ICIC Express Letters
Volume **14**, Number **3**, March **2020**

ICIC International ©2020 ISSN 1881-803X
pp. **219–226**

THE EVALUATION OF SUPERVISED CLASSIFIER MODELS TO DEVELOP A MACHINE LEARNING API FOR PREDICTING CARDIOVASCULAR DISEASE RISK

Ida Bagus Kerthyayana Manuaba[1], Indrajani Sutedja[2]
and Raymond Bahana[1]

[1]Computer Science Department, Faculty of Computing and Media
[2]Information Systems Department, School of Information Systems
Bina Nusantara University
Jl. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
{ imanuaba; rbahana }@binus.edu; indrajani@binus.ac.id

Received August 2019; accepted November 2019

Abstract. *This paper discusses the development of a machine learning API (Application Programming Interface) to predict cardiovascular disease risk based on medical record information as the input features. For building a supervised machine learning to predict cardiovascular disease risk, a best classifier is tested and selected to fit with the model data. In selecting the best classifier model, five classifiers were used to train the data. In this process, each classifier used sixty thousand medical records from Mayapada private hospital for data training and testing. The data was initially divided into 38 attributes. The training model of each classifier was measured and evaluated using classification accuracy, classification error and F-measurement. As a result, Gradient Boosted Trees classifier was selected as the best classifier with the highest accuracy rate more than 80%, F-measure is more than 92% and the classification error is less than 12%. Hence, a backend machine learning API was built based on this Gradient Boosted Trees classifier and it has been tested for training and predicting the cardiovascular disease risk through the Postman application.*

Keywords: Supervised machine learning, API (Application Programming Interface), Classifier

1. **Introduction.**

Nowadays, the trend of Machine Learning (ML) has been widely discussed for many applications and research areas. Based on Goldberg and Holland [1], ML can be defined as a tool in computer science that could be effectively used to perform specific tasks, such as classifying, clustering, and predicting processes. Classification algorithm is a core of all processes in ML, especially for supervised machine learning. There are a lot of classification algorithms today, for example: Decision Trees [2,3], Naïve Bayes [4,5], Artificial Neural Networks (ANN) [6], and k-Nearest Neighbor (KNN) algorithms [7].

According to Reimer et al. [8], a predicting process could lead into the process of detection. In the world of medical health, a detection of disease risk is an important stage that could be used to prevent an illness. Early detection could also be used to improve the successful outcome of treatment.

Cardiovascular disease is one of major deadly illnesses that have a devastating effect on society [9]. In developing countries, such as Indonesia, cardiovascular disease is the highest ranked disease that contributes 35% to the proportional mortality rate based on a WHO report in 2018 [10]. Based on this high-risk of cardiovascular disease in Indonesia, it is important to find a solution to prevent this disease at an early stage. Many factors in

DOI: 10.24507/icicel.14.03.219

people's daily lives also cause limitations to accessing information and consulting experts about their symptoms.

In Indonesia, based on the Digital in 2017: Global Overview report [11], the usage of mobile applications was recorded at 70% for daily usage. Many applications from social media, finance, games and also health applications can be accessed from mobile phones. Hence, mobile application is a potential media that could be utilized to facilitate people accessing information.

The advance of today's technology has indicated a possibility to integrate mobile applications and machine learning [12]. A book by NG [13] stated that Machine Learning (ML) has already served in various fields and now is the time to serve mobile application development. They also mentioned that for healthcare applications, ML could also be used to take care of human health. It is not limited only to providing information in taking care and maintaining human health, it is also possible to substitute the physician temporarily in predicting people's sickness based on their symptoms.

This paper discusses the building of a supervised machine learning API (Application Programming Interface) that can be used to predict the cardiovascular disease risk based on blood test results as the input. The API if trusted could provide an easy implementation and interaction with machine learning that could be accessed from any device. For this purpose, five model classifiers were used to train the medical record data and will be assessed and evaluated using several evaluation techniques. The best of classifier evaluation results in the training and testing model is used as a model classifier of machine learning application with API. Hence it could be connected to any devices to predict someone's cardiovascular disease risk based on specific blood test results as the input attribute.

This paper will discuss the methods, including source data and all stages of the research followed by more discussion in the implementation and detail process of each stage on building a machine learning API. A number of measurement and evaluation methods are also described more comprehensively in the following sections.

2. Methods.

In building a supervised machine learning API to predict the cardiovascular disease risk, we required sample data for training and testing process. For this research, the data that was used for training and testing processes were 60,588 medical records from *Mayapada* Hospital. These medical records were divided into 38 attributes. These attributes were divided into 7 patient's general information categories and 31 of their blood test components. Selecting the right attributes was important in preparing clean data since it would be used as the data features for the Machine Learning (ML) process.

In order to build the ML application, we needed to select or create a model classifier for training the data. Based on the data source above, in this research we tested a number of exciting classifier models that were utilized in previous related research scenarios. This classifier model was measured and evaluated using several evaluation techniques which were expected to help us get the best fit of classifier model for our data and scenario. More detailed information regarding the model classifiers and evaluation techniques is discussed in the following implementation section. Before that, we will explain the stages for this research.

This research was designed and conducted by following three main stages, they were 1) preparing data stage, 2) building a machine learning model, and 3) creating/applying the ML APIs for predicting cardiovascular disease risk based on model and evaluation of ML training and testing. Figure 1 shows the information in each stage of this research.

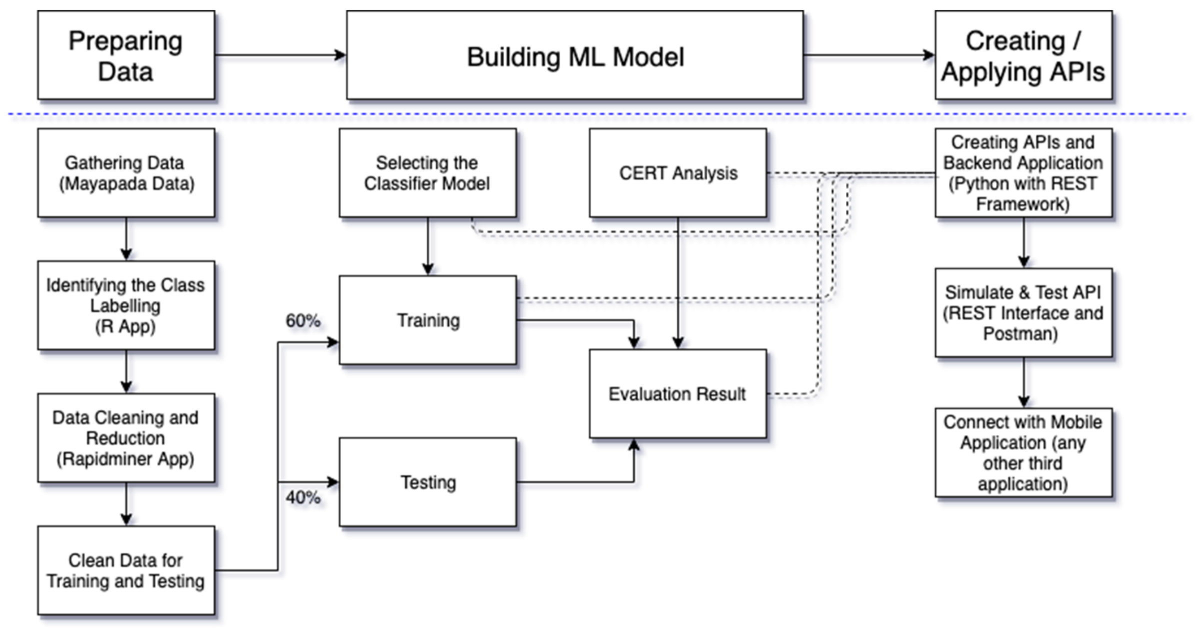

Preparing Data
Building ML Model
Creating / Applying APIs
Gathering Data (Mayapada Data)
Identifying the Class Labelling (R App)
Data Cleaning and Reduction (Rapidminer App)
Clean Data for Training and Testing
Selecting the Classifier Model
Training
Testing
60%
40%
CERT Analysis
Evaluation Result
Creating APIs and Backend Application (Python with REST Framework)
Simulate & Test API (REST Interface and Postman)
Connect with Mobile Application (any other third application)

FIGURE 1. Methods: Three main stages in the research method

3. Implementation, Result and Discussion.

3.1. Preparing data.

For data preparation, it was started by adding a new attribute named ‘condition’ into the data for assigning a status to each medical record. This condition label is categorized into ‘Risk’ and ‘Healthy’. By utilizing Rapplication we created a formula to generate the value of condition labels for each medical record. The formula was created by following the standardization of normal conditions for blood test results [14].

Next the researcher had to clean and reduce the data. In this stage, we identified and excluded any incomplete, incorrect, inaccurate and irrelevant data which might cause problem for training process. These two steps were important to produce clean data input for training and testing of the ML model.

Using *RapidMiner application* [15], we found that 33 out of 38 attributes could be used as data features. In addition, we could see that there were 77% (46,599 records) identified as Risky and 23% (13,989 records) as Healthy. We could also see the quality of the data, specifically the quality of each column data. This data quality could be measured by the correlation, ID-ness, stability, missing, and text-ness values.

In general, attributes for features in ML should have low values for missing, stability, and ID-ness. A high percentage of text-ness is normally avoided since too much text will make the ML process slower. Attributes with high correlations are typically preferred, but not if the high correlation occurs because of a direct cause and effect relationship with the value of data prediction.

This information was useful in conducting data reduction steps to help us consider what to discard in the data columns that provided less value. By measuring the quality of the data, we could see 13 out of the 33 attributes had high correlation score, and low value ID-ness, stability, missing, and text-ness value. The rest of attributes were discarded since they had more than 70% missing values. Detail of the attributes can be seen in Table 1.

In this stage, all the process of preparing data attributes as the features for the Machine Learning process is done. This clean data now was used for training and testing data.

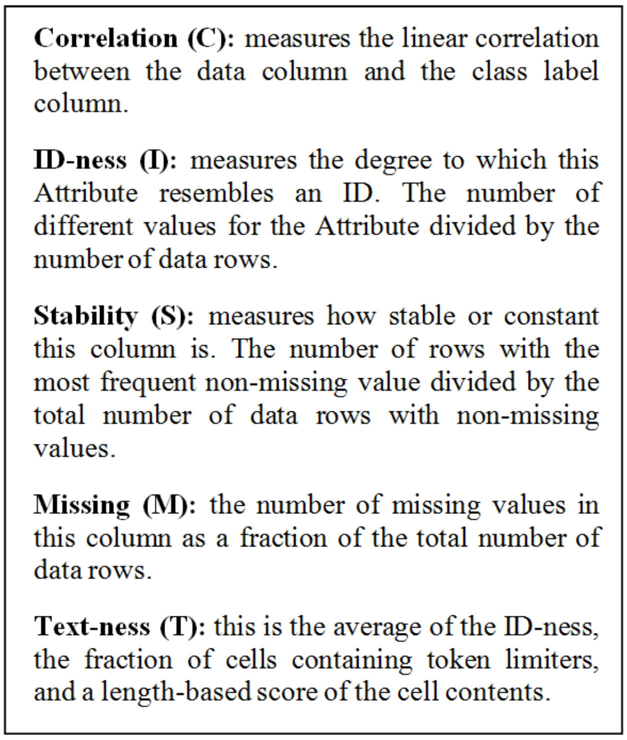
3.2. Building Machine Learning (ML) model.

In building a supervised machine learning model for prediction, we were required to select a classifier that we could use for

Table 1. Recommendation data attributes based on CISMT as reduction processing

	(C)	**(I)**	**(S)**	**(M)**	**(T)**
Age	4.16%	0.17%	2.31%	0.00%	0.00%
SEX	0.09%	0.00%	57.37%	0.00%	2.16%
CHOL	2.19%	0.55%	1.14%	61.74%	0.00%
TRIG	2.97%	1.02%	0.86%	62.41%	2.06%
HDL	1.28%	0.20%	3.45%	62.85%	1.07%
UREA	2.18%	0.54%	5.05%	47.75%	0.00%
CREA	1.32%	?	14.29%	46.46%	0.00%
UA	2.49%	0.29%	2.56%	63.69%	1.52%
SGOT	0.79%	1.03%	4.97%	49.37%	0.00%
SGPT	2.21%	1.03%	3.40%	66.31%	1.94%
X33	1.74%	0.22%	7.82%	68.26%	0.00%
K	0.06%	?	6.90%	68.28%	0.00%
CL	0.61%	0.19%	7.97%	68.26%	0.00%

*‘?’ symbol is missing value



Correlation (C): measures the linear correlation between the data column and the class label column.

ID-ness (I): measures the degree to which this Attribute resembles an ID. The number of different values for the Attribute divided by the number of data rows.

Stability (S): measures how stable or constant this column is. The number of rows with the most frequent non-missing value divided by the total number of data rows with non-missing values.

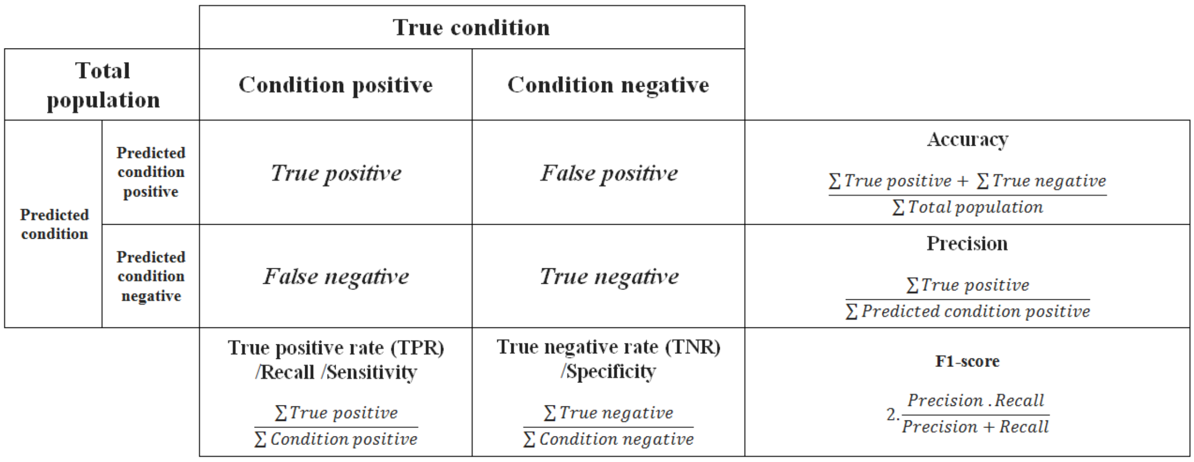
Missing (M): the number of missing values in this column as a fraction of the total number of data rows.

Text-ness (T): this is the average of the ID-ness, the fraction of cells containing token limiters, and a length-based score of the cell contents.

the prediction process. There were many classifier models available for this process, and unfortunately there was no certain way to choose the best classifier model for the data. Hence, in this case, by using RapidMiner application, we ran several trainings and tests for different types of classifiers.

3.2.1. *Applying classifier models.* In order to select a classifier model for building our Machine Learning API, there were five classifier models that were tested for finding the best fit for our scenario and data. These five classifiers were selected and tested based on the previous related work [2,8,16]. These five classifier models were: *Naïve Bayes classifier*; *Logistic Regression classifier*; *Decision Trees classifier*; *Random Forest classifier* and *Gradient Boosted Trees classifier*.

3.2.2. *Evaluation techniques.* In order to choose the best classifier model for our Machine Learning API, there are a number of evaluation techniques that have been applied to assess the performance of prediction result of the class label. The summary of all evaluation techniques can be seen in diagram below (Figure 2). In this work, for the main evaluation techniques, we measured the classification accuracy, classification error, and F-score/F-measurement.



		True condition		
Total population		Condition positive	Condition negative	
Predicted condition	Predicted condition positive	*True positive*	*False positive*	Accuracy $\frac{\sum True\ positive + \sum True\ negative}{\sum Total\ population}$
	Predicted condition negative	*False negative*	*True negative*	Precision $\frac{\sum True\ positive}{\sum Predicted\ condition\ positive}$
		True positive rate (TPR) /Recall /Sensitivity $\frac{\sum True\ positive}{\sum Condition\ positive}$	True negative rate (TNR) /Specificity $\frac{\sum True\ negative}{\sum Condition\ negative}$	F1-score $2.\frac{Precision\ .Recall}{Precision + Recall}$

Figure 2. Summary of the evaluation techniques [22,23]

3.2.3. *Training and testing.* Using the evaluation techniques above, we evaluated performance and prediction results of all the tested classifiers. The real and predicted conditions needed to be categorized into: 1) Risk and Risk as True Positive; 2) Risk and Normal as False Negative; 3) Normal and Risk as False Positive; and 4) Normal and Normal as True Negative.

For the training and testing purposes we split our data into 60% for training and 40% for testing. Based on training purposes, the results of the accuracy evaluation, classification error evaluation and F-measurement evaluation could be seen on the graphs in Table 2.

Table 2. Accuracy evaluation, classification error and F-measure of 5 classifier models

	Naïve Bayes	Logistic Regression	Decision Tree	Random Forest	Gradient Boosted
F-Measure	84.7%	88.6%	86.9%	86.9%	92.5%
Classification Error	20.9%	17.9%	23.2%	23.1%	11.7%
Accuracy	79.1%	82.1%	76.8%	76.9%	88.3%

Based on above results, it clearly shows that the Gradient Boosted Trees classifier has the highest result on accuracy and F-measurement evaluation, followed by the Logistic Regression classifier. Better performance evaluation was also demonstrated by Gradient Boosted Trees classifier with the lowest classification error measurement with only 11.7% followed by the Logistic Regression classifier with 17.9%.

In order for deeper analysis, we also measured the value of Precision, Recall/Sensitivity and Specificity for assessing the performance of prediction results of our data in each model classifier. The results are shown in Table 3.

Precision means the percentage of your results which are relevant. On the other hand, Recall refers to the percentage of total relevant results correctly classified by your algorithm. Highest Precision and Recall evaluation indicated better accuracy of the model. Based on the evaluation matrix test for precision, the Naïve Bayes classifier has the highest number for precision followed by the Gradient Boosted Trees classifier with a value

Table 3. Precision, recall/sensitivity and specificity of 5 classifier models

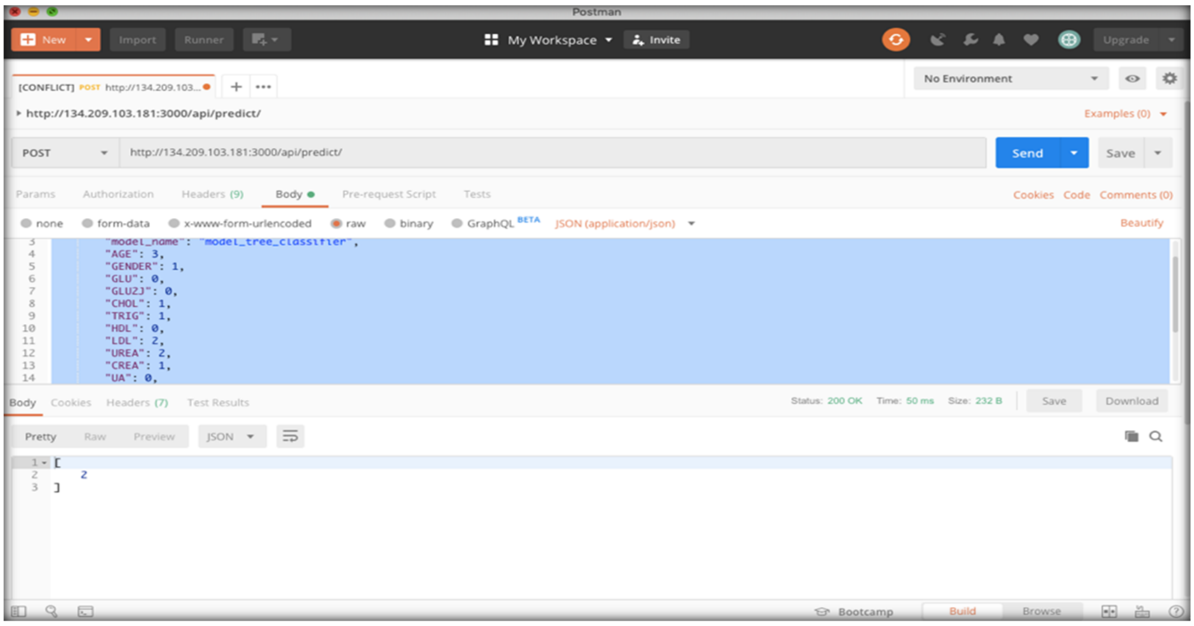
	Naïve Bayes	Logistic Regression	Decision Tree	Random Forest	Gradient Boosted
Precision	97.1%	86.4%	76.8%	76.9%	91.2%
Recall/ Sensitivity	75.1%	91.0%	100.0%	100.0%	93.8%
Specificity	92.4%	52.5%	0.0%	0.0%	70.0%

above 90%. Meanwhile, for Recall evaluation, Decision Trees and Random Forest classifiers held the highest Recall values, followed by Gradient Boosted Trees classifier at 93.8%. For Specificity, the Naïve Bayes classifier had the highest value of 92.4% for Sensitivity followed by the Gradient Boosted Trees classifier with a value of 70%.

3.3. **Creating or applying APIs.** The Gradient Boosted Trees classifier has been selected as the best model classifier for our data model. Here all the designs and parameters of this classifier are used for building a backend application. In building a backend machine learning application, we used Django REST framework and Python library.

As the prerequisite and installation steps, we used Python version 3.6, Django version 2.1 and *DjangoRestFramework* version 3.8.2. Using a droplet by Digital Ocean, we installed all the prerequisites and installations to set the backend server. Then we created a rest application in the root location as the host application for the backend. In applying Gradient Boosted Trees classifier into the backend, we utilized a library from *sckit-learn* version 0.21.2.

For creating the API, URLs and patterns in the backend application needed to be modified. This is an important step to set the API URL so it could be accessed from the outside of the backend application. As the result, the machine learning API could be accessed and tested via Postman application. Figure 3 is an example of API that has been accessed through the Postman application. In this stage, we successfully created the API for training and predicting cardiovascular disease risk.



http://134.209.103.181:3000/api/predict/

POST http://134.209.103.181:3000/api/predict/

```
        "model_name": "model_tree_classifier",
        "AGE": 3,
        "GENDER": 1,
        "GLU": 0,
        "GLU2J": 0,
        "CHOL": 1,
        "TRIG": 1,
        "HDL": 0,
        "LDL": 2,
        "UREA": 2,
        "CREA": 1,
        "UA": 0,
```

Status: 200 OK Time: 50 ms Size: 232 B

```
[
    2
]
```

Figure 3. The testing of the ML API for prediction using the POSTMAN application

4. **Conclusion.**

In summary, this paper's objective was to describe the whole process of building a machine learning API to predict cardiovascular disease risk based on medical records as the input. In conducting this research, we categorized this activity into three stages, which were preparing the data, building the machine learning API based on the best classifiers for the model data training, and creating and simulating the API.

As for the results, based on the training and testing process, Gradient Boosted Trees classifier was chosen as the best model classifier with accuracy above 80%, F-measure above 92% and classification error less than 12%. Further evaluation using precision, recall, sensitivity and specificity measurement also have been conducted, but the result still shows Gradient Boosted Trees classifier can be categorized as second best for the training and testing model.

Then, a backend machine learning application with API was successfully built based on this Gradient Boosted Trees classifier model and it was also tested for training and predicting the cardiovascular disease risk through a web-browsable API and the Postman application.

For further development and work, this machine learning API could be connected and tested using third party applications and devices. More work is also required to improve the accuracy for prediction and make the training process more efficient. Another approach for future researcher could be in terms of predicting another disease.

Acknowledgment. This work has been supported by Directorate General of Higher Education and Bina Nusantara University.

REFERENCES

[1] D. E. Goldberg and J. H. Holland, Genetic algorithms and machine learning, *Mach. Learn.*, vol.3, nos.2-3, pp.95-99, 1988.

[2] J. Tanha, M. van Someren and H. Afsarmanesh, Semi-supervised self-training for decision tree classifiers, *Int. J. Mach. Learn. Cybern.*, vol.8, no.1, pp.355-370, 2017.

[3] H. Blockeel and L. De Raedt, Top-down induction of first-order logical decision trees, *Artif. Intell.*, vol.101, nos.1-2, pp.285-297, 1998.

[4] T. Li, J. Li, Z. Liu, P. Li and C. Jia, Differentially private Naïve Bayes learning over multiple data sources, *Inf. Sci.*, vol.444, pp.89-104, 2018.

[5] Machine Learning Plus, *How Naive Bayes Algorithm Works? (With Example and Full Code) | M-L+*, https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/, Accessed on 25-Jun-2019.

[6] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science*, vol.355, no.6325, pp.602-606, 2017.

[7] Y. Song, J. Liang, J. Lu and X. Zhao, An efficient instance selection algorithm for k nearest neighbor regression, *Neurocomputing*, vol.251, pp.26-34, 2017.

[8] A. P. Reimer, N. K. Schiltz, V. P. Ho, E. A. Madigan and S. M. Koroukian, Applying supervised machine learning to identify which patient characteristics identify the highest rates of mortality post-interhospital transfer, *Biomed. Inform. Insights*, vol.11, pp.1-11, 2019.

[9] B. M. Altura, Deadly communicable diseases and 'superbugs' brought to western societies could exacerbate cardiovascular disease worldwide and why a newly discovered biologic may ameliorate these challenges: A lurking real danger and solution, *EC Pharmacology and Toxicology*, vol.6, no.9, pp.821-828, 2018.

[10] WHO, *World Health Organisation – Noncommunicable Diseases (NCD) Country Profiles*, 2018.

[11] S. Kemp, *Digital in 2017: Global Overview – We are Social*, https://wearesocial.com/special-reports/digital-in-2017-global-overview, Accessed on 22-Jul-2019.

[12] S. Das, A. Dey and N. Roy, *Applications of Artificial Intelligence in Machine Learning: Review and Prospect*, 2015.

[13] K. NG, *Machine Learning Projects for Mobile Applications: Build Android and iOS Applications Using TensorFlow Lite and Core ML*, Packt Publishing Ltd., 2018.

[14] O. Padilla, *Normal Laboratory Values – Resources – MSD Manual Professional Edition*, https://www.msdmanuals.com/professional/resources/normal-laboratory-values/normal-laboratory-values, Accessed on 22-Jul-2019.

[15] RapidMiner, *Lightning Fast Data Science Platform for Teams | RapidMiner©*, https://rapidminer.com/, Accessed on 22-Jul-2019.
[16] C. Ehrentraut, M. Ekholm, H. Tanushi, J. Tiedemann and H. Dalianis, Detecting hospital-acquired infections: A document classification approach using support vector machines and gradient tree boosting, *Health Informatics J.*, vol.24, no.1, pp.24-42, 2018.
[17] S. Narkhede, *Understanding Logistic Regression – Towards Data Science*, https://towardsdatascience.com/understanding-logistic-regression-9b02c2aec102, Accessed on 26-Jun-2019.
[18] T. Ganegedara, *Intuitive Guide to Understanding Decision Trees – Towards Data Science*, https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-decision-trees-adb2165ccab7, Accessed on 26-Jun-2019.
[19] T. Yiu, *Understanding Random Forest – Towards Data Science*, https://towardsdatascience.com/understanding-random-forest-58381e0602d2, Accessed on 26-Jun-2019.
[20] xgboostdeveloper, *Introduction to Boosted Trees – xgboost 0.90 documentation*, https://xgboost.readthedocs.io/en/latest/tutorials/model.html, Accessed on 26-Jun-2019.
[21] C. van Rijsbergen, Retrieval effectiveness, *Progress in Communication Sciences*, 1981.
[22] D. M. W. Powers, Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation, *J. Mach. Learn. Technol.*, vol.2, no.1, pp.37-63, 2011.
[23] L. Derczynski, Complementarity, F-score, and NLP evaluation, *Proc. of International Conference Language Resource Evaluation*, 2016.