# Stacked Generalization - Investigating the impact on predictive performance of basic machine learning models[*]

Szymon Antonik[1,*,†], Bartosz Bąba[1,†]

[1]*Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND*

Abstract
In this work we investigate the influence of stacking different artificial intelligence algorithms on their predictive performance, using real data from the 1994 US Census database [1] to predict whether an individual's annual income exceeds $50 000.
Stacking is an ensemble algorithm, used to combine the predictions from a few machine learning models, already trained on the dataset, to achieve greater predictive accuracy. The basic idea behind stacked generalization is to leverage the strengths of different algorithms by using their predictions to train a final learner, which then produces the final prediction. In this work KNeighborsClassifier, GaussianNB and Support Vector Classifier have been used as base estimators and Logistic Regression has been used as the final estimator.

**Keywords**
artificial intelligence, stacked generalisation, stacking, census income, adult dataset, income prediction

## 1. Introduction

The introduction of machine learning is essential for achieving precise predictions in today's data-driven world [2, 3, 4]. However, relying solely on individual models may limit the attainment of optimal results. In this context, the technique of Stacked Generalization (Stacking) becomes a significant tool for improving predictive performance through the application of ensemble strategies [5, 6].

Stacking is an ensemble learning method that integrates the results of multiple base models, allowing a meta-algorithm (such as logistic regression) to make final predictions effectively [7]. In our study, we plan to apply Stacked Generalization with basic algorithms like Naive Bayes, SVM, and k-NN to examine its impact on improving prediction accuracy.

The primary goal of our article is to explore the potential of Stacked Generalization in enhancing the predictive performance of basic machine learning models. We aim not only to delve into its potential but also to assess the flexibility of this technique across different domains and applications. Our goal is to determine whether the application of Stacked Generalization can effectively increase prediction accuracy compared to individual models on the selected dataset.

[*]*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania*
[1,*] Corresponding author
[†] These author contributed equally.

sa307842@student.polsl.pl (S. Antonik); bb307846@student.polsl.pl (B. Bąba)
0009-0008-2418-1869 (S. Antonik); 0009-0009-7044-3783 (B. Bąba)

Through experimental research and result analysis, our article aims to provide practical insights into the potential benefits derived from using Stacked Generalization.

In this way, our work aims to broaden the understanding of Stacked Generalization as an effective tool in the arsenal of machine learning practitioners, significantly improving prediction quality and opening up new possibilities in data analysis.

## 2. Methodology

### 2.1. Machine Learning Classifier

A machine learning classifier is a type of algorithm that learns patterns from labeled training data in order to categorize or classify new input data into one of several predefined classes or categories. The goal of a classifier is to generalize from known examples and make predictions or decisions about new data based on its learned knowledge.

How a typical machine learning classifier works:

1. Phase of training - The classifier is provided with a set of training examples - each of them contains input data (features) and output labels (class). The classifier learns to map input features to the proper output labels by identifying relations and patterns in the training data. The classifier uses one of the learning algorithms (such as Naive Bayes, support vector machines, and k-nearest neighbors) to build a model based on the training data.
2. Phase of predicting - Once the classifier is trained, it can be used to predict the output classes of new, unseen before data. The classifier takes input features of new data sample and applies the learned model to assign it to one of the predefined classes.

### 2.2. K-Nearest Neighbors Classifier

The K-Nearest Neighbors (KNN) classifier is a simple yet effective machine learning classifier. KNN is considered 'lazy' classifier, because it doesn't explicitly learn a model during training process. It memorizes training sets and makes predictions for new instances based on their similarity to known examples instead.

1. Phase of training - Train and test subsets are picked from the dataset. KNN 'memorizes' (stores) training data.
2. Phase of predicting Number of nearest neighbors K and distance metric (e.g. Euclidean distance) need to be chosen before making predictions. For every data instance in test set, KNN calculates distance to each and every data instance in memorized set, using chosen distance metric. K nearest neighbors of the instance are chosen (samples with the lowest calculated distance). Algorithm checks which output class is the most popular among those neighbors - this class is predicted output for the specimen.

### 2.3. Naive Bayes Classifier

The Naive Bayes classifier is a probabilistic machine learning algorithm based on Bayes' theorem. Term 'naive' refers to the assumption made by the algorithm that all features in the dataset are

independent of each other given the class label. This assumption is considered 'naive' because it's often not true in real-world datasets where features can be correlated or dependent on each other. Despite its simplicity and the 'naive' assumption, Naive Bayes classifiers are widely used for classification tasks, especially in text classification and spam filtering, due to its efficiency and effectiveness.
Bayes' theorem - probability of probability of a hypothesis (class label) given the evidence (features).

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{1}$$

Where $P(y|X)$ - posterior probability of class $y$ given features $X$, $P(X|y)$ - likelihood of features $X$ given class $y$, $P(y)$ - probability of class $y$, $P(X)$ - probability of features $X$.

1. Phase of training - Train and test subsets are picked from the dataset. Naive Bayes Classifier calculates prior probabilities of each class $y$, based on the training set. $P(y) = \frac{Number \quad of \quad instances \quad of \quad class \quad y}{Total \quad number \quad of \quad instances}$ For each feature $x$ classifier calculates the likelihood $P(x|y)$, based on the training data.
2. Phase of predicting - Classifier computes the posterior probability: $posterior = \frac{likelihood \times prior}{evidence}$ , where evidence $P(\mathbf{x})$ is the total probability of observing data $\mathbf{x}$ across all possible classes

$$P(\mathbf{x}) = \sum_{y} P(\mathbf{x}|y)P(y) \tag{2}$$

Classifier selects the class $y$ with the highest posterior probability $P(y|\mathbf{x})$ as the predicted class for the new instance $\mathbf{x}$

## 2.4. Support Vector Classification

Support Vector Machines (SVMs) are powerful learning models used for classification, regression, and outlier detection tasks. The primary goal of SVMs is to find an optimal *hyperplane* that separates data points of different classes with the maximum *margin* in a high-dimensional space. In SVM, a hyperplane is a decision boundary separating data points into different classes in a feature space. In binary classification (with 2 output classes), the hyperplane is (n-1)-dimensional subspace of n-dimensional feature space.
The margin is the distance between the hyperplane and the nearest data points (support vectors) from each class. SVM goal's to maximize this margin to improve robustness and generalization.

- Linear Separability: If data points can't be linearly separated by a hyperplane, SVM uses techniques like kernel functions to map data into higher-dimensional space, where linear separation is possible. Kernel functions are crucial in SVMs - they compute the dot product between transformed feature vectors, thus avoiding the explicit computation of the transformation while still benefiting from the increased dimensionality. Commonly used kernel functions: linear, polynomial, radial basis function.

- Optimization: SVM finds optimal hyperplane by solving an optimization problem that minimizes the classification error, while maximizing the margin. For 2 output classes:

$$min_{w,b}\frac{1}{2}\left\|w\right\|^2 \tag{3}$$

  subject to:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i \in \{1, ..., n\} \tag{4}$$

  where: $w$- weight vector, $b$- bias term, $x_i$- feature vector of the $i$-th data point, $y_i$- class label of the $i$-th data point. The $w$and $b$that solve this problem determine the final classifier, $x \longrightarrow -\ sgn(w^T x_i + b)$, where $sgn(\cdot)$ is the sign function.
- Support Vectors: Support vectors are the data points that lie closest to the hyperplane and directly influence its position. These points determine the margin and are crucial for defining the decision boundary.

Support Vector Classification:

1. Phase of training - Train and test subsets are picked from the dataset. Kernel function needs to be chosen before making predictions. SVM designates the optimal hyperplane by solving the optimization problem. The Algorithm identifies the support vectors that lie on or near the margin.
2. Phase of predicting - For each new data point, SVM calculates its class based on its position relative to the learned hyperplane. The decision function outputs the predicted class.

## 2.5. Stacking Classifier

A stacking classifier, also known as stacked generalization, is an ensemble learning technique that combines multiple base classifiers to improve predictive performance. It operates by training a meta-model on the predictions made by these base classifiers.

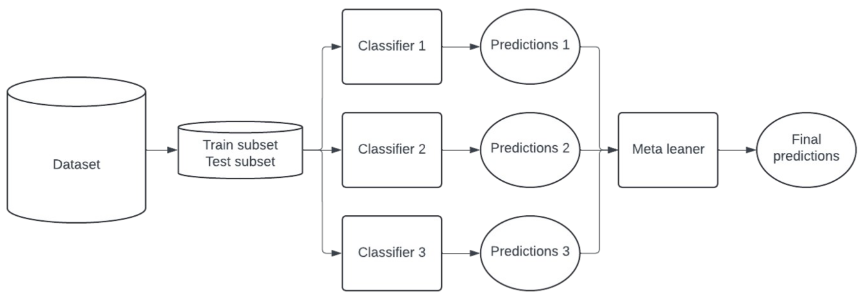


**Figure 1:** Scheme of Stacking Classifier

1. Phase of training - Main train (MTr) and main test (MTe) subsets are picked from the dataset. MTr is splitted to train and test subsets. Each of base classifiers is trained on the train subset and makes predictions. Meta learner is trained on the base classifiers predictions.

2. Phase of predicting - Each of base classifiers makes predictions based on input features of MTe. Meta learner makes final predictions based on base classifiers predictions.

**Algorithm 1:** Algorithm - Stacking Classifier

```
Data: estimators, final_estimator
Function fit(X_main_train, y_main_train) is
    Split X_main_train, y_main_train into 4 subsets: X_train, X_test, y_train, y_test
    LIST base_predictions ⟵ []
    foreach $e \in estimators$ do
        e.fit(X_train, y_train)
        pred ⟵ e.predict(X_test)
        base_predictions.append(pred)
    base_predictions ⟵ column_stack(base_predictions)
    final_estimator.fit(base_predictions)
Function predict(X_test, y_test) is
    LIST test_base_predictions ⟵ []
    foreach $e \in estimators$ do
        base_pred ⟵ e.predict(X_test)
        test_base_predictions.append(base_pred)
    X_final ⟵ column_stack(base_predictions)
    return final_estimator.predict(X_final)
```

## 3. Experiments

### 3.1. Performance metrics

In this work, the Stacking Classifier has been used to combine 3 other classifiers: Naive Bayes Classifier, KNN Classifier and Support Vector Classifier. There are 4 possible outcomes of the predictions made by the classifier during binary classification: True Positive (TP) - occurs when classifier predicts a positive (true) outcome, and the actual outcome is indeed positive. True Negative (TN) - occurs when classifier predicts a negative (false) outcome, and the actual outcome is indeed negative. False Positive (FP) - occurs when classifier predicts a positive (true) outcome, and the actual outcome is negative. False Negative (FN) - occurs when classifier predicts a negative (false) outcome, and the actual outcome is positive. The following metrics have been used to measure the performance of classifiers: Accuracy is the proportion of correctly classified instances (both true positives and true negatives) among all instances.

$$Accuracy = \frac{TP + TN}{n} \tag{5}$$

Precision is the proportion of true positive predictions (correctly predicted positive instances) among all instances predicted as positive.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall is the proportion of true positive predictions (correctly predicted positive instances) among all actual positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

F1 Score is the harmonic mean of precision and recall, providing a balance between the two metrics.

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

Confusion matrix is a table that summarizes the performance of a classification model by listing the counts of true positives, false positives, true negatives, and false negatives.

**Algorithm 2:** Algorithm - Label Encoder

```
Function LabelEncoding(df: DataFrame) is
    foreach column IN df.columns do
        LIST unique_values ← UNIQUE_VALUES(df[column])
        DICT col_map ← {}
        foreach unique_value IN unique_values do
            col_map[unique_value] ← INDEX(unique_values, unique_value)
        foreach row IN df do
            df[colum][row] = col_map[df[column][row]]
    return df
```

## 3.2. Preparing dataset

- Encoding - Encoding categorical data is a crucial step in preparing dataset for machine learning models. Label Encoding assigns a unique integer to each category.
- Normalization (scaling) is a process, which brings all features to a similar scale or range. Scaling is performed to ensure that certain features do not dominate solely, because they have larger magnitudes. Scaling may also improve computing speed and performance. One of commonly used scaling method is min-max normalization.
- Splitting a dataset into training and testing subsets is a fundamental practice in machine learning to evaluate the performance of predictive models. This process helps ensure that the model generalizes well to new, unseen data. One of common splitting strategy is holdout method - dividing original dataset into two subsets: training subset and testing subset, typically in ratio 70:30 or 80:20.

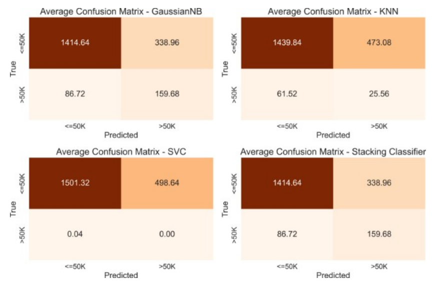
## 3.3. Dataset

*Adult Census Income* dataset contains data from 1994, extracted by Barry Becker from US Census database. Records, that did not meet the following dependency have been removed from the original database: ((AAGE>16)&&(AGI>100)&&(AFNLWGT>1)&&(HRSWK>0)), where AAGE - represents a person's age, AGI - Adjusted Gross Income, AFNLWGHT - Final weight - number of people represented by an entry, HRSWK - number of hours worked per week. As a result, dataset containing 32651 rows and 15 columns has been created. The source of the data is http:\\kaggle.com [1].

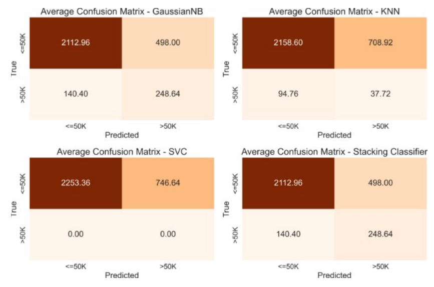
## 3.4. Testing setup

In this section, we present the results of four conducted tests. The tests were performed on both unscaled and scaled data. For each dataset, we applied two splits: 80/20 and 70/30 for training and testing sets. Each test will evaluate the impact of scaling and different split ratios on model accuracy.

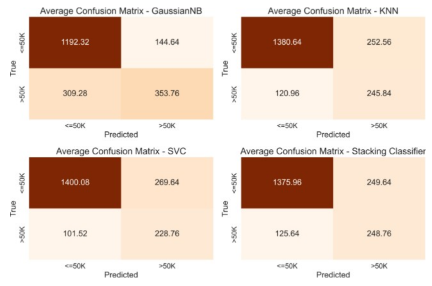
Performance of classifiers for unscaled data with an 80/20 split (Figure 3-2a) varied across different metrics: GaussianNB achieved an average accuracy of 78.72%, with a precision of 64.83%, recall of 32.04%, and an F1 score of 42.80%. KNN showed an average accuracy of 73.27%,



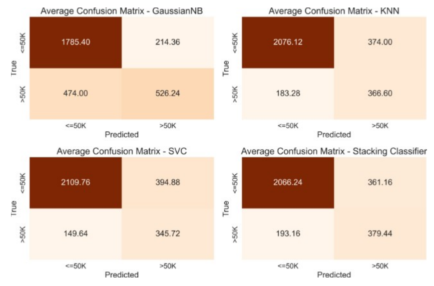
(a) Confusion matrix for data split 80/20 without scaling



(b) Confusion matrix for data split 70/30 without scaling



(c) Confusion matrix for data split 80/20 after scaling



(d) Confusion matrix for data split 70/30 after scaling

**Figure 2:** Comparison of confusion matrices for different data splits and scaling methods

with a precision of 29.47%, recall of 5.13%, and an F1 score of 8.70%. SVC attained an average accuracy of 75.07%, however, precision, recall, and F1 score were 0% due to no false negatives and true positives. The Stacking Classifier obtained identical results to GaussianNB.

Performance of classifiers for unscaled data with an 70/30 split (Figure 4-2b) varied across different metrics: GaussianNB and Stacking Classifier demonstrate comparable performance, achieving an average accuracy of 78.72%, precision of 64.01%, recall of 33.30%, and F1 score of 43.74%. Their confusion matrices exhibit similar patterns with moderate true positive and false negative values. However, KNN performs less effectively with an average accuracy of 73.21%, precision of 28.48%, recall of 5.05%, and F1 score of 8.55%, indicating higher misclassification rates. On the other hand, SVC, despite achieving an accuracy of 75.11%, fails to provide meaningful precision, recall, or F1 score due to classifying all instances as true negative or false positive.

In this analysis, we examine the performance of classifiers on normalized data with an 80/20 split (Figure 5-2c).KNN, SVC, and Stacking Classifier achieved better results than GaussianNB, with higher values of accuracy, precision, and F1 score. KNN showed the highest average accuracy at 81.32%, while SVC and Stacking Classifier attained accuracies of 81.44% and 81.24%

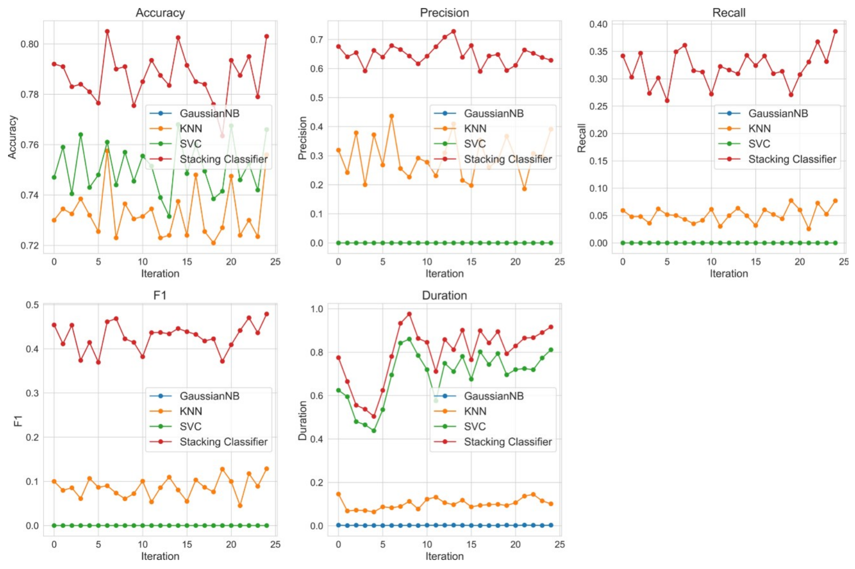


**Figure 3:** Results for data split 80/20 without scaling

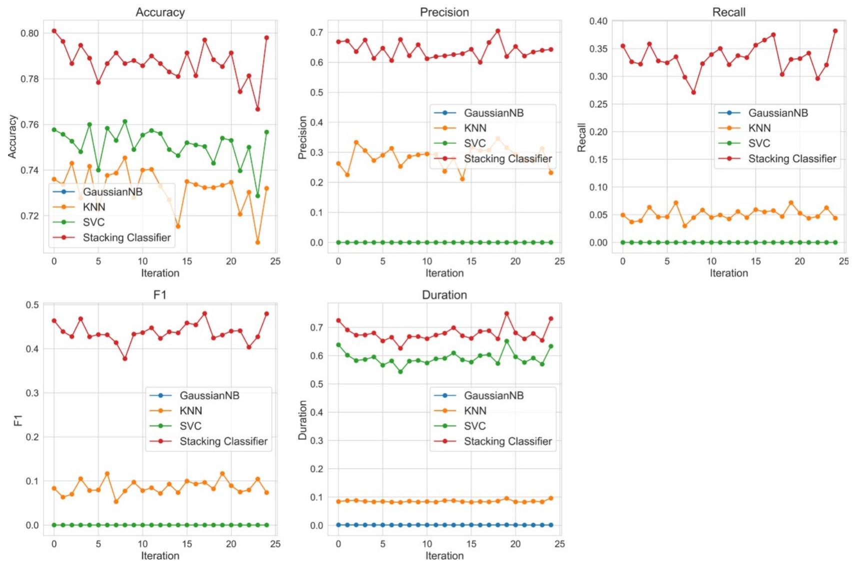


**Figure 4:** Results for data splitted 70/30 without scaling

respectively. Despite differences in performance, all three classifiers demonstrated a tendency to better recognize positive instances compared to GaussianNB, particularly evident in the reduced number of false negatives.

The evaluation of classifiers on normalized data with an 70/30 split (Figure 6-2d) indicates that KNN, SVC, and Stacking Classifier outperform GaussianNB in terms of accuracy, precision, recall, and F1 score. KNN exhibits the highest average accuracy of 81.42%, with SVC and Stacking Classifier closely following, achieving accuracies of 81.85% and 81.52% respectively.

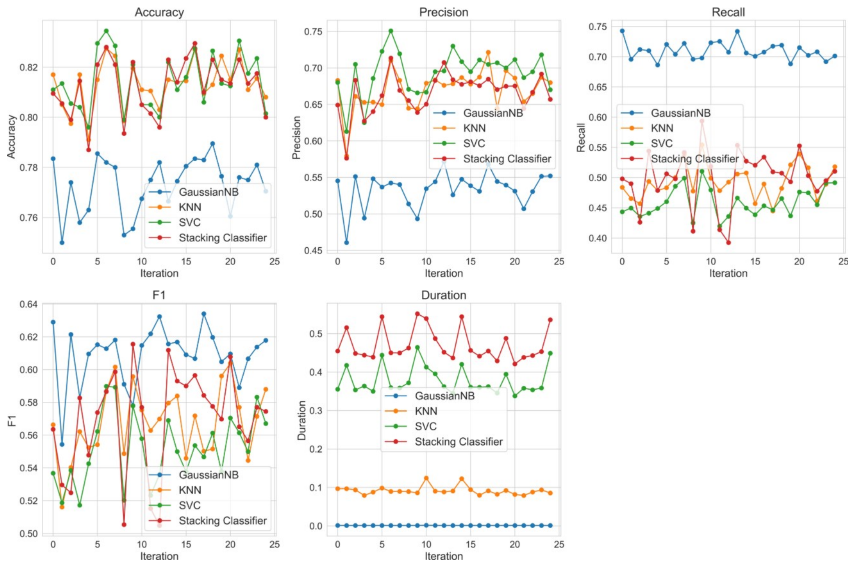


**Figure 5:** Results for data splitted 80/20 after scaling

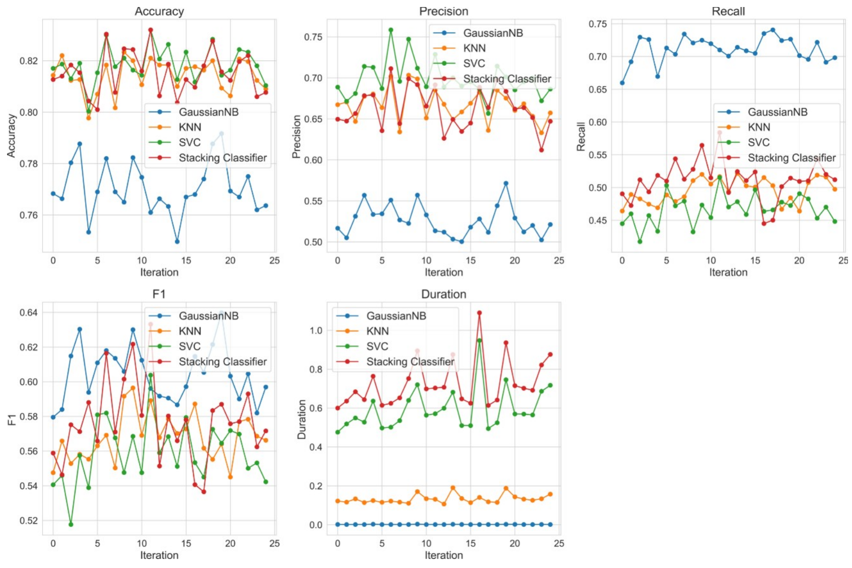


**Figure 6:** Results for data splitted 70/30 after scaling

## 4. Conclusion

Usage of the Stacking Classifier results in increase, especially for not scaled data, in the F1 metric score - this indicates that the classification model has a good balance between precision and recall, meaning it performs well in correctly identifying positive instances while minimizing false positives and false negatives. Higher F1 score signifies a more reliable and effective model, especially in situations where there is an imbalance in the classes or when both types of errors (false positives and false negatives) are critical to consider.
To summarize, using the Stacking Classifier may increase predictive performance and reliability, at the expense of computation time.

## References

[1] B. B. Ronny Kohavi, Adult census income, https://www.kaggle.com/datasets/uciml/adult-census-income (1994).

[2] M. Woźniak, M. Wieczorek, J. Siłka, Bilstm deep neural network model for imbalanced medical data of iot systems, Future Generation Computer Systems 141 (2023) 489–499.

[3] G. P. Kanna, S. J. Kumar, Y. Kumar, A. Changela, M. Woźniak, J. Shafi, M. F. Ijaz, Advanced deep learning techniques for early disease prediction in cauliflower plants, Scientific Reports 13 (2023) 18475.

[4] D. Połap, G. Srivastava, A. Jaszcz, Energy consumption prediction model for smart homes via decentralized federated learning with lstm, IEEE Transactions on Consumer Electronics (2023).

[5] C. Zhang, H. Hu, J. Ji, K. Liu, X. Xia, M. S. Nazir, T. Peng, An evolutionary stacked generalization model based on deep learning and improved grasshopper optimization algorithm for predicting the remaining useful life of pemfc, Applied Energy 330 (2023) 120333.

[6] W. Fu, Y. Fu, B. Li, H. Zhang, X. Zhang, J. Liu, A compound framework incorporating improved outlier detection and correction, vmd, weight-based stacked generalization with enhanced desma for multi-step short-term wind speed forecasting, Applied Energy 348 (2023) 121587.

[7] Y. Xie, W. Sun, M. Ren, S. Chen, Z. Huang, X. Pan, Stacking ensemble learning models for daily runoff prediction using 1d and 2d cnns, Expert Systems with Applications 217 (2023) 119469.