SZABIST

Water Quality Analysis

Artificial Intelligence Project Report

Course Name: CSC4101 - Artificial Intelligence
Course Instructor: Sheikh Usama Khalid

Course Name: CSCL4101 - Lab Artificial Intelligence
Course Instructor: Wali Muhammad Khubaib

Group Member	Reg No.
Mustan Ali	2112121
Umer Amir	2112241

Table Of Content

Introduction

This report focuses on the analysis and prediction of water quality using a dataset containing concentrations of various contaminants. Ensuring safe drinking water is crucial for public health, and predictive models can help in early detection of water safety issues. To achieve this, we utilized several machine learning algorithms to classify water samples as safe or unsafe. The chosen algorithms include Decision Tree, Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), and Logistic Regression. Each of these algorithms offers unique advantages and challenges, which are explored in the context of this dataset. This report provides an overview of the dataset and a brief explanation of each algorithm's methodology and suitability for the task.

Dataset Overview

The dataset consists of 21 features representing the concentrations of various contaminants in water samples. The target variable is a binary indicator (is_safe), where 1 denotes safe water and 0 indicates unsafe water. Below is the detailed information about the features:

Contaminant	Dangerous Threshold (ppm)
Aluminium	2.8
Ammonia	32.5
Arsenic	0.01
Barium	2
Cadmium	0.005
Chloramine	4
Chromium	0.1
Copper	1.3
Fluoride	1.5
Bacteria	0
Viruses	0
Lead	0.015
Nitrates	10
Nitrites	1
Mercury	0.002
Perchlorate	56
Radium	5
Selenium	0.5
Silver	0.1
Uranium	0.3
Is_Safe	Class attribute (0: not safe, 1: safe)

Machine Learning Algorithms

K-Nearest Neighbors (KNN)

Overview: KNN is a simple, instance-based learning algorithm where the classification of a sample is determined by the majority class among its k-nearest neighbors in the feature space. It is widely used for classification and regression tasks.
Advantages: Simple to implement, intuitive to understand, and non-parametric, meaning it makes no assumptions about the underlying data distribution.
Disadvantages: Computationally expensive at prediction time due to the need to compute distances between the sample and all training data, sensitive to the choice of k, and can be affected by irrelevant or redundant features.

Decision Tree

Overview: Decision Trees are hierarchical models used for classification and regression tasks. They split the data into subsets based on feature values, creating a tree structure with nodes representing feature tests and branches representing the outcome of these tests.
Advantages: Easy to understand and interpret, handles both numerical and categorical data, and requires little data preprocessing.
Disadvantages: Prone to overfitting, especially with complex trees, and can be unstable due to small variations in data.

Multi-Layer Perceptron (MLP)

Overview: MLP is a type of artificial neural network consisting of an input layer, one or more hidden layers, and an output layer. Each layer is composed of neurons that are fully connected to neurons in the next layer. MLPs are used for complex pattern recognition tasks.
Advantages: Capable of capturing complex non-linear relationships, highly flexible, and can model complex data distributions.
Disadvantages: Computationally intensive, sensitive to the choice of hyperparameters, requires a large amount of data for training, and is often considered a black-box model due to difficulty in interpreting the weights.

Logistic Regression

Overview: Logistic Regression is a linear model used for binary classification tasks. It models the probability that a given input belongs to a particular class using a logistic function. The model outputs probabilities that can be mapped to binary outcomes.
Advantages: Simple and easy to implement, provides interpretable model coefficients, works well for linearly separable data, and outputs probabilities which can be useful for decision making.
Disadvantages: Assumes a linear relationship between the features and the log odds of the target, may not perform well with complex, non-linear data, and can be affected by multicollinearity among features.

Code Implementation and Output

K-Nearest Neighbors (KNN)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
```

```
df=pd.read_csv('../datasets/WaterQuality.csv')
df.head()
```

	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	...	lead	nitrates	nitrites	me
0	1.65	9.08	0.04	2.85	0.007	0.35	0.83	0.17	0.05	0.20	...	0.054	16.08	1.13	
1	2.32	21.16	0.01	3.31	0.002	5.28	0.68	0.66	0.90	0.65	...	0.100	2.01	1.93	
2	1.01	14.02	0.04	0.58	0.008	4.24	0.53	0.02	0.99	0.05	...	0.078	14.16	1.11	
3	1.36	11.33	0.04	2.96	0.001	7.23	0.03	1.66	1.08	0.71	...	0.016	1.41	1.29	
4	0.92	24.33	0.03	0.20	0.006	2.67	0.69	0.57	0.61	0.13	...	0.117	6.74	1.11	

5 rows × 21 columns

```
df.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
```

```
df.replace('#NUM!', float('nan'), inplace=True)
df = df.dropna()
```

```
x = df.drop(["is_safe"], axis="columns")
y = df.is_safe

scaler = StandardScaler()
x = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=12)

knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(x_train, y_train)

print(knn.score(x_test, y_test))
```

```
0.90375
```

Decision Tree

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
df = pd.read_csv("../datasets/WaterQuality.csv")
df.head()
```

	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	...	lead	nitrates	nitrites	me
0	1.65	9.08	0.04	2.85	0.007	0.35	0.83	0.17	0.05	0.20	...	0.054	16.08	1.13	
1	2.32	21.16	0.01	3.31	0.002	5.28	0.68	0.66	0.90	0.65	...	0.100	2.01	1.93	
2	1.01	14.02	0.04	0.58	0.008	4.24	0.53	0.02	0.99	0.05	...	0.078	14.16	1.11	
3	1.36	11.33	0.04	2.96	0.001	7.23	0.03	1.66	1.08	0.71	...	0.016	1.41	1.29	
4	0.92	24.33	0.03	0.20	0.006	2.67	0.69	0.57	0.61	0.13	...	0.117	6.74	1.11	

5 rows × 21 columns

```
df.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
```

```
df.replace("#NUM!", float("nan"), inplace=True)
df = df.dropna()
```

```
x = df.drop(["is_safe"], axis="columns")
y = df.is_safe

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=10)

dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

print(dt.score(x_test, y_test))
```

```
0.956875
```

Multi-Layer Perceptron (MLP)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
```

```
df = pd.read_csv("../datasets/WaterQuality.csv")
df.head()
```

	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	...	lead	nitrates	nitrites	me
0	1.65	9.08	0.04	2.85	0.007	0.35	0.83	0.17	0.05	0.20	...	0.054	16.08	1.13	
1	2.32	21.16	0.01	3.31	0.002	5.28	0.68	0.66	0.90	0.65	...	0.100	2.01	1.93	
2	1.01	14.02	0.04	0.58	0.008	4.24	0.53	0.02	0.99	0.05	...	0.078	14.16	1.11	
3	1.36	11.33	0.04	2.96	0.001	7.23	0.03	1.66	1.08	0.71	...	0.016	1.41	1.29	
4	0.92	24.33	0.03	0.20	0.006	2.67	0.69	0.57	0.61	0.13	...	0.117	6.74	1.11	

5 rows × 21 columns

```
df.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
```

```
df.replace("#NUM!", float("nan"), inplace=True)
df = df.dropna()
```

```
x = df.drop("is_safe", axis=1)
y = df["is_safe"]

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=12)

mlp = MLPClassifier(hidden_layer_sizes=(200), max_iter=1000)
mlp.fit(x_train, y_train)
mlp.score(x_train, y_train)
```

```
0.9735772357723578
```

Logistic Regression

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
df = pd.read_csv("../datasets/WaterQuality.csv")
df.head()
```

	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	...	lead	nitrates	nitrites	me
0	1.65	9.08	0.04	2.85	0.007	0.35	0.83	0.17	0.05	0.20	...	0.054	16.08	1.13	
1	2.32	21.16	0.01	3.31	0.002	5.28	0.68	0.66	0.90	0.65	...	0.100	2.01	1.93	
2	1.01	14.02	0.04	0.58	0.008	4.24	0.53	0.02	0.99	0.05	...	0.078	14.16	1.11	
3	1.36	11.33	0.04	2.96	0.001	7.23	0.03	1.66	1.08	0.71	...	0.016	1.41	1.29	
4	0.92	24.33	0.03	0.20	0.006	2.67	0.69	0.57	0.61	0.13	...	0.117	6.74	1.11	

5 rows × 21 columns

```
df.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
```

```
df.replace("#NUM!", float("nan"), inplace=True)
df = df.dropna()
```

```
x = df.drop("is_safe", axis=1)
y = df["is_safe"]

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=12)

lr = LogisticRegression(max_iter=1000)
lr.fit(x_train, y_train)
lr.score(x_test, y_test)
```

```
0.9025
```

Conclusion

The predictive models trained on the water quality dataset have demonstrated promising performance across various machine learning algorithms. Below are the accuracy scores achieved by each model:

ML Algorithm	**Accuracy Score**
Multi-Layer Perceptron (MLP)	0.9735772357723578
Decision Tree	0.956875
K-Nearest Neighbors (KNN)	0.90375
Logistic Regression	0.9025

The MLP model achieved the highest accuracy score of approximately 97.36%, indicating its effectiveness in accurately predicting water safety based on contaminant levels. Decision Tree also performed well with an accuracy score of around 95.69%. While KNN and Logistic Regression yielded slightly lower accuracy scores, they still demonstrated considerable performance in classifying water samples.

In conclusion, these models show promise in aiding the early detection of water safety issues, which is crucial for safeguarding public health. Further fine-tuning and optimization of these models could potentially enhance their predictive capabilities and contribute to more robust water quality assessment systems.