JOURNAL OF DATA SCIENCE | Vol.2024:22
eISSN:2805-5160

Enhancing Classification Algorithms with Metaheuristic Technique

Cokro Nurwinto[1], Tri Basuki Kurniawan[1*], Misinem[2], Tata Sutabri[1], Yesi Novaria Kunang[1]

[1]Magister of Information Technology, Universitas Bina Darma, Palembang, Indonesia
[2]Faculty of Vocational, Universitas Bina Darma, Palembang, Indonesia

[*]**Email:** tribasukikurniawan@binadarma.ac.id

Abstract

Classification is a process of grouping or placing data into appropriate categories or classes based on specific attributes or features to predict labels or classes of new data based on patterns observed from previously trained data. Implementing this process uses classification algorithms such as Naïve Bayes, Support Vector Machine, and Random Forest. However, the classification algorithm cannot classify data optimally due to the challenges in dealing with various data sets. Not all available features will make a solid contribution to the label of the data class, often in the form of noise or interference. For this reason, it is necessary to carry out a feature selection process. Currently, many feature selection processes have been carried out using correlation values from chi-square and gain-information, but the accuracy of the results is often still not good enough. This is because the chi-square and gain-information values are fixed. So, the selection of features is minimal and is not based on the previous learning process or what is known as heuristics. For this reason, in this research, several auxiliary algorithms are introduced to improve the performance of the classification algorithm, namely the meta-heuristic algorithm. Meta-heuristic algorithms are search techniques used to solve complex optimization problems, and these algorithms can help provide reasonable solutions in a shorter time than exact methods. In its operation, the metaheuristic algorithm optimizes the feature selection process, which will later be processed using the classification algorithm. Three (3) meta-heuristics were implemented, namely Genetic Algorithm, Particle Swarm Optimization, and Cuckoo Search Algorithm; the experiment was conducted, and the results were collected and analyzed. The result shows that combining Naive Bayes and Genetic Algorithm gives the best performance regarding higher accuracy improvement at +23.77%.

Keywords

Classification, Metaheuristics, Machine Learning, Feature Section

Introduction

Machine Learning has become integral to various fields, from pattern recognition to data-based predictions. Machine learning algorithms classify data into specific categories or classes (Mukhlis et al., 2024). Although various classification algorithms have been implemented successfully, improving their performance to handle diverse data sets is the main challenge.

Submission: 20 June 2024; **Acceptance:** 25 July 2024

Not all available features will make a solid contribution to the label of the data class, often in the form of noise or interference. For this reason, it is necessary to carry out a feature selection process. Currently, many feature selection processes have been carried out using correlation values from chi-square and gain-information, but the accuracy of the results is often still not good enough. This is because the chi-square and gain-information values are fixed. So, the selection of features is minimal and is not based on the previous learning process or what is known as heuristics. For this reason, in this research, several auxiliary algorithms are introduced to improve the performance of the classification algorithm, namely the meta-heuristic algorithm.

Meta-heuristics is an approach to finding close to optimal solutions in a complex and large search space (Setiawan & Ginting, 2014). Various meta-heuristic algorithms, such as Swarm Optimization (PSO), Genetic Algorithm (GA), and Cuckoo Search (CS), have been proven effective in optimizing various problems, including classification algorithm optimization problems.

Although the process of selecting appropriate features using a meta-heuristic algorithm has not been carried out as much by other research compared to using chi-square and gain-information, as discussed in research conducted by (Wang et al., 2016), It can be seen that research movements in this direction have begun to be carried out by several researchers, such as those carried out by Himawan et al. (2023) and Al-Qaness (2020). This is because the optimization process using the meta-heuristic method requires more extended resources and processing time. After all, the optimal feature search process is much more flexible.

Several other researchers, such as Afshar & Usefi (2022) and Gangadhara Moorthy & Pravin (2021), have also carried out research in the same field by improving the capabilities of the sparse least square (SLS) method and global analysis of sensitivity. Their research showed promising results, namely higher accuracy values. For this reason, this research will propose using a meta-heuristic approach to improve performance.

This research aims to explore the study of classification algorithm optimization using a meta-heuristic approach, with a focus on three (3) classification algorithms: Naïve Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF) (Guia et al., 2019). It will also explore using three (3) meta-heuristic algorithms to select the best features, such as PSO, GA, and CS (Akbari and Henteh, 2019).

Three classification techniques are selected depending on whether they offer different and complementary methods to machine learning. Every method has unique qualities that fit various kinds of data and issue settings (Alnuaimi and Albadawi, 2024). Because of their different and complementary optimization mechanisms, which provide insightful analysis of the efficacy of many techniques in choosing the most pertinent features for machine learning models, the researchers also compare PSO, GA, and CS for feature selection. Using a comparison between those algorithms, researchers may utilize each algorithm's extraordinary benefits, enhancing the performance of machine learning models and knowledge of feature selection methods.

By comparing various methods, researchers can grasp the performance of several datasets. Every method has advantages and disadvantages that show themselves differently based on the dataset and current challenges (Taye, 2023). By comparing different techniques, researchers can evaluate the performance of fresh or less often utilized algorithms. Depending on the particular goals of the work (e.g., speed, accuracy, interpretability), researchers can choose the most suitable algorithm for their purposes, employing the best one.

Methodology

The rapid advancements in machine learning have led to the development of various algorithms capable of solving complex classification problems across different domains. However, the performance of these classification algorithms significantly depends on the quality and relevance of the features used for training the models. Feature selection becomes a critical step in the machine learning pipeline to enhance model performance, reduce overfitting, and decrease computational complexity.

Metaheuristic algorithms, known for their flexibility and efficiency in solving optimization problems, have gained popularity in feature selection. Algorithms such as PSO, GA, and CS offer promising solutions for identifying optimal subsets of features that improve the performance of classification models. This research will carry out processes according to the research design, as shown in Figure 1 below.

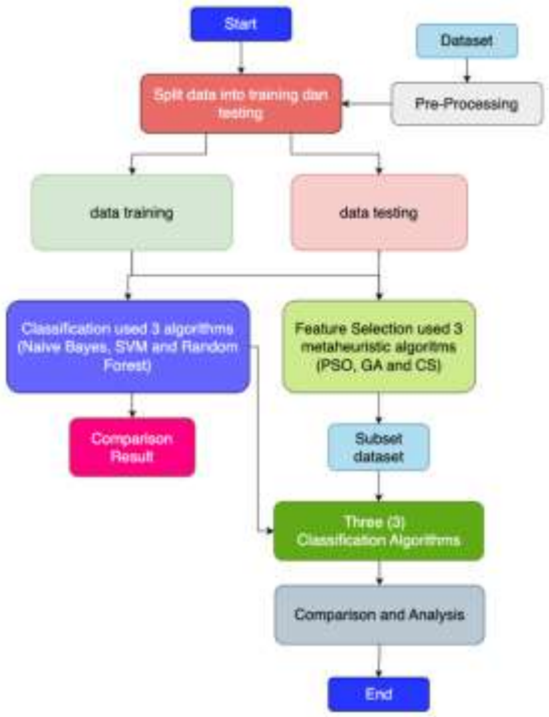
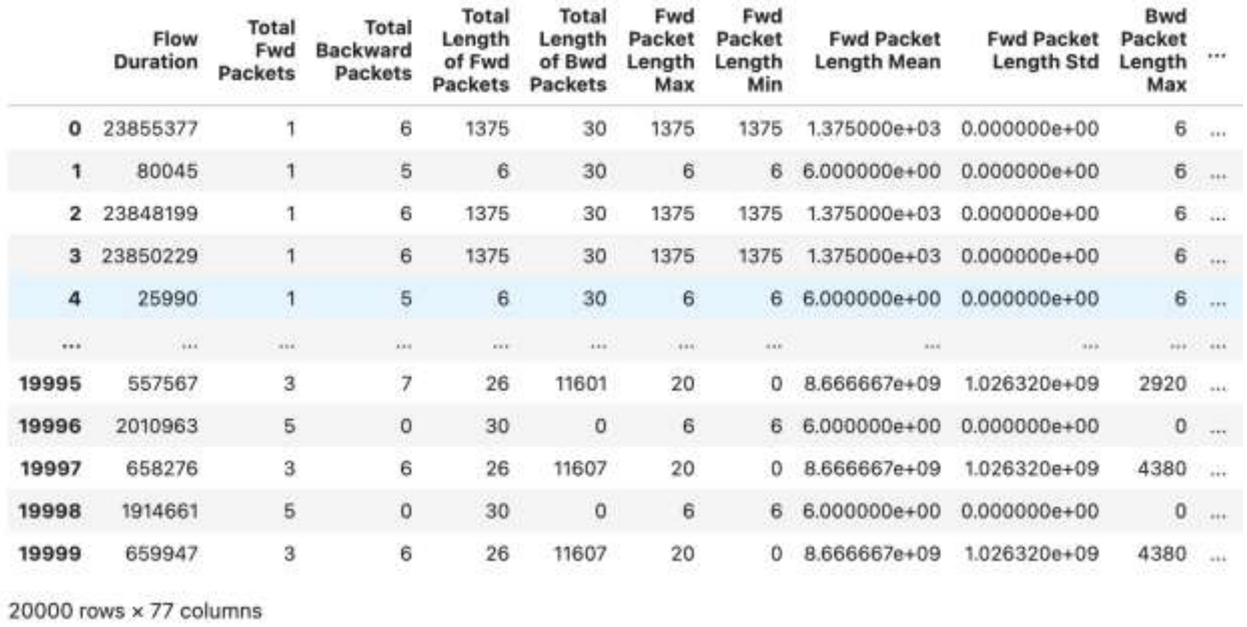

Start
Dataset
Pre-Processing
Split data into training dan testing
data training
data testing
Classification used 3 algorithms (Naive Bayes, SVM and Random Forest)
Feature Selection used 3 metaheuristic algorithms (PSO, GA and CS)
Comparison Result
Subset dataset
Three (3) Classification Algorithms
Comparison and Analysis
End

Figure 1. Research methodology framework

Based on that figure, the research design begins by preprocessing the dataset. We first classify using three (3) classification algorithms. Next, different subset datasets will be generated using three (3) meta-heuristic algorithms. For each subset dataset, a training and testing process will be carried out using three (3) different classification algorithms. For each accuracy, results will be collected and compared. Finally, an analysis will be carried out to compare the accuracy of each subset's results. The three (3) meta-heuristic algorithms for feature selection are implemented using the Py_FS module from the PyPI Python library Guha Ritem et al., (2022).

This research uses data with the extension .xlsx. Data obtained from the Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB). The data used in this research is secondary data in the form of network traffic data containing DDoS data. The data was obtained from the Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB). The data used was selected, and the pre-processing of about 20,000 rows was divided into data training and testing. The total columns were 77, including the Label column, as shown in Figure 2.



	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	...
0	23855377	1	6	1375	30	1375	1375	1.375000e+03	0.000000e+00	6	...
1	80045	1	5	6	30	6	6	6.000000e+00	0.000000e+00	6	...
2	23848199	1	6	1375	30	1375	1375	1.375000e+03	0.000000e+00	6	...
3	23850229	1	6	1375	30	1375	1375	1.375000e+03	0.000000e+00	6	...
4	25990	1	5	6	30	6	6	6.000000e+00	0.000000e+00	6	...
...	...	...	...	...	...	...	...	...	...	...	...
19995	557567	3	7	26	11601	20	0	8.666667e+09	1.026320e+09	2920	...
19996	2010963	5	0	30	0	6	6	6.000000e+00	0.000000e+00	0	...
19997	658276	3	6	26	11607	20	0	8.666667e+09	1.026320e+09	4380	...
19998	1914661	5	0	30	0	6	6	6.000000e+00	0.000000e+00	0	...
19999	659947	3	6	26	11607	20	0	8.666667e+09	1.026320e+09	4380	...

20000 rows × 77 columns

Figure 2. The description of the used dataset

Figure 2 shows 77 columns and 20,000 rows of data. From the data, the pre-processing already conducted, such as converting some values into numerical data and removing some columns, such as the 'Flow ID' and 'Timestamp' columns, because we cannot use them in the classification process, since it does not contain any predictive information about the target variable and Its primary role is to uniquely identify records rather than to provide insights into the relationships between features and the target (Encord Blog, 2023).

Results and Discussion

Some procedures should be done before we do the classification. These procedures include investigating the box plot for each feature in the dataset. By conducting a box plot analysis before classification, you can ensure that the data is well-understood, adequately prepared, and free from issues that could negatively impact the performance of your classification model, like identifying outliers, understanding data distribution, preparing for feature selection, etc. Figure 3 shows a box plot of some features.

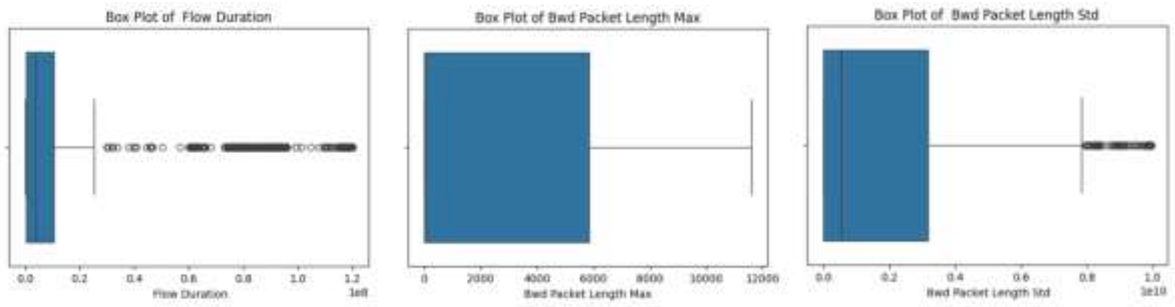
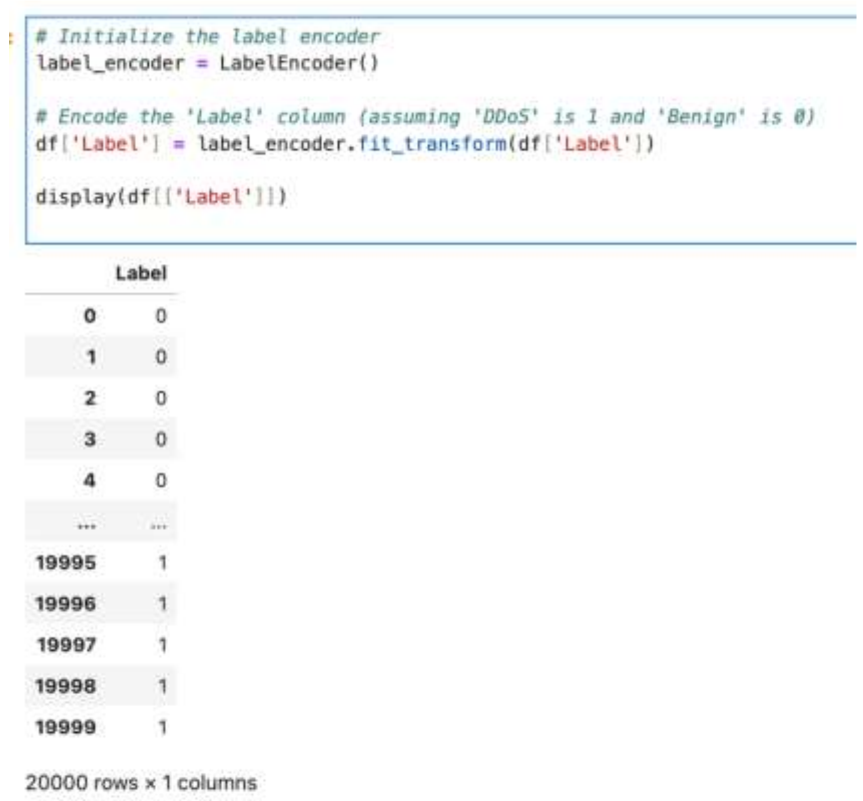


Figure 3. Box-plot of some of the feature

The next step is converting some features, such as the 'Label' columns, to numeric data types. As shown in Figure 4, we will use the ‘LabelEncode’ module from the Sklearn library of Python.


```
# Initialize the label encoder
label_encoder = LabelEncoder()

# Encode the 'Label' column (assuming 'DDoS' is 1 and 'Benign' is 0)
df['Label'] = label_encoder.fit_transform(df['Label'])

display(df[['Label']])
```

```
       Label
0      0
1      0
2      0
3      0
4      0
...    ...
19995  1
19996  1
19997  1
19998  1
19999  1

20000 rows × 1 columns
```
Figure 4. LabelEncode module processing

Figure 4 shows the 'Label' columns already converted into numerical values. Converting data into numerical format is crucial for classification algorithms due to the mathematical nature of the computations involved, the need for distance and similarity metrics, and the requirements of machine learning libraries. This preprocessing step ensures that the algorithms learn from the data and make accurate predictions.

The last step is to perform a correlation analysis. Correlation analysis helps understand the data structure, improve feature selection, and enhance the classification model's robustness and performance, as shown in Figure 5.

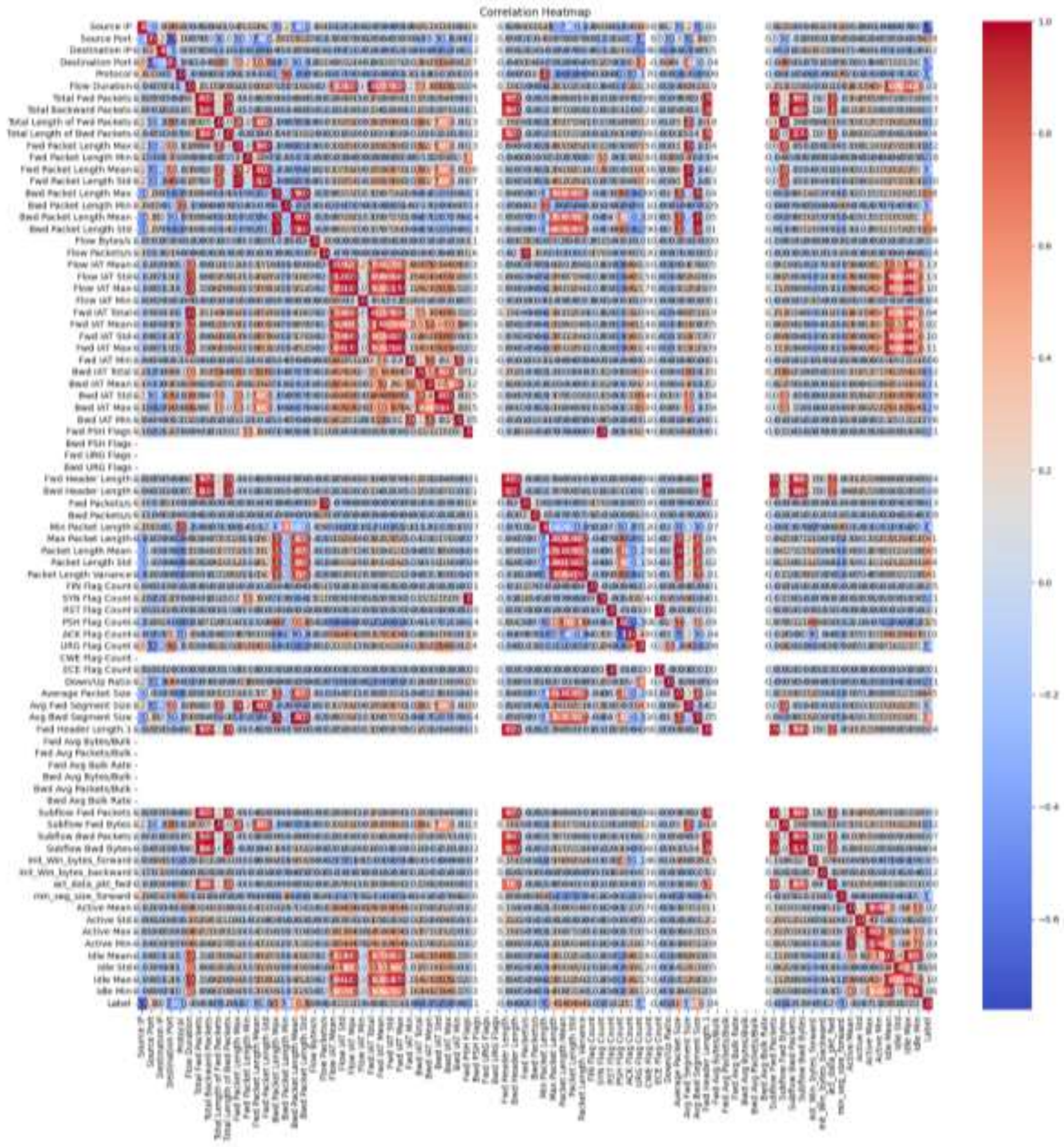
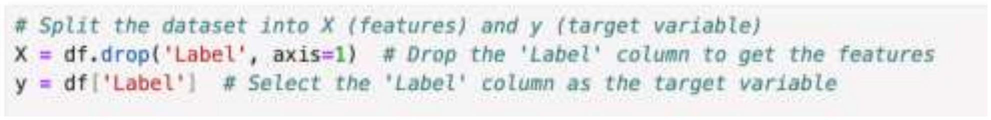


Figure 5. Correlation between all 83 features in our dataset.

Figure 5 shows that not all features strongly correlate with other features, with a value nearest to 1 (or red color). Some features only have low correlation (indicated by blue color). Some other features do not correlate (the figure shows an empty value). A strong correlation means that a feature contributes enormously to our label, and the feature should be selected during the feature selection process.

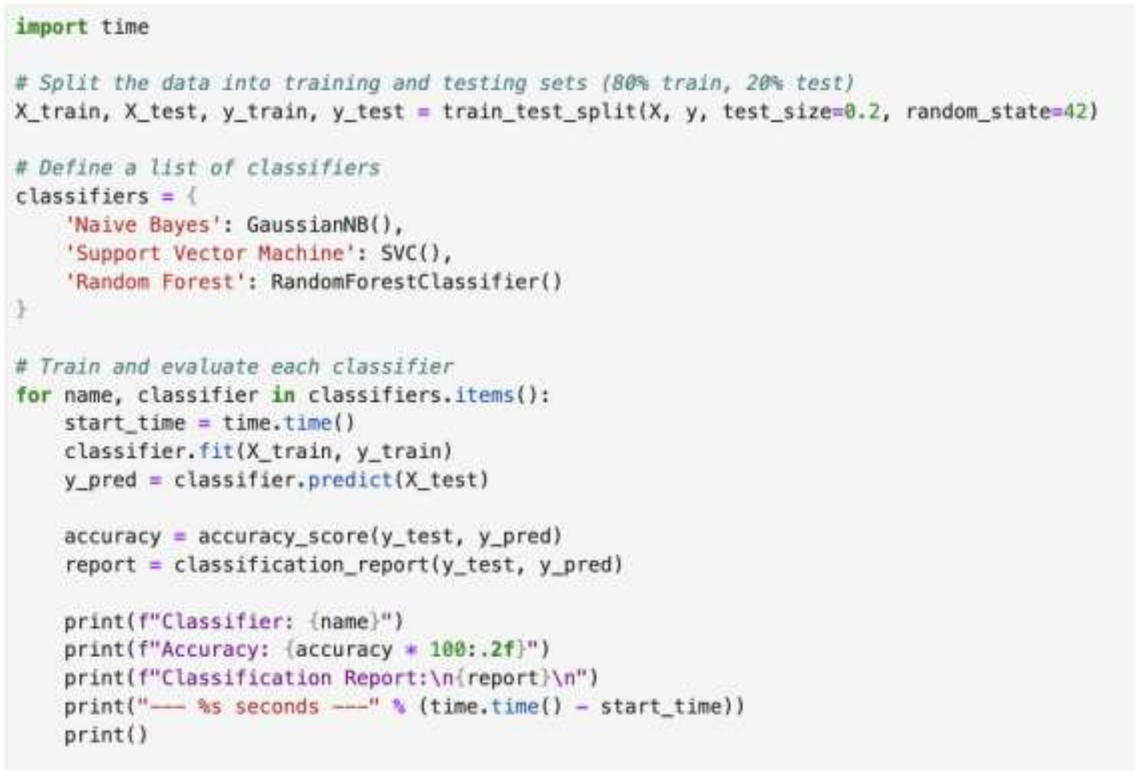
Okay, let us start the classification. First, we need to tell the algorithms which features are independent features (X) and which are dependent features (y), as shown in Figure 6.



```
# Split the dataset into X (features) and y (target variable)
X = df.drop('Label', axis=1)  # Drop the 'Label' column to get the features
y = df['Label']  # Select the 'Label' column as the target variable
```

Figure 6. X and y data from the dataset

Figure 6 shows the Python statement to select independent and dependent variables for classification processing. Next, we split the dataset and do three (3) classification algorithms: Naïve Bayes, Support Vector Machine, and Random Forest, as shown in Figure 7.

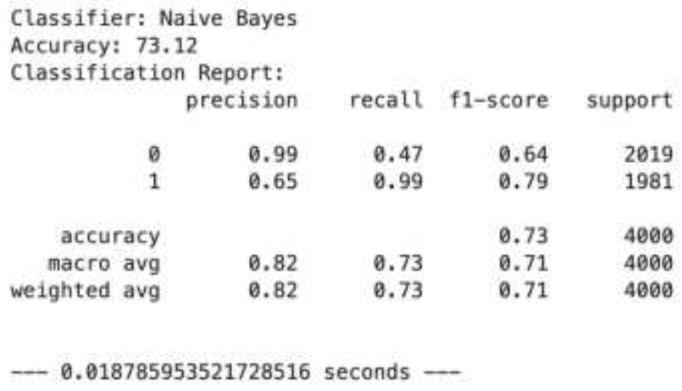
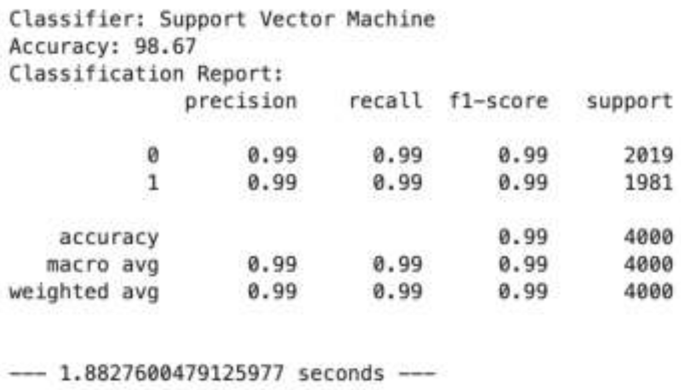
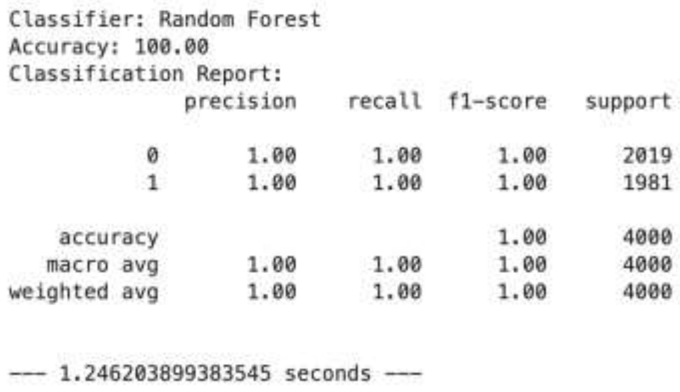


```
import time

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define a list of classifiers
classifiers = {
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(),
    'Random Forest': RandomForestClassifier()
}

# Train and evaluate each classifier
for name, classifier in classifiers.items():
    start_time = time.time()
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    print(f"Classifier: {name}")
    print(f"Accuracy: {accuracy * 100:.2f}")
    print(f"Classification Report:\n{report}\n")
    print("--- %s seconds ---" % (time.time() - start_time))
    print()
```

Figure 7. The classification algorithms code

Based on the coding shown in Figure 7, the dataset will be split into 80% and 20% for training and testing data, respectively. Next, for each classifier, take the start time, and then, before the process finishes, calculate and show the processing time. The training and testing are processed, and the accuracy is taken. Table 1 shows the results and the comparison results.

Table 1. The results and the comparison results

Classifier	Accuracy (%) Execution Time (s)	Classification Report
Naive Bayes (NB)	73.12 0.019	
Support Vector Machine (SVM)	98.68 1.883	
Random Forest (RF)	100.00 1.246	

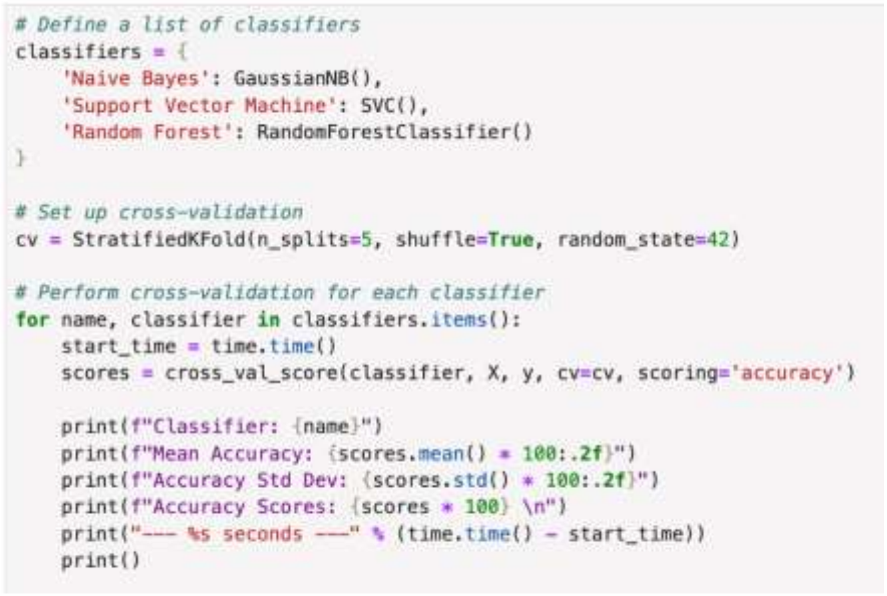
```
Classifier: Naive Bayes
Accuracy: 73.12
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.47      0.64      2019
           1       0.65      0.99      0.79      1981

    accuracy                           0.73      4000
   macro avg       0.82      0.73      0.71      4000
weighted avg       0.82      0.73      0.71      4000


--- 0.018785953521728516 seconds ---
```

```
Classifier: Support Vector Machine
Accuracy: 98.67
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2019
           1       0.99      0.99      0.99      1981

    accuracy                           0.99      4000
   macro avg       0.99      0.99      0.99      4000
weighted avg       0.99      0.99      0.99      4000


--- 1.8827600479125977 seconds ---
```

```
Classifier: Random Forest
Accuracy: 100.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2019
           1       1.00      1.00      1.00      1981

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000


--- 1.246203899383545 seconds ---
```

Table 1 shows that the RF obtained the best accuracy, 100.00%, and the worst NB, 73.12%, and the SVM, at the middle position, at 98.68%.

The worst computational time obtained by the SVM was 1.883 seconds, followed by RF at 1.246% NB at the first position at 0.019 seconds. That means the NB gives the shortest computational time, not even 0.02 seconds; the process is already finished. The RF needs 1.246 seconds and the longest computational time by SVM, almost 1.5 times more than RF and almost 99 times compared to NB. It may be given the same, where we only need to run the algorithms once. Nevertheless, when we need validation of the testing using Cross-Validation, the testing must run 5 (five) times, and the difference will show significantly. Cross-validation is necessary to ensure the results are reliable and generalizable, providing a more accurate assessment of its real-world performance, as shown in Figure 8 and Table 2.



```
# Define a list of classifiers
classifiers = {
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(),
    'Random Forest': RandomForestClassifier()
}

# Set up cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform cross-validation for each classifier
for name, classifier in classifiers.items():
    start_time = time.time()
    scores = cross_val_score(classifier, X, y, cv=cv, scoring='accuracy')

    print(f"Classifier: {name}")
    print(f"Mean Accuracy: {scores.mean() * 100:.2f}")
    print(f"Accuracy Std Dev: {scores.std() * 100:.2f}")
    print(f"Accuracy Scores: {scores * 100} \n")
    print("--- %s seconds ---" % (time.time() - start_time))
    print()
```

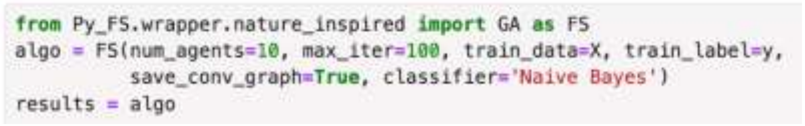
Figure 8. The cross-validation code

Table 2. The cross-validation results and the comparison results

Classifier	Execution Time (s)	Mean Accuracy (%)	Std Dev of Accuracy	Accuracy Scores
NB	0.072	73.11	0.61	[72.95 72.35 73.15 72.875 74.2]
SVM	9.300	98.50	0.14	[98.275 98.425 98.6 98.55 98.65]
RF	5.504	99.95	0.05	[99.98 99.98 99.85 99.95 99.98]

Table 2 shows the consistency pattern with the results in Table 1. The RF still obtained the best mean accuracy at 99.95%, followed by SVM at 98.50% and NB at 73.11%, respectively. The RF also gives a lower standard deviation value at 0.05, which means the RF gives more stable results every time the code is running. The NB gives the most unstable results, indicated by a higher standard deviation of 0.61. Regarding the execution or computational time, the NB still gives the smallest one, and the SVM gives the higher value.

After we get results based on three (3) classification algorithms, we will explore the performance of three (3) meta-heuristic algorithms combined with three (3) classification algorithms in the feature selection process, as shown in Figure 9 and Table 3.



```
from Py_FS.wrapper.nature_inspired import GA as FS
algo = FS(num_agents=10, max_iter=100, train_data=X, train_label=y,
          save_conv_graph=True, classifier='Naive Bayes')
results = algo
```

Figure 9. The feature selection is based on GA and uses the NB classifier.

Table 3. The exploration and the comparison results

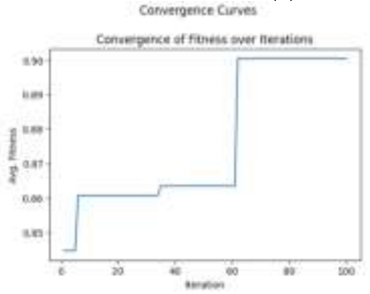
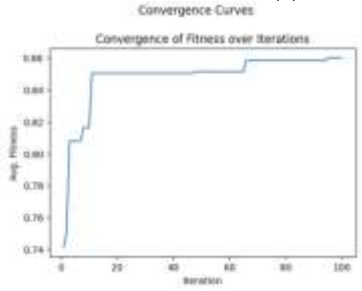
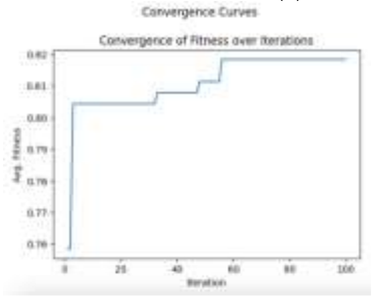
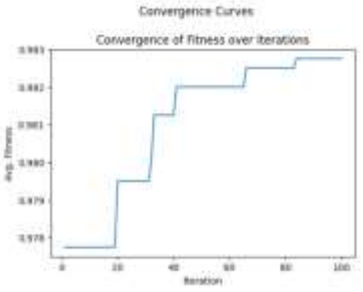
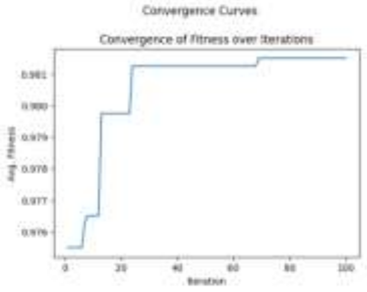
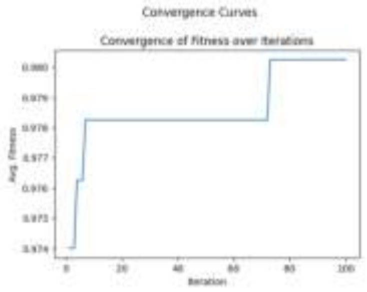
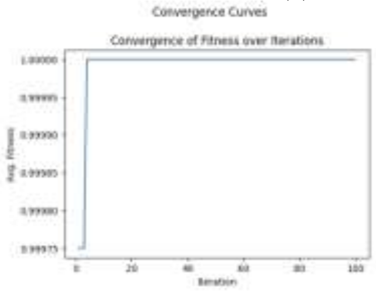
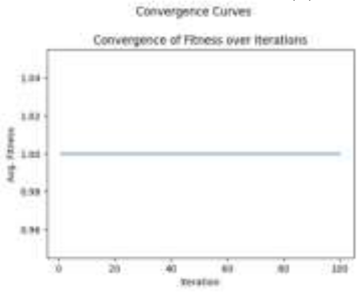
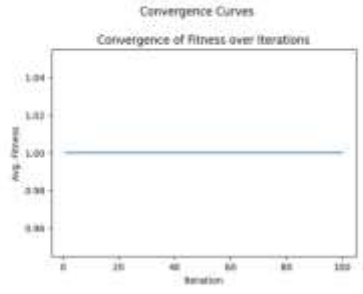
	GA	PSO	SC
NB	Final Accuracy: 90.05% Accuracy +: 23.77% Features dimension: 37 Time: 43.837 second(s) 	Final Accuracy: 86.00% Accuracy +: 14.23% Features dimension: 35 Time: 41.484 second(s) 	Final Accuracy: 81.83% Accuracy +: 10.13% Features dimension: 36 Time: 90.730 second(s) 
SVM	Final Accuracy: 98.28% Accuracy -: 0.41% Features dimension: 39 Time: 523.692 second(s) 	Final Accuracy: 98.15% Accuracy -: 0.54% Features dimension: 36 Time: 507.087 second(s) 	Final Accuracy: 98.03% Accuracy -: 0.66% Features dimension: 44 Time: 1014.056 second(s) 
RF	Final Accuracy: 100.00% Accuracy +: 0.00% Features dimension: 37 Time: 828.092 second(s) 	Final Accuracy: 100.00% Accuracy +: 0.00% Features dimension: 22 Time: 776.483 second(s) 	Final Accuracy: 100.00% Accuracy +: 0.00% Features dimension: 22 Time: 1874.569 second(s) 

Figure 9 shows that each meta-heuristic algorithm uses ten (10) agents running 100 iterations and prints the execution time. Table 3 shows the results for each meta-heuristic combined with classifier algorithms.

Since NB obtained the lowest accuracy, Table 3 shows that NB obtained higher accuracy improvement at +23.77%, +14.23%, and 10.13% for GA, PSO, and CS algorithms, respectively.

These results were obtained when the number of features was only 35 to 37. This means that the combination of NB and GA results improved significantly. Since RF's accuracy was already 100.00%, meta-heuristic algorithms cannot improve. However, the accuracy results are stable at 100.00% when some features are removed until only 22 features are left. In another case, the SVM gives a decline in accuracy between 0.41% and 0.66%. This means the meta-heuristic algorithms could have improved the accuracy when some features were removed. Although reducing the number of features is expected to increase accuracy, sometimes the opposite is true. However, it certainly provides benefits, namely reducing execution time significantly.

Some meta-heuristic algorithms add more complexity to the process, significantly increasing execution time. In the pattern shown in Table 3, CS gives the higher complexity, followed by GA and PSO for all classifier algorithms. The PSO gives a less significant complexity increment, indicated by less improved execution time.

Overall, combining NB and GA gives the best performance regarding higher accuracy improvement, at +23.77%, followed by NB-PSO and NB-CS. The worst performance was when combined SVM and SC at -0.66%, which used 44 features.

Conclusion

The experiment began with the collected dataset and the pre-processing. It explored and evaluated combining some classifier algorithms with meta-heuristic algorithms. The results were collected, and the analysis was done. The first process was to get accuracy without feature selection involved. These results are obtained as a baseline of how much performance improvement can be achieved when applying meta-heuristic algorithms.

The following process was used to determine accuracy and calculate the accuracy improvement based on a combination of classifier and meta-heuristic algorithms. The execution times are also collected. The final results, combining NB and GA, give the best performance regarding higher accuracy improvement, at +23.77%, followed by NB-PSO and NB-CS. The worst performance was when combined SVM and SC, at -0.66%, which used 44 features. Regarding execution time, the results show that CS has the highest complexity, followed by GA and PSO for all classifier algorithms. The PSO gives a less significant increment of complexity.

References

Afshar, M., & Usefi, H. (2022). Optimizing feature selection methods by removing irrelevant features using sparse least squares. *Expert Systems with Applications*, *200*, 116928. https://doi.org/10.1016/j.eswa.2022.116928

Akbari, M., Henteh, M., (2019). Comparison of Genetic Algorithm (GA) and Particle Swarm Optimization Algorithm (PSO) for Discrete and Continuous Size Optimization of 2D Truss Structures, *Journal of Soft Computing in Civil Engineering*. Vol. 3(2). pp-76-79. https://doi.org/10.22115/scce.2019.195713.1117

Al-Qaness, M. A. A., Ewees, A. A., Fan, H., & El Aziz, M. A. (2020). Optimization method for forecasting confirmed cases of COVID-19 in China. *Journal of Clinical Medicine*, *9* (3). https://doi.org/10.3390/jcm9030674

Alnuaimi, A.F. and Albaldawi, T.H.K, (2024). An overview of machine learning classification technique, *Bio Web of Conference* Vol. 97(4). https://doi.org/10.1051/bioconf/20249700133

Encord Blog, (2023). Data Cleaning & Data Preprocessing for Machine Learning, (Online), accessed online on 20/07/2024, https://encord.com/blog/data-cleaning-data-preprocessing/

Gangadhara Moorthy, S., & Pravin, A. (2021). An Approach for Optimal Feature Selection in Machine Learning using Global Sensitivity Analysis. *International Journal of Advanced Computer Science and Applications*, *12*. https://doi.org/10.14569/IJACSA.2021.0120676

Guha, Ritam., et al., (2022). Py_FS: A Python Package for Feature Selection Using Meta-Heuristic Optimization Algorithms. (Online), accessed online on 20/07/2024, https://github.com/Ritam-Guha/Py_FS.

Guia, Marco., Silva, R.R., and Bernandino, J., (2019), Comparison of Naïve Bayes, Support Vector Machine, Decision Tree, and Random Forest on Sentiment Analysis., KDIR 2019 - 11th International Conference on Knowledge Discovery and Information Retrieval. http://dx.doi.org/10.5220/0008364105250531

Himawan, S. N., Rendi, & Nugraha, N. B. (2023). Feature Selection Menggunakan Algoritma Meta-Heuristik. *Journal of Practical Computer Science*, *2*(2 SE-), 84–89. https://doi.org/10.37366/jpcs.v2i2.2289

Mukhlis, I., Pipin, S., Judijanto, L., Reba, F., Mandowen, S., Al-Husaini, M., & Laili, N. (2024). *ALGORITMA PEMBELAJARAN MESIN (Dasar, Teknik, dan Aplikasi)*.

Setiawan, E., & Ginting, M. (2014). PEMILIHAN METODE METAHEURISTIK MENGGUNAKAN FUZZY ANALYTICAL HIERARCHY PROCESS UNTUK MENYELESAIKAN MASALAH PERANCANGAN TATA LETAK FASILITAS BERORIENTASI PROSES. *Jurnal Teknik Dan Ilmu Komputer Ukrida*, *3*, 346–359.

Taye, M.M, (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions, *Computers*. Vol. 12(5). https://doi.org/10.3390/computers12050091

Wang, S., Tang, J., & Liu, H. (2016). *Feature Selection* (pp. 1–9). https://doi.org/10.1007/978-1-4899-7502-7_101-1