

# Low-Level Design Document

## Knee MRI Classification

Revision Number: 1.0

**Imran**  
**Knee MRI Intern Team**

## Document version control

Date Issued	Version	Description	Author
09-06-2021	1.0	First copy of LLD	Imran

## Contents

Knee MRI Classification	1
1.1. Why this Low-Level Design Document?	5
1.2. Scope	5
2. Project Description	6
2.1. Introduction	6
2.2 High-Level Objectives	6
3. Work flow Overall	12
3.3 Details of DL Model	13
3.4.1 Hyper Parameter Tuning	15
2.6.3 Sample GUI	23
3. References	24

## Abstract

Magnetic resonance imaging (MRI) of the knee is the preferred method for diagnosing knee injuries. However, interpretation of knee MRI is time-intensive and subject to diagnostic error and variability. An automated system for interpreting knee MRI could prioritize high-risk patients and assist clinicians in making diagnoses. Deep learning methods, in being able to automatically learn layers of features, are well suited for modelling the complex relationships between medical images and their interpretations.

## 1. Introduction

### 1.1. Why this Low-Level Design Document?

The technical design document gives a design blueprint of the Knee MRI Classification project. This document communicates the technical details of the solution proposed.

Once agreed as the basis for the building of the project, the flowchart and assumptions will be used as a platform from which the solution will be designed.

Note: All the code will be written in python version 3.7

### 1.2. Scope

This document captures the different workflows involved to build the solution, exceptions in the workflows and any assumptions that have been considered. Changes to this business process may constitute a request for change and will be subject to the agreed agility program change procedures.

## 2. Project Description

### 2.1. Introduction

The knee is one of the largest and most complex joints in the body. The knee joins the thigh bone (femur) to the shin bone (tibia). The smaller bone that runs alongside the tibia (fibula) and the kneecap (patella) are the other bones that make the knee joint. Tendons connect the knee bones to the leg muscles that move the knee joint. Ligaments join the knee bones and provide stability to the knee:

- Anterior Cruciate Ligament(ACL) prevents the femur from sliding backwards on the tibia.
- Posterior Cruciate Ligament(PCL) prevents the femur from sliding forward on the tibia
- Medial and Lateral Collateral Ligaments prevent the femur from sliding side to side.
- Two C-shaped pieces of cartilage called the medial and lateral menisci to act as shock absorbers between the femur and tibia.

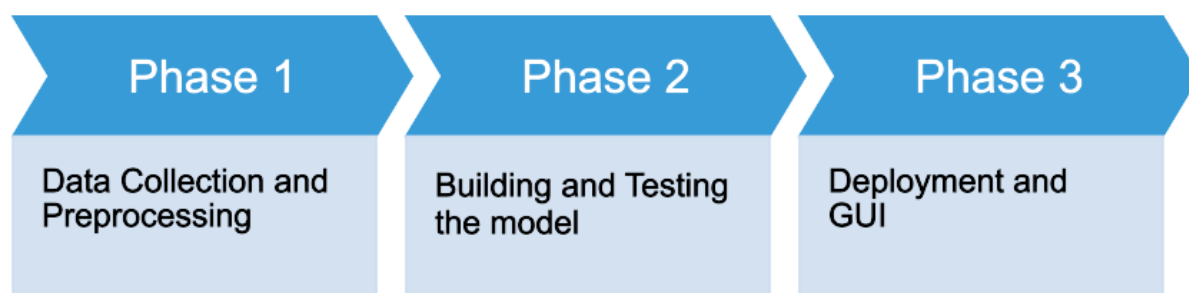
Magnetic Resonance Imaging (MRI) of the knee uses a powerful magnetic field, radio waves and a computer to produce detailed pictures of the structures within the knee joint. It is typically used to help diagnose or evaluate pain, weakness, swelling or bleeding in and around the joint and it can help determine whether you require surgery. In order to better confront the ever-growing workload of musculoskeletal (MSK) radiologists, automated tools for patients' triage are becoming a real need, reducing delays in the reading of pathological cases.

### 2.2 High-Level Objectives

1. To enable reading/loading of MRI Image by the user.
2. To pre-process the MRI image for enhanced quality before further processing.
3. To classify the image into normal and abnormal using Deep Learning Techniques
4. If abnormal, predicting the type and location of the injury.
5. To perform statistical analysis of the results and display them in a user-friendly manner.

## Low Level Design (LLD)

The project can be divided into multiple phases:



### 2.3 Data Collection and Preprocessing

The dataset employed in this research is MRNet V1 taken from Stanford ML Group ([link](#)) [1]. The same can be downloaded through Kaggle ([link](#)) [2]. It consists of a total of 1370 images divided into three classes namely - Abnormal exams (80.6%), ACL tears (23.3%), and Meniscal tears (37.1%). An anterior cruciate ligament (ACL) tear is usually a complete tear of the ligament where it has been split into two pieces resulting in an unstable knee. A meniscal tear is a tear of the cartilage that provides a cushion between the bones in the leg. An abnormal exam is a catch-all classification for these and other knee injuries.

The dataset is split into 3 sets as given in Table 1. As of now, the test set isn't freely accessible and is utilized for scoring models submitted for the competition.

Table 1

Training set	1130 exams	1088 patients	Available
Validation set	120 exams	111 patients	Available
Test set	120 exams	113 patients	Not Available

The dataset consists of a train and validation set, including csv files for the conclusions of the three conditions we're keen on. Up until this point, we've investigated that the train and validation sets are additionally divided into three planes - axial, sagittal and coronal - and each exam series is provided as a .npy file.

Some sample images are shown in Figure 1 to visualize the knee anatomy present in all the three planes.

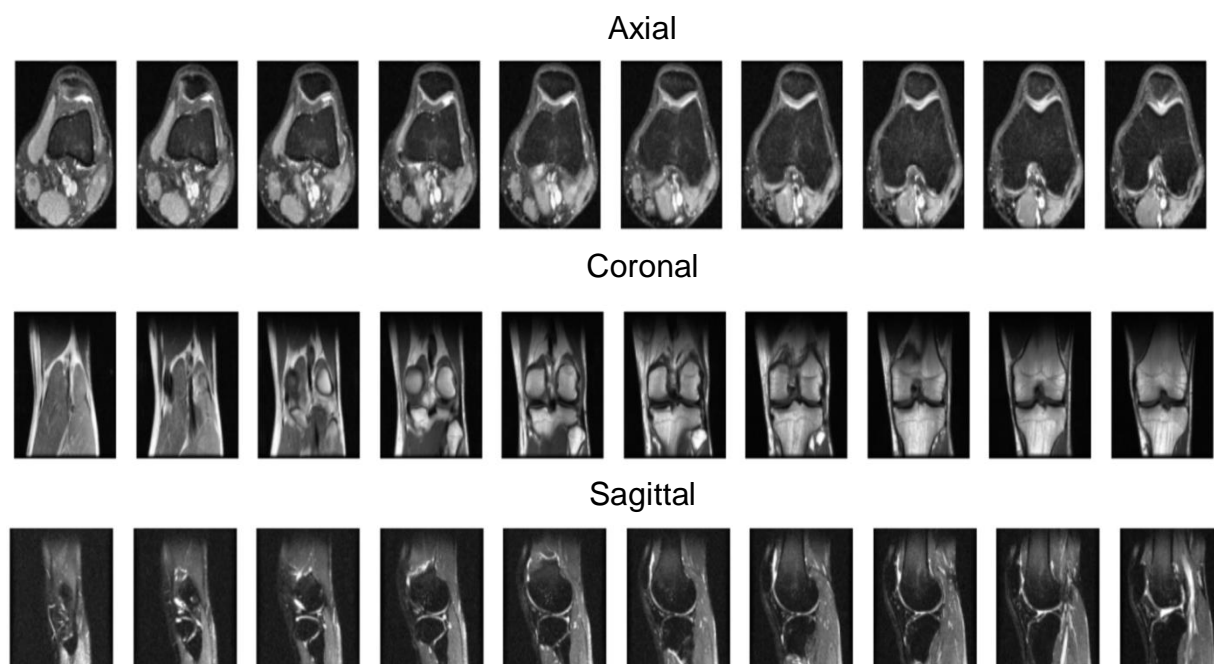


Figure 1

The flowchart given in Figure 2 showing the hierarchical distribution of images.

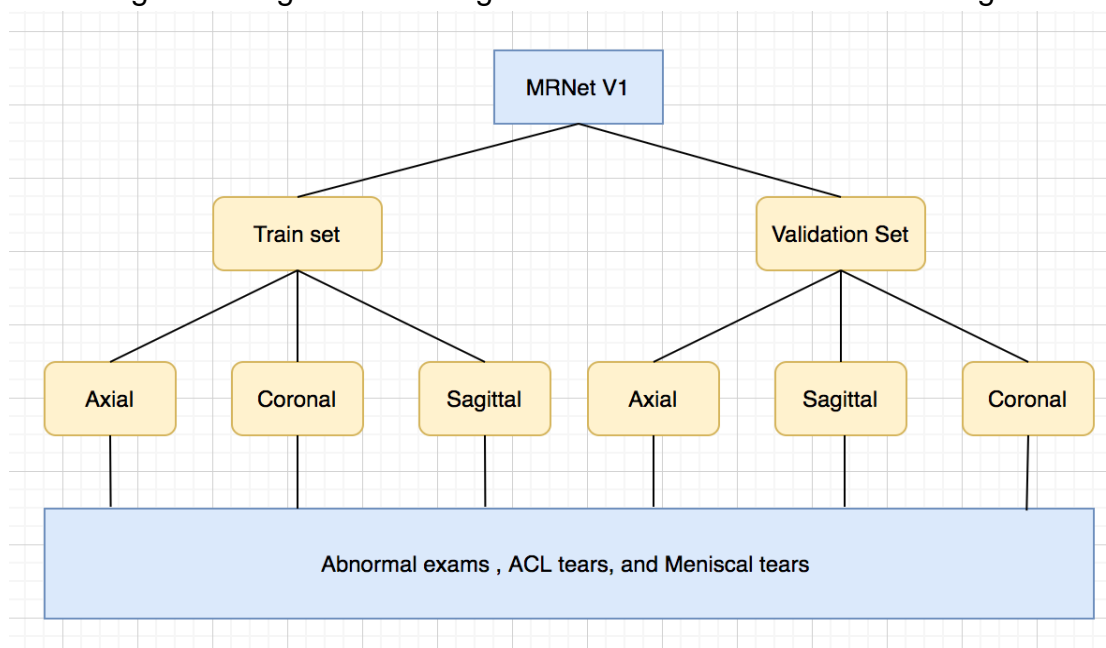


Figure 2

A total of six csv files including training and validation set are also provided with the dataset for each condition i.e abnormal, meniscus and ACL tears. The description of the given csv files are shown in Table 2.



Table 2

CSV File	Description	Shape
train-abnormal.csv	Contains labels of abnormal exams present in training set	(1130,2)
train-acl.csv	Contains labels of ACL tears exams present in training set	(1130,2)
train-meniscus.csv	Contains labels of meniscus exams present in training set	(1130,2)
valid-abnormal.csv	Contains labels of abnormal exams present in validation set	(120,2)
valid-acl.csv	Contains labels of ACL tears exams present in validation set	(120,2)
valid-meniscus.csv	Contains labels of meniscus exams present in validation set	(120, 2)

One point to note is that each exam is divided into several images. For example, as we can see in figure 3 that a sample exam in coronal plane contains 40 images.

```
train_axial_exams = glob(f'{train_dir}/axial/*.npy')[:5]
train_coronal_exams = glob(f'{train_dir}/coronal/*.npy')[:5]
train_sagittal_exams = glob(f'{train_dir}/sagittal/*.npy')[:5]
for exam in train_coronal_exams:
    series = np.load(exam)
    print(series.shape)
```

```
(19, 256, 256)
(40, 256, 256)
(18, 256, 256)
(18, 256, 256)
(21, 256, 256)
```

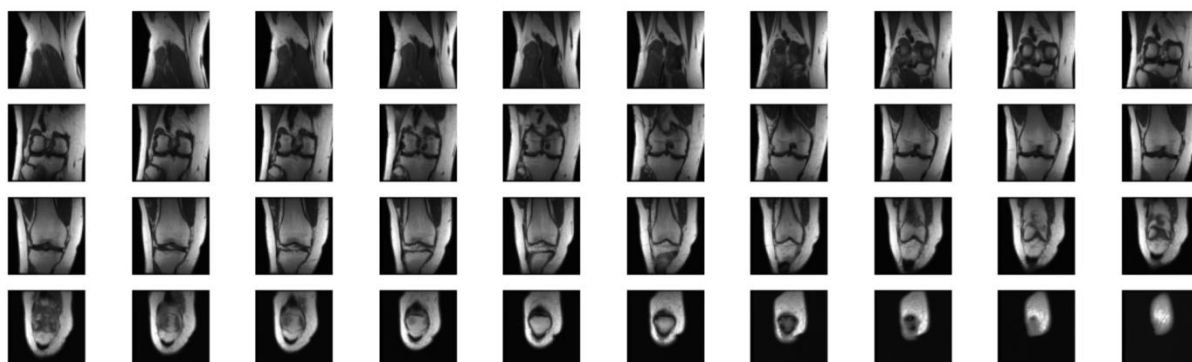


Figure 3

As indicated by the paper, the quantity of arrangement in each exam fluctuates from 17 to 61 (mean 31.48, SD 7.07). It's imperative to take a look at the appropriations for train and validation dataset. The most extreme and least number of slices for each plane is given in Table 3. For each plane the dispersions are fairly comparative between the training and validation set. In most cases, the quantity of pictures falls in the middle of 20 and 40.

Table 3

<u>Set</u>	<u>Plane</u>	<u>Minimum</u>	<u>Maximum</u>
Training set	Axial	19	61
	Coronal	17	58
	Sagittal	17	51
Validation set	Axial	20	52
	Coronal	17	48
	Sagittal	21	45

Table 4 and 5 shows the distribution of classes for training and validation exams. It's important to note that there are inconsistencies between the train and validation dataset as far as the recurrence of co-event of these conditions is considered. For instance, the distribution for training and validation set is 38.3% and 16.7% separately for Abnormal class with no ACL tear and no meniscal tear.

Table 4

Set	Abnormal	ACL	Meniscus	Exams	Percentage
Training	0	0	0	217	19.2
	1	0	0	433	38.3
	1	0	1	272	24.1
	1	1	0	83	7.3
	1	1	1	125	11.1
				1130	100%

Table 5

Set	Abnormal	ACL	Meniscus	Exams	Percentage
Validation	0	0	0	25	20.8
	1	0	0	20	16.7
	1	0	1	21	17.5
	1	1	0	23	19.2
	1	1	1	31	25.8
				120	100%

The dataset can be divided into training, validation, and testing phases. For this, a standard train test split method can be employed. In this project, the validation set is already given which can be considered as a testing set. For training of images, the validation set is particularly required. We can transfer 120 exams from the train set using various sampling methods to ensure equal examples of abnormal, ACL tear and meniscal tear in the validation set (Table 6).

Table 6

Training set	1010 exams
Validation set	120 exams
Testing set	120 exams

Various Data Augmentation Techniques can be applied to enhance the accuracy. As recommended in [5], we need to be very careful using augmentations. Data augmentation even at a low scale can be a tricky aspect which may hamper the performance of the network. However, if done properly can give some boost to the performance. Image Data Generator [3] can be used from the keraspreprocessing package to generate batches of image data with real-time data augmentation. Some common methods like horizontal flip, vertical flip, rescale, zoom range, rotation range, width and height shift range can be included to augment the training set.

It's worth considering that we have to perform data augmentation only on the training set i.e. 1010 exams (after train test split). This is useful to prevent data leakage problems.

In this project, we can also utilize imgaug library [4] to augment the images. This library has strong as well as basic augmentation pipelines that help generate images according to the project goals.

## 2.4 Building and Testing Model

## 3. Work flow Overall

The figure below shows the detailed flow of the proposed project.

### 3.1 Application Flow

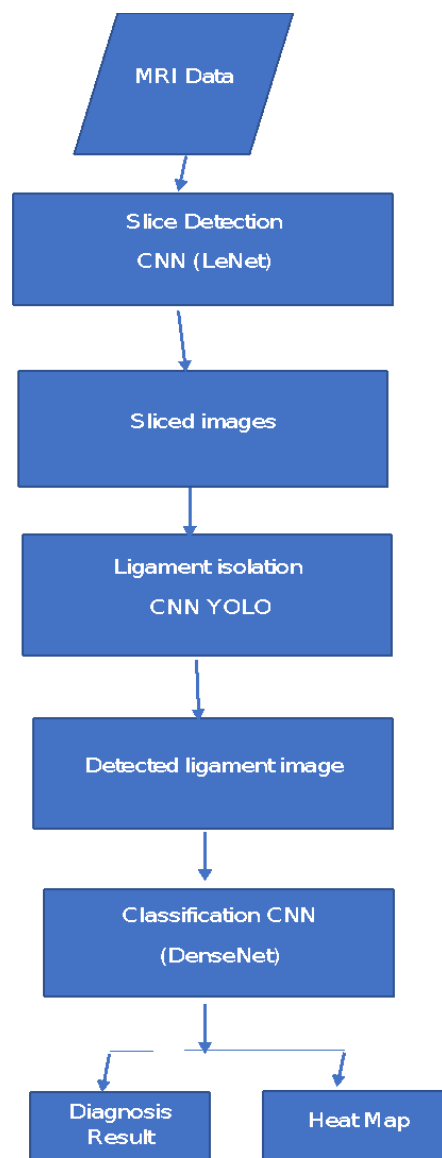
### 3.2 Exception Scenario Overall

Step	Exception	Mitigation

### 3.3 Details of DL Model

The DL model is referred from a Research Article published in the Radiology Artificial Intell Journal. The article is titled as Fully Automated Diagnosis of Anterior Cruciate Ligament Tears on Knee MR Images by Using Deep Learning[\[6\]](#).

Proposed flow as in the following flowchart

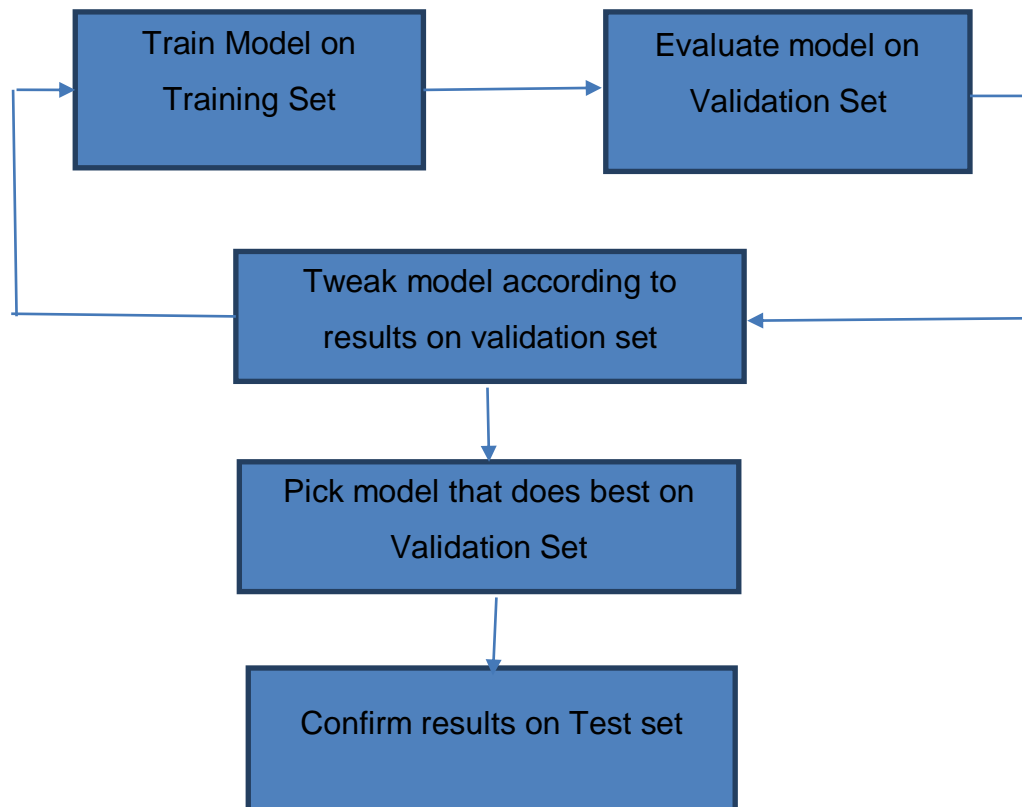


### 3.4 Testing of DL Model

Testing or Validation of DL model requires two sets of data.

1. Test data
2. Validation data

Following flowchart defines the flow of testing/validation of the proposed model.



### 3.4.1 Hyper Parameter Tuning

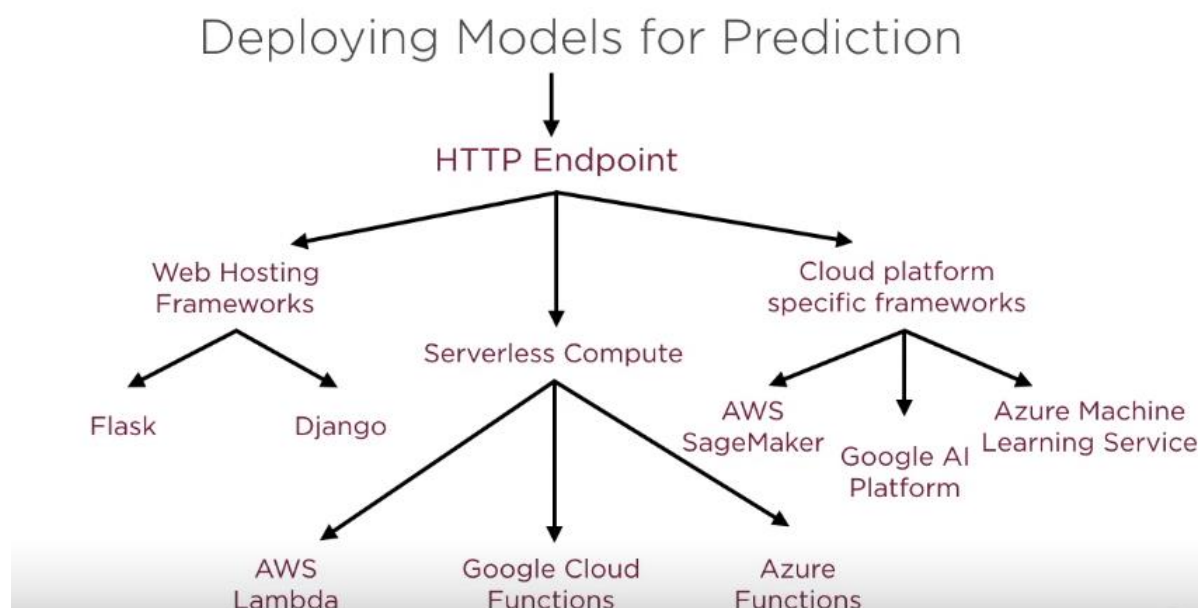
Several parameters that can help improve the performance and reduce losses of the model are as listed below.

Hyper Parameter	Definition	Possible values
Batch Size	The number of training samples to work through before the model's internal parameters are updated	
Regularization	Parameter that helps avoiding over-fitting issue in a model	
Learning Rate	Determines the step size at each iteration while moving toward a minimum of a loss function	
Number of epochs	Controls the number of complete passes through the training dataset.	

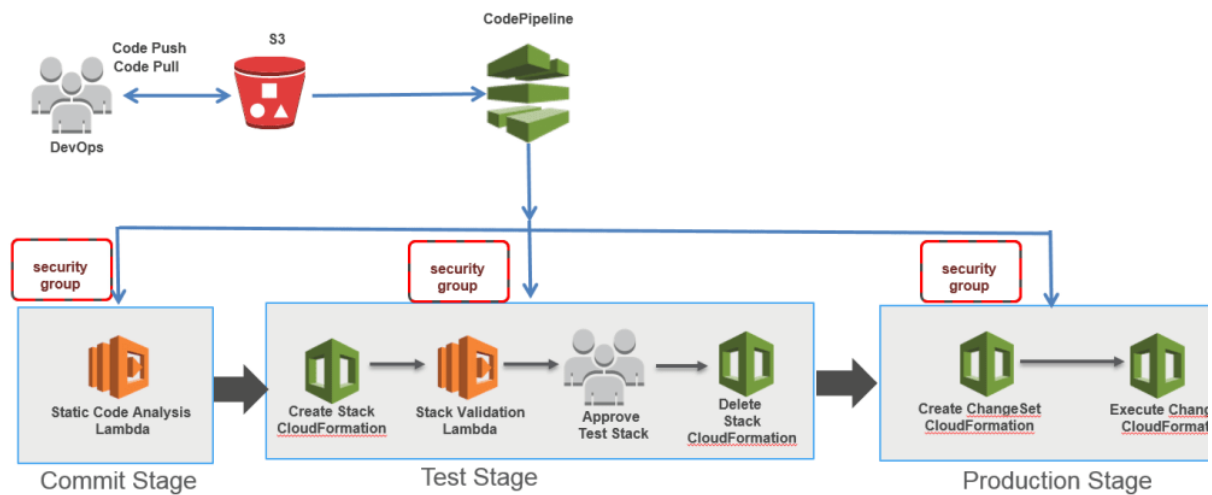
### 3.4.2 Performance Metrics

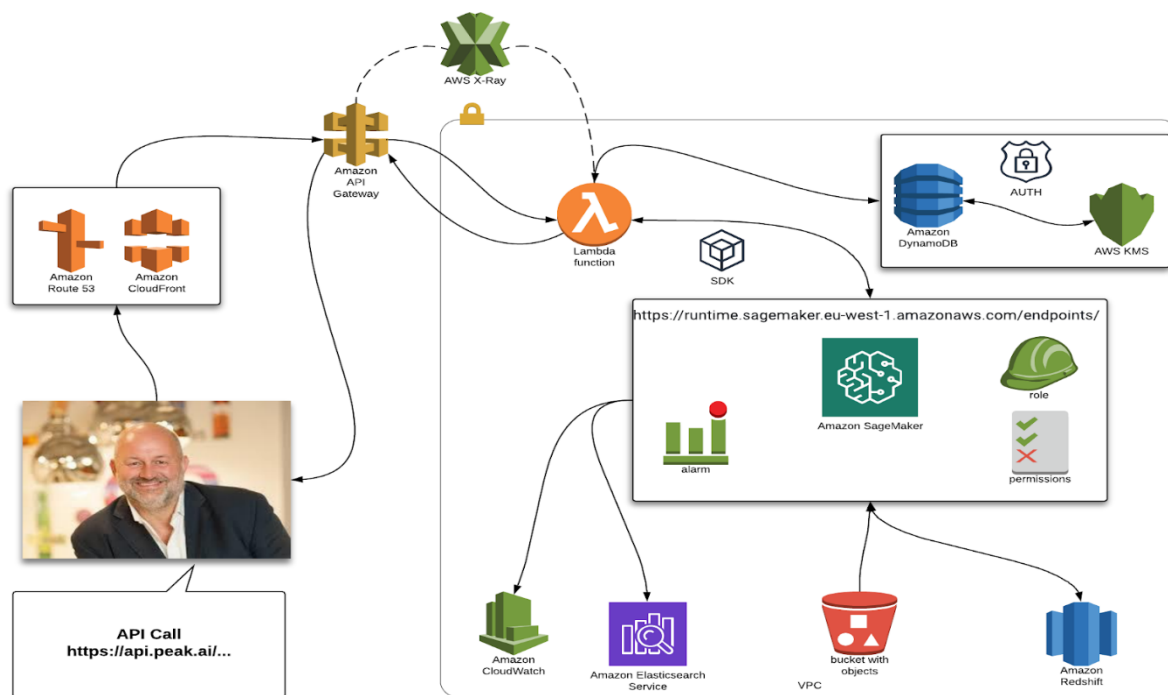
We can use the sensitivity, specificity, and ROC-AUC for determining the performance of the model.

## Deployment



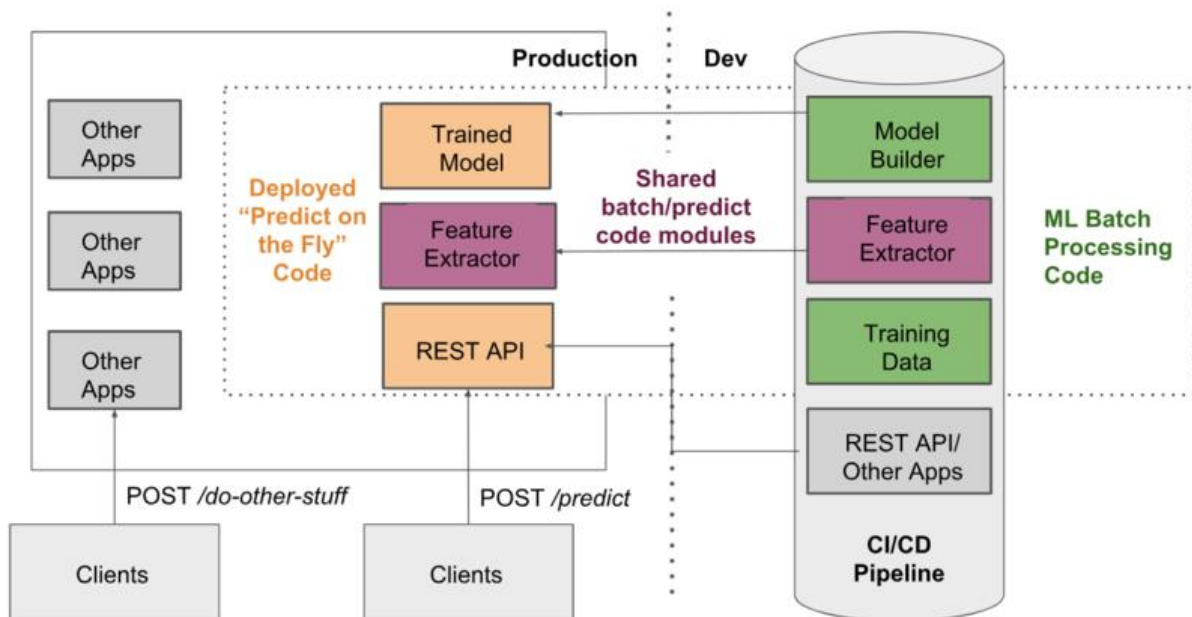






## Amazon SageMaker

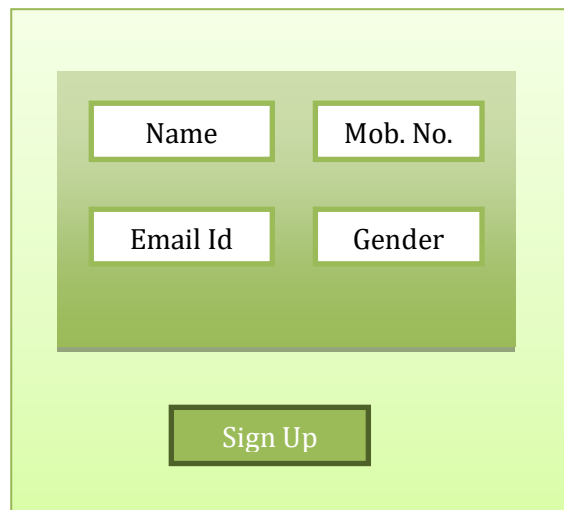
Amazon SageMaker is a cloud machine-learning platform that was launched in November 2017. SageMaker enables developers to create, train, and deploy machine-learning models in the cloud. SageMaker also enables developers to deploy ML models on embedded systems and edge-devices.



## GUI SCREENS

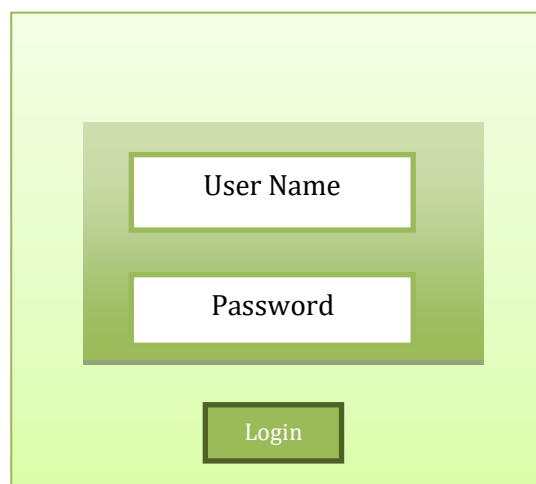
Stepwise format of Sample GUI Screens:

### Step 1: Sign up form



A diagram of a sign-up form. It consists of a light green outer rectangle containing a darker green inner rectangle. Inside the inner rectangle are four white input fields arranged in a 2x2 grid: 'Name' (top-left), 'Mob. No.' (top-right), 'Email Id' (bottom-left), and 'Gender' (bottom-right). Below the inner rectangle, centered within the outer rectangle, is a green button with the text 'Sign Up'.

### Step 2: To login using these inputs



A diagram of a login form. It consists of a light green outer rectangle containing a darker green inner rectangle. Inside the inner rectangle are two white input fields stacked vertically: 'User Name' (top) and 'Password' (bottom). Below the inner rectangle, centered within the outer rectangle, is a green button with the text 'Login'.

Step 3: Search the scan from data base

User queue

Patient Id:

Patient Name:

Save

Refresh

Accession #:

Referring Dr.

Status

Modality:

Search

Clear

Patient Name

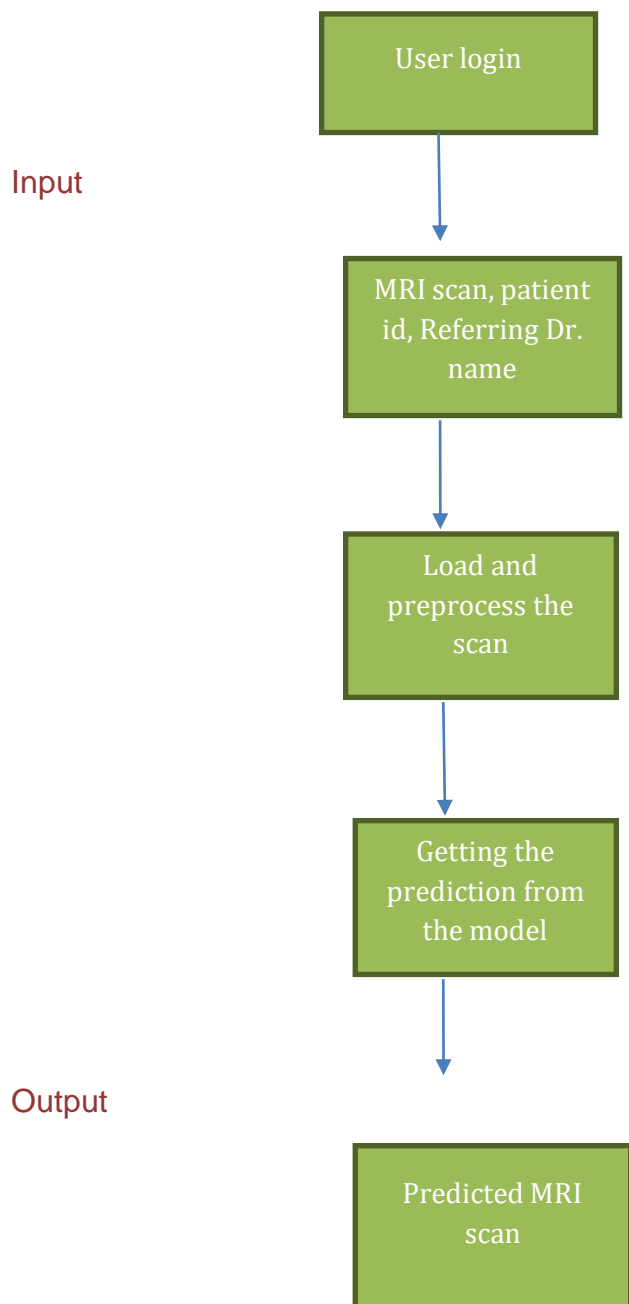
Step 4: Already studied scan Will be stored in Studies table

Pt. id	Pt. name	Access-ion	Ref. Dr name	Modality

Step 5: view the scan zoom, rotate, arrows etc.



## User I/O workflow:



### 2.6.3 Sample GUI



[\(Source\)](#)



[\(Source\)](#)

GUI Screens: Stepwise format of Sample GUI Screens:

### 3. References

1. "MRNet Dataset." Stanford ML Group, <https://stanfordmlgroup.github.io/competitions/mrnet/>
2. "Mrnet Dataset." Kaggle, <https://www.kaggle.com/cjinny/mrnet-v1>
3. ImageDataGenerator [https://www.tensorflow.org/api\\_docs/python/tf/keras/preprocessing/image/ImageDataGenerator](https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator)
4. ImgAug Library, <https://imgaug.readthedocs.io/en/latest/>
5. Azcona, D., McGuinness, K., & Smeaton, A. F. (2020, October). A Comparative Study of Existing and New Deep Learning Methods for Detecting Knee Injuries using the MRNet Dataset. In 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA) (pp. 149-155). IEEE.
6. Liu, F., Guan, B., Zhou, Z., Samsonov, A., Rosas, H., Lian, K., Sharma, R., Kanarek, A., Kim, J., Guermazi, A., & Kijowski, R. (2019). Fully Automated Diagnosis of Anterior Cruciate Ligament Tears on Knee MR Images by Using Deep Learning. Radiology. Artificial intelligence, 1(3), 180091. <https://doi.org/10.1148/ryai.2019180091>