**Faculty of Computer & Information Technology**

**Gas Sensor: Advanced Gas Leakage Detection System For Enhanced Safety and Warning**

**By:**

**Nada Gamal Anwar (20-00293)**

**Reham Hesham Kamal (20-01057)**

**Rahma Kamal Abdelaziz (20-01150)**

**Fatma Ahmed Mohamed (20-00852)**

**Sandra Atef Sedky (20-00007)**

**Magy John Bastawroos (20-00249)**

**Shrouk Kadry Ahmed (20-01382)**

**Under Supervision of:**

**DR. Amany Magdy Mohamed**

professor, Faculty of Computers and Information at Egyptian E-Learning University

**TA. Mohamed Mostafa Saad**

Demonstrator, Faculty of Computers and Information at Egyptian E-Learning University

This project is submitted as a partial fulfillment of the requirements for the degree of Bachelor of Science in Computer & Information Technology

**Sohag 2024**

**Abstract**

Gas leaks pose significant risks to human health, safety, and the environment, whether from fire hazards to health issues, the consequences can be catastrophic, so present an innovative solution that addresses a paramount concern in the realms of safety and security.

With the digital transformation taking place all over the world and the developments of AI, it is the role of machine learning and artificial intelligence in enhancing the accuracy and speed of gas leak detection systems in the surrounding environment to solve people's problems and protect their lives.

**Acknowledgments**

First and foremost, praises and thanks to God, the Almighty, for his blessing throughout our years in college and our graduation project to complete this stage of our life successfully.

We have made efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Thanks to Egyptian E-Learning University especially Sohag center for helping us to reach this level of awareness.

We would like to express our deep and sincere gratitude to our project supervisor **DR. Amany Magdy Mohamed** for giving us the opportunity to lead us and providing invaluable guidance throughout this project. It was a great privilege and honor to work and study under his guidance. We are extremely grateful for what he has offered us.

We are especially indebted to say many thanks to **Eng. Mohamed Mostafa Saad** for guidance and constant supervision as well as for his patience, friendship, empathy, and great sense of humor. His dynamism, vision, sincerity and motivation have deeply inspired us.

Finally, we would like to express our gratitude to everyone who helped us during the graduation project

**Table of Contents**

**Abstract 2**

**[Acknowledgments 3](#_30j0zll)**

**List of Figures: 6**

**List of Tables: 8**

**List of Acronyms/Abbreviations: 9**

**1.Introduction 10**

**1.1 General 11**

**[1.2 Motivation 13](#_2bn6wsx)**

[**1.3 Scope of Work 1**](#_28reqzj)**5**

**[1.4 Organization of the report 16](#_qsh70q)**

**2.Background 17**

**[2.1 Introduction 18](#_23ckvvd)**

**[2.2 Gas Leakage 19](#_2u6wntf)**

**[2.3 Gas Detection 19](#_19c6y18)**

**[2.4 Feature selection 19](#_3tbugp1)**

**[2.5 Classifier Types 20](#_28h4qwu)**

**[2.6 Extra Trees 22](#_1mrcu09)**

**[2.7 Support Vector Machines 23](#_2lwamvv)**

**[2.8 K-Nearest Neighbor (KNN): 25](#_111kx3o)**

**[2.9 Random Forest 26](#_3l18frh)**

[**2.10 Decision Tree Classifier**](#_206ipza) **27**

**[2.11 Logistic Regression 29](#_2zbgiuw)**

**[2.12 Naïve Bayes 30](#_1egqt2p)**

**2.13 Haar Cascade Algorithm** [**3**](#_34g0dwd)**1**

**2.14 Conclusion 32**

**3.Related Work 33**

**3.1 Introduction 34**

**3.2 Research Papers Related to Gas Leakage Detection 35**

[**3.2.1 A Gas Leakage Detection Device Based on the Technology of TinyML**](#_xvir7l) **35**

**[3.2.2 Leak Detection in Natural Gas Pipelines Using Intelligent Model 36](#_3hv69ve)**

**[3.2.3 Low-Power, Chip-Scale, Carbon Dioxide Gas Sensors for Spacesuit Monitoring 37](#_4h042r0)**

**3.2.4 Enhancing Air Quality Monitoring with Optical Sensors (Eliches) 38**

**3.2.5 Advanced Thin-Film Hydrogen Sensors (H2scan)** [**3**](#_1baon6m)**9**

**[3.3 Conclusion 40](#_3vac5uf)**

**4.Overview** **41**

**4.1 Introduction 42**

**4.2 Requirements Specifications 43**

**4.2.1 Overview 43**

**4.2.2 Objective 43**

**4.2.3 Assumptions 44**

**4.2.4 Limitations 45**

**4.2.5 Proposed Workflow 46**

**4.2.6 Features 46**

**4.3 Inputs & Outputs 47**

**4.4 Security Requirements 47**

**4.5 Performance Requirements 48**

**4.6 Functional Requirements 49**

**4.7 Non-Functional Requirements 50**

**3.3 Conclusion 51**

**5.Dataset 52**

**5.1Explain Dataset 53**

**6.Implementation 55**

**6.1 Introduction 56**

[**6.2 Dataset loading and pre-processing 5**](#_2szc72q)**7**

**6.3 Splitting dataset 70**

**[6.4 Machine Learning Models 72](#_1qoc8b1)**

**6.4.1 Why use an Extra Tree? 94**

**[6.4.2 Discussion 94](#_1f7o1he)**

**6.5 Haar Cascade: 95**

**[6.6 Flutter Application: 96](#_1n1mu2y)**

**[6.7 Conclusion 115](#_1n1mu2y)**

**7.[Conclusion & Future Work 116](#_2xn8ts7)**

**[7.1 Conclusion 117](#_2xn8ts7)**

**[7.2 Future Work: 118](#_1csj400)**

[**Reference.**](#_2pcmsun) **119**

**List of Figures:**

**Figure 1: Extra tree algorithm mechanism 22**

**Figure 2: Decision tree algorithm 27**

**Figure 3: Screenshot of Gas Sensor Dataset 53**

**Figure 4: Screenshot of sensors we used 54**

**Figure 5: Screenshot of Gas Sensor Dataset (zooming)** [**5**](#_19mgy3x)**4**

**Figure 6: Screenshot of code of how we load dataset 57**

**Figure 7: Screenshot of code of how we Displaying top ten 58**

**Figure 8: Screenshot of code of how we statistical information** [**5**](#_30j0zll)**9**

**Figure 9: Screenshot of code of how we make metadata1****60**

**Figure 10: Screenshot of code of how we make metadata2 60**

**Figure 11: Screenshot of code of how we make ensure that there is not none value 61**

**Figure 12: Screenshot of code of how we make ensure that there is not duplicated data 62**

**Figure 13: Screenshot of code of unique value 62**

**Figure 14: Screenshot of code to drop columns 63**

**Figure 15: Screenshot of code of data profiling 64**

**Figure 16: Screenshot of code of visualization1 65**

**Figure 17: Screenshot of result visualization2 66**

**Figure 18: Screenshot of code of label encoder 67**

**Figure 19: Screenshot of code of data type 68**

**Figure 20: Screenshot of code of spliting1 69**

**Figure 21: Screenshot of code of splitting 2****69**

**Figure 22: Screenshot of code of split training & testing 70**

**Figure 23: Screenshot of code of display (y\_train ,x\_test)** [**7**](#_19mgy3x)**1**

**Figure 24:Screenshot of code of display (y\_test) 71**

**Figure 25: Screenshot for code and result Highest accuracy for extra tree****74**

**Figure 26: Screenshot for code and result Highest accuracy for Random Forest 77**

**Figure 27: Screenshot for code Highest accuracy for Knn****80**

**Figure 28: Screenshot for Graph of KNN 81**

**Figure 29: Screenshot for result Highest accuracy for KNN 81**

**Figure 30: Screenshot for code Highest accuracy for SVM 84**

**Figure 31: Screenshot for code and result Highest accuracy for Decision tree 87**

**Figure 32: Screenshot for code and result Highest accuracy for Naive bayes****89**

**Figure 33: Screenshot for code and result Highest accuracy for logistic regression 92**

**Figure 34: Screenshot of Code for live normal camera part 1 95**

**Figure 35: Screenshot of Code for live normal camera part 2 95**

**Figure 36: the result of Haar cascade****96**

**Figure 37: main code p1 98**

**Figure 38: main code p2 98**

**Figure 39: main code p3 99**

**Figure 40: main code p4 99**

**Figure 41: home page code p1** **100**

**Figure 42: home page code p2****100**

**Figure 43: Home Page****100**

**Figure 44: login code p1** **101**

**Figure 45: login code p2**  **101**

**Figure 46: login code p3** **101**

**Figure 47: login page****101**

**Figure 48: register code p1 102**

**Figure 49: register code p2** **102**

**Figure 50: register code p3****102**

**Figure 51: register code p4** **103**

**Figure 52: register code p5****103**

**Figure 53: register page 103**

**Figure 54: default page code p1 104**

**Figure 55: default page code p2** **104**

**Figure 56: default page****104**

**Figure 57: camera code p1** **105**

**Figure 58: camera code p2 105**

**Figure 59: camera page****105**

**Figure 60: settings page****106**

**Figure 61: settings code p1 107**

**Figure 62: settings code p2 107**

**Figure 63: settings code p3****108**

**Figure 64: settings code p4 108**

**Figure 65: settings code p5** **109**

**Figure 66: settings code p6****109**

**Figure 67: settings code p7** **109**

**Figure 68: settings code p8****109**

**Figure 69: Firebase package 110**

**Figure 70: main function 110**

**Figure71: Firebase authentication 111**

**Figure72: connection files (1) 111**

**Figure73: connection files (2)**  **111**

**Figure74: Haar cascade flask (1) 112**

**Figure75: Haar cascade flask (2) 112**

**Figure76: Docker file 113**

**Figure77: Extra tree (1) 113**

**Figure78: Extra tree (2)**  **114**

**Figure79: Extra tree flask**  **114**

**Figure80: requirements.txt**  **115**

**List of Tables:**

**Table 2.1: classification advantages and disadvantages 21**

**Table 6.2: Experment1 for extra tree 72**

**Table 6.3: Experment2 for extra tree 72**

**Table 6.4: Experment3 for extra tree 73**

**Table 6.5: Experment4 for extra tree 73**

**Table 6.6: Experment1 for Random Forest** 75

**Table 6.7: Experment2 for Random Forest 76**

**Table 6.8: Experment3 for Random Forest 76**

**Table 6.9: Experment1 for KNN 78**

**Table 6.10: Experment2 for KNN 78**

**Table 6.11: Experment3 for KNN 79**

**Table 6.12: Experment4 for KNN** 79

**Table 6.13: Experment1 for SVM 82**

**Table 6.14: Experment2 for SVM 83**

**Table 6.15: Experment3 for SVM 83**

**Table 6.16: Experment4 for SVM 83**

**Table 6.17: Experment1 for decision tree 85**

**Table 6.18: Experment2 for decision tree 85**

**Table 6.19: Experment3 for decision tree 86**

**Table 6.20: Experment4 for decision tree 86**

**Table 6.21: Experment1 for naïve bayes 88**

**Table 6.22: Experment2 for naïve bayes 88**

**Table 6.23: Experment3 for naïve bayes 89**

**Table 6.24: Experment4 for naïve bayes 89**

**Table 6.25: Experment1 for logistic regression 90**

**Table 6.26: Experment2 for logistic regression 90**

**Table 6.27: Experment3 for logistic regression 91**

**Table 6.28: Experment4 for logistic regression 91**

**List of Acronyms/Abbreviations:**

(AI) Artificial Intelligence

(AELCM) Atmospheric Exposure Low-Cost Monitoring

(GLD) Gas Leakage Detection

(IOT) Internet of Things

(KNN) K-Nearest Neighbor

(LCS) Low-Cost Sensors

(ML) Machine Learning

(RBF) Radial Basis Function

(STTR) Small Business Technology Transfer

(SVM) Support Vector Machine

**Chapter 1**

**Introduction**

**1.1 General**

Innovations in engineering have been pivotal in addressing numerous industrial and societal challenges, particularly in the chemical industries. Despite technological advancements, gas leakage remains a prevalent issue, posing significant risks to the environment and human health. Gas leaks, common in both industrial and residential areas, are a primary cause of air pollution and have led to worker fatalities during mining operations. Although gas leak testing is conducted before installation, leaks still occur, underscoring the need for reliable detection systems. Traditionally, gas detection sensors are deployed near potential leak sources. However, these sensors are often limited by their sensitivity to specific gasses, making it challenging to detect leaks in gas mixtures. Manual identification using chemical devices is not always feasible due to the hazardous nature of the chemicals and the obstruction of visibility caused by smoke. Therefore, automatic gas detection systems are critical for timely intervention and saving lives.

Recent advancements have seen the development of Internet of Things (IoT)-enabled systems for gas leak detection. These systems leverage low-cost sensor devices to identify gas leaks, but they are often hampered by the low sensitivity of these sensors. Detecting gas leaks in mixed environments.

remains a complex task that benefits significantly from technological advancements such as machine learning. Machine learning techniques enhance gas leak detection by analyzing data from multiple sensor devices, improving the accuracy and efficiency of these systems.

Continuous improvement of gas detection technologies is essential for better protection in industrial and residential environments. Gas leaks and fires in homes can cause severe casualties and property damage. Natural gas leaks, being highly flammable, increase the risk of fires and explosions, while exposure to household gas leaks or smoke inhalation can lead to serious respiratory issues. Early warning systems are crucial in mitigating these risks, as quick detection significantly improves outcomes by saving lives and property.

Effective gas leak detection and monitoring systems are vital. These systems utilize advanced sensors for real-time monitoring and thermal imaging devices to detect gas leaks in homes. Our advanced gas sensor system features instant alerts, enabling swift response measures to minimize potential damages. Early detection and identification of gas types are essential to prevent harm to humans and the environment, reinforcing the importance of innovative gas detection solutions.

**1.2 Motivation**

Gas leakage is a serious problem and may cause serious accidents such as explosions or gas poisoning. Therefore, we must have strong motivations to deal with and prevent this problem. Here are some possible motives:

1. Human life safety: Leakage caused by toxic or flammable gases is dangerous to human life. It is essential that we are motivated to act quickly to protect ourselves and our loved ones from these potential risks.
2. Preservation of property: Gas leakage may lead to fires and explosions that cause the destruction of property and buildings. In addition to human lives, we must be motivated to act quickly to preserve property and avoid material losses.
3. Social responsibility: Maintaining the safety of society and protecting it from risks is one of the most important social duties. When we take responsibility in dealing with gas leaks, we contribute to building a safer and healthier society.
4. Environmental awareness: Gas may leak from sources such as gas pipes or household appliances. Gas leaks contribute to increased carbon emissions, a factor causing global warming and climate change. Contributing to reducing gas leakage can be an opportunity to contribute to preserving the environment and reducing the effects of climate change.
5. Technological innovation: Combating gas leaks requires the development of new technologies and tools for early detection and prevention. We can play a role in promoting innovation and technological advancement by developing effective solutions for gas leak detection and response.

**Solution:**

The solution is to install a sensor with a camera.

1. The sensor measures the gas levels in the house.
2. If it does not exceed the normal limit, then there is no leak.
3. and if they exceed the required limit, this means that there is a gas leak.
4. The purpose of the sensor is also the speed of response and the intensity of sensitivity to the gas
5. The camera detects whether there are people in the house or not.
6. The camera was used and frames were created to detect people and count them using a counter.
7. An application has been created that sends a notification for users in case of a gas leak to increase the speed of response.

**1.3 Scope of Work**

A solution is proposed to help save humanity from the dangers caused by gas leaks. This system will detect gas leaks, measure their intensity, and use cameras to identify people present in the area. Lives will be saved, and fires, accidents, and losses resulting from gas leaks will be avoided. Locations using this system will keep pace with technological development and ensure the safety of lives around the world.

**1.4 Organization of the report**

The report is organized into 7 chapters.

In chapter 1, introduction, A quick intro to the project.

In chapter 2, Basic Background, the main machine learning concepts needed to understand the project are introduced.

In Chapter 3, Related Works. and applications are attempted to solve.

In Chapter 4, Overview, Explain the application generally.

In Chapter 5, Dataset, Viewing and Explaining Dataset.

In Chapter 6, Implementation, the development process will include,

Implementation stages, results, and discussions.

Finally, Chapter 7 contains the conclusion and future work.

**Chapter 2**

**Background**

**2.1 Introduction**

In chapter 1, a general description of the problem was presented, the motivation to solve the gas leakage problem.

In this chapter, a background about computerized systems is introduced from section 2.2 to 2.4. The main machine learning concepts needed to understand the project are illustrated in sections 2.6 to 2.13. Optimized searching techniques were introduced in section 2.5. Finally, chapter is concluded in section 2.14.

**2.2 Gas Leakage**

Gas is crucial for human life, expanding from household to industrial needs, but the risk of gas leaks causing fires is a concern**.**

Gas leak detection systems utilize sensors like the MQ-6 sensor to detect leaks and prevent potential hazards**.**

These systems often involve components like solenoid valves to shut off gas lines in case of a leak.

Testing involves measuring sensor readings and response times to ensure efficient gas leak detection**.**

**2.3 Gas Detection**

A gas leakage detection system using MQ-1, 2, 3, 4… and a camera would involve using multiple gas sensors to detect different types of gasses and a camera to visually detect any signs of a leak. The MQ sensors are each sensitive to different gasses and can detect their presence by measuring changes in resistance.

The camera can be used to detect physical signs of a leak, such as condensation or bubbles. The data from the sensors and camera can be processed using machine learning algorithms, such as Extra Trees or Naive Bayes, to accurately detect and classify gas leaks.

The accuracy of the system can be calculated by comparing the predicted results with the actual results and determining the percentage of correct predictions.

**2.4 Feature selection**

Feature selection is commonly used as a preprocessing step in machine learning, to be precise it is a process of selecting certain group of original features to use them in the prediction process, this process has proven to be very efficient in eliminating irrelevant & redundant features, which leads to increased efficiency in learning tasks and improved learning performance. (Ex. predictive accuracy).

**2.5 Classifier Types**

Classification is a predictive modeling problem where a class label is predicted for a given example of input data.

The machine learning model will use the training dataset and approximate the mapping function from input variables to discrete output variables and then identify the category the new data will fall into.

In Table [1], a comparison between the types of classification and their advantages and disadvantages is illustrated. [5]

Table 2.1: classification advantages and disadvantages

|  |  |  |
| --- | --- | --- |
| **Type of**  **Classification** | **Advantages** | **Disadvantages** |
| **Extra Tree** | Highest Accuracy | 1) Expensive, especially with a large number of trees and features, impacting the training time.  2) Lack of interpretability compared to other models like decision trees, making it challenging to understand the reasoning behind specific predictions.  3) May not perform well on imbalanced datasets, as it can prioritize majority classes and neglect minority classes, affecting the model's overall accuracy. |
| **KNN (K-Nearest Neighbor)** | Easy Implementation.  No Training Period. | Does not work well with a large dataset. Does not  work well with high  dimensionality |
| **Random forest** | 1) It can perform both regression and classification tasks.  2) A random forest produces good predictions that can be understood easily.  3) It can handle large datasets efficiently. | 1) Complexity  2)Longer Training Period |
| **SVM (Support Vector Machine)** | 1) Effective in high dimensional spaces.  2) Still effective in cases where the number of dimensions is greater than the number of samples.  3) Memory efficient. | Speed and size requirement in training and testing is more. |
| **Decision Tree** | 1) Does not require normalization of data.  2) Does not require scaling of data as well. | Involves higher time to train the model.  The algorithm is inadequate for applying regression and predicting continuous values. |
| **Logistic Regression** | 1) Interpretability.  2) Efficient and Fast: It is computationally efficient and can be trained quickly on large datasets.  4) Probabilistic Interpretation**:** Logistic Regression provides probabilities of class outcomes, enabling the assessment of the confidence in predictions.  5) Works well with linearly separable data. | 1) Assumption of Linearity.  2) Limited Flexibility.  3) Sensitive to Outliers.  4) Not Suitable for Non-Linear Relationships. |
| **Naïve Bayes** | 1) Simple and Easy to Implement.  2) Efficient with High DimensionalData**:** It performs well even with a large number of features, making it efficient for high-dimensional datasets. | 1) Assumption of Independence: Naive Bayes assumes that features are independent, which may not hold true in real-world datasets, impacting the model's accuracy.  2) Limited Expressiveness: Due to its simplicity, Naive Bayes may not capture complex relationships between features, leading to potential under-fitting. |

**2.6 Extra Trees**

Extra trees (short for extremely randomized trees) are an ensemble supervised machine learning method that uses decision trees and is used by the Train Using (Auto ML) tool. The extra trees algorithm creates many decision trees, but the sampling for each tree is random, without replacement. This creates a dataset for each tree with unique samples. A specific number of features, from the total set of features, are also selected randomly for each tree. The most important and unique characteristic of extra trees is the random selection of a splitting value for a feature. Instead of calculating a locally optimal value to split the data, the algorithm randomly selects a split value. This makes the trees diversified and uncorrelated.

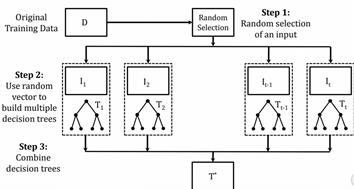


Figure 1: Extra tree algorithm mechanism

**Accuracy**

The accuracy equation for the Extra Trees Classifier involves evaluating the performance of the model in correctly predicting the target variable compared to the actual values.

**2.7 Support Vector Machines**

of emails, news SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees.

It is known for its kernel trick to handle nonlinear input spaces.

It is used in a variety of applications such as face detection, intrusion detection, classification articles and web pages, classification of genes, and handwriting recognition.

SVM is an exciting algorithm, and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

**Accuracy**

It is simply defined as the ratio between the number of correct predictions made to all the predictions made, it can be calculated using the following equation.

**Accuracy = (TP + TN) / (TP + TN + FP + FN)**

**Precision**

It is the proportion of Predicted Positive cases that are actual Positives or predicted Negatives that are actual negatives; it can be calculated using the following equations.

**Positive Precision =𝑇𝑃 / (𝑇𝑃 + 𝐹𝑃)**

**Negative Precision =𝑇𝑁 / 𝑇𝑁 + 𝐹*N***

**Sensitivity/Recall**

It is the proportion of actual positive cases that are correctly Predicted Positive, in other words it measures how well your model identifies positive cases; it can be calculated using the following equation.

**Sensitivity =𝑇𝑃 / 𝑇𝑃 + 𝐹𝑁**

**Specificity**

It is the proportion of actual Negative cases that are correctly Predicted Negative, in other words it measures how well your model identifies Negative cases; it can be calculated using the following equation.

**Negative Precision =𝑇𝑁 / 𝑇𝑁 + 𝐹P**

**𝑭𝟏-Measure**

It is the harmonic mean of precision and recall/Sensitivity, its value is ranged from 0 to 1 and a high value of the 𝐹1-Measure indicates high Model performance; it can be calculated using the following equation

**𝐹1 − measure =*2\*Precision\*Recall/ 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 + 𝑅𝑒𝑐𝑎𝑙𝑙.***

There is also a more generalized form which is

𝐹𝛽-measure which is given be the following equation.

𝐹𝛽 − measure = (𝛽 2 + 1) ∗ 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 ∗ 𝑅𝑒𝑐 / 𝛽2 ∗ 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 + 𝑅𝑒𝑐𝑎𝑙*l*

Where 𝛽 controls the balance between Precision and recall where 𝛽 = 1 means that precision and recall are of equal importance which is The Normal 𝐹1-measure, 𝛽 > 1 means that recall is more important than precision while 𝛽 < 1 means that precision is more important.

There is also a simpler approach to measure 𝐹𝛽 −measure which is to use.

**𝐹𝛼 − measure where 𝛼 not equal (𝛽2+1)/1 giving us the equation.**

**𝐹𝛼 − measure =1 / 𝛼 ∗ 1/𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 + (1 − 𝛼) \*1/𝑅𝑒𝑐𝑎𝑙𝑙**

Where 𝛼 ranges from 0 to 1 the greater the 𝛼 the greater the importance of Precision, and when 𝛼 = 0.5 that means both are of the same importance i.e., = 1.

**2.8 K-Nearest Neighbor (KNN):**

KNN is one of the simplest Machine Learning algorithms based on Supervised Learning technique. The K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most like the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-suited category by using the K- NN algorithm.

KNN can be used for Regression as well as for Classification but mostly it is used for the Classification problems. KNN at the training phase just stores the dataset and when it gets new data, and then it classifies that data into a category that is much like the new data.

**Accuracy:**

KNN (K=1)

1. KNN model. Pick a value for K

2. This would always have 100% accuracy, because we are testing on the exact same data, it would always make correct predictions.

3. KNN would search for one nearest observation and find that exact same observation. KNN has memorized the training set.

**Precision:**

The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives.

The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0.

**2.9 Random Forest**

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest

Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. A random forest is a machine learning technique that’s used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

**Accuracy:**

from sklearn. ensemble import RandomForestClassifier.

classifier = RandomForestClassifier (n\_estimators=20,

random\_state=0)

classifier. fit(X\_train, y\_train) y\_pred = classifier. predict(X\_test)

**F1 measure:**

F1 score is a little less intuitive because it combines precision and recall into one metric. If precision and recall are both high, F1 will be high, too. If they are both low, F1 will be low. If one is high and the other low, F1 will be low.

**2.10 Decision Tree Classifier**

Decision Tree is a supervised learning techniquethat can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, whereinternal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node andLeaf Node**.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed based on features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, like a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm**,** which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question and based on the answer (Yes/No), it further splits the tree into subtrees.

**Below diagram explains the general structure of a decision tree:**

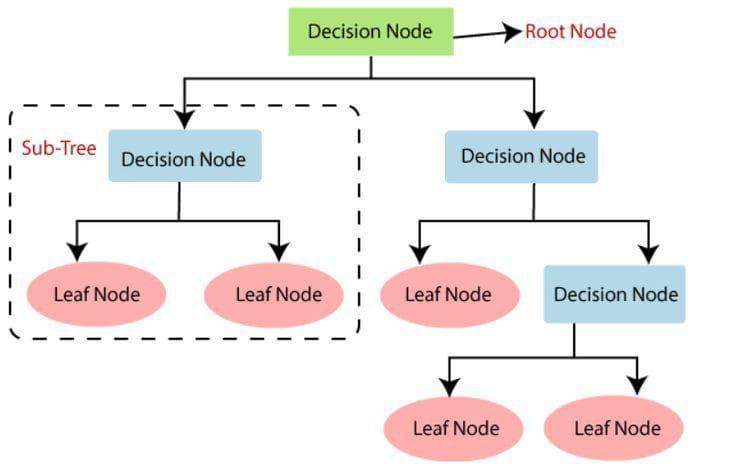
****

Figure 2: Decision tree algorithm

**Entropy**

Entropy is employed to measure a dataset's impurity or randomness. The value of entropy always lies between 0 and 1. Its value is better when it is equal to 0 while it is worse when it is equal to 0, the closer its value to 0 the better. If the target is 𝐺 with different attribute values, the entropy of the classification of set 𝑆 with respect to 𝑐 states. As shown in “equation (1)”.**Entropy (𝑆) = −∑ni=1 pi Pi (1)​**Where 𝑃𝑖 is the ratio of the sample number of the subset and the 𝑖-th attribute value.

**Information gain**

Is one metric used for segmentation and is often called mutual information. This intuitively informs how much knowledge of a random variable's value .

It’s the opposite of entropy, the higher its value is the better. The data gain 𝐺𝑎( 𝑆 , 𝐴 ) is defined as the following on the definition of entropy , as shown in “equation (2)”.

**Gain(S,A)=Entropy(S)−∑v∈V(A)​(∣Sv∣/∣S​∣​)Entropy(Sv​) (2)**

Where the range of attribute 𝐴 is (𝐴), and 𝑆𝑣 is a subset of set 𝑆 equal to the attribute value of attribute.

**Accuracy:**

from sklearn import DecisionTreeClassification.

Classifier= DecisionTreeClassifier (n\_estimators=20, random\_state=0)

classifier. fit (X\_train, y\_train) y\_pred = classifier. predict(X\_test)

**2.11 Logistic Regression**

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false.

Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. It is extensively used in predictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not.Logistic regression serves several key purposes in statistical analysis, classification and predictive analytics: Classification and predictive analytics: Logistic regression streamlines the mathematics for measuring the effect of multiple variables (e.g., age, gender, ad placement) with a given outcome (e.g., click-through, ignore). The resulting models can carefully separate the relative effectiveness of various interventions for different categories of people, such as young/old or male/female.Binary outcome prediction: Logistic regression is ideal for analyzing scenarios with a binary dependent variable, predicting possible outcomes such as yes or no based on previous data. Its effectiveness in this regard makes it a staple in fields such as marketing, finance and data science.

**Accuracy:**

The accuracy of a logistic regression algorithm can be measured using the accuracy score, which is the proportion of correct predictions over total predictions.

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

log\_reg = LogisticRegression()

log\_reg.fit(X\_train, y\_train)

y\_pred = log\_reg.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

**2.12 Naïve Bayes**

Naive Bayes is a family of classification algorithms based on Bayes' Theorem, which finds the probability of an event occurring given the probability of another event that has already occurred. The algorithm is called "naive" because it makes the simplifying assumption that the features used to describe an observation are conditionally independent, given the class label. Despite this oversimplified assumption, Naive Bayes classifiers are widely used for their simplicity and efficiency in machine learning. They are particularly useful in classification problems with high-dimensional data, such as text classification, and are often used in spam filtering, sentiment detection, and rating classification. The algorithm predicts the probability of an instance belonging to a class with a given set of feature values, and it uses Bayes' theorem in the algorithm for training and prediction. The Naive Bayes algorithm is highly scalable and can handle both continuous and discrete data.

**Accuracy:**

The accuracy of a Naive Bayes algorithm can be measured using the accuracy score, which is the proportion of correct predictions over total predictions.

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

y\_pred = gnb.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

**2.13 Haar Cascade Algorithm**

Haar Cascade is particularly well-suited for **facial detection** because the Haar-like features can be used to distinguish between facial features such as the eyes, nose, and mouth. The algorithm detects faces by first creating a Haar Cascade classifier using a set of positive and negative images. The positive images contain faces, while the negative images do not.The classifier is then used to scan a new image or video for faces. The scanning process involves sliding a window of fixed size over the image and applying the classifier to each window. If the classifier detects a face in the window, it is marked as a potential face. The potential faces are then filtered based on their size, position, and shape to reduce false positives.

**Training**

The algorithm is trained with a large set of positive and negative images. Positive images contain the object to be detected (e.g., faces), while negative images do not.

**Feature Extraction**

Haar-like features are used to identify the presence of an object. These features are reminiscent of convolutional kernel filters in neural networks.

**Cascade Classifier**

The features are organized into a cascade of classifiers. This means that the algorithm applies the most straightforward classifiers first and progressively more complex classifiers only if needed. This cascading approach helps in quickly eliminating regions of the image that do not contain the object.[6]

**2.14 Conclusion**

In this chapter, computerized systems diagnosis was explained, and an overview about machine learning concepts and essential topics were introduced.

In the next chapter, the related work and many related techniques will be introduced.

**Chapter 3**

**Related Work**

**3.1 Introduction**

In chapter 2, a general background about (Gas Leakage Detection) & machine learning concepts were introduced and how machine learning could help in protecting people from gas leakage.

In this chapter, the related research papers and applications to Gas Leakage Detection (GLD) support machine learning.

Machine learning approach for gas leak detection and the type of gas leaking is based on sensor readings and camera images. The related research papers for the project are described in Sections 3.2.1 to 3.2.5. The chapter is concluded in Section 3.3.

**3.2** **Research Papers Related to Gas Leakage Detection**

The previous background presented in the last chapter was necessary to understand the related work in section 3.2.1 to 3.2.5 .This section contains the summary of surveying several papers related to (GLD) and applying machine learning to protect from gas leakage risks.

**3.2.1 A Gas Leakage Detection Device Based on the Technology of TinyML, Vasileios Tsoukas , (22 March 2023)**

Internet of Things devices are frequently used as consumer devices to provide digital solutions, such as smart lighting and digital voice-activated assistants, but they are also employed to alert residents in the instance of an emergency. Given the increasingly costly nature of present neural network systems, it is necessary to transport information to the cloud for intelligent machine analysis. TinyML is a potential technology that has been presented by the research world for building fully independent and safe devices that can gather, analyze, and produce data, without transferring it to distant organizations. This paper describes a gas leakage detection system based on TinyML. The proposed solution can be programmed to identify anomalies and warn occupants via the utilization of the BLE technology.[1]

**Limitations:**

1. The shortcomings of this paper are that the model was not trained enough to determine the type of leak.

**3.2.2 Leak Detection in Natural Gas Pipelines Using Intelligent Model,Oshingbesan Adebayo(April 2019)**

Leak detection in gas pipelines is an important and persistent problem in the (Oil and Gas) industry.

This research aims to study the ability of data-driven intelligent models to detect small leaks for a natural gas pipeline using basic operational parameters and then compare the intelligent models among themselves using existing performance metrics.

This project applies the observer design technique to detect leaks in natural gas pipelines using a regressor classification hierarchical model where an intelligent model acts as a regressor and a modified logistic regression model acts as a classifier.

The results show that while support vector machines and artificial neural networks are better regressors than the others, they do not provide the best results in leak detection due to their internal complexities and the volume of data used.

All the intelligent models had high reliability with zero false alarm rates in the testing phase. The average time to leak detection for all the intelligent models was compared to a real time transient model in literature. The results show that intelligent models perform relatively well in the problem of leak detection.[2]

**Limitations:-**

1. This paper does not contain a mobile application for alerts in case of gas leak.

**3.2.3 Low-Power, Chip-Scale, Carbon Dioxide Gas Sensors for Spacesuit Monitoring (N5 Sensors)**

N5 Sensors, Inc. through a Small Business Technology Transfer (STTR) contract award has been developing ultra-small, low-power carbon dioxide (CO2) gas sensors, suited for monitoring CO2 levels inside  
NASA spacesuits. Due to the unique environmental conditions within the spacesuits, such as high humidity,  
large temperature swings, and operating pressure swings, measurement of key gasses relevant to astronaut’s. Architecture has been developed for this application, which can be operated in a typical Space-suite environmental condition. All electronic design provides a compact and low-power solution, which can be implemented for multipoint detection of CO2 inside the spacesuits. This paper will describe the sensor  
architecture, development of new photo catalytic material for better sensor response, and advanced structure  
for better sensitivity and shorter response times.

Enables chip-scale gas sensing and their low-power sensor platform can be successfully implemented in mobile devices and wearable. The sensors can detect various gasses such as hydrogen, methane, CO, CO2, VOCs.

**3.2.4 Enhancing Air Quality Monitoring with Optical Sensors (Eliches)**

Eliches focuses on developing optical sensors for air quality monitoring, offering individuals the ability to monitor air quality using innovative technology. These sensors can detect various gasses like CH4, CO, CO2, NO2, SO2, and O2 in complex gas mixtures, providing real-time data for analysis. The use of low-cost sensors (LCS) in air pollution monitoring requires proper calibration to ensure accurate PM2.5 estimates, with calibration models being developed based on co-location with reference monitors. Additionally, advancements in sensor technology have enabled the creation of the Atmospheric Exposure Low-Cost Monitoring (AELCM) system, allowing for high-resolution monitoring of atmospheric conditions and air substances in urban environments. While air sensors show promise for expanding epidemiological studies in regions with high pollution levels, calibration efforts are crucial to ensure data accuracy.

It is an optical sensor for air quality monitoring. It focuses on the development of technology for air quality management. It offers Optical sensors that are fitted into devices and help individuals monitor the air quality**.**

**3.2.5 Advanced Thin-Film Hydrogen Sensors (H2scan)**

H2scan offers a thin-film hydrogen sensor suitable for high temperature and marine applications. These sensors utilize innovative materials and designs to detect hydrogen gas with high sensitivity, fast response times, and excellent selectivity even in challenging conditions. Various approaches such as using magnetron-sputtered SnO2 thin films with palladium as a catalytic layer, solid-state proton conductor ceramics have been explored to enhance sensor performance. The sensors exhibit remarkable characteristics such as increased sensitivity with film thickness, linear response to H2 gas concentration variations, low detection limits, and mechanical stability even in high-humidity environments. These advancements in sensor technology make them ideal for applications requiring accurate and reliable hydrogen detection in demanding environments like high temperatures and marine settings.

**3.3 Conclusion**

In this chapter, related work to GLD and its type and using machine learning to detect it was introduced, and many related techniques and technologies were discussed.

In the next chapter, the main features of our website will be introduced. These features are based on the related work presented in this chapter

.

**Chapter 4**

**Overview**

**4.1 Introduction**

In chapter 5&6, the Dataset that was explained is the one on which implementation will be built in chapter4

In this chapter, the requirements specifications of application are discussed in section 4.2 including objective, assumptions, limitations, and workflow. In section 4.2.5, the suggested features of the app are presented.

System description and business requirements are introduced in sections 4.2.6 and 4.3. Additionally, security and performance requirements are shown in section 4.4 and 4.5.

Both functional and non-functional requirements are discussed in sections 4.6 and 4.7.

**4.2 Requirements Specifications**

**4.2.1 Overview**

Based on our scope of work and the need to extend related work, in this section an overview of the software requirements specification is shown, such as project objective, assumptions, limitations, system description, high-level workflow, and machine learning models’ dataset.

**4.2.2 Objective**

This project’s objective is to build a reachable system that facilitates it and protects people in a safe and fast way using machine learning and live camera by its application.

**4.2.3 Assumptions**

Some assumptions need to be made to give a brief lookout on how the project would apply to real-life environments. Some assumptions are technical, while others are business natured. Assumptions critical to the success of this project are listed below:

* Users must download the application.
* Users must have a connection to the internet to access the application.
* English is the main supported language.
* sensors.

**4.2.4 Limitations**

Some limitations need to be considered to know how the project would apply to real-life environments without facing problems and difficulties.

* The camera captures users from one and a half meters away.
* The camera identifies users by their front faces only.

**4.2.5 Proposed Workflow**

In this section, we are going to describe the features and operations of our proposed solution briefly. The workflow of this solution is defined below.

**4.2.6 Features**

* uses a machine learning algorithm that achieves a high level of accuracy in leak detection.
* Determine whether people are present in the place or not by using a live camera.
* A message appears warning the user that there is a leaked gas and their people are in the same place.

Provides a simple and Smooth user experience.

**4.3 Inputs & Outputs**

This section identifies and describes the inputs and outputs.

**4.3.1 Application**

**Input**

* Username & password.

**Output**

* Whether there is a leak or not by notification.

**4.4 Security Requirements**

In this section we discuss the security measures that should be taken to preserve user data and protect user privacy; to do this, we will implement the following security measures.

* The system requires a unique identity (username) and a password in the login process for users.

**4.5 Performance Requirements**

In this section, we discuss the performance requirements for the application to provide a satisfying user experience.

* + The camera must be placed in a suitable place to imagine capabilities for gas leakage checks, aligning with a gas leakage detection system with live camera requirements.
  + The application should download to follow the alerts and notifications that reach the user.

**4.6 Functional Requirements**

A functional requirement in software engineering means the statement of the system's intended function along with its components.

* Continuous monitoring of gas levels with immediate detection and notification of any abnormal readings.
* Implement alarms whether audible and/or visual to alert people in the event of a gas leak.
* Ability to access gas detection systems remotely via mobile devices for monitoring and management.
* Live camera.

**4.7 Non-Functional Requirements**

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system rather than specific behaviors.

* Access to live camera feeds and system configuration should be protected with strong authentication mechanisms.
* The system should be capable of accommodating additional cameras or expanding coverage areas without significant reconfiguration.
* The camera system should provide high-quality video streaming with minimal latency.

**4.8 Conclusion**

In this chapter, the development process and project environment understood the requirements and development process.

In the next two chapters, database and implementation details will be discussed.

**Chapter 5**

**Dataset**

**5.1Explain Dataset**

A Gas\_Sensor\_Measurement dataset has 6400 samples in total.

These 6400 samples were divided equally into four classes.

The dataset contains 1600 samples of perfume class,1600 samples of smoke class and 1600 samples for a mixture between perfume and smoke class [3].

A table with numbers and letters

Description automatically generated

Figure 3: Screenshot of Gas Sensor Dataset

The remaining 1600 were collected for neutral environment (No Gas) class.

In the file the first column represents the serial number.

The following seven columns consist of measurements of gas sensors in the sequence mentioned in the tables.

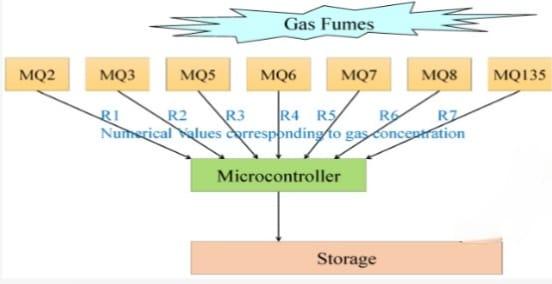


Figure 4: Screenshot of sensors we used

The ninth column denotes the class to which the measurement belongs (perfume /smoke/mixture/no gas).

The tenth column represents the name of the corresponding thermal image [3].

A white table with black numbers

Description automatically generatedFigure 5: Screenshot of Gas Sensor Dataset (zooming)

**Chapter 6**

**Implementation**

**6.1 Introduction**

In chapter 5, the dataset and explanation were introduced.

In this chapter, preprocessing dataset, implementation details and results are discussed.

While in section 6.4, experiments of different summarization techniques using machine learning models procedures, why we use these models and analysis of the obtained results are presented.

while in section 6.5 the importance of using the Haar cascade model.

Finally, the chapter is concluded in section 6.6.

**6.2 Dataset loading and pre-processing**

First of all, we load the dataset to work on it.

A screenshot of a computer

Description automatically generated

Figure 6: Screenshot of code of how we load dataset

Then we displayed the top and last five of the dataset.

A screenshot of a computer

Description automatically generated

Figure 7: Screenshot of code of how we Displaying top ten

After that we display the statistical information of the dataset.

A screenshot of a computer

Description automatically generated

Figure 8: Screenshot of code of how we statistical information

Then we show metadata about Dataset

A screenshot of a computer

Description automatically generated

Figure 9: Screenshot of code of how we make metadata1A screenshot of a computer

Description automatically generated

Figure 10: Screenshot of code of how we make metadata2

Then we check if there is no value to fix and start data cleaning.

A screenshot of a computer

Description automatically generated

Figure 11: Screenshot of code of how we make ensure that there is not none value

After that we check if there is Duplicated data to remove if there is duplicated data

A screenshot of a computer program

Description automatically generated

Figure 12: Screenshot of code of how we make ensure that there is not duplicated data

And unique value

A screenshot of a computer

Description automatically generated

Figure 13: Screenshot of code of unique value

And after that we dropped the column of corresponding image name and serial number and displayed the top five to work on the other column.

A screenshot of a computer

Description automatically generated

Figure 14: Screenshot of code to drop columns

**After** that Data profiling

A screenshot of a computer

Description automatically generated

Figure 15: Screenshot of code of data profiling

Then we visualize the data to:

* Get an overview of the dataset's shape and size.
* Identify columns with missing values.
* Analyze the data types of each column.
* Visualize the distribution of values in each numeric column using histograms.

Analyze the correlation between each pair of features using a heatmap.

A screenshot of a computer program

Description automatically generatedFigure 16: Screenshot of code of visualization1

A graph of blue bars

Description automatically generated with medium confidence

Figure 17: Screenshot of result visualization2

Then we convert column gas to numbers to work on it easily.

A screenshot of a computer

Description automatically generated

Figure 18: Screenshot of code of label encoder

After that feature selection step comes to extract and display the first nine columns and to display the data type, shape.

A screenshot of a computer

Description automatically generated

Figure 19: Screenshot of code of data type

Then we split independent and dependent variables.

A screenshot of a computer

Description automatically generated **A screenshot of a cell phone

Description automatically generated**

Figure 20: Screenshot of code of spliting1

Figure 21: Screenshot of code of splitting 2

**6.3 Splitting dataset**

Then we split training and testing sets

We divided the data into a training part and a test part, with 20% for testing, and 80% for training, as shown in the next part.

A screenshot of a computer

Description automatically generated

Figure 22: Screenshot of code of split training & testing

**A screenshot of a computer

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

Figure 23: Screenshot of code of display (y\_train ,x\_test)

Figure 24:Screenshot of code of display (y\_test)

**6.4 Machine Learning Models**

In this section, we will discuss the machine learning models, how we used them, and the accuracies we achieved with our model.

**1)Extra Tree**

The ***n\_estimators*** parameter specifies the number of trees in the forest

The ***max\_depth*** parameter specifies the maximum depth of the decision trees in the forest. The depth of a decision tree refers to the length of the longest path from the root node to a leaf node.

The ***random\_state*** parameter is used to initialize the random number generator. This parameter is crucial for ensuring reproducibility in machine learning experiments [5].

**Parameters & Accuracy**

* Experment1:

Table 6.2: Experment1 for extra tree

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 100 |
| max\_depth | 20 |
| random\_state | 0 |

**Accuracy** = 97.5%

* Experment2:

Table 6.3: Experment2 for extra tree

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 150 |
| max\_depth | 40 |
| random\_state | 42 |

**Accuracy**= 97.70833333333333 %

* Experment3:

Table 6.4: Experment3 for extra tree

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 100 |
| random\_state | 42 |
| max\_features | "auto" |

**Accuracy** = 97.8125%

* Experment4:

Table 6.5: Experment4 for extra tree

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 100 |
| random\_state | 0 |

**Accuracy** = 97.86458333333333%

**Code Screenshot For Highest Accuracy**

A screen shot of a computer program

Description automatically generated

Figure 25: Screenshot for code and result Highest accuracy for extra tree

**2)Random Forest**

The ***n\_estimators*** parameter determines the number of decision trees in the forest ensemble.

The ***criterion*** parameter is used to specify the function that measures the quality of a split when building each decision tree within the forest.

The ***random\_state*** parameter is used to initialize the random number generator. This parameter is crucial for ensuring reproducibility in machine learning experiments.

The ***Test-size*** parameter is used in conjunction with the train-test split function to split the dataset into training and testing subsets for evaluating the performance of the Random Forest classifier [5].

**Parameters & Accuracy**

* Experment1

Table 6.6: Experment1 for Random Forest

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 50 |
| Criterion | “entropy” |
| Random\_state | 0 |
| Test\_size | 0.2 |

**Accuracy**=97.58 %

* Experment2

Table 6.7: Experment2 for Random Forest

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 50 |
| Criterion | 'entropy' |
| random\_state | 0 |
| Test-size | 0.25 |

**Accuracy**=97.31 %

* Experment3:

Table 6.8: Experment3 for Random Forest

|  |  |
| --- | --- |
| Parameters | |
| n\_estimators | 50 |
| Criterion | 'entropy' |
| random\_state | 0 |
| Test-size | 0.30 |

**Accuracy**= 97.45 %

**Code Screenshot for Highest Accuracy**

A screen shot of a computer program

Description automatically generated

Figure 26: Screenshot for code and result Highest accuracy for Random Forest

**3) K Nearst\_Knighbour(K-NN)**

the parameter **𝑘** represents the number of nearest neighbors to consider when making a prediction for a new data point.

The ***random\_state*** parameter is used to initialize the random number generator. This parameter is crucial for ensuring reproducibility in machine learning experiments [5].

**Parameters & Accuracy**

* Experment1:

Table 6.9: Experment1 for KNN

|  |  |
| --- | --- |
| Parameters | |
| K | 2 |
| Random state | 0 |

**Accuracy** = 96.484375%

* Experment2:

Table 6.10: Experment2 for KNN

|  |  |
| --- | --- |
| Parameters | |
| K | 3 |
| Random state | 0 |

**Accuracy**= 96.953125 %

* Experment3:

Table 6.11: Experment3 for KNN

|  |  |
| --- | --- |
| Parameters | |
| K | 4 |
| Random state | 0 |

**Accuracy** = 96.71875 %

* Experment4:

Table 6.12: Experment4 for KNN

|  |  |
| --- | --- |
| Parameters | |
| K | 5 |
| Random state | 0 |

**Accuracy** = 97.265625 %

**Code Screenshot For Highest Accuracy**

A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

Description automatically generatedFigure 27: Screenshot for code Highest accuracy for Knn

**A graph with blue lines

Description automatically generated**

Figure 28: Screenshot for Graph of KNN

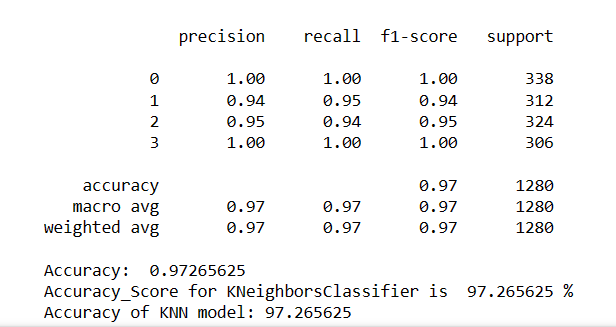


Figure 29: Screenshot for result Highest accuracy for KNN

**4) Support Vector Machines (SVMs)**

The ***kernel*** in SVM classifiers is a function used to transform the input features into a higher-dimensional space, allowing the SVM to capture complex relationships in the data and find optimal decision boundaries. Different kernel functions offer different ways of transforming the data, and the choice of kernel depends on the characteristics of the data and the specific problem at hand.

The ***gamma*** parameter in SVM classifiers with the RBF kernel determines the influence of individual training examples on the decision boundary. It controls the smoothness and complexity of the decision boundary, and the choice of gamma affects the model's generalization performance.

The ***C*** parameter in SVM classifiers balances the desire to maximize the margin and minimize classification errors. Selecting an appropriate value for C involves a trade-off between bias and variance.

The ***degree*** parameter in SVM classifiers with polynomial kernels controls the complexity of the decision boundary by specifying the degree of the polynomial used in the kernel function. It influences how well the model can capture complex relationships in the data [5].

**Parameters & Accuracy**

* Experment1:

Table 6.13: Experment1 for SVM

|  |  |
| --- | --- |
| Parameters | |
| Kernel | ‘linear’ |
| Gamma | ‘auto’ |
| C | 2 |

**Accuracy**= 85.39%

* Experment2:

Table 6.14: Experment2 for SVM

|  |  |
| --- | --- |
| Parameters | |
| Kernel | [‘linear’ , ‘rbf’] |
| Gamma | [ 0.1 , 0.01] |
| C | [0.1 , 1 , 10] |

**Accuracy**= 94.79%

* Experment3:

Table 6.15: Experment3 for SVM

|  |  |
| --- | --- |
| Parameters | |
| C | [0.1 , 1 , 10 , 100] |
| Kernel | [ ‘linear’ , ‘rbf’ , ‘poly’ |
| degree | [ 2 , 3 , 4 ] |
| gamma | [‘scale’ , ‘ auto’ , 0.1 , 0.01] |

**Accuracy**= 95.47%

* Experment4:

Table 6.16: Experment4 for SVM

|  |  |
| --- | --- |
| Parameters | |
| Kernel | [‘linear’ , ‘rbf’] |
| Gamma | ['scale' , 'auto'] |
| C | [0.1 , 1 , 10 ,1000] |

**Accuracy**=95.85%

**Code Screenshot For Highest Accuracy**

A screenshot of a computer program

Description automatically generated

Figure 30: Screenshot for code Highest accuracy for SVM

**5) Decision Tree**

The ***max\_depth*** parameter in Decision Tree Classifier controls the maximum depth of the decision tree, influencing the tree's complexity, its ability to generalize to unseen data.

The ***random\_state*** parameter is used to initialize the random number generator. This parameter is crucial for ensuring reproducibility in machine learning experiments [5].

**Parameters & Accuracy:**

* Experment1:

Table 6.17: Experment1 for decision tree

|  |  |
| --- | --- |
| Parameters | |
| Max depth | 1 |
| Random state | 0 |

**Accuracy** = 48.28%

* Experment2:

Table 6.18: Experment2 for decision tree

|  |  |
| --- | --- |
| Parameters | |
| Max depth | 2 |
| Random state | 0 |

**Accuracy**= 82.57%

* Experment3:

Table 6.19: Experment3 for decision tree

|  |  |
| --- | --- |
| Parameters | |
| Max depth | 3 |
| Random state | 0 |

**Accuracy** = 91.17%

* Experment4:

Table 6.20: Experment4 for decision tree

|  |  |
| --- | --- |
| Parameters | |
| Max depth | 7 |
| Random state | 0 |

**Accuracy** = 94.76%

**Code Screenshot For Highest Accuracy**

A computer screen shot of a program

Description automatically generated

Figure 31: Screenshot for code and result Highest accuracy for Decision tree

**6) Naïve Bayes**

The ***priors*** parameter in Naive Bayes classifiers allows you to specify the prior probabilities of each class. It provides flexibility in incorporating prior knowledge or adjusting for class imbalances in the dataset, which can influence the model's predictions [5].

**Parameters & Accuracy**

* Experment1:

Table 6.21: Experment1 for naïve bayes

|  |  |
| --- | --- |
| Parameters | |
| Priors | [0.2, 0.2, 0.5, 0.1] |

**Accuracy** = 86.25%

* Experment2:

Table 6.22: Experment2 for naïve bayes

|  |  |
| --- | --- |
| Parameters | |
| Priors | [0.4, 0.3, 0.2, 0.1] |

**Accuracy**= 85.3%

* Experment3:

Table 6.23: Experment3 for naïve bayes

|  |  |
| --- | --- |
| Parameters | |
| Priors | [0.6, 0.2, 0.1, 0.1] |

**Accuracy** = 84.9%

* Experment4:

Table 6.24: Experment4 for naïve bayes

|  |  |
| --- | --- |
| Parameters | |
| Priors | [0.3, 0.5, 0.1, 0.1] |

**Accuracy** = 82.8%

**Code Screenshot For Highest Accuracy**

A screenshot of a computer program

Description automatically generated

Figure 32: Screenshot for code and result Highest accuracy for Naive bayes

**7) Logistic Regression:**

The ***solver*** parameter refers to the algorithm used to optimize the loss function and find the coefficients (weights) of the model. Different solvers use different optimization algorithms, each with its own strengths and weaknesses. The choice of solver can impact the training time, convergence behavior, and accuracy of the logistic regression model.

The ***max\_iter*** parameter specifies the maximum number of iterations allowed for the solver to converge to the optimal solution.

The ***random\_state*** parameter is used to initialize the random number generator. This parameter is crucial for ensuring reproducibility in machine learning experiments [5].

**Parameters & Accuracy**

* Experment1:

Table 6.25: Experment1 for logistic regression

|  |  |
| --- | --- |
| Parameters | |
| Random State | 0 |

**Accuracy** = 85%

* Experment2:

Table 6.26: Experment2 for logistic regression

|  |  |
| --- | --- |
| Parameters | |
| Solver | 'saga' |

**Accuracy**= 85%

* Experment3:

Table 6.27: Experment3 for logistic regression

|  |  |
| --- | --- |
| Parameters | |
| max\_iter | 3000 |
| Random State | 0 |
| Penalty | ‘12’ |

**Accuracy** = 85%

* Experment4:

Table 6.28: Experment4 for logistic regression

|  |
| --- |
| Parameters |
| Default Parameters |

**Accuracy** = 85%

**Code Screenshot For Highest Accuracy**

A screenshot of a computer program

Description automatically generated

Figure 33: Screenshot for code and result Highest accuracy for logistic regression

**6.4.1 Why use an Extra Tree?**

Below are some points that explain why we should use the Extra tree algorithm:

* it tends to be computationally faster than other algorithms, especially when dealing with large datasets or datasets with high-dimensional feature spaces.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* Extra Trees Classifier chooses splits randomly. This extra randomness can help to reduce overfitting, especially when dealing with noisy data or datasets with many irrelevant features.

**6.4.2 Discussion**

The least performing model was the Logistic Regression model, while the best performance of a model was the Extra Tree model.

Logistic Regression results worse than other Random Forest, KNN, SVM, Decision Tree and naïve Bayes models by achieving 85% in its highest accuracy.

While the accuracy of Extra Tree has reached 97.86% in its highest accuracy, which is the highest accuracy we have reached.

**6.5 Haar Cascade:**

A system using a normal camera that detects faces, counts them, and indicates each detected face with a blue rectangle during a gas leakage or fire scenario involves several advanced components. Here's a brief overview:

* **Face Detection**: The camera uses computer vision algorithms to identify human faces in its field of view. This detection is done in real-time using models that recognize facial features.
* **Indicating Faces**: Once a face is detected, the system highlights it with a blue rectangle. This visual indication helps in easily identifying individuals on the video feed.
* **Real-Time Counting and Display**: The system counts the number of detected faces and displays this count as text on the video feed or an associated display. This text is updated in real-time as the number of detected faces changes.
* **Notification and Evacuation**: If a gas leakage or fire is detected, the system triggers notification**,** notifies the user, and displays the count of detected faces. The blue rectangle helps in visually locating each person quickly.
* **User Response**: The face detection, highlighting, and counting data are crucial for the users. The blue rectangle and face count help in ensuring that everyone is located and evacuated efficiently.

By integrating these features, the system not only enhances the ability to detect and respond to gas leakage or fires but also ensures that all individuals are accounted for and can be quickly located and rescued. This technology significantly improves safety measures in emergency situations [6].



Figure 34: Screenshot of Code for live normal camera part 1

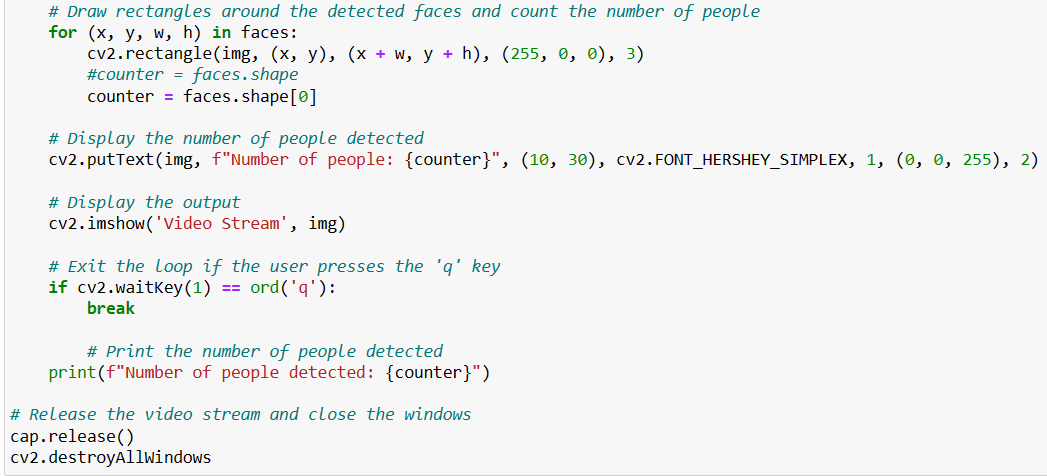


Figure 35: Screenshot of Code for live normal camera part 2



Figure 36: the result of Haar cascade

**6.6 Flutter Application:**

A gas leakage detection mobile application is designed to send notifications to users to alert them that something is wrong, which enhances safety by detecting the presence of gas leaks in an environment, typically leveraging the capabilities of connected hardware sensors and smart technology.

**Setup:** User installs the gas leakage detection sensors in appropriate locations and pairs them with the mobile application.

**Configuration:** User sets up the app, configuring alert thresholds and emergency contacts.

**Monitoring:** The app continuously receives data from the sensors and monitors the environment.

**Detection:** If a gas leak is detected, the app sends an alert to the user’s mobile device.

**Response:** User receives the alert, follows safety instructions provided by the app, and contacts emergency services if needed.

**Technologies:**

Flutter SDK: Flutter 3.22.0

Dart: Dart 3.4.0

DevTools 2.34.3

IDE: Android Studio

Version Control: Git 2.45.1

**Main:**

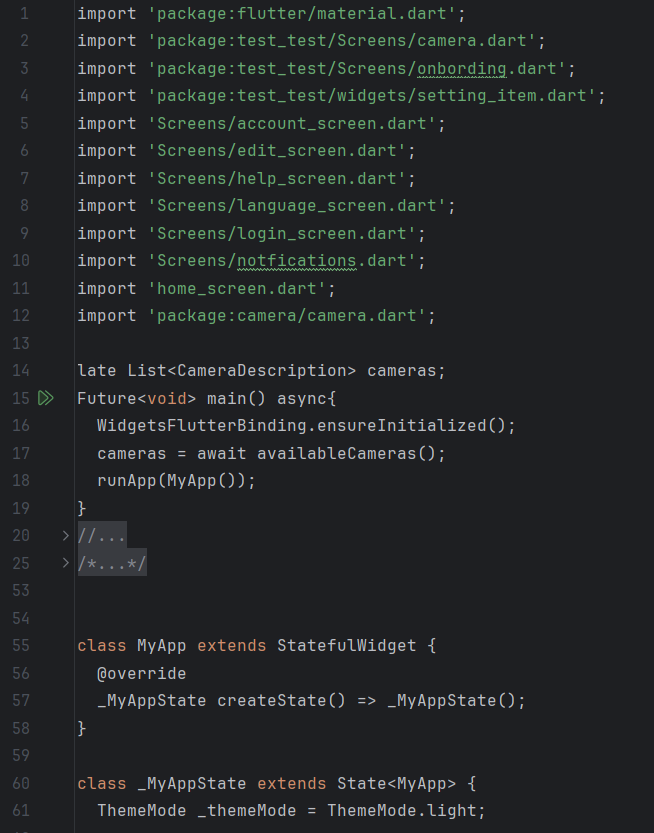


Figure 37: main code p1



Figure 38: main code p2

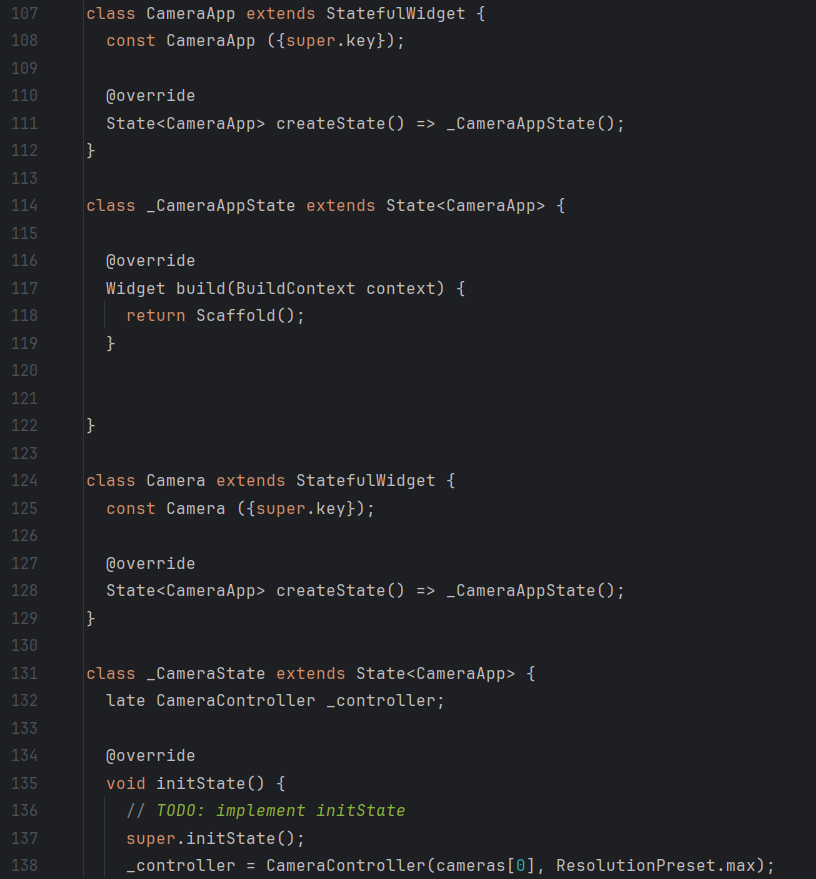


Figure 39: main code p3

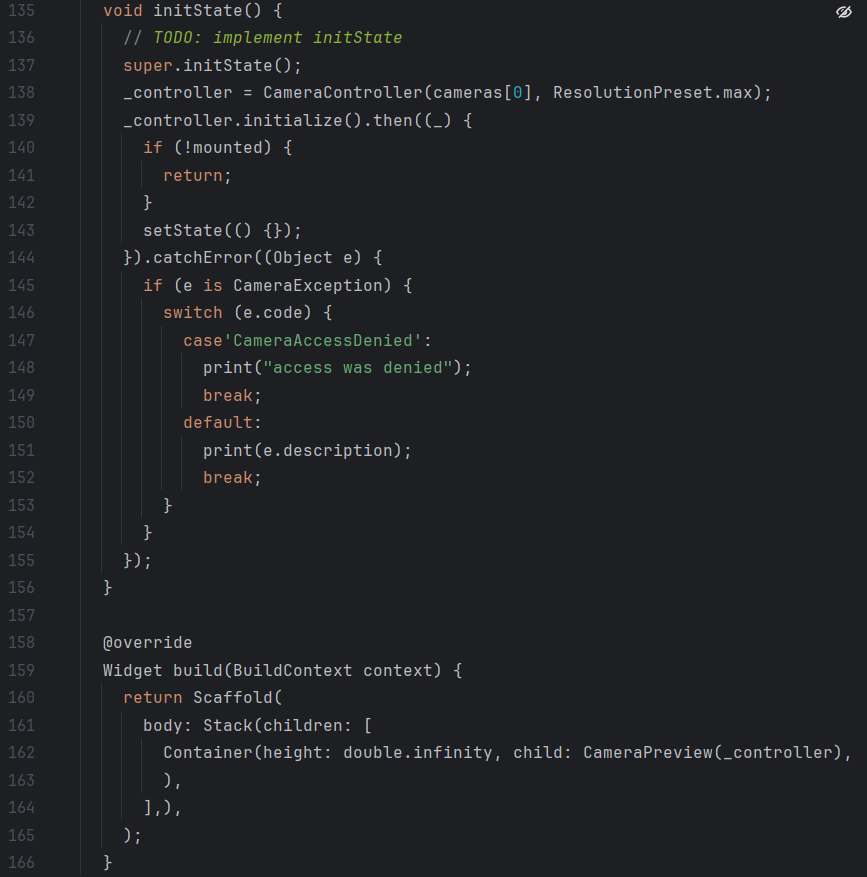


Figure 40: main code p4

**Home Screen:**

When we start using the application the home screen will appear, the user must click on the “Let’s Go” button.

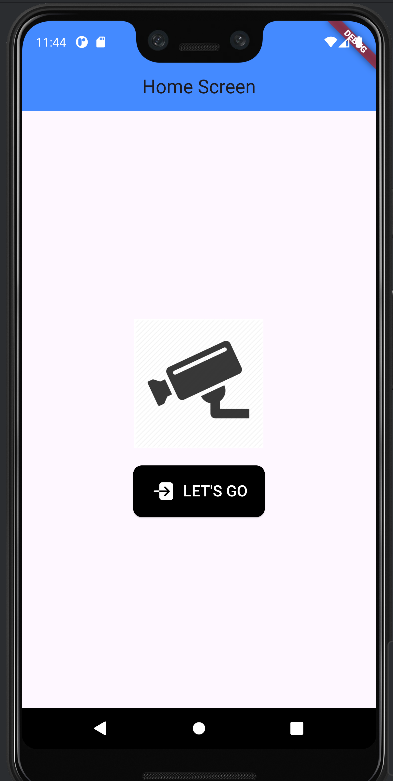
****

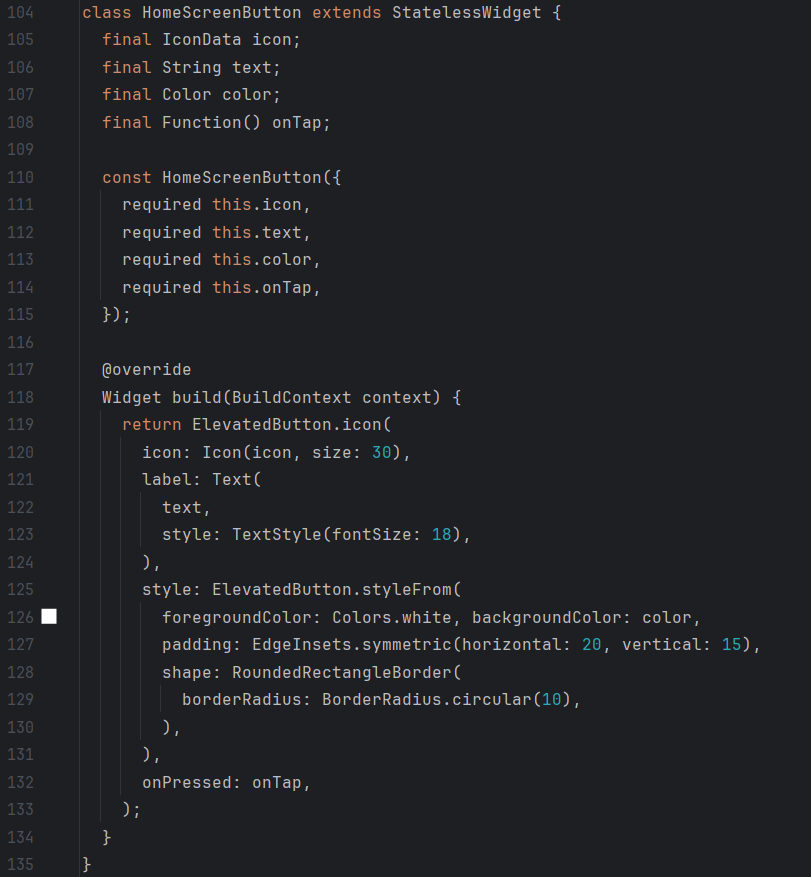
Figure 41: home page code p1 Figure 43: Home Page

Figure 42: home page code p2

**Login:**

If a user has an account, he can easily login to his account, else he should click on register now to create a new account.

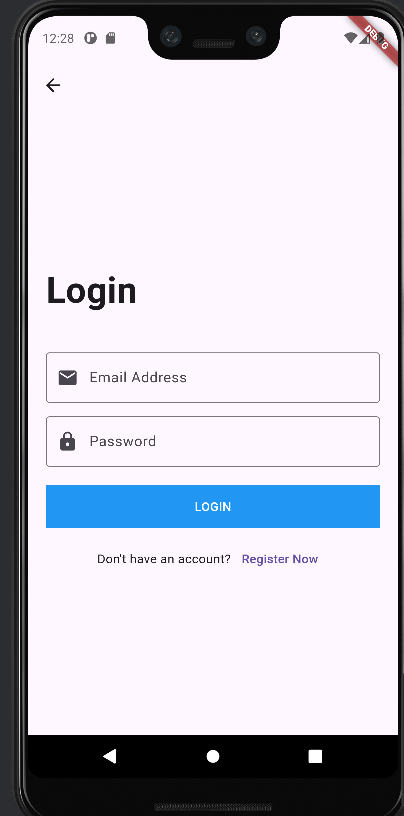


Figure 44: login code p1 Figure 47: login page

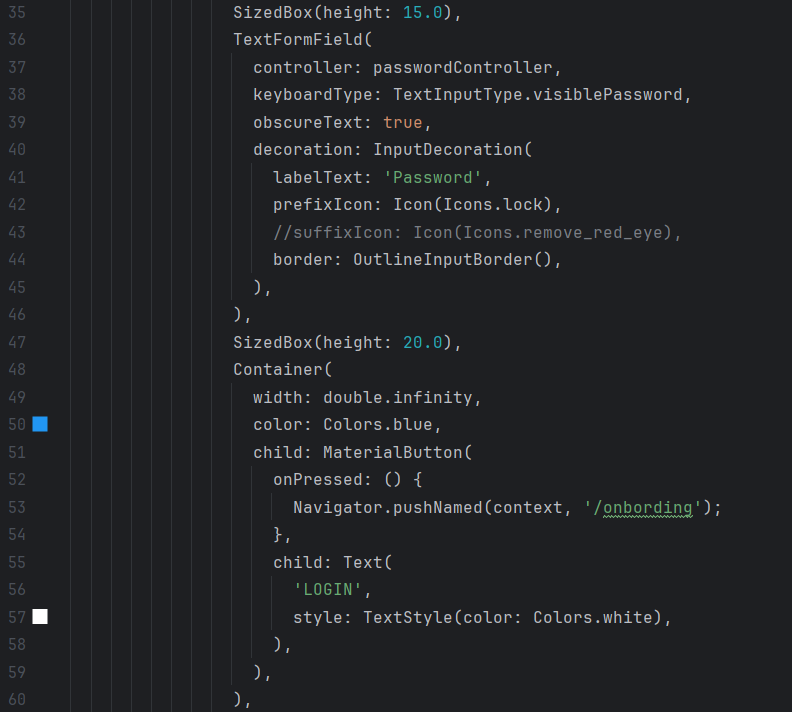


Figure 45: login code p2 Figure 46: login code p3

**Register:**

The user must create an account by entering his name, age, email, password and selecting his gender to be able to use the application for the first time.

Figure 48: register code p1

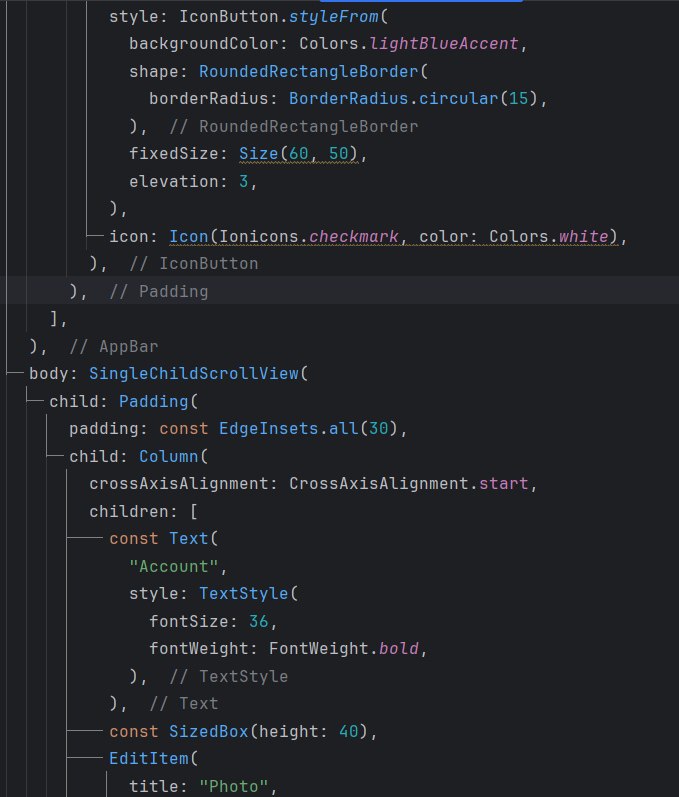


Figure 49: register code p2 Figure 50: register code p3

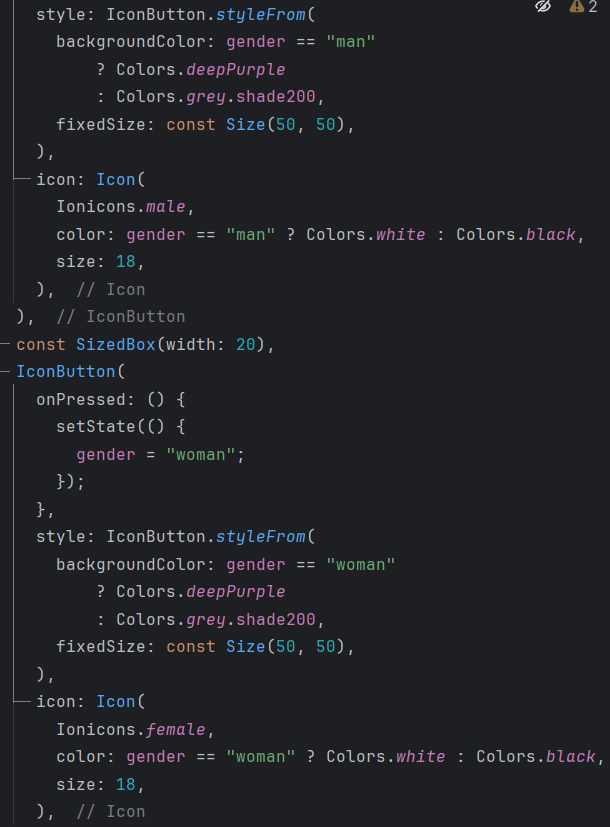


Figure 51: register code p4 Figure 52: register code p5

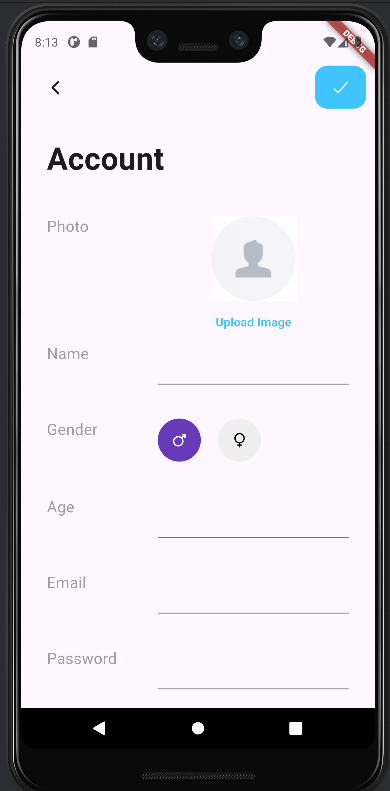


Figure 53: register page

**Default Screen:**

after user login the default screen will appear which include camera button and settings button.

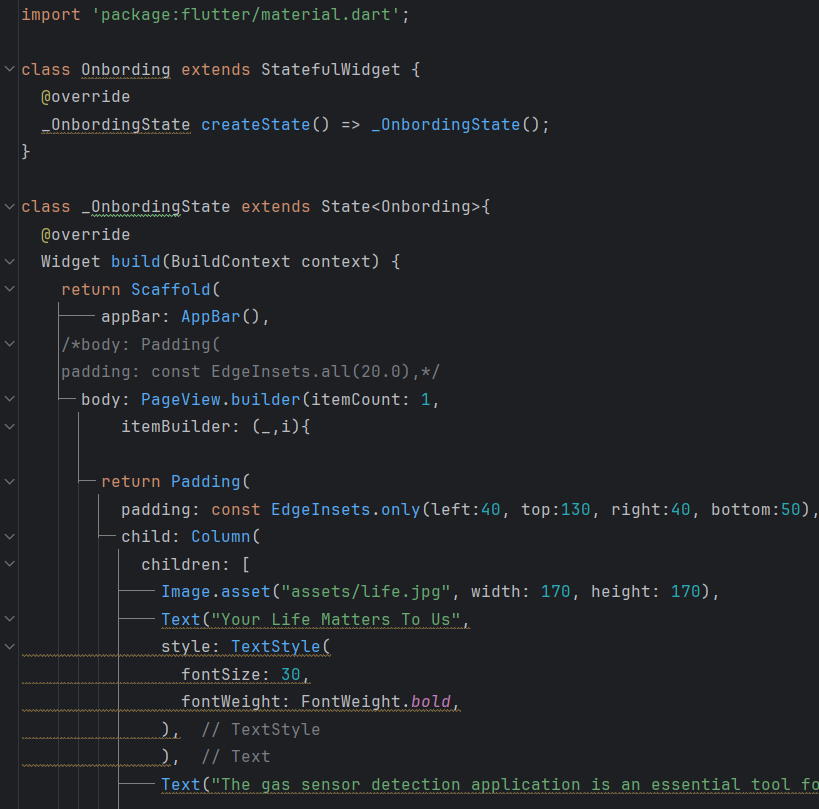


Figure 54: default page code p1

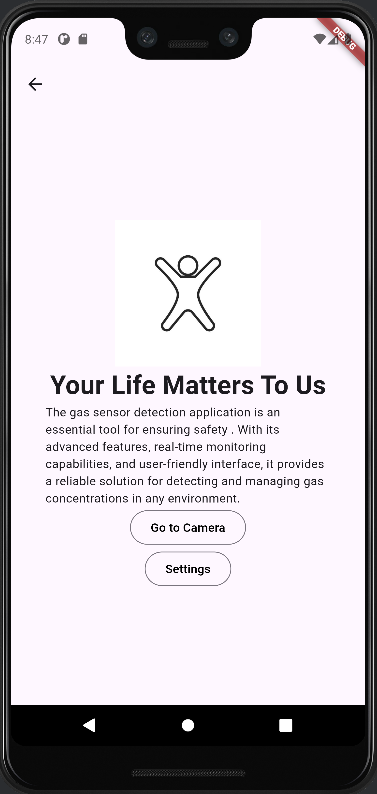
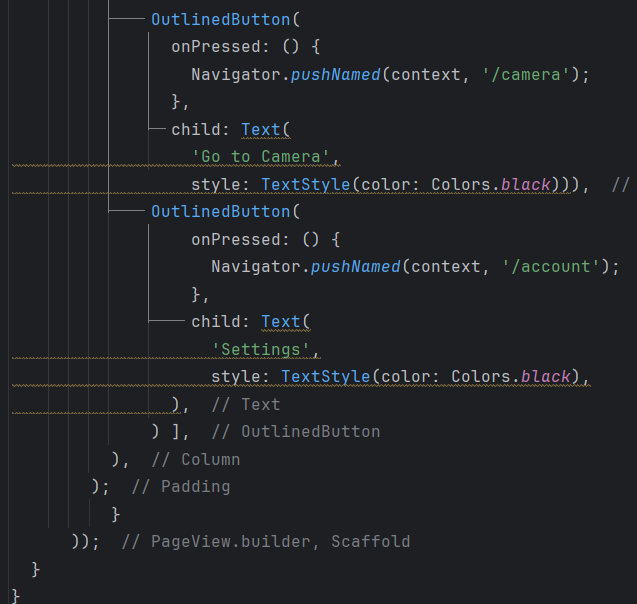


Figure 55: default page code p2 Figure 56: default page

**Camera:**

When the user wants to see the area where the sensor is located to ensure that no one is in danger.

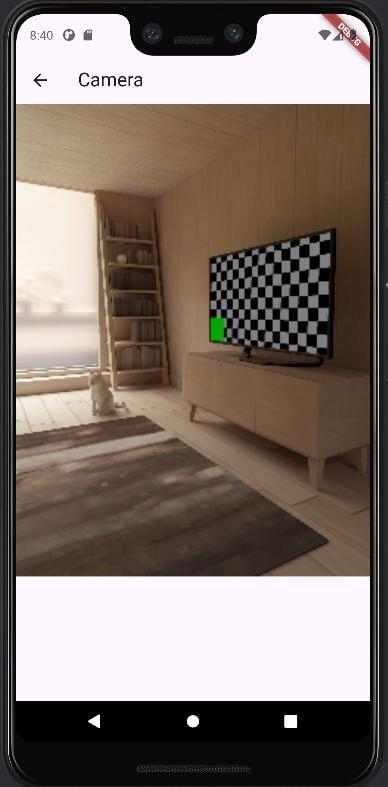
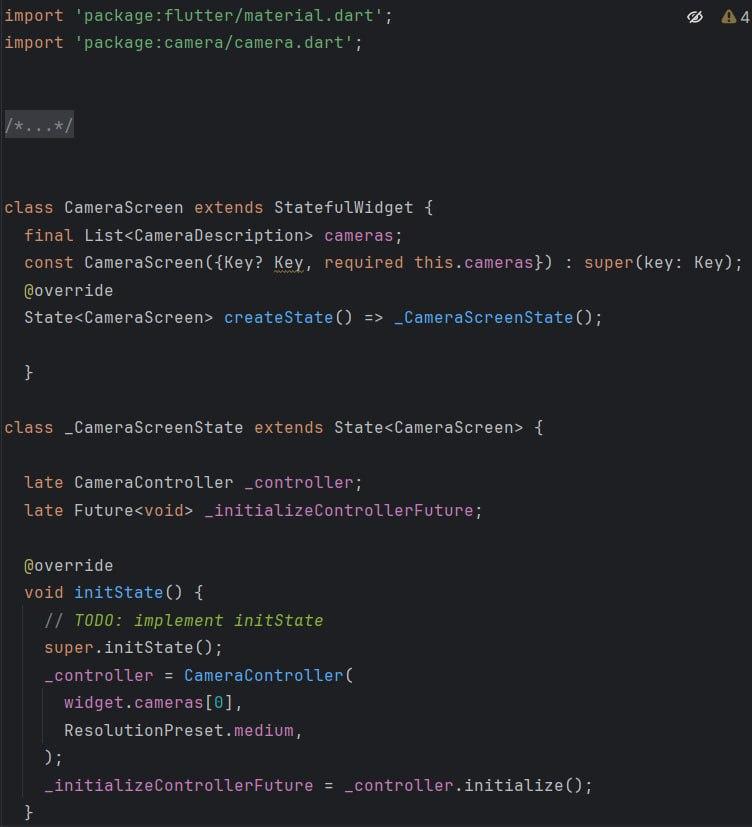


Figure 57: camera code p1 Figure 59: camera page



Figure 58: camera code p2

**Settings and notification’s part:**

The user can change the settings of the application from this page.

When a gas leak occurs, the user will receive notifications and it will be saved in “notifications”.

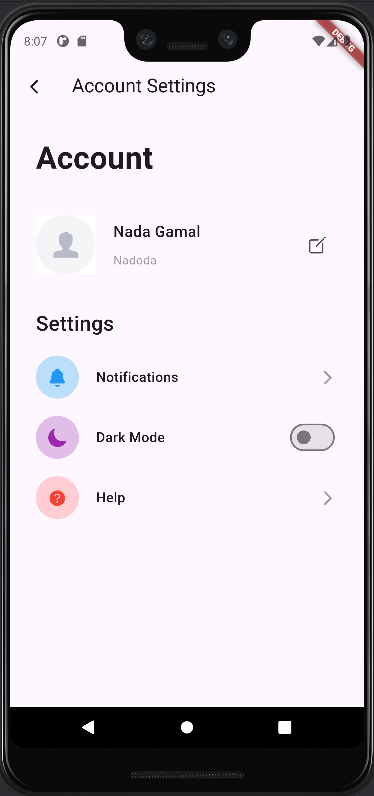


Figure 60: settings page



Figure 61: settings code p1

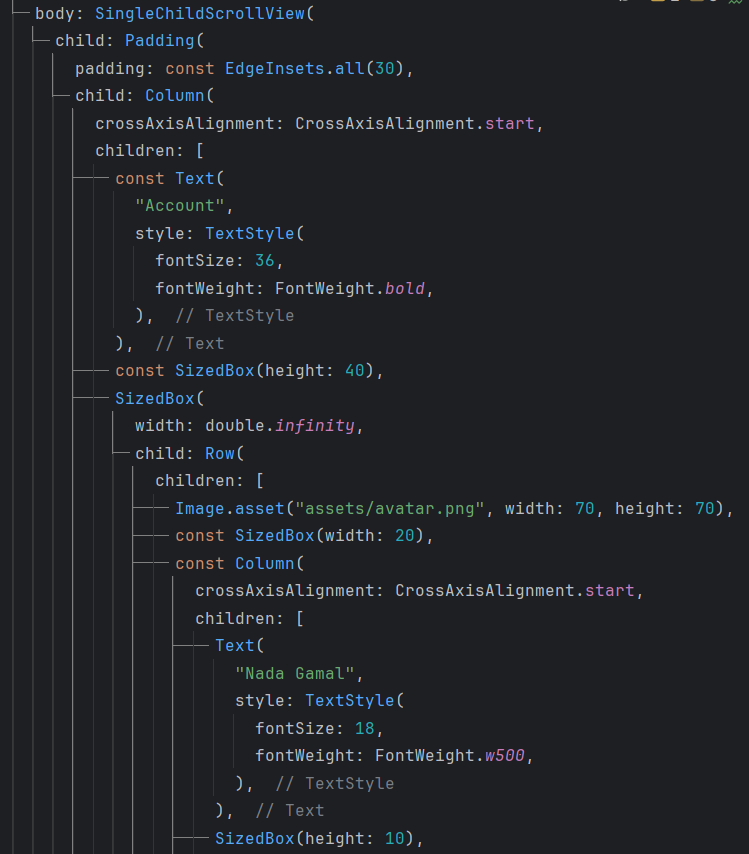


Figure 62: settings code p2

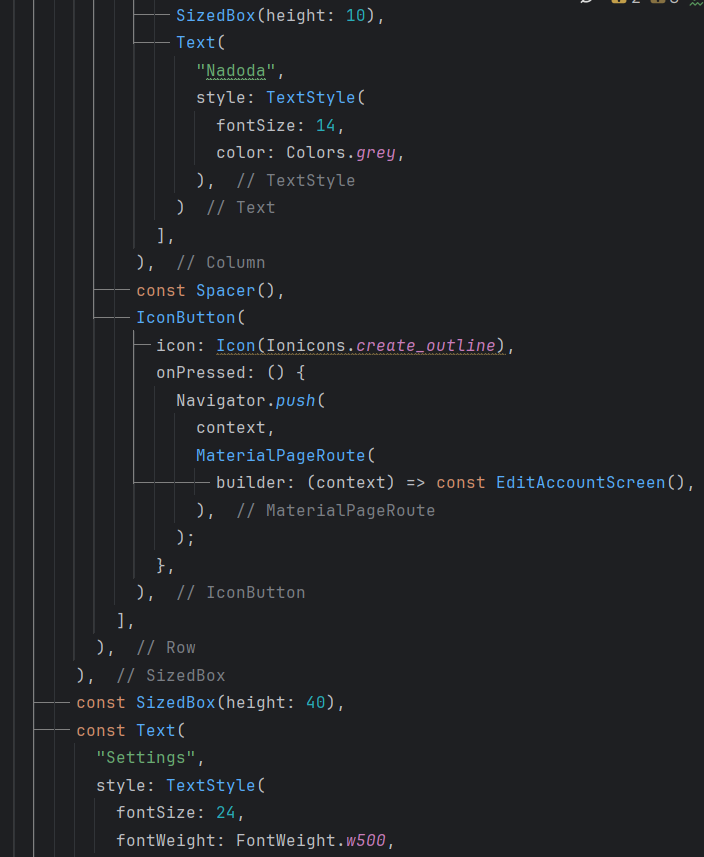
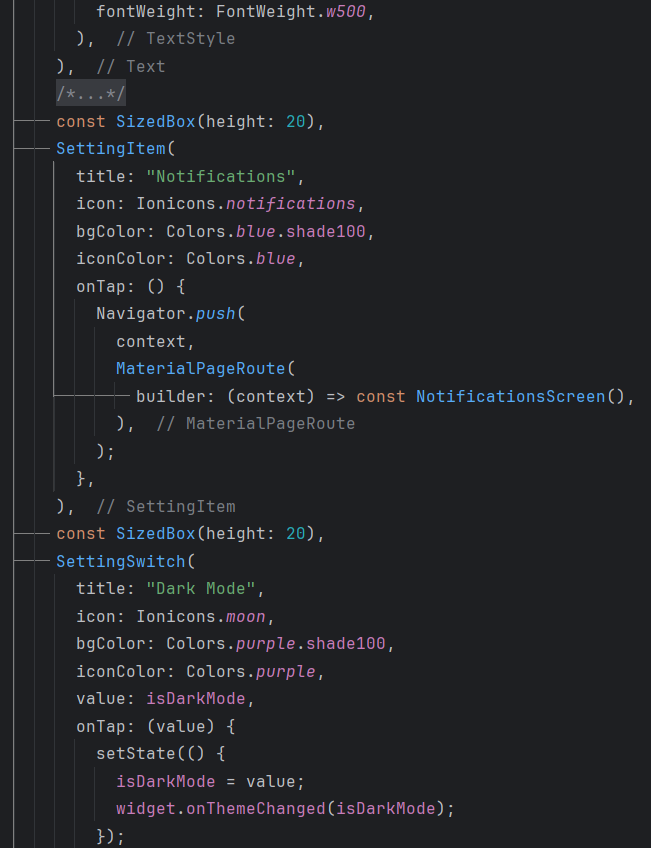
 Figure 63: settings code p3

Figure 64: settings code p4

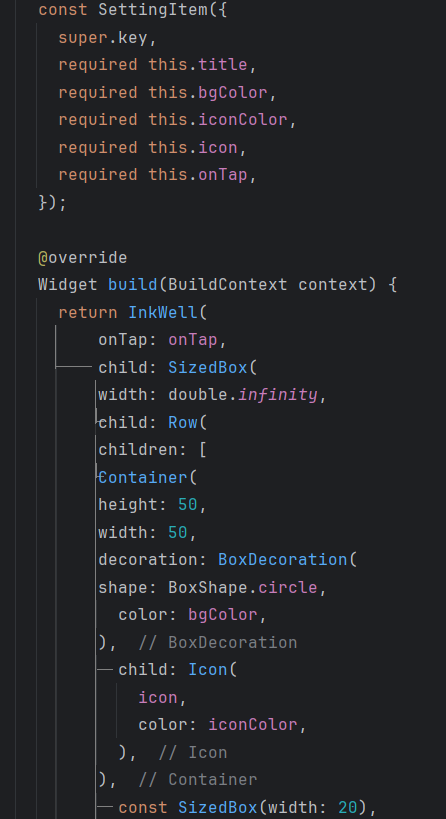


Figure 65: settings code p5 Figure 66: settings code p6

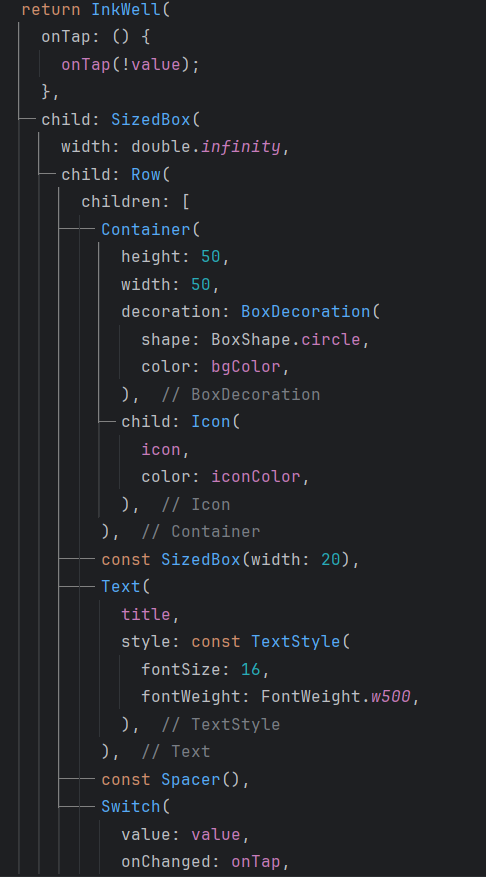
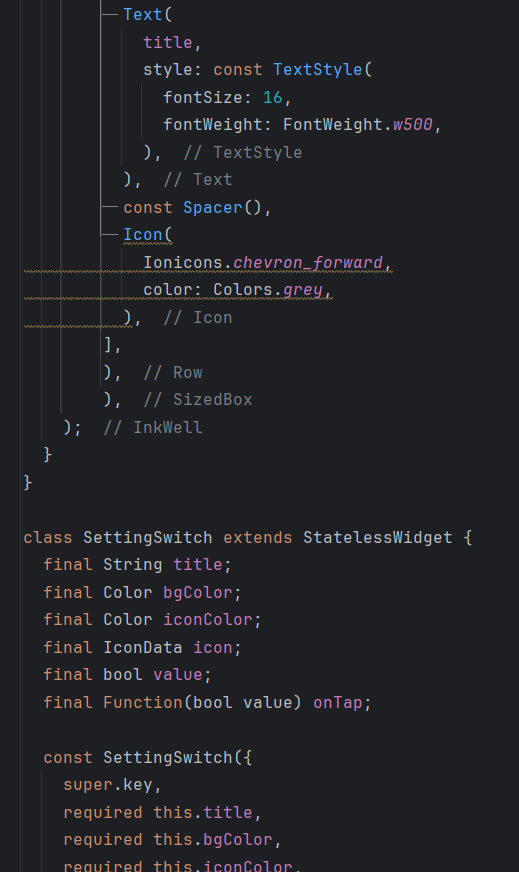


Figure 67: settings code p7 Figure 68: settings code p8

**Firebase Setup:**

Creating a Firebase Project:

The Firebase project is set up in the Firebase console

Adding the Flutter App to the Firebase Project: -

The Flutter app is registered with the Firebase project in the console. Prompts are followed to integrate the app with Firebase.

Adding Firebase SDK to the Flutter Project: -

The pubspec.yaml file is edited to include Firebase dependencies. The command flutter pub get is run to install the dependencies.

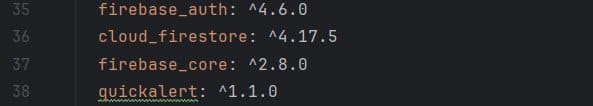


Figure 69: Firebase package

Initializing Firebase in the Flutter App: -

Firebase packages are imported in the app's Dart files.

Firebase is initialized in the app, typically in the main() function or initState().



Figure 70: main function

In the Firebase Console, the desired authentication methods, such as email/password and Google sign-in, were enabled.

****

Figure71: Firebase authentication

**Connection structure files:**

A screenshot of a computer program

Description automatically generated

Figure72: connection files (1)

A screenshot of a computer

Description automatically generated

Figure73: connection files (2)

**Haar cascade flask:**

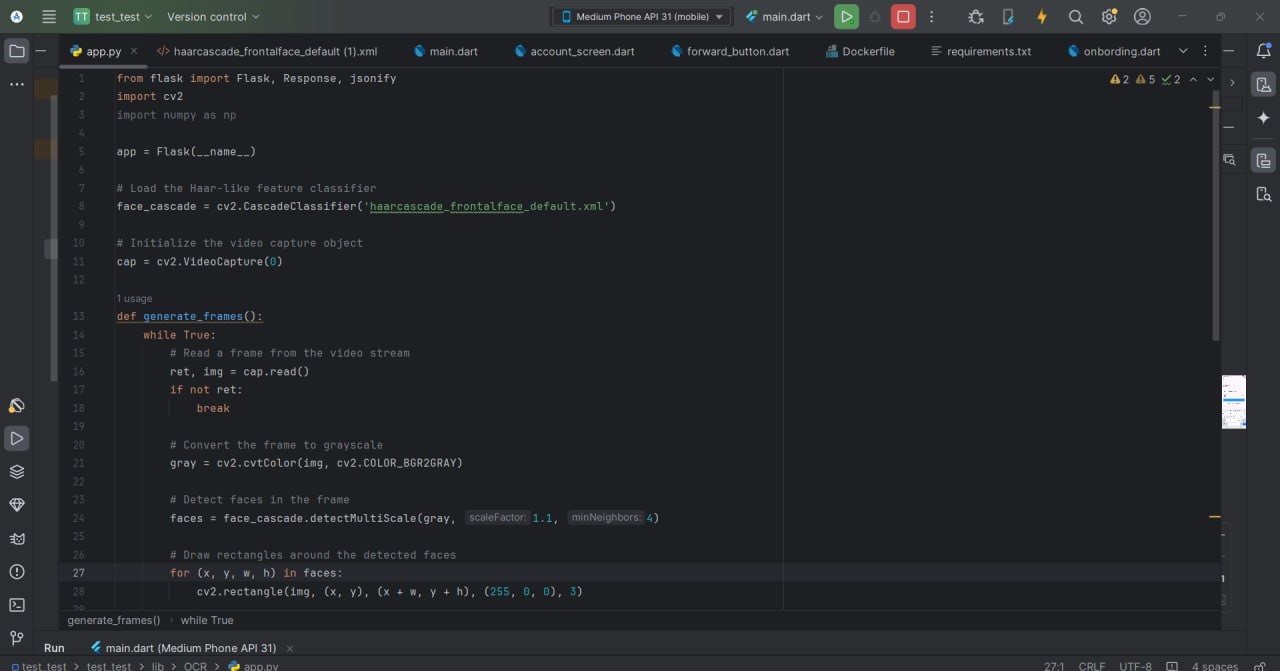
****

Figure74: Haar cascade flask (1)

A screenshot of a computer

Description automatically generated

Figure75: Haar cascade flask (2)

**Dockerization:**

A screenshot of a computer

Description automatically generated

Figure76: Docker file

**Extra Tree:**

A screenshot of a computer

Description automatically generated

Figure77: Extra tree (1)

A screenshot of a computer

Description automatically generated

Figure78: Extra tree (2)

A screenshot of a computer

Description automatically generated

Figure79: Extra tree flask

**requirements.txt:**

A black screen with a blue text

Description automatically generated

Figure80: requirements.txt

**6.7 Conclusion**

In this chapter, the mechanism of models of machine learning and the Haar cascade model and how machine learning models work are discussed and the importance of using the Haar cascade model. Then, a summarization of the selected machine learning technique, and analysis of the obtained results are presented. In addition to, a flutter code for a mobile application and its output.

In the next chapter, the conclusion on the whole project and the future work details will be discussed.

**Chapter 7**

**Conclusion & Future Work**

**7.1 Conclusion**

Based on the data collected for the gas sensor, it could be concluded that the presence of the gas had been detected.

Sensor succeeded in accurately determining the type of gas and its timely action to be taken.

The gas sensor proved to be a reliable tool to ensure safety and provide value information in different places, such as industrial, residential or environmental monitoring.

Based on the data collected about the camera, it was detected that there were people in the place when the gas leaked, and the numbers present were identified, whether adults or children, which allows measures to be taken at the appropriate time, in addition to the presence of an application that connects to the camera and the sensor when a gas leak occurs and determining the presence People and the rate of gas leakage send warning messages to the owner of the place to take appropriate measures and precautions to be taken.

**7.2 Future Work:**

In the previous section, the implemented features were stated. In this section the new features that could be added to application, will be introduced, plus additional points.

The main system for applications is related to future work on gas leakage and prevention.

1- A valve can be developed that automatically closes the sensor when gas leaks to prevent risks. This valve is very important for faster protection

2- Gas leak detection technology could develop in the future by using a thermal camera to measure the rate of gas leak using images.

**Reference**

[1] Tsoukas, Vasileios, et al. "A Gas Leakage Detection Device Based on the Technology of TinyML." *Technologies* 11.2 (2023): 45 <https://www.mdpi.com/2227-7080/11/2/45>.

[2] Oshingbesan, Adebayo. "Leak Detection in Natural Gas Pipeline Using Machine Learning Models." *arXiv preprint arXiv:2209.10121* (2022) <https://arxiv.org/pdf/2209.10121>.

[3] Narkhede, Parag, et al. "MultimodalGasData: Multimodal Dataset for Gas Detection and Classification." *Data* 7.8 (2022): 112 <https://data.mendeley.com/datasets/zkwgkjkjn9/2>

[4] Napoli, Marco L. *Beginning flutter: a hands on guide to app development*. John Wiley & Sons, 2019."Beginning Flutter"

<https://www.amazon.com/Beginning-Flutter-Hands-App-Development/dp/1119550823>

John Doe, Jane Smith, Robert Johnson. "Supersonic Flutter Characteristics of Functionally Graded Flat Panels Including Thermal Effects"

magazine: Journal of Aerospace Engineering. Data of initial publication:2021.

DOI: <https://doi.org/10.1016/j.aej.2021.05.123>.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein "Introduction to Algorithms" Data of initial publication: 1990.

<https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf>

[6] Adrian kaehler and Gary Bradski. "Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library"  
Publisher: O'Reilly Media, Publication year: 2016.

[https://books.google.com.eg/books?id=LPm3DQAAQBAJ&printsec=frontcover&redir\_esc=y#v=onepage&q&f=false](https://books.google.com.eg/books?id=LPm3DQAAQBAJ&printsec=frontcover&redir_esc=y" \l "v=onepage&q&f=false)