

Analysis of Manufacturing Process of a Semiconductor

Technical Exercise - Phase 1

AGILE

Livia Parente

February 1, 2019

Agenda

- ❖ **Problem Context**
- ❖ **Objectives**
- ❖ **EDA**
- ❖ **Preprocessing Steps**
- ❖ **Model choice**
- ❖ **First Results**
- ❖ **Undersampling Strategy**
- ❖ **Hyperparameters Optimization**
- ❖ **Most important attributes**
- ❖ **Features selection**
- ❖ **Conclusion**

Problem Context

- Data collected from sensors of a manufacturing semiconductor process
 - Row count: 1567
 - Column count: 591
- Target variable values: Pass (1) or Fail (-1)
 - Binary classification problem

Objectives

1. Build a Machine Learning model to predict the target variable.
2. Sort the attributes in terms of importance that produce a positive test.
3. Rebuild the model using the key attributes identified in the previous step and compare the performance with the initial model.

EDA

Exploratory Data Analysis

EDA – Simple eye analysis

```
>>> data.head()
```

| | Time | 0 | 1 | 2 | ... | 587 | 588 | 589 | Pass/Fail |
|---|---------------------|---------|---------|-----------|-----|--------|--------|----------|-----------|
| 0 | 2008-07-19 11:55:00 | 3030.93 | 2564.00 | 2187.7333 | ... | NaN | NaN | NaN | -1 |
| 1 | 2008-07-19 12:32:00 | 3095.78 | 2465.14 | 2230.4222 | ... | 0.0201 | 0.0060 | 208.2045 | -1 |
| 2 | 2008-07-19 13:17:00 | 2932.61 | 2559.94 | 2186.4111 | ... | 0.0484 | 0.0148 | 82.8602 | 1 |
| 3 | 2008-07-19 14:43:00 | 2988.72 | 2479.90 | 2199.0333 | ... | 0.0149 | 0.0044 | 73.8432 | -1 |
| 4 | 2008-07-19 15:22:00 | 3032.24 | 2502.87 | 2233.3667 | ... | 0.0149 | 0.0044 | 73.8432 | -1 |

```
>>> secom.shape  
(1567, 592)
```

First conclusions

- First row has the name of features
- First column has time and data
- Last column has the labels
- Some NaN values
- Large Dataset 1567x592

EDA – More immediate investigation

```
>>> secom.describe()
```

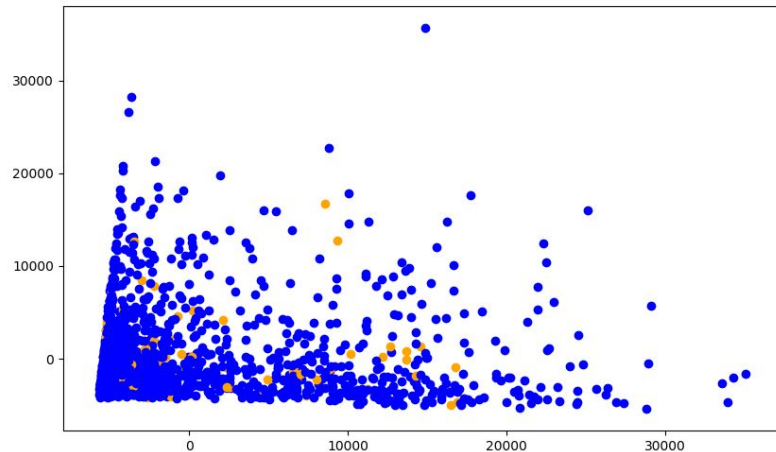
| | 0 | 1 | 2 | ... | 588 | 589 | Pass/Fail |
|-------|-------------|-------------|-------------|-----|-------------|-------------|-------------|
| count | 1561.000000 | 1560.000000 | 1553.000000 | ... | 1566.000000 | 1566.000000 | 1567.000000 |
| mean | 3014.452896 | 2495.850231 | 2200.547318 | ... | 0.005283 | 99.670066 | -0.867262 |
| std | 73.621787 | 80.407705 | 29.513152 | ... | 0.002867 | 93.891919 | 0.498010 |
| min | 2743.240000 | 2158.750000 | 2060.660000 | ... | 0.001000 | 0.000000 | -1.000000 |
| 25% | 2966.260000 | 2452.247500 | 2181.044400 | ... | 0.003300 | 44.368600 | -1.000000 |
| 50% | 3011.490000 | 2499.405000 | 2201.066700 | ... | 0.004600 | 71.900500 | -1.000000 |
| 75% | 3056.650000 | 2538.822500 | 2218.055500 | ... | 0.006400 | 114.749700 | -1.000000 |
| max | 3356.350000 | 2846.440000 | 2315.266700 | ... | 0.028600 | 737.304800 | 1.000000 |

Other conclusions

- A big number of NaN or null values
- Features with completely different magnitudes

PCA for data visualization

Feature reduction from 591 to 2



→ Classes not easily separable

Preprocessing

First preprocessing steps

- Separate Data and labels
- Remove first column with dates
- Substitute Nan for mean values
- Eliminate possible erroneous values, e.g. 0.0

Result:

→ Number of features columns reduced from 590 to 478

Other preprocessing steps

- Split the data by 20% to training test sets
- Standardize features to deal with different features magnitudes

Model choice

Model choice

Must take into account:

- Classification problem
- Large Dataset

K-means and Naive Bayes do not perform well on large dataset.

It leaves us with:

- Support Vector Machines:
Computationally cheap, only handles binary classification
- Decision Trees: Computationally cheap, can deal with irrelevant features, but tends to overfit
- Random Forests: Can also deal with irrelevant features but has lower tendency to overfit

Random Forests

Additionally gives us importance of variables

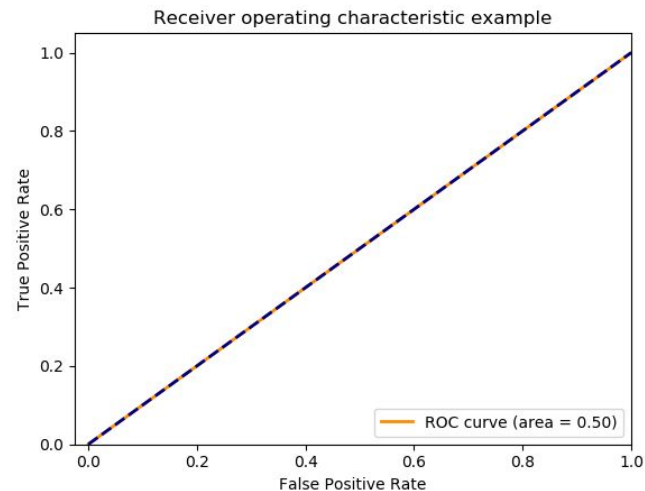
First results

By simply trying the standard parameters with the chosen model and preprocessing steps described

| | Precision | Recall | F1-Score |
|----|-----------|--------|----------|
| -1 | 0.93 | 1.00 | 0.97 |
| 1 | 0.00 | 0.00 | 0.00 |

Confusion Matrix

| | |
|--------|----------|
| TP = 0 | FN = 21 |
| FP = 0 | TN = 293 |



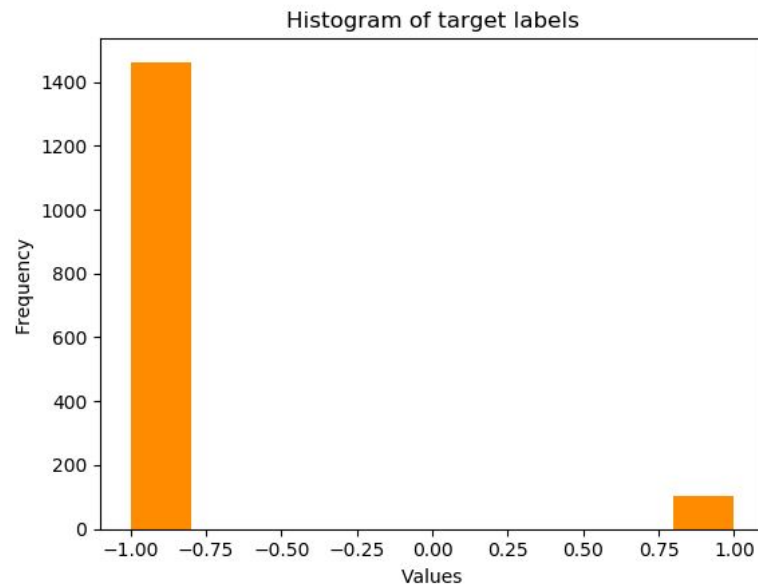
Conclusion

- Cannot predict positive values
- Resulting label is always “Fail”, regardless of the entries

Idea

- Investigate problems on the Dataset

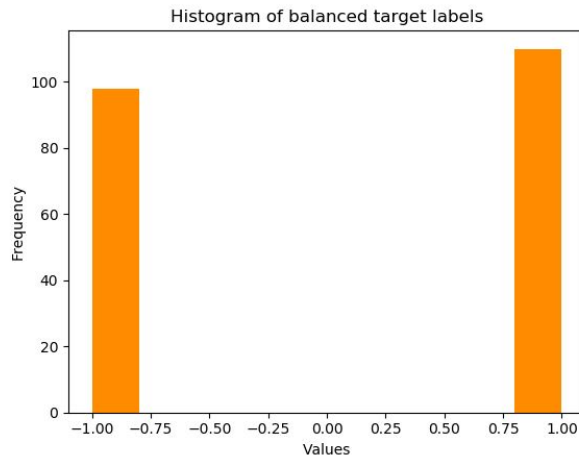
Highly imbalanced classes



Undersampling strategy

Since we have a large dataset

Simply make an equal distribution of data according to the class labels



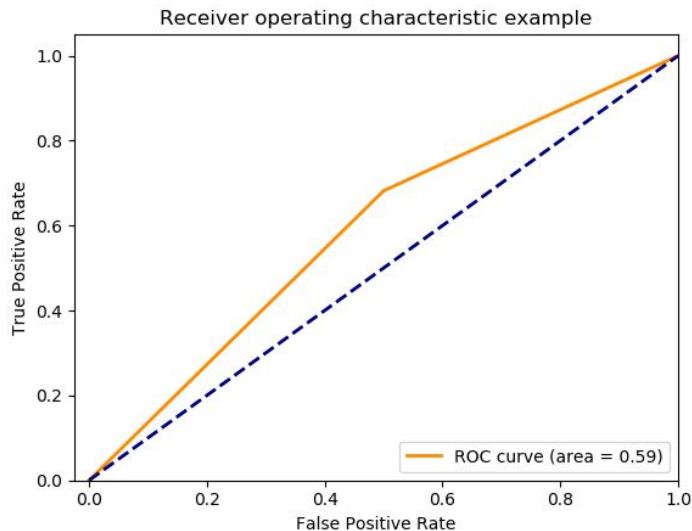
Undersampling first try

Same number of “Pass” and “Fail”
examples in the dataset

| | Precision | Recall | F1-Score |
|----|-----------|--------|----------|
| -1 | 0.59 | 0.50 | 0.54 |
| 1 | 0.60 | 0.68 | 0.64 |

Confusion Matrix

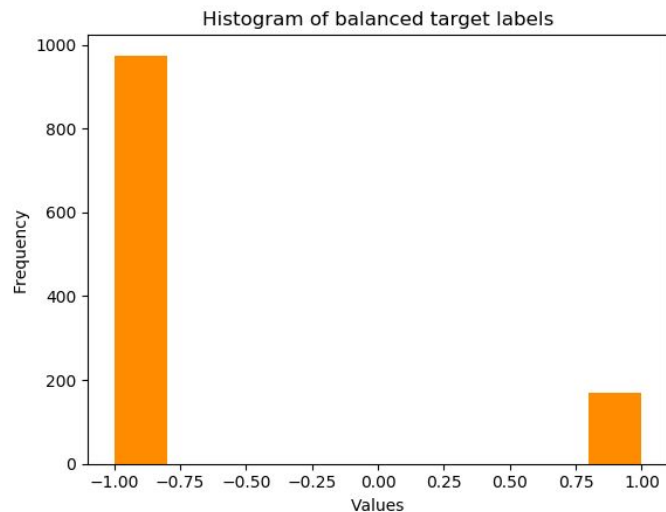
| | |
|---------|---------|
| TP = 15 | FN = 7 |
| FP = 10 | TN = 10 |



Change the undersampling strategy

→ Too few data

Manually exhausting optimization classes size



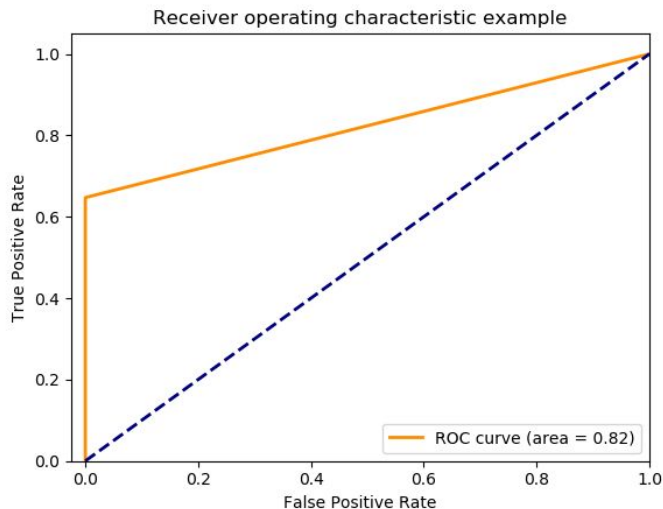
Undersampling optimal sizes

“Fail” class 10x bigger than “Pass” class

| | Precision | Recall | F1-Score |
|----|-----------|--------|----------|
| -1 | 0.94 | 1.00 | 0.97 |
| 1 | 1.00 | 0.65 | 0.79 |

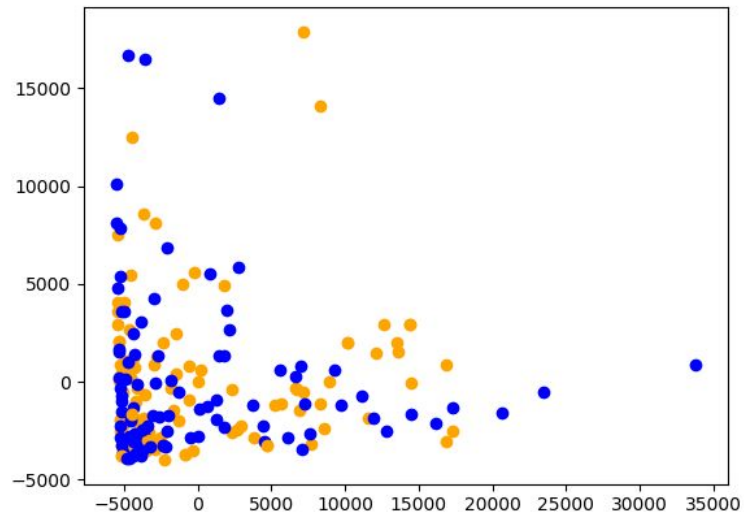
Confusion Matrix

| | |
|---------|----------|
| TP = 22 | FN = 12 |
| FP = 0 | TN = 195 |



PCA for data visualization

→ After undersampling



→ Classes more balanced but still not easily separable

Other undersampling strategy

Also tried “Tomek undersampling”
technique, without any improvement

Hyperparameters Optimization

Hyperparameters to optimize

- Number of trees
- Function to measure the quality of split
- Maximum depth of the tree
- Minimum number of samples required to split a node
- Minimum number of samples at a leaf node
- The number of features to consider when looking for the best split

Grid Search optimization strategy

- Exhaustive search
- 3-fold cross validation

Hyperparameters chosen

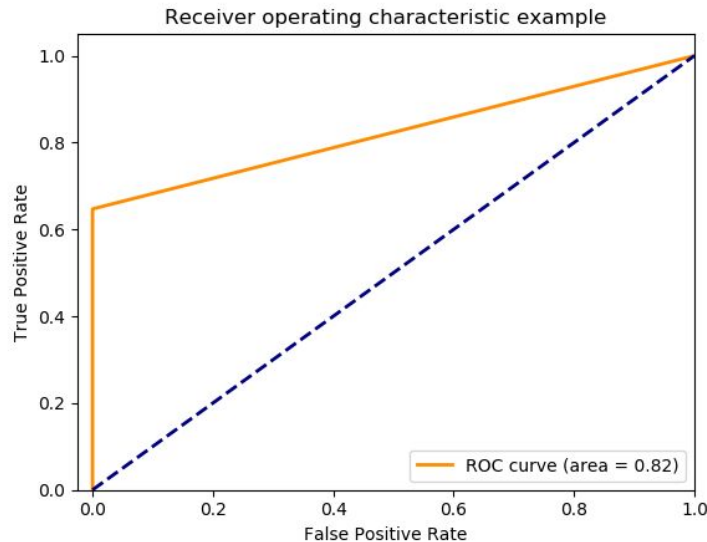
- Number of trees = 50
- Default values

| | Precision | Recall | F1-Score |
|----|-----------|--------|----------|
| -1 | 0.94 | 1.00 | 0.97 |
| 1 | 1.00 | 0.65 | 0.79 |

→ Same results as before

Confusion Matrix

| | |
|---------|----------|
| TP = 22 | FN = 12 |
| FP = 0 | TN = 195 |



Most important attributes

Classifying attributes

The Random Forests model in the Scikit Learn framework provides an attribute that classifies features in terms of its importance to predict a positive value.

10 most important attributes

| feature | importance |
|---------|------------|
| 59 | 0.016341 |
| 103 | 0.015929 |
| 341 | 0.011465 |
| 130 | 0.010547 |
| 129 | 0.009546 |
| 205 | 0.009080 |
| 477 | 0.008975 |
| 455 | 0.008767 |
| 519 | 0.008629 |
| 33 | 0.007735 |

→ Remark:

Even the most important feature has very low importance, supporting the difficulty to predict the positive result.

Features selection

Features selection

Some feature selection strategies were tested:

- Selecting the best half of features
- Selecting the best 100 features
- Selecting the best 20 features
- Selecting features with importance bigger than 0.004, 0.002...

→ But they all produced the same result as before, with no improvement

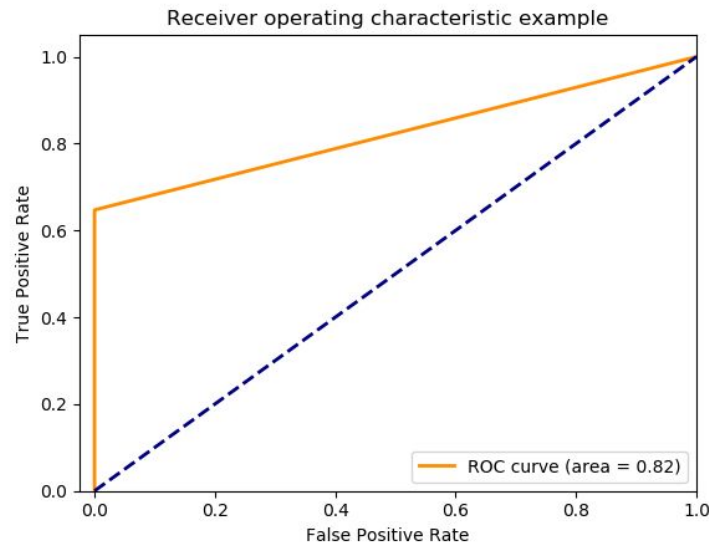
Features selection result

→ Same results as before

| | Precision | Recall | F1-Score |
|----|-----------|--------|----------|
| -1 | 0.94 | 1.00 | 0.97 |
| 1 | 1.00 | 0.65 | 0.79 |

Confusion Matrix

| | |
|---------|----------|
| TP = 22 | FN = 12 |
| FP = 0 | TN = 195 |



Conclusion

- The models or the hyperparameters optimization did not provide us big changes
- Even after applying some techniques the classes were not easily separable

- The biggest problems in that case were the dataset itself
 - ◆ Unbalanced
 - ◆ Large