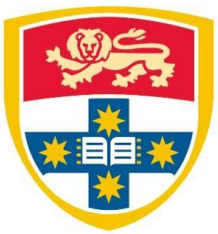

THE UNIVERSITY OF SYDNEY

Project Report for ENGG2112

Making Smoke Alarms Safer and Better

Pratul Singh Raghava, 51064305, Software Engineering
Kevin Aji Kuriakose, 510562278, Biomedical Engineering
Zeeshan Ansari, 510370813, Software Engineering
Seyun Jung, 510075516, Mechanical Engineering

Faculty of Engineering

November 06, 2022

Executive Summary

This report determines the best machine learning algorithm for a smoke alarm by using a smoke alarm dataset to train and test many different models and analysing them on a set of methods of performance. Today's smoke alarms have a high rate of failure, including raising false alarms and also failing to detect real instances of fire. These false alarms are a huge waste of fire department resources and also impact public trust in the effectiveness of fire alarms.

After performing initial data pre-processing on the dataset acquired, different machine learning models were trained on a subset of this data and then tested and measured by their accuracies and number of false negatives. Both the weighted (WNN) and unweighted (KNN) versions of the KNN model were found to be the best models since they had near-perfect accuracies and number of false negatives. This incredible accuracy suggests that the KNN model can distinguish between positive and negative cases quite precisely, thus providing a sense of security that it is very well suited to handle features that affect the decision-making process of a smoke alarm. No other model even came close to the extraordinary results that KNN produced.

There is an opportunity in the Smoke Alarm market for newer, more affordable and more accurate smoke alarms which can be digitally powered by our KNN/WNN machine learning models. These alarms would be readily upgradable through improvements in the technology, unlike today's traditional alarms. As a major potential commercialisation opportunity, these alarms can be pitched to leading smoke alarm manufacturing companies like ABB & Bosch.

Contents

1. Background and Motivation

A smoke alarm is a device that senses smoke, typically as an indicator of fire, and raises an alarm. The risk of a fatality in a home fire is halved if there is a working smoke alarm in a residential dwelling. Smoke detectors save a lot of lives. For example, the number of fire victims fell by more than 48% in France and 56% in the UK from 1982 to 2013. These reductions can largely be attributed to better fire safety regulations and more smoke detectors [1, 2, 3].

But, with an increase in the number of smoke alarms, comes an increase in the number of false alarms too. Unwanted alarms create complacency towards genuine alarms and this can result in serious injury or loss of life. They also divert fire department resources that would otherwise be available for genuine emergencies. A severe issue for firefighters, the number of false fire alarms is increasing continuously and was as high as 97% in Australia in 2015. As can be seen in Appendix-A, millions of false alarms are raised every year [4].

This massive defective inability in today's smoke alarms to correctly identify real fire emergencies poses another question. In how many cases, where there was a real fire emergency, has a smoke alarm failed to detect it and raise an alarm? This is a much more serious issue than a false alarm as it puts into question the very foundation of trust that we lay on smoke alarms to protect our lives.

Shockingly, Ionisation smoke alarms, which are the most prevalent type of smoke alarms available, have a staggering 55% failure rate. This means, in more than half the cases of real fire emergencies, these alarms fail to detect them and sound an alert, leading to many human casualties (see Appendix-B). In response to this, better versions like the Photoelectric smoke alarm have been introduced with lesser failure rates, but these are either very costly or not readily available [5].

The issue is that these technologies, Ionisation alarms (invented in 1940) & Photoelectric alarms (invented in 1972), just haven't kept up with time as newer sources and scenarios of fire emergencies have emerged. There is a need for a shift in the smoke alarm industry to deliver a more reliable and affordable alternative to the smoke alarms that we have today [6, 7].

This project is focused on the use of machine learning to train new smoke alarm technologies and help make them better at identifying and distinguishing between true and false fire emergencies. We intend to do this by making use of existing data which measures various factors that help smoke alarms make a decision and use it to give future smoke alarms better precedents to base their decisions on.

The dataset used for our project was produced by Stefan Blattmann with the goal of developing an AI based ionisation smoke detection device. The data, consisting of more than 60000 observations, includes different environments and sources of fire to improve the quality of the model to be developed [8].

The dataset observations were recorded by testing the IOT smoke detection devices in different environments and recording various parameters like temperature, humidity, pressure, particle size

and CO_2 levels of the environment areas. The result of the smoke test as well as the actual fire present is also present in the dataset for comparison.

Many studies have been performed using this dataset to train machine learning models like we intend to do. We humbly acknowledge them and aim to conduct our own independent study so we can compare our results to these pre-existing findings in order to confirm them, and if possible, suggest improvements [8].

2. Objectives and Problem Statement

The main objective of our project is to determine the best machine learning model by training and testing a variety of different models, all with different strengths and weaknesses. The models will be judged on a variety of measures of performance, but the two main objectives that we want our best machine learning model to excel in are :

a. *Low False Negatives & High True Positives* : To save human lives
b. *Low False Positives & High True Negatives* : To save fire department resources

Both of these objectives aim to improve the accuracy of smoke alarms so that we can restore and instil public trust in their effectiveness. With these objectives in mind, we develop our problem statement :

Enhance safety & efficiency of smoke alarms by optimising a machine learning model to reduce false negatives and increasing accuracy & true positives.

We intend to achieve our objectives by first identifying the features in our dataset that contribute the most to the fire alarm's decisions. We find this by calculating the correlation between each input feature and the target feature and getting rid of those input features that have no relation whatsoever.

Having trained our machine learning models, we aim to analyse each model using the pre-determined measures of performance and how each of their strengths and weaknesses affect their accuracy and whether the models are able to decipher different patterns in the dataset and use it to their advantage.

3. Methodology

As stated accurately in the "No Free Lunch" theorem", no one algorithm works best for every problem, especially in our case, i.e. predictive modelling. Therefore, for our analysis to be fruitful, it is only fair that a number of models be trained and tested on our dataset in order to determine the best one [9].

3.1 Data Pre-Processing

The first and foremost step of any data analysis is to pre-process it before it is used, so as to ensure quality of data and also to avoid a "Garbage In, Garbage Out" scenario where our results are incorrect due to faulty data being provided to the machine learning models.

Our first step was to identify the nature and data type of each column and check whether all columns were numerical in nature and whether any categorical features needed to be converted to numerical type using One-Hot Encoding or other methods. It was observed that there were no categorical features in the dataset (see Appendix C) and therefore no further processing was required in this matter.

Now that we have all numerical features, it was only logical to analyse the range of values of each column to identify any potential issues before we train our machine learning models on them. It was found that all columns had values greater than or equal to zero, except for *Temperature[C]* which had negative values (see Appendix-D) which were technically valid since this column contained values of temperature in °C which has values ranging as low as -273°C. This is a potential issue since some machine learning models, for e.g. - Multinomial Naive Bayes, do not accept values lower than zero.

To solve this issue, we converted all the temperature values from Celsius (°C) to Kelvin (K) since they are always greater than or equal to zero and therefore comply with the requirements of all machine learning models. The column was rightly renamed to *Temperature[K]* to reflect the change in units (see Appendix-E).

The next step of preprocessing was to identify and deal with missing and NaN values within the dataset which would hinder our analysis. Fortunately, upon writing the necessary code for it, it was found that there were none such cases within our dataset (see Appendix-F) and no further processing was required in this matter.

After all this preprocessing, the dataset was finally ready and fit to be used for our further analysis.

3.2 Feature Extraction

Feature Extraction is essential when dealing with datasets with a huge number of features (E.g. - greater than 100). Since we only have 15 features in our dataset, Feature Selection, not Extraction, is a much more appropriate strategy in order to eliminate any columns which do not contribute significantly to the results and are only a source of additional noise.

To do so, a Random Forest Classifier was trained to predict the *Fire Alarm* column using the other 15 features and then rank each feature in order of its importance in determining the final result. 3 columns were adjudged to be the least influential on the results and they are :

1. *CNT* : This is just a simple count.
2. *Unnamed: 0* : This is just the index column.
3. *UTC* : This column tells the time when the observation was recorded

It can be clearly seen that the above features have no relation whatsoever with how the Fire Alarm works and therefore can safely be discarded (see Appendix G). We now have 12 features and therefore utilising all 12 of the features to train our machine learning models is optimal since it doesn't cost us anything computationally and instead gives us just enough parameters to safely justify our results. Therefore, any sort of Feature Extraction/Selection is not required going ahead.

3.3 Methods

Now, on to the main part of our analysis. Having completed all pre-processing of data and feature extraction, it is time to train several different machine learning models on our dataset and then compare their results to find the best model for us. In total, 5 models were trained as part of our analysis and they are :

3.3.1 K-Nearest Neighbours (Weighted/Unweighted)

This model was developed by training and testing the KNN classifier. One issue that presents itself while using KNN is that depending on the value of *k* being too small or too large, the model is more sensitive to outlier values and values from other classes respectively [10].

To overcome this problem, we trained two separate models (see Appendix-H). One, the default unweighted classifier (KNN) , and the other, a weighted-KNN (WNN) , where the nearest *k* points are assigned a weight inversely proportional to their distance.

Since even for a weighted-KNN, results can vary for different values of *k*, both the weighted and unweighted models were trained on a number of different values of *k* to find just the perfect number when the accuracy of the models were the highest.

3.3.2 Multinomial Naive Bayes

This model was developed by training and testing the Multinomial Naive Bayes classifier (see Appendix-I). There are a couple of qualities about our dataset that should be noted :

a. All the features are discrete-valued, i.e. are in a multinomial distribution
b. All the features are reasonably independent from one another

Since the above points actually play to the strengths of our classifier, the model should be a good fit for our dataset.

Also, a specific requirement that a Multinomial Naive Bayes classifier has is that unlike some other classifiers, it cannot take negative values for any feature at all. Fortunately, this is something we predicted while pre-processing our dataset and therefore all the values provided to the classifier are indeed greater than or equal to zero.

3.3.3 Gaussian Naive Bayes

This model was developed by training and testing the Gaussian Naive Bayes Classifier (see Appendix-J). In this model, we use Bayes Rule and assume a probability distribution for each feature variable and find the parameters of those distributions from the data. Although this model conventionally works well with continuous data, it often works quite well even when feature variables are discrete. As with Multinomial Naive Bayes, Gaussian Naive Bayes too works well with features independent of one another and therefore should be a good fit for our dataset.

3.3.4 Logistic Regression

This model was developed by training and testing the Logistic Regression Classifier (see Appendix-K). Like Naive Bayes, this is a classification method based on finding the probability of a feature vector belonging in a certain class, i.e. we find $P\ [C=c|x]$ where C is the random variable representing the unknown class label, and x is the feature vector. We assume that the probability of being in class 1 (in a binary problem) is given by a certain simple formula, and then find the parameters in that formula from the data.

Theoretically, Logistic Regression is only well suited for continuous valued features and should not be much useful in our case. But, we still include it as one of our methods to verify this hypothesis.

3.3.5 Multi-Layer Perceptron Classifier

This model was developed by training and testing a Multi-Layer Perceptron Classifier (MLP) (see Appendix-L). MLP is a type of artificial neural network and the MLP Classifier can take any number of input features, any number of hidden layers, and any number of neurons per hidden layer.

For our model, we fixed the number of hidden layers to a certain number and tried various numbers of neurons to achieve maximum accuracy.

3.4 Simulation Environment

The simulations made use of a 70-30 training-testing split, and were run on Google Colab through Python notebooks. Random cross-validation of 20-100 instances was performed, with the measures of performance being calculated for each one and the best one taken for each model for further comparison with other models.

3.5 Measures of Performance

Since the main purpose of our analysis is to compare the effectiveness and feasibility of different machine learning models with each other, it is only fitting that several measures of performance be chosen to make a fair comparison.

For each of the models listed above in the Methods section, the following measures of performance were calculated and recorded :

a. **Accuracy Score** : A score generated by checking if the set of labels predicted for a sample exactly matches the corresponding set of labels in the set of true values or not [11].
b. **AUC Score** : Computes the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. A larger area, i.e. a greater AUC score is preferred [11].

c. **Mean Absolute Error** : The mean of the absolute errors in the predictions labels with respect to the true labels [11].

d. **Confusion Matrix** : A tabular graphic showing the number of True Positives, True Negatives, False Positives and False Negatives [11]. Furthermore, percentages were calculated for each number to easily compare models.

e. **ROC Curve** : Computes Receiver operating characteristic (ROC) and displays a graph with the true positive rate on the Y-Axis, and the false positive rate on the X-Axis [11].

The above measures were calculated several times for each model as a result of cross-validation and only the best scores were taken for each model for the final comparison.

4. Simulation Results

4.1 Individual Findings

4.4.1 Multinomial Naive Bayes

After performing 100 test_train_splits on the data, the highest accuracy achieved was just a mere 0.56 with a True Positive rate of 49.75% and a False Negative rate of 0.51%. Although 0.51% doesn't seem like much, considering our dataset with over 15,000 rows, that accounts for nearly 90 instances where the Smoke Alarm should have gone off but didn't. The ROC curve also isn't that impressive as the AUC score is just 0.55, which means that the model isn't that effective in distinguishing between positive and negative cases. A high Mean Absolute Error rate of 0.43 and a low accuracy of 0.56 don't seem to show this model as an improvement over the existing smoke alarms in the world. (see Appendix-M)

4.4.2 Gaussian Naive Bayes

After performing 100 test_train_splits on the data, the highest accuracy achieved was just a mere 0.77 with a True Positive rate of 70.42% and a False Negative rate of 1.54%. Although 1.54% doesn't seem like much, considering our dataset with over 15,000 rows, that accounts for nearly 250 instances where the Smoke Alarm should have gone off but didn't. The ROC curve also is quite impressive as the AUC score is nearly 0.94, which means that the model is quite good at distinguishing between positive and negative cases. Though, a Mean Absolute Error rate of 0.23 and a low accuracy of 0.77 don't seem to show this model as an improvement over the existing smoke alarms in the world (see Appendix-N).

4.4.3 Logistic Regression

After performing 100 test_train_splits on the data, the highest accuracy achieved was 0.88 with a False Negative rate of 1.1%. Although 1.54% doesn't seem like much, considering our dataset with over 15,000 rows, that accounts for nearly 200 instances where the Smoke Alarm should have gone off but didn't. The ROC curve is also almost perfect as the AUC score is nearly 0.95, which means that the model is quite good at distinguishing between positive and negative cases. Though, a low Mean Absolute Error rate of 0.11 and a medium accuracy of 0.88 make this model a potential improvement over the existing smoke alarms in the world (see Appendix-O).

4.4.4. Multi-layer Perceptron Classifier

After performing 100 test_train_splits on the data, the highest accuracy achieved was 0.98 with a False Negative rate of 13.95%. Considering our dataset with over 15,000 rows, that accounts for nearly 2600 instances where the Smoke Alarm should have gone off but didn't. The ROC curve is perfect as the AUC score is nearly 0.99, which means that the model is quite good at distinguishing between positive and negative cases. Though, the ROC curve and accuracy suggest that this model is highly effective, the Mean Absolute Error rate of 0.36 and shockingly high false negative rates seem to suggest otherwise and therefore do not show this model to be any sort of improvement over existing options (see Appendix-P).

4.4.5 K-Nearest Neighbours (Weighted/Unweighted)

For the Weighted KNN model, after performing 100 test_train_splits on the data, the highest accuracy achieved was almost a perfect 1.0 with a False Negative rate of 0.04%. This means there were only 2 instances where the smoke Alarm should have gone off but didn't, which is shockingly impressive considering that our dataset has over 15,000 observations. The ROC curve is absolutely perfect as the AUC score is nearly 1.0, which means that the model is extremely good at distinguishing between positive and negative cases. Since the ROC curve and accuracy suggest that this model is extremely effective, the Mean Absolute Error rate and false negative rates of almost 0 confirm our findings and therefore suggest this model to be a massive improvement over existing options (see Appendix-Q).

Like the weighted model, for the Unweighted KNN model, after performing 100 test_train_splits on the data, the highest accuracy achieved was also almost a perfect 1.0 with a False Negative rate of 0.11%. This means there were only 18 instances where the Smoke Alarm should have gone off but didn't, which is shockingly impressive considering that our dataset has over 15,000 observations. The ROC curve is absolutely perfect as the AUC score is nearly 1.0, which means that the model is extremely good at distinguishing between positive and negative cases. Since the ROC curve and accuracy suggest that this model is extremely effective, the Mean Absolute Error rate and false negative rates of almost 0 confirm our findings and therefore suggest this model to be a massive improvement over existing options and only marginally weaker than its weighted counterpart (see Appendix-R).

4.2 Key Findings and Significance

Having seen the individual findings of all models, we can now summarise them into a table for comparison (see Appendix-S).

As noted in their individual findings, KNN and WNN proved to be the most accurate models out of all other models used for this dataset, with KNN having a slight edge over WNN in False Negative rates although they match in all other parameters.

This implies that training KNN Classifiers on recorded observations like in our dataset and then deploying them in real world scenarios can help us achieve flawless accuracy in correctly identifying true and false fire emergency situations.

Since the decisions of these models are based on 12 different input features, each of which have separate mechanisms of being calculated by a smoke alarm, a few false negatives and false positives could be due to a variation in the sensitivity range in a smoke alarm detection device.

4.3 Issues Faced

The analysis process of this project was relatively straightforward and could be completed easily, barring some minor hiccups. They were :

a. **Multinomial Naive Bayes model rejecting negative values**

   While testing machine learning models, we were faced with quite some errors in the case of Multinomial Naive Bayes (MNB). We soon deciphered that this was because the *Temperature[C]* feature in the dataset contained negative values since its unit was Celsius. Since the MNB model doesn't accept negative values anywhere in the input variables, we had to perform some data processing to make the temperature values positive and at the same time, equally valid.

b. **Suspicious accuracy of KNN/WNN**

   One of the major successes of this project was also one of its more worrying points. The fact that we were able to achieve near-perfect values in all the measures of performance, led us to doubt our findings since they seemed too good to be true. It is quite rare for a machine learning model to be this accurate and therefore results like those that we observed for KNN/WNN enforced our verification of our results even further by compelling us to test for different testing dataset sizes, different k values and other sturdy cross validation methods.

5. Potential for Wider Adoption

Based on our findings as a result of this project, as mentioned in our problem statement, we wish to improve the safety and efficiency of smoke alarms worldwide. Since our study was successful

in indicating that this is highly possible, given the success of our KNN & WNN models, this gives rise to an opportunity for us to potentially commercialise our model in the Smoke Alarm Manufacturing industry.

The global fire alarm and detection market size was valued at USD 29.75 billion in 2021. Currently available options are either the moderately priced, low-accuracy Ionisation alarms or the very costly, relatively good accuracy photoelectric alarms. This trade-off between accuracy and price is hurting both the level of fire safety and the potential of this industry [12, 13].

There is quite a big opportunity for change and we firmly believe that newer upgraded versions equipped with our machine learning model, that are way cheaper and way more effective than current options, can help make a dent in this 30 billion dollar industry.

Some key players in this industry include Bosch, Honeywell, ABB and Siemens; and new technologies can help these companies develop advanced products in a bid to beat the competition. Another go-to-market focus will be offering after-sale services to ensure that the installations meet the safety and operation requirements with regular improvements to the machine learning model that powers these smoke alarms which is a feature not currently available in the traditional smoke alarms.

6. Conclusion

Our project was completed to a satisfiable degree. We managed to complete our goal of finding the best machine learning model for smoke alarm data to reduce false negatives and increase true positives, to achieve our goal of improving accuracy of smoke detectors. We completed every aim we set in the project proposal, and took it further by increasing the number of machine learning models tested. We met difficulties especially in the coding section as explained in section 4.3, but our work didn't stray from our proposal. However there were many additions like the introduction of ROC curve or confusion matrix to better illustrate our results to stakeholders. Our team worked efficiently, having 2-hour weekly meetings at 8pm on Mondays and communicating through Facebook Messenger to increase productivity. Every member contributed to the report and presentation, giving their best effort and being communicative team members in general.

Our main findings are as shown:

a) KNN and WNN had near perfect accuracies and AUC scores.
b) MLP classifier had an exceptionally high False Negative Rate.
c) Multinomial Naive Bayes fared quite poorly, contrary to expectations.
d) Low accuracy in current smoke detectors is a real problem, and can definitely be improved by adapting the right ML models (reassurance in our direction).

The project can be improved by running more datasets through our chosen model, and adapting the model to better detect anomalies and increase accuracy. This would be done by analysing the tables and graphs made from running the model and applying solutions to the model code. Once our final machine learning model reaches a satisfiable industry standard, it would be able to be commercialised to possible stakeholders such as leading smoke detector manufacturing companies.

References

[1] NSW Fire + Rescue, "Smoke Alarms," [Online]. Available: https://www.fire.nsw.gov.au/page.php?id=80.

[2] "Home Office statistics highlight fire false alarms remain an issue," 27 November 2020. [Online]. Available: https://www.bafe.org.uk/bafe-news/home-office-statistics-highlight-fire-false-alarms-remain-an-issue.

[3] MODERN BUILDING ALLIANCE, "FIRE SAFETY STATISTICS," [Online]. Available: https://www.modernbuildingalliance.eu/fire-safety-statistics/.

[4] NSW Fire + Rescue, "AUTOMATIC FIRE ALARMS - FALSE ALARMS," [Online]. Available: https://www.fire.nsw.gov.au/page.php?id=9028.

[5] Northern eastern Ohio fire prevention association, "Over half of all smoke alarms FAIL!," [Online]. Available: https://neofpa.org/smoke-alarms/half-smoke-alarms-fail/#:~:text=Over%20half%20of%20all%20smoke%20alarms%20FAIL.

[6] Smoke Detector (1970s), "Smoke Detector (1970s)," [Online]. Available: https://www.orau.org/health-physics-museum/collection/consumer/miscellaneous/smoke-detector.html.

[7] My smoke alarm, "SMOKE ALARM HISTORY," [Online]. Available: https://www.mysmokealarm.org/smoke-alarm-history/.

[8] D. CONTRACTOR, "Smoke Detection Dataset," [Online]. Available: https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset

[9] Elite data science, "Modern Machine Learning Algorithms: Strengths and Weaknesses," 8 July 2022. [Online]. Available: https://elitedatascience.com/machine-learning-algorithms.

[10] M. S. Khan, "What is weighted KNN and how does it work," 16 May 2020. [Online]. Available: https://medium.com/@mohdsaeed.khan25/what-is-weighted-knn-and-how-does-it-work-aa8e461fd5d7.

[11] Pedregosa et al., "Scikit-learn: Machine Learning in Python," 2011 , JMLR 12, pp. 2825-2830.

[12] Thomasnet, "Top Suppliers of Fire Alarms in the USA and Worldwide," [Online]. Available: https://www.thomasnet.com/articles/top-suppliers/fire-alarm-suppliers/.

[13] Grand View Research, "Fire Alarm And Detection Market Size, Share & Trends Analysis Report By Product (Fire Detectors, Fire Alarms), By Fire Detector Type, By Fire Alarm Type, By Application, By Region, And Segment Forecasts," 2021.

[14] My smoke alarm, "SMOKE ALARM STATISTICS," [Online]. Available: https://www.mysmokealarm.org/smoke-alarm-statistics/.

[15] V. Novozhilov, L. Shi , . M. Guerrieri, K. Moinuddin, . D. Bruck and P. Joseph, "Australian Building Codes," 2015.

[16] M. Ahrens, "Smoke Alarms in US Home Fires," February 2021. [Online]. Available: https://www.nfpa.org/News-and-Research/Data-research-and-tools/Detection-and-Signaling/Smoke-Alarms-in-US-Home-Fires.

Appendix

Appendix - A :

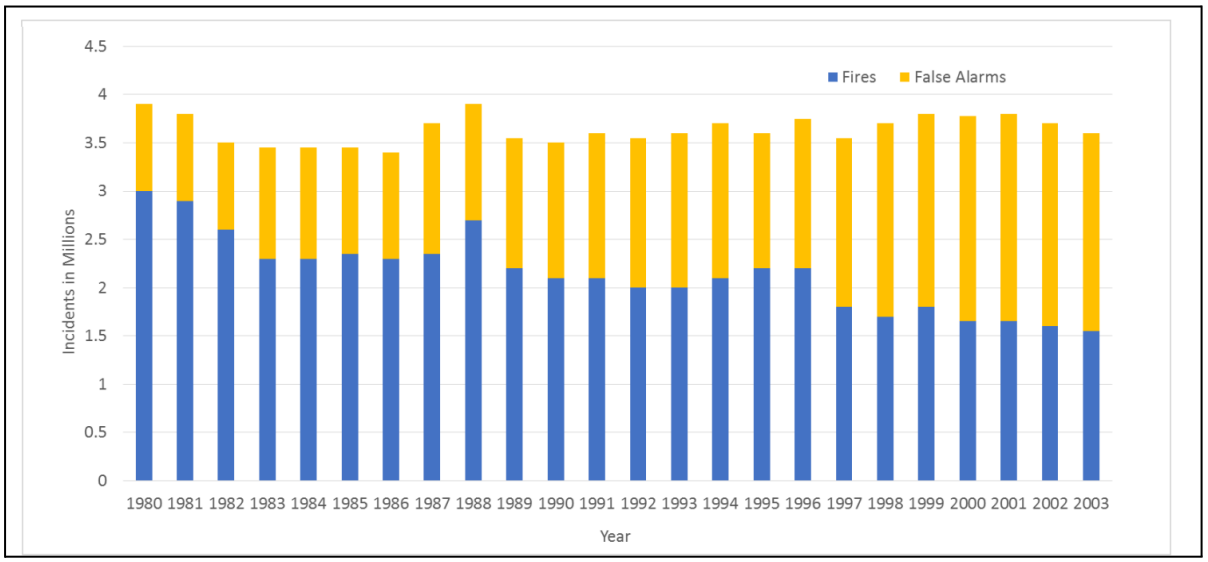

Fig. 1.1 Figure showing ratio of false alarms in reported incidents [14]

Appendix - B :

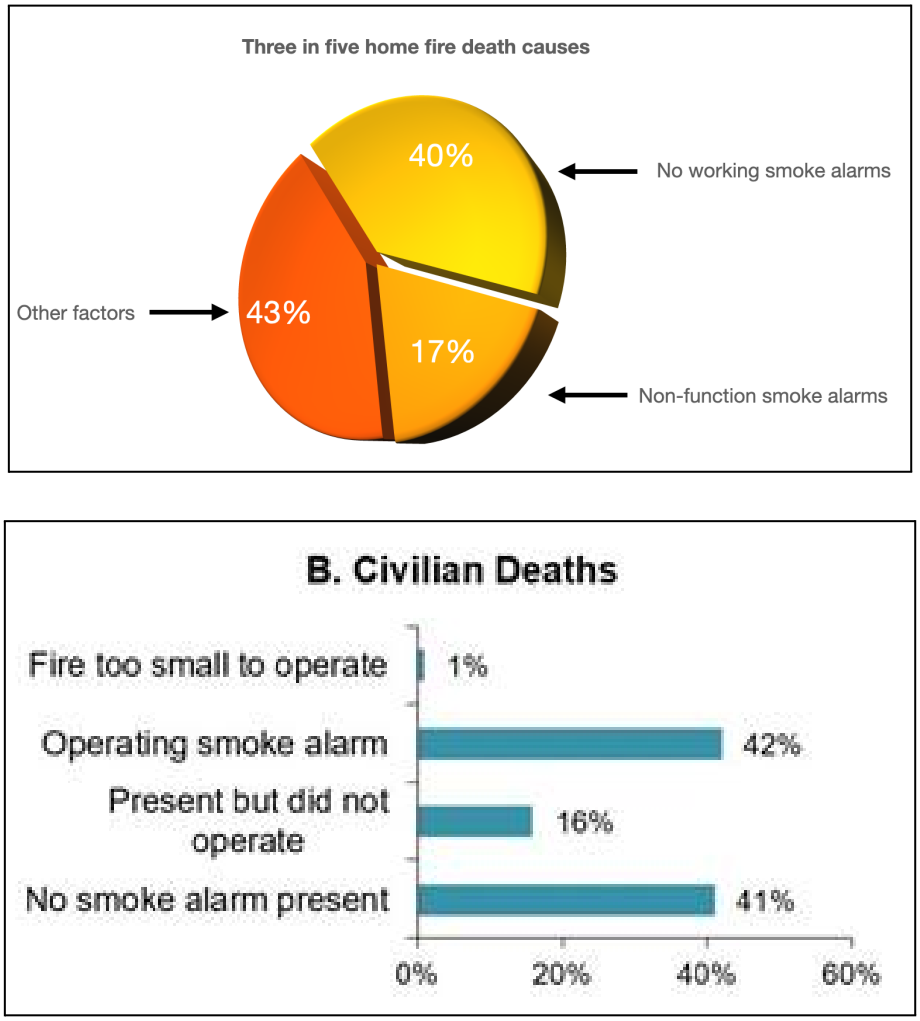

Fig. 1.2 16-17% of human fatalities are caused by faulty smoke alarms [15,16]

Appendix - C :

```
# Display datatypes
smoke.dtypes
```

```
Temperature[C]    float64
Humidity[%]       float64
TVOC[ppb]           int64
eCO2[ppm]           int64
Raw H2              int64
Raw Ethanol         int64
Pressure[hPa]     float64
PM1.0             float64
PM2.5             float64
NC0.5             float64
NC1.0             float64
NC2.5             float64
Fire Alarm          int64
dtype: object
```

Fig. 3.1.2 Finding column data types using Pandas.Dataframe.dtypes

Appendix - D :

```
# Defining a function to find the minimum and maximum values for a given column
def minMax(x):
    return pd.Series(index=['min','max'],data=[x.min(),x.max()])

# Printing the minimum and maximum values of each column in the dataframe
smoke.apply(minMax)
```

	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]
min	-22.01	10.74	0	400	10668	15317	930.852
max	59.93	75.20	60000	60000	13803	21410	939.861

Fig. 3.1.3 Finding Min & Max values using Pandas.Dataframe.apply()

Appendix - E :

```
# Defining a function to convert a given Celsius value to its Kelvin equivalent
def CelsiusToKelvin(t) :
  return t+273

# Converting all Celsius values in the temperature column to kelvin
smoke['Temperature[C]'] = smoke['Temperature[C]'].apply(CelsiusToKelvin)

# Renaming the above column
smoke.rename(columns={'Temperature[C]': 'Temperature[K]'}, inplace=True)
```

Fig. 3.1.4 Converting & Renaming the temperature column using Pandas.Dataframe.apply() & Pandas.Dataframe.rename()

Appendix - F :

```
# Counting NaN in the DataFrame
smoke.isna().sum().sum()
```

```
0
```

Fig. 3.1.5 Counting null values using Pandas.Dataframe.isna().sum().sum()

Appendix - G :

```
# Dropping the columns not required for our analysis
smoke = smoke.drop(['Unnamed: 0','CNT','UTC'], axis=1)
```

Fig. 3.1.1 Dropping columns using Pandas.Dataframe.drop()

Appendix - H :

```
# Training & Fitting the Unweighted KNN Classifier
unweighted_neigh = KNeighborsClassifier(n_neighbors=i**2)
unweighted_neigh.fit(X_train, y_train)
unweighted_y_pred = unweighted_neigh.fit(X_train, y_train).predict(X_test)
unweighted_acc = accuracy_score(y_test, unweighted_y_pred)
unweighted_auc = roc_auc_score(y_test, unweighted_y_pred)
```

Fig. 3.3.1.1 Training the unweighted classifier using sklearn.neighbors.KNeighborsClassifier()

```
# Training & Fitting the Weighted KNN Classifier
weighted_neigh = KNeighborsClassifier(n_neighbors=i**2, weights='distance')
weighted_neigh.fit(X_train, y_train)
weighted_y_pred = weighted_neigh.fit(X_train, y_train).predict(X_test)
weighted_acc = accuracy_score(y_test, weighted_y_pred)
weighted_auc = roc_auc_score(y_test, weighted_y_pred)
```

Fig. 3.3.1.2 Training the weighted classifier using sklearn.neighbors.KNeighborsClassifier()

Appendix - I :

```
# Creating Multinomial NB model and fit to training data
mnb_model = MultinomialNB()
mnb_model.fit(X_train, y_train)

# Getting predicted outputs
mnb_pred = mnb_model.predict(X_test)
mnb_acc = accuracy_score(y_test, mnb_pred)
mnb_prob = mnb_model.predict_proba(X_test)
```

Fig. 3.3.2 Training the classifier using sklearn.naive_bayes.MultinomialNB()

Appendix - J :

```
# Fitting the classifier into the training data
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Predicting the label of the testing data
gnb_y_pred = gnb.fit(X_train, y_train).predict(X_test)
```

Fig. 3.3.3 Training the classifier using sklearn.naive_bayes.GaussianNB()

Appendix - K :

```
# Create Logistic Regression model and fit to training data
lr_model = LogisticRegression(max_iter = 10000).fit(X_train, y_train)

# Predict y_preb and y_prob using the fitted model above
y_pred = lr_model.predict(X_test)
y_prob = lr_model.predict_proba(X_test)
```

Fig. 3.3.4 Training the classifier using sklearn.linear_model.LogisticRegression()

Appendix - L :

```
# Create Logistic Regression model and fit to training data
MLP_model = MLPClassifier(random_state = 1, hidden_layer_sizes = (20,),
                          batch_size = 20, max_iter = 100).fit(X_train, y_train)

# Predict y_preb and y_prob using the fitted model above
mlp_y_pred = MLP_model.predict(X_test)
mlp_y_prob = MLP_model.predict_proba(X_test)
```

Fig. 3.3.5 Training the classifier using sklearn.neural_network.MLPClassifier()

Appendix - M :

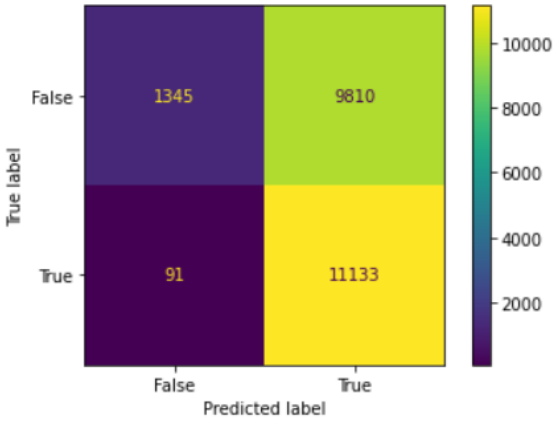

Fig. 4.1.1.1 Confusion Matrix using sklearn.metrics.confusion_matrix()

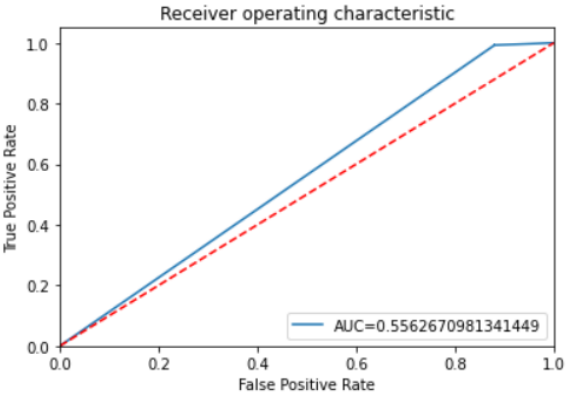

Fig. 4.1.1.2 ROC curve using sklearn.metrics.roc_curve()

```
True Negatives : 6.01%
False Positives : 43.84%
False Negatives : 0.41%
True Positives : 49.75%
```

Fig. 4.1.1.3 Confusion Matrix values as percentages

	Accuracy	AUC	Mean Absolute Error
6	0.563117	0.556580	0.4369
38	0.562402	0.558202	0.4376
58	0.562134	0.559117	0.4379
82	0.562089	0.559216	0.4379
25	0.561509	0.558280	0.4385

Fig. 4.1.1.4 Top 5 scores of Measures of Performance

Appendix - N :

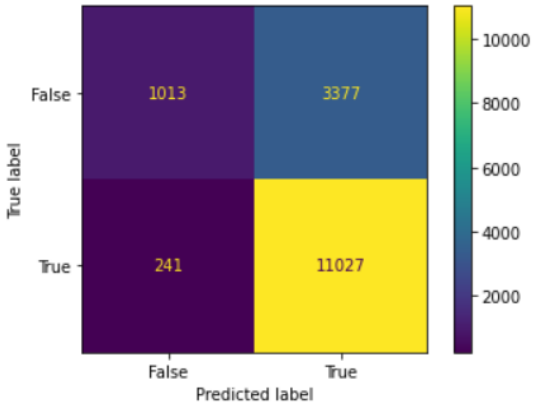

Fig. 4.1.2.1 Confusion Matrix using sklearn.metrics.confusion_matrix()

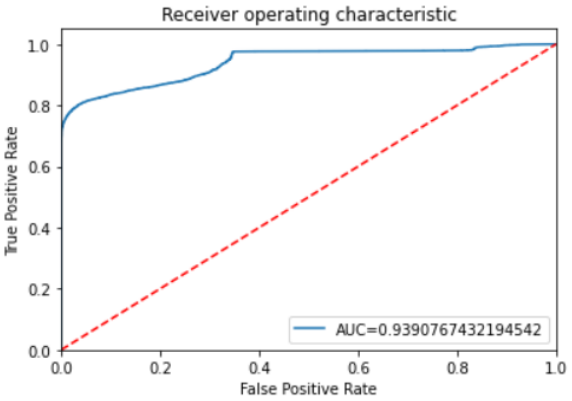

Fig. 4.1.2.2 ROC curve using sklearn.metrics.roc_curve()

```
True Negatives : 6.47%
False Positives : 21.57%
False Negatives : 1.54%
True Positives : 70.42%
```

Fig. 4.1.2.3 Confusion Matrix values as percentages

	Accuracy	AUC	Mean Absolute Error
9	0.770405	0.938159	0.2296
4	0.769830	0.935359	0.2302
6	0.768936	0.937027	0.2311
0	0.766765	0.938739	0.2332
3	0.765168	0.936709	0.2348

Fig. 4.1.2.4 Top 5 scores of Measures of Performance

Appendix - O :

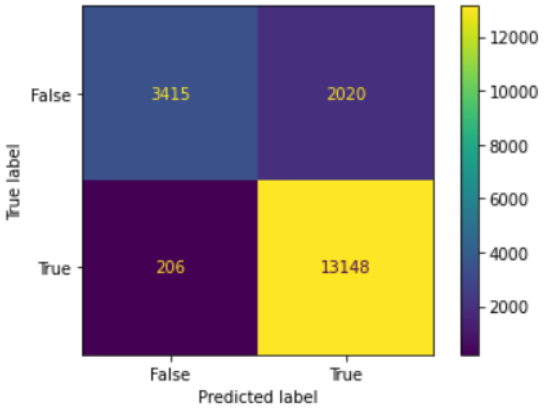

Fig. 4.1.3.1 Confusion Matrix using sklearn.metrics.confusion_matrix()

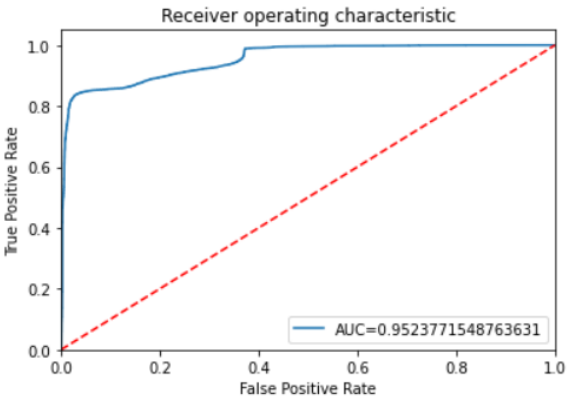

Fig. 4.1.3.2 ROC curve using sklearn.metrics.roc_curve()

```
True Negatives : 18.18%
False Positives : 10.75%
False Negatives : 1.10%
True Positives : 69.98%
```

Fig. 4.1.3.3 Confusion Matrix values as percentages

	Accuracy	AUC	Mean Absolute Error
83	0.886263	0.954632	0.113737
59	0.885518	0.954365	0.114482
93	0.885412	0.951964	0.114588
57	0.884720	0.955486	0.115280
91	0.884667	0.953855	0.115333

Fig. 4.1.3.4 Top 5 scores of Measures of Performance

Appendix - P :

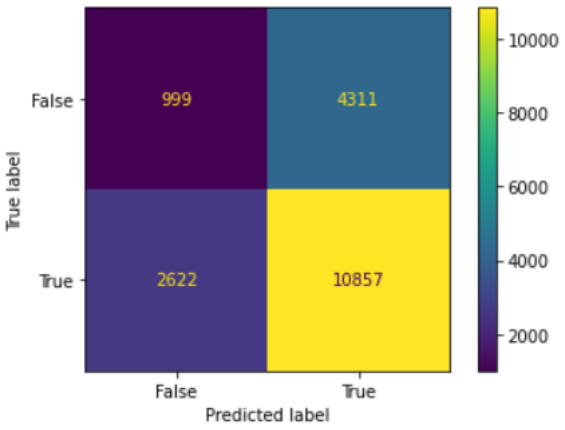

Fig. 4.1.4.1 Confusion Matrix using sklearn.metrics.confusion_matrix()

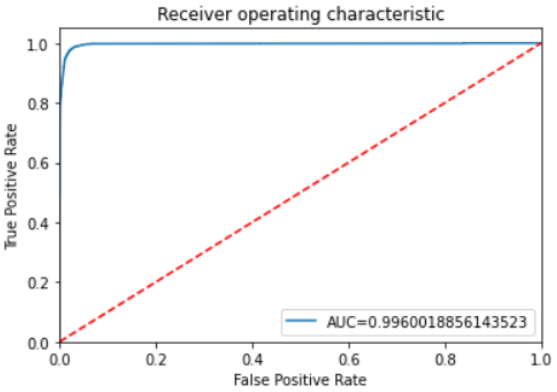

Fig. 4.1.4.2 ROC curve using sklearn.metrics.roc_curve()

```
True Negatives : 5.32%
False Positives : 22.94%
False Negatives : 13.95%
True Positives : 57.78%
```

Fig. 4.1.4.3 Confusion Matrix values as percentages

	Accuracy	AUC	Mean Absolute Error
1	0.983395	0.997659	0.362233
3	0.981532	0.995560	0.373197
2	0.981372	0.997699	0.365852
4	0.967108	0.996002	0.368992
0	0.912608	0.996482	0.367502

Fig. 4.1.4.4 Top 5 scores of Measures of Performance

Appendix - Q :

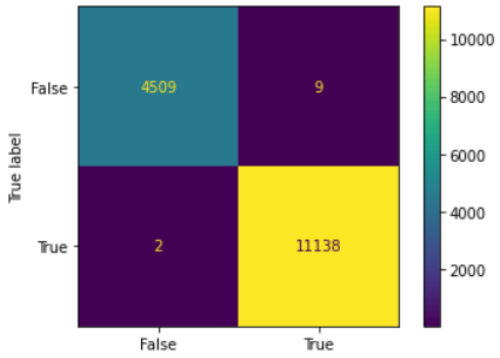

Fig. 4.1.5.1 Confusion Matrix using sklearn.metrics.confusion_matrix()

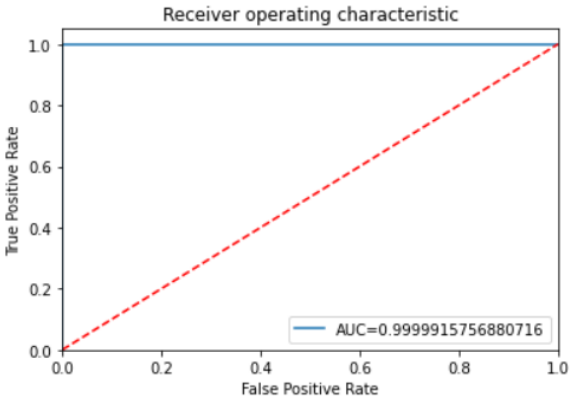

Fig. 4.1.5.2 ROC curve using sklearn.metrics.roc_curve()

```
True Negatives : 28.63%
False Positives : 0.06%
False Negatives : 0.04%
True Positives : 71.27%
```

Fig. 4.1.5.3 Confusion Matrix values as percentages

	Accuracy	AUC	Mean Absolute Error
0	0.999745	0.999754	0.0003
1	0.999745	0.999625	0.0003
6	0.999617	0.999392	0.0004
2	0.999617	0.999391	0.0004
3	0.999553	0.999350	0.0004

Fig. 4.1.5.4 Top 5 scores of Measures of Performance

Appendix - R :

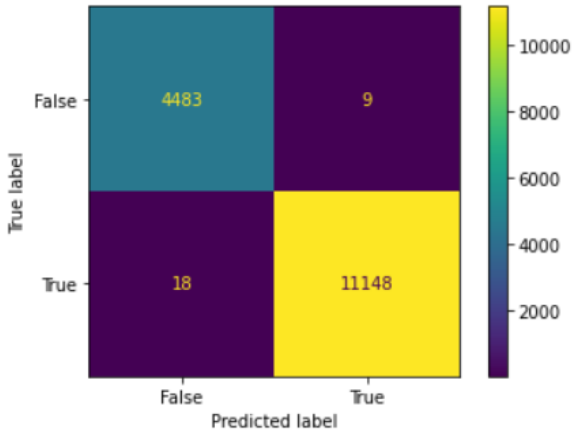

Fig. 4.1.5.1 Confusion Matrix using sklearn.metrics.confusion_matrix()

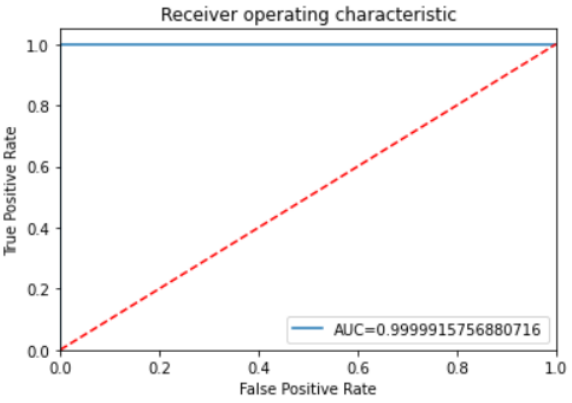

Fig. 4.1.5.2 ROC curve using sklearn.metrics.roc_curve()

```
True Negatives : 28.63%
False Positives : 0.06%
False Negatives : 0.11%
True Positives : 71.20%
```

Fig. 4.1.5.3 Confusion Matrix values as percentages

	Accuracy	AUC	Mean Absolute Error
0	0.999745	0.999754	0.0003
1	0.999681	0.999514	0.0003
2	0.999553	0.999278	0.0004
3	0.999489	0.999237	0.0005
6	0.999425	0.999122	0.0006

Fig. 4.1.5.4 Top 5 scores of Measures of Performance

Appendix - S :

Model	Accuracy	AUC	False Negatives	Mean Absolute Error
Weighted KNN	0.999745	0.999754	0.04%	0.0003
Unweighted KNN	0.999745	0.999745	0.11%	0.0003
MLP Classifier	0.983395	0.997699	13.95%	0.3622
Logistic Regression	0.886263	0.955486	1.10%	0.1148
Gaussian Naive Bayes	0.771427	0.939076	1.54%	0.2286
Multinomial Naive Bayes	0.562402	0.559216	0.41 %	0.4369