NEA – MELANOMA DETECTION

Abhiram Vinjamuri

RAM CORP

Table of Contents

Analysis

Problem Identification

At this moment in time when a patient has skin cancer or a suspected lesion on their skin, they have to wait an average of 18 weeks and general practitioners' sensitivity is 61% to 66% however a computational solution has much more potential [*can be filled into a quantified value when done].* Furthermore, with statistics that over 1 in 5 people will develop skin cancer and in this current pandemic these waiting queues are only getting longer. Additionally, most of these cases are false calls with many just being ordinary moles on people's skin. This means that my solution of an instant and quick check online would be in high demand where it can check how likely someone is to have a cancer and then check if the skin cancer is potentially fatal.

There are several features explained below which the problem needs to have in order to be solvable by computational methods.

Stakeholders

There are many different stakeholders in this project, but the main target stakeholders are the general population to use for free so that everyone can get checked to help reduce the risk of people having dangerous skin tumours. This would mainly be used by anyone who suspects that they may have skin cancer or are just worried because they have a lesion on their skin.

However certain demographics of people will be more at risk of getting skin cancers so they will be targeted more. For example, individuals who are above 50 years old have an exponentially larger risk of developing skin cancer therefore they should be targeted, and the system should be built in a way that is suitable for them to use as they may have certain hardware and software limitations.

Furthermore, it would also be extremely important to collaborate with medical professionals in many ways. Therefore, doctors would also be stakeholders because if someone is deemed to have a skin tumour it is important for the system to have some way of liaising with the patient's general practitioner given their permission for next steps.

These practitioners who normally handle these patients in real life will need to be worked with to make sure that we provide a professional service that can give a personalised that is fitting of their standards

Finally, in areas where the GP surgeries have extremely high waiting lists, we would collaborate with the GP surgeries to offer an initial check therefore the surgeries would also be important stakeholders for the system.

Why is it suited to a computational solution?

The users will have a website interface to interact with and will need to input some medical details as well as a picture of the lesion for the program to use. The end-users will need to enter some details into a website and an initial analysis will take place to advise them on next steps and give them an idea of if their skin lesion is a cancer or not using a machine learning file. I will be using a neural network which will use both medical histories along with the image to build a model for predicting if the user has a tumour or not. Furthermore, the neural network will be using pattern recognition within the data along with in the images to look for certain features which all the images that have tumours have in common to successfully create a model where another image can be inputted and if similar features appear then the cancer will be considered identified. The neural network will be able to identify patterns which cannot be seen to the naked eye allowing for an extremely efficient system solved by computational methods.

Computational methods that the solution lends itself to:

Problem Recognition

The problem of skin cancer identification has two ways of being done, the initial is by experts however, the waiting times are too large so the second is by using computational methods. This is how I propose to use a machine learning algorithm which is able to make a predicative model using vast amounts of data which have been published on the topic. This offers a principled approach for developing sophisticated, automatic and an objective algorithm for analysis of high-dimensional and multimodal biomedical data to predict if someone has a skin cancer or not

Problem Decomposition

The machine learning algorithms are made up of three parts:

- The computational algorithm at the core of making determinations either neural network or support vector machine.
- Variables and features that make up the decision
- Base knowledge for which the answer is known that enables (trains) the system to learn.

  Using all these I will be able to make a model for if people have a skin cancer or not along with its potential severity. Therefore, this problem is clearly solvable by computational methods because since the problem is clearly defined and the machine learning algorithm can easily be decomposed.

Abstraction

The system created will be fed some data about a patient and will need to use a model in order to detect how likely it is that the patient will have a skin tumour or not. The input of this will be some medical data and a picture of the patient’s skin. The output of this program would be the chance however the system should also be able to communicate this to doctors upon patient request.

Data Requirements

The final conditional for making sure that a problem is solvable by computational methods is to identify what data is needed and if the storage capacity is available. For my solution, I will need access to large datasets however, the maximum size is within [*insert value when solved*].

Altogether since the system has all the characteristics of being solved by computational methods such as being able to be abstracted and decomposed as well as having realistic data and processing requirements and is clearly defined it is well suited to a computational solution.

In terms of the database choosing we will need to consider many different factors such as how many different types of lesions have been analysed as well as is it has been based on all skin colour or jus one specific population which will cause a large database bias, finally the number of items in the database and the proportion of each item will also need to be greatly looked into.

Interview

Interview Questions

Medics

My questions are for a medical student and a doctor as doctors are one of the main stakeholders of my product since doctors currently have to deal with many patients that have skin cancer

1. In your opinion do you think that skin cancer is a major issue for the general population?
2. Do you think that an online service to detect skin cancers would be viable?
3. How confident do you think patients would be using this service?
4. Do you think that an online check-up service will be able to reduce workloads for GPs?
5. Do you have any concerns with a system such as this?
6. Do you have any additional comments to add?

Question one has been written to see what the doctors overall think of skin cancer and to gauge how dangerous this cancer is to the population compared to other diseases and cancers

Question two is a very general question which will allow us to understand what medical professional's initial opinions of a system like this is allowing us to understand what they may expect out of a system like this.

Question three covers a problem that I had identified, and it will help to show me how much of a problem this is and start to think of other ways that we can get patients to interact with the system.

Question four covers one of the main objectives in the long term and will help me to understand how viable this sort of aim would be and the potential that the product has.

Finally question five and six cover any potential concerns that they may have with the system from their initial view along with any additional statements they would like to add about my system.

Potential Patients

These questions are aimed at the general population to see what their opinions of a system like this is and to see how likely they are to use the system.

1. Do you think that you are at risk of getting skin cancer?
2. Would you check yourself if you found out that 100,000 cases of skin cancer are diagnosed yearly in the UK?
3. Would you be willing to check any lesions on a website to help keep yourself safe?
4. Would you be able to upload an image on a website?
5. Do you think that a system like this should anonymised or have an anonymous mode?
6. Do you have any potential concerns about this system?
7. Do you have anything else to add?

Question one and two both act to see what the general population's opinions are of skin cancer, trying to find out how at risk they think they were.

Question three and four aim to see how comfortable the patients would be using a website as this could be a cause for concern with people not trusting web applications.

Question five is aimed to find out what sort of privacy modes the patients require and what levels they would think are suitable.

Question six and seven aim to find out any potential concerns the patients may have as well as any points that thy would like to mention that they did not answer in any of the question sections.

Interview

Medics

Ankita – Medical Student

1. **In your opinion do you think that skin cancer is a major issue for the general population?**

   "Yes, it is a condition that is very dangerous however is preventable therefore although it is major killer, I do not think that it should be. It can be caused by small things like sunbeds and going out in the sun without sun cream or any skin protection and can be treated effectively if identified early".

2. **Do you think that an online service to detect skin cancers would be viable?**

   "Possibly, in times like corona very good for online for people who cannot book appointments, however, many people will not have access to the relevant technologies to use a product like these although it may not be able to work as a standalone system for the entire population e.g., elderly who may not be so tech-savvy".

3. **How confident do you think patients would be using this service?**

   "Depends on how easy it is to use; will the public be able to use it and trust it? Does everyone think that skin cancer is a problem for them with most people only getting worried when it is too late therefore may want an actual consultation! Additionally, how would it be advertised to the public?"

4. **Do you think that an online check-up service will be able to reduce workloads for GPs?**

   "Potentially when the system is well developed and trusted by the public with positive reviews, being trusted and people knowing that it works".

5. **Do you have any concerns with a system such as this?**

   "Information security are a major risk, and you have to be transparent about how you will use the data collected from patients and if you need to store it prevent it from being hackable and patient's data being stolen. Furthermore, what is the misdiagnosis potential and are there any back up checks to keep making sure that the system is working."

6. **Do you have any additional comments to add?**

"I think that this seem like a good idea for the future, and I am definitely interested in how the finished product will turn out."

Dr Indira - Doctor

1. **In your opinion do you think that skin cancer is a major issue for the general population?**

   "Yes, it is a major cause for concern for people across the country!"

2. **Do you think that an online service to detect skin cancers would be viable?**

   "Of course, it would be useful, but I would worry about false positives and false negatives, I would wonder if the general population would get more worried than not? It is imperative we don't miss out on a real malignant lesion therefore the system would have to be quite accurate."

3. **How confident do you think patients would be using this service?**

   "They might use it willingly, however, we need to be mindful of how many have access to technology and whether they might be able to interpret results as suggested. Those who have the gadgets to use this and knowledgeable would benefit."

4. **Do you think that an online check-up service will be able to reduce workloads for GPs?**

   "It is hard to say whether it would reduce workload or cause more concern and it would be imperative to test it on a small population first."

5. **Do you have any concerns with a system such as this?**

   "It is a great App, user friendly and simple instructions. The main concern would be availability of technology, and interpretation of results. It is unclear what patient related info will be collected and if confidentiality will be breached."

6. **Do you have any additional comments to add?**
   "Only a good look at the product will allow a good understanding of how it will work. It is a great idea and if it works, will definitely benefit the general population."

Analysis

From these professional opinions the first thing that is clear is that skin cancer is a serious issue for lots of people. Upon initial opinions they both agreed that a system like the one I proposed would be useful however, the problem of false positives was raised. This will mean that it is essential for me to finetune the machine learning algorithm carefully as well as provide contact to additional people such as GPs for people to be confirmed about a positive diagnosis.

They also mentioned that a product like this has lots of potential but must be easy to use because not every patient will be tech-savvy and therefore may not be confident on a website platform. Finally, Ankita also noted about security risks and that the patient's data needs to be well protected to stop hackers having easy access to patient's data on the website.

Potential Patients

Subil Phillips – Person at risk

1. **Do you think that you are at risk of getting skin cancer?**

"No, I do not think that I have a very great risk since people with darker skin tones are less likely to get skin cancer and I am of Indian origin therefore I feel quite safe. Additionally, I do not spend that much time in the sun therefore feel my overall risk is low."

2. **What do you think about 100,000 cases of skin cancer are diagnosed yearly in the UK?**

"This is a surprisingly large figure and makes me feel that makes me more vulnerable and that almost anyone could be getting skin cancer can happen to anyone, it also makes me feel like I may need to get tested in the future to keep safe."

3. **Would you be willing to check any lesions on a website to help keep yourself safe?**

"Personally, I feel like if it were to be a small mole on the skin that I only have suspicions about I would be able to trust a website. However, if it was something which I was seriously concerned about I would be much more confident going to a trained doctor."

4. **Would you be able to upload an image on a website?**

"Yes, I am able to do simple operations on a website and I feel like most others would also be able to."

5. **Do you think that a system like this should anonymised or have an anonymous mode?**

"Making the whole website anonymous may be able to increase privacy but I feel like it would not be safe because if someone actually had a cancer then they would not get contacted. The idea of an anonymous mode upon request would seem much better as it would allow some patients privacy however the majority will be able to be contacted."

6. **Do you have any potential concerns about this system?**

"My main concern with a system like this is that if it claimed that I had a skin cancer when I do not it could cause unnecessary panic and stress for me."

7. **Do you have anything else to add?**

"With the pictures I was wondering how they will be used and if they are going to be stored, I think it will be incredibly important to encrypt the images to make sure they cannot be stolen".

Ambrose Pailing

1. **Do you think that you are at risk of getting skin cancer?**

"No, I protect myself against and use sunscreen as well as cover my skin sufficiently when exposed to the sun."

2. **What do you think about 100,000 cases of skin cancer are diagnosed yearly in the UK?**

“This is a high number of cases, and I did not expect the numbers to be so high however, many of these cases are not life threatening so I am not inclined to feel vulnerable.”

3. **Would you be willing to check any lesions on a website to help keep yourself safe?**

“Yes, I feel confident in the technology and trust it to predict correctly, it is also less hassle because there would hopefully be no long application process and waiting time.”

4. **Would you be able to upload an image on a website?**
   “Yes”
5. **Do you think that a system like this should anonymised or have an anonymous mode?**

“I feel like an anonymous option should be optional as the majority of people would not require such service.”

6. **Do you have any potential concerns about this system?**

“If I was to use a system like this my main concern would be the potential for hack that data would have and if I was to test positive, I wouldn’t want other people knowing about this”.

7. **Do you have anything else to add?**

“No, I think I have covered everything”.

Analysis

Initially it was clear that neither patient though that they have a high risk of getting skin cancer but were shocked by the number of people who get diagnosed with skin cancer yearly. They both agreed that they would be confident checking lesions on a website however, Subil mentioned that of the lesion was serious he would prefer to go to a GP.

When questioned about privacy they both also agreed on the fact that the system should allow varying levels of anonymity depending on what the user would like. Finally with concerns Ambrose mentioned the fact that the website needs to be secure and Subil mentioned how the accuracy of the test would be important. Therefore, security and accuracy should be two clear objectives for this product.

Research

Existing Solutions

Google AI

Overview:

The 'Google AI' managed to detect breast cancer more efficiently than doctors and has also been shown to have the ability to detect breast cancer early before detectable by the human eye. This tells me that the project is viable as similar projects exists. Furthermore, in many cases such as this one they have a s system where the AI is backed by a team of doctors that review some cases.

Furthermore, when making their model they decided to feed it less information than the human experts did by choosing not to feed it the medical histories and only feeding in the X-ray of the mammograms. At first thought this may have made us think that this will cause the Google AI to be less successful however, it could still predict at a much higher level than the human experts could predict at.

After conducting many studies, they also concluded that their AI would be optimized when used in tandem with human radiologists which would be able to reduce the workload of the radiologists and help to free up more time for the radiologists as they only had to look at cases which the Google AI were uncertain with the prognosis.

To do their modelling the google team used 3 different neural networks. Each of these looked at the different levels of detail, allowing them to predict more accurately since the first neural network would be able to give a first answer however the second would then be able to use that result to predict more accurately and then the output from the third neural network would be more accurate. In googles case they trained the neural networks to pass encrypted data from one another for this process.

Parts I can apply to my solution:

From reading the information about the Google AI I think it is essential to work in tandem with general practitioners in my case because it will allow the system to reduce the workload of the GPs as well as a more important factor which is that it allows us to give a more definitive answer in case the system results in a false positive where the patient has not actually got a tumour therefore provides a second check for the patient since being told that one has a cancer is a big event

Furthermore, I will also implement the idea of using multiple machine learning algorithms. Where google AI used 3 trained neural networks I will use [*I think*] a support vector machine as my initial layer then a neural network behind it to give an accurate solution. However, where the Google AI got the neural networks to learn how to interact with each other I will manually set the interaction between my machine learning algorithms.

Skinvision

Overview:

Furthermore, there is a company called SkinVision which provides a skin cancer melanoma app. This is very similar to my proposed solution so there are many different features that I can develop by looking at their solution.

They work off a system where a patient must input a picture of the skin lesion and within 30 seconds, they will feed this into their model and feedback using a three-tier system of low, medium and high from how likely the skin lesion is to be a cancer. They also allow patients to perform regular self-checks on their skin via their app where they send frequent push notifications to patients in order to keep them safe.

The company will also store every photo that is sent in in order to help you monitor changes in your skin over long periods of time. However, the service they offer is quite dear at €6,99 per check.

Additionally, the company also posts lots of information and guidance on skin cancers and simple steps on how to prevent them from developing to increase their user satisfaction by helping the users in every way possible.

Parts I can apply to my solution:

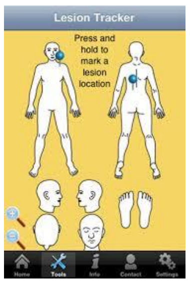
From their solution I could take into account their 3-tier system and make my own system because the machine learning algorithm will output a continuous variable from 0 to 1 where 1 is 100% a cancer cell. Therefore, to make the system easier to understand for the end user it would be important to qualify what the number given to the patient means. Although their service is good and they have a tier system they do not go an extra step and try to classify how dangerous the tumour, they just output the risk a person has of having a tumour. Since the majority of skin tumours are benign and almost harmless however, the ones to worry about are the malignant ones which my system will also be able to categorise. Therefore, I feel that my system would be able to provide a better end product in terms of output where my system will go the extra step if a tumour is suspected and find out if the tumour is benign or malignant.

Moreover, to help the users feel like the application is understanding I think it is also important to publish basic information about skin cancers on the website to help the users feel comfortable using my application and so that it is not very intimidating which is very well executed on SkinVision's website although they run most of their program through an app.

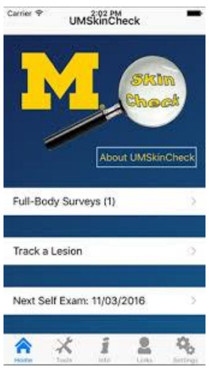
UMSkinCheck

Overview:

This is an app which was developed by the university of Michigan school of medicine which allows a patient to complete a full-body skin cancer self-exam as well as create a history of moles and growths on the skin.



They provide a step-by-step guide on how to complete the exam accompanied with graphics and written instructions. On the right is an example page from their app. It is not available on android and the UI is significantly outdated. On the other hand, the information given on these tutorials are well presented and very informative serving the purpose of helping people self-test well.



Furthermore, one feature that they include is a history of moles and lesions on the skin. This is a simple feature which just requires memory that they have employed but it is very useful. This can be seen in the section called track a lesion. They also allow you to create different sections for different lesions which would allow the patient to view how a lesion is changing and if it appears to be changing.

Finally, this app also has push notifications to make sure that you check your skin frequently. This is very useful for people who may have a suspected developing tumour but there is quite a lot of uncertainty. You can see this at the bottom of the image on the left.

Parts I can apply to my solution:

From this I think it would be ideal to be able to inform patients to take a check frequently on my software. However, since this is a website, this would need to be in the form of an email from a smtp server set up on the website. This would consist of a friendly message to customers in order to tell them that they should consider rescanning their lesions to make sure that none of them are tumours.

Furthermore, I would try to incorporate the system which they have named track a lesion. This is because although a lesion is small and may be determined to be low risk if this lesion begins to grow and the patient does not realise it could be important for them to have reference images to compare how the mole has changed which they could ultimately use to get a prognosis from a GP even if the system does not think that the lesion is dangerous.

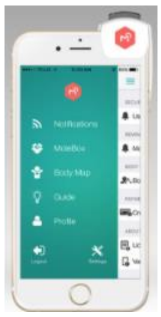
Finally, for my information page I could reference some information from this app in case some of the patients are not fully confident with the system's prognosis. This would allow patients to check if the machine is correct and if they think that it is not, they will be able to get referred properly.

MoleScope

MoleScope is a service provided by a company called metaoptima. They have four main stages of how their product works.

With a clip-on camera, MoleScope uses the ABCD method to complete a risk assessment of your moles.
MoleScope

The initial is to attach an external device onto a compatible smartphone which acts as a lens to take more detailed pictures. They claim that their lens allows better subdermal imaging as well as being dermatologist approved. However, as cameras on phones are developing so rapidly, I believe that a phone camera should have sufficient zooming capabilities to carry out this task. Additionally, the cost of the extra hardware seems unnecessary.



The second is to use their app which would allow patients to take resolution images of lesions using the lens which will enable a better zoom into the lesion. Their app has a very simple UI as can be seen on the right. It works on a basic system where every time the user adds a mole, they give it a label and then you are able to place it on a body map. This allows the user to have a visual of where their moles are and is easier to look at than a table of different moles.

The third stage is to get the moles checked out. However, MoleScope does not any software to analyse or diagnose the moles but allows you to view the mole along with a description of the ABCD method for self-testing which stands for: Asymmetry, Border, Colour and Diameter.

Their final stage is to send any moles that you flag as dangerous to online dermatologists who will be able to do an online check-up and review the pictures of the moles. The user will then wait while their image is processed within 2 days.

Parts I can apply to my solution:

While the unique selling point of the solution by metaoptima is their lens I think that it would be impractical since the aim of my service is to allow everyone to get checked for free.

However, the use of the body map on their app is something that I could use in my solution. Where people view their moles, I could try to include where it is on the body and sort them this way to allow patients to view their moles in a systematic way.

Furthermore, I could also use the feature of being able to send pictures can be utilised in the system where the patient is getting referred to a GP because the pictures and history of lesion would be incredibly important for the GP to give them assessment with as much details as possible in order to maximise the chance of the patient getting the correct overall prognosis.

Check4Cancer

Overview:

Check4Cancer is a clinic which offers a service called SkinCheck. Although they provide help with many different types of cancers I will be focusing on this department.

This company offers private cancer tests. This means that the cancer tests are completely anonymised and that a patient will be able to keep their identity hidden which may be a priority for their customers because cancer is a very sensitive issue where people’s mental state must be monitored carefully.

Additionally, a patient can book an appointment on their website for a fixed rate of £175 for adults above 18 years old and then allow a patient to go to one of their surgeries which are nationwide. Patients can book rom up to 48 hours before the appointment slot. In the surgeries the patients will have high quality images of their lesions taken which will then be sent to consultant dermatologists and the patient will receive their results in 5 working days which is a full week.

Furthermore, they also offer personal guidance on how best to protect your skin as well as a free skin health information booklet. They also take pride in their patient experience by offering high levels of care and attention with a frictionless process. This would be important because getting cancer is a life changing event and it is extremely important to help patients feel comfortable in any way possible. The image on the right is an example of the cover page of some information that they have posted in order to increase patient satisfaction and help them feel positive.

CHECK4 CANCER

How you can reduce your cancer risk for 2021

Simple and easy lifestyle changes to reduce your risk of cancer in the New Year

2021

Parts I can apply to my solution:

While the SkinCheck is a service provided by nurses and doctors for the patients, there are still many features from this which I can look to incorporate into my end product. The initial feature which stood out for me from this service was the positive reviews from how friendly and kind the staff were. In order to integrate this into my solution I think it will be very important to create a friend UI using light colours and to make sure that the website does not come off as unpleasant which may scare users away from using the website because many people are still afraid of technology.

Additionally, since a high majority of their customers preferred anonymity. This will help to get more people to start testing their moles because every user who is afraid of what is happening with their data will be allowed to access the website on an anonymous mode which means that none of their data will be stored but they will still be allowed to get their skin lesions checked out.

Features of my proposed solution.

Initial concept of my solution after research

My solution is an application that will be a website that will enable patients to get their skin checked out for free. Furthermore, if necessary, I will also make sure that the patients will be able to contact a GP or dermatologist if the risk of getting a melanoma increase. Additionally, the website will need to be simple and easy to use to make it accessible for everyone who is using it, therefore each page should have clear and concise instructions that can be followed as well as a frictionless system that the patients can follow through to help to keep the user experience as high as possible.

Limitations of my solution

The problem of online detection is a famous one. There are multiple different solutions which detect cancers online. There is a growing industry where machine learning is used to detect the prognosis of breast cancer by feeding the scans into machine learning industries. Therefore, there are many different sources published online which can aide me in my project to create a skin cancer detection system however, there are still many limitations in this project.

The first glaring limitation is the dataset, in many of the machine learning applications listed below the algorithms would have been trained with millions of images and examples in order to tailor the machine learning algorithm. I will be restricted by the dataset that I will use which in this case has entries.

Furthermore, since I am using a dataset and am not able to survey the data myself there is a high chance that there will be dataset bias which is caused when the people who were collecting the data did not choose a complete random sample. In this case there is a high bias of samples which have a skin cancer than not in the database so the algorithms must be trained carefully to make sure that the model does not overfit the test data and to make sure that it can still give a statistically correct decision for an unknown section of the dataset.

Additionally, another limitation of the product is that the website may not be trusted enough to get many patients. Due to this, I must concentrate on the UI aspect and make sure the UI is friendly and invites people onto the page.

Requirements

Software and hardware requirements

Hardware

A computer that is able to run masses of data – This will be very important when running the machine learning algorithms, they will need lots of processing power and will need to be run multiple times in order to find the optimal algorithm. For this most desktop computers will be capable of running these sort of programs --**Cloud? Or is that software??**

Any device with internet – To access the website system the end-user will need a device connected to Wi-Fi with a web-browser on it. Almost every device will fit this requirement.

A camera – This will be needed by the end user to take a picture of their lesions and add the onto the website. The software will be able to make this a seamless process.

Software

Web browser – This will be all that is needed on the end user's computer; however, it will need to be JS enabled which all newer devices have however, some old devices may not have access to

Python ide – This will be needed to run the machine learning algorithms and to help fine-tine the algorithms on the back end. However, it will also be important on the front end as a temporary web development server to allow the website interface to be created with small changes but not pushing it onto the main server.

Flask/(Django) – This is a web micro-framework that will allow me to create a website using python as the backend and create the front-end using JavaScript, CSS and html. Furthermore, to connect the front and back end I will be using Jinja 2 which allows me to add the functionality of templating environments which means a dynamic web page.

Heroku server – This will be used as the web server which will allow us to push updated files onto a repository and allow live updates onto a website that is hosted online.

NumPy for calculations, Matplotlib to visualise data and Pandas for operations – These are all packages built into python which allow us to do calculations and relevant processing for machine learning more efficiently rather than doing them all inefficiently.

Success Criteria

Criteria	**How to evidence**
Simple and lightweight website for users to interact with	Screenshot of the web page that is clearly laid out and is not cluttered.
Robust protected website	Evidence of attempts to steal data from the website and the precautions put in place
Easy to use website	Screenshot of every stage of the upload process and each stage being explicitly explained for the user
Non-complicated home page	Screenshot of every stage of the process
Anonymous Mode	Screenshot of a user using the website anonymously with none of their data available
Extra information about what skin cancer is	Screen shot of a page that displays what melanomas are and has some public figures
Melanoma extra support	Screenshot of a page that reassures people about next steps after a positive diagnosis
Fine-tuned ML algorithms	Use a test and cross validation set to test how well the algorithm works and make sure the test accuracy is high enough, and that the data is not overfitted.
Integration of ml algorithms together	Use another test set to check if the overall diagnosis given by both machine learning algorithms is accurate.
Accurate Data Analysis	Screenshot each piece of code used for analysis

Design

Decompose the problem

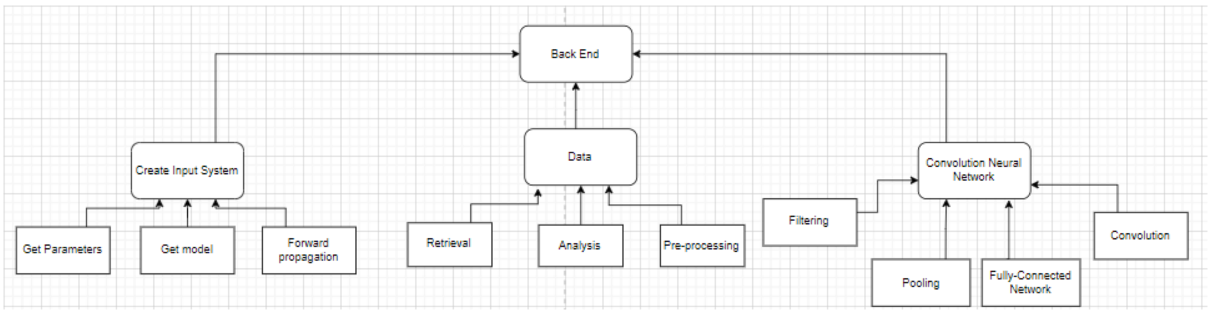
This can be split into the front-end algorithms and the back-end algorithms

The front-end can be described as the user interface for the problem, which will consist of a website that is user-accessible

The back end is the main challenge of this project, which consists of

- Data including retrieval, analysis, and pre-processing
- Creating a neural network that can process the images and check if they are cancerous
- Create a feed-forward system where images can be inputted into the network

Back-End

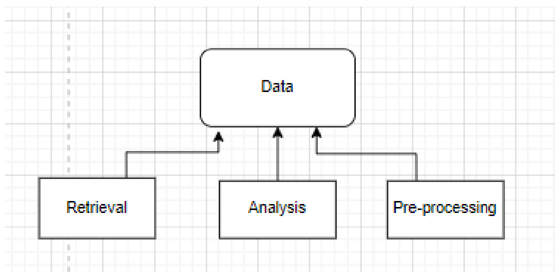


Here is a top-down diagram of how the back end of the system will be decomposed, which will be explored in more detail over the next few pages as well as justified why each section needs to happen

Data

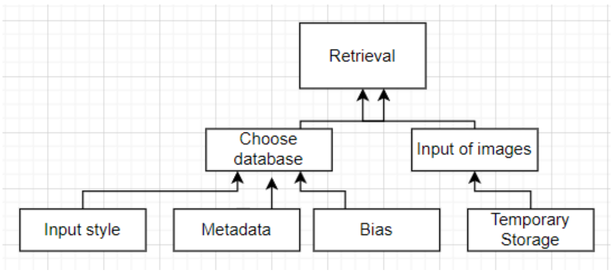
The data section can further be split into 3 as seen in this top-down diagram here.

However, once expanded more the top-down diagram gets very complicated and this will be explored in the subsequent sections below, being the retrieval, analysis and pre-processing of the data



Data Retrieval

This can further be split into the choice of database as well as the inputting of the images in the live system



- Choice of database

  To do this, there are a lot of conditions to decide upon, such as how big the database required should be, what metadata is provided, how many types of lesions are provided, the bias within the database, data formatting…

Condition	Relevance
Size of database	This is how many images of tumours are provided, which is extremely important when deciding how to train the network as the more data points there are the more effective training will be, but the more processing power will be required
Metadata provided	This is extremely important, so we know about the context about the data which can be taken into account by another network that can be used to check the bias of the data, some examples are given in analysis of database
Types of lesions	This is extremely important to analyse because even though the problem ultimately comes to does the patient have cancer or not, if the data base can provide specific diagnoses, it will be more reliable, and the user can be given more

	direct feedback upon what type of skin cancer they have or what type of mole is present
Database bias	This can be gathered through the metadata and is extremely important to gauge from the get-go because if the database only contains lesions from a specific stratum of population (e.g., colour) then if someone from another strata will not be able to use the system
Data format	The data should be in a format that is easy to process, this could be if the images are provided in alternative formats such as already in an array format

My choice of database was HAM10000 database provided by ISIC (https://challenge2018.isic-archive.com)

Condition	
Size of database	This database provides **10015** separate images of lesions on skins. This is more than enough to accurately train and test the network. To get over the obstacle of processing I will utilise google servers and run on a client server set-up
Metadata provided	This dataset also provides lots of metadata, which will be explored n next section
Types of lesions	This dataset provides 7 types of diagnoses, Melanocytic nevi', 'Melanoma', 'Benign keratosis-like lesions', 'Basal cell carcinoma', 'Actinic keratoses', 'Vascular lesions', 'Dermatofibroma'
Database bias	The bias in this database appears to be low with people from a range of ages being chosen
Data format	This database provides HD images of lesions as well as correspond 64x64 matrices of all the images which are much easier to load into system and process

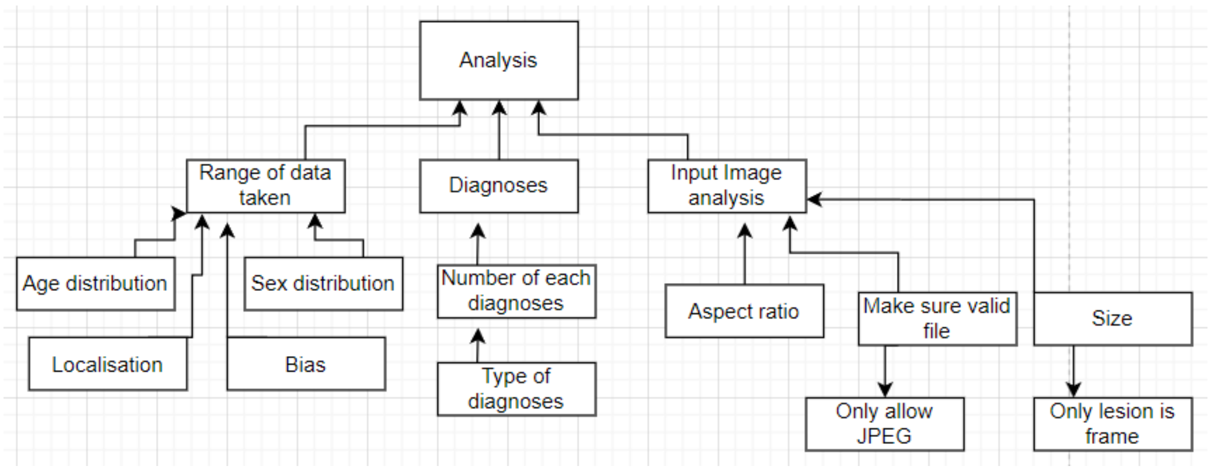
Image inputs

This is an extremely trivial section because the image has to be found and then needs to be saved somewhere temporarily

In order to input the image on the website, a view of the image preliminary view has to be seen to make sure that the user can see their lesion temporarily before they begin analysis on their lesion.

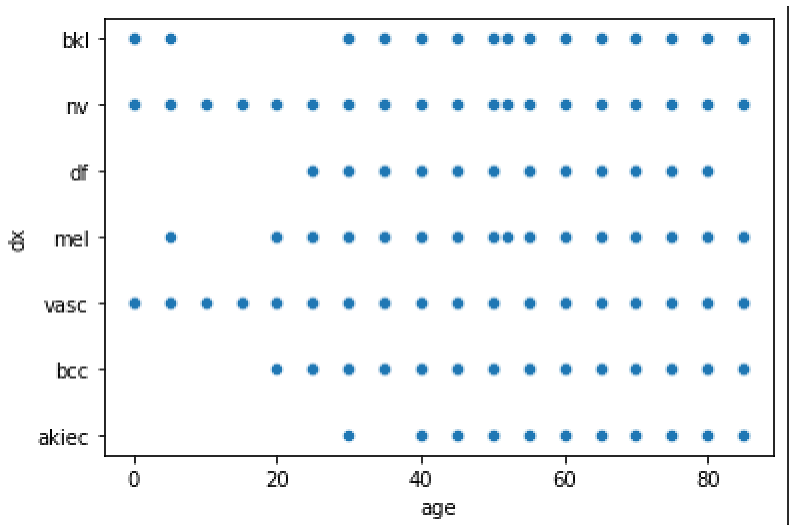
Data Analysis

The main aim of this section is to check the bias of the database using the metadata that has been provided as well as any other information that can be gauged from viewing the images



Age Range

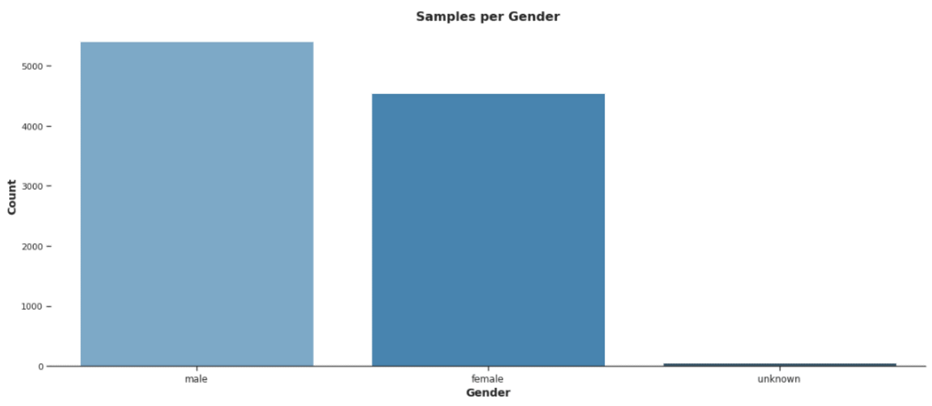
- Important to test general bias of the end product



- From here is can be seen that a wide range of ages have been utilised for each diagnosis
- The median is a bit high, being around 50 but this is to be expected as most tumours are occurrent in elder people

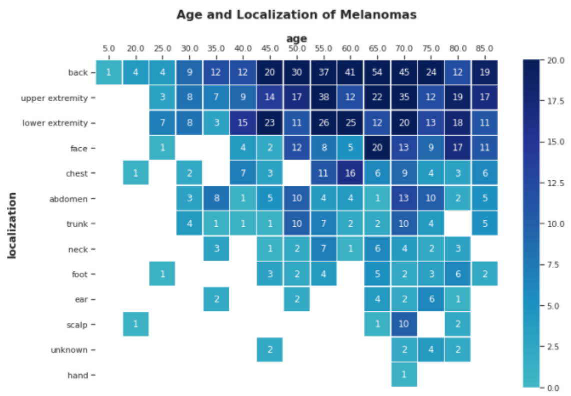
Sex Distribution

- This is also important because often databases are very skew towards one sex



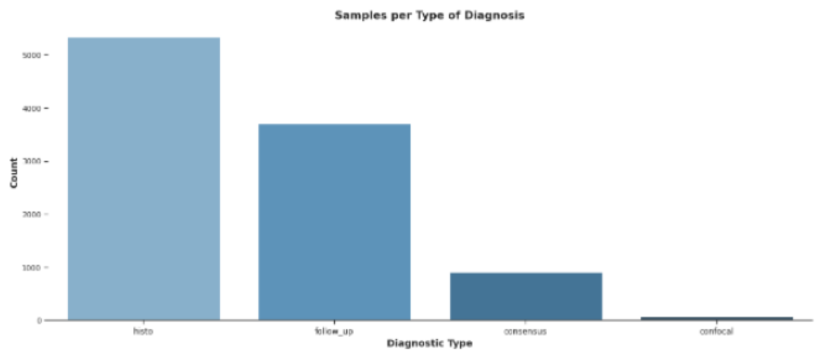
- From here we can see that the majority is male however, upon further analysis the distributions within both male and female are extremely similar, and this is only a 5% difference which will provide littles significance

Localisation



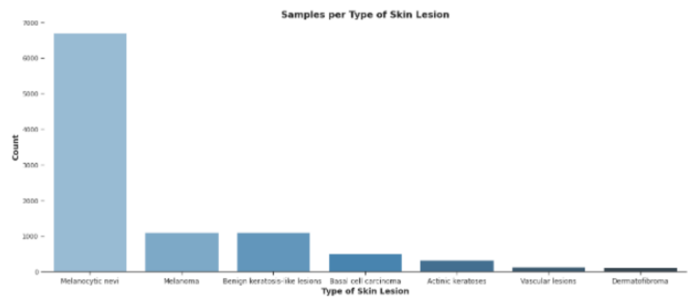
- The scalp seems to be the most common localisation for 70-year-olds and other than that most other localisations are places associated with long exposures to the sun, so it is unlikely that there is any correlation between there 2 variables

Diagnosis Type



- Here we can see that there is a large bias upon the type of diagnosis therefore this might not be useful within the fully connected neural network

Diagnoses



- Here we can see that there is a high bias towards the melanocytic nevi
- This will provide a problem in code generation

Finally, we also have to consider the actual live analysis of the input image, most of this is self-explanatory

- Aspect Ratio – I have chosen a square image to be my target size
- We also have to do both client side and server-side processing to make sure that the image inputted is in the correct format and not a phishing attack, this can be done by restricting data types however, that will not be enough
- The size of the file is also important, making sure that it is not above 1mb as well as that only the lesion is in the frame, which can be achieved by displaying the image back onto the webpage

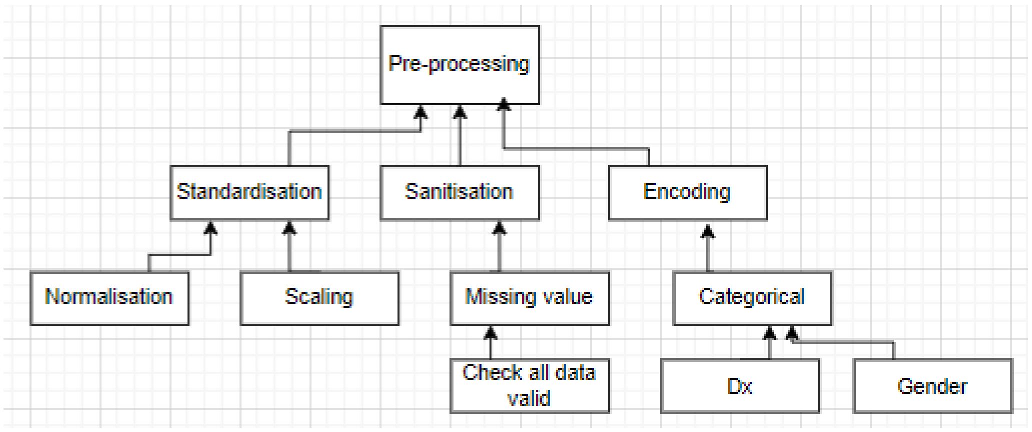
Data Pre-processing

- This is an extremely important part of data procurement as it will reduce the bias of the dataset as well as make the data easier to process for the neural network
- Changes raw data into clean data

Missing Values

- For many items such as gender for people who did not declare a gender and gave the answer as other, the database has recorded a NaN, which will need to be addressed and fixed
- Additionally, all continuous variables will need to be standard scaled or normalised to reduce large number (anomaly) bias as in a normal distribution all data is within 3 standard deviations

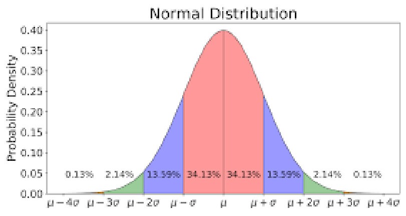
Data Pre-processing



Now we have all the data provided, we need to design a machine learning algorithm which is appropriate and find out if the data has any dependencies within it

Standardization

- This is the first stage which is essential for continuous data points
- There are 2 methods to standardize being Normalisation and linear Scaling
- Normalisation seems to be most apt in this scenario
- This relies upon the principles of the normal distribution and will map all the values to be within this distribution



Sanitisation

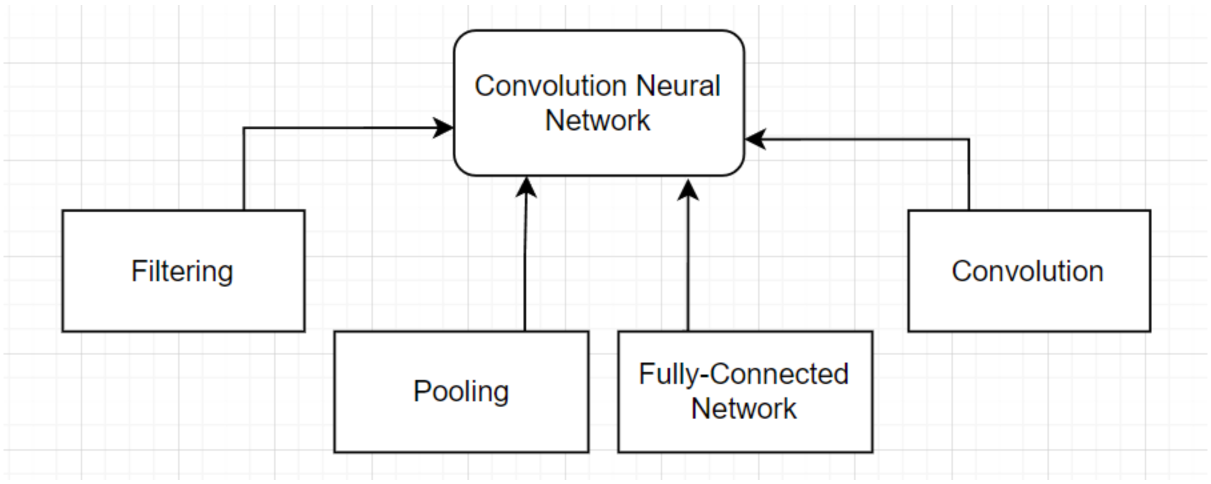
- This is extremely important to stop errors from occurring when the network is running
- This will attempt to fix all NA and NAN values within the dataset
- This will include mean imputing the data (can be median in some cases)
- Additionally, this will include the security of any image sent across the network

Encoding

- For the majority of the CNN encoding is not needed but for the fully connected layer one hot encoding may be useful in case any other categorical variables will need to be included within the model
- Some examples of this would be the actual diagnosis but this is already provided by HAM10000 in a discrete form, and another is the gender which will need to be converted into 1 and 0 or -0.5 and 0.5

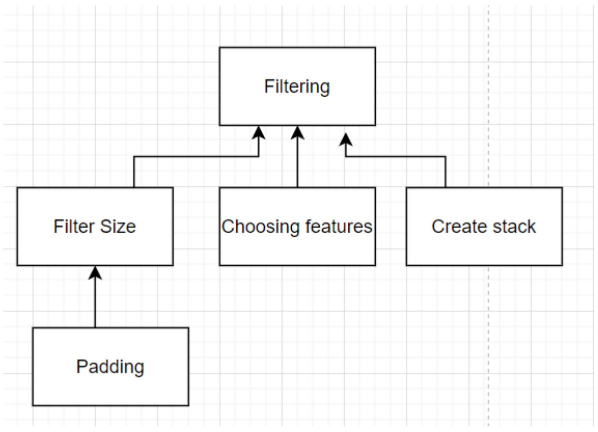
Convolutional Neural Network

This is by far the hardest part of the project, and I split it into 4 different sections



Here we can see the main 4 sections being Filtering, pooling, convolution and the fully connected network. Each will be delved into below and broken down into smaller segments

Filtering:

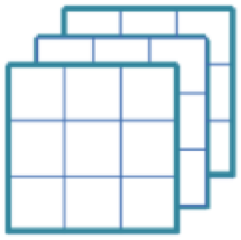


- The filtering layer of the convolution neural network can be split into 3 more segments

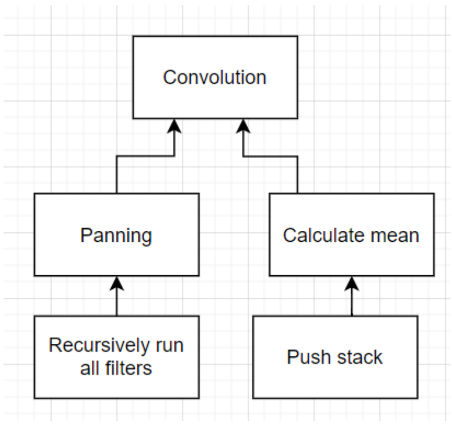
- The first is the filter size and hyperparameters associated with these
- This is important as the filter size has to be specific towards the image size
- Additionally, all the filter values have to be given pre-sets which provide no bias
- Then we also need to add padding to the images to prevent any overlapping issues with the actual filter when it is convolved

- Choosing features may seem like a vague point but is the most difficult part of the model
- Need to provide means to choose different features and then also provide them their weights upon the fully connected network
- This will also help to reduce size of frame of each image but will create more images

➢ A stack then needs to be created which will allow for the stack of convolved features to be put together so they can be accessed later



Convolution:

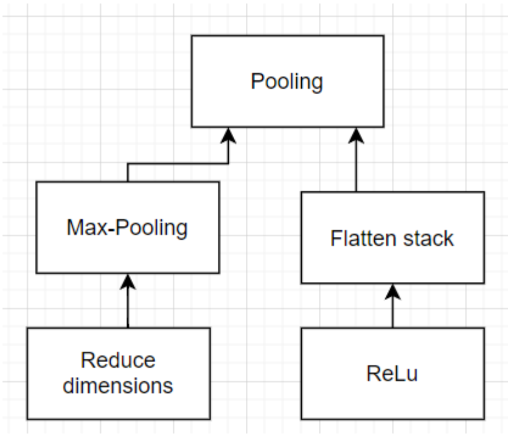


- The actual convolution phase involves panning the filters over the processed image in order to create feature matrices that represent the image at a higher level
- This panning will be run at a set stride and will run for all filters across the whole of all the images

∞ In order to actually create the convolved features, we calculate the pixel value then multiply it by the filter value at the point, then calculate the mean of all these products in the frame and place that in the new image at location of centre of new image

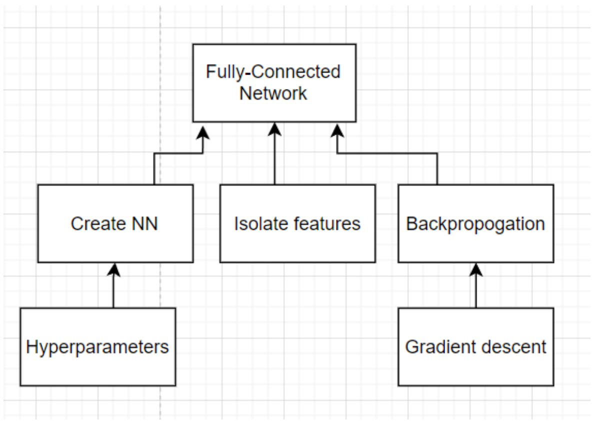
∞ This will therefore reduce the fringe dimensions

Pooling:



- The choice of main algorithm for pooling for this stage is max pooling, this is where a window is placed over the image and the maximum value within the image is taken as the new pixel value at that point
- This can also be used to flatten the stack to make less feature layers
- This helps to make the network less sensitive to positions
- Additionally, we also have to normalise these values and when negatives values show up, we can use the ReLU function to remove any negative values and map them to 0

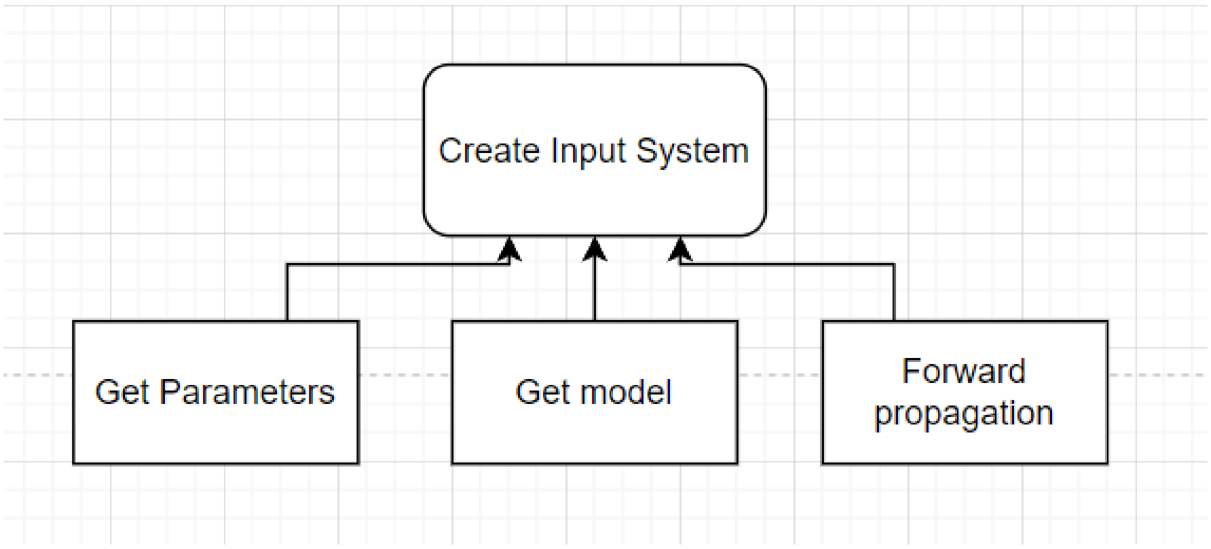
Neural Network:



- This is the most involved part of the CNN which involves taking every feature layer from above and putting them together in order to make a prediction about the skin cancer
- To do this we initially need to isolate all the separate features and make them an input node into the network which can be used to process.
- Once we have all the features, we can then take the training data and feed it into the network
- Once this is done a cost will be calculated and using this cost gradient descent will occur in order to minimise the cost function

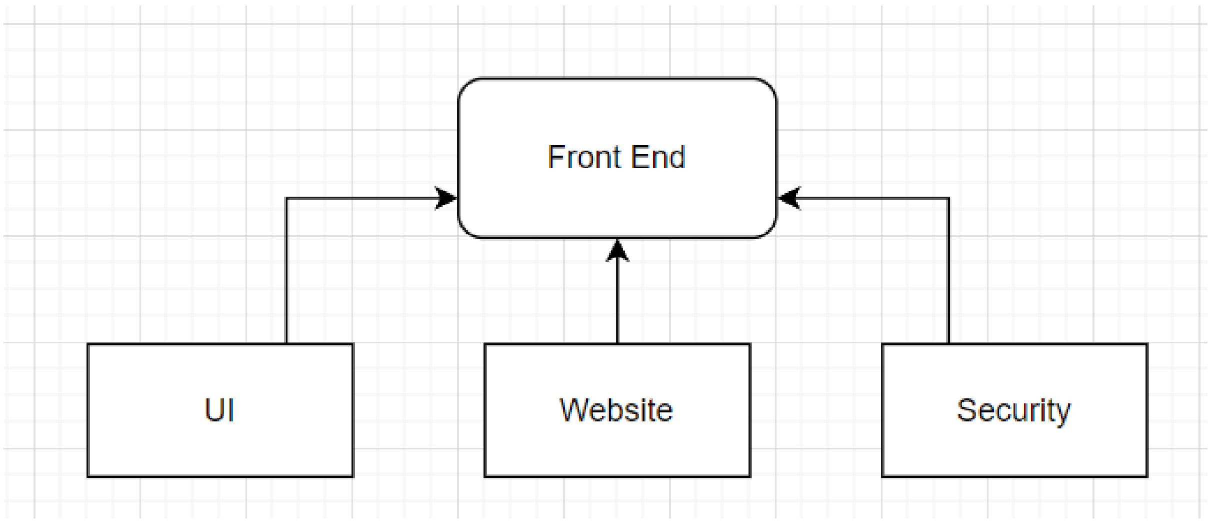
Feed Forward System

Although this can be considered to be part of the CNN, I believe it is important to develop this side separately to allow abstraction to be utilised inside the solution, where the user does not need to know how the CNN works



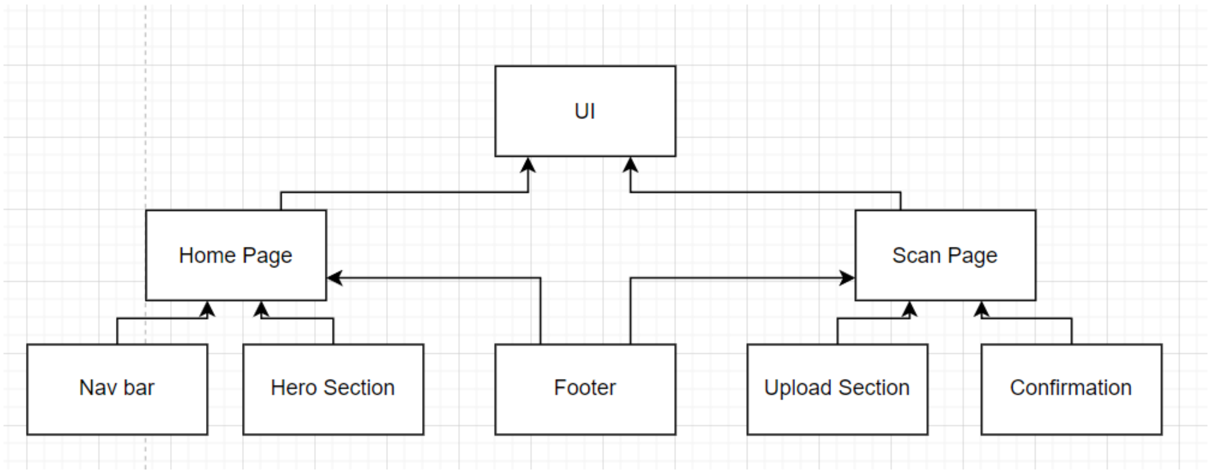
- Getting the parameters will be a trivial task because the parameters for the model will be thousands of lines long and it will be essential to use a pickle file to store the parameters to allow for rapid downloading as well as quick upload into variable explorer
- In order to get the model, I will need to create a class which can be accessed as a module in order to allow for quick processing of the image. It will also need to parameters before this stage
- Finally, we will need to use the previous two stages in order to create a forward propagation algorithm that can be ran on the cloud to allow for the web hosted website to run it.

Front-end



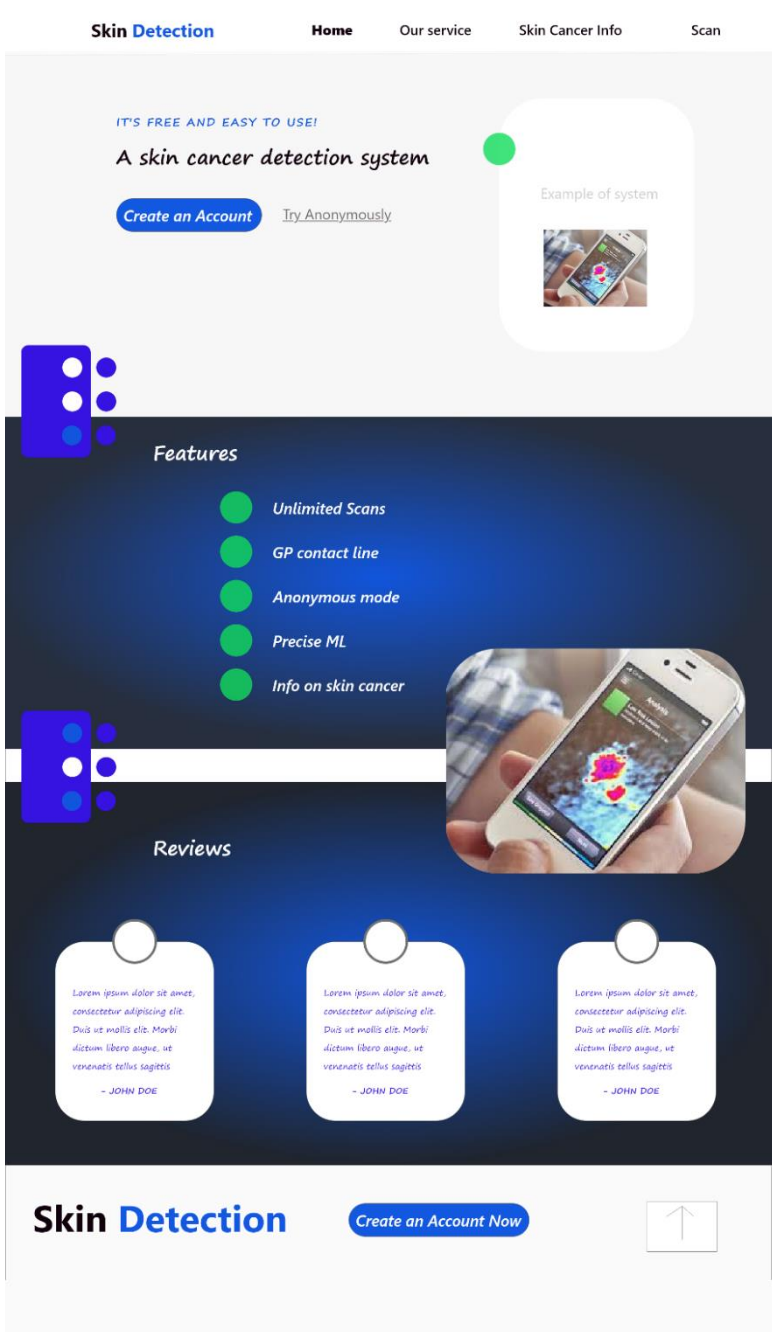
As can be seen from this diagram the front end of this website can be decomposed into 3 main sections, being the User Interface, the actual website development and finally the security of the product in a client-side model.

User Interface Design



Home Page

Here is the initial design for the home page of the website.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ut mollis elit. Morbi dictum libero augue, ut venenatis tellus sagittis

- JOHN DOE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ut mollis elit. Morbi dictum libero augue, ut venenatis tellus sagittis

- JOHN DOE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ut mollis elit. Morbi dictum libero augue, ut venenatis tellus sagittis

- JOHN DOE

Here we have a few main sections, initially at the top we can see the navigation bar. Next the section which covers most of the page is the hero section. This is the section that is made to grab the user's attention. Furthermore, we also have the features sections then we move onto the reviews or testimonials section before finishing off with a footer.

Navigation Bar

As can be seen in the image I have decided on having a navigation bar at the top of the website. An efficient navigation system will enable visitors to travel through the website seamlessly without having to look for links all around the webpage. Furthermore, it also acts a summary for the website being a short and concise way of summing up what the website can do.

By choosing the colour to be white while the background of the page is an off-white grey colour, it will link in with the first point in the success criteria which helps to keep the website light and simple. This is also something I noticed from the UMSkincheck app that has been implemented in my initial design. Furthermore, since in the interviews with the doctors they said that it would be very important to keep the website easy to use I think the nav bar will help to achieve that goal.

Hero Section

The aim of this section is to convey the service that the website provides in a short and succinct way.

The more important part of this section will be to put a form of skeleton UI in the white rectangle on the right-hand side of the page. This would be to give the users an immediate understanding of how the system works and make sure that they fully understand the skin cancer detection service that we are providing. This will help to fulfil criteria 4 and 5 which state that the home page of the website should be simple but also explain what we do. Therefore, this section is essential!

Additionally, for this section, I have used a font called Segoe. As for the titles of the main sections, this helps to keep the text light due to its open and friendly type face. This helps to keep the website friendly which is very important because skin cancer is a serious issue, and it is important to attempt to calm the users down subconsciously. Also, the font has lots of weights which helps to establish a good visual hierarchy.

This visual hierarchy has been established in the hero section mainly. We can see this where the two action buttons are, these are 'Create an account' and 'Try anonymously.' Both are using the same font but the create an account is the primary call to action so has been given a bolder weighting as well as given the primary colour for the website (blue) as the background for the website which helps the main call bright and stand out on the website. This is extremely important, and it immediately incorporates the anonymous feature into the website which is on the success criteria and something that was unanimously want by all the people who were interviewed from the doctors to the potential users of the website.

Finally, to keep the website simple I have decided to keep the colour of the scheme of this section very basic with the main title bring black and the primary colour for this website being the blue. This can be seen in the title in the top left-hand corner which has utilised both the primary colour and white.

Features

This section is intended for users to scroll down to once they have read through the hero section and then chose to keep scrolling through the website therefore registering their interest in the website.

This section is short and succinct and therefore aims to give the user a concise guide to what the website offers and why people should sign up to our skin detection service.

The first thing that we can see here is that the background colour has been changed. This has gone from a grey into a gradient blue background. This will help to shift the focus of the user onto the centre of the page where all the main features of the website are. This will help to keep the users engaging with the website and work to emphasise the main points. This has been done to fulfil the criteria of keeping the website easy to use and very light on the user.

Furthermore, this section also allows us to increase the confidence if the users using this website which was an issue raised with the doctors and therefore having a section that fully explains what the website entails is important. For further development it may be important to include the security of the website on this section as that is a point that was raised by both the potential patients in their interviews.

Reviews

This section has been included underneath the features section and now is filled with placeholder text which would be filled in with the actual reviews that our initial interviewees will give us when the entire system has been included.

This has been incorporated after the doctor's comments which stated that for users to be confident using a system like this it must be reviewed well online therefore, I think it would be important to add these onto the front of the website.

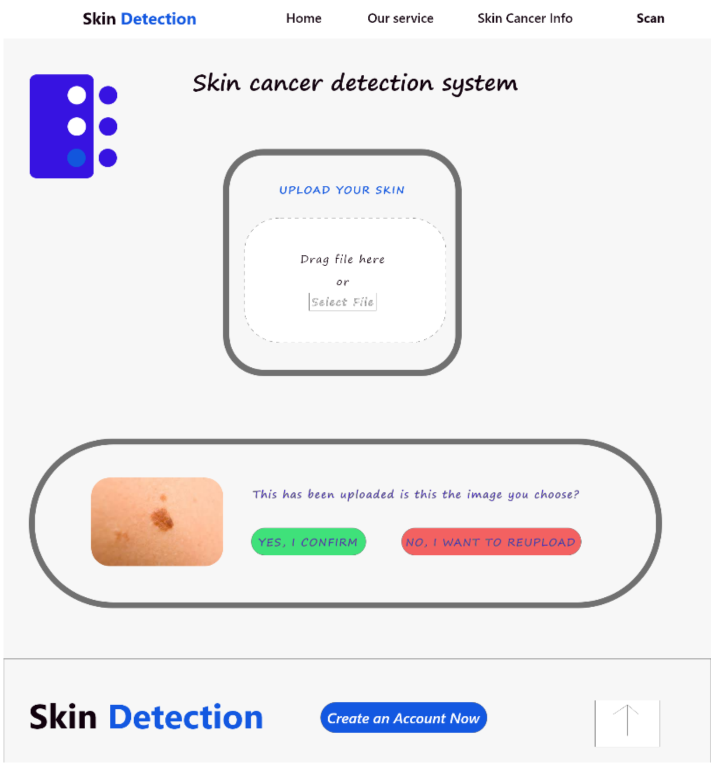
To design this section, I am planning on using a system that is similar to skin vision where they can be scrolled through at the bottom of the website. This is undoubtably a feature that they have used to increase the confidence that customers feel when using the website.

Footer

At this moment in time the footer has been used to reiterate the main message of the website to help to shepherd the users into creating an account on out system. This works to the criteria of keeping the website explanatory and therefore will help to keep the website unnecessarily complicated with another navigation section at the bottom.

Scan Page

Here is the initial design for the main page of the website which is the page where the main function of the website is carried out which is the skin cancer detection.



The design for this page is significantly simpler than that of the home page. We have new 2 main sections evident on this page at this early stage. We have at the top the upload image section and towards the middle we can also see the confirmation section. The website will also have a constant header and footer so in this design I have used the same header and footer from the home page.

Upload Section

This section has been designed with the main idea being to keep the website simple easy to use, which is the first point in the success criteria. To do this I have used a section with a white background to allow the user to upload their images as it will stand out on the page. This section will have potential to be restyled as the development of the web page develops because the current drawing is very rough and can be improved.

Furthermore, at the top of this section there is a clear title in 'UPLOAD YOUR SKIN' which explicitly states what the upload box is for which works to help keep the website more explicit than implicit, allowing it to be explanatory which was one of the bullet points in the success criteria.

Finally the whitespace in this section will also be essential to implement when building the web page to try to make sure that the website is easy to follow and that the page is not too busy, therefore making sure that the website will stay simple, which would be important to make sure that the users of this website are confident when using it.

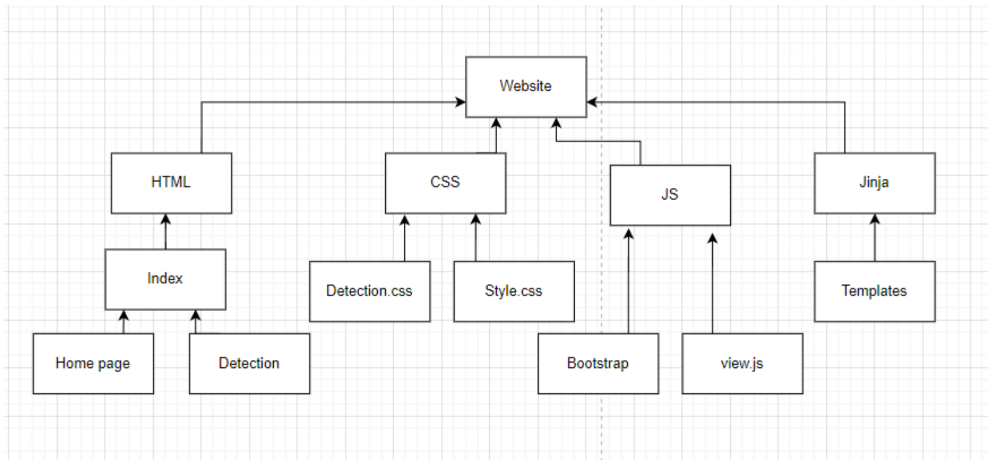
Confirmation Section

This section is very simple and will render the image uploaded onto the website and make sure that the user is happy with their choice. This is essential so that the user knows what they have actually uploaded to the website and make sure that no mistakes have been made when uploading or taking the picture.

The essential part of this section is the confirmation buttons. To help improve usability of the website, I have followed the well-known colour scheme that most people assume, which is that to carry on is green while to reject is red. This will allow the users to traverse this page with ease.

This section has been enclosed in a box to help the user to focus on the section at hand and not get confused with other parts of the website, therefore helping the unclutter the website and keep it clear.

Website Development



HTML

- For this I created templates for the pages, on the website and because many pages have overlapping features, I used the principle of inheritance to create a base instance for the pages which includes the general usability features and then created new instances of these pages with specific styles and themes for each page

CSS

- I had 2 main CSS files that I operated with, a general Style.css that dictated all the general features according to the user surveys with the appropriate fonts loaded in and then Detection.css that deals with the bulk of the detection page items and their features

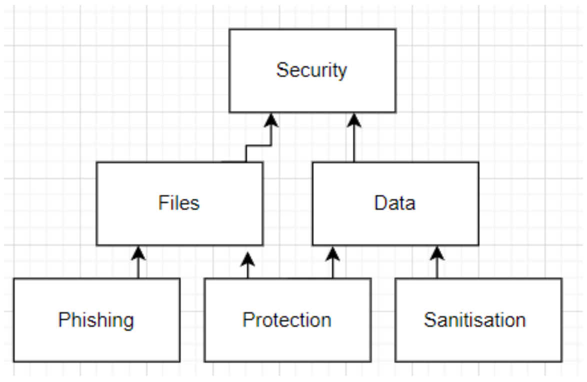
JS

- To deal with the JavaScript and inner workings of the website, I initially imported the bootstrap libraries to get a common theme and they used my view.js file to make appropriate functions for actions and to make items appear and disappear when appropriate using event listeners

JINJA

- This is a fast-templating engine which allowed me to dynamically edit the web pages using flask (without php).
- I used this to render images onto the website as well as any information specific to users
- This can be used to store session data

Security of the website



The security of the website can be bisected into 2 sections, being the image files that are inputted and then any data that is inputted by the user.

Files

- For any image file that is inputted, the input box can be used for phishing therefore it is important to make sure that only images are inputted and that they are of a specific format
- Additionally, the images may be a piece of malware in disguise, therefore sufficient steps will need to be taken in order to avoid this
- Additionally, once the image has been inputted it is important to secure the image and make sure that the raw image cannot be accessed and breach the data protection act

Data

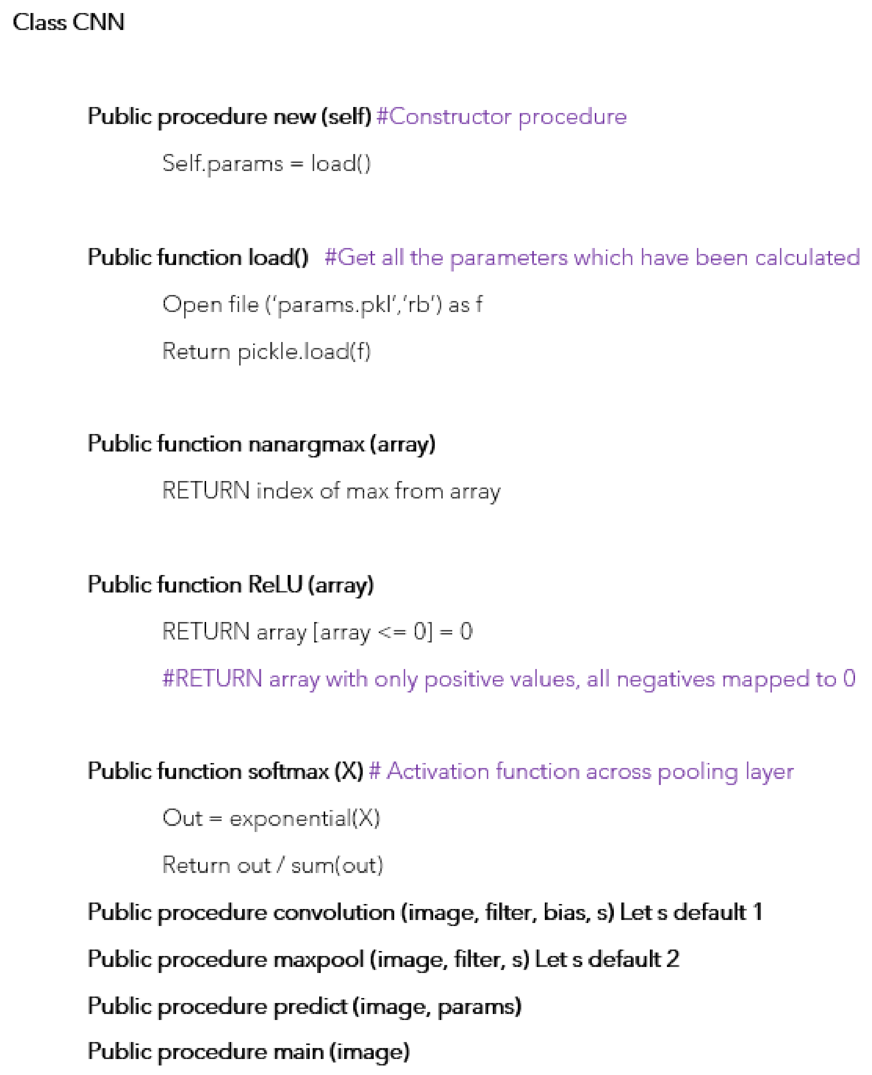
- Data protection act of 1998 & 2018 needs to be abided
- Means that if the user inputs any data onto the website it must be kept securely and deleted unless explicit permission for long term saving is present

- Additionally, all data should be sanitised to make sure that the correct data is inputted as well as to prevent phishing attacks on the system!

Algorithms

The algorithms for the final program will be split into 2 main python files, there will be a third python files used initially to calculate all the parameters. Additionally, there are 2 JavaScript files that will be utilised to perform tasks on the actual website. This structure will be used to help keep the project organised

Cnn.py

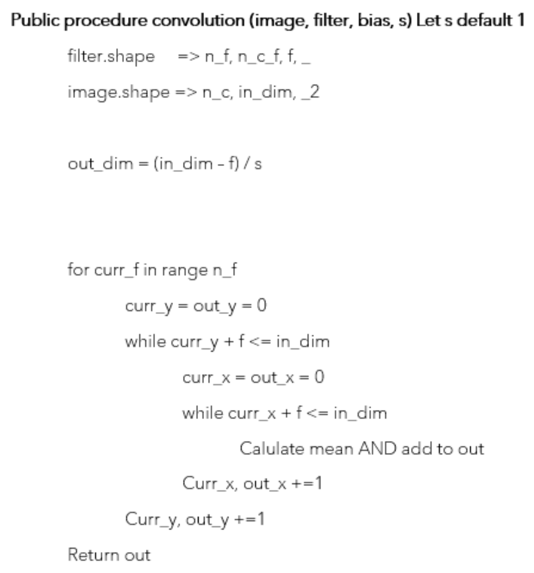


```
Class CNN

    Public procedure new (self) #Constructor procedure
        Self.params = load()

    Public function load()  #Get all the parameters which have been calculated
        Open file ('params.pkl','rb') as f
        Return pickle.load(f)

    Public function nanargmax (array)
        RETURN index of max from array

    Public function ReLU (array)
        RETURN array [array <= 0] = 0
        #RETURN array with only positive values, all negatives mapped to 0

    Public function softmax (X) # Activation function across pooling layer
        Out = exponential(X)
        Return out / sum(out)
    Public procedure convolution (image, filter, bias, s) Let s default 1
    Public procedure maxpool (image, filter, s) Let s default 2
    Public procedure predict (image, params)
    Public procedure main (image)
```

Here is the first python file which is a feedforward CNN, some longer procedures have not been included in this image but will be explained and justified.

The first two procedures are the new constructor and load. These will run upon instantiation and will take the latest parameters that are saved in a file called 'params.pkl' and returns them so that they can be loaded into memory. On one hand it may seem like a redundant process and that the variables can be hard coded into the algorithms however the parameters are many thousands of lines and pickle provides a way to process the variables almost instantly without a long loading time, therefore this stage is essential in order to create an application that can run quick enough for the end user

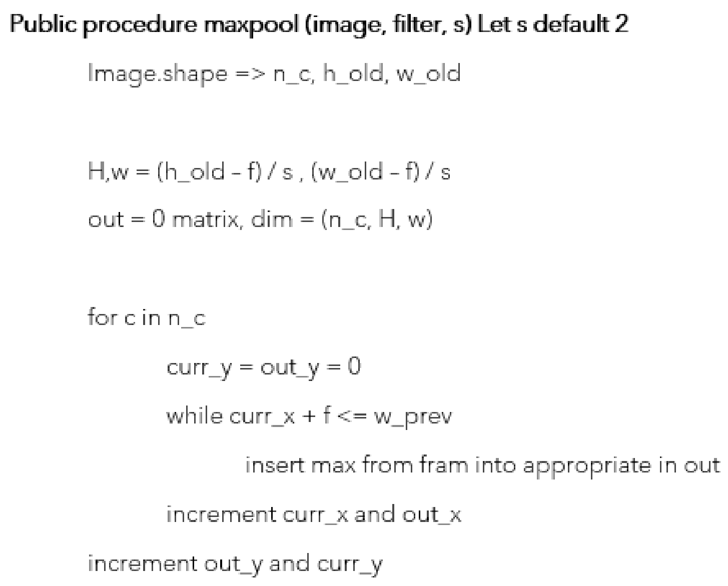
The next 3 procedures are Nanargmax, ReLU and SoftMax which are all essential functions inside the CNN which the next 4 bigger functions will rely on to perform their tasks



```
Public procedure convolution (image, filter, bias, s) Let s default 1
    filter.shape    => n_f, n_c_f, f, _
    image.shape => n_c, in_dim, _2

    out_dim = (in_dim - f) / s


    for curr_f in range n_f
        curr_y = out_y = 0
        while curr_y + f <= in_dim
            curr_x = out_x = 0
            while curr_x + f <= in_dim
                Calulate mean AND add to out
            Curr_x, out_x +=1
        Curr_y, out_y +=1
    Return out
```

The next procedure is the convolution procedure, this initially unravels the filter and image dimensions into different variable which can be used to check if the image has been processed correctly as well as being used on line 3 to calculate the output dimensions

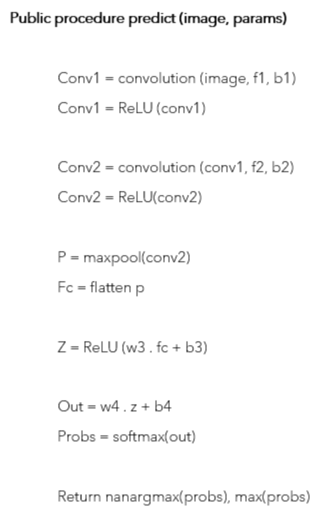
The for loop will then convolve over every part of the image making sure to add the bias term at each stage. The while loop Is used to make sure it only goes to the end of each line and does not go past; padding is important if dimensions are not perfect


```
Public procedure maxpool (image, filter, s) Let s default 2
        Image.shape => n_c, h_old, w_old


        H,w = (h_old - f) / s , (w_old - f) / s
        out = 0 matrix, dim = (n_c, H, w)


        for c in n_c
                curr_y = out_y = 0
                while curr_x + f <= w_prev
                        insert max from fram into appropriate in out
                increment curr_x and out_x
        increment out_y and curr_y
```

This procedure is at first hard to understand but is used to down sample the image a filter f and s as a stride. Like the previous procedure the first few lines are used to get the previous dimensions and use them to calculate the predicted new dimensions of the images

This for loop is used for each channel and in our case, we have 3 channels being R G and B. It will then slide over each part of the image and assign the max value at each stage to the output matrix

Increments are equivalent is x+=1


```
Public procedure predict (image, params)

        Conv1 = convolution (image, f1, b1)
        Conv1 = ReLU (conv1)

        Conv2 = convolution (conv1, f2, b2)
        Conv2 = ReLU(conv2)

        P = maxpool(conv2)
        Fc = flatten p

        Z = ReLU (w3 . fc + b3)

        Out = w4 . z + b4
        Probs = softmax(out)

        Return nanargmax(probs), max(probs)
```

The predict function is a composite of many different functions which are used to try and predict using the parameters if the image is cancerous or not

We start by running the image through each layer of the CNN, by convolving it twice and making sure the ReLU it to remove negative values in comparisons

This is then passed through the pooling layer and then flattened to allow it to be connected as nodes to the artificial neural network, which will predict using the nan argmax function

```
Public function main (image)

        Image.resize(28,28)
        Frame = Image as array reshaped as (3, 28, 28)
        X,y = predict (frame, params)

        Return X,y
```

This is the final subroutine of this file and is a function that allows for an image to be inputted then processed in an appropriate manner which will allow it to be passed into the predict function, where X is the diagnosis and y is the uncertainty in this judgement

Main.py

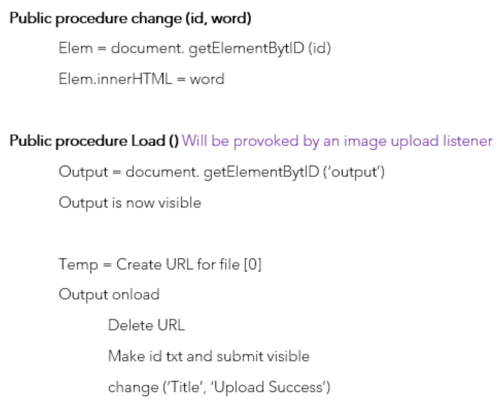
This is the front-end flask python page which allows for html to be rendered on the server, it consists of many static subroutines and the essential functions will be described

```
@path '/'
Public procedure main ()
        Return 'Home.html'
@path '/main'
Public procedure Detection ()
        If post
                F = request['image']
                Save F name = temp + original name
                X = CNN ()
                _ = X. main (F)
                Return _ into webpage

        Return 'Detection.html'
```

Here are the 2 main procedures which have been decorated with their path link. The main subroutine simply returns the html file onto the webpage

The second subroutine if there is a get request will automatically return the html file onto the webpage but if there is a post request that means that the user is attempting to submit information onto the website and to deal with this the code will request the image from the webpage and will then run the image through the CNN before returning the results of the processing back onto the webpage

JS Files



```
Public procedure change (id, word)
    Elem = document. getElementBytID (id)
    Elem.innerHTML = word

Public procedure Load () Will be provoked by an image upload listener
    Output = document. getElementBytID ('output')
    Output is now visible

    Temp = Create URL for file [0]
    Output onload
        Delete URL
        Make id txt and submit visible
        change ('Title', 'Upload Success')
```

These are 2 of the front-end procedures that have been utilised, the first one is a simple function to make the syntax of changing the state of an object within the website

The second subroutine is a simplified view of what will happen when an image is loaded on a website, at the beginning the output box is hidden therefore when this is triggered is will now be visible and then when the image has been fully loaded, a temporary URL will be created to store the image in blob space. Once the image has been saved the URL will then be revoked and txt and submit elements will now be visible before finally changing the title sentence from its original to your image has been uploaded awaiting confirmation that the user is happy with the image

Justifying Choices and Usability

In order to make the UI useable I have decided on having a navigation bar at the top of the website. An efficient navigation system will enable visitors to travel through the website seamlessly without having to look for links all around the webpage. Furthermore, it also acts a summary for the website being a short.

By choosing the colour to be white while the background of the page is an off-white grey colour, it will link in with the first point in the success criteria which helps to keep the website light and simple.

The more important part of this section will be to put a form of skeleton UI in the white rectangle on the right-hand side of the page. This would be to give the users an immediate understanding of how the system works and make sure that they fully understand the skin cancer detection service that we are providing. This will help to fulfil criteria 4 and 5 which state that the home page of the website should be simple but also explain what we do. Therefore, this section is essential!

Additionally, for this section, I have used a font called Segoe. As for the titles of the main sections, this helps to keep the text light due to its open and friendly type face. This helps to keep the website friendly which is very important because skin cancer is a serious issue, and it is important to attempt to calm the users down subconsciously. Also, the font has lots of weights which helps to establish a good visual hierarchy.

This visual hierarchy has been established in the hero section mainly. We can see this where the two action buttons are, these are 'Create an account' and 'Try anonymously.' Both are using the same font but the create an account is the primary call to action so has been given a bolder weighting as well as given the primary colour for the website (blue) as the background for the website which helps the main call bright and stand out on the website. This is extremely important, and it immediately incorporates the anonymous feature into the website which is on the success criteria and something that was unanimously want by all the people who were interviewed from the doctors to the potential users of the website.

This section is intended for users to scroll down to once they have read through the hero section and then chose to keep scrolling through the website therefore registering their interest in the website.

This section is short and succinct and therefore aims to give the user a concise guide to what the website offers and why people should sign up to our skin detection service.

The first thing that we can see here is that the background colour has been changed. This has gone from a grey into a gradient blue background. This will help to shift the focus of the user onto the centre of the page where all the main features of the website are. This will help to keep the users engaging with the website and work to emphasise the main points. This has been done to fulfil the criteria of keeping the website easy to use and very light on the user.

Furthermore, this section also allows us to increase the confidence if the users using this website which was an issue raised with the doctors and therefore having a section that fully explains what the website entails is important.

Key Variables and Data Structures

Name	Data Type/Class/Function	Use
Params	2D array of Matrices	They are used to form the specific model for the data and have been trained using backpropagation of the CNN To form this data, we feed forward around 10000 images through the neural network and work out the cost in batches and subsequently minimise this cost function. The actual values inside the matrices are floats but there are many 1000 values
Bias	Constant/Float	This is used as part of the parameters to prevent overfitting from occurring which Is important because that would mean that the parameters/weights will not become over reliant on the data points within the fully connected layer of the CNN
Image	2D array 28x28x3	These have 3 channels being RGB and also each image is 28 by 28 pixels, this includes all the input images in the training stages as well as during the final stage where these images will be stored temporarily in memory before being loaded into the RAM and then deleted from permanent storage
CNN	Class	This is a very important class that has been discussed within the pseudocode section where us is discussed how this class is used for prediction using the image and the very important parameters that are essential for running the CNN as a whole
Dx_dict	Dictionary/Record	This is important because the prediction class will only output a discrete number and this has to be translated into a diagnosis and this has to be decided if cancerous or not, which is held within this dictionary and able to be accessed using the key to the diagnosis or diagnosis numbers
Df	Matrix/ Data frame	This is most of the inputted features from the excel spreadsheet which is able to be accessed rapidly once uploaded from memory. This contains all the different samples for the data and is used because it is often quicker to access than the images
Secret_key	String	This is used to encrypt all traffic within the network and needs to be set out within the code for the website during the development stage to allow efficient access on the website

Validation routines:

In order to make sure that the program is robust, all the user inputs must be validated to make sure that they are inputting the correct data. Some examples are listed below

Inputting images

- The user inputted images is edited and then put back onto the webpage in order to give the user an opportunity to change their mind about the image that they have inputted in order to let them change their image if it does not fit the square frame and when cropped if the lesions is not within the frame

Buttons

- On the web user interface there are a lot of buttons which are linked to functions and often pass variables between different web pages and through the backend. To make sure that if the JS functions are changed the website index cannot be accessed, the flask backend double checks what the state of each appropriate variable is and if any are in the wrong state even though a POST request is made, the webpage will default

Text Boxes

- In order to prevent injections between the text boxes, all text is 'string-fied' and not treated as raw data which is not able to carry out its appropriate injection to make sure that the website is kept robust and safe from attackers in this respect
- Additionally, another precaution that could be utilised is making sure that only certain characters are validated

File Upload

- Here, even though an image file is requested, at first any file could have been inputted, therefore a java script validation function is being utilised which will check the input prior to its upload and check if it is a JPEG or PNG and any other files will be auto deleted in order to protect the website

Test Plan

Throughout the coding of this project, I have had to test each function and class or make sure that they complete their intended purpose and run without any faults. This will mean that any testing after the program is complete will highlight minor bugs or avenues for improvement

The testing for this should include all data inputted into the program and check at multiple stages if the data is being used correctly in order to complete its desired aim. This will mean constantly checking the terminal for any irregularities in inputs and placing breakpoints throughout the program to check if everything is running correctly because there are so many variables to check.

After the full development is complete, 1 big test should be completed where the inputs will be along the lines of one sample of skin that is clear not to have cancer and one which clearly has a cancerous lesion and then some boundary cases to check if the system is predicting correctly. Additionally destructive testing where file signatures are faked and to see how to website reacts.

I think it will probably be very important to record any cases which are inputted and show screenshots of the terminal for the interim programming stages in this. While doing this I should also go back to the stakeholders and work out any improvements which will be needed and to check is any requirements will need to be changed.

Iterative Development

I will be using an iterative development system to complete this application. This will mean that the program will be broken down into smaller solvable problems using a top-down approach. This has been shown in the diagrams earlier in this section. This will mean that after each stage is complete it will be gradually tested within each section in order to optimise each individual function before combining them all together. This will gradually create a prototype and will keep going until the application is complete

The testing will help this significantly as each prototype can be compared against the test criteria to check if the prototype functions correctly or not

Testing checklist

This is the list that I will use to test the functionality of the application that I create

Action to test	Working
Website loads with no console bugs	
Website runs in non-development mode	
Menu button functional	
Menu adaptive to screen size	
Menu has a drop-down menu with various links	
All links in menu redirect to correct page	
Info page will output general information to user	
Service page will output some information about how works	
Home page is auto loaded and discusses website	
Upload of file (JPEG or PNG)	
Upload of file (non-image)	
Upon upload image is edited	
After editing image is displayed	
After display image confirmation is triggered	
After image is displayed upload option removed	
Allow submit of image	
Allow rejection of image	
Image stored only temporarily	
Image file cannot be accessed	
Instructions on screen clear and well formatted	
After submission data loaded into CNN	
Results from CNN outputted onto website	
Test cases for different types of tumours	

When this thorough list is all complete, I will be sure that the functionality of this program is met and has no major errors, even though more features may come up which need to be tested, which will be gained from further stakeholder interviews.

At the stage where all these actions are working, then all the criteria on the success criteria should be complete and the development would be complete.

Interface inputs and test tables / Further Data Required

File Input

The image input as a file is the most crucial part to test effectively as it would be a big weak point if a user can add random files and data into the database.

This takes the part of the success criteria stating

- **Robust protected website**

Input	Test Type	Result
RGB image from database	Valid	
Skin Lesion image taken	Valid	
Logo of product	Extreme	
Document	Invalid	
B/W image (database)	Valid	
Another RGB image	Valid	
Exe file	Invalid	
Bat file	Invalid	

File Storage

Action/Check	Test Type	Result
RGB Image storage time	Valid	
Logo of product time	Valid	
Rejected image storage	Valid	
Bat File	Invalid	

Tumour test cases

This is incredibly important to make sure that the actual machine learning algorithm is fully functioning, the actual test set is much bigger, but this will be a small sample taken from there to view what is happening behind the scenes in the test stage of the CNN, where we work out a percentage of successes in a confusion matrix

This will fulfil the success criteria part

- **Fine-tuned ML algorithms**

Input	Test Type	Result
Cancer case 1	Valid	
Cancer case 2	Valid	
Cancer case 3	Valid	
Non-cancer case 1	Valid	
Non-cancer case 2	Valid	
Non-cancer case 3	Valid	
Clear skin	Invalid	

As part of **further data,** I should consider finding some skin samples from a separate database and images that I have actually taken for the terminal testing at the end of development to make sure the application is not subject to significant dataset bias.

To do this, I will ask friends and use my own skin samples to test if the product works or not. I could use images from google searches, but some images may be copyrighted, therefore this would not be a good idea!

Here is a lesion taken from Subil Philip Ⓒ

Development

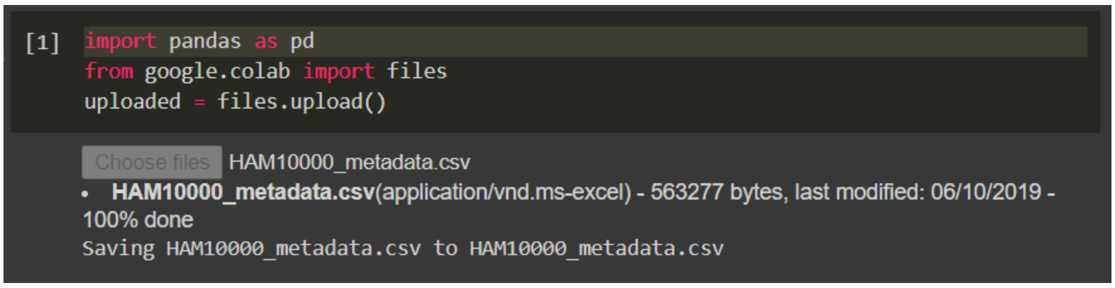
Iterative

1. Data Analysis
2. Build CNN
3. Train CNN
4. Build website
5. Final Testing

Each stage has been tested constantly and inline tests have been described as well as final tests at the end of each section. Most of the testing has then been accumulated into stage 5, which is the testing section and some of the inline tests have been repeated for clarity as well as to make sure that the product still passes all the initial tests from previous stages

The stages have been decomposed in this fashion for the suitability of the project and in order to create sections that overlap the least within the project and will be modularised from one another therefore be less reliant on each other.

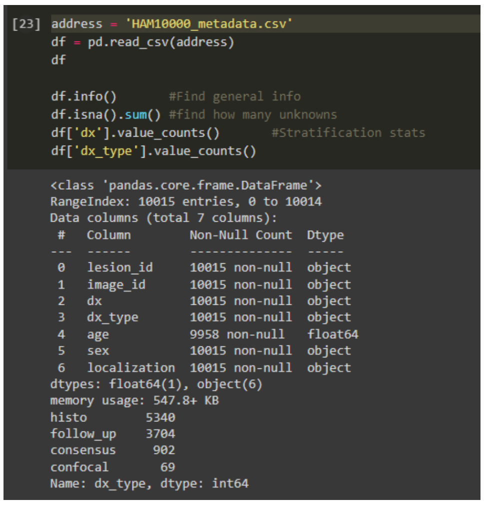
Stage 1 Data Analysis



```
[1] import pandas as pd
    from google.colab import files
    uploaded = files.upload()
```

```
Choose files  HAM10000_metadata.csv
• HAM10000_metadata.csv(application/vnd.ms-excel) - 563277 bytes, last modified: 06/10/2019 -
100% done
Saving HAM10000_metadata.csv to HAM10000_metadata.csv
```

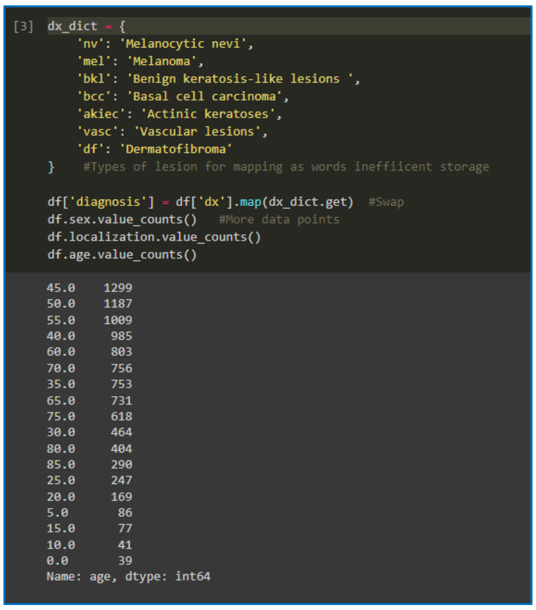
Download the dataset metadata into the RAM test passed as seen in terminal (Upload successful)



```
[23] address = 'HAM10000_metadata.csv'
     df = pd.read_csv(address)
     df

     df.info()        #Find general info
     df.isna().sum() #find how many unknowns
     df['dx'].value_counts()        #Stratification stats
     df['dx_type'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10015 entries, 0 to 10014
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   lesion_id     10015 non-null  object
 1   image_id      10015 non-null  object
 2   dx            10015 non-null  object
 3   dx_type       10015 non-null  object
 4   age           9958 non-null   float64
 5   sex           10015 non-null  object
 6   localization  10015 non-null  object
dtypes: float64(1), object(6)
memory usage: 547.8+ KB
histo        5340
follow_up    3704
consensus     902
confocal       69
Name: dx_type, dtype: int64
```

View the initial data set and check for initial criteria, in this case we get the general info about how many rows and columns there are and then we find out how many N/A counts that there are which will need to be mean imputed.

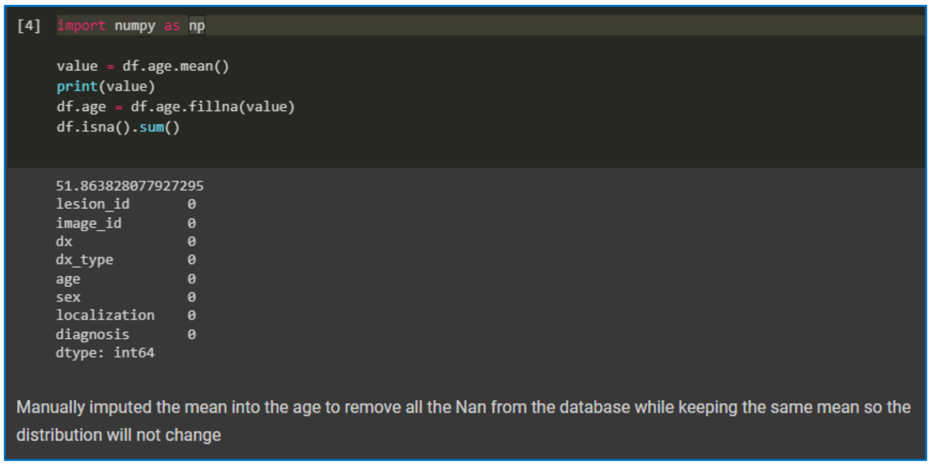


```
[3] dx_dict = {
        'nv': 'Melanocytic nevi',
        'mel': 'Melanoma',
        'bkl': 'Benign keratosis-like lesions ',
        'bcc': 'Basal cell carcinoma',
        'akiec': 'Actinic keratoses',
        'vasc': 'Vascular lesions',
        'df': 'Dermatofibroma'
    }    #Types of lesion for mapping as words ineffiicent storage

    df['diagnosis'] = df['dx'].map(dx_dict.get)  #Swap
    df.sex.value_counts()   #More data points
    df.localization.value_counts()
    df.age.value_counts()
```

```
45.0    1299
50.0    1187
55.0    1009
40.0     985
60.0     803
70.0     756
35.0     753
65.0     731
75.0     618
30.0     464
80.0     404
85.0     290
25.0     247
20.0     169
5.0       86
15.0      77
10.0      41
0.0       39
Name: age, dtype: int64
```

Here the values are from the database analysed have been mapped onto their actual values

Additionally, more data points are taken

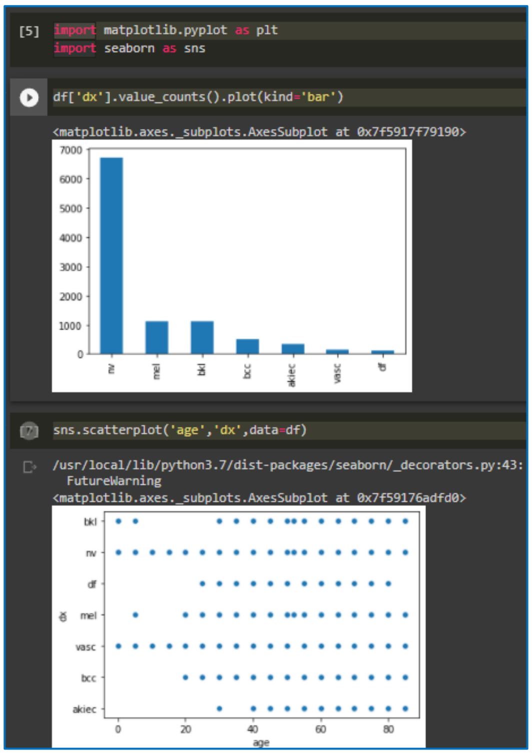


```
[4] import numpy as np

    value = df.age.mean()
    print(value)
    df.age = df.age.fillna(value)
    df.isna().sum()
```

```
51.863828077927295
lesion_id       0
image_id        0
dx              0
dx_type         0
age             0
sex             0
localization    0
diagnosis       0
dtype: int64
```

Manually imputed the mean into the age to remove all the Nan from the database while keeping the same mean so the distribution will not change

Mean imputing the means into the database, test passed as seen above, no more NA in database



```
[5] import matplotlib.pyplot as plt
    import seaborn as sns
```

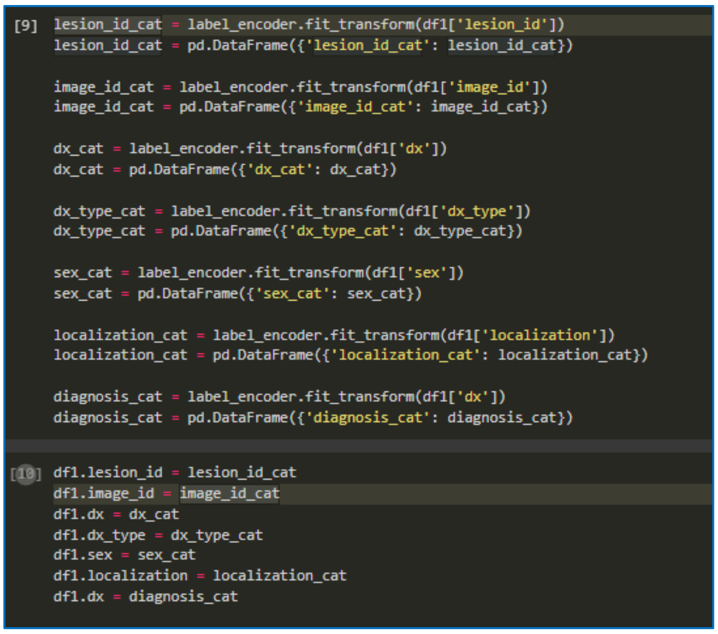
```
df['dx'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5917f79190>
```

```
sns.scatterplot('age','dx',data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f59176adfd0>
```

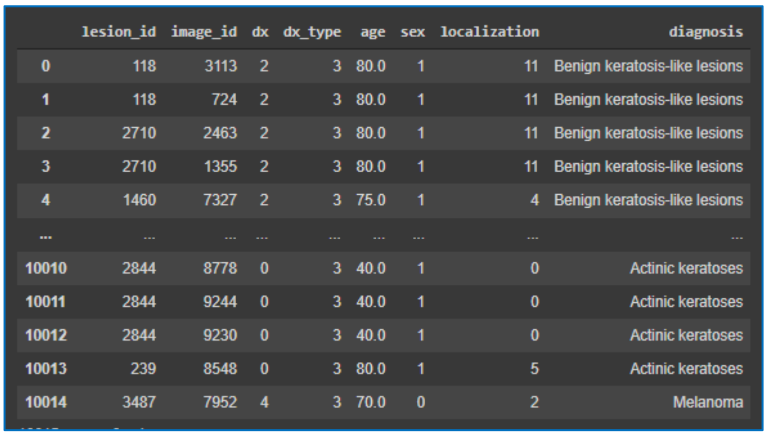
More analysis with some different charts being formed that have been talked about in the design and analysis section. Test passed in graphs that can be seen above.



```
[9]  lesion_id_cat = label_encoder.fit_transform(df1['lesion_id'])
     lesion_id_cat = pd.DataFrame({'lesion_id_cat': lesion_id_cat})

     image_id_cat = label_encoder.fit_transform(df1['image_id'])
     image_id_cat = pd.DataFrame({'image_id_cat': image_id_cat})

     dx_cat = label_encoder.fit_transform(df1['dx'])
     dx_cat = pd.DataFrame({'dx_cat': dx_cat})

     dx_type_cat = label_encoder.fit_transform(df1['dx_type'])
     dx_type_cat = pd.DataFrame({'dx_type_cat': dx_type_cat})

     sex_cat = label_encoder.fit_transform(df1['sex'])
     sex_cat = pd.DataFrame({'sex_cat': sex_cat})

     localization_cat = label_encoder.fit_transform(df1['localization'])
     localization_cat = pd.DataFrame({'localization_cat': localization_cat})

     diagnosis_cat = label_encoder.fit_transform(df1['dx'])
     diagnosis_cat = pd.DataFrame({'diagnosis_cat': diagnosis_cat})
```

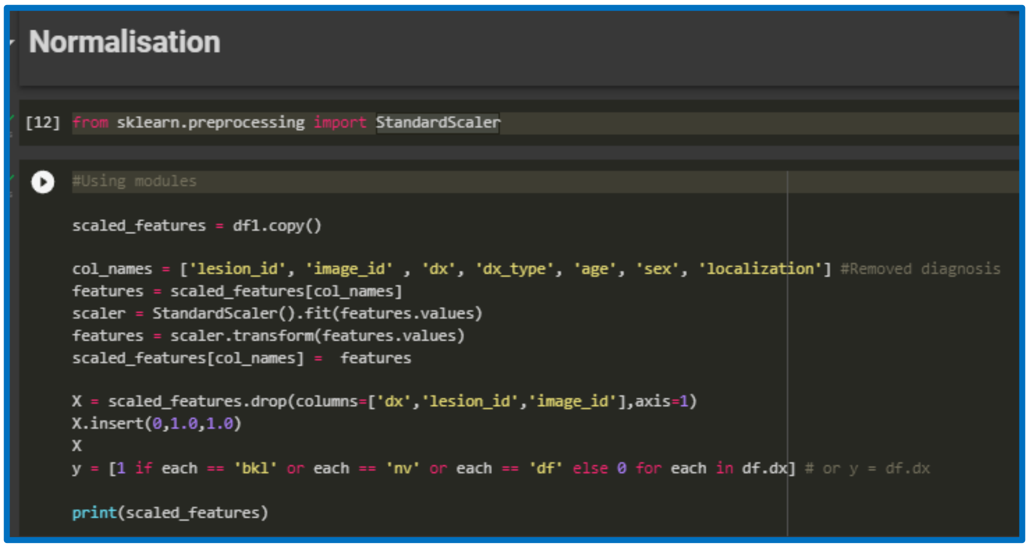
```
[10] df1.lesion_id = lesion_id_cat
     df1.image_id = image_id_cat
     df1.dx = dx_cat
     df1.dx_type = dx_type_cat
     df1.sex = sex_cat
     df1.localization = localization_cat
     df1.dx = diagnosis_cat
```

This is the bulky code that label encodes the data to prevent bias from occurring due to the values of the codes, we have used one hot encoding. Test passed as shown below



	lesion_id	image_id	dx	dx_type	age	sex	localization	diagnosis
0	118	3113	2	3	80.0	1	11	Benign keratosis-like lesions
1	118	724	2	3	80.0	1	11	Benign keratosis-like lesions
2	2710	2463	2	3	80.0	1	11	Benign keratosis-like lesions
3	2710	1355	2	3	80.0	1	11	Benign keratosis-like lesions
4	1460	7327	2	3	75.0	1	4	Benign keratosis-like lesions
...	...	...	...	...	...	...	...	...
10010	2844	8778	0	3	40.0	1	0	Actinic keratoses
10011	2844	9244	0	3	40.0	1	0	Actinic keratoses
10012	2844	9230	0	3	40.0	1	0	Actinic keratoses
10013	239	8548	0	3	80.0	1	5	Actinic keratoses
10014	3487	7952	4	3	70.0	0	2	Melanoma

This is what the data looks like before being normalised

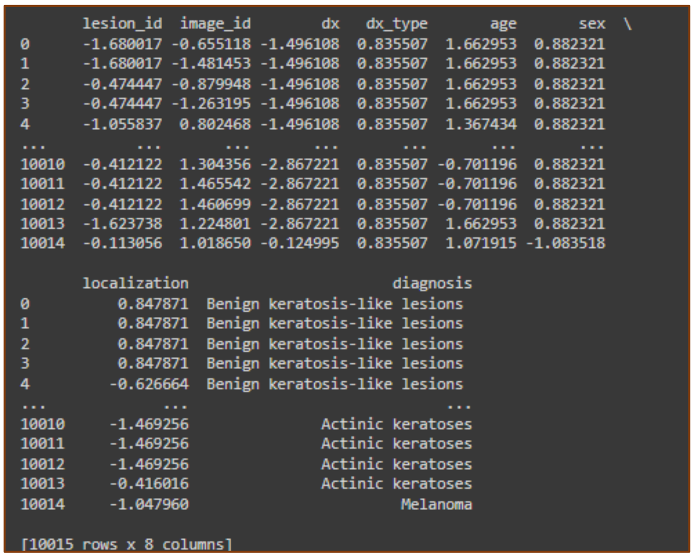


Normalisation

```
[12] from sklearn.preprocessing import StandardScaler
```

```
#Using modules

scaled_features = df1.copy()

col_names = ['lesion_id', 'image_id' , 'dx', 'dx_type', 'age', 'sex', 'localization'] #Removed diagnosis
features = scaled_features[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features[col_names] =  features

X = scaled_features.drop(columns=['dx','lesion_id','image_id'],axis=1)
X.insert(0,1.0,1.0)
X
y = [1 if each == 'bkl' or each == 'nv' or each == 'df' else 0 for each in df.dx] # or y = df.dx

print(scaled_features)
```

Here is the beginning of the normalisation section of the code



```
       lesion_id  image_id        dx   dx_type       age       sex  \
0      -1.680017 -0.655118 -1.496108  0.835507  1.662953  0.882321
1      -1.680017 -1.481453 -1.496108  0.835507  1.662953  0.882321
2      -0.474447 -0.879948 -1.496108  0.835507  1.662953  0.882321
3      -0.474447 -1.263195 -1.496108  0.835507  1.662953  0.882321
4      -1.055837  0.802468 -1.496108  0.835507  1.367434  0.882321
...          ...       ...       ...       ...       ...       ...
10010  -0.412122  1.304356 -2.867221  0.835507 -0.701196  0.882321
10011  -0.412122  1.465542 -2.867221  0.835507 -0.701196  0.882321
10012  -0.412122  1.460699 -2.867221  0.835507 -0.701196  0.882321
10013  -1.623738  1.224801 -2.867221  0.835507  1.662953  0.882321
10014  -0.113056  1.018650 -0.124995  0.835507  1.071915 -1.083518

       localization                       diagnosis
0          0.847871  Benign keratosis-like lesions
1          0.847871  Benign keratosis-like lesions
2          0.847871  Benign keratosis-like lesions
3          0.847871  Benign keratosis-like lesions
4         -0.626664  Benign keratosis-like lesions
...             ...                             ...
10010     -1.469256               Actinic keratoses
10011     -1.469256               Actinic keratoses
10012     -1.469256               Actinic keratoses
10013     -0.416016               Actinic keratoses
10014     -1.047960                        Melanoma

[10015 rows x 8 columns]
```

Here is what the normalised data looks like which is the data that we shall be using (Passed Test)

On line 5 the columns have had to be removed due to errors and then improvements have been made by removing the columns because the words were creating issues un the coding, which needed to be fixed using the current lines of code.

Stage 1 Review

The data analysis is now complete. This takes the raw data and metadata and checks for any patterns and then normalised the data as well as scale and one hot encode all the features.

This has prepared the rest of the project to be completed by completing all the prerequisites therefore does not fulfil many success criteria points however is an essential process for the development of the project.

How is has been tested

Every single function has been initially tested by inputting simple values and tables and manually checking their distributions. I then added more complicated tables to see what the values tested. Most of these can be seen within the terminal screenshots underneath each section of code. This is very explanatory and shows the output after every section of code and this can be physically analysed to check the validity and functionality of each function.

This stage was proceeded by inputting the large full database as well as the metadata database which allowed me to view how the functions work with the expected load. This section is extremely important for me to prepare for further stages of development.

Additionally, most testing can be viewed in the small console underneath each function where the output from each function and section of code can be viewed from each time that it has been viewed. This provided the majority of testing, and most have been passed although any fails have been discussed above and adapted. These have been highlighted.

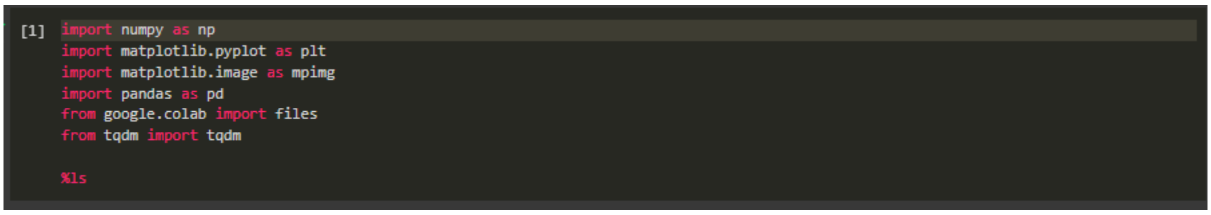
Success Criteria

Accurate Data Analysis	Screenshot each piece of code used for analysis

Changes

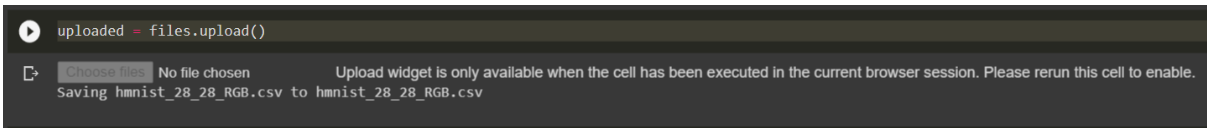
I decided that I will not be able to run my machine learning environment on my local machine therefore investigated using a new IDE being google collab for further development due to the size of data and large amounts of data crunching needed.

Stage 2 Build CNN



```
[1] import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import pandas as pd
    from google.colab import files
    from tqdm import tqdm

    %ls
```

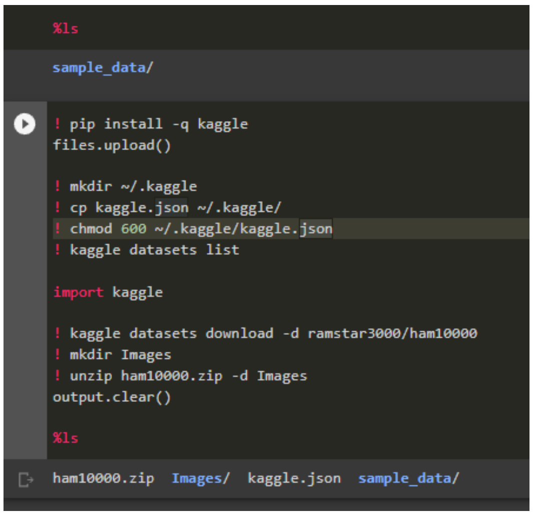
Import the relevant modules, for the program, at first these need to be imported like this but for end development, we will need to make a requirements.txt with all the modules and their versions, therefore this is a preliminary version of this code as it is not fully updated. Additionally, in the final product, not all of these modules will need to be included.



```
uploaded = files.upload()

Choose files  No file chosen     Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving hmnist_28_28_RGB.csv to hmnist_28_28_RGB.csv
```

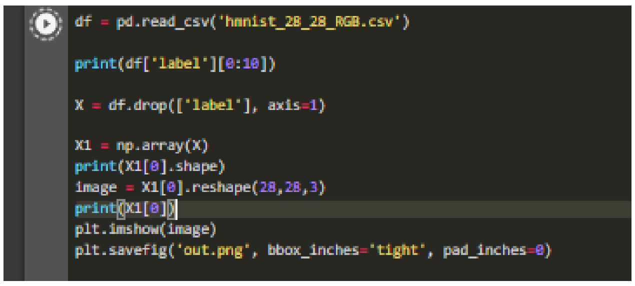
This is where the data is inputted into the program, at first, I chose to upload the images, however the amount of processing power needed to do this was extravagant, so instead I chose to change my approach and choose the HMINST data which chooses the 28x28x3 pixel array for use within the program. Successful Upload = Failed test

This code took around 40 minutes to upload into the database therefore in order to fix this, I have changed the design of this into using a Kaggle dataset in order to import the data after uploading onto Kaggle myself.



```
%ls

sample_data/

! pip install -q kaggle
files.upload()

! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets list

import kaggle

! kaggle datasets download -d ramstar3000/ham10000
! mkdir Images
! unzip ham10000.zip -d Images
output.clear()

%ls

ham10000.zip  Images/  kaggle.json  sample_data/
```

Here is the new code that has been written which now fulfils the requirement to be complete within the time period and therefore makes subsequent development stages much quicker than in previous



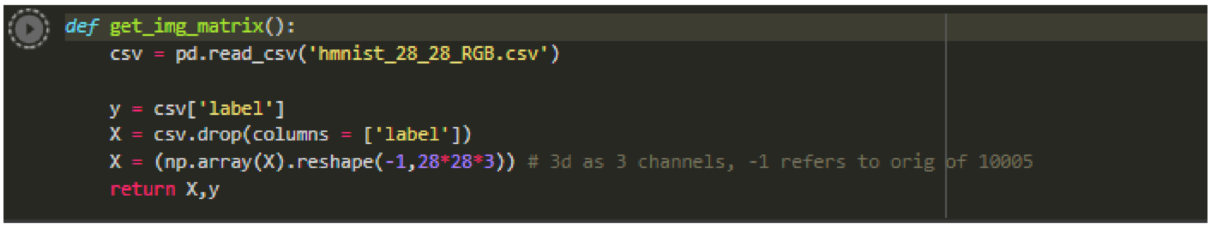
```
df = pd.read_csv('hmnist_28_28_RGB.csv')

print(df['label'][0:10])

X = df.drop(['label'], axis=1)

X1 = np.array(X)
print(X1[0].shape)
image = X1[0].reshape(28,28,3)
print(X1[0])
plt.imshow(image)
plt.savefig('out.png', bbox_inches='tight', pad_inches=0)
```

iterations therefore this time test has now been passed with the proof of the images folder now being created.

Here, the data has been inputted (which takes about one hour to upload every time), then we have to remove the labels because the CNN should not be given these details.

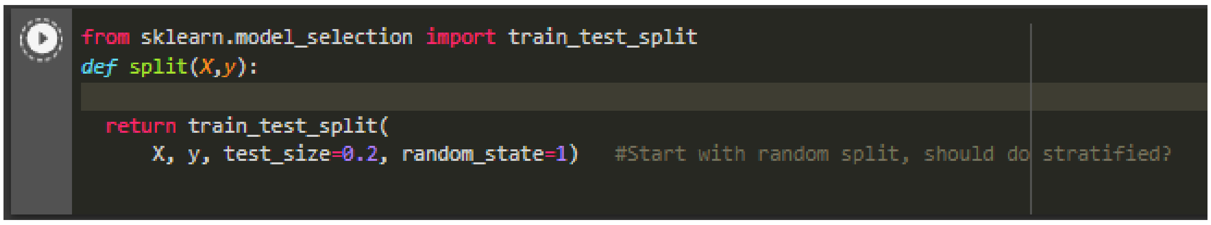
The next stage is to view an image from the dataset to find the format. All the rest of the code will be dependent on this; therefore, it is a very important stage and needs to be coded carefully. Here are some image outputs and we can see that the images are of the form 28x28x3. This stage has been -passed because it has run through without any matrix compatibility errors.

Image 1:



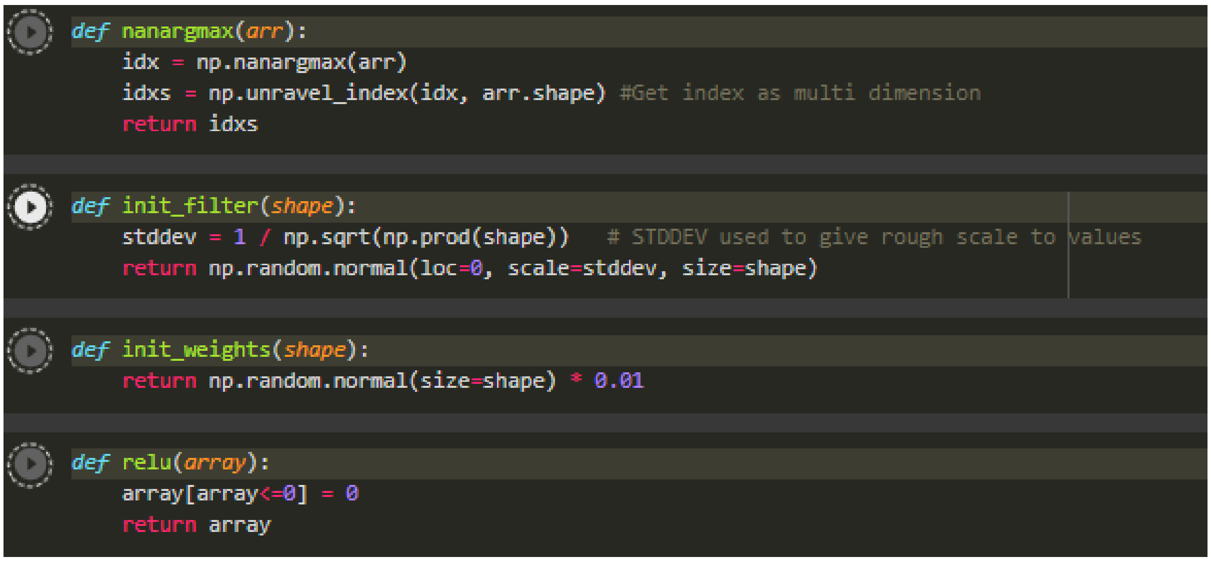
```
def get_img_matrix():
    csv = pd.read_csv('hmnist_28_28_RGB.csv')

    y = csv['label']
    X = csv.drop(columns = ['label'])
    X = (np.array(X).reshape(-1,28*28*3)) # 3d as 3 channels, -1 refers to orig of 10005
    return X,y
```

This is the first stage of the CNN, which takes the HMINST dataset which is stored in memory and takes the label and separates them from their specific instance in a form of data abstraction from the main dataset so the CNN cannot access all the data. Additionally, each image has been given its own 'row' within the set.



```
from sklearn.model_selection import train_test_split
def split(X,y):

  return train_test_split(
      X, y, test_size=0.2, random_state=1)   #Start with random split, should do stratified?
```

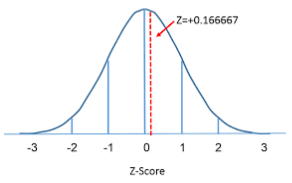
This is a simple algorithm I have made in order to split the dataset into its elementary sections being a random split at the moment however, in later development, this may need to be changed.



```python
def nanargmax(arr):
    idx = np.nanargmax(arr)
    idxs = np.unravel_index(idx, arr.shape) #Get index as multi dimension
    return idxs

def init_filter(shape):
    stddev = 1 / np.sqrt(np.prod(shape))    # STDDEV used to give rough scale to values
    return np.random.normal(loc=0, scale=stddev, size=shape)

def init_weights(shape):
    return np.random.normal(size=shape) * 0.01

def relu(array):
    array[array<=0] = 0
    return array
```

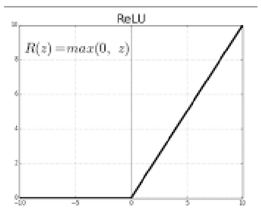
These are all specific functions for within the CNN, nan argmax is used to deal with lots of NaN arguments within, the program

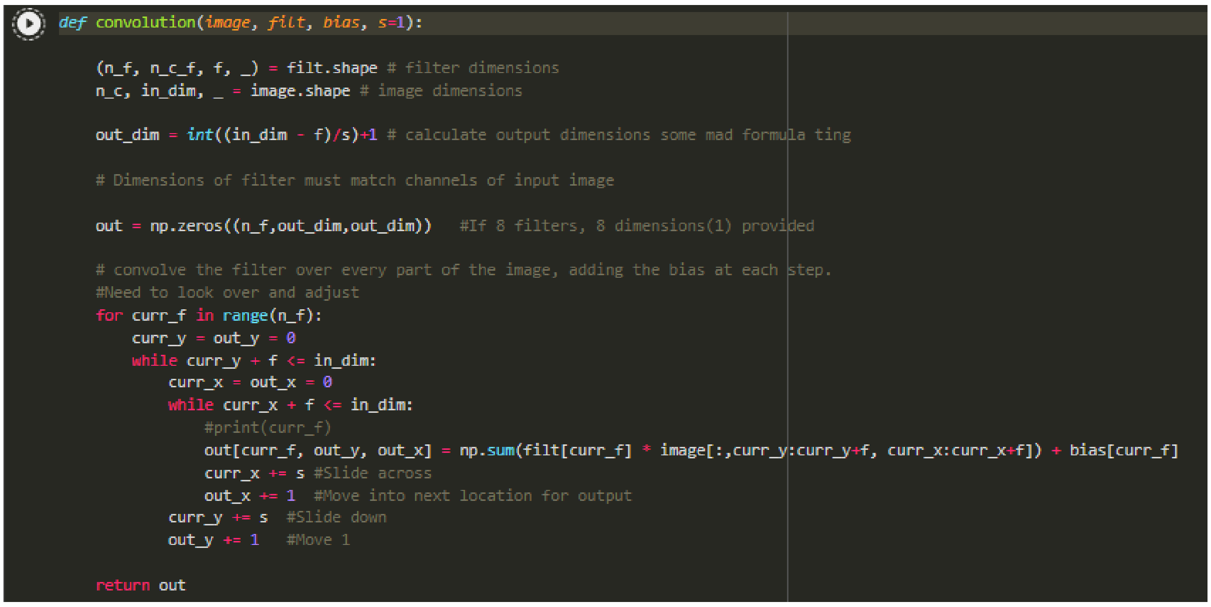
The INIT filter will be used to generate random number about the standard normal distribution.



The INIT weight's function does the same thing with a different size and is less scaled

The ReLU function is also called the rectifier function which is a very specific maths function

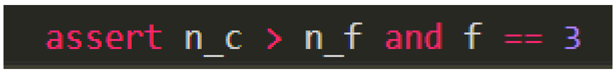




```python
def convolution(image, filt, bias, s=1):

    (n_f, n_c_f, f, _) = filt.shape # filter dimensions
    n_c, in_dim, _ = image.shape # image dimensions

    out_dim = int((in_dim - f)/s)+1 # calculate output dimensions some mad formula ting

    # Dimensions of filter must match channels of input image

    out = np.zeros((n_f,out_dim,out_dim))   #If 8 filters, 8 dimensions(1) provided

    # convolve the filter over every part of the image, adding the bias at each step.
    #Need to look over and adjust
    for curr_f in range(n_f):
        curr_y = out_y = 0
        while curr_y + f <= in_dim:
            curr_x = out_x = 0
            while curr_x + f <= in_dim:
                #print(curr_f)
                out[curr_f, out_y, out_x] = np.sum(filt[curr_f] * image[:,curr_y:curr_y+f, curr_x:curr_x+f]) + bias[curr_f]
                curr_x += s #Slide across
                out_x += 1  #Move into next location for output
            curr_y += s  #Slide down
            out_y += 1   #Move 1

    return out
```

This stage was the hardest stage to code as required the most factual knowledge about the CNN and took a long time to work out how every single stage functioned.

One stage which is not present here which has been added is assertion statements, because the on line one and two of the function, the variables have been unpacked dimensions and these need to be compared to make sure that then the dimensions are compatible with no overlaps or other errors.

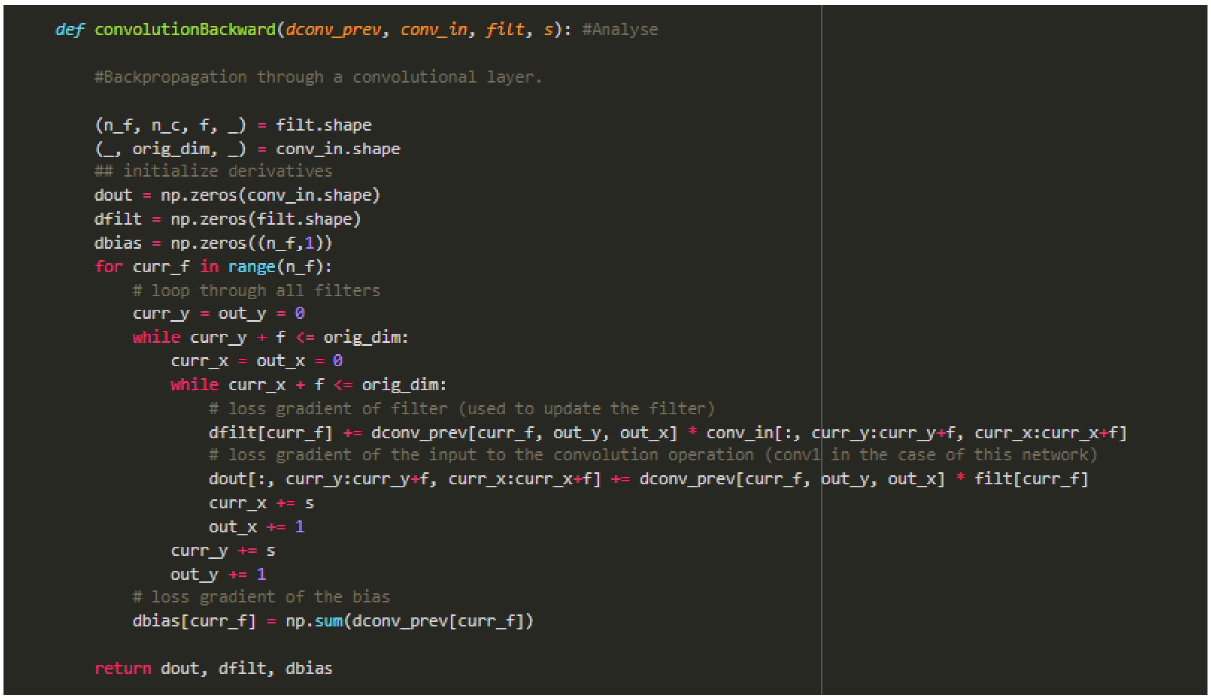


```
assert n_c > n_f and f == 3
```

This addition on line 3 Is very important to stop bad data from being imputing into the CNN network, so has been added into the source code.

Next, an empty matrix of 0 has been generated to represent the output of this function, these values will be populated after the loop has finished running.

Therefore, this test is now passed

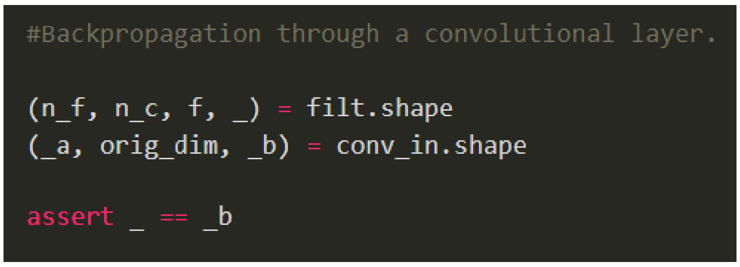


```
def convolutionBackward(dconv_prev, conv_in, filt, s): #Analyse

    #Backpropagation through a convolutional layer.

    (n_f, n_c, f, _) = filt.shape
    (_, orig_dim, _) = conv_in.shape
    ## initialize derivatives
    dout = np.zeros(conv_in.shape)
    dfilt = np.zeros(filt.shape)
    dbias = np.zeros((n_f,1))
    for curr_f in range(n_f):
        # loop through all filters
        curr_y = out_y = 0
        while curr_y + f <= orig_dim:
            curr_x = out_x = 0
            while curr_x + f <= orig_dim:
                # loss gradient of filter (used to update the filter)
                dfilt[curr_f] += dconv_prev[curr_f, out_y, out_x] * conv_in[:, curr_y:curr_y+f, curr_x:curr_x+f]
                # loss gradient of the input to the convolution operation (conv1 in the case of this network)
                dout[:, curr_y:curr_y+f, curr_x:curr_x+f] += dconv_prev[curr_f, out_y, out_x] * filt[curr_f]
                curr_x += s
                out_x += 1
            curr_y += s
            out_y += 1
        # loss gradient of the bias
        dbias[curr_f] = np.sum(dconv_prev[curr_f])

    return dout, dfilt, dbias
```

This function acts as the inverse of the convolution function, it is used in order to fit the parameters and adjust them slightly, however, the parameters will not be in a compatible size and format, therefore this code allows the parameters to be scaled appropriately depending on the size of the convolution filter.

Failed Test:

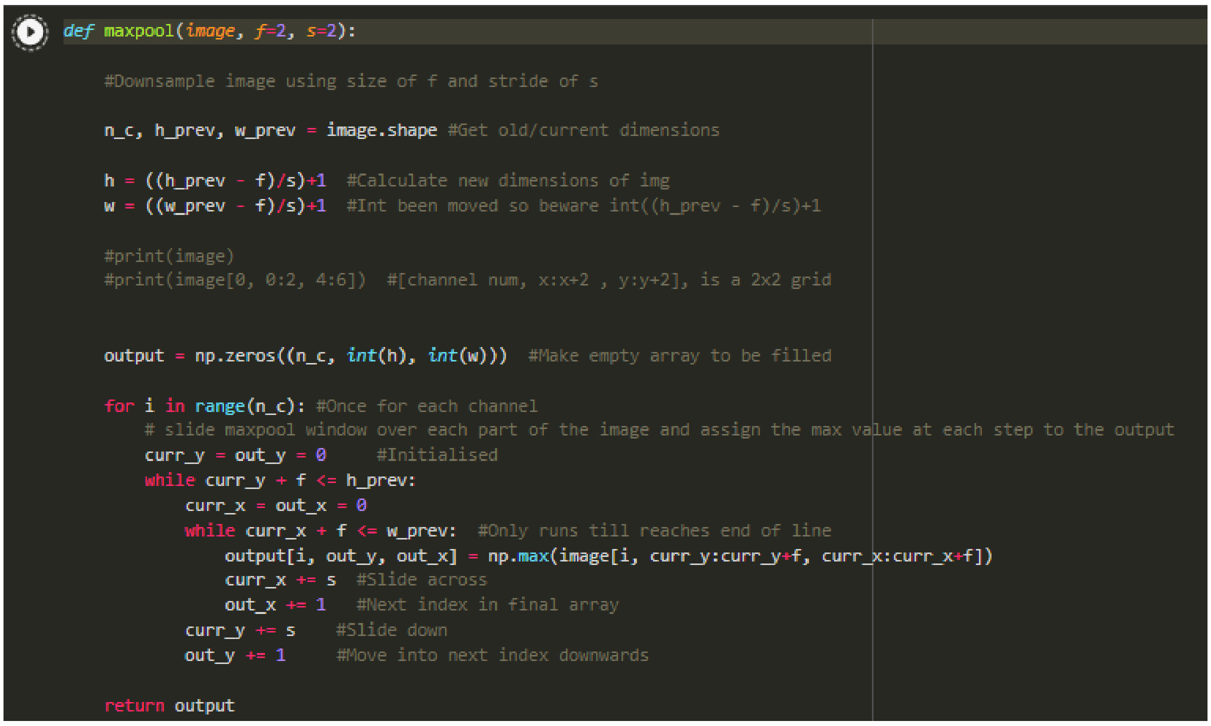
The functions starts of by extracting the filter and convolution sizes, which like explained above has assertion statements added to make sure that _ =_ in this picture, therefore the variables needed to be renamed before this stage could pass testing



```python
#Backpropagation through a convolutional layer.

(n_f, n_c, f, _) = filt.shape
(_a, orig_dim, _b) = conv_in.shape

assert _ == _b
```

This was the change that had to be completed

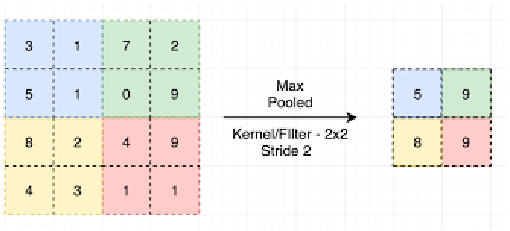
The function then calculates the expected output size then create a blank output matrix after using the formula. It will then populate the output matrix with the appropriate parameters using the nested while loops and the parameter unrolling formulae using the loss gradient of the filter.



```python
def maxpool(image, f=2, s=2):

    #Downsample image using size of f and stride of s

    n_c, h_prev, w_prev = image.shape #Get old/current dimensions

    h = ((h_prev - f)/s)+1  #Calculate new dimensions of img
    w = ((w_prev - f)/s)+1  #Int been moved so beware int((h_prev - f)/s)+1

    #print(image)
    #print(image[0, 0:2, 4:6])  #[channel num, x:x+2 , y:y+2], is a 2x2 grid


    output = np.zeros((n_c, int(h), int(w)))  #Make empty array to be filled

    for i in range(n_c): #Once for each channel
        # slide maxpool window over each part of the image and assign the max value at each step to the output
        curr_y = out_y = 0     #Initialised
        while curr_y + f <= h_prev:
            curr_x = out_x = 0
            while curr_x + f <= w_prev:  #Only runs till reaches end of line
                output[i, out_y, out_x] = np.max(image[i, curr_y:curr_y+f, curr_x:curr_x+f])
                curr_x += s  #Slide across
                out_x += 1   #Next index in final array
            curr_y += s    #Slide down
            out_y += 1     #Move into next index downwards

    return output
```

This is another essential function within the model of a CNN

The design of this function is to replicate what can be seen within the images, where the image is split into multiple different sections and then finds the maximum value within this section



This test was passed by inputting small matrices into it

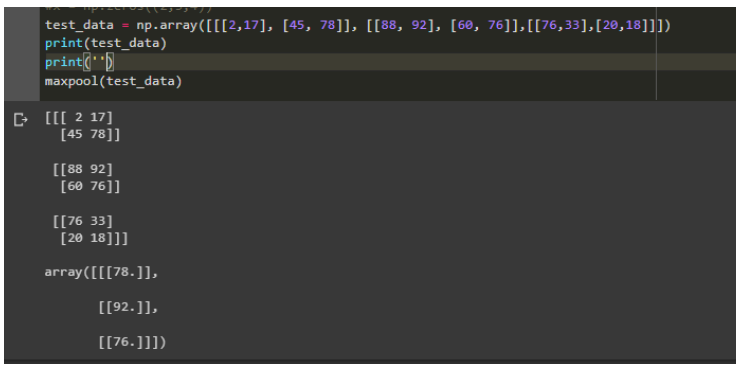
This function starts like most others have been unrolling the old dimensions from the image using the NumPy function shape

It then uses a formula (explained in comments) to work out the new image dimensions

Next, the function uses nested while loops to move across the image and choose the maximum values that the function can use by populating the max pooled matrix

For this, a for loop was added afterwards, because I realised that because the images were in colour, therefore had 3 channels, being R G and B which each needed to be analysed separately, however, this was noticed very early in development so is already added within the code

Additionally, the sliding across was hard to work out which variable was being incremented by s and which by 1, therefore this section had to be tested multiple times with specific data to make sure that there were no errors due to indexing out the matrix and to make sure that each channel was considered separately.

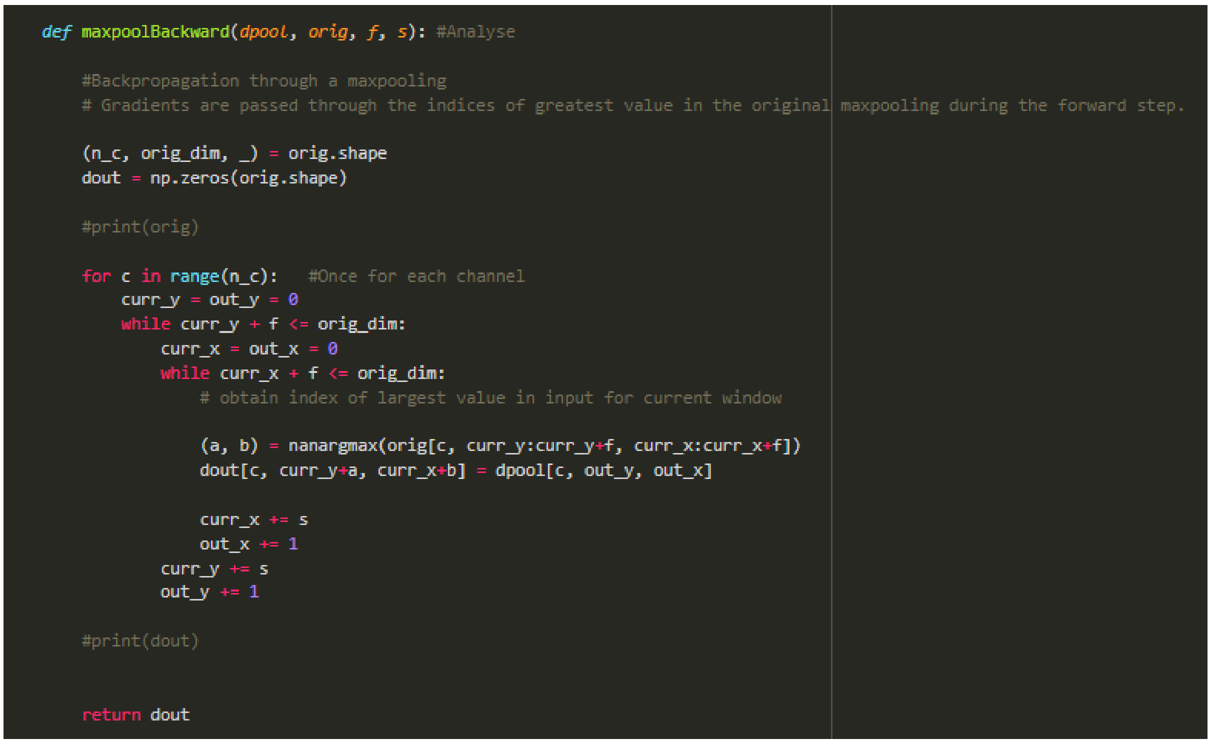


```
test_data = np.array([[[2,17], [45, 78]], [[88, 92], [60, 76]],[[76,33],[20,18]]])
print(test_data)
print('')
maxpool(test_data)
```

```
[[[ 2 17]
  [45 78]]

 [[88 92]
  [60 76]]

 [[76 33]
  [20 18]]]

array([[[78.]],

       [[92.]],

       [[76.]]])
```

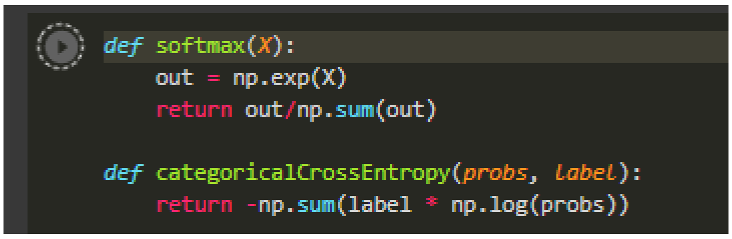
Here is the test example for the max pool layer and this test is an evident pass after viewing the data and noticing that the maximum for each pool has been taken.

A linear test like this is hard for maxpool backwards because the calculations are too complex.



```
def maxpoolBackward(dpool, orig, f, s): #Analyse

    #Backpropagation through a maxpooling
    # Gradients are passed through the indices of greatest value in the original maxpooling during the forward step.

    (n_c, orig_dim, _) = orig.shape
    dout = np.zeros(orig.shape)

    #print(orig)

    for c in range(n_c):   #Once for each channel
        curr_y = out_y = 0
        while curr_y + f <= orig_dim:
            curr_x = out_x = 0
            while curr_x + f <= orig_dim:
                # obtain index of largest value in input for current window

                (a, b) = nanargmax(orig[c, curr_y:curr_y+f, curr_x:curr_x+f])
                dout[c, curr_y+a, curr_x+b] = dpool[c, out_y, out_x]

                curr_x += s
                out_x += 1
            curr_y += s
            out_y += 1

    #print(dout)


    return dout
```

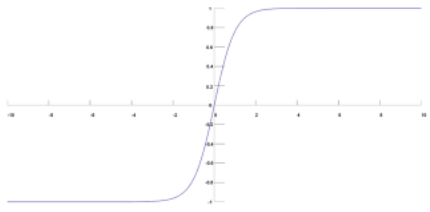
This is the matching function for the maxpool, which was much easier than the previous function because the backpropagation format had already been set out by all the previous backward functions and the max pool turned out to be a relatively simple function to decompose and reverse.

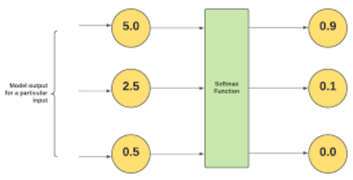


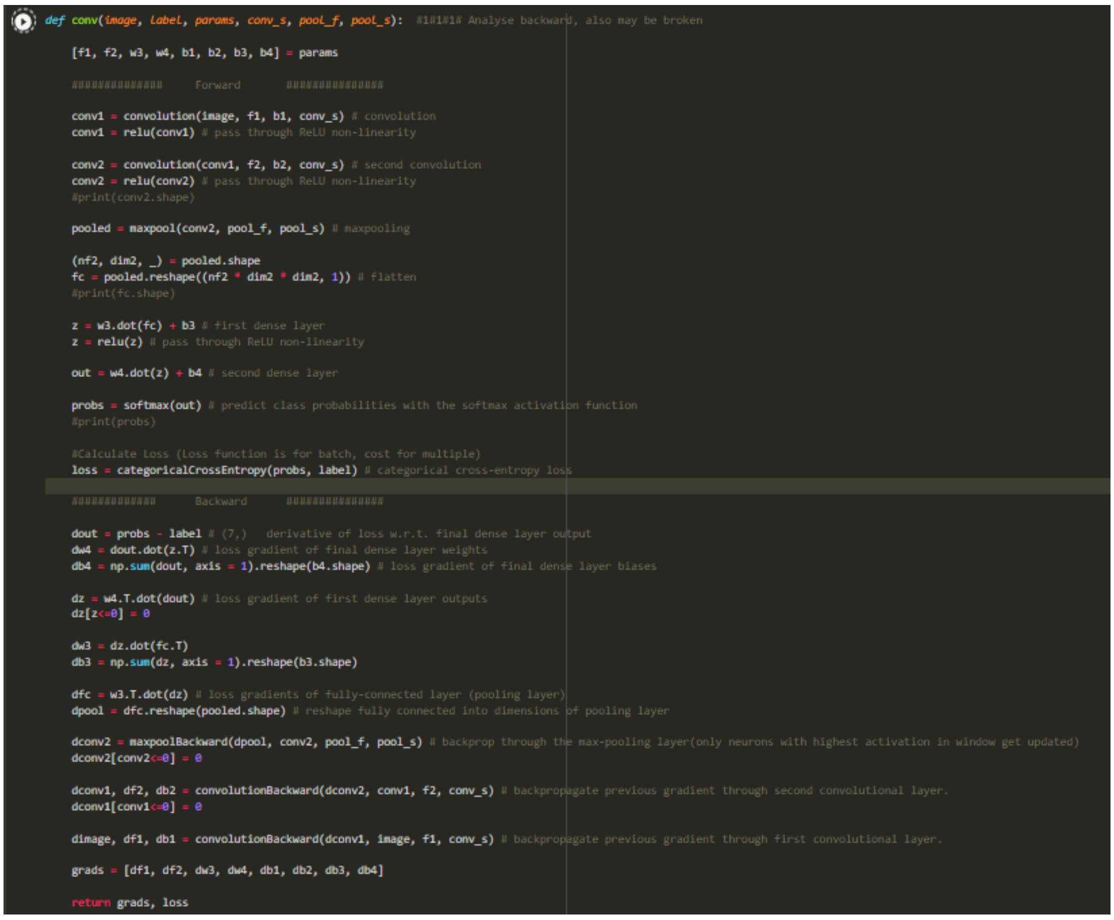
```python
def softmax(X):
    out = np.exp(X)
    return out/np.sum(out)

def categoricalCrossEntropy(probs, label):
    return -np.sum(label * np.log(probs))
```

Here are 2 more functions that were needed in order for my CNN to function.

The first of these is the SoftMax function which will map X using this





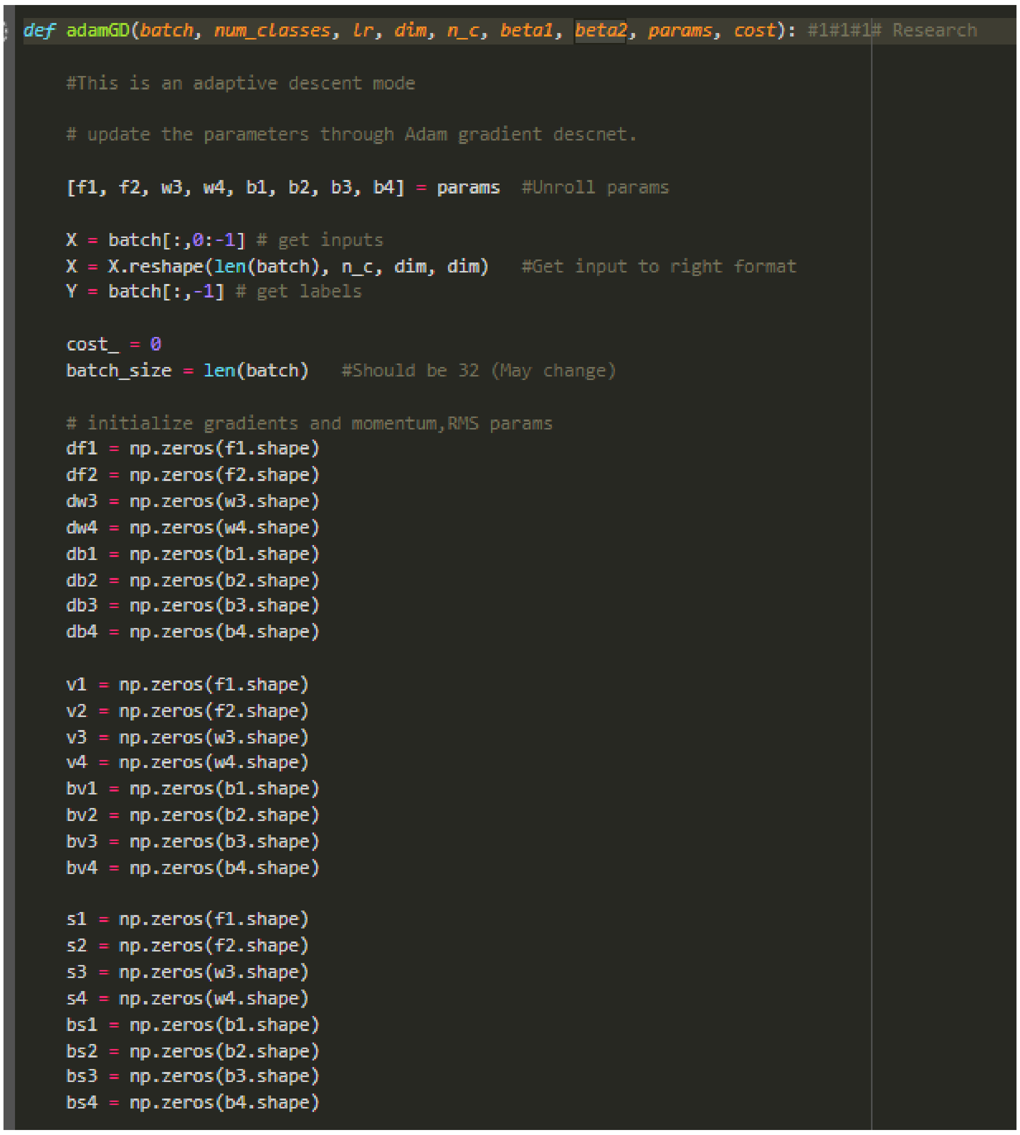


```python
def conv(image, label, params, conv_s, pool_f, pool_s):  #1#1#1# Analyse backward, also may be broken

    [f1, f2, w3, w4, b1, b2, b3, b4] = params

    ##############     Forward       ###############

    conv1 = convolution(image, f1, b1, conv_s) # convolution
    conv1 = relu(conv1) # pass through ReLU non-linearity

    conv2 = convolution(conv1, f2, b2, conv_s) # second convolution
    conv2 = relu(conv2) # pass through ReLU non-linearity
    #print(conv2.shape)

    pooled = maxpool(conv2, pool_f, pool_s) # maxpooling

    (nf2, dim2, _) = pooled.shape
    fc = pooled.reshape((nf2 * dim2 * dim2, 1)) # flatten
    #print(fc.shape)

    z = w3.dot(fc) + b3 # first dense layer
    z = relu(z) # pass through ReLU non-linearity

    out = w4.dot(z) + b4 # second dense layer

    probs = softmax(out) # predict class probabilities with the softmax activation function
    #print(probs)

    #Calculate Loss (Loss function is for batch, cost for multiple)
    loss = categoricalCrossEntropy(probs, label) # categorical cross-entropy loss

    #############      Backward      ###############

    dout = probs - label # (7,)   derivative of loss w.r.t. final dense layer output
    dw4 = dout.dot(z.T) # loss gradient of final dense layer weights
    db4 = np.sum(dout, axis = 1).reshape(b4.shape) # loss gradient of final dense layer biases

    dz = w4.T.dot(dout) # loss gradient of first dense layer outputs
    dz[z<=0] = 0

    dw3 = dz.dot(fc.T)
    db3 = np.sum(dz, axis = 1).reshape(b3.shape)

    dfc = w3.T.dot(dz) # loss gradients of fully-connected layer (pooling layer)
    dpool = dfc.reshape(pooled.shape) # reshape fully connected into dimensions of pooling layer

    dconv2 = maxpoolBackward(dpool, conv2, pool_f, pool_s) # backprop through the max-pooling layer(only neurons with highest activation in window get updated)
    dconv2[conv2<=0] = 0

    dconv1, df2, db2 = convolutionBackward(dconv2, conv1, f2, conv_s) # backpropagate previous gradient through second convolutional layer.
    dconv1[conv1<=0] = 0

    dimage, df1, db1 = convolutionBackward(dconv1, image, f1, conv_s) # backpropagate previous gradient through first convolutional layer.

    grads = [df1, df2, dw3, dw4, db1, db2, db3, db4]

    return grads, loss
```

The conv function is the most important function of the CNN because it is able to use all the previous functions to provide essential inputs by carrying out the actual convolutions

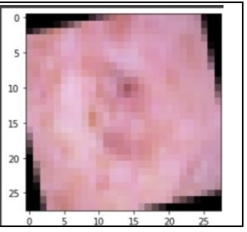
The majority of the function is explained inline through the comments, but essentially this function takes all the information about the image and CNN and will use the different functions in a specific order of convolution, ReLU, convolution, ReLU, max pool, ReLU, fully connected network.

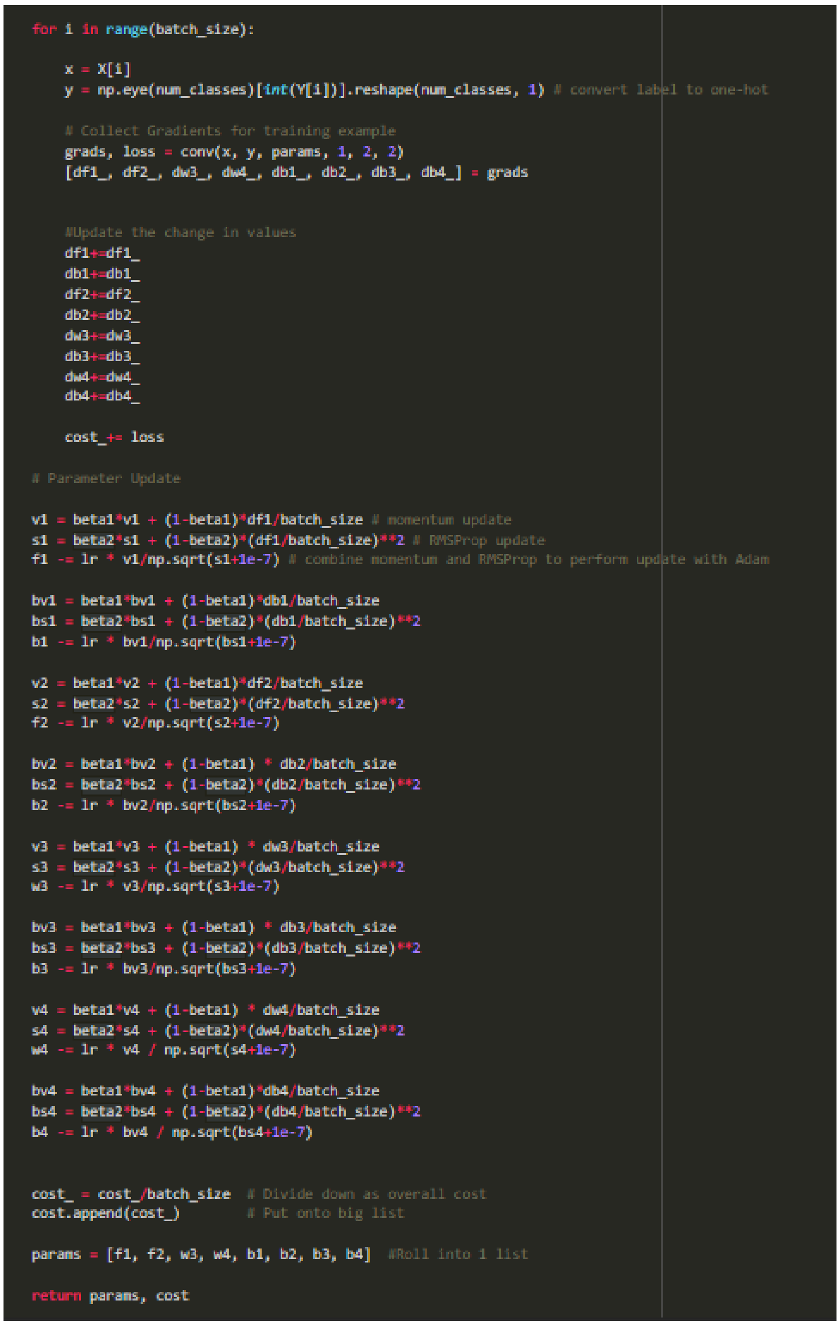
It uses these to output an array of past matrices and the cost of the CNN at this specific point using SoftMax as the loss function in order to calculate the variables



```
def adamGD(batch, num_classes, lr, dim, n_c, beta1, beta2, params, cost): #1#1#1# Research

    #This is an adaptive descent mode

    # update the parameters through Adam gradient descnet.

    [f1, f2, w3, w4, b1, b2, b3, b4] = params  #Unroll params

    X = batch[:,0:-1] # get inputs
    X = X.reshape(len(batch), n_c, dim, dim)   #Get input to right format
    Y = batch[:,-1] # get labels

    cost_ = 0
    batch_size = len(batch)   #Should be 32 (May change)

    # initialize gradients and momentum,RMS params
    df1 = np.zeros(f1.shape)
    df2 = np.zeros(f2.shape)
    dw3 = np.zeros(w3.shape)
    dw4 = np.zeros(w4.shape)
    db1 = np.zeros(b1.shape)
    db2 = np.zeros(b2.shape)
    db3 = np.zeros(b3.shape)
    db4 = np.zeros(b4.shape)

    v1 = np.zeros(f1.shape)
    v2 = np.zeros(f2.shape)
    v3 = np.zeros(w3.shape)
    v4 = np.zeros(w4.shape)
    bv1 = np.zeros(b1.shape)
    bv2 = np.zeros(b2.shape)
    bv3 = np.zeros(b3.shape)
    bv4 = np.zeros(b4.shape)

    s1 = np.zeros(f1.shape)
    s2 = np.zeros(f2.shape)
    s3 = np.zeros(w3.shape)
    s4 = np.zeros(w4.shape)
    bs1 = np.zeros(b1.shape)
    bs2 = np.zeros(b2.shape)
    bs3 = np.zeros(b3.shape)
    bs4 = np.zeros(b4.shape)
```

This is the first half of the ADAMGD optimiser for the CNN this section is mainly initialising empty gradients and momenta for the problem as well as unrolling and finding their relevant size.

A test for this section was to see if an image can be rotated because if this works, it will mean that ask other algorithms within this section are successful	Here is a successful test because the image has been rotated which means all other functions work and has been fed through successfully	



```
for i in range(batch_size):

    x = X[i]
    y = np.eye(num_classes)[int(Y[i])].reshape(num_classes, 1) # convert label to one-hot

    # Collect Gradients for training example
    grads, loss = conv(x, y, params, 1, 2, 2)
    [df1_, df2_, dw3_, dw4_, db1_, db2_, db3_, db4_] = grads


    #Update the change in values
    df1+=df1_
    db1+=db1_
    df2+=df2_
    db2+=db2_
    dw3+=dw3_
    db3+=db3_
    dw4+=dw4_
    db4+=db4_

    cost_+= loss

# Parameter Update

v1 = beta1*v1 + (1-beta1)*df1/batch_size # momentum update
s1 = beta2*s1 + (1-beta2)*(df1/batch_size)**2 # RMSProp update
f1 -= lr * v1/np.sqrt(s1+1e-7) # combine momentum and RMSProp to perform update with Adam

bv1 = beta1*bv1 + (1-beta1)*db1/batch_size
bs1 = beta2*bs1 + (1-beta2)*(db1/batch_size)**2
b1 -= lr * bv1/np.sqrt(bs1+1e-7)

v2 = beta1*v2 + (1-beta1)*df2/batch_size
s2 = beta2*s2 + (1-beta2)*(df2/batch_size)**2
f2 -= lr * v2/np.sqrt(s2+1e-7)

bv2 = beta1*bv2 + (1-beta1) * db2/batch_size
bs2 = beta2*bs2 + (1-beta2)*(db2/batch_size)**2
b2 -= lr * bv2/np.sqrt(bs2+1e-7)

v3 = beta1*v3 + (1-beta1) * dw3/batch_size
s3 = beta2*s3 + (1-beta2)*(dw3/batch_size)**2
w3 -= lr * v3/np.sqrt(s3+1e-7)

bv3 = beta1*bv3 + (1-beta1) * db3/batch_size
bs3 = beta2*bs3 + (1-beta2)*(db3/batch_size)**2
b3 -= lr * bv3/np.sqrt(bs3+1e-7)

v4 = beta1*v4 + (1-beta1) * dw4/batch_size
s4 = beta2*s4 + (1-beta2)*(dw4/batch_size)**2
w4 -= lr * v4 / np.sqrt(s4+1e-7)

bv4 = beta1*bv4 + (1-beta1)*db4/batch_size
bs4 = beta2*bs4 + (1-beta2)*(db4/batch_size)**2
b4 -= lr * bv4 / np.sqrt(bs4+1e-7)


cost_ = cost_/batch_size  # Divide down as overall cost
cost.append(cost_)        # Put onto big list

params = [f1, f2, w3, w4, b1, b2, b3, b4]  #Roll into 1 list

return params, cost
```

This is the 2nd half of the ADAMGD optimiser

This is a very technical piece of code which uses different formulae with small tweaks each time, so was very hard to implement, we also can see the use of the eye function which will generate an identity matrix with the appropriate size and depth.

Stage 2 review

The CNN is now complete. This takes the raw data and metadata and checks for any patterns and then normalised the data as well as scale and one hot encode all the features. This is extremely important as is provides the base CNN needed for the rest of the project.

How is has been tested

Every single function has been initially tested by inputting simple values and tables and manually checking their distributions. I then added more complicated tables to see what the values tested. These have been shown in line.

Additionally, I often found random images from the dataset and outputted their state at frequent intervals which allowed me to check the SoftMax function and generation of filters at random and normal distributions was working correctly. This can be seen above.

Additionally, for the maxpool and convolution functions, I fed in smaller matrices which were of size 2x2 or 3x3 to check that the correct outputs were being formed and this allowed me to create assertion functions to make sure that only images of the appropriate format are added into the CNN.

The images show examples of expected inputs and outputs and as a form of final testing, the functions have been checked against an actual CNN in real world examples and these tests were passed.

Success Criteria points

Fine-tuned ML algorithms	Use a test and cross validation set to test how well the algorithm works and make sure the test accuracy is high enough, and that the data is not overfitted.

Changes

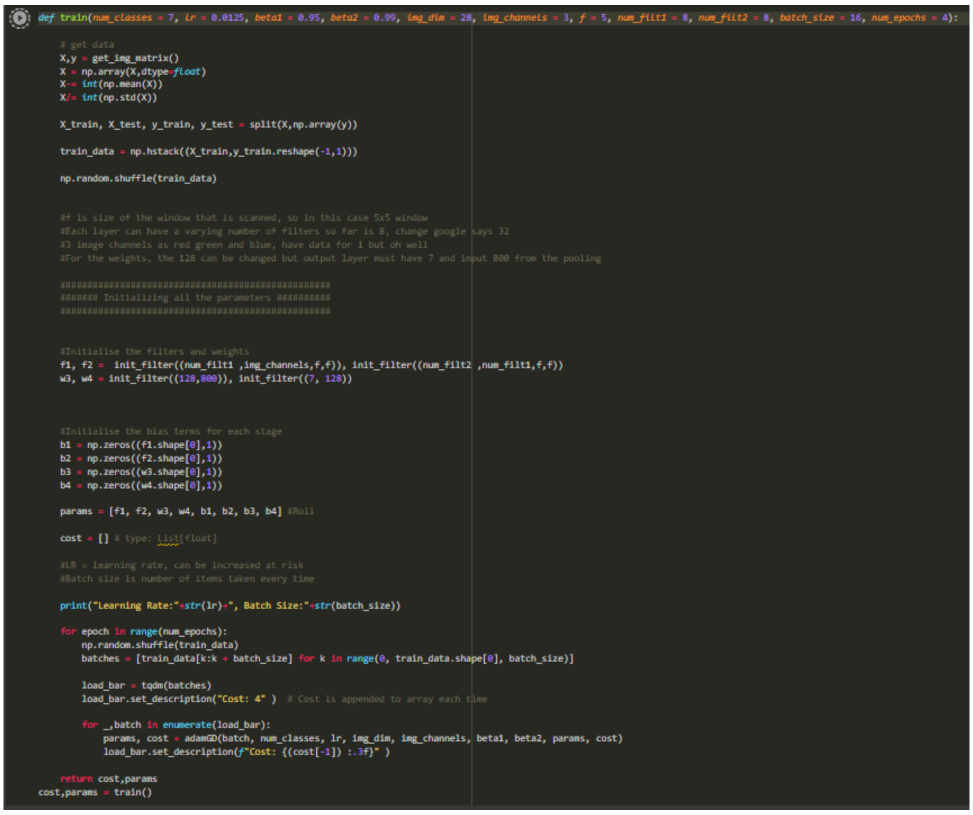
During this process I had to change the architecture of my CNN many times which was essential due to the constant changing algorithms and ended up choosing a 4-layer alternating pattern after finally fixing these algorithms after rigorous and repeated testing.

Summary of prototype

At this stage, the hardest part of coding has been completed, now it is all about doing the visible and most important bulk of the coding with all the background code and functions having been coded and fully tested for functionality.

Stage 3 Train CNN

This stage consists of 2 parts of the development, the building of the CNN train functions and then physically training the function using the code and dataset and therefore training the hyperparameters.



```python
def train(num_classes = 7, lr = 0.0125, beta1 = 0.95, beta2 = 0.99, img_dim = 28, img_channels = 3, f = 5, num_filt1 = 8, num_filt2 = 8, batch_size = 16, num_epochs = 4):

    # get data
    X,y = get_img_matrix()
    X = np.array(X,dtype=float)
    X-= int(np.mean(X))
    X/= int(np.std(X))

    X_train, X_test, y_train, y_test = split(X,np.array(y))

    train_data = np.hstack((X_train,y_train.reshape(-1,1)))

    np.random.shuffle(train_data)


    #f is size of the window that is scanned, so in this case 5x5 window
    #Each layer can have a varying number of filters so far is 8, change google says 32
    #3 image channels as red green and blue, have data for 1 but oh well
    #For the weights, the 128 can be changed but output layer must have 7 and input 800 from the pooling

    ##################################################
    ####### Initializing all the parameters ##########
    ##################################################


    #Initialise the filters and weights
    f1, f2 =  init_filter((num_filt1 ,img_channels,f,f)), init_filter((num_filt2 ,num_filt1,f,f))
    w3, w4 = init_filter((128,800)), init_filter((7, 128))



    #Initialise the bias terms for each stage
    b1 = np.zeros((f1.shape[0],1))
    b2 = np.zeros((f2.shape[0],1))
    b3 = np.zeros((w3.shape[0],1))
    b4 = np.zeros((w4.shape[0],1))

    params = [f1, f2, w3, w4, b1, b2, b3, b4] #Roll

    cost = [] # type: List[float]

    #LR = learning rate, can be increased at risk
    #Batch size is number of items taken every time

    print("Learning Rate:"+str(lr)+", Batch Size:"+str(batch_size))

    for epoch in range(num_epochs):
        np.random.shuffle(train_data)
        batches = [train_data[k:k + batch_size] for k in range(0, train_data.shape[0], batch_size)]

        load_bar = tqdm(batches)
        load_bar.set_description("Cost: 4" )  # Cost is appended to array each time

        for _,batch in enumerate(load_bar):
            params, cost = adamGD(batch, num_classes, lr, img_dim, img_channels, beta1, beta2, params, cost)
            load_bar.set_description(f"Cost: {(cost[-1]) :.3f}" )

    return cost,params
cost,params = train()
```

Here is the train algorithm which is obviously the most important algorithm of the training function. This will take specific hyperparameters, which will be defined later and will use them to use gradient descent in order to find the most efficient parameters for the dataset.

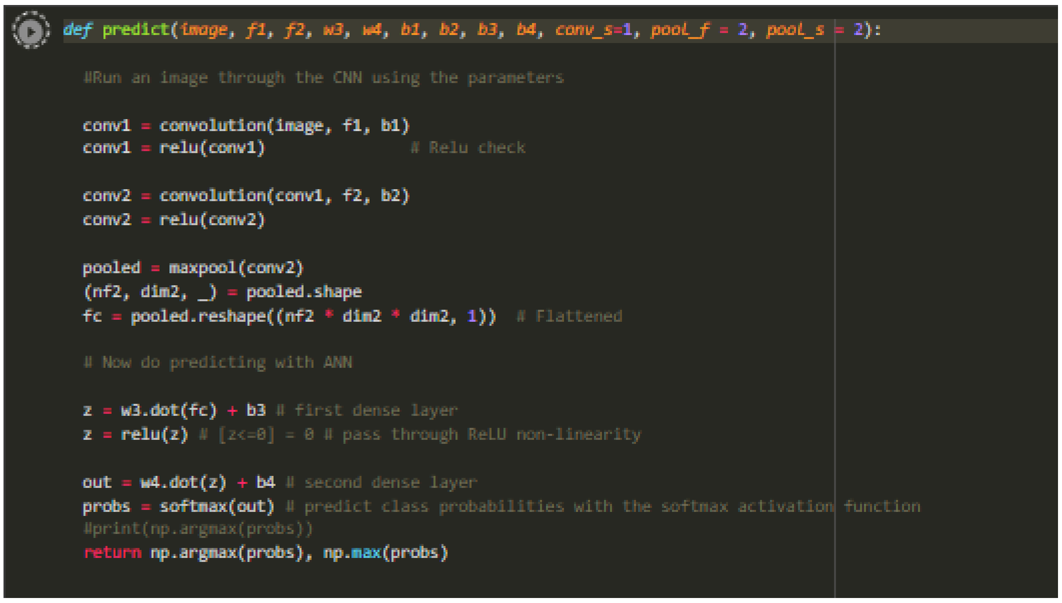
To do this the image matrix is initially taken and it will be split into its actual data and the label which should be predicted, which is represented by X and y.

The dataset will then be split into its training and its test set. At the moment this process should be random and there should be no pattern between the lesion samples which are chosen for the final dataset.

Next the dataset is shuffled because the main dataset is organised, and these will need to be reorganised to make sure the database does not have positional bias within the dataset.

Next, blank filters and bias terms are formed, this time the majority are not 0 matrices but random numbers within a standard normal distribution.

Then at each epoch we use the ADAMGD function in order to run through the dataset and train it for each batch, which is a recursive process and runs like this:

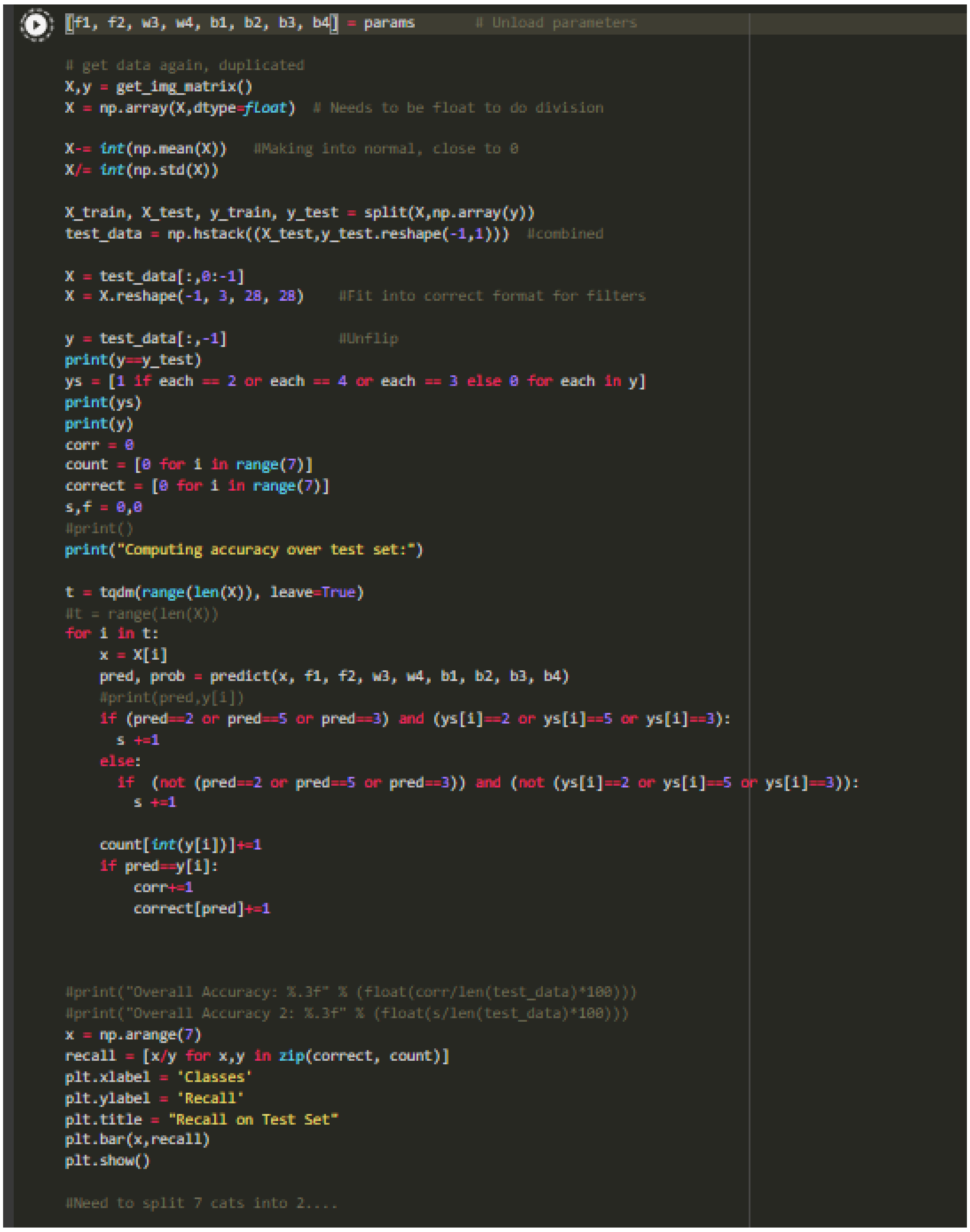


```
def predict(image, f1, f2, w3, w4, b1, b2, b3, b4, conv_s=1, pool_f = 2, pool_s = 2):
    #Run an image through the CNN using the parameters

    conv1 = convolution(image, f1, b1)
    conv1 = relu(conv1)                # Relu check

    conv2 = convolution(conv1, f2, b2)
    conv2 = relu(conv2)

    pooled = maxpool(conv2)
    (nf2, dim2, _) = pooled.shape
    fc = pooled.reshape((nf2 * dim2 * dim2, 1))  # Flattened

    # Now do predicting with ANN

    z = w3.dot(fc) + b3 # first dense layer
    z = relu(z) # [z<=0] = 0 # pass through ReLU non-linearity

    out = w4.dot(z) + b4 # second dense layer
    probs = softmax(out) # predict class probabilities with the softmax activation function
    #print(np.argmax(probs))
    return np.argmax(probs), np.max(probs)
```

This is the next function utilised by the algorithm, which is the predict function which is able to use the parameters which are already provided to the function in order to try and use pattern recognition in order to choose an output image.

To do this, the function initially runs through the CNN layers then the fully connected layer, it then finally uses the SoftMax function to calculate the loss at this position and will the predict appropriately depending on what group is selected.

The function uses the np. argmax to fund the appropriate maximum value and will use the max function to find the actual probability that the function is predicting this value with.



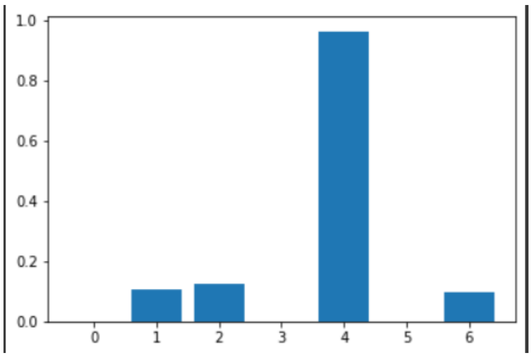
```python
[f1, f2, w3, w4, b1, b2, b3, b4] = params       # Unload parameters

# get data again, duplicated
X,y = get_img_matrix()
X = np.array(X,dtype=float)  # Needs to be float to do division

X-= int(np.mean(X))   #Making into normal, close to 0
X/= int(np.std(X))

X_train, X_test, y_train, y_test = split(X,np.array(y))
test_data = np.hstack((X_test,y_test.reshape(-1,1)))  #combined

X = test_data[:,0:-1]
X = X.reshape(-1, 3, 28, 28)    #Fit into correct format for filters

y = test_data[:,-1]             #Unflip
print(y==y_test)
ys = [1 if each == 2 or each == 4 or each == 3 else 0 for each in y]
print(ys)
print(y)
corr = 0
count = [0 for i in range(7)]
correct = [0 for i in range(7)]
s,f = 0,0
#print()
print("Computing accuracy over test set:")

t = tqdm(range(len(X)), leave=True)
#t = range(len(X))
for i in t:
    x = X[i]
    pred, prob = predict(x, f1, f2, w3, w4, b1, b2, b3, b4)
    #print(pred,y[i])
    if (pred==2 or pred==5 or pred==3) and (ys[i]==2 or ys[i]==5 or ys[i]==3):
      s +=1
    else:
      if  (not (pred==2 or pred==5 or pred==3)) and (not (ys[i]==2 or ys[i]==5 or ys[i]==3)):
        s +=1

    count[int(y[i])]+=1
    if pred==y[i]:
        corr+=1
        correct[pred]+=1



#print("Overall Accuracy: %.3f" % (float(corr/len(test_data)*100)))
#print("Overall Accuracy 2: %.3f" % (float(s/len(test_data)*100)))
x = np.arange(7)
recall = [x/y for x,y in zip(correct, count)]
plt.xlabel = 'Classes'
plt.ylabel = 'Recall'
plt.title = "Recall on Test Set"
plt.bar(x,recall)
plt.show()

#Need to split 7 cats into 2....
```

Here is the main code upon which the code is tested upon. This occurs after the training phase and parameters have been loaded and are used to try and predict the lesion type on an unknown dataset. For this

While creating the training algorithm it is extremely important to train the hyperparameters, I have done this over the test set and then adjusted the hyperparameters accordingly to improve this.

Testing:

Round 1

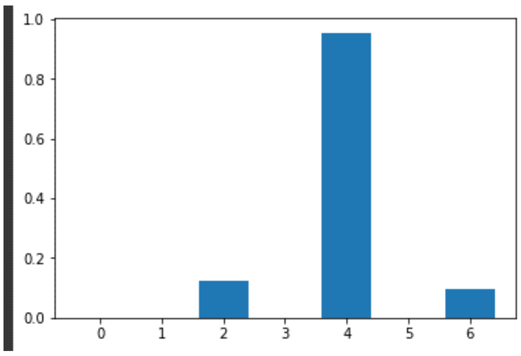


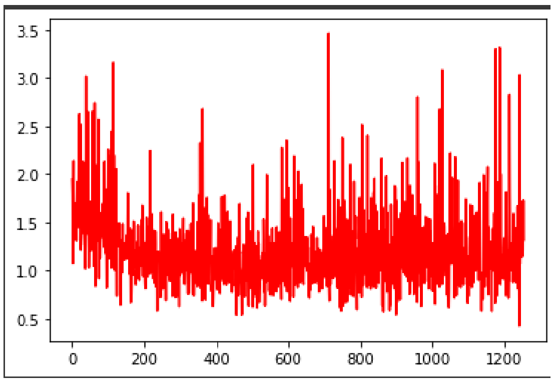
Hyperparameters

((num_classes = 7, *lr* = 0.01, *beta1* = 0.95, *beta2* = 0.99, *img_dim* = 28, *img_channels* = 3, *f* = 5, *num_filt1* = 8, *num_filt2* = 8, *batch_size* = 32, *num_epochs* = 2)

This is the initial state of the hyperparameters, and we can evidently see that the class 4 is being predicted quite highly which is worrying and needs to be improved upon. This may be due to underfitting the dataset therefore it will be important to increase the number of epochs

Round 2

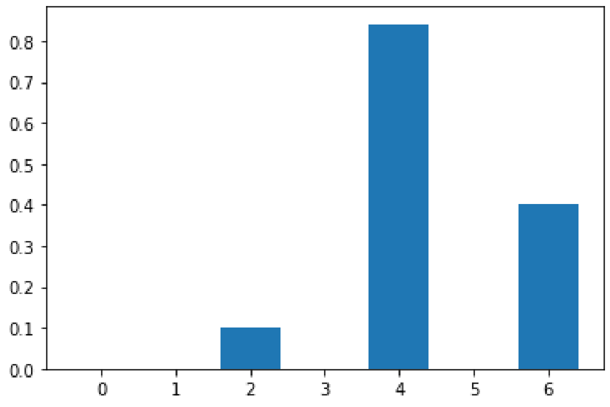


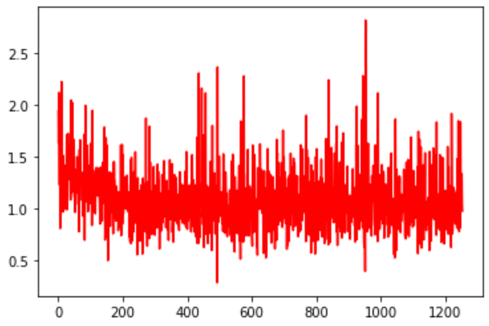


((num_classes = 7, lr = 0.02, beta1 = 0.95, beta2 = 0.99, img_dim = 28, img_channels = 3, f = 5, num_filt1 = 8, num_filt2 = 8, batch_size = 32, num_epochs = 5)

For the second round, I increased the number of epochs and can see a definite improvement that 4 is being predicted less, however upon analysing the graph on the right we can see that the cost function is spiralling which is very concerning and means that the learning rate may be too high as it cannot stay in a local maximum but spiralling in and out of maxima and minima after each batch.

Round 3



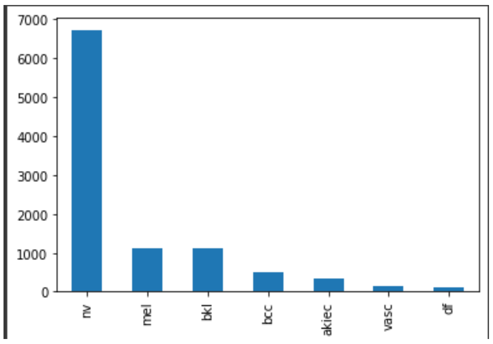


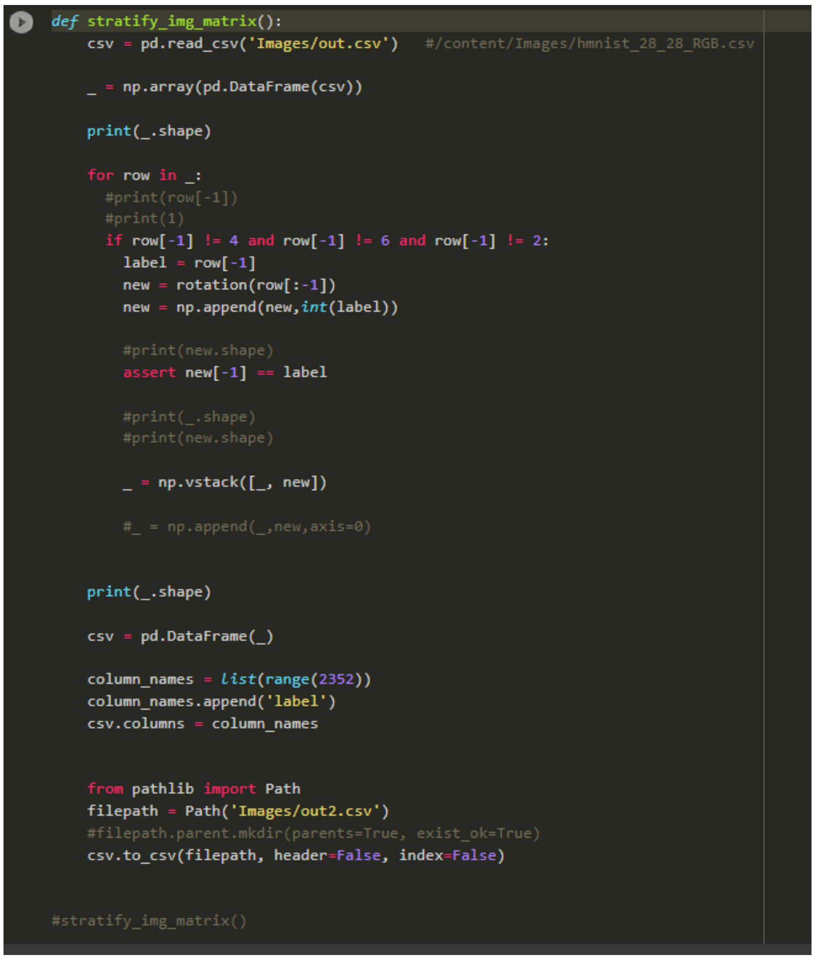
(num_classes = 7, *lr* = 0.015, *beta1* = 0.95, *beta2* = 0.99, *img_dim* = 28, *img_channels* = 3, *f* = 5, *num_filt1* = 8, *num_filt2* = 8, *batch_size* = 32, *num_epochs* = 5)

Here is the third round of hyperparameters where they seem to have been optimised however, we can still see that there is a surplus of predictions of the category 4 therefore we will need to do iterative development in order to fix this error.

To improve this, initially the database needed to be looked at because there was a large sway in data which created an extremely high bias and the CNN could not function. As can be seen in this graph, there are many more nv samples than all others! Therefore this test has failed after 3 iterations.

Here is a view of the initial state of the database which is causing this clear bias as nv maps to class 4 therefore this is a big problem and will need to be fixed by upscaling the database.

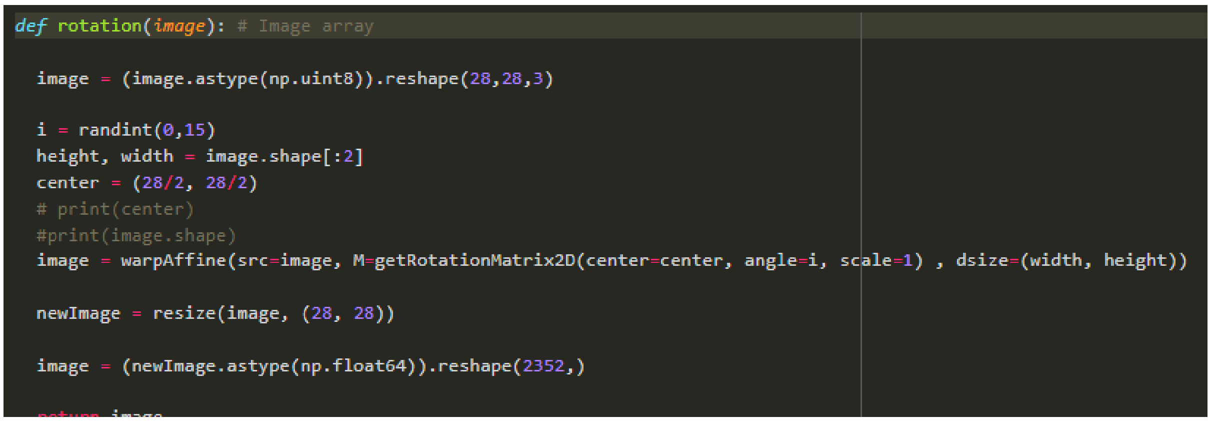




```python
def stratify_img_matrix():
    csv = pd.read_csv('Images/out.csv')   #/content/Images/hmnist_28_28_RGB.csv

    _ = np.array(pd.DataFrame(csv))

    print(_.shape)

    for row in _:
      #print(row[-1])
      #print(1)
      if row[-1] != 4 and row[-1] != 6 and row[-1] != 2:
        label = row[-1]
        new = rotation(row[:-1])
        new = np.append(new,int(label))

        #print(new.shape)
        assert new[-1] == label

        #print(_.shape)
        #print(new.shape)

        _ = np.vstack([_, new])

        #_ = np.append(_,new,axis=0)


    print(_.shape)

    csv = pd.DataFrame(_)

    column_names = list(range(2352))
    column_names.append('label')
    csv.columns = column_names


    from pathlib import Path
    filepath = Path('Images/out2.csv')
    #filepath.parent.mkdir(parents=True, exist_ok=True)
    csv.to_csv(filepath, header=False, index=False)


#stratify_img_matrix()
```

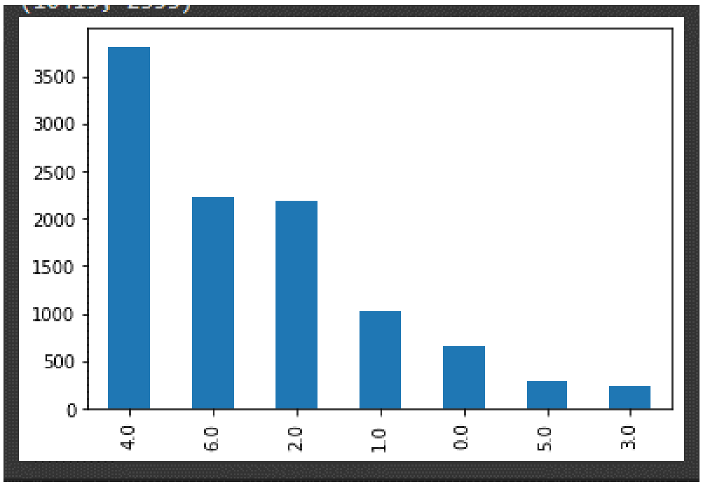
I used this code to choose appropriate values of data points and created new data points within these divisions by rotating the images and flipping them to certain degrees.
The code starts by taking the database and outputting the shape in order for analysis on dimensions. This is merely here for the testing stage.

Next, it chooses certain lesions and will take the individual label and use the new rotation function to change the image slightly and finally stacking this on top of all the other images. It then adds the column labels back and saves it into a new csv file. To test this, I looked through the data points and compared random points with the original data and used frequent assertion statements.

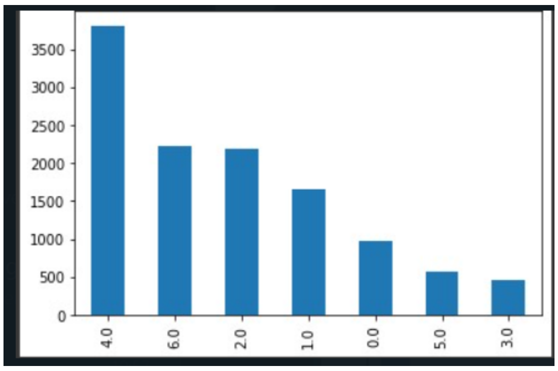


```python
def rotation(image): # Image array

  image = (image.astype(np.uint8)).reshape(28,28,3)

  i = randint(0,15)
  height, width = image.shape[:2]
  center = (28/2, 28/2)
  # print(center)
  #print(image.shape)
  image = warpAffine(src=image, M=getRotationMatrix2D(center=center, angle=i, scale=1) , dsize=(width, height))

  newImage = resize(image, (28, 28))

  image = (newImage.astype(np.float64)).reshape(2352,)

  return image
```

For the rotation function, initially an image array is inputted, this is then reshaped into a 28x28x3 array and is rotated using matrix maths and then outputted subsequently.

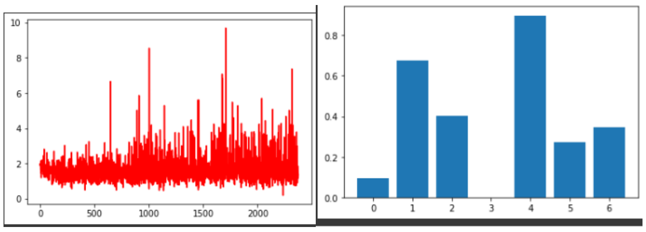


Iteration 1



Iteration 2

Clear improvement seen at each iteration with the algorithm being run twice, which has now allowed me to pass this test and move onto stage 4 of development



Here are the final graphs of the loaded CNN which have cost consistency and more equal predictions therefore this test has now been passed and we can move onto next stage of production.

Stage 3 review

This stage of training the CNN was by far the most intricate process of this CNN which took time to run and ended up failing horrifically, so needed more development time in order to create more images for each class in order to improve these stages.

How is has been tested

To test this, I calculated the cost with the trained hyperparameters and used a series of graphs which can be viewed above as well as passing through data with known results to test if the CNN can predict as accurately as other CNNS are able to do. Most of the testing was adjusting the parameters and trying to find a way to minimise the cost function, therefore, to pass the big test here, the cost graph had to have found a consistent minimum which I believe has been found.

Additionally, the data algorithms were tested with live images which were then viewed using image modules and then checked if rotated correctly or if all 3 colours RGB are showing appropriately.

Here is an example of one of the test results that have been carried out.

Success Criteria points

Fine-tuned ML algorithms	Use a test and cross validation set to test how well the algorithm works and make sure the test accuracy is high enough, and that the data is not overfitted.
Integration of ml algorithms together	Use another test set to check if the overall diagnosis given by both machine learning algorithms is accurate.

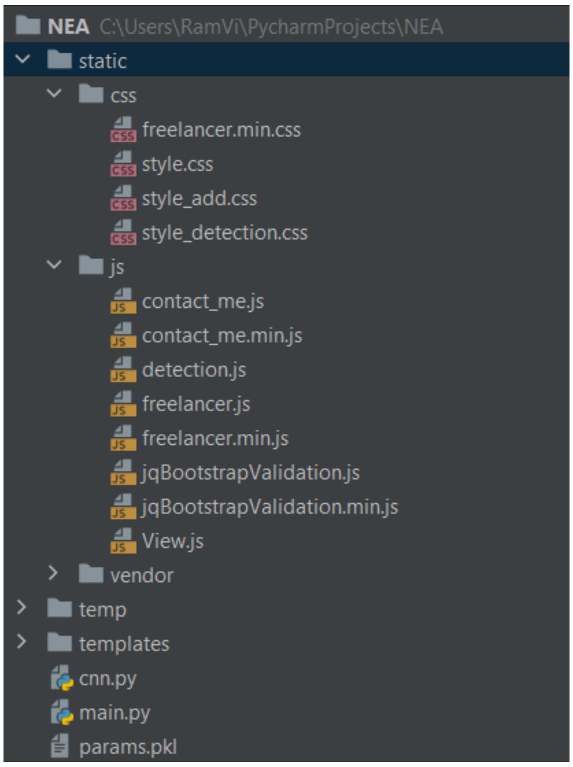
Changes

During this process I had to change the actual structure of the database I was using making sure to include more images of other lesions because the data seemed to be allowing the ML algorithm to massively overfit upon the dataset therefore, I needed to write some new algorithms and map the dataset with new data to allow the algorithm to run more carefully.

Summary of prototype

At this stage, the hardest part of coding has been completed, now it is all about building the website since almost all the backend development has been completed so now it is moving onto the front end and actual features of the web page!

Stage 4 build website



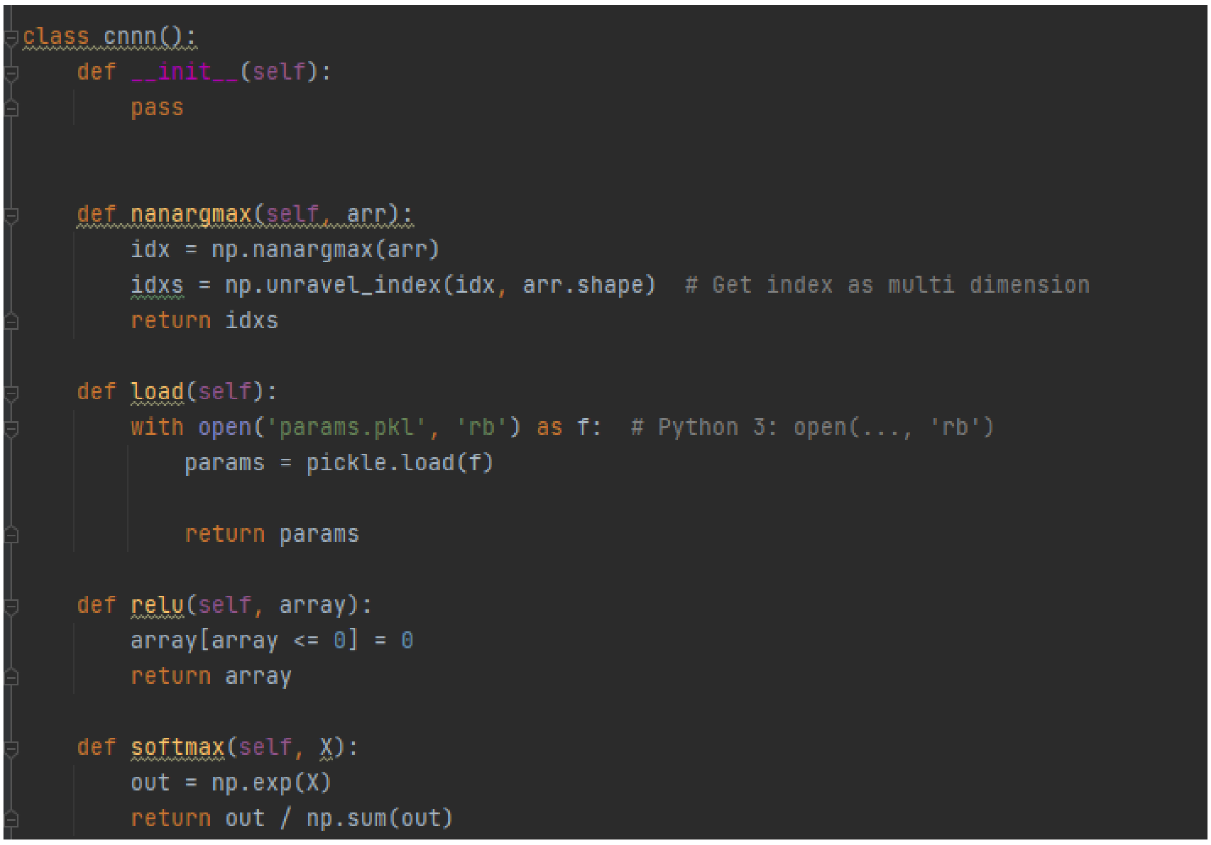
```
NEA  C:\Users\RamVi\PycharmProjects\NEA
  static
    css
      freelancer.min.css
      style.css
      style_add.css
      style_detection.css
    js
      contact_me.js
      contact_me.min.js
      detection.js
      freelancer.js
      freelancer.min.js
      jqBootstrapValidation.js
      jqBootstrapValidation.min.js
      View.js
    vendor
  temp
  templates
  cnn.py
  main.py
  params.pkl
```

Here is the structure of my website, which uses CSS, JS and 2 python files

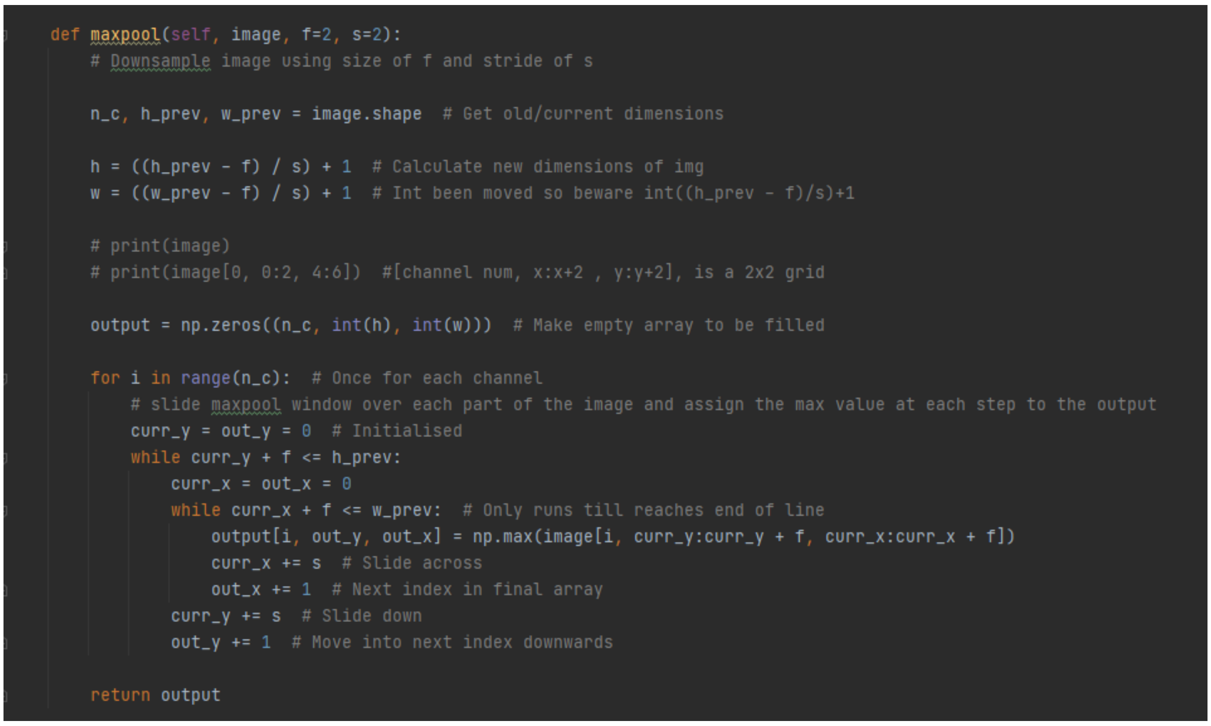
In subsequent development, the View.js file eradicated due to being redundant

Cnn.py

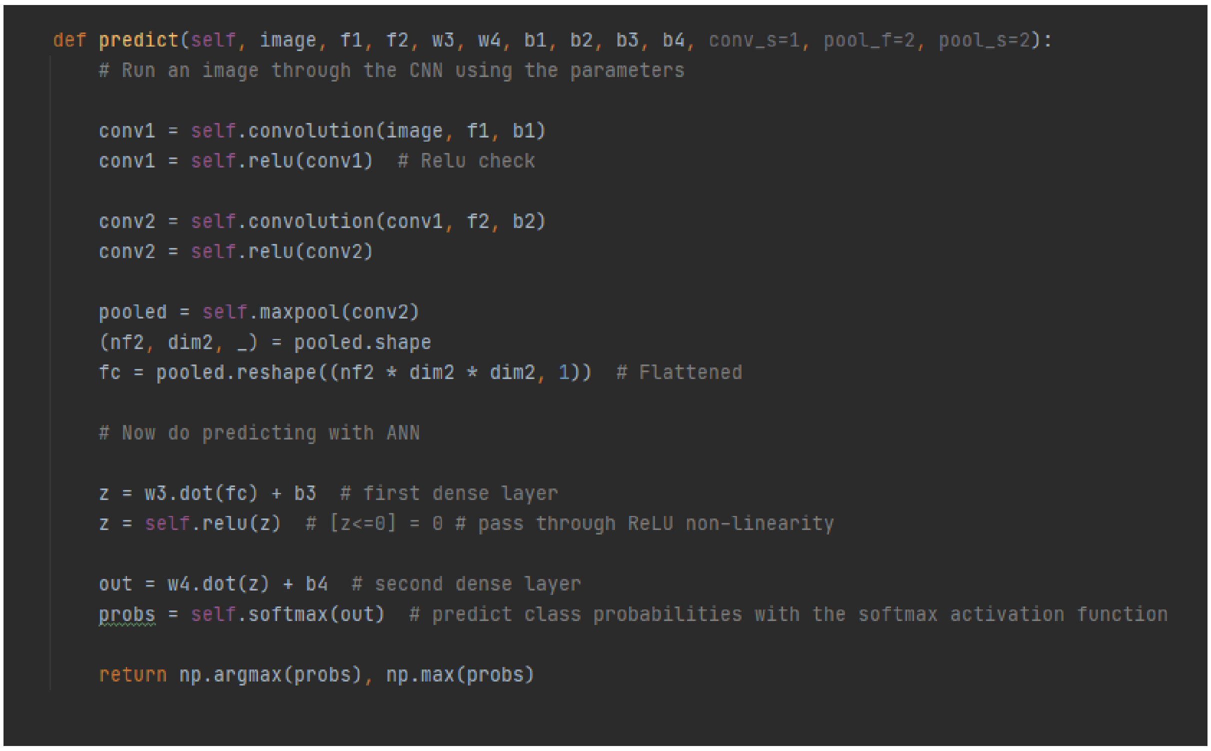
Most of this code is variations of code already discussed within the CNN build section therefore this file does not have as many annotations upon it.



```python
class cnnn():
    def __init__(self):
        pass


    def nanargmax(self, arr):
        idx = np.nanargmax(arr)
        idxs = np.unravel_index(idx, arr.shape)  # Get index as multi dimension
        return idxs

    def load(self):
        with open('params.pkl', 'rb') as f:  # Python 3: open(..., 'rb')
            params = pickle.load(f)

            return params

    def relu(self, array):
        array[array <= 0] = 0
        return array

    def softmax(self, X):
        out = np.exp(X)
        return out / np.sum(out)
```

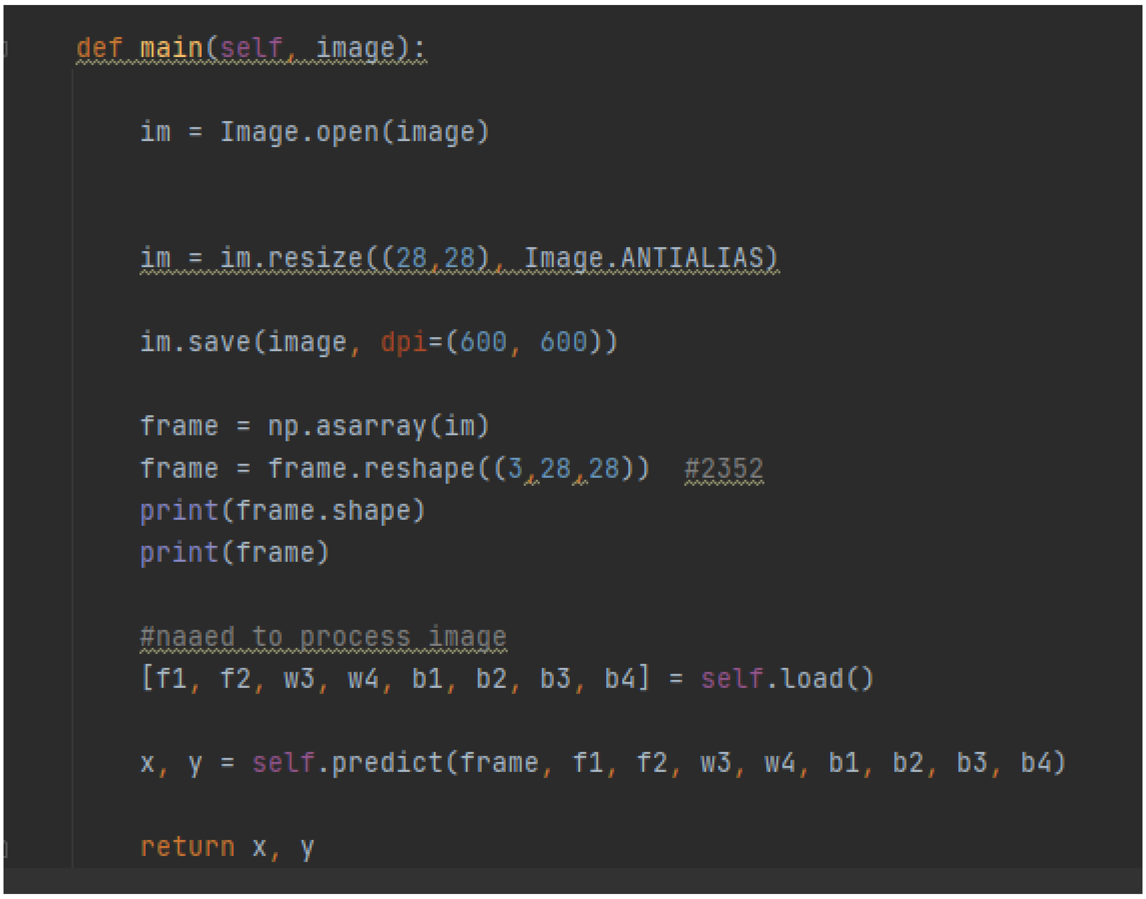


```python
    def maxpool(self, image, f=2, s=2):
        # Downsample image using size of f and stride of s

        n_c, h_prev, w_prev = image.shape  # Get old/current dimensions

        h = ((h_prev - f) / s) + 1  # Calculate new dimensions of img
        w = ((w_prev - f) / s) + 1  # Int been moved so beware int((h_prev - f)/s)+1

        # print(image)
        # print(image[0, 0:2, 4:6])  #[channel num, x:x+2 , y:y+2], is a 2x2 grid

        output = np.zeros((n_c, int(h), int(w)))  # Make empty array to be filled

        for i in range(n_c):  # Once for each channel
            # slide maxpool window over each part of the image and assign the max value at each step to the output
            curr_y = out_y = 0  # Initialised
            while curr_y + f <= h_prev:
                curr_x = out_x = 0
                while curr_x + f <= w_prev:  # Only runs till reaches end of line
                    output[i, out_y, out_x] = np.max(image[i, curr_y:curr_y + f, curr_x:curr_x + f])
                    curr_x += s  # Slide across
                    out_x += 1  # Next index in final array
                curr_y += s  # Slide down
                out_y += 1  # Move into next index downwards

        return output
```



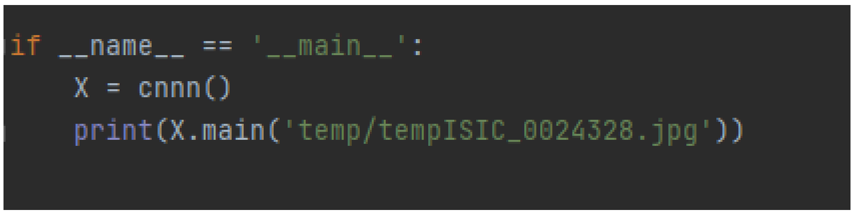
```
def predict(self, image, f1, f2, w3, w4, b1, b2, b3, b4, conv_s=1, pool_f=2, pool_s=2):
    # Run an image through the CNN using the parameters

    conv1 = self.convolution(image, f1, b1)
    conv1 = self.relu(conv1)  # Relu check

    conv2 = self.convolution(conv1, f2, b2)
    conv2 = self.relu(conv2)

    pooled = self.maxpool(conv2)
    (nf2, dim2, _) = pooled.shape
    fc = pooled.reshape((nf2 * dim2 * dim2, 1))  # Flattened

    # Now do predicting with ANN

    z = w3.dot(fc) + b3  # first dense layer
    z = self.relu(z)  # [z<=0] = 0 # pass through ReLU non-linearity

    out = w4.dot(z) + b4  # second dense layer
    probs = self.softmax(out)  # predict class probabilities with the softmax activation function

    return np.argmax(probs), np.max(probs)
```

All code here has been previously discussed within the build CNN phase



```
def main(self, image):

    im = Image.open(image)


    im = im.resize((28,28), Image.ANTIALIAS)

    im.save(image, dpi=(600, 600))

    frame = np.asarray(im)
    frame = frame.reshape((3,28,28))  #2352
    print(frame.shape)
    print(frame)

    #naaed to process image
    [f1, f2, w3, w4, b1, b2, b3, b4] = self.load()

    x, y = self.predict(frame, f1, f2, w3, w4, b1, b2, b3, b4)

    return x, y
```

Here is the main function which only runs when the file is run within the main function if it is run. This function is used as a testing algorithm for the CNN class.

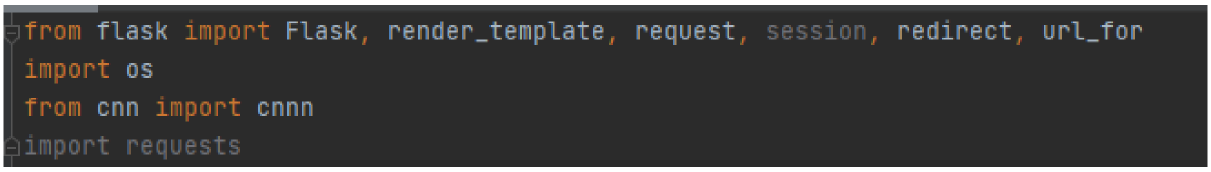


```python
if __name__ == '__main__':
    X = cnnn()
    print(X.main('temp/tempISIC_0024328.jpg'))
```

Here is the instantiation of the CNN class which has the attribute the image of ISIC_002436.jpg.

This is an important testing phase in the end development stage for the CNN production

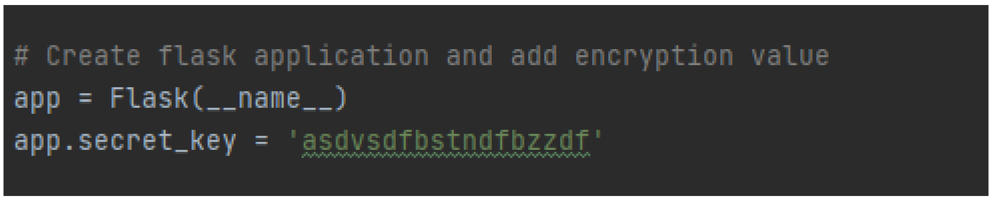
Main.py



```python
from flask import Flask, render_template, request, session, redirect, url_for
import os
from cnn import cnnn
import requests
```

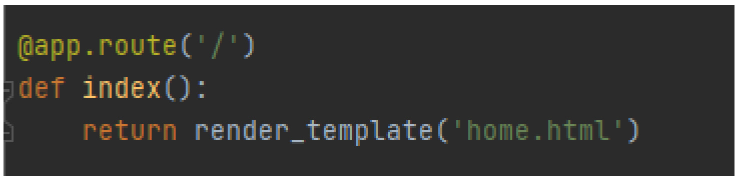
Here are the input modules into the main.py function which holds the backend of the majority of the code which are needed for the function.

The flask module is used for rendering the website within a python console and the appropriate website functions using Jinja2 notation.



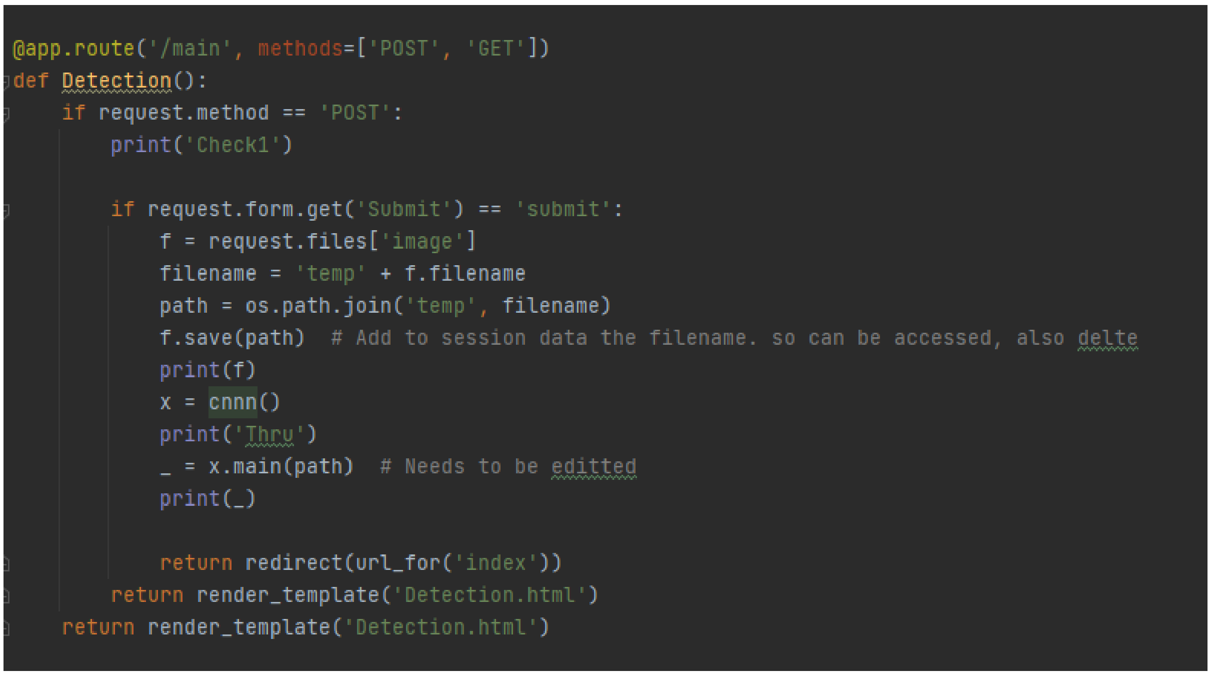
```python
# Create flask application and add encryption value
app = Flask(__name__)
app.secret_key = 'asdvsdfbstndfbzzdf'
```

Here is the code which runs the flask application which



```python
@app.route('/')
def index():
    return render_template('home.html')
```

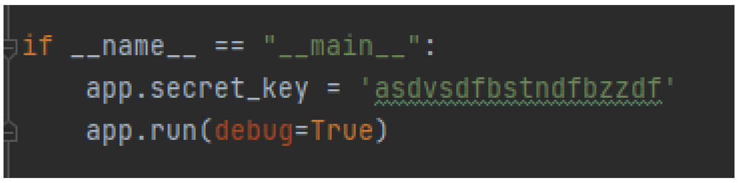
This function is the backend for the main link of the webpage, to test this the looks of the website will be mainly tested by look test which will be analysed within html section


```
@app.route('/main', methods=['POST', 'GET'])
def Detection():
    if request.method == 'POST':
        print('Check1')

        if request.form.get('Submit') == 'submit':
            f = request.files['image']
            filename = 'temp' + f.filename
            path = os.path.join('temp', filename)
            f.save(path)  # Add to session data the filename. so can be accessed, also delte
            print(f)
            x = cnnn()
            print('Thru')
            _ = x.main(path)  # Needs to be editted
            print(_)

            return redirect(url_for('index'))
        return render_template('Detection.html')
    return render_template('Detection.html')
```

Here is the detection function which consists of a decorator being the app. route which tells us the index that the webpage is located and that this page accepts both get and post requestions

If there is a post request and someone is trying to send data to the webpage is will temporarily save the image onto the session data in order for the CNN to be run with the data and then will call the CNN function from the class, which will output the prediction which is then fed to the web page

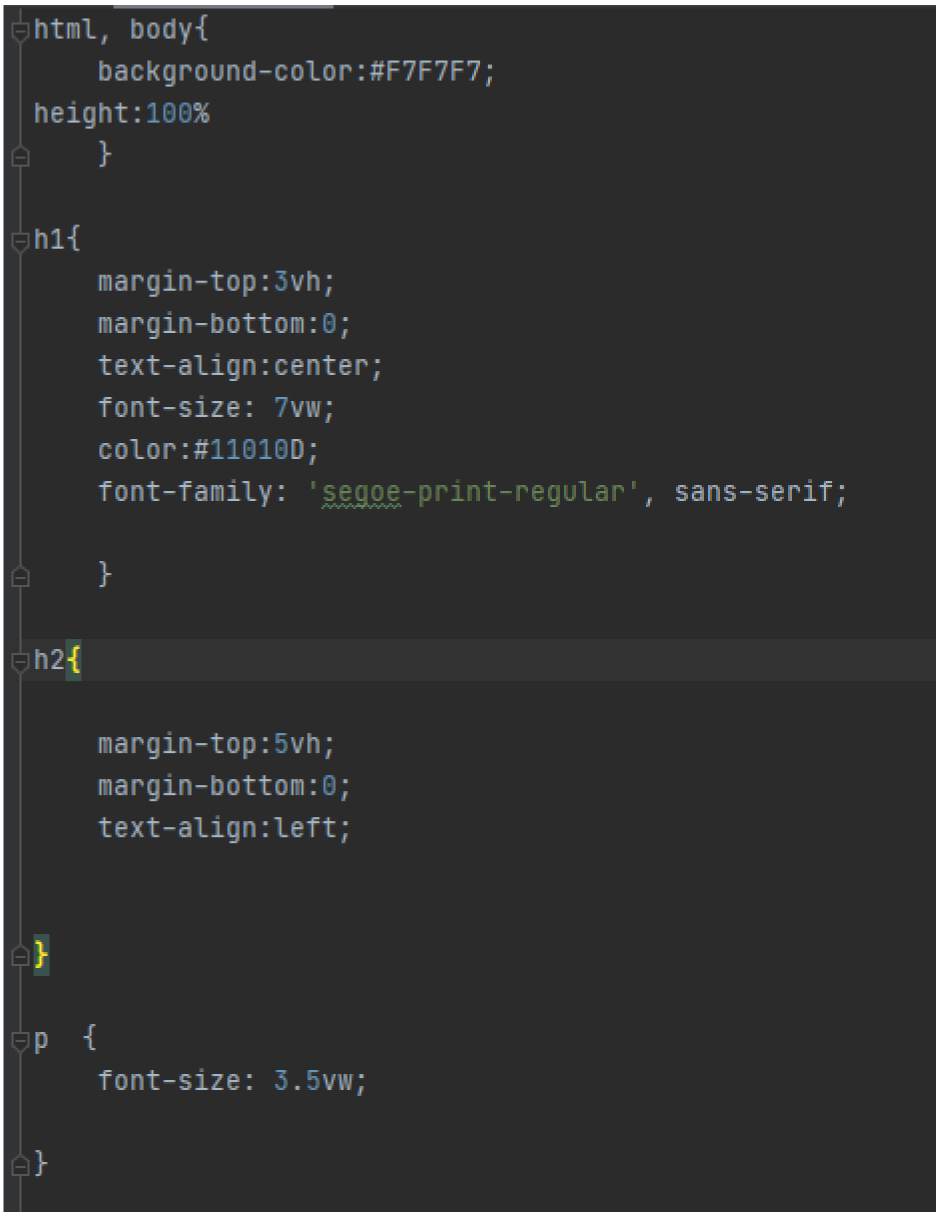
This test has currently failed because data has not been sent properly to the webpage and this needs better validation:


```
if __name__ == "__main__":
    app.secret_key = 'asdvsdfbstndfbzzdf'
    app.run(debug=True)
```

This code is used to tell the local machine which website to render when localhost is run on port 5000

CSS

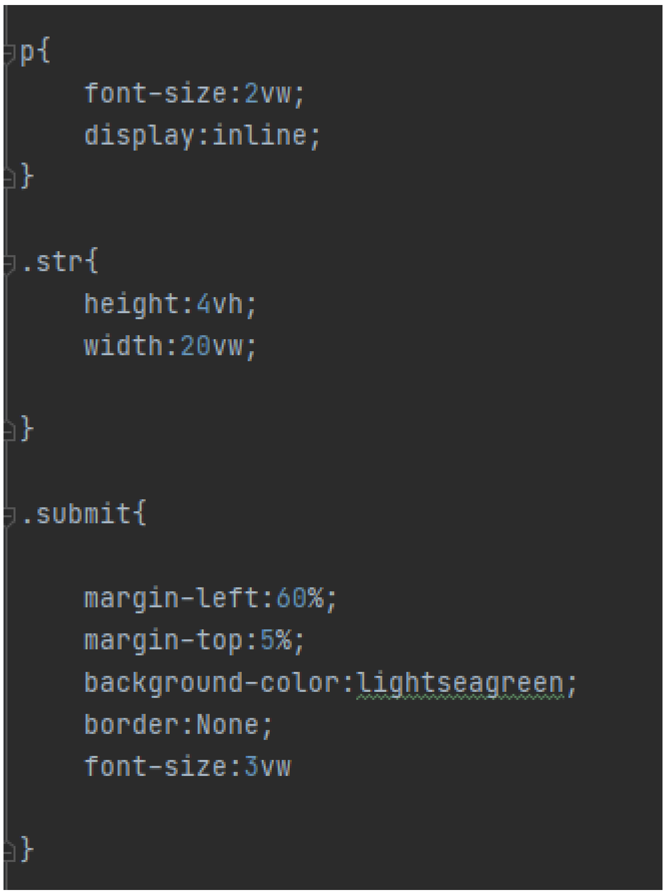
Style.css

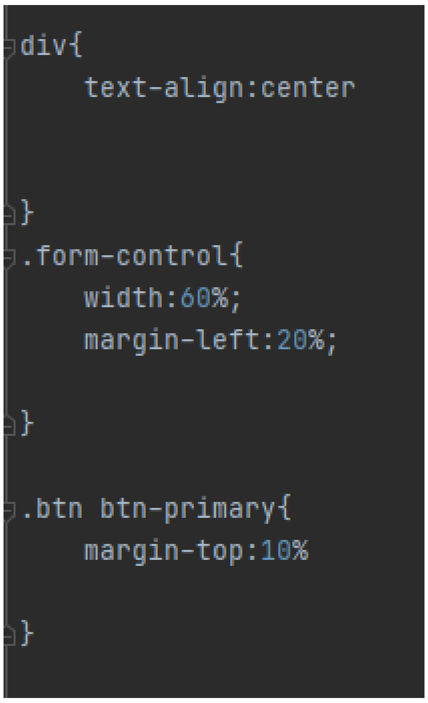


```
html, body{
    background-color:#F7F7F7;
height:100%
    }

h1{
    margin-top:3vh;
    margin-bottom:0;
    text-align:center;
    font-size: 7vw;
    color:#11010D;
    font-family: 'segoe-print-regular', sans-serif;

    }

h2{

    margin-top:5vh;
    margin-bottom:0;
    text-align:left;



}

p  {
    font-size: 3.5vw;

}
```

This code is relatively simple and has been done like this to make it most useable for a website on a larger computer and the looks will be shown in the HTML section. The font has been chosen as picked out within the analysis section.

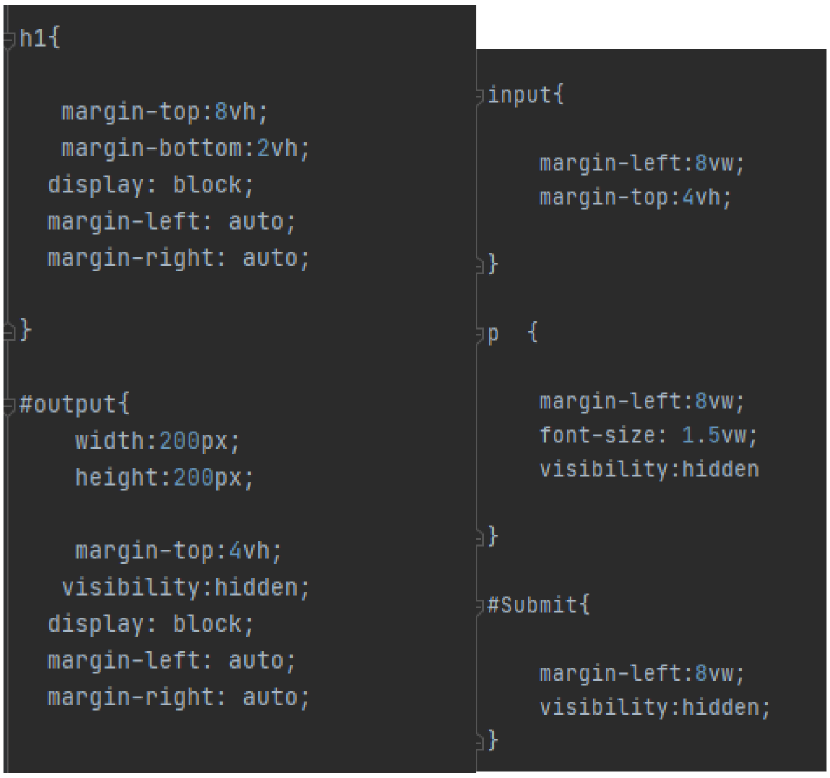
Style_add.css


```
p{
    font-size:2vw;
    display:inline;
}

.str{
    height:4vh;
    width:20vw;

}

.submit{

    margin-left:60%;
    margin-top:5%;
    background-color:lightseagreen;
    border:None;
    font-size:3vw

}
```


```
div{
    text-align:center


}
.form-control{
    width:60%;
    margin-left:20%;

}

.btn btn-primary{
    margin-top:10%

}
```

These 2 snippets are very specific to the add function webpage and have specific values for this webpage which overwrite all base attributes given by the class values from the previous CSS page.

Style_detection.css



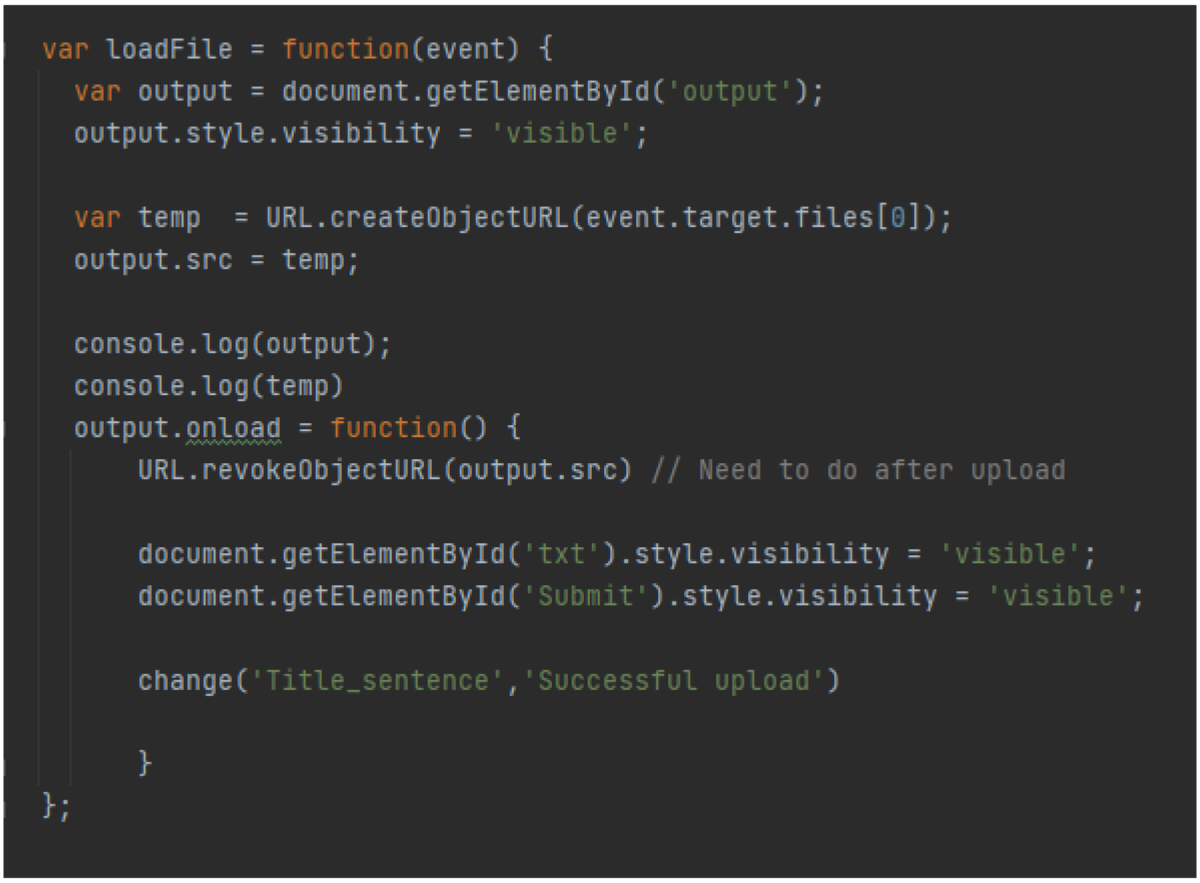
```
h1{

    margin-top:8vh;
    margin-bottom:2vh;
   display: block;
   margin-left: auto;
   margin-right: auto;

}

#output{
     width:200px;
     height:200px;

     margin-top:4vh;
    visibility:hidden;
   display: block;
   margin-left: auto;
   margin-right: auto;
```

```
input{

    margin-left:8vw;
    margin-top:4vh;


}

p  {

    margin-left:8vw;
    font-size: 1.5vw;
    visibility:hidden

}

#Submit{

    margin-left:8vw;
    visibility:hidden;
}
```

These are the final CSS snippets which have been used on the most important page of the website being the detection page and are able to make sure that all motioned objects are able to interact with each other while still looking appropriate and adapting purposefully.

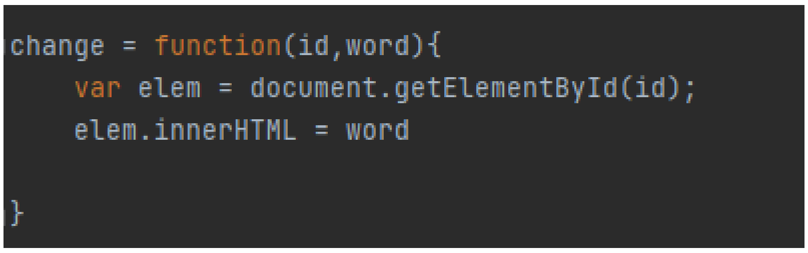
JS

Detection.js



```
var loadFile = function(event) {
  var output = document.getElementById('output');
  output.style.visibility = 'visible';

  var temp  = URL.createObjectURL(event.target.files[0]);
  output.src = temp;

  console.log(output);
  console.log(temp)
  output.onload = function() {
      URL.revokeObjectURL(output.src) // Need to do after upload

      document.getElementById('txt').style.visibility = 'visible';
      document.getElementById('Submit').style.visibility = 'visible';

      change('Title_sentence','Successful upload')

      }
};
```

This is the main front-end programming for the detection algorithm which Is able to control all animations and moving figures around the page when the user wishes to input new data into the webpage and dealing with all the client-side pre-processing to make sure that only an image is given!

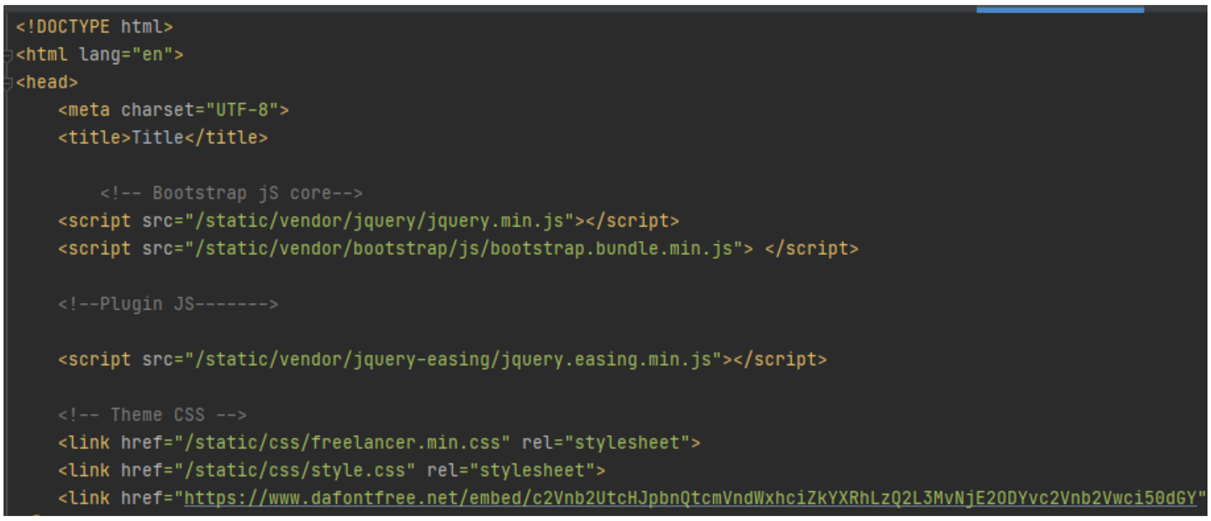


```
change = function(id,word){
    var elem = document.getElementById(id);
    elem.innerHTML = word

}
```

This function was created to make the swaps within the functions to make the code DRY and much simpler to understand.

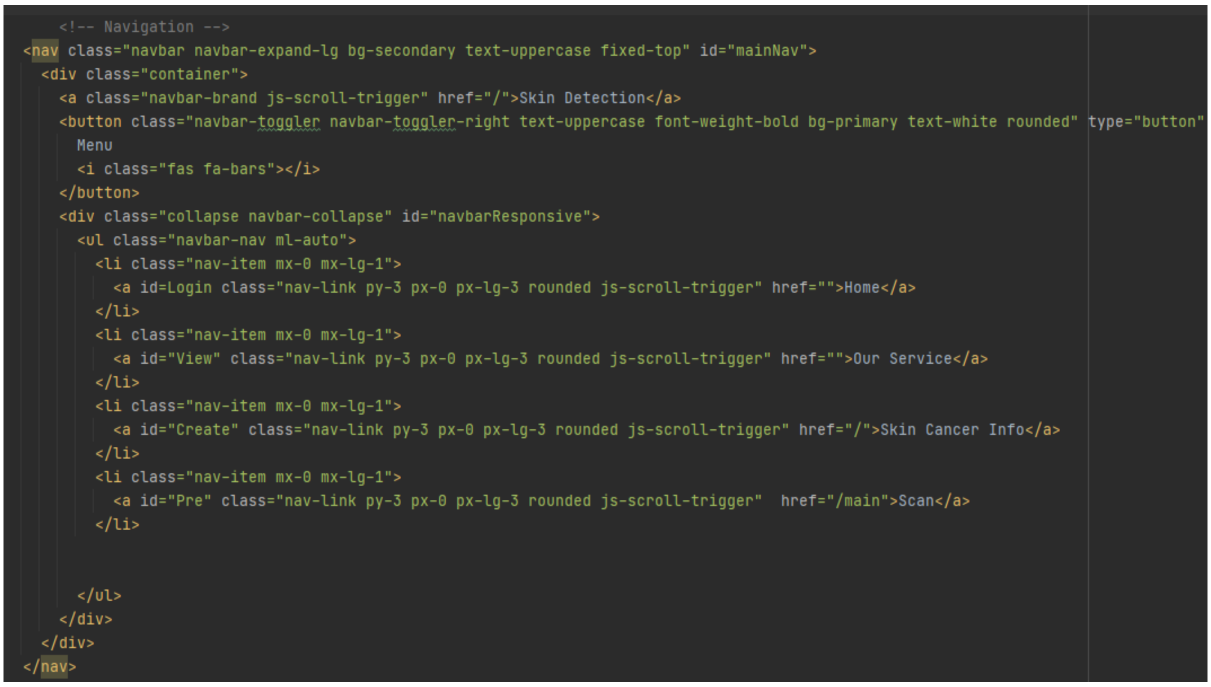
HTML

Index.html



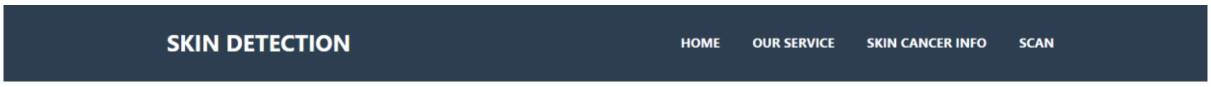
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

        <!-- Bootstrap jS core-->
    <script src="/static/vendor/jquery/jquery.min.js"></script>
    <script src="/static/vendor/bootstrap/js/bootstrap.bundle.min.js"> </script>

    <!--Plugin JS------->

    <script src="/static/vendor/jquery-easing/jquery.easing.min.js"></script>

    <!-- Theme CSS -->
    <link href="/static/css/freelancer.min.css" rel="stylesheet">
    <link href="/static/css/style.css" rel="stylesheet">
    <link href="https://www.dafontfree.net/embed/c2Vnb2UtcHJpbnQtcmVndWxhciZkYXRhLzQ2L3MvNjE2ODYvc2Vnb2Vwci50dGY"
```

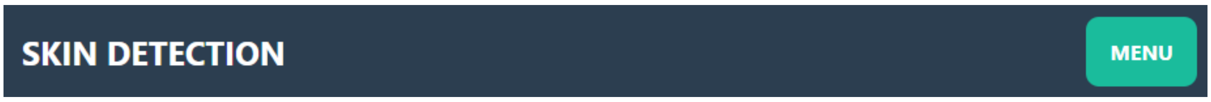
This section of html is the beginning of the head tag of the main template of all the html functions, it has imported all the different CSS and JS files as well as the font that I chose to write this website within. This font has been discussed within the analysis and the other snippets have discussed the CSS and JS files.

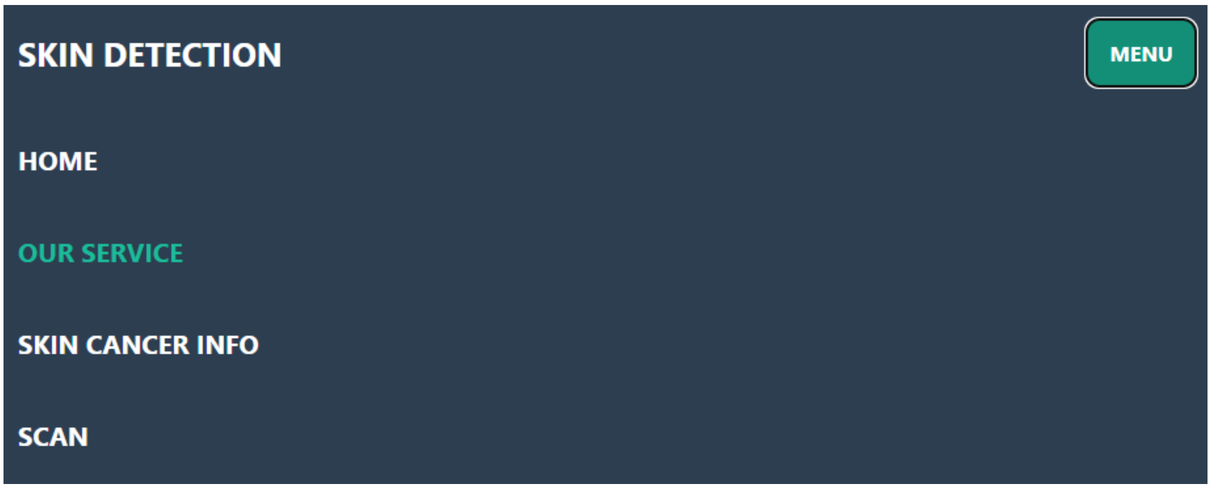


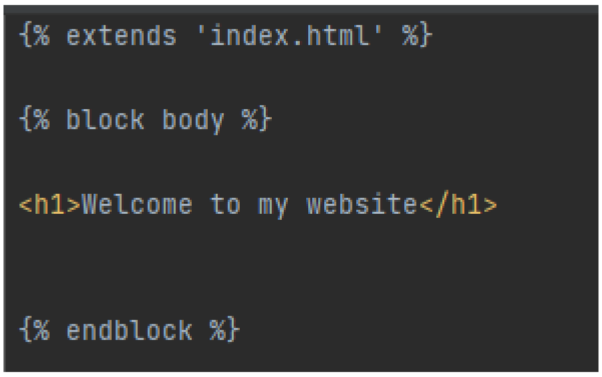
```html
    <!-- Navigation -->
 <nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-top" id="mainNav">
   <div class="container">
     <a class="navbar-brand js-scroll-trigger" href="/">Skin Detection</a>
     <button class="navbar-toggler navbar-toggler-right text-uppercase font-weight-bold bg-primary text-white rounded" type="button"
       Menu
       <i class="fas fa-bars"></i>
     </button>
     <div class="collapse navbar-collapse" id="navbarResponsive">
       <ul class="navbar-nav ml-auto">
         <li class="nav-item mx-0 mx-lg-1">
           <a id=Login class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="">Home</a>
         </li>
         <li class="nav-item mx-0 mx-lg-1">
           <a id="View" class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="">Our Service</a>
         </li>
         <li class="nav-item mx-0 mx-lg-1">
           <a id="Create" class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="/">Skin Cancer Info</a>
         </li>
         <li class="nav-item mx-0 mx-lg-1">
           <a id="Pre" class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger"  href="/main">Scan</a>
         </li>


       </ul>
     </div>
   </div>
 </nav>
```

This is the navigation bar at the top of the page of the base of the website which will be independent of what page you are currently on so will not be within a specific page HTML page but is written as a general case for every single webpage. This can be seen below and has a responsive design which allows it to be viewed on smaller devices such as tablets





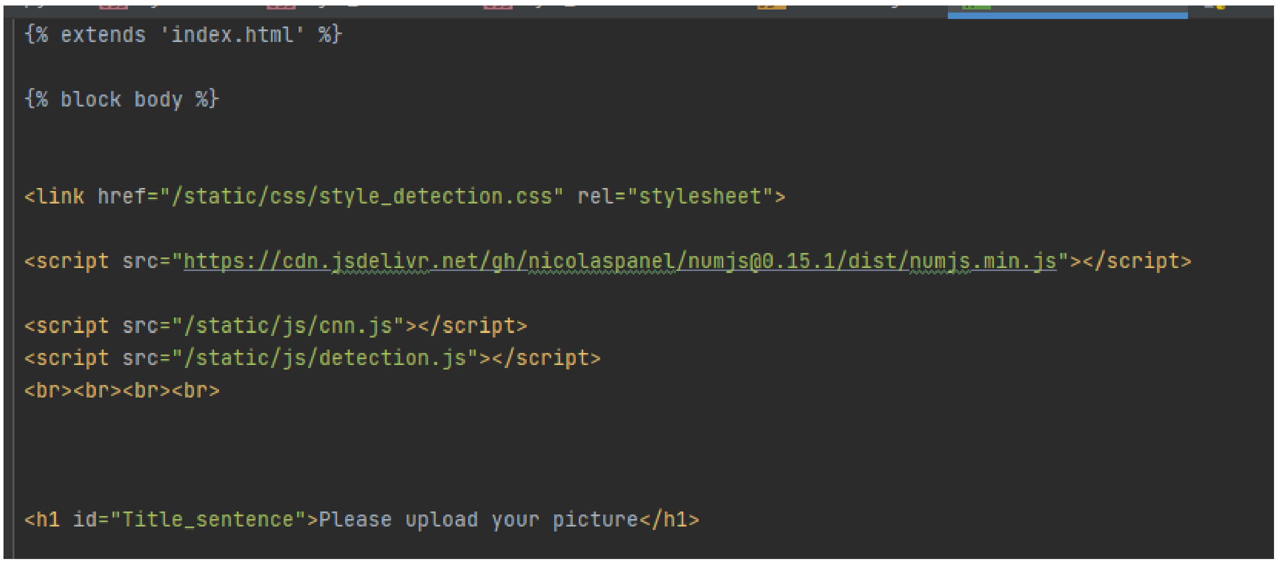




```
{% extends 'index.html' %}

{% block body %}

<h1>Welcome to my website</h1>


{% endblock %}
```

Here is an example of a block within the Jinja2 syntax which works off the premise that each webpage will code something different within this section and has been used to make sure that each page follows a similar theme to make sure that the website looks professional according to success criteria!

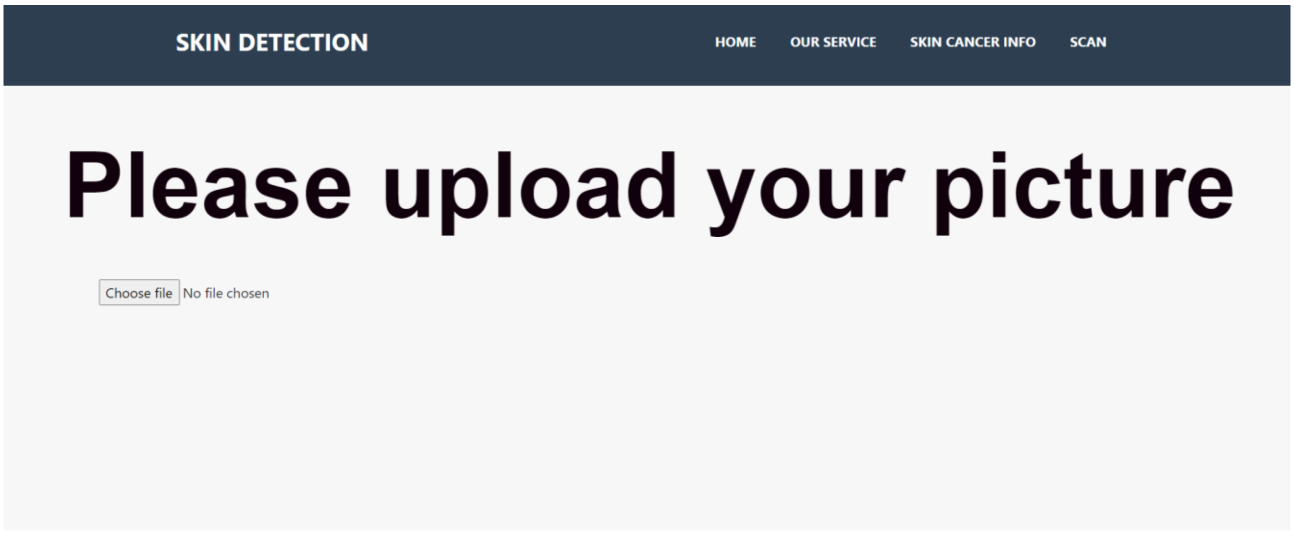
Detection.html



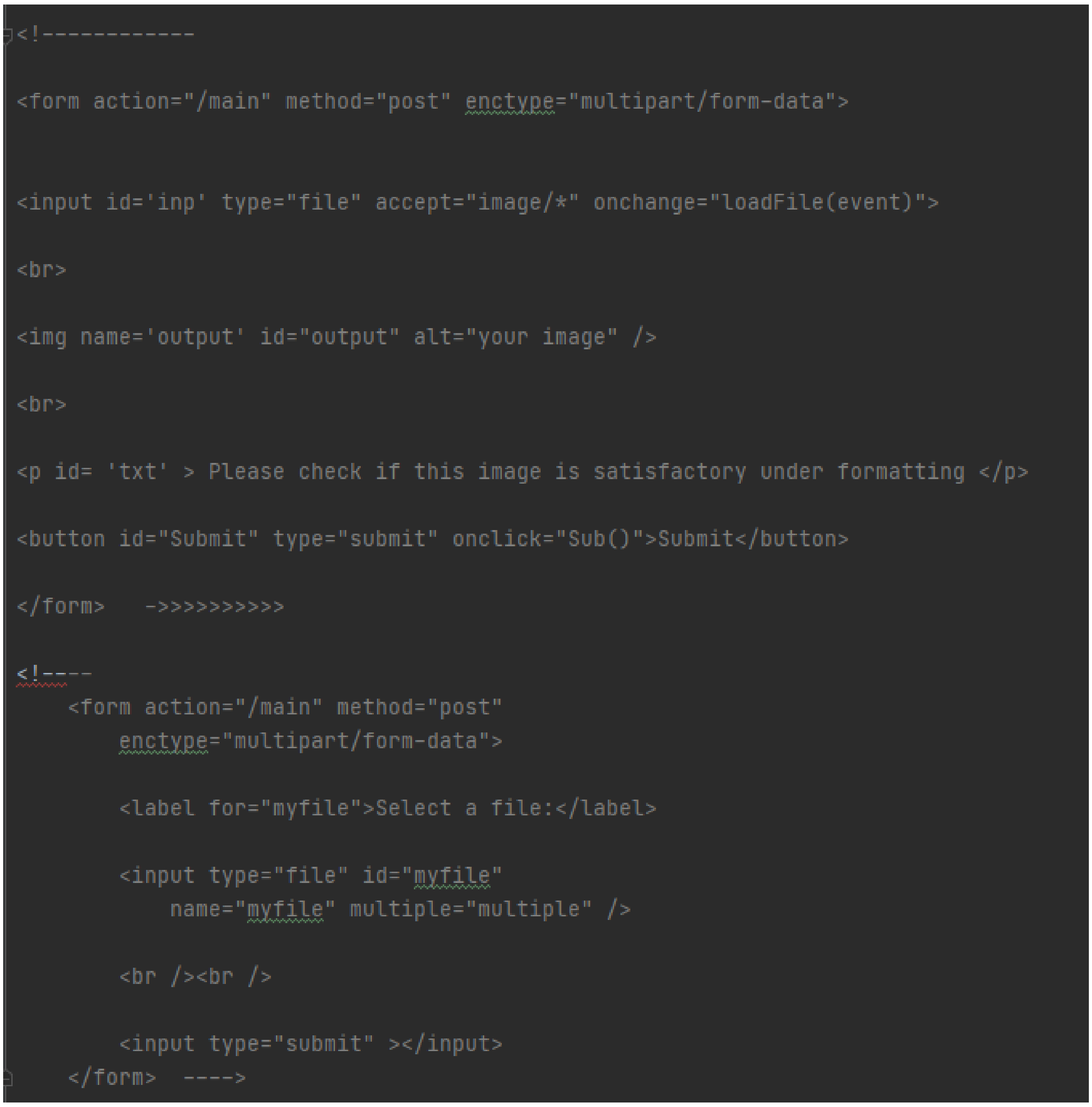
```
{% extends 'index.html' %}

{% block body %}


<link href="/static/css/style_detection.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/gh/nicolaspanel/numjs@0.15.1/dist/numjs.min.js"></script>

<script src="/static/js/cnn.js"></script>
<script src="/static/js/detection.js"></script>
<br><br><br><br>



<h1 id="Title_sentence">Please upload your picture</h1>
```

Here is the first half of the detection HTML page which acts as a head tag and will show the large title that can be seen below in a big font as well as import some more CSS and JS files which are also explained above.

The appropriate IDs have also been used to make sure that the website can be accessed anywhere.



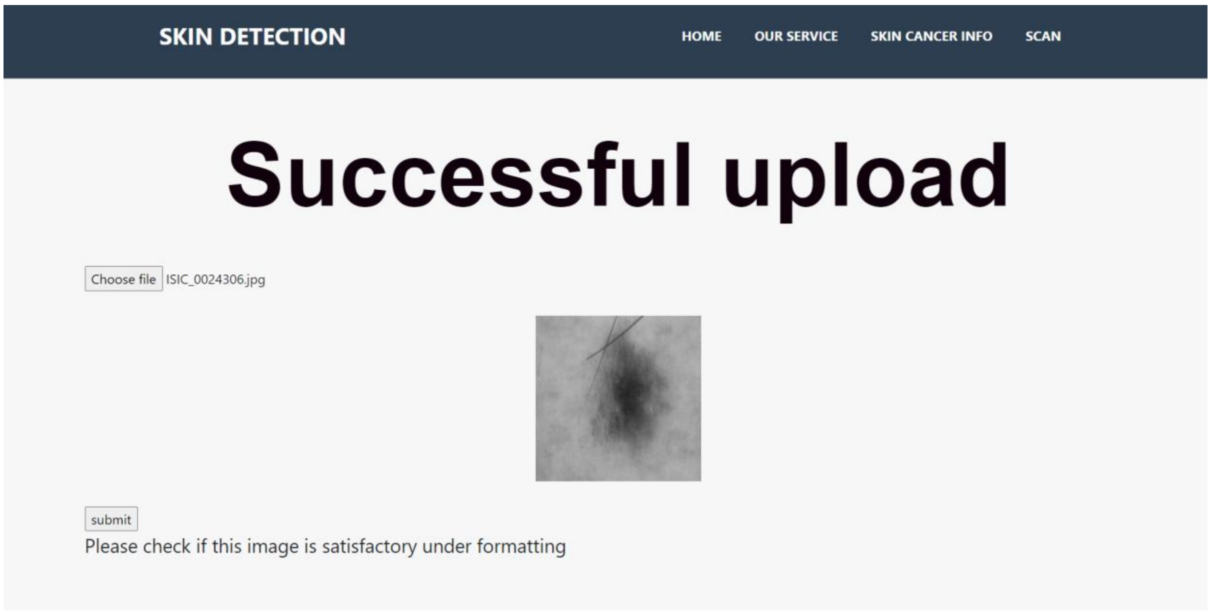
Please upload your picture



```
<!------------

<form action="/main" method="post" enctype="multipart/form-data">


<input id='inp' type="file" accept="image/*" onchange="loadFile(event)">

<br>

<img name='output' id="output" alt="your image" />

<br>

<p id= 'txt' > Please check if this image is satisfactory under formatting </p>

<button id="Submit" type="submit" onclick="Sub()">Submit</button>

</form>   ->>>>>>>>>>

<!----
    <form action="/main" method="post"
        enctype="multipart/form-data">

        <label for="myfile">Select a file:</label>

        <input type="file" id="myfile"
            name="myfile" multiple="multiple" />

        <br /><br />

        <input type="submit" ></input>
    </form>  ---->
```

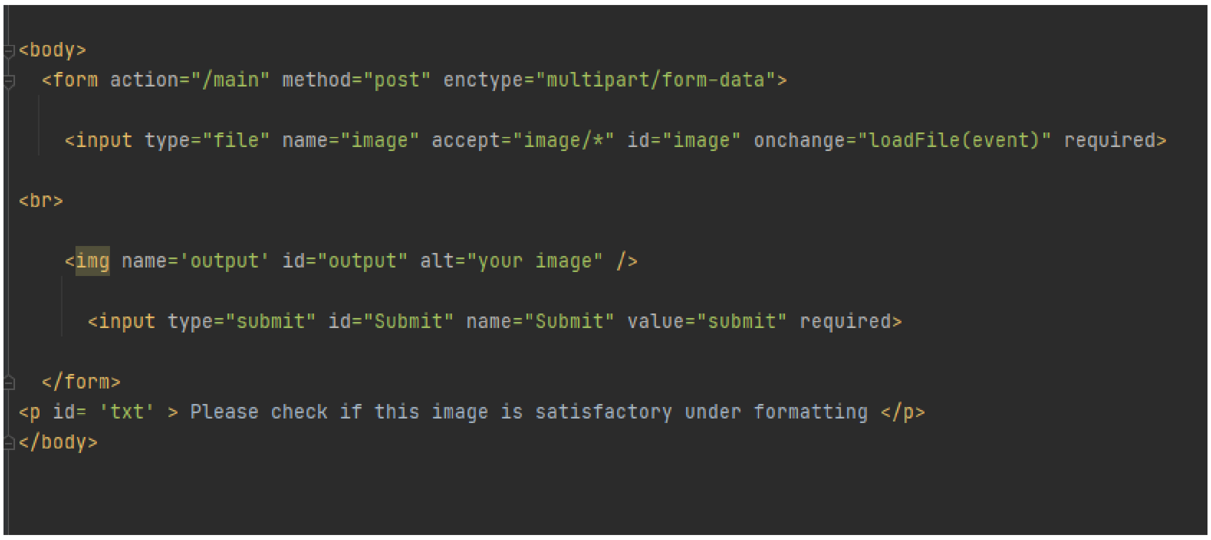
As you may have noticed this code is commented out because it is from a failed attempt to produce the screenshot below. The first method I tried to use was JS based using event loaders however, this method proved to be extremely complicated therefore instead I tried a singularly based JS approach which did work however, this method was completely reliant on client-side processing therefore was not safe for use within a webpage therefore was permanently changed to the code below.

This code failed previous test stages when trying to feed an image through the system therefore has to be rewritten in order to pass the test!



Successful upload

Please check if this image is satisfactory under formatting



```
<body>
  <form action="/main" method="post" enctype="multipart/form-data">

    <input type="file" name="image" accept="image/*" id="image" onchange="loadFile(event)" required>

<br>

    <img name='output' id="output" alt="your image" />

      <input type="submit" id="Submit" name="Submit" value="submit" required>

  </form>
<p id= 'txt' > Please check if this image is satisfactory under formatting </p>
</body>
```

This code here was much simpler than the previous 2 attempts and simply used form tags and a single event loader to send the image to a JS file for client-side pre-processing before sending it to the Python file for actual processing by the CNN.

All movement and animations for this website is located within the JS file to keep the code modularised

Feed an image through the system Here is a lesion taken from Subil Philip	Test pass because it has been uploaded and then printed out into the console which is an accurate prediction therefore this has been successful. `(1, 0.3446460664640328)` This was outputted when a BCC was passed through the system, and it was successfully predicted

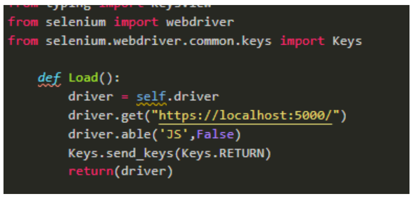
Stage 4 review

This stage was all about the actual building of the website therefore was a stage which was important not only to do accurately but also in a way to make my product presentable to others as this is the face of the product and will be what is scrutinised.

How is has been tested

This has mainly been tested in extremely small increments by making small changes on the website and seeing what physical changes needed to be made upon the HTML and gradually working up the code from there. Then to test robustness, the entry bar was tested with different forms of data as explained within the test plan.

Additionally, to test the website I used crawlers such as selenium and try to disable JS and see how the website performs in order to test the robustness and how the website looks. (Test Passed)



```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

    def Load():
        driver = self.driver
        driver.get("https://localhost:5000/")
        driver.able('JS',False)
        Keys.send_keys(Keys.RETURN)
        return(driver)
```

An example test which I carried out and passed was to take a picture of one of my friend's skins and input it into the system and view If correct predictions have been made. This test is discussed later but was passed at this stage and during final testing.

Success Criteria points

Simple and lightweight website for users to interact with	Screenshot of the web page that is clearly laid out and is not cluttered.
Robust protected website	Evidence of attempts to steal data from the website and the precautions put in place
Easy to use website	Screenshot of every stage of the upload process and each stage being explicitly explained for the user
Non-complicated home page	^

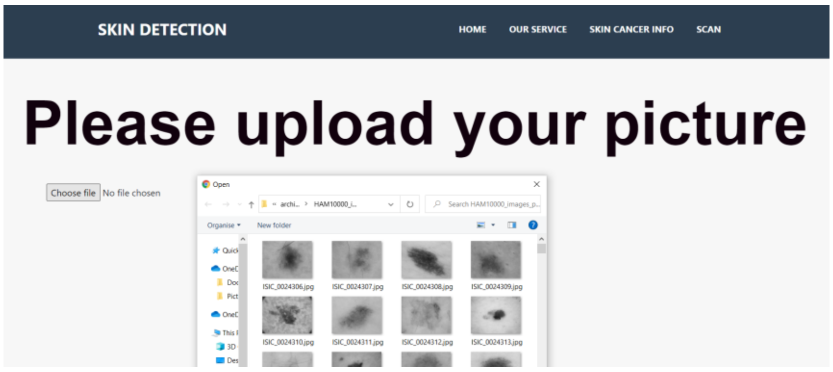
Changes

As part of this stage, I decided to make the change of moving the project from a 7-case classification problem to a 2-case problem in order to help keep the UI stable and make it a more user-friendly experience rather than overwhelming the user with too much detail. This also allows me to get a much higher prediction with the CNN.

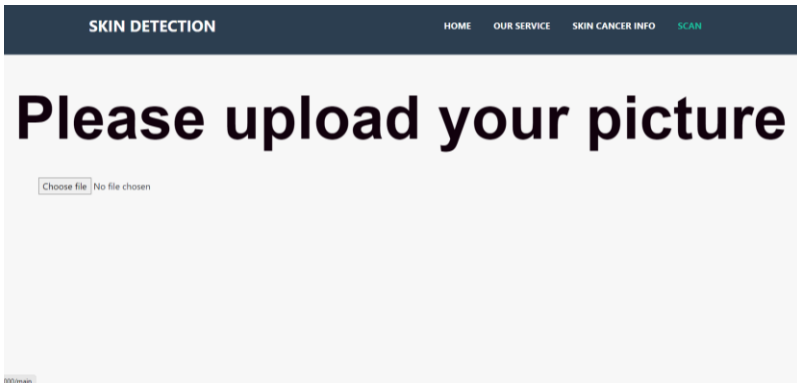
Summary of prototype

At this stage the project is almost complete, all that is needed is the final testing using the stakeholders before this is a complete product. Therefore, this product is almost complete!

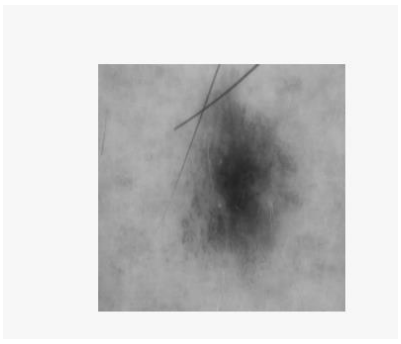
Stage 5 Final Testing using Stakeholders



Easy to use input bar



Highlighting appropriate areas



Picture live pre-processed

These images of the webpage were then showed to the stakeholders in order to get their feedback whilst doing the final stage of testing of the product.

Stakeholder Review

After showing the UI to the users they reviewed that the colours were a bit bland on the website however, after showing them the animation within the website and the change in colours to green upon the website they were satisfied with the colour scheme and mentioned that perhaps I should change my font sizes on the home page.

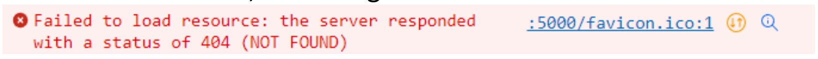
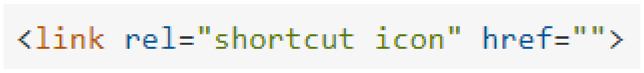
These were then immediately changed to help make the users happier and make the product overall.

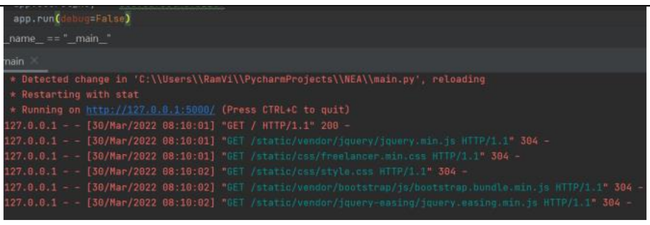
Some of the users as to how the predictions were being formed upon the website therefore, I chose to make an additional page to show the users the code and explain to them how the CNN works.

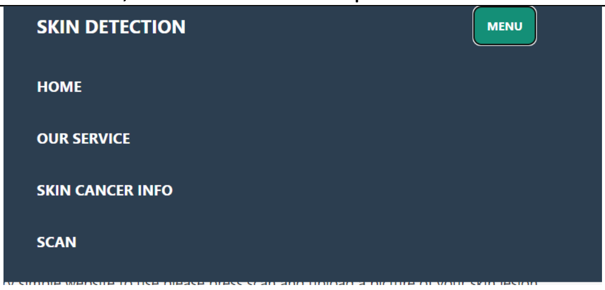
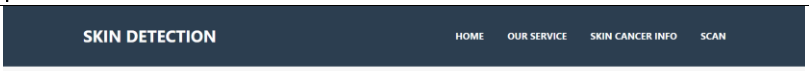
Other than that, the Stakeholders were impressed with the website and said that it was fine to use for a product in real world and had a professional and sleek air about the product

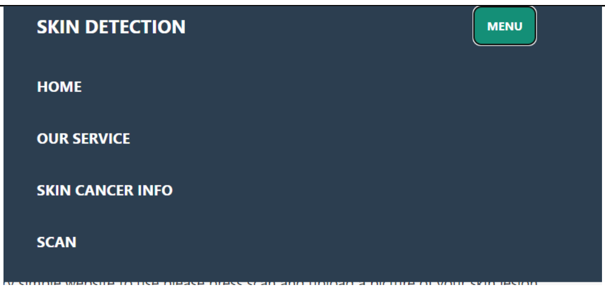
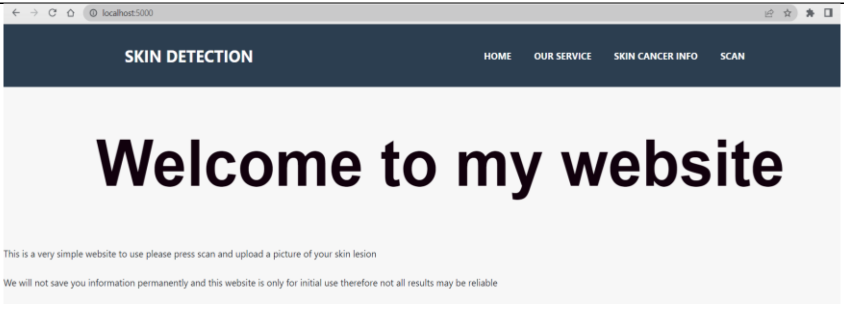
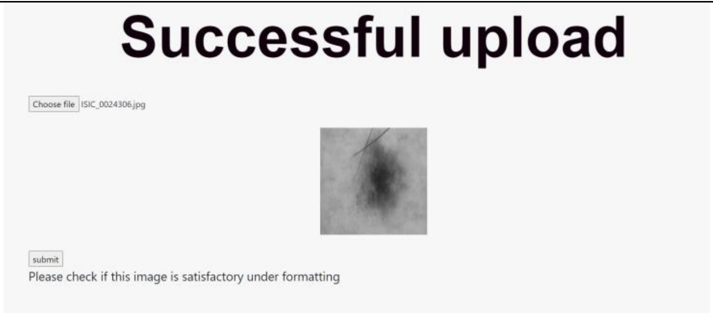
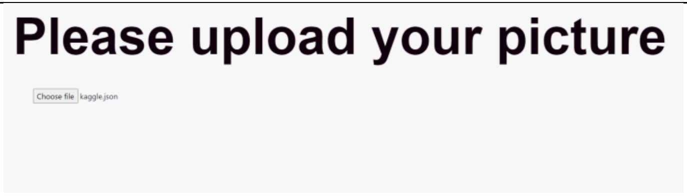
Additionally, there were some questions as to how the website will be accessed and how it could be hosted on different machines and I explained that it could be hosted for free online, which will allow users to use this service!

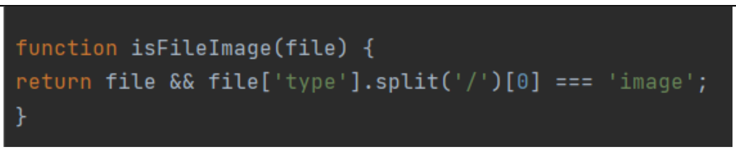
A final comment of this product was that they would like to have an impact on the post development section therefore I feel like their opinions should be added within the evaluation sections after the testing has been fully completed in order to make sure that the product is what the stakeholders were expecting and to make sure that they agree with each success criteria point and if they agree that it works!

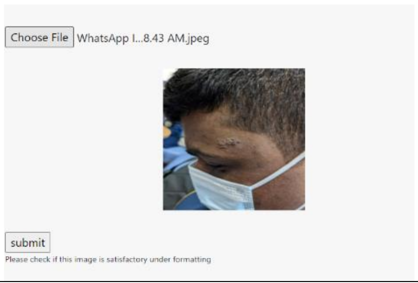
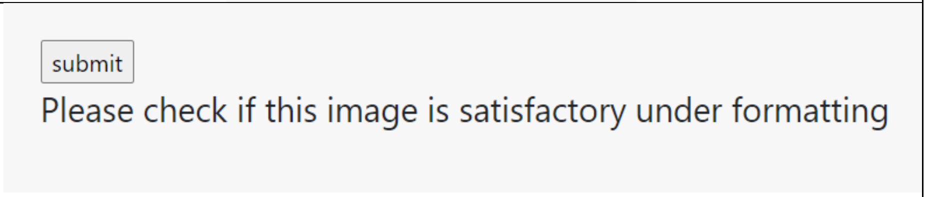
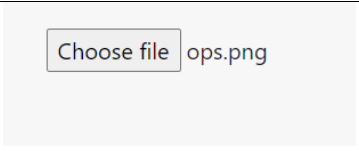
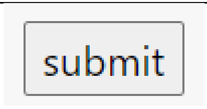
Stg.	Action to test	Working (Y/N)
4	Website loads with no console bugs	Yes, the website loads almost instantaneously when loaded in a google chrome browser. However, on an edge browser there is a small failure being  `Failed to load resource: the server responded with a status of 404 (NOT FOUND)` `:5000/favicon.ico:1` However, this is a standard error and comes with the bootstrap package that have been utilised therefore overall this test is a pass eventhough there was still a small error within the page However, remedial action has since been made to make sure that this error does not appear and was only a small fix being this code added into the header of the website so that there is a default icon which is blank `<link rel="shortcut icon" href="">` Therefore, now this test is fully passed

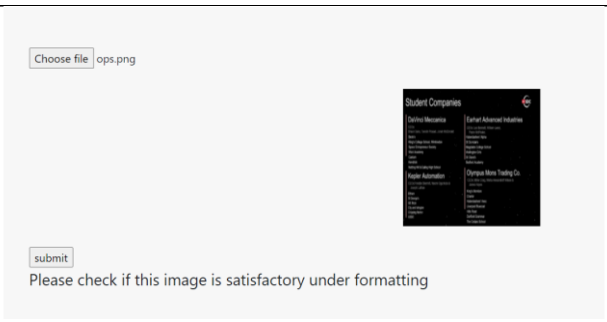
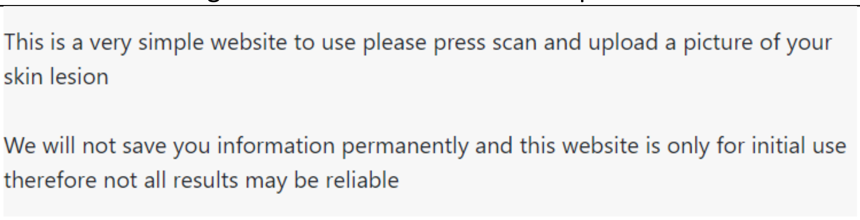
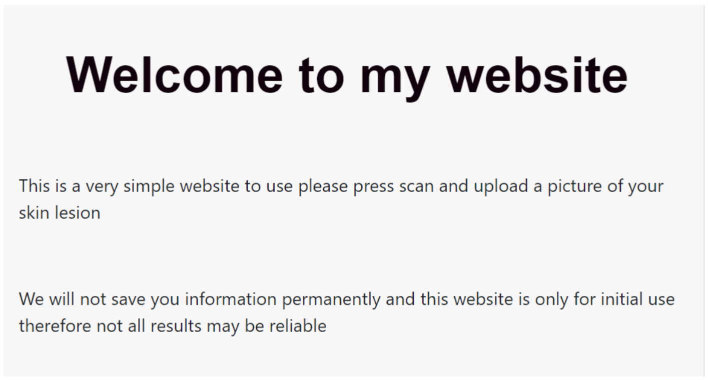
4	Website runs in non-development mode	`app.run(debug=False)`
		`_name_ == "_main_"`
		`main`
		`* Detected change in 'C:\\Users\\RamVi\\PycharmProjects\\NEA\\main.py', reloading`
		`* Restarting with stat`
		`* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)`
		`127.0.0.1 - - [30/Mar/2022 08:10:01] "GET / HTTP/1.1" 200 -`
		`127.0.0.1 - - [30/Mar/2022 08:10:01] "GET /static/vendor/jquery/jquery.min.js HTTP/1.1" 304 -`
		`127.0.0.1 - - [30/Mar/2022 08:10:01] "GET /static/css/freelancer.min.css HTTP/1.1" 304 -`
		`127.0.0.1 - - [30/Mar/2022 08:10:02] "GET /static/css/style.css HTTP/1.1" 304 -`
		`127.0.0.1 - - [30/Mar/2022 08:10:02] "GET /static/vendor/bootstrap/js/bootstrap.bundle.min.js HTTP/1.1" 304 -`
		`127.0.0.1 - - [30/Mar/2022 08:10:02] "GET /static/vendor/jquery-easing/jquery.easing.min.js HTTP/1.1" 304 -`
		In order to pass this test, before running the actual web server, all I needed to do was to turn off development mode so that last loading will not occur, and no python errors will display within the page
		
		Therefore, this test has been passed

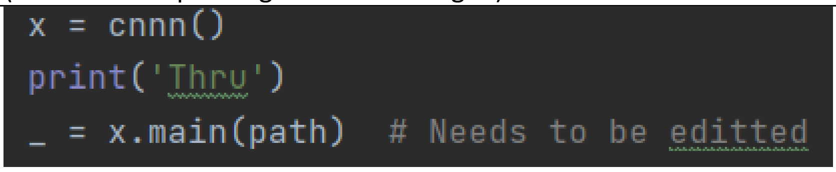
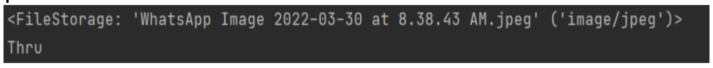
4	Menu button functional	 SKIN DETECTION MENU HOME OUR SERVICE SKIN CANCER INFO SCAN To test the functionality, I pressed the button while in small screen and large screen view to make sure it worked, and the image above is an example of what the menu was supposed to look like therefore this is a passed test
4	Menu adaptive to screen size	 SKIN DETECTION HOME OUR SERVICE SKIN CANCER INFO SCAN Here is a picture of the adaptive screen sized menu which is for larger screens and that for smaller screens in a drop-down menu can be viewed in the previous cell. Therefore, this point has been passed

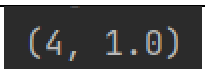
4	Menu has a drop-down menu with various links	 SKIN DETECTION MENU HOME OUR SERVICE SKIN CANCER INFO SCAN This test has been passed as can be seen in this screenshot
4	All links in menu redirect to correct page	This cannot be showed within a screenshot, but within the menu drop down each page links to the correct location within the webpage therefore this section is overall a pass.
4	Info page will output general informatio n to user	This page has been failed, however, to fix this I have made sure to discuss with the stakeholder to find an alternative to this and have put general information onto the home page of the web page alternatively, so even though this test has failed, the stakeholders have been involved and a different solution has been met
4	Home page is auto loaded and discusses website	 localhost:5000 SKIN DETECTION HOME OUR SERVICE SKIN CANCER INFO SCAN Welcome to my website This is a very simple website to use please press scan and upload a picture of your skin lesion We will not save you information permanently and this website is only for initial use therefore not all results may be reliable This is the base page of the website which has been loaded without the need to click on a specific link but the moment the base website is tried to access, it will automatically forward to this page therefore this page is an evident pass.
3	Upload of file (JPEG or PNG)	Successful upload Please check if this image is satisfactory under formatting Image is previewed and confirmation is asked from the user which is the correct procedure therefore this test is a pass
3	Upload of file (non-image)	Please upload your picture This test is a failure because the file can still be submitted. Therefore, client-side processing was required to make sure that this cannot happen! Therefore, this code has been added into the JS file

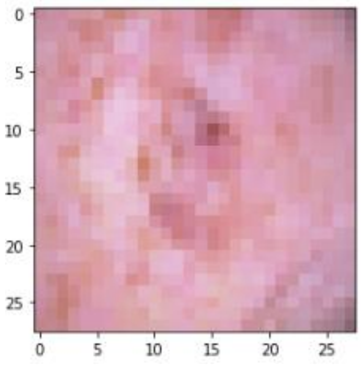
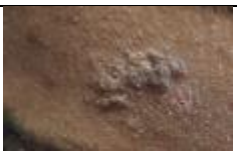
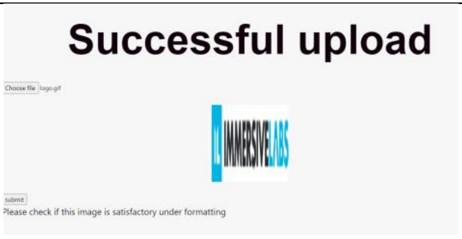
		
		`function isFileImage(file) {`
		`return file && file['type'].split('/')[0] === 'image';`
		`}`
		This code will use file signatures as well as metadata in order to check the type of file that has been uploaded and is an appropriate remedy because now JSON and other files cannot be committed into the database

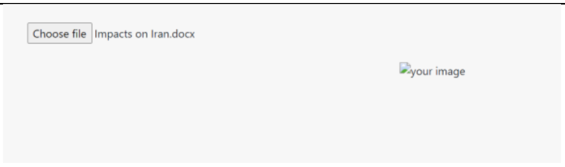
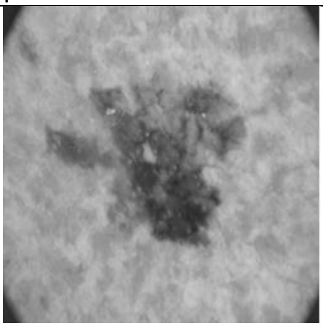
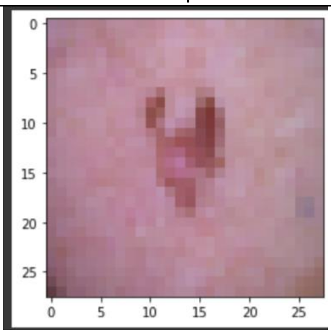
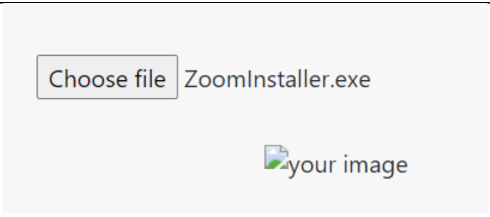
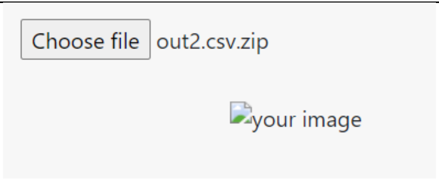
3	Upon upload image is edited	This test has been passed and can be proved by inputting a random image and checking if its formatting has changed This table has clearly been reformatted and edited therefore this test is passed.
3	After editing image is displayed	This test has been passed and can be seen in this screenshot  Choose File WhatsApp I...8.43 AM.jpeg submit Please check if this image is satisfactory under formatting
3	After display image confirmation is triggered	 submit Please check if this image is satisfactory under formatting In order to display the image confirmation, an image has to be inputted, which has done therefore this test has been passed
3	After image is displayed upload option removed	 Choose file ops.png This test has been passed because if they choose to press the choose file, they are allowed to send a new/adapted file for potential upload
3	Allow submit of image	 submit This test has been passed as there is a submission box

4	Allow rejection of image	 Choose file ops.png submit Please check if this image is satisfactory under formatting Choose new file is equivalent to the rejection of an image therefore although this test has been failed, the functionality is already there on the website and perhaps this may need to be presented onto the website.
3	Image stored only temporaril y	Yes, this test cannot be shown but it is placed in a temporary folder where each image is flushed out once it is no longer needed and is deleted from blob storage as soon as possible
3	Image file cannot be accessed	`URL.revokeObjectURL(output.src)` This code makes sure that the image cannot be accessed whilst stored within blob storage therefore this test has been passed
4	Instruction s on screen clear and well formatted	 This is a very simple website to use please press scan and upload a picture of your skin lesion We will not save you information permanently and this website is only for initial use therefore not all results may be reliable This information was present however, this failed the test and has to be reformatted in order to please the stakeholders and eventually pass this test  Welcome to my website This is a very simple website to use please press scan and upload a picture of your skin lesion We will not save you information permanently and this website is only for initial use therefore not all results may be reliable (Font size and padding has been changed)

2	After submission data loaded into CNN	
		`x = cnnn()`
		`print('Thru')`
		`_ = x.main(path)   # Needs to be editted`
		The data has been loaded into the CNN successfully and while debugging the word thru being printed is the indication that the code has reached this point
		
		`<FileStorage: 'WhatsApp Image 2022-03-30 at 8.38.43 AM.jpeg' ('image/jpeg')>`
		`Thru`
		Here is the proof of this which shows the test has been passed

2/3	Results from CNN outputted onto website	 `(4, 1.0)` This test has been purposefully failed due to new stakeholder information that the diagnosis should not be informally outputted on a website therefore the diagnosis has been printed in the console and ideally would be read out to user who is using the application

Stg.	Input	Test Type	Result
2	RGB image from database	Valid	 This is a rendering of the image that has been loaded from within the dataset and plotted using matplotlib to check if the image is being rendered accurately Passed
3	Skin Lesion image taken	Valid	 Here is an actual skin lesion sample that has been taken and worked successfully in this case as it was run through the database and picked up on the pink of the sample and diagnosed correctly
4	Logo of product	Extreme	 **Successful upload** Choose file logo.gif IMMERSIVELABS submit Please check if this image is satisfactory under formatting For the upload of a logo into the system, this test is passed.

3	Document	Invalid	 Choose file Impacts on Iran.docx your image The document has been caught by the new client-side processing algorithm therefore has been passed
2	B/W image (database)	Valid	 This test was a pass and was sent through the CNN
2	Another RGB image	Valid	 Tests passed
4	Exe file	Invalid	 Choose file ZoomInstaller.exe your image Test Passed
4	Bat file	Invalid	 Choose file out2.csv.zip your image Test passed

Action/Check	Test Type	Result
RGB Image storage time	Valid	Successful
Logo of product time	Valid	Successful
Rejected image storage	Valid	Successful
Bat File	Invalid	Successful

Stage 5 Review (Testing)

How is has been tested

Used user feedback and improve website while developing and making small changes to the UI such as a colour changes to make the website looks good.

Success Criteria points

Simple and lightweight website for users to interact with	Screenshot of the web page that is clearly laid out and is not cluttered.
Robust protected website	Evidence of attempts to steal data from the website and the precautions put in place

Summary of prototype

At this stage, the project is complete as all tested that have been attempted have been either passed or adapted to the point where the function operation is reasonable. The next stage is to go back the success criteria and fund out if they have been passed.

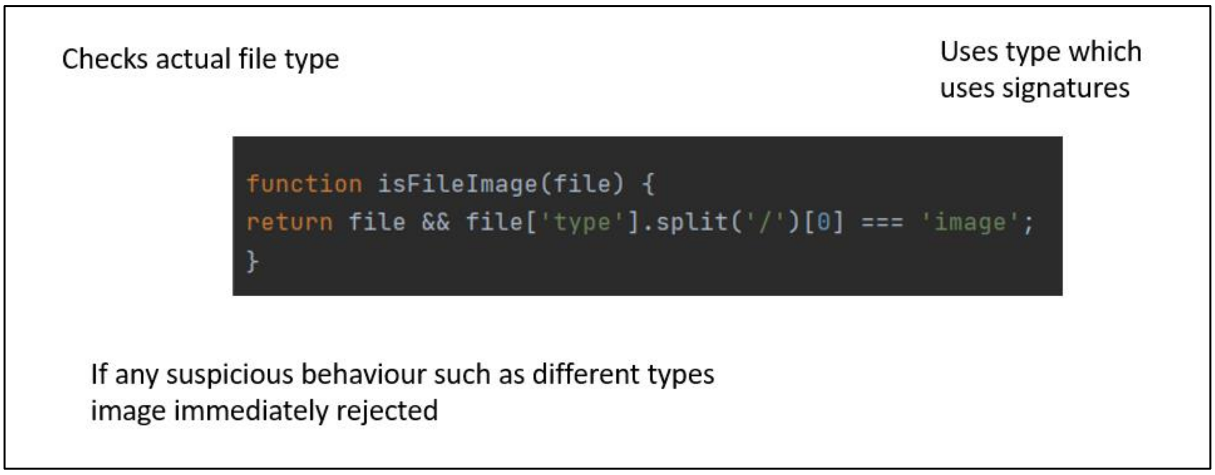
Evaluation

Functions and usability of UI

Robustness

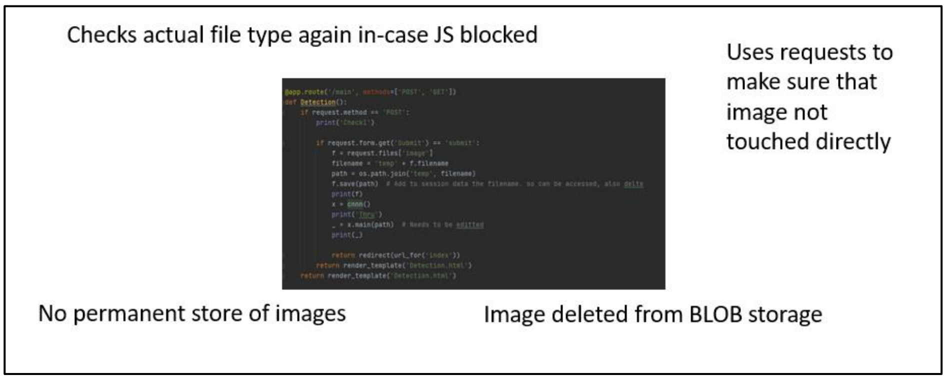
In terms of robustness, overall, the website is quite secure and most of the test cases run through validation routines to make sure that the website is fully functional as well as fully secures in its nature. This includes both client and server-side validation!

Client Side:


Checks actual file type
Uses type which uses signatures
If any suspicious behaviour such as different types image immediately rejected

```
function isFileImage(file) {
return file && file['type'].split('/')[0] === 'image';
}
```

Server Side:


Checks actual file type again in-case JS blocked
Uses requests to make sure that image not touched directly
No permanent store of images
Image deleted from BLOB storage

```
@app.route('/main', methods=['POST', 'GET'])
def Detection():
    if request.method == 'POST':
        print('Check1')

        if request.form.get('Submit') == 'submit':
            f = request.files['image']
            filename = 'temp' + f.filename
            path = os.path.join('temp', filename)
            f.save(path)  # Add to session data the filename. so can be accessed, also delte
            print(f)
            x = cnnn()
            print('Thru')
            _ = x.main(path)  # Needs to be editted
            print(_)

            return redirect(url_for('index'))
        return render_template('Detection.html')
    return render_template('Detection.html')
```

The functionality is shown in the next section along with usability

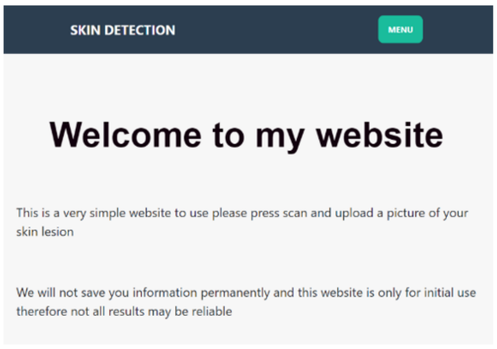
Usability

Many of these are discussed in success criteria being points 1, 3 + 4

First usability + functionality section is within UI:

Structured menu

Simple to look at



Welcome to my website

This is a very simple website to use please press scan and upload a picture of your skin lesion

We will not save you information permanently and this website is only for initial use therefore not all results may be reliable

Clear Hero section

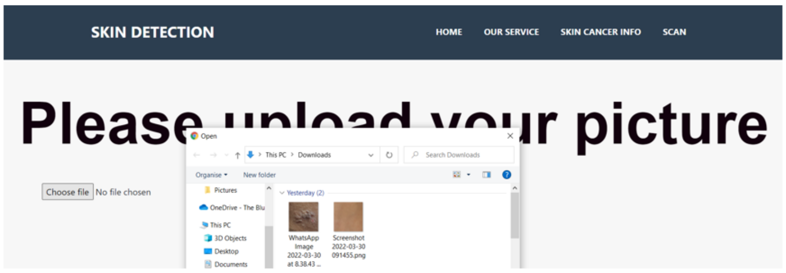
Well designed Hero Section

Useable UI

In this section UI usability has been very successful.

Big instructions on what to do within website

Easy to change images

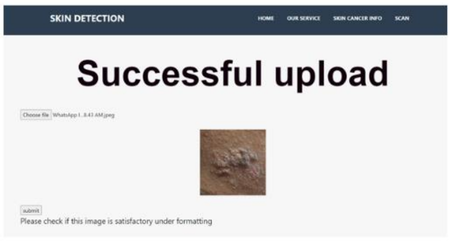


Please upload your picture

Easy to upload image, windows file explorer

Big instructions on what to do within website

Changeable images



Successful upload

Please check if this image is satisfactory under formatting

Preview of image rendered before save

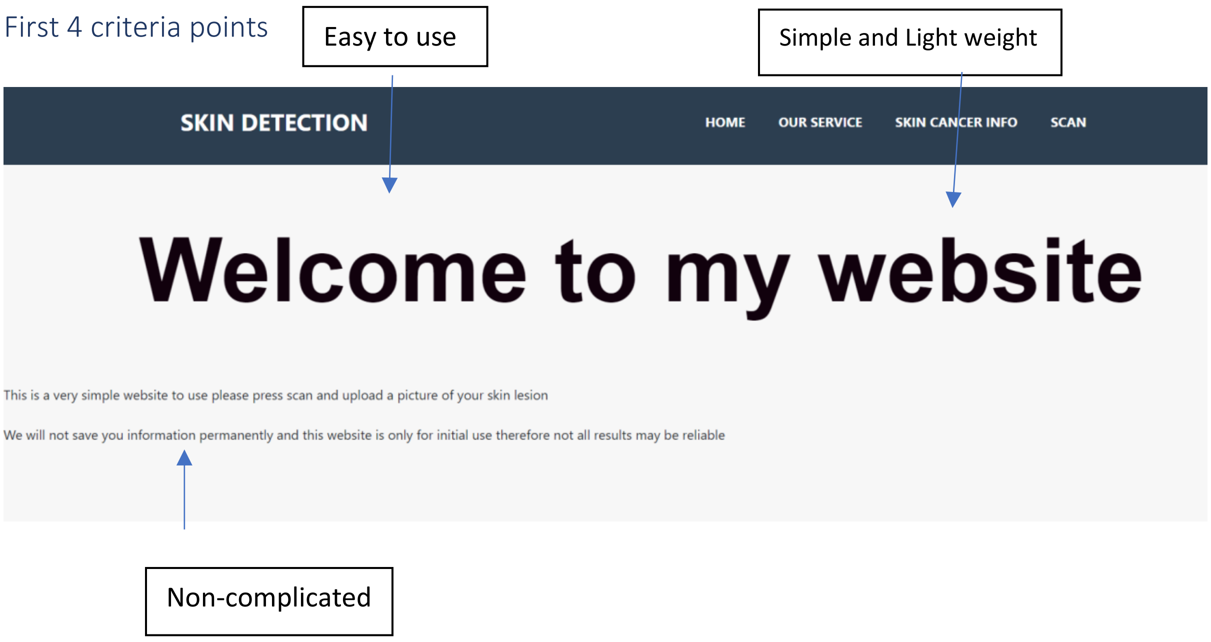
Image not yet stored

Criteria Method using test evidence

Criteria	Met?
Simple and lightweight website for users to interact with	Yes
Robust protected website	Yes
Easy to use website	Yes
Non-complicated home page	Yes
Anonymous Mode	Yes
Extra information about what skin cancer is	No
Melanoma support	Yes
Fine-tuned ML algorithms	Yes
Integration of ml algorithms together	Yes
Accurate Data Analysis	Yes

Evidence are listed as screenshots below and thoroughly analysed after talking to the stakeholders for a final time!

First 4 criteria points



Welcome to my website

This is a very simple website to use please press scan and upload a picture of your skin lesion

We will not save you information permanently and this website is only for initial use therefore not all results may be reliable

This first screenshot as discussed in previous sections shows the skeleton UI of the website which has a simple menu which is adaptive as well as a very light weight UI.

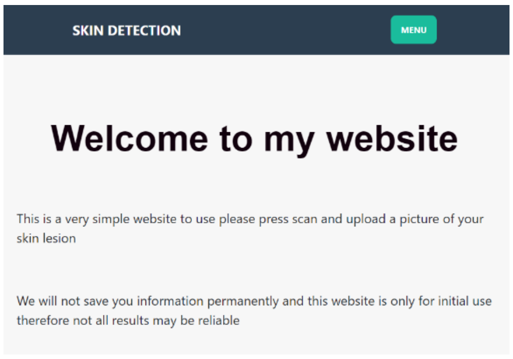
Additionally, the instructions on the page are listed simply and without a lot of meaningless language as these are kept for the information pages to make sure that the standard user will not be overwhelmed by the website therefore meeting success criteria 1 3 and 4.

After talking to stakeholders, they agreed that this page was simple and easy to use, however some argued that perhaps the page is too simple while others claimed that it was the correct amount of simplicity.

In order to meet as many stakeholder requirements as possible and create a solution which pleased as many stakeholders as possible without compromising on any key features. The solution to this was to change the tags of this introduction to <p> as well as change some of the formatting to make sure that the webpage can still be viewed on smaller devices.

I then went back to the stakeholder with this screenshot, and they said it is much improved and were impressed with the improvement whilst still keeping it simple.

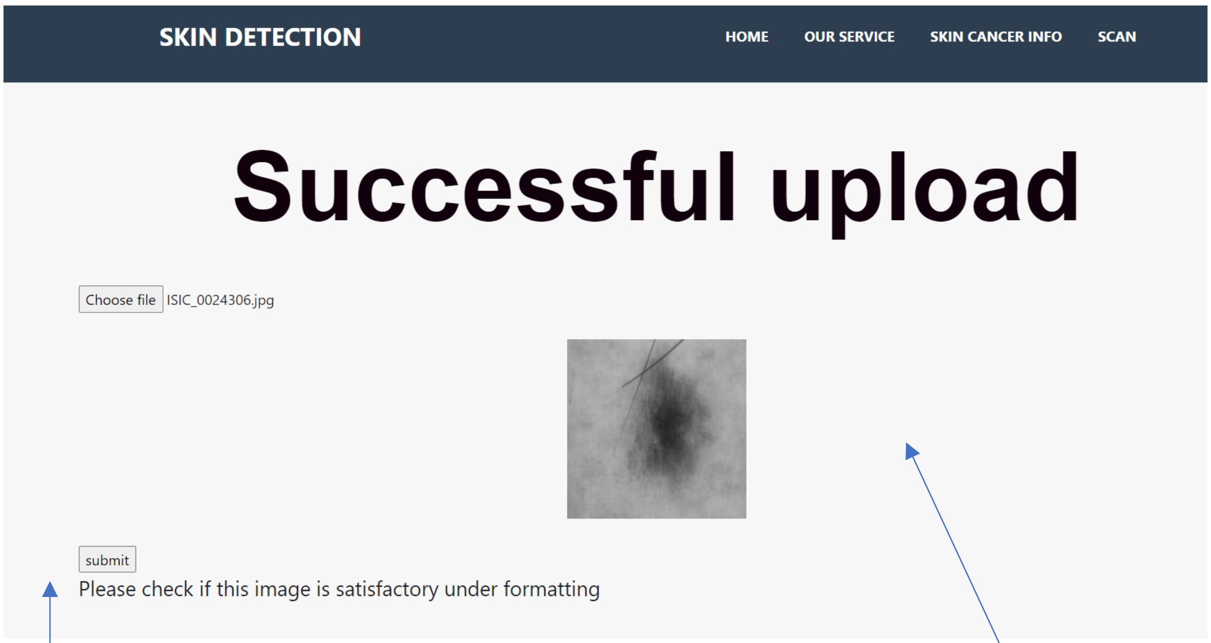
Ankita "The use of borders on either side of the page makes the website look much better and much more professional while still not making it over-whelming"



Welcome to my website

This is a very simple website to use please press scan and upload a picture of your skin lesion

We will not save you information permanently and this website is only for initial use therefore not all results may be reliable



Successful upload

Please check if this image is satisfactory under formatting

Non-complicated

Simple and Light weight

This is the second page that I showed the stakeholders in order to test if the first 4 points of the success criteria have been met. This page was also used for checking the main function of the website to give melanoma support to the users by allowing them to upload an image and outputting the correct answers to the user.

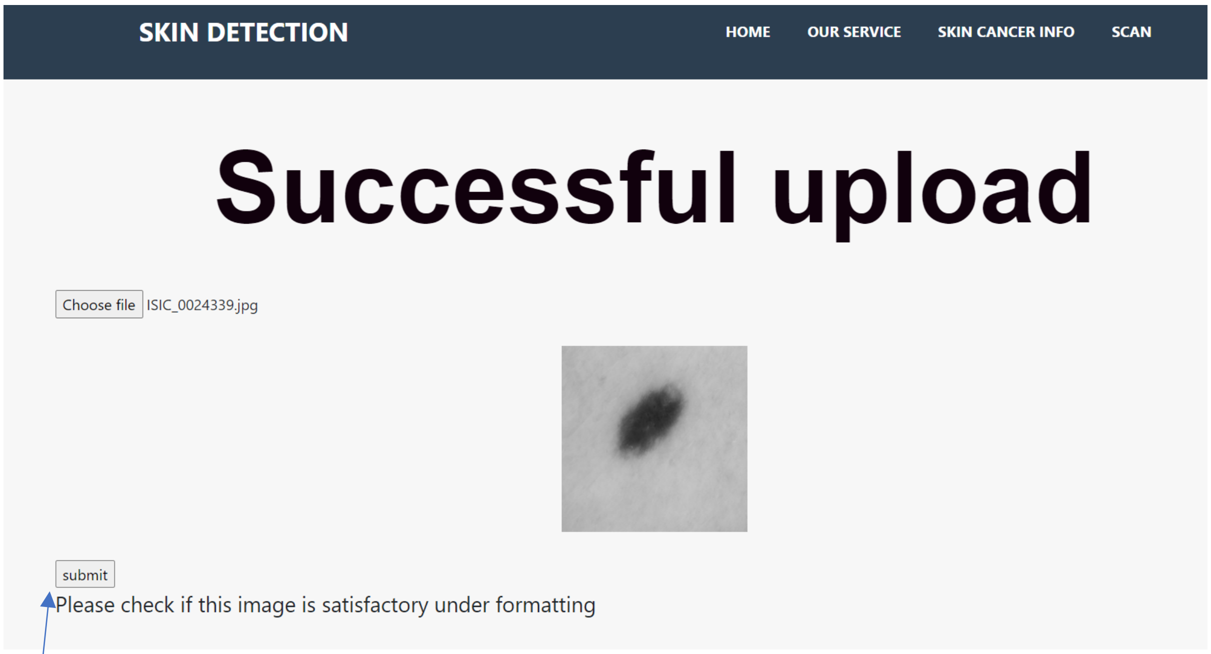
For this page overall the stakeholders were content with the layout, claiming that it was extremely simple to use and stepped through carefully for the user.

Subil "I was impressed with the fact that the website is able to allow me to preview my image before uploading it as well as change the image last minute if that is what is required. It seems simple enough, but I would question the security of this page as it seems easy enough to upload whatever file you would like"

I then assured the stakeholders that the website was still secure by showing them the client-side validation however, I felt the need to explain this which suggests that this should be added onto the information page of the website so users are assured of the security, therefore this is an important change which will need to be implemented

Other than this, the users were in consensus that this website meets the first 4 criteria points as well as the 'Melanoma Support' bullet point.

Point 5

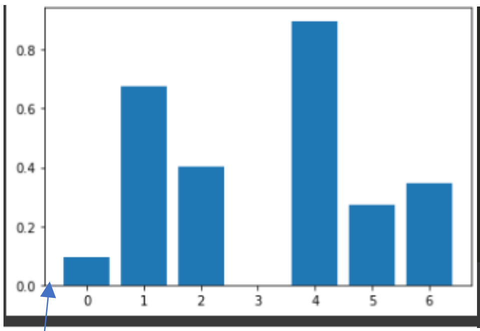


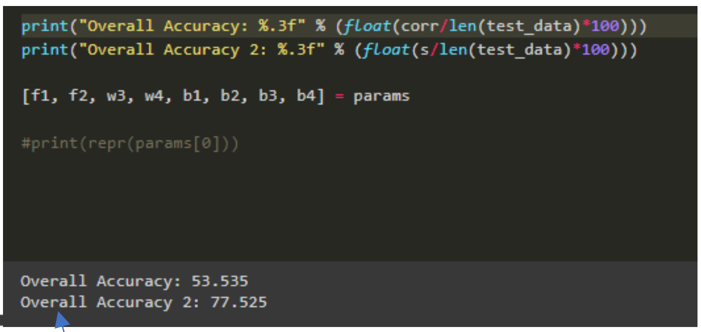
Anonymous; No information of users given

As part of the anonymous mode which has been implemented, the users are not required to input any extra features such as their names or any personal data such as past melanomas and the localisation of any other lesions upon the skin.

Additionally, I have chosen not to display the results of the CNN immediately even though there is direct feedback to make sure that the users are not stressed or overworked due to the results, therefore more personal feedback such as by the local GP or with their personal friends/doctors therefore this success criteria could have been developed further.

Point 8,9,10





```
print("Overall Accuracy: %.3f" % (float(corr/len(test_data)*100)))
print("Overall Accuracy 2: %.3f" % (float(s/len(test_data)*100)))

[f1, f2, w3, w4, b1, b2, b3, b4] = params

#print(repr(params[0]))
```

```
Overall Accuracy: 53.535
Overall Accuracy 2: 77.525
```

Fine-Tuned ML

Data Analysis + Double Integration

Here is the proof of the CNN working and we currently have a 77.5% efficient on my test set. This is much higher than all previous iterations and is clearly not biased towards one type of cancer compared to others.

This section did not need stakeholder input but from the CNN a percentage of 77% from the test data is extremely high because any higher and we run the risk of over-fitting the data onto the dataset which is worse than randomising the parameters therefore 77% would be considered fine-tuned and optimised and the data analysis can be seen via the percentages which have been outputted and the corresponding graph showing us what the parameters have been able to predict.

Unmet criteria

Unfortunately, during the course of this project not every feature from the success criteria has been met even though the desired product that has been created fits all the functionality features and passed almost all the tests first time.

In terms of point 6 and 7. 6 has not been completed and 7 has only been partially completed. This is because these pages proved to be very hard to meet within the deadline of the project and therefore the development of the project was not complete, however I felt that this section had to be reduced. This is because point 6 required me to build a completely new web page which I was not able to do to a high enough standard within this time period therefore this point was unfortunately not complete, and I failed this point

In terms of point 7, I managed to deal with Melanoma support but an ideal product ready to be presented would be able to have a call GP button and a Q&A section which was not completed because it was not deemed completely essential by the stakeholders.

Usability Summary

Usability Features

This would be to give the users an immediate understanding of how the system works and make sure that they fully understand the skin cancer detection service that we are providing. This will help to fulfil criteria 4 and 5 which state that the home page of the website should be simple but also explain what we do. Therefore, this section is essential!

Additionally, I have used a font called Segoe. As for the titles of the main sections, this helps to keep the text light due to its open and friendly type face. This helps to keep the website friendly which is very important because skin cancer is a serious issue, and it is important to attempt to calm the users down subconsciously. Also, the font has lots of weights which helps to establish a good visual hierarchy.

This visual hierarchy has been established in the hero section mainly. We can see this where the two action buttons are, these are ‘Create an account’ and ‘Try anonymously.’ Both are using the same font but the create an account is the primary call to action so has been given a bolder weighting as well as given the primary colour for the website (blue) as the background for the website which helps the main call bright and stand out on the website. This is extremely important, and it immediately incorporates the anonymous feature into the website which is on the success criteria and something that was unanimously want by all the people who were interviewed from the doctors to the potential users of the website.

The first thing that we can see here is that the background colour has been changed. This has gone from a grey into an off-white background. This will help to shift the focus of the user onto the centre of the page where all the main features of the website are. This will help to keep the users engaging with the website and work to emphasise the main points. This has been done to fulfil the criteria of keeping the website easy to use and very light on the user.

Unmet usability features

Some of the features from the design section were not possible due to the time limit on this project. This is because some features such as building a gradient blue background with circular orbits. This is not well documented on the internet and to get the specific conditions that work with the other features within the page.

Some of the positioning of words within the page are not perfect due to the fact that the computer is not fully useable within different sources of media and ideally in the future I could make the web page with a more adaptive design which will allow it to be used on mobiles however, this usability feature was not prioritised due to the fact that the target population was very aged and above 60 therefore the use of smaller technologies such as mobile phones and tablets.

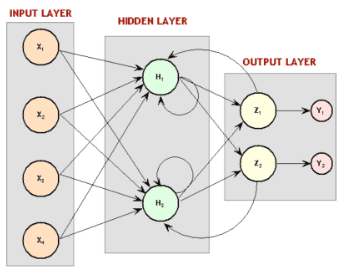
Maintenance Summary

Maintenance limitations

In terms of maintenance, at the moment the app is located within my PC and is run using local host. However, for further development of this project, I should host it on a Heroku server which will allow it to be run online and on the cloud which will allow it to be accessed anywhere and easily tested on other devices.

This process of committing my code onto Heroku would require me to change the hierarchy of my project slightly but because most of the code written has been modular it should not require too much change in order to commit to a higher database. This process is not too extensive

Additionally with maintenance there would be a problem if too many sessions are created that the system could crash therefore, I would need to upgrade my flask application to Django to make sure that does not occur or create a user login system to make sure that there are no other issues with a DDOS attack or others of similar sort.



Finally, for the maintenance of the product I should also make sure to run a constant update cycle within my CNN. This works using a continuous feedback loop or retraining the CNN at set intervals in order to take into account real-time live user feedback in order to make the product able to adapt to any new types of cancer without it being explicitly coded.

The limitations that this brings is the data protection act because the user would have to be asked before saving their image within the database/ inside the csv file. Even though the actual image would be encoded permissions would still have to sorted and a cookie-based system would have to be added to the website in order to implement feedback therefore this would be extremely hard to implement however, could be done in the future to improve the product further.

Potential limitations and changes

In terms of limitations of the solution, the solution works well to function, however it is not accurate enough to be commercially viable or compete against some other companies which have been discussed within the project. Additionally, I will probably need to change my CNN structure to deal with more data therefore should use a TensorFlow backend or maybe a keras back.

In the future with a larger dataset of items which I could take my own pictures from different perspectives and have skin from different nationalities which would allow the CNN to be trained more accurately. However, this would increase the need of a faster backend to allow the training phase to be sped up because at the moment it takes around 3 hours to train 11000 images but when the data is scaled up to around 50000 for a working environment then it will slow down the system a lot which increases the need for TensorFlow to allow TPU acceleration within the system which will make use of more pipelining and will allow the code to run quicker.

The TPU acceleration would work similar to how a GPU can optimise acceleration of video development, the actual image matrices will be split up into individual subsections that are not reliant on each other, meaning that these processes can be overlapped, if I optimised this change it would give me a lot more freedom within the implementation of the CNN and therefore help me to make a more professional project, however within the success criteria of this project was to make my own CNN without frameworks therefore this optimisation would not have been possible without use of low level languages to optimise my project by reconfiguring the python interpreter, which was beyond the scope of this project and would have required a lot more development stages and a lot of learning therefore was not chosen.

Conclusion

Overall, my project has been a remarkable success and I believe has I have done a very good job with the development and creation of the project as well as the documentation. Although some success criteria points are yet to be met, they were not essential for the product to be complete and therefore since compromises have been met with the stakeholders, this section has been a success.

Additionally, I have also learnt a lot during the process of this project such as how to develop my project whilst utilising a spiral development structure which allowed me to get constant feedback correctly iteratively from the users.

I have also computational methods such as back tracking for my neural network to sift through large data which has been an enjoyable prospect. The hardest part of this project was building the usability of the website which I struggled with a lot; however, it was a great learning curve from which I have gained a lot of knowledge that I will use in the future and during my Computer Science degree!

Code Appendix

```
# -*- coding: utf-8 -*-
"""CNN.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1vbkanyorbDrwJ-Yu9euIBkG5oG4KLDuC
"""

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
from google.colab import files,output
from tqdm import tqdm
import matplotlib.pyplot as plt

from random import randint
from cv2 import warpAffine, getRotationMatrix2D, resize

# %ls



# Commented out IPython magic to ensure Python compatibility.
```

```
! pip install -q kaggle
files.upload()


! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets list


import kaggle


! kaggle datasets download -d ramstar3000/ham10000
! mkdir Images
! unzip ham10000.zip -d Images
output.clear()


# %ls


def rotation(image): # Image array


  image = (image.astype(np.uint8)).reshape(28,28,3)


  i = randint(0,15)
  height, width = image.shape[:2]
  center = (28/2, 28/2)
  # print(center)
  #print(image.shape)
  image = warpAffine(src=image, M=getRotationMatrix2D(center=center, angle=i,
scale=1) , dsize=(width, height))
```

```
  newImage = resize(image, (28, 28))

  image = (newImage.astype(np.float64)).reshape(2352,)

  return image

def stratify_img_matrix():
    csv = pd.read_csv('Images/out.csv')
#/content/Images/hmnist_28_28_RGB.csv

    _ = np.array(pd.DataFrame(csv))

    print(_.shape)

    for row in _:
      #print(row[-1])
      #print(1)
      if row[-1] != 4 and row[-1] != 6 and row[-1] != 2:
        label = row[-1]
        new = rotation(row[:-1])
        new = np.append(new,int(label))

        #print(new.shape)
        assert new[-1] == label

        #print(_.shape)
        #print(new.shape)

        _ = np.vstack([_, new])
        #_ = np.append(_,new,axis=0)


    print(_.shape)

    csv = pd.DataFrame(_)

    column_names = list(range(2352))
    column_names.append('label')
    csv.columns = column_names



    from pathlib import Path
    filepath = Path('Images/out2.csv')
    #filepath.parent.mkdir(parents=True, exist_ok=True)
    csv.to_csv(filepath, header=False, index=False)



#stratify_img_matrix()

def get_img_matrix():

    csv = pd.read_csv('Images/out2.csv')
#/content/Images/hmnist_28_28_RGB.csv

    #print(csv)
    #csv = pd.DataFrame(_)
    column_names = list(range(1,2353))
    column_names.append('label')
    csv.columns = column_names



    y = csv['label']

    #csv['label'].value_counts().plot(kind='bar')

    df = pd.DataFrame(y)


    X = csv.drop(columns = ['label'])
    X = (np.array(X).reshape(-1,28*28*3)) # 3d as 3 channels, -1 refers to orig
of 10005


    #Now do rotating nonsesne


    return X[:-1],y[:-1]


#g = get_img_matrix()

from sklearn.model_selection import train_test_split
def split(X,y):
  return train_test_split(
      X, y, test_size=0.2, random_state=2)   #Start with random split, should do
stratified?


def nanargmax(arr):
    idx = np.nanargmax(arr)
    idxs = np.unravel_index(idx, arr.shape) #Get index as multi dimension
    return idxs


def init_filter(shape):
    stddev = 1 / np.sqrt(np.prod(shape))   # STDDEV used to give rough scale to
values
    return np.random.normal(loc=0, scale=stddev, size=shape)


def init_weights(shape):
    return np.random.normal(size=shape) * 0.01


def relu(array):
    array[array<=0] = 0
    return array


def convolution(image, filt, bias, s=1):

    (n_f, n_c_f, f, _) = filt.shape # filter dimensions
    n_c, in_dim, _ = image.shape # image dimensions

    out_dim = int((in_dim - f)/s)+1 # calculate output dimensions some mad
formula ting
```

```
    # Dimensions of filter must match channels of input image

    out = np.zeros((n_f,out_dim,out_dim))   #If 8 filters, 8 dimensions(1) provided

    # convolve the filter over every part of the image, adding the bias at each
step.
    #Need to look over and adjust
    for curr_f in range(n_f):
        curr_y = out_y = 0
        while curr_y + f <= in_dim:
            curr_x = out_x = 0
            while curr_x + f <= in_dim:
                #print(curr_f)
                out[curr_f, out_y, out_x] = np.sum(filt[curr_f] *
image[:,curr_y:curr_y+f, curr_x:curr_x+f]) + bias[curr_f]
                curr_x += s #Slide across
                out_x += 1  #Move into next location for output
            curr_y += s  #Slide down
            out_y += 1   #Move 1

    return out



def convolutionBackward(dconv_prev, conv_in, filt, s): #Analyse

    #Backpropagation through a convolutional layer.

    (n_f, n_c, f, _) = filt.shape
    (_a, orig_dim, _b) = conv_in.shape
    #assert _ == _b

    ## initialize derivatives
    dout = np.zeros(conv_in.shape)
    dfilt = np.zeros(filt.shape)
    dbias = np.zeros((n_f,1))
    for curr_f in range(n_f):
        # loop through all filters
        curr_y = out_y = 0
        while curr_y + f <= orig_dim:
            curr_x = out_x = 0
            while curr_x + f <= orig_dim:
                # loss gradient of filter (used to update the filter)
                dfilt[curr_f] += dconv_prev[curr_f, out_y, out_x] * conv_in[:,
curr_y:curr_y+f, curr_x:curr_x+f]
                # loss gradient of the input to the convolution operation (conv1 in
the case of this network)
                dout[:, curr_y:curr_y+f, curr_x:curr_x+f] += dconv_prev[curr_f,
out_y, out_x] * filt[curr_f]
                curr_x += s
                out_x += 1
            curr_y += s
            out_y += 1
        # loss gradient of the bias
        dbias[curr_f] = np.sum(dconv_prev[curr_f])

    return dout, dfilt, dbias
```

```
def maxpool(image, f=2, s=2):

    #Downsample image using size of f and stride of s

    try:
      n_c, h_prev, w_prev = image.shape #Get old/current dimensions
    except:
      h_prev, w_prev = image.shape
      n_c = 1

    h = ((h_prev - f)/s)+1  #Calculate new dimensions of img
    w = ((w_prev - f)/s)+1  #Int been moved so beware int((h_prev - f)/s)+1

    #print(image)
    #print(image[0, 0:2, 4:6])  #[channel num, x:x+2 , y:y+2], is a 2x2 grid



    output = np.zeros((n_c, int(h), int(w)))  #Make empty array to be filled

    for i in range(n_c): #Once for each channel
        # slide maxpool window over each part of the image and assign the max
value at each step to the output
        curr_y = out_y = 0     #Initialised
        while curr_y + f <= h_prev:
            curr_x = out_x = 0
            while curr_x + f <= w_prev:  #Only runs till reaches end of line
                output[i, out_y, out_x] = np.max(image[i, curr_y:curr_y+f,
curr_x:curr_x+f])
                curr_x += s  #Slide across
                out_x += 1    #Next index in final array
            curr_y += s     #Slide down
            out_y += 1      #Move into next index downwards

    return output

def maxpoolBackward(dpool, orig, f, s): #Analyse

    #Backpropagation through a maxpooling
    # Gradients are passed through the indices of greatest value in the original
maxpooling during the forward step.

    (n_c, orig_dim, _) = orig.shape
    dout = np.zeros(orig.shape)

    #print(orig)

    for c in range(n_c):    #Once for each channel
        curr_y = out_y = 0
        while curr_y + f <= orig_dim:
            curr_x = out_x = 0
            while curr_x + f <= orig_dim:
                # obtain index of largest value in input for current window

                (a, b) = nanargmax(orig[c, curr_y:curr_y+f, curr_x:curr_x+f])
                dout[c, curr_y+a, curr_x+b] = dpool[c, out_y, out_x]

                curr_x += s
                out_x += 1
            curr_y += s
            out_y += 1

    #print(dout)



    return dout


def softmax(X):
    out = np.exp(X)
    return out/np.sum(out)


def categoricalCrossEntropy(probs, label):
    return -np.sum(label * np.log(probs))


def conv(image, label, params, conv_s, pool_f, pool_s):  #1#1#1# Analyse
backward, also may be broken


    [f1, f2, w3, w4, b1, b2, b3, b4] = params


    ###############     Forward       ################


    conv1 = convolution(image, f1, b1, conv_s) # convolution
    conv1 = relu(conv1) # pass through ReLU non-linearity


    conv2 = convolution(conv1, f2, b2, conv_s) # second convolution
    conv2 = relu(conv2) # pass through ReLU non-linearity
    #print(conv2.shape)
    pooled = maxpool(conv2, pool_f, pool_s) # maxpooling


    (nf2, dim2, _) = pooled.shape

    fc = pooled.reshape((nf2 * dim2 * dim2, 1)) # flatten

    #print(fc.shape)


    z = w3.dot(fc) + b3 # first dense layer

    z = relu(z) # pass through ReLU non-linearity


    out = w4.dot(z) + b4 # second dense layer


    probs = softmax(out) # predict class probabilities with the softmax activation
function

    #print(probs)


    #Calculate Loss (Loss function is for batch, cost for multiple)

    loss = categoricalCrossEntropy(probs, label) # categorical cross-entropy loss


    ##############      Backward      ################


    dout = probs - label # (7,)   derivative of loss w.r.t. final dense layer output

    dw4 = dout.dot(z.T) # loss gradient of final dense layer weights

    db4 = np.sum(dout, axis = 1).reshape(b4.shape) # loss gradient of final dense
layer biases


    dz = w4.T.dot(dout) # loss gradient of first dense layer outputs

    dz[z<=0] = 0


    dw3 = dz.dot(fc.T)
    db3 = np.sum(dz, axis = 1).reshape(b3.shape)


    dfc = w3.T.dot(dz) # loss gradients of fully-connected layer (pooling layer)

    dpool = dfc.reshape(pooled.shape) # reshape fully connected into dimensions of
pooling layer


    dconv2 = maxpoolBackward(dpool, conv2, pool_f, pool_s) # backprop through
the max-pooling layer(only neurons with highest activation in window get updated)

    dconv2[conv2<=0] = 0


    dconv1, df2, db2 = convolutionBackward(dconv2, conv1, f2, conv_s) #
backpropagate previous gradient through second convolutional layer.

    dconv1[conv1<=0] = 0


    dimage, df1, db1 = convolutionBackward(dconv1, image, f1, conv_s) #
backpropagate previous gradient through first convolutional layer.


    grads = [df1, df2, dw3, dw4, db1, db2, db3, db4]


    return grads, loss


def adamGD(batch, num_classes, lr, dim, n_c, beta1, beta2, params, cost):
#1#1#1# Research


    #This is an adaptive descent mode


    # update the parameters through Adam gradient descnet.


    [f1, f2, w3, w4, b1, b2, b3, b4] = params  #Unroll params
```

```
X = batch[:,0:-1] # get inputs

X = X.reshape(len(batch), n_c, dim, dim)   #Get input to right format

Y = batch[:,-1] # get labels


cost_ = 0

batch_size = len(batch)   #Should be 32 (May change)


# initialize gradients and momentum,RMS params

df1 = np.zeros(f1.shape)

df2 = np.zeros(f2.shape)

dw3 = np.zeros(w3.shape)

dw4 = np.zeros(w4.shape)

db1 = np.zeros(b1.shape)

db2 = np.zeros(b2.shape)

db3 = np.zeros(b3.shape)

db4 = np.zeros(b4.shape)


v1 = np.zeros(f1.shape)

v2 = np.zeros(f2.shape)

v3 = np.zeros(w3.shape)

v4 = np.zeros(w4.shape)

bv1 = np.zeros(b1.shape)

bv2 = np.zeros(b2.shape)

bv3 = np.zeros(b3.shape)

bv4 = np.zeros(b4.shape)


s1 = np.zeros(f1.shape)

s2 = np.zeros(f2.shape)
```

```
    s3 = np.zeros(w3.shape)

    s4 = np.zeros(w4.shape)

    bs1 = np.zeros(b1.shape)

    bs2 = np.zeros(b2.shape)

    bs3 = np.zeros(b3.shape)

    bs4 = np.zeros(b4.shape)


    for i in range(batch_size):


        x = X[i]

        y = np.eye(num_classes)[int(Y[i])].reshape(num_classes, 1) # convert label
to one-hot


        # Collect Gradients for training example

        grads, loss = conv(x, y, params, 1, 2, 2)

        [df1_, df2_, dw3_, dw4_, db1_, db2_, db3_, db4_] = grads



        #Update the change in values

        df1+=df1_

        db1+=db1_

        df2+=df2_

        db2+=db2_

        dw3+=dw3_

        db3+=db3_

        dw4+=dw4_

        db4+=db4_


        cost_+= loss
    # Parameter Update

    v1 = beta1*v1 + (1-beta1)*df1/batch_size # momentum update
    s1 = beta2*s1 + (1-beta2)*(df1/batch_size)**2 # RMSProp update
    f1 -= lr * v1/np.sqrt(s1+1e-7) # combine momentum and RMSProp to perform
update with Adam

    bv1 = beta1*bv1 + (1-beta1)*db1/batch_size
    bs1 = beta2*bs1 + (1-beta2)*(db1/batch_size)**2
    b1 -= lr * bv1/np.sqrt(bs1+1e-7)

    v2 = beta1*v2 + (1-beta1)*df2/batch_size
    s2 = beta2*s2 + (1-beta2)*(df2/batch_size)**2
    f2 -= lr * v2/np.sqrt(s2+1e-7)

    bv2 = beta1*bv2 + (1-beta1) * db2/batch_size
    bs2 = beta2*bs2 + (1-beta2)*(db2/batch_size)**2
    b2 -= lr * bv2/np.sqrt(bs2+1e-7)

    v3 = beta1*v3 + (1-beta1) * dw3/batch_size
    s3 = beta2*s3 + (1-beta2)*(dw3/batch_size)**2
    w3 -= lr * v3/np.sqrt(s3+1e-7)

    bv3 = beta1*bv3 + (1-beta1) * db3/batch_size
    bs3 = beta2*bs3 + (1-beta2)*(db3/batch_size)**2
    b3 -= lr * bv3/np.sqrt(bs3+1e-7)

    v4 = beta1*v4 + (1-beta1) * dw4/batch_size
    s4 = beta2*s4 + (1-beta2)*(dw4/batch_size)**2
    w4 -= lr * v4 / np.sqrt(s4+1e-7)


    bv4 = beta1*bv4 + (1-beta1)*db4/batch_size
    bs4 = beta2*bs4 + (1-beta2)*(db4/batch_size)**2
    b4 -= lr * bv4 / np.sqrt(bs4+1e-7)



    cost_ = cost_/batch_size  # Divide down as overall cost
    cost.append(cost_)        # Put onto big list


    params = [f1, f2, w3, w4, b1, b2, b3, b4]  #Roll into 1 list


    return params, cost

def train(num_classes = 7, lr = 0.0125, beta1 = 0.95, beta2 = 0.99, img_dim =
28, img_channels = 3, f = 5, num_filt1 = 8, num_filt2 = 8, batch_size = 16,
num_epochs = 4):


    # get data


    X,y = get_img_matrix()
    #print(X,y)


    X = np.array(X,dtype=float)



    #print(X,y)
    X-= int(np.mean(X))

    X/= int(np.std(X))


    X_train, X_test, y_train, y_test = split(X,np.array(y))


    train_data = np.hstack((X_train,y_train.reshape(-1,1)))


    np.random.shuffle(train_data)



    #f is size of the window that is scanned, so in this case 5x5 window

    #Each layer can have a varying number of filters so far is 8, change google
says 32

    #3 image channels as red green and blue, have data for 1 but oh well

    #For the weights, the 128 can be changed but output layer must have 7 and
input 800 from the pooling


#####################################################
##

    ####### Initializing all the parameters ##########

#####################################################
##



    #Initialise the filters and weights

    f1, f2 =  init_filter((num_filt1 ,img_channels,f,f)), init_filter((num_filt2
,num_filt1,f,f))

    w3, w4 = init_filter((128,800)), init_filter((7, 128))
    #Initialise the bias terms for each stage
    b1 = np.zeros((f1.shape[0],1))
    b2 = np.zeros((f2.shape[0],1))
    b3 = np.zeros((w3.shape[0],1))
    b4 = np.zeros((w4.shape[0],1))

    params = [f1, f2, w3, w4, b1, b2, b3, b4] #Roll

    cost = [] # type: List[float]

    #LR = learning rate, can be increased at risk
    #Batch size is number of items taken every time

    print("Learning Rate:"+str(lr)+", Batch Size:"+str(batch_size))

    for epoch in range(num_epochs):
        np.random.shuffle(train_data)
        batches = [train_data[k:k + batch_size] for k in range(0,
train_data.shape[0], batch_size)]

        load_bar = tqdm(batches)
        load_bar.set_description("Cost: 4" )  # Cost is appended to array each
time

        for _,batch in enumerate(load_bar):
            params, cost = adamGD(batch, num_classes, lr, img_dim, img_channels,
beta1, beta2, params, cost)
```

```
            load_bar.set_description(f"Cost: {(cost[-1]) :.3f}" )

    return cost,params
cost,params = train()

import pickle



with open('objs.pkl', 'wb') as f:  # Python 3: open(..., 'wb')
    pickle.dump(params, f)

def predict(image, f1, f2, w3, w4, b1, b2, b3, b4, conv_s=1, pool_f = 2, pool_s
= 2):

  #Run an image through the CNN using the parameters

  conv1 = convolution(image, f1, b1)
  conv1 = relu(conv1)               # Relu check

  conv2 = convolution(conv1, f2, b2)
  conv2 = relu(conv2)

  pooled = maxpool(conv2)
  (nf2, dim2, _) = pooled.shape
  fc = pooled.reshape((nf2 * dim2 * dim2, 1))  # Flattened

  # Now do predicting with ANN

  z = w3.dot(fc) + b3 # first dense layer
  z = relu(z) # [z<=0] = 0 # pass through ReLU non-linearity

  out = w4.dot(z) + b4 # second dense layer
  probs = softmax(out) # predict class probabilities with the softmax activation
function
  #print(np.argmax(probs))
  return np.argmax(probs), np.max(probs)

plt.plot(cost,'r') # Plot the cost over each iteration, maybe consider moving
average
plt.xlabel = 'No. Iterations'
plt.show()

[f1, f2, w3, w4, b1, b2, b3, b4] = params        # Unload parameters

# get data again, duplicated
X,y = get_img_matrix()
X = np.array(X,dtype=float)  # Needs to be float to do division

X-= int(np.mean(X))   #Making into normal, close to 0
X/= int(np.std(X))

X_train, X_test, y_train, y_test = split(X,np.array(y))
test_data = np.hstack((X_test,y_test.reshape(-1,1)))  #combined

X = test_data[:,0:-1]
X = X.reshape(-1, 3, 28, 28)    #Fit into correct format for filters

y = test_data[:,-1]               #Unflip
```

```
print(y==y_test)

ys = [1 if each == 2 or each == 4 or each == 3 else 0 for each in y]

print(ys)

print(y)

corr = 0

count = [0 for i in range(7)]

correct = [0 for i in range(7)]

s,f = 0,0

#print()

print("Computing accuracy over test set:")


t = tqdm(range(len(X)), leave=True)

#t = range(len(X))

for i in t:

    x = X[i]

    pred, prob = predict(x, f1, f2, w3, w4, b1, b2, b3, b4)

    #print(pred,y[i])

    if (pred==2 or pred==5 or pred==3) and (ys[i]==2 or ys[i]==5 or ys[i]==3):

      s +=1

    else:

      if  (not (pred==2 or pred==5 or pred==3)) and (not (ys[i]==2 or ys[i]==5 or
ys[i]==3)):

        s +=1


    count[int(y[i])]+=1

    if pred==y[i]:

        corr+=1

        correct[pred]+=1
```

```
#print("Overall Accuracy: %.3f" % (float(corr/len(test_data)*100)))
#print("Overall Accuracy 2: %.3f" % (float(s/len(test_data)*100)))
x = np.arange(7)
recall = [x/y for x,y in zip(correct, count)]
plt.xlabel = 'Classes'
plt.ylabel = 'Recall'
plt.title = "Recall on Test Set"
plt.bar(x,recall)
plt.show()

#Need to split 7 cats into 2....

print("Overall Accuracy: %.3f" % (float(corr/len(test_data)*100)))
print("Overall Accuracy 2: %.3f" % (float(s/len(test_data)*100)))

[f1, f2, w3, w4, b1, b2, b3, b4] = params

#print(repr(params[0]))

with open('objs.pkl','rb') as f:  # Python 3: open(..., 'rb')
    params2 = pickle.load(f)

print(len(params2))
print(len(params))

df = pd.read_csv('Images/out2.csv')
```

```
column_names = list(range(1,2353))
column_names.append('label')
df.columns = column_names

print(df['label'][0:10])

X = df.drop(['label'], axis=1)




X1 = np.array(X, dtype='uint8')
print(X1[0].shape)
image = X1[0].reshape(28,28,3)

print(image)
print(image.shape)
#print(X1[0])
#print(type(image[0][0][0]))
print('====================================')
new_image = np.array(rotation(image),dtype='uint8').reshape(28,28,3)
#print(';;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;')

plt.imshow(new_image)
plt.savefig('out.png', bbox_inches='tight', pad_inches=0)

#x = (np.array([[3,1,7,2],[5,1,0,9],[8,2,4,9],[4,3,1,1]]))
```

```
#x = np.zeros((2,3,4))

test_data = np.array([[[2,17], [45, 78]], [[88, 92], [60, 76]],[[76,33],[20,18]]])

print(test_data)

print('')

maxpool(test_data)


from typing import KeysView

from selenium import webdriver

from selenium.webdriver.common.keys import Keys


    def Load():

        driver = self.driver

        driver.get("https://localhost:5000/")

        driver.able('JS',False)

        Keys.send_keys(Keys.RETURN)

        return(driver)


import pickle



with open('objs.pkl', 'wb') as f:  # Python 3: open(..., 'wb')

    pickle.dump(params, f)
```

```
# -*- coding: utf-8 -*-
"""HAM10000.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1c6FlVt0bM7Hoi1rYilI-Dx-
7Wd4YZ5Va

#Presets
"""

import pandas as pd
from google.colab import files
uploaded = files.upload()

address = 'HAM10000_metadata.csv'
df = pd.read_csv(address)
df

df.info()        #Find general info
df.isna().sum() #find how many unknowns
df['dx'].value_counts()        #Stratification stats
df['dx_type'].value_counts()

"""dx is the diagnosis and type is how the diagnosis was made
https://arxiv.org/abs/1803.10417"""

dx_dict = {
    'nv': 'Melanocytic nevi',

    'mel': 'Melanoma',

    'bkl': 'Benign keratosis-like lesions ',

    'bcc': 'Basal cell carcinoma',

    'akiec': 'Actinic keratoses',

    'vasc': 'Vascular lesions',

    'df': 'Dermatofibroma'

}    #Types of lesion for mapping as words ineffiicent storage


df['diagnosis'] = df['dx'].map(dx_dict.get)  #Swap

df.sex.value_counts()   #More data points

df.localization.value_counts()

df.age.value_counts()


import numpy as np


value = df.age.mean()

print(value)

df.age = df.age.fillna(value)

df.isna().sum()


"""Manually imputed the mean into the age to remove all the Nan from the
database while keeping the same mean so the distribution will not change


#**DATA ANALYSIS**
"""


import matplotlib.pyplot as plt

import seaborn as sns
```

```
df['dx'].value_counts().plot(kind='bar')


sns.scatterplot('age','dx',data=df)


"""#**Preprocessing**


Need to convert words to numbers, need to do it manually
"""


from sklearn import preprocessing




#Need to label encode and one hot encode/ normalise
label_encoder = preprocessing.LabelEncoder()
df1 = df.copy()
#df1.lesion_id


lesion_id_cat = label_encoder.fit_transform(df1['lesion_id'])
lesion_id_cat = pd.DataFrame({'lesion_id_cat': lesion_id_cat})


image_id_cat = label_encoder.fit_transform(df1['image_id'])
image_id_cat = pd.DataFrame({'image_id_cat': image_id_cat})


dx_cat = label_encoder.fit_transform(df1['dx'])
dx_cat = pd.DataFrame({'dx_cat': dx_cat})


dx_type_cat = label_encoder.fit_transform(df1['dx_type'])
dx_type_cat = pd.DataFrame({'dx_type_cat': dx_type_cat})


sex_cat = label_encoder.fit_transform(df1['sex'])

sex_cat = pd.DataFrame({'sex_cat': sex_cat})


localization_cat = label_encoder.fit_transform(df1['localization'])

localization_cat = pd.DataFrame({'localization_cat': localization_cat})


diagnosis_cat = label_encoder.fit_transform(df1['dx'])

diagnosis_cat = pd.DataFrame({'diagnosis_cat': diagnosis_cat})


df1.lesion_id = lesion_id_cat

df1.image_id = image_id_cat

df1.dx = dx_cat

df1.dx_type = dx_type_cat

df1.sex = sex_cat

df1.localization = localization_cat

df1.dx = diagnosis_cat



convertor = {

    'bkl':2,

    'nv':5,

    'akiec':0,

    'mel':4,

    'bcc':1,

    'vasc':6,

    'df':3,
}
df1

print(np.array(df1['dx'])[64])

"""#**Normalisation**"""

from sklearn.preprocessing import StandardScaler

#Using modules

scaled_features = df1.copy()

col_names = ['lesion_id', 'image_id' , 'dx', 'dx_type', 'age', 'sex', 'localization']
#Removed diagnosis
features = scaled_features[col_names]
scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features[col_names] =  features

X = scaled_features.drop(columns=['dx','lesion_id','image_id'],axis=1)
X.insert(0,1.0,1.0)
X
y = [1 if each == 'bkl' or each == 'nv' or each == 'df' else 0 for each in df.dx]
# or y = df.dx

print(scaled_features)
```

```
# Manual


#scaled = df1.copy()
#Then use maths operations use folder where it is written out
#Will experiment with standardisation vs normalisation


"""#**Linear Regression** """


from sklearn.model_selection import train_test_split


y = [1 if each == 'bkl' or each == 'nv' or each == 'df' else 0 for each in df.dx]
print(y)
print(X)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1)


y.count(0) # == 2096
y.count(1) # == 7919


print(X_train)
print(X_train.shape)


# Normal equation ==


def ComputeCost(X,y,theta):



  m = len(y)
  predictions = X.dot(theta)


  Error = (predictions-y)   #Make sure the right type of squaring should be
dot...#


  SError = (Error)**2



  J = (1/2*m) * sum(SError)
  return J






#ComputeCost(X_train,y_train,theta)


def Regression(X,y):


  theta = np.array([0.79,-0.1,-0.01,-0.14,-0.111]).transpose()
  alpha = 0.05
  m = len(y)
  Js = []


  for __ in range(500):


    predictions = X_train.dot(theta)-y_train
```

```
        #print(predictions)
        #print(alpha/m)
        #print(alpha/m * X_train)

        #print(X_train.shape)
        #print(predictions.shape)
        some = (X_train.transpose().dot(predictions))
        #print(some_.shape)

        change = ((alpha/m) * some)
        #print(change)
        theta = theta - change
        #print(theta)

        J = ComputeCost(X_train,y_train,theta)
        Js.append(J)

    plt.plot(Js)
    print(Js[-1])

    return (theta)



#25345962.0
#25345962.


    #Change theta appropriately
```

```
#print(Regression(X_train,y_train))

def Predict(x,y,theta):

  values  = round(x.dot(theta))

  fp = 0
  p  = 0
  fn = 0
  n  = 0
  for a,b in zip(y,values):

    if a == 1:
      if b == 1:
        p += 1
      if b == 0:
        fp +=1

    if a == 0:
      if b == 0:
        n+=1
      if b == 1:
        fn += 1

  print(p , n)
  print(fp , fn)
  print(f'Overall acurracy is {(p+n) / (fp+fn+p+n)}')
  #Confusion matrix


#Normal Equation
print(X_train.T)
Xs = ((X_train.T)).dot(X_train)
thet = np.linalg.inv(Xs).dot(X_train.transpose())
theta  = thet.dot(y_train)

print(theta)



(ComputeCost(X_train,y_train,theta))


print(theta)


Predict(X_test,y_test,theta)

"""Linear regression bad as takes anomolies therefore is not a great idea!

#Neural Net
"""


def Sigmoid(z):
  return 1/(1+np.exp(-z))


def SigGrad(z):
  return np.multiply(Sigmoid(z),(1-Sigmoid(z)))


m = X_train.shape[0]


Theta1 = np.matrix([[1.1,3.2,3,2,5],[1,2,3,4,5],[1,2,3,4,5],[1,2,3,4,5]])
#Needs to be bigger atm only 1 hidden layer

Theta2 = np.matrix([[1.1],[2],[1],[1]])

print(Theta1.shape)

a1 = X_train.copy()


def NNCostFunc():#weights, input_layer_size, hidden_layer_size, num_labels, X,
y, lamda):

  global Theta1, Theta2, a1

  #Choosing not to unroll parameters into weight matrices and just use weights


  #2 layers, so 2 weight matrices


  z2 = Theta1.dot(a1.T)

  a2 = Sigmoid(z2)

  layer1 = Sigmoid(np.dot(a1,Theta1.transpose()))

  #print(layer1)


  output = Sigmoid(np.dot(layer1,Theta2))

  #print(output.shape)

  #print('Cost Is done')





  #print(y_train)
```

```
Delta3 = (y_train-output.T).T # = (8012,1)
Del21 = (Delta3 * Theta2.T)
#print(Del21.shape)
Del22 = SigGrad(z2)

#add = (np.ones(Del22.shape[1]))
#print(add.shape)
#print(Del22.T.shape)
#Del22 = np.column_stack( (add, Del22.T))

#print(Del22.shape)
#print(Del22) # =2,8013

Delta2 = np.multiply(Del22,Del21.T)

Theta1_grad = (1/m) * (Delta2.dot(a1))
#print(Theta1_grad)

Theta2_grad = (1/m) * (a2.dot(Delta3))
#print(Theta2_grad)

Theta1 += Theta1_grad
Theta2 += Theta2_grad
#print(output)
return Theta1,Theta2,output
```

```
for _ in range(5000):
  Theta1,Theta2,output = NNCostFunc()


def predict():


  z2 = Theta1.dot(X_test.T)
  a2 = Sigmoid(z2)
  layer1 = Sigmoid(np.dot(X_test,Theta1.transpose()))
  #print(layer1)


  output = Sigmoid(np.dot(layer1,Theta2))


  success,fail = 0,0
  p,n,fp,fn =0,0,0,0
  for a,b in zip(y_test,output.tolist()):
    b = round(b[0])
    if a == b:
      success +=1
    else:
      fail+=1
    if a == 1:
      if b == 1:
        p += 1
      if b == 0:
        fp +=1


    if a == 0:
      if b == 0:
        n+=1
      if b == 1:
        fn += 1

  print(p , n)
  print(fp , fn)
    #if b[0] < 0.7:
    #  print(a,b)
  print( f'Overall Success is {success/(success+fail)} ')
predict()

"""#Logistic Regression 1 vs All

We will take a 1 vs many approach, so will need to do a seperate regression for
each of the different values for *y*
"""

y = df1['dx']
all_theta = np.empty((6,5))
X_train, X_test, y_train, y_test = train_test_split(
      X, y, test_size=0.2, random_state=1)
s = y_test+0
for x in range(0,6):
  y_now = [1 if each==x else 0 for each in y]

  X_train, X_test, y_train, y_test = train_test_split(
      X, y_now, test_size=0.2, random_state=1)
    #print(y_now.count(1))
  theta = Regression(X_train,y_train)
  #print(theta)
  all_theta[x] = theta
  pred = X_train.dot(theta)

print(all_theta)



OneVsAllPredict(all_theta,X_test)


def OneVsAllPredict(all_theta,X):


  #Assume X has 0s
  probs = X.dot(all_theta.transpose())
  #print(probs)
  pred = probs.idxmax(1)
  #print(pred)


  print(s[70:80])
  success = 0
  fail = 0
  for a,b in zip(s,pred):
    if a==b:
      success +=1
    else:
      fail+=1
  print(f'Overall Accuracy of {success/(success+fail)}')   #0.6689
  s_1var = [1 if each == 2 or each == 5 or each == 3 else 0 for each in s]
  pred_1 = [1 if each == 2 or each == 5 or each == 3 else 0 for each in s]


  for a,b in zip(s_1var,pred_1):
    if a==b:
      success +=1
    else:
      fail+=1
  print(f'Overall Accuracy 2 of {success/(success+fail)}')   #0.8344





OneVsAllPredict(all_theta,X_test)



#Overall still overpredicts on 5

"""# To do

Here, we need to do standardisation and normalisation using no modules!
If time make a svm but otherwise will be fine
"""

print(y)
```

```
# -*- coding: utf-8 -*-
"""Image_NN

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1HCuwpLHlyY3gFyXs1T8x-
tarmchCIPC_

#The image one
"""

#Preprocessing

#Need to convert to black and white
#Then use pixel values

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from google.colab import files


uploaded = files.upload()

def rgb_to_gray(img):
        grayImage = np.zeros(img.shape)
        R = np.array(img[:, :, 0])
        G = np.array(img[:, :, 1])
        B = np.array(img[:, :, 2])

        R = (R *.299)
        G = (G *.587)
        B = (B *.114)

        Avg = (R+G+B)
        grayImage = img.copy()

        for i in range(3):
           grayImage[:,:,i] = Avg

        return grayImage
address = 'ISIC_0024312.jpg'
image = mpimg.imread(address)
grayImage = rgb_to_gray(image)

plt.imshow(grayImage)
plt.show()
mpimg.imsave('out.jpg',grayImage)

# Commented out IPython magic to ensure Python compatibility.
# %ls
image = mpimg.imread('out.jpg')
plt.imshow(image)
plt.show()

files.download('out.jpg')
```

```
import numpy as np
import pickle
from PIL import Image
import pandas as pd
import matplotlib.pyplot as plt



class cnnn():
    def __init__(self):
        pass


    def nanargmax(self, arr):
        idx = np.nanargmax(arr)
        idxs = np.unravel_index(idx, arr.shape)  # Get index as multi
dimension
        return idxs

    def load(self):
        with open('params2.pkl', 'rb') as f:  # Python 3: open(..., 'rb')
            params = pickle.load(f)

            return params

    def relu(self, array):
        array[array <= 0] = 0
        return array

    def softmax(self, X):
        out = np.exp(X)
        return out / np.sum(out)

    def convolution(self, image, filt, bias, s=1):
        (n_f, n_c_f, f, _) = filt.shape  # filter dimensions
        n_c, in_dim, _ = image.shape  # image dimensions

        out_dim = int((in_dim - f) / s) + 1  # calculate output dimensions
some mad formula ting

        # Dimensions of filter must match channels of input image

        out = np.zeros((n_f, out_dim, out_dim))  # If 8 filters, 8
dimensions(1) provided

        # convolve the filter over every part of the image, adding the bias
at each step.
        # Need to look over and adjust
        for curr_f in range(n_f):
            curr_y = out_y = 0
            while curr_y + f <= in_dim:
                curr_x = out_x = 0
                while curr_x + f <= in_dim:
                    # print(curr_f)
                    out[curr_f, out_y, out_x] = np.sum(filt[curr_f] *
image[:, curr_y:curr_y + f, curr_x:curr_x + f]) + \
                                                bias[curr_f]
                    curr_x += s  # Slide across
                    out_x += 1  # Move into next location for output
                curr_y += s  # Slide down
                out_y += 1  # Move 1
        return out

    def maxpool(self, image, f=2, s=2):
        # Downsample image using size of f and stride of s

        n_c, h_prev, w_prev = image.shape  # Get old/current dimensions

        h = ((h_prev - f) / s) + 1  # Calculate new dimensions of img
        w = ((w_prev - f) / s) + 1  # Int been moved so beware int((h_prev
- f)/s)+1

        # print(image)
        # print(image[0, 0:2, 4:6])  #[channel num, x:x+2 , y:y+2], is a
2x2 grid

        output = np.zeros((n_c, int(h), int(w)))  # Make empty array to be
filled

        for i in range(n_c):  # Once for each channel
            # slide maxpool window over each part of the image and assign
the max value at each step to the output
            curr_y = out_y = 0  # Initialised
            while curr_y + f <= h_prev:
                curr_x = out_x = 0
                while curr_x + f <= w_prev:  # Only runs till reaches end
of line
                    output[i, out_y, out_x] = np.max(image[i, curr_y:curr_y
+ f, curr_x:curr_x + f])
                    curr_x += s  # Slide across
                    out_x += 1  # Next index in final array
                curr_y += s  # Slide down
                out_y += 1  # Move into next index downwards

        return output

    def predict(self, image, f1, f2, w3, w4, b1, b2, b3, b4, conv_s=1,
pool_f=2, pool_s=2):
        # Run an image through the CNN using the parameters

        conv1 = self.convolution(image, f1, b1)
        conv1 = self.relu(conv1)  # Relu check

        conv2 = self.convolution(conv1, f2, b2)
        conv2 = self.relu(conv2)

        pooled = self.maxpool(conv2)
        (nf2, dim2, _) = pooled.shape
        fc = pooled.reshape((nf2 * dim2 * dim2, 1))  # Flattened

        # Now do predicting with ANN

        z = w3.dot(fc) + b3  # first dense layer
        z = self.relu(z)  # [z<=0] = 0 # pass through ReLU non-linearity

        out = w4.dot(z) + b4  # second dense layer
        probs = self.softmax(out)  # predict class probabilities with the
softmax activation function

        return np.argmax(probs), np.max(probs)
    def main(self, image):

        im = Image.open(image)


        im = im.resize((28,28), Image.ANTIALIAS)

        im.save(image, dpi=(600, 600))

        frame = np.asarray(im)
        frame = frame.reshape((3,28,28))  #2352
        print(frame.shape)
        print(frame)

        #naaed to process image
        [f1, f2, w3, w4, b1, b2, b3, b4] = self.load()

        x, y = self.predict(frame, f1, f2, w3, w4, b1, b2, b3, b4)

        return x, y


if __name__ == '__main__':
    X = cnnn()
    print(X.main('temp/tempISIC_0024328.jpg'))
```

```
from flask import Flask, render_template, request, session, redirect,
url_for
import os
from cnn import cnnn
import requests

# Create flask application and add encryption value
app = Flask(__name__)
app.secret_key = 'asdvsdfbstndfbzzdf'


@app.route('/')
def index():
    return render_template('home.html')


@app.route('/main', methods=['POST', 'GET'])
def Detection():
    if request.method == 'POST':
        print('Check1')

        if request.form.get('Submit') == 'submit':
            f = request.files['image']
            filename = 'temp' + f.filename
            path = os.path.join('temp', filename)
            f.save(path)  # Add to session data the filename. so can be
accessed, also delte
            print(f)
            x = cnnn()
            print('Thru')
            _ = x.main(path)  # Needs to be editted
            print(_)

            return redirect(url_for('index'))
        return render_template('Detection.html')
    return render_template('Detection.html')


if __name__ == "__main__":
    app.secret_key = 'asdvsdfbstndfbzzdf'
    app.run(debug=True)
```

```
  var loadFile = function(event) {
    var output = document.getElementById('output');
    output.style.visibility = 'visible';

    var temp  = URL.createObjectURL(event.target.files[0]);
    output.src = temp;

    console.log(output);
    console.log(temp)
    output.onload = function() {
        URL.revokeObjectURL(output.src) // Need to do after upload

        document.getElementById('txt').style.visibility = 'visible';
        document.getElementById('Submit').style.visibility = 'visible';

        change('Title_sentence','Successful upload')

        }
  };


change = function(id,word){
    var elem = document.getElementById(id);
    elem.innerHTML = word

}




function isFileImage(file) {
return file && file['type'].split('/')[0] === 'image';
}
```

```
html, body{
    background-color:#F7F7F7;
height:100%
    }

h1{
   margin-top:3vh;
   margin-bottom:0;
   text-align:center;
   font-size: 7vw;
   color:#11010D;
   font-family: 'segoe-print-regular', sans-serif;

   }

h2{

    margin-top:5vh;
    margin-bottom:0;
    text-align:left;


}

p  {
   font-size: 3.5vw;

}









p{
   font-size:2vw;
   display:inline;
}

.str{
   height:4vh;
   width:20vw;

}

.submit{

   margin-left:60%;
   margin-top:5%;
   background-color:lightseagreen;
   border:None;
   font-size:3vw

}

div{
   text-align:center
}
.form-control{
   width:60%;
   margin-left:20%;

}

.btn btn-primary{
   margin-top:10%

}
```