National College of Ireland

Enhancing Water Potability Prediction Using Sparse Attention-Based Deep Learning Models

MSc Research Project
MSc in Data Analytics - C

Pakeer Saikiran Reddy
Student ID: x23332310

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Pakeer Saikiran Reddy……

Student ID: ……x23332310……

Programme: MSc in Data Analytics…… **Year:** 2024-25……..

Module: MSc Research Practicum……

Lecturer: …Dr. Christian Horn……

Submission Due Date: ……15/09/2025……

Project Title: Enhancing Water Potability Prediction Using Sparse Attention-Based Deep Learning Models

Word Count: ……1150…… **Page Count:** ……08……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Pakeer Saikiran Reddy

Date: 15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Water Potability Prediction Using Sparse Attention-Based Deep Learning Models

Pakeer Saikiran Reddy
Student ID: x23332310

1. Introduction

This configuration manual presents the setup and methodology for building a machine learning pipeline to classify water samples based on their potability.

2. System Hardware Requirements

To effectively execute the water potability classification system, the following system specifications are recommended. These ensure seamless processing of the dataset, efficient training of both classical machine learning models and deep learning architectures, and compatibility with all required libraries.

Operating System

- **Preferred:** Windows 10 or Windows 11
  Fully compatible with Python-based scientific computing libraries and deep learning frameworks such as PyTorch, XGBoost, and TabNet.
- **Optional Alternative:** Ubuntu 20.04+
  Suitable for users experienced with Linux environments. Offers improved resource handling for CLI-based workflows.

Processor (CPU)

- **Minimum:** Intel Core i5 or AMD Ryzen 5
- **Recommended:** Quad-core processor or higher
  Improves performance for data preprocessing, feature transformation, and training ensemble models (e.g., Random Forest, XGBoost).

System Memory (RAM)

- **Minimum:** 16 GB
- **Recommended:** 32 GB
  Especially beneficial during parallel training, batch processing, and when handling large datasets or memory-heavy deep learning models like TabNet.

Graphics Processing Unit (GPU)

- **Recommended:** NVIDIA GPU with CUDA support
  Enables hardware acceleration for training deep learning models such as the custom

neural network and TabNet classifier.
Compatible GPU models include the NVIDIA GTX/RTX series or Tesla (for workstation setups).

Storage Requirements

- **Required Free Space:** 15–20 GB
  To store:
  - Input dataset (water_data_ka.csv)
  - Python environment and dependencies
  - Model artifacts and logs
  - Evaluation outputs and visualizations

3. Software Requirements:

This section describes the essential software components needed to execute the water potability classification pipeline effectively. The system is built using Python and leverages a range of scientific computing and machine learning libraries for data processing, model development, and evaluation.

Python Version

The project requires Python version 3.8 or higher. All major components of the pipeline, including machine learning and deep learning modules, are compatible with Python versions up to 3.11.

Required Libraries

The following Python libraries must be installed and available in the execution environment:

- pandas: used for data loading, manipulation, and summarization
- numpy: supports numerical computations and array-based operations
- matplotlib and seaborn: provide visualization tools for exploratory data analysis and performance reporting
- scikit-learn offers tools for preprocessing, model training, evaluation metrics, and validation strategies
- xgboost: used for implementing gradient boosting models
- torch (PyTorch): required for building and training the custom deep learning model
- pytorch-tabnet: necessary for running the TabNet architecture on tabular datasets

Environment Management

It is recommended to use a virtual environment for package management and to maintain project-specific dependencies. This ensures that the software stack remains isolated and reproducible across different systems or platforms.

Development Tools

The pipeline can be executed in any of the following development environments:

- Jupyter Notebook or JupyterLab
- Visual Studio Code with Python support
- Google Colab for cloud-based execution with optional GPU usage
- Anaconda Navigator for users preferring a graphical interface

Optional GPU Support

To utilize GPU acceleration during training, ensure that the correct versions of CUDA and cuDNN are installed and compatible with the PyTorch library. This is particularly relevant for training neural networks and the TabNet model efficiently.

Output Handling

All evaluation results, including accuracy metrics, confusion matrices, and class distribution visualizations, are generated within the environment. No external tools or logging frameworks are required.

4. Dataset Details

This project uses a structured dataset containing water quality measurements collected from various sources. The dataset is designed to support the binary classification of water samples into potable and non-potable categories, based on a range of physicochemical attributes.

Dataset Name
The dataset file is named **water_data_ka.csv**.

Dataset Size

- Number of records: 100,000 rows
- Number of features: 23 columns (including the target variable)

Target Variable

- **Target**: A binary variable indicating potability
    - Value 0: Non-potable (not safe for consumption)
    - Value 1: Potable (safe for consumption)

Feature Types

- The dataset consists entirely of numerical features. These represent chemical concentrations, physical measurements, and water quality indicators.
- There are no categorical features present in the raw data.
- A subset of columns contains missing values, which are handled during preprocessing.

Key Features

- **pH**: Acidity or alkalinity of water
- **Turbidity**: Clarity of the water
- **Iron, Lead, Zinc, Copper**: Concentrations of metal elements
- **Fluoride, Sulfate, Nitrate**: Key chemical compounds
- **Water Temperature**: Measured in degrees Celsius
- **Manganese**: Trace metal element
- Additional variables represent other relevant water quality indicators.

Data Integrity

- The dataset includes some missing values in numeric columns. These are treated using median imputation during the preprocessing phase.
- Several features exhibit skewness or near-zero medians, which are addressed through transformation techniques such as log scaling.

Class Distribution

- Potability classes are relatively balanced:
    - Approximately 53 percent of samples are labeled as non-potable (class 0)
    - Approximately 47 percent are labeled as potable (class 1)

5. Model Configuration

This section outlines the configuration of all classification models used in the water potability prediction pipeline. The system incorporates a variety of algorithms ranging from decision trees to deep learning architectures, each configured for binary classification of water samples based on physicochemical properties.

Modelling Approach

The classification task is supervised and binary in nature. All models are trained using pre-processed data with the target variable indicating potability status. A standard 80-20 train-test split is applied, stratified to preserve class balance.

Included Models

The following models are configured and evaluated as part of the pipeline:

1. **Decision Tree Classifier**

```
# Decision Tree Model
dt_model = DecisionTreeClassifier(max_depth=3, random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)

print("=== Decision Tree Classifier===")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("\nClassification Report:\n", classification_report(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))

# Plot confusion matrix
plt.figure(figsize=(5,4))
sns.heatmap(confusion_matrix(y_test, y_pred_dt), annot=True, fmt='d', cmap='Blues')
plt.title('Decision Tree Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

2. Random Forest Classifier

```
# Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, max_depth=3, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("\n=== Random Forest Classifier ===")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))

# Plot confusion matrix
plt.figure(figsize=(5,4))
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Greens')
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

3. Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Initialize and train the model
gb_model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
gb_model.fit(X_train, y_train)

# Predict
y_pred_gb = gb_model.predict(X_test)

# Evaluation
print("=== Gradient Boosting Classifier ===")
print("Accuracy:", accuracy_score(y_test, y_pred_gb))
print("\nClassification Report:\n", classification_report(y_test, y_pred_gb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_gb))
```

4. **AdaBoost Classifier**

```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Initialize and train the model
ada_model = AdaBoostClassifier(n_estimators=100, learning_rate=1.0, random_state=42)
ada_model.fit(X_train, y_train)

# Predict
y_pred_ada = ada_model.predict(X_test)

# Evaluation
print("=== AdaBoost Classifier ===")
print("Accuracy:", accuracy_score(y_test, y_pred_ada))
print("\nClassification Report:\n", classification_report(y_test, y_pred_ada))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_ada))
```

5. **XGBoost Classifier**

```python
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Initialize and train the model
xgb_model = XGBClassifier(n_estimators=100, max_depth=3, learning_rate=0.1, use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

# Predict
y_pred_xgb = xgb_model.predict(X_test)

# Evaluation
print("=== XGBoost Classifier ===")
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("\nClassification Report:\n", classification_report(y_test, y_pred_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))
```

6. **Custom Neural Network (PyTorch)**

   - Fully connected feedforward architecture
   - Layers: Input → Dense (64) → ReLU → Dropout → Dense (32) → ReLU → Dropout → Output (2 units)
   - Cross-entropy loss with Adam optimizer
   - Trained for 600 epochs with evaluation on test set

7. **TabNet Classifier (PyTorch TabNet)**

   - Deep learning model designed for tabular data
   - Incorporates feature selection using attentive transformers
   - Trained with early stopping and batch-wise optimization
   - Achieved highest model accuracy in the pipeline

Input and Output

- **Input Features**: All numeric columns except the target
- **Output**: Binary class prediction (0 or 1), indicating non-potable or potable water, respectively

```python
import numpy as np
from pytorch_tabnet.tab_model import TabNetClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import torch

# Convert to NumPy arrays
X_tabnet = X.values
y_tabnet = y.values

# Encode target if not already encoded
if y_tabnet.dtype == 'object':
    y_tabnet = LabelEncoder().fit_transform(y_tabnet)

# TabNet requires float32 input
X_train_tabnet = X_train.values.astype(np.float32)
X_test_tabnet = X_test.values.astype(np.float32)
y_train_tabnet = y_train.values
y_test_tabnet = y_test.values
```

6. Evaluation and Results

The evaluation of model performance is carried out using standard classification metrics including accuracy, precision, recall, and F1-score. Confusion matrices are generated for each model to analyze true and false classifications across potable and non-potable classes.

```
=== TabNet Classifier ===
Accuracy: 0.8726

Classification Report:
              precision    recall  f1-score   support

         0.0       0.91      0.84      0.87     10600
         1.0       0.83      0.91      0.87      9400

    accuracy                           0.87     20000
   macro avg       0.87      0.87      0.87     20000
weighted avg       0.88      0.87      0.87     20000

Confusion Matrix:
 [[8887 1713]
 [ 835 8565]]
```

Conclusion

This project successfully demonstrates a complete machine learning pipeline for classifying the potability of water samples using a rich set of physicochemical features.

References

PyTorch (Official Documentation): https://pytorch.org/docs/stable/index.html

Scikit-learn: https://scikit-learn.org/stable/documentation.html

Matplotlib: https://matplotlib.org/stable/users/index.html

Seaborn: https://seaborn.pydata.org/

OpenML : https://www.openml.org/

Kaggle Learn: https://www.kaggle.com/learn

UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

Pytorch TabNet: https://github.com/dreamquark-ai/tabnet

Google Colab: https://colab.research.google.com/

Anaconda Documentation: https://docs.anaconda.com/

MLflow: https://mlflow.org/docs/latest/index.html

SHAP: https://shap.readthedocs.io/en/latest/