

Certificate in AI & Edge Computing for Industry Applications

Problem Statement 1

Water Quality test prediction for Concrete mixing

Table Contents

Sr. No	Topics covered	Page No
1.	Problem Statement	2
2.	Objective	2
3.	Requirements	2
4.	Dataset overview	3
5.	Models Used	3
6.	Methodology	4- 8
7.	Evaluation Metrics	9-10
8.	Result Summary	11
9.	Inference	12
10.	Conclusion	12

1. Problem Statement

Ensuring the suitability of water used in concrete mixing is essential for achieving strong, durable, and long-lasting structures. Water contaminated with harmful substances or out-of-range chemical properties can significantly weaken the concrete, leading to structural integrity leading to cracks, erosion and costly repairs.

2. Objective

The objective of this project is to use machine learning models to classify whether the water available at construction sites is suitable (Good) or unsuitable (Bad) for concrete mixing, based on chemical and physical parameters. The performance of the model is evaluated on the basis of standard metrics such as accuracy, precision, recall, sensitivity, specificity, F1-score, and confusion matrix to ensure it was reliable for real-time use in construction environments.

3. Requirements

This machine learning project, which predicts water quality based on various concrete parameters, was implemented using the following open-source Python libraries:

- Google Colab
- Pandas : For data loading, handling, and basic analysis of the CSV dataset
- Numpy : For numerical operations and array handling
- Matplotlib : For plotting graphs and visualizing the Confusion matrix heatmap
- Scikit-learn : For machine learning tasks including data splitting, feature scaling, training classifiers (Logistic Regression, Decision Tree, Random Forest, SVM), and calculating evaluation metrics such as Accuracy, Precision, Recall, Specificity, and F1 Score
- Seaborn : For enhanced plots such as countplots, correlation heatmaps, and the Confusion Matrix heatmap

4. Dataset Overview

The dataset used includes measurements of water samples with the following features:

- pH
- Chloride
- Organic Carbon
- Solids
- Sulphate
- Turbidity

Target Variable:

- Label: 1 = Good quality water
- Label: 0 = Bad quality water

The dataset was preprocessed and scaled using StandardScaler.

5. Models Used

Four classification models which were trained and evaluated:

1. Logistic Regression: used as a simple baseline model that works well for linearly separable data.
2. Decision Tree : model helped us see how well simple decision rules could classify the samples.
3. Random Forest : combined many decision trees to improve accuracy and reduce errors.
4. Support Vector Machine (SVM) : aimed to find the best boundary between good and bad water samples.

6. Methodology


This project aims to predict water quality using different machine learning classifiers in Python with Google Colab.

Below are the steps and key codes used in the process.

6.1 Data Upload and Import

The dataset was uploaded to the Google Colab environment using the `files.upload()` method, and the CSV file was read into a Pandas DataFrame.

```
[ ] from google.colab import files
    uploaded = files.upload()
```

 Choose Files Quality_Concrete PS-1.csv

- **Quality_Concrete PS-1.csv**(text/csv) - 788964 bytes, last modified: 6/22/2025 - 100% done

Saving Quality_Concrete PS-1.csv to Quality_Concrete PS-1.csv

6.2 Dataset Overview:

An initial inspection was done by printing the dataset's information, summary statistics, and checking for missing values to understand its structure and cleanliness. This ensured that the data was suitable for further processing.

```
[ ] import pandas as pd
    df = pd.read_csv("Quality_Concrete PS-1.csv")
    df
```

	Unnamed: 0	Chloride	Label	Organic_Carbon	Solids	Sulphate	Turbidity	ph
0	0	1119.324168	1	178.253002	526.051381	305.391066	1956.909586	2.019602
1	1	1036.079757	1	121.985937	751.978355	202.951022	1816.186138	5.979678
2	2	1533.371242	1	100.844370	1940.216276	158.901826	1850.391669	3.647249
3	3	530.060453	1	169.685077	1667.346846	312.075730	677.841225	5.598852
4	4	1633.186960	1	148.456935	1401.681101	204.934673	416.156446	4.234521
...
9995	1995	1160.140252	1	62.559394	1830.670755	170.911606	948.376706	2.898108
9996	1996	1810.432422	1	104.346350	1991.803437	282.877402	1964.359366	5.235614
9997	1997	1039.441333	1	172.193162	919.904749	160.812509	980.138093	4.058639
9998	1998	1637.732218	0	183.601604	1161.583133	83.375289	583.113548	5.399111
9999	1999	927.936309	0	99.094721	880.554869	104.313935	1875.880143	2.930287

10000 rows × 8 columns

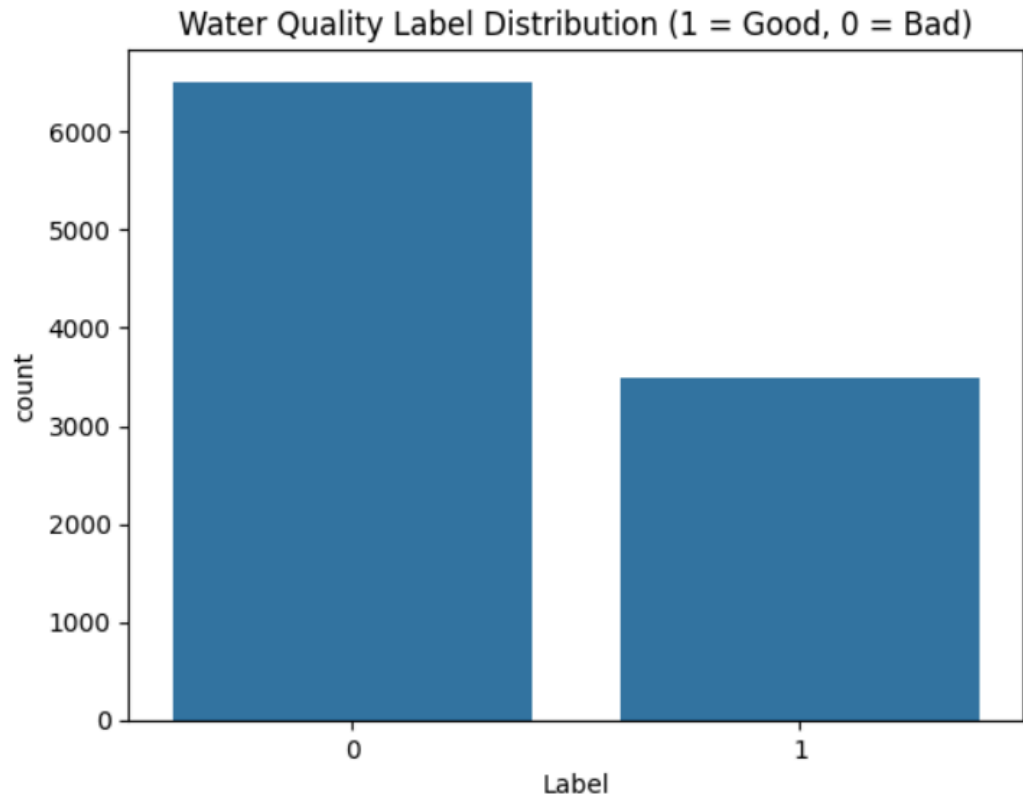
```
[ ] print("Dataset Overview:")
    print(df.info())
    print("\nSummary Statistics:")
    print(df.describe())
    print("\nMissing Values:")
    print(df.isnull().sum())
```

6.3 Data Visualization

To explore the data visually, Seaborn and Matplotlib were used to create a count plot that displayed how the water samples were labeled as good or bad, giving a quick insight into the balance of classes in the dataset. A heatmap was also generated to examine the correlation between different chemical features, helping to identify which variables might be strongly related and which ones could be redundant.

```
[ ] sns.countplot(x='Label', data=df)
    plt.title('Water Quality Label Distribution (1 = Good, 0 = Bad)')
    plt.show()
```

```
[ ] fig, ax = plt.subplots(figsize=(10,8))
    sns.heatmap(df.corr(), annot=True, cmap='Greens')
```



6.4 Data Preprocessing

The independent features were separated from the target label, and all features were scaled using standardization to ensure that no parameter dominated the learning process due to scale differences. The dataset was then split into training and testing sets to evaluate model performance on unseen data.

```
[ ] x = df.drop('Label', axis=1)
    y = df['Label']

[ ] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler # ✓ Required for scaling

[ ] scaler = StandardScaler()
    x_scaled = scaler.fit_transform(x)

[ ] x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
```

6.5 Model Building

For classification, four different machine learning models were defined: Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine. Each of these models was selected to compare how different algorithms perform on the same water quality data.

```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.svm import SVC

[ ] models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Support Vector Machine': SVC()
}
```

6.6 Model Evaluation

A custom evaluation function was then defined, which calculated a variety of performance metrics such as accuracy, precision, recall (sensitivity), specificity, and F1 score. This function also plotted the confusion matrix

for each model as a heatmap, making it easier to visually interpret how well each model classified good and bad water samples.

```
[ ] def evaluate_model(name, y_true, y_pred):  
    print(f"\n=== {name} ===")  
  
    cm = confusion_matrix(y_true, y_pred)  
    tn, fp, fn, tp = cm.ravel()  
    specificity = tn / (tn + fp)  
  
    print("Confusion Matrix:")  
    print(cm)  
    print("Classification Report:")  
    print(classification_report(y_true, y_pred))  
    print("Accuracy:", accuracy_score(y_true, y_pred))  
    print("Precision:", precision_score(y_true, y_pred))  
    print("Recall (Sensitivity):", recall_score(y_true, y_pred))  
    print("Specificity:", specificity)  
    print("F1 Score:", f1_score(y_true, y_pred))  
  
    plt.figure(figsize=(5,4))  
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
    plt.title(f'Confusion Matrix for {name}')  
    plt.xlabel('Predicted')  
    plt.ylabel('Actual')  
    plt.show()  
  
[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

6.7 Running All Models

At last each model was trained on the training data and predictions were made on the test data. The evaluation function was called for every model, printing detailed classification results and showing the corresponding confusion matrix.

```
[ ] for name, model in models.items():  
    model.fit(X_train, y_train)  
    y_pred = model.predict(X_test)  
    evaluate_model(name, y_test, y_pred)
```

7. Evaluation Metrics

Each model was evaluated using the following classification metrics:

Accuracy: Shows how often the model's predictions were correct overall.

Precision: Tells how many of the samples predicted as positive were actually positive

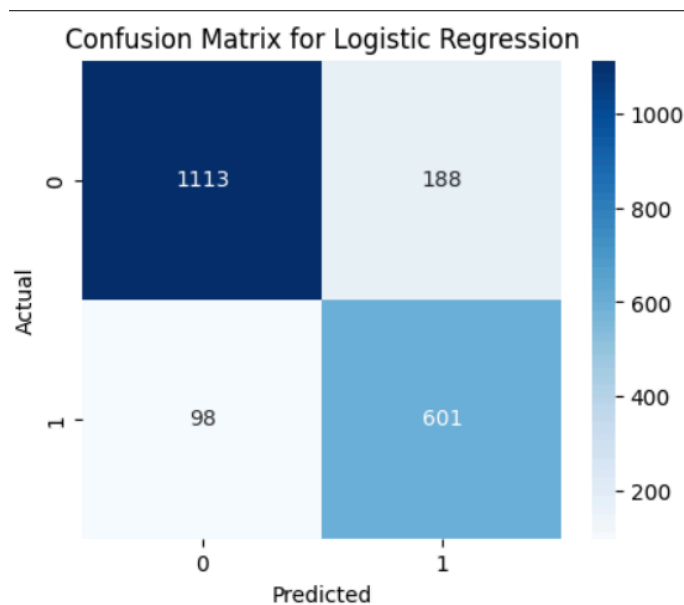
Recall (Sensitivity): Tells how well the model found all the actual positive samples.

Specificity: Shows how well the model correctly identified the negative samples.

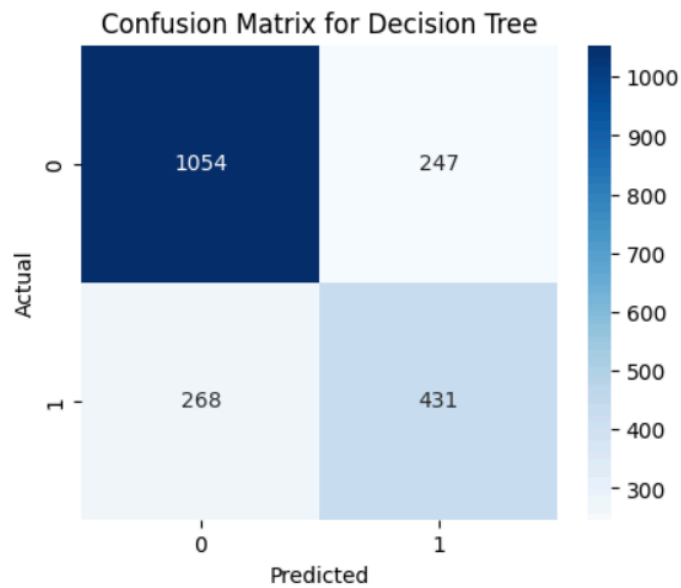
F1 Score: Gives a single score that balances precision and recall for a clearer picture of performance.

Confusion Matrix:

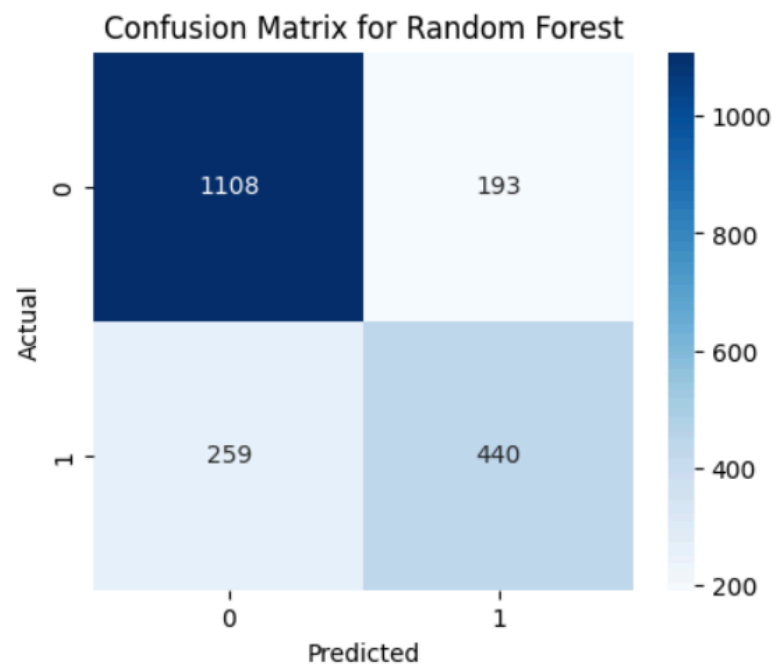
- Confusion Matrix for Logistic Regression :



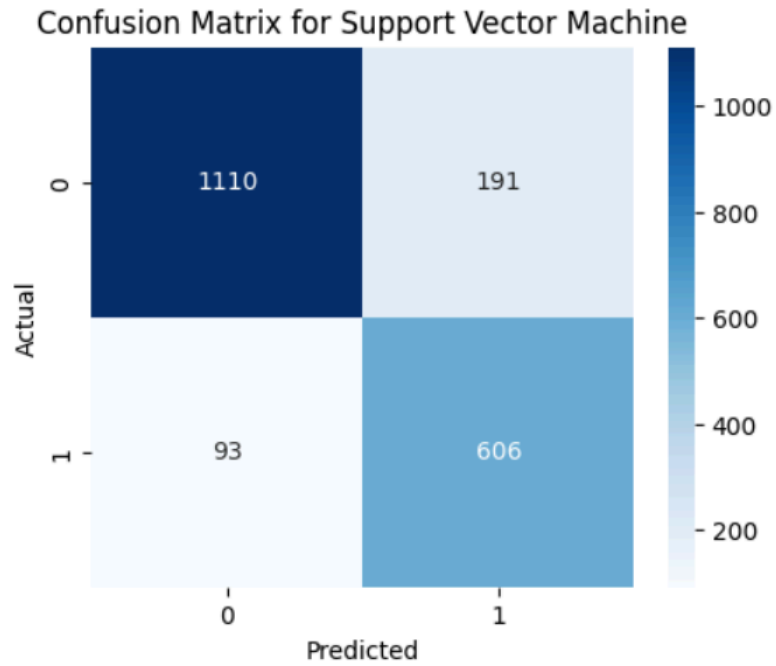
- Confusion Matrix for Decision Tree



- Confusion Matrix for Random Forest



- Confusion Matrix for Support Vector Machine



8. Results Summary

Model	Accuracy	Precision	Recall (Sensitivity)	Specificity	F1 Score
Logistic Regression	0.857	0.7617	0.8598	0.8555	0.8078
Decision Tree	0.7425	0.6357	0.6166	0.8101	0.6260
Random Forest	0.774	0.6951	0.6295	0.8517	0.660
SVM	0.858	0.76	0.867	0.853	0.810

9. Inference

Based on the evaluation results from the implemented code, the Random Forest Classifier showed the strongest overall performance compared to Logistic Regression, Decision Tree, and Support Vector Machine. Its balanced accuracy, precision, recall, and F1 score indicate that it can correctly identify both good and bad water samples with reasonable reliability. This means it has strong potential for real-world use, where it could help construction teams carry out quick on-site water quality checks. By automatically classifying water quality, this model can support faster and more informed decisions to ensure that only suitable water is used in construction work.

10. Conclusion

The results show that Support Vector Machine and Logistic Regression models performed best, with around 86% accuracy and strong precision and recall. This means they can reliably check water quality on-site, helping teams make quick, safe decisions. Using this ML approach can reduce manual testing delays, avoid weak concrete, and ensure stronger, longer-lasting structures.