


Explore An Acoustic Extinguisher Fire Dataset With Machine Learning Models

Yiqian Zhang

School of Computer Science (Department)
University of Nottingham Ningbo China (School)
Ningbo, China
scxyz6@nottingham.edu.cn 

Abstract—Nowadays, sound wave fire extinguishing arouses an increasing research interest. Existing methods for predicting flame status are dedicated to using rule-based methods and ensemble-based models. In this paper, I innovatively propose a Holdout-based manual grid search method and a stratified k-fold cross-validation-based technique applying the GridSearchCV class to train and evaluate ML models, including k-Nearest-Neighbors (k-NN), Decision Tree (DT), Random Forest (RF) and variants. Following Google Cloud’s Data Science Road Map, the project goes through data preprocessing, exploratory data analysis, and a modelling procedure. The experiments illustrate that the implemented k-NN model outperforms other implemented models both on the two model evaluation approaches, and the gap in performance between the state-of-the-art method is as low as 0.13%. The distance between flames and the output of the sound wave, and the airflow produced by the sound wave, are confirmed as the most significant factors in the flame states.

Index Terms—Sound wave flame, grid search, Holdout Method, k-fold cross-validation, Machine Learning

I. INTRODUCTION

Fire is a kind of serious natural or man-made disaster, which can destroy buildings, damage property and even threaten personal safety without proper fire extinguishers. Water, powder and carbon extinguishers are the most common types applied in real life [1]. Recently, researchers have been researching sound wave extinguishers to discover a more effective fire extinguishing approach [2]. It uses sound waves to influence the behaviour of the flame and fire-related medium. To explore the impact of various model evaluation approaches and what media can impact and interfere fire extinguishing effectively, this study utilises a dataset “Acoustic_Extinguisher_Fire_Dataset” [2], [3], which contains experimental data obtained from experiments of three various liquid fuels and LPG fuel under a sound wave extinguishing system in a specially created room [4] to determine whether the current combination of features can successfully extinguish the fire. To be specific, the fuel container moved forward from 10 cm to 190 cm with 10 cm by each moving step. There were 54 low-frequency values to complete experiments under each distance value and flame size. These features include five fuel container sizes representative of flame size plus two LPG fuel modes, fuel types, distance from the fire to the output of sound wave, decibel levels measured by a decibel meter, air flow generated by sound waves and frequencies of computers. Besides, the

Feature	Value Range	Description	Unit	Data Type
SIZE	{1, 2, 3, 4, 5, 6, 7}	Fuel container size representative of flame size. 1: 7cm for liquid fuels; 2: 12cm for liquid fuels; 3: 14cm for liquid fuels; 4: 16cm for liquid fuels; 5: 20cm for liquid fuels; 6: Half throttle setting for LPG fuel; 7: Full throttle setting for LPG fuel.	cm	numerical
FUEL	{'gasoline', 'thinner', 'kerosene', 'lpg'}	Fuel type (first three values are liquid fuels).	/	categorical
DISTANCE	[10, 190]	Distance (from flame to collimator output).	cm	numerical
DESIBEL	[72, 113]	Sound pressure level.	dB	numerical
AIRFLOW	[0, 17]	Air flow generated by sound waves.	m/s	numerical
FREQUENCY	[1, 75]	Low frequency range.	Hz	numerical
STATUS	{0, 1}	Flame extinguished state. 0: non-extinction state; 1: extinction state.	/	categorical

Fig. 1. Description of fields in the dataset. (Reorganised through markdown [2], [3].)

dataset uses a binary label ‘STATUS’ to denote the state of the fire. With 17,442 records, the dataset has the ability to deploy different machine learning models for exploring classification tasks.

The structure of this paper is as follows. Section II reviews related work on the dataset. In Section III, I propose my methodology. Section IV shows the detailed experimental results. Section V lists discussions, and Section VI concludes the paper.

II. LITERATURE REVIEW

This section reviews analyses and machine learning methods from research papers using the same dataset and lists all experimental results of them in Fig. 2.

A. Analysis Methods

Koklu and Taspinar (2021) [2] analysed the distribution of different features using line charts and bar charts. They also gained confusion matrices and some metrics, including accuracy, precision, F1-score and so on, to evaluate trained ML models. Taspinar, Koklu, and Altin (2021) [3] worked further on classification tasks, by deploying box plots, scatter plots, confusion matrices and several model metrics like sensitivity to complete analysis.

B. Neural Network Models

As powerful and more popular tools, artificial neural networks (ANNs) and deep neural networks (DNNs) are more

Model	Hyperparameters	Accuracy
ANN	n_features_in=6, hidden_units=100, output_units=2 activation_function=ReLU, optimizer=Adam, iterations=200 learning_rate=0.0001	96.03%
DNN	n_features_in=6, hidden_units=2(layers)×50(neurons) output_units=2, activation_function=ReLU, optimizer=Adam iterations=200, learning_rate=0.0001, dropout_rate=0.2	94.88%
kNN	n_neighbors=5 distance_metric=Euclidean distance $D(a,b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$	92.62%
DT	max_depth=5	97.28%
RF	n_estimators=10	96.58%
Stacking	[refer to ANN, k-NN and RF]	97.06%
ANFIS	n_layers=5	94.50%
CN2 Rule	n_rules=736	99.91%

Fig. 2. Performance comparison in the related work. (Organised through markdown.)

common in clustering and classification tasks in real applications [5]. Their main characteristic is independent layers, having arbitrary numbers of nodes, enabling neural networks to receive several input features, go through hidden layers whose units can be more than those of the input layer and output layer, and pass calculated results to the output layer.

Koklu and Taspinar [2] implemented an ANN model with 6 input units, 100 hidden units and 2 output units. They also built a DNN architecture using similar parameters to the ANN one, with a difference of 2 layers having 50 units respectively in the hidden layers.

C. k-Nearest-Neighbors, Decision Tree and Random Forest

As a non-parametric classification method, k-Nearest-Neighbors (k-NN) utilises majority voting among a neighbourhood containing k neighbours [6]. Also, as a non-parametric model, Decision Tree (DT) applies splitting, stopping, and pruning during building the model, which includes several nodes and branches [7]. Compared with DT, Random Forest (RF) models are built up with a collection of tree-structured classifiers [8]. In the input, each tree would vote for the most popular class.

Koklu and Taspinar (2021), as well as Taspinar, Koklu, and Altin (2021), created a k-NN model based on 5 nearest neighbours and Euclidean distance. They also built a 5-depth DT and an RF with a capacity of 10 trees to predict classification tasks.

D. Stacking Method

Stacking is an effective approach to combining the performance of multiple ML models. It considers predictions of previous levels' ML models as input for use on the next level [9]. In Koklu's work [2], the stacking model integrated an ANN, k-NN and RF model.

E. ANFIS

Through a hybrid learning procedure, the Adaptive-Network Based Fuzzy Inference Systems (ANFIS) forms an input-output mapping based on fuzzy if-then rules and input-output

data pairs [10]. For Taspinar, Koklu, and Altin (2021), they created a five-layer ANFIS structure for predictions.

F. CN2 Rule

Similar to ANFIS using rules, the CN2 algorithm aims to find the optimal rule set by considering the statistical difference ratio and removing rules that are constrained below a certain threshold [3]. Based on the fire dataset, Taspinar, Koklu, and Altin performed classification with 736 generated rules.

III. METHODOLOGY

In this section, the data preprocessing approaches are first introduced. Then, the details of EDA are described. After that, a detailed ML route map is provided. The whole methodology complies with Google Cloud's Data Science Road Map [11].

A. Data Preprocessing

To obtain the dataset, I utilise `pd.read_excel()` to get the original dataframe. Then, `describe`, `info`, `head`, `tail`, `skew`, and `kurt` are methods used to outline the dataset. For better data representations, visualisation and ML performance, I applied `df.duplicated()`, interquartile range (IQR), `df.isna().any(axis = 1)`, and `df.rename()` to figure out duplicates, outliers and bad data, missing values, and column renaming issues. To be specific, IQR is used to identify outliers where data points are located outside $1.5 \times \text{IQR}$, which can be visualised using boxplots. After identifying and handling the data, `df.to_csv()` stores the cleaned version of the dataset.

B. Exploratory Data Analysis

In order to find the most meaningful features for predicting the flame status, I leverage histplots, boxplots, Pearson correlation, heatmap, and Implots to visualise the distribution of features and relationships among them. The advantages of my work are integrating many helpful diagrams that some of which appear in the referenced research papers [2], [3].

By encapsulating each plot in a function, six features are traversed in a for-loop to plot histograms using `sns.histplot()` from the Seaborn library. The histograms [12] provide an estimate of where values are concentrated. In addition to the histograms, boxplots [12] can display IQR efficiently. Boxplots use a rectangle box and a straight line across the middle of the box to represent the lower quartile (25%), median (50%) and upper quartile (75%). Two whiskers extend from the box to the lower and upper extremes. Data displayed as circles are the outliers located outside the two extremes.

Next, I adopt `df.corr()` and `stats.pearsonr()` with `sns.heatmap()` to measure the linear dependencies between each pair of six features [13]. For diving deeper into correlation, Implots [14] offer the combination of a scatterplot and linear regression. According to the value range of correlation coefficients [2], linear correlation and p-value can be described in the form of Table I.

Algorithm 1 Dataset Splitting Using Two Route Maps**Input:** mode: Data splitting mode**Output:** data and labels

```

1: if mode == 'train-test' then
2:   Call train_test_split method to get the training data
   and labels, as well as the test data and labels.
3:   return data and labels for training and testing
4: else if mode == 'cross-validation' then
5:   Define a StratifiedKfold class.
6:   Initialise an array to store training data and testing data
   for each fold.
7:   for  $i$ , ( $train\_index$ ,  $test\_index$ ) in
    $enumerate(skf.split(X\_scaled, y))$  do
8:     Use  $train\_index$  and  $test\_index$  to get the train-
   ing data, training labels, test data and test labels.
9:     Construct and store the current fold's data in the
   array.
10:  end for
11:  return array
12: end if

```

Algorithm 2 The Route Map Using Holdout Method And Manual Grid Search**Input:** data: A dictionary containing hyperparameters of the trained model.**Input:** X_train , y_train , X_test , y_test : The training data and labels, as well as the test data and labels.**Output:** *train_result*: Results containing the best model, parameters and score.

```

1: Get UUID and grid content from data.
2: Start timing.
3: Initialise variables for storing the best model, parameters
   and score.
4:  $param\_grid$  = Get the hyperparameter grid.
5: if  $uuid == 'knn-knc'$  then
6:   Get all possible values of each hyperparameter from
    $param\_grid$ .
7:   for each combination of hyperparameter configura-
   tions do
8:     Get values of hyperparameters in the current com-
   bination.
9:     Define the KNeighborsClassifier.
10:    Fit the model.
11:    Call model.predict method.
12:    Compare to get the best model, score and param-
   eters.
13:  end for
14: else if other models then
15:   ▷ Other models perform similar operations
16: end if
17: Display overall time.
18: Store training results in train_result
19: return train_result

```

TABLE I
DESCRIPTION OF CORRELATION COEFFICIENT AND P-VALUE

Value Range	Definition
pearson-coef = 1	Perfect positive linear correlation.
pearson-coef = 0	No linear correlation. Two variables likely do not affect each other.
pearson-coef = -1	Perfect negative linear correlation.
pearson-coef $\in [0.01, 0.29]$	A low level of correlation.
pearson-coef $\in [0.30, 0.70]$	A moderate correlation.
pearson-coef $\in [0.71, 0.99]$	A high level of correlation.
p-value < 0.001	Strong evidence that the correlation is significant.
p-value < 0.05	Moderate evidence that the correlation is significant.
p-value < 0.1	Weak evidence that the correlation is significant.
p-value > 0.1	No evidence that the correlation is significant.

C. Feature Engineering

Followed by applying the binning technique for converting the int label 'Status' to categorical type, the data X and label y are derived from the dataset after cleanup. Because of a non-numerical categorical field and unknown specific data distribution, the feature 'Fuel' is transformed to the range [1, 4] and then Min-Max Scaling [15] in (1) is utilised to normalise the data, where the original data is normalized into the range $[C, D]$. In this case, C is 0 and D is 1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} * (D - C) + C \quad (1)$$

D. Data Modelling and Analysis

For current common model evaluation methods, the Train-test split (Holdout) method and k-fold cross-validation are two approaches [16] incorporated in this work. As the simplest method, the Holdout method divides the raw dataset into a training part and a test part, using the former part to train the model and the latter part to evaluate. In Scikit-learn, I utilise the *train_test_split* method with a *test_size* of 0.2 to complete it. The principle of the stratified k-fold cross-validation is to iterate over the dataset k times and select one fold at each time for evaluation, with the rest of k - 1 folds being trained in the training process.

Among hyperparameter optimisation methods, the Grid Search is a traditional and straightforward method, exhaustively exploring all possible combinations of hyperparameter values [17]. In my work, I synthesise the model evaluation method and the Grid Search, forming two route maps to perform modelling. The first route is using the Holdout method and manual grid search. Another is to combine manual stratified k-fold cross-validation with Scikit-learn's *GridSearchCV* class (The '*cv*' parameter is set as 2, and '*k*' is set as 5 to reduce the overall consumption time). To prepare for hyperparameter optimisation, I initialise some dictionaries to store all possible values of hyperparameters and a list storing used ML models for the second route map. Algorithm. 1 to Algorithm. 3 show the pseudo-code for them.

In Koklu's work [2], the state-of-the-art (SOTA) model is the Stacking method. As a result, this model method is reimplemented using SciKeras' *KerasClassifier* [18] to integrate three base estimators, including ANN, k-NN, and RF, with the default final estimator of *LogisticRegression*. After defining two route maps, five models (*KNeighborsClassifier*,

Algorithm 3 The Route Map Using k-fold cross-validation And GridSearchCV

Input: *data*: A dictionary containing hyperparameters of the trained model.

Input: *model*: The model to be trained.

Input: *k_fold_data*: A dictionary containing data and labels for k-fold cross-validation.

Output: *fold_result*: Results containing consumption time, the best model, score and parameters.

```

1: Get UUID and grid content from data.
2: Initialise a variable for storing the best model, parameters and score.
3: if uuid == these five models' uuid then
4:   Initialise a variable to count the overall consumption time.
5:   param_grid = Get the data and labels from the dictionary.
6:   Construct the GridSearchCV
7:   for i, fold_data in enumerate(k_fold_data) do
8:     Start timing for each fold.
9:     Initialise variables for storing the best model, parameters and score.
10:    Get the data and labels from fold_data.
11:    Fit the model.
12:    Compare to get the best model, score and parameters.
13:    Update the overall consumption time.
14:    Store training results of the current fold in fold_result[i].
15:  end for
16:  Display overall time.
17: else
18:   Other models are not supported yet.
19: end if
20: return fold_result

```

RadiusNeighborsClassifier, DT with Bagging, DT, and RF) call two route maps respectively, and the obtained results are stored in the form of *train_result*_{i} and *fold_result*_{i}. Apart from these models, I also implement a self-defined k-NN, which lists the core logic of the k-NN algorithm step by step.

As a binary classification task, models' performances are evaluated through several most popular metrics involving the confusion matrix, accuracy, precision, recall, F1-score, and ROC curve in this work. In these metrics, the result of the confusion matrix can facilitate the computation of other metrics [2], so the confusion matrix and accuracy are the most significant metrics to be considered during experiments.

IV. EXPERIMENTAL RESULTS

In this section, the results of data preprocessing, exploratory data analysis, hyperparameter grid settings and model evaluation with comparison are introduced in sequence.

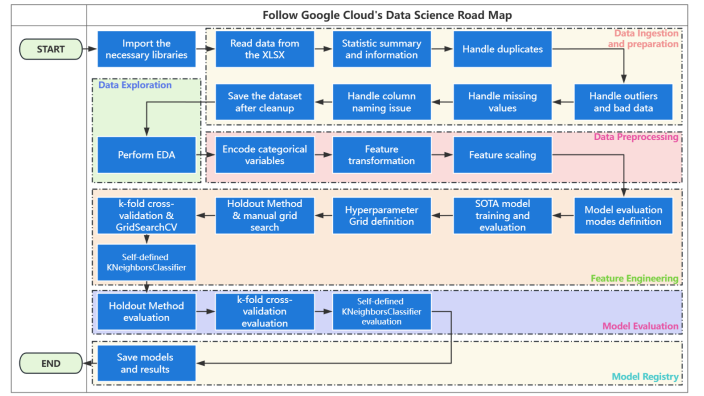


Fig. 3. Flowchart of the project. (Drawn by ProcessOn.)

A. Data Preprocessing

During this step, I use several methods mentioned in Section III. The results of the *skew()* method show that the skew of the label 'Status' is positive and near zero, meaning a positive skew. The mass of the distribution is concentrated on the left of the figure [19], which is consistent with 8759 data points containing a non-extinguishing state and 8683 data points with an extinguishing state. After checking the statistical summary information, there are no missing values or duplicates in the dataset. However, the name of the column 'DESIBEL' is wrong, enabling me to correct it.

B. Exploratory Data Analysis

Through the exploratory data analysis (EDA), I can figure out the distribution of each feature and the correlation among them.

According to Fig. 4, each feature has different distribution characteristics. For example, for the feature 'Fuel', there are 4 different fuel types. When the status is non-extinction, the frequencies of occurrence of gasoline, thinner and kerosene are very high, reaching about 2400, 2650 and 2800, respectively. But for the LPG fuel, its frequency of occurrence is about one-third of that of kerosene. For the extinction state, the frequencies of occurrence of four fuels represent a similar trend.

Fig. 5 shows box plots for the distribution of features under different flame states. In this figure, the data in the pink box represents that they are located between the lower quartile and the upper quartile. The red line in the box is the median, and the yellow plus sign represents the mean. For the feature 'Size' in the non-extinction state, the Q1, median, and Q3 are 2, 4, and 5, respectively, illustrating that data is concentrated in the range of [4, 5], while data for the extinction state is concentrated in the range of [2, 3]. For the feature 'Decibel' with status 0, the gap between Q1 and median is 4, which is shorter than that between median and Q3. It indicates more data is located in [91, 95]. And more data with status 1 are located in [90, 96]. In Fig. 6, the summary information about the best value range for each feature is revealed.

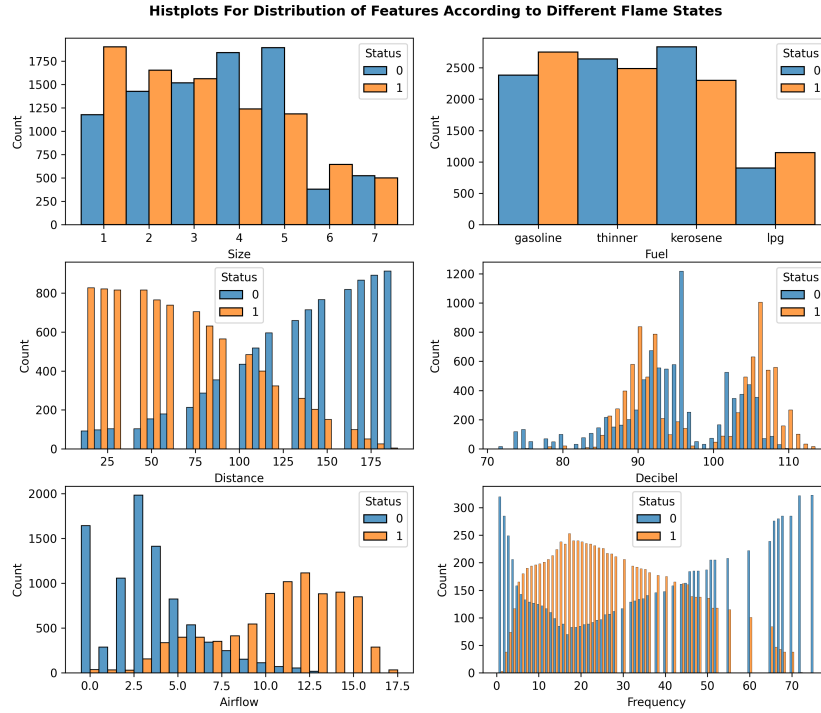


Fig. 4. Histplots for distribution of features according to different flame states. (Drawn by Seaborn.)

Fig. 7 concludes that the features 'Distance' and 'Airflow' can be most significant to the results of flame states. Besides, other features can also influence the status. By following the meanings of the Pearson correlation coefficient and p-value, it shows that the features 'Size', 'Distance' and 'Frequency' with 'Status' have a negative correlation, while the features 'Decibel' and 'Airflow' with 'Status' have a positive correlation.

In addition, from the Implots, the appropriate value range between various features can be found. Specifically, in Fig. 8, flames can be extinguished primarily when the distance is short with high airflow, or the distance is long with low airflow, revealing that the airflow will decrease when the distance becomes longer.

C. Training Procedure and Hyperparameters

When training the implemented models mentioned in Subsection III-D, to fairly compare the performances, only five models are trained and comprehensively evaluated both in the Holdout method and k-fold cross-validation strategy, while the self-defined k-NN is only implemented with the Holdout approach. Table II represents the hyperparameter grids used in the project. Because it consumes a long time to train the GridSearchCV using other hyperparameter values of RadiusNeighborsClassifier, I only implement the default value version.

D. Model Evaluation and Comparison with the State-of-the-art Method

From the results of the two route maps (Holdout Method denoted as I and k-fold cross-validation denoted as II), I mainly focus on the consumption time and accuracy. According to Table III, the performances of all models in route map II are worse than those in the other route map. For both route maps, the KNeighborsClassifier wins the highest accuracy of 97.11% and 96.45%, respectively, where the higher metric has only a 0.13% gap between it and the reimplemented SOTA model of 97.25%. Meanwhile, the overall duration of the KNeighborsClassifier in route map I consumes about 6.65 seconds, ranking in the top three. Compared with it, the training using GridSearchCV needs about 22.87 seconds, which is nearly four times as large as it. The first method has more advantages than the second one. The best hyperparameters can be found in Table IV.

What's more, generated confusion matrices indicate four basic results of a classification task (TP, FP, TN, and FN). Through them, many common metrics, including the accuracy score, can be calculated directly. Based on four results, we can plot the TP rate on the y-axis and FP rate on the x-axis, forming a ROC curve. The area under the curve is larger, the better the performance of the classifiers. Evidently, Fig. 9 and Fig. 10 demonstrate the best performance of the KNeighborsClassifier (The ROC AUC is the highest among all trained models).

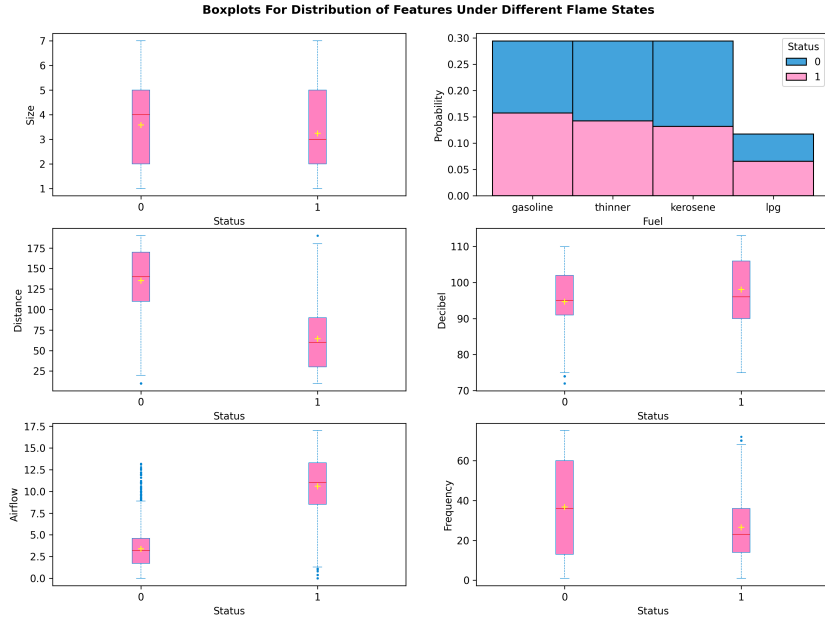


Fig. 5. Boxplots for distribution of features under different flame states. (Drawn by Seaborn.)

V. DISCUSSION

Both route maps applied in this work adopt the idea of grid search, facilitating the program to search for the best hyperparameter configurations among plenty of possible values. My results suggest that the k-NN is the best machine learning model for the binary flame extinction classification task. As shown in Table III, Fig. 9 and Fig. 10, its excellent performance is 4.49% higher than that of its counterpart [2], although other trained models here cannot outperform their counterparts. One possible reason is that the hyperparameters in this paper are confined to some extent, lacking rich hyperparameter values.

VI. CONCLUSION AND FUTURE RESEARCH

In this paper, owing to findings from EDA, all features, especially 'Distance' and 'Airflow', play a vital role in the results of flame extinction states. Six machine learning methods are trained and evaluated, and five among them go through two diverse approaches - Holdout Method with manual grid search, and k-fold cross-validation combined with GridSearchCV class, leading to a better performance. After comparing these models, the major finding is the outperforming results of the k-NN on the binary classification problem. In addition, to dive into this dataset, I compare my re-implemented stacking method with the model in the original paper, contributing to really close performance indicators.

However, because this paper has space to incorporate lots of useful machine learning models and evaluation methods, my future work includes implementing more powerful ML models and deploying other model evaluation methods like Leave-one-out (LOO), making more advanced and interesting results. For example, the LOO method can be compared with

my implemented grid search methods to figure out the most appropriate approach for this classification task.

ACKNOWLEDGMENT

I appreciate the sincere teaching on data modelling and analysis from Dr. Lim and Dr. Zhang.

REFERENCES

- [1] Liu, Z., Kim, A. K., & Carpenter, D. (2007). A study of portable water mist fire extinguishers used for extinguishment of multiple fire types. *Fire safety journal*, 42(1), 25-42.
- [2] Koklu M., Taspinar Y. S., (2021). Determining the Extinguishing Status of Fuel Flames With Sound Wave by Machine Learning Methods. *IEEE Access*, 9, pp.86207-86216, Doi: 10.1109/ACCESS.2021.3088612.
- [3] Taspinar Y.S., Koklu M., Altin M., (2021). Classification of Flame Extinction Based on Acoustic Oscillations using Artificial Intelligence Methods. *Case Studies in Thermal Engineering*, 28, 101561, Doi: 10.1016/j.csite.2021.101561
- [4] Taspinar, Y. S., Koklu, M., & Altin, M. (2022). Acoustic-driven airflow flame extinguishing system design and analysis of capabilities of low frequency in different fuels. *Fire technology*, 58(3), 1579-1597, Doi: 10.1007/s10694-021-01208-9.
- [5] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11).
- [6] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings* (pp. 986-996). Springer Berlin Heidelberg.
- [7] Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2), 130.
- [8] Breiman, L. (2001). Random forests. *Machine learning*, 45, 5-32.
- [9] Pavlyshenko, B. (2018, August). Using stacking approaches for machine learning models. In *2018 IEEE second international conference on data stream mining & processing (DSMP)* (pp. 255-258). IEEE.
- [10] Jang, J. S. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3), 665-685.

- [11] Google Cloud. (2022, February 1). Intro to data science on Google Cloud. <https://cloud.google.com/blog/topics/developers-practitioners/intro-data-science-google-cloud>.
- [12] Rebecca S.. (n.d.). The Data Visualisation Catalogue. <https://datavizcatalogue.com/>.
- [13] MedCalc Software Ltd. (n.d.). Correlation coefficient. <https://www.medcalc.org/manual/correlation.php>.
- [14] Waskom, M. L. (2021). Seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.
- [15] Patro, S. G. O. P. A. L., & Sahu, K. K. (2015). Normalization: A preprocessing stage. arXiv preprint arXiv:1503.06462.
- [16] Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. arXiv preprint arXiv:1811.12808.
- [17] Liashchynskiy, P., & Liashchynskiy, P. (2019). Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv:1912.06059.
- [18] Garcia Badaracco, A. SciKeras [Computer software]. <https://github.com/adriangb/scikeras>.
- [19] Skewness. (2025, April 18). In Wikipedia. <https://en.wikipedia.org/wiki/Skewness>.

Feature	Status	Q1	Median	Q3	Mean	IQR (=Q3-Q1)	Q1-1.5×IQR	Q3+1.5×IQR	Concentrated Range
Size	0	2	4	5	3.58	3	-2.5	9.5	[4, 5]
Size	1	2	3	5	3.24	3	-2.5	9.5	[2, 3]
Distance	0	110	140	170	135.12	60	20	260	Distributed uniformly
Distance	1	30	60	90	64.57	60	-60	180	Distributed uniformly
Decibel	0	91	95	102	94.72	11	74.5	118.5	[91, 95]
Decibel	1	90	96	106	98.05	16	66	130	[90, 96]
Airflow	0	1.7	3.2	4.6	3.39	2.90	-2.65	8.95	[3.2, 4.6] - slightly
Airflow	1	8.5	11	13.3	10.59	4.80	1.30	20.5	[11, 13.3] - slightly
Frequency	0	13	36	60	36.70	47	-57.5	130.5	[13, 36] - slightly
Frequency	1	14	23	36	26.48	22	-19	69	[14, 23]

Fig. 6. The best value range of features. (Organised through markdown.)

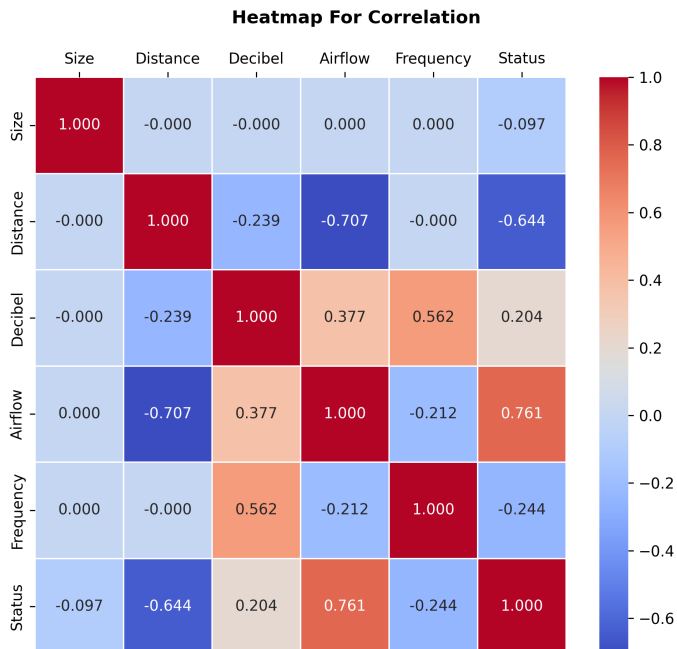


Fig. 7. Heatmap for correlation. (Drawn by Seaborn.)

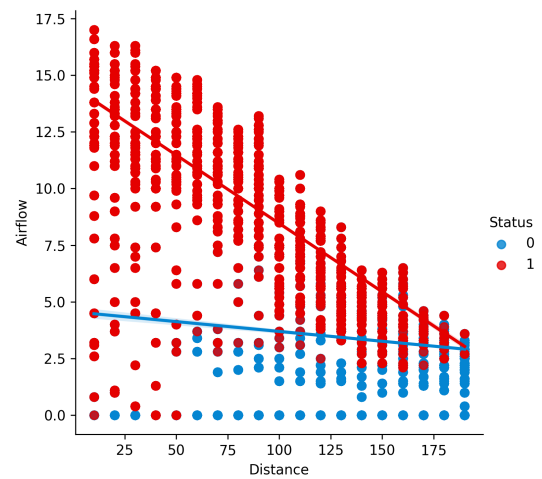


Fig. 8. Lmplots for distribution of 'Distance' and 'Airflow'. (Drawn by Seaborn.)

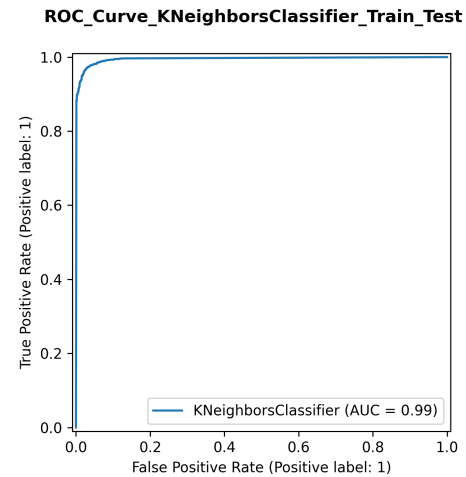


Fig. 9. ROC Curve of KNeighborsClassifier in Route Map I. (Drawn by Sklearn.)

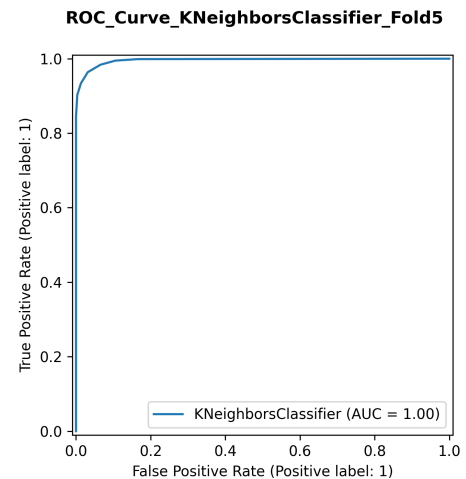


Fig. 10. ROC Curve of KNeighborsClassifier in Route Map II. (Drawn by Sklearn.)

TABLE II
HYPERPARAMETER CONFIGURATIONS OF ALL POSSIBLE VALUES

Model	Hyperparameter	Values
KNeighborsClassifier(UUID = 'knn-knc')	'weights'	['uniform', 'distance']
	'n_neighbors'	[4, 5, 6, 7]
	'algorithm'	['auto', 'ball_tree', 'kd_tree', 'brute']
	'p'	[2]
RadiusNeighborsClassifier(UUID = 'knn-rnc')	'radius'	[1.0]
	'weights'	['uniform']
	'algorithm'	['auto']
	'p'	[2]
DecisionTreeClassifier & BaggingClassifier(UUID = 'dt-bc')	[estimator] 'criterion'	['gini', 'entropy', 'log_loss']
	[estimator] 'splitter'	['best', 'random']
	[estimator] 'max_depth'	[2, 3, 4]
	'n_estimators'	[90, 100, 110]
DecisionTreeClassifier(UUID = 'dt')	'criterion'	['gini', 'entropy', 'log_loss']
	'splitter'	['best', 'random']
	'max_depth'	[2, 3, 4]
RandomForestClassifier(UUID = 'rf')	'n_estimators'	[90, 100, 110]
	'criterion'	['gini', 'entropy', 'log_loss']
	'max_depth'	[2, 3, 4]

TABLE III
EVALUATION RESULTS FOR TWO ROUTE MAPS

UUID	Route Map	Metrics (Round to two decimal places)				
		Duration (s)	Accuracy score	Precision score	Recall score	F1-score
knn-knc	I	-6.65	97.11%	97.11%	97.06%	97.09%
	II	-22.87±0.076	-96.45%	-96.76%	-96.10%	-96.42%
knn-rnc	I	-2.78	87.19%	88.84%	84.89%	86.82%
	II	-6.65±0.041	85.91%	87.40%	83.80%	85.55%
dt-bc	I	-43.13	91.34%	92.67%	89.68%	91.15%
	II	-58.32±0.420	89.04%	90.88%	86.71%	88.71%
dt	I	-0.16	89.85%	92.96%	86.10%	89.40%
	II	-0.26±0.000	86.86%	90.99%	81.69%	86.08%
rf	I	-14.25	89.85%	93.13%	85.93%	89.38%
	II	-19.48±0.155	88.38%	91.87%	84.12%	87.82%

TABLE IV
BEST HYPERPARAMETER VALUES

UUID	Route Map	Hyperparameter	Route Map	Hyperparameter
knn-knc	I	'weights': 'distance'	II	'algorithm': 'brute' 'n_neighbors': 7
knn-rnc	I	<default values>	II	<default values>
dt-bc	I	[DT] 'max_depth': 4 [BC] 'n_estimators': 100	II	[DT] 'max_depth': 4 [BC] 'n_estimators': 90
dt	I	'max_depth': 4	II	'max_depth': 4
rf	I	'max_depth': 4 'n_estimators': 110	II	'max_depth': 4