National College of Ireland

Configuration Manual

MSc Research Project
MSc. Data Analytics

Elizabath Kaimoolayil Thomas
Student ID: x20170131

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Elizabath Kaimoolayil Thomas
Student ID:	X20170131
Programme:	MSc. Data Analytics **Year:** 2021
Module:	Research Project
Lecturer:	Jorge Basilio
Submission Due Date:	16/12/2021
Project Title:	Prediction of Malignant melanoma Using machine learning……
Word Count:	525 **Page Count:** 11

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: Elizabath Kaimoolayil Thomas

Date: 16/12/2021

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

Elizabath Kaimoolayil Thomas
Student ID: x20170131

1 Hardware/Software Requirements

The setup handbook covers the procedures to be followed while executing the study scripts. This handbook will guide you through the process of properly executing the algorithm. The hardware setup of the machine on which the code was run is also detailed in this handbook. The bare minimum configuration required for the system is also mentioned.

2 System Specification

The hardware specs for the machine upon which research study is run are listed below.

- Processor: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
- RAM: 8 GB
- Storage: 963 GB SSD
- Operating System: windows

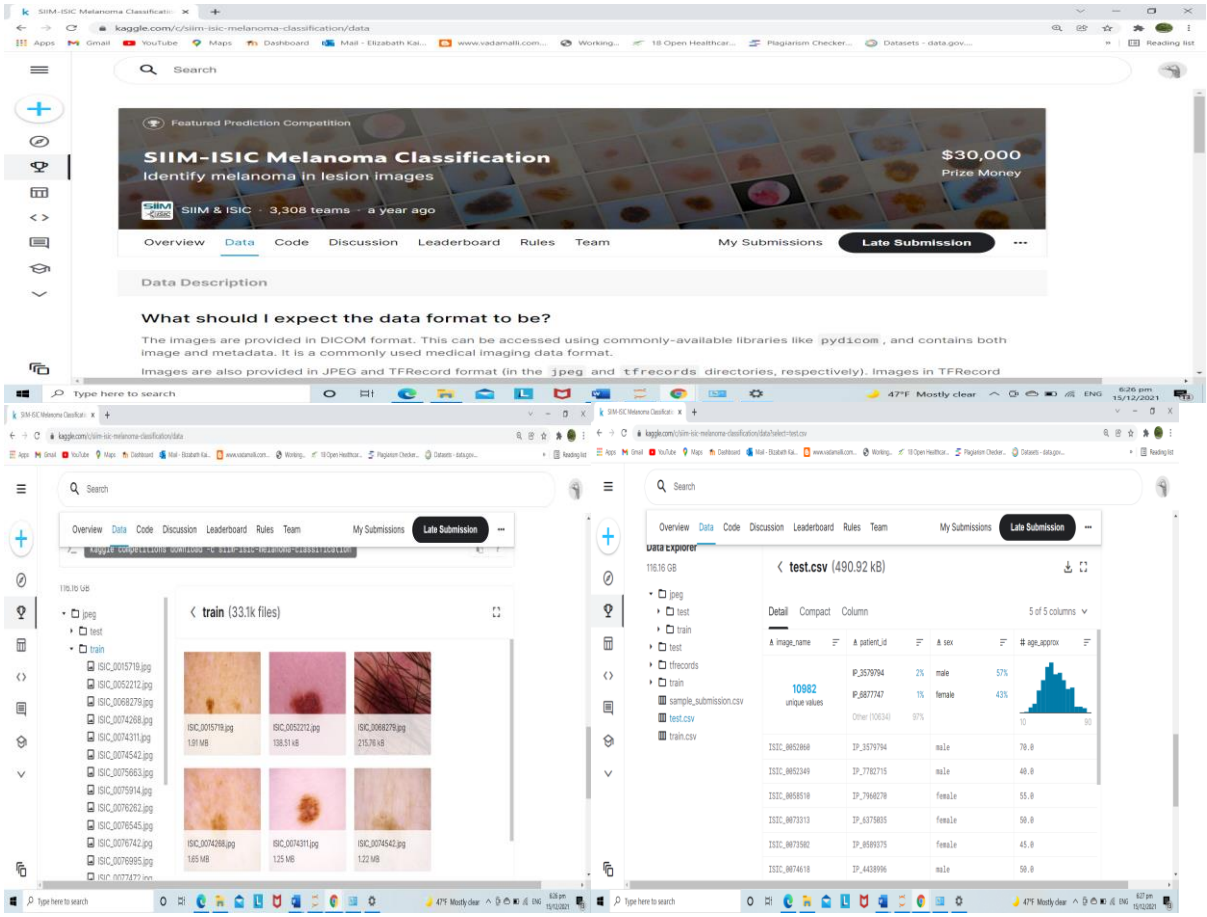
3 Software Requirements

Appropriate the programming tools required for the study are given below.

- Python version 3.8
- Jupiter notebook
- Microsoft Excel
- Word.

4 Data Collection

The datasets for the study collected from Kaggle. Named as SIIM-ISIC Melanoma Classification. Which contain more than 30000 images and csv files contain variables related to the images.



5 Implementation

Imported the required libraries for the research

- Numpy
- Matplotlib
- Os
- Webcolors
- Pandas
- Sklearn
- Ploty
- Opencv
- PIL

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import cv2
        import PIL
        from IPython.display import Image, display
        import os
        import plotly.graph_objs as go
        import plotly.graph_objects as go
        import seaborn as sns
        from glob import glob
        import matplotlib.pyplot as plt
        import webcolors
        from PIL import Image
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import roc_curve
        from sklearn.metrics import classification_report
        from sklearn.svm import SVC
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn import metrics
```

5.1 Reading data

Reading the 2 data set using the below code.

```
In [2]: image_data = r'C:\Users\ELIZABATH K THOMAS\Documents\nci modules\research\ISIC_2020_Training_JPEG\train'

In [3]: df=pd.read_csv('C:/Users/ELIZABATH K THOMAS/Documents/nci modules/research/train.csv')
```

```
In [5]: print(df)
```

```
        image_name  patient_id     sex  age_approx  \
0      ISIC_2637011  IP_7279968    male        45.0
1      ISIC_0015719  IP_3075186  female        45.0
2      ISIC_0052212  IP_2842074  female        50.0
3      ISIC_0068279  IP_6890425  female        45.0
4      ISIC_0074268  IP_8723313  female        55.0
...             ...         ...     ...         ...
33121  ISIC_9999134  IP_6526534    male        50.0
33122  ISIC_9999320  IP_3650745    male        65.0
33123  ISIC_9999515  IP_2026598    male        20.0
33124  ISIC_9999666  IP_7702038    male        50.0
33125  ISIC_9999806  IP_0046310    male        45.0

      anatom_site_general_challenge diagnosis benign_malignant  target
0                         head/neck   unknown           benign       0
1                     upper extremity   unknown           benign       0
2                     lower extremity     nevus           benign       0
3                         head/neck   unknown           benign       0
4                     upper extremity   unknown           benign       0
...                               ...       ...              ...     ...
33121                          torso   unknown           benign       0
33122                          torso   unknown           benign       0
33123                 lower extremity   unknown           benign       0
33124                 lower extremity   unknown           benign       0
33125                          torso     nevus           benign       0

[33126 rows x 8 columns]
```

5.2 Extracting features from images

Extracting the colours of the image using webcolors. And adding it to the data frame df.

In [6]:
```
m=[]

for filename in os.listdir(image_data):
    img = Image.open(image_data+'\\'+filename)

    colors = img.getpixel((320,240))
    #print(colors)
    m.append(colors)
```

In [7]:
```
n=[]
import webcolors

def closest_colour(requested_colour):
    min_colours = {}
    for key, name in webcolors.CSS2_HEX_TO_NAMES.items():
        r_c, g_c, b_c = webcolors.hex_to_rgb(key)
        rd = (r_c - requested_colour[0]) ** 2
        gd = (g_c - requested_colour[1]) ** 2
        bd = (b_c - requested_colour[2]) ** 2
        min_colours[(rd + gd + bd)] = name
    return min_colours[min(min_colours.keys())]

def get_colour_name(requested_colour):
    try:
        closest_name = actual_name = webcolors.rgb_to_name(requested_colour)
    except ValueError:
        closest_name = closest_colour(requested_colour)
        actual_name = None
    return actual_name, closest_name
for i in m:
    requested_colour = i
    actual_name, closest_name = get_colour_name(requested_colour)

    #print ("Actual colour name:", actual_name, ", closest colour name:", closest_name)
    n.append(closest_name)
```

In [8]:
```
df['color'] = n
# df=df.append(M)
df
```

Out[8]:

	image_name	patient_id	sex	age_approx	anatom_site_general_challenge	diagnosis	benign_malignant	target	color
0	ISIC_2637011	IP_7279968	male	45.0	head/neck	unknown	benign	0	olive
1	ISIC_0015719	IP_3075186	female	45.0	upper extremity	unknown	benign	0	silver
2	ISIC_0052212	IP_2842074	female	50.0	lower extremity	nevus	benign	0	gray
3	ISIC_0068279	IP_6890425	female	45.0	head/neck	unknown	benign	0	gray
4	ISIC_0074268	IP_8723313	female	55.0	upper extremity	unknown	benign	0	silver
...	...	...	...	...	...	...	...	...	...

5.3 data cleaning and preprocessing

this step includes removing null values, Removing unwanted columns from the data, making the unbalanced data balanced and converting categorical values to numerical.

In [55]:
```
df1 = df.dropna()
```

In [56]:
```
print(df1.isnull().sum())
```

```
image_name                     0
patient_id                     0
sex                            0
age_approx                     0
anatom_site_general_challenge  0
diagnosis                      0
benign_malignant               0
target                         0
color                          0
dtype: int64
```

In [149]:
```
#df.age_approx=df.age_approx.fillna(df.age_approx.mean())
```

In [57]:
```
df3= df1.drop(columns='image_name')
df4= df3.drop(columns='patient_id')
melanoma= df4.drop(columns='benign_malignant')
melanoma=melanoma.drop(columns='diagnosis')
```

In [58]:
```
def gender_conv(x):
    if x == 'male' : return 1
    if x == 'female' : return 2
    return 0
def extrimity(x):
    if x== 'torso' : return 1
    if x=='lower extremity' : return 2
    if x== 'upper extremity' : return 3
    if x=='head/neck' : return 4
    if x=='palms/soles' : return 5
    if x=='oral/genital': return 6
    return 0

def col(x):
    if x== 'silver': return 0
    if x== 'gray': return 1
    if x== 'olive': return 2
    if x== 'maroon': return 3
    if x== 'black': return  4
    if x== 'white': return  5
    if x== 'purple': return 6
    if x== 'yellow': return 7
    if x== 'teal': return   8
```

In [59]:
```
melanoma.sex=melanoma.sex.apply(gender_conv)
melanoma.anatom_site_general_challenge=melanoma.anatom_site_general_challenge.apply(extrimity)
# melanoma.diagnosis=melanoma.diagnosis.apply(diag)
melanoma.color=melanoma.color.apply(col)
```

In [60]:
```
melanoma['target'].value_counts()
```

Out[60]:
```
0    31956
1      575
Name: target, dtype: int64
```

In [61]:
```
from sklearn.utils import resample
#create two different dataframe of majority and minority class
df_majority = melanoma[(melanoma['target']==0)]
df_minority = melanoma[(melanoma['target']==1)]
# upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples= 31956, # to match majority class
                                 random_state=42)  # reproducible results
# Combine majority class with upsampled minority class
melanoma_df = pd.concat([df_minority_upsampled, df_majority])
```

In [62]:
```
melanoma_df
```

Out[62]:

	sex	age_approx	anatom_site_general_challenge	target	color
6454	2	45.0	3	1	1.0
25238	2	25.0	2	1	1.0
16139	1	55.0	3	1	1.0
6628	2	35.0	1	1	1.0
4449	1	65.0	4	1	0.0
...	...	...	...	...	...
33121	1	50.0	1	0	1.0
33122	1	65.0	1	0	0.0
33123	1	20.0	2	0	1.0
33124	1	50.0	2	0	1.0
33125	1	45.0	1	0	1.0

63912 rows × 5 columns

5.4 Prediction

The prediction was done in 3 different machine learning techniques Random forest, Logistc Regression and support vector machine.

In [80]:
```
X =melanoma_df.loc[:,melanoma_df.columns!='target']
y= melanoma_df['target']
```

In [81]:
```
from sklearn.model_selection import train_test_split
# i.e. 70 % training dataset and 30 % test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

In [ ]:
```
```

In [82]:
```
X_test = X_test.fillna(X_train.mean())
```

In [83]:
```
X_test=np.nan_to_num(X_test.astype(np.float32))
X_train=np.nan_to_num(X_train.astype(np.float32))
y_train=np.nan_to_num(y_train.astype(np.float32))
y_test=np.nan_to_num(y_test.astype(np.float32))
```

Random forest

In [84]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100)
```

In [85]:
```
clf.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = clf.predict(X_test)

# metrics are used to find accuracy or error
from sklearn import metrics
print()

# using metrics module for accuracy calculation
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
```

```
ACCURACY OF THE MODEL:  0.7439527715375372
```

In [86]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

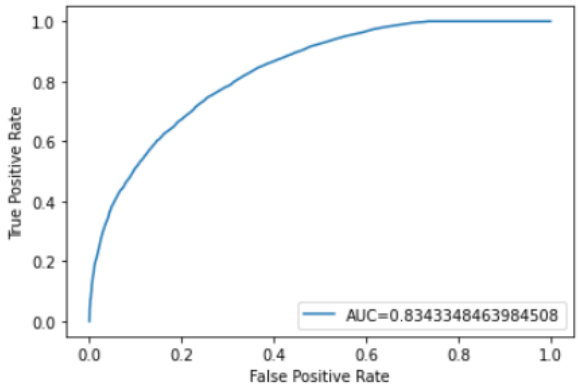
```
              precision    recall  f1-score   support

         0.0       0.75      0.72      0.74      9496
         1.0       0.73      0.77      0.75      9645

    accuracy                           0.74     19141
   macro avg       0.74      0.74      0.74     19141
weighted avg       0.74      0.74      0.74     19141
```

In [87]:
```
y_pred_proba = clf.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)

#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```



In [ ]:

Support Vector Machine

In [43]:
```
from sklearn.svm import SVC
svc = SVC(probability=True, kernel='rbf')
svc.fit(X_train,y_train)
y_predict = svc.predict(X_test)
print("SVM Accuracy:",metrics.accuracy_score(y_test,y_predict))
print("SVM F1 score:",metrics.f1_score(y_test,y_predict))
```

```
SVM Accuracy: 0.6396217543493026
SVM F1 score: 0.6056482963640522
```

In [44]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict))
```

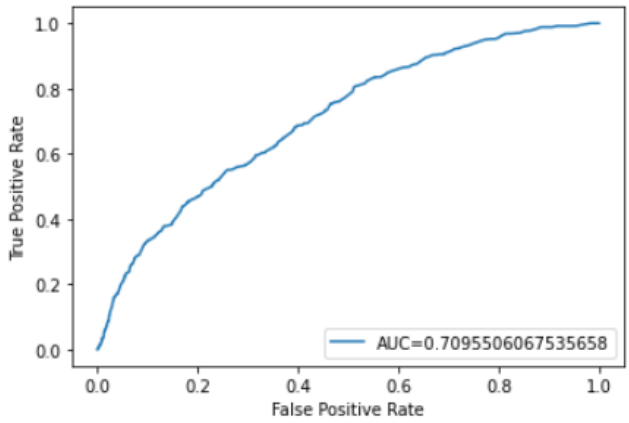
```
              precision    recall  f1-score   support

         0.0       0.63      0.72      0.67      9708
         1.0       0.66      0.56      0.61      9433

    accuracy                           0.64     19141
   macro avg       0.64      0.64      0.64     19141
weighted avg       0.64      0.64      0.64     19141
```

In [45]:
```
y_pred_proba = svc.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)

#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```



Logistic Regression

In [88]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [89]:
```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[89]:
```
LogisticRegression()
```

In [90]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.65
```

In [91]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.65      0.65      0.65      9496
         1.0       0.66      0.65      0.65      9645

    accuracy                           0.65     19141
   macro avg       0.65      0.65      0.65     19141
weighted avg       0.65      0.65      0.65     19141
```

In [92]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```