National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Sarthak Gupta
Student ID: 20247575

School of Computing
National College of Ireland

Supervisor: Mr. Aaloka Anant

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Sarthak Gupta

Student ID: x20247575

Programme: MSc Data Analytics **Year:** 2022-2023

Module: Research Project

Lecturer: Mr Aaloka Anant

Submission Due Date: 01.02.2023

Project Title: Skin Lesion Classification Based on Various Machine Learning Models Explained by Explainable Artificial Intelligence

Word Count: 736 **Page Count:** 8

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: Sarthak Gupta

Date: 01.02.2023

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

Skin Lesion Classification Based on Various Machine Learning Models Explained by Explainable Artificial Intelligence

Sarthak Gupta
Student ID:20247575

1 Introduction

The aim of this project is to build a classification model based on a machine learning model, the XGB Classifier, and two convolutional neural network models. and with the help of SHAP and LIME, explain the decision-making process. In this document important code snippets are present that can be used to recreate the project code.

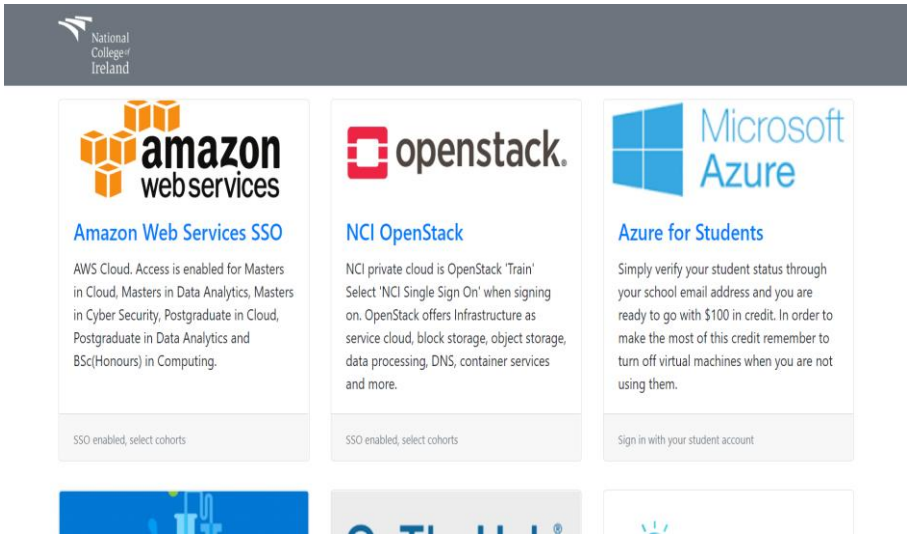
2 System Requirements

To implement this project, the Amazon Web Services platform was used to run the proposed models, which required high computational power.

2.1 Software Configuration

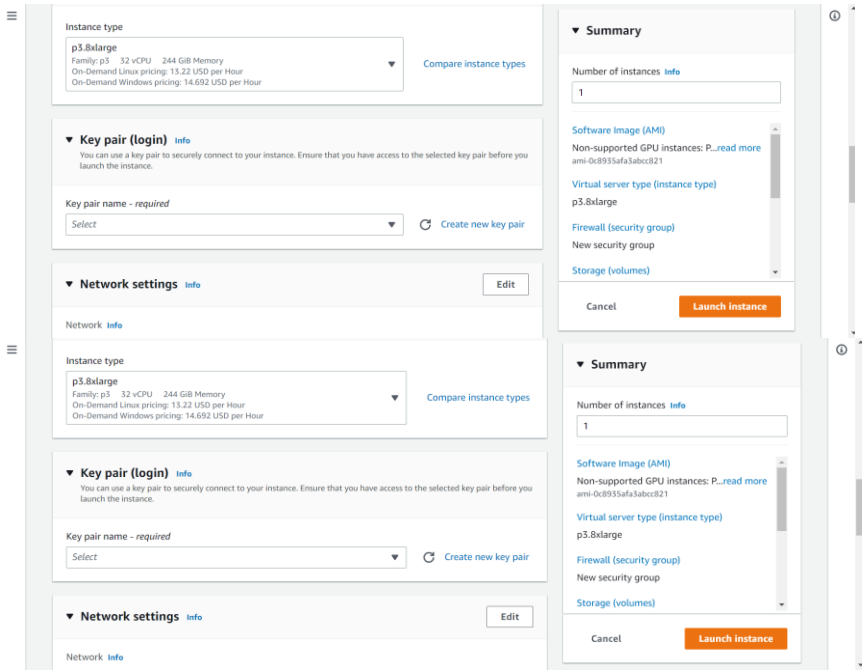
Amazon Web Services Setup

- Go to https://cloud.ncirl.ie/ and click on aazon web services



Create an EC2 Instance

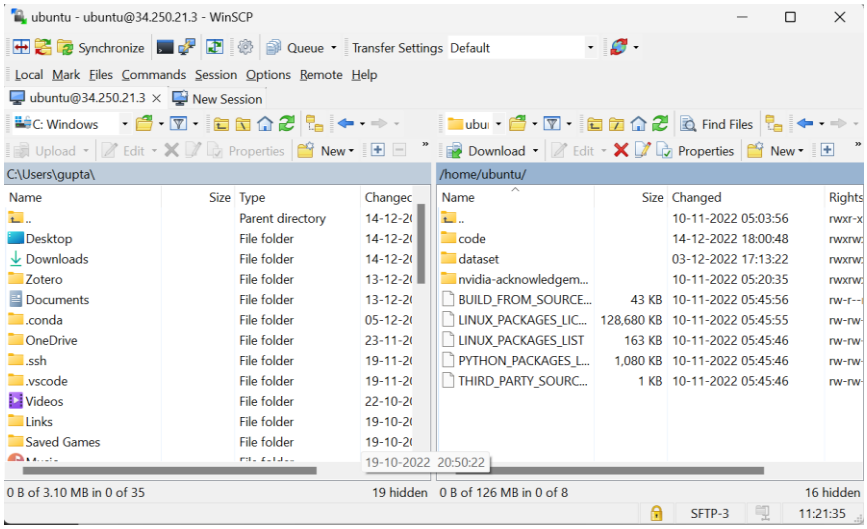
- Using the settin shown in the screenshot, create an instance



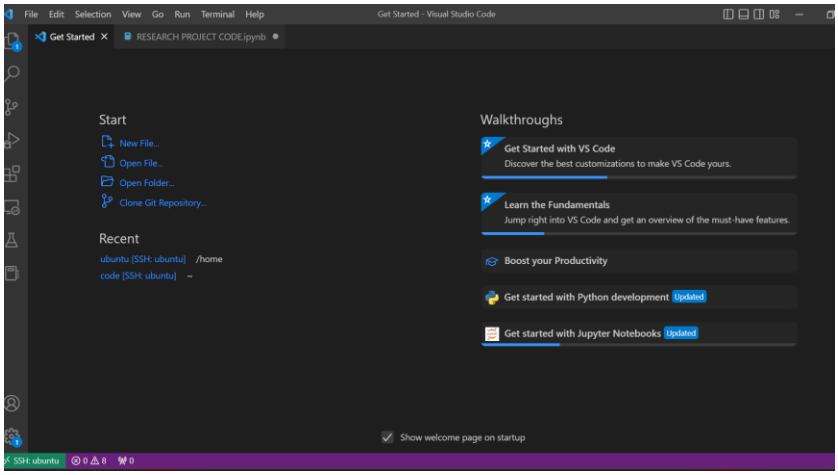
- Select 75 GB as storage of the setup

The following should be the setup of the instance.

- To transfer files WinSCP is used with the credentials



- To run the python script, we used visual studio code connected to the server via credentials



3 Environment Setup

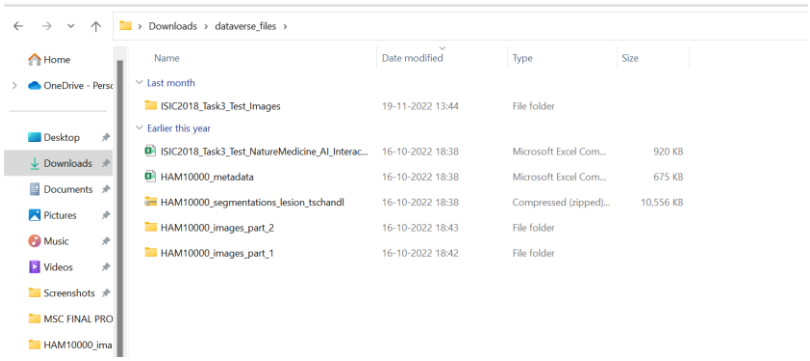
The following libraries are installed and imported.

```
Pandas
Numpy
CV2
Matplotlib
OS
TenserFlow
Keras
LIME
SHAP
Seaborn
Sklearn
xgboost
```

4 Implementation

4.1 Data Collection

- The dataset was downloaded for the Havard Dataverse website and should be unzipped before use. The link of the dataset is provided below https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T



- The CSV file path is given to the and imported in the environment.

```
metadata=pd.read_csv('/home/ubuntu/dataset/HAM10000_metadata.csv', on_bad_lines='skip')
```

```
metadata.sample(n=5)
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	dataset
3845	HAM_0006947	ISIC_0031662	nv	histo	45.0	female	foot	rosendahl
7902	HAM_0002015	ISIC_0033771	nv	histo	30.0	female	lower extremity	vidir_modern
1472	HAM_0002032	ISIC_0030575	mel	histo	25.0	male	back	rosendahl
2723	HAM_0004662	ISIC_0024331	bcc	histo	65.0	male	lower extremity	rosendahl
2810	HAM_0006386	ISIC_0029035	bcc	histo	70.0	male	face	rosendahl

```
metadata.rename(columns = {'localization':'lesion_location'}, inplace = True)
metadata.rename(columns = {'dx':'lesion_type'}, inplace = True)
metadata.rename(columns = {'dx_type':'lesion_confirmation_type '}, inplace = True)
```

4.2 Data Preprocessing

- Null values are removed using the following code

```
metadata['age'].fillna((metadata['age'].mean()), inplace=True)
metadata.isnull().sum()
```

- Data visulasization is done using matplotlib, seaborn

4.3 Data Transformation

- The images from the dataset were resized fro 450*600 to 120*160

```
metadata['image'] = metadata['path'].map(lambda x: np.asarray(Image.open(x).resize((160,120))))
```

- The dataset is splitted into test and train

```
features=metadata.drop(columns=['lesion_type_categorical','dataset'],axis=1)
target=metadata['lesion_type_categorical']


trainX, testX, trainY, testY = train_test_split(features, target, test_size=0.20,random_state=1234)
```

- The dataset is Standardized
- The attributes of the dataset are converted to categorical values.

4.4 Model Building

- Three models are proposed in this dataset two built on the same CNN architecture
- CNN Architecture

```
# CNN Model Architecture
input_shape = (120, 160, 3)
num_classes = 7

model = Sequential()
model.add(Conv2D(16, (3, 3), padding='same',input_shape=input_shape, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer = 'adam', loss = "categorical_crossentropy", metrics=["accuracy"])
```

Model 1 CNN on orignal dataset

- Model fitting CNN on original dataset.

```
Epochs=50
batchSize=5
t0=time.time()
# model fit

print('-----------------------------------------------')
print('Running Model 1 without data augmentation.')
print('-----------------------------------------------')
ModelT = model.fit(x_train, y_train,
                   validation_data=(x_test, y_test),
                   epochs=Epochs,
                   batch_size=batchSize,
                   verbose=1)

print('-----------------------------------------------')
t1=time.time()
print(t1-t0," seconds")
```

Mdel 2 CNN after Image Augmentation

- Image augmentation

```
trainDatagen = ImageDataGenerator(
    rotation_range = 20, # Tried a variety of rotations but made little difference
    width_shift_range = 0.1,  # 0.2
    height_shift_range = 0.1, # 0.2
    #shear_range = 0.1,  # 0.2
    zoom_range = 0.1, # 0.2, 0.3
    horizontal_flip = True,
    #vertical_flip = True # tended to add a bit more overfitting
)

trainDatagen.fit(x_train) # fit the training data in order to augment.
```

- Model fit after Image augmentation

```
epochs = 50
batchSize = 10
t0=time.time()

print('-----------------------------------------------')
print('Running Model 2 - Data Augumentation Included.')
print('-----------------------------------------------')

Model = model.fit_generator(trainDatagen.flow(x_train,y_train, batch_size=batchSize),
                            epochs = epochs, validation_data = (x_validate,y_validate),
                            verbose = 1, steps_per_epoch=x_train.shape[0] // batchSize,
                            callbacks=[lrReduction])

t1=time.time()
print(t1-t0," seconds")
```

Model 3 XGB Classifier

- Categorization of columns
- Split into test and train

```
X = tile_df[features]
y = tile_df['lesion_type_categorical'].values
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
```

- Model fit

```
model = XGBClassifier(random_state=1)
model = model.fit(X_train, y_train)
```

5 Evaluation

Results of Model are calculated using the same code as the model are same but different parameters there the following snippets can be used to evaluate all the models.

- For model accuracy and loss

```
print('-----------------------------------')
print('Model 2 Accuracy and Loss Scores')
print('-----------------------------------')
scores = model.evaluate(x_test, y_test, verbose=2)

print("CNN Error: %.2f%%" % (100-scores[1]*100))
print("CNN Acc: %.2f%%" % (scores[1]*100))

final_loss, final_acc = model.evaluate(x_test, y_test, verbose=1)
print("Final loss: {0:.4f}".format(final_loss, final_acc))
print('-----------------------------------------')
```

- To generate classification report

```
from sklearn.metrics import classification_report
print('Model 2 Classification Report')
print('---------------------------------------')

model_report= classification_report(y_true, y_pred, target_names=targetNames)
print(model_report)
```

- To calculate area under the curve

```
def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = preprocessing.LabelBinarizer()
    lb.fit(y_test)
    y_test1 = lb.transform(y_test)
    y_pred1 = lb.transform(y_pred)
    return roc_auc_score(y_test1, y_pred1, average=average)

aucScore =multiclass_roc_auc_score(y_test, y_pred)
print('-----------------------------------')
print('Model 2 Area Under Curve')
print('-----------------------------------')
print("AUC: %.2f%%" % (aucScore*100))
```

- To calculate confusion matrix

```
confusion_mtx = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(9,6))

sns.heatmap(confusion_mtx, annot=True, fmt='d',cmap=plt.cm.BuPu )
plt.show()
```

6 Explainable AI

- To implement install LIME AND SHAP Using !pip install command.

SHAP explainer

- Define model and SHAP Value

```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
```

- Create plots to explain the models using the following codes

```
shap.summary_plot(shap_values, X_test, plot_type="bar")
```

```
shap.summary_plot(shap_values[0], X_test)
```

```
shap.initjs()
shap.force_plot(explainer.expected_value[0], shap_values[0][:100,:], X_test.iloc[:100,:])
```

```
shap.initjs()
shap.force_plot(explainer.expected_value[0], shap_values[0][15,:], X_test.iloc[15,:])
```

LIME Explainer

Install and import LIME

- LIME is inputted CNN model to explain the images

```
np.random.seed(222)

Xi = x_test[0]
preds = model.predict(Xi[np.newaxis,:,:,:])
top_pred_classes = preds[0].argsort()[-7:][::-1] # Save ids of top 5 classes
top_pred_classes
```

- The following code can be used to proce explaination for an image present in the datset
- Select an image at random from the datset to provide explainations

```
#Generate segmentation for image
superpixels = skimage.segmentation.quickshift(Xi, kernel_size=4,max_dist=200, ratio=0.2)
num_superpixels = np.unique(superpixels).shape[0]
skimage.io.imshow(skimage.segmentation.mark_boundaries(Xi, superpixels))
print("The number of super pixels generated")
num_superpixels
```

```
#Generate perturbations
num_perturb = 150
perturbations = np.random.binomial(1, 0.5, size=(num_perturb, num_superpixels))

#Create function to apply perturbations to images
import copy
def perturb_image(img,perturbation,segments):
    active_pixels = np.where(perturbation == 1)[0]
    mask = np.zeros(segments.shape)
    for active in active_pixels:
        mask[segments == active] = 1
    perturbed_image = copy.deepcopy(img)
    perturbed_image = perturbed_image*mask[:,:,np.newaxis]
    return perturbed_image

#Show example of perturbations
print(perturbations[0])
```

```
skimage.io.imshow(perturb_image(Xi,perturbations[5],superpixels))
```

```
#Estimate linear model
from sklearn.linear_model import LinearRegression
class_to_explain = 4
simpler_model = LinearRegression()
simpler_model.fit(X=perturbations, y=predictions[:,:,class_to_explain], sample_weight=weights)
coeff = simpler_model.coef_[0]

#Use coefficients from linear model to extract top features
num_top_features = 4
top_features = np.argsort(coeff)[-num_top_features:]

#Show only the superpixels corresponding to the top features
mask = np.zeros(num_superpixels)
mask[top_features]= True #Activate top superpixels
skimage.io.imshow(perturb_image(Xi,mask,superpixels))
```