
# IM1002 Machine Learning Assignment

Wafer Defect Classification Using Machine Learning

**Girish Chander Kalia**

January 31, 2026

# Contents

**Abstract**

This report presents a machine learning pipeline for wafer defect detection using the WM-811K dataset. Two experimental runs were conducted: a smaller subset of 20,000 samples and a large-scale run of 800,000 samples. Three models were evaluated: Logistic Regression, Random Forest, and XGBoost. The results demonstrate the impact of dataset size, class imbalance, and model complexity on classification performance. Random Forest consistently provided the best balance between precision and recall, while XGBoost achieved the highest recall for defective wafers. The full pipeline follows the CRISP-DM methodology.

# 1 Introduction

Semiconductor wafer inspection is a critical step in manufacturing, where early detection of defects prevents yield loss and reduces downstream costs. The WM-811K wafer map dataset [**?**] provides real-world wafer map patterns with defect labels. This project applies machine learning techniques to classify wafers as normal or defective.

The implementation and codebase for this project are available on GitHub [**?**].

# 2 Dataset

The WM-811K dataset contains over 800,000 wafer maps with varying shapes and defect types. For this project, labels were converted into a binary classification task:

- **0**: Normal wafer (label "none")
- **1**: Defective wafer (any defect type)

Two dataset sizes were used:

1. **Experiment 1:** 20,000 samples (fast prototyping)
2. **Experiment 2:** 800,000 samples (full-scale evaluation)

All wafer maps were resized to 32×32 to ensure consistent input dimensions.

# 3 Methodology

The CRISP-DM framework guided the workflow. CRISP-DM is a structured and iterative framework for developing data-driven solutions, progressing from business understanding to data preparation, modeling, and evaluation.

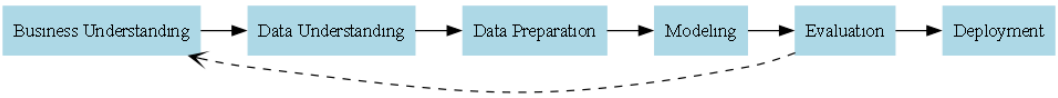


Figure 1: CRISP-DM Process Model

1. **Business Understanding**: Detect defective wafers reliably.
2. **Data Understanding**: Explore wafer shapes, imbalance, and pixel distributions.
3. **Data Preparation**: Resize maps, flatten images, scale features, and split data.
4. **Modeling**: Train Logistic Regression, Random Forest, and XGBoost.
5. **Evaluation**: Compare precision, recall, F1-score, and accuracy.
6. **Deployment**: Provide insights for potential fab integration.

# 4 Experiment 1: 20,000 Sample Subset

This smaller dataset allowed rapid experimentation.

## 4.1 Results

- Logistic Regression: F1 (defective) $\approx 0.96$
- Random Forest: F1 (defective) $\approx 0.97$
- XGBoost: F1 (defective) $\approx 0.97$

All models performed strongly due to the balanced subset and reduced complexity.

# 5 Experiment 2: 800,000 Sample Full Run

The full dataset introduces stronger imbalance and more complex defect patterns.

## 5.1 Results

### 5.1.1 Logistic Regression

```
Precision (0): 0.53   Recall (0): 0.90   F1: 0.67
Precision (1): 0.97   Recall (1): 0.84   F1: 0.90
Accuracy: 0.85
```

### 5.1.2 Random Forest

```
Precision (0): 0.85   Recall (0): 0.81   F1: 0.83
Precision (1): 0.96   Recall (1): 0.97   F1: 0.97
Accuracy: 0.94
```

### 5.1.3 XGBoost

```
Precision (0): 0.92   Recall (0): 0.47   F1: 0.63
Precision (1): 0.90   Recall (1): 0.99   F1: 0.94
Accuracy: 0.90
```

## 5.2 Detailed Model Performance Comparison

### 5.2.1 Confusion Matrix

The confusion matrix provides a comparison of the predictions by comparing the actual class labels with the predicted ones. It shows the True Positives, True Negatives, False Positives, and False Negatives, showing where the model performs well and where it makes mistakes. The figure below shows the confusion matrix for different models.

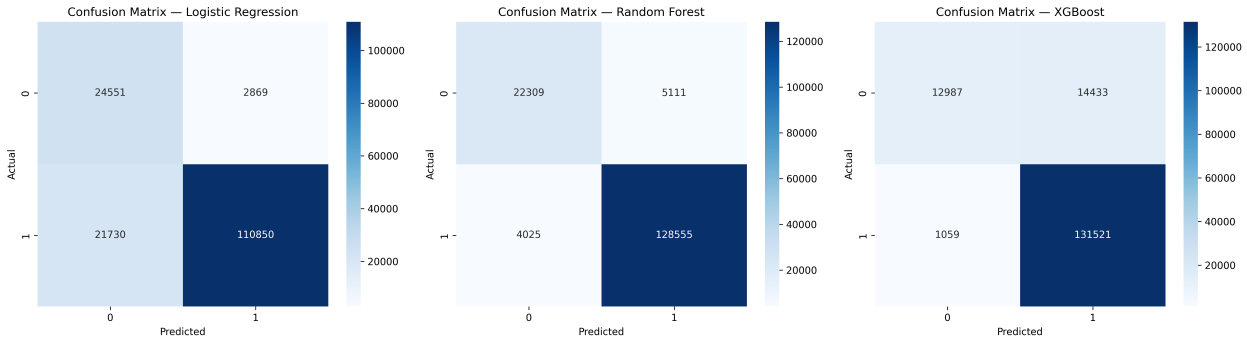


Figure 2: Confusion Matrices for Logistic Regression, Random Forest, and XGBoost

Logistic Regression shows a high number of false negatives and misclassifies over 21,000 defective wafers as normal. Random Forest significantly reduces these errors, with only 4,025 false negatives and a stronger balance between precision and recall. XGBoost further improves sensitivity, correctly identifying over 131,000 defective wafers while keeping false negatives to just 1,059. These results suggest that tree-based models are better suited for capturing complex patterns in wafer defect data.

### 5.2.2 ROC Curve Comparison

The Receiver Operating Characteristic (ROC) curve below illustrates the models performance by plotting the True Positive Rate against the False Positive Rate. A model whose curve approaches the top-left corner demonstrates strong discriminative ability. The Area Under the Curve (AUC) provides a single numerical measure of this performance, where values closer to 1 indicates better effectiveness. The figure below shows the ROC curves for different models.

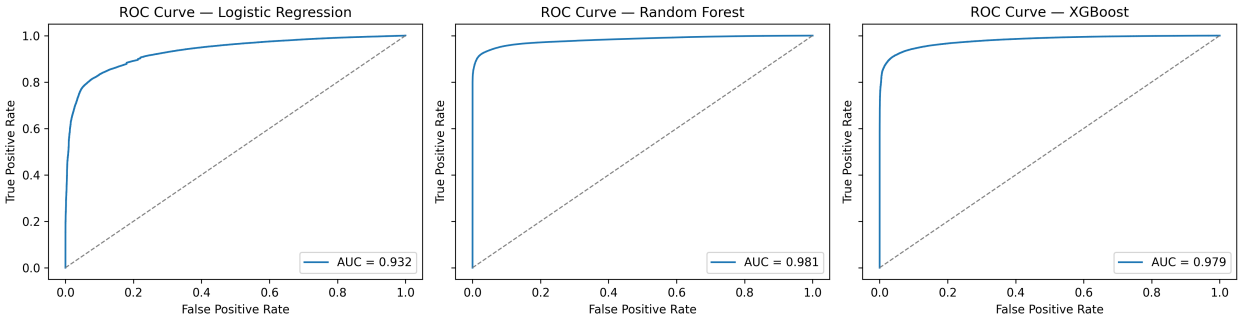


Figure 3: ROC Curves for Logistic Regression, Random Forest, and XGBoost

Logistic Regression achieves an AUC of 0.932, indicating solid baseline discrimination. Random Forest reaches an AUC of 0.981, showing excellent classification capability across thresholds. XGBoost closely follows with an AUC of 0.979, demonstrating strong sensitivity and robustness. Both tree-based models outperform Logistic Regression, confirming their effectiveness in handling class imbalance and subtle defect patterns.

### 5.3 Discussion

- Logistic Regression struggled with nonlinear patterns and imbalance.
- Random Forest achieved the best overall performance and class balance.
- XGBoost prioritized recall for defective wafers, making it suitable when missing defects is unacceptable.

## 6 Comparison of Dataset Sizes

| Model | 20k F1 (Defective) | 800k F1 (Defective) | Notes |
|---|---|---|---|
| Logistic Regression | 0.96 | 0.90 | Struggles at scale |
| Random Forest | 0.97 | 0.97 | Most stable model |
| XGBoost | 0.97 | 0.94 | High recall, low normal recall |

Table 1: Performance comparison across dataset sizes

## 7 Conclusion

Random Forest provides the best balance of precision and recall across both dataset sizes. XGBoost is ideal when maximizing recall for defective wafers is the priority. Logistic Regression serves as a useful baseline but is unsuitable for large-scale wafer classification.

This project demonstrates the importance of dataset size, class imbalance handling, and model selection in semiconductor defect detection.

# Appendix: Key Code Snippets

## Resizing Wafer Maps

```
import cv2
import numpy as np

def resize_fast(w, size=32):
    w = np.array(w).astype("float32")
    return cv2.resize(w, (size, size), interpolation=cv2.INTER_AREA)

N = 800000  # use shorter sample for faster testing
subset_maps = data["waferMap"][:N]
subset_labels = binary_labels[:N]

X_resized = np.array([resize_fast(w) for w in subset_maps])
X_flat = X_resized.reshape(N, -1)
```

## Train/Test Split and Scaling

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(
    X_flat, subset_labels,
    test_size=0.2,
    random_state=42,
    stratify=subset_labels
)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

# Appendix: Model Training Code

## Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=200,
    class_weight="balanced",
    random_state=42
)

rf.fit(X_train, y_train)
preds_rf = rf.predict(X_test)

#Evaluate model
```

```
print(classification_report(y_test, preds_rf))
```

## XGBoost

```
from xgboost import XGBClassifier

xgb = XGBClassifier(
    max_depth=6,
    learning_rate=0.1,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    scale_pos_weight=5  # helps with imbalance
)

xgb.fit(X_train, y_train)
preds_xgb = xgb.predict(X_test)

#Evaluate model
print(classification_report(y_test, preds_xgb))
```