Published *online* on the journal's web page : http://innovatics.unsil.ac.id

Innovation in Research of Informatics (INNOVATICS)

| ISSN (Online) 2656-8993 |

Comparison of Machine Learning Algorithms in Detecting Contaminants in Drinkable Water

Souhayla Elmeftahi[1], Maulana Decky Rakhman[2], Alam Rahmatulloh[3]

[1]*Data Engineer, École Nationale des Sciences Appliquées d'Al Hoceima, 32003, Morocco*
[2]*Department of Informatics, Faculty of Engineering, Siliwangi University, Tasikmalaya, 46115, Indonesia*

[1]*souhayla.elmeftahi@etu.uae.ac.ma,* [2]*207006036@student.unsil.ac.id,* [2]*alam@unsil.ac.id*

ARTICLE INFORMATION

Article History:
Received: March 5, 2024
Last Revision: April 2, 2024
Published *Online*: April 4, 2024

KEYWORDS

Water Quality,
Machine Learning,
Classification Algorithms,
Algorithm Comparison

CORRESPONDENCE

Phone: +212674500530

E-mail: souhayla.elmeftahi@etu.uae.ac.ma

ABSTRACT

Water is a natural resource that is crucial for the continuity of human life. The availability of clean, quality water is the human right of every individual and an important factor for living a decent life. However, water quality can be affected by various harmful substances, minerals, and contaminants, often originating from various sectors such as industry, agriculture, residential and energy. One effort to maintain water quality is by direct manual inspection such as the WQI and STORET methods. However, this method requires a lot of time. Therefore, machine learning is needed to help check water quality quickly. There have been many previous studies that have studied this problem with various algorithms. However, there is still a gap between which algorithm is best in classifying water quality because there are many existing algorithms. For this reason, a comparison of 7 algorithms was carried out to determine which method is best for classifying water quality by comparing metric values. The accuracy results obtained show that the Random Forest algorithm is the most effective in classifying water quality with the highest accuracy of around 84.8%, followed by the XGBoost and CatBoost algorithms which also show good performance, namely with an accuracy of 82.9% and 80.2%. Behind that is followed by the Decision Tree algorithm with an accuracy of 77.3%, SVM with an accuracy of 72.3%, K-NN with an accuracy of 70.6%, and finally AdaBoost with the smallest accuracy value, namely 63.33%.

1. INTRODUCTION (capital, 10pt, bold)

Water is one of the natural resources that is vital for human life, with around 71% of the earth's surface consisting of water [1], [2]. Every individual has the human right to clean water, which is an important prerequisite for living a decent and dignified life. Therefore, it is necessary to maintain the quality and quantity of water well [3], [4]. Water is a complex substance with many substances and minerals. However, water is susceptible to contamination by dangerous bacteria and minerals so that some of these substances and minerals are not safe for human consumption [2], [5]. Water pollution generally comes from various sectors, including industry, agriculture, housing, and energy. One example of the impact is rivers which often become polluted and dirty [4], [6].

There are many efforts to maintain water quality, which involve checking for disease or bacterial contamination in the water. Precautions will be taken if there is a decline in water quality [7]. Water quality can be assessed based on various parameters, including microbiological aspects, inorganic chemistry, physical characteristics, and other chemical parameters. Water quality parameters relate to minerals dissolved in water. To determine whether water meets health standards, you must understand the composition of the minerals and substances contained in the water. Water quality classification is usually carried out through manual calculations such as using the Water

Quality Index (WQI) and STORET methods. However, this method requires a lot of time to calculate, so an automatic system is needed to simplify the process [2], [8].

Machine Learning is a branch of artificial intelligence that can overcome this. Machine Learning can focus on utilizing data and algorithms to imitate the human learning process with the aim of increasing accuracy and the level of intelligence [9], [10]. Machine Learning can be used in various contexts to solve various problems by analyzing existing data and executing specific tasks [11], [12]. In this research, the Support Vector Machine, Random Forest, Decision Tree, Extreme Gradient Boosting, Adaptive Boosting, CatBoost and K-Nearest Neighbors (K-NN) algorithms were used. These algorithms can be used in the case of water quality classification [2], [5], [8], [13]–[16].

This research aims to compare metric values, namely accuracy, precision, recall, and f1-score, from various algorithms such as SVM, Random Forest, Decision Tree, XGBoost, AdaBoost, CatBoost, and K-NN in the context of water quality identification so that methods can be found. optimal algorithm for water quality identification based on maximum accuracy results.

2. RELATED WORK

Several previous studies have used various machine learning algorithms to assess water quality. Some of these studies are used as references or comparisons because they use research methods or topics that are like those carried out in this study. Research by Weiskhy, et al. [17] concluded that the use of the SVM-PSO algorithm resulted in an accuracy of 84.81% and an AUC value of 0.898. Then, the C4.5-PSO algorithm produces an accuracy of 80.00% and an AUC value of 0.787. Priscolius Evrolino Jennes, et al. [14] analyzed the feasibility of water sources in Indonesia and stated that the accuracy results were 71% for the SVM algorithm, 61% for the Decision Tree algorithm, and 67% for the Random Forest algorithm. Fauzi, et al. [2] concluded that the Decision Tree algorithm achieved an accuracy of 94.94% and an AUC of 0.865, the Naïve Bayes algorithm obtained an accuracy of 84.79% and an AUC of 0.814 and the K-NN algorithm achieved an accuracy of 87.86% with an AUC of 0.725. So the Decision Tree algorithm is considered the most accurate algorithm in classifying water quality. Maulana, et al. [5] states that the K-NN algorithm gets an accuracy of 82.42% and the Naïve Bayes algorithm gets an accuracy of 70.32%. This research confirms that the KNN method is the best method for water quality classification. G L Pritalia [8] summarized the research results covering the accuracy of various algorithms. Decision Tree has an accuracy of 79%, Random Forest 85%, SVM 68%, Logistic Regression 50%, K-NN 77% and Naïve Bayes 57%. The best accuracy is obtained by the Random Forest algorithm. Muhammad, et al. [13] concluded that the Random Forest algorithm can predict water quality for 82% of data that can be classified as water that can be consumed or not. This shows that Random Forest produces good precision and sensitivity. Research by Taufik, et al. [15] shows that the CatBoost algorithm gets an accuracy of 68%, the Gradient Boosting algorithm is 60%, and the AdaBoost is 58%, so the CatBoost algorithm has the highest accuracy. Hasriq, et al. [16] concluded that the XGBoost model had better performance with 94% accuracy compared to the SVM model which only had 67% accuracy.

Algorithms such as SVM, K-NN, Naive Bayes, ANN, Hierarchical Clustering, Decision Tree and Random Forest are machine learning algorithms that are commonly used for classification [10]. However, the machine learning algorithms AdaBoost, XGBoost and CatBoost also show quite good results in water quality classification [15], [16]. The novelty of this research is comparing the metric results of the SVM, Random Forest, Decision Tree, XGBoost, AdaBoost, CatBoost, and K-NN algorithms to find out which method is the most efficient and optimal for water quality classification. This research was conducted because no similar research has been found regarding the comparison of the Support Vector Machine, Extreme Gradient Boostin, Random Forest, Decision Tree, Adaptive Boosting, CatBoost and K-Nearest Neighbors algorithms in water quality prediction.

3. METHODOLOGY

The stages of this research start from data collection and continue to analysis of the results. Figure 1 below shows the sequence of steps in this research.

FIGURE 1. RESEARCH STAGES

The stages in Figure 1 carry out data collection regarding water quality. Next, the data preprocessing process is carried out. After that, the SVM, Random Forest, Decision Tree, XGBoost, AdaBoost, CatBoost and K-NN algorithms were implemented on the water quality data. The next step is to test the model with various metrics, namely Accuracy, Precision, Recall, and F1-Score to evaluate model performance. Finally, analyze the model testing results.

3.1 Data Collection

The data used is data regarding water quality, including values such as pH, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_carbon, Trihalomethanes, Turbidity, and Potability. This data was obtained from Kaggle sources which were published by Aditya Kadiwal in 2021 (*https://www.kaggle.com/datasets/adityakadiwal/water-potability*), consisting of 3276 rows and 10 columns. This data is used to make predictions regarding the suitability of water, namely whether the water can be consumed or not.

3.2 Data Pre-Processing

Data pre-processing is the initial stage in data preparation, which includes cleaning, handling missing data, and adapting raw data to fit the format required for subsequent analysis. Steps in preprocessing include attribute selection, missing data handling, outlier handling, and data transformation.

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

FIGURE 2. WATER QUALITY DATASET

3.2.1 Feature Selection

The attributes used for prediction in this research are the Potability attribute as a label or target to be predicted with binary values 0 and 1 while the other 8 attributes are used as features that will be used to build a prediction model.

3.2.2 Missing Values Processing

Handle missing values in the data, either by filling in missing values or using imputation techniques. In this case, to avoid bias, missing values are handled by deleting the rows containing the missing data.

3.2.3 Data Transformation

Perform data transformation if necessary, such as normalization or standardization. In this case using a standard scaler method, this can help the machine learning model perform better.

3.3 Model Training

At this stage, implementation is carried out using the Support Vector Machine, Extreme Gradient Boosting, Random Forest, Decision Tree, Adaptive Boosting, CatBoost and K-Nearest Neighbors algorithms.

3.3.1 Algorithm SVM

Support Vector Machine operates by dividing the training data using an optimal hyperplane. This hyperplane is the plane that separates two classes with the largest distance between them. A support vector is a portion of the training data in the input space. C, Gamma, and kernel values are some of the parameters used by the SVM algorithm. The C and Gamma values used in this study range between 0.001 and 1000. Three types of kernels (linear, poly, and radial basis) are used by SVM [8].

3.3.2 Algorithm XGBoost

Extreme Gradient Boosting is a technique in machine learning that is used for regression analysis and classification based on the Gradient Boosting Decision Tree (GBDT) concept. XGBoost combines the concepts of boosting and optimization in the construction of a Gradient Boosting Machine (GBM). In the boosting method, new models are built to predict errors from previous models, and additions to these models continue until there is no longer a significant improvement in the errors. This algorithm uses gradient descent to minimize errors when creating new models, so it is known as gradient boosting [18].

$$\hat{y}_i^t = \sum_{k=1}^{t} f_k(x_i) \quad (1)$$

Information:
$\hat{y}_i^t$ = Final tree model
$f_k(x_i)$ = New model built
t = The total number of models from the base tree models

3.3.3 Algorithm Random Forest

Random Forest is a group of trees that work together to make decisions. Random Forest has many slightly different trees. The main concept of Random Forest is that any tree may provide good predictions in some cases but may be too precise for the training data. To get more reliable results and reduce overfitting, many different trees are built, and their prediction results are combined. The Random Forest approach combines various Decision Trees. The result is obtained by taking the average prediction, which helps improve accuracy and control overfitting [8].

3.3.4 Algorithm Decision Tree

Decision Trees are one of the algorithms commonly used in classification, famous for their ability to produce decision rules that are easy to understand. Basically, a Decision Tree learns from data by building a hierarchy of "if-else" questions that lead to a decision. The Decision Tree process involves transforming tabular data into a tree structure, which can then be simplified into rules. Some of the algorithms used to build Decision Trees include ID3, CART, and C4.5. These algorithms simplify the complex relationships between input variables and target variables by dividing the original variables into more meaningful groups. In this research, parameters such as gini, entropy, and max_depth (maximum depth of the tree) are set within a certain range for the formation of a Decision Tree [8].

3.3.5 Algorithm AdaBoost

The AdaBoost algorithm builds a combined tree model repeatedly. Wrongly classified data is given a higher weight than correct data at each iteration, so as to correct data that was wrongly classified in the previous iteration. Predictions for each model are combined, usually through voting, to determine a class label. New data predictions are based on majority weights [19].

3.3.6 Algorithm CatBoost

CatBoost is an algorithm that adopts the gradient boosting method, using a binary decision tree as a basic predictor. CatBoost can handle categorical and ordered features and prevent overfitting through Bayesian estimators. In the CatBoost algorithm, the use of Prediction Values Change (PVC) or Loss Function Change (LFC) is used to determine the ranking of features in model development [20].

3.3.7 Algorithm K-NN

The K-Nearest Neighbor algorithm is based on the basic idea of finding several k nearest neighbors in the training data while testing new data by calculating the distance between them. This method groups new data by measuring the distance between the new data and several nearest neighbors in the training data. KNN is included in the instance-based learning category, where training data is stored and when it must classify new data that does not yet have a label, the process is done by comparing the similarity of the new data with existing training data [5].

$$euc = \sqrt{\sum_{i=1}^{n} (x_{2i} - x_{1i})^2} \quad (2)$$

Information:
x_1 = test data
x_2 = training data
i = data variables
n = data dimensions

3.4 Model Testing

Before testing, the initial data is divided into two parts, namely train data and test data with a ratio of 80:20. In the model testing process, the Metric method is used as a tool for evaluation. Metrics are indicators used to measure the performance of a machine learning model. The performance measurements used in this research consist of accuracy, precision, recall and f1-score.

3.4.1 Accuracy

Measures the percentage of overall prediction accuracy. A value of 0 for accuracy indicates a perfect prediction, while a value of 0 indicates a prediction that is not correct at all.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} euc = \sqrt{\sum_{i=1}^{n} (x_{2i} - x_{1i})^2} \quad (3)$$

3.4.2 Precision

Calculates the ratio of all correct positive data. Recall shows how well the machine learning model finds all positive data. The recall value ranges from 0 to 1.

$$Precision = \frac{TP}{TP+FP} Accuracy = \frac{TP+TN}{TP+TN+FP+FN} euc = \sqrt{\sum_{i=1}^{n} (x_{2i} - x_{1i})^2} \quad (4)$$

3.4.3 Recall

Calculates the ratio of correct positive predictions for each prediction. Precision shows how often a machine learning model makes correct positive predictions; the value ranges between 0 and 1.

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

3.4.4 F1-Score

Harmonic mean of precision and recall. The F1 score is a measure to measure the balance between the two metrics, and a high value indicates a good balance between the two metrics.

$$F1 - measure = \frac{2\,x\,precision\,x\,recall}{precision + recall} \quad (6)$$

Information:
TP = True positive
TN = True negative
FN = False Negatif
FP = False Positif

3.4.5 Analysis of Result

At this stage, we discuss the comparison of the results of each algorithm that has been explained previously, to find out which algorithm has the best test results.

4. RESULT AND DISCUSSION

4.1 Dataset

The dataset used in this research is water quality data with CSV data type for the identification process in comparing the accuracy results of the seven methods used, namely Support Vector Machine, Random Forest, Decision Tree, Extreme Gradient Boosting, Adaptive Boosting, CatBoost and K-Nearest Neighbors . Based on previous data pre-processing, the following dataset is obtained.

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
1560	-0.407446	0.568164	1.573662	-0.372683	-0.617157	-0.748305	-1.892862	0.662178	-0.555826
423	0.555940	-0.475971	-0.314589	1.699008	-0.189016	-0.491868	-0.942301	0.030787	0.114900
1324	0.947519	-0.656177	-1.799511	-0.041414	1.813628	-0.277269	0.529336	-0.896731	-0.983079
2823	-0.119364	1.365294	-1.653110	0.577283	2.559799	0.754484	1.212247	-1.317969	0.455175
3203	1.249434	-2.285840	0.010198	1.391305	-1.150261	-0.660452	0.049381	-0.162005	0.354107
...	...	...	...	...	...	...	...	...	...
1917	-0.148668	-0.766217	-0.190093	-0.020895	0.776554	-0.309461	-0.112573	-0.931348	0.648187
744	0.245351	-0.310176	-0.959942	-1.513441	1.048789	-1.253825	-0.814454	-0.029422	0.077977
660	0.285319	-0.125936	0.243408	0.082071	0.065342	-0.695288	1.155312	0.234418	-0.345227
550	0.232140	1.230347	0.358026	0.104292	-0.402319	-0.594303	-2.563289	-0.290229	-0.036325
1213	0.026784	-0.514756	-0.259436	0.617962	0.108220	-0.755290	0.911611	1.325753	-0.259269

2400 rows × 9 columns

FIGURE 3. WATER QUALITY DATASET AFTER PREPROCESSING

4.2 Algorithm Implementation and Testing

4.2.1 Algorithm SVM

A Support Vector Machine (SVM) classifier is implemented using the SVC class from the scikit-learn library. Hyperparameter tuning for SVM is performed using grid search (GridSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
svm = SVC()

params_svm = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf'], 'gamma': ['scale', 'auto']}
grid_svm = GridSearchCV(svm, param_grid=params_svm, cv=10)
grid_svm.fit(X_train, y_train)

svm_predict = grid_svm.predict(X_test)
svm_acc_score = accuracy_score(y_test, svm_predict)

print("Akurasi SVM:", svm_acc_score * 100, '\n')
print(classification_report(y_test, svm_predict))
```

```
Akurasi SVM: 72.29166666666667

              precision    recall  f1-score   support

           0       0.74      0.75      0.74       256
           1       0.71      0.69      0.70       224

    accuracy                           0.72       480
   macro avg       0.72      0.72      0.72       480
weighted avg       0.72      0.72      0.72       480
```

FIGURE 4. SVM ACCURACY

4.2.2 Algorithm XGBoost

An XGBoost classifier is implemented using the XGBClassifier class from the XGBoost library. Hyperparameter tuning for XGBoost was performed using random search (RandomizedSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
xgb = XGBClassifier(eval_metric='logloss', use_label_encoder=False)

params_xgb = {'n_estimators': [50, 100, 250, 400, 600, 800, 1000], 'learning_rate': [0.2, 0.5, 0.8, 1]}
rs_xgb = RandomizedSearchCV(xgb, param_distributions=params_xgb, cv=10)
rs_xgb.fit(X_train, y_train)

xgb_predict = rs_xgb.predict(X_test)
xgb_acc_score = accuracy_score(y_test, xgb_predict)

print("Akurasi XGBoost:", xgb_acc_score * 100, '\n')
print(classification_report(y_test, xgb_predict))
```

```
Akurasi XGBoost: 82.91666666666667

              precision    recall  f1-score   support

           0       0.87      0.80      0.83       256
           1       0.79      0.86      0.82       224

    accuracy                           0.83       480
   macro avg       0.83      0.83      0.83       480
weighted avg       0.83      0.83      0.83       480
```

FIGURE 5. XGBOOST ACCURACY

4.2.3 Algorithm Random Forest

A Random Forest classifier is implemented using the RandomForestClassifier class from scikit-learn. Hyperparameter tuning for Random Forest is performed using grid search (GridSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
rf = RandomForestClassifier()

params_rf = {'n_estimators': [100, 200, 350, 500], 'min_samples_leaf': [2, 10, 30]}
grid_rf = GridSearchCV(rf, param_grid=params_rf, cv=10)
grid_rf.fit(X_train, y_train)

rf_predict = grid_rf.predict(X_test)
rf_acc_score = accuracy_score(y_test, rf_predict)

print("Akurasi Random Forest:", rf_acc_score * 100, '\n')
print(classification_report(y_test, rf_predict))
```

```
Akurasi Random Forest: 84.79166666666667

              precision    recall  f1-score   support

           0       0.87      0.84      0.86       256
           1       0.83      0.85      0.84       224

    accuracy                           0.85       480
   macro avg       0.85      0.85      0.85       480
weighted avg       0.85      0.85      0.85       480
```

FIGURE 6. RANDOM FOREST ACCURACY

4.2.4 Algorithm Decision Tree

A Decision Tree classifier is implemented using the DecisionTreeClassifier class from scikit-learn. Hyperparameter tuning for the Decision Tree is carried out using grid search (GridSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
dt = DecisionTreeClassifier(class_weight='balanced', random_state=2021)

para_dt = {'criterion':['gini','entropy'],'max_depth':np.arange(2, 22, 2),
           'min_samples_split': [2, 3, 4], 'min_samples_leaf':[1, 2, 3, 4]}
grid_dt = GridSearchCV(dt, param_grid=para_dt, cv=10) #grid search decision tree for 5 fold cv

dt_predict = grid_dt.predict(X_test)
dt_acc_score = accuracy_score(y_test, dt_predict)

print("Akurasi Decision Tree:", dt_acc_score * 100, '\n')
print(classification_report(y_test, dt_predict))
```

```
Akurasi Decision Tree: 77.29166666666667

              precision    recall  f1-score   support

           0       0.86      0.68      0.76       256
           1       0.71      0.88      0.78       224

    accuracy                           0.77       480
   macro avg       0.78      0.78      0.77       480
weighted avg       0.79      0.77      0.77       480
```

FIGURE 7. DECISION TREE ACCURACY

4.2.5 Algorithm AdaBoost

An AdaBoost classifier is implemented using the AdaBoostClassifier class from scikit-learn. Hyperparameter tuning for AdaBoost was performed using grid search (GridSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
ada = AdaBoostClassifier()

params_ada = {'n_estimators': [50, 100, 250, 400, 500, 600], 'learning_rate': [0.2, 0.5, 0.8, 1]}
grid_ada = GridSearchCV(ada, param_grid=params_ada, cv=10)
grid_ada.fit(X_train, y_train)

ada_predict = grid_ada.predict(X_test)
ada_acc_score = accuracy_score(y_test, ada_predict)

print("Akurasi AdaBoost:", ada_acc_score * 100, '\n')
print(classification_report(y_test, ada_predict))
```

```
Akurasi AdaBoost: 63.33333333333333

              precision    recall  f1-score   support

           0       0.66      0.66      0.66       256
           1       0.61      0.61      0.61       224

    accuracy                           0.63       480
   macro avg       0.63      0.63      0.63       480
weighted avg       0.63      0.63      0.63       480
```

FIGURE 8. ADABOOST ACCURACY

4.2.6 Algorithm CatBoost

A CatBoost classifier is implemented using the CatBoostClassifier class from the CatBoost library. Hyperparameter tuning for CatBoost was performed using grid search (GridSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
catboost = CatBoostClassifier(verbose=False)

params_catboost = {'n_estimators': [100, 200, 300, 400],
                   'learning_rate': [0.1, 0.2, 0.3, 0.4], 'depth': [4, 6, 8]}
grid_catboost = GridSearchCV(catboost, param_grid=params_catboost, cv=10)
grid_catboost.fit(X_train, y_train)

catboost_predict = grid_catboost.predict(X_test)
catboost_acc_score = accuracy_score(y_test, catboost_predict)

print("Akurasi CatBoost:", catboost_acc_score * 100, '\n')
print(classification_report(y_test, catboost_predict))
```

```
Akurasi CatBoost: 80.20833333333334

              precision    recall  f1-score   support

           0       0.83      0.79      0.81       256
           1       0.77      0.81      0.79       224

    accuracy                           0.80       480
   macro avg       0.80      0.80      0.80       480
weighted avg       0.80      0.80      0.80       480
```

FIGURE 9. CATBOOST ACCURACY

4.2.7 Algorithm K-NN

A K-Nearest Neighbors (K-NN) classifier is implemented using the KNeighborsClassifier class from scikit-learn. Hyperparameter tuning for K-NN was performed using grid search (GridSearchCV) to find the best combination of hyperparameters. The model is trained on training data, and predictions are made on test data.

```
knn = KNeighborsClassifier(n_jobs=-1)

para_knn = {'n_neighbors':np.arange(3, 20)}  #parameters of knn
grid_knn = GridSearchCV(knn, param_grid=para_knn, cv=10) #search knn for 5 fold cross validation
grid_knn.fit(X_train, y_train)

knn_predict = knn.predict(X_test)
knn_acc_score = accuracy_score(y_test, knn_predict)

print("Akurasi K-Nearest Neighbors:", knn_acc_score * 100, '\n')
print(classification_report(y_test, knn_predict)
```

```
Akurasi KNN: 70.625

              precision    recall  f1-score   support

           0       0.75      0.67      0.71       256
           1       0.67      0.75      0.70       224

    accuracy                           0.71       480
   macro avg       0.71      0.71      0.71       480
weighted avg       0.71      0.71      0.71       480
```

FIGURE 10. K-NN ACCURACY

4.3 Prediction Result

From the method comparison results, the water quality classification prediction results obtained using the SVM, Random Forest, Decision Tree, XGBoost, AdaBoost, CatBoost and K-NN algorithms are as follows.

TABLE 1. PERFORMANCE COMPARISON OF MACHINE LEARNING ALGORITHM

Algorithm Machine Learning	Accuracy	Precision	Recall	F1-Score
SVM	0.72	0.72	0.72	0.72
XGBoost	0.83	0.83	0.83	0.82
Random Forest	0.85	0.85	0.84	0.85
Decision Tree	0.77	0.78	0.78	0.77
AdaBoost	0.63	0.64	0.64	0.64
CatBoost	0.80	0.80	0.80	0.80
K-NN	0.71	0.71	0.71	0.70

From this table it can be seen that the difference in the average Accuracy, Precision, Recall and F1-Score values of each method only has a slight difference, namely around 0.01 or even no difference.

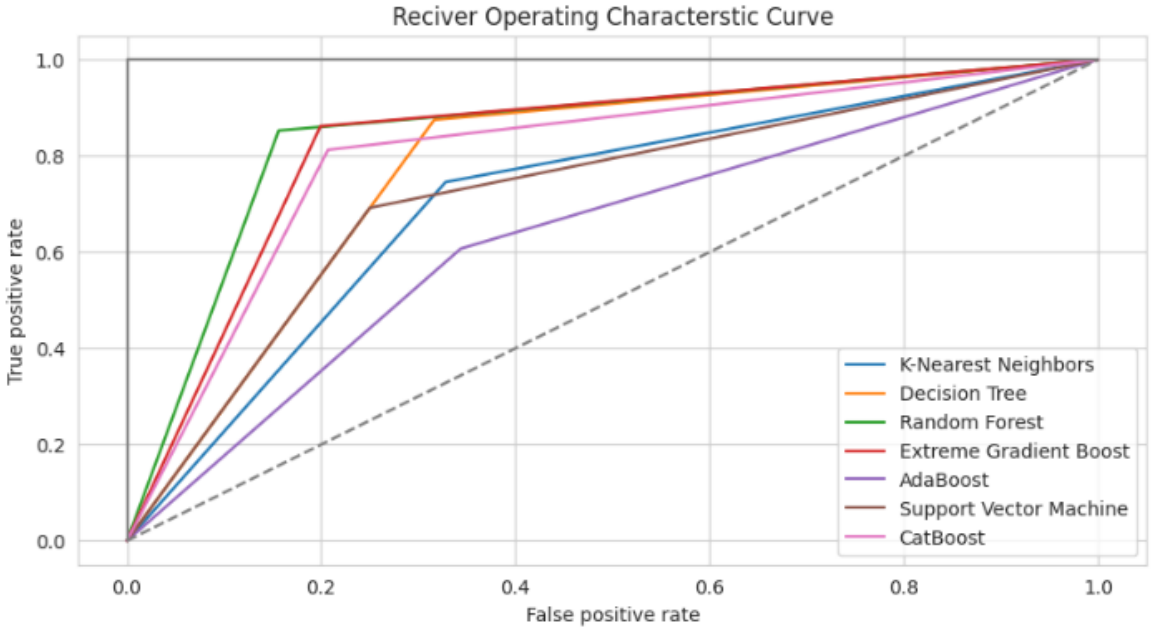

FIGURE 11. ROC CURVE COMPARISON OF MODEL ACCURACY

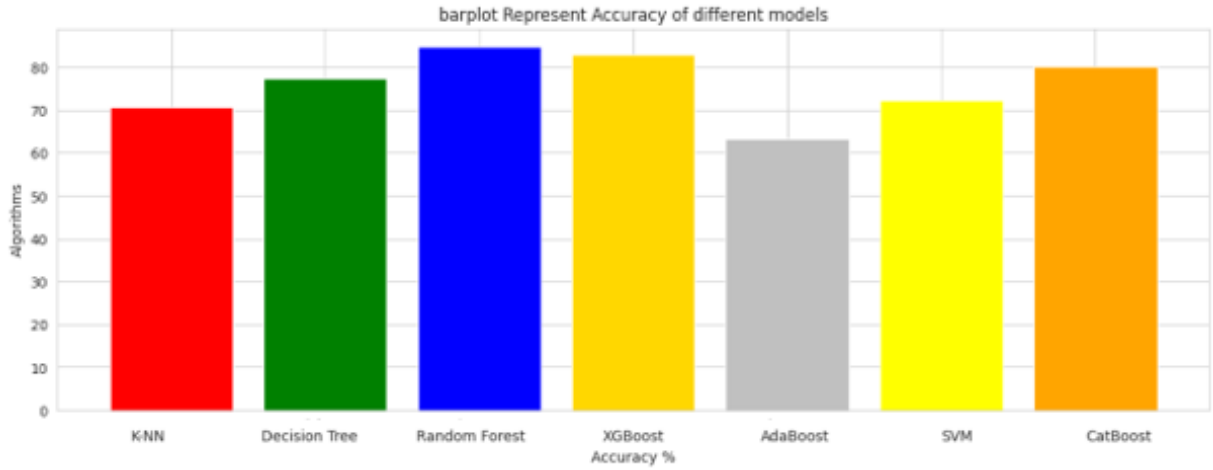

FIGURE 12. MODEL ACCURACY COMPARISON BAR GRAPH

After making a comparison using the same test data and training data with an initial dataset of 3276 rows and 10 columns, the results show that the methods used have different levels of accuracy. The algorithm with the lowest accuracy value is AdaBoost which has an accuracy of around 63.33%, K-NN around 70.6%, SVM around 72.3%, Decision Tree around 77.3%, CatBoost around 80.2%, XGBoost around 82.9%, and Random Forest with the highest accuracy value around 84.8 %. From the test results it can be seen that the Random Forest, XGBoost and CatBoost algorithms can get better accuracy than other algorithms, namely above 80%.

5. CONCLUSIONS

Based on research that has been carried out for water quality classification, the implementation uses the Support Vector Machine, Extreme Gradient Boosting, Random Forest, Decision Tree, Adaptive Boosting, CatBoost, and K-Nearest Neighbors algorithms with a division of 80% training data and 20% test data, resulting in Accuracy values are quite varied. The AdaBoost algorithm gets an accuracy of around 63.33%, the K-NN algorithm gets an accuracy of around 70.6%, the SVM algorithm gets an accuracy of around 72.3%, the Decision Tree algorithm gets an accuracy of around 77.3%, the CaBoost algorithm gets an accuracy of around 80.2%, the XGBoost algorithm

gets an accuracy of around 82.9% , and the Random Forest algorithm gets an accuracy of around 84.8%. Therefore, it can be concluded that the Random Forest algorithm has proven to be the most effective in classifying water quality with the highest accuracy, namely around 84.8%. Followed by the XGBoost, CatBoost, Decision Tree, K-NN, SVM algorithms and finally AdaBoost with the lowest accuracy value.

REFERENCES

[1] S. Kusumawardani and A. Larasati, "ANALISIS KONSUMSI AIR PUTIH TERHADAP KONSENTRASI," Jurnal Holistika, vol. 4, no. 2, p. 91, Nov. 2020, doi: 10.24853/holistika.4.2.91-95.

[2] F. Y. Rahman, I. I. Purnomo, and N. Hijriana, "PENERAPAN ALGORITMA DATA MINING UNTUK KLASIFIKASI KUALITAS AIR," Technologia : Jurnal Ilmiah, vol. 13, no. 3, p. 228, Aug. 2022, doi: 10.31602/tji.v13i3.7070.

[3] D. Hartanti and A. I. Pradana, "Komparasi Algoritma Machine Learning dalam Identifikasi Kualitas Air," SMARTICS Journal, vol. 9, no. 1, 2023, doi: 10.21067/smartics.v9i1.8113.

[4] D. Kamalia, "Analisis Pencemaran Air Sungai Akibat Dampak Limbah Industri Batu Alam di Kecamatan Depok Kabupaten Cirebon." [Online]. Available: http://jurnalkesehatan.unisla.ac.id/index.php/jev/index

[5] M. A. Rahman, N. Hidayat, and A. A. Supianto, "Komparasi Metode Data Mining K-Nearest Neighbor Dengan Naïve Bayes Untuk Klasifikasi Kualitas Air Bersih (Studi Kasus PDAM Tirta Kencana Kabupaten Jombang)," 2018. [Online]. Available: http://j-ptiik.ub.ac.id

[6] I. Wayan Eka Artajaya and N. Kadek Felyanita Purnama Putri, "FAKTOR-FAKTOR PENYEBAB TERJADINYA PENCEMARAN AIR DI SUNGAI BINDU," Jurnal Hukum Saraswati (JHS), 2021, doi: 10.36733/jhshs.v2i2.

[7] Y. T. K. Yunior and K. Kusrini, "Sistem Monitoring Kualitas Air Pada Budidaya Perikanan Berbasis IoT dan Manajemen Data," Creative Information Technology Journal, vol. 6, no. 2, p. 153, Feb. 2021, doi: 10.24076/citec.2019v6i2.251.

[8] Generosa Lukhayu Pritalia, "Analisis Komparatif Algoritme Machine Learning dan Penanganan Imbalanced Data pada Klasifikasi Kualitas Air Layak Minum," KONSTELASI: Konvergensi Teknologi dan Sistem Informasi, vol. 2, no. 1, Apr. 2022, doi: 10.24002/konstelasi.v2i1.5630.

[9] N. Muniroh and E. Agus Priatno, "PENERAPAN ALGORITMA K-NN PADA MACHINE LEARNING UNTUK KLASIFIKASI KUALITAS AIR BUDIDAYA AKUAPONIK BERBASIS IoT," Jurnal Teknologi dan Bisnis, vol. 4, no. 2, pp. 73–86, Dec. 2022, doi: 10.37087/jtb.v4i2.87.

[10] I. M. Faiza, G. Gunawan, and W. Andriani, "Tinjauan Pustaka Sistematis: Penerapan Metode Machine Learning untuk Deteksi Bencana Banjir," Jurnal Minfo Polgan, vol. 11, no. 2, pp. 59–63, Aug. 2022, doi: 10.33395/jmp.v11i2.11657.

[11] A. Roihan, P. A. Sunarya, and A. S. Rafika, "Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper," IJCIT (Indonesian Journal on Computer and Information Technology), vol. 5, no. 1, May 2020, doi: 10.31294/ijcit.v5i1.7951.

[12] Y. Herdiana and W. Adhitya Geraldine, "PENERAPAN MACHINE LEARNING DENGAN MODEL LINEAR REGRESSION TERHADAP ANALISIS KUALITAS HASIL PETIK THE DI PT. PERKEBUNAN NUSANTARA VIII KEBUN SEDEP."

[13] M. M. Mutoffar et al., "KLASIFIKASI KUALITAS AIR SUMUR MENGGUNAKAN ALGORITMA RANDOM FOREST," vol. 04, 2022.

[14] P. Evrolino Jennes, Y. Wahyuningsih, and A. Dwi Nur Fadlilah, "Prediksi Kelayakan Sumber Air Minum Menggunakan Algoritma Support Vector Machine (SVM)," Prosiding Seminar Nasional Energi, vol. 8, p. 2022.

[15] T. Z. Jasman, M. A. Fadhlullah, A. L. Pratama, and R. Rismayani, "Analisis Algoritma Gradient Boosting, Adaboost dan Catboost dalam Klasifikasi Kualitas Air," Jurnal Teknik Informatika dan Sistem Informasi, vol. 8, no. 2, Aug. 2022, doi: 10.28932/jutisi.v8i2.4906.

[16] H. I. H. Yusri, A. A. Ab Rahim, S. L. M. Hassan, I. S. A. Halim, and N. E. Abdullah, "Water Quality Classification Using SVM And XGBoost Method," in 2022 IEEE 13th Control and System Graduate Research Colloquium (ICSGRC), IEEE, Jul. 2022, pp. 231–236. doi: 10.1109/ICSGRC55096.2022.9845143.

[17] W. S. Dharmawan, "KOMPARASI ALGORITMA KLASIFIKASI SVM-PSO DAN C4.5-PSO DALAM PREDIKSI PENYAKIT JANTUNG," I N F O R M A T I K A, vol. 13, no. 2, p. 31, Jan. 2022, doi: 10.36723/juri.v13i2.301.

[18] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," The Annals of Statistics, vol. 29, no. 5, Oct. 2001, doi: 10.1214/aos/1013203451.

[19] I. A. Rahmi, F. M. Afendi, and A. Kurnia, "Metode AdaBoost dan Random Forest untuk Prediksi Peserta JKN-KIS yang Menunggak," Jambura Journal of Mathematics, vol. 5, no. 1, pp. 83–94, Jan. 2023, doi: 10.34312/jjom.v5i1.15869.

[20] A. Ilmiah Aplikasi Teknologi, Y. Purbolingga, D. Marta Putria, A. Rahmawatia, and B. Wajhi Akramunnas, "JURNAL APTEK Perbandingan Algoritma CatBoost dan XGBoost dalam Klasifikasi Penyakit Jantung," vol. 15, no. 2, pp. 126–133, 2023, [Online]. Available: http://journal.upp.ac.id/index.php/aptek

AUTHORS

Souhayla Elmeftahi

Is a student in the Data Engineering program at the National School of Applied Science, Morocco. Her research interests include data engineering, data science, data analytics, Internet of Things, cloud computing, and related informatics fields.

Alam Rahmatulloh
is a lecturer and researcher in the field of Informatics at the Department of Informatics, Faculty of Engineering, Siliwangi University, Indonesia. He obtained his master's degree in informatics from the School of Electrical and Informatics Engineering, Bandung Institute of Technology in 2015. His research interests include: Microservices, Web Programming, IoT, Cryptography. Experienced in making information systems such as smart campuses, academic systems, and others.

Maulana Decky Rakhman
is student at Department of Informatics, Faculty of Engineering, Siliwangi University, Indonesia. His research interest iclude: datas science Robotics, and Internet of Things..