or not this film will result in a decent or meagre film-viewing experience.

This project aims to alleviate that pressure on viewers by taking all of the factual data already present and using it to make accurate predictions on the overall rating of unreleased films, but by taking into account each of the most important film attributes and even finding relationships between them.

## Stakeholders

A total of three stakeholders were identified for this particular project, revealing how they would be affected by or provide assistance in achieving the outcome of the project.

### Developer – Kane Small

The developer, Kane Small, is responsible for advancing the project through all phases of the development lifecycle. This involves making sure that the problem statement outlined in section *3.1* is resolved and this will be accomplished by fulfilling the objectives outlined in section *3.3*. Due to time constraints and other important engagements throughout the year, the developer will also be responsible for maintaining an effective pace through time management and organisation, to ensure that project goals are met on time.

### Project supervisor – Jonathan Boyle

The project supervisor, Jonathan Boyle, is the person who will provide continued support and assistance toward guiding the project in the right direction at all times. This will be accomplished by scheduling weekly and/or bi-weekly meetings whereby a detailed discussion will comprise of the work that has been carried out and any problems encountered since the last meeting ending with a summary of what will be tackled by the next meeting.

### User – Film Viewers

The user of the finished product will be, any film viewer that is uncertain on whether or not to see an upcoming film. They will expect to use a system that is easy-to-view and understand, where the content has been organised in a concise and logical manner but most importantly that the prediction result being displayed is as accurate as possible; as they will be relying on this factor to make their decisions for them.

## Technical specification

The application to be developed will reside on a server that will deliver the content to the internet. There will be two pages, the *main* application page that will display all of the film metadata including the prediction result and a *trends* page that will provide users with updated trend graphs for the data being held.

Utilising the previously created PID, the project must satisfy the objectives outlined. However, the objectives that are listed below have been altered in order to conform to realistic time constraints and developer experience:

**Data acquisition**

* Implement a viable API data source
* Utilise the YouTube API for film trailer acquisition
* Design, build and configure the database to store the film data
* Implement a viable Python framework
* Build the scripts that will perform the data acquisition
* Setup and configure a server to host the scripts
* Create a questionnaire to aid with feature selection

**Data cleansing/pre-processing**

* Use imputation to format NULL data values
* Use one-hot encoding to format categorial features
* Use Principal Component Analysis (PCA) to reduce dimensions

**Machine learning implementation**

* Select an appropriate machine learning algorithm
* Fit the formatted data to the algorithm
* Train the algorithm to produce a model that can output a set of predictions
* Test the accuracy of the model using different regression metrics

**API routing**

* Create API routes to serve data to the front-end
* Implement basic API route authentication

**Front-end design and user experience**

* Create a functional front-end interface for the user to interact with
* Implement the typeahead and bloodhound autocomplete and suggestion engine
* Allow the user to query for films
* Implement charting/graphing capabilities to display data trends

### Project motivation

The main motivation behind carrying out this project is, the developer’s passion for film. Having had the idea to implement some form of film aggregation and search platform in the past, but not knowing exactly how to make the application stand-out, combining a field of interest such as machine learning has enabled the project to reach a state that is worthy of development.

The application itself, also has a lot of possibility for future implementations such as, including TV show support, or allowing custom search parameters to be user-inputted to test different film recipes – which would be extremely useful for production companies.

### Project constraints

The biggest constraints to consider are scheduling and scope. Ensuring that, within the given time it will be possible to research, learn and implement a successful machine learning algorithm that will be able to output a set of predictions. But more importantly, guaranteeing that the project is not over-scoped and that the objectives outlined are focused on first and foremost, where additional features and functionality can be implemented if there is additional time at the end of the development lifecycle.

# Literature Review

In order to produce a viable solution to the problem articulated in section 2.1, first research must be conducted into existing projects that are similar to this, in order to learn from what they accomplished well but also to avoid the limitations they presented. Existing rating platforms will be analysed to determine what rating information is often presented to users and how it is calculated. A viable data source must be selected to handle the acquisition of film metadata, so a comparison of available sources will be carried out. In order to implement a machine learning model, the different categories and problems within the field must be properly examined so that an informed decision can be made on the best technique to utilise. Lastly, the front and back-end technologies will be explored to decide what will be the best setup for a project like this and will ultimately satisfy the problem presented.

## Similar projects

A paper released by students at the University of Salford, aimed to implement data mining techniques to analyse and predict film ratings (Saraee, et al., 2004). The paper outlines IMDb as their data source, retrieving film data in a collection of un-formatted text documents that were all linked by the film title. This is definitely not a viable source for acquiring data, as each crucial film attribute would have to be extracted manually from the relevant text documents and combined into a single structured format for each film. As mentioned in the paper, this would require some form of natural language processing (NLP) to extract the key pieces of information.

The paper split the evaluation phase into four separate sub-phases, pre-selection, cleansing and integration, selection and transformation and data mining. Pre-selection, involved filtering out the unwanted files that contained data that was unnecessary or where the information was already present in an alternate file such as, cinematographers, or actors and so on. The cleaning and integration sub-phase revolved around processing each file into comma separated value (CSV) file so that these could later be read by their choice of database management system, which happened to be Microsoft Access. Within Microsoft Access, tables could be formed from the CSV files linking all attribute tables (actors, ratings and so on) to a table with the unique film entries.

Next came the selection and transformation sub-phase, centring around calculating numerical rating values for fields such as directors and actors, as it was to their understanding that these particular film attributes would heavily influence a films success. In order to calculate these values, Structured Query Language (SQL) queries were constructed that computed the average user rating for every film an actor or director had been in or directed, respectively. This produced a singular rating value for each individual actor and director stored in the database.

The final sub-phase could only be accomplished once the data had been cleaned and transformed, via the previous two steps. This project trained a classifier which would be used to predict a class or each film into one of the following categories:

|  |  |
| --- | --- |
| Scale | Category |
| 7.5 – 10 | Excellent |
| 5 – 7.4 | Average |
| 2.5 – 4.9 | Poor |
| 1 – 2.4 | Terrible |

Table 1 - Rating categories from the academic paper - A data mining approach to analysis and prediction of movie ratings

So, this method focused on generalising the prediction rating instead of providing a continuous numerical value as the output. It could be argued that this provides users with a much simpler method of seeing whether a film has been predicted to do well or not, to influence their decision on whether to see it upon release.

|  |  |
| --- | --- |
| Film | Rating |
| 5-25-77 (2005) | Average |
| Ask the Dust (2005) | Average |
| Batman Begins (2005) | Excellent |
| Because of Winn-Dixie (2005) | Poor |
| Bewitched (2005) | Average |
| Bridget Jones: The Edge of Reason (2004) | Average |
| Cars (2005) | Excellent |
| … | … |

Table 2 - Predicted film ratings from the academic paper - A data mining approach to analysis and prediction of movie ratings

A subset of the results they achieved can be found in Table 2 above. Taking a couple of the films as an example, Batman Begins which was set to release in 2005 at the time the paper had been written, sits at a certified fresh rating of 84% on Rotten Tomatoes (Rotten Tomatoes, 2018). On IMDb itself, Batman Begins has an average user rating of 8.3/10 (IMDb, 2018). On the opposite end of the scale, Because of Winn-Dixie has a *rotten* rating of 54% on Rotten Tomatoes (Rotten Tomatoes, 2018) and an average user rating of 6.4/10 on IMDb (IMDb, 2018). Consequently, these results do illustrate at least some level of accuracy even 14 years on, but a larger sample of films would need to be tested to confirm the accuracy.

Ultimately, this academic paper has highlighted a lot of interesting points to consider when carrying out this project. They had a lot of trouble with data source selected, so locating one that returns the film metadata in a viable format that is still structured would be recommended. They also did not take into account a large selection of film attributes and therefore will potentially be biased toward the actors and directors of each film.

## Rating platforms

### Rotten tomatoes

Rotten tomatoes is one of the top aggregator of film and TV show reviews from expert critics in the industry (Rotten Tomatoes, 2018). They are well known for the *Tomatometer*, which essentially is a scoring system calculated based on the opinions of hundreds of film and TV show critics. The value itself is a percentage rating that illustrates how positively or how negatively a film or TV show has been reviewed by critics, but is only displayed after at least five reviews have been acquired.

There are different categories within the Rotten Tomatoes platform, such as the *Certified Fresh* and *Rotten* statuses that reveal to a user whether a film or TV show has either scored above 75% or below 60% in total positive reviews. This is useful to the user for establishing a visually eye-catching marker that distinguishes a truly terrible film or TV show, from a critically acclaimed film or TV show.

Lastly, there is the *Audience rating* which reveals the percentage of Rotten Tomatoes users that have rated the show. The interesting value here is the value that is displayed for an unreleased film or TV show, which indicates how many users have added an upcoming film to their *Want to-See* list. This however, is a biased rating as it only takes into account whether or not a user *wants* to see a film and not any additional factual information that can be acquired. This is a value that would be useful as a feature when training a model, but does not reveal to the user a predicted value of whether or not the film *will* be good. But, it isn’t as biased as a critic rating, as the number of critic reviews is always smaller than the total number of users that have added a film to their *Want to-See* list.

### Internet Movie Database (IMDb)

IMDb is another (IMDb, 2018) is another film and TV content aggregator, tailored more toward providing the content and metadata for films and TV shows as opposed to average critic review ratings. But the platform does provide a popularity score on each film and TV show page, indicating to users how popular that film or TV show has been by the number of visitors, for that week. They also have a user rating value, that represents the mean value of all users that have casted a vote for a film or TV show, on a rating scale of 1-10.

IMDb’s user rating value, isn’t as biased as Rotten Tomatoes’ critic rating, as there is a much larger audience or sample to calculate a mean rating from. A lot of content on IMDb, films especially, can receive hundreds of thousands of votes. However, as IMDb mentions here (IMDb, 2018), the vote averages that are displayed on a film or TV show page are weighted averages as opposed to raw data averages. So, the values have been filtered to try to reduce *vote stuffing*, whereby users create multiple accounts to purposefully boost or reduce a rating. Though, the methods by which IMDb carries out the filtering are not released to the public, most likely to prevent attackers from creating bots that can overcome the filtering.

### Metacritic

The third and final rating platform (Metacritic, 2018), is similar to the previous two in that it provides user with most of the film or TV show metadata as well as a *Metascore*, but it also does this for both video games and music. The Metascores are calculated very similarly to both Rotten Tomatoes and IMDb, by utilising a weighted average of critic reviews, but for film and music they also normalise the end result (Metacritic, 2018).

### Summary

So, with each rating platforms there are limitations as well as benefits. Rotten Tomatoes is the only one out of the three platforms to display some form of indicator as to what it’s users think about an upcoming film by means of the *Want to-See* list, which could be useful as a feature for training purposes. IMDb has a lot more votes on average, per film or TV show and because these are *supposedly* all user ratings, the end result should be a much less biased one. Metacritic is very similar to Rotten Tomatoes, except it offers generated ratings for video games and music, which neither Rotten Tomatoes or IMDb offer.

## Data sources

A variety of data sources were researched to see which could be used as the primary origin to obtain film metadata from. The most important feature was how easy it would be to retrieve the data from the source, so the preferred method would be via an Application Programming Interface (API). An API essentially, abstracts the complex code from the user and reveal simple and formatted data instead (MDN Web Docs, 2018). They also interface with the database so that the front-end never has to directly make database calls, which is slow and insecure.

### The Open Movie Database (OMDb)

The OMDb was the first data source considered, simply because it happened to be one that the developer had heard of in the past. It is an open source service that offer an API to retrieve film metadata (OMDb, 2018). The site however, does limit the number of requests that a developer can make using an API key to 1,000 a day. This limit is then increased and eventually removed, depending on the reward tier you are subscribed to if you become a patron. This involves paying a monthly donation to the developer of the service (Patreon, 2018). At $1 per month, users receive basic API access and 100,000 daily requests. At $5 per month, users receive access to the basic API as well as the poster API and are limited to 250,000 requests a day. Lastly, at $10 per month users are provided with their own server that has no limits and access to the API’s mentioned above.

### Rotten Tomatoes

Rotten Tomatoes was also considered and has been analysed as a rating platform above in section 3.2.1. It provides access to the Tomatometer and Audience rating values, as well as a subset of critic reviews per film but does not provide detailed film or TV show metadata (Rotten Tomatoes - Developer Network, 2018). The access to the API is also granted on a per proposal application basis and therefore, depending on their turnaround time may not have been reviewed before the project commenced.

This source would not be viable for the project, but could be used for the ratings it provides, which could be used for feature selection.

### IMDb

IMDb does not provide an official API. They do however, offer subsets of their data that can be downloaded manually (IMDb - Datasets, 2018). These separate tab-separated-values (TSV) files would have to be formatted and then combined before being stored in a database. Due to time constraints, this might not be the most viable data source solution. The data would also have to be filtered, i.e. filtering out adult films, after the data had been stored in a database.

### The Movie Database (TMDb)

Lastly, there is TMDb. Another open source service that offer an API to gain access to their data. They offer an extremely detailed documentation, that lists each of the routes available and how to use them including information on the query strings and the responses given (TMDb - API, 2018). The API also provides a range of film metadata that satisfy the project requirements.

The API is rate-limited, whereby developers can only perform 40 requests every 10 seconds, but this is not as much of a problem as the OMDb which had a maximum number of daily requests. The rate limit will just have to be observed so that the application doesn’t exceed it. This can easily be accomplished by reading the *X-RateLimit* request headers and keeping track of the *X-RateLimit-Remaining* value. If this value is exceeded then, an HTTP status will be returned with the *Retry-After* response header and after this cool down period expires requests can continue (TMDb - Rate Limiting, 2018).

The response from the API is in a JavaScript Object Notation (JSON) format, which enables the key information to be extracted with ease. JSON is format for displaying/storing structured data and is most commonly used for data transmission within web applications (MDN Web Docs, 2018). Data is stored within a JSON file in a key/value pair format, where the keys are always strings and the value is a supported data type.

If selected, this option would be able to provide the required metadata, in a format that would be easy to manipulate and without any serious performance/time limitations.

### Summary

Again, each data source in this list have their defining strengths and weaknesses but ultimately, the source selected will be the one that offers the best API support and satisfies the requirements/objectives of the project. The OMDb definitely offers a wider selection of metadata such as reward information and Rotten Tomatoes, Metacritic and IMDb ratings. But, with the recurring monthly payment, it would have to be decided whether the additional metadata would be worth the cost. TMDb also did say that they were in the process of providing API support for awards in 2017, so this could be added very soon (TMDb, 2018).

## Machine learning technologies

### Algorithm/model selection

Machine learning and data mining often get confused and was something that warranted additional research. Machine learning is the process of building models using certain data mining techniques, in order to predict future outcomes on a given dataset. Data mining, is the process of finding patterns and/or relationships within datasets.

Before selecting an appropriate model, the different categories in which machine learning problems are organised must first be understood.

There are two main categories that machine learning algorithms can be split into, supervised and unsupervised learning (Scikit-learn, 2018). Supervised learning is focused on finding the relationships in a dataset between the target value that is trying to be predicted and the remaining features that influence the target. Problems within this category will be one of two types, classification or regression. The easiest way to distinguish the two is to look at the target value, if it is a class value then predicting the target will be a classification problem, whereas if the target value is a continuous value then it is a regression problem.

Unsupervised learning concentrates on, not being trained alongside any target values and predicting an output based on those attributes. Two of the common problems within this category are, clustering and association. Clustering problems look to find and group data values that have similarities. A common real-world example of a clustering problem is Netflix’s recommendation engine. They cluster groups of similar films together in order to better suggest films to users based on what they have viewed in the past. Association problems on the other hand, looks at a collection of items and produces association rules that will attempt to predict the occurrence of one item based on the occurrence of other similar items. A common real world example of this type of problem being used is for the Amazon *Frequently bought together* section, which will recommend related items based on the item currently being searched as well previous user purchases.

Data often needs to be pre-processed before fitting it to the correct model. This step is often just as, if not more important than training the model itself as, poorly formatted data can affect the predicted results of a model. Data pre-processing can be split into the following categories:

**Data cleaning**

* Focuses primarily on removing duplicates and handling NULL values.

**Data transformation**

* Implementing standardisation to format all attributes in a given dataset onto the a uniform scale, where 0 is the baseline

**Data reduction**

* Implementing techniques such as principal component analysis (PCA), which reduces down the number of dimensions (columns) within a given dataset

There is a lot to consider when it comes to selecting the correct model, but once the problem being addressed has been filtered into one of the main categories it makes the process somewhat easier to manage.

### Useful libraries

**Tensorflow**

Tensorflow is an open source machine learning and data graphing library originally built by Google (Tensorflow, 2018) and is being used by a lot of big-name brands such as AMD, Dropbox, Snapchat, Intel and many more. From an initial overview, it seems as if Tensorflow is not as welcoming to newcomers in the field of machine learning. The documentation is somewhat complicated and it isn’t too clear what is the best way to start using the library. It does however, offer guides on how to install it on at least three unique platform operating systems and even offers a simplified interface that is supposed to mimic the more established scikit-learn library to easy users through the transition (GitHub - Tensorflow, 2018).

**Scikit-learn**

Scikit-learn is another popular open source machine learning library for python. It does have a few dependencies such as NumPy, SciPy and matplotlib as it was built around these tools, but these are again all open source libraries (Scikit-learn, 2018).

From a cursory overview, scikit-learn is a lot more tailored to inviting users who are both new to machine learning and data analytics as well as experienced industry professionals. The documentation is thorough and provides detailed and walked through examples in the form of tutorials, which makes implementing the library easier to understand. It also provides information on how to select the appropriate model for the dataset you have access to, which again for someone starting out in the field is extremely helpful.

## Feature selection

Feature selection is an important part of machine learning and centres around finding the most important features that directly influence the value being targeted for prediction.

### Questionnaire

To assist with feature selection, a questionnaire was carried out to determine what a subset of real world viewers thought their most important film attributes were and to rank those in order of how they influence the decision of whether or not to see a film.

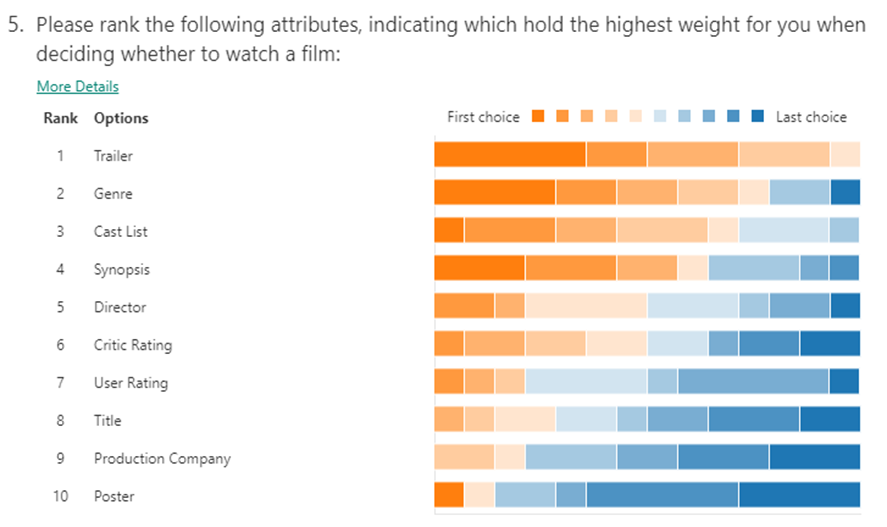
The full list of both questions and answers can be found under 11.3 Appendix 3. But the most important results are those from question 5 which asked users to rank a predefined list of film attributes in order of how they influence your film-watching decisions.

Figure 1 - Film attribute ranking questionnaire results

Figure 1 above illustrates, from the small subset of users who responded, that the most important film attribute feature is the trailer. Therefore, taking into account a film trailer’s view, like and dislike count as model features could help increase the predictions accuracy. Surprisingly, both the critic and user ratings came in at positions 6 and 7 respectively, proving to a certain degree that the current platforms that present these values to users are not helping viewers decide on whether an upcoming film will be worth watching. Taking into account as many features as possible however, will provide a much more accurate and unbiased prediction result.

### Sentiment analysis

Sentiment analysis is a technique that focuses around natural language processing (NLP), being able to train a computer to understand human language (Bird, et al., 2009) and from that, decide whether or not the language being analysed is positive or negative (or neutral).

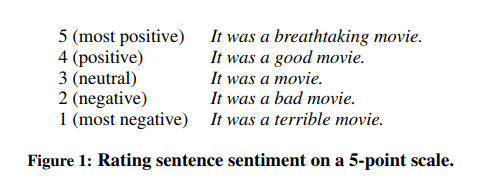
The following academic paper uses sentiment analysis to rate film reviews on a scale of 1-5, see Figure 2 for the full scale categories (Roberts & Yan, 2014). They utilised a dataset of ~10,000 film reviews/sentences and split those into an 80%, 20% training and test set respectively to train their model to classify film reviews.

Figure 2 - Sentiment rating scale (Academic paper)

This technique could be extremely valuable in this project, by performing sentiment analysis on tweets from Twitter, or comments on Facebook posts in order to gain a popularity rating for upcoming films and provide an additional feature when training the model.

### Summary

In the end, the features selected will be the ones that can be pre-processed to satisfy the input requirements of a given machine learning model.

## Back-end technologies

### Database

**NoSQL**

NoSQL is a database system that was researched as a newer technology to familiar ones such as, Structured Query Language (SQL). NoSQL are non-relational database systems that are focused more toward performance and scaling (Amazon Web Services, Inc, 2018). Unlike traditional relational database models, that define the relationships between tables NoSQL databases store data in JSON formats for example. NoSQL databases can also be queried a lot more efficiently than traditional relational database management systems (RDBMSs) to the JSON or equivalent format data is stored in.

Depending on how the data is acquired and from which source, a NoSQL database system could work for this project but, the underlying relationships between the film data attributes would first have to be considered.

**SQL**

SQL database systems rely on more traditional methods of storing data into tables with rows and columns and finding relationships between the tables, linking them by primary and foreign keys. SQL queries can then be sent to an RDBMS for execution.

RDBMSs are something the developer is a lot more confident with and has experience implementing, however depending on the hardware being utilised storing the required amount of film metadata might prove both challenging to store and/or query later on.

## Front-end technologies

### Web frameworks

Web frameworks allow developers to build web applications and utilise a specific back-end programming language. They often allow additional modules to easily be connected and maintained for extra features to be implemented within the application. For this project, the developer would like to use their most experienced language which is python, so the following libraries were considered:

**Pyramid**

* A lot newer that the other two, so may have less technical support and less plugin support
* Intimidating with the sheer number of setup options available to the user

**Django**

* Targeted toward large scale projects
* Is the oldest framework of the three and as a result, has a lot more user feedback
* The documentation is extremely detailed

**Flask**

* The fastest to setup and start prototyping with, can be up and running with 7 lines of code
* Targeted toward smaller projects

The framework utilised will ultimately be the one that satisfies the requirements and appeals to the developer.

### Application programming interface (API)

As mentioned above in section 3.3, an API is used to interface and retrieve data directly from a web application without exposing access to the code behind it. An API will also have to be created to interact with the data stored in the database that is selected. This will enable the front-end web pages to request data from an intermediary point instead of calling directly from the database, which is an insecure process.

Python web frameworks all provide methods that enable API routes to be constructed alongside templates that can render the data returned from the database, so this should not be a problem.

### Grid systems

**Bootstrap**

Grid systems are extremely useful for the alignment and positioning of content on a web page. Bootstrap has been around for a long time now and was originally developed by Twitter. It is an open source tool that allows developers to build websites and web applications with a responsive grid system, that transforms and adapts to fit the resolution of the device that is being used to view the content on (Otto, 2018).

Utilising Bootstrap would allow for the front-end user experience to incorporate a set well-organised and laid out styles. The documentation is also extremely detailed, which is exactly what the developer is after in this case to ensure that the front-end can be built with ease leaving more time on the core features of the project.

### Autocomplete/suggestion engine

**Typeahead JS and Bloodhound**

Lastly, an autocomplete engine will be required for the searching of films within the application. Typeadhead.js is an autocomplete library by Twitter, that combines the user interface (UI) view and a suggestion engine, Bloodhound to make informed suggestions to users (GitHub, 2018). The suggestion engine, Bloodhound, will be perfect for providing efficient lookups on the film data provided and offers additional features such as prefetching, which allows the browser to cache subsets of data so that the suggestions can be returned to the user at a much faster rate (GitHub, 2018).

# Solution Approach

In this section, a solution will be clearly outlined influenced heavily by the options outlined in the literature review section above.

## Similar project takeaways

## Data sources

Utilising the results from the research carried out in section 3.3, the data source that will be best suited for this project is TMDb API. This option provides the means to interface with the required data via an extremely well documented API, it does not limit the number of queries that can be executed on a daily basis and provides a very active forum that can be used to ask any questions that may arise during the development of the project.

## Machine learning implementation

For the machine learning implementation, based on the dataset that will be acquired from the data source selected above a supervised learning technique will be the most suited option for the problem presented. A continuous user rating value is what the model will attempt to predict, so a regression algorithm will need to be implemented here.

Following on, scikit-learn offers a variety of regression methods but a decision tree regressor would suit the problem and dataset well. They are able to handle data features that are both categorical such as genres, as well as numerical such as budget, or runtime and so on (Scikit-learn, 2018).

## Coding languages and frameworks

## Chosen solution

Based on the sections listed above the chosen solution will utilise TMDb for the acquisition of data, a MySQL RDBMS to store the data retrieved into the relevant tables, the features that have been selected (genres, budget, runtime, trailer view count, trailer like count, trailer dislike count and user rating) will provide a decent spread of attributes for the training of the model, a supervised machine learning technique will be implemented and utilise a decision tree regression algorithm and Flask will be used as the web framework to assist with building the application.

A lot of different methods, techniques and technologies could be applied to this project, but the solution outlined above will satisfy the problem articulated in section 2.1.

# Design and Implementation

## Environment configuration

### Raspberry pi

### DigitalOcean Droplet

## Data acquisition

### Libraries utilised

**Acquisition:**

* requests
  + Handling POST/GET HTTP requests
* pymysql
  + Connecting Python to the MySQL database
* progressbar
  + Visual aid for time-consuming operations
* time
  + Sleeping the code to wait out the rate limiting

**Error handling:**

* csv
  + Printing errors to a csv
  + Prevents any operations from halting
* traceback
  + Allows for the printing of stack traces

### Functions created

* getLatestFilmID()
  + Checking latest film ID via API
* insertData(sqlQuery, params)
  + Inserting data into the database
* getTrailerData(film\_id)
  + Acquiring trailer data
* matchActors(film\_id)
  + Matches actors to film\_id (separate API routes)

## Data cleansing/pre-processing

### Data imputation

### One-hot encoding

### Principal component analysis (PCA)

## Regression algorithm and predictions

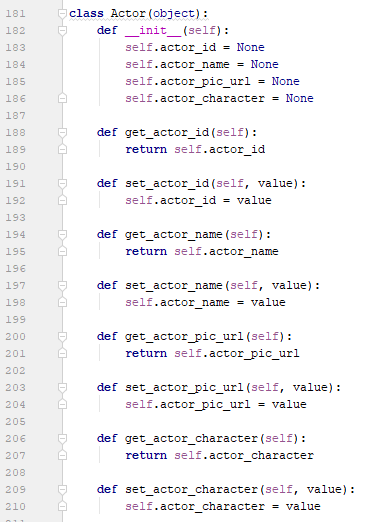
### Model persistence

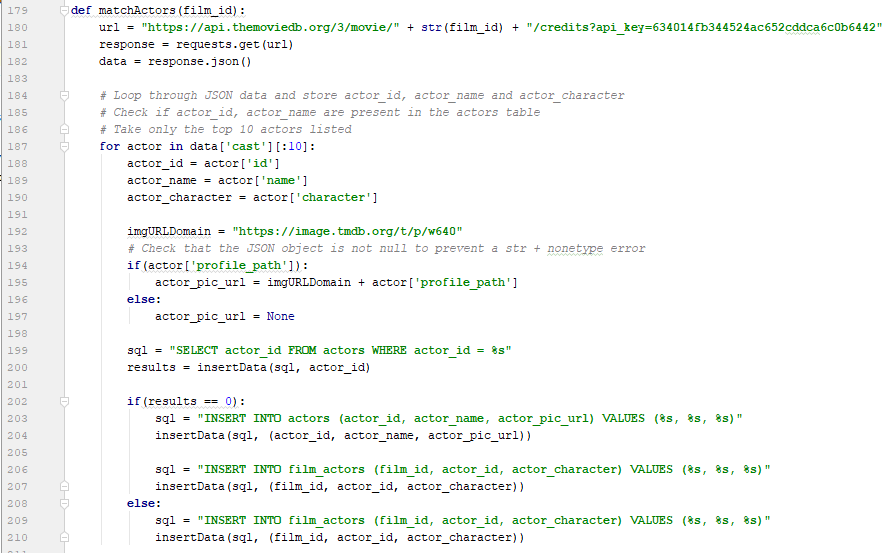
## API routing

Unique search example: <https://ksmall.me/api/film/58>

Query example: <https://ksmall.me/api/search/films?query=deadpool>

## Front-end design and user experience





* 1 class per database table
* 5 separate SQL queries
* Map results to an object
* Return jsonpickled object

# Testing Verification and Validation

## Unit testing

Unit testing is a method that involves testing the individual components of an application, such as functions, classes and so on, to ensure that they carry out their intended purpose (ISTQB Exam Certification, 2018). This project however, did not conduct any automated unit testing with specialist testing software, instead the unit tests were carried out manually. This involved utilising Python’s *print* function, which provides a full traceback detailing where exactly an exception has been raised and this helps identity and locate the erroneous code as quickly as possible. A collection of the vital unit tests carried out are provided below inTable 3.

**Back-end code (Data acquisition)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Component | Expected Outcome | Actual Outcome | Action Taken |
| 1 | getLatestFilmID() function | Retrieves the latest ID from TMDb API | The latest film ID is returned from TMBd API | None |
| 2 | insertData() function | Takes 2 parameters, a SQL query and the values being inserted into the database, queries the database and returns the results | Inserts the data passed and returns the results of the execution when called | None |
| 3 | selectData() function | Takes a SQL query and selects data from the database, returning the results | Selects data from the database and returns the results | None |
| 4 | insertGenres() function | Fetches a list of genres and inserts them into the database | Fetched genres are successfully inserted into the database | None |
| 5 | getTrailerData() function | Takes in a film id, queries TMDb API for a YouTube video key, queries the YouTube API with that key and returns all trailer details related to the key | Returns the trailer URL, view, like and dislike counts as expected | None |
| 6 | matchGenres() function | Checks to make sure if genres related to the current film iteration are already in the genres table, if they aren’t add them and if they are do not | Adds genres to the genres table if not present, otherwise updates the association table with the genres for the current film iteration | None |
| 7 | matchProdComps() function | Checks to make sure if production companies related to the current film iteration are already in the production companies table, if they aren’t add them and if they are do not | Adds production companies to the production companies table if not present, otherwise updates the association table with the production companies for the current film iteration | None |
| 8 | matchActors() function | Checks to make sure if actors related to the current film iteration are already in the actors table, if they aren’t add them and if they are do not | Adds actors to the actors table if not present, otherwise updates the association table with the actors for the current film iteration | None |
| 9 | matchDirectors() function | Checks to make sure if directors related to the current film iteration are already in the directors table, if they aren’t add them and if they are do not | Adds directors to the directors table if not present, otherwise updates the association table with the directors for the current film iteration | None |
| 10 | checkIfPresent() function | Simply performs a check to see if the data being passed to it is there or not, if it is return the data, if it isn’t return *None* | Returns the data if present, returns *None* if it isn’t | None |
| 11 | insertFilms() function | The main function that runs via a while loop across a list of all film ids and acquires all of the film metadata utilising each function listed above and stores the data in the database | Executes via a while loop, displays a progressbar, sleeps to avoid exceeding rate limitations, filters out adult and foreign films, prints any errors to a csv file | None |

Table 3 - Back-end code (Data acquisition) unit test results

**Back-end code (Regression modelling and predictions)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Component | Expected Outcome | Actual Outcome | Action Taken |
| 1 | Training data mapping | Training data csv is mapped correctly to a pandas dataframe structure | A pandas dataframe is created, populated with the csv data and with the relevant header names | None |
| 2 | Genre categories mapping | Genre categories are read from a csv file and mapped to a Python dictionary object | Genre categories successfully stored in Python dictionary | None |
| 3 | Film genres acquisition | A list of genres for each film are acquired, using the genre categories dictionary as a reference | List of genres for each film are successfully acquired and stored in a list | None |
| 4 | One-hot encoding genres | Genres are subjected to a formatting technique called one-hot encoding | Each genre is given its own category and a binary value represents a genres relation to a certain film | None |
| 5 | Handling NULL values | NULL values are imputed using the *fillna()* function within pandas | All NULL values are replaced with an arbitrary -1 value | None |
| 6 | PCA | The dataset is subjected to PCA, which reduces the number of dimensions | The number of dimensions are reduced successful and the results and the target are appended back into a single dataframe | None |
| 7 | Regression fit | The formatted dataframe is fitted to the decision tree regressor | The decision tree regressor is trained successfully | None |
| 8 | Saving the model | The model is then saved to an external file, so that it can be loaded at any point | Model is saved to an external file successfully | None |
| 9 | Regressor predictions | The regressor makes predictions on the test data | Predictions on test data are stored successfully in a list | None |
| 10 | Storing prediction ratings | Predicted ratings are inserted into the database | Predicted ratings are successfully stored in the database | None |
| 11 | Regression metrics | The explained variance and mean absolute error values are printed to the console | Regression metrics are successfully printed to the console | None |

Table 4 - Back-end code (Regression modelling and predictions) unit test results

**Front-end code (API routing)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Component | Expected Outcome | Actual Outcome | Action Taken |
| 1 | Genre() class | The required variables are created and the appropriate setters and getters are created | Class created to store genre data as objects | None |
| 2 | Actor() class | The required variables are created and the appropriate setters and getters are created | Class created to store actor data as objects | None |
| 3 | Director() class | The required variables are created and the appropriate setters and getters are created | Class created to store director data as objects | None |
| 4 | Production\_Company() class | The required variables are created and the appropriate setters and getters are created | Class created to store production company data as objects | None |
| 5 | Film() class | The required variables are created and the appropriate setters and getters are created, including instances of the genre, actor, director and production company classes as lists | Class created to store film data as objects | None |
| 6 | Trends() class | The required variables are created and the appropriate setters and getters are created | Class created to store trend data as objects | None |
| 7 | index route (“/”) | The home route that simply returns the index.html template | Route directs users to the index.html template page | None |
| 8 | trends route (“/trends”) | The home route that simply returns the trends.html template | Route directs users to the trends.html template page | None |
| 9 | getTrends route (“/api/get\_trends/<int:trend\_id>”) | The route that queries the database, maps the returned data to the trends object and stores the data into a json file | Queries database for trends, maps results to trend object, saves object to json file | None |
| 10 | getTrends route (“/api/show\_trends/<int:trend\_id>”) | The route that loads a trend json file by the provided id and returns the results to the page as an HTTP response | Returns loaded json data to the page as an HTTP response | None |
| 11 | getFilms route (“/api/search/films”) | The route responsible for taking in an additional query parameter and querying the results returned from the database, returning an HTTP json response | Uses query parameter in SQL query, returns an HTTP json response with the results | None |
| 12 | getFilm route (“/api/film/<int:film\_id>”) | The route takes a single film id and queries the database for the required metadata | Returns an HTTP json response with the metadata related to that single film id | None |

Table 5 - Front-end code (API routing) unit test results

**Front-end code (HTML/CSS/JS)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Component | Expected Outcome | Actual Outcome | Action Taken |
| 1 | Navigation links function as intended upon user selection | User relocated to alternate page | Alternate page loads | None |
| 2 | Navbar menu dropdown button toggles on smaller devices when clicked | Toggled button reveals dropdown menu | Nothing happens when selected | See section 7.1.1 |
| 3 | Input field produces results when a query is provided | A list of (up to 10) results are displayed to the user | Results are displayed to the user | None |
| 4 | Selecting an item from the search results updates the page content | Film data should update dynamically | Film data is updated dynamically | None |
| 5 | YouTube video trailer is playable | The YouTube video should be playable | The YouTube video can be played | None |
| 6 | YouTube controls are accessible/usable | All of the default YouTube controls should be visible and usable by the user | The user can see and select any of the default YouTube controls | None |
| 7 | Trend graphs display data to the user upon page load | When the user visits the trends page, the graphs should load | Upon visiting the trends page the trend graphs load immediately | None |
| 8 | The trend graphs should display additional data when hovered over | If a user hovers over the trend graph additional data should be displayed | When the user hovers over the trend graph additional data is displayed | None |
| 9 | selectFilmStatus() JS function | Takes in a film status value and based on that value returns the relevant status badge class | Successfully returns status badge class | None |
| 10 | Bloodhound suggestion engine assignment | The bloodhound engine is initialised, returning a mapped JS array of film attributes provided by the API | Bloodhound variable assigned to a list of film attributes from the API | None |
| 11 | User/predicted rating bar assignments | The progressbar.js variables are assigned, configuring the options to the developers requirements | Progressbar.js variables assigned | None |
| 12 | grabFilmData() JS function | Takes a film id parameter, calls the API, returns the results, formats them accordingly and passes the data to an HTML page | Formats results returned from API call and displays the data to an HTML page | None |
| 13 | Typeahead UI instantiation | The typeahead.js dropdown that will display the search results is instantiated, defining the appropriate options | typeahead.js UI object instantiated | None |
| 14 | getJSON() asynchronous functions | For the trend JS, the getJSON() functions take an API route as an input and use the returned JSON object to map the results to a newly defined chart.js object | getJSON() function calls from the API, uses JSON object response to map to chart.js object and display to an HTML page | None |

Table 6 - Front-end code (HTML/CSS/JS) unit test results

All of the functions listed above in Table 3, Table 4, Table 5 and Table 6 completed their unit tests successfully, For Table 3, this resulted in integrating the functions tested into the *main()* method for use in the acquisition of the entire film dataset.

## Compatibility testing

It is always essential to test any web application on a variety of operating systems, devices and browsers, to ensure that it reaches and is accessible to the widest range of users. Due to the nature of the project, the front-end interface is the platform in which users will interact with, therefore only this part of the application has been compatibility tested and the prediction modelling has not. At the time of writing this, the most up-to-date versions of the operating systems and browsers were used for each individual test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Device | Operating System | Web Browser | Pass/Fail |
| 1 | Desktop | Windows 10 | Google Chrome | Pass |
| 2 | Desktop | Windows 10 | Mozilla Firefox Quantum | Pass |
| 3 | Desktop | Windows 10 | Microsoft Edge | Pass |
| 4 | Desktop | Windows 10 | Internet Explorer 11 | Fail |
| 5 | Laptop | macOS High Sierra | Safari | Pass |
| 6 | Laptop | macOS High Sierra | Google Chrome | Pass |
| 7 | Laptop | macOS High Sierra | Mozilla Firefox | Pass |
| 8 | Mobile | Android 8.0 “Oreo” | Google Chrome | Pass |
| 9 | Mobile | Android 8.0 “Oreo” | Mozilla Firefox Focus | Pass |
| 10 | Mobile | iOS 11 | Safari | Pass |
| 11 | Mobile | iOS 11 | Google Chrome | Pass |

Table 7 - Application compatibility test results

*Table 3* clearly identifies that almost every compatibility test passed, but happens to fail when tested on a Windows 10 desktop device running Internet Explorer 11. This test result was inspected further and the results are discussed in section 7.1.1.

## Usability testing

Usability testing revolves around making sure that the user experience when performing tasks using an application is an easy and problem-free process. Therefore, the following set of tests outlined in *Table 4* reveal the results of two external users that the developer knew and asked to test the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Pass/Fail | | Action Taken |
| **User 1** | **User 2** |
| 1 | Can you search for a film using the search input field? | Pass | Pass | None |
| 2 | Can you play the YouTube video trailer loaded for a film? | Pass | Pass | None |
| 3 | Can you full screen the YouTube video trailer loaded for a film? | Pass | Pass | None |
| 4 | Can you locate a film of your choice via the search input field with ease? | Pass | Fail | See section 7.1.1 |
| 5 | Can you hover over the trend graphs to display additional information? | Pass | Pass | None |
| 6 | Can you load a film within a reasonable amount of time? | Pass | Fail | See section 7.1.1 |
| 7 | Is the site easily accessible via the URL provided? | Pass | Pass | See section 7.1.1 |

Table 8 - Front-end application usability test results

The results from the usability tests were fairly reasonable, with only two failures relating mainly to the querying of films. The outcome of these tests and how they were resolved are covered in detail in section 7.1.1.

## Regression metrics

For a regression algorithm, there are a few metrics that can be used to gather information relating to how accurate the model that has been trained is. The two metrics that were used for this project were, explained variance and mean absolute error.

The explained variance metric helps identify how spread apart are the prediction results from their mean value. So, a regression model like the decision tree regressor essentially looks for the inter-linking relationships between the independent variables (the features used to train the model) and the dependant variable (the target value that is being predicted). The closer the value is to 1, the more accurate the model should be. There are however other factors that can affect a model’s explained variance.

The mean absolute error on the other hand, is the average spread of errors from the dataset, the differences between the actual results and the prediction results. If these values were to be plotted on a graph whereby the X values were the actual (true) results and the Y values were the prediction results, in machine learning the mean is usually represented by 0. So, imagine a straight-line being drawn where both X and Y = 0, the prediction values closer to that line illustrate a higher level of accuracy as the mean spread of errors will be lower. A MAE that is negative (below the mean) reveals an underestimation in the predicted values, whereas a positive MAE represents an overestimation in the predicted values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Regression Metric | Result | Training Data Used | Additional Regressor Parameters | Action Taken |
| 1 | Explained variance | 0.1535 | All training data | max\_depth = 2 | Retrain |
| 2 | Explained variance | 0.0512 | All training data, with NULL values ignored | max\_depth = 2 | Retrain |
| 3 | Explained variance | -0.5685 | All training data | max\_depth not set | None |
| 4 | Mean Absolute Error | 32.3410 | All training data | max\_depth not set | None |

Table 9 - Regression metric results table

Both metrics assisted with helping the developer better understand the accuracy of the model that had been trained and from Table 9, it is evident that the model is overestimating the prediction values and that a lot of work needs to be carried out in improving the overall accuracy of the model. The details of the accuracy predictions are discussed in section 7.1.2.

## Testing limitations

# Discussion: Contribution and Reflection

## Discussion

The section prior to this illustrated the results of unit, compatibility and usability testing that was carried out during the development of this project. This also included, utilising regression metrics to identify the accuracy of the model selected. This section will aim to discuss the discoveries of the tests in more detail and look to fix any major flaws that were identified.

### Known issues

The testing that was conducted as part of section 6, exposed some important features of the application that needed improving, mainly to do with the usability of the application.

|  |  |  |  |
| --- | --- | --- | --- |
| Reference ID | Description | Priority | Action |
| 1 (Unit – Front-end) | On mobile devices, the navigation menu reduces to a togglable button, but this was not functioning as intended | High | Completed |
| 4 (Compatibility) | Display of film data and input field functionality not functioning on Windows 10 Desktops using Internet Explorer 11 | Low | To do |
| 4 (Usability) | Difficult to locate certain films using the input field | Medium | Completed |
| 6 (Usability) | Initial loading of the search results and autocomplete/suggestion features are sometimes slow | Low | Completed |
| 7 (Usability) | The site does not have secure communication via HTTPS | High | Completed |

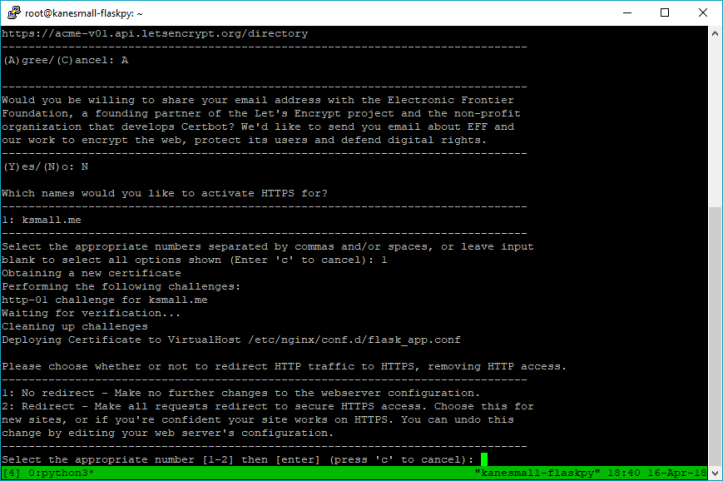
Table 10 - Known issues table

When it came to correcting the issues referenced in *Table 5* above, the higher priority tasks were resolved first. This is because, these issues are more likely to result in missing or reduced application functionality or potential security risks.

To begin with, the first issue related to being able to access the menu items (home/trend page links and search bar) on mobile devices. On mobile devices, the menu button that is supposed to toggle when click and reveal the navigation items and search bar was not functioning. This is a major problem, as it prevents mobile users from being able to use the application for its intended purpose. To rectify the issue however, was an extremely easy fix. It was a matter of making sure that the *div* element containing the menu items had an id value that matched the *data-target­* element attached to the togglable button.

Upon configuring the Flask application on a DigitalOcean droplet, a valid certificate had not been acquired from a Certificate Authority (CA) at that time. However, there are plenty of sources that offer free certificates, the most notable being Let’s Encrypt (Let's Encrypt, 2018). As this application does have shell access to the server over SSH, an automated certificate issuance and installation tool can be used called Certbot (Certbot, 2018). The instructions are straight forward to follow and allowed for an extremely smooth installation process.

Figure 3 below illustrates a selection of the type of questions that were asked during the setup and installation process. Most notably the last question, which asks whether or not all HTTP traffic should be redirected to HTTPS, removing HTTP access as a result. Option 2 was selected and this was for two reasons. The first being that, HTTPS access is more secure and should be used across the web. In July 2018, Chrome will be one of the first browsers to mark non-HTTP as *not secure* with the release of Chrome 68 (Google, 2018). The second reason behind implementing HTTPS access is due to the way in which in the API handles credentials that are used to connect to the database. When passed in plaintext and using an HTTP connection, the credentials would be readily available for any experienced hacker to grab them in transit. However, by implementing an HTTPS connection and forcing all users to do the same, it is not possible to access the credentials in transit thus implementing basic API authentication.

Figure 3 - Certbot installation via PuTTY

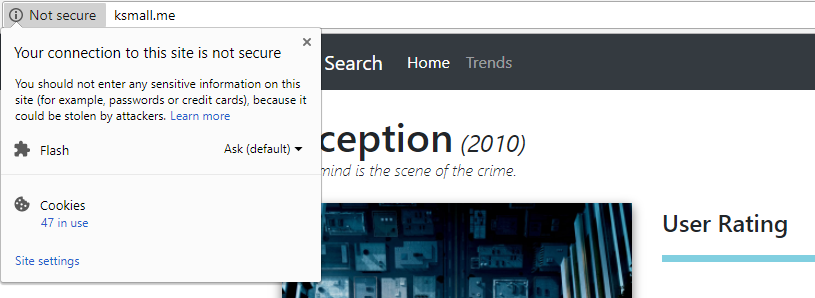
The browser that the user is viewing the application will update from saying that the connection is not secure to the site as illustrated in Figure 4.

Figure 4 - HTTP (Not secure) connection

To a message stating that the certificate for this site is valid and that the connection *is* secure. The URL will now also include the HTTPS protocol and is illustrated in Figure 5 below.

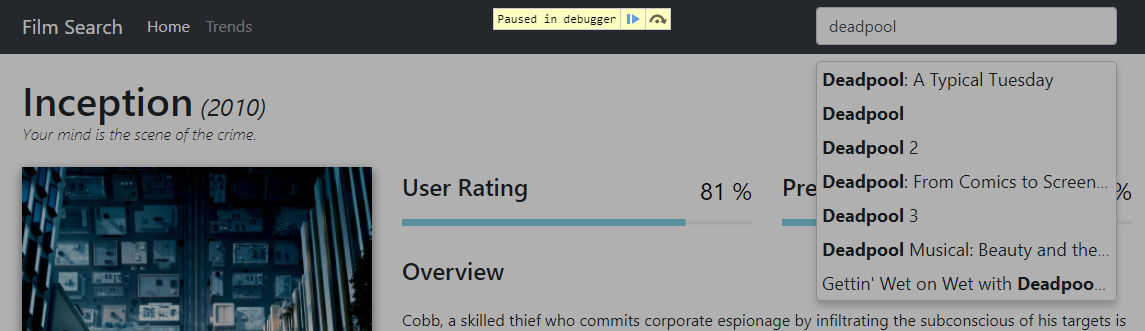
Next up was the issue relating to locating certain films using the input field. The user who marked this usability test as a fail informed the developer that this was due to the way in which the results were returned after a query had been entered making it difficult to locate certain films. To rectify this issue, additional film metadata was added to the query results so that users could better pinpoint films with similar titles, or films with the same name due to a rebooting of the series for example. This is illustrated by the comparison of Figure 4 and Figure 5.

Figure 5 - HTTPS (Secure) connection

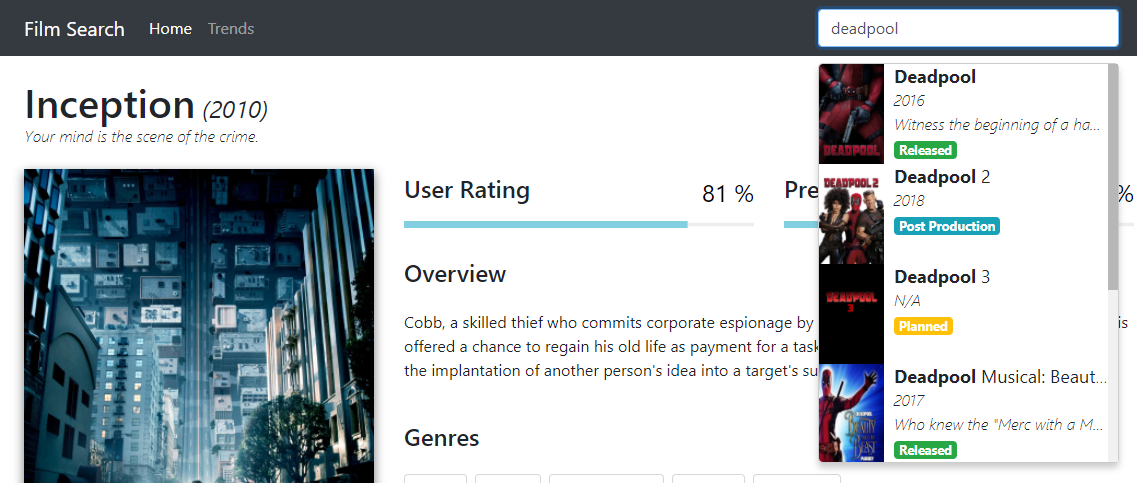
Figure 6 - Query results (old)

Figure 7 - Query results (new)

Next, the issue relating to the load time of the query results which was only documented by one of the users as a fail. The user in question stated to the developer that it was not an issue all of the time, but did occur on a rather regular basis. They also mentioned that the load times were already more than reasonable, but that an increase in speed would result in a much more fluid experience for the user.

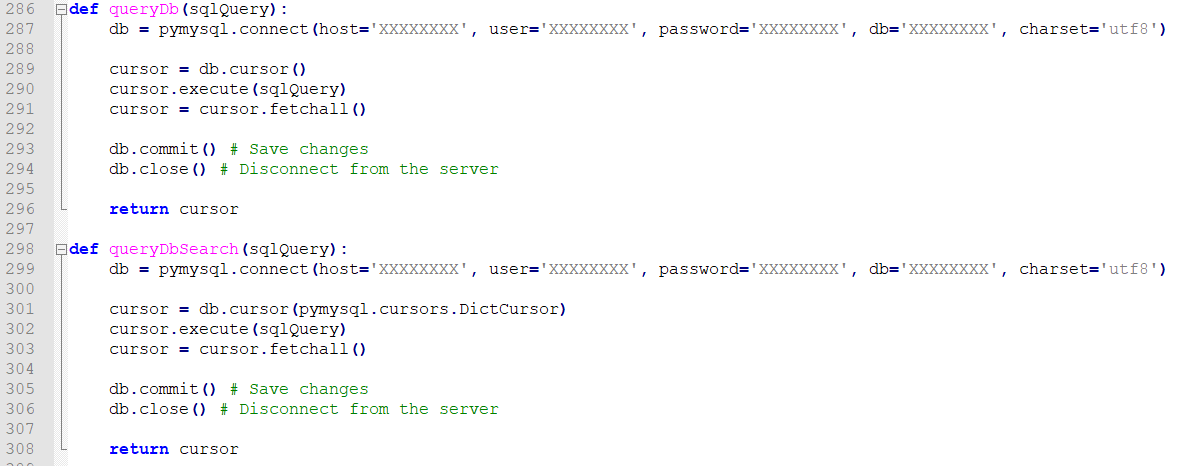
This issue required a bit more attention than the last, mainly due to the way in which the search results were queried via the API. But after a bit of research, a pymysql has a class (pymysql.cursors.DictCursor) that can be passed to the *cursor()* object as a parameter allowing the cursor to return the results as a python dictionary (pymysql.readthedocs, 2018).

Figure 8 - Database query function comparison

As illustrated by Figure 6 above, by duplicating the already existing database connection function and altering the cursor parameters, it would now be possible to utilise this function within the correct API route. Now, as part of the */api/search/films* route, because the *queryDbSearch()* function will now return a python dictionary of results, this can simply be serialised using jsonpickle and returned as a response to the web page. As evident in Figure 7, the code that has been commented out was responsible for mapping each element returned from the SQL query to a python class. This was the part of the code that was rather expensive and was causing slow load times for the query results.

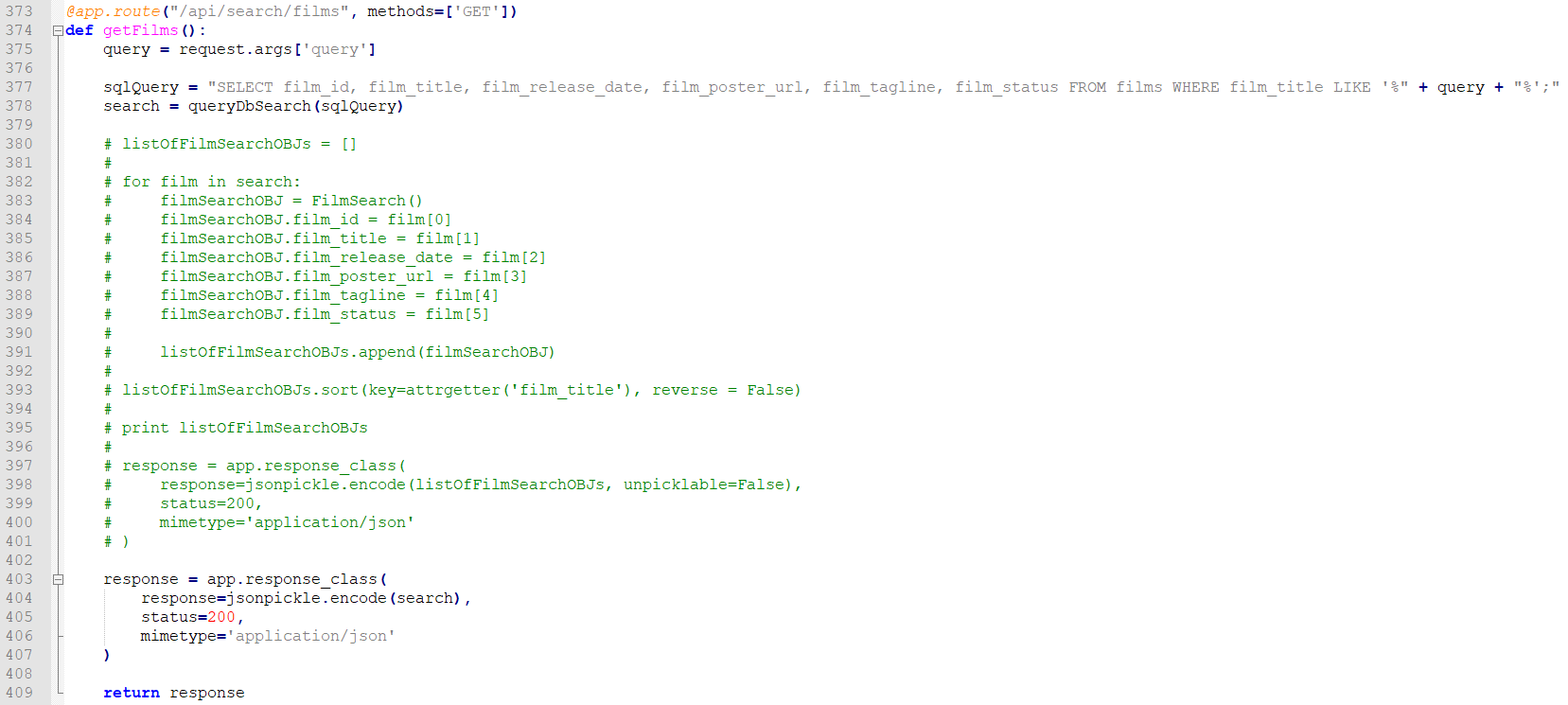
Lastly, there is the issue relating to the displaying of data and input field querying not functioning as intended whilst using a Windows 10 desktop running Internet Explorer 11. From a cursory overview within Internet Explorer 11 using the in-built developer tools, it appeared that the *progressbar.js* library being utilised to display the animated user and predicted rating progress bars, is not supported for the Internet Explorer 11 browser. Due to the fact that it would have taken a while to locate a replacement library for generating animated progress bars that was compatible with Internet Explorer, this issue has been left with the status of *To do*. It was also more important to ensure that support/compatibility for the most popular browsers was the main priority and as can be seen on *statcounter* (statcounter, 2018), Internet Explorer is only used by 3.13% of total users, whereas Chrome, Safari and Firefox total of 77.88% of users worldwide.

Figure 9 - /api/search/films route

### Prediction accuracy

The results shown are difficult to make assumptions on by themselves, but it is clear that the model trained isn’t exceedingly accurate and does overestimate the prediction values. This could be a result of a number of factors and when it comes to machine learning in general, there are so many factors to consider that may have influenced the end results.

**Model selection**

When it comes to selecting the appropriate model for a particular dataset, there isn’t necessarily a full proof method, lots of different variables affect the decision. Scikit-learn provide a *cheat sheet* (Appendix 4: Scikit-learn algorithm cheat sheet), which can be used as a starting point for understanding which types are better suited toward specific datasets. However, this should not be the only tool used when selecting a model.

Even today, one of the best approaches to selecting the best model for your dataset is trial and error (Sief, 2018). Implementing, training and then comparing the prediction accuracy results of each will give a better understanding of the factors that improve accuracy. But, due to time constraints, training different models was simply not possible considering how long it takes to do just that.

That is why, in the future it would be best to try implementing different models and continually attempting to improve feature selection and pre-processing techniques to learn which methods and which combination of methods, result in the most accurate results. A potential neural network solution could even be something to consider, as they are exceedingly effective at modelling highly complex non-linear relationships (Sief, 2018) and that is what this dataset has a lot of. More details on the future improvements that have been considered however, can be found in section 9.2.1.

**Feature selection**

Another reason for the results attained could simply be a consequence of the features that have been selected. The features, budget, runtime, trailer view/like/dislike counts and user rating, might just not have been the right attributes for making predictions on the user ratings. Given more time and a better understanding of feature selection and pre-processing, the model would have also been trained using additional metadata values as features such as, actors, directors, production companies and other film-related attributes that were not acquired for this project. Again, more information on additional feature selection can be found in sections 9.2.5 and 9.2.8.

**Summary**

Ultimately, this isn’t the end of the project even after submission. There are a variety of additional methods to try, new features to acquire and use to train models and even entirely different machine learning solutions to learn about. The main objective of, implementing a regression algorithm to train a model that could make predictions on previously unseen data has been accomplished. It is now the responsibility of the developer to improve the accuracy of those predictions over time, utilising the infrastructure that has been developed over the course of this projects lifetime.

### Application limitations

There are definitely still limitations related to this application, primarily with the absence of interactivity whereby users can create custom search parameters in order to receive a prediction output. But, one could argue that this was never the intended purpose of the project and is simply an idea that has presented itself over the course and nearer to the end of the project’s lifetime. Section 9.2 addresses in detail, a number of improvements that could be made given more time and if the project were to be continued even after university.

## Reflection

As a final year student completing his/her degree, you are required to undertake a final year project that must be completed throughout the last year and alongside other pieces of coursework. I will be looking at how I personally felt at each point during the project’s lifetime and ultimately, what I learned from this experience and how I could improve if given a second chance.

My initial feelings for the project were ambivalent, mainly due to the sheer size of the undertaking and because we had to continually work on the project throughout the year alongside multiple pieces of coursework being set by each module lecturer. However, once I had settled on a project idea and started working on it, because I was working on something that I find of interest to me I think this helped in viewing the project as an enjoyable set of tasks as opposed to mandatory *work*.

One of the worst parts of the project for me was before the year had even started. A few months before beginning our final year, we were tasked with either selecting a pre-defined project from a list that was provided or coming up with our own idea and submitting a PID for it. For my placement I worked at a cyber security company and so for the best part of that year I had been thinking of cyber security related projects that could have been interesting to complete. However, I eventually decided that cyber security was no longer a field I wished to pursue, as I wasn’t enjoying it as much as I was in the beginning. Therefore, completing a project related to that field no longer made sense. So, this left me with a few weeks of having to think of an entirely new project.

The stress and anxiety that the final year project caused me was severe and at times, very overwhelming. Having to juggle separate pieces of coursework for different modules I found challenging enough in the first two years of university, but having this additional project weighing down on you and also knowing that it’s worth a third of the entire year in terms of marks was again, at times too much to handle.

That being said, I have learned with the help of my incredible Mother, certain coping techniques and strategies to alleviate some of the pressures of work similar to this. So, there were definitely good experiences that I had whilst completing my project. I thoroughly enjoyed improving my programming ability with every piece of code I had to write. I decided to build my application in Python, because I believe that I have the most experience with this language over any other that I have used in the past, yet after having completed this project I feel as if I’ve learned so much more about the language then I could have imagined.

Learning how to implement machine learning has been a very interesting journey. I didn’t quite anticipate the level of difficulty that using these tools would provide but learning how to implement them as well as *why* to implement a specific algorithm for example has been really enjoyable. I have only just touched the surface when it comes to machine learning but, with so many industries beginning to adopt this technology it could be a very interesting field to break into after university.

In conclusion, from my experience I now know that it is vital to carry out research first and at the earliest point in development as possible. Before you can tackle any task, you need to have all of the tools, including the knowledge of how to implement *with* those tools.

Looking at my personal experiences and my unique style of working I would say that, industry experts spend months and months, sometimes even years, attempting to perfect machine learning models. I can’t expect to have achieved outstanding results on my first attempt and that shouldn’t hinder me from trying to get better results, trying different models and comparing accuracy scores. Just because this is the end of my project from an academic-perspective doesn’t mean it *has* to be the end of my journey of continuing to expand my knowledge in this vast field.

I think, if I were to undertake this or a similar project again I would try to focus on asking more questions. Making sure that I came prepared to each supervisor meeting with detailed questions, where we could then have discussions about. I would also ensure to utilise strategies to help me, such as recording feedback my supervisor gives me so that I don’t forget what has been said by the time I get home. I need to focus on my worst qualities first next time, such as time management and organisation. If I create a plan of action of *how* I am going to tackle these obstructions, it will be easier when it comes to starting.

# Social, Legal, Health and Safety and Ethical Issues

## Social, legal and ethical issues

When constructing the PID initially no social, legal or ethical issues were identified. However, upon reflection there are a couple of social and ethical issues that could in fact relate to this project.

Firstly, one of the social issues concerning the usage of the application itself relates to users who speak another native language other than English and would therefore not be able to understand the content that is being displayed to them. A simple solution to this problem would be to implement some form of translation feature, possibly with the help of an existing library that supports hundreds of languages and give the user the option to change the language manually. However, a more streamlined approach would be to automatically detect the user’s location via their browser session and to convert the language for them. But, having the manual option present would still be required for users that are visiting the application whilst using a VPN.

Secondly, an additional social issue which again, concerns the usage of the application itself, would be the fact that the application does not cater toward users with disabilities such as visual impairments. There are a variety of solutions of for this type of issue including but not limited to, ensuring that all *<img>* tags are making proper use of the ‘alt’ tag that describes what an image (often used for screen readers), ensuring that the web application is fully-operable via *­just* the keyboard, providing users with the control over text sizing for all elements and having an option for users to be able to alter the contrast levels of the application or include specific colour-blindness modes.

Finally, an ethical implication of this project could be that if the prediction accuracy of the ratings was high enough some could argue that there would no longer be a need for film critics, thus putting a lot of people out of work. A second ethical issue, that follows on closely from the last relates to the emotional side of films. A lot of people believe that computers are not capable of replicating human emotion and therefore, they will never be as accurate as a human critic. There are often films that do draw heavily on a viewer’s emotion, regardless of the known variables such as the genres, budget, trailer view count, etc and therefore it is hard to envisage a scenario in which a computer would consider these parameters when formulating predictions. Such film’s that are considered *cult-classics* would also be extremely difficult to predict accurately for, since a lot of the time these films are liked by viewers regardless of critic ratings/reviews, or factual metadata such as the actors in the film.

## Health and safety risks

There were a few risks that were identified during the creation of the PID relating to the work area and these were eye strain, hardware failure and repetitive strain injury. Eye strain was reduced by taking regular breaks and by utilising an application call *f.lux* (f.lux, 2018), which reduces blue light exposure. To reduce the risk of hardware failure, all documents and code were backed up to a private GitHub repository as well as Google Drive. Lastly, the risk of repetitive strain injury was reduced by taking regular breaks, using an ergonomic chair with lumbar support and height adjustments and by simply adjusting the workstation to appropriately conform to the user’s requirements.

# Conclusion and Future Improvements

## Conclusion

To conclude the report and the project as a whole, the overarching objective to produce an application that makes predictions on user ratings for unreleased films was a success. In order to carry out the objectives, research had to be conducted on the technique of regression modelling to implement, as well as the libraries that would accommodate machine learning algorithms into a python coding environment, various underlying technologies/frameworks were also researched in order to determine how the application could be built and accessed by the end user.

By completing the sections within this report, the main objectives have been achieved. Data has been acquired from a viable source that provided all of the metadata required, this data was then stored in an appropriate normalised database structure, a suitable python framework was selected that would form the base of the entire application, the application itself was configured on a robust and reliable DigitalOcean Droplet which at the same time future-proofed the application and lastly; the features selected were influenced by the results of a questionnaire.

For the data cleansing/pre-processing phase, three separate pre-processing techniques were used to properly transform the data into a format that the regression algorithm could handle. Imputation of NULL values, replaced all NULL entries throughout the dataset with the arbitrary value of -1, one-hot encoding converted the categorial genres feature into separate unique fields that were then represented by binary values and PCA reduced the number of dimensions from a total of 42 down to 2, resulting in an appropriate X input value for the regression algorithm. The regression algorithm itself was selected after research had been conducted, with a decision tree learner being chosen as part of the scikit-learn library for its benefits when working with large and complex datasets that are attempting to predict a numerical output. This was then trained and then used to make predictions, by learning from the scikit-learn library and making reasonable adjustments when needed.

For a first attempt at implementing any form of machine learning logic, the developer has managed to produce a working regression model and that in its own right is a success. From these initial results and using what has been learned throughout the entire project, the results can be improved upon as the project transitions past this deadline.

Researching the techniques used to implement a strong and robust API, along with the routes to access it was a challenging process, but like the rest of the objectives this one also resulted in success. The API can serve data from the database to the front-end web pages, either by predefined identifiers, or by unique queries. It was also secured with basic authentication as a result of carrying out useful usability testing, which makes it more difficult for attackers to access the credentials being transmitted for use when interfacing with the database.

Lastly, the front-end design and user experience objective has been satisfied. This in-part was due to the use of the bootstrap grid system, which handles different user configurations well, such as different resolutions and helps with making the application responsive. But, also thanks to the layout and positioning of the elements on the page, making sure to house the most important information at the top of the page and sticking with this method for the rest of the metadata.

As a final point, there is a lot to take away from this project, not only from a technical stand point but also from a soft skill-perspective. It has enabled the developer to improve their abilities on all fronts, from programming efficiency and code management, to organisation and time management; all of which can be utilised moving forward into a career within the field of computer science.

## Future Improvements

A lot of time and thought has gone into how this project could be improved if more time had been allocated, or simply as the project continues to grow past a university project.

### Improving prediction model accuracy

One of the most important considerations is improving the accuracy of the prediction model. This is the core feature of the application and being able to specify to your users that this application is capable of predicting user ratings for unreleased films, to certain high-level degree of accuracy, would encourage a lot more people to use the platform.

In machine learning it is common practice to implement multiple algorithms to see which is best suited toward your specific dataset. So, with more time it would be possible to train different models and compare the accuracy scores (explained variance, MAE) of each to see which is the most precise at predicting user ratings.

### TV show support

Implementing TV show support as well as film, would allow for the application to have a lot more range and to appeal to a wider audience. TMDb (TMDb, 2018) already allows developers to retrieve metadata for TV shows, so by utilising the same scripts that have already been created for the film data acquisition and altering them slightly all TV show metadata could be acquired.

### Video game support

Further to the addition of TV show support, once an effective model has been trained and can produce accurate prediction ratings on a regular basis for film and TV, video game support could also be added to the application. This would again, increase the range of the audience by tapping into an ever-growing and fast-paced industry. If users and game developers alike, had the ability to predict how well a game would do, it would provide users with the knowledge of whether or not the game is worth purchasing and developers the information to decide whether or not to put time and money into creating a game that may or may not be well-received.

### Additional trend graphs

It would be advantageous to provide the user with additional trend graphs via the *trends* page, but also allow for more user interactivity by having some form of filtering option available on the existing and any future graphs. An added interactive feature could include the option to search for a specific production company for example and to show a variety of trend analytics specifically for that search query.

### External ratings

Requesting ratings from other external sources such as Rotten Tomatoes and/or Metacritic would not only allow users to see all ratings for a film in a singular location but could also help influence prediction models by providing these specific ratings as additional features. Rotten Tomatoes for example, has a *want-to see list*, which indicates how many users have added an unreleased film to their want-to see list. So, utilising this feature could help increase the accuracy of the model by revealing the percentage of users that are already interested in seeing the film.

A similar idea could be applied to both Twitter and Facebook pages whereby, users can follow or like pages that have been created for upcoming films. But, this could be taken a step further and tweets and/or Facebook comments could be analysed using sentiment analysis to determine whether the content is positive or negative and again used as additional features for training the model.

### Updater function

This is an important function to have for an application like this one, whereby on a set schedule (once a week, or once a month) a function would trigger that would be responsible for looking at all existing data in the database and comparing it against The Movie Database API to see if any new films have been added, but more importantly to check if any existing films have been updated or altered in anyway. If a film’s metadata has changed, then the function will retrieve the new data and replace the old data in the database.

Once updated or additional data has been acquired, the function will then be responsible for re-training the model to account for the new data. This will ensure that as new data enters the system, all prediction rating values are also updated to better reflect any changes in the relationships that the model had previously been trained on.

### Targeting businesses

By adding additional functionality to the application whereby allowing users to configure their own custom test cases for unreleased films, this application could be targeted toward production companies. These industry experts are always trying to find the *winning* formula for a film and one that will generate the highest amount of revenue. If they had the ability to compute different combinations of variables such as, which actors, or which director would best suit an upcoming film based on the predicted rating output, then it would be extremely beneficial and lucrative for them.

### Oscar nominees and winners

A very interesting model feature was considered toward the end of the project and as such, has been constructed into a future improvement. The idea centred around utilising Oscar nomination, winner and potentially other film award or accolade data, as an additional feature when training the model; which could have a huge advantage on the outcome of the prediction values. It would work so that, whenever a film is either nominated or wins, a unique category would then be populated for each attribute of the film. Taking this year’s Academy Awards for example, Blade Runner 2049 won an Oscar in best visual effects category (Donnelly, 2018). So, each film attribute, the genres, the actors, the director, the production company and so on; would also receive numerical value appended onto this unique category mentioned above. Each value could also be weighted differently depending on the category in which it has originated from and depending on whether the end result was simply a nomination or an actual win.

# References

Amazon Web Services, Inc, 2018. *What is NoSQL? – Amazon Web Services (AWS).* [Online]   
Available at: https://aws.amazon.com/nosql/

Bird, S., Klein, E. & Loper, E., 2009. Preface. In: *Natural Language Processing with Python: Analyzing Text with the Natural Toolkit.* s.l.:O'Reilly Media, p. ix.

Certbot, 2018. *Nginx on Ubuntu 16.04 (xenial) - Install.* [Online]   
Available at: https://certbot.eff.org/lets-encrypt/ubuntuxenial-nginx

Donnelly, J., 2018. *Oscar Winners 2018: See the Full list.* [Online]   
Available at: http://oscar.go.com/news/2018/oscar-winners-2018-see-the-full-list

f.lux, 2018. *justgetflux.* [Online]   
Available at: https://justgetflux.com/

GitHub - Tensorflow, 2018. *tensorflow/skflow.* [Online]   
Available at: https://github.com/tensorflow/skflow

GitHub, 2018. *twitter/typeahead.js.* [Online]   
Available at: https://github.com/twitter/typeahead.js

GitHub, 2018. *twitter/typeahead.js - Bloodhound.* [Online]   
Available at: https://github.com/twitter/typeahead.js/blob/master/doc/bloodhound.md

Google, 2018. *A secure web is here to stay - Google Online Security Blog.* [Online]   
Available at: https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html

IMDb - Datasets, 2018. *IMDb Datasets.* [Online]   
Available at: https://www.imdb.com/interfaces/

IMDb, 2018. *Batman Begins.* [Online]   
Available at: https://www.imdb.com/title/tt0372784/

IMDb, 2018. *Because of Winn-Dixie.* [Online]   
Available at: https://www.imdb.com/title/tt0317132/?ref\_=fn\_al\_tt\_1

IMDb, 2018. *IMDb | Help - Weighted Average Ratings.* [Online]   
Available at: https://help.imdb.com/article/imdb/track-movies-tv/weighted-average-ratings/GWT2DSBYVT2F25SK?ref\_=cons\_tt\_rt\_wtavg#

IMDb, 2018. *IMDb | Help - What is IMDb?.* [Online]   
Available at: https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref\_=helpsect\_cons\_1\_1#

ISTQB Exam Certification, 2018. *What is unit testing?.* [Online]   
Available at: http://istqbexamcertification.com/what-is-unit-testing/

Let's Encrypt, 2018. *Getting Started.* [Online]   
Available at: https://letsencrypt.org/getting-started/

MDN Web Docs, 2018. *Introduction to web APIs.* [Online]   
Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side\_web\_APIs/Introduction

MDN Web Docs, 2018. *Working with JSON.* [Online]   
Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

Metacritic, 2018. *About Us - Metacritic.* [Online]   
Available at: http://www.metacritic.com/about-metacritic

Metacritic, 2018. *How We Create the Metascore Magic - Metacritic.* [Online]   
Available at: http://www.metacritic.com/about-metascores

OMDb, 2018. *OMDb API - The Open Movie Database.* [Online]   
Available at: http://www.omdbapi.com/

Otto, M., 2018. *Bootstrap.* [Online]   
Available at: https://getbootstrap.com/

Patreon, 2018. *Brian Fritz is creating The Open Movie Database API | Patreon.* [Online]   
Available at: https://www.patreon.com/omdb

pymysql.readthedocs, 2018. *Cursor Objects - PyMySQL 0.7.2 documentation.* [Online]   
Available at: http://pymysql.readthedocs.io/en/latest/modules/cursors.html#pymysql.cursors.DictCursor

Roberts, I. & Yan, L., 2014. *Vector-based Sentiment Analysis of Movie Reviews,* s.l.: Stanford University.

Rotten Tomatoes - Developer Network, 2018. *Fandango - Welcome to the Rotten Tomatoes® Developer Network.* [Online]   
Available at: https://developer.fandango.com/rotten\_tomatoes

Rotten Tomatoes, 2018. *Batman Begins.* [Online]   
Available at: https://www.rottentomatoes.com/m/batman\_begins/

Rotten Tomatoes, 2018. *Because of Winn-Dixie.* [Online]   
Available at: https://www.rottentomatoes.com/m/because\_of\_winndixie

Rotten Tomatoes, 2018. *Rotten Tomatoes: About.* [Online]   
Available at: https://www.rottentomatoes.com/about/

Saraee, M., White, S. & Eccleston, J., 2004. A Data Mining Approach To Analysis And Prediction Of Movie Ratings. *WIT Transactions on Information and Communication Technologies,* Volume 33, p. 10.

Sas.com, 2018. *Machine Learning - What it is and why it matters.* [Online]   
Available at: https://www.sas.com/en\_gb/insights/analytics/machine-learning.html

Scikit-learn, 2018. *1.10. Decision Trees — scikit-learn 0.19.1 documentation.* [Online]   
Available at: http://scikit-learn.org/stable/modules/tree.html#tree

Scikit-learn, 2018. *An introduction to machine learning with scikit-learn.* [Online]   
Available at: http://scikit-learn.org/stable/tutorial/basic/tutorial.html

Scikit-learn, 2018. *Documentation of scikit-learn 0.19.1.* [Online]   
Available at: http://scikit-learn.org/stable/documentation.html

Sief, G., 2018. *Selecting the best Machine Learning algorithm for your regression problem - Towards Data Science.* [Online]   
Available at: https://towardsdatascience.com/selecting-the-best-machine-learning-algorithm-for-your-regression-problem-20c330bad4ef

statcounter, 2018. *StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share.* [Online]   
Available at: http://gs.statcounter.com/

Tensorflow, 2018. *An open source machine learning framework for everyone.* [Online]   
Available at: https://www.tensorflow.org/

TMDb - API, 2018. *API Documentation - TMDb.* [Online]   
Available at: https://developers.themoviedb.org/3/getting-started/introduction

TMDb - Rate Limiting, 2018. *API Documentation - Request Rate Limiting.* [Online]   
Available at: https://developers.themoviedb.org/3/getting-started/request-rate-limiting

TMDb, 2018. *Get TV Details API Documentation.* [Online]   
Available at: https://developers.themoviedb.org/3/tv/get-tv-details

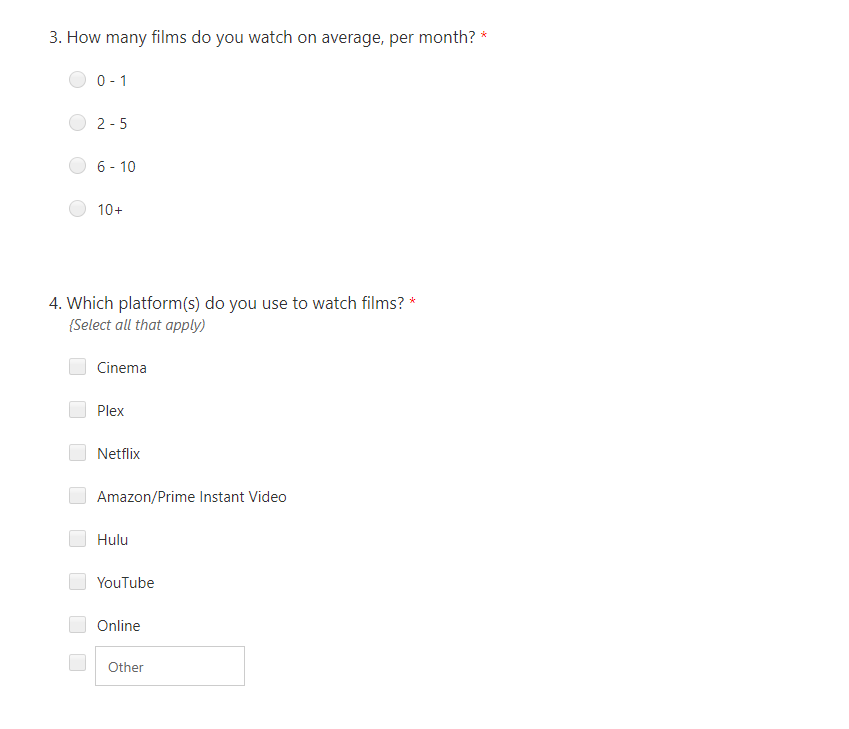
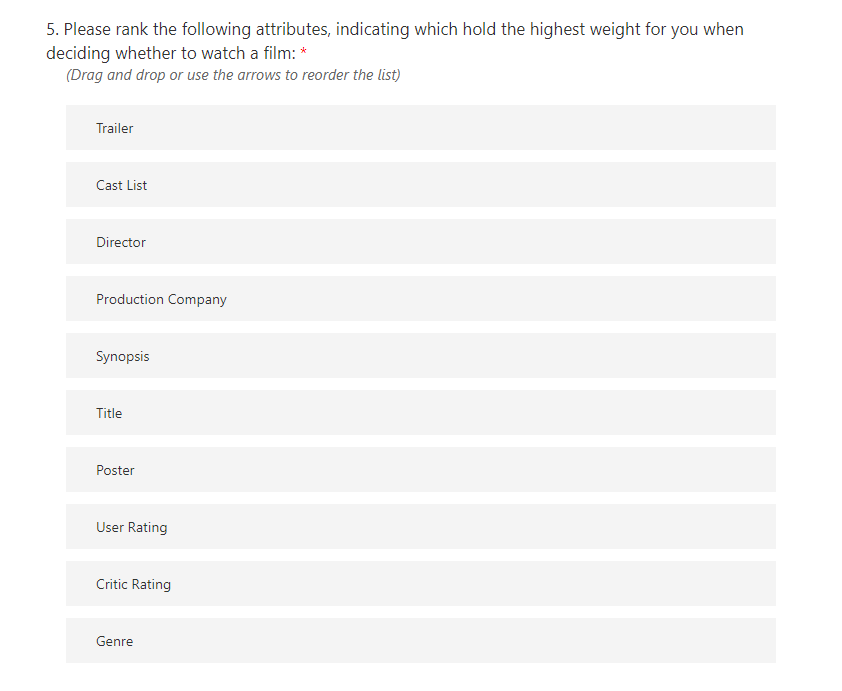
TMDb, 2018. *New APIs for external reviews and awards — Talk — The Movie Database (TMDb).* [Online]   
Available at: https://www.themoviedb.org/talk/5904943092514169c000c5f8

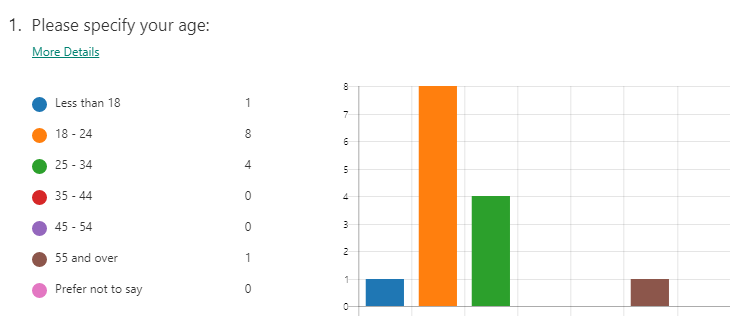
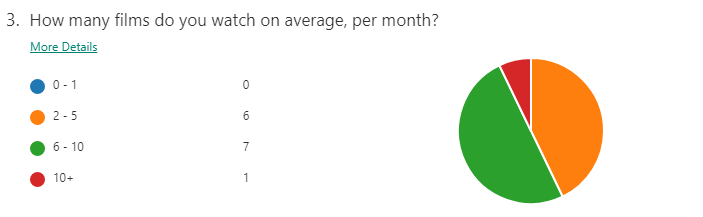
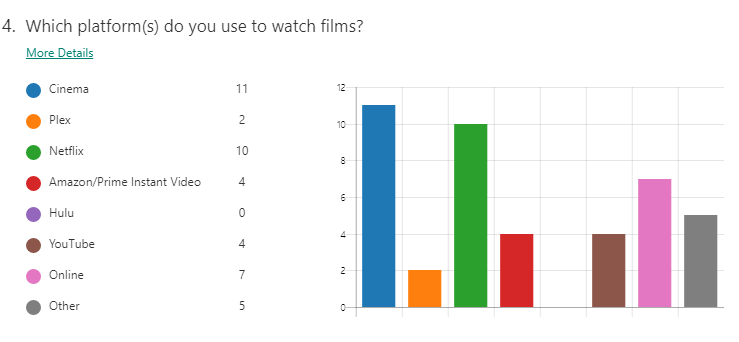
# Appendices

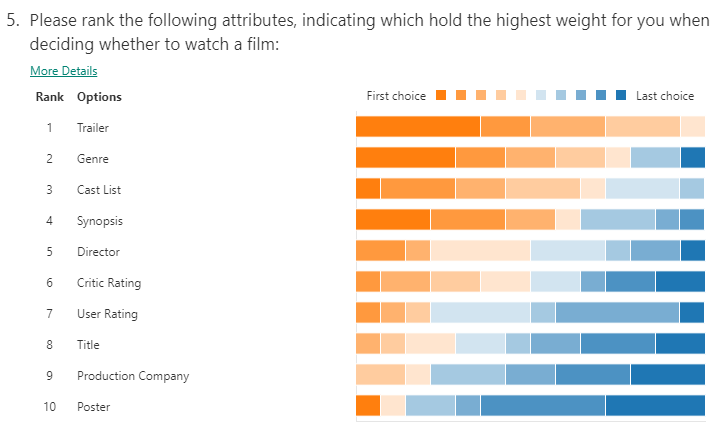
## Appendix 1: Project Initiation Document

## Appendix 2: Logbook

## Appendix 3: Questionnaire







## Move mouse over imageAppendix 4: Scikit-learn algorithm cheat sheet