Wine Data Predictive Analysis
Final Year Project Report

DT211
BSc in Computer Science Infrastructure

David O'Neill
Bryan Duggan

School of Computing
Dublin Institute of Technology

03/04/2019

Abstract

The area of study is data analysis, deep learning and restful web application development, along with studying the worldwide wine industry and studying how wine features are affected and how that knowledge can be used alongside data analysis and deep learning to produce an intuitive user-friendly product.

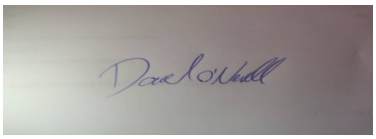
The problem area is that there are not many applications that can improve the job of a wine taster/reviewer. This application comes to the forefront of this issue and solves that problem. Studies show that wine ratings can be predicted to help the end user in their day to day jobs to improve efficiency and be more accurate.

The problem will be tackled using data analysis techniques, deep learning technologies and flask web application development to produce a product which provides a service to help the end user. The solution is implemented using these technologies as well as cloud servers to create a fully functioning solution. The outcome results that will allow users to avail of deep learning to predict ratings of their wines with the help of natural language processing.

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:



David O'Neill

03/04/2019

Acknowledgements

I would like to thank my supervisor Bryan Duggan for all of his help throughout this project. He helped me to solve problems when I needed help and he also got me to think about problems in different ways and to gain an understanding of the subject that this document covers. His help is greatly appreciated.

Also, I would like to thank Lucy Griffin for giving me a better insight into the world of wine tasting and production.

I would also like to thank everyone who took part in testing the application and thank you to my friends who helped me with challenges I faced and kept me motivated to complete this project.

Table of Contents

TABLE OF FIGURES

1. Introduction

This project of Wine Data Analysis through Wide and Deep Learning is to try and see what the data of wines can tell us and how different features of different wines can affect its variables and how reviews can predict what a wine's rating could be. The project researches and uses Natural Language Processing along with Machine Learning to predict a wine rating from the review given by tasters. This prediction data is available through a web application where wine enthusiasts or people in the hospitality industry can find the information they need and also give their reviews and keep track of wines they use and taste. The goal for the project is to have completed two main parts which include, the predictive analysis and the web application.

1.1 Overview & Background

'Data Analysis is the process of systematically applying statistical and logical techniques to describe and illustrate, condense and recap, and evaluate data.' [1] Data analysis is used every day throughout the world. It is used in major businesses and research fields and is becoming a significant aspect of science. Data analysis plays a significant role throughout this project as it helps us to analyse and visualise why certain aspects are the way they are, and it also aids us in other aspects of the projects. Data and data analysis drive a lot of what goes on day to day in the world, it is a valuable resource, hence why it has been previously referred to as 'the new oil.' [2]

Deep Learning is extracting useful patterns from data in an automated way with as little human involvement as possible. It does this with the optimization of neural networks by using libraries such as Python [3] and TensorFlow [4]. The hardest part of applying deep learning is asking good questions about the topic and problem that needs a solution, as well as finding good data sources. Deep Learning is a rapidly growing area of machine learning, it is relatively new, and it is growing fast. New technologies are emerging every day, and new ways to apply deep learning techniques are developing rapidly.

The above lead to numerous areas where the use of data assists projects. In this project, we dive into the analysis of data and the use of deep learning in the world of wine reviews and ratings. Using these techniques along with growing technologies the project shows critical aspects of how the wine industry could use data and deep learning.

1.2 Project Objectives

The objective at hand is to have a functioning web application that will take a user's input into the web application and run the data they input through the created deep learning algorithm. The outcome is that the user gets a predicted wine rating from 0-100 after they give their wine description/review, the variety of grape and the country. There are a lot more sub-objectives that also have to be completed as the gathering of data, the cleaning and analysis of the data so that useful insights can be found and visualised from that data, and the data can then be used to implement the wide & deep neural network from which the rating prediction comes from. Other sub-objectives include a section on the application that allows a user to create an account on the application and then review their wines and share their wine reviews with others on the site. The application has to be clean and user-friendly, have

a minimal design that improves functionality and user experience. Wine reviews then saved to a database where they are pulled from the database and displayed on the application. Lastly the ability to view several wineries on a map on the application would be beneficial to give the user an idea of how vast the world of wine is.

1.3 Project Challenges

A significant challenge of this project is finding a dataset that is large enough to be beneficial to a neural network — the more data, the better when designing neural networks. The data must include features that will directly and significantly influence the deep learning algorithm to give the best possible result — the latter results in much time being used to study the data and different data analysis techniques.
Another challenge of this project is the implementation of a wide & deep neural network. It is a new category of deep learning, being researched and developed by TensorFlow in 2016. Firstly, the author must have achieved an understanding of deep learning and from there an understanding of how to concatenate two neural networks and how they work together and what is beneficial about the concatenation of the networks to achieve the desired result from the wide & deep algorithm. The Keras library has a functional API that allows us to implement this form of deep learning which helps overcome the challenge.
One more challenge is finding a way to implement the model and having it communicate with the application to take data from the user's inputs and return them a prediction. The latter is a challenge as it is something the author has never done before but is keen to figure out how to achieve the final goal, along with hosting the application to Amazon web services. It is something exciting to learn and implement.

1.4 Structure of the Document

Section 1 – Introduction

This introduction of the report acts as a preface that gives a high-level overview of the main aspects of the project, the objectives, challenges, and structure of the document. It helps the reader to understand the direction the project is taking.

Section 2 – Research

The research section of the report goes through the relevant research conducted throughout the project lifecycle from beginning to end. It consists of background research which goes into specific machine learning techniques, data analysis and how these techniques become used in the context of wine. This section also covers some history and relevant information on wine briefly. Next, the primary research consists of a domain expert interview which is where the report delves deep into the factors of what makes a great wine as well as other aspects of the production process of wine and what influences the flavour, price, and ratings as well as some information on the wine lifecycle. All the presented information is from an interview with a domain expert in the wine tasting and production industry. This section leads us into the

existing solutions section and then to the technologies researched section where debated, and decided technologies were chosen to be used or not.

Section 3 – Design

This design section of the document takes a look at the design of the overall system and the design choices of the application. It takes a look at the database design and its tables. It also goes through some early on prototype ideas and designs and discusses how these designs and ideas developed throughout the project's lifecycle along with aspects of the project that did not make it to final production. The design section also includes some user scenarios and use cases and a look at source code layout.

Section 4 – Architecture and Development

The architecture and development section of the document looks at the architecture of the system how the systems created along with the details of how the developments implemented and a look at the development itself. There is a diagram of the architecture, and it discusses how each component of the system works together and why each component is necessary for the system to succeed. This section also explains the explanation of code components in the project.

Section 5 – System Validation

The system validation section of the document is where the report takes a look at various testing that has taken place to ensure the application and its underlying features all work successfully and make sense in the overall scheme of the project. This section consists of unit tests of the software code, user tests to test the usability of the application, a domain expert test, and review and statistical analysis of the deep learning model.

Section 6 – Project Plan

The project plan section discusses the plan of the project from the interim submission and report, the original Gantt chart and the comparisons of the original plan to how the plan changed throughout the lifecycle of the project. This section will then also go into the plans for the project after completion and what direction parts of the project could take and features that did not make final production but will become implemented in the future.

Section 7 – Conclusion

The conclusion section takes a look at the project from start to finish; It goes through how each component performs. It discusses any changes that could happen in the future, the unincluded aspects and the reasoning behind them, and an overall look at what went well in the project as well as the analysis of components that did not meet the desired standards.

Section 8 – Appendixes

This section will include a source to a video demonstration of the application.

Section 9 – Bibliography

This section contains all of the references used throughout the research of the project and the report.

2.0 Research

2.1 Background Research

This project is a project in which the author plans to predict wine varieties and prices through methods of machine learning and natural language processing and predictive analytics. Predictive Analytics is the art of building using models that make predictions based on patterns extracted from historical data.[5] Just as the analysis of grapes, must and wines is a regular and vital part of the winemaking process,[6] the analysis of data is the most crucial part of the data analytics process. To go deep into the subject matter and adequately understand all of the details and every aspect of the project the author must complete research. The history of wine, what makes certain wines taste different from other wines? How long is the process? What are the main factors that affect the final product of a wine? All of these questions needed answers along with the interview of a domain expert with extensive knowledge in this field.

The earliest evidence of wine grape-based fermented drinks was found in China as long ago as 7000BC. From there the rest of the world followed and discovered wines. The oldest wine production facility is almost 6100 years old, and it is the Areni-1 winery in Armenia. It is still

used daily, now more than ever. A bottle of wine can sell from as low as a few cents, to thousands if not millions of euro.[7]

Many factors play a part in the pricing of wine, and we look at that in the next section with the interview of the domain expert. First, we discuss other attempts at similar studies as a part of the completed research. One study was Predicting Wine Points using sentiment analysis. [8] (Wine points also mean the rating of the wine, usually from 0 – 100). In this research, the authors used sentimental analysis to predict the rating of wine. They concluded that sentimental analysis was an excellent way to try to predict the rating of a wine, they used logistical regression, and the authors said the outcome of the classifier done reasonably well. Another study titled, 'Wineinformatics: A Quantitative Analysis of Wine Reviewers.' [9] This study uses a White-Box classification algorithm: Naïve Bayes and Black-Box classification algorithm: Support Vector Machine and receive almost 87% accuracy when evaluated with the SVM method. There is also studies and experiments completed in 'Machine Learning In Python: Essential Techniques For Predictive Analysis' [10] that predict how a wine tastes depending on its acidic and sugar levels.

The data in this project is from Winemag.com [11], A wine review site where there are over 150,000 wine reviews which also include useful features for this project. The author acquired the dataset through Kaggle [5], a data science website. It has everything needed to proceed with the project. A sample of the raw data is:

country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery
Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos
US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm
US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

Figure 1: Raw Data Sample

As you may notice from the first few lines of data, the data needs to be cleaned and manipulated to suit the project. A data cleansing process has to take place and the data manipulated so that some columns change, new columns added and null values dealt with among other quality issue tasks.

2. 2 PRIMARY RESEARCH

2.2.1 DOMAIN EXPERT INTERVIEW

Throughout the research of this report, to gain knowledge on different factors and to find out the details to understand what provokes wine prices to fluctuate and what causes the wine to taste the way that it tastes, an interview with a domain expert in the field of study and research had to be carried out. The information below has gathered from an interview with Lucy Griffin. Lucy is a professional wine taster. Lucy has been in the industry for almost ten years and has lived in Bordeaux on a winery as part of her training. She has boundless knowledge in this domain, and it was a pleasure to interview her. The interview gave an in-depth analysis of many things that help to improve this project on wine review analysis. Below is the information gained from the interview.

There are many questions to ask yourself when researching a predictive model on wine such as what factors affect the price? Why is one wine more expensive than the other? Why do people pay more for grand cru classé Bordeaux than they do for an Italian Pinot Grigio?

To start, the following things have to be taken into account. The price or value of the land where the grapes are grown plays an impact on the pricing of different wines. For example, The average price of land for one hectare of grand cru vineyard in Burgundy is over €5 million and can be up to €10 million and more for the really renowned vineyards like Romanée-Conti. To put that into perspective, the max yield defined by the Burgundian appellation is about 35hl/ha – that's a maximum of 5000 bottles of wine.

The cost of labour – varies from country to country. There is a lot of articles about modern day slavery in South African vineyards (There is a documentary about this called Bitter Grapes.[13]) and many Irish and UK retailers now demand fair-trade wines hence prices are higher. How manual the grape growing/harvest is also comes into play. It is cheaper to mechanically harvest but that's not always possible. Some wine regions forbid mechanical processes and insist on a manual harvest. The grand cru calssé vineyards on the left bank in Bordeaux are all harvested by hand. During the interview Lucy stated that when she was at Ch. Mouton Rothschild in the Medoc, she was told that the Chateau employs an amount of 500 people over 4 weeks to harvest their grapes by hand. Some vineyards - along the Mosel in Germany for example are so steep they are not accessible by any machinery so all work is done by hand - ploughing, leaf plucking to encourage ripening, harvest, pruning. There are also wines that have to be hand harvested because of the winemaking style - sweet wines and wines where the grapes are dried or partially dried like Amarone Della Valpolicella have to be hand harvested.

Max yields are defined by the appellations laws. Each region has their own max yields. The best winemakers often come in good bit lower than the max but they can't produce any wine over the max for their appellation so even if they have a really great harvest in a good year, they can't sell the extra wine under their appellation. As it's limited quantity, it can affect the cost.

Another factor to consider is Trends - One of the best examples of this is Chateauneuf-du-pape. It always sells for a premium and people in Ireland know it well and love it. There are smaller lesser known areas that are right next to Chateauneuf-du-pape (like Lirac or Vacqueyras) that have identical terroir and weather and grow the same grapes and make wine in the same style but you can get it at a fraction the price. Along with this is, How easy or difficult the grapes are to grow - Pinot Noir is a fussy thin skinned grape. It is susceptible to rot and disease. It needs near perfect conditions to grow - this is usually more expensive although Chile have started growing Pinot Noir and can make it relatively inexpensively thanks to their dry moderate climate in the central valley areas.

Another factor to consider is a big one. It is the Weather/Vintage - Great Vintages cost a premium. It is most prevalent in Bordeaux where good vintages can be aged for longer and increase in price with time. Weather is one of the prominent factors on how a wine can vary from year to year - it is mainly down to how ripe the grapes get or if there's an event which causes a low harvest. The frosts across France last year and this year would have pushed up prices a lot especially across Burgundy and Chablis. The hot summer and drought this year meant grapes were getting "sugar ripe" without phenolic ripeness - this is where the grape produces lots of sugar but the phenolics in the skin which contributes most of the flavour isn't ripe. It results in high alcohol wines without good flavour.

Next factor to discuss is the Aging - not all wine is aged, some is kept in stainless steel tanks until filtration and bottling. Other is aged in stainless steel and oak chips can be added to give it some flavour. Other wine is aged in either French or American oak barrels. The size and origin of the barrel depends on the wine and what complexities the winemaker wants to add to their wine. A bigger barrel is used to add more oxidative flavour to the wine and little oak influence. A smaller barrel is to add oak flavours . A French barrel is usually 225L and costs upward of €700 per barrel. Some winemakers opt for second hand barrels from neighbouring chateaux or age a portion of their wine in one or two year old barrels. Along with Aging, Irrigation is forbidden in most of Europe but widely used in the New World.

The trade position or how many intermediaries are involved play a big part in pricing. Burgundy is really fragmented - a grower might own 2 rows of vines and sell their grapes to a winemaker (possibly through a 3rd party). These negotiates (importer, exporter of wines) blend, bottle, age and sell the wine under their own name. Whereas in other regions, the grape grower is the winemaker and bottler as well. Along with, legislation around use of pesticides/fungicides and if the wine is organic etc - grapes are susceptible to rot/disease. The age of the vineyard also comes into play, older vines generally make better quality wine which is more concentrated, but are also lower yielding.

Supply and demand is another factor to consider, and Champagne is an excellent example of how supply and demand affect the price. Alongside this is where the wines bottled – not all wine travels in bottles. Wine is increasingly bulk shipped in large bladders in freight containers and bottled in the UK and Germany for the Irish market. The Winemaking process is a good pricing factor too - sweet wine like Sauternes/Ice Wines made by concentrating the grape

juice so that there is so much sugar in the juice that the yeast dies during fermentation and the wines left with some residual sweetness. It uses two different methods, but the result is that more grapes make less wine which makes it more expensive. The same said for Amarone wine in Italy. It is a dry style, but the bunches of grapes are left out to dry before fermentation. It makes a concentrated wine, but the volumes are much less than if the grapes weren't allowed to raisin first. Also, what's rare is expensive. There is a couple of vineyards in Europe which can claim they are pre-phylloxera (they are mostly in the sandy soils of Northern Spain). The oldest vines in Europe.

When it comes to predicting wine prices on let's say, a yearly basis, you have to consider why are some vintages more expensive than other and what economic factors make the wine price go up and down? The following should be taken into account when examining why prices may fluctuate:

1. Weather – is the biggest variable from year to year. Wine seems really 'glam' but most are regular fruit farmers for 360 days of the year and winemakers for five and just like the poor grain harvest is affecting every agricultural commodity in Ireland at the minute and driving up prices from butter to deadweight chickens, the poor grape harvest this Autumn in the Northern Hemisphere is going to put wine prices through the roof.
2. Cost of Diesel and Crude Oil prices - the shipping has to be taken in to account. This will also have a bearing on the cost at cellar door as the farmer will use diesel to plough, tend the land etc.
3. Cost of Glass (Bottle), Cork/ Metal (Closure), Paper (Label)
4. Changes in cost of labour

All of the above is what plays a factor in wine prices changing. This interview was invaluable to the research process of this project and it really shows the level of detail you have to consider when working with wine features.

2.3 ALTERNATE EXISTING SOLUTIONS

2.3.1 Vivino

Throughout the research stage, two applications came to light that was found to reflect similarly to what this project plans to achieve. The first is 'Vivino' [14] Vivino is an application on the google play store that holds a database of wines and their prices and reviews and ratings. The application lets users view these reviews and also produce their review as well as creating their own 'wine cellar' in the application. This application is similar to what this projects final product may tend to look like plus the added sections that include the machine learning algorithm that runs against user input data. Below is what the application looks like:

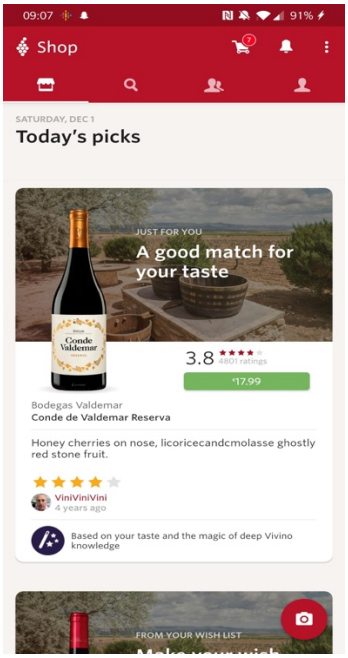


Figure 2: Vivino Homepage

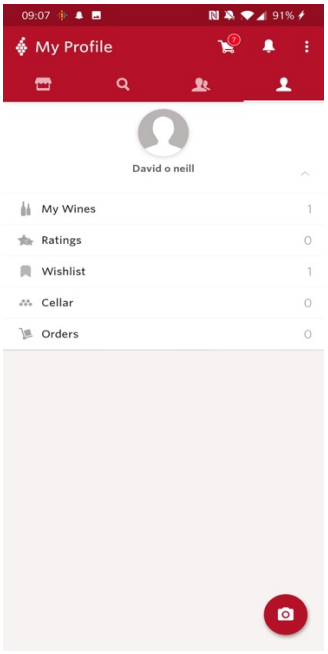


Figure 3: Vivino Userpage

2.3.2 MINTEC

The second existing technology is a tool called MINTEC. [15] MINTEC is an advanced analytical tools that's widely used in the wine industry. The tool tracks wine prices for the year ahead and each year following depending on the cost of different commodities. This tools not just used in the wine industry, it is used in all types of businesses where analytics is needed. It follows a similar structure to how this project is structured, regarding the input and output.

2.4 Data Science Technologies Researched

Below are the steps involved in the data science section of this project. The data process, the model used and how it's implemented into the web application.

Gather Data:
The data is gathered from an online data repository. The data was originally going to be scraped from multiple sources, the final dataset was found through 'kaggle.com', a data science website with hundreds of datasets available for use.

Clean Data:
The ensuing action in the process is to clean the data. Cleaning the data along with feature engineering prepares the data for use in the model. This process requires the use of appropriate technologies.

Model:
Once the model has been created, and the results returned. The models saved as a .h5 file, a file used for saving deep learning models. The .h5s saved alongside the tokenizer that is used to tokenize the user's input data.

Access to the Model:
The model then has to be implemented through an API where the user can input data, run it through the model and receive a value from the model.

2.5 Technologies Researched

2.5.1 Python

Python is a simple language that people pick up quickly quite often enough as a hobby or for work to automate tasks. It has excellent libraries for building web apps. It’s great for handling HTTP, especially with its libraries such as Flask and Django. It scales well and used on websites such as Nasa and Reddit. Used heavily for scientific computing. It’s libraries NumPy and SciPy are great examples of this. It has libraries that support sound, mouse and keyboard interactions such as PyGame, which is excellent for Creating games.

Python is great for data analysis and machine learning. It comes with libraries such as Pandas, SciKitLearn, Seaborn and MatPlotLib. These libraries are great for implementing numerous machine learning models, plotting and visualizing data, as well as manipulating datasets for running data analysis. It is also excellent for Natural Language Processing. It uses the library NLTK.

2.5.2 R

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control. [16]

2.5.3 Java

The Java™ Programming Language is a general-purpose, concurrent, strongly typed, class-based object-oriented language. It is normally compiled to the bytecode instruction set and binary format defined in the Java Virtual Machine Specification. [17]

Java can be used for machine learning, although it is not as popular as Python. Some libraries used for Machine Learning with Java include ELKI, DeepLearning4j, JSAT and MALLET.

2.5.4 JavaScript

JavaScript is one of the three core languages of websites, It adds behaviour and interactivity to a website. JavaScript is classed as a scripting language so it doesn't have the same features such as C++ or Java.

2.5.6 React Native

React Native is a framework which uses a mixture of React and JavaScript which gives you a native iOS and Android application. It was created by Facebook and instead of targeting the browser it targets mobile platforms.

2.5.7 MongoDB

MongoDB is a NoSQL database. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use.[18]

2.5.8 MySQL

MySQL is an open-sourced database management system developed and supported by Oracle. MySQL databases are relational. It stores data is separate tables rather than one large table. It is very fast, reliable, scalable and easy to use.

The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). [19]

2.5.9 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system.[20] It has a strong reputation for its proven architecture, reliability and data integrity. It is capable of efficiently handling multiple tasks at once, otherwise known as concurrency.

2.5.10 SQLite

SQLite is a self-contained, file-based relational database. It is a “serverless” database. It’s lightweight, has a small footprint, is transferable and it is all stored in one file which makes it a great choice for testing and development purposes. SQLite will be used in the development and testing stages throughout the project, but it is not adequate for production. It has limited concurrency, no user management and lacks adequate security. For these reasons it will only be used for the development and testing and will then be migrated to a PostgreSQL instance.

2.5.11 NodeJS

NodeJS is an open-source, cross platform environment for executing JavaScript code outside of a browser. It is used to call APIs and to talk to the backend of a web interface. It can be used to build highly-scalable, data-intensive and real-time backend services that power client applications. It also as the largest amount of open source libraries available.

2.5.12 Flask

Flask is a Python web framework which means that flask gives you the needed tools, technologies and libraries that allow you to build a web application. It is a micro-framework, so it has little to none external libraries.

Flask is light, there are little dependencies to update and watch out for security bugs. Although because of this you have to do most of the work yourself and install plug-ins and not depend on external libraries.

2.5.13 Django

Django is a high-level web framework developed with Python. It is extremely fast, secure and exceedingly scalable. It encourages rapid development and a clean design.

It comes with dozens of external libraries that you can use to help you develop your web framework. It can take care of things such as user authentication, content administration and

sitemaps. It has higher security than Flask and a lot of other web frameworks. It is also incredibly versatile and can be used to build numerous types of websites.

2.5.14 Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.[21]
If you want to work with data using python than you need to use Pandas. Using Pandas you can load, prepare, manipulate, model and analyse data. You can merge data from multiple sources and analyse it.

Pandas uses a data frame to store data. It stores data in a data frame and indexes each column and row of the dataset. IT allows us to view the information of the data, the object type and how much data is in the dataset. It allows us to delete columns or rows from the data as well as add more in.

2.5.15 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.[22]

NumPy is another library that is essential for data science and analysis, it can crunch numbers much faster than just using standard python. It helps us to create arrays that put our data into a readable form for our models.

2.5.16 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.[23]
Matplotlib is perfect for data visualising. Data visualisation is an extremely useful tool as it gives us more insight into our data. You can generate histograms, bar charts and other types of visualisations.

2.5.17 Scikit-Learn

Scikit-Learn is a simple and efficient tool for data mining and data analysis. It is built on NumPy, SciPy and Matplotlib and it is open source.[24] It contains a wide variety of Supervised and Unsupervised machine learning algorithms such as KNN and Naïve Bayes.

2.5.18 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.[4] TensorFlow was developed by Google, it takes input as a multi-dimensional array, otherwise called a tensor. The input goes in at one end and it flows through a number of operations and it comes out the other side. It is extremely versatile, it can be ran on a GPU, CPU and even mobile. It also includes TensorFlow Serving which is a great tool for deploying TensorFlow models to production. TensorFlow Serving was considered for this project, in the end it was apparent that it wasn't needed but it is something that could be implemented at a later stage.

2.5.19 Keras

Keras is an interface that allows you to customise and access frameworks such as TensorFlow or Theanos. In relation to this project we will use Keras and TensorFlow as Keras backend. Keras is great for implementing neural networks. It is user friendly and widely used. It is also written in Python which fits in well with the project.

2.5.20 React Native

React Native allows you to build native mobile acts using JavaScript and React. This gives you a full native app for android and iOS. It is built using Facebook's JavaScript library for building user interfaces. React Native makes it easier to develop cross platform.

2.5.21 Xamarin

Xamarin was recently purchased by Microsoft and has become open source. It allows you to build cross platform native applications using C#. It gives native user interfaces, native APIs and native user performance. It allows you to build applications and also has an added bonus of the 'Xamarin University'. A free resource to help you get started with Xamarin.

2.5.22 Dash

Dash is great for creating reactive, web-based applications, it uses an open source python library. It is still very new as it is only three years old. It is used to create analytic web applications. It can help to show stakeholders the data analysis, visualization and exploration of their data in the form of a web application. It uses mostly plot.ly, which is a python library for graphs and displaying data, along with Flask on the backend. It uses pure python to get the same results as you would by using some JavaScript charting libraries.

2.6 Web Hosting Technologies

The goal for this application is to have it available for everybody to use, at any time of the day. It should be 'always on' and ready to use. Therefore, setting up and configuring a server to host our application is essential. Below are two of the technologies that were researched and why they were chosen.

2.6.1 Digital Ocean

Digital Ocean is an Infrastructure as a Service provider, it is a simple cloud hosting environment built for developers. Digital Ocean gives you servers that you can create droplets or containers on. You can run your website/host your database on a droplet or you can use it for test purposes. Controlling your droplets is very easy due to its built in GUI and it is on a similar level to Amazon Web Services.

2.6.2 Amazon Web Services (AWS)

Amazon Web Services is a secure cloud services platform where you can host your applications on a cloud server, you can then use your databases and store on their services and use them on your application. You can also use their computing power. AWS gives you the resources you need to build sophisticated, scalable applications of any size. A particular feature of Amazon Web Services which was used in this project is the EC2 instances. An EC2 instance is a server where you can host your application code to make it available to the public on a public domain. It is used in this project to host our application. It also adds extra security to the hosted application.

2.7 Other Relevant Technology Research

2.7.1 Digital Ocean vs Amazon Web Services

Digital Ocean and Amazon Web Services are similar so the choice to use either one was mostly personal. First here are some similarities and differences…

- Digital Ocean targets small developers who need to start up small high-performance instances. It has a user-friendly, clean interfaces with few features and one-click deployments.
- It has Hourly Billing, Built-In Control Panel, SSH Key Setup and it has 7 Data Centres on 3 Continents. It has Monitoring Charts as well as REST API functionality.
- Amazon Web Services has very high virtual machine performance. It offers a broad range of IaaS/PaaS products with nearly all cloud services.
- It has Hourly Billing, Built-In Control Panel, SSH Key Setup and it has 10 Data Centres on 4 Continents. As well as, Monitoring Charts and REST API functionality.

They are both similar in a lot of ways with few differences. Amazon Web Services was chosen for this Project as it easier to navigate and use, as well as being more reliable. Amazon Web Services also has a very good student credit deal.

2.7.2 MongoDB vs MySQL vs PostgreSQL

MongoDB is a NoSQL database which means it does not use Sequel Querying Language. Whereas, MySQL uses SQL, SQL language is the preferred method of database querying for this project as it has been used on many projects before this one and the author is more familiar with the MySQL database than with MongoDB and NoSQL.

Although, PostgreSQL is the database that will be used on this project as MySQL is not compatible with the newer versions of python a last-minute decision was made to migrate to a PostgreSQL data instead.

2.7.3 AWS RDS vs PostgreSQL Local Server

The author debated whether to use a PostgreSQL local server on their local machine or whether to create an Amazon Relational Database Server on Amazon Web Services. It was decided that creating an Amazon RDS was the best option here as it is easy to set up, operate and scale in the cloud. The size is easily changeable, it takes away a lot of the time-consuming administrated tasks that the local server would involve. It backs up your database at regular Intervals and is fast preforming, has high availability and it also has excellent security compared to a local PostgreSQL instance and it is easily scalable, which will be useful for future use after the project has been completed.

2.7.4 Node.js vs Django vs Flask

Nodejs allows you to use open source JavaScript on the server side of your project and not just on the front end. It is good if you are using JavaScript on the front end and want to keep a similar coding style. Node.js is incredibly scalable but for this project we won't need to be scaling massively. Node.js follows event driven programming architecture. It has good performance, but it is less complex than Django. It is widely used all across the globe, used by many big companies and is quiet ahead technology wise compared to Django. Node.js has good security but the developers have to be aware of this and make sure that it is secure.

Django is an open source web framework coded in Python. It is not as scalable compared to Node.js and it follows the Model, View, Controller architecture, unlike the event driven architecture that Node.js follows. It has better performance and it a lot more complex. It is relatively new compared to Node.js, leading it to be a bit more behind in terms of usage. Django has excellent security and the developer doesn't have to worry about it too much.

Flask is a lot more lightweight than Django. It comes with less right out of the box, although it is still very powerful. It is also commonly used with hosting deep models.

Flask has been chosen to be used in this project as the project will involve working with a very large dataset and it will need to have a powerful backend. It will also have to be able to work well alongside a deep model. Therefore, Flask is a better fit for this use case.

2.7.5 React Native vs Xamarin

React Native and Xamarin have major common points. They are both great and do practically the same thing. In this section you will see the comparison of the two and their benefits. React Native is still relatively new, there is not as much support compared to Xamarin. When using Xamarin you used Visual Studio as an IDE, this keeps everything together, you can code your application, connect your API's all through here. React Native has less features like this. When it comes to code compilation, Xamarin wins. Cross platform environment, Xamarin wins here too as everything is done through Visual Studio, React Native usually runs on Expo as an IDE, which doesn't support all of React Natives features. In terms of documentation, React Native is better than Xamarin, everything you need to know is in one place and well documented. Whereas, Xamarin documentation could do with some improvement. Since Xamarin is an older platform, the community is a lot larger than React Native which is relatively new, getting assistance will be a lot easier while using Xamarin. In terms of performance, Xamarin is better as it supports 64-bit Android whereas, React Native does not.

With all these in mind, the application section of this project will hopefully be completed using Xamarin, given time constraints. Not only has it better support and more features needed for this project. It has never been used in a Final Year Project before, which will be a great challenge to complete to be the first project to use this platform.

2.8 MACHINE LEARNING ALGORITHMS

This section of the report is to discuss the researched predictive algorithms and techniques that were considered for use on this project.

2.8.1 KNearestNeighbors

KNearestNeighbor or KNN for short, is a machine learning algorithm. It can be used for classification or regression, mostly classification. KNN was used in the prototype for this project. KNN is a very simple algorithm. It identifies the K nearest neighbours of C. If we have classes of plus and minus and k = 5 then we have to find the 5 nearest neighbours to C. The algorithm then uses these results to predict the outcome.

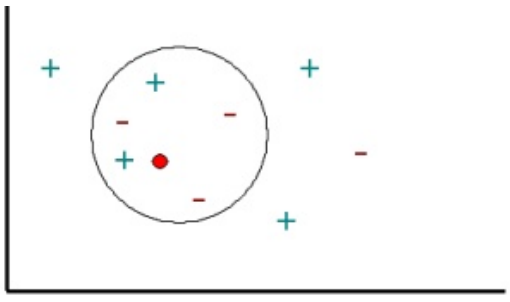
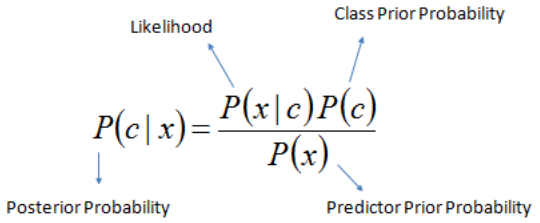


Figure 4: KNN example

2.8.2 Naïve Bayes

This is a machine learning algorithm for classification problems. It is mostly used for text classification like sentimental analysis. It is simple yet extremely effective. It learns the probability of an object with certain features belonging to a particular class. It uses the Bayes theorem which states that probability of event C given X is equal to the probability of the event X given C multiplied by the probability of X upon probability of C.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

[25]

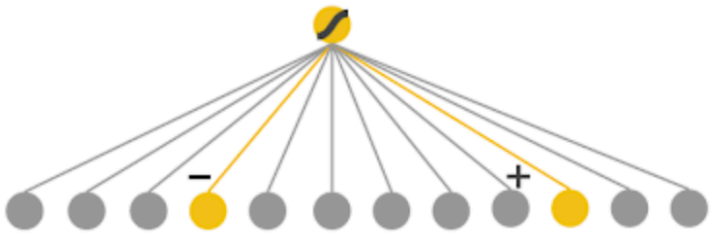
Figure 5: Bayes Theorem.

2.8.3 Wide & Deep Learning

'Wide & Deep Learning'[26] is a new technique, developed in 2016 and released by TensorFlow. It uses memorisation and generalisation, and it is a joint classification system that trains traditional hand-engineered features along with word-embedding vectors that share the same target feature while keeping the neural networks different. Traditional classification systems can predict based on what they have already seen in the data. Whereas, word embeddings create new relationships between data points and the data. When you concatenate the two together, you get a much more advanced model. It works very well with recommendation systems and ranking problems. Which benefit the project as the system recommends or predicts a rating from 0-100. Google tested and implemented this model in the Google Play Store and saw a significant increase in app acquisitions.[26] In essence, the model returns a rating that matches with most of the input data.

2.8.4 The Wide Model

The Wide Model uses memorisation. You want to memorise what rating works best for each user input or query. It uses a linear model with numerous wide cross-product feature transformations to capture how the concurrency of the input data matches with the target label; in this case, the rating of a wine. The wide model predicts the probability of a particular rating returning for each set of features. As you can see per the image below, a query would represent a wines description/review, and an item would be the variety of grape. A Merlot and a Pinot Noir could have a similar description, but the model memorises which returns a particular rating.



AND(query="Dense and Rich Smooth Taste", rating="88")AND(query="Dense and Rich Smooth Taste", rating="83.5")
Figure 6: Wide Model

2.8.5 The Deep Model

The Deep Model gives us a more advanced way to tackle the problem at hand. Our model is returning predicted ratings well, but it needs to be improved even more. The latter is where the deep model enters our hypostasis. With the deep model, we are learning 'lower dimensional-dense representations' or so-called word embeddings for every description or 'query'. By using this, our application uses generalisation by matching items to queries or varieties to descriptions, that are close together in the embedding space and matches them with the best-suited rating.

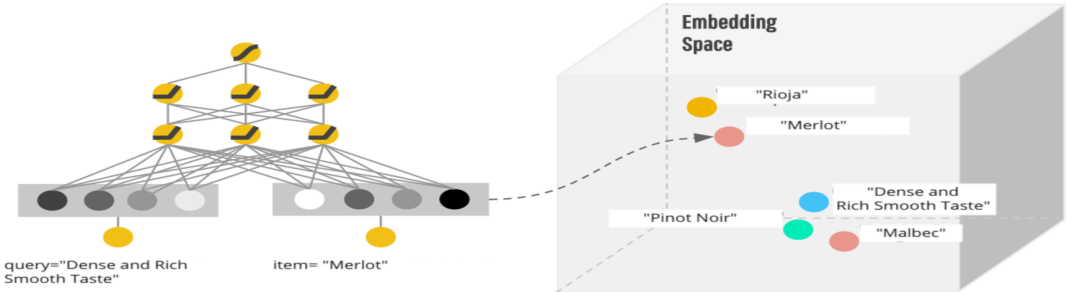


Figure 7: Deep Model

2.8.6 Combining Wide & Deep Models

By combining the models, we can prevent the deep model over generalising by letting the wide model have input, and this leads us to a more accurate rating result. While training the model, the prediction errors are backpropagated to both sides to train the model's parameters. The wide model memorises the inputs and relates them to which rating suits best, while the deep model can generalise which descriptions match the rating based on the

word embedding.

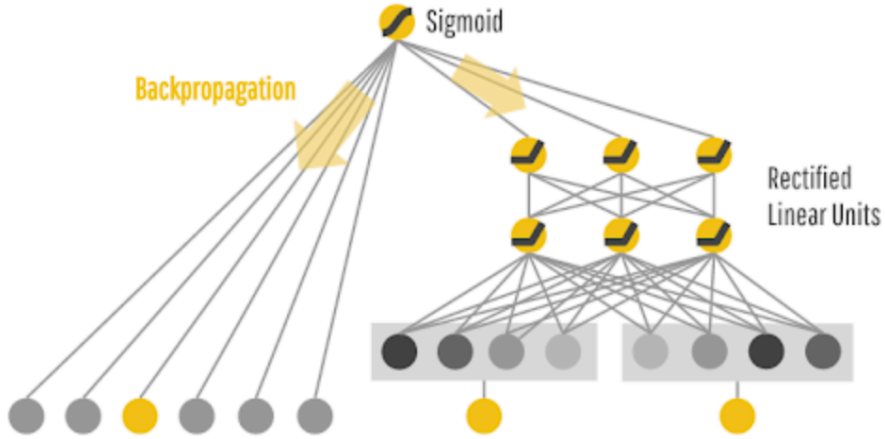


Figure 8: Wide & Deep Model

2.9 Machine Learning Techniques

2.9.1 Natural Language Processing

Python has a vast library called Natural Language Tool Kits. *'NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. '*[27]

2.9.2 Bag Of Words

We also use Bag Of Words – this embeds sentences as a list of 0 or 1, 1 represents containing a word. This model extracts features you could use in your predictive model. Its called a “bag” of words, because any information about the order or structure of words in the wine descriptions discarded. The model is only concerned with whether known words occur in the descriptions, not where in the descriptions. As you can see in figure 9, there are three descriptions. After the bag of words is implemented in figure 10, the words are mixed up and given a numeric vector of 0 or 1.

D1 -	Dark, menthol, warm earth.
D2 -	dark and dense, warm bodied
D3 -	light bodied gathering of cola and pine

Figure 9: Descriptions Pre Bag of Words

	Dark	menthol	warm	earth	and	dense	bodied	light	gathering	of	cola	pine
D1	1	1	1	1	0	0	0	0	0	0	0	0
D2	1	0	1	0	1	1	1	0	0	0	0	0
D3	0	0	0	0	1	0	1	1	1	1	1	1

Figure 10: Descriptions post Bag of Words

This depicts the training features containing term frequencies of each word in every description. The bag of words approach has its name since the number of occurrence and not sequence or order of words matters in this approach.

2.9.3 One-Hot Encoding

The chosen dataset has primarily categorical features. A categorical feature is, in essence, a variable that contains label values rather than numeric values. The issue here is that when using a neural network, the model won't react well to large amounts of categorical data. It works best with numeric features and vectors. So that is where one-hot encoding aids in the creation of the model.

One-Hot Encoding helps us to create a relationship between each feature, one which we cannot achieve unless the values are numerical. When a feature is one-hot encoded, the model reads our data as a vector of integers. Our categorical features become 0s and 1s. The length of these vectors is equal to the number of categories that our model is expected to classify. Below we look at an example.

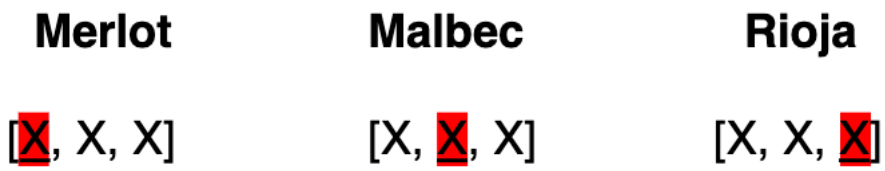


Figure 11: One-Hot Encoding of Varieties (i)

We have three varieties of grapes. Merlot, Malbec and Rioja. Our vectors would each be of length three as there are three varieties. Each element or index of the vector corresponds to one of the three varieties. In each vector, every element will be zero, except the element that corresponds to the category of the given input.

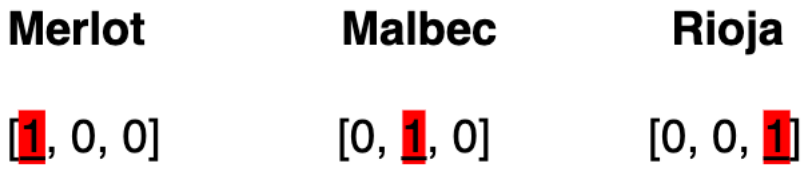


Figure 12: One-Hot Encoding of Varieties (ii)

When the model receives an input of Merlot, It's not reading the label as Merlot, it is reading it as 1, 0, 0. When the model receives an input of Malbec it reads it as 0, 1, 0 and when it receives an input of Rioja, it reads it as 0, 0, 1. If another variety is added, the vectors will then be of a length of four, as per the image below.

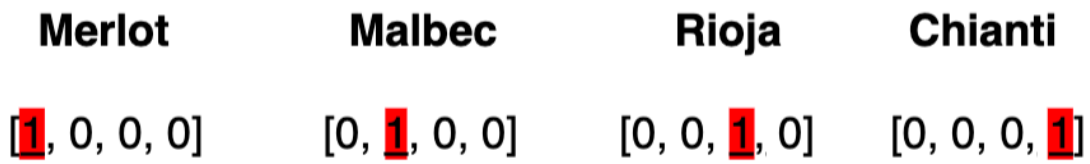


Figure 13: Figure 13: One-Hot Encoding of Varieties (iii)

2.9.4 TF-IDF (Term Frequency, Inverse Document Frequency)

The next model we will look at is **TF-IDF (Term Frequency, Inverse Document Frequency)** – this model is weighing words by how frequent they are in our dataset, discounting words that are too frequent. It works by finding out how often a word appears over the total number of words.

$$tf(t,d) = \frac{number\ of\ occurrences\ of\ term\ in\ document}{total\ number\ of\ all\ words\ in\ document}$$

The Term Frequency (TF) of a term, t, and a document, d.

Figure 14: Term-Frequency equation

2.10 Resultant Findings and Selected Technologies

2.10.1 Data Analysis & Models

After analysing the research, the author found that KNN was not the best approach to this problem. KNN was not accurate, and it would not give the desired results. The author decided that Naïve Bayes, although giving a good result, does not give the needed result using probability. This lead to the wide & deep model being used for the remainder of the project. It gave better results and also achieved these results in an exciting and new way. It was found that TF-IDF (Term Frequency, Inverse Document Frequency), would not be of use in this project as it is used primarily in finding the frequency of terms in large documents. Our descriptions are only short sentences, so there was no need to use this technique. Instead, we use 'bag of words' and 'one-hot encoding' to aid our wide & deep model, along with other natural language processing techniques. The Python language is used along with libraries such as Pandas and NumPy to analyse the data, see trends in the data, view the correlation of features between each other and also to clean the data so that it is prepared and ready to be used in the model.

A massive amount of knowledge on this subject has been gained from the domain expert interview. It gave an excellent insight into what makes each feature of the dataset different and the factors that determined why each feature is what it is. The data can now be adequately analysed and cleaned accordingly as well as the models become more accurate since we know the background of the features.

2.10.2 Frontend

For the prototype, Dash was used. This is an incredible front-end choice for the prototype as it gave us an insight to the data in a visual heavy way. For the overall project, Flask will be used along with HTML, CSS, JavaScript and Bootstrap to create a front-end for our Flask server and a place for users to interact with the model. It will also be a place where a user can make an account and review their own wines for other users to see. I chose these technologies as I love what dash can do in regard to visualizing data, it gave us an interesting perspective for the prototype, but flask alongside the chosen front-end technologies will give us a user-friendly UI that will help to interact with our trained model.

2.10.3 Backend

I have chosen MySQL as the backend for this project as our userbase will be small. There is no need for a NoSQL database such as MongoDB, but it is a possibility in the future if the user base begins to exceed 100,000+ daily users, then it will be necessary to migrate to a database provider such as MongoDB. For the project as of now, MySQL will do a perfect job.

2.10.4 Server Side

Amazon Web Services with Flask.
I have chosen this combination as I feel like Amazon Web Services has a lot of support in regards of documentation and help. It also has a lot of features that could be useful throughout the production of this project. I have researched that it works well with Flask and Flask was chosen as it is a python backend as this project involves a great amount of python coding. For the server side on the prototype, it was Amazon Web Services with Flask since is a built-in feature of Dash.

2.11 PROPOSED SOLUTION

Using the above findings, The goal is to create an Amazon Web Services Server which will host all of the application. Flask will contain the backend APIs which will call the web application and also connect to the MySQL database using Restful API calls. The data cleaning and predictive analysis will be completed through Jupyter Notebooks, An interactive Python/R IDE.

3.0 Design

In this section of the report I will take you through the design approaches and methodologies that were used throughout the project. These were followed and used to structure and each component of the project. Such as the data analysis, machine learning, database design, system's architecture, the web application and use cases.

There was two approaches and methodologies used. The project took on the Agile Methodology as well as CRIP-DM as it is data and machine learning heavy.

3.1 Agile Methodology

Agile Methodology is a term that covers several project management approaches that allow teams to respond to changing requirements through incremental and iterative work. Agile came about in the 1990s when the software industry began to rise, and this is when rigid, sequential software development methods couldn't keep up with the rapidly changing requirements and priorities in projects. How this works is Business Users relay their product requirements to developers through user stories. The team takes the user stories and figure out how to make the product in stages and make sure that the product gets delivered on time to the business users. User stories are sorted into order of what needs to get done first according to the priority of the user story. These are then completed in sprints, which are usually one or two week blocks of where specific issues and user stories are assigned to developers to complete within that time frame. A Scrum Master all overlooks this. Developers add to the product as they go and improve 'on the go' then adjust accordingly depending on feedback from other developers. This feedback stage is called a retrospective. The sprints continue until the product is delivered and the cycle starts over again.

This model suits this project as many variables changed throughout the project, and this is a more flexible model compared to the waterfall model. Working on this project with an agile mindset helped to keep everything moving smoothly, The agile way of working that was gained through college experience and work experience when applied to the project made hitting targets and goals a lot smoother.

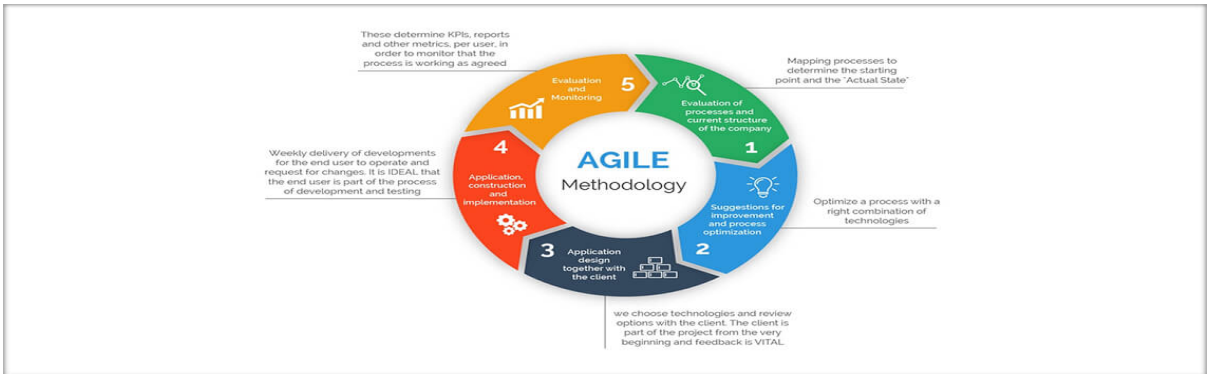


Figure 15: Agile Methodology

3.2 Crisp-DM

When working with data it is a good idea to follow a Crisp-DM model. Crisp-DM stands for 'Cross Industry Process for Data Mining. It is a very popular methodology which uses a structured approach to plan a data mining project. It has six phases which are Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation and Deployment.

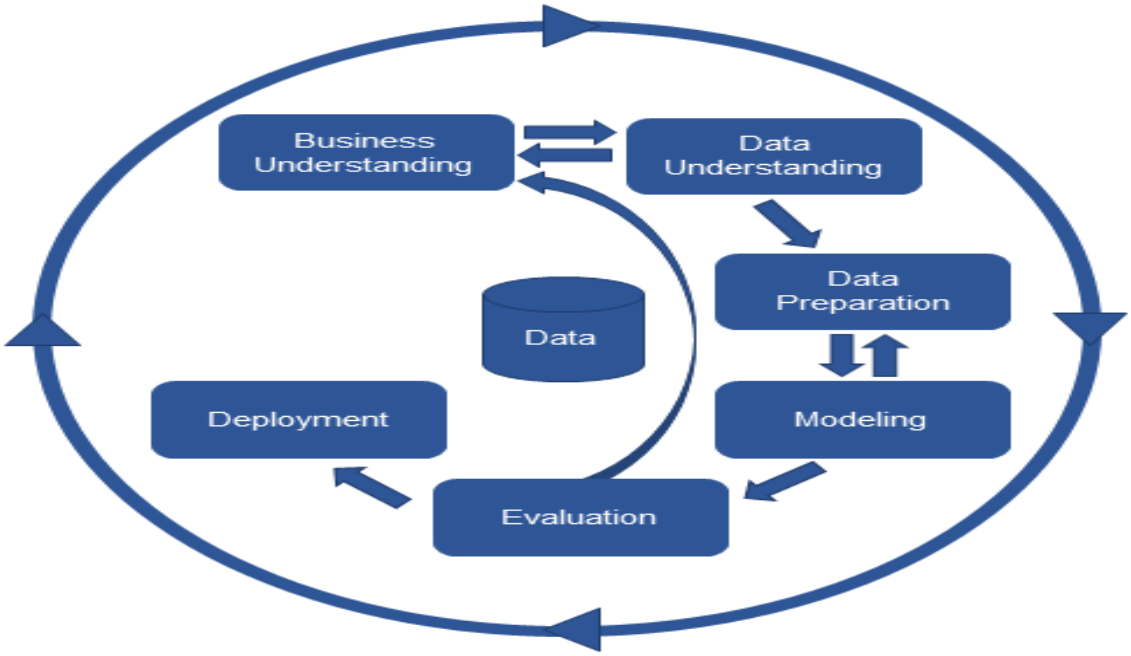


Figure 16: Crisp-DM Methodology

3.2.1 Business Understanding

In the Business understanding phase, you have to figure out what is that has to be done. The business objectives have to be determined, determine the project goals and produce the project plan.

3.2.2 Data Understanding

The Data understanding stage is where the data has to be found and has to be analysed to see if the data will suit to the needs of the project as well the data's strengths and limitations. The data has to be collected, explored and the quality has to be verified.

3.2.3 Data Preparation

In the data preparation stage the data has to be prepared. This stage is the most time consuming. The data has to be selected, cleaned, constructed and integrated with the product. This stage and the previous stage are the most time consuming.

3.2.4 Modelling

In the Modelling stage the model you wish to use has to be applied to the data. This stage is completed side by side with the data preparation stage. Models are built as you find your data and changed depending on the results of the data preparation stage.

3.2.5 Evaluation

In the Evaluation stage the models have to be evaluated for suitability with the project and end product. Does the model help to satisfy the business goals? If so then the project moves to the deployment stage, if not then the data gets brought back to the business.

3.2.6 Deployment

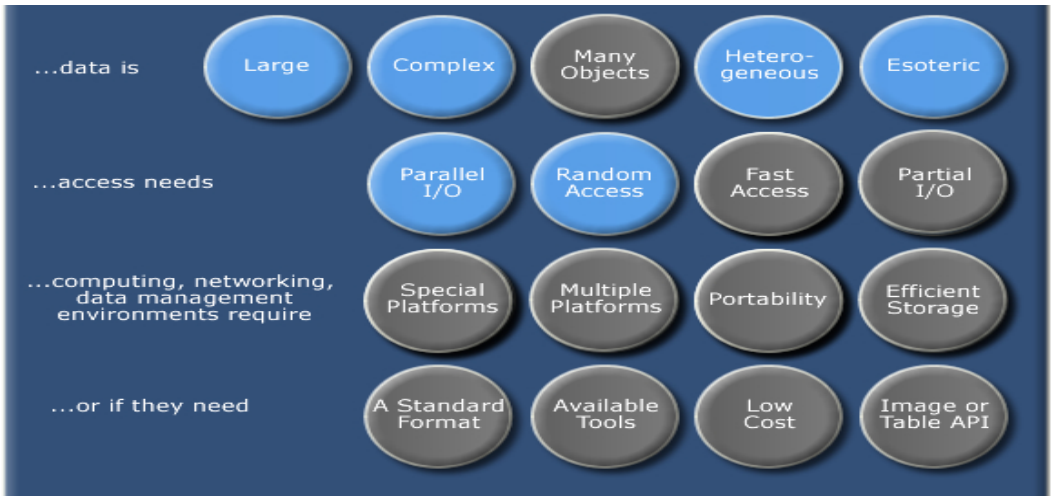
The Deployment stage is the final stage in the CRISP-DM life-cycle, it is when the prototype has to be developed to a final working product and it then gets deployed to a production environment.

3.3 Database Design

This project only has two database tables within the database. Users and Reviews. The data for the deep learning model is saved as a .h5 file on the AWS server. In this section we will go through the HDF5 file design and the PostgreSQL database design.

3.3.1 Hierarchical Data Format

The Hierarchical Data Format version 5 (HDF5), is an open source file format that supports large, complex, heterogeneous data.[28] HDF5 uses a file directory like structure that slows you to organize data within the file in different ways. Organizations would use this file format for the following reasons:



It structures data inside the file like this:

Figure 17: Why use HDF5?

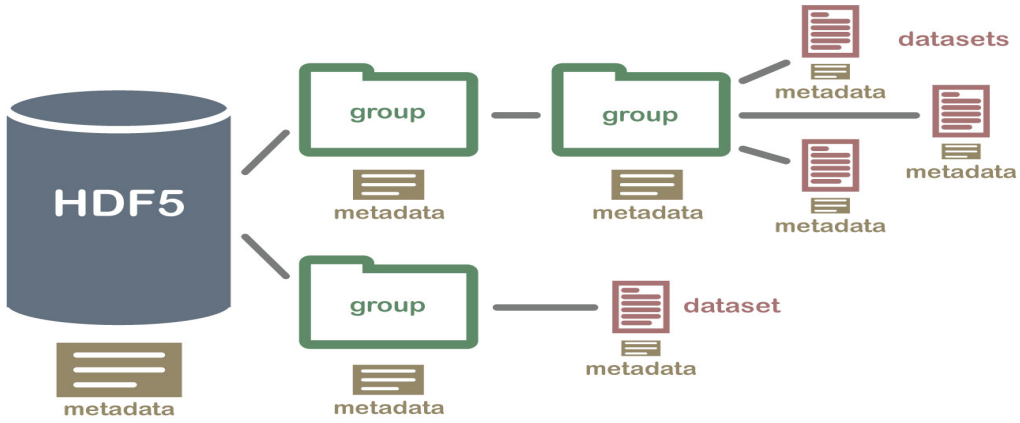


Figure 18: HDF5 Structure

It is a compressed data format meaning that any data stored inside of the file is optimized to create a smaller file, which is great when you are using it to store a deep leaning model as they can often contain large amounts of big data and be quite large in size. HDF5 files have the ability to 'slice data' which is where it can choose a particular subset of data and extract that subset to be used. Meaning that the file allows reads that slice or subset into memory which in turn means that it will keep speed and efficiency high. It also has heterogeneous data storage, the file can store multiple data types within the same file such as numerous datasets, all with different datatypes e.g. string, integer etc. It is very popular for saving deep models as the different data types it can store range from temperature, precipitation and photosynthetic active radiation data, to images that each have specific spatial information. It also has an open format, which allows it to be used with a number of different programming languages.

3.3.2 PostgreSQL Database

I used pgAdmin4 for any administrator purposes when using the PostgreSQL database. It's a tool from Postgres which allows you to manage your database and view relevant information and processes of your database. It also gives a dashboard where you can check server sessions and transactions per second among other features.

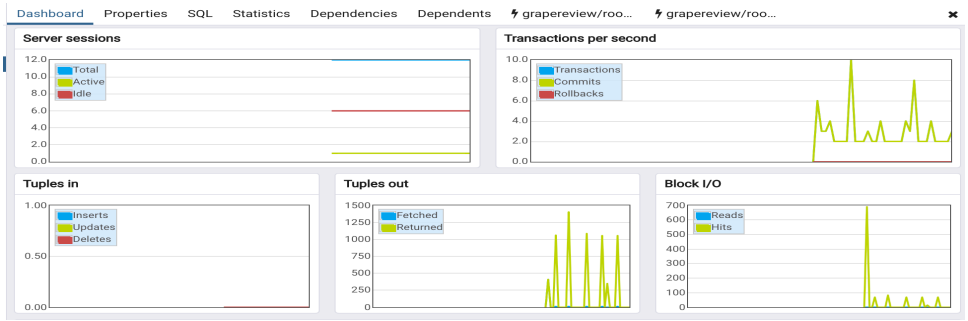


Figure 19: pgAdmin4 Dashboard

The PostgreSQL database is an amazon web services relational database instance (RDS). AWS RDS is a great service it allows use to easily host out database in amazons cloud and it takes care of security and maintenance. AWS RDS also comes with great dashboard features for monitoring your instance.

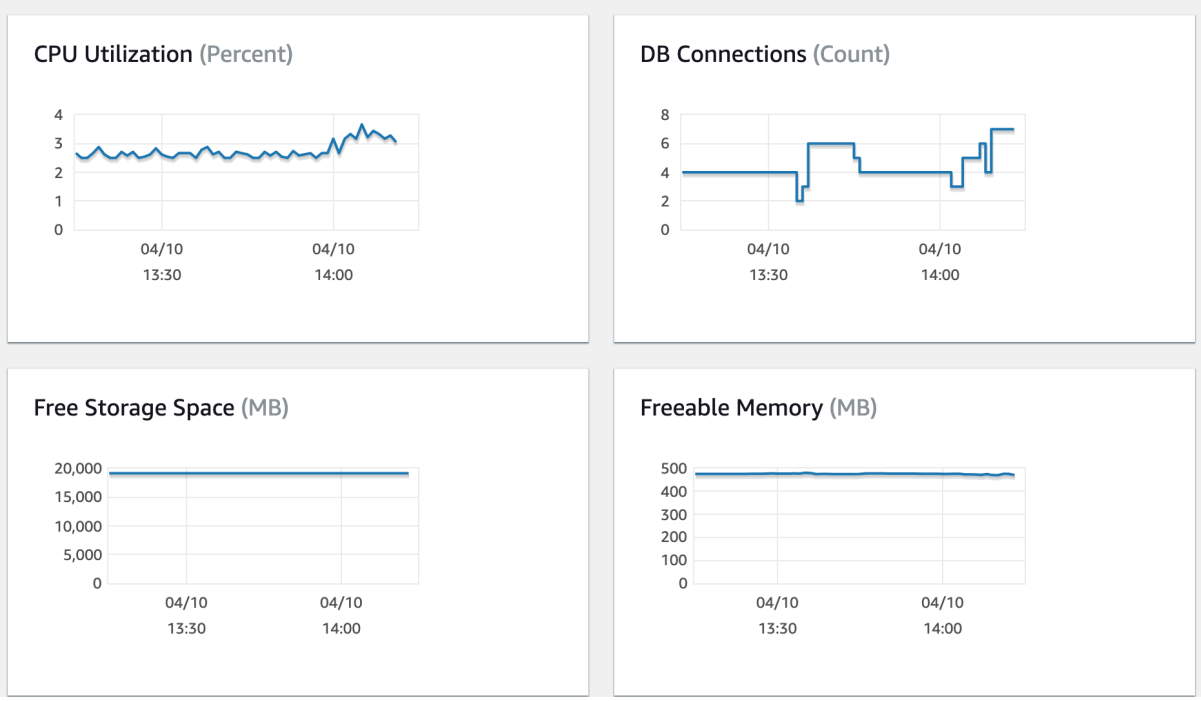


Figure 20: AWS RDS Dashboard

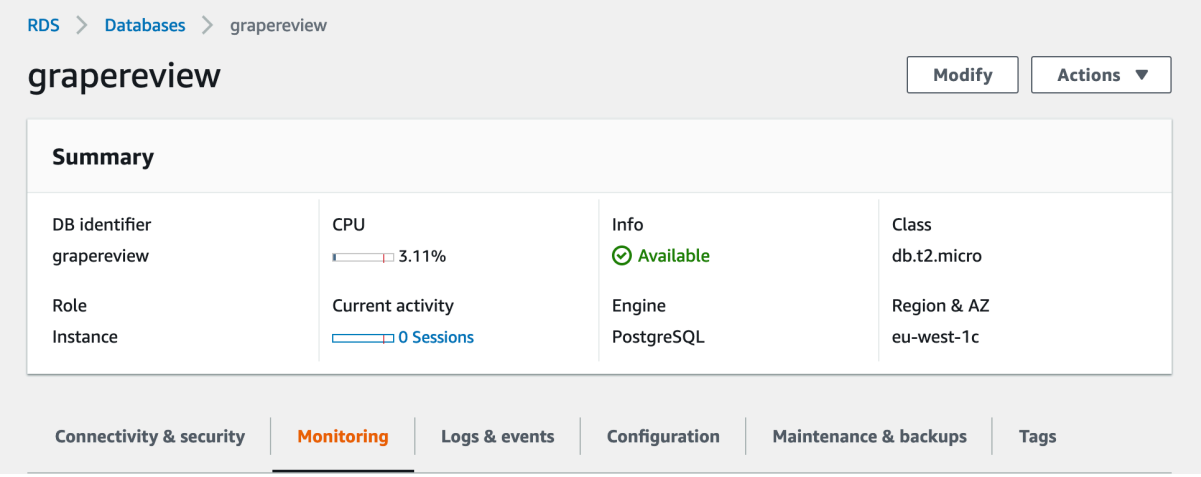
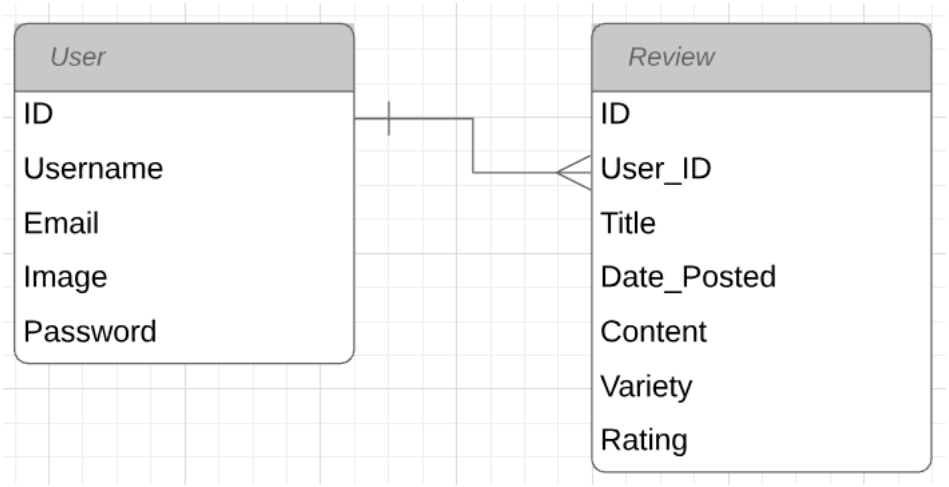


Figure 21: AWS RDS Overview

The database consisted of just two tables, ‘User’ and ‘Review’. The User table was necessary for each user to register and log in and have their own reviews and the Review table was necessary to store and post reviews created by the users.



The User table is used to store all of the information that a user needs to get authenticated ad to have an account. It includes their unique ID, their username which can be anything that they choose, their email, an image which is auto assigned by a default image but can be changed on their account page and also has their password which can be any length and is stored in the database in an encrypted format using the Bcrypt module that can be installed with Flask. The Review table consists of a review ID, a title, review content and the date the review was posted. This allows the user to create a review on any wine that they would like to review. It gives them a place to share their reviews for other users to see. The tables have a One-to-Many relationship as one user can have many reviews, but a review can only have one user.

User Table:
The user table consists of 5 attributes which we will go through here:

ID: Int
This field is a unique identifier for each user on the database. It cannot be null. It is the Primary Key.

USERNAME: Varchar (20)
This field is a unique identifier. It refers to the username of a user. The username must be within 20 characters. This could be a nickname or anything the user chooses. It cannot be null.

EMAIL: Varchar (40)
This field is a unique identifier. It refers to email that the user will use. It must be within 40 characters. It must also be in an email format. It cannot be null.

IMAGE: Varchar (20)
This field holds either the users default image or the image that they upload. The data type is a varchar because when the user uploads and the flask application will change the image into an 8-byte token so that it can be stored in the database. It cannot be null.

PASSWORD: Varchar (60)

The password field can be a string up to 60 characters. The user must have a password when registering an account. It cannot be null

Review Table:
This is where users can create reviews that are stored in the database. We will go through it's 5 attributes here:

ID: Int
This field is a unique identifier for each user on the database. It cannot be null. It cannot be null. It is the Primary Key.

USER_ID: Int
This field is a unique identifier for each user on the database. It cannot be null. It is a Foreign Key that is related to the users table. It is the key that links each user to their reviews.

Title: varchar (100)
This field is the title of the review. It was must within 100 characters. It cannot be null.

Date_Posted: DateTime
This field contains the date and time that the review was posted. It is set to autofill the current timestamp at the time the user submits a review. It cannot be null.

Content: Text ()
This field contains the content of the review, or the main body of the review. It stores data in a text format which allows the user to input long sentences. It cannot be null.

Variety: String(20)
This field contains the variety of grape/wine that the user is reviewing. It cannot be null.

Rating: Int()
This field contains the rating a user gives their wine. It cannot be null, It is set to a minimum of 0, maximum of 100.

3.3.3 SQLAlchemy

SQLAlchemy is used in this project. SQLAlchemy is the Python SQL toolkit and Object Relational Mapper (ORM) that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.[29]

3.4 Use-Case Diagram

Site Usability Use Case:

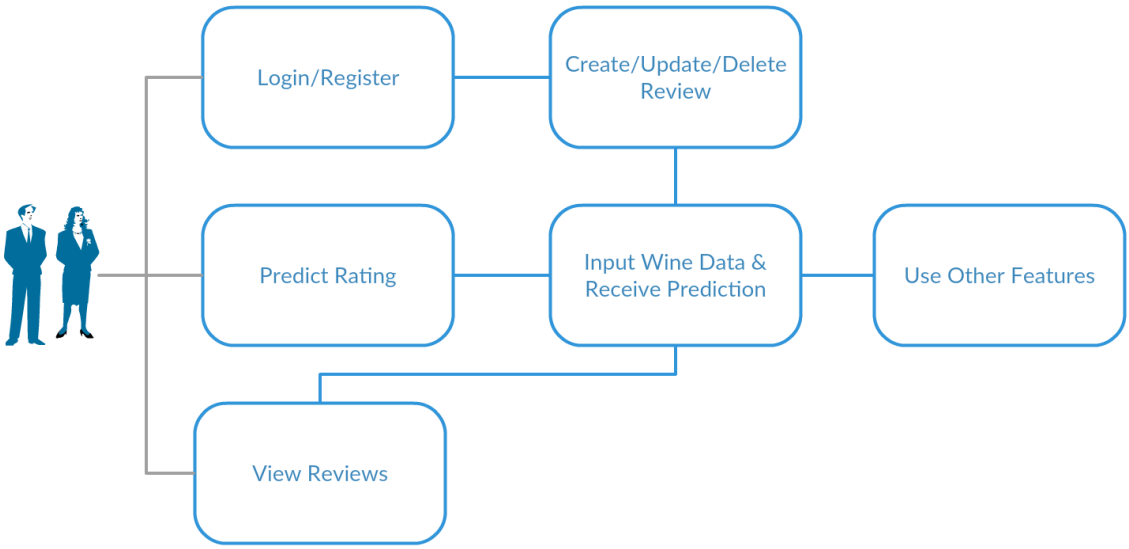


Figure 22: Use-Case Diagram for a user using all of the features on the application

Figure 22 represents the users' overall experience from start to finish while using the application. It is a use case that shows every aspect of the application and its main features, the flow of which the user can go through the application and where they start and finish. The application user has three choices when they enter the site. They can either view the reviews on the application, Use the prediction functionality of the application to predict a rating for their wine, or they can log in/Register to the application to use certain user-only features.

If the user chooses not to log in or register, then they have the option to read other users reviews, and they can also start using the predictor. The user views the prediction form on the home page of the web application. The user then will be able to input their own wine description/review, the variety of grape that the wine is made from and the country from where the wine has originated. They input these details into the form and press the predict button which will then bring you to a new page and give you the result of your wine rating. From here the user can use the other features on the application available to them.

If the user wishes to go the route of logging in or registering, the user will then be allowed to go to their account to update their account details, and they could also create a new wine review for the rest of the users on the application to see. They could update and delete that wine review, and they could also use the prediction function just as the other users have, they could also use other functions of the application.

3.5 Web-Application Source Code Layout

Flask does not follow the Model, View, Controller structure such as Django but it does follow a structure that is similar. Flask is a very popular web development framework that is lightweight and fast. The file structure of the source code is made up of a folder that includes another folder named winesite as well as our app.py file which initialises the server and a requirements.txt file which has all of the required modules to run the program on a new machine. In the winesite folder there is a h5 folder, a static folder, a templates folder and files such as routes.py, models.py, __init__.py and forms.py. Within the static folder there is a folder for images on the site, the CSS code for the site as well as the folder that holds the profile photos. Within the templates folder there are all of the HTML pages that are in the application. Finally, within the h5 folder there is our h5 file and our tokenizer which are used while making a prediction.

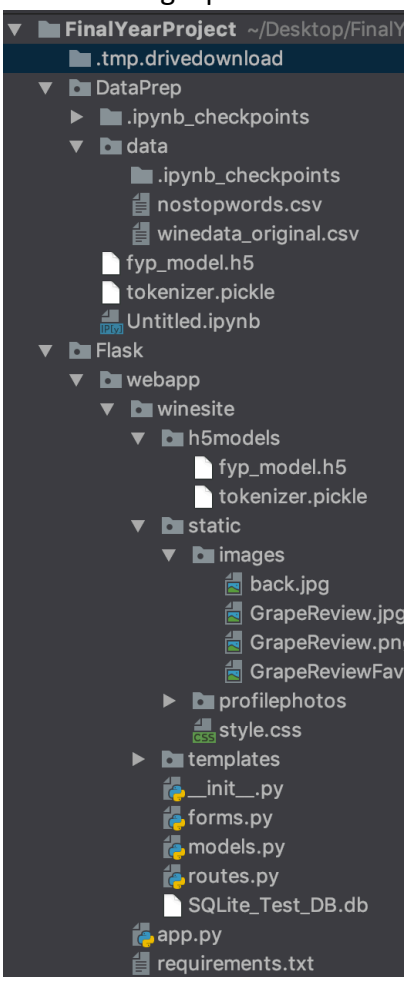


Figure 23: Source Code Layout

DataPrep

The dataprep folder contains the dataset as a csv file and data analysis code and model in a jupyter notebook

H5models

This folder contains the .h5 file which is our Wide & Deep neural network ready to be used. This folder also includes the tokenizer.pickle which is necessary to one-hot encode the users input to make it a readable format for the model.

Static
The static folder includes all of the user's profile photos and the other images used in the application. It also is where the style.css file is allocated, where we can change design aspects of our application.

Templates
The templates folder is where all of the .html pages are. Every page the user sees on while using the application will be here.

__init__.py
The file keeps all of the applications configurations. It calls in the PostgreSQL database from Amazon Web Services. It includes a 'SECRET_KEY', which is gotten by using the python secrets modules and creating a 16-bit token, which allows us to access the application once that key is included in this file. The file also includes the app variable initialisation and flask-LoginManager module configurations which allow us to have a working login form along with other features.

Forms.py
This is where any form included in the site is created and called from. There are a number of forms on the site, one for registering, logging in, creating a review, viewing the reviews and also for the prediction. An Example of code to create forms would be *Figure 25.*

Routes.py
The routes file holds all of the API routes created to access our html files on an endpoint. Along with that it is also where our prediction functions are written where it takes user input from a form and runs it through the necessary steps to prepare it for the deep model which it then gets passed through and returns a prediction on a different page. The routes file is also where functions such as updated the account and creating reviews are coded. Among a number of other functions.

Models.py
This file is the file that holds our two models which are creating with the assistance of the ORM SQLAlchemy. Using SQLAlchemy makes it easy to create tables such as User and Review that are in this file. Minimal amounts of SQL were needed as this model file creates all of that instead.

App.py
This file is used to call the flask app and run it, it is linked to the __innit__.py file so there is not much in this app except the one command to run the application.

Requirements.txt

This file stores all of the modules required to run the application. This file is useful when setting up the virtual environment in the AWS Ubuntu Server that is used to deploy the application to the web.

3.6 Webpage Design

3.6.1 Prototype & Mock Design

The prototype of the project was developed using Dash (See *Figure 25*). Dash is great for visualizing data, but it acts as a dashboard for data visualisation and for this project a more user-friendly application was needed as the application needs to take in users data and run it through the deep model, as well as having a sections for users to review wines and more.

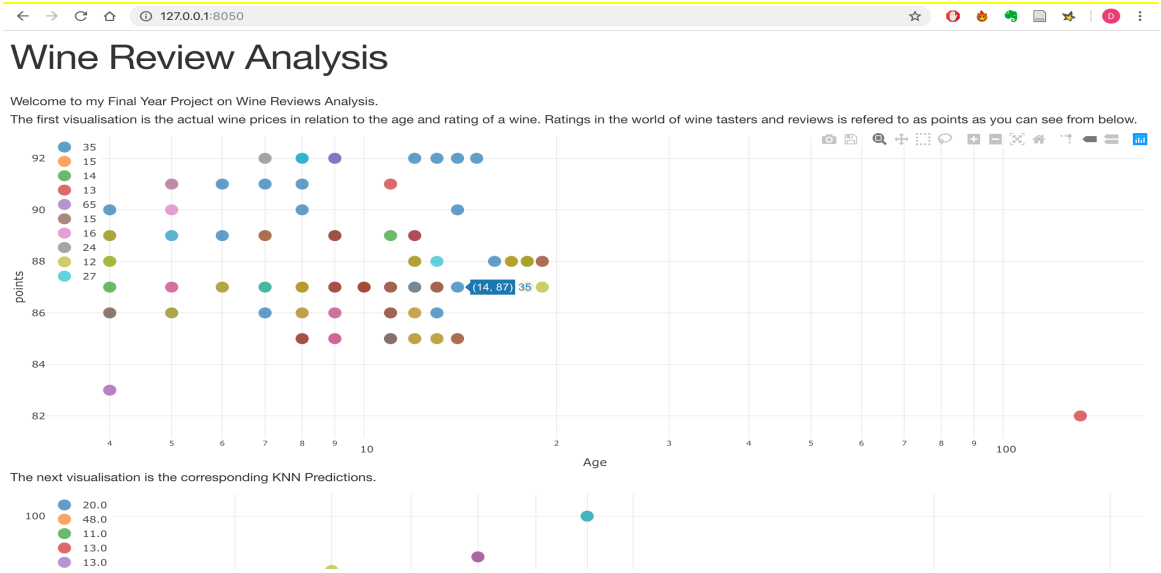


Figure 24: Dash Prototype

This led to the decision to not use Dash for the final product. The author drew some user interface design to make it easier to develop the application. Once these designs were on paper, it gave the author the ability to design and map the application using mock designs as a reference.

3.6.2 Final Design

The final designs were then created with the help of the mock designs. The author went for a minimal and clean application design has they felt that websites today have become extremely flashy and packed full of JavaScript effects and other features that take away from the main idea of the application at hand. Hence the minimal design style was chosen by the author. The application implementation included the use of Bootstrap[30] which is mentioned in a previous section of this report. Bootstrap is a premade style sheet essentially, allowing the creator to not have to worry too much about using CSS to style their applications. It is built on jQuery[31] and includes pre built components that the user

can reference through tags in their code and it will style the sections of the application which is divided into divs. Although CSS is still used to change some bootstrap design aspects. Using bootstrap also allows the application to be responsive to mobile devices as well as computer web browsers. Here we will go through some of the screenshots from the applications final design.

First is the navbar. The navbar is the top of the site that is used to navigate throughout the site with ease. The first image is the navbar a user sees if they are in the application and not signed in to their user account. Using Flask it's possible to change the navbar to show different buttons and content for users that are authenticated as you can see in the image of the second navbar design and also below that a simple footer to give the application some sense of a border.

Figure 25: Navbar

Figure 26: Navbar with User Authentication

Figure 27: Footer

The next section a user sees is the home page. The home page consists of the input form where a user can input their wine data and receive a prediction of their rating. Below this form is user reviews. This is where users wine reviews show after they have been created.

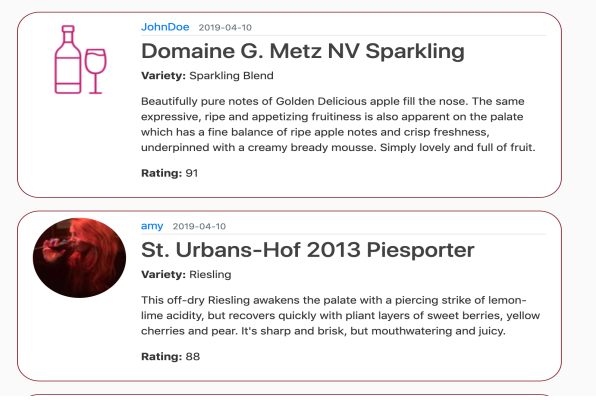


Figure 28: Homepage with Prediction Form & Reviews

The next page is the creating a new review page which is similar to the updating a review page. The flask module WTF-Forms was used to create these forms which is a great tool for creating forms in flask.

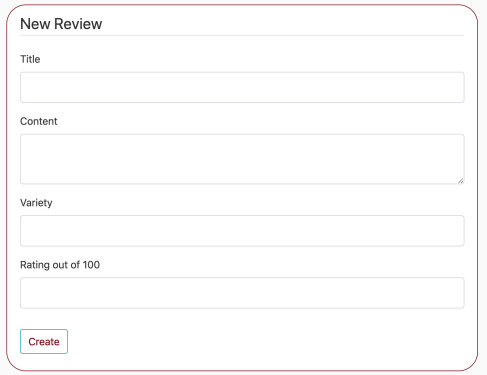


Figure 29: New Review Page

There is also the login and registration forms which are also built using the flask forms modules. These are standard forms used in many applications. All of these forms are styled using bootstrap.

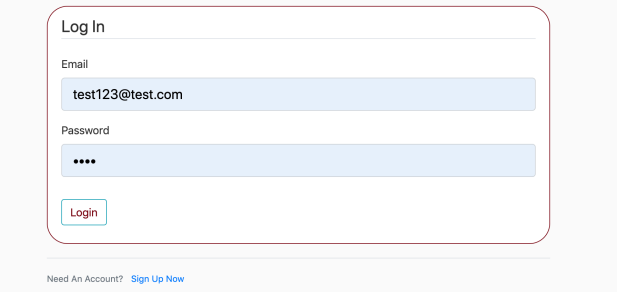


Figure 30: Login Form

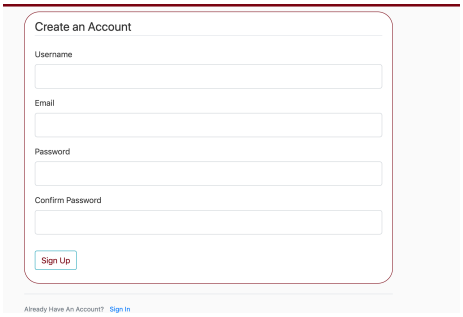


Figure 31: Registration Form

Here we have the page where the user can view all of the reviews made by one user. This uses pagination, as does the homepage, which allows the users reviews to move onto another page after there are 5 reviews on the current page. After that we have the page where a user can update or delete a review that they have made.

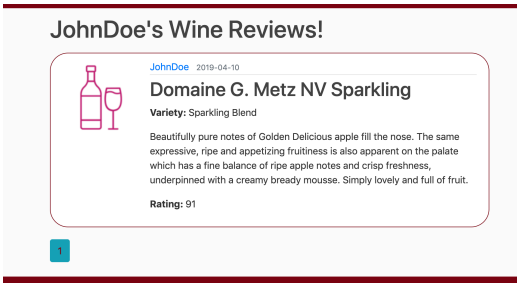


Figure 32: Users Reviews

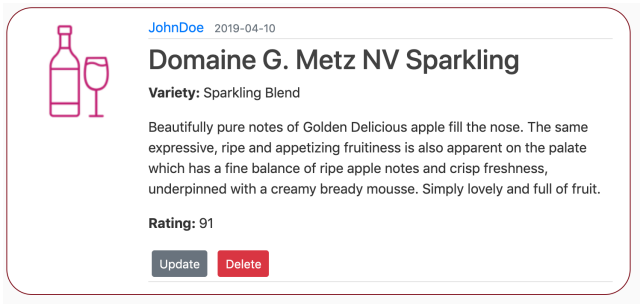


Figure 33: Update/Delete Review

There is also a user account page which shows the users account information and allows them to change their username, email and profile picture.

Figure 34: Account Update Form

Finally, there is the Google Maps page which uses the Google Maps API and has markers for wineries and vineyards all over the world.

Figure 35: Google Maps Page

4.0 Architecture and Development

4.1 Technical Architecture

This section of the report is dedicated to the architecture of the project. It will describe each component and how everything works together. For the project the author opted for an architecture which incorporates a hosting server (AWS EC2), a development server (Flask), a database (Postgres) and a client (Web Interface). The architecture is structured as displayed in *Figure: 25*.

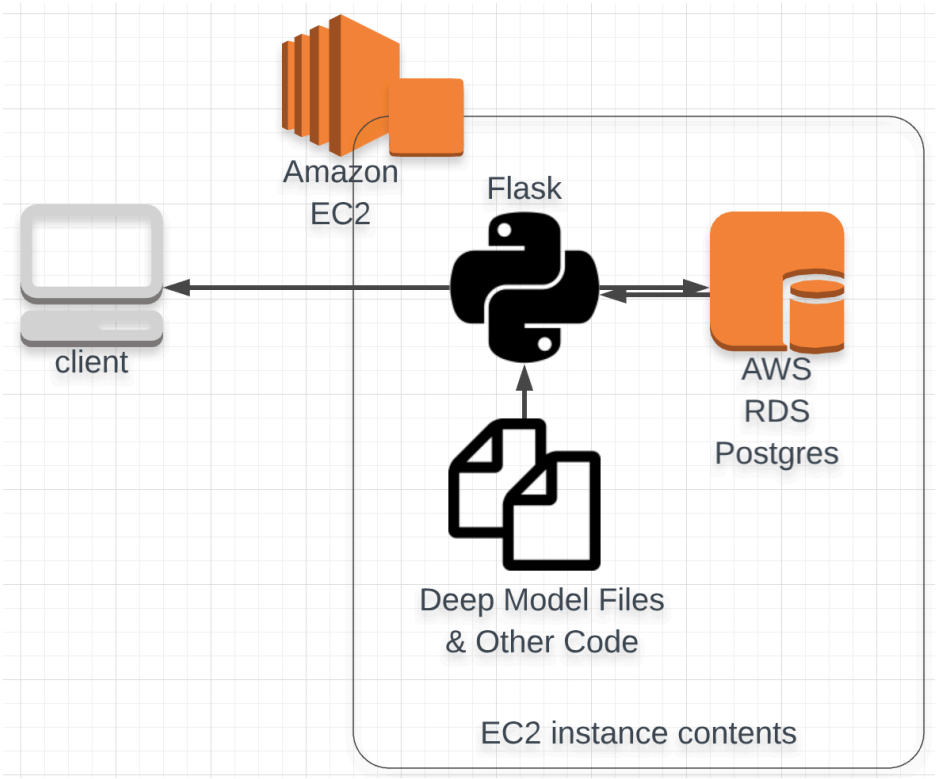


Figure 36: Technical Architecture

4.1.1 Flask

The flask server is where all the code is written and stored so that it can be used to produce a web application. The html files, CSS, python and deep model files are all kept here and configured to work together to produce the application.

4.1.2 Client

The client is where the user can access the application. This can either be a mobile device though it's browser or on any browser on a PC.

4.1.3 Database

The database (as discussed in the previous design section) is a PostgreSQL database that is hosted on an Amazon RDS instance. The author chooses this database for production as it is more secure, scalable and reliable. The database stays on constantly, it never shuts down which allows the application to stay live. It takes the database server off of the authors personal machine, allowing the database to run smoother and it is also more secure. It is also the chosen database as in the future the author plans to expand the application to also be available through a mobile application. This will allow the mobile application to connect to this database instance easily and there is no need to have separate databases for the different applications.

4.1.4 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud.[32] This allows you to have a ubuntu server to host your application, without having to purchase any hardware. It has a free tier which is perfect for this project. It is also essential as it allows the author to easily set any security needs that are necessary to protect the project in production. AWS EC2 supplies you with a region where you server will be hosted , you can also pick where in the world you would like. Once the author purchases a domain, they must set up an Elastic IP.[33] The Elastic IP, once set up to connect to the domains nameservers will then allow the application to be reachable on the internet from the given domain. The EC2 instance hosts the flask server, any code to deploy to project as well as the deep models used for the prediction. It calls a connection to the AWS RDS database too, essential it holds the project together and gives it a platform to be used by the user.

4.2 Component Implementation

In this section we will discuss code implementation and the different sections of code and their functionality. We will begin with the data analysis and cleaning of the code and then move onto the wide & deep model and finally we will look at some main aspects of the applications code.

4.2.1Data Analysis & Cleaning Code

Data analysis and cleaning is the most important part of any project following a Crisp-DM design methodology. It allows the author to identify the problems at hand and allows them to be solved in the best way possible.

The author began the analysis by getting the details and information of the data using Pandas.[34] The author noticed there was no age column which was needed for the prototype, The age was extracted from the title as the title had the year in it. It was achieved like so:

```
years = data.title.str.extract('([1-2][0-9]{3})').astype('float64')
data = data.assign(year = years)
```

```
data['Age'] = 2019 - data['year']
```

Figure 37: Create Age Feature

Next the author had to check the data types within the dataset and there was three data types present.

```
data.get_dtype_counts()
```

```
float64     3
int64       1
object     11
dtype: int64
```

Figure 38: Data Types

The author checked the shape of the data and seen that the dataset has 129971 rows of data which could be used, although a lot of this could be bad data which would have to be cleaned. The author then checked the data for nulls, there was a lot of them.

```
In [128]: data.isnull().sum()

Out[128]: country                     63
          description                  0
          designation              37465
          points                       0
          price                     8996
          province                    63
          region_1                 21247
          region_2                 79460
          taster_name              26244
          taster_twitter_handle    31213
          title                        0
          variety                      1
          winery                       0
          year                      4609
          Age                       4609
          dtype: int64
```

Figure 39: Sum of Nulls

We then checked to see what country has the most wine reviews and which has the least as this would aid in the analysis as we want to remove countries that don't have enough wine reviews.

```
plt.figure(figsize=(14,10))
country = data['country'].value_counts().to_frame()[0:20]
sns.barplot(x= country['country'], y=country.index, data=country, color="re
plt.title('Distribution of Wine Reviews by Top 20 Countries');
```

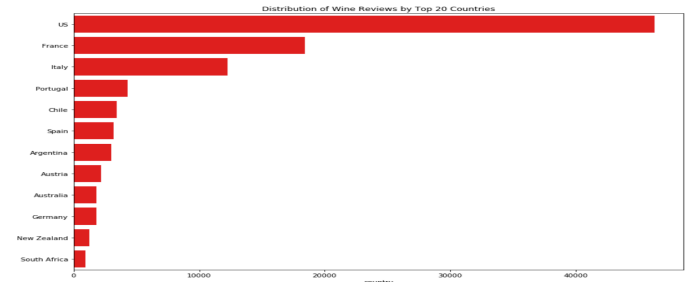


Figure 40: Reviews Per Country

The author then done some more analysis which is explained as comments in the code itself. To clean the data the author removed nulls, removed any country that occurs less than 1000 times in the dataset and also removes any variety that appears less than 650 times in order to get a more accurate result from the model. The description had 8159 duplicate so they were removed. The text within the description also had to be cleaned. The stop words had to be removed, along with any other unnecessary characters, apostrophes for example.

```
new_list =[]
snow = nltk.stem.SnowballStemmer('english')
for description in stop_description:
    description = description.lower()
    cleanr = re.compile('<.*?>')
    description = re.sub(cleanr, ' ', description)
    description = re.sub(r'[?|!|\'|"|#]',r'',description)
    description = re.sub(r'[.|,|)|(|\|/]',r' ',description)

    words = [snow.stem(word) for word in description.split() if word not in stopwords.words('english')]
    new_list.append(words)

stop_description = new_list
```

Figure 41: Cleaning Description Data

Wide & Deep Model Code

The Wide & Deep Model is an interesting aspect to this model as it is still relatively new and not many projects are using it. It gives great results and can be implemented with some ease when you use the 'Keras Functional API'.[35]

To begin the data is split into 80% training data and 20% testing data. This ensures a greater accuracy in the algorithm. Next the author assigned each column/feature being used in the dataset to either training(X) or testing (Y).

```
trainsplit = int(len(data) * .8)
print ("Train size: %d" % trainsplit)
print ("Test size: %d" % (len(data) - trainsplit))


description_x = data['description'][:trainsplit]
variety_x = data['variety'][:trainsplit]
country_x = data['country'][:trainsplit]
points_x = data['points'][:trainsplit]
description_y = data['description'][trainsplit:]
variety_y = data['variety'][trainsplit:]
country_y = data['country'][trainsplit:]
points_y = data['points'][trainsplit:]
```

```
Train size: 79156
Test size: 19790
```

Figure 42: Training & Testing Split

Next a tokenizer was created, which allows the data to be tokenized and in turn, produced a bag of words which is discussed in the research section of this report. The tokenizer is then saved for use later in the application and the descriptions are put with the tokenizer and Keras texts_to_matrix function which creates the bag of words.

```
vocab_size = 20000
tokenize = keras.preprocessing.text.Tokenizer(num_words=vocab_size, char_level=False)
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenize, handle, protocol=pickle.HIGHEST_PROTOCOL)
tokenize.fit_on_texts(description_x)
BOW_x = tokenize.texts_to_matrix(description_x)
BOW_y = tokenize.texts_to_matrix(description_y)
```

Figure 43: Bag of Words Implementation

Example of the bag of words implemented and the difference from the original:

```
print(data['description'][:2])
```

```
34444    ['despit', 'ampl', 'alcohol', 'level', 'soft',...
19737    ['love', 'cool-clim', 'pinot', 'grigio', 'woul...
Name: description, dtype: object
```

```
print(BOW_x[0:2])
```

```
[[0. 1. 1. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]]
```

Figure 44: Bag of Words Difference

A label encoder is then used from the label encoder module. This encodes the data given with numeric values and then one-hot encoding is implemented which creates vectors and gives each country and variety it's own vector. Num_classes is a variable to create the shape which is implemented in the models layers.

```
label_encoding = LabelEncoder()
label_encoding.fit(variety_x)
variety_x = label_encoding.transform(variety_x)
variety_y = label_encoding.transform(variety_y)
num_classes = np.max(variety_x) + 1
label_encoding.fit(country_x)
country_x = label_encoding.transform(country_x)
country_y = label_encoding.transform(country_y)
num_classes_1 = np.max(country_x) + 1
variety_x = keras.utils.to_categorical(variety_x, num_classes)
variety_y = keras.utils.to_categorical(variety_y, num_classes)
country_x = keras.utils.to_categorical(country_x, num_classes_1)
country_y = keras.utils.to_categorical(country_y, num_classes_1)
```

Figure 45: One-Hot Encoding

Difference between varieties after one-hot encoding was implemented. This sllows the model the read the data as it does not perform well with categorical data.

```
print(data['variety'][:7])
```

```
34444          Pinot Noir
19737        Pinot Grigio
27064               Glera
49089          Pinot Noir
17932               Glera
23476           Nebbiolo
27348    Sauvignon Blanc
Name: variety, dtype: object
```

```
print(variety_x[:7])
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
```

Figure 46: One-Hot Encoding Difference

The wide model is then created. It's layers are created by the vocab_size variable that was set earlier. The num_classes variable is used to create the layer of the other two feature inputs. The layers are then concatenated and joined and used with the functional API and the Dense variable to implement the operation being run, the activation variable is the element-wise activation function. The inputs are then set and the outputs. The model is then compiled.

```
BOW = layers.Input(shape=(vocab_size,))
variety = layers.Input(shape=(num_classes,))
country = layers.Input(shape=(num_classes_1,))
joined_layers = layers.concatenate([BOW, variety, country])
joined_layers = layers.Dense(256, activation='relu')(joined_layers)
predictions = layers.Dense(1)(joined_layers)
wide_model = keras.Model(inputs=[BOW, variety, country], outputs=predictions)
```

Figure 47: Wide Model Implementation

The metrics that we are looking for are the 'mae' or mean squared error. If the loss and this are continuously getting lower at each training epoch then the model is performing correctly. Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.[36] This part of the project was a challenge for the author as deep learning is new to the author and it took a lot of time and research to understand.

```
wide_model.compile(loss='mse', optimizer='adam', metrics=['mae'])
print(wide_model.summary())
```

Figure 48: Wide Model Compile

A summary can be viewed which shows us information on the layers of the wide deep model.

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)         (None, 20000)             0
--------------------------------------------------------------------------------------------------
input_2 (InputLayer)         (None, 35)                0
--------------------------------------------------------------------------------------------------
input_3 (InputLayer)         (None, 12)                0
--------------------------------------------------------------------------------------------------
concatenate (Concatenate)    (None, 20047)             0           input_1[0][0]
                                                                   input_2[0][0]
                                                                   input_3[0][0]
--------------------------------------------------------------------------------------------------
dense (Dense)                (None, 256)               5132288     concatenate[0][0]
--------------------------------------------------------------------------------------------------
dense_1 (Dense)              (None, 1)                 257         dense[0][0]
==================================================================================================
Total params: 5,132,545
Trainable params: 5,132,545
Non-trainable params: 0
--------------------------------------------------------------------------------------------------
None
```

Figure 49: Wide Model Summary

Next the description is worked against a word embedding using the functional API. This code pads all of the descriptions sequences to the same length. A word embedding is where words and phrases of the descriptions are matched to vectors of real numbers and involves a mathematical embedding from a space of one-dimension per word to a continuous vector space which much lower dimensions.

```
embedded_x = tokenize.texts_to_sequences(description_x)
embedded_y = tokenize.texts_to_sequences(description_y)

max_seq_length = 170
embedded_x = keras.preprocessing.sequence.pad_sequences(
    embedded_x, maxlen=max_seq_length, padding="post")
embedded_y = keras.preprocessing.sequence.pad_sequences(
    embedded_y, maxlen=max_seq_length, padding="post")
```

Figure 50: Word Embedding Implemented

An example of the description against the word embedding.

```
print(embedded_x[1])
```

```
[ 247  865  797   70  794  370  251 2269 4304 3566   31   36    1   99
   53  216    3  143    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
```

Figure 51: A Description against a Word Embedding

Next the deep model is implemented similarly to the wide model and the two are concatenated to produce the wide & deep model.
The features are then ran against the compiled model and returns a predicted rating. The average difference between original and predicted was used to test the accuracy of the model. The difference was much larger, in the 10-15s, after some more analysis and cleaning for the features the difference dropped to around 1.5.

```
[2] - ['year', 'bottl', 'age', 'wine', 'show', 'ripe', 'fruit', 'tannin', 'well', 'integr', 'combin', 'juici', 'aci
d', 'ripe', 'fruit', 'dri', 'core', 'bring', 'develop', 'wine', 'well']
Predicted:  88.3928 Actual:  88

Average prediction difference:  0.5576343536376953
[3] - ['90-92', 'pack', 'new', 'wood', 'polish', 'wine', 'tannin', 'smooth', 'velvet', 'delici', 'juici', 'red', 'b
erri', 'fruit', 'charact', 'show', 'fine', 'eleg']
Predicted:  93.09297 Actual:  91

Average prediction difference:  1.0808773040771484
[4] - ['one', 'harder', 'grape', 'tame', 'california', 'yet', 'want', 'walk', 'razor', 'blade', 'balanc', 'ripe',
'fresh', 'copi', 'josh', 'jensen', 'version', 'candi', 'nectarin', 'white', 'peach', 'cherimoya', 'jasmin', 'show',
'ripe', 'honey', 'nose', 'palat', 'restrain', 'exuber', 'chamomil', 'sour', 'pear', 'peel', 'sea', 'salt', 'flavo
r']
Predicted:  94.17429 Actual:  92

Average prediction difference:  1.62445068359375
```

Figure 52: Prediction Outcome

4.2.3 Application Code

In this section the author will explain some aspects of the code in the application. The application uses multiple python modules such as flask-login and SQLAlchemy which is an ORM and lets the author communicate with the database without the use of SQL.

The first part of code is the models. These are used to set the database tables and make it easy to create the tables in the database, all that is needed to create these classes and then run db.create_all() in the python interpreter to create the tables in the database. Here the author can set each attribute of the table and set them to ints, varchars etc, also make relations between the tables.

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(40), unique=True, nullable=False)
    image = db.Column(db.String(20), nullable=False, default="default.jpg")
    password = db.Column(db.String(60), nullable=False)
    reviews = db.relationship("Review", backref="author", lazy=True)

    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image}')"


""" This class in the models table is the review table, this sets all the attributes for a review a
    relationship to the user table """


class Review(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    date_posted = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    content = db.Column(db.Text, nullable=False)
    variety = db.Column(db.String(20), nullable=True)
    rating = db.Column(db.Integer(), nullable=True)
    user_id = db.Column(db.Integer, db.ForeignKey("user.id"), nullable=False)

    def __repr__(self):
        return f"Review('{self.title}','{self.date_posted}','{self.variety}','{self.rating}')"
```

Figure 53: Model File Code

The form classes are how we define the forms used in the application. Here is the predict input form and the registration form. The fields of the form are set and the input types and validators are then set too. A validator is a rule that the form must follow. E.g. data is required in a field or the minimum/maximum length of an input.

```
class Register(FlaskForm):
    username = StringField(
        "Username", validators=[DataRequired(), Length(min=2, max=20)]
    )
    email = StringField("Email", validators=[DataRequired(), Email()])
    password = PasswordField("Password", validators=[DataRequired()])
    password_confirm = PasswordField(
        "Confirm Password", validators=[DataRequired(), EqualTo("password")]
    )
    submit = SubmitField("Sign Up")

    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError("Username already taken!!")

    def validate_email(self, email):
        email = User.query.filter_by(email=email.data).first()
        if email:
            raise ValidationError("Username already taken!!")
class PredictData(FlaskForm):
    description = StringField("Description", validators=[DataRequired()])
    variety = StringField("Variety", validators=[DataRequired()])
    country = StringField("Country", validators=[DataRequired()])
    submit = SubmitField("Predict")
```

Figure 54: Form File Code

The __init__ file allows us to set configurations in the application. Our secret key is set to allow authentication, The database connection is configured and other modules are initialised such as bcrypt which is used for hashing users passwords.

```
"""Initialises application and other modules of the application"""
app = Flask(__name__)

""" Secret key was gotten by using a python interpreter and importing the secrets module
    and then using 'secrets.token_hex(16)' to get a 16 byte secret token, this is necessary for the flask application to run"""

app.config["SECRET_KEY"] = "1deef2f0b5b64295bfe291975201d7c9"
app.config[
    "SQLALCHEMY_DATABASE_URI"] = "postgresql:///root:finalyearproject@grapereview.ckmq0nb39mlo.eu-west-1.rds.amazonaws.com/grapereview
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
login_manager.login_view = "login"
login_manager.login_message_category = "info"
""" imports routes """
from wineSite import routes
```

Figure 55: __init__ File Code

The routes file allows us to set our endpoints and make restful API calls to the browser. A route to look at the detail is the home and prediction as this is where the models implemented. The comments below in the figure explain the code thoroughly, The first section of home sets the pagination which allows the reviews to go to another page if there are more than 5 reviews present on a page at once. The next is the user input is taken from the predict form and inputted into a data frame that's passed to the predict function where our deep models implemented. The description input is used with the tokenizer to create the bag of words matrix, the other features are one-hot encoding to create a vector and the model is loaded with its weights and the data run through it and the outputs presented on another page. This section of the application was extremely challenging for the author has the author had ‘hit a wall’ and got confused and stuck at small errors. After two meeting with the author's supervisor who assisted the author with the challenges at hand, they overcame these errors.

```python
@app.route("/", methods=["GET", "POST"])
@app.route("/home", methods=["GET", "POST"])
def home():
    page = request.args.get('page', 1, type=int)
    reviews = Review.query.order_by(Review.date_posted.desc()).paginate(page=page, per_page=5)
    form = PredictData()
    if form.validate_on_submit():
        input_one = form.description.data
        input_two = form.variety.data
        input_three = form.country.data
        """Users inputs to the form are take and inputted into a dataframe, the variable df is made global to be used
        in another function"""
        global df
        df = pd.DataFrame(
            data=[[input_one, input_two, input_three]],
            columns=["description", "variety", "country"],
        )
        return redirect(url_for("predict"))
    return render_template("home.html", reviews=reviews, form=form)
```

```python
@app.route("/predict", methods=["GET"])
def predict():
    """The dataframe from the previous function is take and the inputs are removed"""
    input_one = df["description"]
    input_two = df["variety"]
    input_three = df["country"]
    """The tokeinzer from the beginning of the file is used witht the description input (input_one) then bag of words
        is implemented"""
    tokenize.fit_on_texts(input_one)
    input_one_bow = tokenize.texts_to_matrix(input_one)
    """The variety and country are put through the encoder to be encoded and then one-hot encoded. The num classes is
        defined so that the inputs hae the same shape as the deep model"""
    encoder = LabelEncoder()
    encoder.fit(input_two)
    input_two = encoder.transform(input_two)
    num_classes = 33
    encoder.fit(input_three)
    input_three = encoder.transform(input_three)
    num_classes_1 = 12
    input_two = keras.utils.to_categorical(input_two, num_classes)
    input_three = keras.utils.to_categorical(input_three, num_classes_1)
    """The description input is then used with the keras word embedding and the three inputs and made into an input
        variable"""
    input_one_embedded = tokenize.texts_to_sequences(input_one)
    max_seq_length = 170
    input_one_embedded = keras.preprocessing.sequence.pad_sequences(
        input_one_embedded, maxlen=max_seq_length, padding="post"
    )
```

```python
input_one_embedded = tokenize.texts_to_sequences(input_one)
max_seq_length = 170
input_one_embedded = keras.preprocessing.sequence.pad_sequences(
    input_one_embedded, maxlen=max_seq_length, padding="post"
)
input = [input_one_bow, input_two, input_three] + [input_one_embedded]
"""The graph variable is made global and the model.h5 file is loaded into the application, the graph is defined and
    the models weights are called. The rest of the graph function is called and the inputs are put onto the model to
    be used to return a prediction"""
global graph
model = tf.keras.models.load_model("winesite/h5models/test_model.h5")
graph = tf.get_default_graph()
model.get_weights()
with graph.as_default():
    prediction = model.predict(input)
    prediction = prediction.round(2)
    """The prediction is in the format of a numpy array so it must be changed to a string and then the predict page
        is called"""
    retval = np.array2string(prediction, separator=",")
    print(retval)
return render_template("predict.html", title="Prediction", prediction=retval)
```

Figure 56: Routes File Code (i)

Another route to look at is the new review route. This code shows that a login must be require before the route can be accessed. The form is linked to the review form and the validate on submit function runs when inputs are submitted to the form. This adds the new data inputted by the user to the database and commits it. It then in turn, calls the relevant html page.

```python
@app.route("/review/new", methods=["GET", "POST"])
@login_required
def new_review():
    form = ReviewForm()
    if form.validate_on_submit():
        review = Review(title=form.title.data, content=form.content.data, variety=form.variety.data,
                        rating=form.rating.data, author=current_user)
        db.session.add(review)
        db.session.commit()
        flash("Review Created!", 'success')
        return redirect(url_for('home'))
    return render_template('newreview.html', title='New Review', form=form, legend='New Review')
```

Figure 57: Routes File Code (ii)

There is numerous html files in this application. They are majority standard bootstrapped forms and the navbar. The layout of the page is linked to each html page which is a helpful feature of flask. Flask also allows us to link the form variables to the html forms and relay error messages back to the user also. The div class is the name of the bootstrap class which styles the application. An example:

```
{% extends "layout.html" %}
{% block content %}


<div class="content-section col-centered">
    <form method="POST" action="">
        <legend class="border-bottom mb-4">Input your wine details and get a rating</legend>

        {{ form.hidden_tag() }}
        <div class="form-group">
            {{ form.description.label(class="form-control-label") }}
            {% if form.description.errors %}
            {{ form.description(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
                {% for error in form.description.errors %}
                <span>{{ error }}</span>
                {% endfor %}
            </div>
            {% else %}
            {{ form.description(class="form-control form-control-lg") }}
            {% endif %}
        </div>
        <div class="form-group">
            {{ form.variety.label(class="form-control-label") }}
            {% if form.variety.errors %}
            {{ form.variety(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
                {% for error in form.variety.errors %}
                <span>{{ error }}</span>
                {% endfor %}
```

Figure 58: HTML Example

Code has also been taken from bootstrap such as this modal which allows a dialog box with option to appear allowing the user to confirm the deletion of a review.

```
<!-- Bootstrap Model Code Taken from Bootstrap.com-->
<div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModal" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="deleteModalLabel">Are you sure you want to Delete?</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">×</span>
                </button>
            </div>
            <div class="modal-body">
                You can always make a new one!
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                <form action="{{ url_for('delete_review', review_id=review.id) }}" method="POST">
                    <input class="btn btn-danger" type="submit" value="Delete">
                </form>
            </div>
        </div>
    </div>
</div>
```

Figure 59: Modal Example

The wineries page uses JavaScript to implement the google maps API. To use this the author had to gain access to an API key though googles API platform. From there the author added the key to the layout page and then used the following code to get the map to appear and the markers to bounce at the locations of wineries, the for-loop loops through the markers with are coordinates and then displays them on the map.

```
<h3 class="text-center">Check out the Vineyards spread all over the world!</h3>
<!--The div element for the map, This uses the google maps API to call coordinates of wineries and input them
    through a for loop to show up on the map -->
<div class="content-section" id="map"></div>
<script>

function initMap() {
    var map=document.getElementById('map');

    var markerLocations=[
    [47.2260,4.9687],
    [47.1595,4.9515],
    [47.1591,4.9546],
    [47.2164,4.9660],
    [45.2140,123.1878],
    [47.2085,4.9638],
    [47.2191,4.9588],
    [34.6546,139.1010],
    [34.9063,138.8397],
    [38.4871,122.4104],
    [38.4274,122.3943],
    [38.5027,122.4284],
    [38.4478,122.4120],
    [38.0751,78.2894],
    [38.0153,78.8363],
    [37.8928,8.5568],
    [45.3504,123.1361]
  ];
```

```
  var mapProperties={
     center: new google.maps.LatLng(47.2260,4.9687),
     zoom:2.5
};
    var map=new google.maps.Map(map,mapProperties);
    var j;
    for (j=0 ; j<markerLocations.length ; j++){
        var marker=new google.maps.Marker({
            position:new google.maps.LatLng(markerLocations[j][0],markerLocations[j][1]),
            animation:google.maps.Animation.BOUNCE
        });

        marker.setMap(map);
      }
    }


</script>
```

5.0 System Validation

System Validation is an essential part of any project lifecycle, and it ensures the final product works as intended and it allows for user feedback for suggestions on how to improve the product. It lets us find any issues or bugs before the products deployed to production for everyday users to use. It is one of the most critical aspects of a project. There are numerous ways to test a system. This project includes a review from a domain expert, giving their perspicacity and opinion on whether the product is a product that they would use in their day to day life to assist them with their job in the wine industry. There are also requirements testing which include making sure each requirement works the way it is supposed to, the login and registration, for example, they also test different aspects of the project's functionality and code, these also fall under Blackbox testing. There is usability and users testing which allows the users to test the application functionality from the outside, this is always good as it gets another person's perspective and brings new ideas and changes to the final product. Testing was executed continuously throughout the project cycle, using unit testing as the form of continuous testing. As a component developed it was tested to ensure it worked correctly; errors still tend to slip through which is why other testing methods are essential.

5.1 Domain Expert Review

The author felt it was necessary to get an insight to the final application from the domain expert who kindly let us interview her at the start of the project. Lucy Griffin as stated in the research section is a professional wine taster and has worked on wineries in France for over a year. She has extensive knowledge in this domain and has agreed to complete a short review of the site, the feedback from this review will help the author to gain knowledge of where to improve in the application.

David's GrapeReview.ml is an excellent and unique website. As a professional wine taster I have rarely came across a service like which David has created. It is an intuitive idea. I particularly enjoyed the rating predictor as it is sometimes hard to pick a rating for a wine and this allows you to do just that! I also enjoyed that you can view other user's reviews and ratings. It will serve as a great hub for all wine enthusiasts to share reviews and opinions.

Recommendations I would suggest to improve on would be to have a drop down menu of a list of wine's in the Variety and country section of the website for ease of access. I would also

suggest that on the wineries page you include a heatmap of the vineyard locations to give a better visual aesthetic. I would also suggest that you add a wine marketplace making it easy for wines to be purchased!

I have really enjoyed the service David has provided with his unique website and I wish him all the best in his future endeavours.

5.2 Requirements testing

5.2.1 White Box Testing

Unit testing falls under White Box Testing, also known as Clear Box Testing. It 'is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs.'[37] What is being looked for while testing is being complete is, internal security holes, broken or poorly structured paths in any code, The flow of inputs and the expected output. It is checking pre-determined inputs and trying to find expected outputs from that. There are two main steps to white box testing which fall under these two headings:

1. Understand the source code.

The tester needs to understand how the source code works. Why does ‘that loop’ work with ‘that function.’

2. Create test cases and execute.

When creating the test cases, you have to cover every aspect of the code — all paths (if-loops). You want to cover around 90% of your code to test.

Examples of types of white box testing would be, Unit Testing/Requirements Testing – Usually the first call of action when testing an application. The latter is where each block of code or unit of codes tested. It helps to identify almost all of the bugs early on in your testing which will lead to a better product. Another type would be White Box Penetration Testing – During this test the tester wants to try to attack the code from every angle and aspect of the code to test the applications level of security. The latter will be covered in the form of requirements testing in a future section of this report.

Advantages of White Box Testing are that:

- Hidden errors will be found, and the code can be optimised.
- These types of tests can be easily automated.
- Covers every aspect of your code and every path.

Requirements-based testing[39] is a testing approach where test cases, conditions, and data are created using project requirements. These tests include functional, reliability and usability tests. There are six areas of these tests that will be covered in this project. Establishing test completion criteria, designing test cases, execute tests, verify results, verify the coverage of the tests in regard to aspects of the project and track and manage defects. When analysing

the overall project, it's decided that this type of testing was essential as many components could cause an error and this would help to classify these components before the errors became fatal or they started to show in production. The tests were carried out, and they help to identify small inconsistencies in the application, by the end of the application all components and test cases had passed.

Test scripts were written up and used as a basis for the testing. The test cases were as follows:

5.2.2 Database Connection Test Cases

Testcase	Feature	Test Action	Expected Result	Result
1	Database Connectivity	Connect to database using correct credentials	Successful connection	Pass
2	Insert	Insert a user and review in database	User and Review are visible in the database	Pass
3	Query	Create a query to find information in the database	Data is returned	Pass
4	Update	Create a query to update data in the database	Updated data is in the database	Pass
5	Delete	Create correct delete query	Data will be gone from the database	Pass

Table 1: Database Test Case

5.2.3 Registration/Login Test Cases

Testcase	Feature	Test Action	Expected Result	Result
1	Open application	Open application with the correct URL	Application opens	Pass
2	Open registration page	Navigate to and open the registration page	Registration page opens	Pass
3	Input incorrect returns error	Input the incorrect data types into the registration forms	Error Returned	Pass
4	Input correct returns null	Input the correct data types into the registration forms	Form gives no error	Pass
5	Submit button submits registration form	After inputting correct information, press submit button	Flash message should appear confirming registration	Pass
6	Open login page	Navigate to login page	Login page opens	Pass

7	Input correct login credentials	Input the correct login credentials	Login form should return no error	Pass
8	Input incorrect returns error	Input the incorrect login details	Login form should return error	Pass
9	Submit button submits login form	After inputting correct information, press submit button	Flash message should appear confirming login	Pass

Table 2: Login/Registration Test Case

5.2.4 Create/Update/Delete Review Test Cases

Testcase	Feature	Test Action	Expected Result	Result
1	Open application	Open application with the correct URL	Application opens	Pass
2	Login in as user	Login as user with correct credentials	System logs in user	Pass
3	Open new review page	Navigate to new review page	New review page opens	Pass
4	Insert & Submit Review	Insert data into review form and Submit	Review submits to the database and appears on homepage	Pass
5	View review	Users selects review to view it	Review Opens	Pass
6	Update Review	Click update button and enter new review details and submit	Review should update	Pass
7	Delete Review	Select delete button, confirm the deletion	Review should be deleted from application	Pass

Table 3: Create/Update/Delete Review Test Cases

5.2.5 Generate Prediction Test Cases

Testcase	Feature	Test Action	Expected Result	Result
1	Open application	Open application with the correct URL	Application opens	Pass
2	Enter data	Enter the data that will be used to make a prediction	Prediction form should return no error	Pass
3	Predict	Select the predict button	Prediction data should submit and be processed from backend	Pass
4	View prediction	Verify a prediction has been returned	Prediction should be on prediction page	Pass

Table 4: Prediction Test Cases

5.3 Usability & User Tests

5.3.1 Black Box Testing

These tests fall under the category of Black Box Testing, also known as Behavioural Testing. The latter is a software testing method in which the internal structure/design/implementation of the item tested is unknown to the tester. These tests can be functional or non-functional, though usually functional. This type of testing tests for how the application reacts to user inputs, interface errors, errors in the database and termination errors. The tester can test the input and the output without knowing about the internal structure of the application. [38] The steps to black box testing are:

1. Functional Testing
2. Non-Functional Testing
3. Regression Testing

To perform black box testing the tester chooses valid inputs and checks if the application processes them correctly and compares the output to the desired output.

To test the usability, functionality, and design of the application, the authored prepared a survey which questions relating to the application. The author also provided a link to a deployed test application on amazon web services with an accessible URL. The overall feedback was excellent as we can see in the figures below. There was some positive feedback along with some constructive criticism. This section of testing was very worthwhile as any errors that arose the test users could quickly inform the author, which resulted in faster fixes. For example, a user could not log in as the database had read-only permissions, an easily fixable issue whereby changing the permissions of the database allowed access. The deep model did not load which led to an error for another user, but they informed the author, and it was easily fixable allowing them to continue with their testing.

The survey given to the users involved seven questions, which the users answered through a google form and the results came back anonymously to the author.

Question One showed the author that the demographics of the testers suited the project, 88.5% were occasional wine drinkers which meant they might find some use from the application.

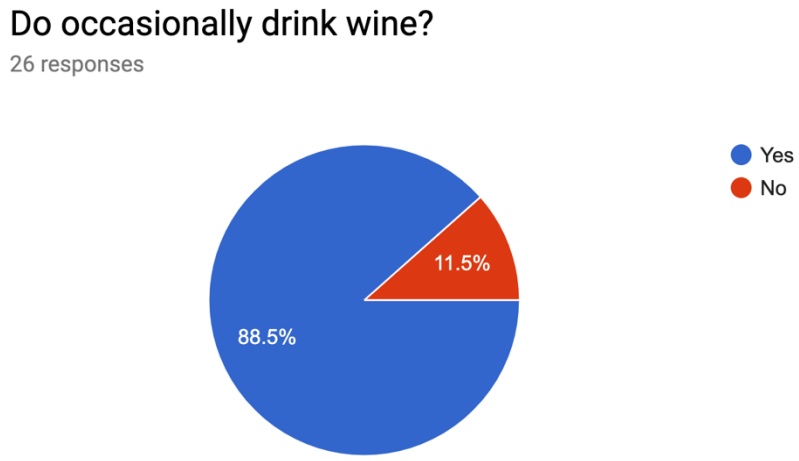


Figure 60: Survey Question One

Question Two showed that 80.8% of the users would use an application like GrapeReview.

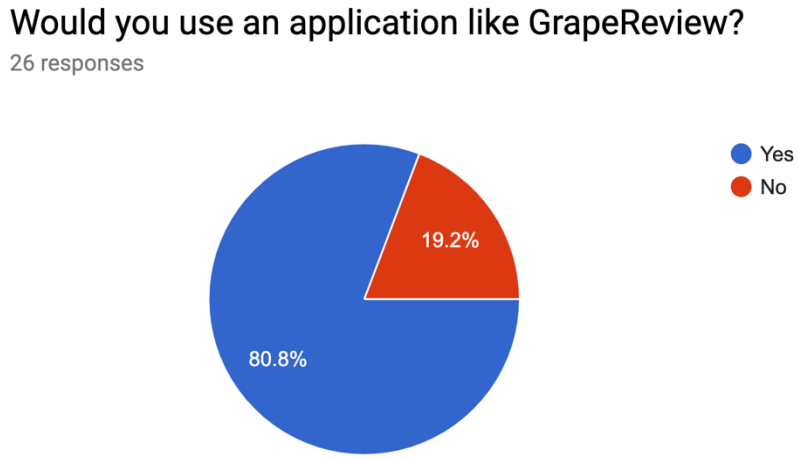


Figure 61: Survey Question Two

Question Three gave some interesting feedback in regards to the applications design. The author had thoughts that the users would find the design different to the 'social norm' of websites today but the feedback was positive.

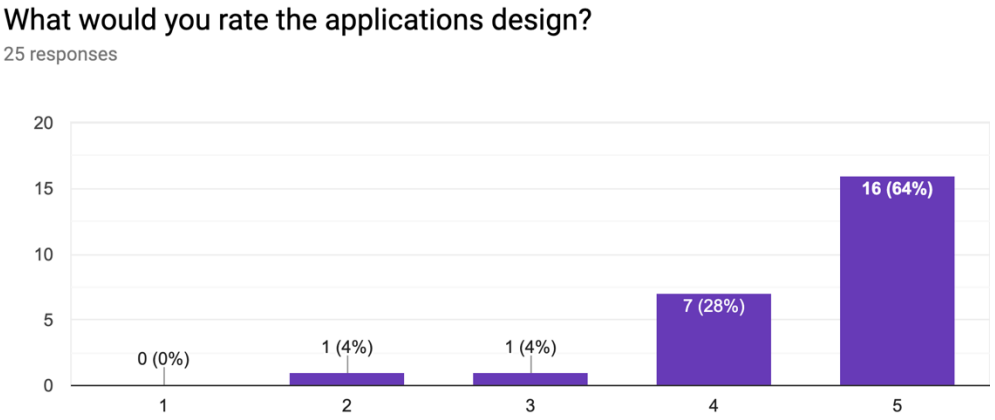


Figure 62: Survey Question Three

Question Four involved asking the user the rate the functionality of the application and the overall feedback was good for this.

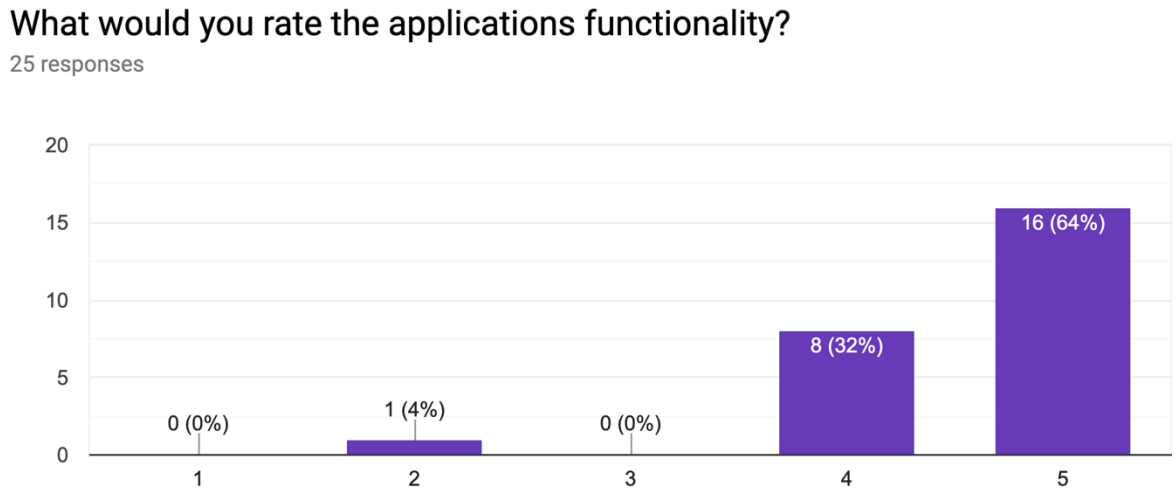


Figure 63: Survey Question Four

Question Five gave the users a chance to give their opinion on changes that could be made to improve the application. Very good answered came back for example, one user suggested extending the application to include other beverages other than wine. Another suggested having a drop down list to select wines rather than typing your variety of wine in yourself. A User also suggested implemented a description predictor which is something that would be difficult but will be considered in future work. A user also wanted to save their favourite wines and also to center the review content to the middle of the screen. All great ideas, especially the centering of the reviews feed. At the moment it is positioned slightly to the left, this is a bootstrap design which allows it to refit on mobile devices.

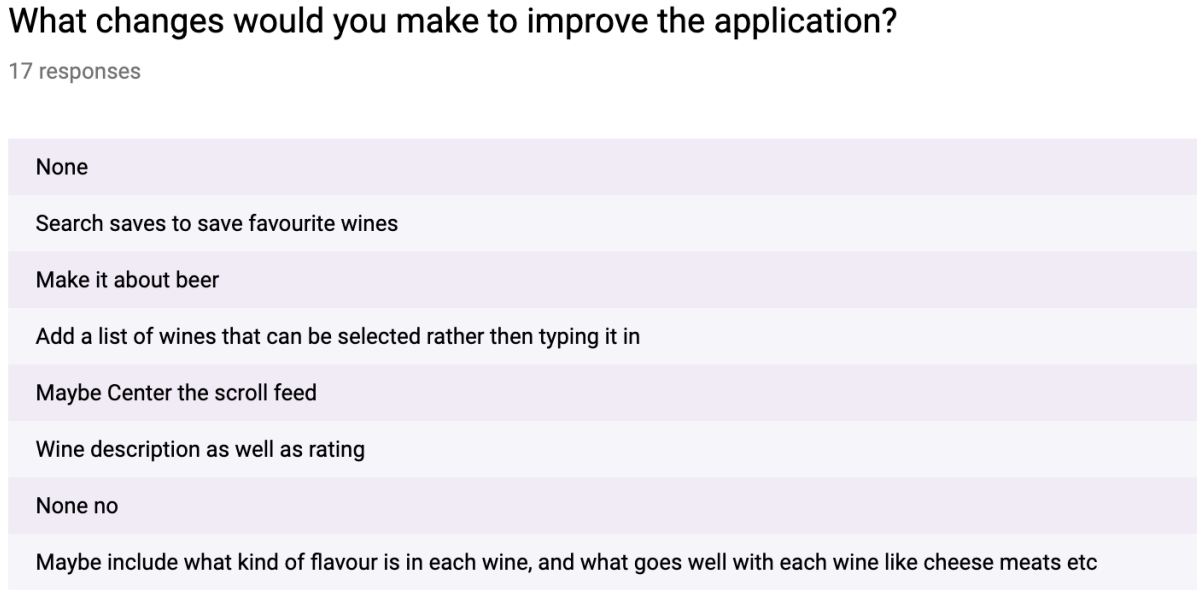


Figure 64: Survey Question 5

Question Six asked which features would you change/add/improve in the application and two users suggested to widen the selection of beverages available to predict and store and another suggested password constraints which is also a great idea that will be implemented in the future plans.

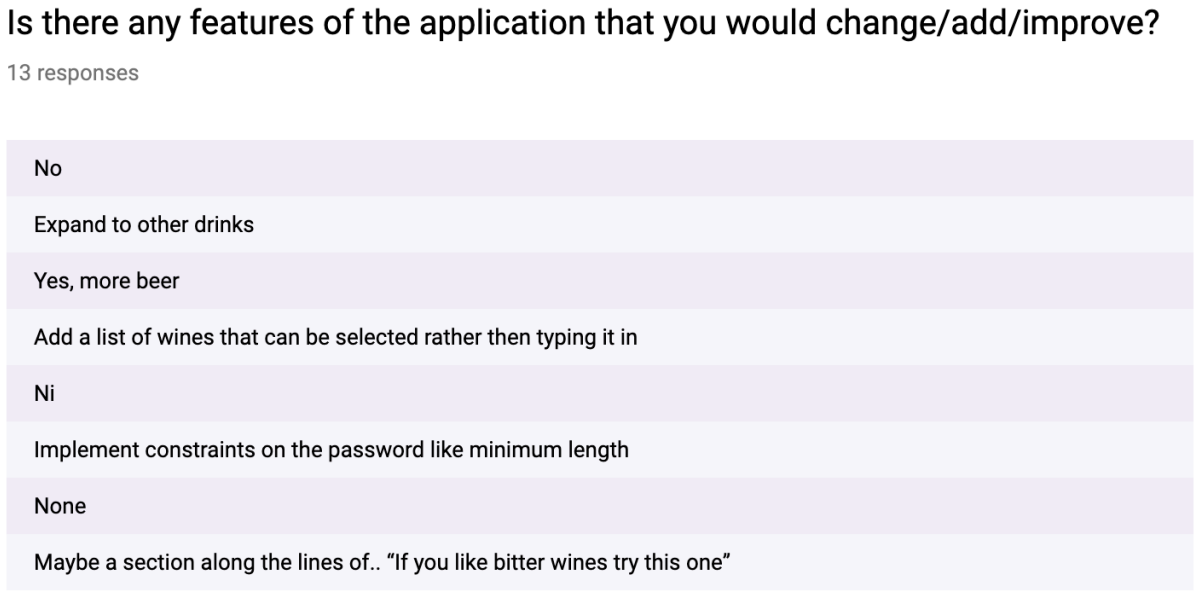


Figure 65: Survey Question Six

Question 7 involved asking the users do they have any further thoughts on the application and all users said they did not.

This section of testing gave the author great ideas to improve the application in the future as well identifying problems which arouse during testing.

6.0 Project Plan

The project plan followed the agile methodology as that is the new standard when it comes to technology design in the field. Rightly so as it allows for deadlines to move and features to change along with the project, it grants more leeway when it comes to development. The plan laid out for the project was followed well; deadlines achieved with a few roadblocks along the way. Something that became an exciting challenge towards the end of the project was time constraints. The author was keen to implement a cross-platform Xamarin application to work alongside the web application, but sadly, the requirement fell short and was unable to be completed within the project life-cycle. In this section, we talk through some of the main points of the projects plan as well as look at the future plans of the project.

6.1 Prototype

The prototype is an integral part of a project as it gives an idea of the direction of a project. It shows whether the proposed approach is set to succeed or fail. Prototyping reduces the overall time spent developing the project as you realise what works and what doesn't. It is easier to identify potential problems and what changes need to take effect before beginning the product development.

In this project, a prototype was created using the K-nearest neighbors Classifier as well as Dash and Flask. When reviewing the results of the prototype classifier, the author sub sequentially discovered that KNearestNeighbors was not an ideal approach. K-nearest neighbors only used three features of the dataset, The Age and Points/Rating, and the Price. These were used to try and predict the price. While the model was giving a prediction, the outcome cannot be trusted as the predicted compared to the original was off by a large number.

	points	Age	price	Predictions
0	87	6	35	20.0
1	87	8	15	48.0
2	87	6	14	11.0
3	87	6	13	13.0
4	87	7	65	13.0

Figure 66: KNN Comparison

In Figure 43, you can see the first 5 predictions made under the 'Predictions' column. We can compare these to the 'price' column which is the actual price of the wine given its rating/points and age. You can see that the predictive algorithm is notably off from the tangible result. Only predicting correctly once in this example, with all other predictions being +10 away from the actual price except for one row. The latter was the main decision to use different technologies, along with different features and try to predict a rating rather than a price.

6.2 Original to Current Plan

The original plan of the project had been produced before the interim, during the research phase. Although some technologies seemed like a perfect fit for the project, some technologies changed throughout the project, as time went on, it was determined through implementation that some technologies work better with others. Flask, for example, is more commonly used when deploying Machine learning models as it is lightweight and efficient.

Some changes were made, like changing from Django to Flask. MySQL local database changed to an Amazon Web Services RDS PostgreSQL instance. The reason for this change is that the author discovered that when attempting to establish and configure a MySQL database server on AWS, MySQL would not work well with the newer python versions. There were longwinded ways around this, but PostgreSQL was a better fit after some more in-depth research. That is when the decision was made to migrate the data on the SQLite3 test database to the AWS Postgres instance. It was an excellent choice as everything fell into place nicely and worked well together. Another change was the decision to use a deep learning model instead of a standard machine learning model such as Naïve Bayes. This decision was because after some tests of both of these models on the dataset it was discovered that the deep learning model performed much better than Naïve Bayes. As a result, the decision was made to go ahead with the deep learning model.

The author had initially been planned to predict the price of a wine, after careful consideration, tests and research it was decided to predict a wine rating. From the research conducted with the domain expert and other research done by the author, it was clear that the price of a wine is too complicated to predict. There are so many varying factors that decide a wines price ranging from the soil the grapes grown in, to the appellation laws of a wine region.

The author had also planned on having a heat map of some wineries across a world map, this was not possible due to time constraints but chose to still have the map just with markers of the wineries rather than a heat map. The concluding change to the original plan was the cross-platform mobile application. This decision also came down to time restrictions, which the author did not take lightly, it was an exciting aspect of the project that the author was keen to complete if there was time, but time did not allow it. Luckily it was decided that this can

be implemented at a future point in time, post completion of the project. Figure 44 is where you can see the original plan of the project and Figure 45 is the updated final plan.

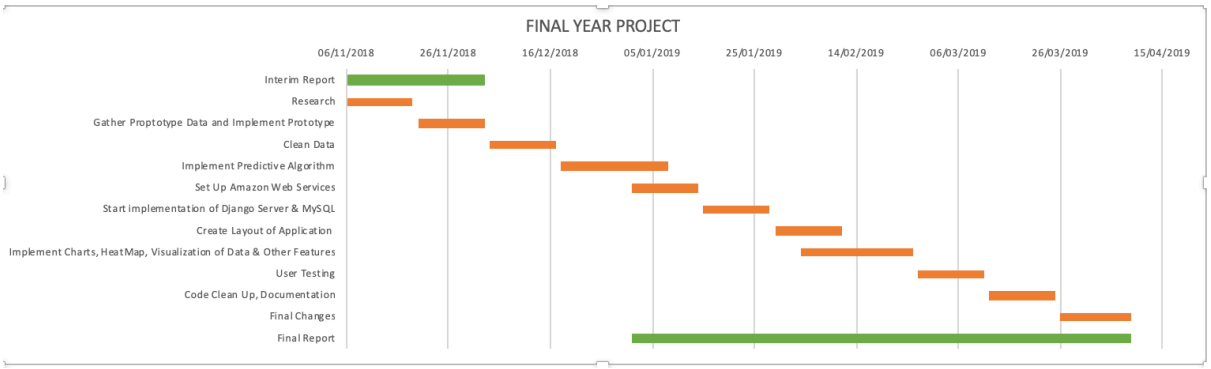


Figure 67: Original Gantt Chart

The changes to the plan happened for the reasons stated above, as well as the exams in January, and the author underestimated the workload for the exams, so the project fell on the back burner for two weeks during exam season. Nevertheless, work hastily continued as proposed and quickly started to retake shape. The current project plan below in Figure 27 shows the plan has shifted from the original.

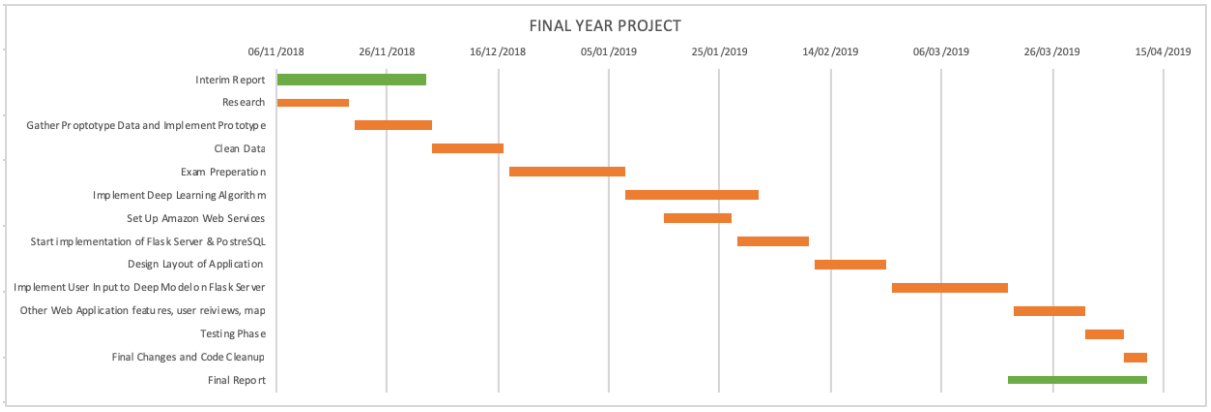


Figure 68: Current Gantt Chart

From what you can see there are some considerably changes regarding technologies and timeframes. It was great working in an agile way following the agile methodology because it gave some flexibility to the project. The planned two-week sprints could change to three weeks or one week depending on how challenges the current aspect of the project was. Some aspects of the project were practically challenging, for example as you can see the implementation of the deep learning algorithm was challenging and took longer to complete than other aspects as there was a lot of study involved, not just implementation of the code. The author had to understand the concepts and how they worked with the dataset and what could possible outcomes be. A great way of learning these was through the weekly meeting with the project's supervisor trying to explain my understanding of the aspects of the project and if some points sounded not fully understood the supervisor was there to help and advise what needs to be improved. Another aspect of the project which took more time than expected was implementing the deep model onto the flask server so that users could input

data and retrieve a prediction from the model. The author hit a roadblock as they struggled to try to implement it, getting stuck on minor syntax errors and trying to understand more how it could work when used in a flask server on a web application. The latter is another time where help from the authors' supervisor was greatly appreciated as it helped the author to see the problem at hand from another person's perspective, particularly someone who had a distinct experience with coding and problem solving and it also helped to see the issues at hand from a different perspective. Once that problem was solved the author was able to get more into the flow of the rest of the development of the web application and was able to implement the other components with a bit more ease.

6.3 Future Plan

The plan for the future of this project is bright as it is a project that the author has loved. The feedback from the user tests was excellent and also had some great ideas for future implementations. For example, in the future, the author hopes to expand the web application to include more than just wines. The application can include other beverages of all kinds, with more data which can be obtained through scraping and datasets, this is an aspect which excites the author as the application could grow to be something that could be used worldwide in the future. Another component of the projects future plan is the cross-platform application built using Xamarin. The latter is a concept the author is excited to dive into after the projects completed so that the author can use the Xamarin application to grow the overall concept and idea of this project. The mobile application is an avenue that will lead to multiple sources of expansion. The author proposes to implement a barcode scanning system where the user could scan their wine and load its information into the application which could then be used to predict a rating or to share and review with others. Another feature would be a friends list for users to interact with people they know. It also it allows companies in the hotel industry to be able to source high-quality wines straight from the wineries through the app, which leads to another future plan to implement a new section of the app which allows users to buy wines directly from wineries that supply wines and sell through the app. It could benefit multiple industries in the future if the projects expanded to these areas. Other aspects of future plans are to gather more data and improve the prediction model as deep learning models perform best with more and more data. Doing more with this side of the project will grow the authors' skills even more in a sector or industry in which they are looking to enter after the completion of their studies.

7.0 Conclusion

The final year project was a mix of experience and understanding. The author has gained vast knowledge and expanded their views on the subject of data analysis, deep learning, web applications, and general computer science. The author has used everything that they have learned over the past four years of my computer science course, and they have used that knowledge and experience to create a project that they can call my their own. The project has thought the author how to think, how to reason, how to problem solve, understand and ask for help when they need it. It has thought the author how to think on their own and how to think in different ways. It has thought them how to approach a problem and how to handle

that problem. To conclude this report, we look at the closing notes of each chapter as well as the personal thoughts of the author of the overall report.

The research that had been carried out in chapter two was extensive and in depth. The domain expert review gave a fascinating insight into the world of wine production. It thought us critical factors about the productions of wine and how features of wines are affected by variables either natural or otherwise. It also showed us how we could use this knowledge when designing our system. We looked at similar studies, technologies, and solutions. We also look at the research of technologies that could have been used and that were used in the project and go into depth into the technologies related to data science, natural language processing and wide & deep learning. Personally, I feel this chapter gave us the knowledge that was needed to complete the project.

The design methodologies in chapter three gave the project a clear design and structure. It allowed the author to tackle the challenges at hand in a professional and thought out fashion. We looked at the design of the database which clearly explained what's involved in creating the database and hosting it on an Amazon Web Services RDS instance. We also explain the source code layout and what it contains. We explained the reasons for the user interface designs and what the prototype looked like and how it had changed. We then discussed a full application use case that showed how a user could use the application. This section relayed the information that we needed to understand the design.

The next section was the architecture and development. This section took a look at the technical architecture chosen and the reasoning for it. It clearly explained what technologies the author had chosen and why, and it showed how they all work together. I feel that the architecture of this project is one of the principal successful components. Everything worked well together and flowed perfectly, connected as one to create an architecturally clean system. Next, we talked about the implementation of coding components. We discussed the ins and outs of some of the main parts of the code from the data analysis and cleaning, to the implementation of the wide & deep model and the code to create the web app using restful API calls to create endpoints for people to access. This section successfully explains the relevant information, and I feel that it shows the complexity thats involved in researching and executing this project.

The next section was chapter five, explained the processes that were involved in testing the application. It started with a review and analysis of the application by the same domain expert as seen in section two of the projects researched. It was great to get the opinion and suggestions from someone who uses systems like this daily with the explanation of white box testing and stated the functional requirement testing that had taken place to ensure the application was working correctly. The next testing approach walked through was the black box testing and usability testing. This section gave a great insight into the perspective of a user that doesn't understand how the underlying project works on the inside. However, it is perfect for testing how it works and functions from a user's perspective. The feedback was positive, and all comments and suggestions acknowledged for future reference. The testing

was very triumphant overall, I would add in more unit testing if time allowed it, but overall, I was happy with how this section turned out.

I have learned how to complete a project from start to end. Including the research, planning, prototyping, trying new things, failing at some and succeeding at some. I have learned how to set goals and how to reach them realistically. The project started as several different ideas, and it grew into one. There were many challenges along the way, some I overcame, for example, teaching myself how deep learning works and how to implement it into web applications. How to set up an Amazon Web Services server instance and how to create a domain and host the code that I have written to it for everybody to see. There were also some challenges that I was not able to overcome such as the implementation of the Xamarin app, which I will take as a challenge that I have still learned from as it gives me great achievement to try to continue to overcome these challenges in the future. The project started as an idea in my head, which turned to an idea on paper and from there I began to work at it, trying to tackle it from all angles smartly and efficiently, getting the advice and guidance from my supervisor along the way.

I feel that the project has achieved its overall goal. It created a platform where people can share their reviews of wines and also a place where deep learnings integrated into an industry where you would not think it would be. This project opens the doors to a world of ideas and concepts that I hope to continue to work on in the future.

8.0 Appendixes

1. Access the survey here:
https://docs.google.com/forms/d/e/1FAIpQLSfSOYpnX-GKCbWEe4Lu0fqnMQgWqwmvOA2LX0ga-Q8dr_OgHQ/viewform?usp=sf_link
2. Video demo of application here:
https://drive.google.com/file/d/1AgGAakFurt9rPMXxCOztFL2WEhz5zL1-/view?usp=sharing

9.0 BIBLIOGRAPHY

[1] "Data Analysis." [Online]. Available: https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/datopic.html. [Accessed: 03-Apr-2019].
[2] "Data is the New Oil," *ANA Marketing Maestros*. [Online]. Available: https://ana.blogs.com/maestros/2006/11/data_is_the_new.html. [Accessed: 03-Apr-2019].
[3] "Welcome to Python.org," *Python.org*. [Online]. Available: https://www.python.org/. [Accessed: 03-Apr-2019].
[4] "TensorFlow." [Online]. Available: https://www.tensorflow.org/. [Accessed: 03-Apr-2019].
[5] J. D. Kelleher, B. M. Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT Press, 2015.
[6] J. Robinson and J. Harding, Eds., *The Oxford Companion to Wine*. Oxford University Press, 2015.
[7] A. Kassam and N. Davis, "Evidence of world's earliest winemaking uncovered by archaeologists," *The Guardian*, 13-Nov-2017.
[8] Y. Lun, K. Gu, and S. Yang, "Predicting Wine Points using sentiment analysis," p. 7.
[9] B. Chen, V. Velchev, J. Palmor, and T. Atkinson, "Wineinformatics: A Quantitative Analysis of Wine Reviewers."
[10] M. Bowles, *Machine Learning in Python: Essential Techniques for Predictive Analysis*. John Wiley & Sons, 2015.
[11] "Wine Enthusiast Magazine | Wine Ratings, Wine News, Recipe Pairings," *Wine Enthusiast Magazine*. [Online]. Available: https://www.winemag.com/. [Accessed: 30-Nov-2018].
[12] "Kaggle: Your Home for Data Science." [Online]. Available: https://www.kaggle.com/. [Accessed: 30-Nov-2018].
[13] "Bitter Grapes," *Bitter Grapes*. [Online]. Available: http://www.bittergrapes.net/. [Accessed: 04-Dec-2018].
[14] "Vivino.com - Find and buy wine in seconds." [Online]. Available: https://www.vivino.com/. [Accessed: 04-Dec-2018].
[15] M. LTD, "Mintec Global." [Online]. Available: https://www.mintecglobal.com. [Accessed: 01-Dec-2018].
[16] "R: What is R?" [Online]. Available: https://www.r-project.org/about.html. [Accessed: 27-Nov-2018].
[17] "Java Programming Language." [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html. [Accessed: 27-Nov-2018].
[18] "What Is MongoDB? | MongoDB." [Online]. Available: https://www.mongodb.com/what-is-mongodb. [Accessed: 27-Nov-2018].
[19] "MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?" [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. [Accessed: 28-Nov-2018].

[20] "PostgreSQL: The world's most advanced open source database." [Online]. Available: https://www.postgresql.org/. [Accessed: 08-Apr-2019].
[21] "pandas: powerful Python data analysis toolkit — pandas 0.24.2 documentation." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/. [Accessed: 07-Apr-2019].
[22] "What is NumPy? — NumPy v1.13 Manual." [Online]. Available: https://docs.scipy.org/doc/numpy-1.13.0/user/whatisnumpy.html. [Accessed: 07-Apr-2019].
[23] "Matplotlib: Python plotting — Matplotlib 3.0.3 documentation." [Online]. Available: https://matplotlib.org/. [Accessed: 07-Apr-2019].
[24] "scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation." [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 07-Apr-2019].
[25] "Naive Bayesian." [Online]. Available: https://www.saedsayad.com/naive_bayesian.htm. [Accessed: 02-Dec-2018].
[26] H.-T. Cheng *et al.*, "Wide & Deep Learning for Recommender Systems," *ArXiv160607792 Cs Stat*, Jun. 2016.
[27] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed. O'Reilly Media, Inc., 2009.
[28] "Hierarchical Data Formats - What is HDF5? | NSF NEON | Open Data to Understand our Ecosystems." [Online]. Available: https://www.neonscience.org/about-hdf5. [Accessed: 08-Apr-2019].
[29] "SQLAlchemy - The Database Toolkit for Python." [Online]. Available: https://www.sqlalchemy.org/. [Accessed: 08-Apr-2019].
[30] M. O. contributors Jacob Thornton, and Bootstrap, "Bootstrap." [Online]. Available: https://getbootstrap.com/. [Accessed: 10-Apr-2019].
[31] J. F.- js.foundation, "jQuery." .
[32] "What Is Amazon EC2? - Amazon Elastic Compute Cloud." [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. [Accessed: 09-Apr-2019].
[33] "Elastic IP Addresses - Amazon Elastic Compute Cloud." [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html. [Accessed: 09-Apr-2019].
[34] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2017.
[35] "Guide to the Functional API - Keras Documentation." [Online]. Available: https://keras.io/getting-started/functional-api-guide/. [Accessed: 12-Apr-2019].
[36] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," *Machine Learning Mastery*, 02-Jul-2017. .
[37] "White Box Testing," *Software Testing Fundamentals*, 19-Dec-2010. .
[38] "Black Box Testing," *Software Testing Fundamentals*, 19-Dec-2017. .
[39] P. Skokovic and M. Rakic-Skokovic, "Requirements-based testing process in practice," *Int. J. Ind. Eng. Manag.*, vol. 1, pp. 155–161, Jan. 2010.