**Medical Imaging Matching (1): Milestone Requirement**

Team: Alex Rogov, Emily Medema, Marieke Gutter-Spence, Niklas Tecklenburg

## Project Information:

*Project Description:*

The Medical Imaging Matching project is aimed at developing an image matching algorithm for BC Cancer. BC Cancer has a mammogram database containing hundreds of thousands of images from women all over BC. When collecting these data BC Cancer noticed that there are several images, where the identifying data match very closely but not a hundred percent. For example, name and birthday match but the health number slightly varies. As of now, these cases are resolved manually by having a medical specialist determine if the mammograms are from the same person. The goal of this project is to come up with an AI algorithm that can speed up the process, reduce the reliance on medical specialists, and help BC Cancer eliminate mismatched mammograms easier. Eliminating mismatches ensures the accuracy of further medical research, specifically in the use of AI, as unknowing duplicates, mismatches, or missing information within a dataset can cause bias. In the following report, we would like to walk you through our project requirements, architecture, tech stack, and testing strategy as well as questions that have been raised.

*User Groups:*

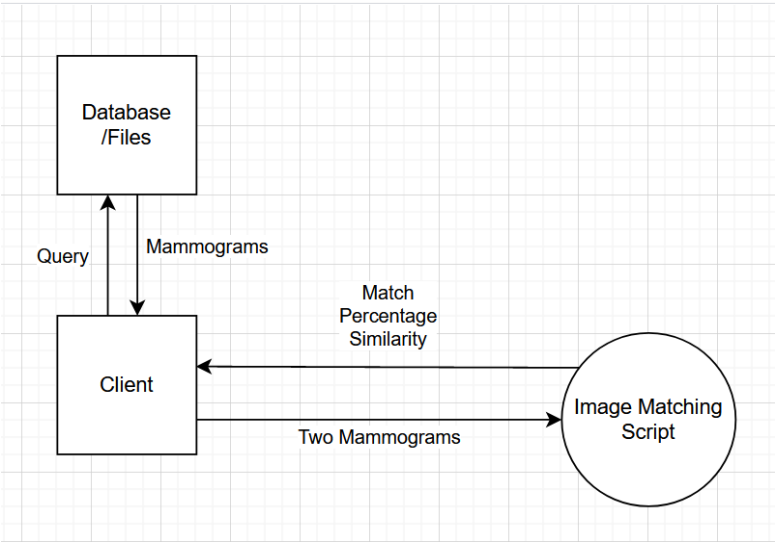
The main user group for the project is BC Cancer Researchers. They will be using the model we develop to curate datasets and go through the mismatch they have found. The result expected from BC Cancer researchers is a script that they can run that returns the similarity index of two images. If the results are acceptable, the script will eventually be publicly available for anyone who wants to curate a large set of images from an institution. Therefore, another possible user group is anyone using the open-source code after it is published. They would use the model in the same way via command line, but they would not be required to run it within BC Cancer's network.

## System Architecture:

### *Data Flow Diagrams:*

*DFD Level 0:*

The client will query and receive the specified mammograms which are stored in the File System. The client will then send at least two mammograms to the Image Matching Script. This Image Matching script will then return a match similarity percentage to the client.

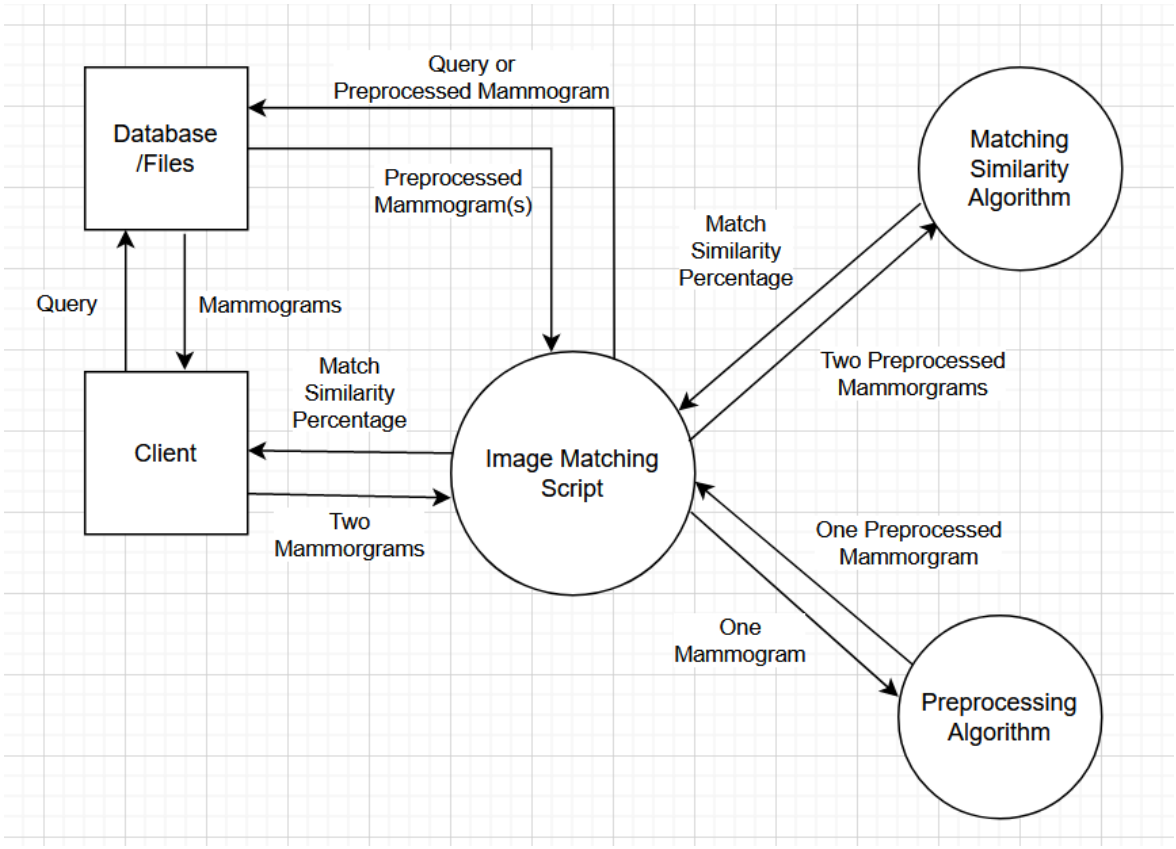



*DFD Level 1*:

The client will query and receive the specified mammograms which are stored in the File System.

The client sends at least two mammograms to the Image Matching Script component. It will then query the Database/Files to see if the preprocessed version of the mammogram exists. If it does, it will directly send them to the Matching Similarity Algorithm component. If not, This script will then send one mammogram to the Preprocessing Algorithm component.

The Preprocessing Algorithm component then returns a preprocessed mammogram to the Image Matching Script. The Image Matching Script will then save it to the Database/File System. Once the Image Matching script has two preprocessed mammograms, it will send them to the Matching Similarity Algorithm component to calculate a matching similarity percentage. An overall percentage will be returned to the Image Matching Script component. Finally, the Image Matching Script will then return this match similarity percentage to the client.




***Components*:**

*Client* - Completed by Peer Testing 1

Clients will see a match percentage response to their mammogram input via the command line. If time is given, a UI can be created for a better user experience, this specific feature would be planned for the final milestone.

*Preprocessing Algorithm* - Completed by Peer Testing 1

This component “cleans” and prepares the mammogram for matching. This would include cutting the file and highlighting/tagging important sections of the mammogram. This will improve the speed and accuracy of the similarity match.

*Image Matching Script* - Completed by Peer Testing 2

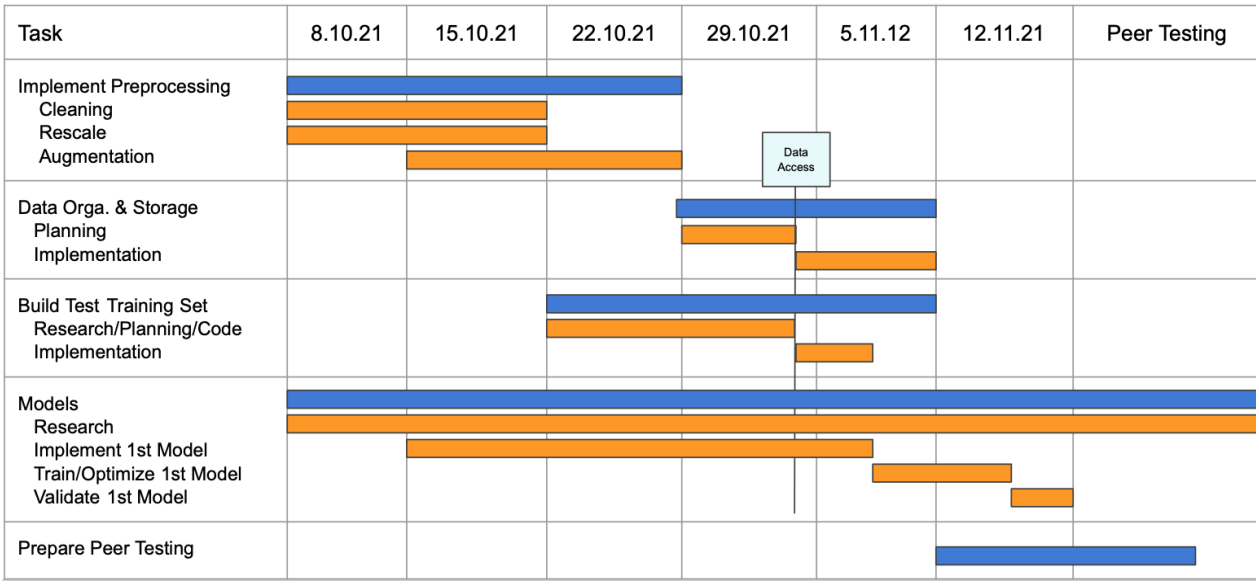
This component holds the logic of the script. It will take in the mammograms and check if the mammograms have already been preprocessed. If they have, it will collect the already preprocessed mammograms from the file structure. If they have not, it will send them for preprocessing. Once enough preprocessed mammograms are collected, the component will send the mammograms to a separate component to calculate a match percentage. Once a percentage is returned, the percentage will be returned to the client. If we have time, a possible feature we could implement would be storing the resultant vectors for all mammograms we

have processed so we can compare each new mammogram to the stored vectors.

*Matching Similarity Algorithm* - Completed by Final Milestone

This component takes two preprocessed components and compares the similarities of the image. Based on built-in parameters of comparison, it will return a similarity match percentage to the main “image matching script” component. As well, many smaller tests will occur in which mammograms will be altered and scored. These values will come together to form an overall score/percentage which is returned. Once this algorithm is returning a similarity index that is reasonable and accurate, we will be focusing on refining the algorithm and exploring hyperparameters that will improve the overall accuracy and precision of the model.

**Time Plan to Peer Testing Milestone 1:**




**Requirements:**

***Functional Requirements:***

*“The model must return a similarity score for mammograms with matching accuracy of +70%”*
The main requirement of this project is to deliver a model that can deliver reliable similarity scores for pairs of mammograms, in order to verify mismatches in the data set. In particular, it should take a CSV file, with the pre-identified potential mismatches and return a CSV, containing the similarity score. This task is split down into 4 parts. The preprocessing, the Data organisation, the test-training set builder, and the model itself.

*Preprocessing:*
*“The preprocessing must come up with a consistent set of images with same dimensions”*
To feed the model with proper, consistent input we are going to implement several steps of preprocessing. This firstly includes data cleaning, with standardizing of the pixels to [0,1]. Secondly, we are going to resize the images with a 4x4 filter to reduce the size by 4, as the

images have a high resolution as of now. Finally, we are going to apply Data Augmentation on the data set, to expand the database of available duplicates, as this is the limiting factor. Here we are considering rotations, shiftings, changing the brightness, and other augmenters.

*Data Organisation / Storage:*
*“Data must be organised, so the data is easily accessible for learning”*
In the client meetings, we were informed that the current way the images are accessed might be a bit tedious for our use cases. Therefore we are planning to come up with an easier structure for our use case to store the preprocessed images.

*Test/Training Set Builder:*
*“Test/Training sets must represent the actual use case and support fast learning progress”*
As the final result highly depends on the learning process of the model and the learning highly depends on the data that are used for it, we are planning to invest some more time into this. Since we are comparing two images per mismatch the training and test dataset selection is incredibly important. We need to make sure we have enough mismatches and enough matches to train our model for both. If we have too many mismatches our model won't get used to matches and the processing of one might be biased or take too long.
We are planning to set up positive examples where we are trying to find mammograms of the same person that are as different as possible and find negative examples that are similar to the one we are comparing it to. We are hoping that this improves the time it takes to train significantly.

*Models:*
*“Models must return proper similarity scores and must be trained in reasonable time”*
The model has to take two sets of mammograms or just two mammograms, as some sets are incomplete, and return a similarity score, to assess whether the mammograms are from the same patient. In order to come up with a similarity score, we are planning to test several image similarity algorithms - such as Euclidean Distance Model, RootSIFT, MSE/SSIM, and network models like Siamese Networks as well as other DNNs for image recognition. For the later, more complex models, we will be facing the problem that it will take a very long time and lots of data to train these networks from scratch. Therefore, we are planning to implement transfer learning, to profit from existing models that have already been trained on big data sets. The desired output of the networks would be an n-dimensional vector (n about 128) for each mammogram. These could then be compared with each other using euclidean distance or even another neural net to decide whether it is the same patient or not. To train this model, we would like to look into different cost functions like euclidean distance or triplet loss, based on euclidean distance, to speed up the learning process. In order to improve the performance of our model, we are also planning to optimize several hyperparameters like the learning rate, drop-out rate, momentum, batch size, and others.

Table 1: Pros and Cons of Each Model

| **Model** | |
|---|---|
| Euclidean Distance | ● Use when calculating the distance between two rows of data containing numerical values, for example, two pixels<br>● Measure the ‘similarity’ between two vectors |
| RootSIFT | ● Used to describe and detect features within digital images, through locating key points and descriptors, often used for object recognition |
| Siamese Networks | ● Very Powerful networks used in facial recognition<br>● Requires more training time than other network models<br>● Finds similarity through comparison of feature vectors |
| MSE/SSIM | ● Simple technique used in order to find similarity of two images<br>● SSIM takes texture into account<br>● MSE will calculate mean square error (not highly indicative of similarity), SSIM will look for the similarity in the pixels of two images |
| Convolutional Neural Net | ● Widely used in the field of Computer Vision, to analyse images and condense information out of them<br>● Easy to build with eg. tensorflow<br>● Lots of documentation |
| FaceNET | ● Lots of documentation, and great performance<br>● Learns face-specific features especially in deeper layers, which are not applicable for us |

We are planning to try a Convolutional Neural Net with 4 Convolutional and 4 pooling layers, connected to a Neural Net first, to experiment with TensorFlow and start things off. We are not expecting really good results from this, but think that it is a good approach to set everything up and see whether every component of the system works before we start trying more complex models and experiment with the other components.

***Non-functional Requirements:***

Regarding the non-functional requirements, the client is flexible. The model should be written in python and able to return similarity scores in a reasonable time. Additionally, the confidentiality of the data is really important to the client. In particular, the mammography data and patient information cannot leave the BC Cancer network. We will also need to ensure that our code is maintainable and well documented, as the development is expected to be an ongoing process that will extend past our 8 months of working on the model.

**Technology Stack**:

***Language***: Python, required by the client.

***Libraries***:

Table 2: Pros and Cons of Each Library

| **Option** | Pros | Cons |
|---|---|---|
| TensorFlow, Keras | ● Well documented<br>● Easy to deploy<br>● Tensor board offers good visualization<br>● Great option for high-level model development | ● Models sometimes feel like they are behind a wall,<br>● Hard to find errors, making it difficult to debug<br>● Comes with a steep learning curve |
| Pytorch | ● Easy to debug, and parallelize<br>● Very dynamic implementation<br>● More Python friendly, simple syntax making easy to learn | ● Not as extensive as TensorFlow<br>● Limited visualization interface<br>● Newer and not as widely known |
| SciKit Learn | ● Very user friendly<br>● Well documented, open-source<br>● Deliberately limited scope<br>● Efficient for predictive data analysis | ● Less extensive, than other libraries for example TensorFlow<br>● Not specifically useful for deep-learning<br>● Not great for in-depth projects |
| OpenCV | ● Extensive useability<br>● Includes a multitude of image processing | ● Image processing tool, whereas TensorFlow for |

| | | |
|---|---|---|
| | functions<br>● Enables you to easily manipulate pixels<br>● Includes very good computer vision algorithms | example is a machine learning tool<br>● Limitations when it comes to face detection<br>● Written natively in c++ |

Based on the pros and cons list surrounding each of the different libraries and frameworks we have researched, we are currently going to be using TensorFlow. We have chosen to go with TensorFlow since it appears to be the best option for high-level modeling due to its extensiveness. TensorFlow also offers great documentation and visualization, making it easy to deploy. Although TensorFlow may come with a steep learning curve and difficulties debugging, the pros outweigh these cons. Besides Tensorflow we are planning to use OpenCV for several preprocessing tasks.

**Testing:**

First and foremost, we will be implementing unit testing throughout our project. Each method in the model(s) as well as within our preprocessing script will have a unit test with which it is associated. All of our unit tests will utilize the built-in python testing library `unittest`. This testing will ensure that all of our methods are functioning and returning the values we expect.

In addition to unit testing, we will also be performing integration testing on our codebase. This is an essential element to our testing as there are multiple scripts and interesting data requirements within this project. We need to ensure that the model script and the preprocessing script are working together properly; that the data is accessed safely and securely; and that the data is not leaving BC Cancer's network.

Likewise, we will also be performing regression testing. This will ensure we are not breaking our code whenever we add a new piece of code. To do this, we will utilize the `pytest-regtest` python plugin to check that the format of the output remains the same as before (ie. no errors).

Lastly, we will be performing functional testing. It is essential to our project that we select the correct images in accordance with the mismatches we input and that we return a similarity index that makes sense. Functional testing will help us ensure that this is true.

***CI/CD:***

We will also be implementing Continuous Integration/Continuous Development procedures within our project to ensure our code does not break when we make changes. We will run premade test cases on each model we develop. These test cases include a set of mammograms in which we already know the results (mammograms match or not). If the success rate is too low, we will know we must improve the model. We will also configure static

analysis, specifically the pre-commit framework so that our code is consistently formatted and linted. This will run via git-hooks and will catch things like whitespace and common issues.

*CI Workflow:*
On Github we will be following the workflow detailed below to ensure CI:

1. We will be creating issues for each feature, separated into the task, chore, and exploration labels depending on the work needed for each issue.
    a. Tasks: A task is a step in the process of completing a feature. A feature is completed when underlying tasks are completed. Usually, tasks will include many aspects such as coding, testing, and documentation.
    b. Chores: A chore is an item that provides value to the team rather than the client. Examples include implementing continuous integration, linting the project, or refactoring a codebase.
    c. Explorations: Exploration is required when breaking a feature into tasks is not possible, either because you are unclear what needs to be done or unclear how it should be accomplished. This is the label used when additional research needs to be done for us to determine how to move forward.
2. One person on the team, unless otherwise specified, will be assigned at least two issues per sprint to work on.
3. Once the assignee determines the work to be done, they will pull and merge any changes that occurred on the main branch into their feature and they will run the test cases that were created. Once those tests pass, they will create a pull request.
4. The pull request will be reviewed by at least two other team members. All comments and concerns must be addressed and the assignee must be given the explicit go-ahead to merge.
5. After the merging of a PR, the main codebase will be subjected to the premade test cases and everyone will be notified that they need to pull to keep their local repository up to date. **Note:** When the PR is merged the issue is automatically closed.

**Questions from Presentation:**

*Do you have any specific facial recognition models in mind for your project?*

We are currently looking into FaceNet as it is one of the most popular models with really good performance and wide documentation, but we are unsure if this will work for our data as it focuses on face-specific attributes, especially in the deeper layers, which may make it not applicable.

*Do you have any concerns about the efficiency of this program?*

While we will be attempting to make this as efficient as possible, our client is more focused on the accuracy of the results rather than the time it takes to get the results.

*Do you have an implementation plan for the project yet?*

We will be following the Gantt chart. We will begin working on the preprocessing algorithm. Then we will create the Similarity Matching Algorithm. Finally, we will create the image matching script which holds the logic, manages the other two components, and communicates with the client. At this point, our focus will be to refine the algorithms to increase its performance and accuracy.

*Do you have plans to host your software? Will it remain local software?*
It will most likely remain local due to the heavy security and hosting requirements that come with having to upload medical data. Remaining local on the BC Cancer network is the better option currently.

*You say that all you're building is an algorithm that the end user will be interfacing with through a command line. And then you say your final milestone is a 'possible ui'. A doctor or nurse won't be able to type command line and pass in images to this thing. Does the client have a trained IT person do this task? Right now you say that they have a professional come in, and manually compare scans to identify matches. But if you give them this algorithm, won't they still have to have an IT person come in to use the command line/ or GUI that you may make?*

We are creating this project for BC Cancer researchers, of which there may be nurses and doctors but they are not a group for which we are developing. According to our client, all BC Cancer researchers that will be using this script will have the knowledge required to use the command line or terminal to run the script. We will also be providing documentation detailing how to run the project. So no, the intended user group will be able to run the script.

*For your non-functional requirements, under security, you say the data has to stay in the BC cancer network. The client has to give you guys real training and testing sets. Isn't that a breach of security? What steps have you and the client taken to ensure security during development?*

We will be accessing the BC Cancer network remotely when using the provided test and training sets that must remain in the BC Cancer network. This in conjunction with the ethics course and confidentiality agreement we signed ensures that there is no breach of security. We will also be using open-sourced data to test on our local machines while we wait to be granted access to the data.

*How do you plan to run your continuous integration testing? This seems like a fairly comprehensive system so I'm wondering if you're planning to run it through github actions or another solution like Jenkins, CircleCI, or something else?*

We were planning on running it through GitHub actions, but we will take a look at your suggestions.

*The video says that the unit test library will be used to test for accessing the data safely. Will the team attempt any form of intrusion testing?*

We had not considered intrusion testing. We are not sure if the testing framework would fit our project as the script/models will be run on BC Cancer's servers, there should be no additional legwork for us required. However, we will keep that in mind in case the requirements or the hosting of the script changes.

*Have you identified the weak points that you will need to focus on in order to ensure security of the system, and if so how are you going to address them?*

As we will be running the script on BC Cancer's server we don't anticipate security problems in regards to data privacy. However, we will be ensuring that all the models and libraries we use are open-sourced and can be used without allowing access to their system.

*How do you plan on getting a test set of images?*

The official training and test set will be provided by BC Cancer, while we are waiting for that we will be using an open-source dataset containing mammograms.

*How do you plan to store and access the processed data?*

We will be storing processed data in a "processed" folder on their file system. This way we can avoid reprocessing images. As we will be reading in mammograms from the file structure, we will first check if it has already been processed. If it has, we can avoid processing it again, if not we will continue with processing the image.

*There are four types of results, namely True Positive, False Positive, True Negative, and False Negative. How do you define the 70-90 percent accuracy considering the four results?*

We will be using the standard accuracy, precision, recall, and F1 Score to calculate the efficacy of our model. The calculations are as follows:

$$Accuracy = \frac{number\ of\ correctly\ classified\ test\ sample}{total\ number\ of\ test\ sample}$$

In terms of the question the equation would be as follows:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

However, this number can be misleading or not show the full picture, so we will also be using the equations below to get a robust understanding of the model's performance.

$$Precision = \frac{number\ of\ relevant\ results\ that\ were\ ranked\ or\ retrieved}{number\ of\ retrieved\ results}$$

This can be defined as of those predicted to be true, proportion that are true. In the terms of the question the calculation would be:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{number\ of\ relevant\ results\ that\ were\ ranked\ or\ retrieved}{number\ of\ relevant\ results}$$

This can be defined as the actual true results, proportion predicted to be true. In the terms of the question the equation would look like this:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

We will also look at specificity, which is of the actual false results, proportion predicted false:

$$Specificity = \frac{True\ Negative}{False\ Positive + True\ Negative}$$

$$F1Score = 2 * \frac{precision * recall}{precision + recall}$$

As it would be more costly for us to miss out on a match, we will be focusing mostly on precision, accuracy, and/or recall rather than specificity (ie. maximizing precision). As precision and recall are essentially similar measures, we will be focusing on the F1Score which is a harmonic mean between precision and recall. Research has shown that F1Scores are generally more reliable than misclassification rates (especially for unbalanced datasets, which is what ours is), therefore that is what we will be focusing on maximizing and what will be informing our models' performance.