



Report “SmartMethane”


Authors: Aina Nurgaliyeva & Raziya Abutalipova

GitHub:

<https://github.com/rosiettan/SmartMethane>

Google Colab:

Google Colaboratory

 https://colab.research.google.com/drive/1ZySJBybNnubR_zHyzMFxcWVcmIMWx4Qw?usp=sharing



Video on YouTube: <https://youtu.be/UbLPQ5otoyA>

1.Introduction

1.1 ■ *Problem*

Detecting explosives and combustibles in the air is important in many industries, such as mining, chemical manufacturing, and oil and gas exploration. These materials can cause malfunctions and accidents, damage infrastructure and pose serious health and safety risks to workers and surrounding communities.

SmartMethane - is a website that tries to address these issues by implementing deep learning techniques. By analyzing air samples, program detect the presence of explosives like methane and give workers early warning before an explosion occurs. This early warning system helps prevent serious accidents and save lives. It not only provides an early warning system, but also helps mining and other industries comply with regulations on methane concentration and explosion protection. SmartMethane monitors methane levels and provides real-time data to help companies comply with safety regulations and avoid potential fines and legal obligations.

By detecting potential hazards early, the system can prevent costly equipment, infrastructure damage, and unplanned downtime from reducing productivity. It can also store methane level data over time, allowing historical data analysis. By analyzing this data, the industry can identify trends and patterns in methane levels, inform future safety measures, and prevent accidents before they occur.

Overall, implementing SmartMethane's deep learning technology has the potential to significantly improve safety, prevent accidents, save money, and protect the health and well-being of workers and surrounding communities in many industries.

1.2. ■ Literature review with links (another solutions)

Deep learning techniques have also been proposed for detecting explosive gases in the air. For example, a recent study by Wei et al. (2021) proposed a deep learning-based approach for detecting methane leaks in underground coal mines. The approach involved training a convolutional neural network (CNN) on a dataset of methane concentration images captured by a thermal camera. The approach was found to be highly accurate in detecting methane leaks in real-time. [link](#)

In addition to these solutions, several other technologies have been proposed for detecting explosive gases in the air, including infrared sensors, catalytic sensors, and electrochemical sensors. However, the effectiveness of these technologies depends on various factors such as the type of gas being detected, the concentration of the gas, and the environmental conditions. [link](#)

In general, the literature suggests that there are several effective solutions for detecting explosive gases in air. However, choosing the best solution depends on the specific needs of the industry and the environment in which the solution will be

used. A SmartMethane approach using deep learning techniques is a promising solution, and its effectiveness can be evaluated through further research and testing.

1.3. ■ Current work (description of the work)

The current work aims to develop a model using recurrent neural networks (RNN), convolutional neural networks (CNN), and the TensorFlow library. The data for the model will be collected from sensors placed in mines to detect methane gas levels. The collected data will be analyzed, and visualizations will be created to understand the patterns in the data.

1.3.1. Creating the model for predicting methane emission

1.3.1.1. Creating a dataset

We take a dataset from Kaggle and normalize it using Power BI application. Drop unnecessary data, use unpivot function to separate data. Instead of several columns for each year from 1990-2018 we make columns Year and Value. After that we made some small modifications like replacing word “year” (e.g. “year2015” become 2015) and changing data types.

1.3.1.2. Import libraries

The code for deep learning model started by importing all necessary libraries:

- `import numpy as np`
- `import pandas as pd`
- `import matplotlib.pyplot as plt`
- `import tensorflow as tf`
- `from sklearn.model_selection import train_test_split`
- `from keras.preprocessing import sequence`
- `from keras.models import Sequential`
- `from keras.layers import Dense, Dropout, Embedding, LSTM, Bidirectional, Conv1D, MaxPooling1D, Flatten`
- `from keras.utils import to_categorical`

1.3.1.3. Preprocessing data

- Fill missing values with the mean of the column.

- Set dependent and independent variables: we choose columns “Country”, “Sector” and “Year” like independent ones, since base on that model can give a forecast for future indicators which are dependent, in our case it is “Value” column. When model trains it memorize significant features which helps for making predictions.
- Convert categorical variables: “Country”, “Sector” into dummy/indicator variables.
- Splitting data on train and test: test dataset includes records from 2013 up to 2018 and train from 1990-2012.

1.3.1.4 Creating the models

For saving the model and using it on the website we have selected CNN model due to the fact that it can be used to classify time-series data of methane-producing industrial processes, identify emission sources, and optimize processes to reduce methane emissions. In both cases, Conv1D layers can learn complex patterns and features from time series data that can help detect and prevent methane emissions.

1.3.1.5 Classification results

- Write function for evaluating methane concentration based on prediction of the model.
- Identify three categories: normal, middle and dangerous level of concentration methane in the air.

| 1.3.2. Deploying the model

After saving our model we deploy it to website using Flask. Flask - a framework for creating web applications in the Python programming language.



How we deployed our deep learning model using Flask:

1. We train and save deep learning model using a Python library like TensorFlow.
2. Create a Flask application and add a route to handle POST requests for our model.
3. In the route, load saved model in .pkl format and preprocess the input data.
4. Make a prediction using the loaded model and return the result to the user.
5. Add index.html which create and structure content for the website, and style.css, which help us to describe the presentation and layout of html.

1.3.2.1. Website concept

Our website “SmartMethane” is the best solution for predicting methane emission, which prevent explosions, particularly in mines. The site's interface is concise, but even so it carries functionality and user interaction.

Users only need to enter three values: “Country”, “Sector”, “Year”- based on which a prediction will be made. After entering this data, a message with the classification by hazard level and concentration will appear on the main screen.

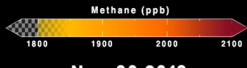
Predict the probability of methane explosion

COUNTRY

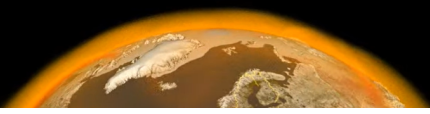
SECTOR

YEAR

The classification is NORMAL. The predicted methane concentration is about = 7.593925488436071e-18 for period = 1.2 years



Nov 30 2018



[top](#)

1.3.2.2. Code concept

The functional part is in the file `app.py`:

- Inputting data by users;
- Converting string values to numerical by using one-hot encoding;
- Calculation of concentration by formula and division into hazard classes

The model's predictions used to calculate the concentration of methane in a mixture and classify the concentration level based on predetermined conditions.

```
# Calculate the concentration and classification
concentration = np.mean(prediction) / 1000
if concentration < 0.0003:
    return render_template('index.html',
                           concentration='The classification is NORMAL. \n The predicted methane concentration is '
                           'about = {} for period = {} years\n'.format(abs(concentration), period),
                           bhai="Your prediction")

elif 0.0003 <= concentration <= 0.0005:
    return render_template('index.html',
                           concentration='The classification is MEDIUM. \n The predicted methane concentration is '
                           'about = {} for period = {} years\n'.format(abs(concentration), period),
                           bhai="Your prediction")

elif concentration > 0.0005:
    return render_template('index.html',
                           concentration='The classification is DANGEROUS. \n The predicted methane concentration '
                           'is about = {} for period = {} years\n'.format(abs(concentration), period),
                           bhai="Your prediction")
```

index.html form includes input fields for the country, sector, and year data that will be used for making a prediction. When the user submits the form, the data is sent to the Flask app using the POST method.

2. Data and Methods

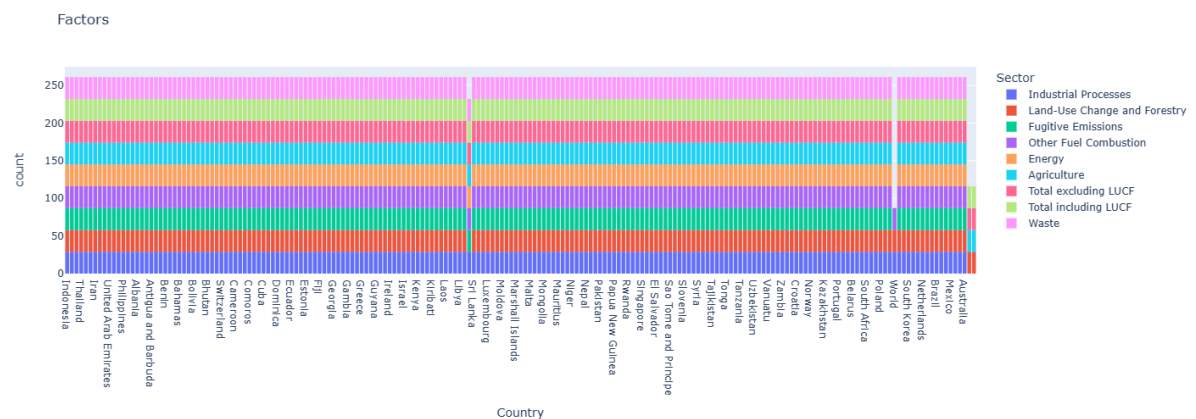
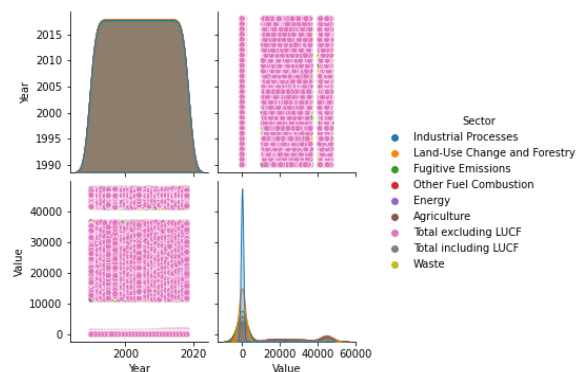
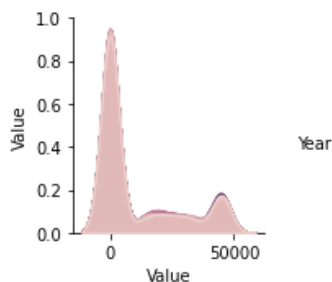
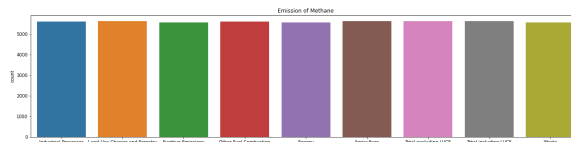
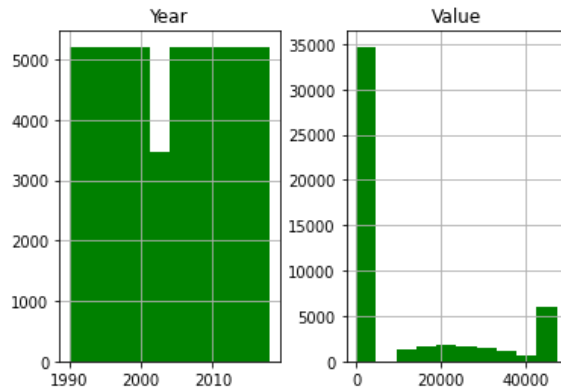
2.1. ■ Information about the data (probably analysis of the data with some visualizations)

	Year	Value
count	50402.000000	50402.000000
mean	2004.000000	10111.335469
std	8.366683	16441.907724
min	1990.000000	0.000000
25%	1997.000000	0.010000
50%	2004.000000	0.740000
75%	2011.000000	19054.000000
max	2018.000000	47453.000000

```

Country    object
Sector     object
Year       int64
Value      float64
dtype: object

```



2.2. ■ Description of the ML models you used with some theory

The models is trained on data consisting of Xtrain and ytrain. where Xtrain is a 3D shape tensor (number of samples, number of time steps, number of features) and ytrain is a 1D shape tensor (number of samples) containing binary labels. The model is trained using the Adam optimizer and binary cross-entropy loss function, and the model performance is evaluated on a separate validation set consisting of xtest and ytest.

- **RNN:**

A deep learning model that uses the Keras library to classify data using a combination of Long Short-Term Memory (LSTM) layers and dense layers. LSTM layers allow models to store information over long periods of time, making them well-suited for time-series data and sequence modeling tasks. A dense layer at the end of the model is used to make final predictions based on the output of the LSTM layer.

It can be used to predict and detect potential sources of methane emissions from industrial sites, landfills, or livestock, and provide real-time feedback to take preventive action. Additionally, deep learning models can be used to optimize energy consumption and reduce the carbon footprint of industrial processes, indirectly helping to avoid methane emissions.

- **CNN:**

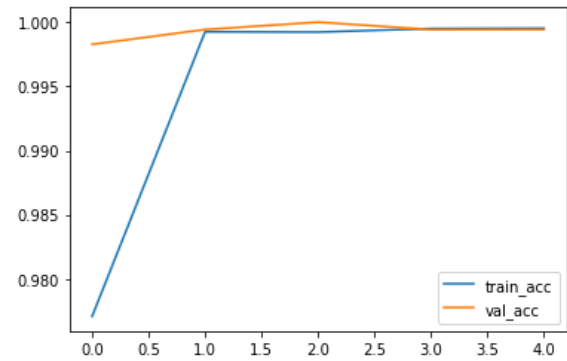
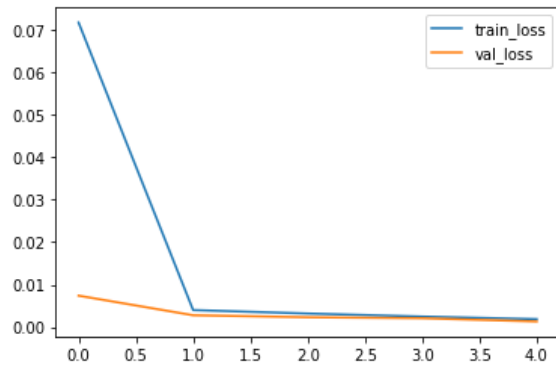
A deep learning model that uses the Keras library to classify data using a combination of Conv1D layers (1D convolutions) and dense layers. The Conv1D layer uses a sliding window to learn patterns and features from the input time series data, and the MaxPooling1D layer is used to downsample the feature map, reducing model complexity and improving computational efficiency. The flattening layer is used to transform the 2D feature map produced by the Conv1D layer into his 1D feature vector. This is fed into the dense layer to make the final prediction.

In the context of avoiding methane emissions, this model can be used in many ways. For example, it can be used to classify time-series data from sensors that monitor methane concentration in the atmosphere, enabling early detection and prevention of methane leaks.

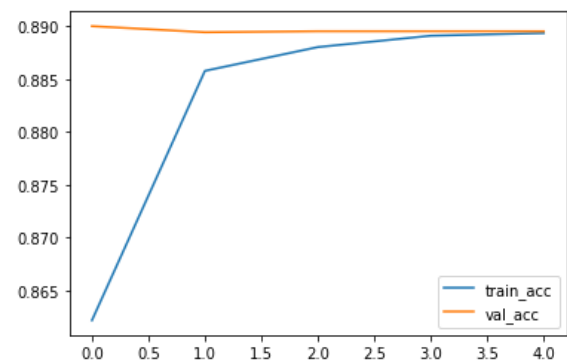
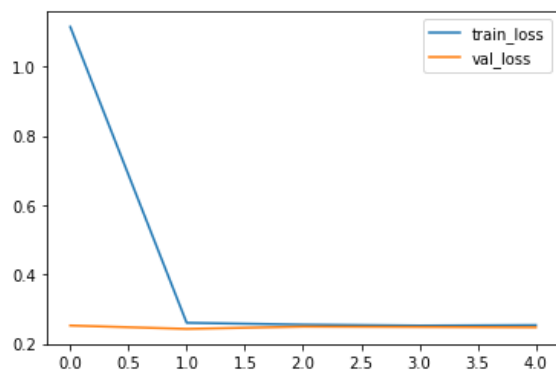
3. Results

3.1. ■ Results with tables, pictures and interesting numbers

- **LSTM MODEL**



- **CNN MODEL**



Our models achieved a high level of accuracy in detecting methane emissions from time series data. The LSTM model achieved 99% accuracy and the CNN model achieved 89% accuracy. These results demonstrate that our models can accurately identify the presence of methane in the atmosphere and detect and prevent methane leaks early.

4. Discussion

4.1. ■ Critical review of results

Our Smart Methane web application provides a user-friendly interface where users can easily enter data and obtain information about the concentration or amount of methane present. An algorithm developed for a web application involves analyzing time series data from a methane sensor and comparing absorbance readings to a threshold representing the minimum level of methane considered dangerous.

One limitation of our project is that we trained and tested the models on only one dataset. Although the data set was large and varied, it may not represent all possible scenarios under which methane emissions could occur. Therefore, the models may not perform as well in other contexts and additional data may need to be collected and used to train the models

The high accuracy of our models suggests that they can be used for a variety of purposes, such as monitoring methane concentrations in the atmosphere or detecting methane leaks in industrial environments.

4.2. ■ Next steps

Our project has several possible next steps. The next important step is to collect and use additional data to train and test the model. This includes collecting data from a variety of sources and contexts to ensure models are robust and effective in a variety of environments.

Moreover, our model can be integrated into larger systems for monitoring and regulating methane emissions. For example, models can be used to trigger alarms and alerts when methane concentrations reach certain levels, or provide real-time feedback on industrial processes to reduce emissions. Finally, you can explore using other deep learning models or techniques such as: Hybrid models combining RNN and CNN layers, or unsupervised learning techniques that can be used to identify patterns and anomalies in data. This may improve the accuracy and effectiveness of our approach.

Our models reach high accuracy, but there is still room for improvement. Future work will explore more sophisticated model architectures and explore the use of other data types such as video and image data to improve model accuracy.

Overall, our project demonstrates the potential for deep learning to address environmental challenges and highlights the importance of technological solutions in mitigating the impact of greenhouse gas emissions on our planet.

5. References list:

1. SibElectro. (n.d.). Когда есть опасность взрыва? [When is there a risk of explosion?]. Retrieved from [click](#)
(An article in Russian that discusses the conditions and factors that can lead to an explosion in various settings.)

2. Mining Media. (n.d.). Химические основы изменения концентрационных пределов и скорости реакций возгорания и взрыва метано-воздушных смесей в горных выработках [Chemical foundations for changing concentration limits and rates of combustion and explosion reactions of methane-air mixtures in mining excavations]. Retrieved from [click](#)
(A scientific article in Russian that discusses the chemical foundations for changing the concentration limits and rates of combustion and explosion reactions of methane-air mixtures in mining excavations.)
3. GeeksforGeeks. (n.d.). Data Visualization Different Charts in Python. Retrieved from [click](#)
(An article that provides an overview of different data visualization techniques and how to implement them using Python).
4. Liguori, A. (2020, June 12). Deploying a Predictive Python ML Model with Flask and Heroku - Part 3. Retrieved from [click](#) (This article is part of a series on deploying a machine learning model with Flask and Heroku. Part 3 focuses on setting up the Flask app and deploying the model to Heroku).
5. Vartanov, A. (2019, February 13). One-Hot Encoding для машинного обучения. Retrieved from [click](#) (This article in Russian discusses one-hot encoding, a common technique used in machine learning to convert categorical variables to numerical format).
6. Corey Schafer. (2018, September 4). Python Flask Tutorial: Full-Featured Web App Part 3 - Templates. Retrieved from [click](#) (This YouTube video is part of a series that provides a tutorial on building a web application using the Flask framework in Python).
7. Stack Overflow. (2019, February 25). Prediction after one hot encoding [Web log post]. Retrieved from [click](#) (This Stack Overflow post provides a discussion on how to perform predictions after performing one-hot encoding on categorical data).
8. Sofiane Kadri. (n.d.). Methane Emissions around the World (1990-2018). Retrieved from [click](#)
(This Kaggle dataset provides data on methane emissions around the world from 1990 to 2018).
9. Vision Systems Design. (2019, June 25). What is deep learning and how do I deploy it in imaging? Retrieved from [click](#) (This article explains what deep learning is and how it can be deployed for use in imaging).

10. Vision Systems Design. (2019, June 27). Five steps for building and deploying a deep learning neural network. Retrieved from [click](#) (This article provides a step-by-step guide for building and deploying a deep learning neural network).