Stacking Ensembles-based Approach for Pattern Diagnosis of Tabular Data

JunKoo Lee
Artificial Intelligence Major
Sungkyunkwan University
Suwon, Republic of Korea
dlwnsrn0727@g.skku.edu

Tai-Myoung Chung[*]
College of Computing and Informatics
Sungkyunkwan University
Suwon, Republic of Korea
tmchung@skku.edu

***Abstract*—The introduction of an automation system in facilities has enabled the continuous flow of data streams, facilitating easier and more accessible data collection. Moreover, advancements in machine learning and deep learning techniques have made it possible to detect abnormalities in data streams, such as electrical energy and equipment diagnosis data, almost in real-time. Early detection of such anomalies allows for proactive maintenance and cost savings, ensuring stable operation of the entire system. Conventionally, the State of Charge (SOC) and State of Health (SOH) have been used as major criteria for assessing facility abnormality in electrical energy data. However, these measures are not suitable for real-time diagnostics due to their design for post-discharge battery analysis. In this paper, we propose the utilization of ensemble stacking technique from the ensemble models to enable accurate and real-time health condition monitoring of electrical equipment using open data sources. Additionally, we perform a comparative analysis among various machine learning and deep learning techniques and optimize the model using a grid search algorithm to achieve a high-performing, robust, and generalizable health condition monitoring system.**

***Index Terms*—Machine Learning, Ensemble Learning, Pattern Diagnosis, Stacking, Tabular Data**

I. Introduction

Generally, State of Charge (SOC) and State of Health (SOH) are used as criteria to measure facility abnormalities in electrical equipment data. SOC and SOH are used to assess the remaining lifespan and current performance status of batteries. The measurement of SOC and SOH is often achieved by repeated measurement of battery charge and discharge cycles [1, 2]. However, due to the time lag between charging and discharging, measurement noise is smoothed using the cubic smoothing spline method to mitigate any faults introduced. Similarly, SOC and SOH are not suitable for real-time analysis of the state of electrical equipment in the context of building an automated monitoring system.

In this paper, various learning techniques are employed to effectively analyze abnormalities in electrical equipment systems using open datasets. We present comparative performance analysis in terms of accuracy and F1 score to classify the current state of the system into ”normal,” ”caution,” or ”warning” categories. Additionally, we provide the training and inference times for each model to demonstrate their real-time inference capabilities in the continuous flow of data streams. Finally, we optimize and evaluate the performance of the models using grid search to identify the best-performing model.

II. Related work

The increasing availability of tabular data in power energy facilities, numerous machine learning and deep learning methods have been proposed to achieve better generalization performance. Previous research has identified several inefficient aspects when analyzing tabular data using machine learning and deep learning models.

In the case of machine learning-based analysis, tabular data typically consists of a large number of columns (features), which can lead to the curse of dimensionality. The curse of dimensionality refers to the degradation of model performance and the potential for overfitting as the number of features increases. To overcome this, various methods such as feature selection or dimensionality reduction techniques have been proposed. Additionally, tabular data often exhibits a static nature, with a lack of temporal continuity between data points. In such cases, it can be challenging for models to capture the dynamic changes in the data, requiring the use of models specifically designed for handling time series data or considering temporal features.

For analysis using deep learning models, tabular data follows a structured format with columns and rows, whereas deep learning models are predominantly designed to process continuous data. As a result, these models may not effectively handle such structured information. Therefore, various methods have been proposed to preprocess or transform tabular data before directly inputting it into deep learning models. Techniques such as pretrained models or RNN [11], Transformer [9, 10], Word Embedding, and tokenization are used to format the data. Subsequently, models like RNN [11], CNN [9, 10], and others are adopted for classification tasks. Complex models like deep neural networks require significant computing resources and time for training. To address these issues, a deep learning model based on Graph Convolutional Networks (GCN) that models multiple relationships simultaneously was used for multi-task classification, demonstrating promising performance [3]. However, deep learning faces challenges in interpreting learned representations or providing rationales for

predictions. In the case of tabular data, where interpretation is important, deep learning has limitations in explaining the process. Therefore, in this paper, we propose a methodology that combines machine learning and deep learning techniques by utilizing the stacking technique of ensemble models to process tabular data effectively.

The stacking technique in ensemble models combines multiple different base models to create a powerful model, and there are several advantages of using ensemble stacking in tabular data:

Performance improvement: Combined the predictions of individual models, ensemble stacking can result in more accurate predictions.

Reduction of overfitting: Combined models with different biases, ensemble stacking can produce more generalized predictions and reduce overfitting.

Learning from diverse features: Each model in the ensemble can learn different aspects of the data, allowing the ensemble model to leverage the diverse features present in tabular data.

III. METHOD

In this section, we extensively discuss the methodology for performing pattern classification on tabular data from power facility energy patterns and fault analysis sensor data, utilizing various models. After visualizing and preprocessing the data, we conducted data training and evaluation using XGBoost + Deep, Deep, XGBoost, SVM, Random Forest, KNN, and Naive Bayes models. Furthermore, we compared and analyzed the reasons for selecting these models and their performance. Based on this analysis, we selected the most suitable model for the tabular data of power facility energy patterns and fault analysis sensor data. Additionally, we compared and analyzed the application of the Grid Search algorithm to determine the impact of its usage.

A. Exploratory Data Analysis

we describe dataset, preprocessing steps, and the baseline machine learning models for the experiment.

a) DATASET: Power Facility Energy Pattern and Fault Analysis Sensor' data from AI Hub, which we collected, represented SOH values by inputting power quality measurements directly measured for 10 types of facilities. Generally, the SOH is determined by comparing the battery's current capacity to its initial capacity when it was new. The Equation 1 is commonly used to calculate SOH as a percentage:

$$SOH = \frac{\alpha}{\beta} * 100 \tag{1}$$

- α : The Current Capacity
- β : The Initial Capacity

the Current Capacity refers to the actual capacity of the battery at the time of measurement, while the Initial Capacity represents the capacity of the battery when it was brand new. Dataset segments raw power quality data collected in 1-minute increments to power factor average, voltage harmonic. SOH diagnostic labeling (normal, caution, and warning types) are present, which states the condition of the system. The labeling criteria is expressed as follows:

- normal : Power factor 80% or above, voltage harmonics 3% or below
- caution : Power factor 60-80%, voltage harmonics 3-5%
- warning : Power factor 60% or below, voltage harmonics 5% or above

b) Preprocessing: We extracted training dataset 1,692,057 SOH diagnostic labeling and Test dataset 233,091 SOH diagnostic labeling using labeled pump general motor data, and the analysis showed class balance as shown in Figure 1.

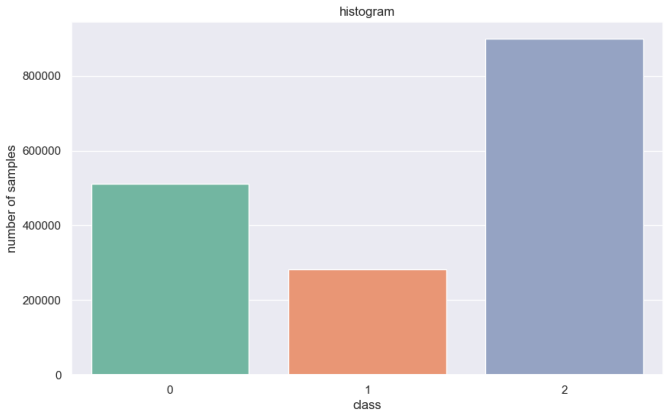

Fig. 1. **Class balance for each label.** Training dataset includes 510,679 cases, 282,501 cases, and 898,877 cases for normal, caution, and warning, respectively.

To classify the patterns among the three states, we utilized three features from SOH diagnostic dataset where, 'E' stands for voltage harmonic average, 'F' stands for current harmonic average, and the 'G' stands for power factor average.

After removing the outlier using IQR, oversampling was conducted to solve the overfitting problem due to class imbalance. We oversampled the other classes to match the largest label (2). The result after the oversampling, the class distributions are shown in Figure 2.

TABLE I
MEAN AND VARIANCE FOR EACH FEATURES.

Features	Mean Value	Var Value
voltage harmonics	4.42	7.80
Current harmonics	12.89	338.01
Power factor	0.51	0.14

We also analyzed the mean and variance values for each features which is shown in Table 1. Also, the distribution of each features are described in Figure3. As we can see, the mean and variance are largely different among the selected features, we scaled the entire dataset into the range of values from 0 to 1 using min max scaling to preserve the distribution

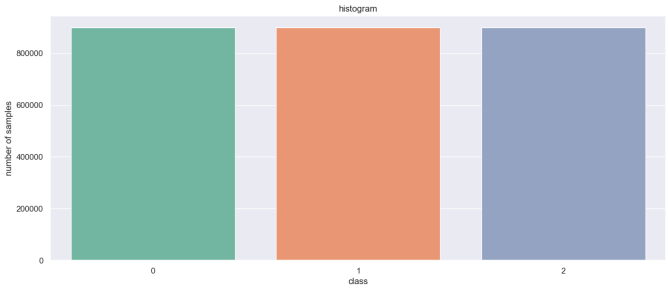

Fig. 2. Class balance for each label after oversampling. Training dataset after oversampling includes 898,877 cases for all three different classes.

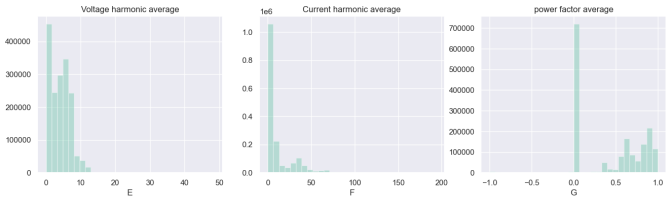

Fig. 3. **Distribution of the features** a) stands for E (voltage harmonic average), b) stands for F (current harmonic average), and c) stands for G (power factor average).

of the original data. For the experiment, we utilized the smaller portion of the training dataset for training and validation considering the size of the collected dataset compared to the capacity of the model.

B. Machine Learning Models

In this section, we describe dataset, preprocessing steps, and the baseline machine learning models for the experiment.

XGBoost: XGBoost is known for its high prediction performance and accuracy. It can handle complex relationships and capture non-linear patterns in the data effectively. XGBoost provides feature importance scores, allowing for feature selection and dimensionality reduction. It includes regularization techniques to control overfitting and offers fast execution speed, making it suitable for large-scale datasets. The algorithm 1 represents the functioning principle of the XGBoost model[6].

RandomForest Classifier: Random Forest Classifier is robust to overfitting due to its ensemble of decision trees. It can handle high-dimensional data with complex relationships and noisy features. Random Forest Classifier can handle both numerical and categorical features without requiring feature scaling or one-hot encoding. It also provides feature importance scores, aiding in feature selection and understanding the data. The algorithm 2 represents the functioning principle of the RandomForest Classifier model.[11].

K-Nearist-Neighbors Classifier: KNN is a simple and intuitive algorithm that is easy to understand and implement. It doesn't make strong assumptions about the underlying data distribution, making it suitable for various types of data. KNN can handle multi-class classification without modifications

Algorithm 1 XGBoost Algorithm

Input: Training data: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$
Output: XGBoost model

Initialize the model: $F_0(x) = 0$
Set the learning rate: η
Set the number of iterations: T
Boosting Iterations:
for $t = 1$ to T **do**
  Compute the negative gradient: $r_{it} = -\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}$ for $i = 1, 2, ..., n$
  Fit a base learner to the negative gradients: $h_t(x) = \text{BaseLearner}(x, r_{1t}, r_{2t}, ..., r_{nt})$
  Update the model: $F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$
end for
return XGBoost model: $F_T(x)$

Algorithm 2 RandomForest Classifier Algorithm

Input: Training data: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$
Output: RandomForest model

Set the number of trees: N
Set the number of features: M
Set the maximum depth of each tree: D
Set the size of the random feature subset: K
Updated node embeddings $H^{(L)}$: Node embeddings after passing through L layers.
Compute Graph Laplacian:
Calculate the adjacency matrix A and degree matrix D of the graph.
Compute the graph Laplacian matrix $L = D - A$.
Compute Graph Laplacian:
for $i = 1$ to N **do**
  Update node embeddings: $H^{(l)} = \text{ReLU}(D^{-0.5} \cdot A \cdot D^{-0.5} \cdot H^{(l-1)} \cdot W^{(l)})$
  Here, $H^{(l-1)}$ is the node embeddings from the previous layer, and $W^{(l)}$ is the weight matrix for layer l.
  Randomly select K features from the total M features: $\text{SelectedFeatures} = \text{RandomSubset}(\text{Features}, K)$
  Build a decision tree using the subset and selected features: $\text{Tree} = \text{BuildDecisionTree}(\text{Subset}, \text{SelectedFeatures}, D)$
end for
return the updated node embeddings $H^{(L)}$ from the last layer.

and can effectively capture non-linear relationships between features. KNN algorithm identifies the k nearest neighbors from the training dataset based on the equation2 and assigns the class label that is most prevalent among the neighbors. The value of k is a hyperparameter that needs to be specified before applying the algorithm[4].

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2}$$

SVM: SVM is effective in high-dimensional spaces and performs well with complex datasets. It can handle a large

TABLE II

THE COMPARATIVE PERFORMANCE OVER DIFFERENT MACHINE LEARNING TECHNIQUES. THE BEST-PERFORMING MODEL IS MARKED AS BOLDFACE. WE TRAINED EACH MACHINE LEARNING MODELS ON THE SAME DATASET AND TESTED ON THE SAME TEST DATASET FOR THE FAIR COMPARISON.

Model	Accuracy	F1 Score	Running Training	Runtime Prediction
XGBoost + Deep	**0.85**	**0.85**	23.49	0.31
XGBoost	0.84	0.84	1.30	0.08
Random Forest	0.84	0.84	1.27	0.08
KNN	0.81	0.81	0.02	3.50
SVC	0.79	0.80	7.16	120.30
Naive Bayes	0.71	0.72	0.01	0.03
Deep	0.80	0.80	6.90	118.46

number of features and effectively capture complex relationships between them. SVM is robust against overfitting and provides the ability to control the balance between maximizing the margin and minimizing training error. It can also handle non-linear decision boundaries through the use of kernel functions. SVMs attempt to find a hyperplane that maximizes the margin(Equation 3), which is the distance between the hyperplane and the closest data points of each class[5].

$$margin = \frac{2\left|w^T x + b\right|}{\|W\|} \tag{3}$$

This margin maximization approach allows SVMs to be effective in handling complex decision boundaries and dealing with data that may not be linearly separable.

Naive Bayes: Naive Bayes is computationally efficient and performs well with large datasets and high-dimensional features. It is robust to irrelevant features and can handle both categorical and continuous features. Naive Bayes provides interpretable probabilistic predictions, allowing for easy understanding of the model's confidence in its predictions. The Naive Bayes algorithm calculates the probability of a particular data instance belonging to a specific class by estimating the conditional probabilities of the features given each class(Equation 3)[12].

$$P(C|X) = \frac{P(C|X)p(C)}{p(X)} \tag{4}$$

C. Deep Learning Model

MLPClassifier: MLPClassifier is composed of multiple layers, making it suitable for modeling nonlinear relationships. The addition of the ReLU function introduces nonlinearity, allowing the model to effectively learn and classify complex patterns in tabular data. Furthermore, the MLPClassifier can be well combined with other engineering techniques, enhancing its ability to handle complex patterns and perform classification tasks on tabular data.

D. Ensemble Model

XGBoost + Deep: The stacking technique was utilized as an ensemble model. Stacking combines the strengths of various machine learning and deep learning models by allowing each model to learn different aspects of the data and combining their predictions. This approach enables more accurate predictions. Moreover, by combining models with different biases, stacking can achieve better generalization performance. It also enables the models to learn and utilize the diverse features present in tabular data.

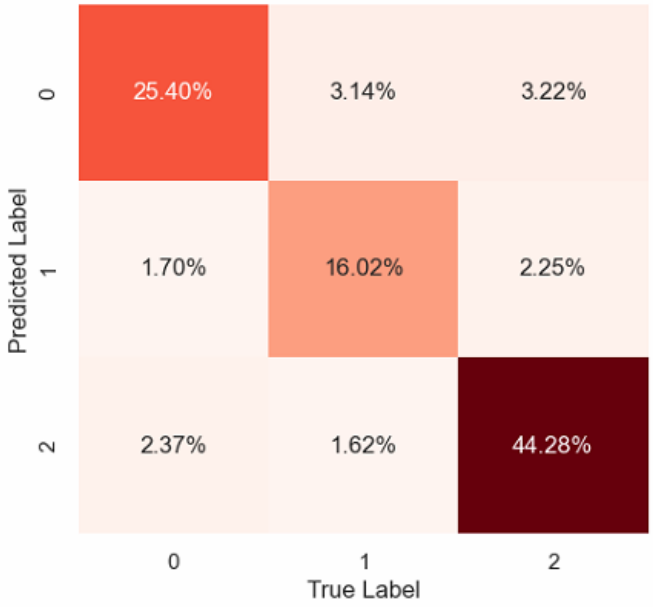

Fig. 4. **Confusion matrix of the optimized XGBoost model**.

E. Grid Search

Grid Search: For finding the optimal set of hyperparameters, we utilized the grid search algorithm by setting the limited range of parameter values. After comparing the performance of machine learning models, we selected the best-performing model for optimizing the parameters.

IV. EXPERIMENTAL RESULTS

A. performance of Machine Learning Models

One of the main challenge of the task is to determine whether the machine learning model can be utilized for the real-time inference over the continous flow of the data stream. Thus, we first trained the selected machine learning

TABLE III
THE TRAINING RESULT OF GRID SEARCH. WE ANALYZED THE ACCURACY OF THE BEST PARAMETERS AND TEST DATA IN THE RESULTS OF FOUR CROSS-VALIDATIONS WITH SPLIT TRAINING.

Model	Mean Test Score	Ranking	Learning Rate	Number of Estimators	Max depth	Alpha
model1	**0.86532**	**1**	0.003	130	4	0.001
model2	0.86522	2	0.002	150	3	0.001
model3	0.86520	3	0.002	120	4	0.002
model4	0.86520	3	0.002	100	3	0.003
model5	0.86520	3	0.002	110	3	0.004
model6	0.86516	6	0.002	120	4	0.003
model7	0.86516	6	0.004	100	4	0.003
model8	0.86512	8	0.001	110	4	0.004
model9	0.86511	9	0.001	120	4	0.001
model10	0.86511	9	0.002	120	3	0.001

models over the preprocessed traning dataset and evaluated the performance on the test dataset. We also measured the training time with the prediction time in seconds. The Table 2 shows the experimental result of different machine learning models. As we can see, the XGBoost outperformed over the other machine learning techniques where the accuracy and f1 score reported 0.85 and 0.85, respectively. Also, in terms of prediction time it only took 0.11 seconds to infer over 200,000 samples of test datasets, which indicates the capability of the model to be used for real-time inference.

B. Result of the Grid Search

As the the superior performance of the XGBoost+Deep model, we optimized the parameters: number of estimators (increased by 10 from 120 to 160), learning rate (increased by 0.001 from 0.001 to 0.005), max depth (2, 3, 4), and alpha (increased by 0.001 from 0.001 to 0.005). The parameter number of estimators represents the number of trees, learning rate determines the amount of weight update at each step, max depth indicates the maximum depth of trees, and alpha serves as the L2 penalty parameter controlling the weights. We trained 192 models with different parameter values for number of estimators, learning rate, max depth, and alpha. The best-performing model achieved an accuracy of 86.53%, showing an improvement of 1.08% in accuracy. We selected the top 10 models based on mean test score among the 192 models and presented the parameters and results in Table 3. The optimized XGBoost+Deep model's confusion matrix is shown in Figure 4. The confusion matrix presents the prediction of values labeled as 'Normal,' 'Caution,' and 'Warning' with an accuracy of approximately 85.7%.

V. DISCUSSION

We analyzed a dataset consisting of 510,679 records labeled as 'normal'. The average power factor is 74.43%, and the average voltage harmonic is 4.25%. However, the construction guidelines suggest that for labeling the normal state of power quality data, the power factor should be above 80% and the voltage harmonic should be below 3%. The criteria for measuring the normal state of electrical equipment data need to be supplemented, as shown in Table 1.

We applied various algorithms to analyze patterns in tabular data from electrical diagnostic equipment and explored them to predict the state of the equipment. From the experimental results, we found several interesting findings. The distribution of each class is highly distinguishable. In Table 2, the ensemble model XGBoost+Deep achieved the highest performance with an accuracy of 85.45%. Generally, boosting-based approaches are vulnerable to overfitting, but XGBoost+Deep performed well even on unseen data. This can be attributed to the ensemble learning characteristics, normalization techniques, outlier handling, and the use of simple models within the ensemble, which enhance generalization and mitigate the risk of overfitting. Additionally, the performance of KNN was surprisingly high, demonstrating fast training time of 0.02 seconds and excellent accuracy. This indicates clear decision boundaries for each class.

However, the dataset lacks richness in information. Despite performing hyperparameter tuning for key parameters that could strongly impact the results, the differences were minimal. There are two main possibilities when hyperparameter tuning is not effective. One is that the tuning process focused on a limited set of parameters such as learning rate or the number of estimators, and the other is that the dataset itself may not have significant features that could greatly improve performance, indicating that useful predictive information may have already been extracted without tuning. Considering the exploration of key hyperparameters and the effectiveness of KNN, the latter possibility is more likely.

VI. CONCLUSION

The pattern analysis of tabular data from electrical diagnostic equipment was effectively performed using the XGBoost+Deep model, which was attributed to the stacking technique of ensemble models. The ensemble model diversified the data characteristics analyzed by each model, helping to address issues such as the curse of dimensionality, overfitting, and handling structural information when analyzing the structure of tabular data with machine learning and deep learning models.

Furthermore, the stacking technique of ensemble models is expected to be effective not only for tabular data but also for

other data formats. Therefore, research should be conducted in this direction, exploring the combination of models other than XGBoost+Deep to improve performance.

ACKNOWLEDGMENT

This research was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00421, AI Graduate School Support Program (Sungkyunkwan University)) and supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-00990, Platform Development and Proof of High Trust & Low Latency Processing for Heterogeneous·Atypical·Large Scaled Data in 5G-IoT Environment).

REFERENCES

[1] Chang W-Y. The state of charge estimating methods for battery: a review. ISRN Appl. Math. 2013; vol:1–7.

[2] H. Tian, P. Qin, K. Li, Z. Zhao, A review of the state of health for lithium-ion batteries: research status and suggestions, J Clean Prod 261 (2020), 120813.

[3] L. Du, F. Gao, X. Chen, R. Jia, J. Wang, J. Zhang, and D. Zhang, TabularNet: A Neural Network Architecture for Understanding Semantic Structures of Tabular Data, In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 322-331, August 2021.

[4] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. American Statistician, 46, 1992. ISSN 15372731. doi: 10.1080/00031305.1992.10475879.

[5] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. 2018. ThunderSVM: A Fast SVM Library on GPUs and CPUs. Journal of Machine Learning Research 19 (2018), 797–801.

[6] T. Chen and C. Guestrin, XGBoost: A scalable tree boosting system, in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[7] Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In Proceedings of the AAAI Conference on Artificial Intelligence.

[8] Haoyu Dong, Shijie Liu, Zhouyu Fu, Shi Han, and Dongmei Zhang. 2019. Semantic Structure Extraction for Spreadsheet Tables with a Multi-task Learning Architecture. In Workshop on Advances in Neural Information Processing Systems.

[9] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. 2019. Tablesense: Spreadsheet table detection with convolutional neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence. 69–76.

[10] Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In Proceedings of the AAAI Conference on Artificial Intelligence.

[11] Biau, G.Ã.Š. Analysis of a random forests model. J. Mach. Learn. Res. 2012, 13, 1063–1095.

[12] I. Rish An empirical study of the naive Bayes classifier IJCAI 2001 Work Empir. Methods Artif. Intell. (2001), pp. 41-46

[13] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," 2021, arXiv:2106.03253.

[14] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, andT. Goldstein, "SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training," 2021, arXiv:2106.01342.

[15] A. Vaswani et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.