CS634 Final Term Project Report

Student Name: Sebastian Skubisz

Course: CS634 - Data Mining

Instructor: Dr. Yasser Abduallah

Project Title: Performance Evaluation of Classification Models for Water Quality Prediction Using SMOTE and Cross-Validation

Date: November 16, 2025

1. Introduction

This project aims to predict whether water is safe to drink based on chemical attributes like ammonia, arsenic, and mercury. The goal is to classify water samples as "safe" or "not safe," helping identify when water needs treatment.

We use supervised binary classification, which predicts one of two outcomes. The main models are Random Forest, a strong ensemble method; LSTM, a deep learning model that captures sequences; and K-Nearest Neighbors (KNN), a simple and effective algorithm. These models are tested and compared to find the best way to predict water safety.

2. Dataset

The Water Quality Dataset from Kaggle is used to classify whether water is safe or not. It has 8,000 rows and 21 columns, with each row representing a water sample and features like ammonia, arsenic, barium, mercury, and other quality indicators. The target variable, is_safe, is 1 if the water is safe and 0 if not.

https://www.kaggle.com/datasets/mssmartypants/water-quality

To prepare the data, missing values were filled with the median of each feature. The data was then standardized to have a mean of 0 and a standard deviation of 1, which helps many machine learning models perform better. The target labels were encoded as 0 and 1. Since there were more safe samples than unsafe ones, SMOTE was used to balance the classes by creating synthetic examples of the minority class.

3. Algorithms Overview

Random Forest is an ensemble learning method that builds many decision trees during training and predicts based on the majority vote of all trees. It's effective for both classification and regression, especially when data has complex or non-linear feature interactions. Random Forest handles large datasets with many features well and is resistant to overfitting. It works by creating multiple decision trees trained on different random subsets of the data, then combining their predictions for a final result. This makes it more robust than a single decision tree, especially with noisy or diverse data like water quality information.

The deep learning model used is LSTM (Long Short-Term Memory), which is good at capturing sequential patterns in data. Even if water quality data isn't strictly sequential, LSTM can still find complex relationships across features. It retains information over long periods, making it useful for modeling intricate patterns that traditional models might miss, especially in complex environmental datasets with many interrelated features.

For a simpler machine learning approach, K-Nearest Neighbors (KNN) was chosen. KNN classifies a new data point based on the majority class among its closest neighbors in the feature space. It's easy to understand and works well when the decision boundary isn't linear. KNN performs well with high-dimensional data but can be slow during prediction because it calculates distances to all training points. It is also sensitive to irrelevant features and requires good data preprocessing and normalization to perform optimally.

4. Implementation

This project is developed using Python, a popular language in data science and machine learning due to its powerful libraries and ease of use. The main development environment is Jupyter Notebook, which supports interactive coding and easy visualization of results, ideal for experimenting with algorithms and viewing outputs. Python scripts (.py files) are also used to organize the project in a modular way, making testing and deployment easier.

Several Python packages are needed to run the project. These include numpy for numerical calculations, pandas for data handling, and scikit-learn for machine learning algorithms like Random Forest and KNN. For deep learning, tensorflow and keras are used to build and train models like LSTM. Visualization tools such as matplotlib and seaborn help display data and results. To install all necessary packages, run the appropriate installation command.

```
pip install numpy pandas scikit-learn tensorflow keras matplotlib seaborn
```

Or

```
pip install -r requirements.txt
```

```
python training.py
```

Average performance over 10 folds (rounded):

	KNN	RF	LSTM
TP	58.700	49.800	47.500
TN	358.800	481.600	473.400
FP	137.100	14.300	22.500
FN	5.100	14.000	16.300
TPR	0.920	0.780	0.745
TNR	0.724	0.971	0.955
FPR	0.276	0.029	0.045
FNR	0.080	0.220	0.255
Precision	0.300	0.779	0.679
F1_measure	0.453	0.779	0.710
Accuracy	0.746	0.949	0.931
Error_rate	0.254	0.051	0.069
BACC	0.822	0.876	0.850
TSS	0.644	0.752	0.699
HSS	0.339	0.750	0.670
Brier_score	0.173	0.042	0.057
AUC	0.890	0.972	0.945
Acc_by_package_fn	0.746	0.949	0.931

```
Loaded dataset with shape: (7999, 21)

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7999 entries, 0 to 7998
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   aluminium    7999 non-null   float64
 1   ammonia      7999 non-null   object 
 2   arsenic      7999 non-null   float64
 3   barium       7999 non-null   float64
 4   cadmium      7999 non-null   float64
 5   chloramine   7999 non-null   float64
 6   chromium     7999 non-null   float64
 7   copper       7999 non-null   float64
 8   flouride     7999 non-null   float64
 9   bacteria     7999 non-null   float64
 10  viruses      7999 non-null   float64
 11  lead         7999 non-null   float64
 12  nitrates     7999 non-null   float64
 13  nitrites     7999 non-null   float64
 14  mercury      7999 non-null   float64
 15  perchlorate  7999 non-null   float64
 16  radium       7999 non-null   float64
 17  selenium     7999 non-null   float64
 18  silver       7999 non-null   float64
 19  uranium      7999 non-null   float64
 20  is_safe      7999 non-null   object 
dtypes: float64(19), object(2)
memory usage: 1.3+ MB
```

```
Converting feature columns to numeric...
Converting 'is_safe' to numeric and dropping invalid rows...
Dropped 3 rows with invalid 'is_safe' values.

Imputing zeros/NaNs in feature columns with median values...
 - Column 'aluminium': 251 zeros, 0 NaNs before imputation
 - Column 'ammonia': 6 zeros, 0 NaNs before imputation
 - Column 'arsenic': 187 zeros, 0 NaNs before imputation
 - Column 'barium': 14 zeros, 0 NaNs before imputation
 - Column 'cadmium': 236 zeros, 0 NaNs before imputation
 - Column 'chloramine': 102 zeros, 0 NaNs before imputation
 - Column 'chromium': 208 zeros, 0 NaNs before imputation
 - Column 'copper': 94 zeros, 0 NaNs before imputation
 - Column 'flouride': 25 zeros, 0 NaNs before imputation
 - Column 'bacteria': 2793 zeros, 0 NaNs before imputation
 - Column 'viruses': 1303 zeros, 0 NaNs before imputation
 - Column 'lead': 1 zeros, 0 NaNs before imputation
 - Column 'nitrates': 1 zeros, 0 NaNs before imputation
 - Column 'nitrites': 16 zeros, 0 NaNs before imputation
 - Column 'mercury': 371 zeros, 0 NaNs before imputation
 - Column 'perchlorate': 7 zeros, 0 NaNs before imputation
 - Column 'radium': 18 zeros, 0 NaNs before imputation
 - Column 'selenium': 403 zeros, 0 NaNs before imputation
 - Column 'silver': 246 zeros, 0 NaNs before imputation
 - Column 'uranium': 594 zeros, 0 NaNs before imputation
Finished preprocessing. New dataset shape: (7996, 21)
```

```
Preview of preprocessed data:
   aluminium  ammonia  arsenic  barium  cadmium  chloramine  chromium  copper  \
0       1.65     9.08     0.04    2.85    0.007        0.35      0.83    0.17
1       2.32    21.16     0.01    3.31    0.002        5.28      0.68    0.66
2       1.01    14.02     0.04    0.58    0.008        4.24      0.53    0.02
3       1.36    11.33     0.04    2.96    0.001        7.23      0.03    1.66
4       0.92    24.33     0.03    0.20    0.006        2.67      0.69    0.57

   flouride  bacteria  ...   lead  nitrates  nitrites  mercury  perchlorate  \
0      0.05      0.20  ...  0.054     16.08      1.13    0.007        37.75
1      0.90      0.65  ...  0.100      2.01      1.93    0.003        32.26
2      0.99      0.05  ...  0.078     14.16      1.11    0.006        50.28
3      1.08      0.71  ...  0.016      1.41      1.29    0.004         9.12
4      0.61      0.13  ...  0.117      6.74      1.11    0.003        16.90

   radium  selenium  silver  uranium  is_safe
0    6.78      0.08    0.34     0.02        1
1    3.21      0.08    0.27     0.05        1
2    7.07      0.07    0.44     0.01        0
3    1.72      0.02    0.45     0.05        1
4    2.41      0.02    0.06     0.02        1

[5 rows x 21 columns]
```

```
Features shape: (7996, 20)
Labels shape: (7996,)

Class distribution (is_safe):
is_safe
0    7084
1     912
Name: count, dtype: int64

----------Checking for Data Imbalance------------
Number of Positive Outcomes:  7084
Percentage of Positive Outcomes: 88.59%
Number of Negative Outcomes :  912
Percentage of Negative Outcomes: 11.41%
-------------------------------------------------
```

```
Correlation of each feature with 'is_safe':
aluminium      0.333740
chloramine     0.186434
chromium       0.182355
silver         0.102017
barium         0.090753
perchlorate    0.075791
radium         0.065655
nitrites       0.048392
copper         0.028511
flouride       0.005288
lead          -0.009360
bacteria      -0.018284
ammonia       -0.022743
selenium      -0.024732
mercury       -0.030667
viruses       -0.045848
nitrates      -0.071490
uranium       -0.077183
arsenic       -0.124948
cadmium       -0.271881
Name: is_safe, dtype: float64

Highest absolute correlation with 'is_safe': aluminium (corr = 0.3337)
```

```
Histograms plotted for all feature columns.

Skewness of each feature:
nitrites      -0.495900
mercury       -0.092567
lead          -0.060667
viruses       -0.052418
nitrates      -0.042061
flouride      -0.039333
uranium       -0.016994
bacteria      -0.014163
ammonia        0.026424
selenium       0.062251
copper         0.241822
cadmium        0.462695
radium         0.548674
barium         0.662255
chloramine     0.889899
perchlorate    0.937986
chromium       1.037485
silver         1.048531
arsenic        1.990500
aluminium      2.014929
dtype: float64

Features with approximate symmetry (|skew| < 0.5):
['ammonia', 'cadmium', 'copper', 'flouride', 'bacteria', 'viruses', 'lead', 'nitrates', 'nitrites', 'mercury', 'selenium', 'uranium']
```

```
Training set shape: (5597, 20)
Test set set shape: (2399, 20)

Training data (standardized) summary:
          aluminium       ammonia       arsenic        barium       cadmium  \
count  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03
mean  -1.396457e-17  1.491670e-16  7.045761e-17 -4.633698e-17 -6.474483e-17
std    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min   -5.220567e-01 -1.626751e+00 -6.415196e-01 -1.283456e+00 -1.221047e+00
25%   -4.986138e-01 -8.626112e-01 -5.250050e-01 -8.307739e-01 -9.949998e-01
50%   -4.673567e-01 -1.406668e-02 -4.446501e-01 -3.122473e-01 -1.190666e-01
75%   -3.110708e-01  8.873712e-01 -2.437630e-01  7.412669e-01  7.286107e-01
max    3.416347e+00  1.740417e+00  3.573093e+00  2.774220e+00  2.423965e+00

         chloramine      chromium        copper      flouride      bacteria  \
count  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03
mean  -6.760122e-17  2.126423e-17 -6.601433e-17  3.618093e-16  1.142556e-17
std    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min   -8.442890e-01 -8.909774e-01 -1.252761e+00 -1.762467e+00 -2.091531e+00
25%   -8.090765e-01 -7.429517e-01 -1.114053e+00 -8.427510e-01 -4.898023e-01
50%   -6.291014e-01 -5.949260e-01 -9.686630e-02 -1.500680e-02  2.967725e-02
75%    7.793996e-01  7.373052e-01  8.740850e-01  8.817161e-01  4.625769e-01
max    2.493076e+00  2.402594e+00  1.814212e+00  1.663475e+00  2.194175e+00

           viruses          lead      nitrates      nitrites       mercury  \
count  5597.000000  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03
mean      0.000000 -1.777309e-17  2.278764e-16  1.015605e-17 -1.485322e-16
std       1.000000  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min      -1.163468 -1.696924e+00 -1.778221e+00 -2.327137e+00 -1.628551e+00
25%      -1.149252 -8.890785e-01 -8.509969e-01 -5.722683e-01 -9.000794e-01
50%       0.312161  3.908419e-02  1.465387e-02  1.472277e-01  1.926278e-01
75%       0.823940  8.984940e-01  8.640058e-01  7.438830e-01  9.210992e-01
max       1.676905  1.723528e+00  1.811151e+00  2.779530e+00  1.649571e+00

        perchlorate        radium      selenium        silver       uranium
count  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03  5.597000e+03
mean  -5.458878e-17  8.505693e-17  1.142556e-16 -2.012168e-16 -2.183551e-16
std    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min   -9.359252e-01 -1.264000e+00 -1.611566e+00 -9.912686e-01 -1.606983e+00
25%   -8.134339e-01 -9.102931e-01 -8.523122e-01 -7.068548e-01 -7.677685e-01
50%   -4.941626e-01 -2.201339e-01 -9.305844e-02 -4.224411e-01  7.144648e-02
75%    7.466287e-01  7.547160e-01  6.661953e-01  6.441105e-01  9.106615e-01
max    2.466611e+00  2.178169e+00  1.805076e+00  2.492800e+00  1.749876e+00
Number of features (for LSTM): 20
```

```
Original training label distribution:
Counter({0: 4959, 1: 638})

Balanced training label distribution after SMOTE:
Counter({0: 4959, 1: 4959})

Class distribution before and after SMOTE:
Original training set class distribution: Counter({0: 4959, 1: 638})
Balanced training set class distribution after SMOTE: Counter({0: 4959, 1: 4959})
```

--- Cross-Validation Iteration 9 ---

Iteration 9:
----- Metrics for all Algorithms in Iteration 9 -----

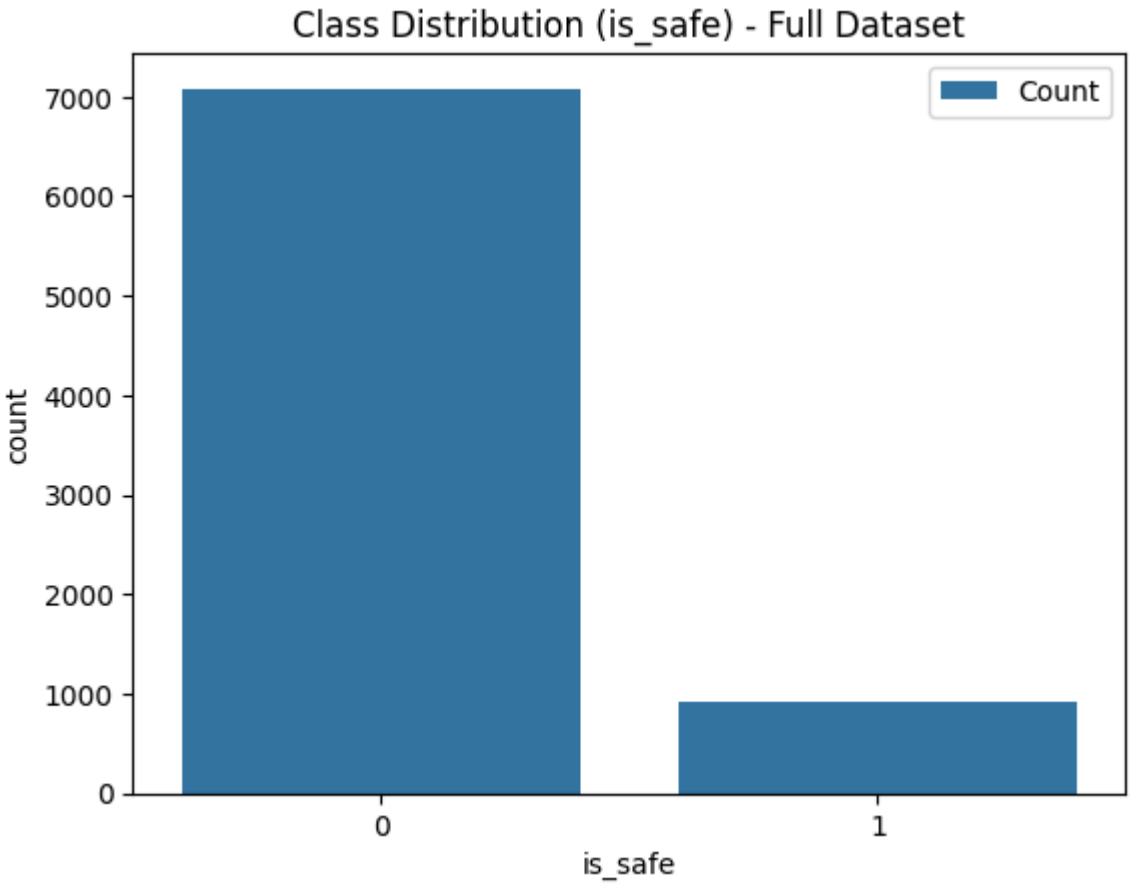
	KNN	RF	LSTM
TP	59.00	46.00	46.00
TN	363.00	480.00	467.00
FP	133.00	16.00	29.00
FN	4.00	17.00	17.00
TPR	0.94	0.73	0.73
TNR	0.73	0.97	0.94
FPR	0.27	0.03	0.06
FNR	0.06	0.27	0.27
Precision	0.31	0.74	0.61
F1_measure	0.46	0.74	0.67
Accuracy	0.75	0.94	0.92
Error_rate	0.25	0.06	0.08
BACC	0.83	0.85	0.84
TSS	0.67	0.70	0.67
HSS	0.35	0.70	0.62
Brier_score	0.17	0.05	0.06
AUC	0.89	0.96	0.94
Acc_by_package_fn	0.75	0.94	0.92

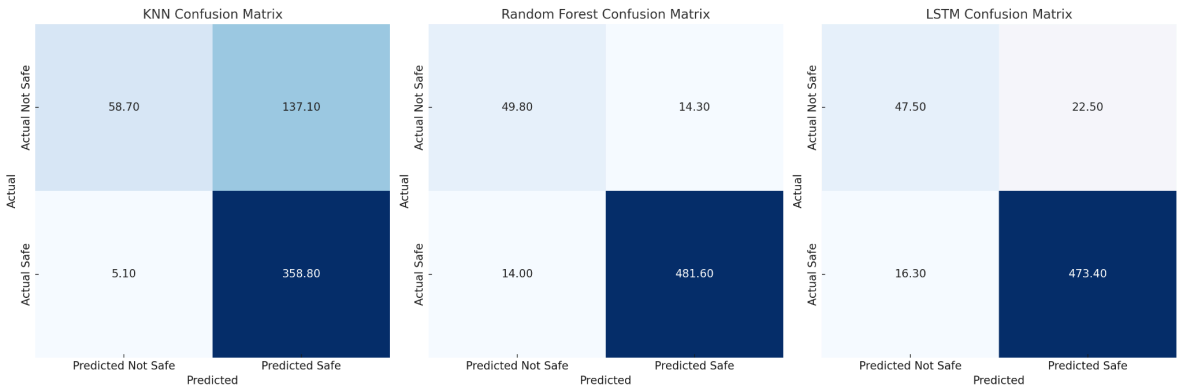
--- Cross-Validation Iteration 10 ---

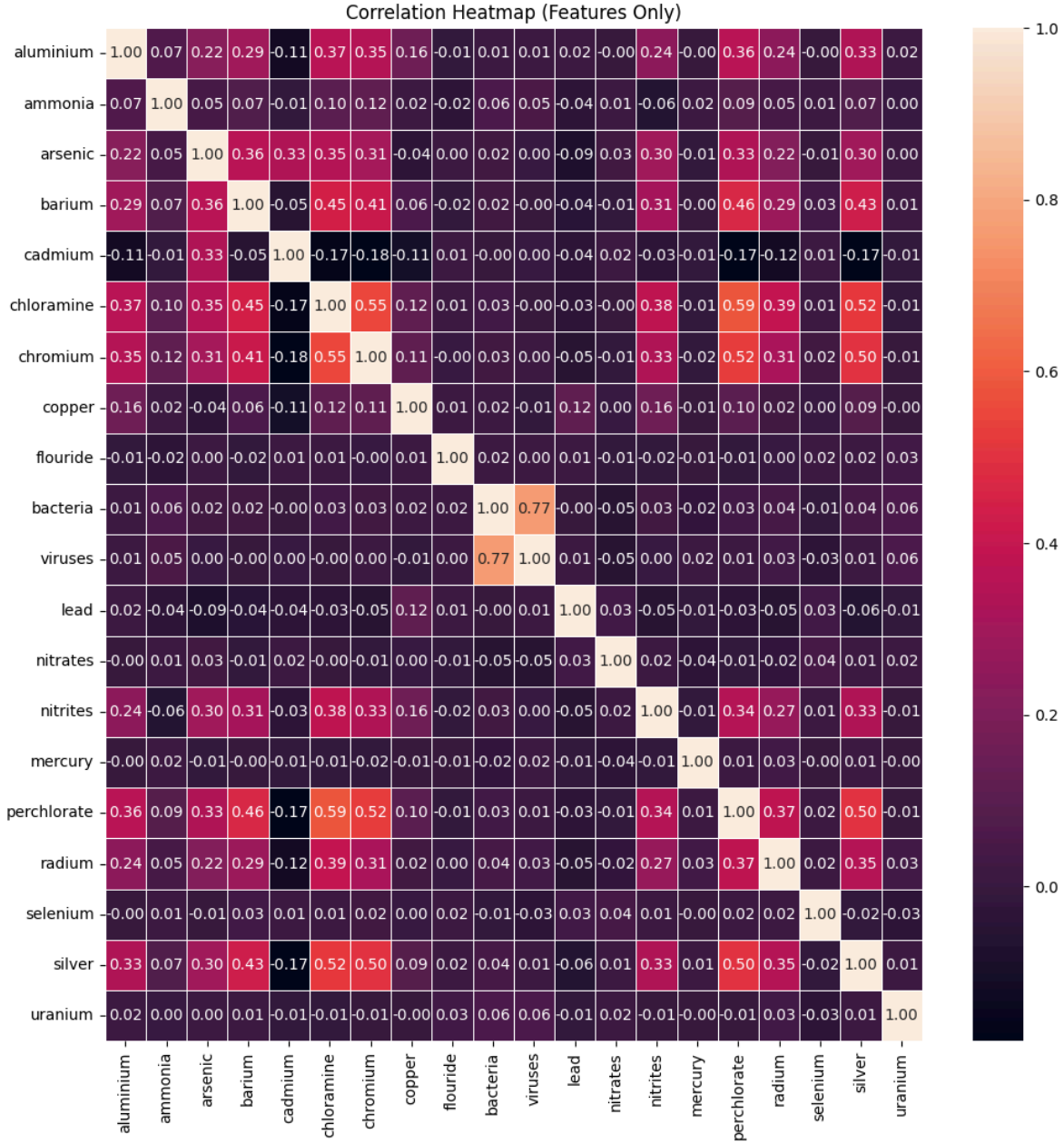
Iteration 10:
----- Metrics for all Algorithms in Iteration 10 -----

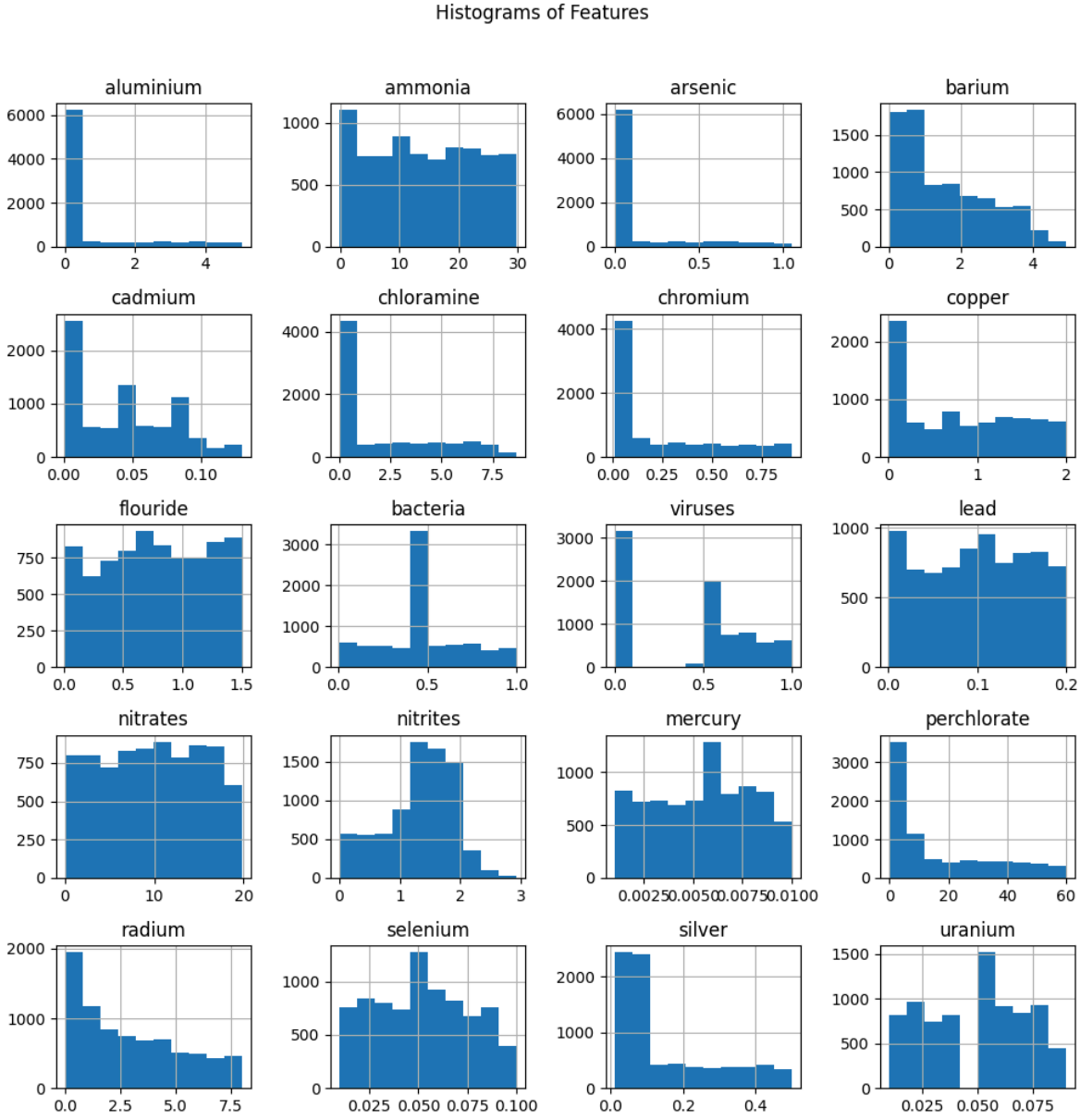
	KNN	RF	LSTM
TP	60.00	57.00	48.00
TN	348.00	483.00	460.00
FP	147.00	12.00	35.00
FN	4.00	7.00	16.00
TPR	0.94	0.89	0.75
TNR	0.70	0.98	0.93
FPR	0.30	0.02	0.07
FNR	0.06	0.11	0.25
Precision	0.29	0.83	0.58
F1_measure	0.44	0.86	0.65
Accuracy	0.73	0.97	0.91
Error_rate	0.27	0.03	0.09
BACC	0.82	0.93	0.84
TSS	0.64	0.87	0.68
HSS	0.32	0.84	0.60
Brier_score	0.18	0.04	0.07
AUC	0.90	0.98	0.93
Acc_by_package_fn	0.73	0.97	0.91

5. Evaluation Setup

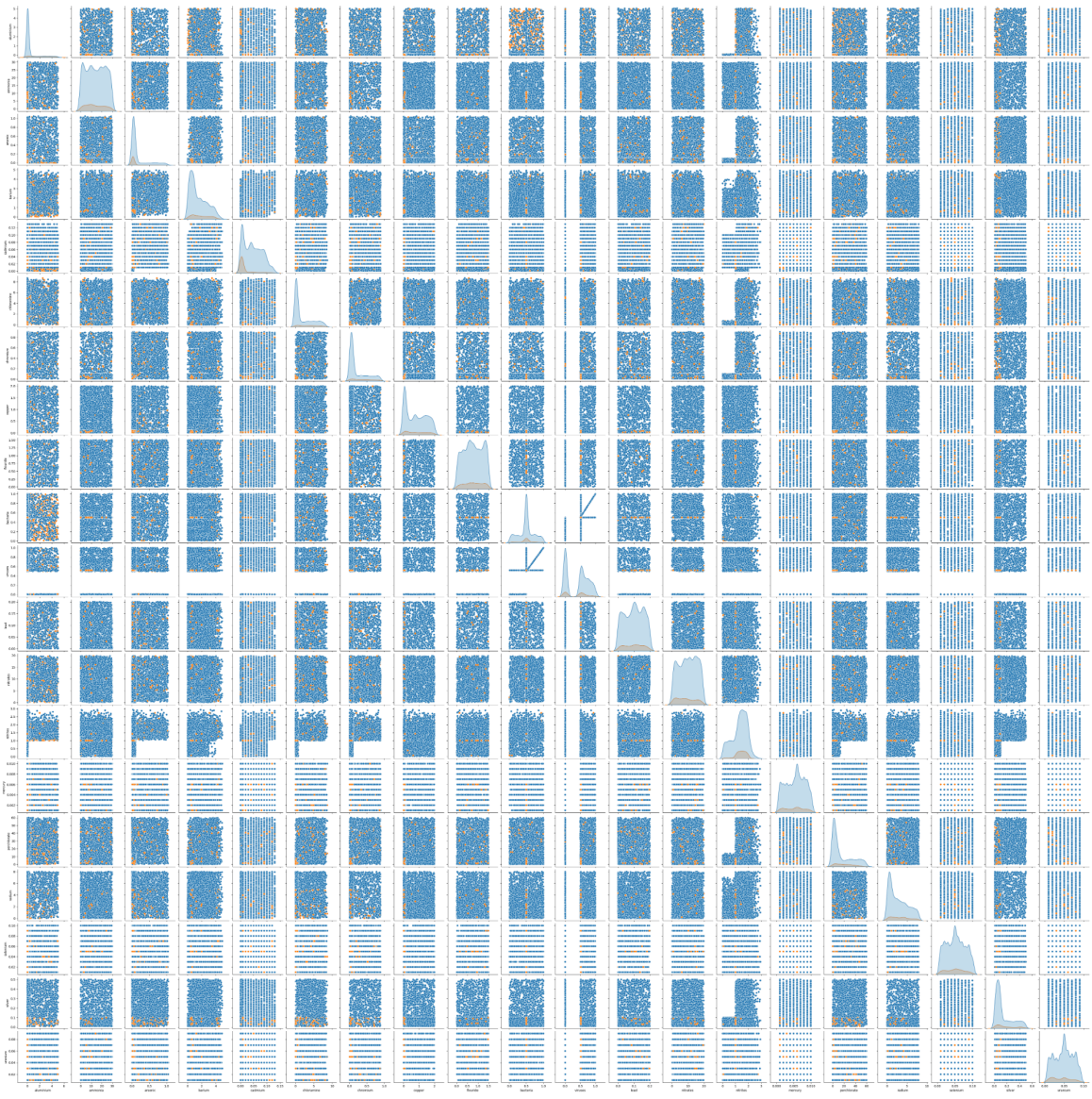





Correlation Heatmap (Features Only)
aluminium 1.00 0.07 0.22 0.29 -0.11 0.37 0.35 0.16 -0.01 0.01 0.01 0.02 -0.00 0.24 -0.00 0.36 0.24 -0.00 0.33 0.02
ammonia 0.07 1.00 0.05 0.07 -0.01 0.10 0.12 0.02 -0.02 0.06 0.05 -0.04 0.01 -0.06 0.02 0.09 0.05 0.01 0.07 0.00
arsenic 0.22 0.05 1.00 0.36 0.33 0.35 0.31 -0.04 0.00 0.02 0.00 -0.09 0.03 0.30 -0.01 0.33 0.22 -0.01 0.30 0.00
barium 0.29 0.07 0.36 1.00 -0.05 0.45 0.41 0.06 -0.02 0.02 -0.00 -0.04 -0.01 0.31 -0.00 0.46 0.29 0.03 0.43 0.01
cadmium -0.11 -0.01 0.33 -0.05 1.00 -0.17 -0.18 -0.11 0.01 -0.00 0.00 -0.04 0.02 -0.03 -0.01 -0.17 -0.12 0.01 -0.17 -0.01
chloramine 0.37 0.10 0.35 0.45 -0.17 1.00 0.55 0.12 0.01 0.03 -0.00 -0.03 -0.00 0.38 -0.01 0.59 0.39 0.01 0.52 -0.01
chromium 0.35 0.12 0.31 0.41 -0.18 0.55 1.00 0.11 -0.00 0.03 0.00 -0.05 -0.01 0.33 -0.02 0.52 0.31 0.02 0.50 -0.01
copper 0.16 0.02 -0.04 0.06 -0.11 0.12 0.11 1.00 0.01 0.02 -0.01 0.12 0.00 0.16 -0.01 0.10 0.02 0.00 0.09 -0.00
flouride -0.01 -0.02 0.00 -0.02 0.01 0.01 -0.00 0.01 1.00 0.02 0.00 0.01 -0.01 -0.02 -0.01 -0.01 0.00 0.02 0.02 0.03
bacteria 0.01 0.06 0.02 0.02 -0.00 0.03 0.03 0.02 0.02 1.00 0.77 -0.00 -0.05 0.03 -0.02 0.03 0.04 -0.01 0.04 0.06
viruses 0.01 0.05 0.00 -0.00 0.00 -0.00 0.00 -0.01 0.00 0.77 1.00 0.01 -0.05 0.00 0.02 0.01 0.03 -0.03 0.01 0.06
lead 0.02 -0.04 -0.09 -0.04 -0.04 -0.03 -0.05 0.12 0.01 -0.00 0.01 1.00 0.03 -0.05 -0.01 -0.03 -0.05 0.03 -0.06 -0.01
nitrates -0.00 0.01 0.03 -0.01 0.02 -0.00 -0.01 0.00 -0.01 -0.05 -0.05 0.03 1.00 0.02 -0.04 -0.01 -0.02 0.04 0.01 0.02
nitrites 0.24 -0.06 0.30 0.31 -0.03 0.38 0.33 0.16 -0.02 0.03 0.00 -0.05 0.02 1.00 -0.01 0.34 0.27 0.01 0.33 -0.01
mercury -0.00 0.02 -0.01 -0.00 -0.01 -0.01 -0.02 -0.01 -0.01 -0.02 0.02 -0.01 -0.04 -0.01 1.00 0.01 0.03 -0.00 0.01 -0.00
perchlorate 0.36 0.09 0.33 0.46 -0.17 0.59 0.52 0.10 -0.01 0.03 0.01 -0.03 -0.01 0.34 0.01 1.00 0.37 0.02 0.50 -0.01
radium 0.24 0.05 0.22 0.29 -0.12 0.39 0.31 0.02 0.00 0.04 0.03 -0.05 -0.02 0.27 0.03 0.37 1.00 0.02 0.35 0.03
selenium -0.00 0.01 -0.01 0.03 0.01 0.01 0.02 0.00 0.02 -0.01 -0.03 0.03 0.04 0.01 -0.00 0.02 0.02 1.00 -0.02 -0.03
silver 0.33 0.07 0.30 0.43 -0.17 0.52 0.50 0.09 0.02 0.04 0.01 -0.06 0.01 0.33 0.01 0.50 0.35 -0.02 1.00 0.01
uranium 0.02 0.00 0.00 0.01 -0.01 -0.01 -0.01 -0.00 0.03 0.06 0.06 -0.01 0.02 -0.01 -0.00 -0.01 0.03 -0.03 0.01 1.00
aluminium
ammonia
arsenic
barium
cadmium
chloramine
chromium
copper
flouride
bacteria
viruses
lead
nitrates
nitrites
mercury
perchlorate
radium
selenium
silver
uranium
1.0
0.8
0.6
0.4
0.2
0.0


Histograms of Features
aluminium
ammonia
arsenic
barium
cadmium
chloramine
chromium
copper
flouride
bacteria
viruses
lead
nitrates
nitrites
mercury
perchlorate
radium
selenium
silver
uranium

Pairplot of Features Colored by is_safe




Class Distribution Before and After SMOTE (Train Set)
Dataset
Original Train
SMOTE Train
Count
5000
4000
3000
2000
1000
0
0
1
Class

Metrics for Algorithm 1 (KNN):

	iter1	iter2	iter3	iter4	iter5	iter6	iter7	iter8	iter9	iter10
TP	58.00	61.00	59.00	58.00	59.00	56.00	57.00	60.00	59.00	60.00
TN	362.00	366.00	352.00	377.00	352.00	363.00	356.00	349.00	363.00	348.00
FP	134.00	130.00	144.00	119.00	144.00	133.00	140.00	147.00	133.00	147.00
FN	6.00	3.00	5.00	6.00	5.00	8.00	7.00	3.00	4.00	4.00
TPR	0.91	0.95	0.92	0.91	0.92	0.88	0.89	0.95	0.94	0.94
TNR	0.73	0.74	0.71	0.76	0.71	0.73	0.72	0.70	0.73	0.70
FPR	0.27	0.26	0.29	0.24	0.29	0.27	0.28	0.30	0.27	0.30
FNR	0.09	0.05	0.08	0.09	0.08	0.12	0.11	0.05	0.06	0.06
Precision	0.30	0.32	0.29	0.33	0.29	0.30	0.29	0.29	0.31	0.29
F1_measure	0.45	0.48	0.44	0.48	0.44	0.44	0.44	0.44	0.46	0.44
Accuracy	0.75	0.76	0.73	0.78	0.73	0.75	0.74	0.73	0.75	0.73
Error_rate	0.25	0.24	0.27	0.22	0.27	0.25	0.26	0.27	0.25	0.27
BACC	0.82	0.85	0.82	0.83	0.82	0.80	0.80	0.83	0.83	0.82
TSS	0.64	0.69	0.63	0.67	0.63	0.61	0.61	0.66	0.67	0.64
HSS	0.34	0.37	0.32	0.38	0.32	0.33	0.32	0.33	0.35	0.32
Brier_score	0.17	0.16	0.18	0.16	0.17	0.18	0.19	0.18	0.17	0.18
AUC	0.90	0.92	0.88	0.88	0.89	0.87	0.86	0.89	0.89	0.90
Acc_by_package_fn	0.75	0.76	0.73	0.78	0.73	0.75	0.74	0.73	0.75	0.73

Metrics for Algorithm 2 (RF):

	iter1	iter2	iter3	iter4	iter5	iter6	iter7	iter8	iter9	iter10
TP	52.00	50.00	50.00	51.00	51.00	48.00	49.00	46.00	46.00	57.00
TN	489.00	477.00	480.00	482.00	488.00	481.00	475.00	480.00	480.00	483.00
FP	7.00	19.00	16.00	14.00	8.00	15.00	21.00	16.00	16.00	12.00
FN	12.00	14.00	14.00	13.00	13.00	16.00	15.00	17.00	17.00	7.00
TPR	0.81	0.78	0.78	0.80	0.80	0.75	0.77	0.73	0.73	0.89
TNR	0.99	0.96	0.97	0.97	0.98	0.97	0.96	0.97	0.97	0.98
FPR	0.01	0.04	0.03	0.03	0.02	0.03	0.04	0.03	0.03	0.02
FNR	0.19	0.22	0.22	0.20	0.20	0.25	0.23	0.27	0.27	0.11
Precision	0.88	0.72	0.76	0.78	0.86	0.76	0.70	0.74	0.74	0.83
F1_measure	0.85	0.75	0.77	0.79	0.83	0.76	0.73	0.74	0.74	0.86
Accuracy	0.97	0.94	0.95	0.95	0.96	0.94	0.94	0.94	0.94	0.97
Error_rate	0.03	0.06	0.05	0.05	0.04	0.06	0.06	0.06	0.06	0.03
BACC	0.90	0.87	0.87	0.88	0.89	0.86	0.86	0.85	0.85	0.93
TSS	0.80	0.74	0.75	0.77	0.78	0.72	0.72	0.70	0.70	0.87
HSS	0.83	0.72	0.74	0.76	0.81	0.72	0.69	0.70	0.70	0.84
Brier_score	0.04	0.05	0.04	0.04	0.03	0.05	0.05	0.04	0.05	0.04
AUC	0.98	0.96	0.97	0.97	0.98	0.96	0.97	0.98	0.96	0.98
Acc_by_package_fn	0.97	0.94	0.95	0.95	0.96	0.94	0.94	0.94	0.94	0.97

Metrics for Algorithm 3 (LSTM):

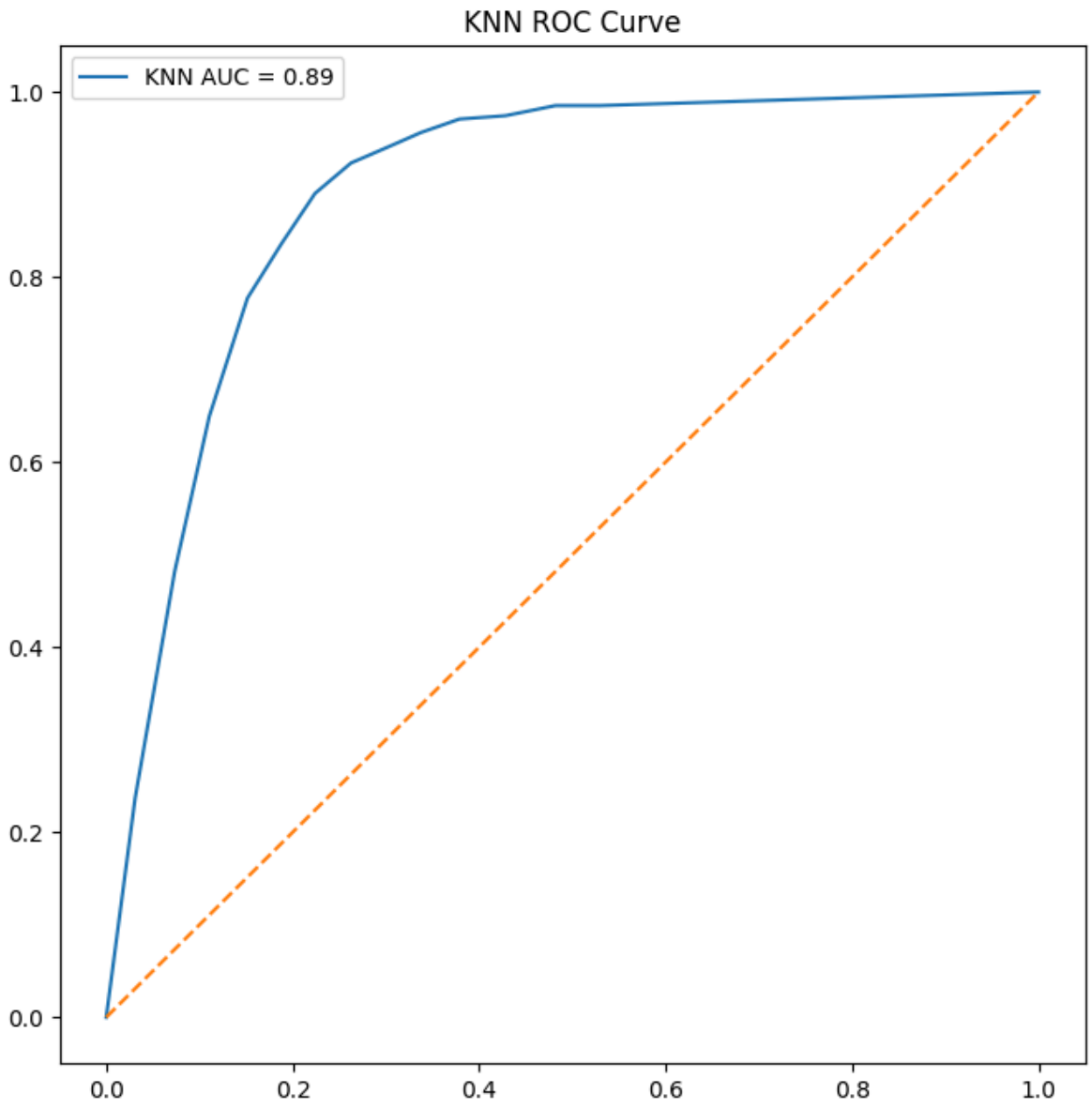
	iter1	iter2	iter3	iter4	iter5	iter6	iter7	iter8	iter9	iter10
TP	50.00	52.00	50.00	51.00	52.00	45.00	49.00	46.00	46.00	48.00
TN	483.00	463.00	461.00	455.00	477.00	474.00	467.00	478.00	467.00	460.00
FP	13.00	33.00	35.00	41.00	19.00	22.00	29.00	18.00	29.00	35.00
FN	14.00	12.00	14.00	13.00	12.00	19.00	15.00	17.00	17.00	16.00
TPR	0.78	0.81	0.78	0.80	0.81	0.70	0.77	0.73	0.73	0.75
TNR	0.97	0.93	0.93	0.92	0.96	0.96	0.94	0.96	0.94	0.93
FPR	0.03	0.07	0.07	0.08	0.04	0.04	0.06	0.04	0.06	0.07
FNR	0.22	0.19	0.22	0.20	0.19	0.30	0.23	0.27	0.27	0.25
Precision	0.79	0.61	0.59	0.55	0.73	0.67	0.63	0.72	0.61	0.58
F1_measure	0.79	0.70	0.67	0.65	0.77	0.69	0.69	0.72	0.67	0.65
Accuracy	0.95	0.92	0.91	0.90	0.94	0.93	0.92	0.94	0.92	0.91
Error_rate	0.05	0.08	0.09	0.10	0.06	0.07	0.08	0.06	0.08	0.09
BACC	0.88	0.87	0.86	0.86	0.89	0.83	0.85	0.85	0.84	0.84
TSS	0.76	0.75	0.71	0.71	0.77	0.66	0.71	0.69	0.67	0.68
HSS	0.76	0.65	0.62	0.60	0.74	0.65	0.65	0.69	0.62	0.60
Brier_score	0.04	0.06	0.07	0.08	0.05	0.06	0.06	0.05	0.06	0.07
AUC	0.95	0.96	0.94	0.93	0.96	0.94	0.95	0.96	0.94	0.93
Acc_by_package_fn	0.95	0.92	0.91	0.90	0.94	0.93	0.92	0.94	0.92	0.91

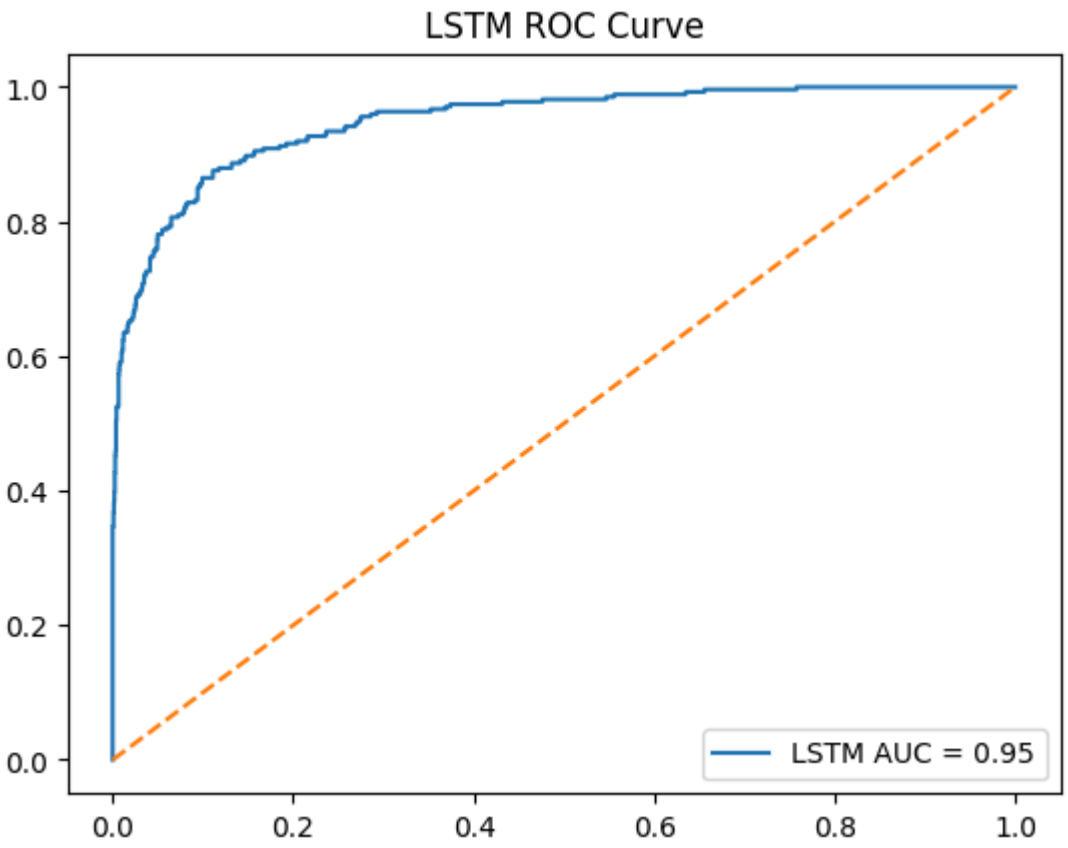
6. Results

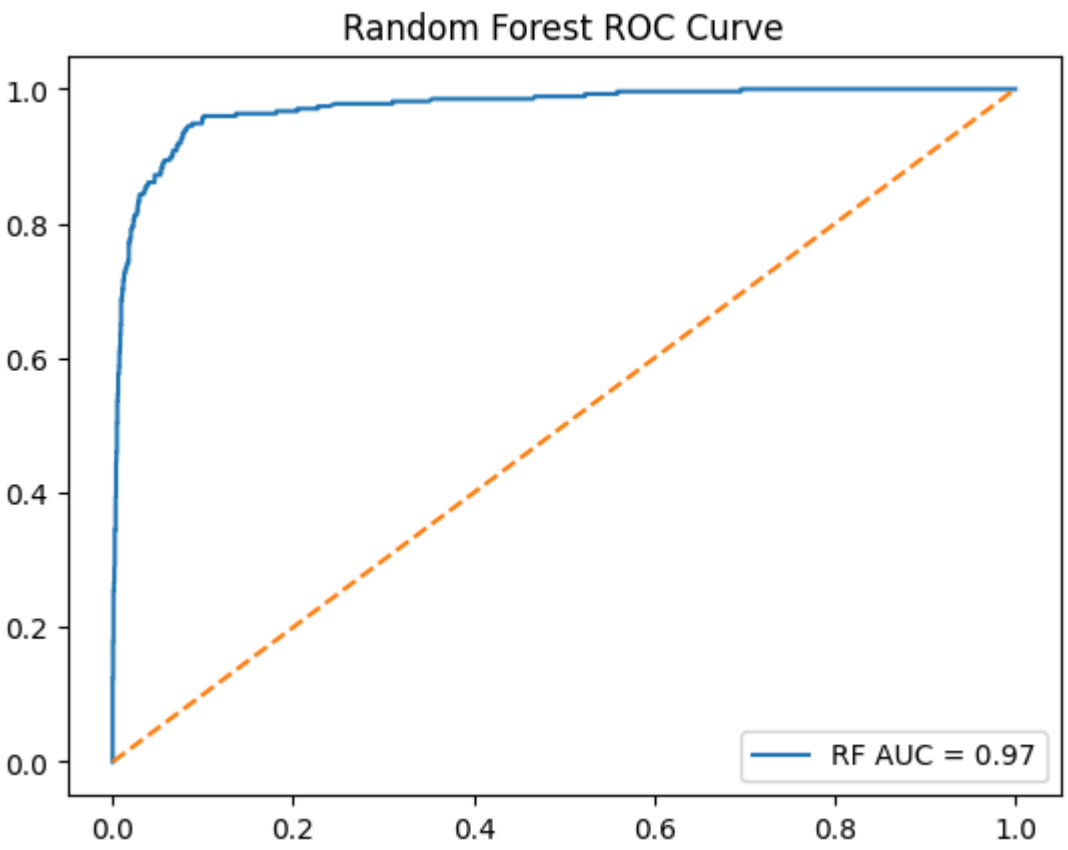
Average performance over 10 folds (rounded):

	KNN	RF	LSTM
TP	58.700	50.000	48.900
TN	358.800	481.500	468.500
FP	137.100	14.400	27.400
FN	5.100	13.800	14.900
TPR	0.920	0.784	0.766
TNR	0.724	0.971	0.945
FPR	0.276	0.029	0.055
FNR	0.080	0.216	0.234
Precision	0.300	0.778	0.649
F1_measure	0.453	0.780	0.700
Accuracy	0.746	0.950	0.924
Error_rate	0.254	0.050	0.076
BACC	0.822	0.877	0.856
TSS	0.644	0.754	0.711
HSS	0.339	0.752	0.658
Brier_score	0.173	0.042	0.060
AUC	0.890	0.972	0.945
Acc_by_package_fn	0.746	0.950	0.924

The performance of the three algorithms KNN, LSTM, and Random Forest is evaluated using their ROC curves and AUC scores. KNN achieved an AUC of 0.89, showing it can reasonably distinguish between safe and unsafe water, though it's less accurate than the other models. The LSTM model scored an AUC of 0.95, outperforming KNN by capturing complex patterns in the data, especially useful for time-series or detailed datasets. The best performer is the Random Forest, with an AUC of 0.97, demonstrating very high accuracy. Its ensemble of decision trees makes it more robust and reliable for predictions. Overall, Random Forest is the most accurate, LSTM captures complex data relationships well, and KNN remains a solid, though less precise, alternative.


KNN ROC Curve
KNN AUC = 0.89
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


LSTM ROC Curve
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
LSTM AUC = 0.95



7. Discussion

The Random Forest algorithm was the best-performing model in this project, achieving the highest AUC score of 0.97. Its success is due to its ability to handle high-dimensional data by combining multiple decision trees, which reduces overfitting and variance. This makes it well-suited for complex datasets with many features, like this one with 21 attributes. Its robustness against noise and feature interactions contributed to its superior accuracy, making it ideal for this binary classification task.

Several challenges arose during preprocessing, training, and evaluation. One major issue was class imbalance, with more safe (positive) samples than unsafe (negative) ones. To address this, SMOTE (Synthetic Minority Over-sampling Technique) was used to create synthetic samples of the minority class, helping the model learn from both classes effectively. Missing values were also a problem; these were filled using the median of each feature to maintain data consistency without bias. Since models like KNN and LSTM are sensitive to feature scales, all features were normalized to have a mean of 0 and a standard deviation of 1, ensuring better training and convergence.

For future improvements, additional feature engineering such as creating interaction terms or applying domain-specific transformations could provide better insights. Hyperparameter tuning, especially for Random Forest (like adjusting tree depth or the number of trees), could further enhance performance. Besides SMOTE, alternative methods like ADASYN or undersampling might better address class imbalance. For deep learning, exploring architectures like Bidirectional LSTM (Bi-LSTM) or Gated Recurrent Units (GRU) could capture sequential relationships more effectively. Additionally, using different cross-validation strategies, such as stratified or time-series cross-validation, could improve understanding of how well the models generalize.

Overall, Random Forest delivered the best results, but future work should focus on fine-tuning models, advanced feature engineering, and experimenting with different deep learning architectures to boost classification accuracy even further.

8. Conclusion

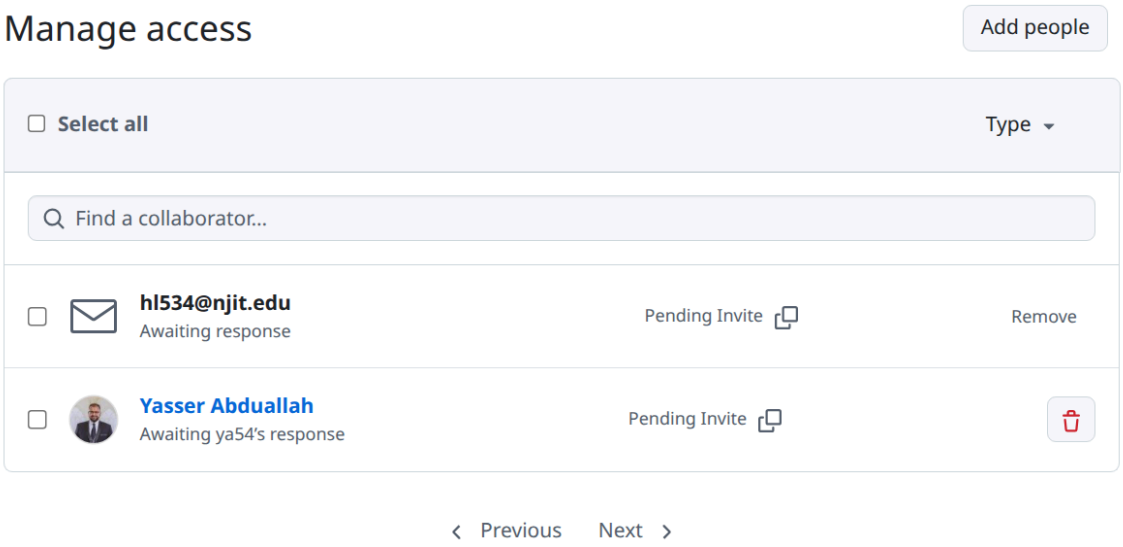
In summary, Random Forest performed the best among the models, as it was well-suited for this task and effectively managed bias and variance. The evaluation process, which included using SMOTE for class balancing and cross-validation for reliable performance measurement, emphasized the importance of proper data preprocessing. This approach helped ensure fair and accurate model assessments. Future improvements, such as advanced feature engineering and exploring more sophisticated architectures, could lead to even better results. Overall, the choice of algorithm and evaluation methods played a key role, with Random Forest demonstrating the strongest ability to generalize to new, unseen data.

9. References

Dataset

10. GitHub Link

https://github.com/SebaSkub/skubisz_sebastian_finaltermproj



11. Appendix

Operating System: Windows, macOS, or Linux

Python: 3.9 – 3.13

Shell: PowerShell, bash, or terminal

1.1 Create a Virtual Environment (recommended)

Windows (PowerShell):

```
Install python from Windows Store 3.13
python -m venv .venv
.venv\Scripts\activate
```

macOS/Linux (bash):

```
sudo apt install python3.13
python3 -m venv .venv
source .venv/bin/activate
```

1.2 Install Required Libraries

Install lightweight dependencies:

```
sudo apt install pip
pip install -r requirements.txt
```

1.3 Run Program

```
python training.py or python3 training.py
```

Operating System: Ubuntu 6.14.0-35-generic (x86_64)

Python Version: Python 3.13.3

Development Environment: VS Code (Version 1.105.1)

Hardware:

CPU: 13th Gen Intel Core i5-13600K (14 cores, 20 threads)

RAM: 30GB (10GB used, 14GB free, 7.2GB buffer/cache)

GPU: NVIDIA GeForce RTX 4070 (1056MiB / 12282MiB used)

Graphics Driver: NVIDIA Driver Version 580.95.05

CUDA Version: 13.0

Swap: 8.0GiB (65MiB used, 7.9GiB free)