MACHINE LEARNING FOR SAFE DRINKING WATER ASSESSMENT

Machine Learning Fundamentals Coursework

ABSTRACT

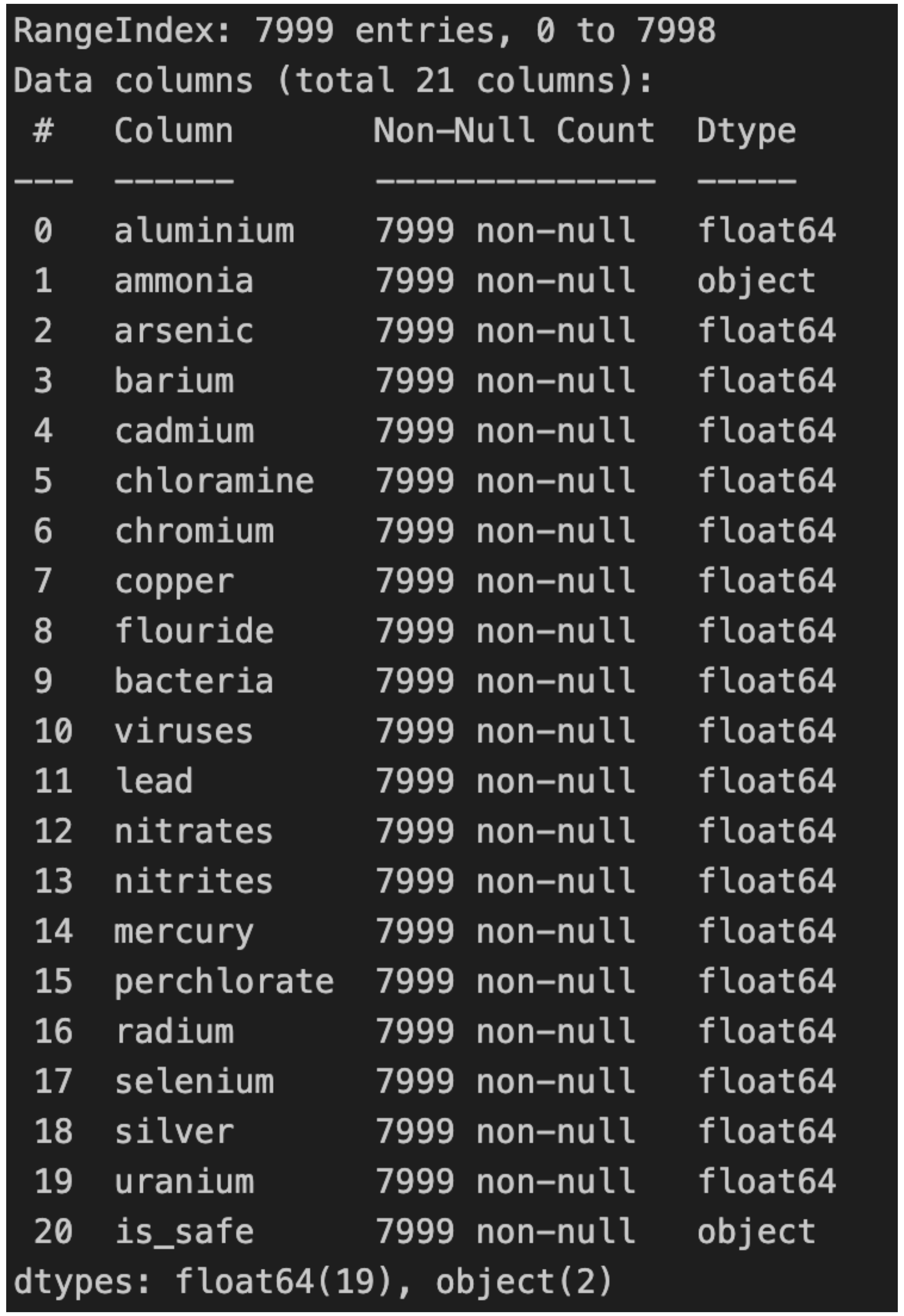
Access to clean drinking water is a fundamental human right, yet millions globally lack this essential resource. This project explores the potential of machine learning to assess water potability. We propose a machine learning model that can analyze various water quality parameters to predict the presence of contaminants and classify water as safe or unsafe for consumption.

PROJECT OBJECTIVE

- This project will explore how ML can leverage a vast array of water quality data to classify water sources as safe or unsafe for drinking.
- We will compare machine learning algorithms to identify the most effective approach for accurately classifying water potability.
- By pinpointing the most critical water quality indicators, we will gain insights to develop target monitoring strategies, maximising efficiency, and effectiveness.
- This project bridge the gap between traditional methods and cutting-edge AI, paving teh way for faster, more efficient water safety assessments.

DATASET OVERVIEW

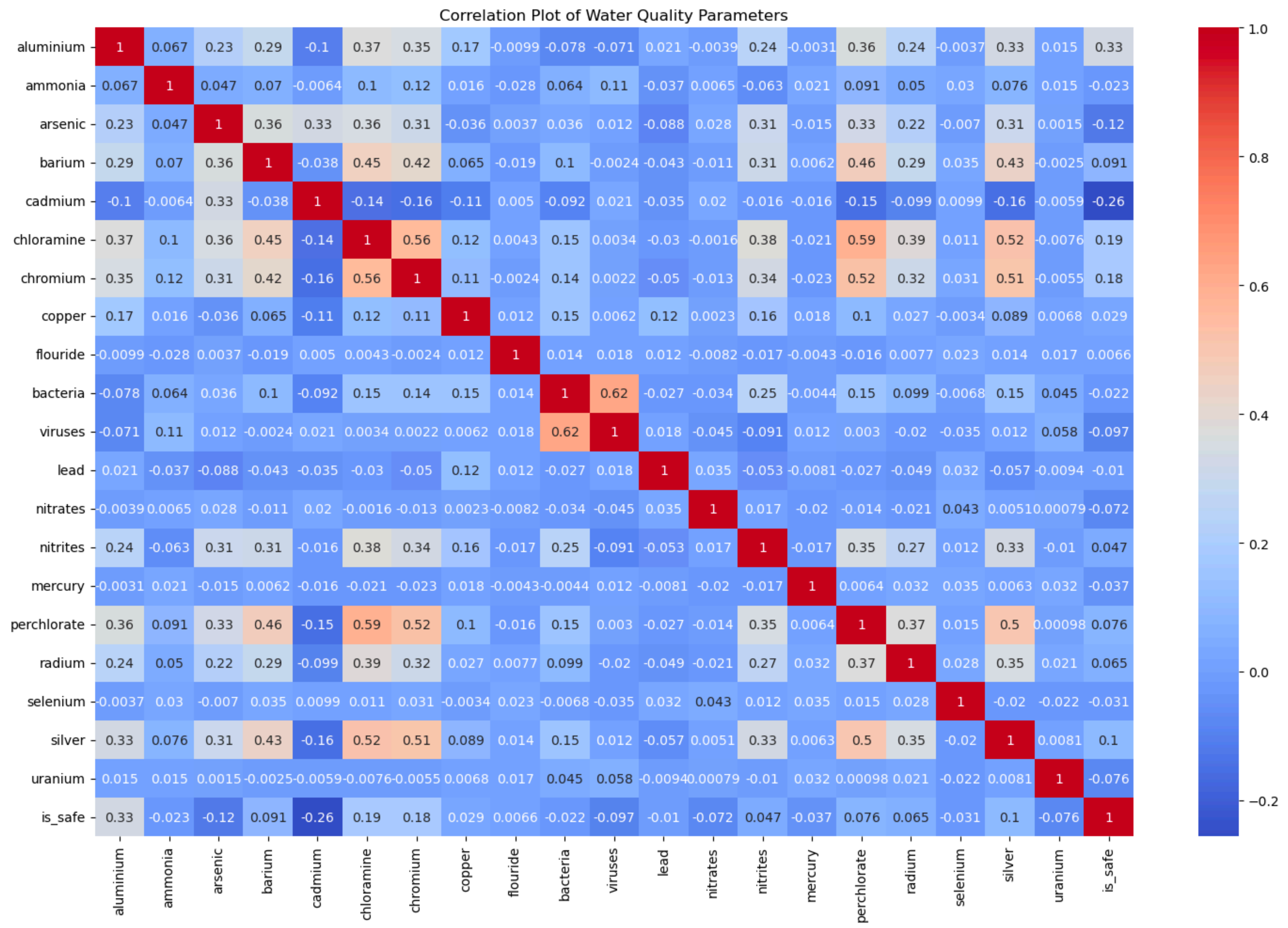
- This data obtained from Kaggle contains 21 different measurements for 8,000 water samples, allowing us to build powerful models to assess water safety.
- Let's dive into what these measurements mean:
  - Metals: Aluminum, Arsenic, Barium, Cadmium, Chromium, Copper, Lead, Mercury, Silver, Uranium, Selenium, Radium - These can be harmful at high levels.
  - Chemicals: Ammonia, Chloramine, Fluoride, Nitrates, Nitrites, Perchlorate - These can indicate pollution or treatment processes.
  - Biological: Bacteria, Viruses - These directly impact water safety.

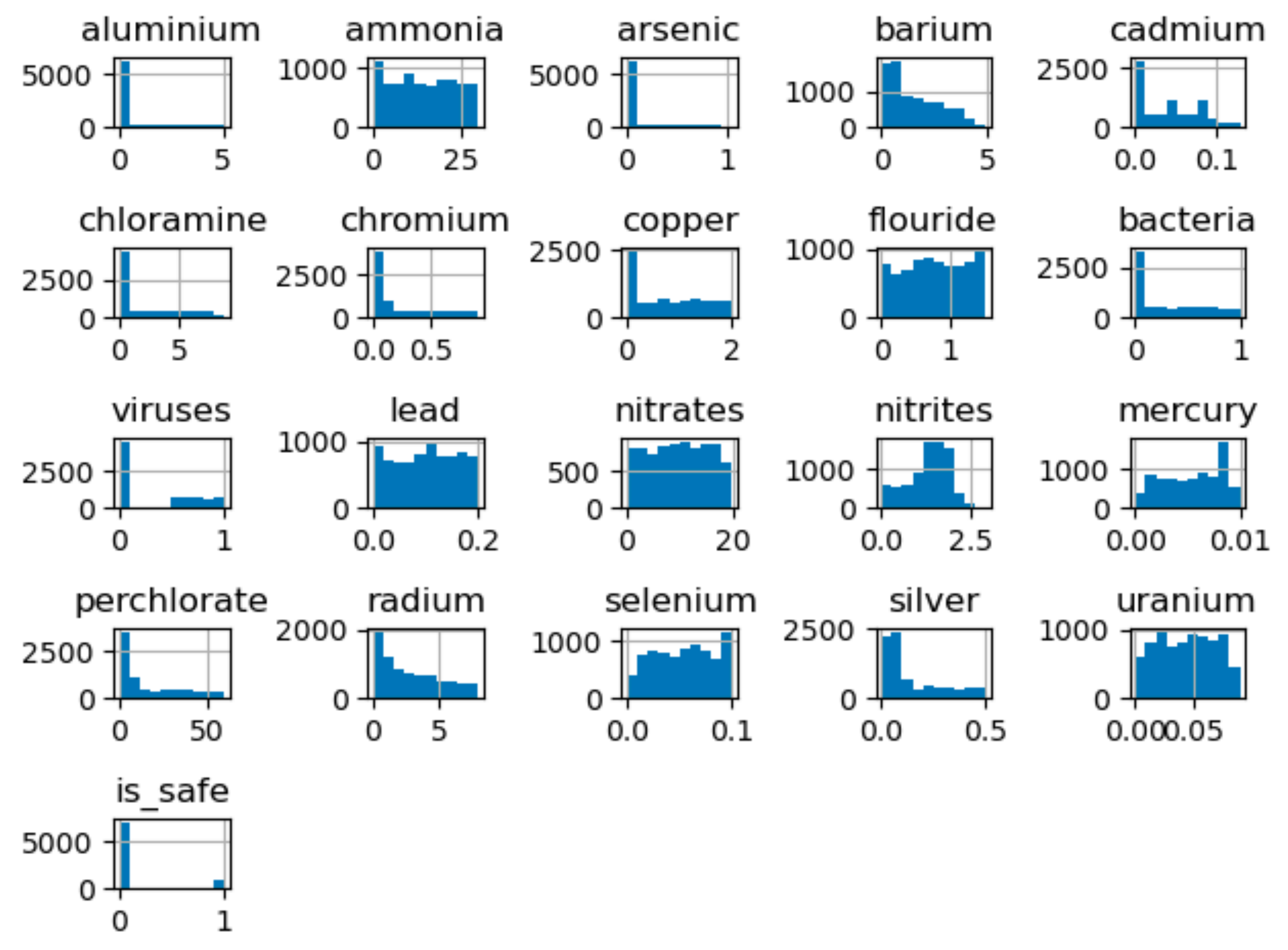


```
RangeIndex: 7999 entries, 0 to 7998
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   aluminium    7999 non-null   float64
 1   ammonia      7999 non-null   object
 2   arsenic      7999 non-null   float64
 3   barium       7999 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)
```

EXPLORATORY DATA ANALYSIS

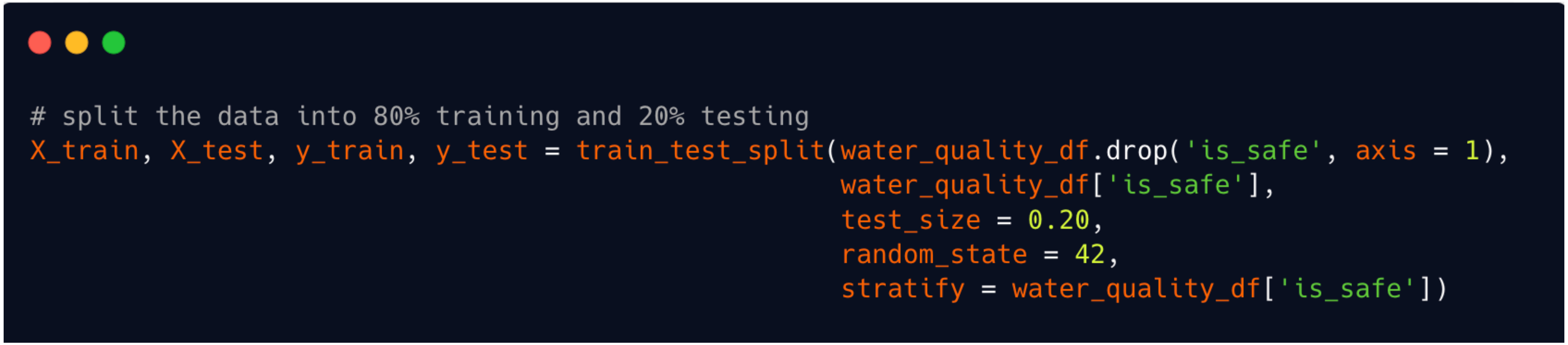
EDA provides insights into the dataset's characteristics and relationships.





DATA SPLITTING

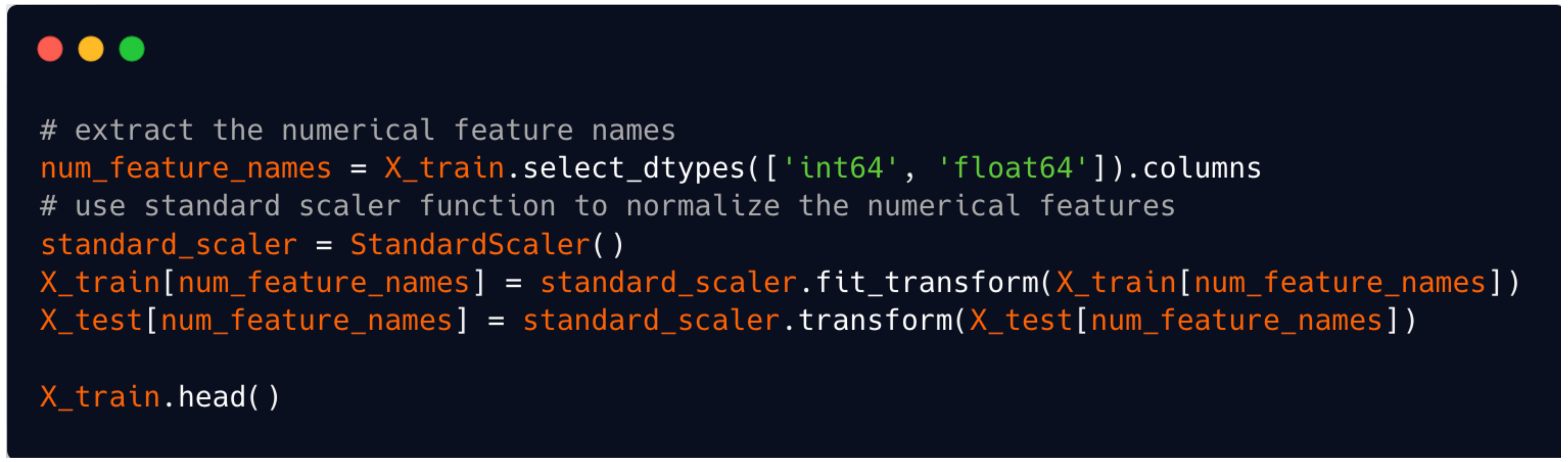
Data splitting aids in training and evaluating models effectively, allowing for robust validation and testing procedures.



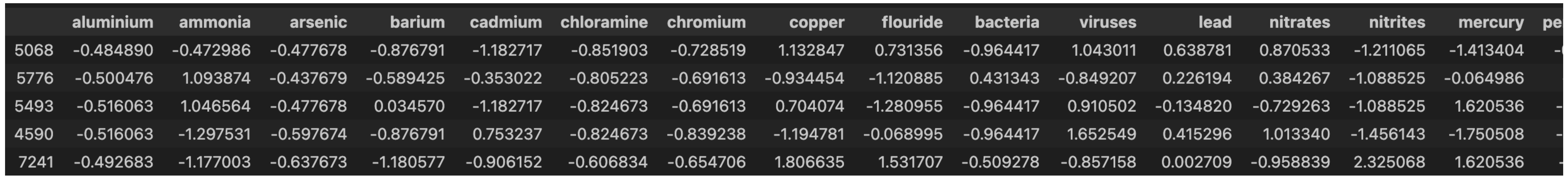
```python
# split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(water_quality_df.drop('is_safe', axis = 1),
                                                    water_quality_df['is_safe'],
                                                    test_size = 0.20,
                                                    random_state = 42,
                                                    stratify = water_quality_df['is_safe'])
```

DATA NORMALISATION

Standard scaling, a preprocessing technique in machine learning, is applied to features to ensure they have a mean of zero and a standard deviation of one. This normalisation process is crucial for many machine learning algorithms, especially those that involve distance calculations or gradient descent optimisation. By standardising the features, the algorithm treats all features equally in terms of their influence on the model, preventing any particular feature from dominating the others due to its scale. This not only helps algorithms converge faster but also enhances their performance by making them less sensitive to the scale of input features, leading to more stable and reliable models overall.

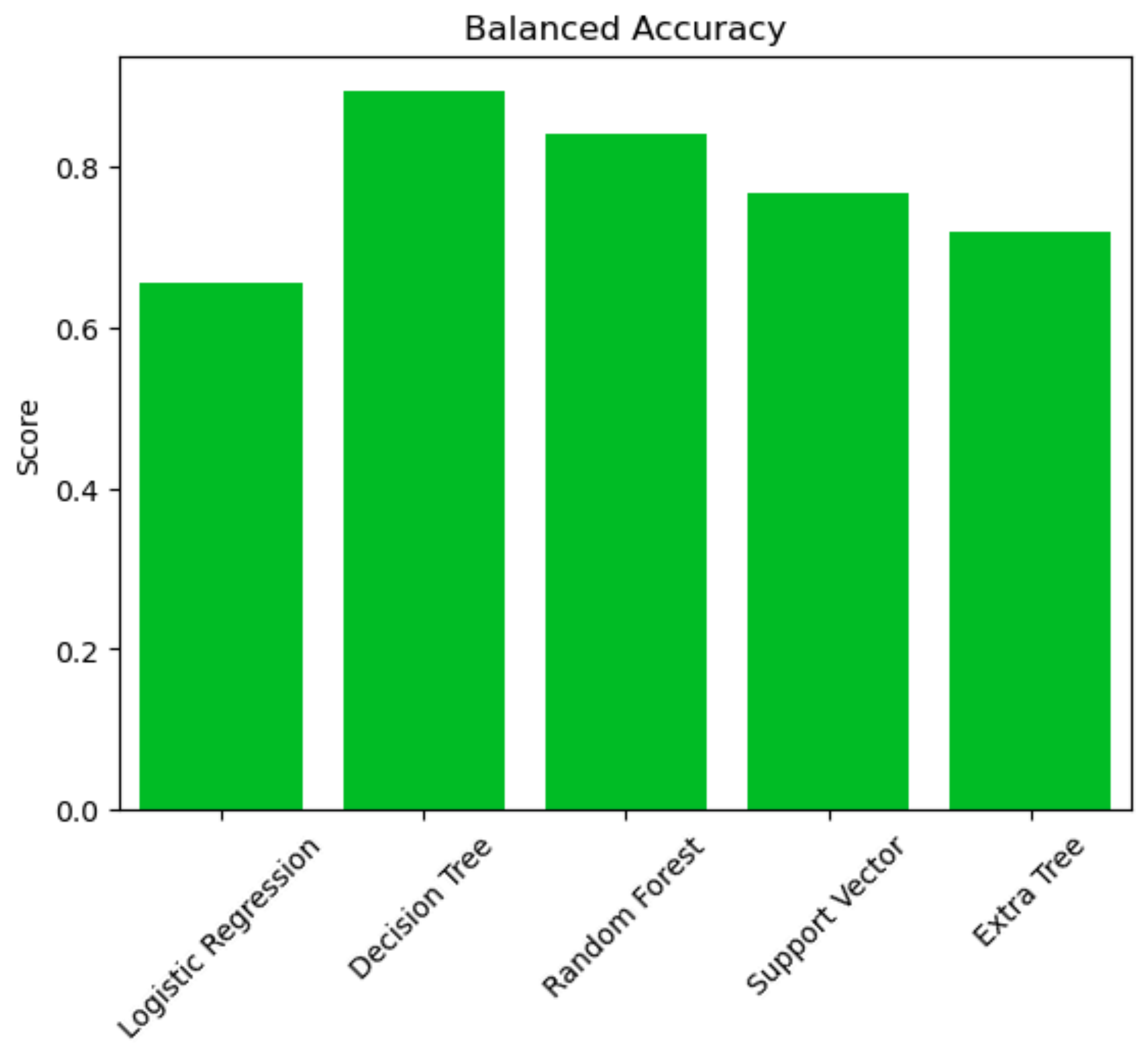


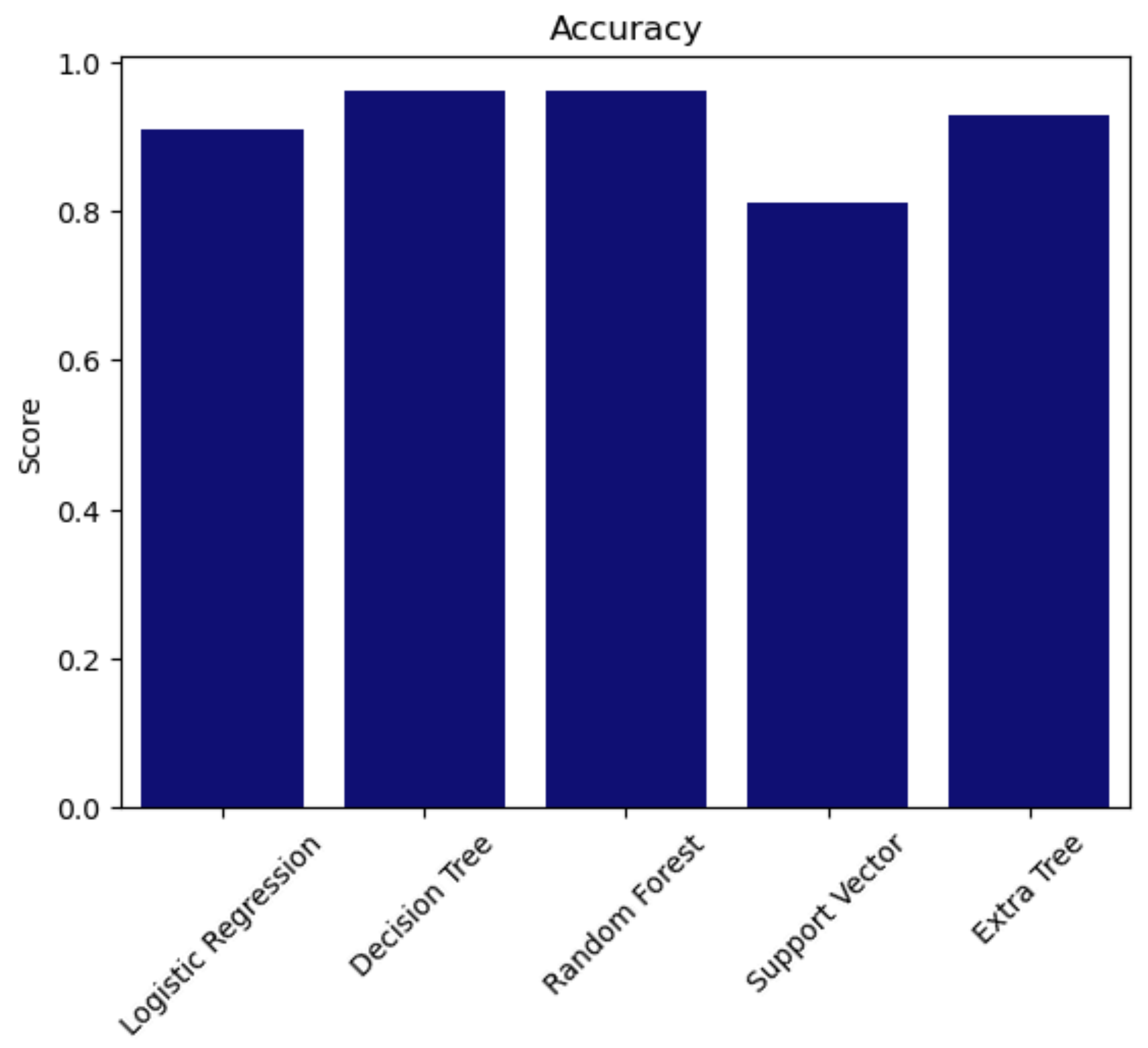
```
# extract the numerical feature names
num_feature_names = X_train.select_dtypes(['int64', 'float64']).columns
# use standard scaler function to normalize the numerical features
standard_scaler = StandardScaler()
X_train[num_feature_names] = standard_scaler.fit_transform(X_train[num_feature_names])
X_test[num_feature_names] = standard_scaler.transform(X_test[num_feature_names])

X_train.head()
```

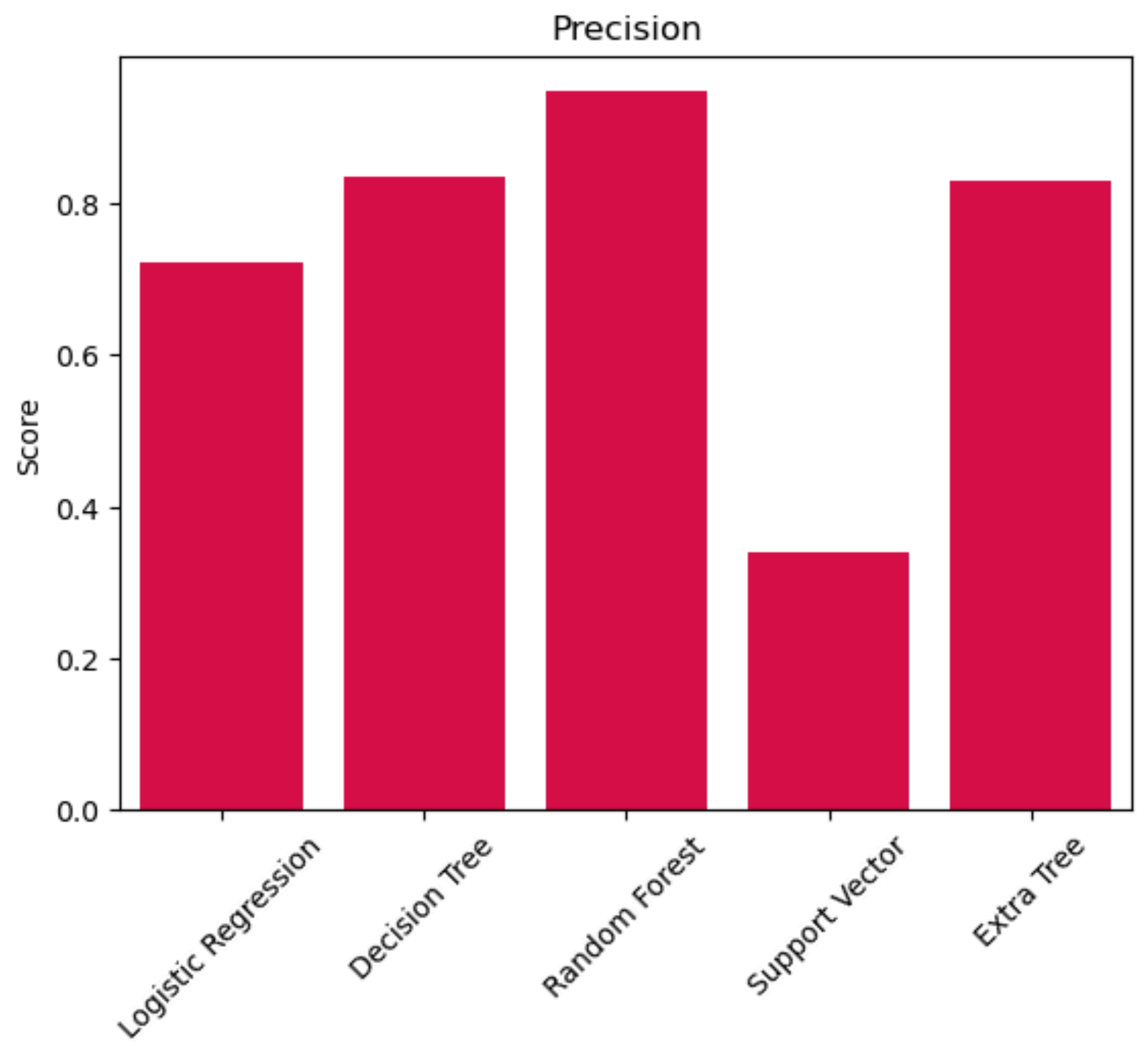


	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	viruses	lead	nitrates	nitrites	mercury	pe
5068	-0.484890	-0.472986	-0.477678	-0.876791	-1.182717	-0.851903	-0.728519	1.132847	0.731356	-0.964417	1.043011	0.638781	0.870533	-1.211065	-1.413404	-
5776	-0.500476	1.093874	-0.437679	-0.589425	-0.353022	-0.805223	-0.691613	-0.934454	-1.120885	0.431343	-0.849207	0.226194	0.384267	-1.088525	-0.064986	
5493	-0.516063	1.046564	-0.477678	0.034570	-1.182717	-0.824673	-0.691613	0.704074	-1.280955	-0.964417	0.910502	-0.134820	-0.729263	-1.088525	1.620536	-
4590	-0.516063	-1.297531	-0.597674	-0.876791	0.753237	-0.824673	-0.839238	-1.194781	-0.068995	-0.964417	1.652549	0.415296	1.013340	-1.456143	-1.750508	-
7241	-0.492683	-1.177003	-0.637673	-1.180577	-0.906152	-0.606834	-0.654706	1.806635	1.531707	-0.509278	-0.857158	0.002709	-0.958839	2.325068	1.620536	-

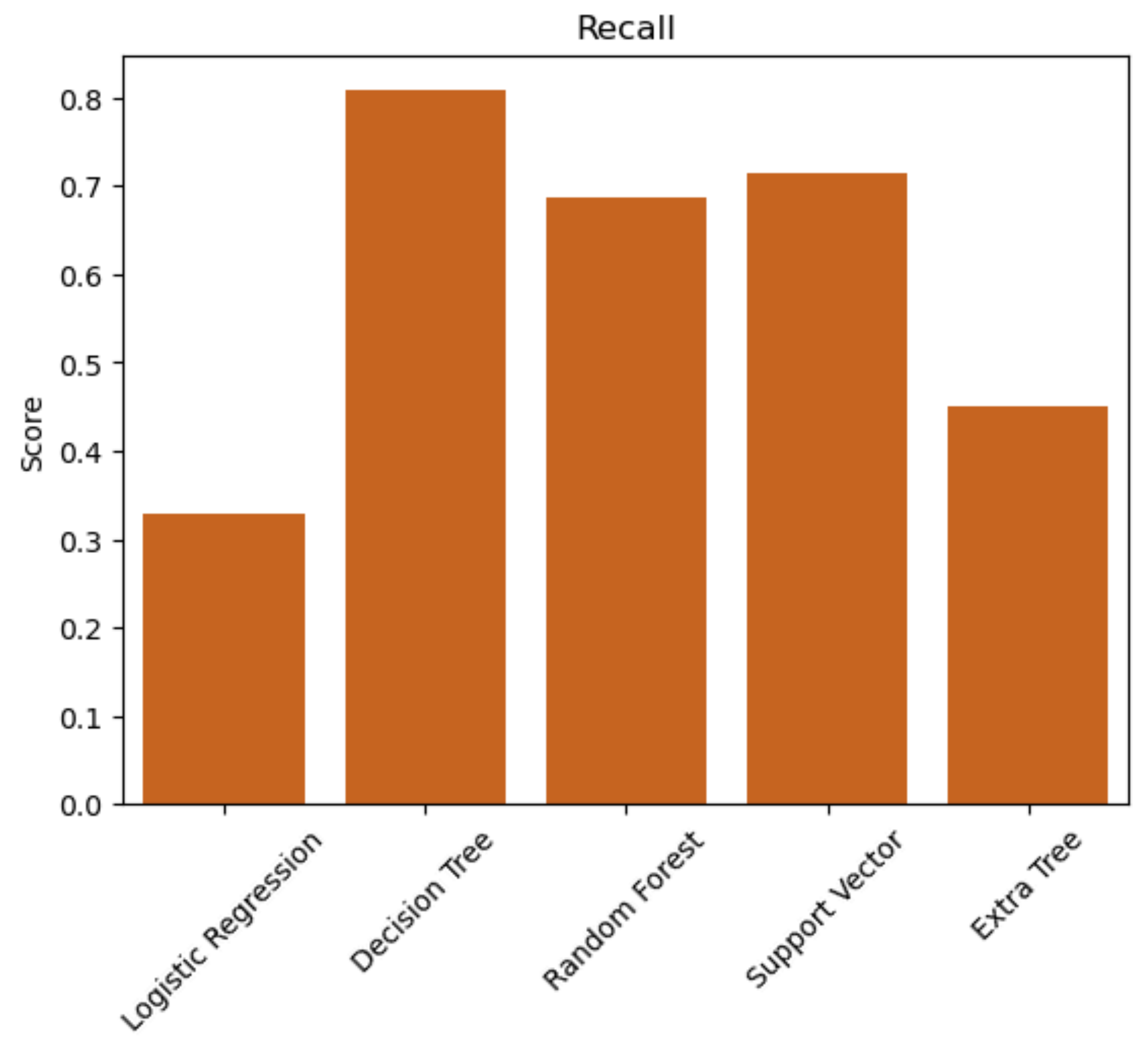
BASELINE SELECTION – MACHINE LEARNING ALGORITHMS

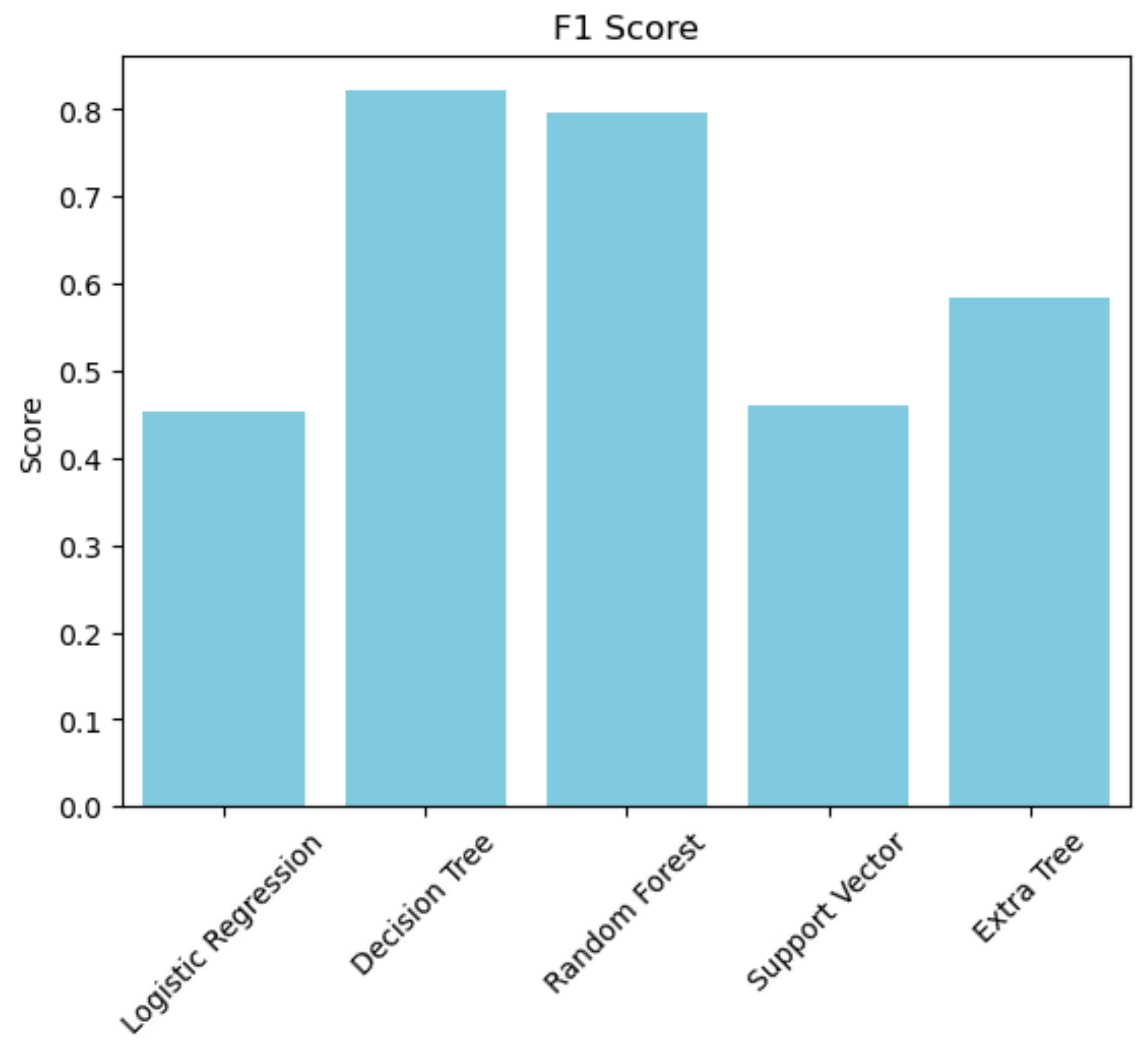






BASELINE SELECTION – MACHINE LEARNING ALGORITHMS





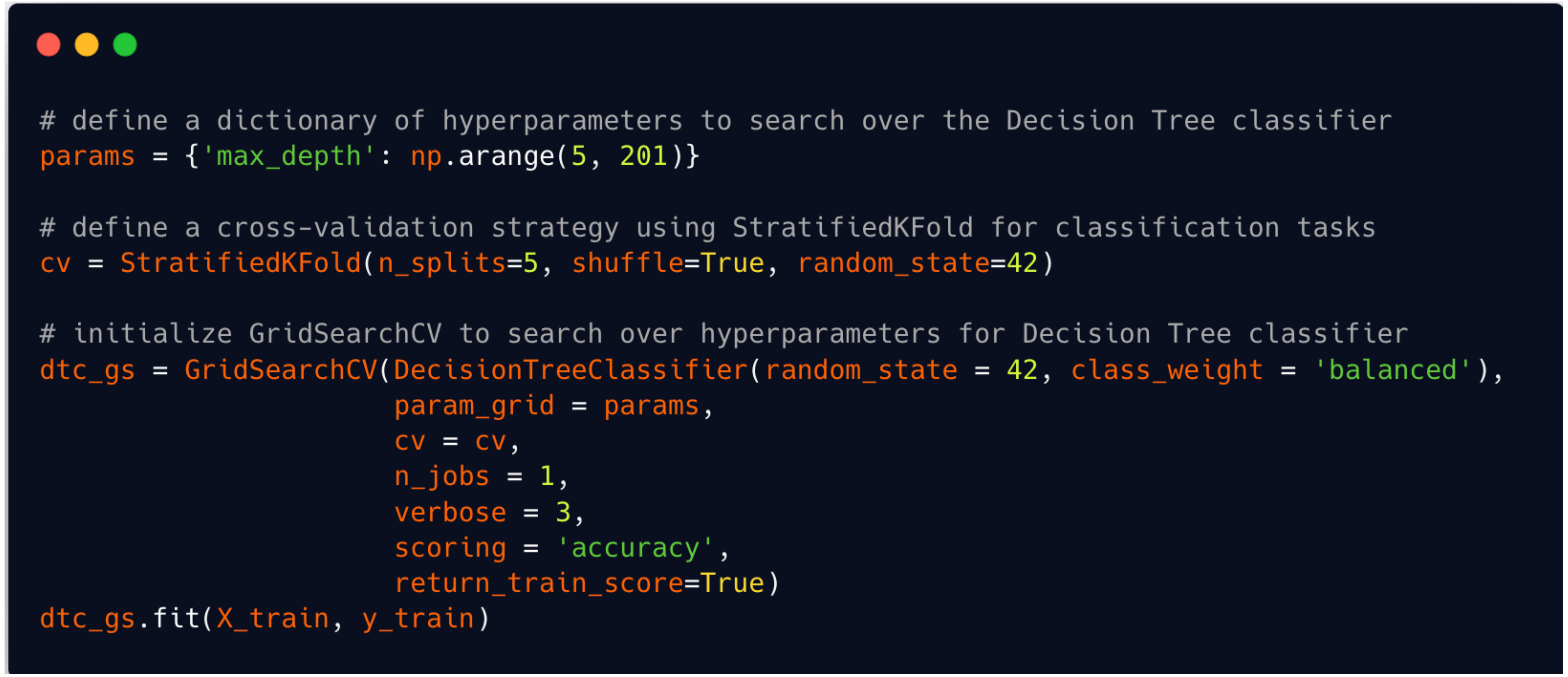
BASELINE SELECTION - MACHINE LEARNING ALGORITHMS

- Tree-based models demonstrate superior performance in accuracy and F1 scores compared to linear models. For instance, Decision Trees (DT) achieved an accuracy of 0.96 and an F1 score of 0.90, while Random Forest (RF) attained the similar accuracy and F1 score.
- Notably, tree-based models exhibit lower overfitting on imbalanced data because they require less data for training, making even a few samples from the minority class sufficient.
- Conversely, linear models like Logistic Regression (LR) and Support Vector Machines (SVM) also exhibit good accuracy scores, achieving same scores of 0.91, respectively.
- However, their F1 scores were lower, each registering a 0.71 and 0.70, respectively. This disparity stems from linear models' need for larger datasets to train effectively, leading to a good fit for majority class data but an underfit for minority class data.

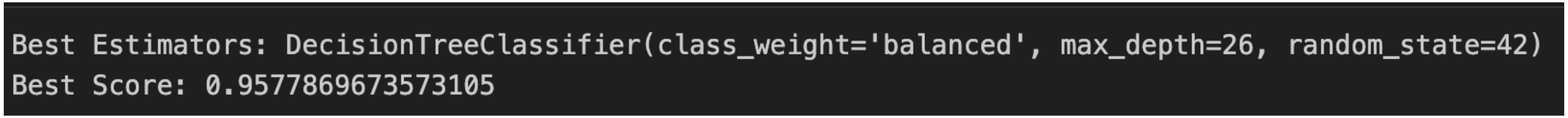
BASELINE SELECTION – MACHINE LEARNING ALGORITHMS

- In general, the models exhibited overfitting on the majority class data, resulting in poor performance on minority class samples.
- This overfitting caused the accuracy score to be considerably higher than the F1 score.
- Notably, the models performed notably well in predicting the 'not safe' (0) class, with similar precision, recall, and F1 scores across all models.
- However, for the 'safe' class (1), the models showed inadequate performance due to underfitting.
- This was because they did not receive sufficient training samples for the 'safe' class, leading to poor performance on minority class predictions.

HYPERPARAMETERS TUNING ON DECISION TREE CLASSIFIER

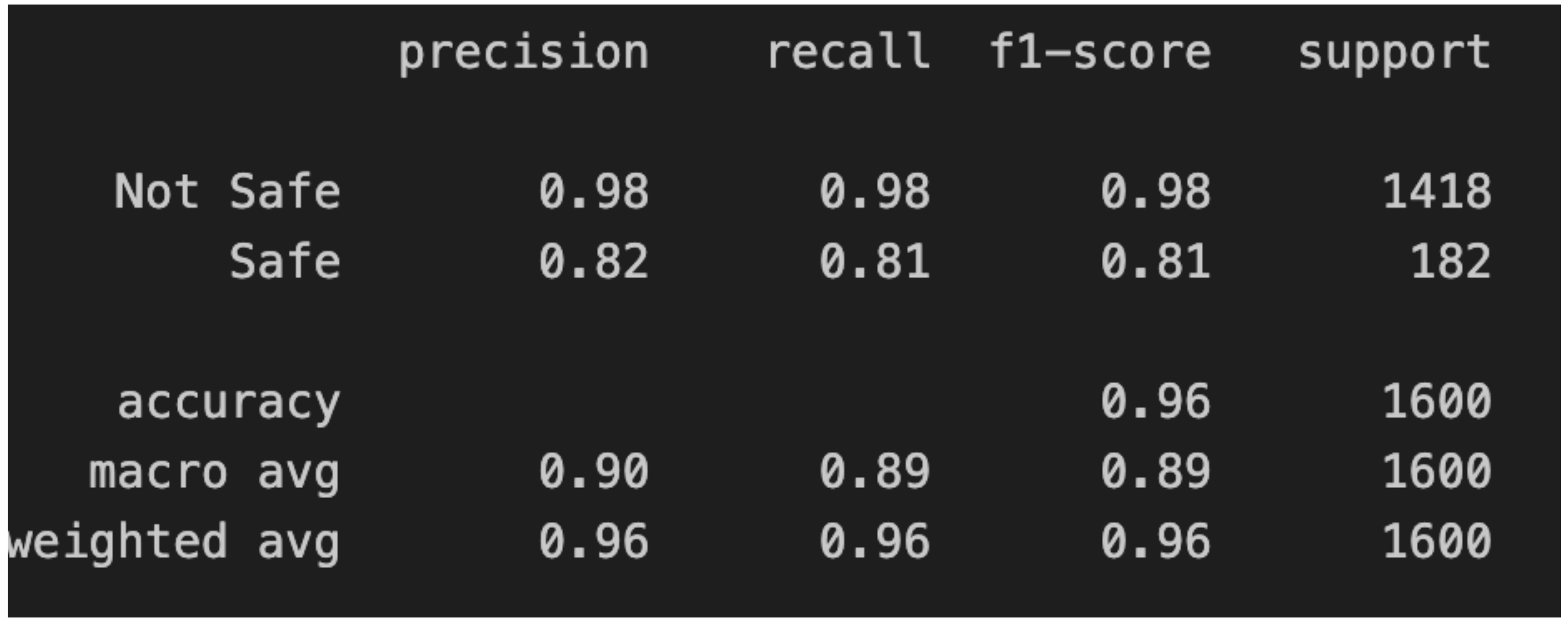


```python
# define a dictionary of hyperparameters to search over the Decision Tree classifier
params = {'max_depth': np.arange(5, 201)}

# define a cross-validation strategy using StratifiedKFold for classification tasks
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# initialize GridSearchCV to search over hyperparameters for Decision Tree classifier
dtc_gs = GridSearchCV(DecisionTreeClassifier(random_state = 42, class_weight = 'balanced'),
                      param_grid = params,
                      cv = cv,
                      n_jobs = 1,
                      verbose = 3,
                      scoring = 'accuracy',
                      return_train_score=True)
dtc_gs.fit(X_train, y_train)
```



```
Best Estimators: DecisionTreeClassifier(class_weight='balanced', max_depth=26, random_state=42)
Best Score: 0.9577869673573105
```

EVALUATE HYPERPARAMETERS TUNING ON DECISION TREE CLASSIFIER

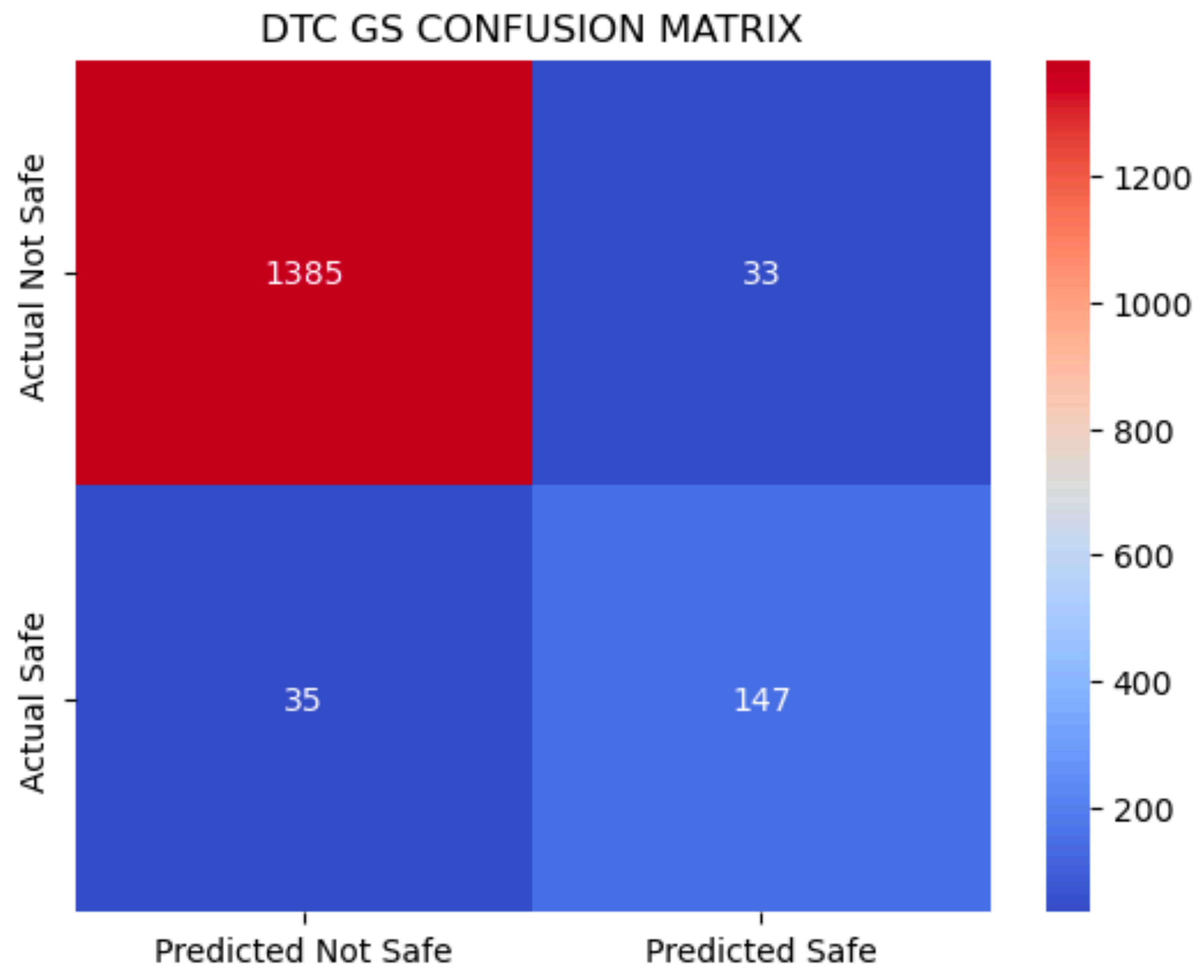


```
              precision    recall  f1-score   support

    Not Safe       0.98      0.98      0.98      1418
        Safe       0.82      0.81      0.81       182

    accuracy                           0.96      1600
   macro avg       0.90      0.89      0.89      1600
weighted avg       0.96      0.96      0.96      1600
```

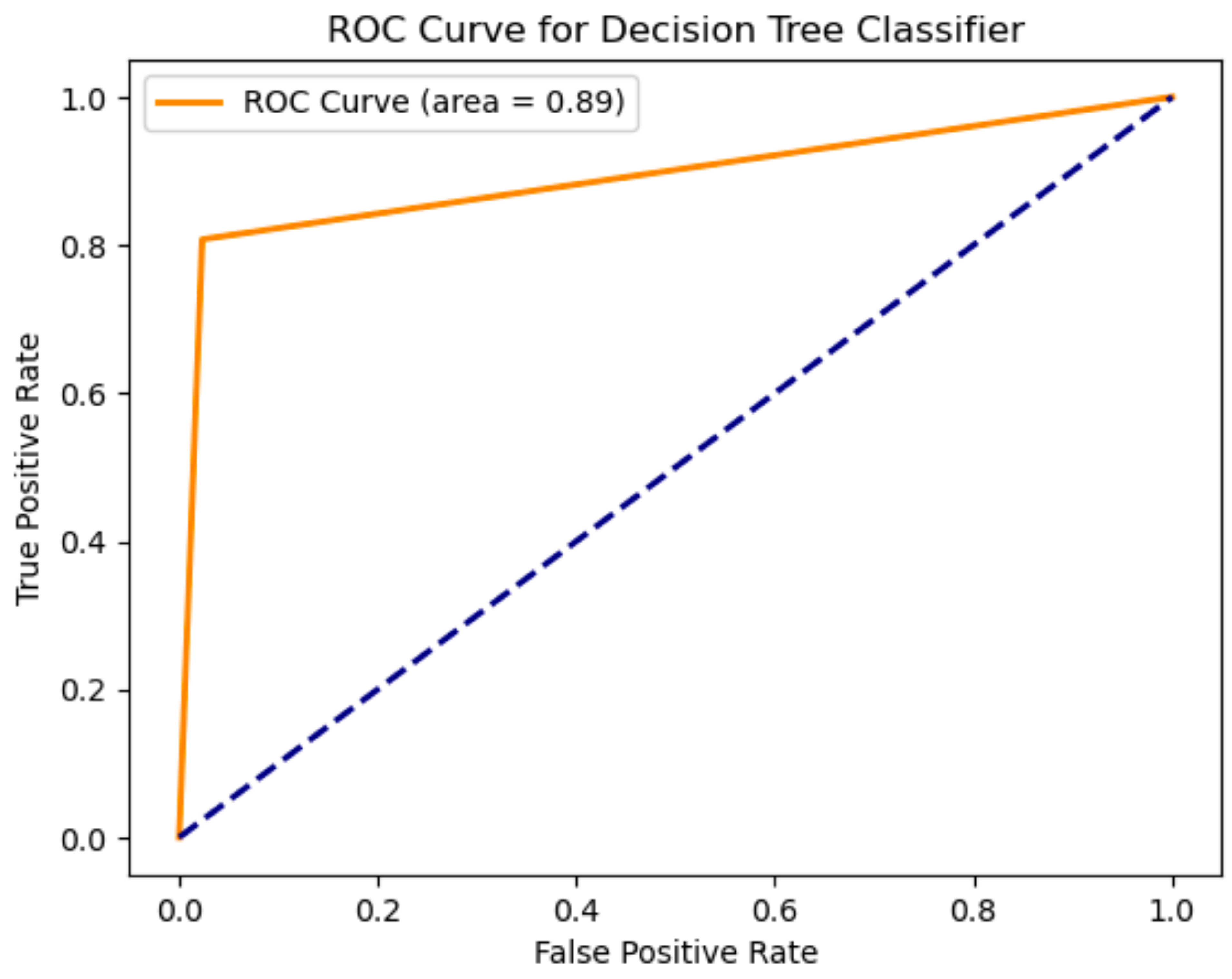
- The classification results indicate a high level of performance overall.
- For the "Not Safe" class, precision, recall, and F1-score are all high, at 0.98, suggesting that the model effectively identifies instances belonging to this class with few false positives and negatives.
- Similarly, for the "Safe" class, precision, recall, and F1-score are slightly lower but still respectable, at 0.84, indicating that while the model identifies instances of this class with good accuracy, there are some false positives and negatives. The accuracy of 0.96 reflects the overall proportion of correctly classified instances in the dataset. The macro and weighted averages of precision, recall, and F1-score are both high, indicating a balanced performance across classes.
- Overall, the model demonstrates strong predictive capabilities with a weighted average F1-score of 0.96, suggesting its suitability for classification tasks.

EVALUATE HYPERPARAMETERS TUNING ON DECISION TREE CLASSIFIER

- The top-left cell (1385) indicates the number of instances that were correctly classified as belonging to the "Not Safe" class (True Negatives).
- The top-right cell (33) indicates the number of instances that were incorrectly classified as "Safe" when they actually belonged to the "Not Safe" class (False Positives).
- The bottom-left cell (35) indicates the number of instances that were incorrectly classified as "Not Safe" when they actually belonged to the "Safe" class (False Negatives).
- The bottom-right cell (147) indicates the number of instances that were correctly classified as belonging to the "Safe" class (True Positives).
- From this confusion matrix, we can see that the model performs well overall, with a high number of true positives and true negatives.
- However, there are some misclassifications, as indicated by the false positives and false negatives. Overall, the model appears to have a higher accuracy in predicting the "Not Safe" class compared to the "Safe" class, as there are fewer false positives and more true negatives.



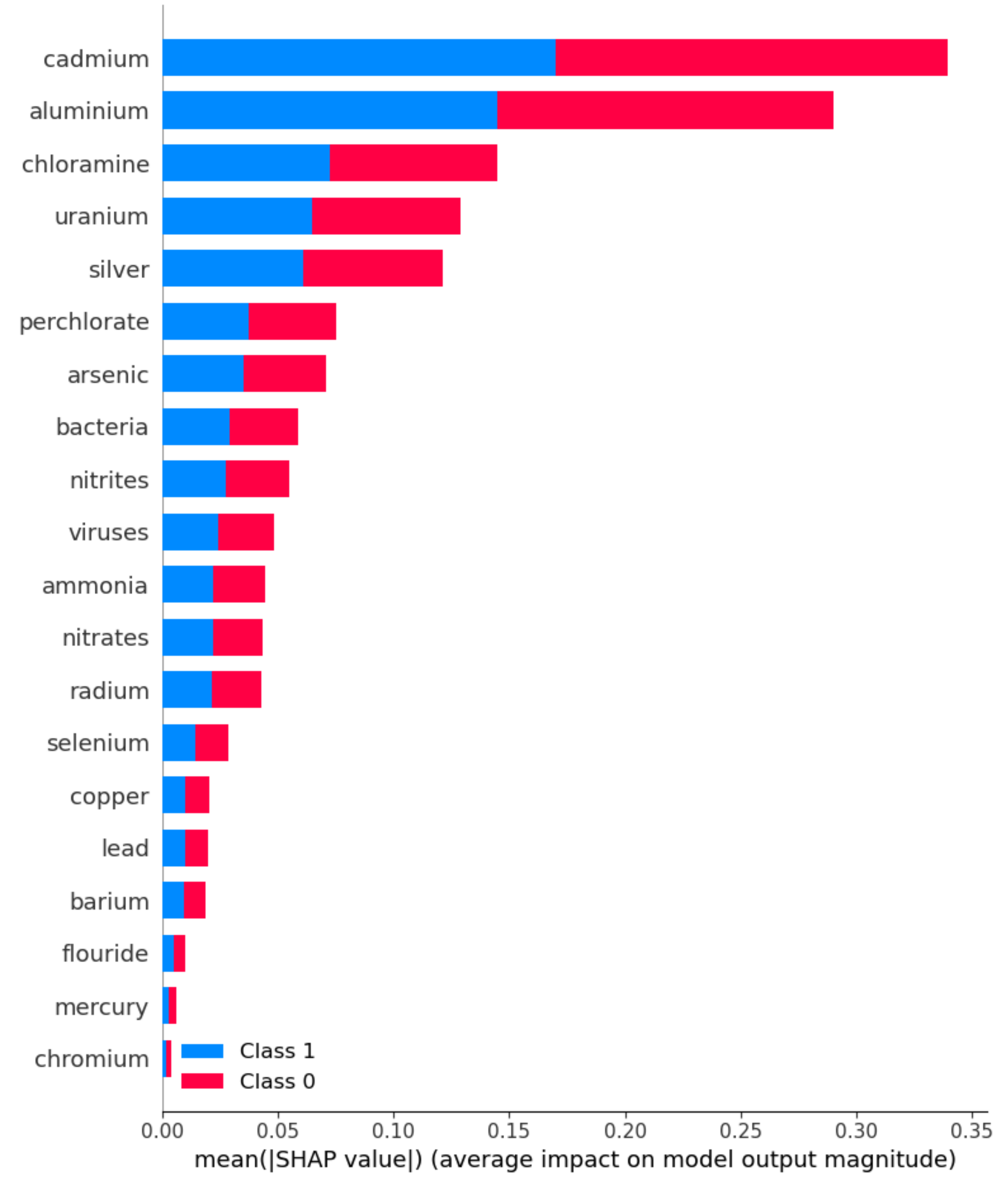
EVALUATE HYPERPARAMETERS TUNING ON DECISION TREE CLASSIFIER



- ➤A higher AUC value (closer to 1) suggests better discrimination between the positive and negative classes, indicating that the model can effectively distinguish between them.

FEATURE IMPORTANCES ON OPTIMISED DECISION TREE CLASSIFIER

- Based on the information provided above, it seems that cadmium, aluminium, chloramine, uranium, and silver have the highest average SHAP values in a model predicting water quality. This suggests that these features are the influential in determining whether the water is classified as safe or not safe.
  - Cadmium, Uranium: These elements are known to be highly toxic and can cause severe health problems even at low levels. Their high SHAP values suggest they strongly influence the model's prediction of unsafe water.
  - Aluminium: While aluminium can be harmful at high concentrations, it's often present in treated water at regulated levels. The model might be sensitive to aluminium levels, but further analysis is needed to determine its impact on the "safe" classification.
  - Chloramine: This disinfectant is commonly used in water treatment. Its SHAP value could indicate that the model considers chlroamine presence as a positive factor for water safety, but it's essential to ensure proper levels are maintained to avoid harmful byproducts.
  - Silver: Colloidal silver is sometimes used for water purification, but its effectiveness and safety are debated. A high SHAP value for silver might require further investigation into the model's training data and assumptions.
- SHAP values are for feature importance, not absolute safety. Just because a feature has high SHAP value does not mean it is sole determinant of safety. Other factors and interactions might also play a role.



FURTHER DISCUSSIONS

This isn't just about our research; it's about making clean water a reality! Our models show promise for real-time water quality monitoring, which could be a major breakthrough in safeguarding drinking water. Imagine being able to instantly identify risky water sources! This could revolutionise public health, especially in areas with unpredictable water quality. By using these machine learning tools, authorities can effectively monitor water, identify problems quickly, and take action to protect people's health.

LIMITATIONS

- While our study shows promise, it's important to acknowledge some limitations. We used historical data, which might not reflect the ever-changing nature of water quality. Additionally, we relied on standard safety thresholds that may not apply everywhere due to local factors like geography and environment.
- Moving forward, incorporating real-time data and considering regional variations would strengthen our approach. Overall, this research highlights the potential of machine learning for water quality assessment. By making the results understandable, identifying key factors, and demonstrating practical applications, we emphasize the value of automated systems in safeguarding clean drinking water.
- The limitations identified here serve as a guide for future research, paving the way for continuous improvement in water quality management and global public health.

FUTURE SCOPE

1. The next step is to make our water quality assessments even more dynamic. Imagine incorporating live data feeds from sensors and other sources directly into our machine learning models. This constant flow of information would allow the models to learn and adapt in real-time, becoming more adept at spotting water quality issues the moment they arise. By ditching the reliance solely on historical data, we can achieve faster and more precise interventions, safeguarding public health more effectively.

2. Investigate the potential of using powerful AI methods, like convolutional neural networks and recurrent neural networks, to unlock insights from intricate water quality data that has many variables.

3. Let's build AI tools that explain their reasoning! This will help people who manage water quality trust the decisions these models make. Tools like LIME and SHAP can shed light on how these models arrive at their predictions, making them more transparent and reliable.