International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)
Manuscript Received: 03 July 2021, Received in Revised form: 08 August 2021, Accepted: 13 August 2021 DOI: 10.46338/ijetae0821_02

Rapid Software Framework for the Implementation of Machine Learning Classification Models

Abdullah Sani Abd Rahman[1], Suraya Masrom[2], Rahayu Abdul Rahman[3], Roslina Ibrahim[4]
[1]*Faculty of Science and Information Technology, Universiti Teknologi Petronas, Perak, Malaysia.*
[2]*Faculty of Computer and Mathematical Sciences, Universiti Teknologi Mara, Perak Branch, Tapah Campus, Malaysia.*
[3]*Faculty of Accounting, Universiti Teknologi Mara, Perak Branch, Tapah Campus, Malaysia.*
[4]*Razak Faculty of Technology & Informatic, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia*

***Abstract*- Reseachers have acknowledged that machine learning is useful to be utilized in many different domains of complex real life problem. However, to implement a complete machine learning model involves some technical hurdles such as the steep learning curve, the abundance of the programming skills, the complexities of hyper-parameters, and the lack of user friendly platform to be used for the implementation. This paper provides an insight of a rapid software framework for implementing machine learning. This paper also demonstrates the empirical research results of machine learning classification models from the rapid software framework. Additionally, this paper explains comparisons of results between two platforms of rapid software; the proposed software and Python program. The machine learning model in the two platforms were tested on breast cancer and tax avoidance datasets with Decision Tree algorithm. The results indicated that although the software framework is easier than the programming platform for implementing the machine learning model, the results from the software framework were highly accurate and reliable.**

***Keywords*- Software framework, rapid, implementation, machine learning**

I. Introduction

Nowadays, deployment of machine learning has moved from the expert users of computer scientists into the inexperienced users of many domains. From academia to industries, they started to realize on machine learning benefits to daily life applications[1]. Despite opportunities, a sort of challenges appeared to implement machine learning. To gain an adequate skill instantly is not possible because it involves vary of knowledges on concept, mathematical, technical and programming. Thus, easy toolkit or rapid software will be useful to them while simultaneously benefit to the expert. With the toolkit, expert users can conduct rapid and extensive trials of the machine learning evaluations.

Due to the rising demands of machine learning by the inexpert users and the requirement of rapid platform by expert data scientists, automation of some process of machine learning has been the important subject of many recent works[2]. Most of the works used scripting programming as the front-end of the toolkits, which remain difficulty to the novice.

Therefore, the objective of this paper is to introduce a software framework that support simple and fast implementation with Graphical User Interface (GUI). Since the software framework was an enhancement from the original toolkit, this paper presents the software architecture and the related components for software expansion. Additionally, this paper provides the finding of research that tested the rapid software on two datasets; breast cancer and tax avoidances. It demonstrates the results based on different input of algorithm configurations. Before the concluding remarks, a discussion about the proposed software with Python toolkit is provided.

II. Literature Review

There exists excellent toolkits software that support rapid implementation of machine learning in different kinds of platform[3]. These platforms provides GUI or scripting based programming as to enable the rapid implementation. In [4], the researchers used GUI as the front-end to R program. Similar for R program, the GUI supported by [5] can be used to implement machine learning for brain tumor detection. Machine learning based model for the heart stroke prediction can be implemented with easy GUI in [6]. PyLZJD is also a GUI based machine learning to assist users to implement image classification and recognition[7]. There exists software that used scripting based programming to allow rapid implementation of machine learning. In [8], comparisons of different scripting tools has been done.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

The tools are named as AutoML or automated machine learning. In [9], the scripting program can be written by the data scientist users in a simplified form to be then converted into different programming languages. This software framework can generate the machine learning codes either in C/C++, JAVA from the GUI input and scripting. Similar with machine learning codes generator in [10], from the front-end GUI, the toolkit can generate Python codes but used for Deep learning, convolutional neural network and images classification applications only. A scripting based programming is supported in toolkit [11] to allow users to write the codes in simple declarative programming for users to design their machine learning model. Similar idea in [12], from the users scripts written in R, the toolkit translates the codes into a HTML website so that users can view their machine learning modeling. Web based modelling interface for management of machine learning is supported in [13]. Users have to write scripting of programming either in Python or R to be converted into a machine learning flowcharts. To the best of our knowledges, these software remain some difficulties to novice users as the GUI as well as the scripting consists an assortment of input configurations that requires details understanding on the machine learning theories and concepts.

III. Research Method

A. The software framework

The proposed software framework used in this research is a free software, which the original source codes in Python programming can be downloaded from the developer websites at https://github.com/anchal27sri/ML-GUI-using-PyQt5. In this paper, we describe the extendable modules of the software framework. Rapid and extendable are two advantages features of the software framework[14]. GUI of the software at the front-end is associated with rapid and easy implementation. The back-end of the software is a set of Python codes developed by the developer based on object oriented paradigm hence flexible modification or amendment can be done to suit the expert computer scientists' need. Figure 1 presents the software framework architecture.

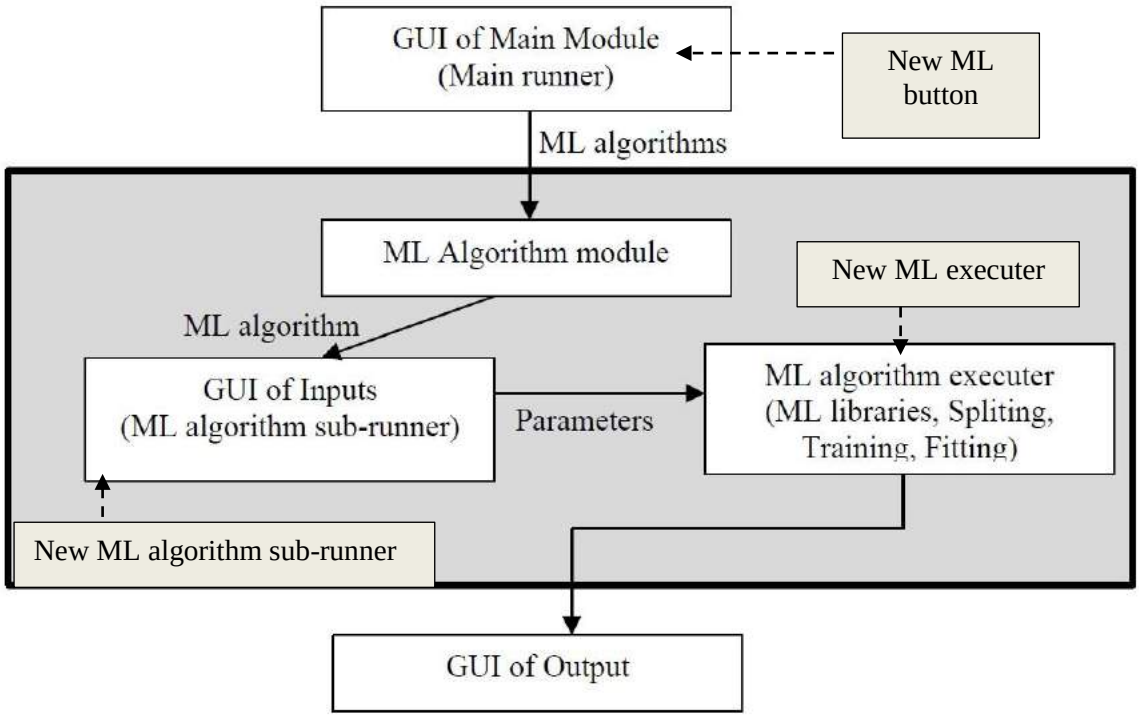


Figure 1 The architecture of the software framework

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

Components with dotted arrow in Figure 1 denote that the modules are extendable for new button, machine learning sub-runner and executers respectively.

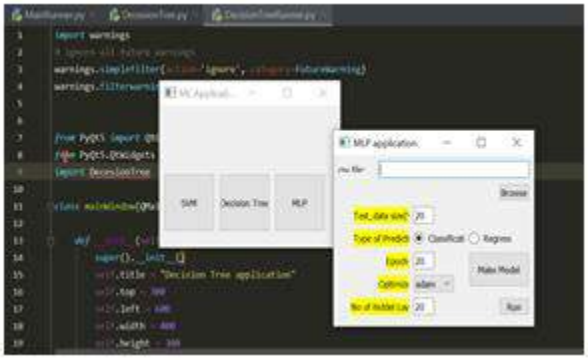


Figure 2 The original software framework

The original software only supported three kinds of machine learning algorithms namely MLP, SVM and Decision Tree as in Figure 2. Extending the main runner, sub-runner and ML algorithm executer has been done to provide five machine learning algorithms as in Figure 3.

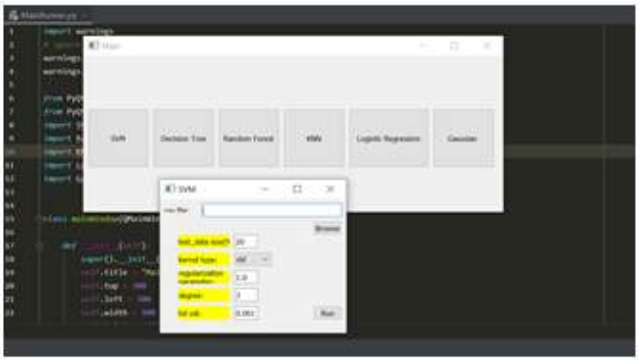


Figure 3 The extension software framework

Refer to Figure 1, the main runner modules can be added with new machine learning sub-runner as shown in Figure 4.

```
def callSVMRunner(self):
    self.m = SVMRunner.Main()

def callDecesionTreeRunner(self):
    self.m = DecisionTreeRunner.Main()

def callRandomForestRunner(self):
    self.m = RandomForestRunner.Main()
```

Figure 4 Python codes in part of main runner module

Once the sub-runner has been created in the main runner module, two Python files for the machine learning sub runner and executors have to be created. For examples, if Random Forest algorithm is created, the *DecisionTree.py* and *DecisionTreeRunner.py* must be created. By using the available machine learning sub runner, replication of codes can be done. Modification on the *runDT* method of the sub-runner to call the Decision Tree Runner via the *DecisionTree.run()* statement. See Figure 5.

The *DecisionTree.py* is the program codes to execute the individual machine learning algorithm that calls the library of the Decision Tree algorithm from *sklern* library and calling the Decision Tree classifier as depicted in Figure 6.

PyQt5 is the library used for the GUI. It consists 35 extension modules that can be used including *Qt Widgets.* There are several UI elements in *QtWidget* modules such as QLabel, QLineEdit, QTextEdit,QPushButton, QradioButton and QcomboBox.

```
def runDT(self):
    if self.fileName != "":
        self.splitSize = int(self.split_lineEdit.text())
        if self.splitSize <=40:
            if self.splitter_button1.isChecked() is False:
                self.splitter = 'random'
            if self.crit_button1.isChecked() is False:
                self.criterion = 'entropy'
            self.results = DecisionTree.run(self.fileName,self.splitSize,self.criterion,self.splitter)
        else:
            pass print("cannot train on such small dataset")
    else:
        pass  print("incorrect file name!")
```

Figure 5 Python codes in the sub runner module

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

```
#Y.ravel()
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=ts,random_state=109)

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
#print(X_train)
#Y_train.ravel()
clf = DecisionTreeClassifier(criterion = crit,splitter = split)
clf.fit(X_train,Y_train)

Y_predict = clf.predict(X_test)
cm = np.array(confusion_matrix(Y_test, Y_predict, labels=[0,1]))
```

Figure 6 Python codes in the machine learning executor module

In this paper, we focused on the Decision Tree. Figure 6 is the GUI to run Decision Tree. Users can select the dataset from the Browse button.

B. The experiments

Based on the GUI in Figure 7, the testing size for all the models was set to 20%. Therefore, 80% from the total dataset numbers were used for training.

The machine learning models of the Decision Tree were named according to the criterion and splitter types. The models' name are GiniBest, GiniRandom, EntropyBest and EntropyRandom.

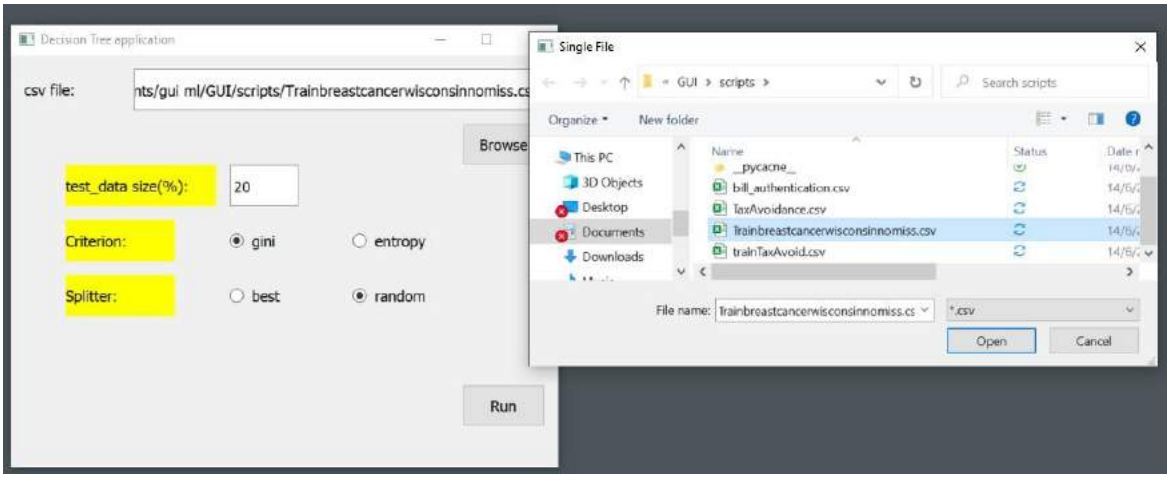


Figure 7 The GUI for Decision Tree

C. Datasets

In order to test the results of Decision Tree in the software framework, two datasets were used. The first dataset is related to breast cancer and the second is on tax avoidance. Both applications are a kind of binary classification where the 1 value indicates positive breast cancer or the occurrence of tax avoidance. The number of records for the breast cancer and tax avoidance dataset is 500 and 75 respectively.

The breast cancer dataset is a benchmark dataset widely used in machine learning research downloaded from the Kagle.com. However, the tax avoidance dataset is a real cases on Government-Linked Companies in Malaysia in 2001-2006. From the 500 breast cancer dataset, 100 records were used for testing and 400 for training. Besides, 15 records from tax avoidance dataset were set for testing and the rest 60 records were used for training. Information about the tax avoidance dataset can be referred in our previous research reported in [15], [16].

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

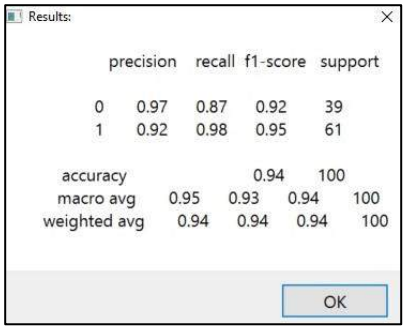
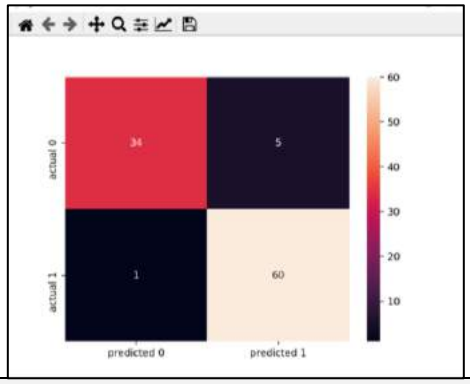
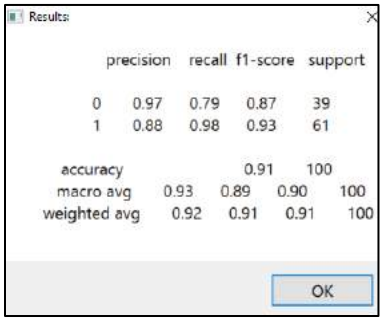
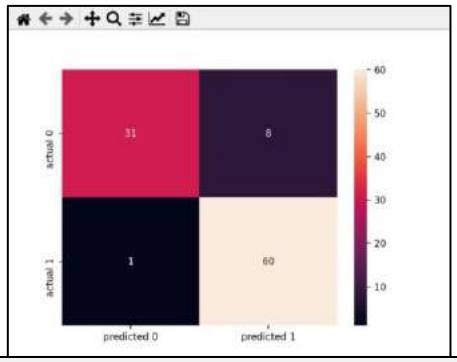
IV. Results And Discussion

The results can be viewed in two windows either metrics window or confusion matrix window. Figure 8 presents the results of GiniBest and Gini Random of Decision Tree when tested on the breast cancer dataset.

As shown in the confusion matrix in Figure 7, the Decision Tree testing has taken 100 out of 500 records from the dataset of breast cancer (20% test sampling). All the f1-score results from the four models are very high (above 90%) with precision value between 88-92 percentages. Precision is the sentitivity ability of the models in detecting the positive cases of breast cancer.

Decision Tree with GiniBest showed the best ability in detecting the positive cancer. The software framework correctly calculate the positive and negative cases as all models displayed 39 negative cases and 61 positive cases, in total 100 testing cases (at the suppot column of the metrics window).

The same breast cancer dataset was used to be tested with the Decision Tree algorithm implemented with Python programming in Jupyter Notebook platform as presented in Figure 8. The accuracy is 92%, which is in the ranges of the results in Figure 9. Therefore, it can be considered that both results from the two platforms are reliable.

<table>
<tr><td colspan="2">Decision Tree with GiniBest Model</td></tr>
<tr><td>Metrics Window</td><td>Confusion Matrix Window</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Decision Tree with GiniRandom Model</td></tr>
<tr><td>Metrics Window</td><td>Confusion Matrix Window</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Decision Tree with EntropyBest Model</td></tr>
<tr><td>Metrics Window</td><td>Confusion Matrix Window</td></tr>
</table>

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

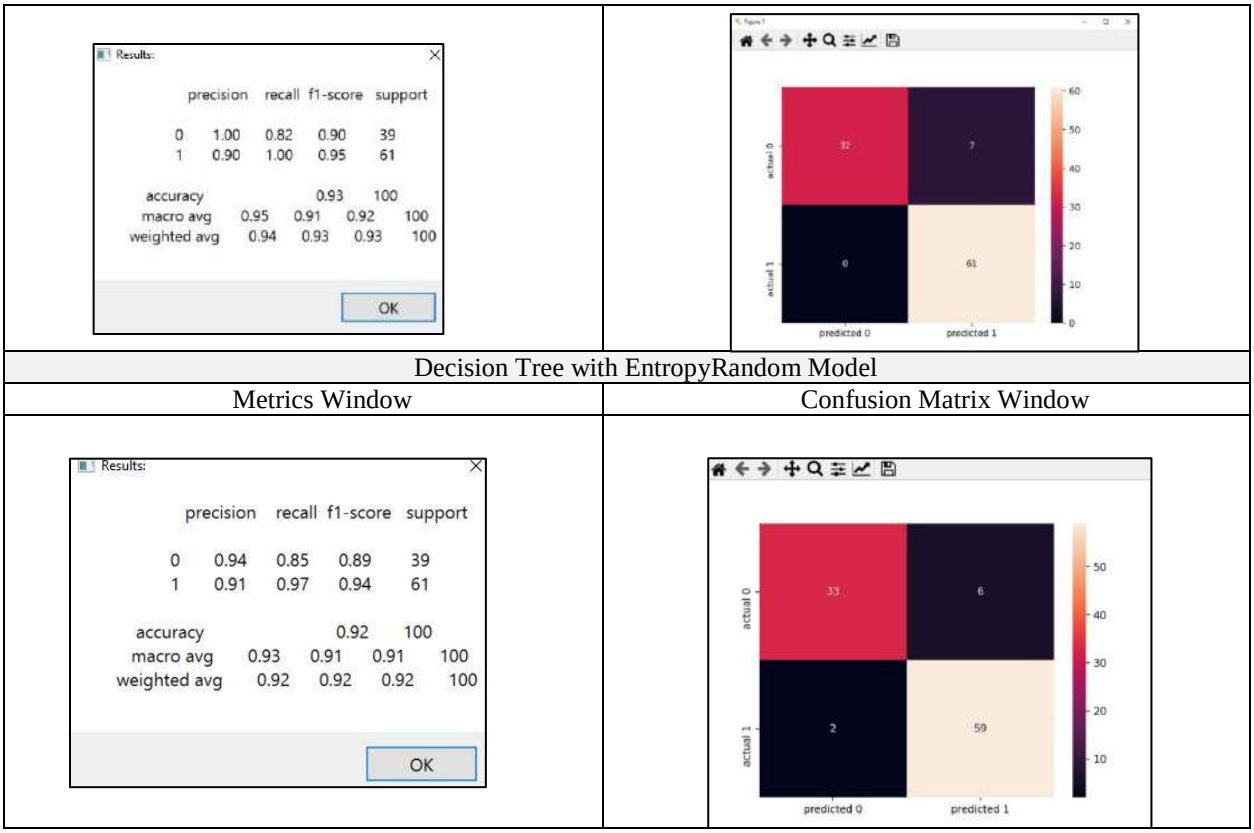


Figure 8 Results of Decision Tree on breast cancer dataset from the software framework

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

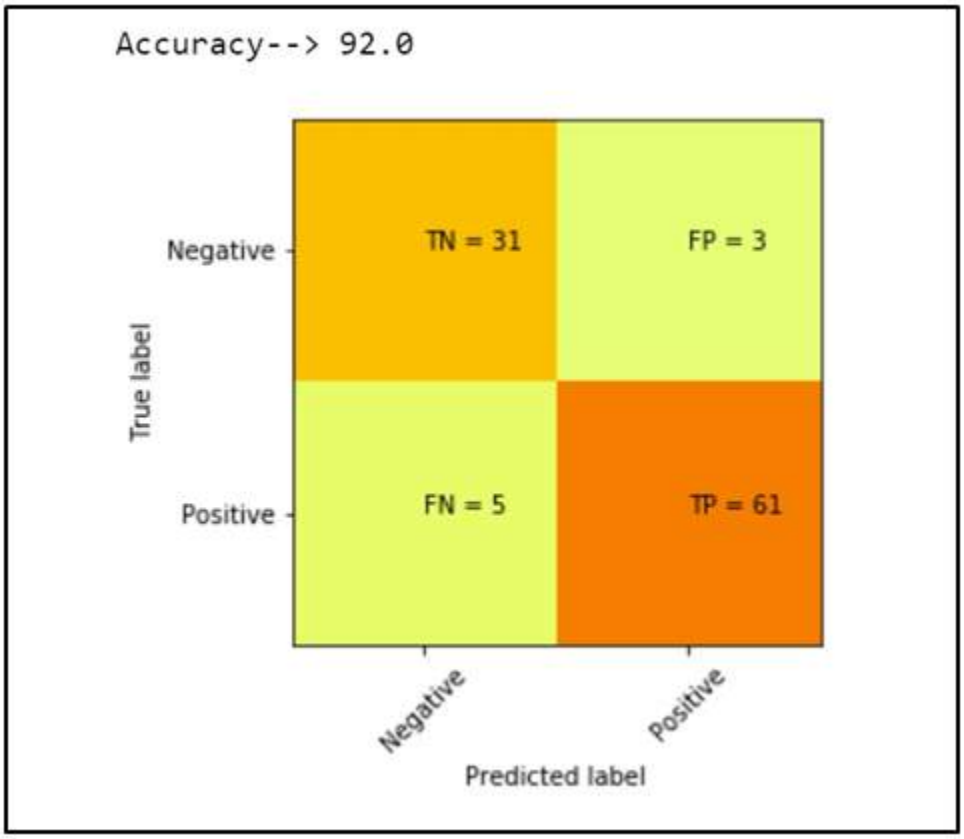


Figure 9 Results of Decision Tree on breast cancer dataset from the Python program

Futhermore, the results in Figure 10 presents the accuracy and the confusion matrix from Decision Tree on the tax avoidance dataset. The software framework correctly used 20% of the dataset for the machine learning testing as the number of support in total from the metrics window is 15 from the 75 records. The ability to detect the occurance of tax avoidance is between 58-80 percentages, which the highest presision provided by GiniRandom model(80%). Similarly, the ability to detect truly both cases is 80% presented by the accuracy result.

In Decision Tree, gini and entropy are different formulation on calculating information to split a set of nodes while developing subtree branch. On spliting the nodes, the index information can be used to decide which feature should be chosen as the parent of the next sub tree. Equation 1 and Equation 2 are the formula for gini and entropy.

$$Gini = 1 - \sum_{i=1}^{n} p^2\,(c_i) \quad (1)$$

$$Entropy = \sum_{i=1}^{n} - p(c_i) log_2\,(p(c_i)) \quad (2)$$

where $p(c_i)$ denotes the probability of that a target feature c in a node i. Gini index can be determined by subtracting 1 with the sum of each feature squared probabilites. Therefore, the gini index always between 0 to 0.5. On the other hand, entropy index lies between 0 to 1 as it measure the collected information of a random features to be substracted with the total entropy.

Therefore, the effect of gini or entropy in a Decision Tree is highly dependent on the tested dataset values[17].

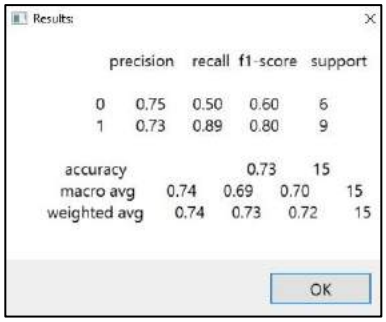
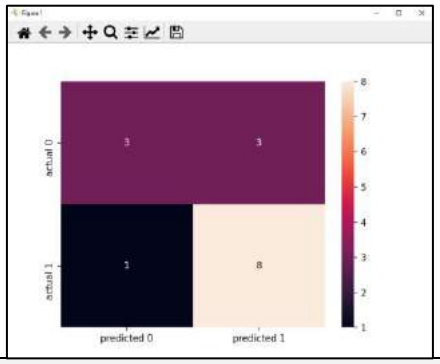
On the tax avoidance dataset, there seem a obvious constrast between the Decision Tree criterion setting (gini vs entropy). However, in the breast cancer case, both gini and entropy criterion did not sign a big change on the Decision Tree performances (Refer Figure 8).

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

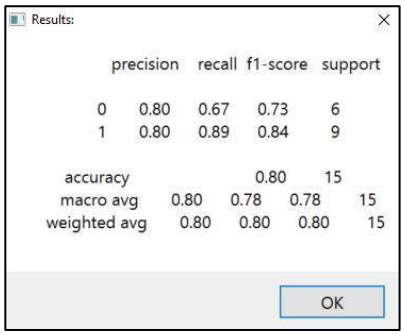
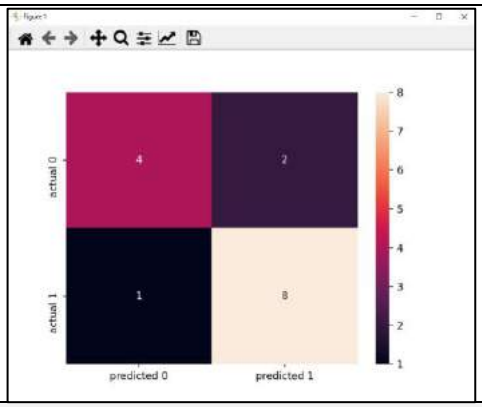
Figure 11 is the results of Decision Tree tested on the tax avoidance dataset implemented with Python in Jypyter Notebook platform. The results in Figure 11 disclosed that the software framework able to produce reliable results like Python programming mainly with the GiniBest and GiniRandom models.

Figure 12 shows the Python program codes to call and run Decision Tree. The codes can be complicated for novice users.

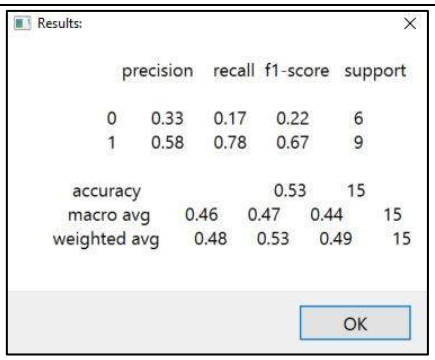
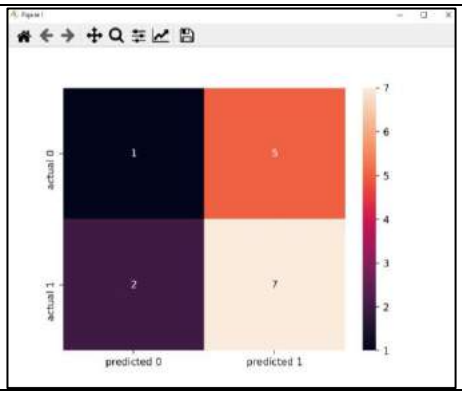
Decision Tree with GiniBest Model

Metrics Window	Confusion Matrix Window
	

Decision Tree with GiniRandom Model

Metrics Window	Confusion Matrix Window
	

Decision Tree with EntropyBest Model

Metrics Window	Confusion Matrix Window
	

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

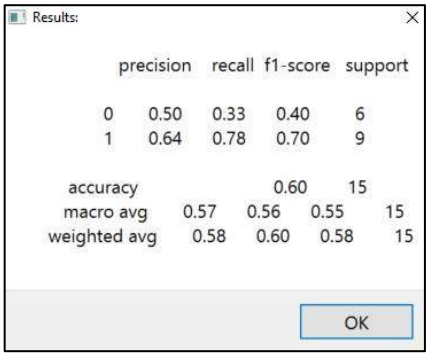
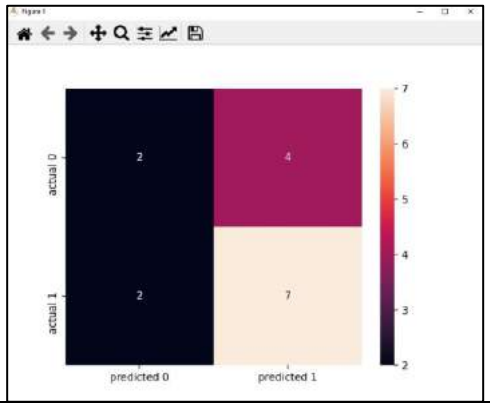
Decision Tree with GiniRandom Model	
Metrics Window	**Confusion Matrix Window**
	

Figure 10 Results of Decision Tree on tax avoidance dataset from the software framework

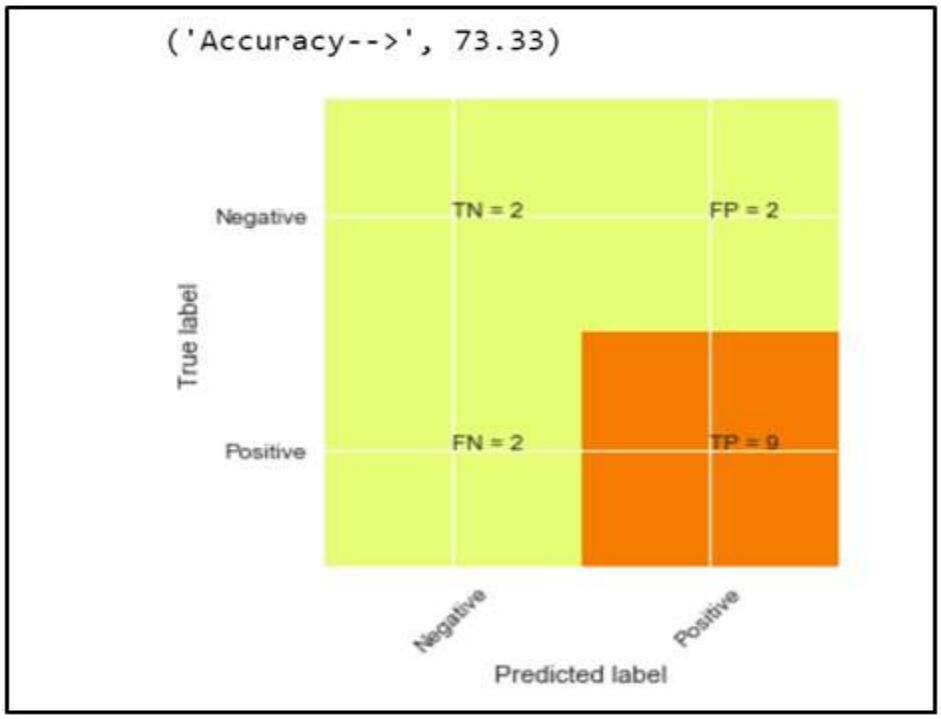


Figure 11 Results of Decision Tree on tax avoidance dataset from the Python program

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 08, August 2021)

```
from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve

decision_tree = DecisionTreeClassifier(max_depth=7)
decision_tree.fit(Feature_Train, Class_Train)
Y_pred = decision_tree.predict(Feature_Test)
acc_decision_tree = round(decision_tree.score(Feature_Test, Class_Test) * 100, 2)
print("Accuracy-->",  acc_decision_tree)
conmat = confusion_matrix(decision_tree.predict(Feature_Test), Class_Test)
plt.imshow(conmat, interpolation='nearest', cmap=plt. conmat.Wistia)
classNames = ['0-Negative','1-Positive']
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['True Negative','False Positive'], ['False Negative', 'True Positive']]
for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(conmat[i][j]))
plt.show()
```

Figure 12 Python codes to call Decision Tree and display results as in Figure 7 and Figure 9

Based on the two cases of dataset testing, it can be thought that the software framework is functional to generate accurate results as Python program. Although the rapid software used GUI to support easy platform for the machine learning, the software is reliable to be used as the Jupyter Notebook platform that Python used programming.

Table I describes the implementation comparisons between the software framework and with the Python program in Jupyter Notebook. Rapid implementation means no complex programming that needs more than 200 lines of codes. Extendable criteria is to allow expert users to add new machine learning algorithms or other functionalities.

TABLE I
COMPARISON OF THE PROPOSED SOFTWARE FRAMEWORK WITH PYTHON PLATFORM

Criteria	Software framework	Python in Jupyter Notebook
Rapid implementation	Yes and very simple with GUI	Yes but can be complicated to novice users
Extendable	Yes	Yes
Installation	Easy	Complicated with different libraries installation
Type of users	Novice and expert	Expert users with good programming skill

The software framework is very simple to be used by novice users such as students or researchers who need to learn about machine learning models. Although Python is a kind of scripting programming language, the codes can be complicated and tricky to be grasped by these users.

V. CONCLUSION

This paper presents the review and empirical works for the implementation of machine learning classification model based on rapid software framework and Python program.

Despite the advantages to support rapid and extensive implementation of machine learning models, the software framework nonetheless has some limitation that should be improved. The main extension on the software framework can focus on the following:

- Currently, there are no option is provided for users to set the splitting approach for training and testing.
- There is no GUI to allow users to select features from the dataset. Currently, the possible way is to set the features on the dataset with Microsoft Excel.
- The software framework can only be used for machine learning classification models but not for prediction model.

Acknowledgement

This project was funded by the Ministry of Education, Malaysia for project under FRGS(600-IRMI/FRGS5/3 (140/2019). Thank you to the Ministry of Education, Malaysia and the Universiti Teknologi MARA for the support.

REFERENCES

[1] M. Mohammed, M. B. Khan, and E. B. M. Bashie, Machine learning: Algorithms and Applications, no. July. 2016.

[2] J. Waring, C. Lindvall, and R. Umeton, "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," Artif. Intell. Med., vol. 104, no. January, p. 101822, 2020, doi: 10.1016/j.artmed.2020.101822.

[3] R. S. Olson and J. H. Moore, "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning," in Automated Machine Learning: Methods, Systems, Challenges, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 151–160.

[4] G. Williams and others, "Rattle: a data mining GUI for R," 2009.

[5] D. Sherawat, Sonia, and A. Rawat, "Brain Tumor Detection Using Machine Learning in GUI," in Proceedings of Integrated Intelligence Enable Networks and Computing, 2021, pp. 9–17.

[6] Y. P. Kadtan, A. P. S. Chauhan, and R. Brindha, "GUI based Prediction of Heart Stroke Stages by finding the accuracy using Machine Learning algorithm," Ann. Rom. Soc. Cell Biol., pp. 4571–4577, 2021.

[7] E. Raff, J. Aurelio, and C. Nicholas, "PyLZJD: An easy to use tool for machine learning," UMBC Fac. Collect., 2019.

[8] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools," in 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, pp. 1471–1479.

[9] O. Castro-Lopez and I. F. Vega-Lopez, "ML2ESC: A source code generator to embed machine learning models in production environments," in Proceedings of the international conference on data science, CSREA, Las Vegas, USA, 2018, vol. 14, pp. 70–73.

[10] P. F. Ferreira Pereira, F. Rodrigues, and C. Ferreira, "Code Generator from Mockups," in 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019, pp. 1–7, doi: 10.23919/CISTI.2019.8760681.

[11] M. E. Schüle, M. Bungeroth, A. Kemper, S. Günnnemann, and T. Neumann, "Mlearn: A declarative machine learning language for database systems," in Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning, 2019, pp. 1–4.

[12] K. Romaszko, M. Tatarynowicz, M. Urbański, and P. Biecek, "modelDown: automated website generator with interpretable documentation for predictive machine learning models," J. Open Source Softw., vol. 4, no. 38, p. 1444, 2019.

[13] M. Vartak et al., "ModelDB: a system for machine learning model management," in Proceedings of the Workshop on Human-In-the-Loop Data Analytics, 2016, pp. 1–3.

[14] S. Petrovic, P. Milosavljevic, and J. Lozanovic Sajic, "Rapid evaluation of maintenance process using statistical process control and simulation," Assessment, vol. 9, p. 16, 2018.

[15] S. Masrom, R. A. Rahman, N. Baharun, and A. S. A. Rahman, "Automated Machine Learning with Genetic Programming on Real Dataset of Tax Avoidance Classification Problem," in Proceedings of the 2020 9th International Conference on Educational and Information Technology, 2020, pp. 139–143.

[16] R. A. Rahman, S. Masrom, and N. Omar, "Tax Avoidance Detection Based on Machine Learning of Malaysian Government-Linked Companies," no. 2, pp. 535–541, 2019, doi: 10.35940/ijrte.B1083.0982S1119.

[17] K. Mathan, P. M. Kumar, P. Panchatcharam, G. Manogaran, and R. Varadharajan, "A novel Gini index decision tree data mining method with neural network classifiers for prediction of heart disease," Des. Autom. Embed. Syst., vol. 22, no. 3, pp. 225–242, 2018.