Feedforward neural nets with hyperparameter optimization with Pima diabetes data set

Introduction

Diabetes is the top non-communicable disease that makes problems to the world and national public health. Nowadays there are many factors that many people are leading to diabetes, for example, Age, Blood Pressure, BMI, Skin Thickness, Pregnancies, Diabetes Pedigree Function, Insulin, and Glucose. When diagnosing body disorders, it is difficult to understand all the test results from laboratories. This report shows the efficiency of results for data classification by machine learning.

Machine learning has a performance for learning from previous observations. This allows the resulting model to be proactively used. Decisions on new instances were previously unseen. Therefore, the trial presented in this paper is by using machine learning classification performed on a medical health check dataset about diabetes. The mission is to practice learning algorithms with predictable data sets. Diabetes onset Considers four learning algorithms which are using neural network, random forest, Naïve Bayes, and decision tree and using random search and grid search, for improving the model and predicting the outcome of type 2 diabetes.

Background

Diabetes background

The data was obtained from Kaggle open data platform (Akturk, 2020). The diabetes data has 768 rows and 9 columns including Age, Blood Pressure, BMI, Skin Thickness, Pregnancies, Diabetes Pedigree Function, Insulin, Glucose, and outcome.

Related work

Usman Gulumbe et al. (2019) used neural networks to predict diabetes and compared them with other classifiers such as ANN and KNN. Finally, they displayed the ROC curve.

Jivani (2020) used decision tree to predict diabetes in more than 700 patients. However, no data cleaning. Finally, they displayed decision tree by using Graphviz function.

Nayak (2022) used random forest classifier to predict diabetes and clean data by using mean. However, my project used median. Finally, they displayed the ROC curve.

Paudyal (2020) used Naïve Bayes to predict diabetes and showed precision, recall, and f1-score. However, no data cleaning.

Wang et al. (2021) used neural network to predict diabetes with random search and grid search to improve the model. However, they compared with other classifiers such as ANN, Adaboost, XGBoost, LightGBM, and GBDT.

Machine learning background

Neural network

Usman Gulumbe et al. (2019) explained that neural networks are modeled after the human brain's heterogeneous structure. In the human brain, a biological neural network is dedicated to human tasks, such as reading, speaking, problem-solving, and data storage. Artificial neural networks imitate a portion of brain operations.

Decision tree

J. R. Quinlan and R. L. Rivest (1989) presented the technique. Splits a given data set into two or more two. decision tree help isolate datasets and create decision models to predict unknown class labels.

Random forest

Random forest is an assembly of tree-structured classifiers (Breiman, 1999). Random forest reinforces objects from the input array to all trees in the forest. Each unit vector is voted on individually and classifies each tree and separate categories from the forest.

Naïve Bayes

Sunil (2017) explained and used Naive Bayes as a classification technique to calculate the outcome of probability by counting uniformity as well as including values defined in data. Attributes are independent as well as dependent on the variable of each class.

RG Hyperparameter Optimization

Wang et al. (2021) explained RG hyperparameter optimization. R stands for Random search, while G refers to Grid search. Both have a significant impact on prediction accuracy. Accuracy is improved by using the right set of hyperparameters.

Grid Search

Lutins (2017) described a grid search to build a model based on the combination of available parameters. Performs all possible combinations of parameters and saves each model.

Random Search

Bergstra and Bengio (2012) used random search to get the most suitable solution. It generates random points over time and computes values of constraint function and objective function. Each value of the objective function is compared one by one and choose the proper value.

Objective

The objective of this assignment aims to establish the norm. This is useful for categorizing diabetes complication data by creating and developing model to predict much more efficiently by using supervised learning for learning risk of type 2 diabetes including neural network, random forest, Naïve Bayes, and decision tree. Moreover, using random search and grid search to improve the model and then display the visualization such as graphs, and tables. Finally, finding the best model to predict who is risky of diabetes.

Methodology

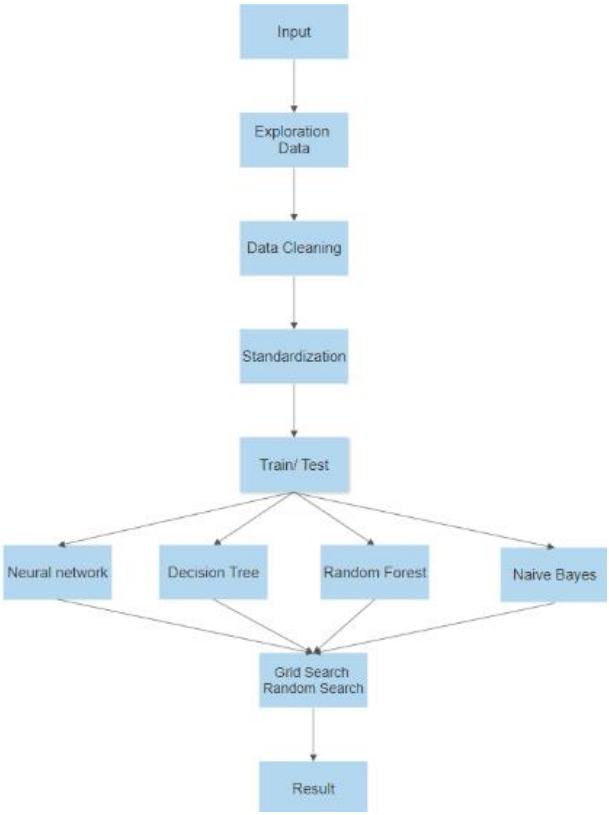


Figure 1

Machine learning is one important technique in data science that can make computer learning by itself by the form of learning is supervised learning. Moreover, Figure 1 shows the steps of this report are as follows:

Step 1 Input, import library read diabetes.csv into Jupyter notebook.

Step 2 Exploration data, understanding rows, and features.

Step 3 Data cleaning, before we train data information there may have some problems and wrong information such as missing values. Therefore, we should modify or remove missing values.

Step 4 Standardization, using numerical features for standardization, which is the best practice to increase accuracy, to scale the data before processing it.

Step 5 Train test split, using supervised learning model and using train data to do hyperparameter tuning. before comparing each model classifier.

Step 6 Developing model, using hyperparameter tuning, for instance, random search and grid search, and apply to increase training accuracy by using training data. Moreover, using validation curve to observe the training graph and time (hyperparameter tuning).

Step 7 Result, now we have many outputs in each model to compare and choose which one is the best result to predict the risk of type 2 diabetes. The result will show validation curves, bar charts, and tables. To choose which classifier is the best train accuracy, time (hyperparameter tuning), test accuracy, precision, recall, and f1-score.

Experiments

Step 1 Read Diabetes.csv

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Figure 2

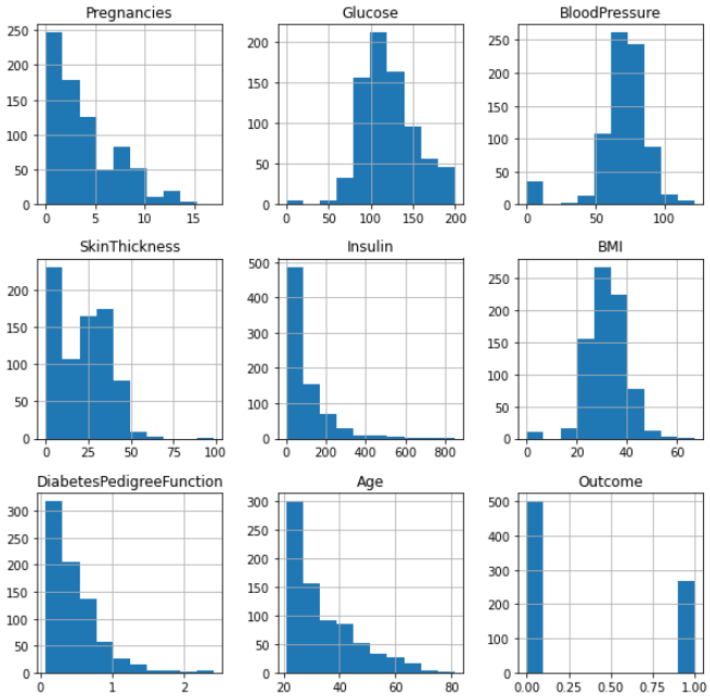


Figure 3

Step 2 Exploration data – Figures 2 and 3 show 768 rows, and 8 features are age, Blood Pressure, BMI, Skin Thickness, Pregnancies, Diabetes Pedigree Function, Insulin, and Glucose. On the other hand, the outcome is 0, and 1 represents 0 is not type 2 diabetes as well as 1 represents type 2 diabetes.

	Outcome	Glucose
0	0	107.0
1	1	140.0

	Outcome	Insulin
0	0	102.5
1	1	169.5

	Outcome	SkinThickness
0	0	27.0
1	1	32.0

	Outcome	BloodPressure
0	0	70.0
1	1	74.5

	Outcome	BMI
0	0	30.1
1	1	34.3

Figure 4

Step 3 Data cleaning – Finding the median of 0 and 1 in each feature such as Glucose, Insulin, Skin Thickness, Blood Pressure, and BMI and replacing them with the missing value in figure 4.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Figure 5

Step 4 Standardization – Using standardization for numerical features in figure 5 to adjust the size of the point of interest in a particular range to increase accuracy.

```
The shape of X_train is      (614, 8)
The shape of X_test is       (154, 8)
The shape of y_train is      (614, 1)
The shape of y_test is       (154, 1)
```

Figure 6: train test split

Step 5 Designing model – Using supervised learning to separate data 80 percent (614 rows, 8 columns) for the training set and 20 percent (154 rows, 8 columns) for the testing set in figure 6.

Step 6 Developing model – Using hyperparameter tuning, for instance, random search and grid search to improve the model.

Hyperparameter tuning

Hashmi (2020) explained the meaning of n_iter and cv. GridSearchCV will try all choices, while RandomSearchCV allows you to define how many you want to test out of all of them by giving the "n_iter" parameter

In this project, grid search used cv = 3 and random search used cv = 3 and n_iter = 20

Grid search

cv = 3 is the model that will be tested (cross-validated) 3 times

Random search

n_iter x cv = 20 x 3 = 60 times

Neural network

Brownlee (2016) explained the definition of hyperparameters in neural networks.

1. Batch size refers to the number of patterns shown to neural network.

2. Epochs is the number of times you can train.

3. Learning rate determines how much weight is distributed and is updated at the end of each group's operation.

4. Momentum determines the extent to which prior update has an impact on current weight.

5. Nonlinearity of each neuron is controlled by activation function.

6. Dropout processes to improve model and solve over-fitting problem by removing some neurons.

7. Neurons play a crucial role in neural networks. The ability to represent network is determined by the number of neurons in a layer.

8. Optimization functions are used to reduce loss and offer the highest level of accuracy.

Hyperparameter	Random search	Parameter
batch_size	10	10, 20, 40, 50, 100
epochs	40	10, 20, 40, 50, 100
optimizer	SGD	RMSprop, Adadelta, Adamax, SGD, Adagrad, Adam, Nadam
activation	hard sigmoid	softplus, relu, sigmoid, hard sigmoid, softmax, softsign, tanh
weight_constraint	2	1 to 5
dropout_rate	0.1	0.0 to 0.8
learn_rate	0.7	0.1 to 0.9
momentum	0.1	0.0 to 0.9
neurons	10	1, 10, 20, 40,60
Time: 2 min 35 seconds		

Table 1: Neural networks with Random search

Hyperparameter	Grid search	Parameter
batch_size	50	50, 100
epochs	20	10, 20
optimizer	Nadam	RMSprop, Adadelta, Adamax, SGD, Adagrad, Adam, Nadam
activation	tanh	softplus, relu, sigmoid, hard sigmoid, softmax, softsign,tanh
weight_constraint	1	1,5
dropout_rate	0.3	0.3, 0.8
learn_rate	0.9	0.1,0.9
momentum	0.2	0.2,0.8
neurons	40	10, 40
Time: 3 hours 17 min 54 seconds		

Table 2: Neural networks with Grid search

Table 1 and table 2 show the different hyperparameters between random search and grid search. When the range of parameters is large and large parameters, the grid search process is slow. Because taking a long time for hyperparameter tuning, I decided to reduce the range of parameters.

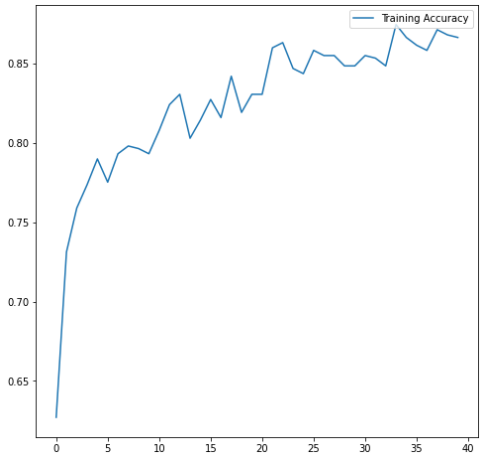


Figure 7: After Random search

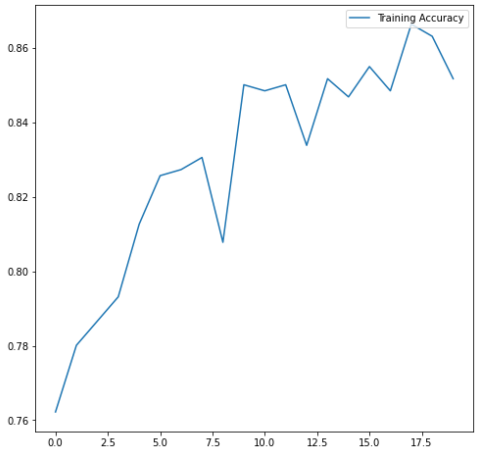


Figure 8: After Grid search

Figure 7 (Random search) is better than figure 8 because of higher train accuracy. Although epochs of random search equal 40 and it's not overfitting. On the other hand, grid search epochs are 20. However, it's underfitting.

Decision tree

Hyperparameter	Random search	Grid search	Parameter
criterion	entropy	entropy	gini,entropy
splitter	best	best	best, random
Max depth	4	9	2 to 14
Max features	log2	sqrt	auto, sqrt, log2
	Time: 295 ms	**Time: 1.54 s**	

Table 3: Decision tree with Random search and Grid search

Gulati (2022) explained the definition of hyperparameters in decision tree.

The technique of determining the split quality of decision trees, such as Gini and entropy, is known as the criterion.

Max depth is the longest route between the node of the root as well as the node of the leaf.

Splitter is a node splitting method.

Max features assess the optimum approach to divide.

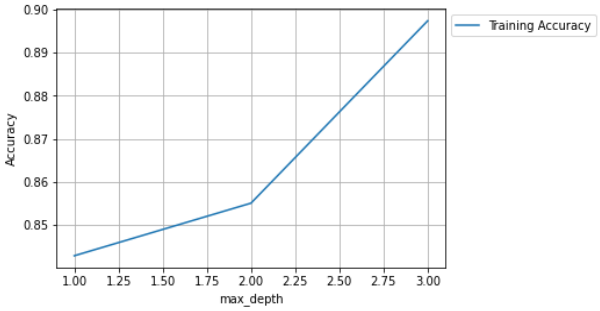


Figure 9: After Random search

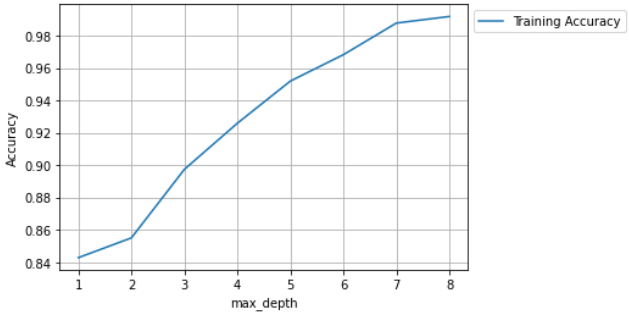


Figure 10: After Grid search

Figure 10 (Grid search) is better than figure 9 (random search) because grid search runs all values. It is not overfitting. On the other hand, random search is underfitting because it stops learning early.

Random Forest

Hyperparameter	Random search	Grid search	Parameter
N_estimators	200	200	200 to 2000
Max_features	auto	auto	auto, sqrt
Max_depth	13	10	0 to 15
	Time: 1 min 16 seconds	**Time: 15 min 11 seconds**	

Table 4: Random Forest with Random search and Grid search

Gulati (2022) explained the definition of hyperparameters in random forest.

The number of trees in forest is n estimators.

Max depth is the longest route between the node of the root as well as the node of the leaf.

Max features assess the optimum approach to divide.

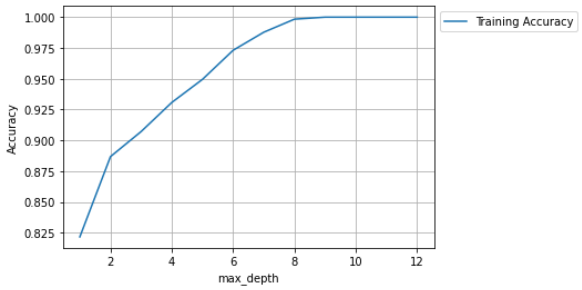


Figure 11: After Random search

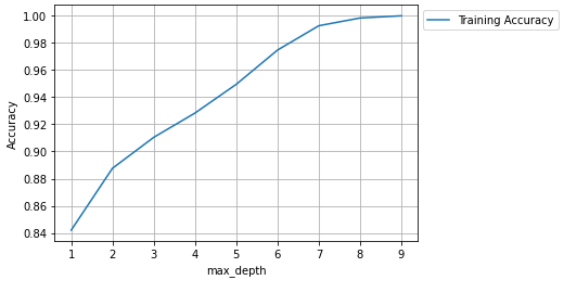


Figure 12: After Grid search

Figure 12 (Grid search) is better than figure 11 (random search) because grid search runs all values. It reaches 100 percent accuracy then stops learning and it's not overfitting. On the other hand, random search is still learning. Even though it's 100 percent already.

Naïve Bayes

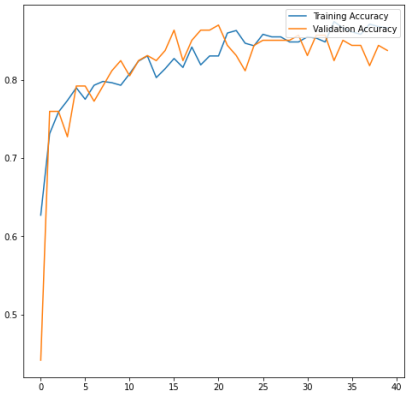
Hyperparameter	Random search	Grid search	Parameter
var_smoothing	0.35	0.35	0, ends at -9, and generates 100 samples.
	Time: 192 ms	**Time: 802 ms**	

Table 5: Naïve Bayes with Random search and Grid search

Jain (2021) explained that var_smoothing is the calculation for improving stability to widen and smooth. Therefore, consider more samples that are far away from mean distribution.

There is no random search and grid search graph for Naïve Bayes var smoothing has a very small value. Moreover, Naïve Bayes with random search and Naïve Bayes with grid search results are the same. However, grid search takes a long time for hyperparameter tuning than random search.

Compared accuracies result



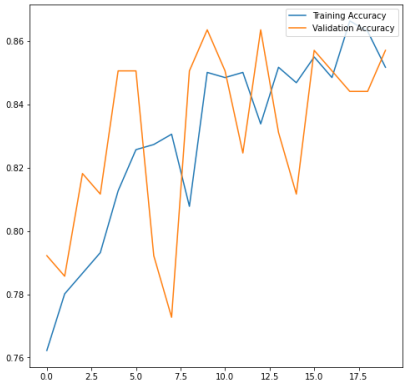
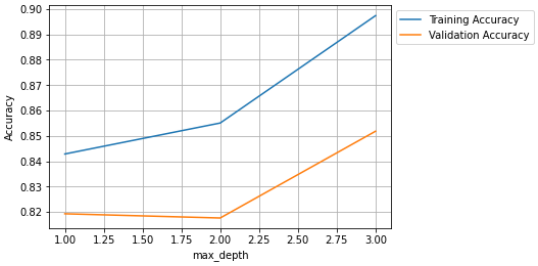


Figure 13: Neural network with Random search **Figure 14: Neural network with Grid search**

The hyperparameter tuning did well with neural network with random search. Due to longer processing, I decided to reduce the range of parameters. It affected random search which gets better accuracies because it randoms more range of parameters than grid search.



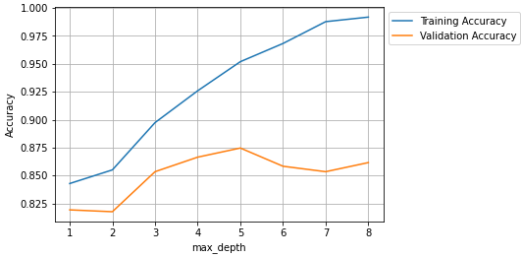
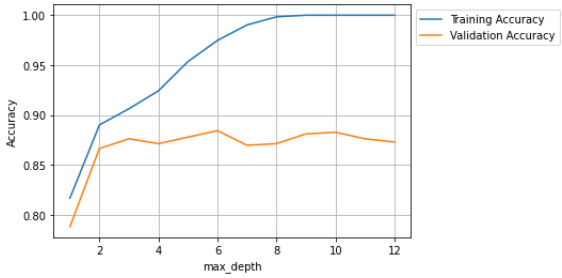


Figure 15: Decision Tree with Random search **Figure 16: Decision Tree with Grid search**

The hyperparameter tuning did well with decision tree with grid search which is better than random search because random search is underfitting.



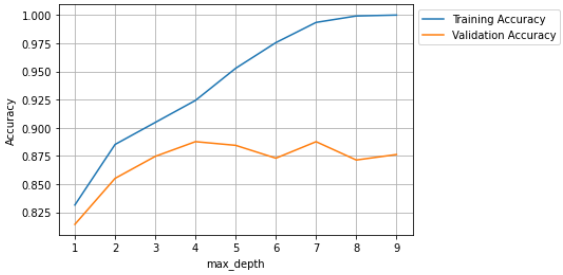


Figure 17: Random Forest with Random search **Figure 18: Random Forest with Grid search**

The hyperparameter tuning did well with random forest with grid search and random search. Because random forest gets the best train accuracies at 100 percent. However, the test accuracy of random forest with grid search is better than random forest with random search.

Naïve Bayes with Random search and Random Forest with Grid search

There is no random search and grid search graph for Naïve Bayes because var smoothing has a small value.

Results

Result of hyperparameter tuning

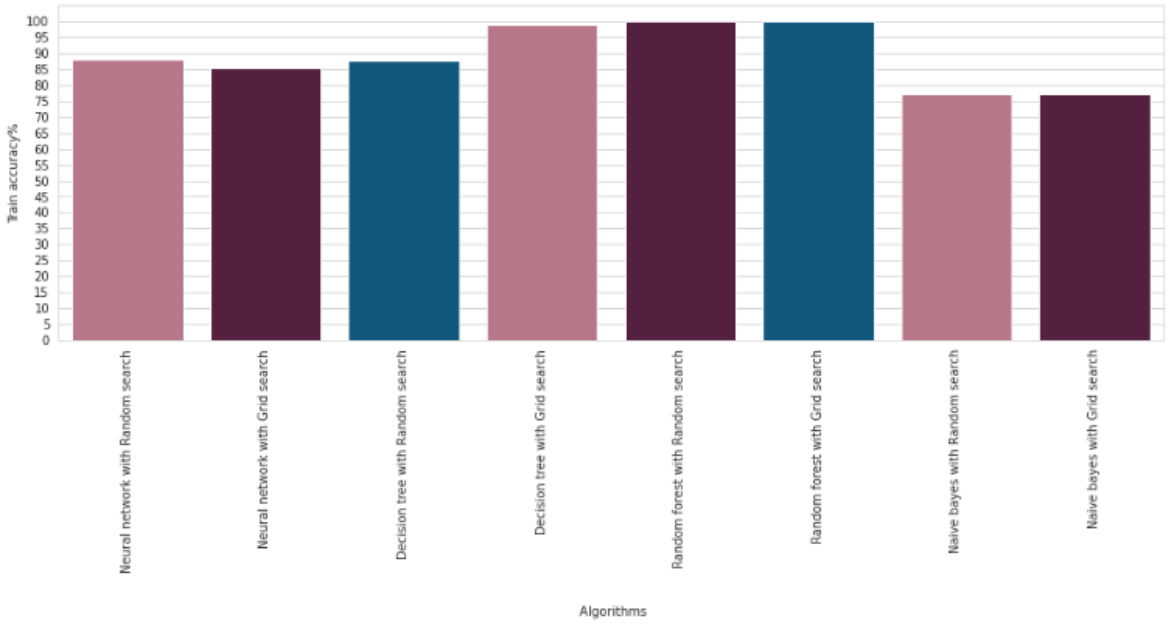


Figure 19

	Random Search				Grid Search			
Classifier	Random forest	Decision tree	Neural network	Naïve Bayes	Random forest	Decision tree	Neural network	Naïve Bayes
Train Accuracy	100.0	87.62	87.94	77.19	100.0	98.69	85.17	77.19
Time (hyperparameter tuning)	1 min16 s	295 ms	2 min35 sec	192 ms	15 min 11 s	1.54 s	3 h 17 min	802 ms

Table 6

Figure 19 and table 6 show the results of training accuracies, and the best training accuracy is random forest with random search and grid search followed by decision tree, neural network, and Naïve Bayes respectively.

Decision tree and random forest work well in grid search because there are small parameters and hyperparameters. Therefore, it can use the same range of parameters.

On the one hand, the result of random search work well with neural networks because I reduced the range of parameter for grid search due to large parameter and a long time to train. It affected training accuracy of grid search which is worse than random search. However, it still spends a long time, 3 hours 17 minutes.

On the other hand, Naive Bayes gets the same results from both random search and grid search. However, both results aren't good.

The best test accuracies result

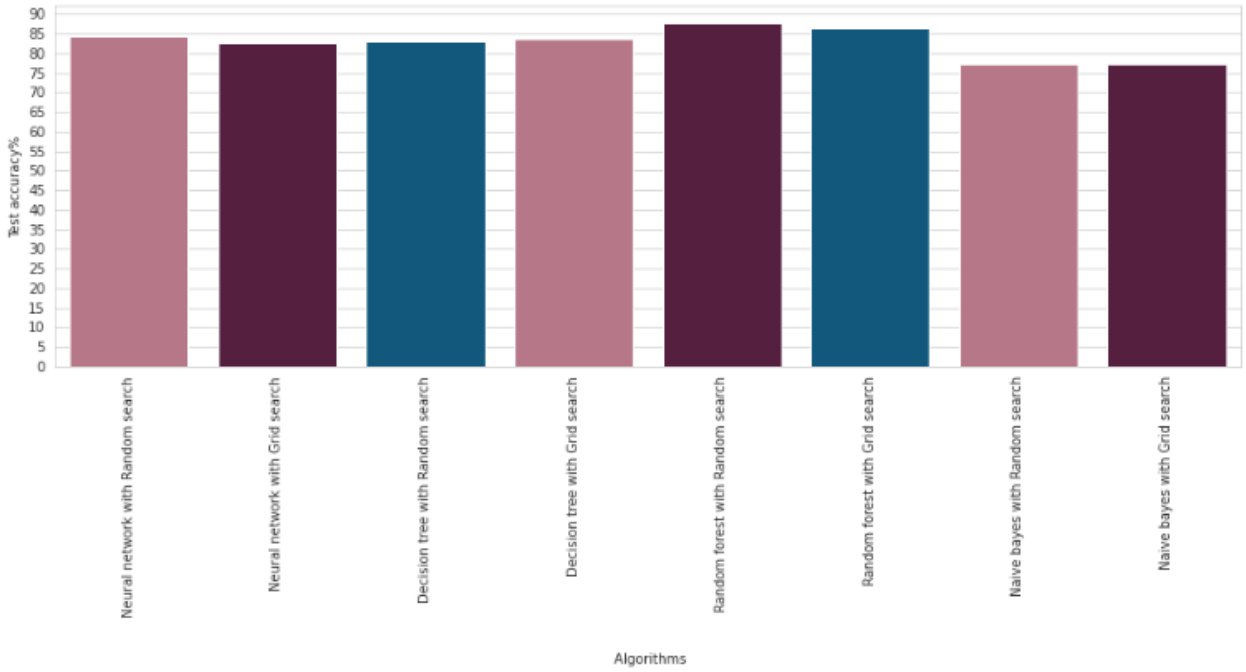


Figure 20

	Random Search				Grid Search			
Classifier	Random forest	Decision Tree	Neural network	Naïve Bayes	Random forest	Decision Tree	Neural network	Naïve Bayes
Test Accuracy	87.66	83.11	84.41	77.27	86.36	83.76	82.46	77.27
Precision	81.03	77.35	76.27	67.85	77.41	76.78	83.33	67.85
Recall	85.45	74.54	81.81	69.09	87.27	78.18	63.63	69.09
F1-score	83.18	75.92	78.94	68.46	82.05	77.47	72.16	68.46

Table 7

Table 7 shows the best classifier is random forest with random search, because the best test accuracy at 87.66 percent, precision, recall, and f1- score are also best at 81.03, 85.45, and 83.18 percent respectively, followed by random forest with grid search, neural network with random search, decision tree with grid search, decision tree with random search, neural network with grid search, and Naïve Bayes.

Critical thinking with random search and grid search

When the hyperparameter range and parameters are large, and grid search process runs very slowly. Senapati (2018) implied that random search is better training data than grid search depending on the dataset. This is because the model can be trained with the best possible parameters without aliasing as well as it is more likely to find the best parameters. Moreover, random searches are useful for low-dimensional data because they take less time and fewer iterations to find the best results. Therefore, small-dimensional random search is the best parameter search method.

Conclusion

This research shows the best result to predict who is risky of diabetes by using machine learning and the best classifier is random forest with random search by comparing with other classifiers such as neural networks, random forest, Naïve Bayes, and decision tree with RG hyperparameter optimization. When using many parameters such as neural network with random search it's faster than grid search. CPU doesn't have to run for a long time and the accuracies are not too much different. On the one hand, grid search is the better result but takes more time. Finally, machine learning has a significant impact to train and predict outcomes (diabetes) and RG hyperparameter optimization could improve the model.

Future work

First, using Bayesian optimization, to compare more results of hyperparameter tuning. Second, adding more parameter range of neural networks with grid search, we should use a better computer because my computer takes a long time for processing.

References

Brownlee, J. (2016) *How to grid search hyperparameters for deep learning models in Python with Keras*, *Machine Learning Mastery*.

Available at: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

(Accessed: March 18, 2022).

Wang, Y. *et al.* (2021) "RG hyperparameter optimization approach for improved indirect prediction of blood glucose levels by boosting ensemble learning," *Electronics*, 10(15), p. 1797.

doi: 10.3390/electronics10151797.

(Accessed: March 18, 2022).

Bergstra, J. and Bengio, Y. (2012) *Random search for hyper-parameter optimization*, *Jmlr.org*.

Available at: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

(Accessed: March 19, 2022).

Usman Gulumbe, S. *et al.* (2019) "Predicting diabetes mellitus using artificial neural network through a simulation study," *Machine Learning Research*, 4(2), p. 33.

doi: 10.11648/j.mlr.20190402.12.

(Accessed: March 19, 2022).

J. R. Quinlan and R. L. Rivest (1989) *Inferring decision trees using the minimum description length, Slideplayer.com*.

Available at: https://slideplayer.com/slide/7753078/

(Accessed: March 19, 2022).

Breiman, L. (1999) *Random forests--random features*, *Berkeley.edu*.

Available at: https://www.stat.berkeley.edu/~breiman/random-forests.pdf

(Accessed: March 19, 2022).

sunil (2017) *Learn Naive Bayes Algorithm*, *Analytics Vidhya*.

Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

(Accessed: March 19, 2022).

Jain, K. (2021) *How to improve naive Bayes?*, *Analytics Vidhya*.

Available at: https://medium.com/analytics-vidhya/how-to-improve-naive-bayes-9fa698e14cba

(Accessed: March 20, 2022).

Senapati, D. (2018) *Grid Search vs Random Search - Deepak Senapati*, *Medium*.
Available at: https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318
(Accessed: April 14, 2022).

Hashmi, F. (2020) *How to tune hyperparameters using Random Search CV in python*, *Thinking Neuron*.
Available at: https://thinkingneuron.com/how-to-tune-hyperparameters-using-random-search-cv-in-python/
(Accessed: April 14, 2022).

Nayak, L. (2022) *Predicting diabetes with random forest classifier, Towards Data Science.*

Available at: https://towardsdatascience.com/predicting-diabetes-with-random-forest-classifier-c62f2e319c6e

(Accessed: April 14, 2022).

Jivani, U. (2020) *Decision Tree on Diabetic patients dataset - Analytics Vidhya - Medium*, *Analytics Vidhya*.
Available at: https://medium.com/analytics-vidhya/decision-tree-on-diabetic-patients-dataset-9529c1265ef4
(Accessed: April 14, 2022).

Paudyal, P. (2020) *Classification of Diabetes using Naive Bayes in Python*, *Medium*.
Available at: https://medium.com/@pragya_paudyal/classification-of-diabetes-using-naive-bayes-in-python-44385b279277
(Accessed: April 27, 2022).

Akturk, M. (2020) *"Diabetes Dataset.", Kaggle.*

https://www.kaggle.com/datasets/mathchi/diabetes-data-set?select=diabetes.csv

(Accessed: April 27, 2022).

Lutins, E. (2017) *Grid searching in machine learning: Quick explanation and python implementation, Medium.*

Available at: https://elutins.medium.com/grid-searching-in-machine-learning-quick-explanation-and-python-implementation-550552200596

(Accessed: April 28, 2022).

Gulati H. (2022) *Hyperparameter tuning in decision trees and Random forests, Engineering Education (EngEd) Program | Section.*

Available at: https://www.section.io/engineering-education/hyperparmeter-tuning/

(Accessed: May 1, 2022).