National College of Ireland

# Configuration Manual

MSc Research Project
Data Analytics

Shivani Saxena
Student ID: x22168729

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

**Student Name:** ……. Shivani Saxena……

**Student ID:** ……x22168729……..……

**Programme:** ……Data analytics…… **Year:** ……..

**Module:** ……Msc Research Project……..……

**Lecturer:** ……Vikas Tomer……..……

**Submission Due Date:** ……14/12/2023……..……

**Project Title:** Disease detection for potato, tomato and pepper plants using ML algorithms

**Word Count:** ……1043…… **Page Count:** ……12……..……..……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** ……

**Date:** ……

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Shivani Saxena
Student ID: x22168729

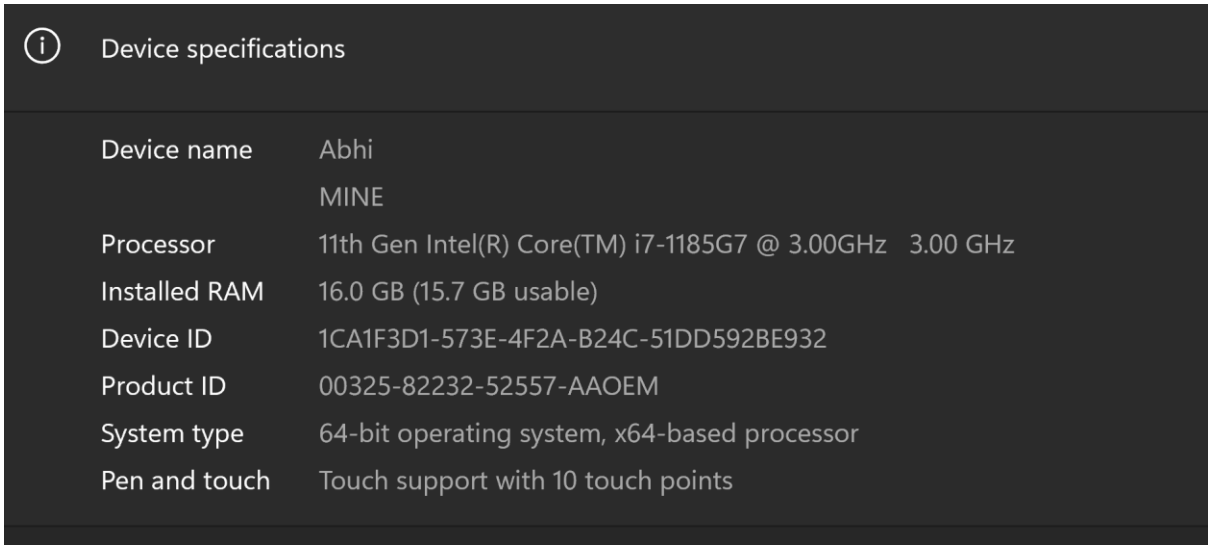
# 1 Introduction

This configuration manual consists of all the details of al the required parts for the project which includes hardware requirements, software requirements, design details, implementation of the project, environmental setup

# 2 System configuration

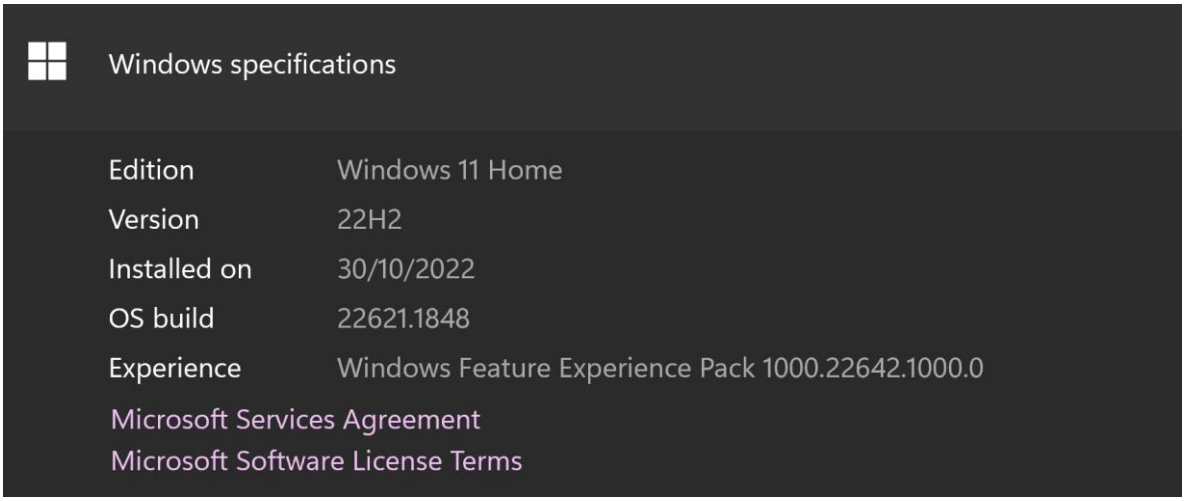
## 2.1 hardware configuration

The research was carried out the local machine below are the given hardware specification
Processor- 11th Gen Intel(R) Core(TM) i7-118G7 @3.00GHz
System type- 64-bit operating system, x64-based processor




## 2.2 Software Configuration

The research was carried out on windows 11 Home version 22H2

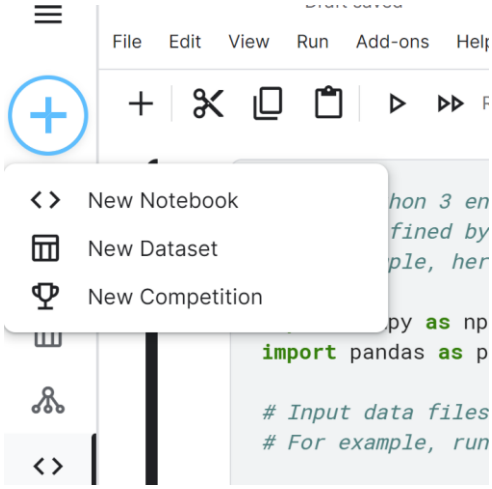



# 3 Environment Setup

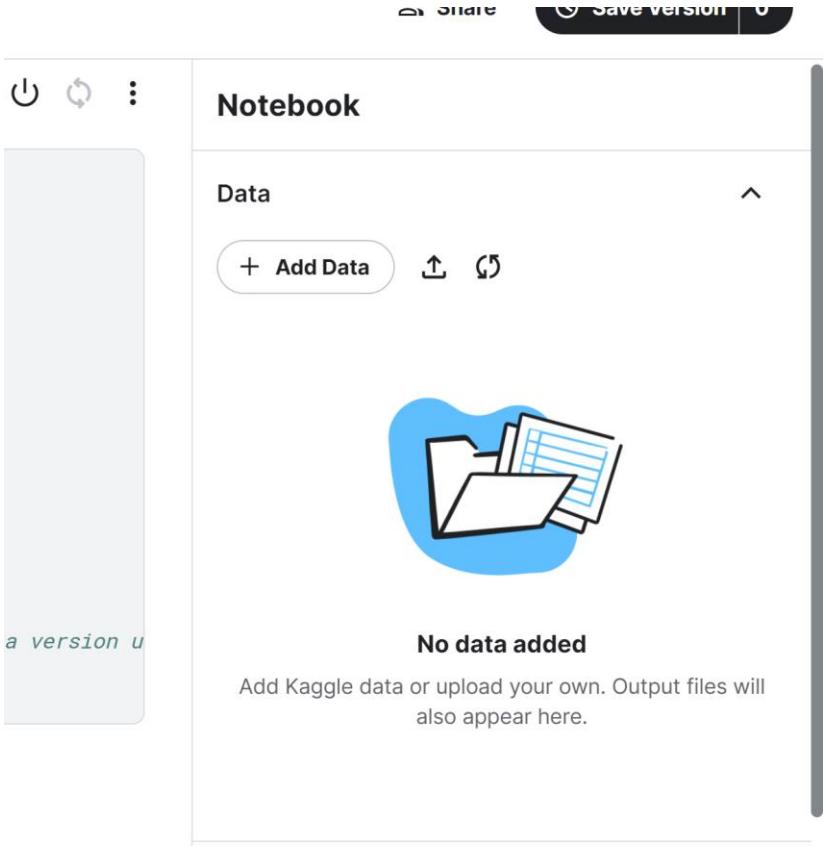
The environment used to carried this research was Kaggle IDE since I need to run my code on GPU and Kaggle provides free 30hrs of GPU unit on your account when phone number is verified (code was taking too long while running on collab while running on cpu and gpu of the collab is only available on paid versions )

| IDE | Kaggle |
|---|---|
| Programming Language | Python |
| Device | GPU |
| Other Tools | Microsoft Excel and word |

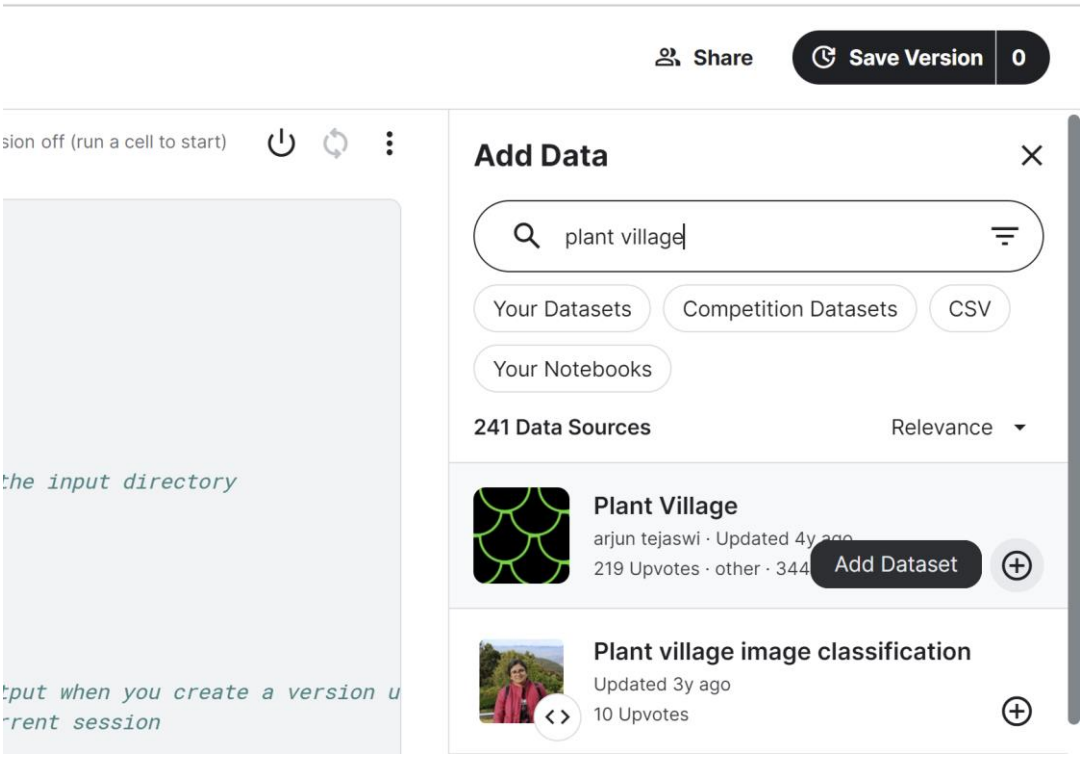
1. Go to Kaggle create notebook




2. Click on add data which is available on the right most side of the window and click on the + button there and search of the dataset




3. Once your dataset is visible click on the the button which is available on the right side + to add the dataset to work on



# 4 Project Implementation

## 4.1 Python libraries

While carrying out the research I have used multiple python libraries.
Numpy, pandas, matplotlib, pickle, Pytorch, sklearn

```
import numpy as np
import pandas as pd
import os
import torch
import random
import shutil

from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader, Dataset
import matplotlib.pyplot as plt
from torchvision.models import (
    resnet50, ResNet50_Weights,
    alexnet, AlexNet_Weights,
    efficientnet_v2_l, EfficientNet_V2_L_Weights
)
from torch import nn
from torch import optim
import torchvision
from torchvision import datasets, models, transforms
from torch.utils.data.sampler import SubsetRandomSampler
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for
this version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

## 4.2 data preprocessing and understanding

In this section we will perform the data preprocessing where will be transforming the data using various augmentation that are flips, rotate , they will be resized, and normalized.

```
train_folder = "/kaggle/working/train"
validation_folder = "/kaggle/working/validation"
test_folder = "/kaggle/working/test"

data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ]),
    'validation': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ]),
}

image_datasets = {
    'train': datasets.ImageFolder(train_folder, transform=data_transforms['train']),
    'validation': datasets.ImageFolder(validation_folder, transform=data_transforms['validation']),
    'test': datasets.ImageFolder(test_folder, transform=data_transforms['validation'])
}

dataset_size = {
    "train": len(image_datasets["train"]),
    "val": len(image_datasets["validation"]),
    "test": len(image_datasets["test"])
}
```

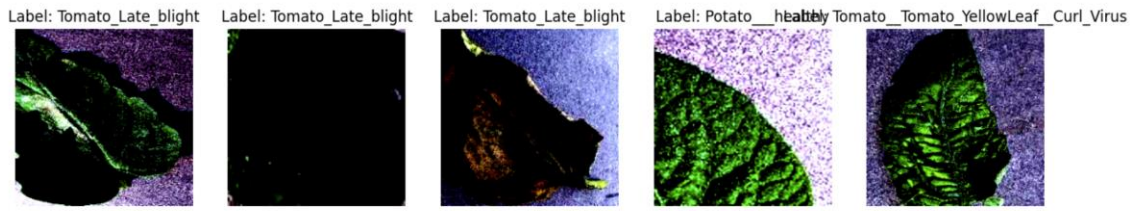
After this we willl retrive all the classes of the images which are there in the dataset and size of the dataset too.

```
['Pepper__bell___Bacterial_spot',
 'Pepper__bell___healthy',
 'Potato___Early_blight',
 'Potato___Late_blight',
 'Potato___healthy',
 'Tomato_Bacterial_spot',
 'Tomato_Early_blight',
 'Tomato_Late_blight',
 'Tomato_Leaf_Mold',
 'Tomato_Septoria_leaf_spot',
 'Tomato_Spider_mites_Two_spotted_spider_mite',
 'Tomato__Target_Spot',
 'Tomato__Tomato_YellowLeaf__Curl_Virus',
 'Tomato__Tomato_mosaic_virus',
 'Tomato_healthy']
```

```
dataset_size
```

```
{'train': 12377, 'val': 4122, 'test': 4139}
```

After that we will be generating the images present on which we need to apply the models




## 4.3 model implementation and Evaluation

we will be implementing 3 models restNet, efficientNet, and alexnet to achive the best accuracy out of the three models
we will save the models in the best_model for restNet model
best_model_eff for the effieinet model
best_model_alex for the alexnet model
we ran the model on the 3 learning rates 0.1,0.01 and 0.001 with batch size 32 and 3 different step sizes 5,7,10

RestNet model

a. chekcing for the best model

```
LR: 0.1 Step Size: 5 Best Accuracy: 86.02620087336244
LR: 0.1 Step Size: 7 Best Accuracy: 88.5977680737506
LR: 0.1 Step Size: 10 Best Accuracy: 89.39835031538088
LR: 0.01 Step Size: 5 Best Accuracy: 89.68947113051917
LR: 0.01 Step Size: 7 Best Accuracy: 89.8592916060165
LR: 0.01 Step Size: 10 Best Accuracy: 90.22319262493934
LR: 0.001 Step Size: 5 Best Accuracy: 90.22319262493934
LR: 0.001 Step Size: 7 Best Accuracy: 90.22319262493934
LR: 0.001 Step Size: 10 Best Accuracy: 90.22319262493934
Best LR: 0.01 Best Step Size: 10
```

b. chekcing the accuracy of the model

```
        predicted = torch.argmax(outputs, dim=1)
        test_total += labels.size(0)
        test_correct += (predicted == labels).sum().item

test_accuracy = 100. * test_correct / test_total
print(f'Test Accuracy: {test_accuracy:.4f}%')
```

```
Test Accuracy: 89.3694%
```

c. checking the precison, recall and accuracy of the model

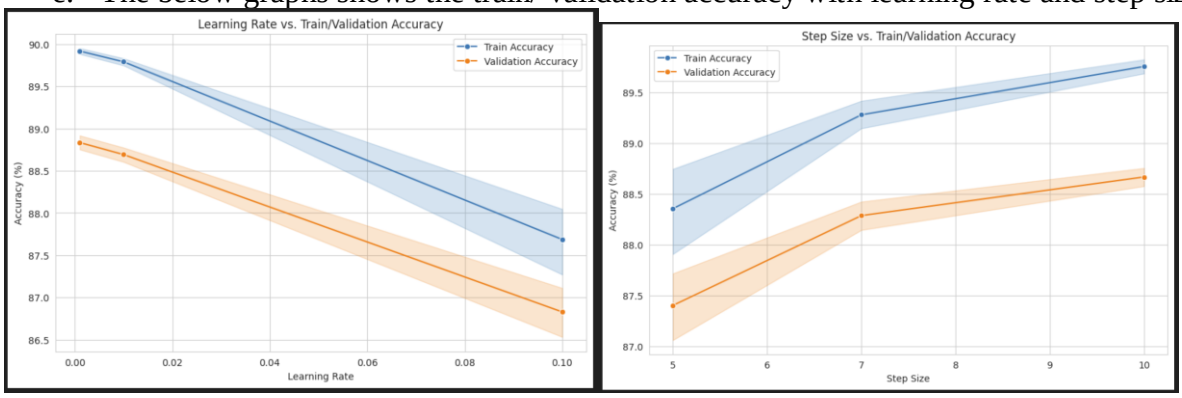
```
Precision: 0.8910
Recall: 0.8860
F1 Score: 0.8847
```

d. Saving the new dataframe in the result_df csv file

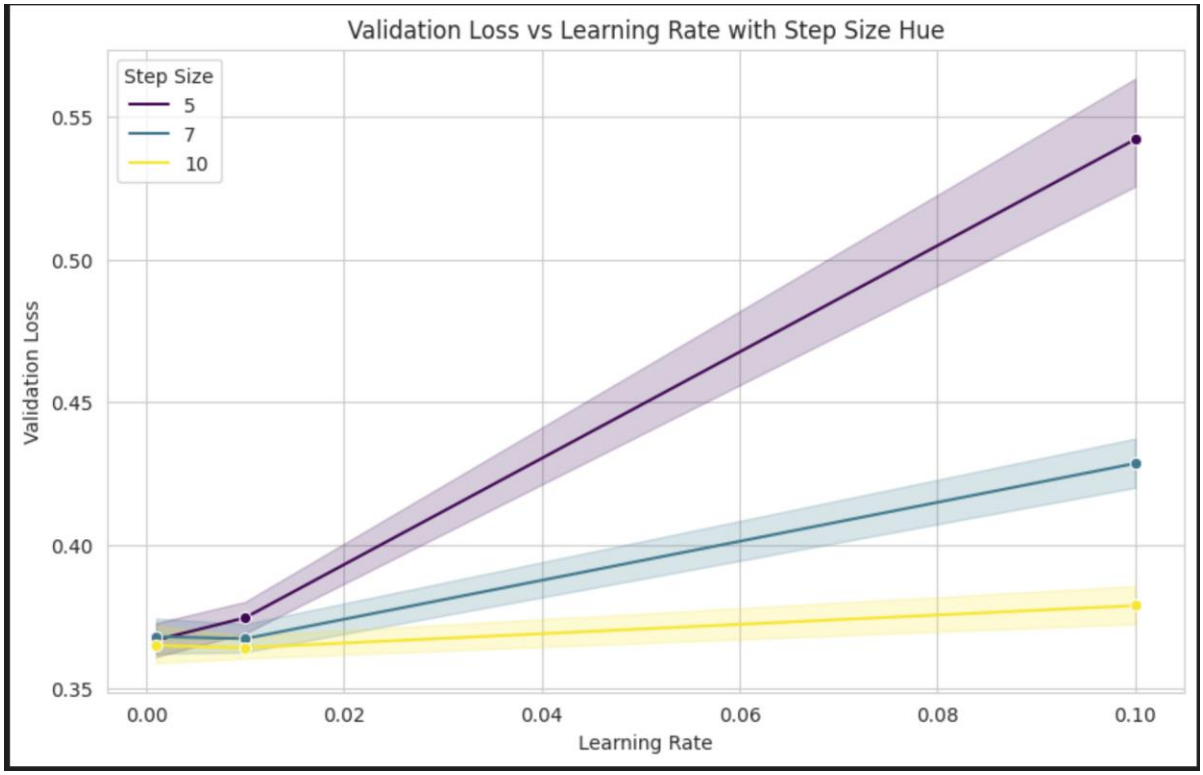
| | Learning Rate | Step Size | Train Loss | Train Accuracy | Validation Loss | Validation Accuracy |
|---|---|---|---|---|---|---|
| 0 | 0.100 | 5 | 1.209047 | 66.058011 | 0.911607 | 76.637555 |
| 1 | 0.100 | 5 | 0.754563 | 78.936737 | 0.773006 | 79.015041 |
| 2 | 0.100 | 5 | 0.630841 | 81.942312 | 0.660324 | 82.241630 |
| 3 | 0.100 | 5 | 0.566265 | 83.614769 | 0.607370 | 82.702572 |
| 4 | 0.100 | 5 | 0.533164 | 84.430799 | 0.555256 | 84.036875 |
| .. | ... | ... | ... | ... | ... | ... |
| 445 | 0.001 | 10 | 0.326353 | 89.989497 | 0.372721 | 88.403688 |
| 446 | 0.001 | 10 | 0.323305 | 90.151087 | 0.343228 | 89.058709 |
| 447 | 0.001 | 10 | 0.324072 | 90.094530 | 0.354622 | 89.058709 |
| 448 | 0.001 | 10 | 0.327621 | 90.005656 | 0.389722 | 88.015526 |
| 449 | 0.001 | 10 | 0.323081 | 89.835986 | 0.357290 | 89.034449 |

[450 rows x 6 columns]

e. The below graphs shows the train/ validation accuracy with learning rate and step size




f. The below graph shows the validation loss vs learning rate on different step sizes




AlexNet model

a. chekcing for the best model

```
LR: 0.1 Step Size: 5 Best Accuracy: 81.97320341047504
LR: 0.1 Step Size: 7 Best Accuracy: 85.01827040194884
LR: 0.1 Step Size: 10 Best Accuracy: 85.01827040194884
LR: 0.01 Step Size: 5 Best Accuracy: 85.01827040194884
LR: 0.01 Step Size: 7 Best Accuracy: 85.1400730816078
LR: 0.01 Step Size: 10 Best Accuracy: 85.87088915956151
LR: 0.001 Step Size: 5 Best Accuracy: 85.87088915956151
LR: 0.001 Step Size: 7 Best Accuracy: 85.87088915956151
LR: 0.001 Step Size: 10 Best Accuracy: 85.87088915956151
Best LR: 0.01 Best Step Size: 10
```

b. chekcing the accuracy of the model

```
Test Accuracy: 85.5596%
```

c. checking the precison, recall and accuracy of the model

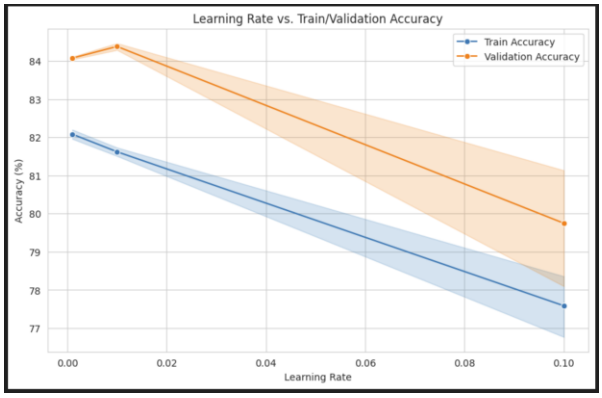
```
Precision: 0.8651
Recall: 0.8556
F1 Score: 0.8524
```

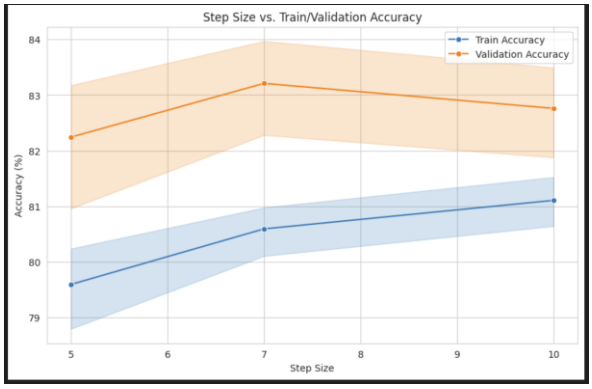
d. Saving the new dataframe in the result_df csv file

| | Learning Rate | Step Size | Train Loss | Train Accuracy | Validation Loss | Validation Accuracy |
|---|---|---|---|---|---|---|
| 0 | 0.100 | 5 | 11.048008 | 45.870445 | 31.624462 | 20.950061 |
| 1 | 0.100 | 5 | 6.444064 | 57.570850 | 14.997222 | 53.958587 |
| 2 | 0.100 | 5 | 5.567074 | 60.485830 | 7.554774 | 63.580999 |
| 3 | 0.100 | 5 | 4.872732 | 63.967611 | 11.251095 | 47.746650 |
| 4 | 0.100 | 5 | 4.971214 | 64.089069 | 14.695096 | 59.196102 |
| .. | ... | ... | ... | ... | ... | ... |
| 445 | 0.001 | 10 | 1.081323 | 83.238866 | 1.238125 | 84.287454 |
| 446 | 0.001 | 10 | 1.267841 | 81.417004 | 1.238128 | 84.287454 |
| 447 | 0.001 | 10 | 1.093792 | 82.388664 | 1.238126 | 84.287454 |
| 448 | 0.001 | 10 | 1.217089 | 81.133603 | 1.238127 | 84.287454 |
| 449 | 0.001 | 10 | 1.149771 | 82.307692 | 1.238129 | 84.287454 |

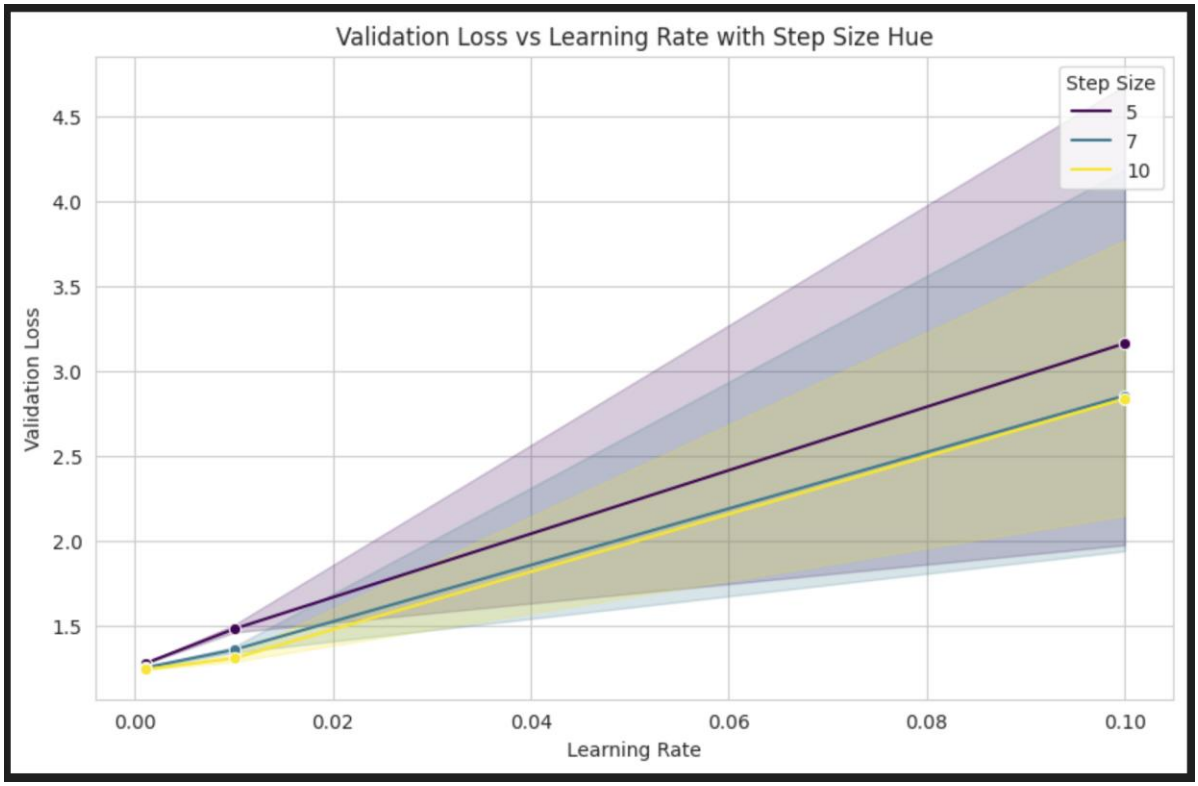
[450 rows x 6 columns]

e. The below graphs shows the train/ validation accuracy with learning rate and step size




f. The below graph shows the validation loss vs learning rate on different step sizes




EfficientNet model

a. chekcing for the best model

```
100%|██████████| 20.5M/20.5M [00:00<00:00, 89.7MB/s]
Epoch: 49, LR: 0.1, Step Size: 5, Best Accuracy: 97.32034104750305
Epoch: 49, LR: 0.1, Step Size: 7, Best Accuracy: 98.29476248477467
Epoch: 49, LR: 0.1, Step Size: 10, Best Accuracy: 98.53836784409258
Epoch: 49, LR: 0.01, Step Size: 5, Best Accuracy: 98.53836784409258
Epoch: 49, LR: 0.01, Step Size: 7, Best Accuracy: 98.66017052375152
Epoch: 49, LR: 0.01, Step Size: 10, Best Accuracy: 98.66017052375152
Epoch: 49, LR: 0.001, Step Size: 5, Best Accuracy: 98.78197320341047
Epoch: 49, LR: 0.001, Step Size: 7, Best Accuracy: 98.78197320341047
Epoch: 49, LR: 0.001, Step Size: 10, Best Accuracy: 98.78197320341047
Best LR: 0.001 Best Step Size: 5
```

b. chekcing the accuracy of the model

```
Test Accuracy: 97.3526%
```

c. checking the precison, recall and accuracy of the model

```
Precision: 0.9740
Recall: 0.9735
F1 Score: 0.9735
```

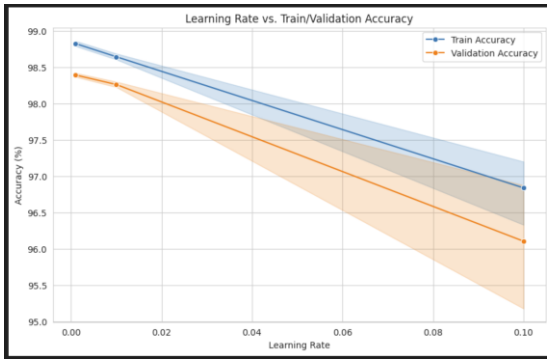
d. Saving the new dataframe in the result_df csv file

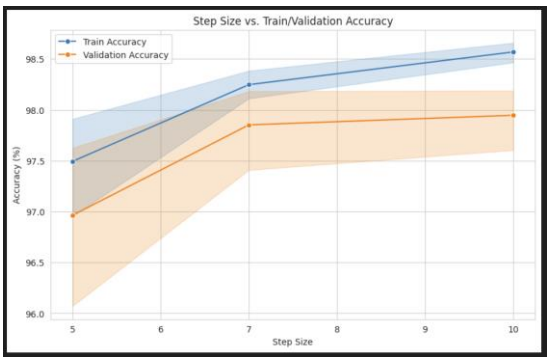
| | Learning Rate | Step Size | Epoch | Train Loss | Train Accuracy | \ |
|---|---|---|---|---|---|---|
| 0 | 0.100 | 5 | 0 | 1.560533 | 67.287449 | |
| 1 | 0.100 | 5 | 1 | 0.494857 | 86.032389 | |
| 2 | 0.100 | 5 | 2 | 0.369051 | 89.190283 | |
| 3 | 0.100 | 5 | 3 | 0.316108 | 90.485830 | |
| 4 | 0.100 | 5 | 4 | 0.230657 | 93.036437 | |
| .. | ... | ... | ... | ... | ... | |
| 445 | 0.001 | 10 | 45 | 0.027517 | 99.190283 | |
| 446 | 0.001 | 10 | 46 | 0.031766 | 99.109312 | |
| 447 | 0.001 | 10 | 47 | 0.045659 | 98.623482 | |
| 448 | 0.001 | 10 | 48 | 0.036997 | 98.987854 | |
| 449 | 0.001 | 10 | 49 | 0.028252 | 98.987854 | |

| | Validation Loss | Validation Accuracy |
|---|---|---|
| 0 | 1.132215 | 72.959805 |
| 1 | 2.000496 | 49.086480 |
| 2 | 0.950446 | 73.934227 |
| 3 | 0.367253 | 89.159562 |
| 4 | 0.800227 | 85.261876 |
| .. | ... | ... |
| 445 | 0.044411 | 98.416565 |
| 446 | 0.047185 | 98.172960 |
| 447 | 0.047537 | 98.416565 |
| 448 | 0.048228 | 98.294762 |
| 449 | 0.048468 | 98.294762 |

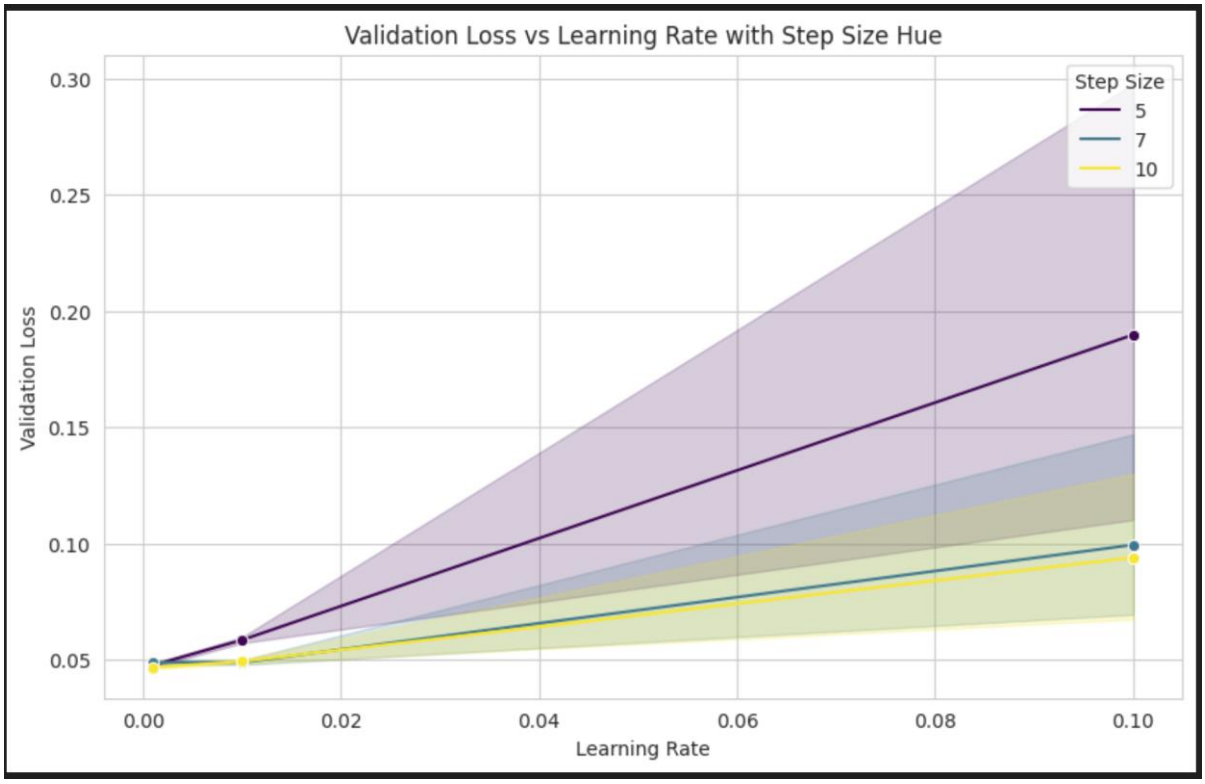
[450 rows x 7 columns]

e. The below graphs shows the train/ validation accuracy with learning rate and step size




f. The below graph shows the validation loss vs learning rate on different step sizes




## 4.4 Result

The final result of all the models are shown below we can see that the efficidnet models performs the best out of all the three model with highest accuracy.

Results of all the models

| model | accuracy | precision | f1score | recall |
|---|---|---|---|---|
| ResNet | 88.59% | 89.10% | 88.47% | 88.60% |
| AlexNet | 85.55% | 86.50% | 85.24% | 85.56% |
| EfficientNet | 97.35% | 97.40% | 97.35% | 97.35% |

# References

**References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at https://libguides.ncirl.ie/referencing**
**You can use a reference management system such as Zotero or Mendeley to cite in MS Word.**

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* 27(5): 1310–1333.

Feng, G. and Buyya, R. (2016). Maximum revenue-oriented resource allocation in cloud, *IJGUC* 7(1): 12–21.

Gomes, D. G., Calheiros, R. N. and Tolosana-Calasanz, R. (2015). Introduction to the special issue on cloud computing: Recent developments and challenging issues, *Computers & Electrical Engineering* 42: 31–32.

Kune, R., Konugurthi, P., Agarwal, A., Rao, C. R. and Buyya, R. (2016). The anatomy of big data computing, *Softw., Pract. Exper.* 46(1): 79–105.