Predicting Movie Ratings with Data Mining
Final Project Report

DT288
BSc in Computer Science

Povilas Kubilius

C16370803

Leo Tilson

School of Computer Science

Technological University, Dublin

04/04/2020

Abstract

The goal of this project is to predict the success of a planned movie, what ratings it will get, how much income it will make in the box office, based on planned input variables such as movie budget, genera and cast. This will be a web application, where users will be able to fill out these planned movie details and my models will make a prediction and show the user what type of rating and revenue the planned movie could get. The data will be gathered about movies from available to download datasets and use web scraping techniques to fill in any gaps in the data and acquire any additional needed information. After cleaning and processing my datasets, the movie data will be used to train an artificial neural network model to make predictions on the success of movies. The project end goal is to host the model online on the interactive web application. Using the entries from the end user, web scrape any necessary additional metadata about the entries, such as how many awards does the actor which was inputted by the user have, or how many movies has a director directed and what ratings those movies had, and then use my pre-trained models to make the predictions.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Povilas Kubilius

Povilas Kubilius

10/04/2020

Acknowledgements

I would like to thank my family and friends for the support during the development of this project. I would like to acknowledge and thank my project supervisor for the support and guidance during the weekly meet ups. His help and insightful advice were invaluable in completing the work done so far on this project and on this report. I want to thank my best friends who have strength in areas where I lacked knowledge and would sit down with me to explain key concepts. It went along way to help me achieve the goals of this project and I am forever grateful.

Table of Contents

1. Introduction

1.1. Project Background

What makes a movie successful? We have all watched movies that evoked strong emotions in us. Perhaps it was the relatable characters, or maybe a resonant ideology that was presented, or maybe simply the visuals and art in the movie were so impressive that it left its mark on you. Storytelling is a core aspect of our cultures. We have told stories for as long as we can remember, and we don't even know what was the first story was ever told [1]. Even back into prehistory, humans drew pictures on cave walls to tell stories. As humans evolve and develop more advanced tools and civilizations, so do the methods of storytelling developed. Before civilization, we told stories only by word of mouth, then with the discovery of fire we also discovered that charcoal is great for making marks on stone walls, Chauvet Cave is probably one of the most famous examples of prehistoric cave paintings. Later with the discovery of writing, it transformed how we tell stories again, taking on the form of symbols on clay tablets. One of most ancient surviving literary work we have today is the Ancient Sumerian "Epic of Gilgamesh" [2], composed in Mesopotamia around 1800 BC, but only second after the Pyramid Texts in Egypt which have been dated to about 2400–2300 BC [3]. Surely these stories didn't survive thousands of years only because they were written on persistent material, because if that were the case then we would a lot more clay tablets of ancient literature. It is more likely that the Epic of Gilgamesh was such a profound tale to the ancient people of Mesopotamia that they made effort to make copies of the clay tablet and put in effort into keeping them safe for preservation. Today the most advanced form story telling comes in the form of digital media, such as movies, TV series and even video games.

But what makes a story popular, successful and with more chance to survive into the future for future generations to hear or watch these stories? Is it solely dependent the story itself? Do all good stories become successful based on their merit alone or maybe there are more factors involved? Perhaps the way the story is told is more important than the story itself? Two different people can retell the exact same story and one can bore us to death, and the other can grip our attention with such intensity that we sit on the edge of our seats, totally immersed in the plot of the story being told. The same is probably also true of movies. The production, cast and delivery of the movie can have a major impact on the movie's ability to grip us and leave a strong impression on us or leave us bored, forgetting about the movie after a few days. This project aims to explore the relationships between production variables in making a movie and how successful the movie was after its release. How does one measure the success of a movie in the first place? Everyone has their own subjective view on which movie was amazing and which ones disliked, but yet there is a general consensus of the majority on which movies were generally better than others. Main gauge used in determining this are movie ratings and reviews. Ratings by the regular audience and ratings by movie critics. This places a numerical value on movie success. One can even consider the revenue the movie produces as a measure of success. By placing numerical values on production variables, such as the budget to produce the movie, the amount of awards the main actors have, the success of previous movies directed by the director, we can use computational data analytic techniques to spot patterns and correlations between production variables and the variables which dictate the success of the movie.

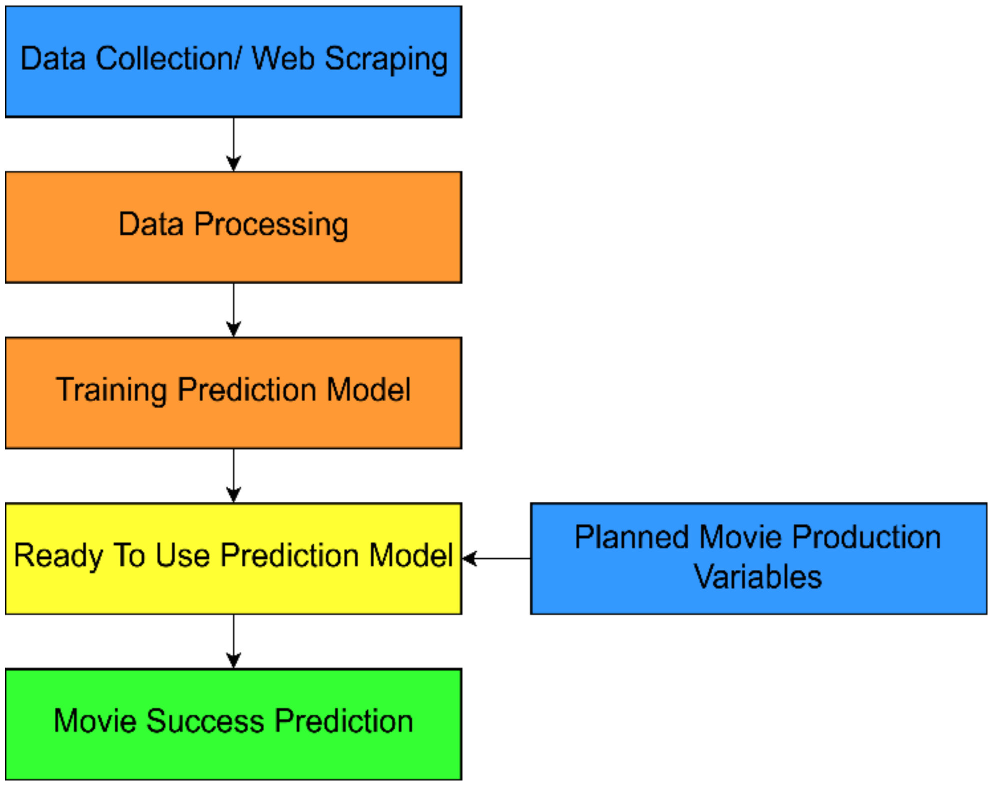
1.2. Project Description

This project will use artificial neural networks and machine learning to learn about the production variables and the star ratings of thousands of movies from the past, and using these models predict the star ratings of planned movies once given the planned production variables, such as the budget of the movie, its runtime, the main actors, directors, genre and other relevant factors.

To be able to use this system, it will be made in the form of an online web application. The end users will have an interactive and friendly user interface to fill out the details of the planned production variables and then get an rating prediction on what the rating of the movie would be based on the user's input.

To make the model in the first place, the key aspect of this project will be data acquisition, which then we can mine to acquire valuable information from. Basic datasets will be acquired online from sites like kaggle.com and IMDb. But these datasets will not be enough, to gain more relevant data it will use web scraping techniques to gather relevant data from movie websites like IMDb and Rotten Tomatoes.

The datasets will then be processed into a single table or file in a format that will be readily acceptable to train the artificial neural network. This will include hot encoding categorical data into binary representations, normalizing highly variant data like budget and runtime and placing the data into formatted arrays. The neural network model will train on the provided dataset to predict movie ratings and will also be tested for accuracy on a subset of the dataset which the model has not been trained with.


Data Collection/ Web Scraping
Data Processing
Training Prediction Model
Ready To Use Prediction Model
Planned Movie Production Variables
Movie Success Prediction

The end system will have a client-server architecture. The client side will be what the end users will interface will. The processing of input data, the prediction model and the database will be located on the server. Some variables are constantly changing, such as the number of awards won by the actors and directors, to keep it up to date, the server will web scrape for that data and use it as input for the prediction model to get an estimate on the movie rating and return to the result to be visually viewed by the end users.

1.3. Project Aims and Objectives

The overall aim of the project is to make an estimate prediction of movie ratings with accuracy greater than 70% and to provide a web-based user interface for the system that anyone can use.

In order to have the most accurate possible predictions, a large amount of data is required. Another objective is to compile a full dataset of over 30,000 movies with data about their budget, genre and feature actors. Web scraping will be used to search online movie pages, like IMDB movie pages, for the missing data to create a bigger, completer and more accurate dataset.

The main objective is to create an artificial neural network, that will be trained using data from the database, to make new predictions about planned movie's ratings and box office income, using user inputted data. This will be done by using python libraries which make neural network implementation accessible and flexible.

The end goal for this project will be to make this widely available by having a web application to host this system online and make it easily accessible by everyone on most deceives. This can be broken down into 5 smaller objectives:

- To gather as much data as possible by downloading free available movie datasets such as from Kaggle and from IMDb. To web scrape any additional data absent from these datasets from the IMDb website
- To use the gather data to train an artificial neural network. Using the neural network to produce a prediction movie to predict movie ratings to be used by the web application.
- To create web based front-end, that is user friendly, easy to use and is accessible from most deceives, including mobile phones.
- To create a web server which holds all the logic to process the user input and use that input to make movie predictions with the neural network model.
- To create a database back-end that holds data about movies and actors. The database will need to be updated one a week to ensure that the data is current and accurate.

1.4. Project Scope

This project is a probabilistic estimate of the movie's rating. It cannot predict with total certainty. The project also does not take into account the actual quality of the content of the movie. It doesn't analyse the plot, complexity of the story, character development and quality of acting by the actors. It doesn't analyse the style of the movie or its visuals. It doesn't analyse the content of the movie's scripts, or the title of the movie or its summary. To predict movie success based on its content it would require the movie to have already been filmed and edited for analysis, which at that point is impractical because one would want to know a rough estimate of the movie success before any money is invested into its production. Using sentimental or thematic analysis for things like movie title, plot summary or script doesn't return meaningful numerical data that be used to train an artificial neural network. This project doesn't predict success for TV series. The success of a TV series is highly variable and can depend on season and even episode. First season of a TV series can be highly successful then dwindle over time, so having only initial production data cannot predict the success of a TV series over the long term.

1.5. Thesis Roadmap

Research

This chapter explores the background research done related to the use of datamining and artificial neural networks in analysing data and predicting trends, like movie success. Looking at examples of software that already use artificial intelligence in different areas of the movie production business.

Experimental Design

This chapter discusses the designed for the overall web application system. Exploring the technologies used and the dataflow between the multi-layered online architecture. This chapter also discusses the software development methodologies that will be used using the development of this project. Also discusses the design choices for the neural network,

Experimental Development

This chapter discusses the development on this project. Explaining the techniques being used with the current approach of data mining, implementation of artificial neural network, and development of the web application.

Testing and Evaluation

This chapter outlines the methods which will be employed in testing the software. It also discusses how the web application will be evaluated by test users and what metrics will be used to measure project evaluation.

Conclusions and Future Work

This chapter provides a summer of all previse chapters and provides conclusion about them. Summarizing why the used approaches in this project were right and what advantages they had over alternative approaches. The last section discusses the possible future work that can done with project. Explores possible improvements and which area of the project could be expanded upon, such as adding more functionality and feature to the web application and creating more neural network prediction model to predict other things about movies, such as the box office income.

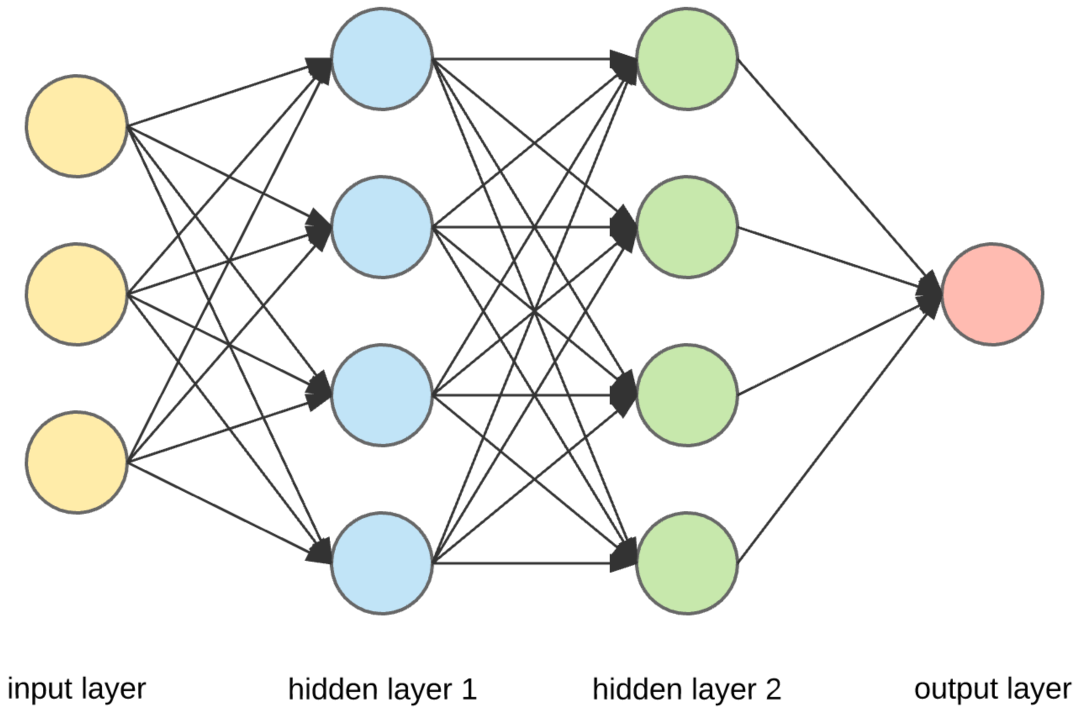
2. Literature Review

2.1. Introduction

This chapter explores the background research related to artificial neural networks, data mining and role of artificial intelligence in movie success prediction. There are a lot of problems in the world and for most of human history it was the humans themselves who could solve their problems. This has worked very efficiently for thousands of years as this is how we developed and progressed our civilisation from the stone age to the modern age of information and technology. As our inventions and civilization became more complex, so have our problems. We now have access a tremendous amount of raw data that we collect with our technology but trying to extrapolate meaningful information about the data is becoming increasing more difficult. Dealing with Big Data is becoming a new problem. Manually analysing Big Data and hoping to find patterns and extracting useful information to make prediction is impractical for humans alone. For example, given 30,000 rows of movies with many columns of metadata about the movie, how would it be possible to predict a planned movie's ratings or box office income? This is where the use of artificial intelligence comes into play. The most powerful aspect of artificial intelligence comes from machine learning techniques such as the use of artificial neural networks. Using the data acquired, we can train the neural networks to find and recognize patterns in the data, creating a model which can be used to make predictions from new observations as inputs. The whole processes of acquiring large data sets, processing them and using artificial intelligence to extract useful information and make future predictions is called data mining.

Artificial Neural Networks

Artificial neural networks are statistical/mathematical models that try to imitate real biological neural networks like the human brain. Artificial neural networks are made up of interconnected nodes, or "neurons", separated out into layers. The connections between the nodes are called synapses, which send signals from one neuron to another. The synapses also have an assigned "weight" to it. Each neuron has an activation function, it calculates the total sum of the weights it received from signals from other neurons, and if the sum is greater than the threshold, it actives and send a single to each neuron it's connected to in the next layer.



What makes Artificial Neural Networks very powerful is backpropagation. When the output of the model does not match the expected output during training, it calculates the degree of error from inputs, expected output and actual output, then iterates backwards through the networks, adjusting the weights and biases of the synapses, in hopes that this will create an output to more closely match the expected output.

Once an Artificial Neural Network has been sufficiently trained, we can input new data into the network to see what the predicted output is. This project will use an artificial neural network, trained on datasets of metadata of previously made movies to then make predictions with new given input.

Data Mining

Data mining is the whole process of finding patterns in large dataset and extracting useful new information. Data mining involves using artificial intelligence, machine learning and statically analysis. Data mining also involves databases and data management, working with pre-existing datasets, pre-processing the data, using machine learning techniques to analyse the data to find new patterns, create statistical prediction models, post-process the raw data from the models then use visualisations to display the data in a meaningful and comprehensive way as to be understood by humans. Data mining can produce models with powerful predictive abilities which can solve business problems and predict trends.

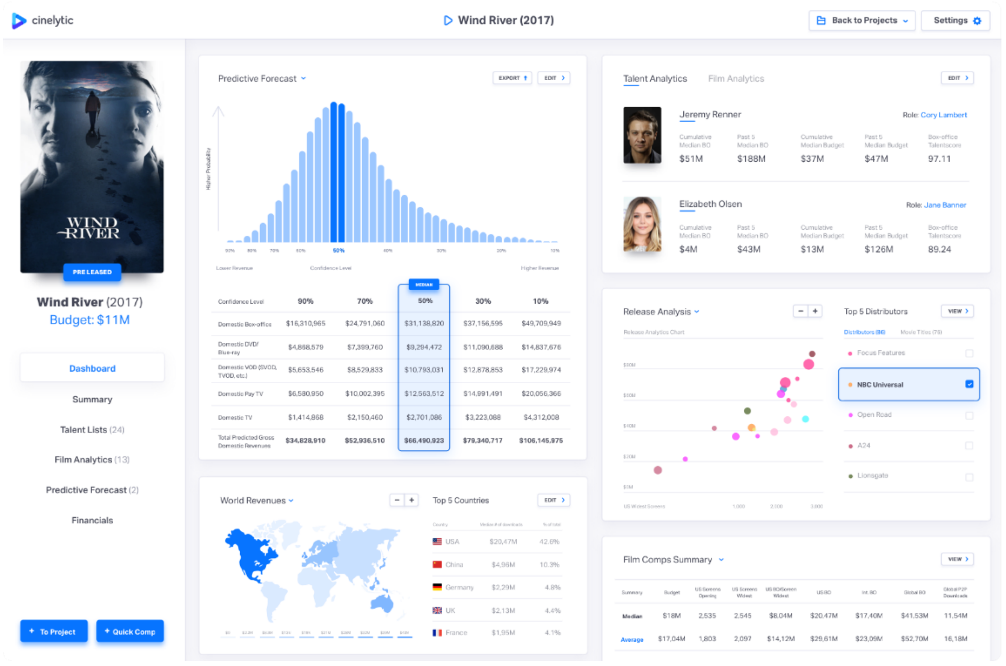
2.2. Alternative Existing Solutions to Your Problem

The use of artificial intelligence in Hollywood's film industry is become more prevalent in the last decade. Making movies can be very expensive, so the production companies want to know if their idea for a movie will be worthwhile making. There are several companies that have started up in the last decade to provide a software solution for this. Companies like Cinelytic, Vault and ScriptBook use data mining and artificial intelligence to make estimate predictions on what will be the movie's box office based on budget, planned actors for the cast and other relevant factors. [4]

Cinelytic

Cinelytic is a software platform which uses artificial intelligence to provide analytic insight for movie studio and independent film makers on possible success and profit for the movie they plan to produce. Cinelytic uses data that is both proprietary and licensed by third parties to create prediction models that allow the users to evaluate the project value and minimised risks. [5]

Cinelytic makes movie success predictions from inputs such as the movie's budget and the main actors cast in the movie. It provides comprehensive visualisation of data and graphs of the results, each after easily understood and business decisions can be made quickly. Cinelytic has been proven to be useful to film makers for "unparalleled accuracy". Cinelytic software doesn't touch on the creative parts of the movie projects. CEO of Cinelytic, Tobias Queisser, said "we want to bring the 'gut' part of the decision down to 60%. The creative part should probably still override, but in order to create a better product, execute and market it better, and find a more financially satisfying outcome, it helps to use a more methodical approach to project evaluation and risk assessment" [6].



Wind River (2017)
Budget: $11M

Confidence Level	90%	70%	50%	30%	10%
Domestic Box-office	$16,310,965	$24,791,060	$31,138,820	$37,156,595	$49,709,949
Domestic DVD/ Blue-ray	$4,868,579	$7,399,760	$9,294,472	$11,090,688	$14,837,676
Domestic VOD (SVOD, TVOD, etc.)	$5,653,546	$8,529,833	$10,793,031	$12,878,853	$17,229,974
Domestic Pay TV	$6,580,950	$10,002,395	$12,563,512	$14,991,491	$20,056,366
Domestic TV	$1,414,868	$2,150,460	$2,701,086	$3,223,088	$4,312,008
Total Predicted Gross Domestic Revenues	$34,828,910	$52,936,510	$66,490,923	$79,340,717	$106,145,975

Jeremy Renner — Role: Cory Lambert

Cumulative Median BO	Past 5 Median BO	Cumulative Median Budget	Past 5 Median Budget	Box-office Talentscore
$51M	$188M	$37M	$47M	97.11

Elizabeth Olsen — Role: Jane Banner

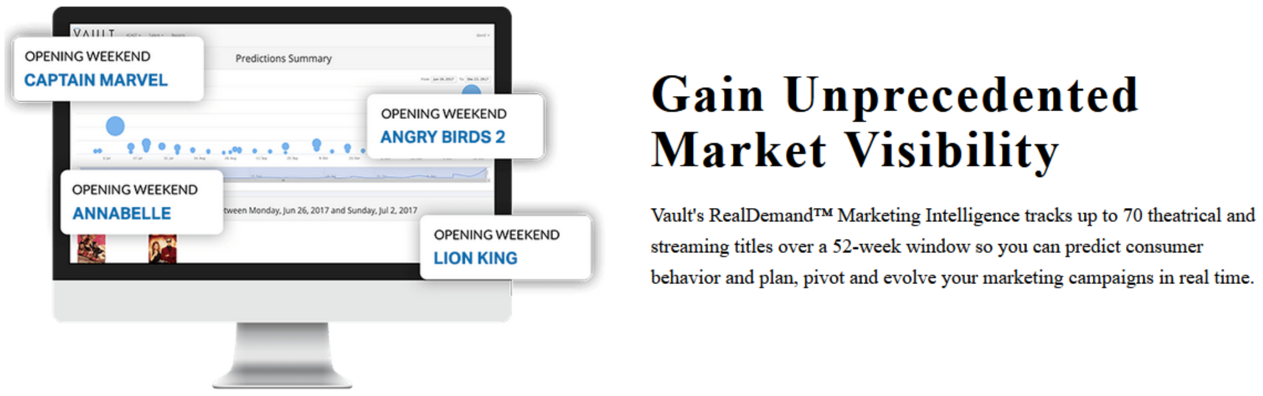
Cumulative Median BO	Past 5 Median BO	Cumulative Median Budget	Past 5 Median Budget	Box-office Talentscore
$4M	$43M	$13M	$126M	89.24

Country	Median # of downloads	% of total
USA	$20,47M	42.6%
China	$4,96M	10.3%
Germany	$2,29M	4.8%
UK	$2,13M	4.4%
France	$1,95M	4.1%

Summary	Budget	US Screens Opening	US Screens Widest	US BO/Screen Widest	US BO	Int. BO	Global BO	Global P2P Downloads
Median	$18M	2,535	2,545	$8,04M	$20,47M	$17,40M	$41,53M	11,54M
Average	$17,04M	1,803	2,097	$14,12M	$29,61M	$23,09M	$52,70M	16,18M

Vault AI

Vault AI is an Israeli company that uses artificial intelligence that analyses the actual content of movies. Vault "reads" the screenplay and analyses the script, things that happen in the scene, character development and even plot structure. Then it is able to make a prediction of the box office income of the movie [7]. Vault AI have several products that offer different type of analysis of movies. One of them is "RealAudience" which uses artificial intelligence to predict who will be the main global demographic who will watch the movie, and how much tractions the movie will get from these audiences, even suggest improvements for the movie to attract even more people of a specific demographic. [8]

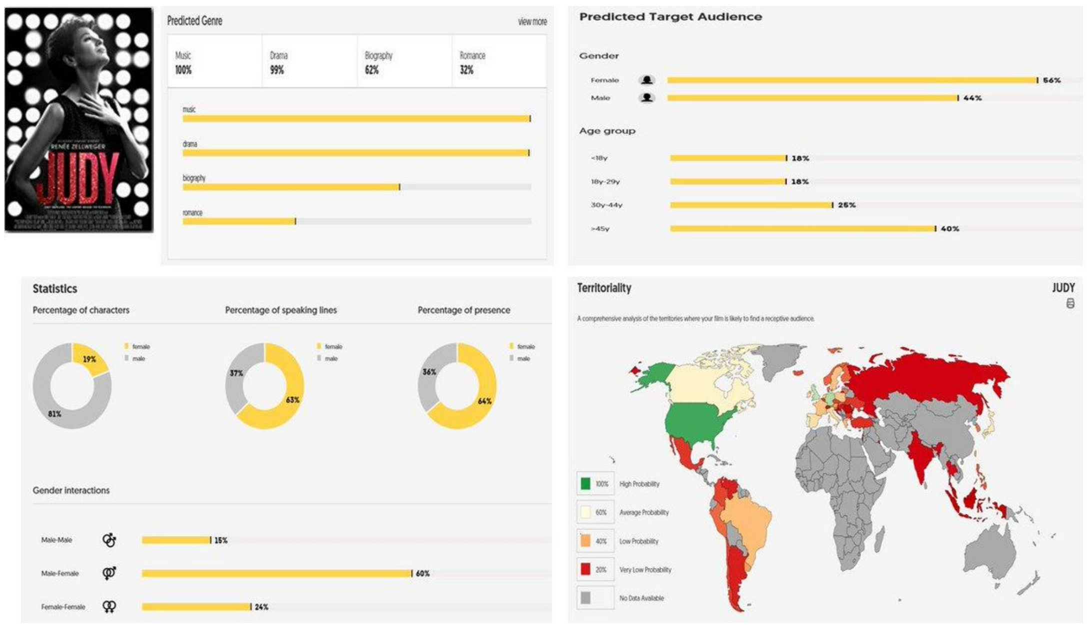


Gain Unprecedented Market Visibility

Vault's RealDemand™ Marketing Intelligence tracks up to 70 theatrical and streaming titles over a 52-week window so you can predict consumer behavior and plan, pivot and evolve your marketing campaigns in real time.

ScriptBook

ScriptBook is a company that uses artificial intelligence with natural language processing to analyse the script and story of the movie. It analyses the characters and story of the movie script and makes predictions of what the audience ratings will be like, which characters are likeable, the genre of the script and even to which audience does the script appeals to most. This doesn't make box office predictions like the previous software do but provides deeper insights into the story itself [9]. ScriptBook have another very interesting product based on artificial intelligence called DeepStory. Having analysed thousands of movie scripts, DeepStory can generate compelling stories and scripts by itself. DeepStory has advanced character awareness, creating characters with their own personalities and traits. [10]



2.3. Technologies you've researched

Machine Learning Technologies

There are several machine learning technologies that implement artificial neural networks. It's is possible to implement an artificial neural network with most object-oriented programming languages such Java and C++. Instead of trying to build an artificial neural network from scratch, it was a better to option to use programming language libraries that allow easy set up and utilisation of artificial neural networks. Python is very popular today due to its simplicity, flexibility and availability of many libraries to be used. This project will use python with third-party python libraries to create, train and use artificial neural networks.

TensorFlow is a python library that provides all machine learning capabilities. TensorFlow is a free and open source library. TensorFlow can be used to implement any type of machine learning technique such as deep learning. Due to this, TensorFlow has a very flexible architecture making this library applicable to most artificial intelligence projects. TensorFlow was chosen for this project due to its wide application and its popularity online means there is a lot of support and documentation for this library.[11]

Keras is another open source python library that implements artificial neural networks. The main advantage of Keras is the ease of use. Keras has a very simple and straight API functions to allow for fast creation of neural networks and testing. Even though Keras is a standalone package, TensorFlow uses Keras as a submodule in its architecture. Keras is the front end of the TensorFlow library. This combination provides a powerful library, with user-friendly API functions to create and train neural networks with the impressive capabilities of the TensorFlow backend giving the library wide application and flexibility. This high level of abstraction allows the focus to remain on the high-level design and implementation of the neural network without having to worry about the low-level technicalities of neural networks. [12]

I have chosen TensorFlow and Keras as technologies to be used in creating and training movie success prediction models.

Web Application Technologies

My goal is creating a web application as the front end for my project. Everyone has access to the internet and hosting my prediction model online gives it wide availability and ease of use. I have chosen python as the programming language of choice for this project, due it the simplicity of use and fast development, this project will also use python-based web framework to host my web application.

Django is a python based free open source web framework. The main advantage of using Django is due to its simplicity of use, plug-and-play architecture and flexibility. This allows for faster design and creation of website without having to deal with the technical low-level details associated with web frame servers. It implements website secure as core of its design allow the developer to avoid the most common security flaws in website design. Django is lightweight and highly configurable. It is compatible with most common sever technologies such as Apache and Nginx. It supports a database backend for most common database technologies as well, such as, SQLite, MySQL and MongoDB. Django is highly extensible and allowed for configuration of third-party application and packages. This project will use Django as the web frame to create the web application due to the many benefits stated above, especially ease of use and rapid development.[13]

As part of the website, I will use JavaScript and Bootstrap to create an interactive and dynamic website. JavaScript is most widely used and well documented technology, there is no reason to use anything unorthodox. Bootstrap templates will be used to create a foundational look of the web application web pages. It takes mobile-first approach, creating mobile compatible and automatically scalable web pages that look good no matter what device it is viewed on.

2.4. Other Research you’ve done

There are several research papers online that have experimented with using data mining to predict movie success. They are very similar in approach to this project. This project uses some of the ideas presented in these research papers.

The most recent paper talking about movie success prediction is by Saurabh Kumar from VIT University. In that project, a dataset was downloaded from Kaggle.com that had 651 rows, meaning 651 movies. Additional data about the movie such as its run time was acquired by web scraping IMDb pages with the relevant movie. Movie’s success was measured by IMDb ratings, number of votes, critic and audience ratings, critic and audience scores for the movies. After cleaning the dataset, the data was used to train a naïve Bayes classifier model with supervised learning method like stochastic gradient decent. Using input variables like budget, number of Oscars won by the cast playing in the movie, and using audience score of the movie, made predictions of the IMDb ratings with almost 80% accuracy. [14]

Another very similar research project was conducted by Manisa Celal Bayar University in Tukery. [15] Using datasets from IMDb they collected a dataset of about 6500 movies. They excluded the director and actor data from the datasets. They used 6 data points to in their final dataset.

1. Gross Revenue for movie theatres
2. Release year of the movie
3. Runtime of the movie
4. Total number of rating votes in IMDb
5. The average rating for the movie
6. Cluster rating with values 0-9.

Using this dataset, and after scaling and normalizing it in the data processing part of the research. They tried out 7 different machine learning algorithms with varying results. From 73% up to 92%.

But they did you use a neural network, and they did not use the prediction model to make movie rating prediction for future movies. This project takes a different approach, with using a neural network to make movie rating prediction as well as using a lot more data point for neural network training, such as actor and director awards to use as a metric of the movie's casts' talent and skill which is an important aspect that determines movie ratings.

2.5. Existing Final Year Projects

Secure Document Sharing - Owen Kane

This project creates a secure online system to create, edit and share documents over the internet. It uses client-side AES encryption algorithm to encrypt the files before they are sent over the internet. This way the data will never be sent in plain text format for any man-in-the-middle to see the contents of the data in case where they are sniffing and capturing passing packets online.

This is a good approach to file sharing. This increases the privacy and security of data from being access by unauthorized users. The technologies used are also like what I want use, like Python and JavaScript, in a client-server architecture. Any transition of data between the tiers in the architecture use a secure encrypted transfer protocol, SSL/TLS. SSL is used when data is retrieved from the database to the server, and again when data is sent from server to client and vice versa. This a good approach, with I'll have do the same in my own project.

The project was very well tested. Used multiple types of tests, such as ad-hoc testing, unit testing and integration testing. Testing is vital to any coding project, but more so to project with computer security as possible bugs in the guys can expose vulnerabilities and opportunities for hackers to steal confidential or sensitive data.

Education Tool for Web-Based Vulnerabilities - Cormac Kelly

Interesting project scans your Java files for possible SQ L Injection vulnerabilities. It is designed as an education tool. I like the way it is a web application, making it accessible and easy by the user. It encourages to design code with security in mind and using this tool as quick test for any obvious security flaws pertaining to SQL Injection. I like the idea behind the project, to raise awareness about computer security and encouraging to write secure code.

The project used many technologies and languages. For the code base, Python, Java and JavaScript were used. These are well suited and straightforward languages to use to make a web application and the server back end. These languages also have graphical user interface libraries to make the program easily accessible.

I like this project due to its emphasis on the user interface. It's perhaps the most important aspect of any software because that's all the user is going to see. It's important that is comprehensive and easy to use. As I will also need a user interface for my web application that doesn't look confusing or bland.

2.6. Conclusions

With the current research available, it provides a good guideline of what is a good approach to the problem of trying to predict movie success, and which variables are highly correlated to the movie ratings and which are correlated to movie box office. The research shows that it is indeed possible to have a good estimate of around 70% - 90% accuracy what the movie ratings will be just by taking into account production variables such as budget, runtime, genre and the awards and success of the actors in the movie. Based on the successful use of different machine learning techniques, it seems that a neural network would work well and provide similar level of accuracy as the prediction models that did not use artificial neural networks.

Next Chapter will discuss the overall design of the project based on the researched technologies discussed in this chapter. It will discuss the software development methodologies used during the development of this project, the design and architecture of the neural network and the design of the Django server that hosts the web application.

3. Experiment Design

3.1 Introduction

Following on from the previous chapter, this chapter will discuss how those technologies will be implemented in this project and what the overall design of the project looks like. This chapter will discuss software development methodology which were used during the development of this project, the design choice in building the neural network and the architecture design of the web application. This will include diagrams of system design, class diagrams of code, diagrams of web application architecture and use case diagrams. The final section in the chapter will discuss the type of testing that will be used and how this projected will be evaluated.

3.2. Software Methodology

Agile Methodology

Agile methodology is a type of software development. Agile uses an incremental approach in developing software. It is a fast approach to writing software whilst remaining flexible to any changes in the requirements of the software. With Agile, the work is split up into time periods called "Sprints", usually about 2 weeks, where the development and testing of the software is done. At the start of each sprint, the team, called a "scrum team" have a meeting to plan out what tasks and features of the software should be developed over the course of sprint. The team discusses the plans with a product manager who relays any requests from the product customers, takes into account any needed changes in the software requirements and plans out the work to be carried out. The team then works to achieves the set-up goals and complete the software development tasks laid out in the meeting. This makes Agile mythology into a cyclical development method, where the work done is reviewed and taking into consideration when planning the next steps in development. [16]

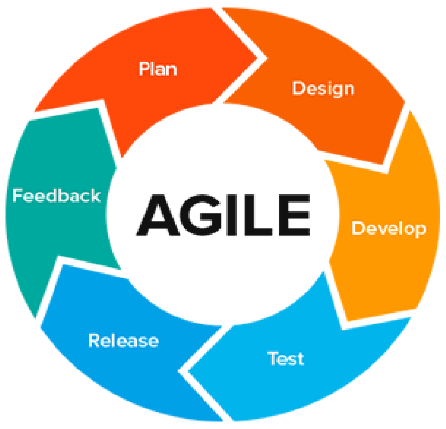


Diagram of Agile development cycle

Feature-driven development

Feature-driven development (FDD) is a type of software development methodology similar to Agile methodology. FDD is also an iterative and incremental approach to software development. FDD's main focus is creating feature for the software, focusing one feature at a time before moving on to the next feature. This is where it differs from Agile methodology, Agile do not focus on individual feature as much as they focus on breaking up the project development plan into small to-d0 tasks and plan to implement a certain number of tasks in a Sprint. FDD instead focus on developing fully working individual features in accordance to the principles outlined in the "Agile Manifesto". With this, FDD combined the best industry practices of software development into one cohesive whole. The main advantage of using FDD is that it's simple 5 step process makes it easy to fast develop and deliver tangible results of the software, present working software features to the costumer. Just like agile, FDD also remains very flexible to any change in software requirements and rapidly adapts does changes.[17]

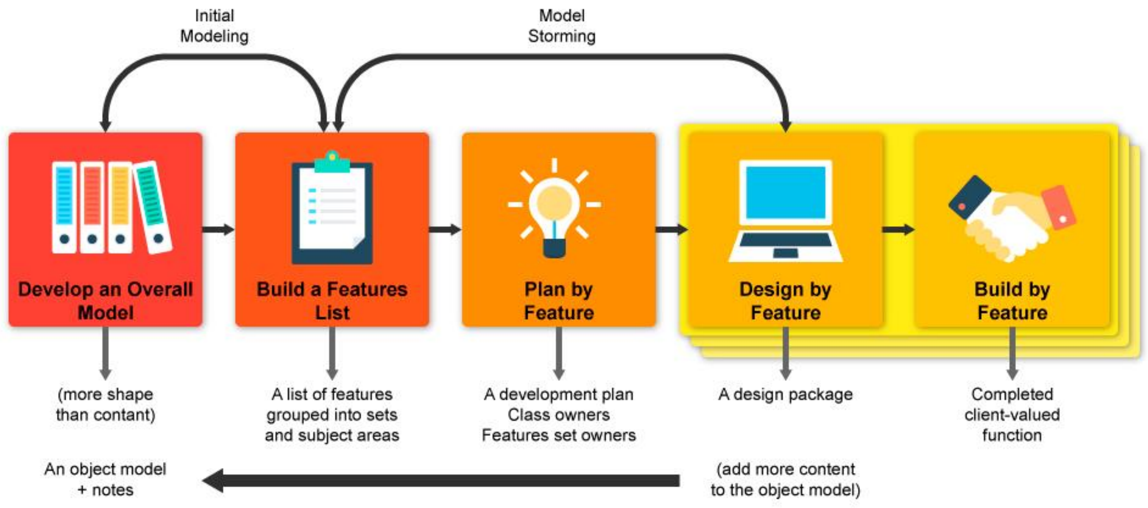
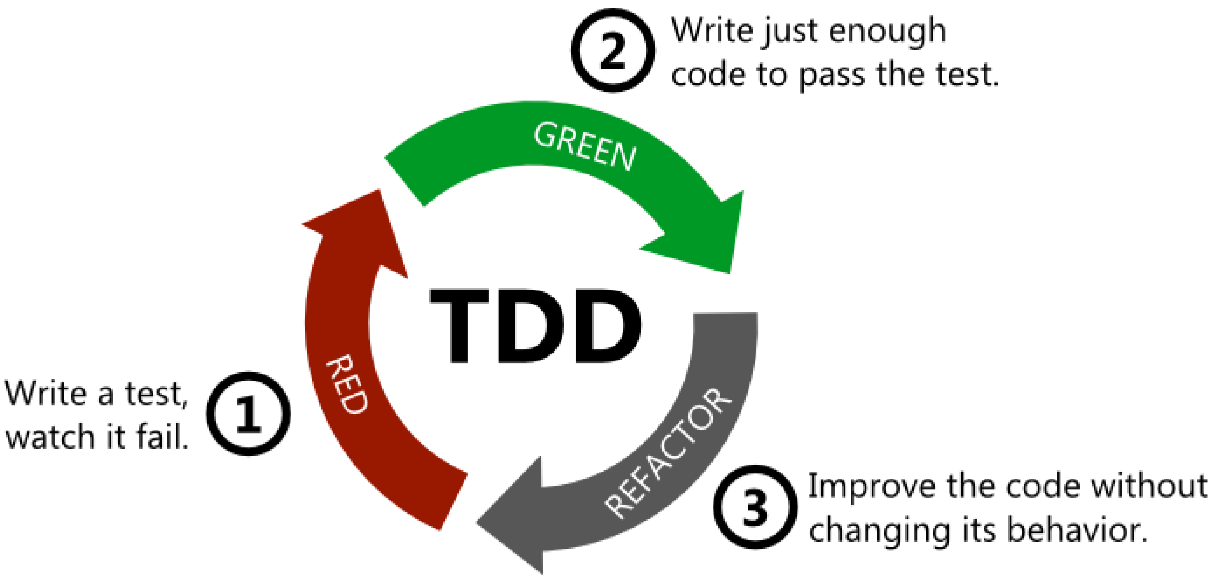

Initial Modeling
Model Storming
Develop an Overall Model
Build a Features List
Plan by Feature
Design by Feature
Build by Feature
(more shape than contant)
A list of features grouped into sets and subject areas
A development plan Class owners Features set owners
A design package
Completed client-valued function
An object model + notes
(add more content to the object model)

Diagram of FFD development

Test Driven Development

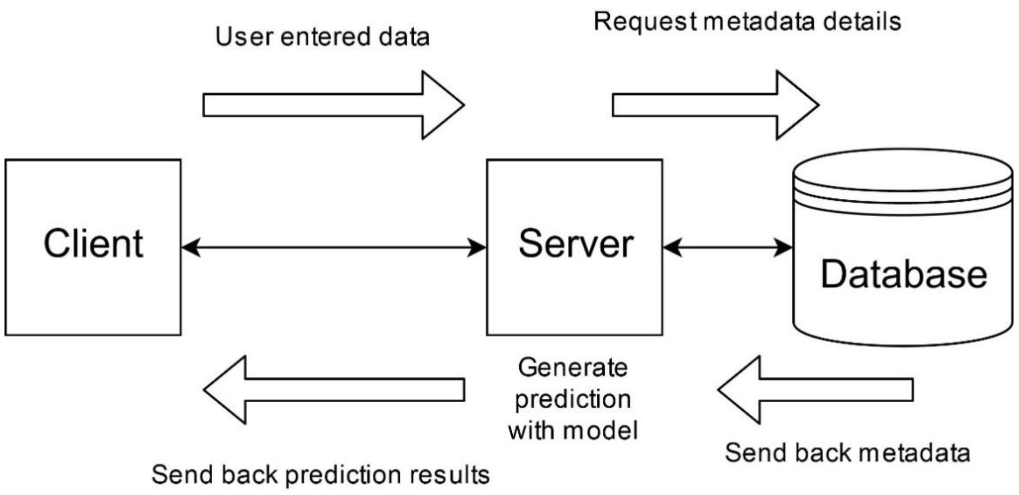
Test Driven Development (TDD) has become a very popular software development cycle in the industry. It can be incorporated with the Feature-Driven Development and agile. TDD approach is to create unit tests for functions that are to be developed first, then the developers will implement minimum amount of code needed to pass those tests. This is a very useful approach in software development. Its main advantage is to ensure that the entire code base remains tested for, if there occurs a bug somewhere in the codebase, the test will pinpoint where the error is. It can easy for developers to focus so much on implementing features first, then trying to create the firsts, but writing tests and some testing for key scenarios can easily forgotten, risking of making hard to find bugs in the codebase. To prevent this, TDD approaches first focusing on the scenarios and test that should be written first, and writing test are prioritised before any further development of the software. This advantage is why TDD approach was used during the development for this project, mainly in the implementation of the Django web app, where the pages and server logic functions need to be tested for. Another advantage of TDD approach, with the principle of writing minimum amount of code in order to pass the tests, is that it prevents writing any unnecessary features and extra code that isn't require. This prevents cluttering of code, allowing for easier reliability, faster implementation and quicker bug fixing. It is clear why this software development has become so popular in the industry, and due to its advantages, the TDD approach was also used during the development of this project.[18]


2
Write just enough code to pass the test.
GREEN
TDD
1
Write a test, watch it fail.
RED
REFACTOR
3
Improve the code without changing its behavior.

TDD software development life cycle

3.3. Overview of System

The main system of this project will be in a form of a web application. It will be a 3-teir architecture, client-server with a database in the backend. Python Django will be the web frame used to host and run the web application. The front-end will be a light client web page, made from HTML\CSS, Bootstrap and using JavaScript to make it interactive and responsive. The front end will look like a form with fields to fill out with details about the movie they want a prediction on. The data will be sent to the server for processing. The database stores record of all IMDb actors and director names with matching IMDb ID. These will used to create IMDb links for each actor and director to web scrape their awards and nominations. The server will do the web scraping for those awards, and using the data from the user submitted form, it will scale and normalise the data and processes into a format ready to be used by the model. The model will analyse the given data and give a prediction of the movie ratings, the server will post process the results from the model into a comprehensible form, make visual representations of the data and send it back to the client to be viewed by the end user.



Overview diagram of the web application and data flow

The Design of the Neural Network

The Neural Network is the core of this project. The details of how the data was gather and processed will be discussed in the following chapter. Here we will discuss the overview of the neural network why it is the way it is.

The goal of this project is it predict movie rating from planned movie production details, like what actors are going to be casted in the movie, director, the budget of the movie, the runtime of the movie and the genre that was picked. These are the main planned movie details that are quantifiable into numbers and can be used to make prediction models on.

Actor/Director Awards and Nominations

This project is unique to what other projects have done in the research is that this project takes into the account actors and directors that were cast for the movie. But how do we quantify actors and correlate their talent with the movie ratings? This project uses the approach of looking at how many Oscars the actor or director has won or been nominated for. as well as how many other awards they had won or been nominated for. Other Awards would be everything that is not an Oscar (The Academy Award), such the BAFTAs, The Golden Globes Awards and Broadcast Film Critics Association Awards etc, with many more for each individual country even [19]. The Oscars are the highest award an actor or director could achieve, and it highly reflects on their talent as an actor or director. This has been separated from so called “Other awards”, which can be more numerous and would be considered same significance as an Oscar. There is also a distinction of being nominated and won, for Oscar or other awards. Being Nominated for an Oscar would be a very good indication of talent, perhaps more so than winning some other award, or maybe not, that will be for the Neural network to decide after seeing thousands of examples and correlating them with movie ratings, but the diction would be an important one to make.

The neural network needs to have a certain amount of input nodes. Initially in the development of this project, a set of nodes for each actor and director, maybe 5 – 10 of them, was considered for implementation, but this approach would have required the end users to enter exactly a set amount of actors every time, removing any sort of flexibility for movie planning, not to mention the size of the neural network. A different approach was taken to solve this problem, instead of considering awards and nominations of each individual actor for each movie, the number of awards and nominations for all actors in the would be summed into a total amount of awards won and nominated for. This reduces the complexity of the network, while still being a good representation for the level of talent. The distinction between actors and directors will remain though, because the talent of the actors and director have different roles during movie production. The actors could be very talented, but the director may lack in skill and vision for the movie and create a movie that would receive low ratings. This is why there is a distinction between the two, and the neural should pick up the significance of the impact the directors and actors have on the movie ratings individually. To achieve this, the training dataset must have data for awards and nomination. In the training dataset has separate column for

- Total number of Oscars won by the movie's actors.
- Total number of Oscars nominated for the movie's actors
- Total number of other awards won by the movie's actors.
- Total number of other awards nominated for the movie actors.
- Total number of Oscars won by the movie's director.
- Total number of Oscars nominated for the movie's director.
- Total number of other awards won by the movie's director.
- Total number of other awards won nominated for the movie's director.

One concern about this approach is that when looking at movies in the past, those actors may have not yet won the awards shown on their IMDb page, is it fair to assign the current number of awards won by the actors to a movie that acted it when they haven't yet won those awards? This is a valid concern, but the assumption behind this approach is in the focus on the talent of the actors, so the assumption is even though the actors may not yet have won the awards for the money, their talent is still the same as when they have multiple awards later on. For example, Leonardo DiCaprio is a phenomenal actor, he was nominated for Oscars four times before winning one. Does that mean DiCaprio's talents back in 1994, when he was first nominated for an Oscar, was not yet good enough to win one until 2016? [20] No, this project assumes DiCaprio's talent was the same (given that over time it may vary and improve) back in 1994 as 2016, so when making a dataset for movies casting DiCaprio before 2016, the movie will include his won Oscar amount, reflecting his talent through his career. This is the assumption for all actors and directors when data gathering for movies total awards won and nominated by the cast of the movies in the dataset.

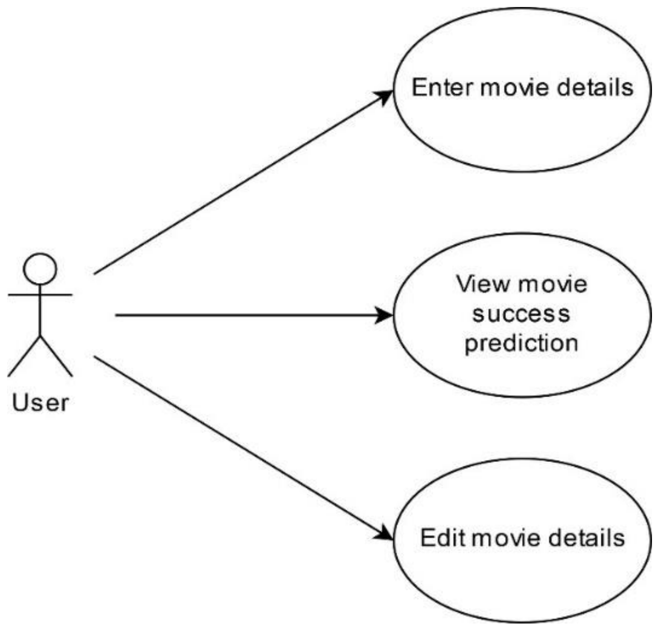
The Neural network will have 28 input nodes for movie details, movie budget, movie runtime, a node for each of the 18 movie genre (due to One Hot Encoding, as will be discussed in the next chapter the technical reason for this), 4 nodes for actors won/nominated Oscars and other awards, and 4 for 4 director won/nominated Oscars and other awards.

The network will have 11 output nodes, for each of 10-star ratings, 0 – 10 inclusive. The neural network's approach to predicting star ratings for movies is a classification approach instead of predicting a numerical value with a single output node. The reason for this will be discussed in the next chapter, but in summer, the multi classification is more accurate in prediction results. The number of middle layers, and the induvial details for the whole neural network will be discussed in depth in the next chapter.

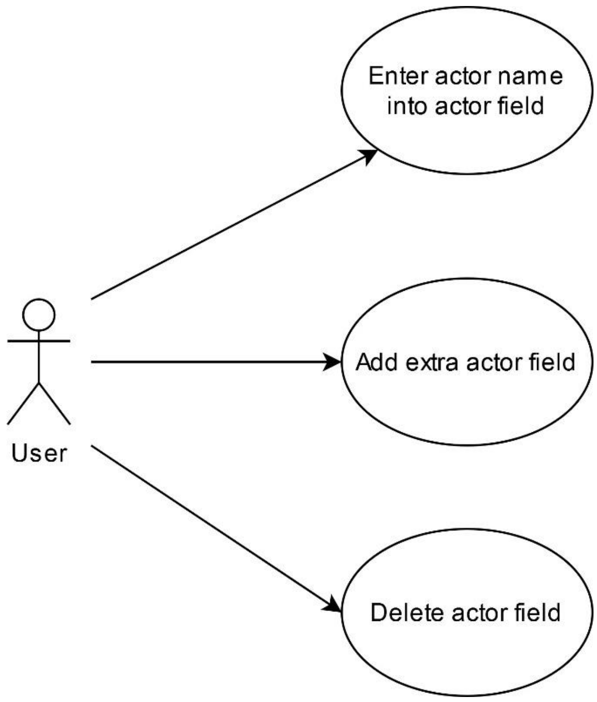
3.4. Front-End

The front-end, the client, are 2 web pages, one having the form to submit movie details the user wants to see movie rating prediction for and the results page showing the movie prediction result. Written in HTML for basic web elements, and CSS for styling and making the website look good. There will be several web pages, first one to be the form where the users fill out the details about the movie and a results page where the prediction and visual data will be shown to the user. None of the processing of data will be done client side, the web pages will just take user input and send that data to the sever for processing.

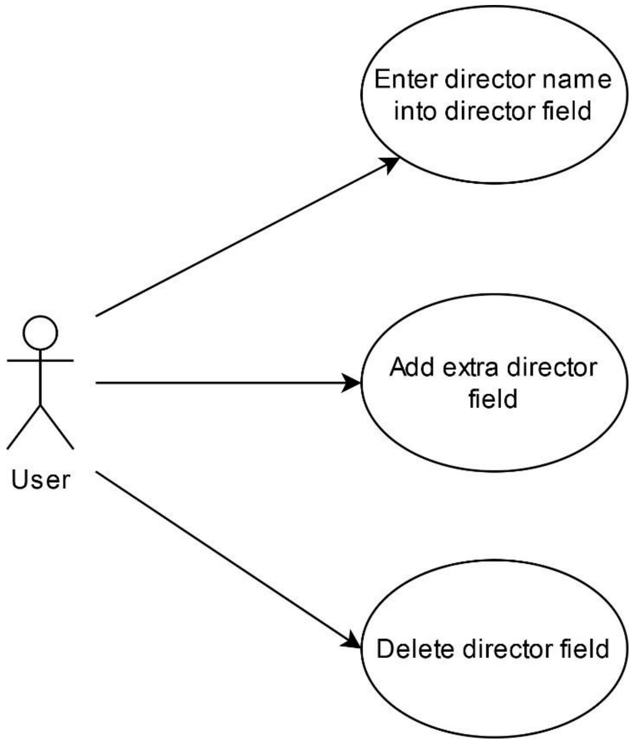
The use case for the web app from the user perspective is simple. The user goes on the web page, they will enter the details for the movie and click the button to predict success. The button will take the user to the results page to view the movie predictions. The use than will have the option to edit the movie details, which will bring them back to the original form page, but the old data is still in the fields, so the user only needs to change some fields as they wish instead of filling out the whole again.



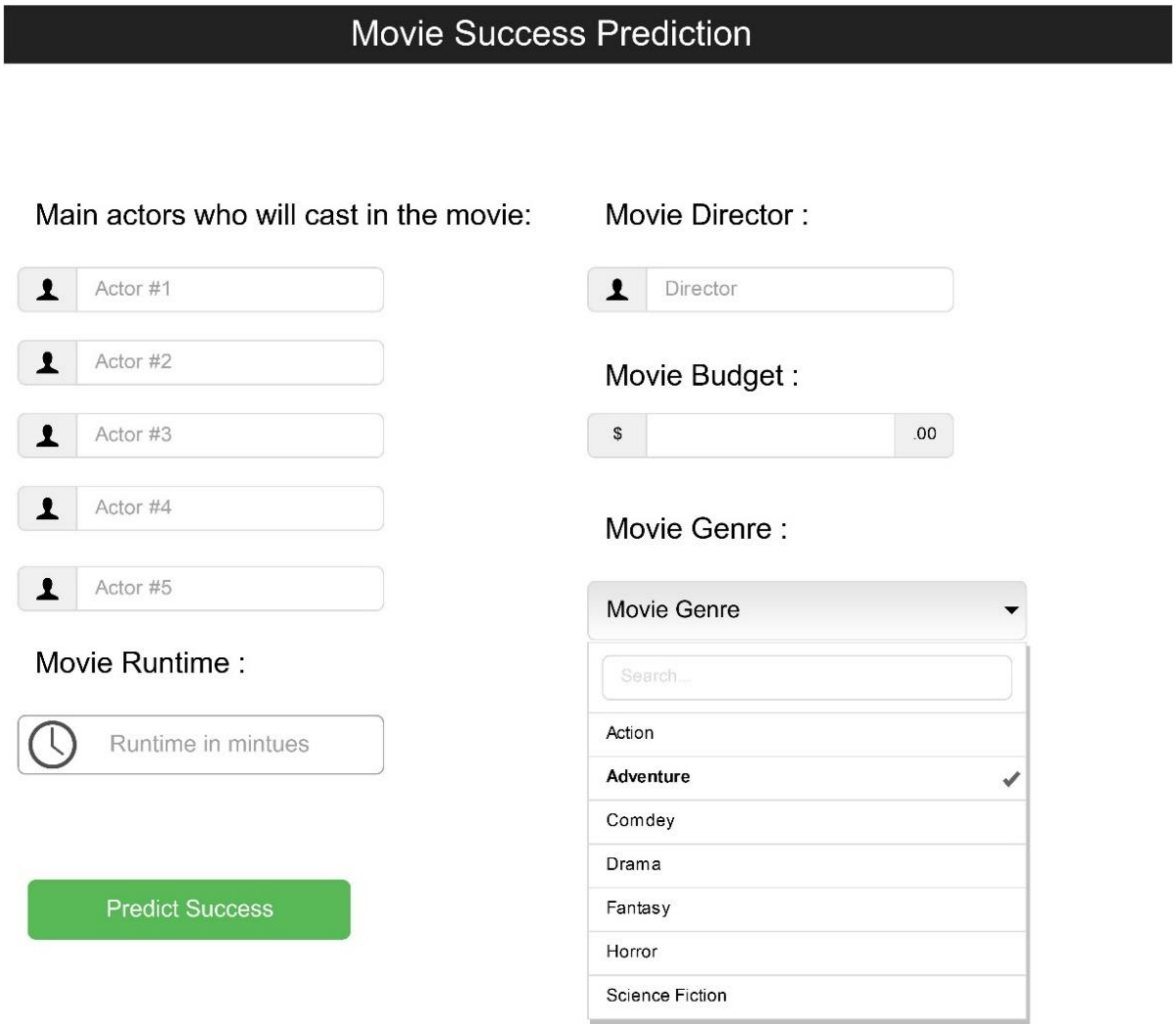
User use case diagram of general use of the web application



Use case of interacting with the dynamic actor fields on the form



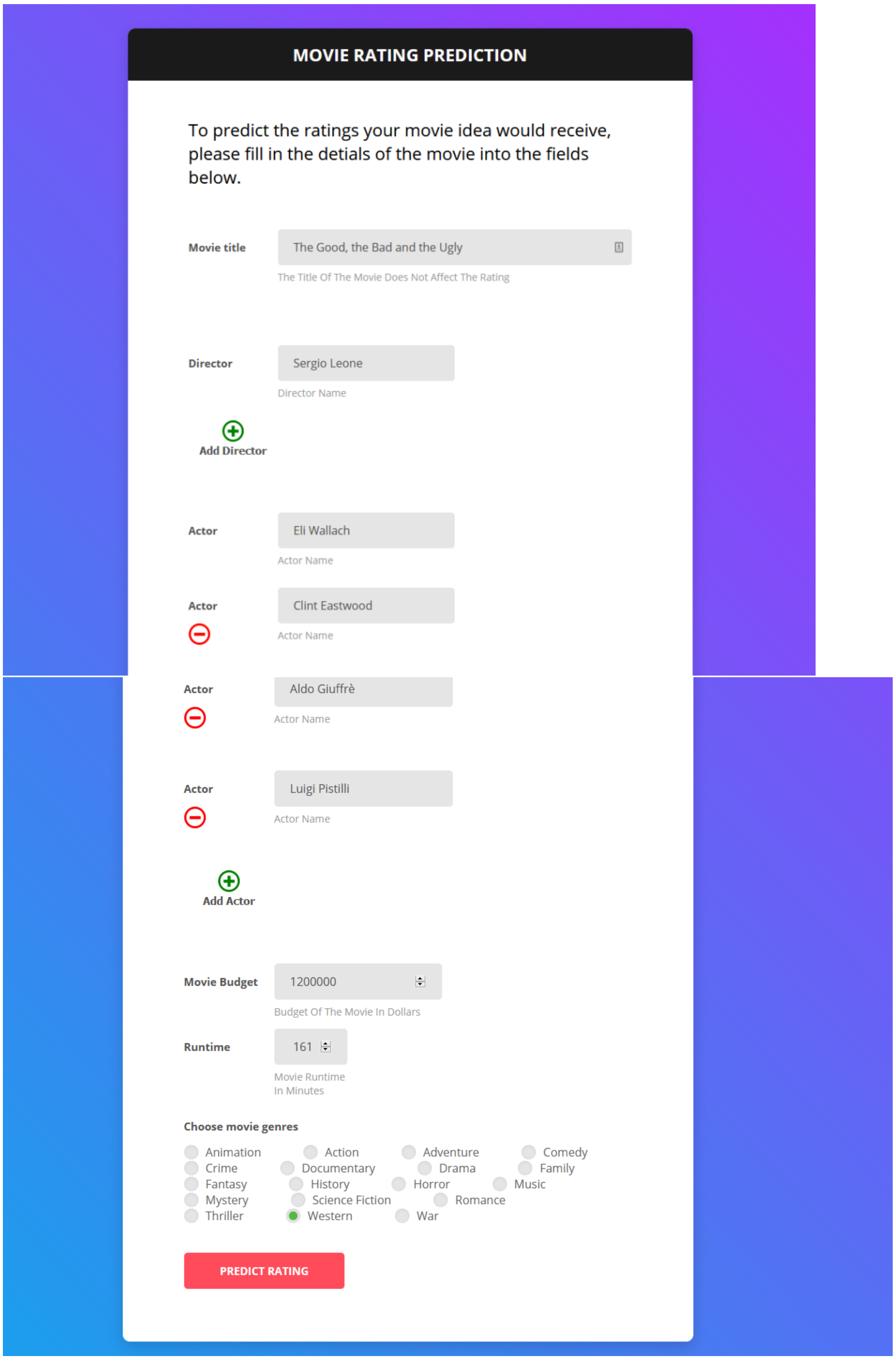
Use case of interacting with the dynamic actor fields on the form



Prototype of from web page

This was the initial website prototype design of the web page. Since then it has evolved to be compacted and scalable on mobile devices. The new look of the web application has more colour to not look bland and more pleasing to the eye.

Final Front page design



MOVIE RATING PREDICTION

To predict the ratings your movie idea would receive, please fill in the detials of the movie into the fields below.

Movie title: The Good, the Bad and the Ugly
The Title Of The Movie Does Not Affect The Rating

Director: Sergio Leone
Director Name

Actor: Eli Wallach
Actor Name

Actor: Clint Eastwood
Actor Name

Actor: Aldo Giuffrè
Actor Name

Actor: Luigi Pistilli
Actor Name

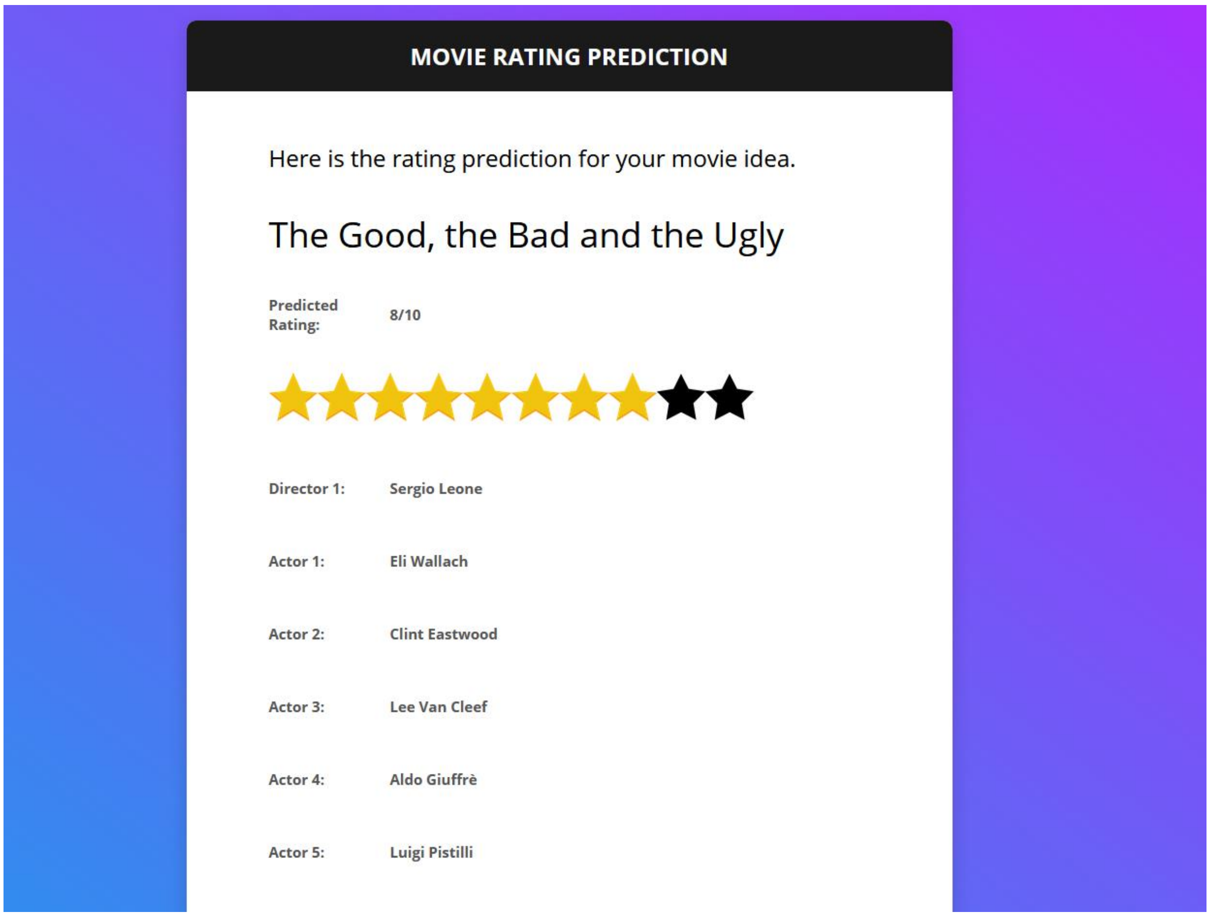
Movie Budget: 1200000
Budget Of The Movie In Dollars

Runtime: 161
Movie Runtime In Minutes

Choose movie genres

Animation, Action, Adventure, Comedy, Crime, Documentary, Drama, Family, Fantasy, History, Horror, Music, Mystery, Science Fiction, Romance, Thriller, [✓ Western], War

Final Result page design


MOVIE RATING PREDICTION

Here is the rating prediction for your movie idea.

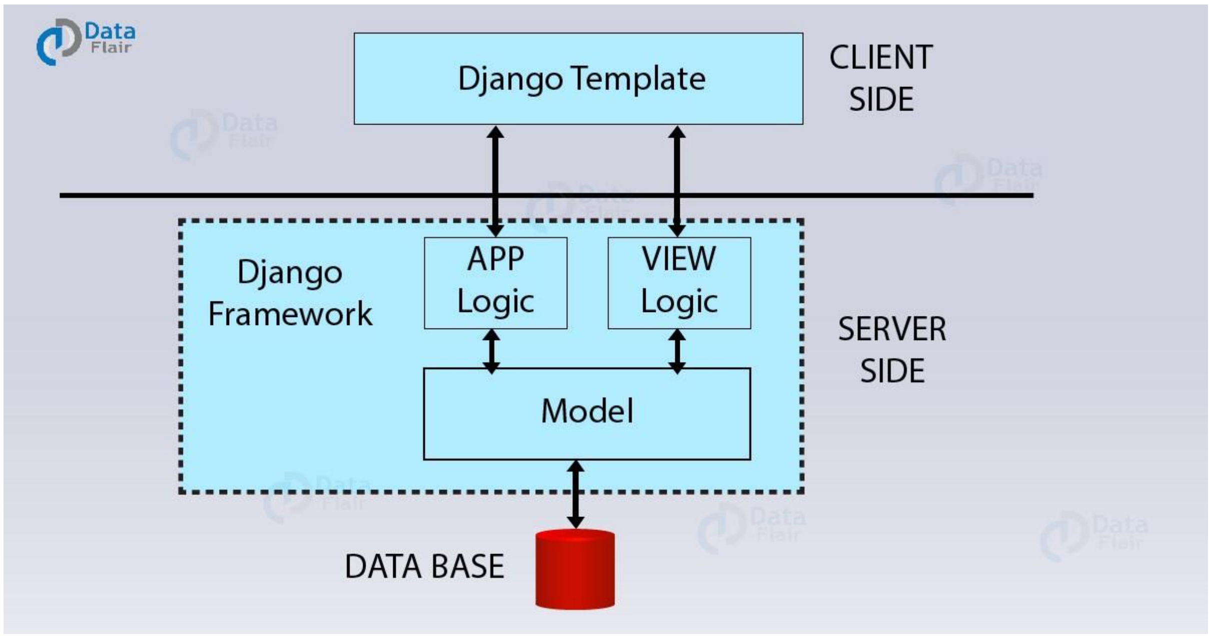
The Good, the Bad and the Ugly

Predicted Rating:	8/10
Director 1:	Sergio Leone
Actor 1:	Eli Wallach
Actor 2:	Clint Eastwood
Actor 3:	Lee Van Cleef
Actor 4:	Aldo Giuffrè
Actor 5:	Luigi Pistilli

There are 2 pages web pages on the web application. This was not for lack of effort but a design decision. The goal was to keep the user interface very straight forward and clear to use. There is a form page where you enter movie details, and a results page where you see the predicted ratings for that movie, with a bottom of each page to go back and forth. In art they say, “less is more”, with less user interface clutter and no unnecessary menus and navigation bars, the users are no overwhelmed and have a good user experience. The evaluation of the user experience will be discussed in the next chapter.

3.5. Middle-Tier

Middle-tier will be run on a python Django server. The server has all the logic behind the web application. The movie success prediction model is stored on the server. The sever will receive input from the client, movie data to make a prediction on. The server will query the back-end database to get metadata on the data, such as number of awards won by the actors, using web scraping, which the user entered. Python scripts on the Django servers will process the data and normalise the data to be used by the prediction model. Run the data through the prediction model to request an output. The prediction model used in this project will make a prediction for the movie ratings. The sever will have to run the processed data through all the prediction models. The results are going to be post processed into a readable comprehensive format. The movie rating result is sent back to the client to be displayed. The prediction result is formatted in visual form with stars and displayed to the user on the results page.



This the model of how the Django server will look in relation to the other tiers

Django templates are the HTML web pages that are generated by the Django server. Each results page is dynamically generated by the server using HTML templates, this makes a web page to reflect the movie rating prediction results along with the summery of the movie details that have been entered by the user in the front page form. The App and View logic handle the data submitted by the user and are processed, extra data that is needed is web scraped and prediction model is used.

All the technical details of the server, the inner logic and functions will be discussed in depth in the next chapter, is just to provide an overview design of the web app using Django web frameworks.

3.6. Back-End

For the back end, it is going to an SQLite database. Originally it was planned to be a MySQL database keeping movie, actor and director details, but the architecture since have evolved. Instead of MySQL, SQLite database was used because it was already fully integrated with the Django Web Framework. This saved a lot of effort in the development process. Another advantage of using SQLite is that it does not require a separate database server, the whole SQLite database is stored in a file. Python uses sqlite3 library to connect and manage the database, and since Django is all python, sqlite3 is used internally in the framework and all data is stored on a sqlite3 file.

Instead storing all movie data, only a list of actor and director names are stored with their corresponding IDs. The design choice for this was made to in order to save a database space, and the movie details will be utilised by the web app. The movie data and all extra data like actor awards are only used to create a movie rating prediction model when training the neural network. The need for the name of actors and directors with their IMDBs IDs is to generate URLs to the actor pages for web scarping of their awards. This actually has the advantage of having the latest most updated data on the movie actor awards, but it comes at the cost waiting a few seconds to make prediction as the web server web scrapes the actor and director award data. The database is non-relational database, as each table does not hold any connection to other tables. There are 2 database tables, or in Django terms "models", which keep all Actor and director details, with which the server than query for. The technical details of the database will be discussed in the next chapter.

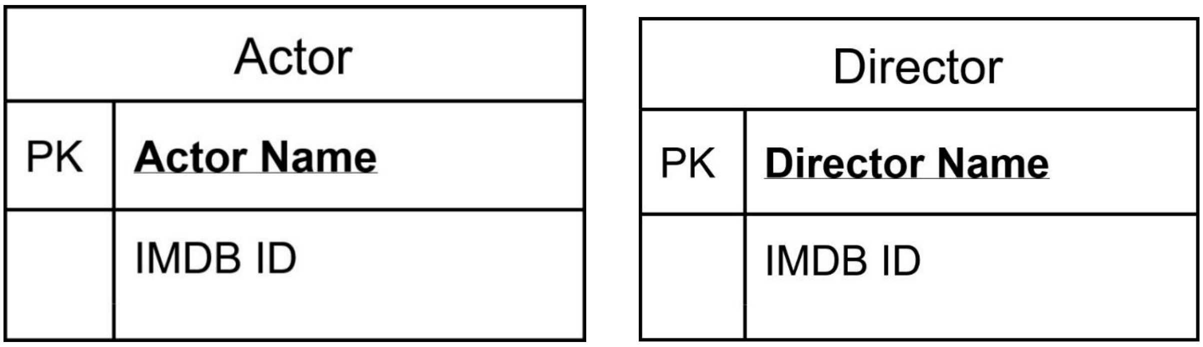


Table diagram for Actor and Director tables

3.7. Conclusions

This chapter discussed the overall design of the web application and the technologies used. Discussed the software development methodologies, with their advantages. Showed the dataflow between architecture tiers, from the front end all the way to the back end. Also discussed the design choices made for the neural network and which movie data was used to train the neural network and to produce movie rating prediction model.

The next chapter will discuss the development details of the project, what has been implemented, explanation of the code logic. It will discuss the technical code design decisions for the neural network, method of data gathering and processing. Also discuss the technical details of the Django web application.

4. Experiment Development

4.1. Introduction

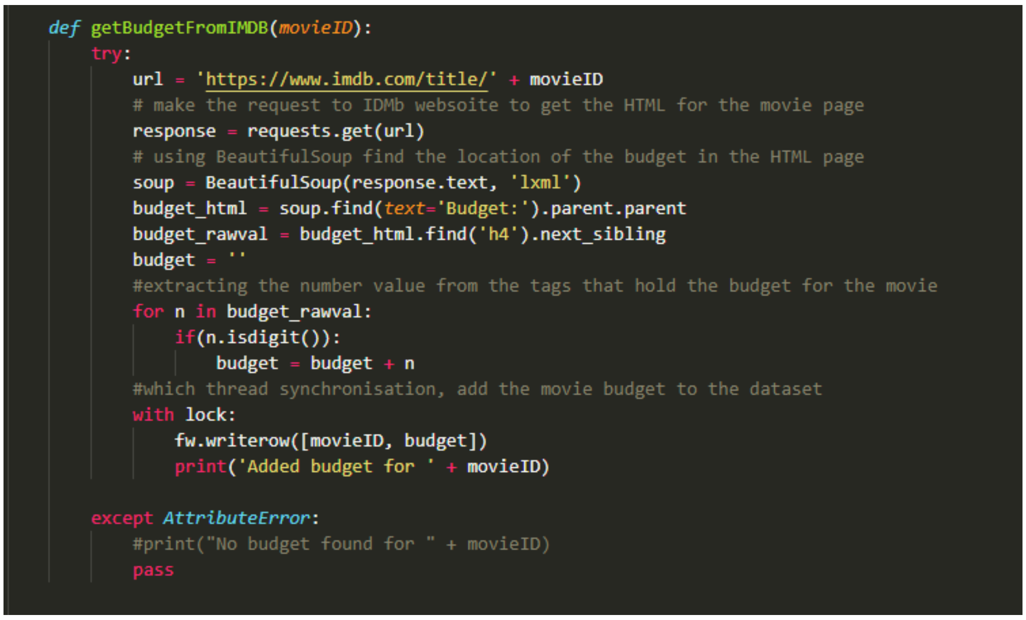
Last chapter we discussed the overall design for the system and the web application. This project is mainly divided into two main parts, the development of the movie rating prediction model and the web application. This chapter will discuss the development of the movie rating prediction model, using data mining techniques and neural network machine learning. This chapter will also discuss the web application developing, delving into details about the front-end of the web app, the logic used by the Django server and the article of the back-end database. We will how all the parts work together to produce a single interactive system to predict movie ratings for new movie ideas.

4.2. Development of the neural network

Gathering of Data

In order to train an artificial neural network a large amount of data is required. The goal was to have a single file CSV called "*movies.csv*" which would hold all the movies with all their metadata present. The python script to train the neural network would read from the *movies.csv* file and put all the rows into a panda data frame to then be normalised, separated into training set and test set, then used to train the neural network.

Kaggle.com is an online website that publishes datasets, free to download and use. I found a dataset with 350'000 movies with details like budget, genre and runtime [21]. That dataset also had tables with main casting in the movie. IMDb website also have a dataset available to download [22]. This dataset contains all the movies and tv series on the IMDb website, the movie ratings and titles with their IMDb IDs. The Kaggle dataset was incomplete, there were movies that were missing the budget for example. To get this additional information, I web scraped IMDb website to find movie budget. Using python libraries "Requests", the web scraping script made HTTP requests to the IMDb websites, using IMDb movie IDs as part of the request URL. Using python library "BeautifulSoup" [23], the received HTML DOC was parsed to the budget listed for that movie on the web page, extracted the number value and added the budget to the in the movie dataset row where the movie budget was missing.

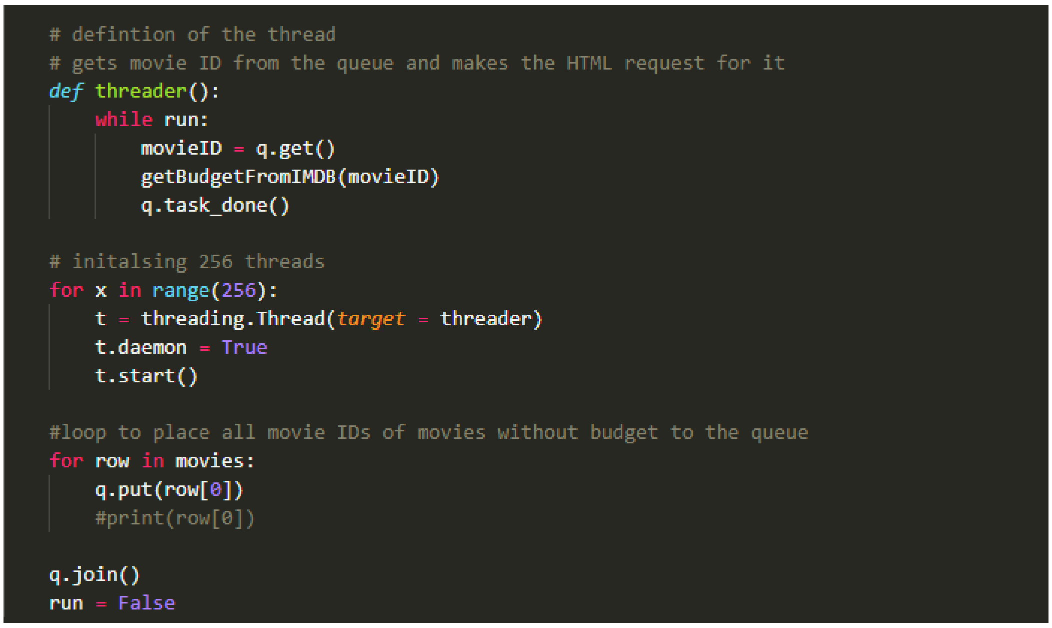


```
def getBudgetFromIMDB(movieID):
    try:
        url = 'https://www.imdb.com/title/' + movieID
        # make the request to IDMb websoite to get the HTML for the movie page
        response = requests.get(url)
        # using BeautifulSoup find the location of the budget in the HTML page
        soup = BeautifulSoup(response.text, 'lxml')
        budget_html = soup.find(text='Budget:').parent.parent
        budget_rawval = budget_html.find('h4').next_sibling
        budget = ''
        #extracting the number value from the tags that hold the budget for the movie
        for n in budget_rawval:
            if(n.isdigit()):
                budget = budget + n
        #which thread synchronisation, add the movie budget to the dataset
        with lock:
            fw.writerow([movieID, budget])
            print('Added budget for ' + movieID)

    except AttributeError:
        #print("No budget found for " + movieID)
        pass
```

Python code used in the requesting the movie web page and parsing for the budget

From the multiple dataset and filled in I wrote a python script to parse through the datasets and merge the data into a single CSV file. This file will then be read by python script to train the neural network

One challenge in the web scraping was the speed of getting and processing the HTML docs. There were about 175’000 movies which didn’t have budget metadata. Making a request one at a time to check for budget on all the movies would have taken a few days to complete. To speed up the process, I used 256 threads to make 256 requests at a time and parse the received HTML doc. This greatly improved web scraping performance, instead of taking few days to complete, it took a few hours instead.



```
# defintion of the thread
# gets movie ID from the queue and makes the HTML request for it
def threader():
    while run:
        movieID = q.get()
        getBudgetFromIMDB(movieID)
        q.task_done()

# initalsing 256 threads
for x in range(256):
    t = threading.Thread(target = threader)
    t.daemon = True
    t.start()

#loop to place all movie IDs of movies without budget to the queue
for row in movies:
    q.put(row[0])
    #print(row[0])

q.join()
run = False
```

Python code using threads to make HTML requests

Data pre-processing

Downloading datasets from IMDb, Kaggle and web scraping for extra metadata created multiple CSV files. It needed to be cleaned, pre-processed and consolidated into one single CSV file that will be ready for training. For this “*data_cleaning.py*” python script was created.

There are 5 main CSVs that have all necessary data needed to train the neural network model.

1. **imdb_movie_ratings.tsv** – this is a dataset downloaded from IMDb website directly. It holds movie ratings for all movies on the website
2. **AllMoviesDetailsCleaned.csv** – The name is a little misleading, as this is a dataset that was downloaded from Kaggle. It holds 30’000 movies with their main details, like their names and IMDb ID, genre, budget and runtime. It has extra data like movie description, year of release etc which is not needed. The consolidated *movies.csv* is created around this main dataset.
3. **movies_with_scrapped_budget.csv** – This dataset holds movies for which there was missing budget data in the *AllMoviesDetailsCleaned.csv* dataset, along with budget that was web scrapped from the IMDB website.
4. **director_awards.csv** – this dataset holds the movies with the number of awards won or nominated for the director for the movies listed in the *AllMoviesDetailsCleaned.csv* dataset.
5. **movies_with_actor_awards.csv** - like *director_awards.csv* dataset, this holds the movies with the total amount of actor awards won and nominated for the movies listed in the *AllMoviesDetailsCleaned.csv* dataset.

The python script *data_cleaning.py* open all the datasets, adds key data like actor awards, director awards and movie budgets into a dictionary to make merging the datasets easier. The key of the dictionary is always the IMDb ID of the movie. The main loop in the script parses through all the rows in *AllMoviesDetailsCleaned.csv* dataset, finds the extra metadata like missing movie budget and awards from the dictionary and puts it into a single array and writes it as a new row in the new *movies.csv* dataset.

To make later data pre-processing easier, the movie genre was encoded into a binary 1s and 0s. This way no extra effort would be required to One Hot Encode the movie genres before training the neural network. Hot Encoding is a data processing technique used to turn classification data, like movie genres, into a binary numerical representation [24]. The neural network only deals with numbers, so the classification data has to be converted into a number. Number genres as 1, 2, 3 … will not work, as the higher number implies a higher value. If the “Action” genre was a represented as 1 and “Romance” genre as 2, the neural network will treat the “Romance” genre as higher value than the “Action” genre. To solve this problem, Hot Encoding is used. With Hot Encoding, all classes are represented by a binary number, or true or false. With the “Action” and “Romance” example, the genres will be represented as an array of 1s and 0s. If the movie is an “Action” movie, the array will look like: [1,0] where the first place in the array represents “Action” and the second place represents “Romance”. If the movie was of both genres, then the array would like [1,1], if the movie was neither genre, then it would look like [0,0]. This way each genre is treated the same by the neural network. This creates a balanced input. In the training dataset, there are 19 total movie genres, represented by 19 binary columns for each movie genre.

Thus *movies.csv* dataset is created and now is ready for further pre-processing and used to train the neural network.

Training the artificial neural network

The training of the neural network is done in the *"training_neural_network.py"* python script. Here the data is taking from the *movies.csv* dataset, put into a data frame using the python library called *Pandas*. The necessary columns of the data are normalised using *MinMax* scaling. The data frame is divided into training set and testing set. The sequential neural network is build using python *Keras* library. The data frame holding the training set is inputted into neural to train. After the training is finished, the network in tested with the test data in the testing data frame to show the accuracy of the model. The neural network prediction is saved as a "*PredictionModel.h5*" file

Data Normalisation

The first necessary step in the training the neural network is to ensure that the input data is normalised. Normalisation is data processing technique to transform the data to be in the same scale without distorting the proportion of the values [25]. This is important when training a neural to ensure a balanced data input, that no input node would overpower the network to render other input data useless. This will result in accurate prediction. But normalising input data to be on the same scale, this will allow a balance adjustment of the neural network's weights and biases, which will make accurate predictions.

To normalise the movie data, all rows from the movies.csv file are loaded into a data frame using the python Pandas library. Only the necessary data is loaded into the data frame, this only excludes the IMDb ID column which is will not be used to the train the neural network. To normalise and scale the rows in the data frame another useful python library is used called "*scikit-learn*". *Scikit-learn* (or "*sk-learn*") is a python library that provides a lot of use tools for machine learning and data analysis, such as creating models and processing data tools. In this project *sk-learn* is used for the quick normalising function.

During the development of this project, then normalisation method used changed. At the start, from the *sk-learn* library, *StandardScaler* was used, then later in the development was changed to use *MinMax* scaling was used to normalise the data. There is a significant difference between the techniques.

StandardScaler is a method of normalising and scaling the data that the mean average of the data is 0 and that the standard deviation of the data is 1. This is done by using this equation ***z = (x - u) / s*** where ***x*** is the sample number from the data frame, ***u*** is the mean average of the whole data frame column from where ***x*** was taken and s is the standard deviation of the data in that column. This way each value is changed to be smaller and closers to 0 in relation to all other data in that column. This value scaling transformation is applied to each column individually to created values on the same scale while keeping their relational proportions. [26]

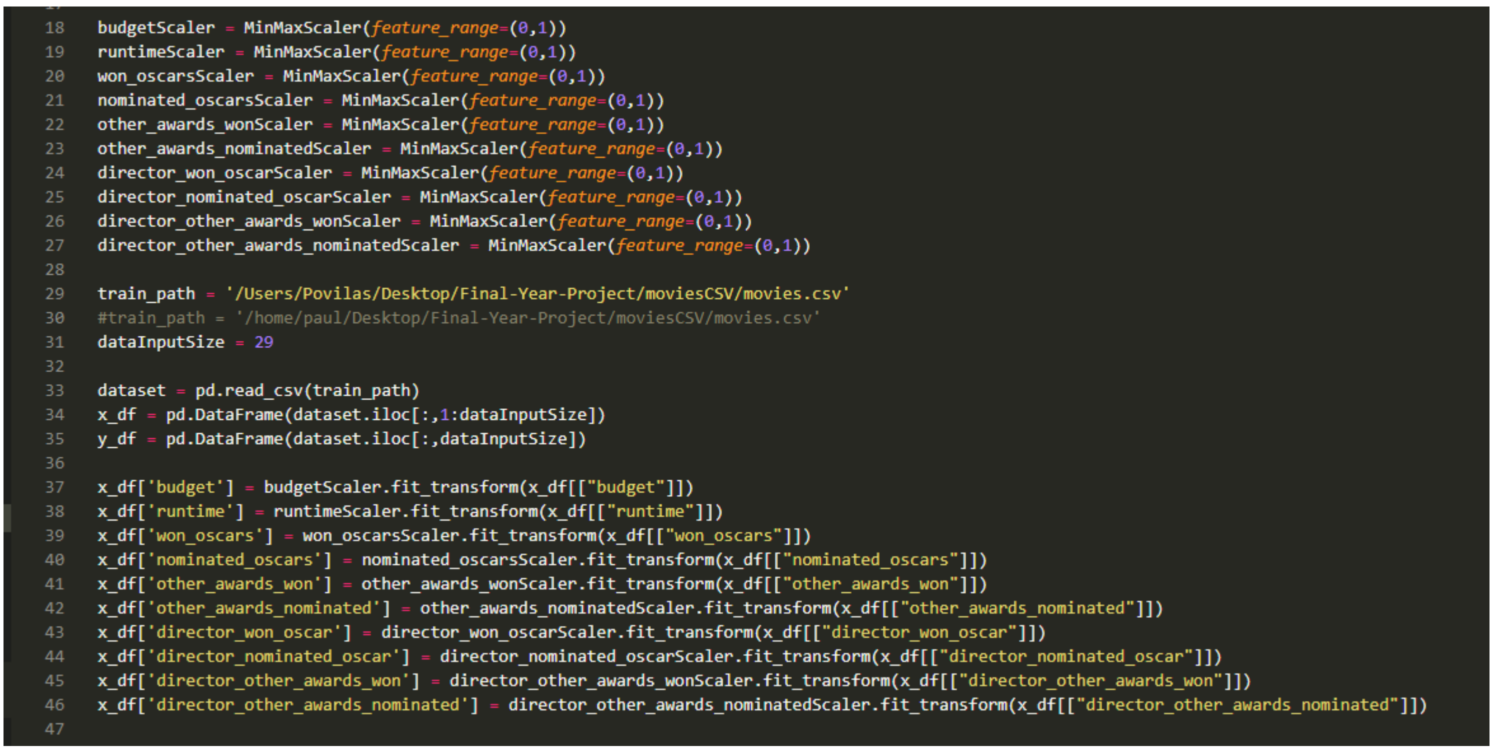
MinMax is a method of normalising and scaling the data to put all values proportionally between 0 and 1. This a simple mapping where the lowest value is 0, and the highest value in the dataset and set to be 1, the rest of the value in the column is mapped proportionally between 0 and 1. This transformation is given by:

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X_scaled = X_std * (max - min) + min

[27]

While both methods manage to create data on a similar scaler, *StandardScaler* transforms data to have both positive and negative values in order to ensure the mean average is 0. The scaling transformation must be applied to each column individually, which means the data scales will not be exactly uniforms. If there are significant outliers in the dataset, it will skew the data scale even more, creating slight data scale mismatches between the data columns. This could mean that the scaled version of movie column could have higher values in general than actor Oscar winnings column. To minimise this possible imbalance, *MinMax* was which put all data between values of 0 and 1 ensure all data was now on the same scale guaranteed. No input node would outweigh any other nodes during neural network training.



```python
budgetScaler = MinMaxScaler(feature_range=(0,1))
runtimeScaler = MinMaxScaler(feature_range=(0,1))
won_oscarsScaler = MinMaxScaler(feature_range=(0,1))
nominated_oscarsScaler = MinMaxScaler(feature_range=(0,1))
other_awards_wonScaler = MinMaxScaler(feature_range=(0,1))
other_awards_nominatedScaler = MinMaxScaler(feature_range=(0,1))
director_won_oscarScaler = MinMaxScaler(feature_range=(0,1))
director_nominated_oscarScaler = MinMaxScaler(feature_range=(0,1))
director_other_awards_wonScaler = MinMaxScaler(feature_range=(0,1))
director_other_awards_nominatedScaler = MinMaxScaler(feature_range=(0,1))

train_path = '/Users/Povilas/Desktop/Final-Year-Project/moviesCSV/movies.csv'
#train_path = '/home/paul/Desktop/Final-Year-Project/moviesCSV/movies.csv'
dataInputSize = 29

dataset = pd.read_csv(train_path)
x_df = pd.DataFrame(dataset.iloc[:,1:dataInputSize])
y_df = pd.DataFrame(dataset.iloc[:,dataInputSize])

x_df['budget'] = budgetScaler.fit_transform(x_df[["budget"]])
x_df['runtime'] = runtimeScaler.fit_transform(x_df[["runtime"]])
x_df['won_oscars'] = won_oscarsScaler.fit_transform(x_df[["won_oscars"]])
x_df['nominated_oscars'] = nominated_oscarsScaler.fit_transform(x_df[["nominated_oscars"]])
x_df['other_awards_won'] = other_awards_wonScaler.fit_transform(x_df[["other_awards_won"]])
x_df['other_awards_nominated'] = other_awards_nominatedScaler.fit_transform(x_df[["other_awards_nominated"]])
x_df['director_won_oscar'] = director_won_oscarScaler.fit_transform(x_df[["director_won_oscar"]])
x_df['director_nominated_oscar'] = director_nominated_oscarScaler.fit_transform(x_df[["director_nominated_oscar"]])
x_df['director_other_awards_won'] = director_other_awards_wonScaler.fit_transform(x_df[["director_other_awards_won"]])
x_df['director_other_awards_nominated'] = director_other_awards_nominatedScaler.fit_transform(x_df[["director_other_awards_nominated"]])
```

Code snippet for data normalisation

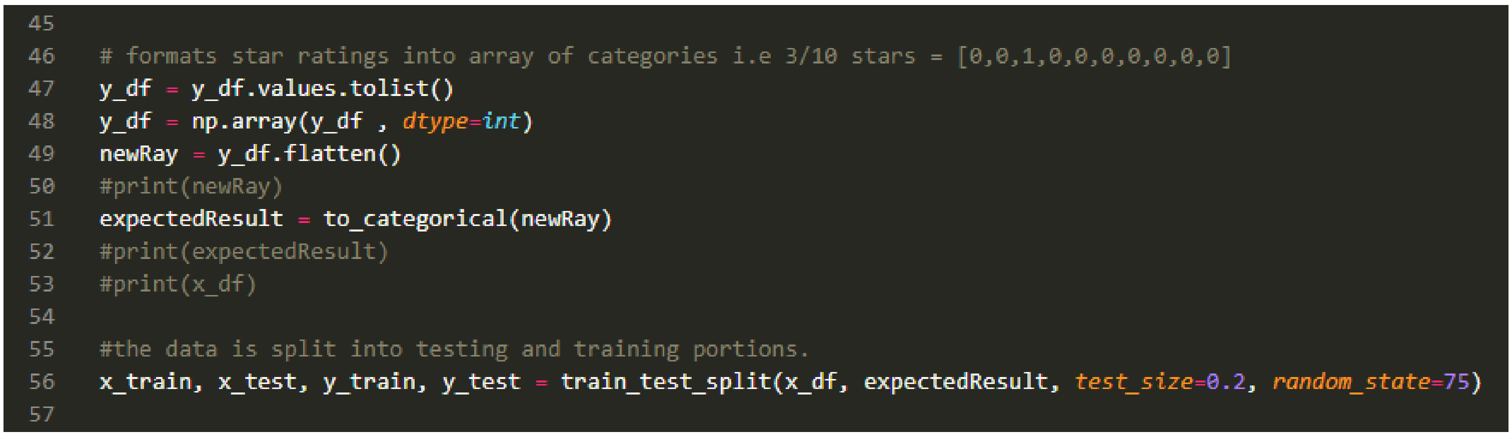
The code snippet shows the code used to normalise and scale the data. The variables between line 18 – 27 hold the scaling objects, in this case they are MinMaxScalers with feature range between 0 and 1. The data is loaded from the *movies.csv* file into a "*dataset*" *Panda* data frame. Then the dataset is split into the features and labels dataset. ***x_df*** is the features, with movie budget, runtime, genre and awards. ***y_df*** is the labels, the ratings for each movie.

In machine learning, the data used as input to train the model are called features, and the data used to check the output the neural network is called labels. The model's weights and biases will be adjusted according to how close the model's output matched with the label.

The genre part of the dataset does not need to normalise. They have been already converted into a binary representation. Each genre will have an input node, and the values are already either 0 or 1, hence no further pre-processing is required on the genre columns. All other columns, such as budget, runtime and awards need to be normalised. Each column is normalised separately and transformed. In one function, the scaler object is fitted with what the minimum and maximum values are and the value is transformed and written to back to the ***x_df*** data frame. The input features have been finally normalised and now ready to be used to train the neural network.

One last step to the data pre-processing is turning the ***y_df*** labels dataframe into binary array. This project will use a multiclassification approach in neural network architecture. The ratings between 0 – 10 will be representing by 11 output nodes. Current ***y_df*** only has the single numerical movie ratings, i.e 4, 6, 7, 5. To be able to match the networks output nodes, each value will need to be converted into another *HotEncoded* binary array, such that if the movie has a rating of 5, the representative *HotEncoded* array will look like ***[0,0,0,0,0,1,0,0,0,0,0]*****.** To do this, Keras library has a quick function *to_categorical()* to do this. First ***y_df*** is converted into a NumPy array, because that is the expected parameter of the *to_categorical()* function. The new NumPy version of y_df is stored in ***newRay*** variable which is passed into the *to_categorical()* function, the encoded results is stored in a variables called ***expectedResult***, which will be used as labels for the neural network training.

Lastly, the features and labels are split into training and testing datasets, called ***x_train, x_test, y_train, y_test x_train, x_test, y_train, y_test*****,** where ***x*** variables gold the features and ***y*** variables hold the labels. These will be used to train and test the neural network.

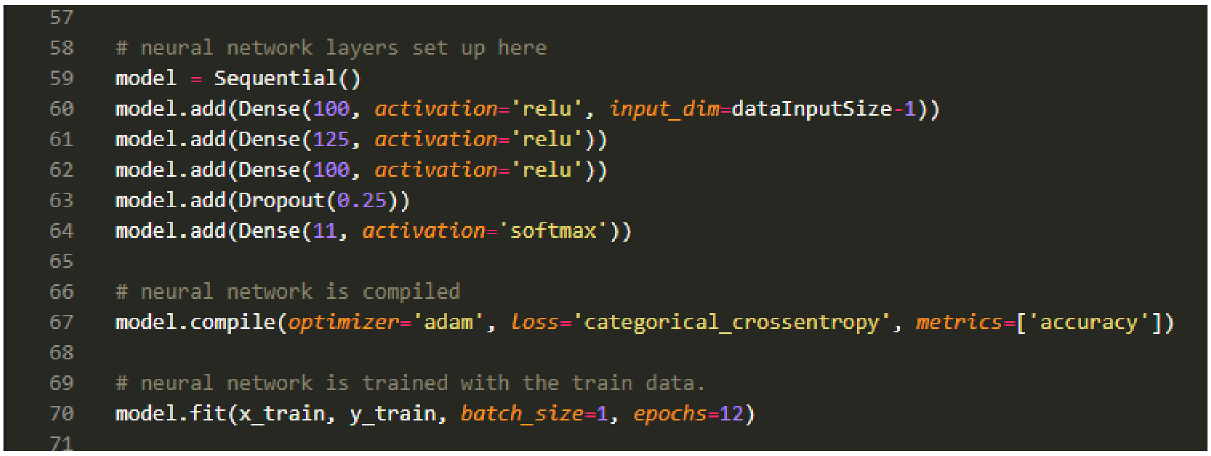


```python

# formats star ratings into array of categories i.e 3/10 stars = [0,0,1,0,0,0,0,0,0,0,0]
y_df = y_df.values.tolist()
y_df = np.array(y_df , dtype=int)
newRay = y_df.flatten()
#print(newRay)
expectedResult = to_categorical(newRay)
#print(expectedResult)
#print(x_df)

#the data is split into testing and training portions.
x_train, x_test, y_train, y_test = train_test_split(x_df, expectedResult, test_size=0.2, random_state=75)

```

Code snipper for encoding movie ratings into encoded binary array

The Neural Network

The neural network used in this project is a 6 layered dense neural network. It takes 28 input data about the movie, such as the budget, runtime, genre and awards and makes a prediction which star rating the movie will get. Dense neural network use “*dense*” layers, which means that every node in the layer is connected to every other node in the previous and next layer of the neural network. There are 11 output nodes, which represent the star ratings as classes, and only 1 of them is activated. The neural network uses ***ReLU*** activation functions for the middle layers and Softmax for the output layer. The neural network uses ***categorical cross entropy*** loss function to generate loss error values and uses the “***adam***” optimiser to optimise the parameters of the neural network. The details of each part of the neural network will be discussed in the following sections.



```python

# neural network layers set up here
model = Sequential()
model.add(Dense(100, activation='relu', input_dim=dataInputSize-1))
model.add(Dense(125, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(11, activation='softmax'))

# neural network is compiled
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# neural network is trained with the train data.
model.fit(x_train, y_train, batch_size=1, epochs=12)

```

Code snippet showing the creation and training of the neural network

Neural Network Layers

To create the neural network the python library Keras was used [28]. Keras provides an easy to use functions as an interface, or like an API, with a TensorFlow backend. Under the hood the python library TensorFlow is doing all the machine learning functionally while Keras provides easy to use functions to quickly set up and manage the neural network, speeding up the development process. This was the main reason that Keras/TensorFlow were chosen for this project.

The neural network is crated as a *Sequential()* model. This means that the layers are added one after the other in order. The network used in this project consists of 6 total layers.

1. **The Input layer** – has 28 input nodes. In the code it's specified as "***dataInputSize-1***" because the total columns in the *movies.csv* file are 29, which is the ***dataInputSize***, but after removing the IMDb ID column from the data frame, there is only 28 columns left**.** Each node for each column from the data, for budget, runtime, for each genre and the nodes for the awards won or nominated etc. All input values have been normalised and scaled beforehand, so each node receives an input value that's somewhere between 0 and 1.
2. **Middle layer** with 100 nodes.
3. **Middle layer** with 125 nodes.
4. **Middle layer** with 100 nodes.
5. **A Dropout layer** with a drop rate of 25%. This layer is used as a measure to prevent overtraining of the neural network. It achieves by switching off 25% of the nodes from the previous middle layer, for each run. This helps to ensure that no nodes in the all previous layers die permanently and all nodes contribute to the final prediction. Basically, keep the neural network on its toes.[29]
6. **The Output Layer –** has 11 nodes. The 11 nodes correspond to each star rating for the movie, 0 – 10 inclusive. The value of each node is compared to the corresponding value of binary array in the labels (***y_train***). How close the values where to the expected value are used to calculate the loss, to then correct the weights biases during the backpropagation of the neural network.

There is no specific correct number of layers or nodes that must be used. Creating a neural network is more of an artform, in that there is a lot of trial, error and experimenting to see which types of models, number of layers or nodes, types of activation function and different loses provide the most accurate results. During the development of this project, a variety of combinations of nodes and layers have been tried to see which gives the most accurate results and how long it takes to train the network. If there are too many layers used, the accuracy drops quietly significantly. If too many nodes are used the accuracy drops as well and the training time can take twice as long or longer. The chosen combination used in this project is a balance between having a large number of nodes with a slight variation between the layers to allow a more accurate capture of trends and patterns in the multi-dimensional input used, and balance between the speed of training, quickest training time without losing accuracy. The drop out rate for the drop out layer was also chosen as a balance between help prevent overtraining and not affecting the accuracy of the results. Too many turned off nodes will prevent the model from making accurate predictions.

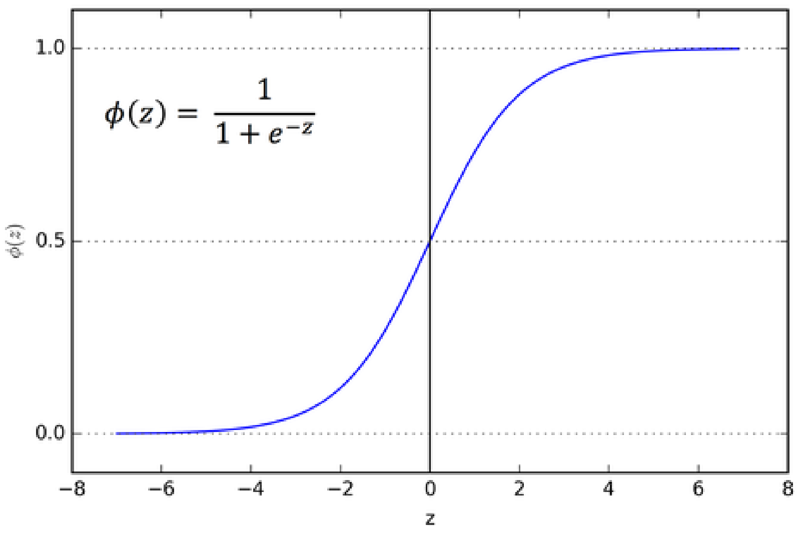
But the activation functions and loss functions were deliberately picked, as will be discussed further on.

Activation functions

The action function is a mathematical function that takes the sum of all received inputs from other neurons (the nodes) and runs the values through the activation function. Each activation function has different activation thresholds, meaning that the neuron will only activate when a value is above a certain threshold value. The activation functions are what make neural networks special when to comes to creating prediction models and allow the training of the network. There are many activation functions to choose from when creating a neural network, the main ones sigmoid, TanH, Reu and Softmax to name a few.

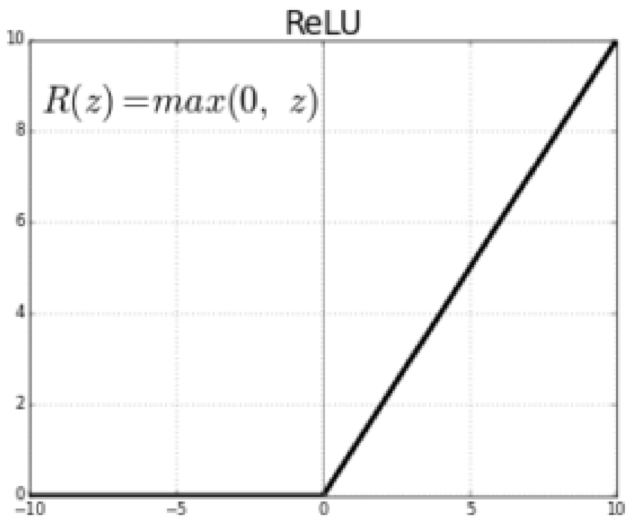
In the development of this project, a few main functions have been researched and experimented with. The ones used in the project are ReLU and Softmax. Sigmoid activation function is a common one, but it has a few problems, for which ReLU is a better choice.

The sigmoid activation function is based on a mathematical sigmoid function that gives a “S” shape to the line. In machine learning, it can used as function to determine if the neuron should activate, by taking all node inputs and passing them into the function to determine if the output of the function meets the threshold. It squashes the inputs to give an output that is between 0 and 1. Sigmoid is commonly used to determine probabilities, and used as an activation function in the last layer that has a single node, for binary classification models which only need to predict 0 or 1, or rather a probability outcome how likely is it to be a True or False [30]. Since this project uses multi classification, the output layer has 11 nodes, so instead of sigmoid function, Softmax activation is used for reason discussed further on. Sigmoid has some inherent problems. Vanishing gradient problem which occurs with neurons using sigmoid during backpropagation. the sigmoid functions squashes large inputs into small output of between 0 and 1, this means the degree of error is smaller hence correction is smaller, the model learns slower. When there are many layers in the network, the small outputs keep getting smaller until the neuron no longer even activates, and any further training in the model is halted, the gradient vanished. The solution to this problem is to use a different kind of activation function for the middle layers of the neural network, the ReLU function. [31]



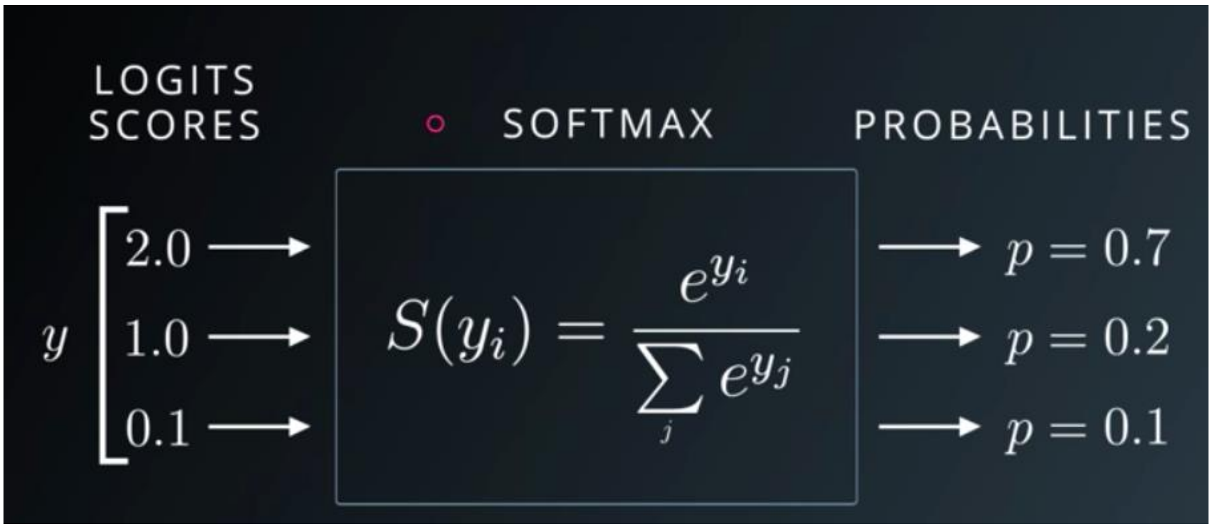
Sigmoid activation function graph

Rectified linear unit (ReLU) activation function is a very simple activation function where any positive input activates the neuron. Of course, this does not mean that all neurons will be active, because the weights connecting the neurons can have negative values. Unlike the sigmoid activation function, the inputs are not squashed into a small value, but preserved through the network. This solves the vanishing gradient problem, but it doesn't make my probability prediction. This function is ideal to use in middle layer activation function of neuron and is currently the most popularly used activation in neural networks [32].



ReLU activation function

Softmax activation function is similar to the sigmoid function in graph shape, but the key difference is that is applied to multiple neurons instead of just the single one. Softmax function takes in large inputs from multiple nodes and squashes the values between 0 and 1 to give a probability distribution for the whole layer's output. This function is used for multi-classification neural networks, like the one used in this project. Later on, categorical cross entropy function encodes the output of the nodes into a binary array where the highest probability node is 1, and the rest are 0, as will be discussed further on. [33]



Softmax action function

Loss Functions

In neural networks, loss functions are methods of optimizing the parameters of the neural networks. Ways of adjusting the weights and basis in the network in order to match the predicted output to the expected output as closely as possible. This is arguable the most important aspect the design of the neural network, as the architecture of the neural network depends on the activation function that was chosen and vice versa. Every on in the development, the model used multiple output nodes by was using a "binary cross entropy" the model has 0% accuracy. After more research, the mistake was solved and the type of loss function that was picked matched the architecture which finally gave expected accuracy results.

There are mains approaches to predicting movie ratings. Since the movie rating is a numeral value, the neural network could predict the actual rating value using "Mean Squared Error" loss function. Also, since movie ratings only range between 0-10, each rating can be treated as a class, meaning there will be 11 classes and the model will predict to which class the movie belongs to. Multi classification neural network uses categorical cross entropy loss function.

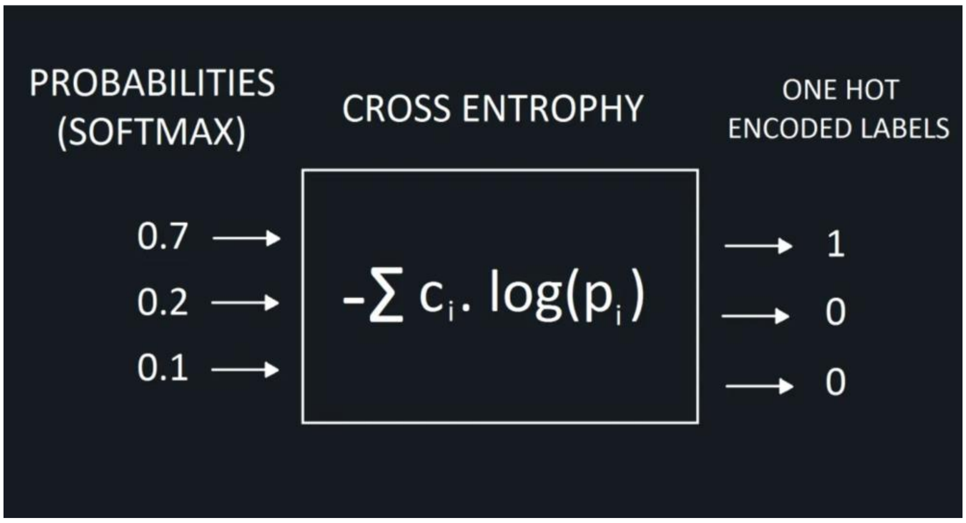
Mean Squared Error (MSE) is a loss function that finds the error value, by which to train the networking during backpropagation, by getting the average mean of the squared error. What this means is that the output value is minused from the expected value, this gives an error value. The error value is squared and summed up with other squared error values and the sum is divided by the total number of error values, giving the mean squared error value. This value is used to optimise the neural network parameters. This loss function optimises the network to predict numerical values. This way the neural network could predict the numerical value for the movie ratings. [34]

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

The equation used to calculate the Mean Squared Error

But since the movie ratings can be divided up into 11 classes, the neural network could give more accurate ratings prediction if it uses multi classification loss functions like the categorial cross entropy function.

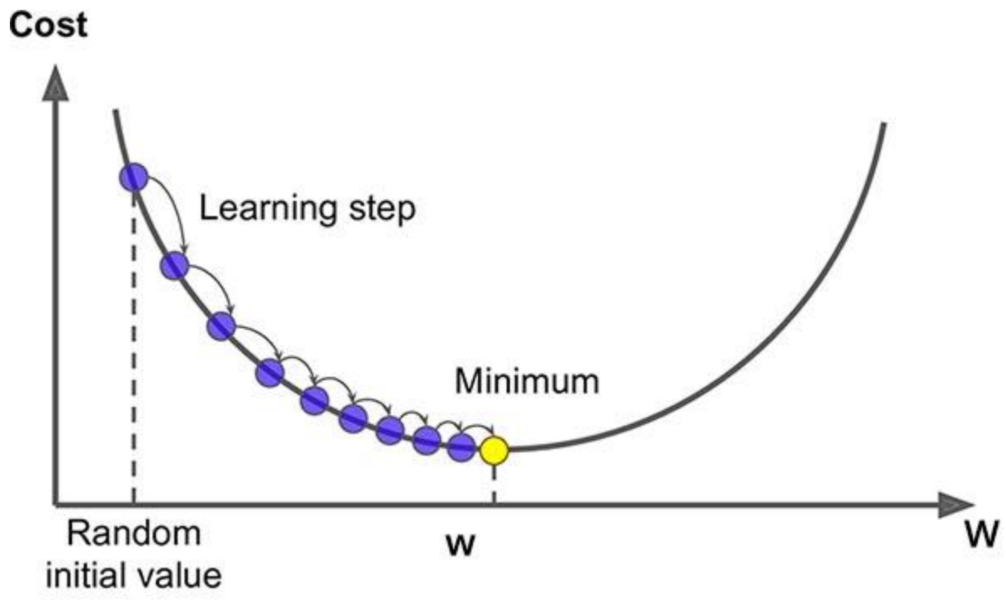
Categorical cross entropy is a loss function that is used for multiple class labels. This loss function is based on the logarithmic loss function (log loss) in that both functions take in a probability distribution as input to calculate the error value between the predicted output versus the actual output. This is the type of loss function that is used in this project. Categorical cross entropy looks at the predicted class and the expected class and generates an error based on the predicted probability of that class. In the neural network to predict movie ratings, the output later uses the Softmax activation function, to attach a probability value to each output node so that all node probability values add up to 1. Categorical cross entropy treats the node with the highest probability as the final predicted class and calculates the loss error value, to be used to optimise the neural network. This is why the movie rating labels had to be One Hot Encoded into binary array [35].



How cross entropy loss functions treats the probability distribution from Softmax

Optimizers

The optimizers in neural networks are the functions that adjust, optimise, neural network parameters, the weight and biases. When talking about weight and error values, there is an important concept is necessary to understand, that of **gradient decent**. Given a set of data points, the neural network must adjust the weights in the network to minimise the error value. There are optimal values for all these weights that give the minimum error values (the cost). The weight values and error values could be graphed on a graph to show a curved gradient, which also has a lowest point where a certain weight value will provide the lowest error value.[36]



In the graph above, the X-axis is the cost, the error value given by loss function after the first prediction attempt. The y-axis is the weight value of a single given weight. There are many weights with their own gradients. Initially all weights are set some random value, then optimized. The learning step is learning rate of the function, how much the weight should be adjusted given the error (cost) value. Thus, the cost value is minimised as weights are adjusted to optimal value, this is called **gradient decent.** To do this manually it would take too much effort hence, automatic gradient algorithms, optimisers, are used.

Stochastic gradient descent optimizer is an algorithm that uses step-by-step approach to minimising costs, as opposed to calculating the gradient decent after seeing the whole dataset. The iterative approach is much more efficient as it calculates the needed adjustment after seeing a few examples at a time. Providing faster learning rates. But the stochastic gradient descent optimisers has a few problems, such what in the case where there might only be local valleys in the gradient and the weights get stuck on a value? This algorithm can be improved upon by making it more adaptive, such as adding *momentum.* Meaning if a weight was being adjusted in a certain direction, increase the value of the adjustment more so it would get to the minimum point faster. But this can overshoot the optimal valley sometimes. [37]

Adam (Adaptive Moment Estimation) is an updated adaptive Stochastic gradient descent optimizer that works generally well for a wide variety of machine learning problems. It is adaptive to the data and has adaptive moment, such that I can reach the optimal valley quickly, and minimise risks of overshooting. Using Adam optimizers has shown to give the most accurate results in this project. [38]

Batch and Epochs

Having set up the loss functions and optimizers, the neural network is ready to be trained but for how long and how many examples do we show the network before optimising the network?

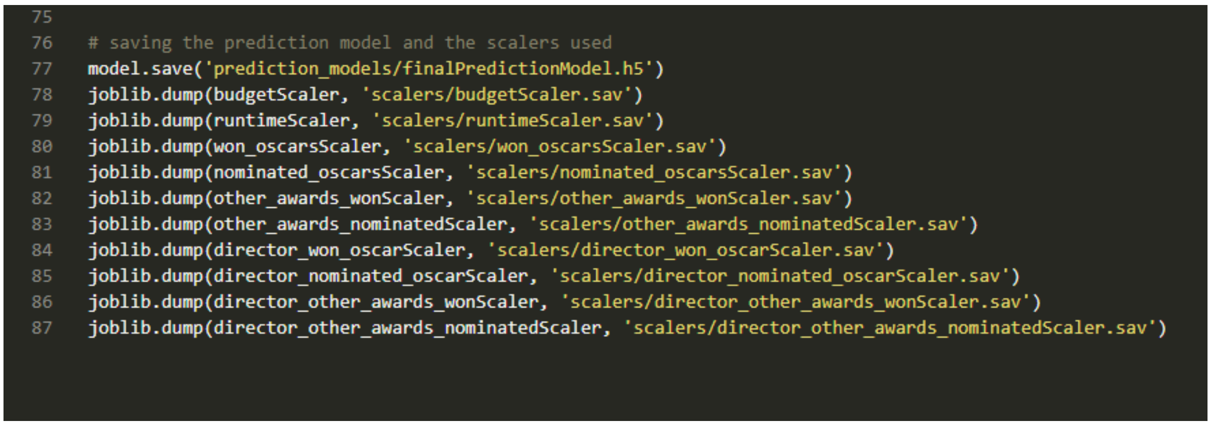
The number of examples shown to the network before the optimiser adjusts the network is called batch. The bigger the batch the quicker the training of the model as there less optimizes that must be made. But this comes at the cost of accuracy. While developing and experimenting with the architecture of the neural network, the batch number was kept high to quickly run through the training to see the estimated prediction accuracy. Ideally the network would optimize itself after every example it sees but can be time consuming when wanting to quickly build and test a specific configuration of the neural network. After the neural network is completed, batch number of 1 is used to produce the most accurate version of the prediction model.

Epochs are the number of times the entire dataset is put through the neural network. Having a larger epoch can create a more accurate prediction model. There is a risk of overtraining the model with too many epochs that the accuracy drops when the model is exposed to unseen data. The Dropout layer helps to prevent overtraining. During the development of this, different epoch values have been experimenting with and after 12 epochs the accuracy of the model plateaus or decreases slightly, meaning any more epochs are redundant or decrease the predictive power of the model.

Results and saving the prediction model

The neural networked was trained on 80% of the 30’000 movies from the *movies.csv* dataset. The 20% was kept for testing to evaluate how accurate was the model in prediction movie rating on the movies it was not trained on. On average, the accuracy result is 37%. This means the model got the star rating exactly right 37% percent of the time, but this does not account for prediction that were off by even 1-star rating. If the movie was a rating of 8 but the model predicted 7, it’s treated as 0% accurate. The tests and evaluation of the prediction model will be discussed in depth in the following chapter.

After training, the model is saved to a “finalPredictionModel.h5” file on the computer. This can be loaded up by another script and be used to make prediction. The scaling values used to normalise the data are each saved to a file, so the same scaling and normalising values can be used on new single point data before running through the model to the movie rating new predictions



```python

# saving the prediction model and the scalers used
model.save('prediction_models/finalPredictionModel.h5')
joblib.dump(budgetScaler, 'scalers/budgetScaler.sav')
joblib.dump(runtimeScaler, 'scalers/runtimeScaler.sav')
joblib.dump(won_oscarsScaler, 'scalers/won_oscarsScaler.sav')
joblib.dump(nominated_oscarsScaler, 'scalers/nominated_oscarsScaler.sav')
joblib.dump(other_awards_wonScaler, 'scalers/other_awards_wonScaler.sav')
joblib.dump(other_awards_nominatedScaler, 'scalers/other_awards_nominatedScaler.sav')
joblib.dump(director_won_oscarScaler, 'scalers/director_won_oscarScaler.sav')
joblib.dump(director_nominated_oscarScaler, 'scalers/director_nominated_oscarScaler.sav')
joblib.dump(director_other_awards_wonScaler, 'scalers/director_other_awards_wonScaler.sav')
joblib.dump(director_other_awards_nominatedScaler, 'scalers/director_other_awards_nominatedScaler.sav')
```

Code snippet of writing the prediction model and scaler values to file

4.3. Development of the web application

The second part of this project was the development of an interactive web application that users could enter their own details about a movie that plan to see the predicted rating for that movie using the prediction model. The goal was to make it simple and clear to use, without too many fields to not overwhelm the user. The user would need to enter movie details like who will be the actors in the movie, the director, the budget for the movie production, runtime of the movie and select the genres for the movie. The user clicks predict, the server runs the prediction and displays a results page with the star prediction and movie detail summary underneath, with a button at the to go back to the form page to predict another movie.

Front End

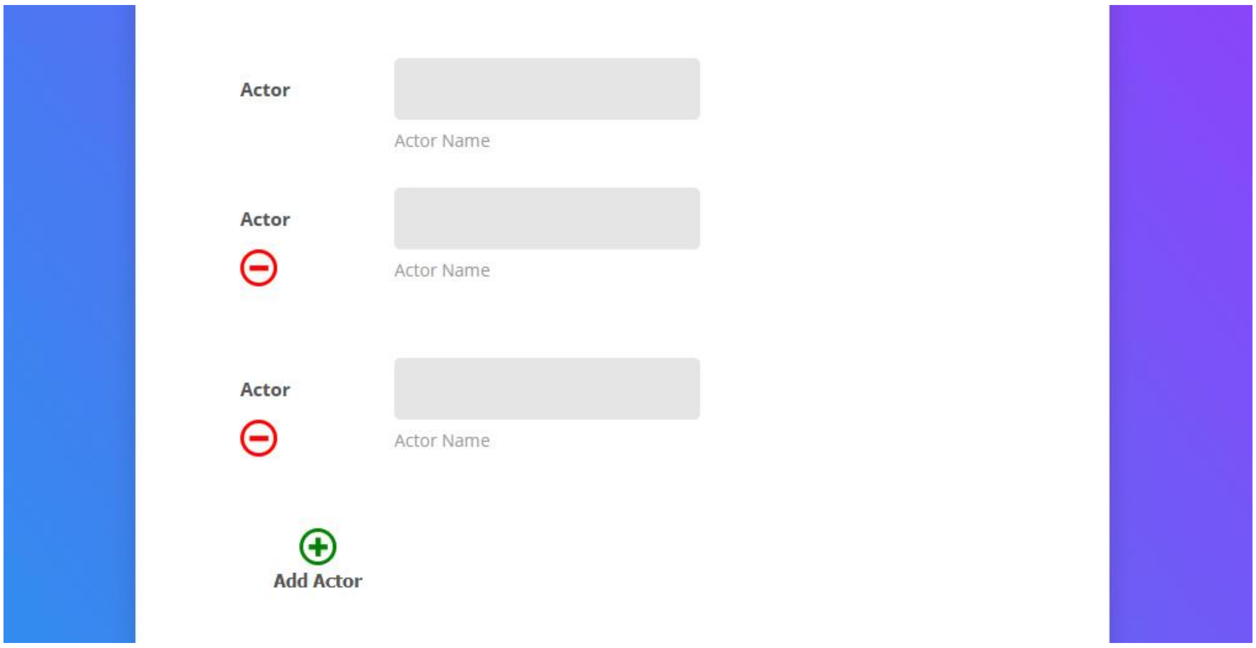
The front end of the website is a HTML web pages, since it is a web app. To make the web page dynamic, such as pressing a button to add extra actor input fields, JavaScript was used. JavaScript also does a part of the form validation, discussed further on. To make the website look clean and appealing to users, Bootstrap was used. A free to use Bootstrap template was downloaded only to provide the base CSS code, the HTML aspects were rewritten to make a form for movie details [39].

The User Interface

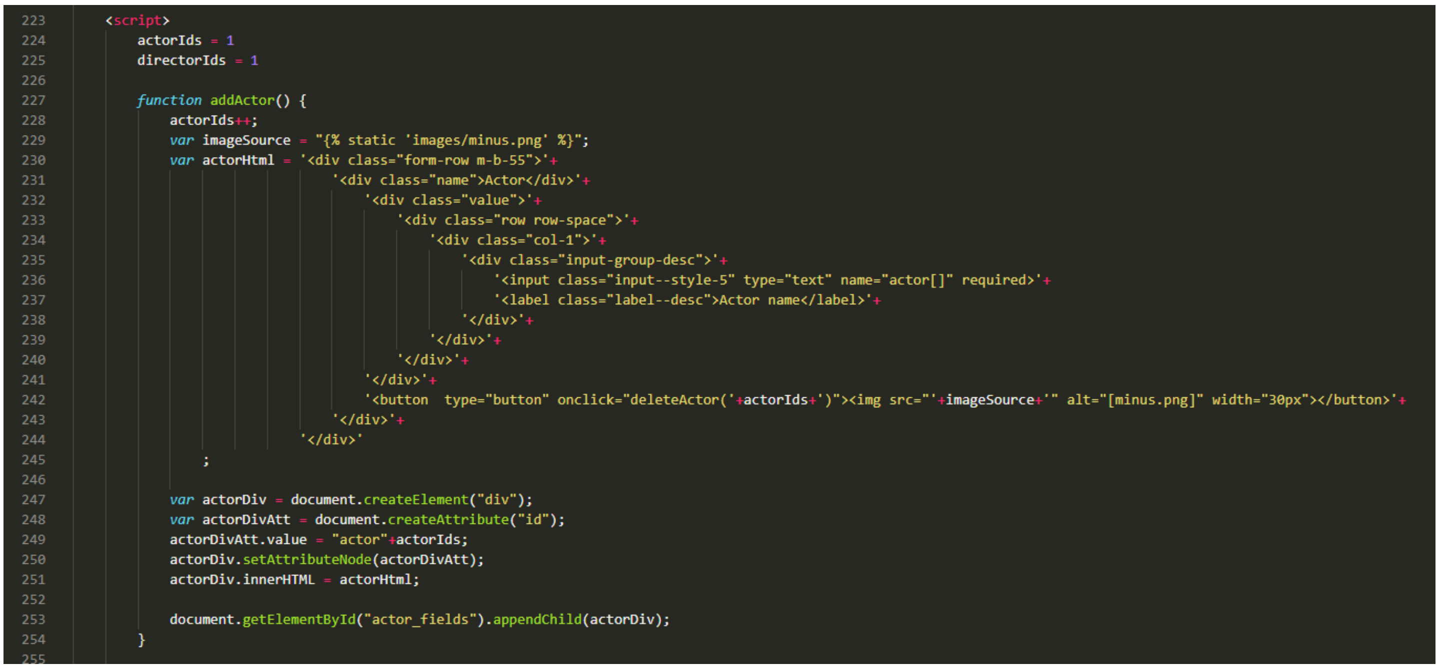
There are 2 pages web pages on the web application. This was not for lack of effort but a design decision. The goal was to keep the user interface very straight forward and clear to use. There is a form page where you enter movie details, and a results page where you see the predicted ratings for that movie, with a bottom of each page to go back and forth. In art they say, “less is more”, with less user interface clutter and no unnecessary menus and navigation bars, the users are no overwhelmed and have a good user experience. The evaluation of the user experience will be discussed in the next chapter.

Bootstrap is an open source front-end development framework. This project used bootstrap to make the nice-looking web page design [40]. Bootstrap was chosen for this project for it's easy of use, automatic scaling of webpages to make them suitable for mobile devices and availability of web page bootstrap templates to get a foundation and build the rest on top of. This sped up the development process of the project and allowed more time and focus on development the inner logic of the server.

On the first page of the web app, there is the movie detail form. The form is written in HTML with some JavaScript elements to make it dynamic. There buttons to add and remove actor and director fields. During the development and evaluation of earlier version of the form page, there would have been static amount of actor fields. For example, there were 5 actor fields, only allowing for maximum of 5 actors to be entered. During user experience evaluations, the users wanted to add more actors or felt obligated to fill all 5 fields even if only wanted to add less than 5 actors. To solve this, JavaScript was used to make the form dynamic and add buttons to dynamically add actor and fields input fields into the form and well as buttons to delete unwanted actor fields.



Here is a screenshot of the form with extra actor fields. The first actor field stays static, this is a design choice to ensure that there is always an actor field available and that an actor is submitted with the form. The extra fields can be removed if so inclined. The same logic applied to movie director fields.



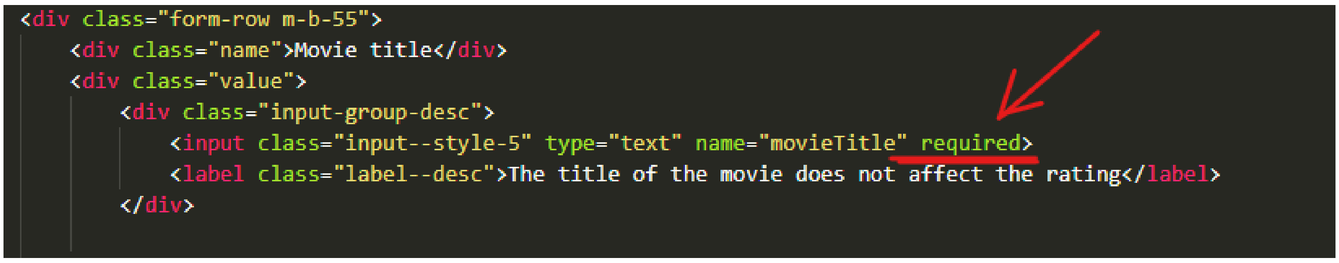
```
<script>
    actorIds = 1
    directorIds = 1

    function addActor() {
        actorIds++;
        var imageSource = "{% static 'images/minus.png' %}";
        var actorHtml = '<div class="form-row m-b-55">'+
                            '<div class="name">Actor</div>'+
                                '<div class="value">'+
                                    '<div class="row row-space">'+
                                        '<div class="col-1">'+
                                            '<div class="input-group-desc">'+
                                                '<input class="input--style-5" type="text" name="actor[]" required>'+
                                                '<label class="label--desc">Actor name</label>'+
                                            '</div>'+
                                        '</div>'+
                                    '</div>'+
                                '</div>'+
                                '<button  type="button" onclick="deleteActor('+actorIds+')"><img src="'+imageSource+'" alt="[minus.png]" width="30px"></button>'+
                            '</div>'+
                        '</div>'
            ;

        var actorDiv = document.createElement("div");
        var actorDivAtt = document.createAttribute("id");
        actorDivAtt.value = "actor"+actorIds;
        actorDiv.setAttributeNode(actorDivAtt);
        actorDiv.innerHTML = actorHtml;

        document.getElementById("actor_fields").appendChild(actorDiv);
    }
```

Here is the JavaScript code for the button functionally and how it adds the extra actor fields. The director button has the exact functionally except the variable's names are different. This JavaScript is located within the *index.html* (the front page) file and not external in a separate JavaScript file. This allows for quicker development and simplicity of the code base. The JavaScript used here is ONLY used by this HTML page and no where else, for quick client dynamic form functionality.

Form Validation

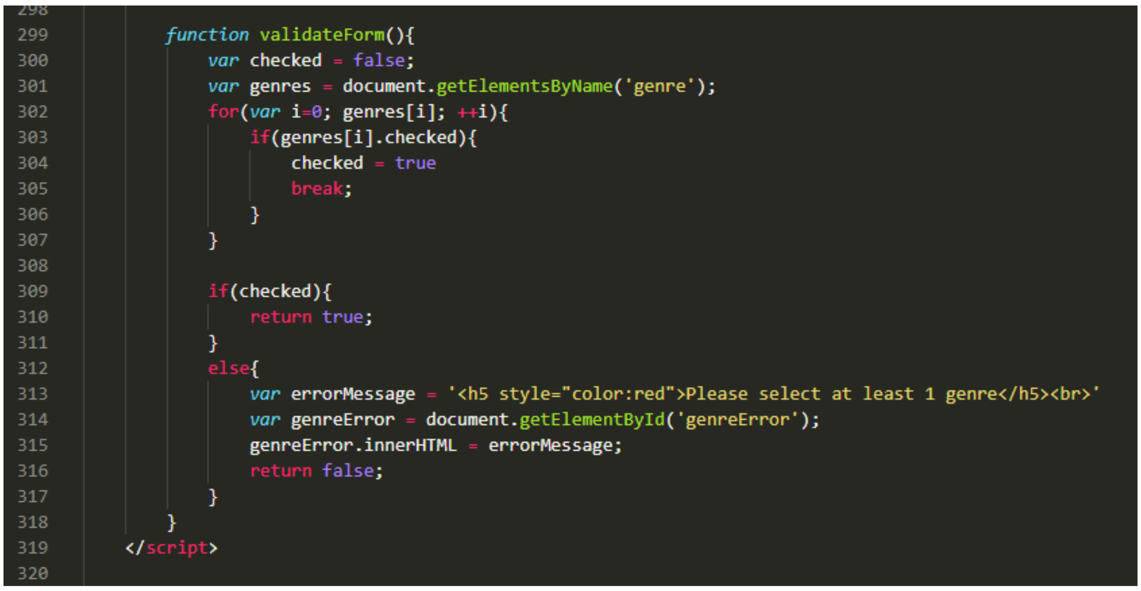
Before the form is submitted to the server, it is important to ensure the data entered the fields is valid and not must importantly not left empty, otherwise it will cause expression error on the server. Like when the server tries to read from the empty array of genres.

HTML 5 has some handy features that make form validation a lot easier to implement. In simple text and number fields, the *<input>* HTML tags can have a parameter called "required". If the user tried to submit a from with an empty field, the browser will create a pop up with a pointer to the empty field with a message saying, "Please fill out this field".



```
<div class="form-row m-b-55">
    <div class="name">Movie title</div>
    <div class="value">
        <div class="input-group-desc">
            <input class="input--style-5" type="text" name="movieTitle" required>
            <label class="label--desc">The title of the movie does not affect the rating</label>
        </div>
```

The only exception to this rule is the genre field. The genre are checkbox boxes, so having a required parameter in their input would mean the browser would require the user to check all those boxes. But the validation should only ensure that at least 1 genre was checked. HTML does not provide a quick solution to this, so there had to be a JavaScript function written to check this.



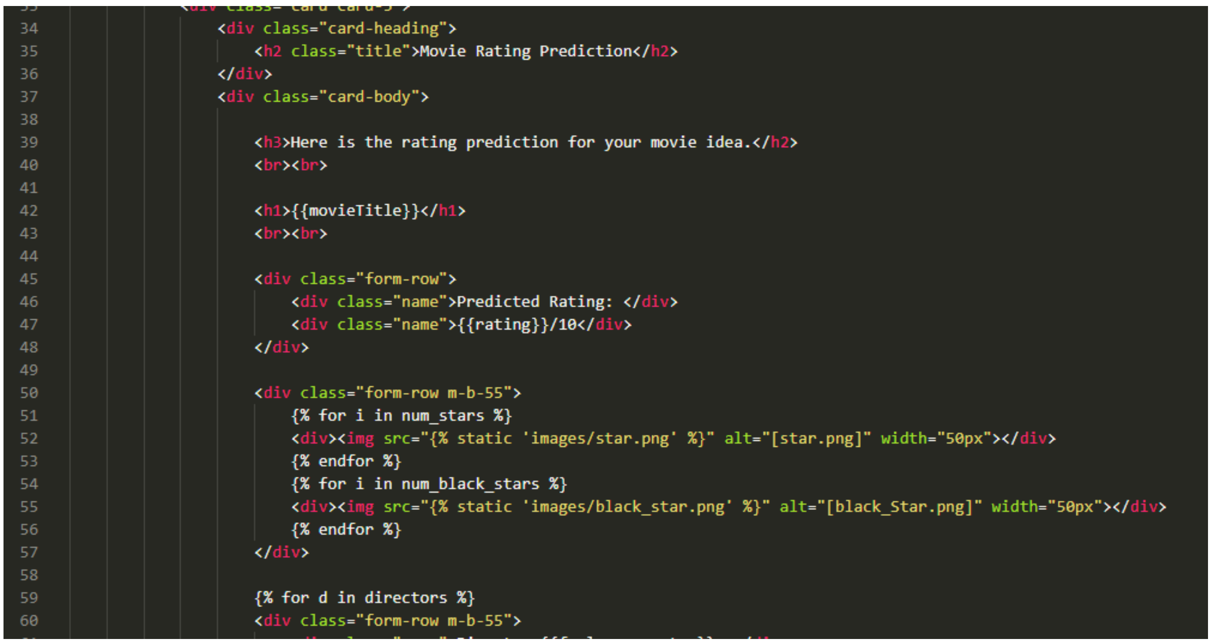
```
    function validateForm(){
        var checked = false;
        var genres = document.getElementsByName('genre');
        for(var i=0; genres[i]; ++i){
            if(genres[i].checked){
                checked = true
                break;
            }
        }

        if(checked){
            return true;
        }
        else{
            var errorMessage = '<h5 style="color:red">Please select at least 1 genre</h5><br>'
            var genreError = document.getElementById('genreError');
            genreError.innerHTML = errorMessage;
            return false;
        }
    }
</script>
```

Here is the form validation JavaScript function (located on the *index.html* file). This is only needed for the genres fields as all other field use the “required” parameter in their input tags. This iterates through all genre input checkboxes and checks if they have been “checked” selected. If none have been selected, the JavaScript adds an error message just above the Submit button to let the user know to make at least one genre selection. Once everything is in order, the form gets submitted to the server.

Results page

Once the server processes the form and makes a movie rating prediction, it redirects the browser to the results page. The results page is a dynamically generated HTML page. The code is in the results *results.html* file.

Unlike the from page, the dynamic generation of the results page is done with the Django framework. No JavaScript is needed here because there are no dynamic elements that need to be changed on the client side. The results page is generated by the Django server, with the movie prediction and the star images with a summary of the passed in movie details from the form.



```
<div class="card-heading">
    <h2 class="title">Movie Rating Prediction</h2>
</div>
<div class="card-body">

    <h3>Here is the rating prediction for your movie idea.</h2>
    <br><br>

    <h1>{{movieTitle}}</h1>
    <br><br>

    <div class="form-row">
        <div class="name">Predicted Rating: </div>
        <div class="name">{{rating}}/10</div>
    </div>

    <div class="form-row m-b-55">
        {% for i in num_stars %}
        <div><img src="{% static 'images/star.png' %}" alt="[star.png]" width="50px"></div>
        {% endfor %}
        {% for i in num_black_stars %}
        <div><img src="{% static 'images/black_star.png' %}" alt="[black_Star.png]" width="50px"></div>
        {% endfor %}
    </div>

    {% for d in directors %}
    <div class="form-row m-b-55">
```

Here is a snippet of the HTML code for the results page. There is some Django code within the HTML, all use elements that are surrounded by curly brackets. That's python code that is used to dynamically generate HTML elements and display the variables taken from the passed in web page "context", which will be discussed further. So, for example if the movie was rated 6 starts, the Django code generated the *<img>* tag of the star image 6 times to make 6 stars appear on the web page. This also dynamically creates HTML tags for each actor and director which the user had entered in the previous form.

The results page still uses the same bootstrap elements and CSS code from the index page.

Django Server (Middle-Tier)

The web application is hosted on a Django server. Django is an open source python web framework. It handles a lot of the technical stuff in the background on its own allowing to focus on implementing the logic for your server for faster development. Since Django is python based, the prediction model made with python Keras can be easily and seamlessly integrated into the Django web application.

Server Architecture

In this project, the web application is in the "DjangoApp" folder. There is a "manage.py" python that needs to be run with python. To start up the server, run "***python ./manage.py runserver***" and it will startup the server and host it on localhost port 8000, or just at ***http://127.0.0.1:8000/***

The Django server has a main folder where it keeps the settings python scripts and management settings for the whole server. In this project it is the "***MovieRatingPredicto***r". There is a URL settings and list of installed app.

The way Django framework works is by modularising projects within the server into app components. Each part of the web app is an app by itself. In the main settings of the Django server, the URL is used to navigate to different apps. In a sense, one Django server can host multiple web apps simultaneously. This project only needs one web app to act a user-friendly interface for the prediction model. Each app is modularised into a separate folder, and the web app folder that holds all the server logic and web pages and python scripts for the movie ratings app is "***moviepredict***".

Since the "***moviepredict***" app folder lie all the python scrips and folder that hold the logic for the movie ratings prediction app. There are several settings python files, but the important one that have this projects logic are:

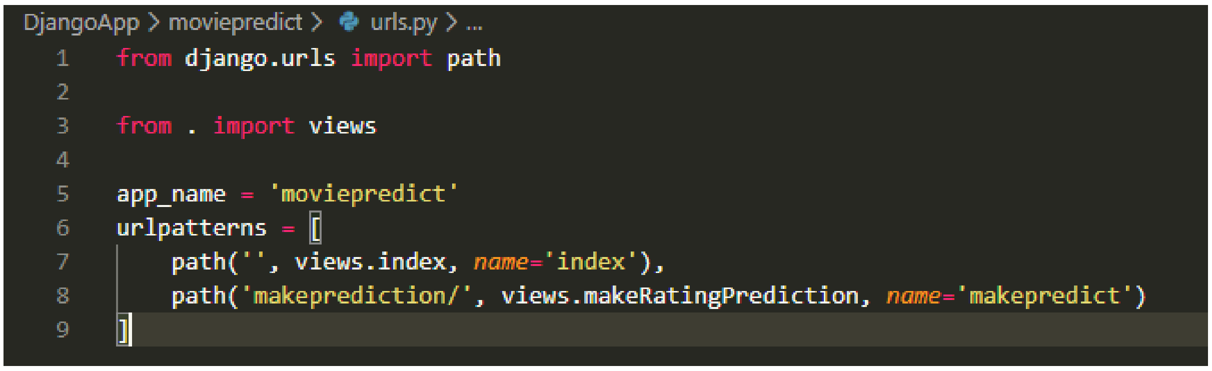
- make_prediction.py
- models.py
- tests.py
- views.py

There are sub-folders that keep extra files, like "***prediction_models***" hold the prediction model file that is used to make movie rating predictions. There is the "***scalers***" folder that holds files for the Scaler values used during data normalisation. "***static***" holds the image, CSS and JavaScript files that are used by the web apps pages. Here is where the bootstrap CSS and JavaScript are stored. ***"templates***" hold the HTML files that are used to generate the web page views.

When a URL is used to access a page within the web app, the ***urls.py*** picks that and runs a function from ***views.py*** to generate a HTML page to send to the client user. ***views.py*** returns HTML pages from the "templates" folder.

Views

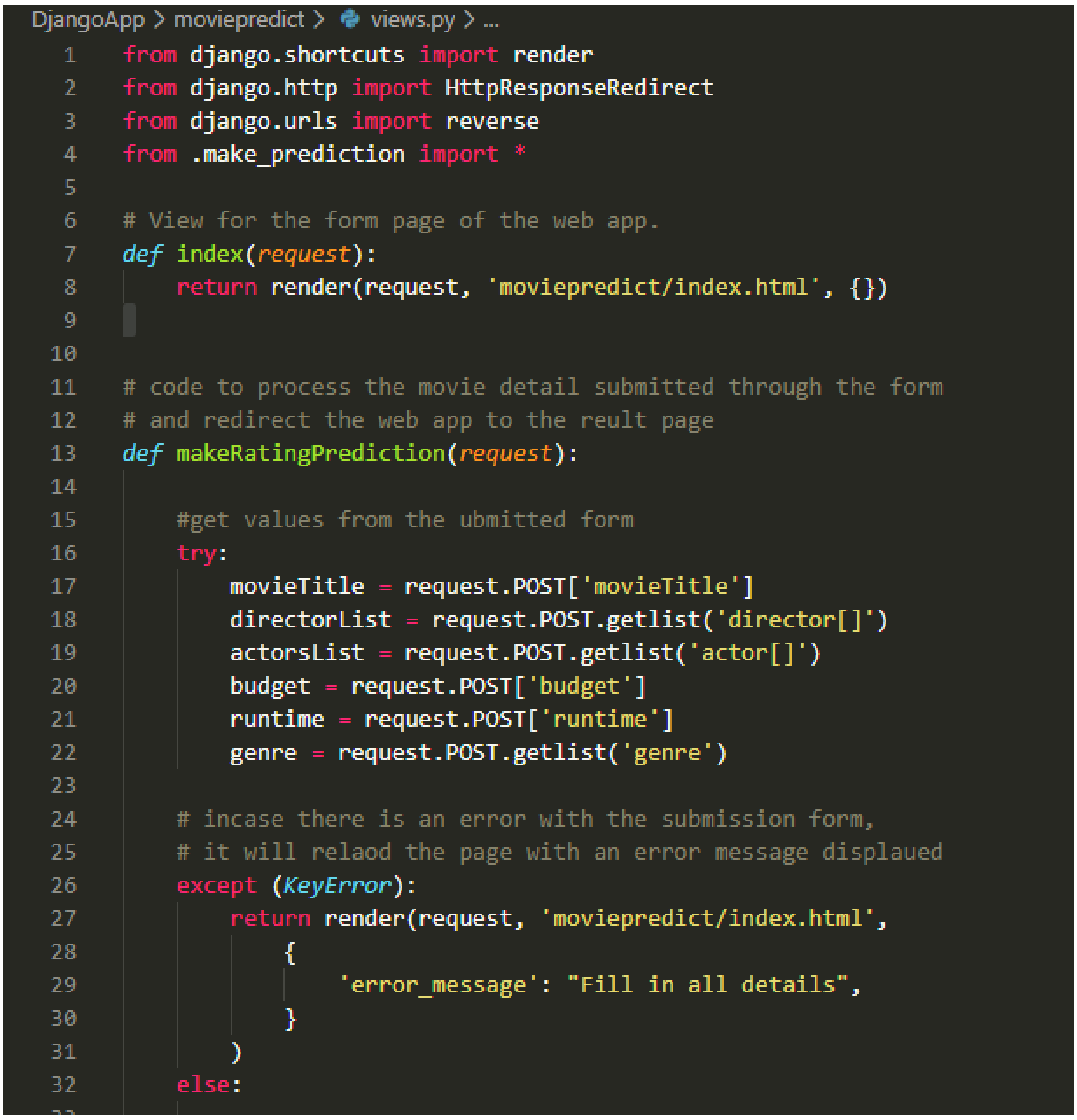
View.py is the main python script in the Django app that generates web pages for the users. ***Views.py*** functions are called by the Django server when a specific URL is access by the client. ***urls.py*** maps the URLs to ***views.py*** functions.



DjangoApp > moviepredict > urls.py > ...

```python
from django.urls import path

from . import views

app_name = 'moviepredict'
urlpatterns = [
    path('', views.index, name='index'),
    path('makeprediction/', views.makeRatingPrediction, name='makepredict')
]
```

urls.py



DjangoApp > moviepredict > views.py > ...

```python
from django.shortcuts import render
from django.http import HttpResponseRedirect
from django.urls import reverse
from .make_prediction import *

# View for the form page of the web app.
def index(request):
    return render(request, 'moviepredict/index.html', {})


# code to process the movie detail submitted through the form
# and redirect the web app to the reult page
def makeRatingPrediction(request):

    #get values from the ubmitted form
    try:
        movieTitle = request.POST['movieTitle']
        directorList = request.POST.getlist('director[]')
        actorsList = request.POST.getlist('actor[]')
        budget = request.POST['budget']
        runtime = request.POST['runtime']
        genre = request.POST.getlist('genre')

    # incase there is an error with the submission form,
    # it will reload the page with an error message displaued
    except (KeyError):
        return render(request, 'moviepredict/index.html',
            {
                'error_message': "Fill in all details",
            }
        )
    else:
```

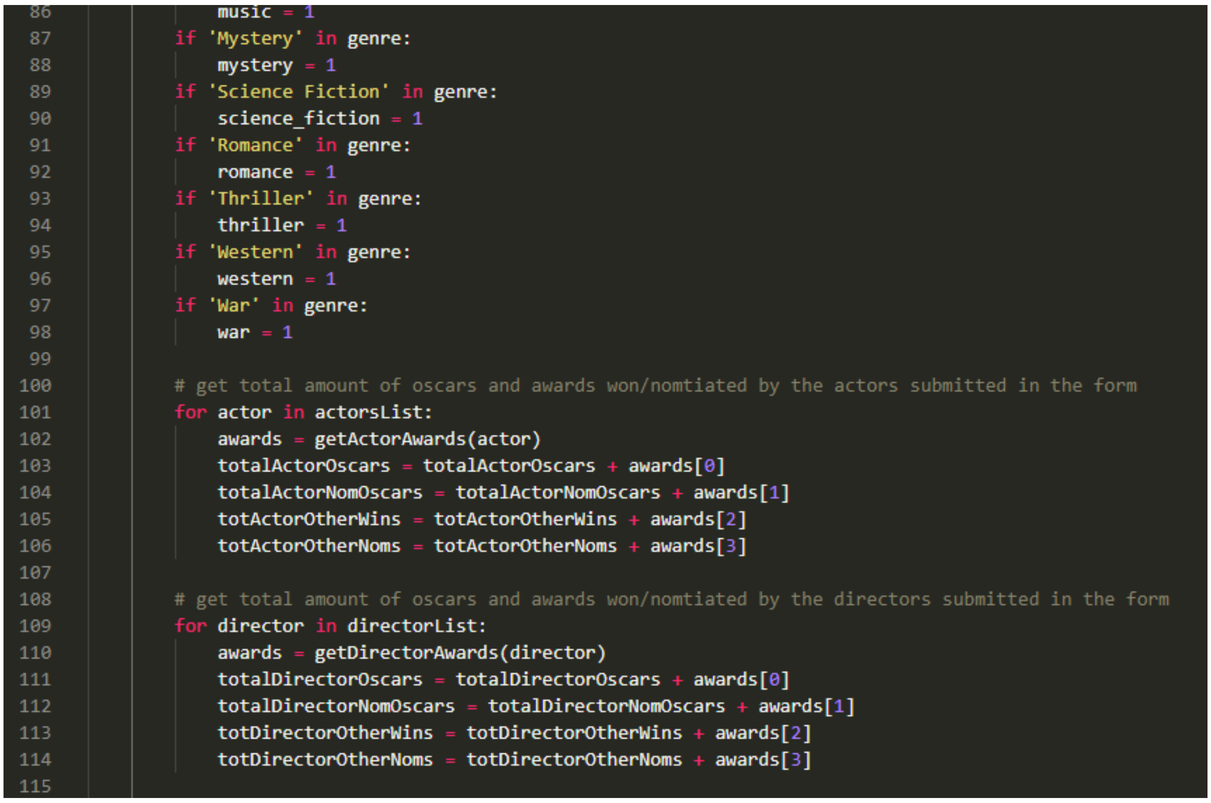
Code snippet of views.py

Views.py has two functions, corresponding the 2 web app pages. ***index(***) is called when the client access the home page of the web app. This sends the user the form page HTML, ***index.html*** (in the templates folder) to fill in movie details of the movie what a ratings prediction for. When the user clicks the submit button on the form, the form is submitted and the ***makeRatingPrediciton()*** is called.

makeRatingPrediciton() function in ***views.py*** processes the data submitted by the forms, uses functions from make_prediction.py to get the rating and returns the results page. In ***makeRatingPrediciton()*** "requests" is passed into the function as a parameter, and this objects holds all the submitted data from the form. The list of actors, list of directors, list of selected movie, genres, budget and runtime are extracted from the request object and stored in variables.

First set in the data pre-processing is to One Hot Encode the genres into 1s and 0s. This is exactly how the genres where pre-processed during the neural network training. It's essential the data is pre-processed that same way when using the prediction model.

Next steps are to find how many awards have been won or nominated by each actor and director that was submitted by the client user. There are two loops, one for actors and one for directors, which iterate through each actor, and call the ***getActorAwards()*** function from the ***make_prediction.py*** script to get the amount of words they have received. The ***getActorAwards()*** function return an array with integer number which corresponds to number of Oscars won, number of Oscar nominated for, number of other awards won and number of other awards nominated for, respectfully. How this function works will be discussed in the next section.

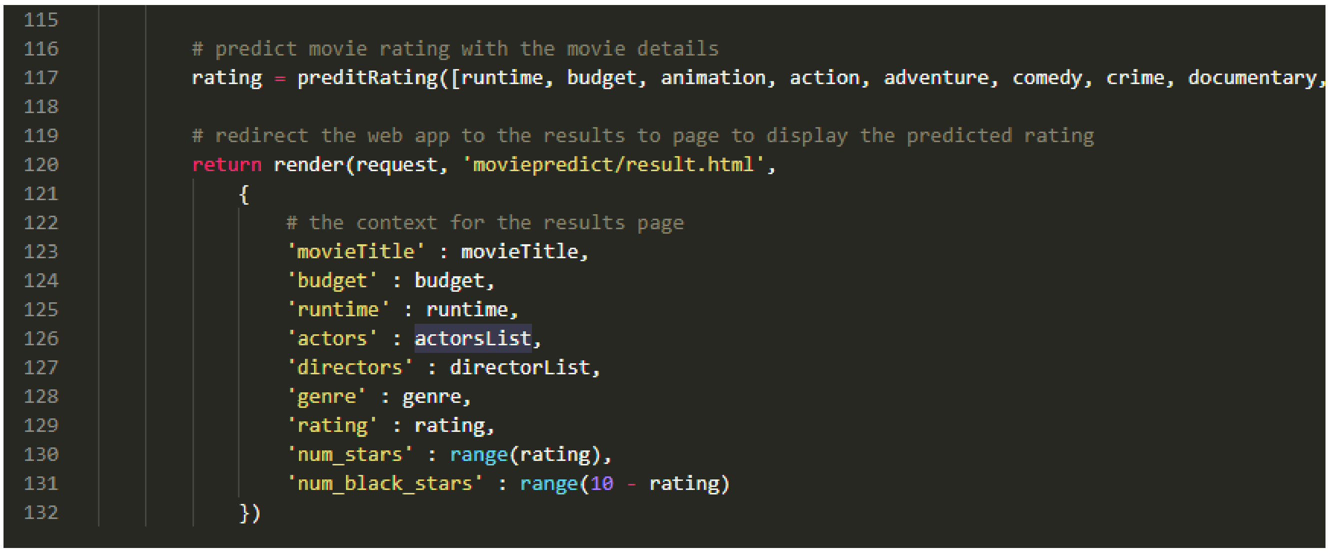


```
            music = 1
        if 'Mystery' in genre:
            mystery = 1
        if 'Science Fiction' in genre:
            science_fiction = 1
        if 'Romance' in genre:
            romance = 1
        if 'Thriller' in genre:
            thriller = 1
        if 'Western' in genre:
            western = 1
        if 'War' in genre:
            war = 1

        # get total amount of oscars and awards won/nomtiated by the actors submitted in the form
        for actor in actorsList:
            awards = getActorAwards(actor)
            totalActorOscars = totalActorOscars + awards[0]
            totalActorNomOscars = totalActorNomOscars + awards[1]
            totActorOtherWins = totActorOtherWins + awards[2]
            totActorOtherNoms = totActorOtherNoms + awards[3]

        # get total amount of oscars and awards won/nomtiated by the directors submitted in the form
        for director in directorList:
            awards = getDirectorAwards(director)
            totalDirectorOscars = totalDirectorOscars + awards[0]
            totalDirectorNomOscars = totalDirectorNomOscars + awards[1]
            totDirectorOtherWins = totDirectorOtherWins + awards[2]
            totDirectorOtherNoms = totDirectorOtherNoms + awards[3]

```

Code snippet showing some One Hot Encoding of the genres into 1s and 0s, and the loops used to sum up the total of actor and director awards.

The loops sum up all each set of won/nominated awards for actors and directors separately. This is also how the actor and director awards were treated in the processing for the neural networking training. Once this set up correctly, it's passed into the ***predictRating()*** function from ***make_prediction.py*** script. This returns an integer movie rating prediction.

Lastly, ***makeRatingPrediciton()*** sends the ***results.html*** web page back to the client. The python dictionary list is passed as the context for the webpage, with variables holding the data that will be used to dynamically generate the results page for the client. For example, the predicted movie rating is passed to the ***results.html*** and the python code within the ***results.html*** will use the rating value to generate that many number of star images on the page, to visually represent the movie rating prediction for the user.

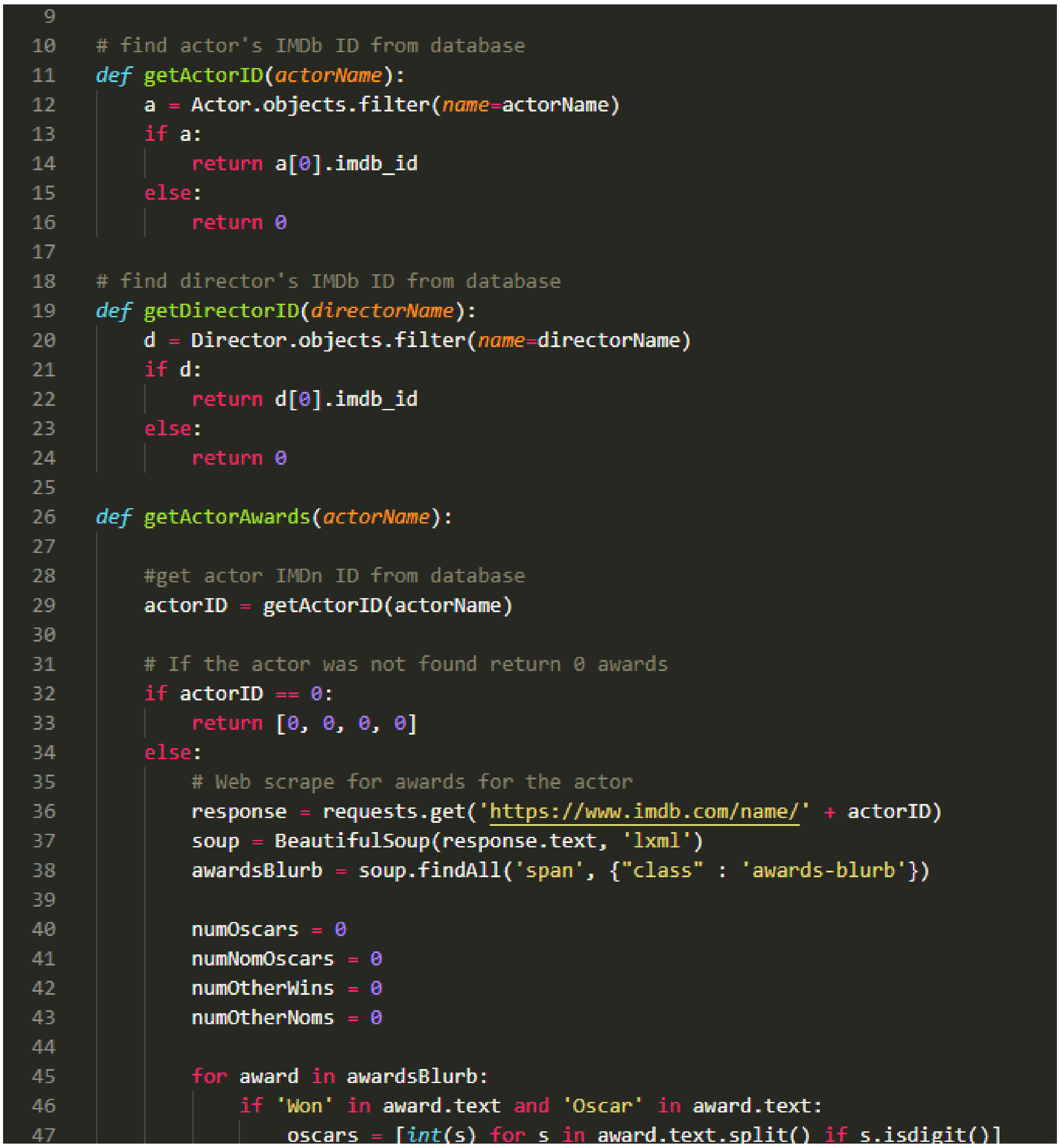


```

        # predict movie rating with the movie details
        rating = preditRating([runtime, budget, animation, action, adventure, comedy, crime, documentary,

        # redirect the web app to the results to page to display the predicted rating
        return render(request, 'moviepredict/result.html',
            {
                # the context for the results page
                'movieTitle' : movieTitle,
                'budget' : budget,
                'runtime' : runtime,
                'actors' : actorsList,
                'directors' : directorList,
                'genre' : genre,
                'rating' : rating,
                'num_stars' : range(rating),
                'num_black_stars' : range(10 - rating)
            })
```

*Code snippet showing where the **predictRating()** function is called and the web app redirect to results.html with context passing the movie prediction parameters to be displayed by the results web page*

Making a rating prediction

make_prediction.py is the python script that uses the prediction model to generate the prediction. The code was separated from ***views.py*** as to not clutter up the code there and keep this decoupled and modularised, for good coding practices. ***make_prediction.py*** has multiple functions which are called by views.py. The first one used was ***getActotAwards()*** function (or ***getDirectorAwards()*** for directors), this takes in the actor's name as a perimeter. Since the awards for actors is not stored on the Django database, they need to be web scraped off the IMDb website. To do this, the actors IMDb ID is needed to create the URL link for the actor to web scrape. So, there is a quick function called ***getActorsID()*** that queries that Django database for an actor of that name to return the IMDb ID. If the actor is not found, then ID of 0 is return, and that actors is treated as have no awards won or nominated.

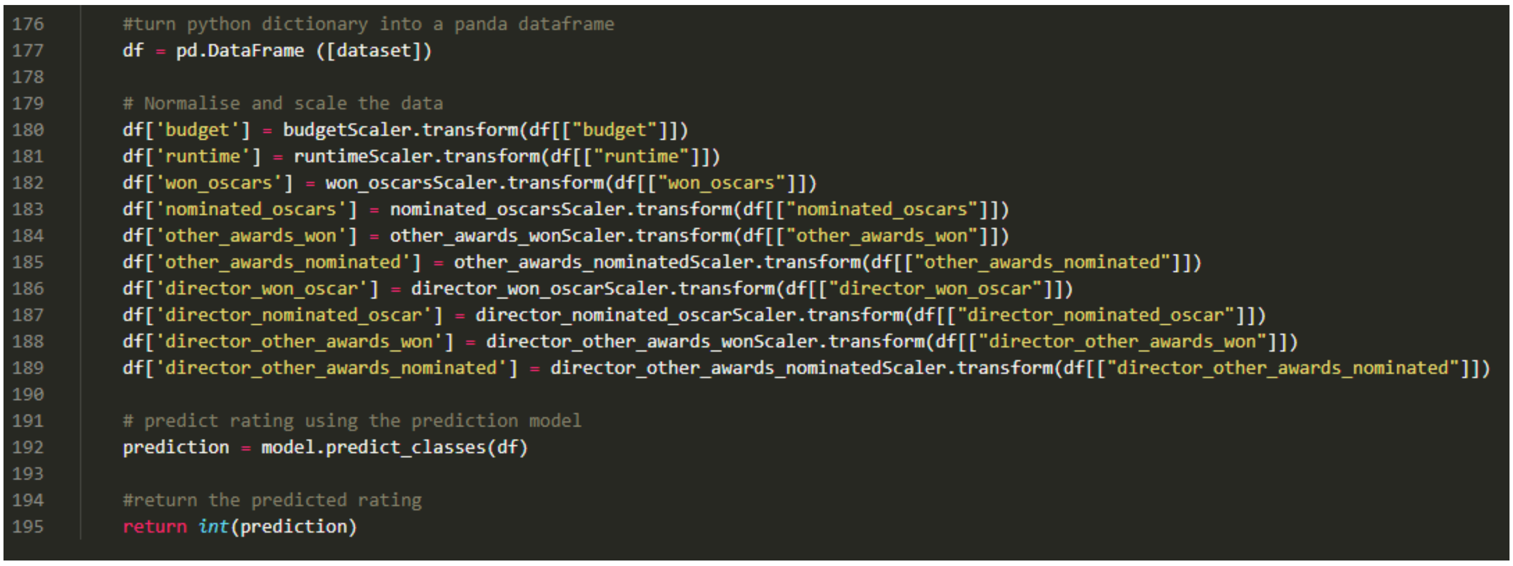


```python

# find actor's IMDb ID from database
def getActorID(actorName):
    a = Actor.objects.filter(name=actorName)
    if a:
        return a[0].imdb_id
    else:
        return 0

# find director's IMDb ID from database
def getDirectorID(directorName):
    d = Director.objects.filter(name=directorName)
    if d:
        return d[0].imdb_id
    else:
        return 0

def getActorAwards(actorName):

    #get actor IMDn ID from database
    actorID = getActorID(actorName)

    # If the actor was not found return 0 awards
    if actorID == 0:
        return [0, 0, 0, 0]
    else:
        # Web scrape for awards for the actor
        response = requests.get('https://www.imdb.com/name/' + actorID)
        soup = BeautifulSoup(response.text, 'lxml')
        awardsBlurb = soup.findAll('span', {"class" : 'awards-blurb'})

        numOscars = 0
        numNomOscars = 0
        numOtherWins = 0
        numOtherNoms = 0

        for award in awardsBlurb:
            if 'Won' in award.text and 'Oscar' in award.text:
                oscars = [int(s) for s in award.text.split() if s.isdigit()]
```

Here is the code snippet for functions discussed above. When the IMDb ID is retuned for that actor, it is used to generate a URL link to that actor. Then similarly to how actor awards were web scraped during the data gather part of this project, the python library “Requests” is used to make the HTTP request and return the HTML for that website. Python library “BeautifulSoup” is used to parse through the retuned HTML and extract the number of Oscars/awards that been won/nominated and retuned to views.py as an array. If the actor had won an Oscar and had 5 other awards and no nominations the returned array would look like this: [1,0,5,0].

The other, and perhaps the most important function of the whole Movie Rating Prediction app is the ***predictRating()*** function. Here the prediction model is loaded from file, along with the scaler values that were made during the neural network training. This takes in all the movie details from views.py as an array, converts into a python dictionary. This step is done to add column names to the data. This will make it easy to scale and normalise. The python dictionary is converted into a Panda data frame. Now we can reuse code from before to scale and normalise the data. The scaler values that are used need to be same as the ones used when normalising the training dataset, as this makes the ne new movie data on the same scale as all other movies from before which were used to train the model. This will ensure to give the most accurate results.

Once the data scaled and normalised is passed into prediction model and the movie rating prediction is generated. It's stored in the "prediction" variable. Since it's a string, it's cast into an int and returned to ***views.py*** which is then past to the results page via the context parameter.



```
    #turn python dictionary into a panda dataframe
    df = pd.DataFrame ([dataset])

    # Normalise and scale the data
    df['budget'] = budgetScaler.transform(df[["budget"]])
    df['runtime'] = runtimeScaler.transform(df[["runtime"]])
    df['won_oscars'] = won_oscarsScaler.transform(df[["won_oscars"]])
    df['nominated_oscars'] = nominated_oscarsScaler.transform(df[["nominated_oscars"]])
    df['other_awards_won'] = other_awards_wonScaler.transform(df[["other_awards_won"]])
    df['other_awards_nominated'] = other_awards_nominatedScaler.transform(df[["other_awards_nominated"]])
    df['director_won_oscar'] = director_won_oscarScaler.transform(df[["director_won_oscar"]])
    df['director_nominated_oscar'] = director_nominated_oscarScaler.transform(df[["director_nominated_oscar"]])
    df['director_other_awards_won'] = director_other_awards_wonScaler.transform(df[["director_other_awards_won"]])
    df['director_other_awards_nominated'] = director_other_awards_nominatedScaler.transform(df[["director_other_awards_nominated"]])

    # predict rating using the prediction model
    prediction = model.predict_classes(df)

    #return the predicted rating
    return int(prediction)
```

Code snippet showing the normalisation of movie details and generation of movie rating prediction.

Database (Back-end)

The original planned database technology for this project was to use a non-rational database like MongoDB. But since then, SQLite was a better option.

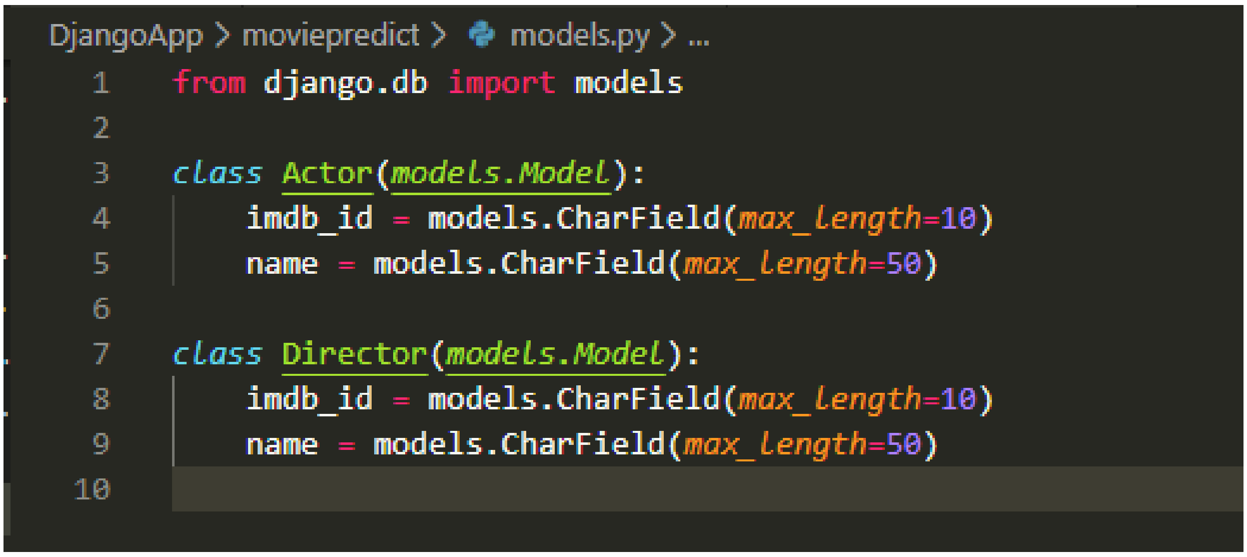
SQLite

SQL lite is a C based light version of SQL database that does not require running server [41]. Django natively integrates an SQLite database in its server and stores it a single SQLite3 file. Sqlite3 is the python module to connect and use SQL lite databases [42]. Django automatically takes of the connection to the database and to uses the models.py script to create objects to be stored serialised and stored within the SQLlite database.

This projected did not need a relational database. It simply needs to store IMDb ID to be matched to existing actors, to the actor awards can be web scraped for rating prediction.

Models

Each Django app has a ***model.py*** python script. Here Actor and Director objects structures are declared to then be stored on the database.

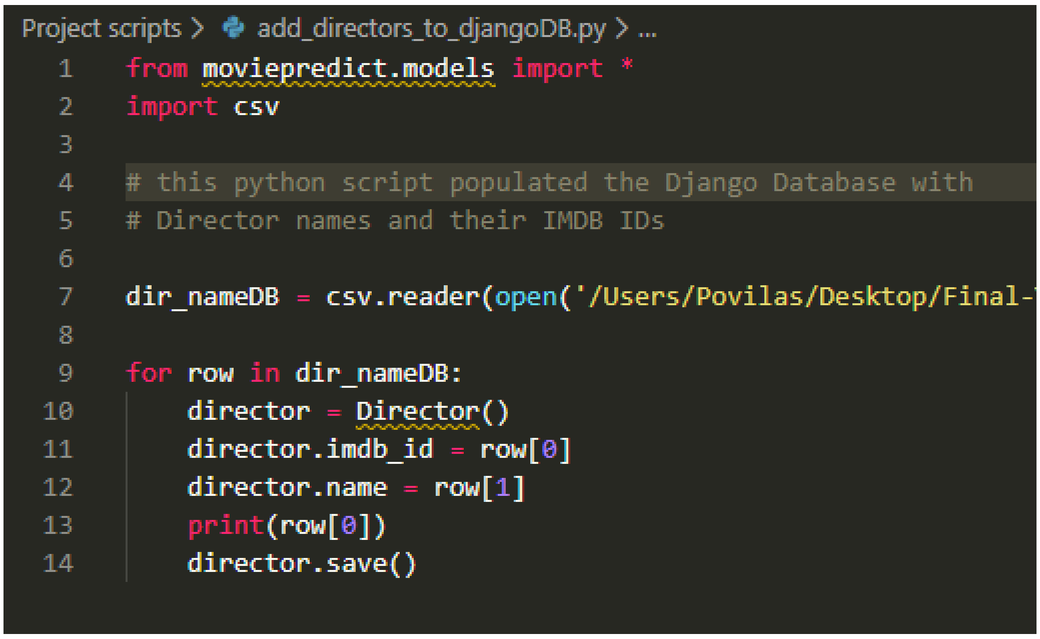


DjangoApp > moviepredict > models.py > ...

```python
from django.db import models

class Actor(models.Model):
    imdb_id = models.CharField(max_length=10)
    name = models.CharField(max_length=50)

class Director(models.Model):
    imdb_id = models.CharField(max_length=10)
    name = models.CharField(max_length=50)

```

Code Snipper of models.py showing how Actor and Director objects are stored in the database

The database was populated using a python script that used Django's underlying API of where it would import the models from ***model.py*** and it would create Actors objects, populate those fields with the actor names and their IMDb and store them on the database. The same logic goes for Directors.



Project scripts > add_directors_to_djangoDB.py > ...

```python
from moviepredict.models import *
import csv

# this python script populated the Django Database with
# Director names and their IMDB IDs

dir_nameDB = csv.reader(open('/Users/Povilas/Desktop/Final-

for row in dir_nameDB:
    director = Director()
    director.imdb_id = row[0]
    director.name = row[1]
    print(row[0])
    director.save()
```

Code snippet of python script used to parse csv file of director and add their names and IMDb to the Django database.

4.3. Conclusions

This chapter discussed the development of the neural network to predict movie ratings, and the development of the web application to use an easy to use interface to enter movie details of movies not made yet to predict what ratings they will get. It was discussed what kind of architecture the neural network had, and what approach was used to train the neural network. Discussed the type of algorithms and why they were chosen by design and experimentation to give the most accurate results. We discuss the architecture and the inner design of the logic used by the Django web app server to take user input and generate movie prediction and how it was processed to be displayed concisely to the end user.

In the next chapter we will discuss how the prediction model was tested and evaluated and what the prediction results means, and metrics were used to determine the accuracy of the model. As well as how the Django web app was tested and evaluated by real users using Nelsons heuristics.

5. Testing and Evaluation

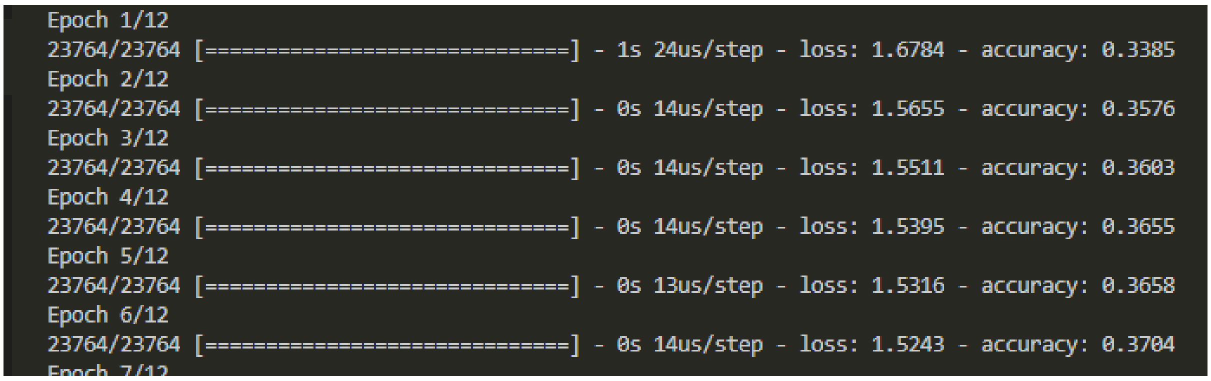
5.1 Introduction

Last chapter we discussed the project development. This chapter will explore how the code is tested to ensure proper functionally. Discuss how the neural network was tested and how the its accuracy is evaluated. Also, this chapter will discuss how the web application is unit tested and how was evaluated.

5.2 Prediction Model Evaluation

The testing and evaluation of the prediction model made by the neural network is the key to this whole project. Afterall, what good is a prediction model is if its nots make accurate, or at least close to accurate results. Therefore, there are several methods to test and evaluate the prediction model made in this project.

The first way to measure the accuracy of the prediction model is at the time of training the neural network. As training data is inputted into the model, the Keras python library gives a real time feedback of the loss and accuracy of the model. With the final accuracy value show at the end of each training epoch.



```
Epoch 1/12
23764/23764 [==============================] - 1s 24us/step - loss: 1.6784 - accuracy: 0.3385
Epoch 2/12
23764/23764 [==============================] - 0s 14us/step - loss: 1.5655 - accuracy: 0.3576
Epoch 3/12
23764/23764 [==============================] - 0s 14us/step - loss: 1.5511 - accuracy: 0.3603
Epoch 4/12
23764/23764 [==============================] - 0s 14us/step - loss: 1.5395 - accuracy: 0.3655
Epoch 5/12
23764/23764 [==============================] - 0s 13us/step - loss: 1.5316 - accuracy: 0.3658
Epoch 6/12
23764/23764 [==============================] - 0s 14us/step - loss: 1.5243 - accuracy: 0.3704
Epoch 7/12
```

Training accuracy results being outputted to the terminal during runtime

This quick feedback enables quick development and modification the neural network to see if any improvement of the results can be made. Experimenting with different neural network sets and combinations of middle layers and nodes provides accuracy differences, then the set up that gives the highest possible accuracy is kept.

The last aspect of accuracy evaluation during neural network training is evaluating the freshly trained network with the testing portion of the overall movie dataset. Before the network is trained, the dataset is split into training and testing datasets, each with the correction features and labels. In this project, 20% of the dataset was kept as testing set, so the neural network is trained on only 80%

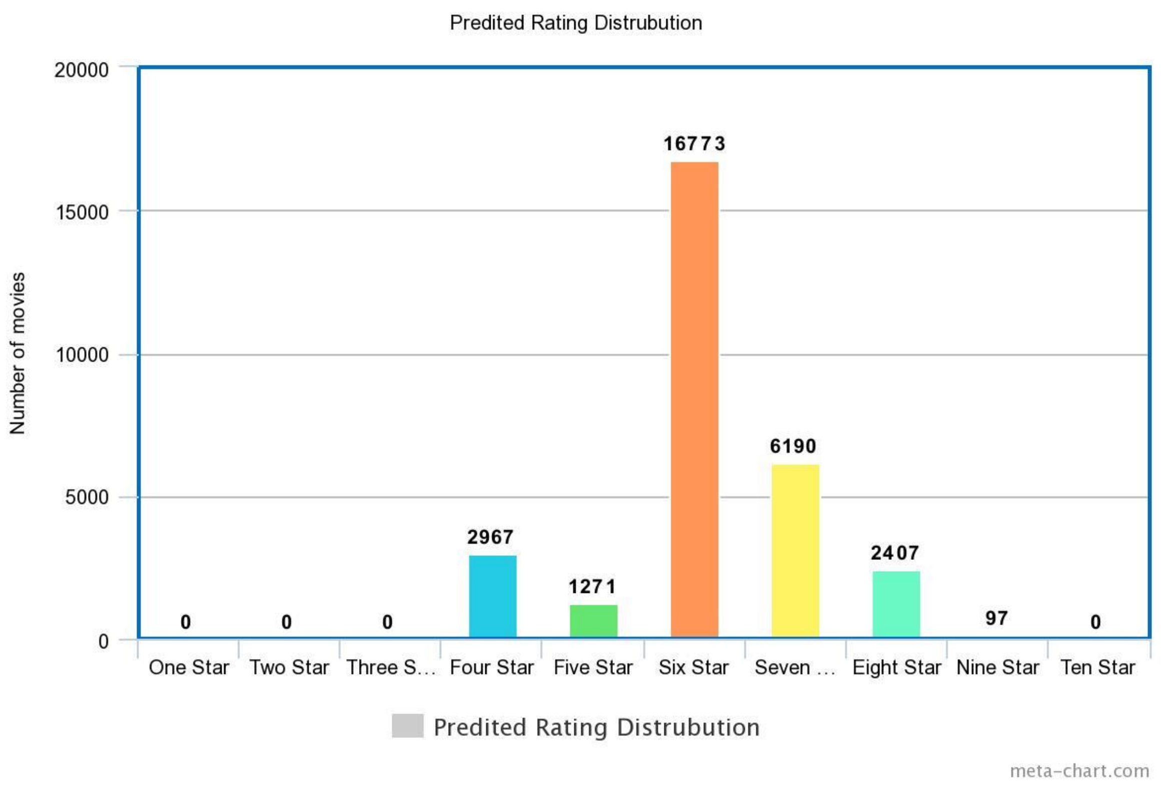
of the movies. It is important to determine if the neural network is not being over trained. It could be giving accurate results to examples it had seemed ten times but give be inaccurate with examples it had not seen before. Separating the training and testing data is a quick way to evaluate the accuracy of the neural network. This provides value feedback to how accurate the model is, and as mentioned before, allow for fast development of the neural network.

The average accuracy of the neural network during training is around 39%, and during evaluation of model using unseen data, is 37%. These values are very close and hence the model is not being over trained. The number here seems small, what it means is that out of 11 possible rating classes, the model gets it perfectly spot on 37% of the time. If the prediction was only 1 star off, for example, actual rating is 7 but the prediction was 6, the model treats it as 0% accurate. The neural network during training does not take into account the closeness of the prediction to the actual value. Even though the prefect prediction is 37%, that is significantly greater than pure chance, and demonstrates the model has effective predicting power.

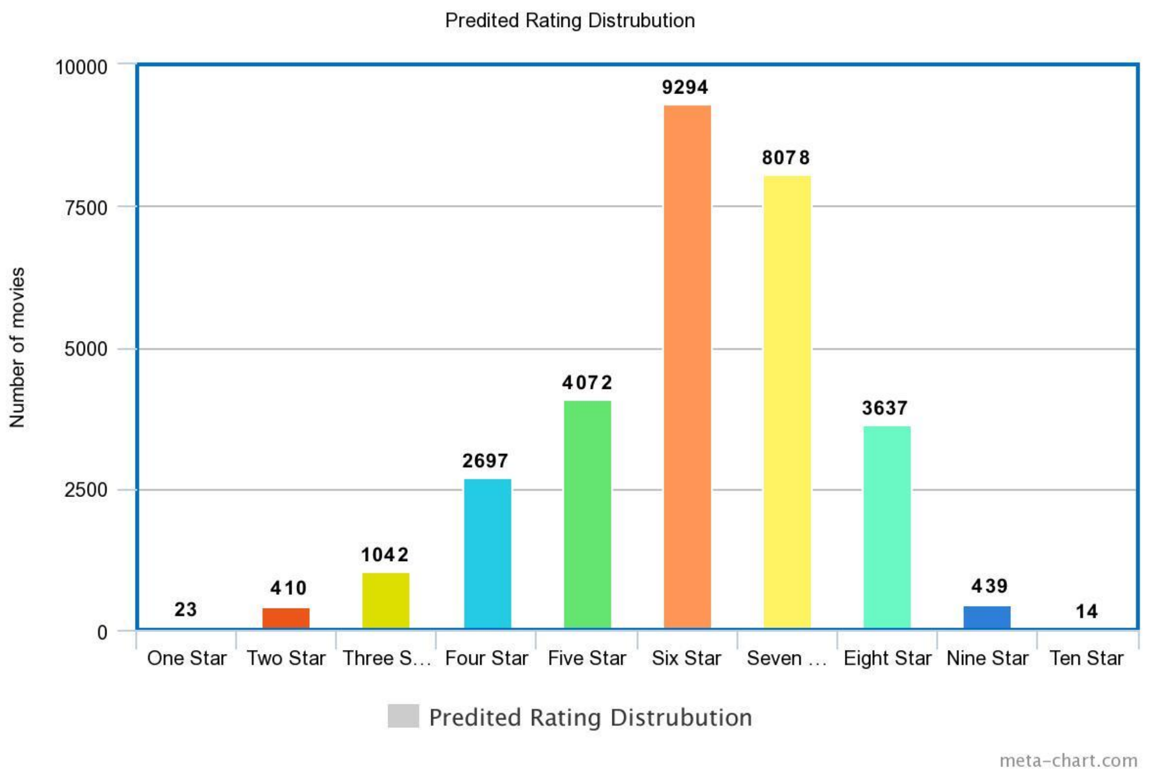
A more useful prediction model evaluation metric would be to measure how close the predicted values are to the actual prediction. Since there are 11 ratings, 0 – 10 inclusive, it' easy to calculate how the predictions are to the actual rating in terms of percentages. If the movie is rating 7, and the prediction is 6, it is 1 star off, the accuracy is considered 90% accurate, is the prediction was 7, that's spot on, so it is 100% accurate. If the movie rating is 7, but the predicted rating is 4, that's 3 stars off, the prediction was 70%. This percentage of closeness provides a good evaluation of how close on average is the model it's prediction. If it's 100% right only 37% of the time, how close where the other prediction.

To measure this, a ***prediction_rating_test.py*** python script was written. There the saved prediction model was loaded from its file, ***movies.csv*** was loaded in as well, all 30'000 movies. They were normalised using the saved Scaler values which were created during model training. Then all the movies were put through the prediction model and each prediction was saved into an array. The actual move rating was also saved into an array. The order of movie rating is the same, the index of movie rating from the movie dataset and index from prediction model are exactly the same. A for loop iterates through both array and compared the star rating difference between actual ratings and predicted rating for all 30'000 movies. The difference was summed up in the "***total***" variable, and divided by the total number of movies. This gives the average of star difference the prediction model predict and actual rating. Each star rating difference was multiplied by 10 to provide a % value for the difference, and then it is minuses from 100. For example, 2 star difference, multiplied by 10 = 20, then 100 – 20 = 80, a 2 star difference is 80% accurate.

This this metric, and evaluating the model again on all 30'000 movies, calculating the average star difference and deducing accuracy percentage, the final result, the actual effective accuracy of the movie prediction model on average is **<u>91%</u>.** This means on average, the model predicts a movie rating with an average of being 1 star off from the actual movie rating.



Above is a bar chart to demonstrate number of movies that were predicted for each star rating. No movies have been predicted 1 to 3 stars, or even 10 stars, the most common prediction is 6 stars, as seemingly is an average. Comparing with the actual movie rating distribution shown below, there is some interesting difference to be discussed.



Even though the movie rating accuracy is about 90%. There is still some mismatch to the actual movie predictions. Why is that? The model seems to not predict movies below 4 stars, when in reality there are movies below 4 stars. The key here is that fact that the content of the movie plays a much greater role in determining the movie rating. The prediction model can only determine the rating from pre-planned movie metadata that be turned into numbers. It cannot account for the complexity of plot and cinematography of movie itself. During training, the neural network would have seen many movie examples of perhaps no budget movies, with cast that don't have many awards, and saw that the movie has a high rating. This influences the prediction model to higher ratings to even the lowest budget movies. The model finds what are the averages rating is movies with a given budget and awards of the cast and genre and runtime of the movie. The model inherently takes the assumption, that even with low budget, the movie would average around 6-star ratings, and this has the assumption the movie would also have an averagely good plot and cinematography and acting.

What is interesting to note, that the model predicted more 4-star movies, than 5-star movies, as opposed to a more evenly stepped distribution we see in actual movie prediction. The possible reason behind this that it would lump all movies of 4 star and below into 1 single category. Hence the rating 4-star kind of becomes a catch-all class for movies with lowest production values. The total of which is higher than movies that are just 5 stars.

The number of movies rated 6-star is very high. It loosely reflects the actual movie rating distribution where most common, movies are 6-star movies. But the model skews to predict more 6-star movies. Likely to the reason stated above, again this highly to do with the quality of the content itself, of which the model only assumes average of this rating. Very high rated movies, with ratings of 8 stars and above, would have similar production values to movies that are 6 or 7 star, and hence more 6-star predictions are made. The production values would have be higher than average, such as higher budget and director and actors with many awards that are all higher than average to predict movies above 6-star.

Despite the fact the model cannot take into account the soul of the movie, it can still correlate the production value to the ratings with above average accuracy.

5.3 Django Web App Unit Testing

The main software development methodology used during the development of this project is Test Driven Development (TDD). The key focus of this methodology is to write the tests first, that fail, then write the minimal code to pass those tests. Writing unit test were key and essential to the development of this project.

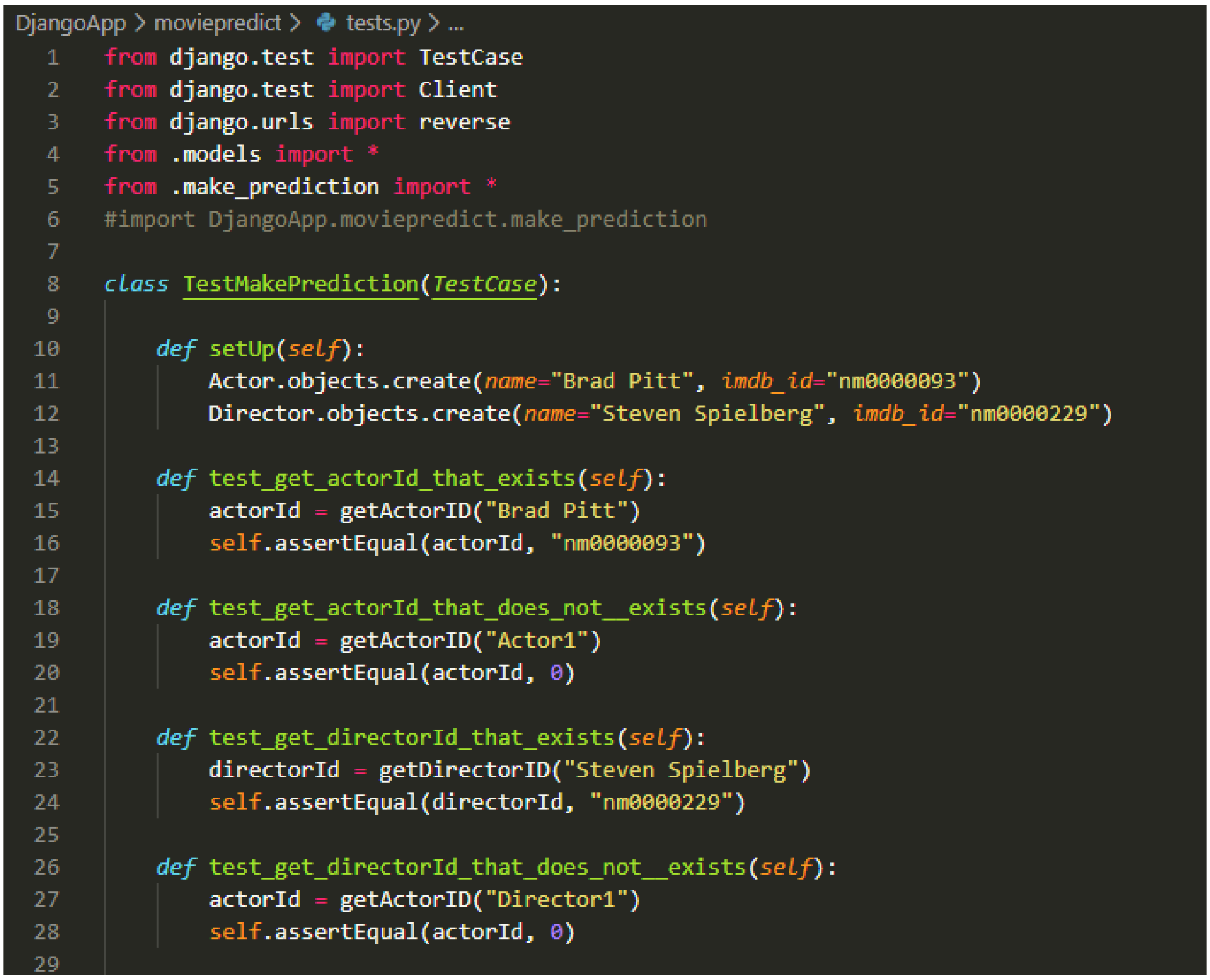
There are no proper ways to unit test a neural network, the test for accuracy and measure was done through a variation of black box testing, where movie data was put into the crated prediction model and results were examined, special measurement metric applied. But for the Django app, white box type of testing was used. This was in the form of unit tests.

Django is a integrated framework that does a love the things in the background already. For instance, to run python unit test on the whole sever, it can be done with a single command,

```
python ./mange.py test
```

This will find and run all test.py files within all the individual installed Django apps on the Django server. There are several important functions used in the web app that need to be tested for. All written function that access the database to retrieve actor data, functions that web scrape for actor awards and even the function that predicts the movie rating is tested.

The Django python unit test use the ***TestCase*** class like any other python unit tests, except it imports it from Django's own python library called "test". Mock database objects are created and movies with known predicted ratings are used to tests these functions to ensure the output is the same as expected output. This was used extensively during the development of the project to ensure all functions in the web app are working as expected.



```python
from django.test import TestCase
from django.test import Client
from django.urls import reverse
from .models import *
from .make_prediction import *
#import DjangoApp.moviepredict.make_prediction

class TestMakePrediction(TestCase):

    def setUp(self):
        Actor.objects.create(name="Brad Pitt", imdb_id="nm0000093")
        Director.objects.create(name="Steven Spielberg", imdb_id="nm0000229")

    def test_get_actorId_that_exists(self):
        actorId = getActorID("Brad Pitt")
        self.assertEqual(actorId, "nm0000093")

    def test_get_actorId_that_does_not__exists(self):
        actorId = getActorID("Actor1")
        self.assertEqual(actorId, 0)

    def test_get_directorId_that_exists(self):
        directorId = getDirectorID("Steven Spielberg")
        self.assertEqual(directorId, "nm0000229")

    def test_get_directorId_that_does_not__exists(self):
        actorId = getActorID("Director1")
        self.assertEqual(actorId, 0)

```

Code snippet of the test.py in "moviepredict" Django web app that tests the web apps functions

5.4 Django Web App User Evaluation

The web application's front end was designed with Nielsen's heuristics in mind. with Nielsen's heuristics are methods to improve software usability [43]. There is a list of 10 things to look out for when making a user interface. Using Nielsen's heuristics is a good approach as increase accessibility to the software's functions and have good comprehension of what is happening with it. It is essential to be adhered to in order to increase the accessibility and usability of the application to a wider audience of people, even the ones who are not tech savvy.

The Neilson's heuristics were taken into account when for example creating the design and layout of the web app pages. Number #8 heuristic is "*Aesthetic and minimalist design*". The web app used as little buttons and clutter as possible. As mentioned earlier, it was a design decision to make it 2 pages and have as little form fields as possible.

Another heuristic which was taken seriously was #5 *"Error prevention"*. This is especially important for form-based web apps. The user could input wrong type of data or leave fields blank which could lead to server errors. Preventing these are important. This is why extensive form validation had to be implemented. In form of using "required" tags in input tags and some JavaScript to ensure at least one genre was selected.

The original plan was to get as many users, with different technical abilities to test out and use my movie rating prediction web application. Unfortunately, due to the recent social-distancing measures enforced for the COVID-19 outbreak, the number of users that could try out the system was limited. 14 users were asked to use the Movie Rating Prediction web app, then afterwards they are asked questions (based around Nielsen's heuristics) and give a feedback rating of 0 – 10. 0 being worst rating, and 10 being the best rating feedback. The ratings were then summed up to total and divided by the number of feedbacks to get the average rating for each feedback.

1. ***How clear was the goal of the web app?*** – 7.8
2. ***How easy was it to use the web app? -*** 9.1
3. ***How clear where the results page of the web app?*** – 8.7
4. ***Rate the overall layout and design of the web app. –*** 9.6
5. ***How likely are you to use this web app again? –*** 7.4

5.5 Conclusions

This chapter discussed the evaluation of the prediction model, how it was tested for accuracy, what metric were used to determine that accuracy. More importantly discussed what the results means and why the model was predicting rating in what it did, and what that means for the usability of the prediction model. This chapter discussed what type of code testing was used during Test Driven Development to ensure the proper functionality of code logic in the Django server. Lastly, this chapter discussed how the web app was evaluated by user to ensure the best user experience and what kind of methods were employed to improve and optimise the usability of the web app. The next chapter will discuss the final conclusions of the project and possible future work in developing this project further.

6. Conclusions and Future Work

6.1. Introduction

Last few chapters discussed the design, development and evaluation of the project. Discussed the summary overview of the project’s architecture, in depth analysis of the code logic that makes everything work and finally how the project code was tested and what metrics were used to evaluate the accuracy of the movie ratings prediction model. This chapter will go over the conclusions about this project, and a general summary of the taken approach to this project.

6.2. Conclusions

In the literature review chapter, we have seen that predicting movie rating using data mining and machine learning is in fact an important problem to movie producers. It’s being tackled by several companies already to create movie prediction software’s. This shows how important data mining is to industry and how to manages to solve very complicated problems with machine leaning techniques like the use neural networks.

From reviewed researcher papers, even they did not use artificial neural networks to make movie rating predictions but used other prediction models like support vector machines (SVMs), they could correlate and predict movie rating based off movie production values. This project taken a more unique and specific approach in that it utilised an artificial neural network to make a prediction model. Other projects who attempted to make movie prediction did not take into account the actors who play in the movie nor the director, and here is where this project is most different from other similar projects. Due to using more data variables, and more determining factors to movie rating such as the vital role the actors have in making movies that are good and liked by the audiences, the prediction model produced by this project is more accurate in predicting movie ratings than any other project before.

In the design chapter we discussed the over all the design the project. Discussed the type of software methodologies that were used during the development of this project. Test Driven Development in junction with Agile proved to be very efficient in developing the code and functionality of this project. Along with feature driven development, the focus on implement working features individually with the support of test-driven development proved to be a great combination to use. Features would be planned, then the tests for those features would be planned out. After all testing scenarios have been written up, the feature would be implemented. Then taking a step back to see what good next steps would be to take in the overall development of this project. This broke down hard problems into smaller and easier to manage problems, making the development of this whole project very feasible.

The decision choices made for the neural network made an accurate prediction model, perhaps more so than other movie rating prediction models. The use of actor and director awards provides value insight into what sort of quality the movie would be. Previous project used methods like support vector machines and used limited amount of production variables, which would make linear prediction on the movie rating, unlike an artificial neural network which could look at each feature of the movie production values and correlate then to each other production value and provide a more balanced prediction. Neural network is a better approach to this problem, and as seen from literature review, the movie industry and companies producing movie rating prediction think so too.

The making of a web application is the best way to create a user interface that could interact with the model and play around with it seeing what combinations of actors, budget, and genres would give the highest rating prediction. Using Django server to manage the logic and provide the with web pages visually displaying the results is much easier to implement than making a software user interface from scratch. Using the available frameworks such as Django and Bootstrap, very clean and easy to use interfaces can be developed and very easy integration of code logic and access to the database back-end. On top of this, this same web app could be hosted online, giving easy access to millions of people. The web application approach to making user friendly interface to the movie rating prediction model is better than the offline software approach.

In the development chapter we discussed the entire development process and inner working of the project in great detail. The approach to gather data for make a dataset set was foundational. IMDb provides a ton of movie and actor information, and it even allows everyone to use and web scrape their website as long is it not for commercial purposes (otherwise would have to ask for licence permission). Using multiple data sources to start with and merging all available data sources created a large movie training dataset with lot of production variable that could provide key insights the correlation between movie rating and the production values of the movies, allowing for future predictions.

The neural network used the approach of speaking each star rating into a sperate class and then using multiclassification approach in predicting. This proved to have more accuracy than the prediction of rating as a single numerical value. This creates a generalised prediction that is more accurate in approximation than trying to predict very specific values which would generate a larger of loss error values, throwing off the accuracy of the model. The number of middle layers, activation functions and even the type of optimizers used have been chosen to fit the overall neural network architecture and provide the most accurate movie prediction results.

To make the web application for the movie prediction model, Django was the best choice. The neural network training and models have been made with python already due to python being a very flexible language and great libraries to be used, and using Django created seamless connection with the prediction model. A lot of the functions could be reused from the training scripts in the Django server, such for example the data scaling and normalization code. Using the prediction model with Keras python library required no extra effort to manage any compatibility issues between the two frameworks.

On top of this, Django is very flexible framework. It allows easy integration with other frameworks and technologies like Bootstrap and the integration of SQLite database libraries. This meant more focus and effort could be utilised on development the web app logic without worrying about the technical details of under the hood operation of these frameworks.

In the testing and evaluation chapter, we discussed how the neural network was tested and evaluated for the accuracy. One of the main ways to determine the accuracy was using the terminal accuracy number report during neural network training at runtime. This provide instant feedback as to how well the model was doing. Any changes in the neural network could quickly be tested by running the training script and seeing those accuracy numbers.

But those numbers were not fully representable of how accurate the model really was. As discussed in the chapter, all those numbers meant was that the model was exactly right 37% of the time but did not take into account of predictions that were close to the right answer. In order to test and evaluate the model to determine how accurate was the prediction model in terms of closeness a testing script was written to calculate the average star rating difference between predicted rating and actual rating. With this new metric evaluation, the prediction model was evaluated to be 91% accurate. Meaning on average, the prediction was only 1 star rating off from the actual star rating of the movie.

Despite the absence of direct quantifiable data about the plot and cinematography, the prediction model still provides an valuable prediction. The distribution of movies rated for each star class might seem unbalanced at first when compared to the actual movie rating distribution, but as discussed, this result is expected when the soul of the movie is not taken into account. Low budget movies can be great and high budget movies can be a total flop. These factors would push a lot of prediction take on some average value. This could be possibly be improved upon if more data is used to train the model, pedicular data about movie plot and themes used in the movie.

To evaluate the user experience of the web application, Nielsen's heuristics were used to ensure the best possible usability of the web application. Each of Nielsen's heuristics were kept in mind when developing the web pages on the web app, such as having form validation with understand errors, and helpful text to explain what was expected of the user and an uncluttered concise layouts that were user friendly. And finally, to actually make sure the usability of the website was as comfortable to use as expected, a user servery was carried out. 14 people were asked to use the web app and then asked 5 question were based on some heuristic evaluations and asked to give a rating out of 10. The resulting average were high, indicating the user experience was great for most people, thanks to the utilisation of Nielsen's heuristics as guidelines in the development of the movie rating prediction web application.

6.3. Future Work

There can be a lot more developed for future work for this project.

There can be another prediction model be made that predicts movie box office incomes. This will be a whole separate neural network and prediction model, but it will still use the same production values. Each neural network could incorporate more production value data, for example it could take into account the month the movie would be released. For each movie in the past, data would have gathered to see what other movies where released during the month of it's release, and their production values to be taken into consideration for the neural network. This alone would add a layer of complexity on top of the current system.

More online movie data source could be used to make prediction. Rotten Tomatoes ratings could be used to compare with IMDb ratings, along with other possible movie datasets and online sources to be data mined.

One massive possible off-spin for this project could add movie description as an input. This perhaps would require a whole new type of neural network architecture of multiple networks merged together somewhere. The key here would be the language and word analysis of the movie descriptions used in IMDb websites and seeing how well the movie did. This would provide the insight into the plot lines of the movie and the type of content the movie has. The neural network could analyse with plot themes have what kind of ratings, so then on the web app the users could write their plot summary for their movie and use that to generate predictions.

On the Web app server side, future work would include expanding and using a larger offline database that includes the awards and nominations won for each actor. Could also include a whole movie database, with updates every week to add new movies and actors to the database and use it to retrain the model to always have an up to date prediction model. This would speed up the processing time, as it the server would not have to web scrape for any additional production value metadata about the movie.

There could be more handy features added to the form part of the web app page, such as a drop-down menu of Actors from the database to autofill the field, with pictures of the actor too. Include more visual metrics. And ofc If going too add more prediction like movie box office income and such, it's going to need a lot more info fields for the web pages.

Bibliography

[1] Storytellingday.net. (2019). History Of Storytelling – How Did Storytelling Begin?. [online] Available at: http://www.storytellingday.net/history-of-storytelling-how-did-storytelling.html

[2] Miller, L. (2019). The oldest story ever written. [online] Salon. Available at: https://www.salon.com/2007/04/24/gilgamesh/

[3] Mark, J. (2019). The Pyramid Texts: Guide to the Afterlife. [online] Ancient History Encyclopaedia. Available at: https://www.ancient.eu/article/148/the-pyramid-texts-guide-to-the-afterlife/

[4] Vincent, J. (2019). Hollywood is quietly using AI to help decide which movies to make. [online] The Verge. Available at: https://www.theverge.com/2019/5/28/18637135/hollywood-ai-film-decision-script-analysis-data-machine-learning

[5] Cinelytic.com. (2019). Cinelytic | Built for a Better Film Business. [online] Available at: https://www.cinelytic.com

[6] Kay, J. (2019). How data company Cinelytic aims to reduce risk in the film business. [online] Screen. Available at: https://www.screendaily.com/features/how-data-company-cinelytic-aims-to-reduce-risk-in-the-film-business/5136245.article

[7] En.wikipedia.org. (2019). VaultML. [online] Available at: https://en.wikipedia.org/wiki/VaultML

[8] Vault-ai.com. (2019). RealDemand™ Market Intelligence. [online] Available at: https://www.vault-ai.com/RealDemand-market-intelligence.html

[9] ScriptBook. (2019). ScriptBook. [online] Available at: https://www.scriptbook.io

[10] ScriptBook. (2019). ScriptBook. [online] Available at: https://www.scriptbook.io/#!/deepstory

[11] TensorFlow. (2019). TensorFlow. [online] Available at: https://www.tensorflow.org/

[12] Keras.io. (2019). Home - Keras Documentation. [online] Available at: https://keras.io/

[13] Zublenko, E., 2020. Why Django Is The Best Web Framework For Your Project. [online] Steelkiwi.com. Available at: https://steelkiwi.com/blog/why-django-best-web-framework-your-project/

[14] Kumar, Saurabh. (2019). Movie Success Prediction using Data Mining For Data Mining and Business Intelligence (ITA5007) of Master of Computer Application School Of Information Technology and Engineering. [online] Available at: https://www.researchgate.net/publication/332396741_Movie_Success_Prediction_using_Data_Mining_For_Data_Mining_and_Business_IntelligenceITA5007_of_Master_of_Computer_Application_School_Of_Information_Technology_and_Engineering

[15] Indexive.com. 2020. Movie Ratingprediction With Machine Learning Algorithmson IMDB Data Set. [online] Available at: http://indexive.com/uploads/papers/icatces2018-51.pdf

[16] Stackify. 2020. What Is Agile Methodology? Tools, Best Practices & More. [online] Available at: https://stackify.com/agile-methodology/

[17] Productplan.com. 2020. What Is Feature Driven Development (FDD)? | Definition. [online] Available at: https://www.productplan.com/glossary/feature-driven-development/

[18] Agile Alliance. 2020. What Is Test Driven Development (TDD)? [online] Available at: https://www.agilealliance.org/glossary/tdd

[19] En.wikipedia.org. 2020. List Of Film Awards. [online] Available at: https://en.wikipedia.org/wiki/List_of_film_awards

[20] IMDb. 2020. Leonardo Dicaprio - Imdb. [online] Available at: https://www.imdb.com/name/nm0000138/awards?ref_=nm_awd

[21] Stephanerappeneau. (2017). 350 000+ movies from themoviedb.org. [online] Available at: https://www.kaggle.com/stephanerappeneau/350-000-movies-from-themoviedborg

[22] IMDb. (2019). IMDb Datasets. [online] Available at: https://www.imdb.com/interfaces/.

[23] Crummy.com. 2020. Beautiful Soup Documentation — Beautiful Soup 4.9.0 Documentation. [online] Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[24] DelSole, M., 2020. What Is One Hot Encoding And How To Do It. [online] Medium. Available at: https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179

[25] Jaitley, U., 2020. Why Data Normalization Is Necessary For Machine Learning Models. [online] Medium. Available at: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

[26] Scikit-learn.org. 2020. Sklearn.Preprocessing.Standardscaler — Scikit-Learn 0.22.2 Documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[27] Scikit-learn.org. 2020. Sklearn.Preprocessing.Minmaxscaler — Scikit-Learn 0.22.2 Documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[28] Keras.io. 2020. Home - Keras Documentation. [online] Available at: https://keras.io/

[29] Santos, L., 2020. Dropout Layer · Artificial Inteligence. [online] Leonardoaraujosantos.gitbooks.io. Available at: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/dropout_layer.html

[30] En.wikipedia.org. 2020. Sigmoid Function. [online] Available at: https://en.wikipedia.org/wiki/Sigmoid_function

[31] Adventures in Machine Learning. (2018). The vanishing gradient problem and ReLUs - a TensorFlow investigation - Adventures in Machine Learning. [online] Available at: https://adventuresinmachinelearning.com/vanishing-gradient-problem-tensorflow/

[32] Danqing Liu (2017). A Practical Guide to ReLU. Medium. [online] 30 Nov. Available at: https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

[33] Uniqtech (2018). Understand the Softmax Function in Minutes - Data Science Bootcamp - Medium. [online] Medium. Available at: https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

[34] Wikipedia Contributors (2019). Mean squared error. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Mean_squared_error

[35] Github.io. (2018). Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. [online] Available at: https://gombru.github.io/2018/05/23/cross_entropy_loss/

[36] Jason Brownlee (2013). Gradient Descent For Machine Learning. [online] Available at: https://machinelearningmastery.com/gradient-descent-for-machine-learning/

[37] Srinivasan, A.V. (2019). Stochastic Gradient Descent — Clearly Explained !! - Towards Data Science. [online] Medium. Available at: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

[38] Jason Brownlee (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. [online] Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[39] Md Akhtar Hossain (2019). 20 Stunning Free Bootstrap Form Templates 2019 - Colorlib. [online] Colorlib. Available at: https://colorlib.com/wp/bootstrap-form-templates/

[40] Careerfoundry.com. (2016). What is Bootstrap: A Beginners Guide. [online] Available at: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/

[41] Wikipedia Contributors (2019). SQLite. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/SQLite

[42] Python.org. (2019). 11.13. sqlite3 — DB-API 2.0 interface for SQLite databases — Python 2.7.17 documentation. [online] Available at: https://docs.python.org/2/library/sqlite3.html

[43] Nielsen Norman Group. (2017). 10 Heuristics for User Interface Design: Article by Jakob Nielsen. [online] Available at: https://www.nngroup.com/articles/ten-usability-heuristics/