# Brain Metastases MRI Segmentation and Survival Analysis Pipeline using Machine Learning

ELEC 490/498

Team 3

Nicolas Campbell, 20054041, 16nbc2@queensu.ca

Simon Crête, 20047585, 16sjcc@queensu.ca

Arie Moffat, 20052122, 16anm4@queensu.ca

Andrew Simonds, 20056566, andrew.simonds@queensu.ca

Supervised by:

Ali Etemad, Ph.D.

Amber Simpson, Ph.D.

April 12th, 2021

# Executive Summary

Brain metastases diagnosis severely impacts the prognosis of cancer patients. Accurate segmentation of tumors and prediction of survival time would enhance prognosis and treatment selection. The use of machine learning utilizing magnetic resonance imaging (MRI) radiomic features has been investigated in patients with brain metastases, but these studies do not undertake a complete MRI to survival prediction pipeline, with deep learning segmentation and machine learning survival prediction. The team proposes a pipeline model utilizing deep learning and machine learning that has the potential to deliver higher accuracy prognostication in this population.

The proposed pipeline takes a brain MRI as input, segments tumors using a deep learning nnU-Net model, extracts radiomic features, uses these features in a time-dependent random survival forest machine learning model, and outputs a survival curve.

Patients diagnosed with brain metastases treated at Kingston General Hospital from 2016-2020 were included in the study. Clinicopathological variables were collected and analyzed for association with survival. Treatment MRI were pulled from the health information system, segmented by a clinical team, and analyzed using quantitative techniques. Before training, 10% of the data was split as a holdout testing set for evaluation of the pipeline's performance. A nnU-Net [1] segmentation model was trained using five-fold cross validation with an 80% training and 20% validation split. Physician-segmented image intensities were normalized based on z-score, and voxel spacing was resampled to 0.9x0.9x0.9mm. Standard radiomic features (116) were extracted using PyRadiomics. These features were grouped by their high collinearities with other features using their variance inflation factor. Features with the highest variance were selected from each group. The Random Survival Forest model [2] from the open-source PySurvival library was trained using radiomic features extracted from physician drawn segmentations and significant clinical variables to predict overall survival. The model used four-fold cross validation (CV) on the remaining data to prevent overfitting, with a 75% training and 25% validation split within each fold. The pipeline was then implemented using the trained models.

In total, 157 patients diagnosed with brain metastases (mixed histologies) were analyzed. The segmentation model obtained a mean dice score of 0.745. Sex ($p = 0.018$) and 14 radiomic features were included in the final survival model. The survival model obtained a C-index of 0.74 and IBS of 0.17 and C-index of 0.675 and IBS of 0.175, on the training and validation data, respectively. The pipeline takes as input a specified MRI scan, and outputs a survival curve plot. The pipeline obtained a holdout test set C-index of 0.68 and IBS of 0.15.

The proposed method utilized radiomic features to train a model for brain metastases segmentation and survival estimation. Our findings suggest that brain metastases radiomic features might be applicable for predicting survival in patients with brain metastases. Results showed that the model produces accurate and meaningful survival predictions. This shows potential for the creation and implementation of a clinical tool using these methods.

# Table of Contents

Executive Summary ..... i
1.0 Project Background and Goals ..... 1
1.1 Background ..... 1
1.2 Problem Definition ..... 1
1.3 Problem Statement ..... 1
1.4 Health Canada Approval for Software as a Medical Device ..... 1
1.4.1 Medical Device Definition ..... 1
1.4.2 Software as a Medical Device ..... 2
1.4.3 Approval Process ..... 2
1.5 Stakeholders ..... 3
1.5.1 Cancer Patients ..... 3
1.5.2 Radiologists ..... 3
1.5.3 Oncologists ..... 3
1.5.4 Hospital IT Staff ..... 4
1.5.5 Health Canada ..... 4
1.6 System Features and Specifications ..... 4
1.6.1 Segmentation ..... 4
1.6.2 Feature Extraction ..... 5
1.6.3 Random Survival Forest ..... 5
1.6.4 Pipeline ..... 6
1.6.5 Overall Specifications ..... 7
1.7 Constraints ..... 7
2.0 Design Process ..... 7
2.1 Project Iterations ..... 7
2.2 Key Steps Taken ..... 8
2.3 Safety, Social, Economic, Environmental, Regulations and Professional Influences ..... 8
3.0 Solution Implementation ..... 9
3.1 Data ..... 9
3.2 Segmentation ..... 10
3.3 Radiomic Feature Extraction & Selection ..... 11
3.4 Survival Analysis ..... 12
3.5 Pipeline Implementation ..... 13
4.0 Testing and Evaluation ..... 14
4.1 Segmentation ..... 14
4.2 Survival Analysis ..... 16
4.3 Pipeline ..... 19

5.0 Discussion of Results ..... 19
6.0 Project Planning and Budgeting ..... 20
6.1 Adherence and Changes to Project Plan ..... 20
6.2 Project Planning Tools ..... 21
7.0 Project Reflections ..... 21
8.0 Conclusion ..... 22
References ..... 23
Appendix A – Github ..... 25
Appendix B – Code Snippets ..... 25

# Table of Figures

Figure 1: Assignment of classes to SaMD devices [13]. ..... 2
Figure 2: Summary of the required documents to be submitted in a Medical Device License application for the various classes [16]. ..... 3
Figure 3: High level pipeline illustration ..... 9
Figure 4: Example patient MRI scan from data set ..... 9
Figure 5: Model generated brain tumor segmentation overlayed on MRI scan ..... 10
Figure 6: Standard U-Net architecture ..... 10
Figure 7: Example terminal output for pipeline ..... 13
Figure 8: Predicted tumor mask along with MRI scan displayed in 3D Slicer ..... 13
Figure 9: Pipeline output patient survival curve showing probability of survival by days following diagnosis ..... 14
Figure 10: First iteration model training progress graph showing convergence of evaluation metric (green), and loss functions (blue and red) ..... 15
Figure 11: Second iteration model training progress graph showing convergence of evaluation metric (green), and loss functions (blue and red) ..... 15
Figure 12: Physician segmented tumor (left) and model segmented tumor (right). ..... 16
Figure 13: Log Rank test of Kaplan Meier analysis of sex with Sig. being the p value. ..... 17
Figure 14: Kaplan-Meier survival curve with sex as a factor showing increased survival probabilities for females. ..... 17
Figure 15: Output plots of survival analysis ..... 18

# 1.0 Project Background and Goals

## 1.1 Background

Brain metastases (brain mets) are secondary tumors that occur when cancer cells from a primary cancer spread to the brain. They occur in 20 to 40% of cancer patients [3], more often when the patient's cancer is in a dormant or quiescent state [4]. Primary cancers that are most likely to lead to metastasis in the brain are cancers of the lung, breast, and melanoma [5]. Although it varies based on the type of primary cancer, the average survival after the time of diagnosis of brain mets is less than 6 months [6]. As these tumors get larger, they affect the surrounding brain tissue, causing symptoms such as headaches, nausea, seizures, memory loss, and other mental changes [7].

## 1.2 Problem Definition

The unpredictable nature of a tumor is reflected in the decision making involved in cancer treatment. A radiologist will read a scan, compare it to previous known cases, and based on his knowledge and experience, will give a survival estimate and treatment course for the patient. Different types of treatments may have varied results, and there is always an uncertainty in which specific treatment is the best. The limiting factor in this situation is the level of rigor at which specialists can read cancer imaging, and how much information about previous cases they synthesize to select the best possible treatment course. Current methods of cancer treatment do not use computer-based prediction techniques such as machine learning models, which reveals an area of potential improvement for a more accurate prognosis.

When brain metastases are to be studied, they must first be segmented on an MRI or CT scan. This task is repetitive and highly time-consuming for a radiologist. Once the tumor volume has been outlined, the physician will select a course of treatment based on many factors, such as the size of the mass and its location. One important element that influences their decision is the patient's estimated time of survival. This can also be quantified by a machine learning prediction model, which can be used to supplement the physician's prognosis. In survival prediction, most current estimation techniques consider only the tumor's volume, the number of mets, and the primary tumor status, factors which may not have much prediction power over the outcome. The estimations made by physicians were shown to lack accuracy in survival predictions. Furthermore, an estimation of the patient's survival is based on subjective judgements of a physician, prone to inter and intra-observer variability [8]. Overall, this leads to poor survival predictions, which can lead to a sub optimal treatment course for the patient. An accurate prediction assures physicians have context to weight different treatments and patients can plan their future with more certainty of the survival estimate.

## 1.3 Problem Statement

Brain metastases are always accompanied by a grave prognosis. The use of a machine-learning based prediction model for the segmentation of tumors on medical imaging data can help improve the quality-of-care physicians provide to the patient by spending more time discussing treatment options, while eliminating the risk of human error. Improving the accuracy of survival prediction further increases the physician's ability to understand each patient's case and recommend the best treatment course for them.

The team proposes a pipeline that automates medical imaging segmentation for brain mets using the nnU-Net model [1] and delivers an accurate survival prediction with a Random Survival Forest model [2] that considers radiomic features of the tumors, paired with the clinical data from each patient.

## 1.4 Health Canada Approval for Software as a Medical Device

### 1.4.1 Medical Device Definition

Canada's Food and Drug Act, defines a "device" to mean, among other things, an apparatus or other similar article for diagnosing, treating, mitigating, or preventing a disease [9]. This definition applies to devices for use in animals, while the term "medical device" encompasses anything considered a device, but only refers to those intended for humans [10].

Medical devices are divided into four classifications which are Class I, Class II, Class III and Class IV where the classification is a reflection of the risk associated with the device [11].

### 1.4.2 Software as a Medical Device

Software products for medical applications are not explicitly addressed in the Canadian Food and Drug Act. As a result, their regulation and classification processes are not as clearly defined. For example, the normal classification system for traditional medical devices is appropriate for physical devices but does not make sense when considering software.

Health Canada has addressed the issue of a lack of clarity regarding software by introducing guidelines for a category of device called Software as a Medical Device (SaMD). These guidelines set out criteria for what devices fall under SaMD. First, a product can only fall under SaMD if it meets the definition of "device" and "medical device" under the Food and Drug Act [12]. The product must also perform one of many general functions listed in the guidelines. Next, the software must meet the two so-called inclusion criteria: the product it is intended for a purpose listed under the definition of device, and the software performs this purpose without being part of a hardware medical device [12]. At this point, a product may still be excluded from SaMD status if it meets all of the exclusion criteria: It does not process a medical image or signal, it simply handles medical information, it only supports health professionals or non health professional caregivers in making decisions about prevention, diagnosis or treatment, and it is not intended to replace clinical judgement in making a diagnosis or treatment decision regarding an individual patient [12].

If a product fits the requirements to make it a SaMD, it must then be assigned a class. For this step, information provided by the device is assessed on two metrics: the significance of the information provided by the product, and the state of the health situation when the information is to be used. Under information significance, "treat or diagnose" means that the information will be used to make an immediate or near-term action [12]. "Drive" means that the information will be used to identify early signs of a disease that will guide next diagnosis. "Inform" means that the information will not be used to make a near-term action. Under state of the situation, "critical" means that timely action is needed to avoid death, "serious" means that timely action is needed to mitigate irreversible consequences, and "non-serious" means that accurate diagnosis and treatment are not critical to mitigate long term effects [13]. Classes are assigned as shown in Figure 1.

| | Significance of information provided by SaMD to healthcare decision | | |
|---|---|---|---|
| **State of healthcare situation or condition** | **Treat or diagnose** | **Drive** | **Inform** |
| Critical | **III** | III | I or II |
| Serious | II or III | II or III | I or II |
| Non-serious | I or II | I or II | I or II |

*Figure 1: Assignment of classes to SaMD devices [13].*

### 1.4.3 Approval Process

To apply for a medical device license, certain documentation must be submitted depending on the class of device. A checklist of required documents for each class is shown in Figure 2. The three main documents that must be submitted are the medical device license application form, evidence of safety and effectiveness, and a Quality Management System (QMS) certificate. The application form asks for details about the manufacturer, the intended use of the device, and the standards the device meets. The safety and effectiveness requirement usually means clinical trials must be completed [14]. To meet the QMS certificate requirement, the company producing the device must comply with the standards of the Medical Device Single Audit Program (MDSAP) and then get certification from a Health Canada and MDSAP-accredited auditing organization [15].

| Required Submission Documents | | | | | Regulatory Requirements | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Responsible Parties | Manufacturer | | | All parties engaged in importation or sales activities | Manufacturer Importer Distributor | | | Manufacturer Importer | |
| Documentation | Medical device licence application and fee form | Objective evidence for safety & effectiveness | Quality Management System (QMS) Certificate | Compliant Label | Complaint Handling Record | Distribution Records | Must hold an active establishment licence | Recall Notice | Mandatory Problem Preliminary and Final Report |
| Class I | Requires Medical Device Establishment Licence | | | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| Class II | ✔ | Provide attestation | ✔ | ✔ | ✔ | ✔ | Exempt if holding an active medical device licence | ✔ | ✔ |
| Class III / IV | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | Exempt if holding an active medical device licence | ✔ | ✔ |

*Figure 2: Summary of the required documents to be submitted in a Medical Device License application for the various classes [16].*

## 1.5 Stakeholders

As this project presents itself as a very involved and novel application, there are a multitude of stakeholders to consider. Each stakeholder has a specific interest or concern with the project, and their insights will play a huge role in shaping the development of the application. Although the list could be expanded upon, the following are the most important stakeholders to consider.

### 1.5.1 Cancer Patients

Cancer patients must be able to rely on the diagnosis given from the application, and trust that the data is properly formatted, manipulated, and transferred among the components. Their critical non-functional requirements to be included in the design are reliability and security. The application must be reliable in a sense that it has a long mean time before error. It must also be secure from a technical standpoint to ensure the patient data is anonymized adequately. The success of the project directly impacts their quality of life and will help oncologists to determine their course of treatment.

### 1.5.2 Radiologists

Radiologists will be paramount in the success of the application through their expertise in segmenting tumors. Although they will not be a primary stakeholder for the system, the MRI tumor mask data must be created by them. They will be responsible for segmenting the tumor by hand and compiling the segmentations into a usable format. Radiologists will also be used as the basis for comparison, as the system will only prove to be a viable alternative if it can outperform a trained radiologist.

### 1.5.3 Oncologists

Oncologists will be the most hands-on user. They will be responsible for selecting the correct patient data and analyzing the output of the pipeline to determine a treatment course for the patient. Their critical non-functional requirements are usability and performance. They value usability because there is no expectation that they will have extensive technological knowledge, so they should be able to learn, use, and remember how to use the system as quickly as possible with little knowledge of the inner workings. They also value performance because the system must be a superior and efficient alternative to studying the data themselves.

### 1.5.4 Hospital IT Staff
The hospital IT staff will be responsible for the setup of the system, as well as dealing with any errors that arise. As a result, they prioritize installability along with robustness for the non-functional requirements. With these requirements in mind, the system must be designed for ease of install and reinstall, as well as designed to handle unexpected situations without failing.

### 1.5.5 Health Canada
Health Canada must approve the system before it can be used with patients. This process is described in the previous section. They have similar priorities as cancer patients which are to maintain data anonymity and ensure the system is very reliable.

## 1.6 System Features and Specifications
Expectations for this project changed between the time the project was chosen in ELEC 390 and when the team began work. As the team got familiar with the nature of the field of bioinformatics, it became clear that this project would be significantly technical and require more work than expected to obtain satisfying results. Following this realization, the team opted to focus on backend development and left behind the original plans it had to develop a user-friendly interface. Packaging the software produced in this project into a more accessible form would be necessary for commercial use, but is not required to present the functionality of the product.

The functional requirements for the system are documented using Hierarchical Textual Tags which facilitates referencing requirements later. It also ensures that when referencing requirements, the name has a meaning. An example of this would be referencing requirement Segmentation.SaveModel versus referencing requirement 1.5.1.2.2. The system features are split into four main components: segmentation, feature extraction, a random survival forest model, and the pipeline.

### 1.6.1 Segmentation
#### Description
The segmentation feature allows the user to train a segmentation model on various MRI scans to save and test for later use. The feature is of high priority.

#### Functional Requirements
Segmentation: Segmenting a tumor from a series of MRI scans.

| | |
|---|---|
| . ImportFiles | The system shall be able to import NIfTI files to use for training and testing. |
| . TrainModel | The system must train a segmentation model on the available MRI scan data. |
| . IterativeSave | The system must iteratively save the current state of the model during training to increase robustness |
| . CrossValidation | The system must split the data into a desired number of groups to perform k-fold cross validation to find the best model. |
| . SaveModel | The system must be able to save the model on the machine for later use on test cases and in the pipeline. |
| . TestModel | The system shall allow a user to access the trained model for testing on holdout data sets. |
| . Segment | The system must segment brain tumors from MRIs accurately. |

### 1.6.2 Feature Extraction

#### Description

The feature extraction feature allows a user to extract various elements of the segmentation data, the original images and the segmentation masks, into a usable format for further analysis. The feature is of high priority.

#### Functional Requirements

FeatureExtraction: Extract repeatable radiomic features from segmentation of MRI scans.

| | |
|---|---|
| . Extraction | The system shall be able extract a vast number of features from the MRI scans. |
| . DropFeatures | The system shall be able calculate colinear features and drop them. |
| . ExportFeatures | The system shall be able export the modified list of features to a csv for later use. |

### 1.6.3 Random Survival Forest

#### Description

The random survival forest feature takes extracted radiomic features along with other information about the patient to train a regression survival model which outputs survival predictions. This feature is of high priority.

#### Functional Requirements

RandomSurvivalForest: Predicting survival probabilities based on radiomic features.

| | |
|---|---|
| . TrainModel | The system must train a survival model based on the radiomic features. |
| . SaveModel | The system must be able to save the model on the machine for later use on test cases and in the pipeline. |
| . TestModel | The system shall allow a user to access the trained model for testing on holdout data sets. |
| . Survival | The model must output a survival prediction of the patient using inputted features. |
| . CensoredData | The model must consider censored data. |

### 1.6.4 Pipeline

#### Description

The pipeline features are an accumulation of the previous features, with modifications to allow for a new set of MRI scans to seamlessly pass through the features all in one script. This feature is of high priority.

#### Functional Requirements

Pipeline:

| | |
|---|---|
| . FormatData | When the user starts the application, the system must check that the required data files are in the correct folder. |
| . Error | If there are missing data dependencies, notify the user of which files are missing and terminate the program. |
| . CreateDirectories | The system must create a new directory for the output data of the segmentation predictions to be stored, and an additional directory for the desired '.nii.gz' file. |
| . MoveFiles | The system must move the required '.nii.gz' file into its own directory. |
| . PredictSegmentation | The system must be able import the trained segmentation model and pass it the formatted MRI scan data. |
| . DisplaySegmentation | When the prediction has completed, the system shall open a new terminal within 3D slicer to display the segmented tumor. |
| . FeatureExtraction | When the prediction has completed, the systems shall pass the original scan and the segmentation mask to the feature extraction script. (Sub-requirements are the same as the main section or feature extraction) |
| . FeatureSelection | When the feature extraction has completed, the system shall pass the csv of features to the feature selection script. |
| . LoadData | The system shall load comparison data from original segmentation and survival model training. |
| . DropColumns | The system shall perform a column wise comparison to drop all columns in the current csv that do not exist in the segmentation comparison data. |
| . JoinColumns | The system shall perform a join based on patient number with the current csv and the survival prediction data. |
| . SurvivalPrediction | The system shall pass in the preprocessed csv file to the trained random survival forest model to run a prediction. |
| . LoadModel | The system shall load in the pre trained survival prediction model. |
| . ExportResults | The system shall export the resulting predictions to a csv file. |
| . DisplayGraph | The system shall display a line graph demonstrating days passed and probability of the patient being alive. |

### 1.6.5 Overall Specifications

The final iteration of the project's specifications are summarized in Table 1 below.

*Table 1: Project Specifications*

| | |
|---|---|
| **1** | **Functional requirements** |
| 1.1 | Train machine-learning based segmentation model using MRI dataset from patients with brain mets. |
| 1.2 | Train model for survival prediction using a combination of features extracted from MRI data and tumor segmentation produced by model from 1.3, as well as corresponding clinical data for each patient. |
| 1.2 | Select best models for segmentation and survival prediction based on resulting accuracies. |
| 1.3 | Produce accurate segmentation of MRI data in a holdout testing set. |
| 1.4 | Produce accurate prediction of survival for patients in dataset. |
| 1.5 | Combine segmentation and survival prediction into a pipeline, while maintaining accuracy in both tasks. |
| **2** | **Interface requirements** |
| 2.1 | Allow user to select an MRI from a dataset, paired with clinical data, and obtain a visual representation of the tumor's outline as well as a survival prediction graph. |
| **3** | **Performance requirements** |
| 3.1 | Entire pipeline, as well as individual pieces satisfy validation metrics. |
| 3.2 | Ensure that final product is easily deployable as a script in the command line. |

## 1.7 Constraints

During the ideation phase of the project there were many constraints to take into consideration which would affect the success of the project. The first and most prominent constraint is the quality of the training data. This heavily depends on the skill and experience of the radiologist, who segments the tumors by hand, the number of scans available for this type of tumor, and the graphical quality of those scans. The other constraints are the computation power required to train such an intensive model, and regulatory restrictions which come in the form of data anonymity as well as the difficulty in making this project truly deployable.

# 2.0 Design Process

Since ELEC 390 the purpose of this project has been to create a piece of software using machine learning that can augment a physician's skill in completing technical tasks related to cancer patients. The team considered several approaches to serve this general purpose, and over time the project shifted focus onto different approaches. The project is heavily based on what data the team was able to acquire. This ultimately led to the final product which is a pipeline that processes an MRI image and outputs a segmentation mask of the tumor, and finally a prediction of the survival time of the patient.

## 2.1 Project Iterations

### ELEC 390

In ELEC 390 the team proposed to use machine learning to process a patient's MRI, CT scans, as well as other relevant data to find the best treatment method. This would help physicians by providing an unbiased and reliable tool to determine course of action for a cancer patient. This project would have required an archive of patients' MRIs, personal data, treatments used, and outcome data. This detailed

data was not available, so the approach was modified. In fact, it became clear toward the end of ELEC 390 that availability of data would heavily influence the scope of the project.

### Start of ELEC 49X (Fall 2020)

At the start of ELEC 49X and with better knowledge of what data would be available, the team decided to produce segmentation software for MRIs of patients with brain metastases. This would provide a faster and consistent alternative to physicians whose time could then be diverted to other important tasks. This process would also remove random error from the segmentation process.

### Mid-End of ELEC 49X

After the team was on track to come up with a good segmentation model, it was decided to integrate a patient survival analysis model created by Simon Crête which led to the development of the final integrated pipeline.

## 2.2 Key Steps Taken

The previous section outlines the key shifts in scope of the project through the past 15 months. The specific actions that led to the final pipeline are described here.

First, research was done on various types of cancers and methods. To further broaden understanding of machine learning as applied to medical images, the team explored data provided by University of Pennsylvania's Brain Tumor Segmentation (BraTS) challenge. One of the challenge's top performing models, "Unet3D", available to the public on GitHub was also run on the data by the team. This process yielded insights to the types of libraries that would be required and the types of errors that would be faced. It also helped familiarize the team with 3D Slicer software for visualization of medical images and the data format itself.

Having already decided to include the survival analysis model, research was done to determine the best models to use in each stage of the pipeline. For image segmentation in general, a model known as U-Net is very popular. An adaptation of U-Net, known as nnU-Net [1] is specifically tailored to segment medical images so this model was selected. For the survival analysis, the Random Survival Forest model [2] was chosen. Both of these models are well documented and well researched. Thus, by using these models, the pipeline may be safer, more reliable, more accurate, and more likely to be approved than if less-known models were chosen.

In Week 18, the team decided against building out a complete GUI as originally intended. This was done to allow the team's focus to be shifted to improving the model accuracy. This decision was partially influenced by the acknowledgement of priorities for the project. One of the reasons this project was done was to further machine learning for medical applications. It was determined that this cause would be better supported by showing better technical capability. Furthermore, the team was aware that prior to applying for a Medical Device License, if developed during this project, the GUI would likely need to be reworked later to meet precise specifications.

## 2.3 Safety, Social, Economic, Environmental, Regulations and Professional Influences

In this project, social concerns were relevant when handling patient data. Patient MRI and clinical variables are confidential information which must be maintained private to anyone outside of the study. The team assured patient privacy by only using anonymized data, leaving no identifiable patient information. The main safety consideration was the degree of reliability and accuracy of the system as it would affect the treatment of patients. For this reason, effort was made to improve the model even when adequate results had already been achieved. The economic potential of this device rests on its usefulness in aiding physicians. By introducing the second model to form the pipeline, the capability of the system was expanded, making it more useful. Finally, as the pipeline is purely software, meaning it does not require manufacturing or physical distribution, there are no direct environmental considerations.

Regulatory compliance is a massive consideration for real world implementation of this project. Before it can ever be used in hospitals, it must be approved. Success in the approval process requires detailed documentation and understandability of the software. Great care was taken to well document

the development process. This extends to creating readable code and recording the methodology used at each phase of the project.

# 3.0 Solution Implementation

Due to the nature of the project being solely software based, there is no true bill of materials to be presented. The only external entity used in the development process was a powerful computer to train and test the segmentation model, which was provided by Dr. Simpson's lab. As a result, there was also no money spent on the project. Figure 3 illustrates each step taken in the pipeline.

## 3.1 Data

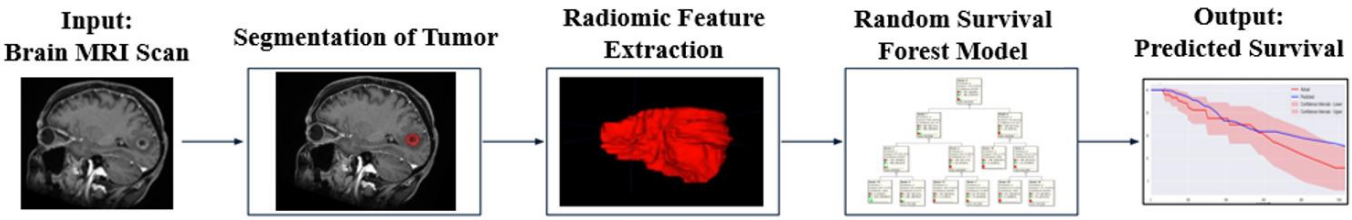


*Figure 3: High level pipeline illustration*

The data was acquired from a Kingston General Hospital (KGH) and a Queen's University study consisting of 200 patients diagnosed with brain metastases with mixed histologies. This dataset has not been studied before and is therefore novel. Each patient has an MRI with a physician-drawn segmentation of the brain metastases tumors saved as NIfTI files. Of those 200 samples, 157 patients have clinical variables and outcome data consisting of patient ID, number of brain metastases, age, sex, vital status, death event, date of death or last follow up, date of diagnosis, and gross tumor volume. From the above-mentioned dates, a survival period was calculated in days for each patient using MATLAB's daysact function and is formatted in a comma separated values (csv) file. This data now acts as our clinical variable information data. A 10% holdout set is saved from the 157 samples to be used to evaluate the survival model and the pipeline.

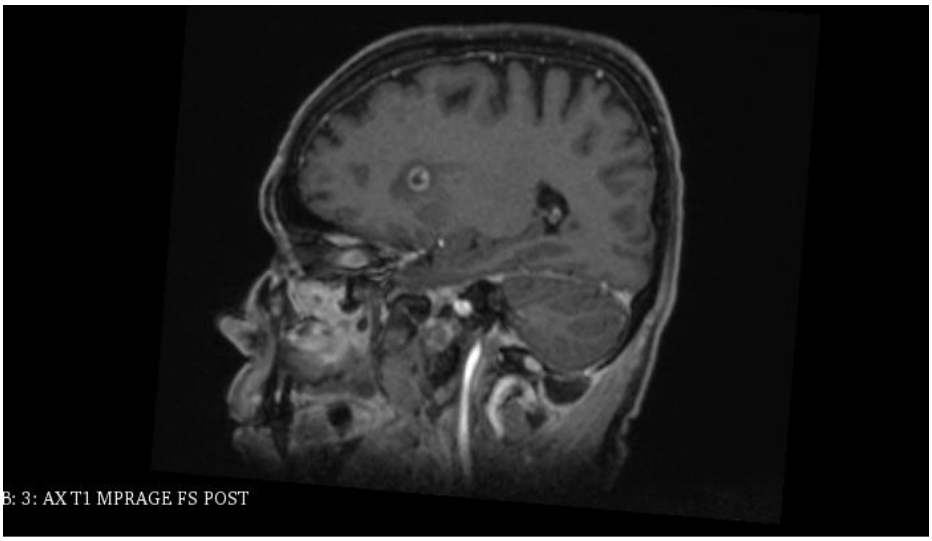


*Figure 4: Example patient MRI scan from data set*

## 3.2 Segmentation

Segmentation, as it relates to this project, means taking an MRI containing one or more tumors and creating a map of voxels (3D pixels) classified by whether these voxels are part of the tumor. This voxel map can be interpreted as a 3D image of the same dimensions as the MRI scan, in which the tumor is represented by voxels of intensity 1, and all other pixels are of intensity 0. A more applicable use of the tumor segmentation map is to overlay it on top of the original MRI as seen in Figure 5.

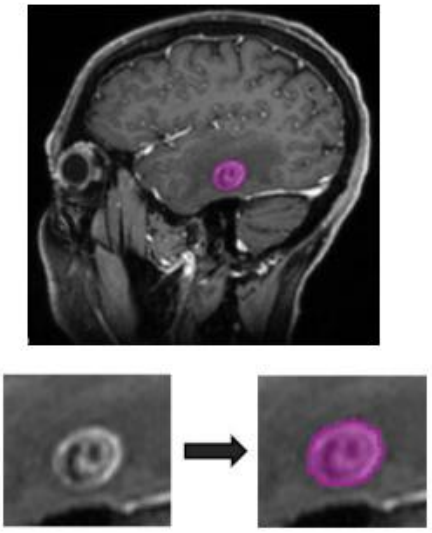

*Figure 5: Model generated brain tumor segmentation overlayed on MRI scan*

For this task, the team opted to use a model called the nnU-Net [1]. It is an open-source neural network based on the U-Net [17], a widely recognized machine learning algorithm designed specifically for biomedical imaging segmentation.

The U-net is built off the Convolutional Neural Network (CNN) model, which is a model typically used for classification tasks. In the case of biomedical image segmentation, this means classifying each pixel as being part of the tumor or not. A supplementary step is added to the original CNN model, where the resulting down-sampled image is then further convolved and up-sampled to obtain a higher resolution and more accurate segmentation output. The architecture of the U-Net can be seen in Figure 6 below.

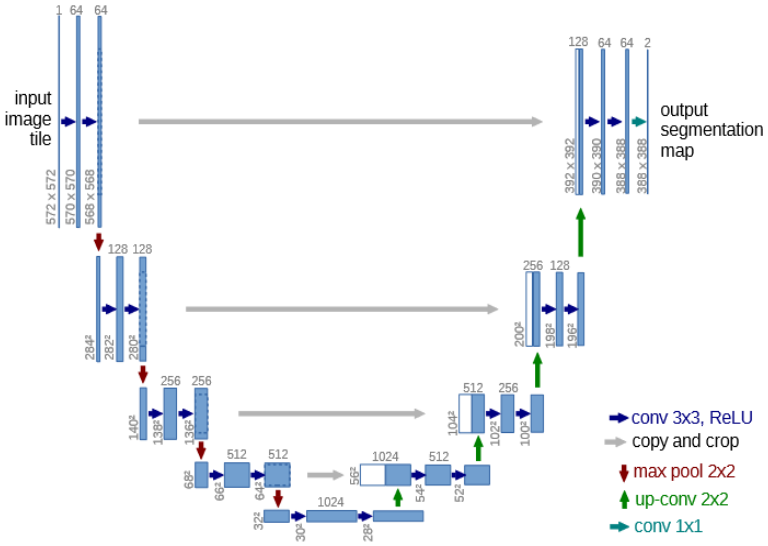


*Figure 6: Standard U-Net architecture*

The nnU-Net uses a slightly modified U-Net architecture, incorporating pre-processing and post-processing into a process that typically produces higher accuracy than the original U-Net. The nnU-Net provides different versions of the modified U-Net architecture, and the team opted to use a 3D U-Net architecture for this project.

The nnU-Net takes the MRI scans of the patient as its input, as well as the segmentation of each tumor done by radiologists. In each case, the nnU-Net trains using 5-fold cross-validation, which means the data is split into 5 subsets, where 4 subsets are used for training, and one is used for validation. This process is repeated five times, so that the model has trained and validated itself on the entirety of the dataset. During each fold, the training dataset is passed through the neural network many times, during which it makes changes to the operations it effectuates on the data before categorizing it. Each time the network has iterated over the entire dataset, it is said to have undergone one epoch. For the nnU-Net model, each fold runs for around 1000 epochs. The goal is for the model to improve its accuracy in segmenting the tumor after each epoch. The metric used to assess accuracy is called Dice Score, which is discussed in more detail in 4.1 Segmentation.

The result of this training step is five trained models, each corresponding to one fold. The next step is to use this model to make predictions on data outside of training dataset. This process is called "inference", which means the model is using what it has learned to make predictions on data it has never been trained on. To maximize the accuracy of the output, all 5 models are used to make these predictions, which is typically referred to as "ensemble learning". The use of this method allows the nnU-Net to make predictions based on 5 networks that have collectively been trained on more data than any of the individual models.

## 3.3 Radiomic Feature Extraction & Selection

Radiomic features can be extracted from the MRI segmentations using computational methods. These features consist of statistics, pixel studies, and geometry of the segmented portion of the image. Feature extraction occurs twice, once initially on all samples to provide training and testing data to the survival model, and once in the pipeline to provide the survival model with an input. This method was implemented in Python, using the open sources libraries mentioned below.

Radiomic features were extracted using the PyRadiomics [18] open-source package. Image intensities were normalized using z-scores on a per-image basis. It is typical to resample image voxels to isotropic voxels [19] [20], therefore images were resampled to 0.9x0.9x0.9. These values were chosen based on the mean x/y axis resolution of the images across all patients. The mean was chosen across all images so that some are up-sampled, and others are down-sampled, since there is no consensus as to which is best [21] [22], though with this technique the Z axis will typically be up-sampled. The only values that were set differently from default in PyRadiomics were: binWidth = 0.116, voxelArrayShift = 0.8. The bin width was based on computing the 0.001, and 0.999 quantiles of intensity values after z-score normalization across all images, which were: lower, upper = (-0.587, 5.234) then performing binWidth = (upper-lower)/50. The binWidth parameter is used for discretization of the image for some of the feature computations. The PyRadiomics documentation recommends a choice of binWidth as to let the number of bins be between 30-150. The approximate range for a typical image was taken and divided by 50. The voxel array shift was chosen based on the minimum intensity inside the masks after normalization, which is approximatively -0.82. The voxelArrayShift is meant to ensure the data is mostly positive, therefore choosing a value close to the minimum will assure that this is the case. Overall, 116 radiomic features were extracted per segmentation from the data set. The same process is followed in the pipeline to extract features for the survival model input (see feature extraction Appendix B – Code Snippets).

The survival model cannot evaluate all 116 features effectively: only around 15 features should be used. Only the most predictive and informative features must be kept to provide the survival model with optimal data. A random survival forest is robust to weak features, meaning that a more conservative approach is acceptable when performing feature selection. Certain features that may

have been removed in a harsher selection can remain in the model and potentially provide good predictive power.

Certain subsets of features provide the model with similar information. These redundant features serve only to confuse the model, and within a subset of similar features, all but one should be removed. To determine the subsets, features with high variance inflation factors (colinear) are found and grouped. All features are removed except the one with the highest variance (see feature selection Appendix B – Code Snippets). The variance inflation factor threshold for grouping can be varied to arrive at a feature set providing an optimal output. A variance inflation factor threshold of 6 was found to be optimal for model accuracy. This was done by experimentally trying different threshold values. The chosen threshold removed 101 features, leaving 15 for prediction. Feature selection is an iterative process which was conducted throughout the rest of the model development.

The model is primarily utilizing the extracted radiomic features as data, however clinical variables included in the outcome data may prove to be useful as well. To determine if any features have significant influence on the patient survival outcome, a Kaplan-Meier analysis and a log rank test can be performed on each of these metrics using a statistical analysis program called IBM SPSS. If a feature is considered to have significant influence, meaning a p value inferior to 0.05, then it should be added on the model. Prior to the analysis, all Kaplan-Meier assumptions should be met. The event and censoring are mutually exclusive and collectively exhaustive, which is easily verifiable by looking at our data and from its origin at KGH. The time to an event or censoring is accurately measured as a date and was converted to a number of days. Left censoring is minimized within the study. There is independence of event and censoring, patient with the event are only ones dead due to cancer or cancer related complications. Secular trends are minimized since the data was acquired over only four years, from 2016 to 2020. For the division of the features that are analysed, it was assured that the proportions of censorship between groups was similar.

## 3.4 Survival Analysis

The Random Survival Forest is an extension of the Random Forest model introduced by Breinman et al in 2001 [23], the Random Survival Forest model was developed by Ishwaran et al. in 2008 [2]. This model is a randomized ensemble of trees to analyse right censored data. The model is implemented in Python, utilizing the PySurvival open-source library [24], which provides a Random Survival Forest API. Hyperparameters for the model include number of trees, depth of trees, minimum number of nodes. To determine optimal hyperparameter values, a grid search is performed, the model is tested with multiple combinations. The model will iterate with nested for loops sequentially changing the value of one parameter each iteration. The concordance index (C-index) is calculated for each model, and the parameters for the model with maximum C-index is selected, as it is the one with the most discriminating power. The selected parameters are set on the final model, which uses 5-fold cross validation to prevent overfitting and to validate. To evaluate the final model, two evaluation metrics are utilized: C-index and Integrated Brier score (IBS). Plots of the observed survival vs. the predicted survival are used to visualize outputs (see Random survival forest code Appendix B – Code Snippets).

## 3.5 Pipeline Implementation

The pipeline is the culmination of the above components into one seamless script which can perform all the aforementioned actions in sequence on an MRI of the user's choice. The process is automated through a bash script which progressively passes data through each step of the pipeline. Each pipeline step is generally a python file or a nnU-Net command (see Bash Pipeline Script Appendix B – Code Snippets). Figure 7 is an example of the terminal in action running the bash script.

```
(base) lab@simpsonlab:~/capstoneGit/Capstone/pipeline$ bash capstone-pipeline
Enter folder where desired image is (FSRTCASE_XXX.nii.gz)
/home/lab/capstoneGit/Capstone/pipeline/inputTest
---START Segmentation---

---START Feature Selection---

---FINISH Feature Selection---

---START Survival Prediction---

Loading the model from ./survival_model.zip
randomSurvivalForest.py:9: DeprecationWarning: 'pyarrow.deserialize' is deprecat
ed as of 2.0.0 and will be removed in a future version. Use pickle or the pyarro
w IPC functionality instead.
  survivalModel = load_model('./survival_model.zip')
```

*Figure 7: Example terminal output for pipeline*

First, the user is asked to input the file location of the MRI scan to analyze which must be in the named in the format of 'FSRTCASE_XXX.nii.gz' where the XXX represents a patient identification number. The script then takes this file location and calls the nnU-Net prediction function to generate a mask for the tumor. Once the mask has been generated, the first output of the script is visible as 3D Slicer is opened with the new mask and the original MRI scan for viewing. The user has the opportunity to iterate over the slices and isolate where the tumor is, which will be highlighted in green. Figure 8 is an example output from the script.

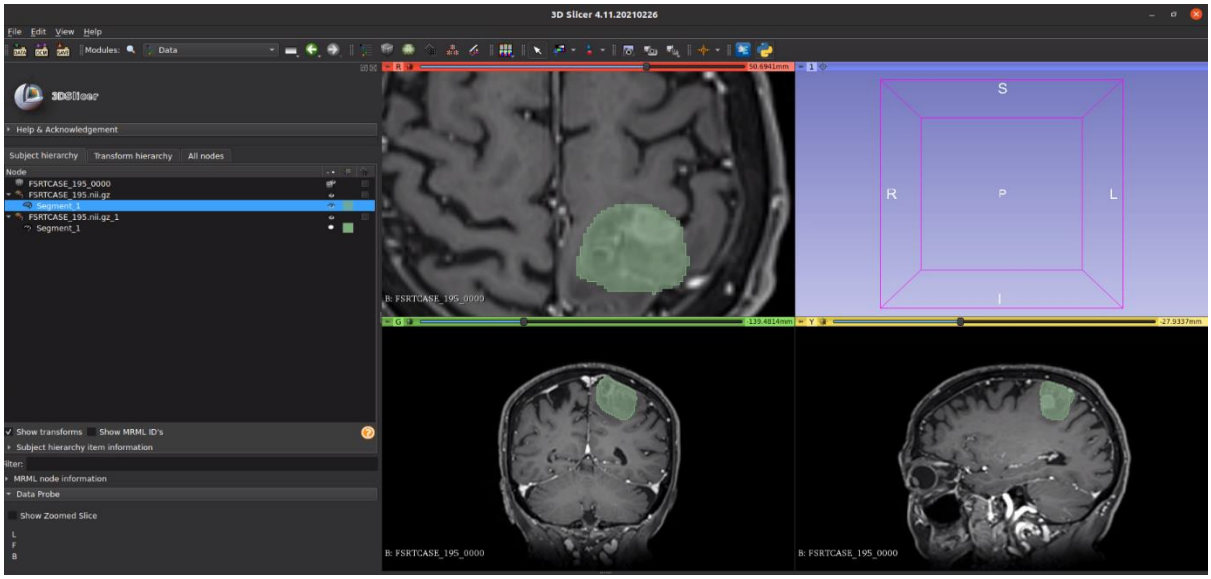


*Figure 8: Predicted tumor mask along with MRI scan displayed in 3D Slicer*

This scan is then passed to the segmentation component which outputs the tumor mask to an independent results folder. This new mask generated by the prediction model along with the original MRI are then passed to the radiomic feature extraction component, which performs all the necessary preprocessing for the data to be passed to the random survival forest model. The process is the same as that described in section 3.3 Radiomic Feature Extraction & Selection.

The survival prediction imports a saved random survival forest model which was trained as described in section 3.4 Survival Analysis and runs the predict survival function built into the model. After some calculations, the survival model outputs a comprehensive graph which demonstrates survival probabilities each day, as seen in Figure 9 below.

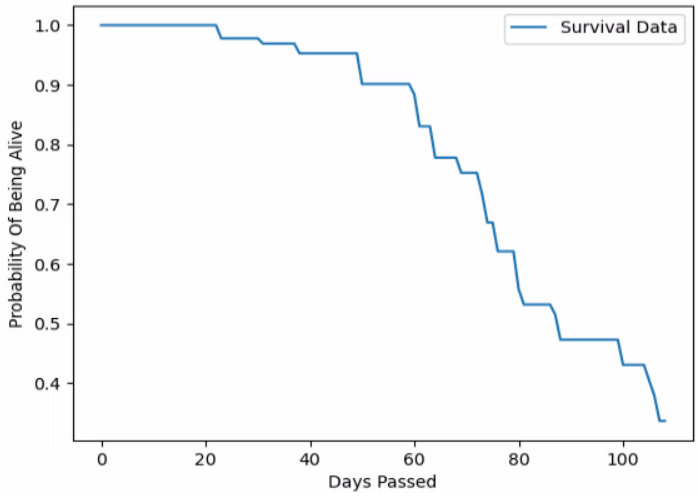


*Figure 9: Pipeline output patient survival curve showing probability of survival by days following diagnosis*

# 4.0 Testing and Evaluation

To assess the pipeline's accuracy, a small 10% holdout set is used. These samples are not used to train either models. It is crucial to evaluate the output of both the segmentation and survival models individually to improve their respective accuracies. Thereafter, the integrated pipeline can be evaluated, to determine how the random survival forest is able to use the output of the segmentation model.

## 4.1 Segmentation

Segmentation models are quantitively evaluated using several metrics, the most important being the Dice Score (dice), which represents the spatial difference between two sets of segmentations.

$$Dice\ Score = \frac{2|X \cap Y|}{|X| + |Y|}$$

Where X and Y are the number of pixels/voxels in each respective segmentation. The two segmentations examined are the physician drawn brain metastases segmentations and the model generated ones.

The segmentation nnU-Net model utilized 5-fold cross validation to provide multiple validation models and scores. The full model was trained three separate times, with iteration two and three having the input data modified to improve accuracy. The nnU-Net computes a dice for each validation sample, and outputs a mean score for the fold. The mean of the 5-folds is then calculated for evaluation purposes.

The first iteration was performed on unaltered data, this model provided a mean dice accuracy of 0.60 averaged across all five validation sets. The results varied significantly between samples, some had a dice of 1 while others near 0, showing that the model was able to accurately segment only certain tumors. This iteration did not provide satisfying results and several methods were utilized to increase model accuracy. First, the results for each sample outputted by the model were tabulated with data such as voxel size, gross tumor volume, segment volume, to determine if any patterns emerged. When looking at the training graph for the first iteration of training, we see the dice and the loss function

fluctuate significantly which is abnormal for training, suggesting problems wit the data. No patterns were found between low accuracy and characteristics of the data. All the outputs were then individually visually analysed using 3D Slicer to find any possible issues with the data. Nine images were found to not have segmentations of the tumors, but instead segmentations of the whole brain, which resulted in a low dice. The team decided to remove these samples as there was no way to fix them at the time.

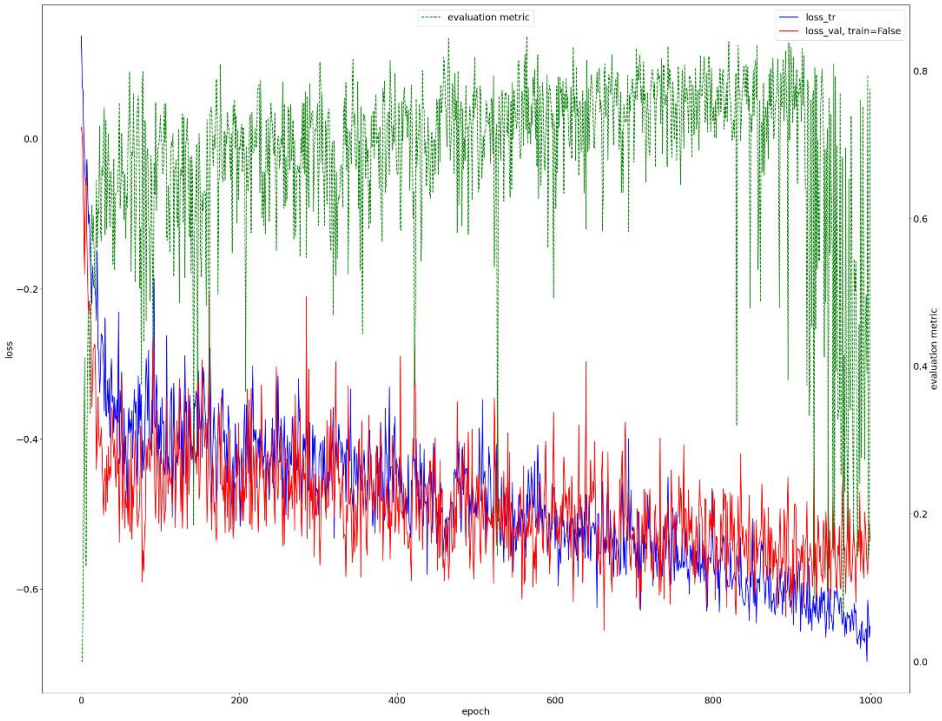


*Figure 10: First iteration model training progress graph showing convergence of evaluation metric (green), and loss functions (blue and red)*

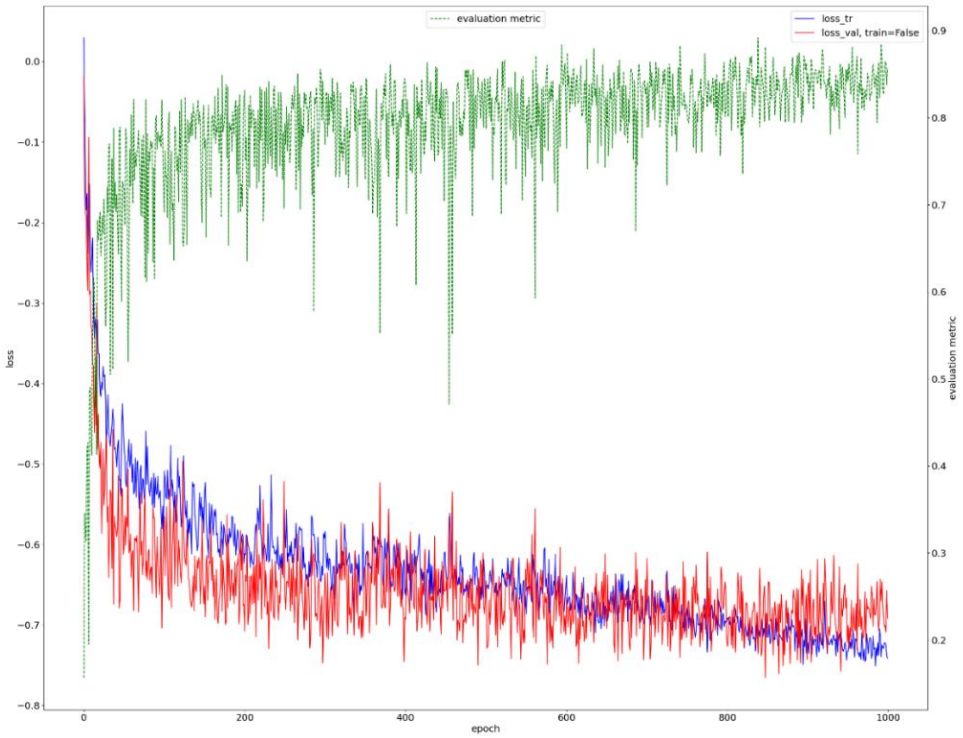


*Figure 11: Second iteration model training progress graph showing convergence of evaluation metric (green), and loss functions (blue and red)*

The second iteration did not include the samples with problematic segmentations, and this resulted in higher accuracies with a mean dice of 0.745 across the 5 cross validation folds. The removal of the bad samples improved the accuracy by removing their low scores but also by allowing the model to train more efficiently. For the second iteration the graphs fluctuate less and stabilize in the last 200 epochs, showing better training. Even with the improvement in mean dice, there are still some cases with very high accuracies and some with lower accuracies.

To fix the above-mentioned problem, eight segmentations were redrawn with the assistance of radiation oncologist Dr. Fabio Ynoe De Moraes. The model was then trained again, but surprisingly, lower dices of 0.72 were achieved. This result is counter intuitive, as a more accurate segmentation should increase accuracy, however, it is possible that the newly drawn segmentations had formatting issues. The team believed that by creating more accurate ground truths, the model metric would be higher. The model takes roughly 24 hours to train one-fold on a GTX3090 GPU, therefore roughly 5 days to train 5 folds. Therefore, the team did not have time to redraw the segmentations and decided to use the previous model. The team opted to use the second iteration of the model.

Overall, this model has an acceptable accuracy for its purpose. As previously mentioned, the qualitative analysis of the model output showed it performed better than what its dice score suggests. The samples were once again visually inspected, and multiple samples were found to have inaccurate physician drawn segmentations and accurate model generated segmentations, resulting in a lower dice. Therefore, we see that the dice metric is not perfect at evaluating this model. The final segmentation model is an ensemble of all 5 validation folds, which creates a stronger and more robust model. The output of this ensemble seems to present even better results. It is safe to suggest that our segmentation model is likely to be sufficiently accurate for its purposes, but issues are still present on some edge cases.

Figure 12: shows a physician drawn segmentation on the left and a model generated output segmentation on the right. As we can see, the physician segmentation is not as tight around the tumor and covers normal brain tissue. The computer-generated tumor seems to only cover tumor tissue. This would result in a segmentation that is more accurate of the tumor and is potentially better at extracting radiomic features.

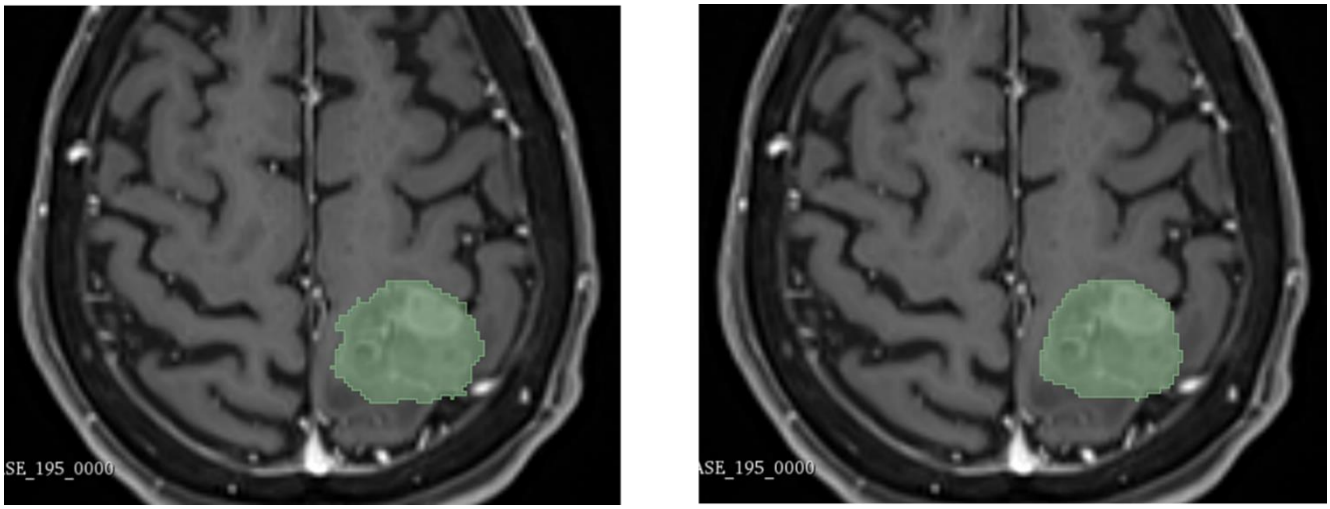


*Figure 12: Physician segmented tumor (left) and model segmented tumor (right).*

## 4.2 Survival Analysis

The feature selection was successful at removing features that were highly correlated, effectively removing redundant variables for the model. This is verified by simply comparing a correlation matrix of the features before and after feature selection. Before feature selection, multiple features had very

high correlation, some even approaching a correlation of 1. After feature selection, most features are not highly correlated. This created a useable set of radiomic features for the survival model.

Kaplan Meier analyses of the clinical variables outputted a survival curve plot and a log rank test with p value for each feature tested. Overall, only sex had a significant log rank score, with a p value of 0.018. From the survival curve, females are shown to have consistently greater survival rates than males. Sex was shown to be the most significant discriminating factor between the three clinical variables studied. This analysis demonstrates that sex should be added as a feature to the model as it is likely to increase performance as a feature. Performance of the feature should be assessed after addition.

**Overall Comparisons**

| | Chi-Square | df | Sig. |
|---|---|---|---|
| Log Rank (Mantel-Cox) | 5.593 | 1 | .018 |

Test of equality of survival distributions for the different levels of GenderM0F1.

*Figure 13: Log Rank test of Kaplan Meier analysis of sex with Sig. being the p value.*

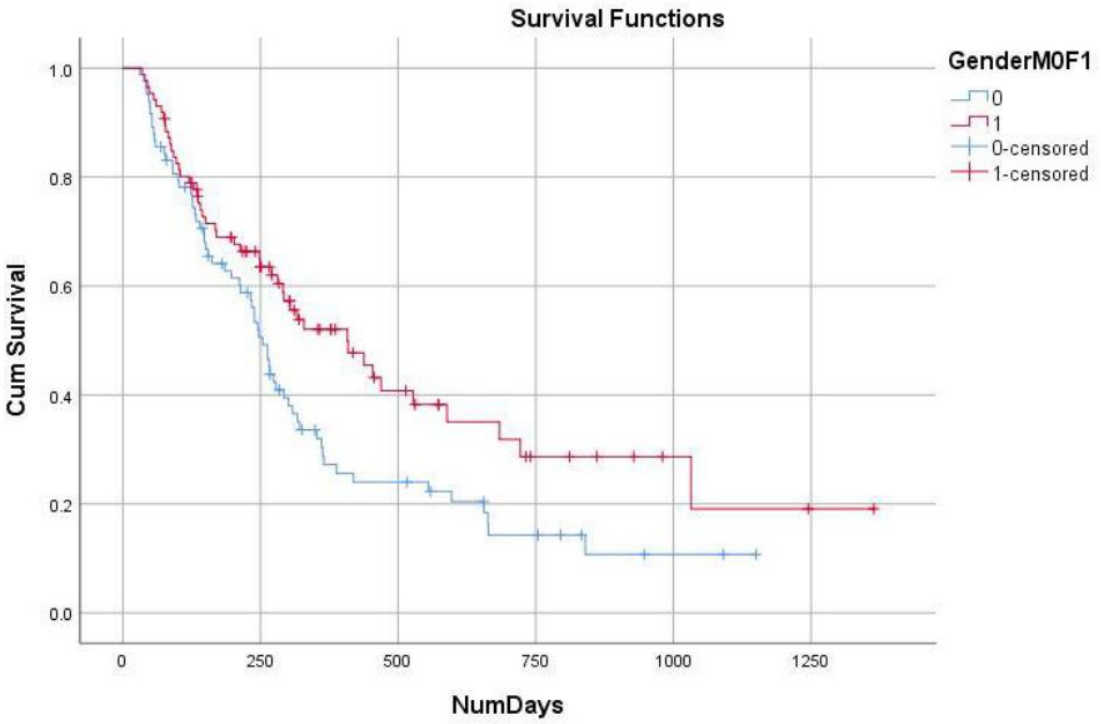


*Figure 14: Kaplan-Meier survival curve with sex as a factor showing increased survival probabilities for females.*

The metrics used to evaluate the final model were average C-index and IBS. The C-index is a generalization of the area under the ROC curve (AUC) that can take into account censored data. It represents the global assessment of the model discrimination power: this is the model's ability to correctly provide a reliable ranking of the survival times based on the individual risk scores, and ranges from 0 to 1, with 1 being the best possible value [24]. The Brier score is used to evaluate the accuracy of a predicted survival function at a given time; it represents the average squared distances between the observed survival status and the predicted survival probability and ranges from 0 to 1, with 0 being the best possible value [24].

The optimal model hyperparameters varied each iteration, but remained around 15-20 trees, a depth of 10-15, and a minimum node count of 5-15. Note that some variations of the parameters provided similar results.

The model used 5-fold cross validation with 75% training 25% testing split, and a 10% holdout set for evaluation. The survival model obtained a C-index of 0.74 and IBS of 0.17 and C-index of 0.675 and IBS of 0.175, on the training and validation data, respectively. The holdout test set obtained a C-index of 0.674 and IBS of 0.168. Figure 15: Output plots of survival analysis displays the validation metrics; a) the Brier Score for every time unit remaining under the 0.25 threshold, b) the predicted survival vs. the actual survival with the confidence interval of the prediction.

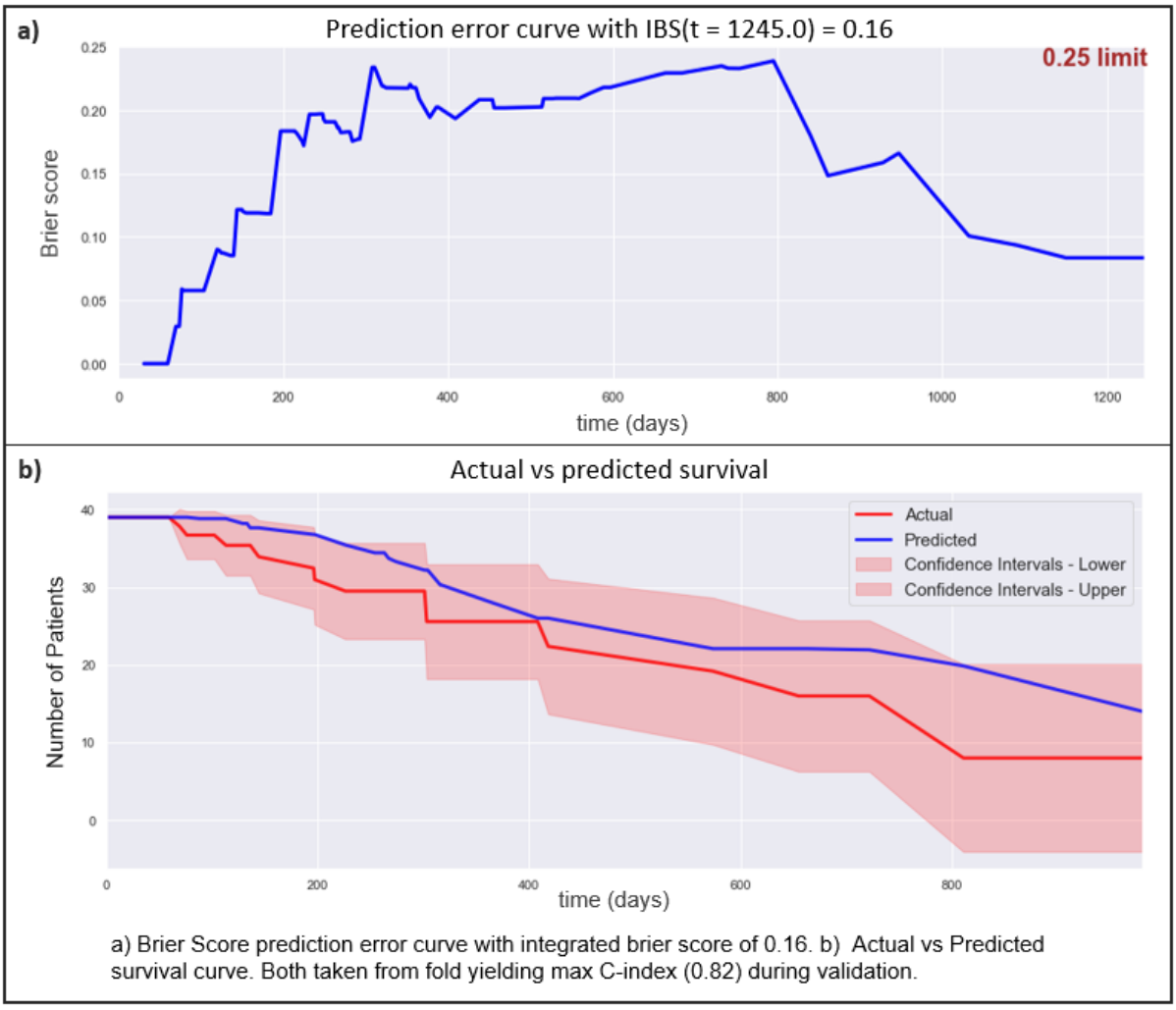


*Figure 15: Output plots of survival analysis*

The model has C-index values that vary from 0.58 to 0.76 depending on the validation sets. Typically, the model outputs one validation set with a low score, one with a high score, and three with a score close to the average value. It is clear that the model performs unevenly on the data, and perhaps performs poorly on a certain subset of patients. The Integrated Brier Score returns values consistently under the threshold of 0.25 required for a model to be considered useful, values typically range between 0.15 to 0.20, and the brier score consistently stays under the 0.25 threshold for each time unit.

These results show that the model is predicting survival with a reasonable accuracy. A mean C-Index of 0.675 presents a clearly non-random survival ranking. A mean IBS of 0.17 shows that the survival probability prediction is accurate over the survival time period. While the C-Index is not particularly high, the IBS score impressive. The low C-index could potentially be due to multiple patient deaths at

a similar time, making it hard to distinguish exactly which death occurs before the other. However, the high IBS shows that the survival prediction would still be relevant in prediction a general survival for these patients.

### 4.3 Pipeline

The pipeline's accuracy is evaluated by scores of the survival model. The team will compare the results of the same 10% holdout set in the independent survival model with features extracted from physician segmentations, and the pipeline integrated survival model with features extracted from computer drawn segmentations. This will show if the pipeline is able to reproduce accurate segmentation, feature extraction, and survival prediction from those computer-generated inputs.

The 10% holdout set MRI images are passed through the pipeline, segmentations are made, features are extracted, sex is added from the clinical data, and these are passed to the survival model. A C-index of 0.68 and an IBS of 0.15 were achieved. Therefore, the pipeline has slightly higher accuracy on the holdout set, due to a higher C-Index, and a lower IBS. The only varying factor between the two are the segmentations (physician vs. computer), which are used to generate the input features for the holdout set. Therefore, it is possible that the improvements come from more accurate segmentations, meaning the computer might be performing slightly better segmentations in our dataset for the purposes of radiomic feature extraction. However, the difference in accuracy is not significant and this hypothesis is not proven.

## 5.0 Discussion of Results

As we have seen from the previous section, the models and pipeline show good accuracy and meaningful prediction. Considering all the factors and evaluation metrics, it is reasonable to postulate that the model is capable of moderately accurate and reliable survival predictions of patients with brain metastases using radiomic features. The prediction model can detect a signal and predict meaningful outcomes from the radiomic features alone. This shows that these radiomic features are predictive of survival outcomes. This is a significant finding, as it suggests that there is potential for accurate brain metastases survival models using radiomic features. However, there are some constraints to the findings. Patients in the data set have mixed primary cancers. This could lead to a lack of repeatable results for patients with different primary cancer histories, and the consequences of this should be extensively reviewed by qualified physicians. Our findings are limited to one independent and novel dataset, limiting the generality of the findings. This means that various datasets from different provenances must be evaluated. Finally, the model does not perform as desired on certain samples. This is likely a direct effect of a small dataset with edge cases, which further increases the need for more data to be analysed.

Comparing the pipeline's result is difficult as no other groups have worked on this dataset and no similar pipeline exists which evaluates brain metastases. No paper was found which uses a random survival forest utilizing radiomic features for brain metastases overall survival, meaning the evaluation metrics used in this paper cannot be compared to other research. However, multiple other papers suggest radiomic features have predicting power over primary brain cancer and brain metastases [25] [26] [27]. These papers use cox regression and other machine learning approaches, which also seem to output similar results, but do not consider time dependent survival.

Creating a clinical survival estimation tool to assist physicians with this method is still distant. However, this can be considered to be a meaningful step which shows that the method has potential, and it is reasonable to pursue further research the subject.

# 6.0 Project Planning and Budgeting

## 6.1 Adherence and Changes to Project Plan

As mentioned in 2.0 Design Process, the project scope was changed when the survival analysis model was added. This reflected a shift in focus from building out the complete software package including a GUI to improving the capability of the software. The team's resources were put into developing this second model and writing additional code to integrate the two models into the data pipeline. Acknowledging that reasonable visualization could still be achieved by using 3D Slicer, the team decided not to pursue development of a GUI. Apart from this change, various delays in receiving access to data and a capable computer for training resulted in some tasks being finished later than expected. The specific additions and deletions to the original project plan are shown in Table 2.

*Table 2: Original project plan showing changes made. Red text represents a removal and green text represents an addition to the project plan.*

| No. | Milestone | Due date | Responsible member(s) |
|---|---|---|---|
| 1 | **Understanding and organizing the preprocessed data that was acquired** | Week 5 | **All members** |
| 2 | **MNIST Handwritten number classification warm up project using TensorFlow and Python** | Week 6 | **All (individually)** |
| 3 | **Proof of concept completed demonstrating trends between various CT scans**<br>**(Eliminated from plan)** | Week 10 | **Simon** |
| 4 | **Functioning prototype** | Week 12<br>Week 17 | **Nicolas**<br>**Nicolas + Simon** |
| 5 | **Research similar models, evaluate design, discuss possibility of expanding scope**<br>A. **Research indicates nnU-Net to be used as base of segmentation model**<br>B. **Scope expanded to integrate survival analysis model** | Week 14<br>Week 11-14 | **Arie**<br>**Nicolas + Simon** |
| 6 | **Considerable improvement of accuracy** | Week 20 | **Nicolas + Simon** |
| 7 | **Begin GUI development**<br>**Begin integration of pipeline** | Week 20 | **Andrew** |
| 8 | **Finish developing the GUI pipeline for the presentation** | Week 22<br>Week 23 | **Andrew** |
| 9 | Final deliverable | Week 23 | All members |
| 10 | Final project report, presentation | Week 24 | All members |

## 6.2 Project Planning Tools

### Github

Code written for the project was always pushed up to Github to allow other team members to see updates and to keep all code for the project in one place.

### Outlook

Communications with Dr. Simpson and members of Dr. Simpson's lab team as well as teaching assistant Laura Connolly were done through Outlook.

### Microsoft Teams, Zoom, Google Meet

In-person meetings with the project supervisors, teaching assistant and within the team were conducted through video communication.

### Messenger

Messenger was used for quick communications within the team including to schedule meetings and to ask questions.

### Slack

A slack group with several channels was created for the team. Channels included "resources" for links to relevant websites, "feature extraction", "segmentation" and "survival analysis".

# 7.0 Project Reflections

Looking back on the project goals set out in the blueprint, the team met and exceeded original expectations. At that point in the year, the goals we had set out were quite general and needed adjustments (as seen in Section 6.1), however the team was still able to achieve great success in each aspect of the application. The team members worked effectively together, all contributing high quality and equal amounts of work.

There was a steep learning curve with regards to the biomedical aspects of the project. Most of the team has had exposure the machine learning practices, and we are all experienced software developers, however the manipulation and processing of medical data was a new challenge. Simple tasks such as searching through MRI scans and their tumor masks in 3D Slicer proved to be difficult at the beginning of the project, but the team learned and adapted quickly. Now, all team members are well aware of the process of acquiring MRI data, formatting the data correctly, and training a segmentation model with this data.

Medical image analysis and survival predictions are complex fields with very specific knowledge, making it difficult to understand issues, identify problems, and develop solutions. Furthermore, machine learning radiomics is a new field with little research performed, making information on the topic scarce. Now, the team has a thorough understanding of this field, and has the ability to identify problems and design effective solutions.

Another issue that was faced during the project development was the general difficulties of team communication on a remote project such as this one. Although the team used the tools mentioned in Section 6.2 effectively, it remained difficult to sync up with team members to make sure everyone was on the same page since we were never able to meet in person. Also, in order to use the lab provided computer to develop and test the system, team members would need to gain remote access if they were not in possession of the physical computer. This caused difficulties if more than one person was trying to do work at one time, or if the computer was off when there was testing to be done.

Lastly, as was in the team's original plan, it would have been very beneficial to have a graphical user interface as a part of the final product. Considering there was no showcase, the technical aspect of the system was prioritized, however upon reflecting on the project, developing even a simple interface for

a user not familiar with the code to interact with would have wrapped up the project well. As seen in Section 3.5, the current system in a command line interface, which may prove to be difficult to use for certain people.

# 8.0 Conclusion

This project serves as a proof of concept for a pipeline that integrates multiple steps of analysis to supplement physicians with useful information about their patients' tumors and an estimate of their survival. This was shown through the validation of each component's accuracy.

Although improvements can be made on this project with regards to increasing the accuracy of the pipeline's components, the most impactful improvements that could be done relate to the usability of the pipeline. Firstly, data is currently input to the pipeline by copying spreadsheets containing clinical and brain MRI scans to a specific folder and upholding a certain naming scheme. Secondly, the pipeline is currently executed as script directly in a command line interface. These two user-friendliness issues could be solved by the implementation of a GUI application, as mentioned above. This application could prompt the user to browse through directories and select the folder containing the input data. The user could also be guided through the process of using this pipeline in a smooth, low-level manner. This would be essential to putting this pipeline into commercial use, and even more crucial in demonstrating its capabilities when trying to get it approved as a medical device. Because of the heavy workload associated with developing the pipeline itself, the team decided that it could not fulfill this task although it was initially a requirement.

Another future step for this project is getting it approved to be used in the Canadian health care system. As the pipeline is designed to help diagnose and treat a disease, and it is not part of a hardware medical device, it meets the Software as a Medical Device (SaMD) inclusion criteria. Furthermore, since it processes medical images, the pipeline does not meet the first exclusion criterion and therefore should be classified under SaMD guidelines. To determine the classification of the pipeline, the state of the health care situation in which it would be used as well as the significance of the information it provides must be considered. The state of the health care situation would either be "serious" or "critical" because timely action would be required to avoid death or to mitigate irreversible long-term effects. Moreover, the significance of the information the pipeline provides is that it helps to "treat or diagnose". These factors indicate that the pipeline should be classified as a Class III SaMD device.

This pipeline in its entirety cannot be compared to other approaches to the problem, as the idea of combining segmentation and survival prediction into a larger unit has not been done, as mentioned in 5.0 Discussion of Results. Although brain mets segmentation has been done in very similar situations, it has not been used on this novel dataset. Survival prediction in cancer is also not novel but has not previously been executed in literature using a Random Survival Forest utilizing radiomic features.

Overall, the outcome of this project shows that high-resolution tumor segmentations produced by computer-based methods such as the nnU-Net have the potential to improve the accuracy of machine-learning survival prediction models. It also demonstrated the predictive power of radiomic features for brain metastasis patient survival.

# References


[1] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen and K. H. Maier-Hein, "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation," Nature Methods, 1-9, 2020.

[2] H. Ishwaran and et al., "Random Survival Forests," The Annals of Applied Statistics, 2008.

[3] R. A. Patchell, "The management of brain metastases," 2003.

[4] J. B. Posner, "Management of brain metastases," Revue Neurologique, Paris, 1992.

[5] C. S. K. Mark J Ambaugh, "Brain Metastasis," StatPearl, 2020.

[6] K. J. Stelzer, "Epidemiology and prognosis of brain metastases," 2013.

[7] Mayo Clinic, "Brain metastases," [Online].

[8] A. E. Taylor, "Observer error in grading performance status in cancer patients," 1999.

[9] Government of Canada, "Food and Drug Regulations," 18 March 2021. [Online]. Available: https://laws-lois.justice.gc.ca/eng/acts/f-27/FullText.html. [Accessed 11 April 2021].

[10] Government of Canada, "Medical Devices Regulations," 16 December 2019. [Online]. Available: https://laws-lois.justice.gc.ca/eng/regulations/sor-98-282/FullText.html. [Accessed 11 April 2021].

[11] Government of Canada, "Guidance Document - Guidance on the Risk-based Classification System for Non-In Vitro Diagnostic Devices (non-IVDDs)," 12 June 2015. [Online]. Available: https://www.canada.ca/en/health-canada/services/drugs-health-products/medical-devices/application-information/guidance-documents/guidance-document-guidance-risk-based-classification-system-non-vitro-diagnostic.html. [Accessed 11 April 2021].

[12] Government of Canada, "Guidance Document: Software as a Medical Device (SaMD): Definition and Classification," 18 December 2019. [Online]. Available: https://www.canada.ca/en/health-canada/services/drugs-health-products/medical-devices/application-information/guidance-documents/software-medical-device-guidance-document.html#a2.3.1.1.1. [Accessed 11 April 2021].

[13] Government of Canada, "Guidance Document: Software as a Medical Device (SaMD): Classification Examples," 18 December 2019. [Online]. Available: https://www.canada.ca/en/health-canada/services/drugs-health-products/medical-devices/application-information/guidance-documents/software-medical-device-guidance/examples.html#a2. [Accessed 11 April 2021].

[14] Government of Canada, "Guidance Document: Guidance on supporting evidence to be provided for new and amended licence applications for Class III and Class IV medical devices, not including In Vitro Diagnostic Devices (IVDDs)," 5 July 2012. [Online]. Available: https://www.canada.ca/en/health-canada/services/drugs-health-products/medical-devices/application-information/guidance-documents/guidance-document-guidance-

supporting-evidence-provided-new-amended-licence-applications-class-class-medical-devices-including. [Accessed 11 April 2021].

[15] Emergo, "Health Canada Regulatory Approval Process for Medical Devices," 15 May 2019. [Online]. Available: https://www.emergobyul.com/resources/canada-process-chart#:~:text=Documents%20must%20be%20submitted%20in%20English%20or%20French.&text=For%20Class%20I%20devices%2C%20submit,Pay%20Health%20Canada%20fees.. [Accessed 11 April 2021].

[16] Government of Canada, "Licensing a Medical Device in Canada," 24 January 2019. [Online]. Available: https://www.canada.ca/en/health-canada/services/drugs-health-products/public-involvement-consultations/medical-devices/software-medical-device-draft-guidance/requirements.html. [Accessed 11 April 2021].

[17] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," Medical Image Computing and Computer Assisted Intervention Society, 2015.

[18] P. Community, "Pyradiomics," 2016. [Online]. Available: https://pyradiomics.readthedocs.io.

[19] B. Altazi and et Al., "Reproducibility of F18-FDG PET radiomic features for different cervical tumor segmentation methods, gray-level discretization, and reconstruction algorithms," Journal of Applied Clinical Medical Physics, 2017.

[20] I. Shiri and et Al., "The impact of image reconstruction settings on 18F-FDG PET radiomic features: multi-scanner phantom and patient studies," European Radiology, 2017.

[21] M. Vallières and et Al., "Radiomics strategies for risk assessment of tumour failure in head-and-neck cancer," Scientific Reports, 2017.

[22] A. Zwanenburg and e. al., "Assessing robustness of radiomic features by image perturbation," Scientific Reports, 2019.

[23] L. Breiman, "Random forests," Machine Learning 45, 2001.

[24] P. Community, "PySurvival," 2019. [Online]. Available: https://square.github.io/pysurvival/.

[25] S. Bakas and et al, "Identifying the Best Machine Learning Algorithms for Brain Tumor," 2018.

[26] X. Feng, N. Tustison and C. Meyer, "Brain Tumor Segmentation using an Ensemble of 3D," 2018.

[27] B. Chen and et al, "Predicting Survival Duration With MRI Radiomics of Brain Metastases From Non-small Cell Lung Cancer," Frontier in Oncology, 2021.

# Appendix A – Github

The entirety of the code can be found at https://github.com/andrewsimonds14/Capstone

# Appendix B – Code Snippets

Feature Extraction code:

pipeline > extraction.py

```python
import nibabel as nib
import nibabel.processing as nibproc
import os
import SimpleITK as sitk
import radiomics
import csv
import sys

data_path = str(sys.argv[1]) #FSRTCASE_200_0000.nii" # Directory holding NIFTI files images
data_path_mask = str(sys.argv[2]) #FSRTCASE_200.nii # Directory holding NIFTI files masks
row_flag = 1

print('MASK PATH: ', data_path_mask)

with open('radiomicfeatures.csv', mode='w') as csv_file:


    for root, dirs, files in os.walk(data_path, topdown=False):
        for name in sorted(files):
            img_name = os.path.join(root,name)
            mask_name = os.path.join(data_path_mask,name.split('_')[0] + '_' + name.split('_')[1])
            patient_name = name.split('_')[0] + name.split('_')[1]

            # Read the .nii image containing the volume with SimpleITK:
            raw = sitk.ReadImage(img_name)
            mask = sitk.ReadImage(mask_name)

            # Load raw image into variable "raw"
            zsc = sitk.GetArrayFromImage(raw)
            zsc = (zsc - zsc.mean())/zsc.std()
            zsc = sitk.GetImageFromArray(zsc)
            zsc.CopyInformation(raw)
            # Save zsc

            #Setting up the pyradiomics extractor object:
            shift_val = 0.8
            lower, upper = (-0.587027480803037, 5.234361187485263)
            bin_width = (upper-lower)/50
            mean_spacing = (0.9,)*3
            voxelArrayShift=shift_val
            parameters = dict(
                    preCrop=True, # Speed up and reduce memory
                    binWidth=bin_width, # use computed binwidth above # TODO Determine why this causes linalg error
                    correctMask=False,
                    voxelArrayShift=shift_val,
                    resampledPixelSpacing=(mean_spacing[0],)*3
            )

            extractor = radiomics.featureextractor.RadiomicsFeatureExtractor(**parameters)

            # Now for each image, you can extract features via
            features_dict = extractor.execute(zsc, mask)
            usefulfeatures_dict = features_dict.copy()

            for k in features_dict.keys():
                if k.startswith('diagnostics') or isinstance(features_dict[k],list):
                    usefulfeatures_dict.pop(k)

            if row_flag:
                csv_writer = csv.writer(csv_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
                column_names = ['patient']
                column_names.extend(list(usefulfeatures_dict.keys()))
                csv_writer.writerow(column_names)

            row_flag = 0

            csv_writer = csv.writer(csv_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)

            column_values = [patient_name]
            column_values.extend(list(usefulfeatures_dict.values()))
            csv_writer.writerow(column_values)
```

Feature Selection code:

survivalAnalysisCode > featureSelection.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
sns.set()
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.cluster import KMeans

# import data as a dataframe
brainMetsFeaturesRaw = pd.read_csv('./brainMetsMriRadiomicFeatures.csv')

# visualize data
# NOTE: most of visualization done in MATLAB
pd.set_option('display.max_columns', 500)
brainMetsFeaturesRaw.columns
brainMetsFeaturesRaw
brainMetsFeaturesRaw.describe()

# visualize correlations of all features as a heatmap (VERY SLOW)
#corrMatrix = brainMetsFeaturesRaw.corr()
#plt.rcParams['figure.figsize'] = [40, 20]
#fig = sns.heatmap(corrMatrix, annot=True)
#plt.show()

# remove letters from patient number
for i, name in enumerate(brainMetsFeaturesRaw.patient):
    brainMetsFeaturesRaw.patient[i] = name.replace('FSRTCASE', '')

# create patient dataframe and covert patient id to numerical
patients = brainMetsFeaturesRaw['patient']
patients = pd.to_numeric(patients)

# Remove colinear features
X = brainMetsFeaturesRaw.drop(columns = ['patient'])
thresh = 5.0
variables = list(range(X.shape[1]))
dropped = True
while dropped:
    dropped = False
    vif = [variance_inflation_factor(X.iloc[:, variables].values, ix)
        for ix in range(X.iloc[:, variables].shape[1])]
    maxloc = vif.index(max(vif))
    if max(vif) > thresh:
        print('dropping \'' + X.iloc[:, variables].columns[maxloc] +'\' at index: ' + str(maxloc))
        del variables[maxloc]
        dropped = True
print('Remaining variables:')
print(X.columns[variables])
brainMetsSelectedFeatures = X.iloc[:, variables]

# Apply min-max column wise normilization
brainMetsSelectedFeaturesNormalized = (brainMetsSelectedFeatures-brainMetsSelectedFeatures.min())/
                                      (brainMetsSelectedFeatures.max()-brainMetsSelectedFeatures.min())

#Remove low variance features
for cols in brainMetsSelectedFeaturesNormalized.columns:
    if np.var(brainMetsSelectedFeaturesNormalized[cols]) <= 0.1:
        print('Dropping: ' + cols)
        brainMetsSelectedFeaturesFinal = brainMetsSelectedFeaturesNormalized.drop(columns=cols)
brainMetsSelectedFeaturesFinal= brainMetsSelectedFeaturesNormalized

# add patient number back to dataframe
brainMetsSelectedFeaturesFinal['patient'] = patients

# create outcome data dataframe
outcomeData = pd.read_csv('brainMets_features_survivalInDays.csv')

# join outcome and feature tables using patient number as key
predictingData = pd.merge(left=outcomeData, right=brainMetsSelectedFeaturesFinal, left_on='Study', right_on='patient')
predictingData = predictingData.drop(columns = [ 'Study','patient', 'ofMets', 'FSRTcourse', 'Ariacourse', 'Age',
                                                 'DateofdeathorLastFU', 'DateofBrainmetdiagnosis', 'FSRTcompletiondate',
                                                 'GTVVolumecc', 'GTVEqRadiuscm' ])

# visualize correlations as a heatmap
corrMatrix = predictingData.corr()
plt.rcParams['figure.figsize'] = [40, 20]
fig = sns.heatmap(corrMatrix, annot=True)
plt.show()

# save selected features as csv for prediction model
predictingData.to_csv(r'./predictionData.csv', index = False)
```

Random Survival Forest code:

pipeline > randomSurvivalForestModel.py

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
from scipy import stats
from sklearn.model_selection import StratifiedKFold
from pysurvival.models.survival_forest import RandomSurvivalForestModel
from pysurvival.utils.metrics import concordance_index
from pysurvival.utils.display import compare_to_actual
from pysurvival.utils.display import integrated_brier_score

# import selected features from previous script
predictionData = pd.read_csv('./predictionDataForHoldOut.csv')
holdOutSet = pd.read_csv('./holdoutSet.csv')

# create necessary variables for model training
features = (predictionData.drop(columns = [ 'AliveStatus0Dead1Alive','NumDays']).columns).tolist()
X = predictionData.drop(columns = ['AliveStatus0Dead1Alive'])
E = predictionData['AliveStatus0Dead1Alive']

# create temp variables necessary for model parameter selection
Xtemp = X
Etemp = E
featuresTemp = features

# Create holdout set variables
X_hold = holdOutSet[features]
E_hold = holdOutSet['AliveStatus0Dead1Alive']
T_hold = holdOutSet['NumDays']

# setting tested parameters for selection
num_tree=(10, 15, 20, 50, 100)
max_depth=(1, 2, 3, 5, 10, 12, 15)
min_node=(1, 2, 3, 5, 10, 12)

for a in num_tree:
    for b in max_depth:
        for c in min_node:
            cc = []
            kf = StratifiedKFold(n_splits=5, random_state=42, shuffle=True)
            i = 1
            for train_index, test_index in kf.split(Xtemp,Etemp):
                X1_train, X1_test = Xtemp.loc[train_index], Xtemp.loc[test_index]
                X_train, X_test = X1_train[featuresTemp], X1_test[featuresTemp]
                T_train, T_test = X1_train['NumDays'].values, X1_test['NumDays'].values
                E_train, E_test = Etemp.loc[train_index].values, Etemp.loc[test_index].values
                xst = RandomSurvivalForestModel(num_trees=a)
                xst.fit(X_train, T_train, E_train, max_features = 'sqrt', max_depth = b,
                min_node_size = c, num_threads = -1,
                sample_size_pct = 0.63, importance_mode = 'normalized_permutation',
                seed = None, save_memory = False )
                c_index = concordance_index(xst, X_hold, T_hold, E_hold)
                cc.append(c_index)
                i = i+1
            print(a, b, c, mean(cc))

CI = []
IBS = []
best_num_tree = 10
best_depth = 5
best_min_node = 12
k_folds = 5
i = 1
kf=StratifiedKFold(n_splits = k_folds, random_state = 1, shuffle = True)
for train_index, test_index in kf.split(X,E):
    print('\n {} of {}'.format(i,kf.n_splits))
    X1_train, X1_test = X.loc[train_index], X.loc[test_index]
    X_train, X_test = X1_train[features], X1_test[features]
    T_train, T_test = X1_train['NumDays'].values, X1_test['NumDays'].values
    E_train, E_test = E.loc[train_index].values, E.loc[test_index].values
    xst = RandomSurvivalForestModel(num_trees=best_num_tree)
    xst.fit(X_train, T_train, E_train, max_features = 'sqrt', max_depth = best_depth,
        min_node_size = best_min_node, num_threads = -1,
        sample_size_pct = 0.63, importance_mode = 'normalized_permutation',
        seed = None, save_memory=False )
    c_index_train = concordance_index(xst, X_train, T_train, E_train)
    c_index_val = concordance_index(xst, X_test, T_test, E_test)
    c_index_holdout = concordance_index(xst, X_hold, T_hold, E_hold)
    results = compare_to_actual(xst, X_test, T_test, E_test, is_at_risk = True,  figure_size=(16, 6), metrics = ['rmse', 'mean', 'median'])
    ibs_train = integrated_brier_score(xst, X_train, T_train, E_train, t_max=2000, figure_size=(15,5))
    ibs_val = integrated_brier_score(xst, X_test, T_test, E_test, t_max=2000, figure_size=(15,5))
    ibs_holdout = integrated_brier_score(xst, X_hold, T_hold, E_hold, t_max=2000, figure_size=(15,5))
    results = compare_to_actual(xst, X_hold, T_hold, E_hold, is_at_risk = True,  figure_size=(16, 6), metrics = ['rmse', 'mean', 'median'])
    CI.append(c_index_val)
    IBS.append(ibs_val)
    print('C-index validation: {:.2f}'.format(c_index_val))
    print('C-index train: {:.2f}'.format(c_index_train))
    print('C-index holdout: {:.2f}'.format(c_index_holdout))
    print('\nIBS validation: {:.2f}'.format(ibs_val))
    print('IBS train: {:.2f}'.format(ibs_train))
    print('IBS holdout: {:.2f}'.format(ibs_holdout))
    i = i + 1
print('C val mean: {:.2f}'.format(mean(CI)))
print('IBS val mean: {:.2f}'.format(mean(IBS)))
```

Bash Pipeline Script code:

pipeline > capstone-pipeline

```
#!/bin/bash
# Run pipeline on desired image

#Current test path: /home/lab/capstoneGit/Capstone/pipeline/inputTest

#Pre requisites:
#    Need to have a single scan to test on in the format of FSRTCASE_XXX.nii.gz and need the directory where it is.
#    Need a trained 3D nnUNet model to use to predict the mask of the scan.
#    Need a trained survival model that must be saved in a zip folder in this directory as survival_model.zip
#Data Requirements:
#    brainMets patient data for survival in days
#    raw and preprocessed prediction data to normalize and extract features from our new single feature set
#

# Change working directory to where we wanna be in case it's being run from somewhere else
cd /home/lab/capstoneGit/Capstone/pipeline

# SEGMENTATION

# Ask user for NIFTY file path
echo 'Enter folder where desired image is (FSRTCASE_XXX.nii.gz)'
read inputFolder
mkdir -p $inputFolder/results
mkdir -p $inputFolder/resultsDump
echo -e '---START Segmentation---\n'

nnUNet_predict -i $inputFolder -o $inputFolder/resultsDump -t 502 -m 3d_fullres

# New NIFTY saved in $inputFolder/../outputTest, we need to access it here to open in slicer for viewing if possible
# Open new terminal to open slicer and display segmentation after getting file paths
inputFile=$(ls -p $inputFolder | grep -v / ) # Parse out folder names from ls command
outputFile=$(ls $inputFolder/resultsDump | head -n1 | cut -d " " -f1) # Grab .nii.gz file
mv /home/lab/capstoneGit/Capstone/pipeline/inputTest/resultsDump/$outputFile /home/lab/capstoneGit/Capstone/pipeline/inputTest/results
inputFilePath="$inputFolder/$inputFile"
outputFilePath="$inputFolder/results/$outputFile"

gnome-terminal -- /home/lab/Slicer-4.11.20210226-linux-amd64/Slicer --python-script display_scene.py $inputFilePath $outputFilePath
echo -e '---FINISH Segmentation---\n'

# FEATURE EXTRACTION
echo -e '---START Feature Extraction---\n'
python extraction.py $inputFolder $inputFolder/results/
echo -e '---FINISH Feature Extraction---\n'

# Normalize feature and remove colinear features in csv
echo -e '---START Feature Selection---\n'
python featureSelection.py
echo -e '---FINISH Feature Selection---\n'

# SURVIVAL PREDICTION
echo -e '---START Survival Prediction---\n'
# Give pre-processed csv to survival prediction model and output survival prediction graph
# We need a survival_model.zip already saved in this directory from the survival forest training file (Done as of now but may need to update)
python randomSurvivalForest.py

echo -e '---FINISH Survival Prediction---\n'
```