ECE 157A/272A Homework 2: Wafer Map Failure Pattern

Due Date: Wednesday, October 26th, 2022, 4:59PM

Introduction

In homework 2, we will introduce you to the WM-811K wafer dataset[1], which is real-world data produced at a fab. This is a well-known dataset used for wafer failure analysis research in recent years. Unlike the first homework, where you are given processed features, this homework require you to perform feature engineering to define and extract features from wafer failure maps. From your map features, you will build a decision tree model to evaluate how well those features function for classification.

Motivating story:
You are a machine learning expert working in a semiconductor company. You arrive at work one day and find out from your boss that the product test team has sent over test results for a new product line. The product test engineers have labeled commonly seen defects.

Your company wants you to build machine learning models to detect these commonly seen defects, to speed up the testing process. The features should be intuitive enough to help the experts better understand the failure patterns and to relate them back to the fab for adjustments. The team want you to start with traditional models as a baseline, then to move to neural network in the next iteration. (For us, that will be the next homework.)

Background: Wafer Map Failure Pattern Analysis

In the semiconductor manufacturing process, batches of functional circuits are fabricated on a slice of silicon material which we call a *wafer*. After going through complicated test stages, the wafer is cut (diced) into pieces, each containing one copy of the circuit. Each of these pieces is called a die. At companies like Intel and Qualcomm, one of the important factors for their revenue is ensuring a high *yield* of their manufactured wafers. *Yield* is the fraction of dice on the yielding wafers that are not discarded during the manufacturing process due to some defect. The *yield per wafer* is quantified as the number of working (good/passing) dice divided by the total number of dice fabricated on that wafer.

Wafer maps provide visual patterns of the bad/failing dice on the wafer. By analyzing the failure pattern on a wafer, experienced engineers can often identify issues in manufacturing and/or test process that caused the failure, then take actions accordingly. Thus, these wafer maps are important for increasing the quality of production and ultimately to increase revenue.

Dataset

We have taken a subset of the original data as training data of this homework (wafermap_train.npy). It contains 1693 real-world wafers and their attributes as shown in Table 1. There are 5 types of known labels: `Center`, `Edge-Loc`, `Scratch`, `Donut`, and `Near-full`.

Column	Count	Type	Description
dieSize	1693	float	the number of dice on a wafer
failureType	1693	string	the type of failure pattern from 5 categories: "Center", "Edge-Loc", "Scratch", "Donut", "Near-full"
lotName	1693	string	the manufacturing lot name of a wafer
trainTestLabel	1693	string	denotes whether the sample is for training or testing in the original dataset
waferIndex	1693	integer	denotes the wafer index within a lot
waferMap	1693	numpy array	contains wafer maps *of varying size*. 0s denote no dice, 1s denote passing dice, 2s denote failing dice

Table 1: Real-world wafers [1]

Tasks

For this homework, we will perform feature engineering to extract features from the wafer maps. We will use these features to build decision tree models to classify the wafer map failure patterns. Lastly, we will extract the decision rules from the trained decision tree model. To get started, we provided you with a skeleton code in the homework zip file.

Dataset Inspection

Before we dive into designing the features, it is important to inspect the dataset and visualize the failure patterns. Implement the two helper functions below to visualize the data:

- Implement a function that takes a pandas data frame of the form described above and plots an example wafer for each failure type. **Show the plot in the report.**
- Implement a function that takes a pandas data frame and outputs all wafer maps to corresponding failure type directories. This allows you to simply click through to see all the wafer maps of each failure type.

Data Preparation (Part 1)

After inspecting the dataset, you should see the following problems:

- The dieSize column varies, meaning the wafer maps have different shapes. Implement a function that takes a wafer map and resizes it to (64, 64). **Show a before-and-after resize plot for a wafer map from each failure type**.
- The failureType is a string type, but the model training requires numerical number. Implement a function that takes the failure type string and converts it to numerical value. **Show the string to numerical mapping.**

Feature Engineering

Now that we have a better understanding of the failure patterns, we will start creating features for the wafer maps based on section **III.B**. of this paper [2]. Most of the features we implement will be based on the salient region of the wafer map. Salient region (for this homework) is defined as the 8-neighborhood connected component with the largest area. We will implement the ten features listed below:

- Area Ratio: Ratio of the area of the salient region to the area of the wafer map
- Perimeter Ratio: Ratio of the perimeter of the salient region to the radius of the wafer map
- Max Distance from Center: Maximal distance between the salient region and the center of the wafer map
- Min Distance from Center: Minimal distance between the salient region and the center of the wafer map
- Major Axis Ratio: Ratio of the length of the major axis of the estimated ellipse surrounding the salient region to the radius of the wafer map
- Minor Axis Ratio: The ratio of the length of the minor axis of the estimated ellipse surrounding the salient region to the radius of the wafer map
- Solidity: The proportion of failed dice in the estimated convex hull in the salient region
- Eccentricity: The shape of the estimated eclipse surrounding the salient region, where the value is 0 for a circle, or 1 for a line
- Yield Loss: Ratio of the failed dice on the wafer map to the total number of dice on the wafer map
- Edge Yield Loss: Ratio of the failed dice on the outermost two rings of the wafer map to the total number of dice on the outermost two rings of the wafer map

For each failure class, show a wafer map and its salient region along with all the above feature values.

Note: many of these functions can be implemented easily with the Scikit-Image library

Data Preparation (Part 2)

Select the features defined above as your dataset and prepare the dataset for training and validation as we did in the previous homework. Perform your own train/test split to get your validation set, varying the `random_state` and `test_size` as you see fit. Do not normalize your data, as DecisionTree does not benefit from data normalization, and we would like the DecisionTree's rules to tell us the exact splitting values of our important features.

Model Training and Validation

We will train a decision tree model on the training set, evaluate its performance on the validation set, and analyze which feature is important based on the decision tree plot. For simplicity, you can start with a decision tree depth of 3 and increase it as needed. For this part you will need to show the following:

- **Per failure type accuracy for the training samples and validation samples**.
- **Confusion matrix for training samples and validation samples. (confusion matrix should be labeled using the original failureType string)**
- **For each failure type, use the decision tree plot to find a rule that leads you to the majority of that failure type. Explain whether or not this rule fits the intuitive description of the failure type.**

Model Testing

Load the test dataset, preprocess it, predict the failure type, output prediction to a `scores.csv` with a single column, `failureType`, and submit it to GradeScope. Try to achieve at least 80% on the test set.

What to turn in

- scores.csv: A CSV file with one column, failureType, containing your model's predictions for the wafers in wafermap_testing.npy.
- For the report, make *slides* that give a brief overview of the task objective, an overview of what you have done, and thoughtfully answer any question and show the results highlighted in bold in the Tasks section. Export and submit the slides as a PDF document.
- Submit your code either in notebook format (.ipynb) or script format (.py).

References

[1] Jyh-Shing R. Jang Ming-Ju Wu and Jui-Long Chen. *MIR-WM811K*. 2015. URL: `http://mirlab.org/dataset/public/`.

[2] Ming-Ju Wu, Jyh-Shing R. Jang, and Jui-Long Chen. "Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets". In: *IEEE Transactions on Semiconductor Manufacturing* 28.1 (2015), pp. 1–12. DOI: `10.1109/TSM.2014.2364237`.