BHASKARACHARYA NATIONAL INSTITUTE FOR SPACE APPLICATIONS AND GEO-INFORMATICS

WEEKLY PROGRESS REPORT (27/03/2023 – 02/04/2023)

WEEK 10

PROJECT NAME	MALWARE DETECTION USING ML

PROJECT DESCRIPTION :	**DESIGN AND IMPLEMENT ML MODEL TO DETECT MALWARE IN SYSTEM**
GROUP MEMBER :	**PRATHAM PATEL, Shubham Patel, Shashank Sharma, Yash Soni**
GROUP ID :	**I2**
GROUP GUIDE :	**HARSH KIRATSATA**
GROUP COORDINATOR :	**SIDHDHARTH PATEL**
COLLEGE GUIDE NAME :	**MAYANK PATEL, RITIKA LADHA, KINJAL CHAUDHARI**
COLLEGE GUIDE No. :	**+91 9033280445,8866438698, 9723173567**
COLLEGE GUIDE EMAIL. :	**mayank.patel@adaniuni.ac.in, ritika.ladha@adaniuni.ac.in, kinjal.chaudhari@adaniuni.ac.in**
COLLEGE NAME :	**ADANI INSTITUTE OF INFRASTRUCTURE AND ENGINEERING**

27/03/2023 TILL 02/04/2023 (7 DAYS)

27/03/2023	**Deciding Learning rate for model.**
28/03/2023	**Programming for normalization and train test split.**
29/03/2023	**Coding stat matrices, architect.**
30/03/2023	**Holiday(Ram navmi)**
31/03/2023	**Programing remaining functions(gradient descent, placeholders)**
01/04/2023	**Debugging and model improvisation.**
02/04/2023	**Holiday (Sunday)**

WEEK 11(PLAN)	**In the next week we are planning to implement this model for other viruses and malwares.**

REFERENCE:

- https://research.google/pubs/pub46484/
- https://www.kdd.org/kdd2018/hands-on-tutorials/view/deep-learning-with-keras
- https://www.researchgate.net/publication/332824465_Intrusion_Detection_by_Deep_Learning_with_TensorFlow

Screenshots :

```
A=x_train.shape[1] #features ~about 40 features
B=len(y_train_onehot[0])    #columns

############################################################
## print stats

precision_scores_list = []
accuracy_scores_list = []

def print_stats_metrics(y_test, y_pred):
    print('Accuracy: %.2f' % accuracy_score(y_test,   y_pred) )
    #Accuracy: 0.84
    accuracy_scores_list.append(accuracy_score(y_test,   y_pred) )
    confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
    print ("confusion matrix")
    print(confmat)
    print (pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted']
    precision_scores_list.append(precision_score(y_true=y_test, y_pred=y_pred))
    print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
    print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
    print('F1-measure: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

```
[[ 97212    109]
 [   149 392373]]
Predicted      0       1     All
True
0          97212     109   97321
1            149  392373  392522
All        97361  392482  489843
Precision: 1.000
Recall: 1.000
F1-measure: 1.000
epoch 999 out of 1000
----------------------------------------
------------------------------
Accuracy score
Run 999,0.9994732737541199
Accuracy: 1.00
confusion matrix
[[ 97212    109]
 [   149 392373]]
Predicted      0       1     All
True
0          97212     109   97321
1            149  392373  392522
All        97361  392482  489843
Precision: 1.000
Recall: 1.000
F1-measure: 1.000
<<<<<<DONE>>>>>>
```

```
def layer(input, weight_shape, bias_shape):
    weight_stddev = (2.0/weight_shape[0])**0.5
    w_init = tf.random_normal_initializer(stddev=weight_stddev)
    bias_init = tf.constant_initializer(value=0)
    W = tf.get_variable("W", weight_shape, initializer=w_init)
    b = tf.get_variable("b", bias_shape, initializer=bias_init)
    return tf.nn.relu(tf.matmul(input, W) + b)

##################################################################
def inference_deep_layers(x_tf, A, B):
    with tf.variable_scope("hidden_1"):
        hidden_1 = layer(x_tf, [A, 30],[30])
    with tf.variable_scope("hidden_2"):
        hidden_2 = layer(hidden_1, [30, 20],[20])
    with tf.variable_scope("hidden_3"):
        hidden_3 = layer(hidden_2, [20, 15],[15])
    with tf.variable_scope("hidden_4"):
        hidden_4 = layer(hidden_3, [15, 10],[10])
    with tf.variable_scope("output"):
        output = layer(hidden_4, [10, B], [B])
    return output
##################################################################
def loss_deep(output, y_tf):
    xentropy = tf.nn.softmax_cross_entropy_with_logits(logits=output, labels=y_
```