JUSIKOM PRIMA (Jurnal Sistem Informasi dan Ilmu Komputer Prima)
Vol. 7 No. 1, Agustus 2023 E-ISSN : 2580-2879

COMPARISON OF ENSEMBLE LEARNING ALGORITHM IN CLASSIFYING EARLY DIAGNOSTIC OF DIABETES

Okta Jaya Harmaja1, Irvan Prasetia2,Yosi Victor Hutagalung3, Hendra Ardanis Sirait4***
Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Prima Indonesia
Jalan Sampul No.3, Sei Putih Barat, Medan Petisah
*E-mail : *oktajaya.h@unprimdn.ac.id*

ABSTRACT- Diabetes is a significant public health problem affecting millions worldwide. This study will perform a comparative analysis of three ensemble learning algorithms (Random Forest, AdaBoost, and XGBoost) in classifying diabetes diagnoses. Based on the research that has been carried out, it is concluded that the model with the highest accuracy is Random Forest with a value of 0.86, XGBoost with a value of 0.85, and AdaBoost with a value of 0.82. It can also be concluded that the three models perform well and can be used to classify diabetes. Based on the visualization of the results of Feature Importance that has been made, it can be concluded that the Random Forest and XGBoost algorithms have in common the 3 most essential features, namely Glucose, BMI, and Age. As for AdaBoost, the 3 most critical features are DPF, BMI, and Glucose.

Key Word: Ensemble Learning, Data Science, Comparison, Feature Importance, Diabetes

1. INTRODUCTION

Diabetes is a public health problem affecting millions worldwide [1],[2]. *Diabetes* is a chronic metabolic disorder characterized by high blood glucose levels (hyperglycemia) due to issues with insulin production or insulin resistance. Early identification and diagnosis of diabetes are crucial to preventing serious complications and providing sufferers proper care. In recent years, the development of information technology has brought changes in medicine and health. The use of machine learning algorithms, particularly machine learning algorithms, has shown great potential in assisting the diagnosis and monitoring of diseases, including diabetes.

Previous research has proposed a different machine-learning approach to detect diabetes. Study [3] implements the Support Vector Machine method and achieves an accuracy of 80.73%. Research [4] uses Decision Tree C4.5 and achieves an accuracy of 70.32%, while study [5] uses the K-Nearest Neighbor method and achieves an accuracy of 65%. Although previous research has shown promising results, there is still potential to improve the accuracy and effectiveness of diabetes diagnosis using ensemble learning algorithms. Ensemble learning algorithms, such as Random Forest, AdaBoost, and XGBoost, have effectively solved complex classification problems by combining several learning models.

Therefore, to solve the existing problems, this study will conduct a comparative analysis of three ensemble learning algorithms (Random Forest, AdaBoost, and XGBoost) to classify diabetes diagnoses. This study will evaluate the performance of the three algorithms based on the accuracy, precision, and ROC-AUC curve. By comparing the results of previous research with the proposed ensemble learning method, it is hoped that this research can contribute to increasing the accuracy of diagnosing diabetes.

2. METHODOLOGY

This research was conducted at the System Analyst Laboratory at Universitas Prima Indonesia to develop a classification algorithm model that could determine the feasibility of promoting contract employees to permanent employees. The tools in this study include the Google Colaboratory and various libraries used to process data and create machine learning models. The research phase begins with data collection (Data Acquisition) from the relevant dataset repository. The data is then processed (Data Preprocessing) to translate column names, resolve missing data, and create a balanced dataset. The following critical stage is model building (Model Building) using ensemble learning algorithms such as Random Forest, AdaBoost, and XGBoost. After the model is built, a model evaluation (Model Evaluation) is carried out. Model performance is measured based on metrics such as accuracy, precision, confusion matrix, and area under the curve (AUC) of the ROC curve. This evaluation provides an overview of how well the model can accurately classify diabetes diagnoses.

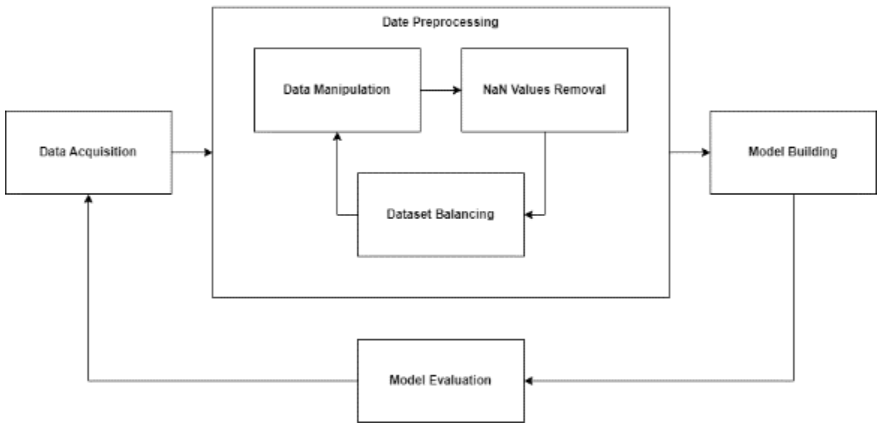
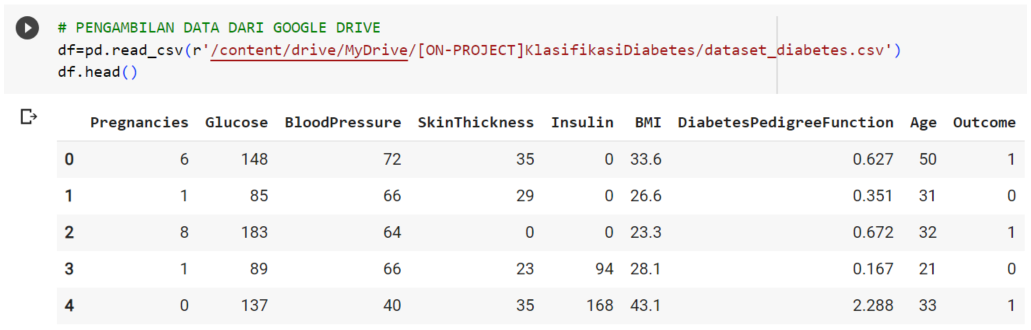


Figure 1. Research's Methodology

2.1. Data Acquisition

The data collection process in this study refers to the steps for collecting data. This process involves identifying, collecting, and processing data from various relevant sources [6], [7]. In this study, the Data Retrieval Process was carried out by retrieving data from "Pima Diabetes India," which belongs to UCI Machine Learning, from the Kaggle international dataset repository [8]. The dataset used in this study consists of 9 columns and 768 rows of data, which includes information about the diagnosis of diabetes. More detailed information about this dataset can be seen in Figure 2.2. During the Data Collection process, these data will be accessed and identified to ensure their quality and accuracy before being used in the next phase of this research.



```
# PENGAMBILAN DATA DARI GOOGLE DRIVE
df=pd.read_csv(r'/content/drive/MyDrive/[ON-PROJECT]KlasifikasiDiabetes/dataset_diabetes.csv')
df.head()
```

	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

Figure 2. Dataset Detail

2.2. Data Preprocessing

2.2.1. Data Manipulation

Data manipulation in data science refers to a series of processes of transforming and organizing data to obtain more valuable or relevant information. This process involves applying various techniques and operations to the dataset to manipulate, change, or manage data according to the needs of the analysis [9], [10]. More meaningful and valuable information can be generated from existing datasets by manipulating data properly.

2.2.2. NaN Values Removal

NaN Values Removal aims to identify and resolve empty or NaN values in the dataset. Proper handling is essential because it can affect modeling and decisions made based on data [11], [12]. The technique used in this study is to use the imputation technique.

2.2.3. Balancing Datasets

Dataset balancing refers to the amount of data on each class or target label in a dataset. Unbalanced datasets can cause bias in the model to be made. To overcome this problem, the oversampling technique was used (adding minority class data) in this study [13], [14].

2.3. Building Models

2.3.1. Random Forest Classifier

The Random Forest Classifier is a machine-learning algorithm model for classification problems. This algorithm is a form of several independent decision trees. Each decision tree is built randomly using a random training data sample [15], [16]. The Random Forest Classifier algorithm is used with the default configuration in this study to classify data on the problem under investigation.

2.3.2. AdaBoost Classifier

Adaboost (Adaptive Boosting) is a machine learning algorithm model that combines several weak decision trees into a robust model. Each decision model is built sequentially with more emphasis on misclassified data. In this way, Adaboost can gradually improve prediction accuracy and overcome complex classification problems [17],[18]. The Random Forest Classifier algorithm is used with the default configuration in this study to classify data on the issue under investigation.

2.3.3. XGBoost Classifier

XGBoost (Extreme Gradient Boosting) Classifier is an efficient machine learning algorithm model for classification tasks. This is an extension of the gradient-boosting method with multiple decision trees. XGBoost uses an ensemble learning approach, building sequential decision trees to correct previous prediction errors. This algorithm has advantages in handling extensive data, computational efficiency, and resistance to overfitting [19], [20]. The Random Forest Classifier algorithm is used with the default configuration in this study to classify data on the problem under investigation.

2.4. Model Evaluation

Model evaluation metrics are critical in measuring model performance and accuracy. Accuracy describes how well the model does the classification (Formula 1). Precision shows how accurately the model identifies positive data (Formula 2), while recall measures how well the model recognizes all positive data (Formula 3). The confusion matrix helps analyze model performance in more detail with true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. The ROC-AUC curve is used to evaluate the ability of the model to differentiate between the two classes. The closer to the value 1, the better the model performance [21].

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} * 100\% \ldots\ldots\ldots\ldots\ldots(1)$$

$$Presisi = \frac{TP}{TP+FP} * 100\% \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

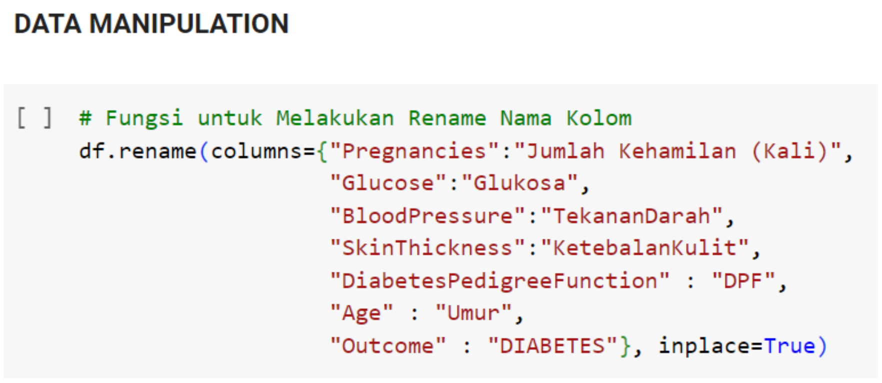
$$Recall = \frac{TP}{TP+FN} * 100\% \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

3. RESULT AND DISCUSSION

3.1. Data Preprocessing

3.1.1. Data Manipulation

Data Manipulation was carried out to translate into several column names with English text to make it easier for researchers to understand the attributes (parameters) contained in the dataset. The "PD.rename" function from the pandas library is used to perform data manipulation, including the column names before and after changing as function parameters, as shown in Figure 3.



DATA MANIPULATION

```
[ ]  # Fungsi untuk Melakukan Rename Nama Kolom
     df.rename(columns={"Pregnancies":"Jumlah Kehamilan (Kali)",
                        "Glucose":"Glukosa",
                        "BloodPressure":"TekananDarah",
                        "SkinThickness":"KetebalanKulit",
                        "DiabetesPedigreeFunction" : "DPF",
                        "Age" : "Umur",
                        "Outcome" : "DIABETES"}, inplace=True)
```

Figure 3. Data Manipulation Steps

3.1.2. NaN Values Removal

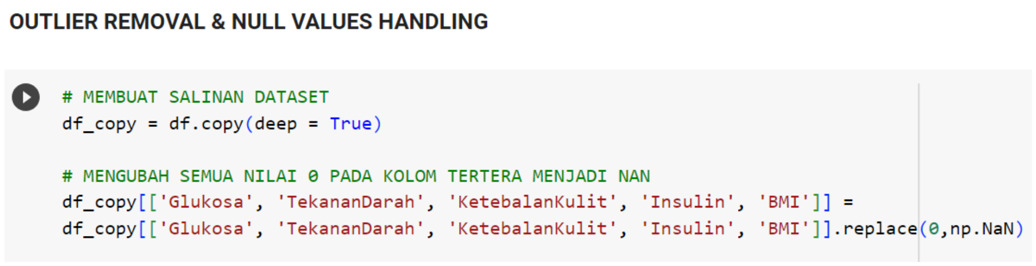
The NaN Values Removal stage begins with detecting whether there are data anomalies in the dataset used. Checking for anomaly data is done by using the "describe" function from pandas to see a statistical description of the dataset, where the results of this function can be seen in Table 1.

Tabel 1. Dataset Descriptive Statistic

	Glukosa	Tekanan Darah	Ketebalan Kulit	Insulin	BMI
Count	768	768	768.	768	768.
Mean	120.8945	69.1054	20.5364	79.7994	31.9925
Std	31.9726	19.3558	15.9522	115.2440	7.8841
Min	0.0000	0.0000	0.0000	0.0000	0.0000
25%	99.0000	62.0000	0.0000	0.0000	27.3000
50%	117.0000	72.0000	23.0000	30.5000	32.0000

75%	140.2500	82.0000	32.0000	127.2500	36.6000
Max	199.000	122.000	99.000	846.000	67.1000

It can be seen in the table above that there are data anomalies in the minimum values for the Glucose to BMI column (the glucose to BMI values cannot touch 0). Therefore, data with a value of 0 must be converted into NaN using the "replace" function from pandas, as shown in Figure 4.



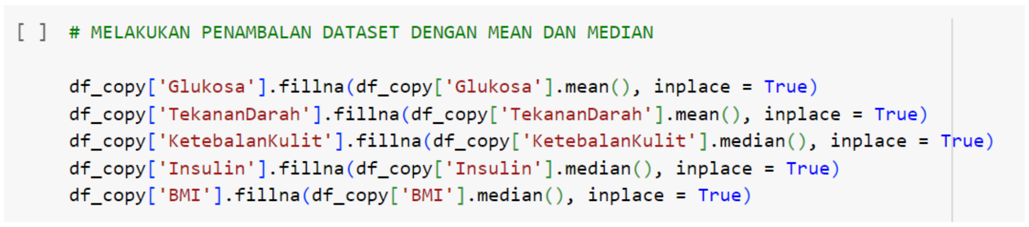
OUTLIER REMOVAL & NULL VALUES HANDLING

```
# MEMBUAT SALINAN DATASET
df_copy = df.copy(deep = True)

# MENGUBAH SEMUA NILAI 0 PADA KOLOM TERTERA MENJADI NAN
df_copy[['Glukosa', 'TekananDarah', 'KetebalanKulit', 'Insulin', 'BMI']] =
df_copy[['Glukosa', 'TekananDarah', 'KetebalanKulit', 'Insulin', 'BMI']].replace(0,np.NaN)
```

Figure 4. Replacing 0 With NaN

After all 0 data has become NaN, the last step is to input the data using the Mean (Glucose and Blood Pressure Columns) and Median (Skin Thickness, Insulin, and BMI Columns) with the pandas “fill” function, as shown in Figure 5.



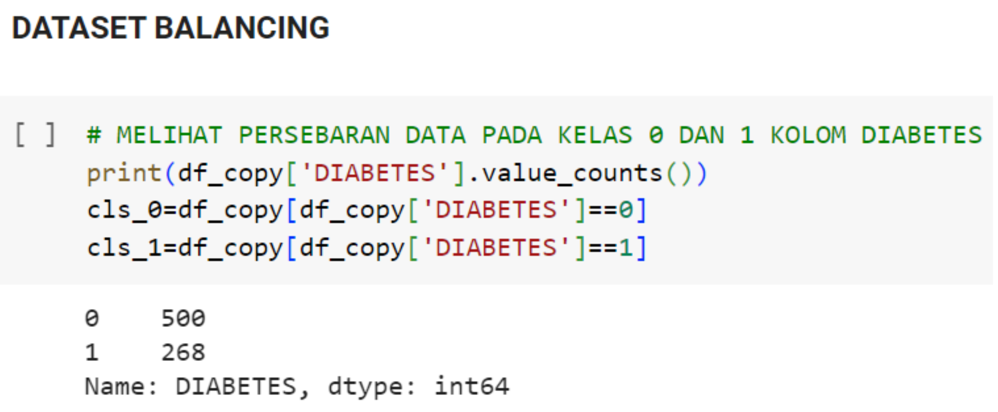
```
[ ] # MELAKUKAN PENAMBALAN DATASET DENGAN MEAN DAN MEDIAN

    df_copy['Glukosa'].fillna(df_copy['Glukosa'].mean(), inplace = True)
    df_copy['TekananDarah'].fillna(df_copy['TekananDarah'].mean(), inplace = True)
    df_copy['KetebalanKulit'].fillna(df_copy['KetebalanKulit'].median(), inplace = True)
    df_copy['Insulin'].fillna(df_copy['Insulin'].median(), inplace = True)
    df_copy['BMI'].fillna(df_copy['BMI'].median(), inplace = True)
```

Figure 5. Fill NaN Values

3.1.3. Dataset Balancing

Dataset Balancing is carried out in two stages: checking the data distribution in each class (Class 0 and 1) and then balancing the dataset. The first stage uses "value_counts," shown in Figure 6.



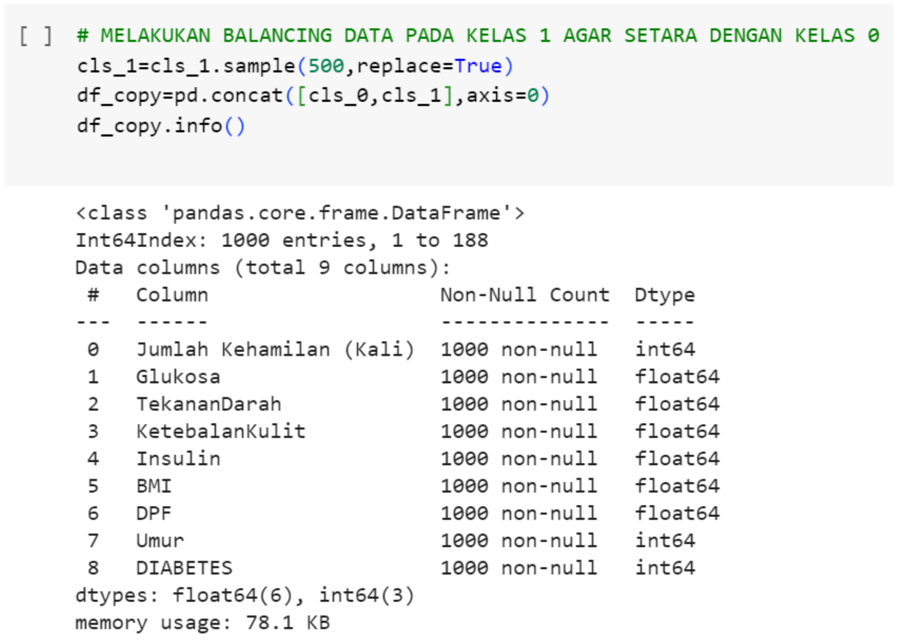
DATASET BALANCING

```
[ ] # MELIHAT PERSEBARAN DATA PADA KELAS 0 DAN 1 KOLOM DIABETES
    print(df_copy['DIABETES'].value_counts())
    cls_0=df_copy[df_copy['DIABETES']==0]
    cls_1=df_copy[df_copy['DIABETES']==1]

    0    500
    1    268
    Name: DIABETES, dtype: int64
```

Figure 6. Class Distribution Target Variable

It can be concluded from Figure 3.4 that there is more data on people without diabetes (class 0) than data on people with diabetes (class 1). Therefore, the dataset must be balanced to prevent bias in the modeling. Balancing the dataset is carried out using the "sample" function in class 1, and the results of the sampling model will be combined (concatenated) with class 0 so that the distribution of the dataset will be 500 data for class 0 and 500 data for class 1. Details of the stages can be seen in Figure 7.



```
[ ] # MELAKUKAN BALANCING DATA PADA KELAS 1 AGAR SETARA DENGAN KELAS 0
    cls_1=cls_1.sample(500,replace=True)
    df_copy=pd.concat([cls_0,cls_1],axis=0)
    df_copy.info()
```

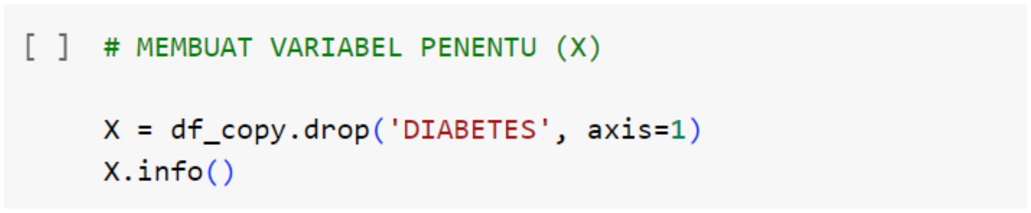
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 1 to 188
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Jumlah Kehamilan (Kali)  1000 non-null   int64
 1   Glukosa                  1000 non-null   float64
 2   TekananDarah             1000 non-null   float64
 3   KetebalanKulit           1000 non-null   float64
 4   Insulin                  1000 non-null   float64
 5   BMI                      1000 non-null   float64
 6   DPF                      1000 non-null   float64
 7   Umur                     1000 non-null   int64
 8   DIABETES                 1000 non-null   int64
dtypes: float64(6), int64(3)
memory usage: 78.1 KB
```

Figure 7. Dataset Balancing

3.2. Model Building

3.2.1. Data Partitioning

Before creating a classification algorithm model, the first step must be to divide the dataset into two to determine the independent variable (X) and the dependent/target variable (Y). Details of the distribution of X and Y variables can be seen in Figure 8.

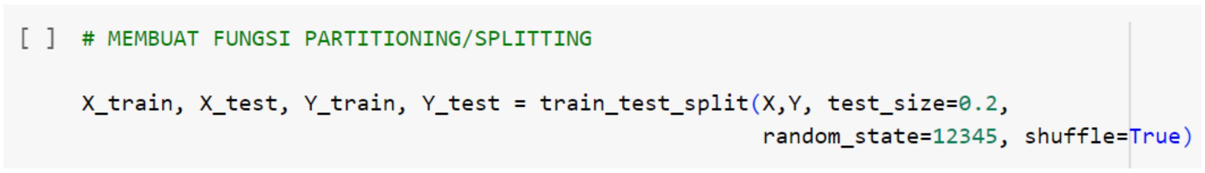


```
[ ] # MEMBUAT VARIABEL PENENTU (X)

    X = df_copy.drop('DIABETES', axis=1)
    X.info()
```

Figure 8. Distribution of Dataset X dan Y

After knowing the columns (variables) that become X and Y, the next step is to create a Training and Testing dataset for each variable. To make it, we use the "train_test_split" function from the sci-kit learn library with the configuration "test_size = 0.2" or 20% of the total dataset will be test data while 80% will be training data, "random_state = 12345" and "Shuffle = True" to allow function randomizes the dataset. After executing this function, the dataset will have X_train and X_test and Y_train and Y_test for modeling. Details can be seen in Figure 9.



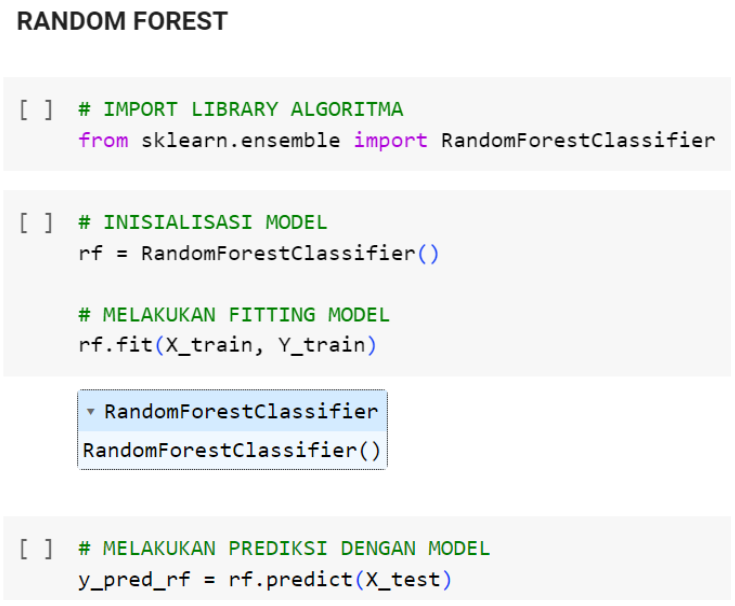
```
[ ]  # MEMBUAT FUNGSI PARTITIONING/SPLITTING

     X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2,
                                                          random_state=12345, shuffle=True)
```

Figure 9. Data Splitting

3.2.2. Random Forest Classifier

This study created a classification model using the Random Forest (RFC) algorithm using the default configuration without any modifications. The RFC model-building process begins by importing the model library from the "sklearn. ensemble" module. Next, the RFC model is initialized, and the fitting or model training process is carried out using the available training data. After the training process is complete, the model is tested or tested using the prepared testing dataset. Details and detailed steps for creating the RFC model can be found in Figure 10.



RANDOM FOREST

```
[ ]  # IMPORT LIBRARY ALGORITMA
     from sklearn.ensemble import RandomForestClassifier
```

```
[ ]  # INISIALISASI MODEL
     rf = RandomForestClassifier()

     # MELAKUKAN FITTING MODEL
     rf.fit(X_train, Y_train)
```

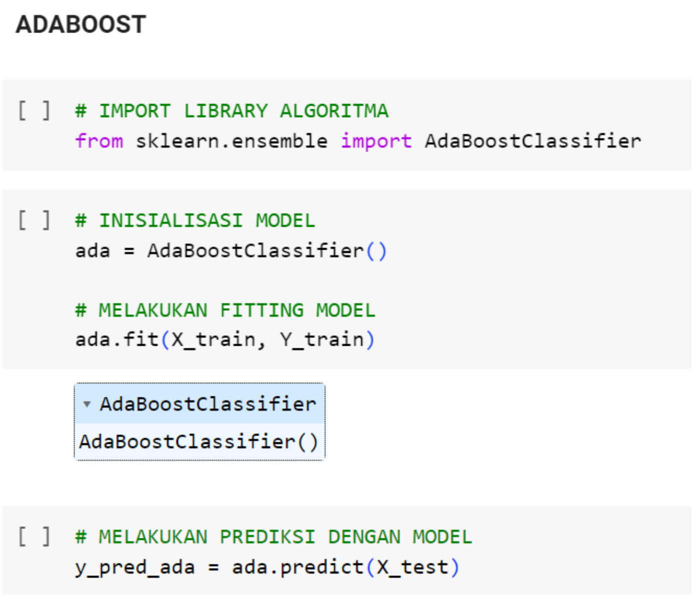
```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
[ ]  # MELAKUKAN PREDIKSI DENGAN MODEL
     y_pred_rf = rf.predict(X_test)
```

Figure 10. Model Building Random Forest

3.2.3. AdaBoost Classifier

This study created a classification model using the AdaBoost (Ada) algorithm using the default configuration without any modifications. The AdaBoost model-building process begins by importing the model library from the "sklearn.ensemble" module. Next, the AdaBoost model is initialized, and the fitting or model training process is carried out using the available training data. After the training process is complete, the model is tested or tested using the prepared testing dataset. Details and detailed steps of modeling AdaBoost can be found in Figure 11.



ADABOOST

```
[ ] # IMPORT LIBRARY ALGORITMA
    from sklearn.ensemble import AdaBoostClassifier
```

```
[ ] # INISIALISASI MODEL
    ada = AdaBoostClassifier()

    # MELAKUKAN FITTING MODEL
    ada.fit(X_train, Y_train)
```

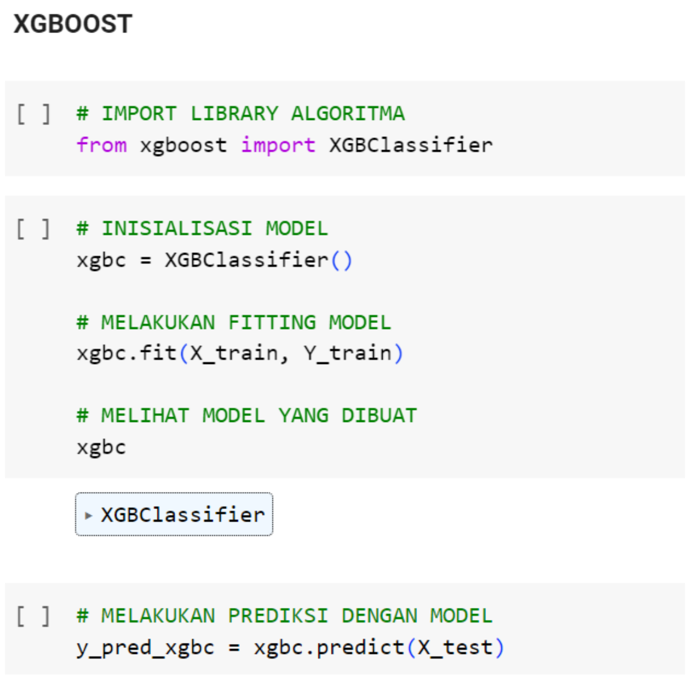
```
▾ AdaBoostClassifier
AdaBoostClassifier()
```

```
[ ] # MELAKUKAN PREDIKSI DENGAN MODEL
    y_pred_ada = ada.predict(X_test)
```

Figure 11. Model Building AdaBoost

3.2.4. XGBoost Classifier

This study created a classification model using the XGBoost (XGBC) algorithm using the default configuration without any modifications. The XGBC model-building process begins by importing the model library from the "sklearn.ensemble" module. Next, the XGBC model is initialized, and the fitting or model training process is carried out using the available training data. After the training process is complete, the model is tested or tested using the prepared testing dataset. Details and detailed steps for creating the XGBC model can be found in Figure 12.



XGBOOST

```
[ ] # IMPORT LIBRARY ALGORITMA
    from xgboost import XGBClassifier
```

```
[ ] # INISIALISASI MODEL
    xgbc = XGBClassifier()

    # MELAKUKAN FITTING MODEL
    xgbc.fit(X_train, Y_train)

    # MELIHAT MODEL YANG DIBUAT
    xgbc
```

```
▸ XGBClassifier
```

```
[ ] # MELAKUKAN PREDIKSI DENGAN MODEL
    y_pred_xgbc = xgbc.predict(X_test)
```

Figure 12. Model Building XGBoost

3.3. Model Evaluation

3.3.1. Classification Report

The first step of the Model Evaluation is the Classification Report, where the model's performance will be analyzed in detail to determine the accuracy, recall, and precision values of the model for the two classes in the target variable. This study will create a Classification Report to evaluate the Random Forest Classifier, AdaBoost Classifier, and XGBoost Classifier. Complete information about the results of the Classification Report can be found in Table 2.

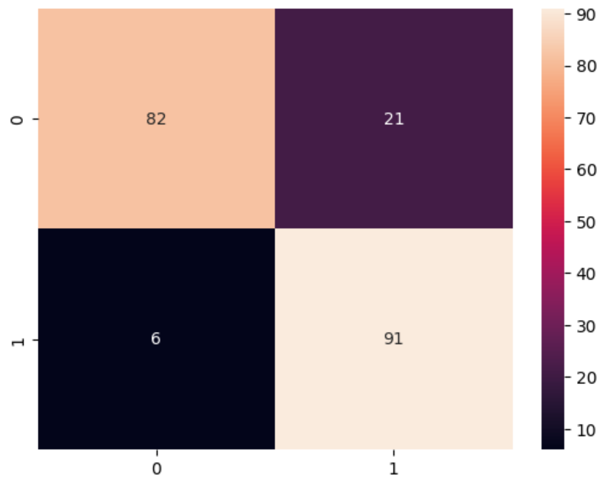
Tabel 2. Classification Report

Algoritma	**Metrik**		**Nilai**
RFC	Akurasi		**0.86**
	Presisi	Kelas 0	**0.80**
		Kelas 1	**0.94**
	Recall	Kelas 0	**0.93**
		Kelas 1	**0.81**
ADA	Akurasi		**0.82**
	Presisi	Kelas 0	**0.81**
		Kelas 1	**0.85**
	Recall	Kelas 0	**0.85**
		Kelas 1	**0.80**
XGBOOST	Akurasi		**0.85**
	Presisi	Kelas 0	**0.77**
		Kelas 1	**0.95**
	Recall	Kelas 0	**0.94**
		Kelas 1	**0.79**

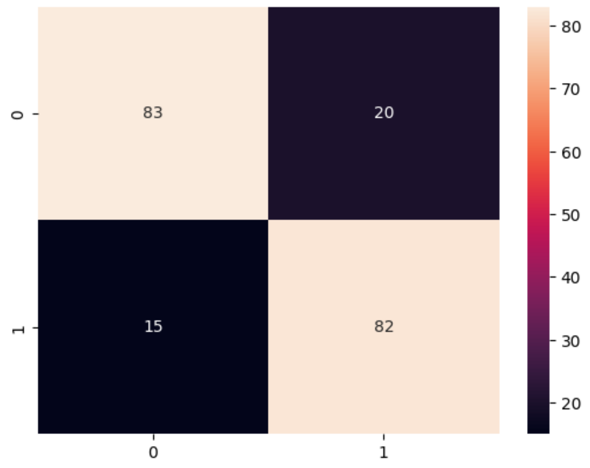
Based on the table above, the model with the highest accuracy is Random Forest with a value of 0.86, XGBoost with a value of 0.85, and AdaBoost with a value of 0.82. It can also be concluded that the three models perform well and can be used to classify diabetes.

3.3.2. Confusion Matrix

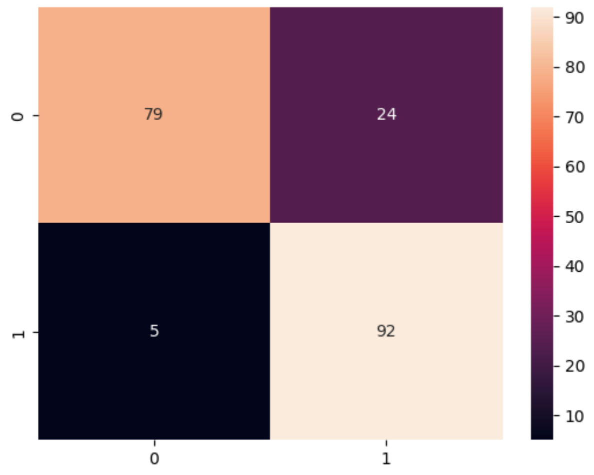
The next step is to check the Confusion Matrix visualization, where this stage is only relevant for classification models and aims to find out the amount of data that is classified correctly (True Positive/TP and True Negative/TN) and the amount of data that is incorrectly classified by the model (False Positive/FP and False Negative/FN). The results of the Confusion Matrix for Random Forest, AdaBoost, and XGBoost can be seen in Figure 13.



(a)



(b)



(c)

Figure 13.Confusion Matrix a) Random Forest, b) AdaBoost and c) XGBoost

Based on the Confusion Matrix above, it can be concluded that the model that has a precision value is the Random Forest. AdaBoost can predict both classes equally, in contrast to XGBoost, which only optimally indicates class 1.

3.3.3. ROC-AUC Curve

The ROC curve's AUC (Area Under the Curve) value reflects the model's ability to distinguish between positive and negative classes. If the AUC value is close to 1, the model has an excellent ability to distinguish categories. In contrast, if it is close to 0.5, the model has poor skills and tends to be random in differentiating classes. The results of the ROC-AUC curve in this study indicate that all models have a value close to 1. Detailed results can be seen in Figure 14.

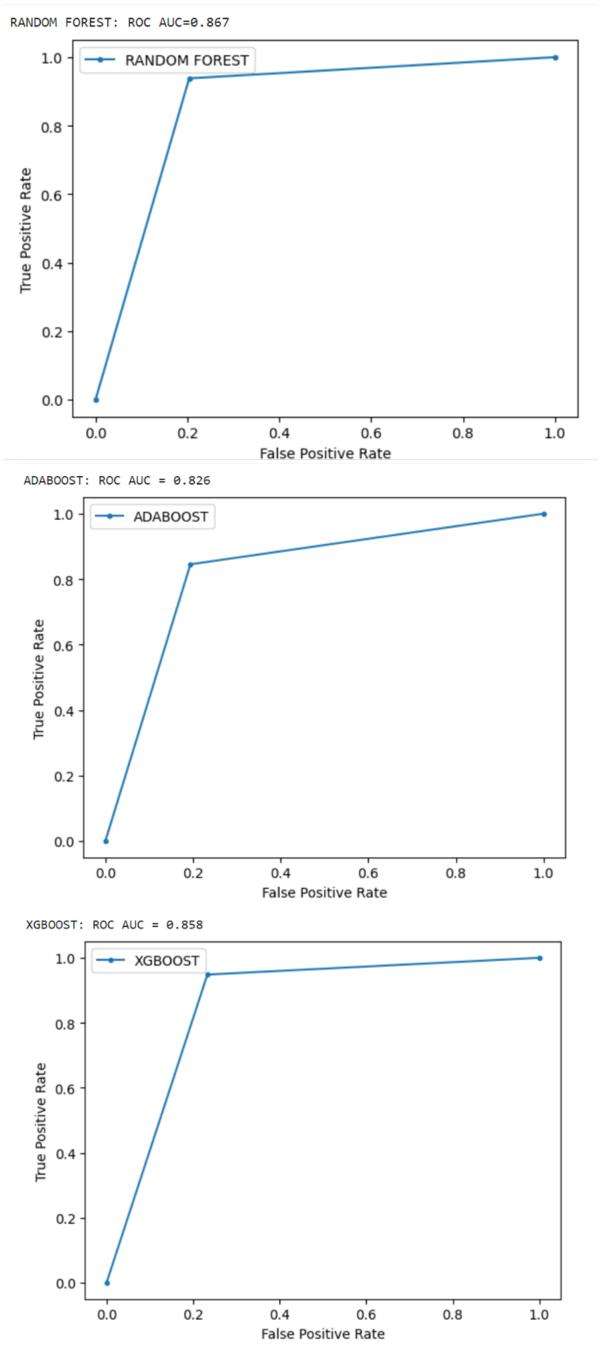


Figure 14. Hasil ROC-AUC Curve Tiap Model

3.3.4. Feature Importance

Feature Importance aims to identify variables that have the most significant influence on modeling. This step involves using the "feature_names" function in the three models developed in this study, namely the Random Forest Classifier, AdaBoost Classifier, and XGBoost Classifier. The results of this process can be found in Figure 15.

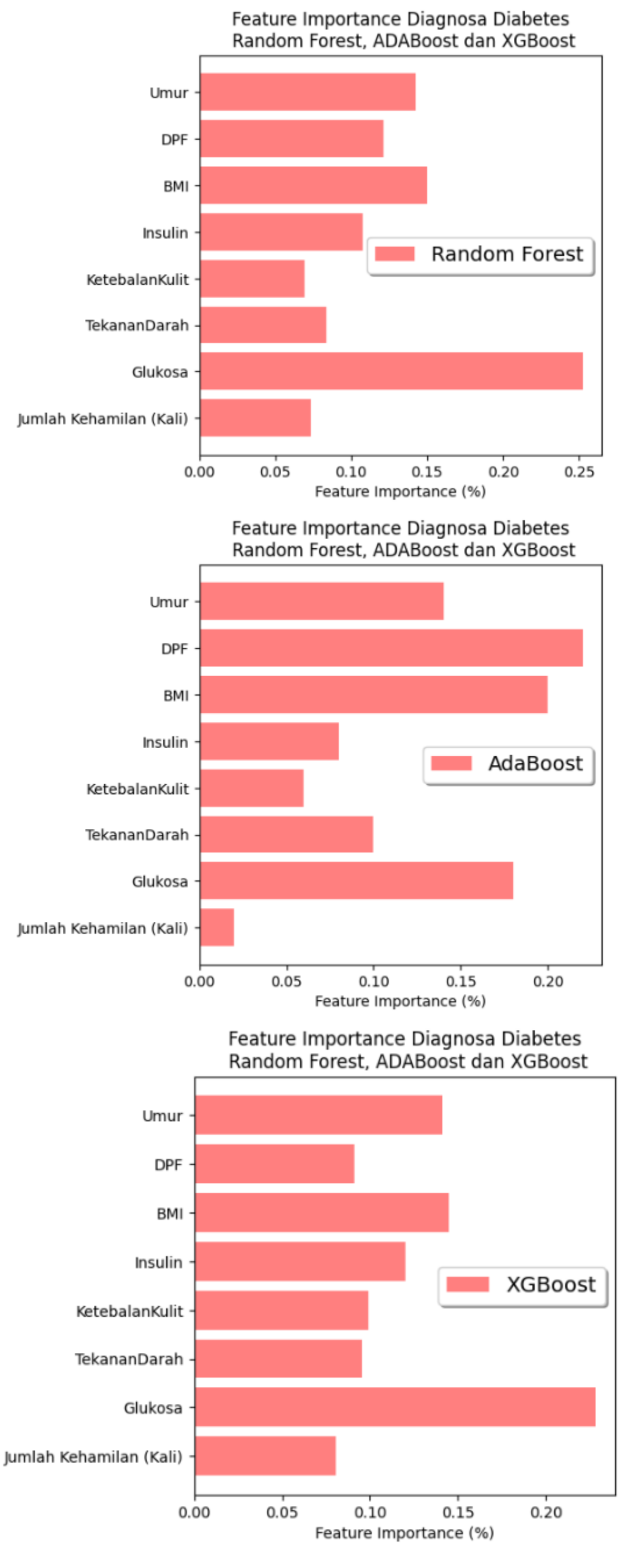


Figure 15. Feature Importance Model

Based on the visualization of the Feature Importance results that have been made above, it can be seen that the Random Forest and XGBoost algorithms have similarities to the 3 most important features, namely Glucose, BMI, and Age. As for AdaBoost, the 3 most essential features are DPF, BMI, and Glucose.

4. CONCLUSION

Diabetes is a significant public health problem and affects millions of people worldwide. This study will perform a comparative analysis of three ensemble learning algorithms (Random Forest, AdaBoost, and XGBoost) in classifying diabetes diagnoses. Based on the research that has been carried out, it is concluded that the model with the highest accuracy is Random Forest with a value of 0.86, XGBoost with a value of 0.85, and AdaBoost with a value of 0.82. It can also be concluded that the three models perform well and can be used to classify diabetes. Based on the visualization of the results of Feature Importance that has been made, it can be concluded that the Random Forest and XGBoost algorithms have in common the 3 most essential features, namely Glucose, BMI, and Age. As for AdaBoost, the 3 most critical features are DPF, BMI and Glucose.

BIBLIOGRAPHY

[1] World Health Organization (WHO), “Diabetes,” *https://www.who.int/news-room/fact-sheets/detail/diabetes*.

[2] Kementrian Kesehatan, “Kasus Diabetes Melitus,” 2020, Accessed: Jul. 27, 2023. [Online]. Available: https://www.kemkes.go.id/downloads/resources/download/pusdatin/infodatin/Infodatin%202020%20Diabetes%20Melitus.pdf

[3] I. M. Karo Karo and Hendriyana, “KLASIFIKASI PENDERITA DIABETES MENGGUNAKAN ALGORITMA MACHINE LEARNING DAN Z-SCORE,” *Jurnal Teknologi Terpadu*, vol. 8, no. 2, pp. 96–99, 2022.

[4] Noviandi, “Implementasi Algoritma Decision Tree C4.5 untuk Prediksi Penyakit Diabetes Jurnal INOHIM,” *Jurnal INOHIM*, vol. 6, no. 1, p. 1, 2018, [Online]. Available: www.kaggle.com/uciml/pima-indians-diabetes-database

[5] A. M. Argina, “Penerapan Metode Klasifikasi K-Nearest Neigbor pada Dataset Penderita Penyakit Diabetes,” *Indonesian Journal of Data and Science*, vol. 1, no. 2, pp. 29–33, 2020.

[6] A. Amalia, M. Radhi, S. H. Sinurat, D. R. H. Sitompul, and E. Indra, “PREDIKSI HARGA MOBIL MENGGUNAKAN ALGORITMA REGRESSI DENGAN HYPER-PARAMETER TUNING,” *Jurnal Sistem Informasi dan Ilmu Komputer Prima(JUSIKOM PRIMA)*, vol. 4, no. 2, pp. 28–32, Feb. 2022, doi: 10.34012/jurnalsisteminformasidanilmukomputer.v4i2.2479.

[7] A. Gehlot, B. K. Ansari, D. Arora, H. Anandaram, B. Singh, and J. L. Arias-Gonzales, “Application of Neural Network in the Prediction Models of Machine Learning Based Design,” in *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, IEEE, Jul. 2022, pp. 1–6. doi: 10.1109/ICSES55317.2022.9914184.

[8] UCI Machine Learning, “Pima Indians Diabetes Database,” *https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database*, 2022.

[9] Z. Tian, L. Cui, J. Liang, and S. Yu, “A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning,” *ACM Comput Surv*, vol. 55, no. 8, pp. 1–35, Aug. 2023, doi: 10.1145/3551636.

[10] M. Vermeulen *et al.*, “XRFast a new software package for processing MA-XRF datasets using machine learning," *J Anal At Spectrom*, vol. 37, no. 10, pp. 2130–2143, 2022, doi: 10.1039/D2JA00114D.

[11] A. Elhassan, S. M. Abu-Soud, F. Alghanim, and W. Salameh, “ILA4: Overcoming missing values in machine learning datasets – An inductive learning approach,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 4284–4295, Jul. 2022, doi: 10.1016/j.jksuci.2021.02.011.

[12] G. A. Lyngdoh, M. Zaki, N. M. A. Krishnan, and S. Das, “Prediction of concrete strengths enabled by missing data imputation and interpretable machine learning,” *Cem Concr Compos*, vol. 128, p. 104414, Apr. 2022, doi: 10.1016/j.cemconcomp.2022.104414.

[13] I. Ul Hassan, R. H. Ali, Z. Ul Abideen, T. A. Khan, and R. Kouatly, “Significance of Machine Learning for Detection of Malicious Websites on an Unbalanced Dataset,” *Digital*, vol. 2, no. 4, pp. 501–519, Oct. 2022, doi: 10.3390/digital2040027.

[14] Y. Yao, T. Sullivan, F. Yan, J. Gong, and L. Li, “Balancing data for generalizable machine learning to predict glass-forming ability of ternary alloys,” *Scr Mater*, vol. 209, p. 114366, Mar. 2022, doi: 10.1016/j.scriptamat.2021.114366.

[15] A. B. Adetunji, O. N. Akande, F. A. Ajala, O. Oyewo, Y. F. Akande, and G. Oluwadara, “House Price Prediction using Random Forest Machine Learning Technique,” *Procedia Comput Sci*, vol. 199, pp. 806–813, 2022, doi: 10.1016/j.procs.2022.01.100.

[16] S. Demir and E. K. Sahin, “Comparison of tree-based machine learning algorithms for predicting liquefaction potential using canonical correlation forest, rotation forest, and random forest based on CPT data,” *Soil Dynamics and Earthquake Engineering*, vol. 154, p. 107130, Mar. 2022, doi: 10.1016/j.soildyn.2021.107130.

[17] M. T. Ramakrishna, V. K. Venkatesan, I. Izonin, M. Havryliuk, and C. R. Bhat, “Homogeneous Adaboost Ensemble Machine Learning Algorithms with Reduced Entropy on Balanced Data,” *Entropy*, vol. 25, no. 2, p. 245, Jan. 2023, doi: 10.3390/e25020245.

[18] W. LI *et al.*, “Implementation of AdaBoost and genetic algorithm machine learning models in prediction of adsorption capacity of nanocomposite materials,” *J Mol Liq*, vol. 350, p. 118527, Mar. 2022, doi: 10.1016/j.molliq.2022.118527.

[19] H. Yan, Z. He, C. Gao, M. Xie, H. Sheng, and H. Chen, “Investment estimation of prefabricated concrete buildings based on XGBoost machine learning algorithm,” *Advanced Engineering Informatics*, vol. 54, p. 101789, Oct. 2022, doi: 10.1016/j.aei.2022.101789.

[20] Z. Li, “Extracting spatial effects from machine learning model using local interpretation method: An example of SHAP and XGBoost,” *Comput Environ Urban Syst*, vol. 96, p. 101845, Sep. 2022, doi: 10.1016/j.compenvurbsys.2022.101845.

[21] A. Wang, V. V Ramaswamy, and O. Russakovsky, “Towards Intersectionality in Machine Learning: Including More Identities, Handling Underrepresentation, and Performing Evaluation,” in *2022 ACM Conference on Fairness, Accountability, and Transparency*, New York, NY, USA: ACM, Jun. 2022, pp. 336–349. doi: 10.1145/3531146.3533101.