Indian Institute of Technology Roorkee
ECN 358 Machine Learning Project Report

DEFECT PREDICTION

Vaishnavi Andhalkar

19116086

1 Introduction

Semiconductor manufacturing intersects with nearly all other IC process technologies, including design, fabrication, integration, assembly, and reliability. The end result is an electronic system that meets all specified performance, quality, cost, reliability, and environmental requirements. We need to make this manufacturing process, more and more productive to get optimum results. Many important technologies, in semiconductor manufacturing are invested over centuries ago, and still there is the evolution that is going on. There are over hundreds of unit processes throughout the process flow.

In this project, we will discuss the classifiers to predict whether or not a certain patterning defect mode will occur based on two inline measurements (scanner exposure and focus). This defect mode is typically only detected in post-etch inspection. However, if we could predict immediately after the exposure step (based on available scanner data) we could simplify the rework process and avoid potential yield loss. Our approach will loosely follow the OSEMN process. The computational work will be handled by Python and associated libraries.

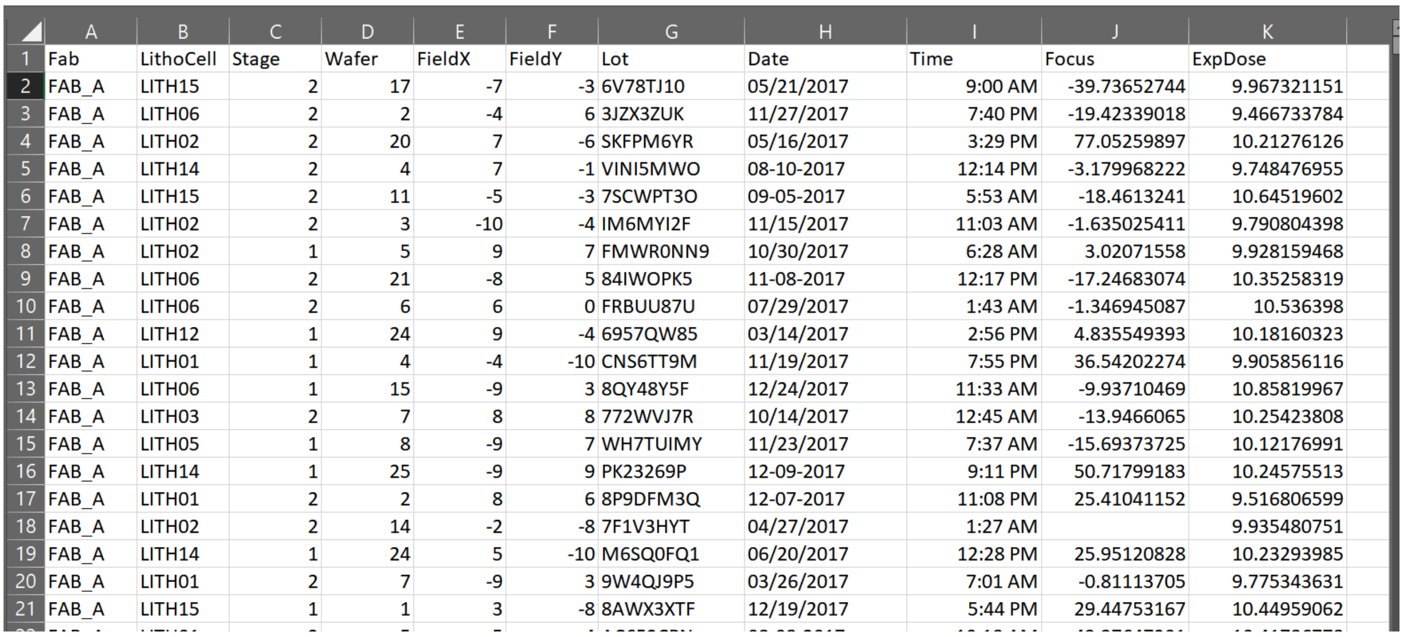
2 Motivation

As I mentioned ICs are the building blocks of any electronic system and we need to get higher yield through this process, we need to find ways to deal with possible failures within semiconductor manufacturing process. We check the devices time to time whether they are defected or not. At each stage we check the devices, but sometimes it is even possible that devices get damaged within a stage and if we don't do something, we will waste the yield of that part as well as the cost and efforts we put there. This is the main reason to do this project.

Also, to fasten up the process of detect prediction we use machine learning algorithms here. They are supervised algorithms which gives the output based on the training dataset available, we use some classifier models that I learned in the course. By using such models, and getting high accuracy, we can also increase the quality of the products. Detection of defects in early stage is way better than detecting them at later stages of semiconductor manufacturing process. We can get so much better results with these models rather than earlier technologies that we used to detect the defects.

3 Snapshot of data

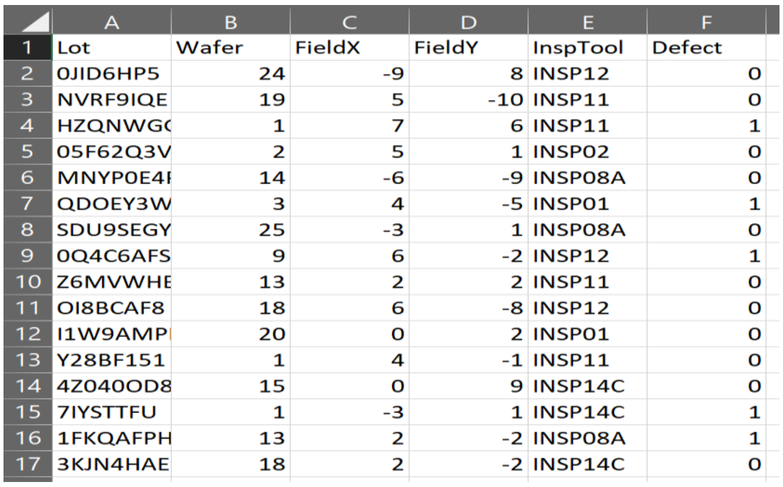
In this problem we are given two datasets, both in CSV(comm-separated values) format. 1st one is exposure data. It contains sample lithography scanner log containing following features.



	A	B	C	D	E	F	G	H	I	J	K
1	Fab	LithoCell	Stage	Wafer	FieldX	FieldY	Lot	Date	Time	Focus	ExpDose
2	FAB_A	LITH15	2	17	-7	-3	6V78TJ10	05/21/2017	9:00 AM	-39.73652744	9.967321151
3	FAB_A	LITH06	2	2	-4	6	3JZX3ZUK	11/27/2017	7:40 PM	-19.42339018	9.466733784
4	FAB_A	LITH02	2	20	7	-6	SKFPM6YR	05/16/2017	3:29 PM	77.05259897	10.21276126
5	FAB_A	LITH14	2	4	7	-1	VINI5MWO	08-10-2017	12:14 PM	-3.179968222	9.748476955
6	FAB_A	LITH15	2	11	-5	-3	7SCWPT3O	09-05-2017	5:53 AM	-18.4613241	10.64519602
7	FAB_A	LITH02	2	3	-10	-4	IM6MYI2F	11/15/2017	11:03 AM	-1.635025411	9.790804398
8	FAB_A	LITH02	1	5	9	7	FMWR0NN9	10/30/2017	6:28 AM	3.02071558	9.928159468
9	FAB_A	LITH06	2	21	-8	5	84IWOPK5	11-08-2017	12:17 PM	-17.24683074	10.35258319
10	FAB_A	LITH06	2	6	6	0	FRBUU87U	07/29/2017	1:43 AM	-1.346945087	10.536398
11	FAB_A	LITH12	1	24	9	-4	6957QW85	03/14/2017	2:56 PM	4.835549393	10.18160323
12	FAB_A	LITH01	1	4	-4	-10	CNS6TT9M	11/19/2017	7:55 PM	36.54202274	9.905856116
13	FAB_A	LITH06	1	15	-9	3	8QY48Y5F	12/24/2017	11:33 AM	-9.93710469	10.85819967
14	FAB_A	LITH03	2	7	8	8	772WVJ7R	10/14/2017	12:45 AM	-13.9466065	10.25423808
15	FAB_A	LITH05	1	8	-9	7	WH7TUIMY	11/23/2017	7:37 AM	-15.69373725	10.12176991
16	FAB_A	LITH14	1	25	-9	9	PK23269P	12-09-2017	9:11 PM	50.71799183	10.24575513
17	FAB_A	LITH01	2	2	8	6	8P9DFM3Q	12-07-2017	11:08 PM	25.41041152	9.516806599
18	FAB_A	LITH02	2	14	-2	-8	7F1V3HYT	04/27/2017	1:27 AM		9.935480751
19	FAB_A	LITH14	1	24	5	-10	M6SQ0FQ1	06/20/2017	12:28 PM	25.95120828	10.23293985
20	FAB_A	LITH01	2	7	-9	3	9W4QJ9P5	03/26/2017	7:01 AM	-0.81113705	9.775343631
21	FAB_A	LITH15	1	1	3	-8	8AWX3XTF	12/19/2017	5:44 PM	29.44753167	10.44959062

It has features like fab name, lithography tool name (LithoCell), Scanner stage number (stage), wafer ID, Column number and row number on wafer, focus and exposure dose are measured in nm and mJ respectively etc.

2nd dataset is sample wafer inspection data for a particular defect type containing the following columns.



	A	B	C	D	E	F
1	Lot	Wafer	FieldX	FieldY	InspTool	Defect
2	0JID6HP5	24	-9	8	INSP12	0
3	NVRF9IQE	19	5	-10	INSP11	0
4	HZQNWGC	1	7	6	INSP11	1
5	05F62Q3V	2	5	1	INSP02	0
6	MNYP0E4I	14	-6	-9	INSP08A	0
7	QDOEY3W	3	4	-5	INSP01	1
8	SDU9SEGY	25	-3	1	INSP08A	0
9	0Q4C6AFS	9	6	-2	INSP12	1
10	Z6MVWHE	13	2	2	INSP11	0
11	OI8BCAF8	18	6	-8	INSP12	0
12	I1W9AMPI	20	0	2	INSP01	0
13	Y28BF151	1	4	-1	INSP11	0
14	4Z040OD8	15	0	9	INSP14C	0
15	7IYSTTFU	1	-3	1	INSP14C	1
16	1FKQAFPH	13	2	-2	INSP08A	1
17	3KJN4HAE	18	2	-2	INSP14C	0

This dataset contains the lot ID and some other features. The last column of the dataset Boolean value representing the absence or presence of the defect. 0 represents absence while 1 represents presence of defect. This all data was generated randomly and does not correspond to any actual fabrication data.

4 methodology analysis

4.1 Importing necessary libraries

We will import some useful libraries in python particularly, matplotlib (for data visualization), pandas (for data manipulation and analysis), NumPy (for numerical computing), scikit-learn (for machine learning). These will help us to analyse the data.

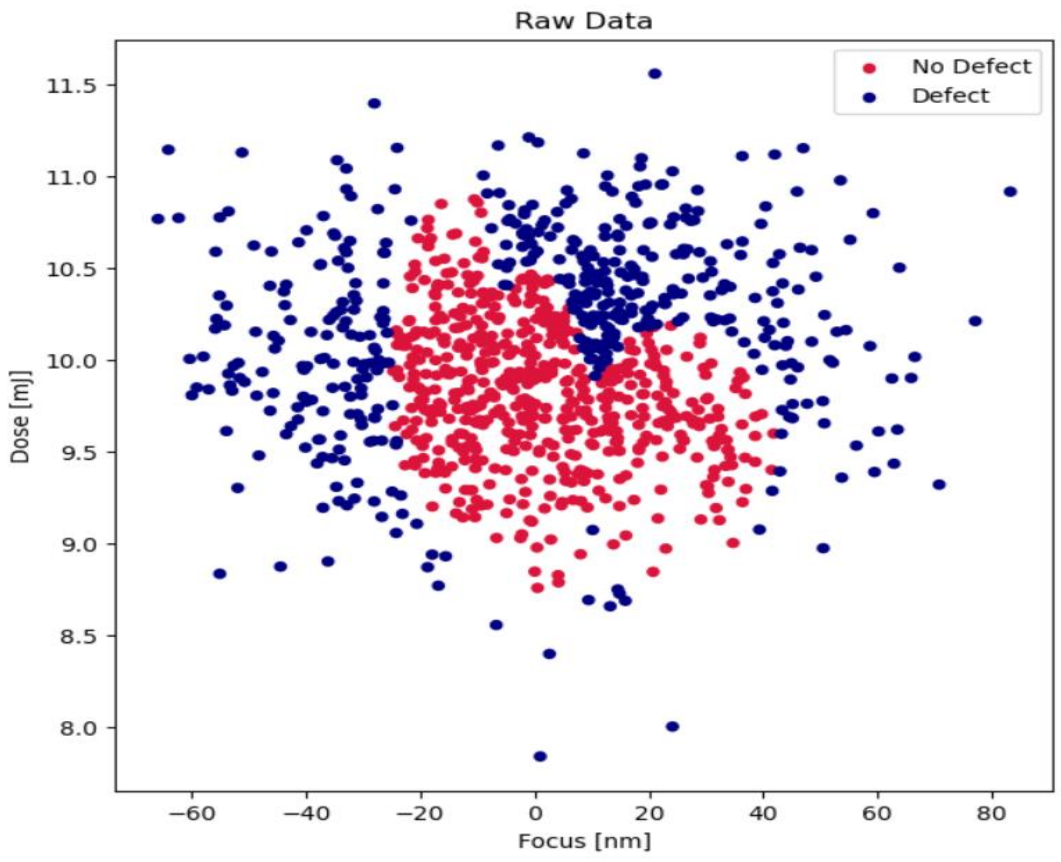
4.2 Scrubbing the given data

Before starting the analysis, we need to look the data once, we need to make some modifications in the data before that. We import the data in a data frame, using pandas read.csv function. After that we will check if there are any missing values in the dataset. As we can see in the code there are some missing values in focus and exposure dose column. To deal with this we will drop the entire row. We will repeat this process for both the datasets.

Finally, we are remained with 1000 rows. Then we merge these datasets together to associate the correct inspection results for each focus/exposure point. After doing this, we will convert the larger dataset into two different arrays, X containing the feature data and Y containing the label data.

4.3 Explore the data

Now, to get the intuition of the given data, we will scatter plot the given data.



Here we can see that there does appear to be a relationship between scanner dose/focus and the occurrence of a defect. The 'no-defect' observations seem to be clustered together into a region (often referred to as a process window). We should be able to train a model to learn the boundaries of this region and predict whether or not a defect will occur based on a given dose/focus observation.

In this project, we will use three different classifiers to classify this given data and then we will compare the accuracy of those models. We will use Support Vector classifier (SVC), Random Forest classifier and K nearest neighbours (KNN) classifiers to classify the given data.

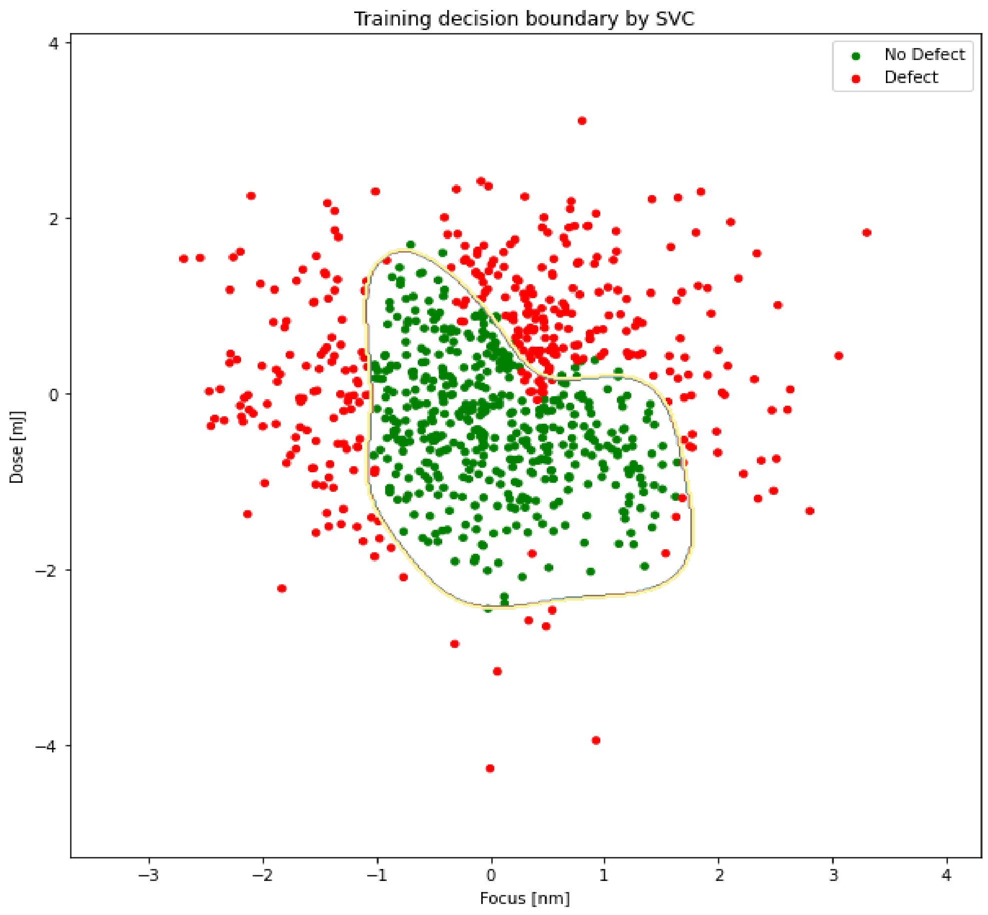
Before defining the models and training them, we will split the given data into two groups, training dataset and test dataset. We will train our model using training dataset and then check the accuracy on test dataset. Sci-kit learn library has inbuilt functions to do so. In addition, we may note that the range of our two features is quite different: the range of focus values is approximately 140nm, while the range of dose values is only about 3.5 mJ. While this isn't too extreme, some efforts to normalize the data so that all the features have a similar mean and range may improve the performance of the optimizer used to fit the model.

4.4 Models

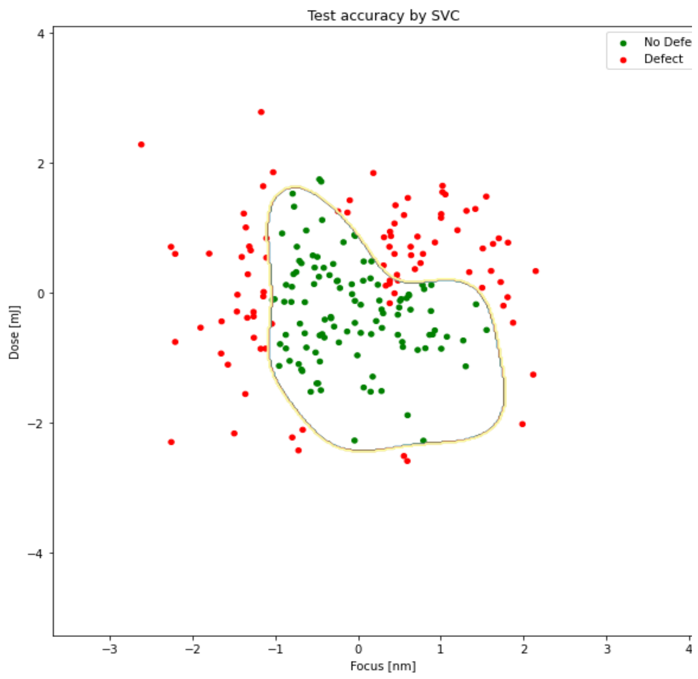
To classify the data between defects and non-defects, we'll train three different models. During the training of our models, we will tune the hyperparameters such that we will get optimum accuracy.

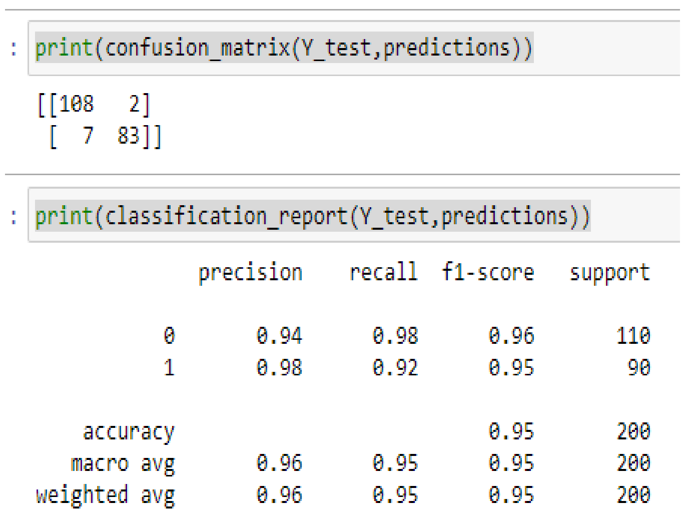
4.4.1 Support Vector Classifier

Support Vector classifier is a supervised leaning method for classification. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.



This is the decision boundary we get by training the data using SVC. Now, keeping the boundary same we'll plot the test dataset and check the test accuracy





```
: print(confusion_matrix(Y_test,predictions))

[[108   2]
 [  7  83]]
```

```
: print(classification_report(Y_test,predictions))

              precision    recall  f1-score   support

           0       0.94      0.98      0.96       110
           1       0.98      0.92      0.95        90

    accuracy                           0.95       200
   macro avg       0.96      0.95      0.95       200
weighted avg       0.96      0.95      0.95       200
```

As we can see here, the test data is plotted against the decision boundary we got using training dataset. We then can see the confusion matrix and accuracy of this model.

$$\begin{vmatrix} \text{true negatives (TN)} & \text{false positives (FP)} \\ \text{false negatives (FN)} & \text{true positives (TP)} \end{vmatrix}$$

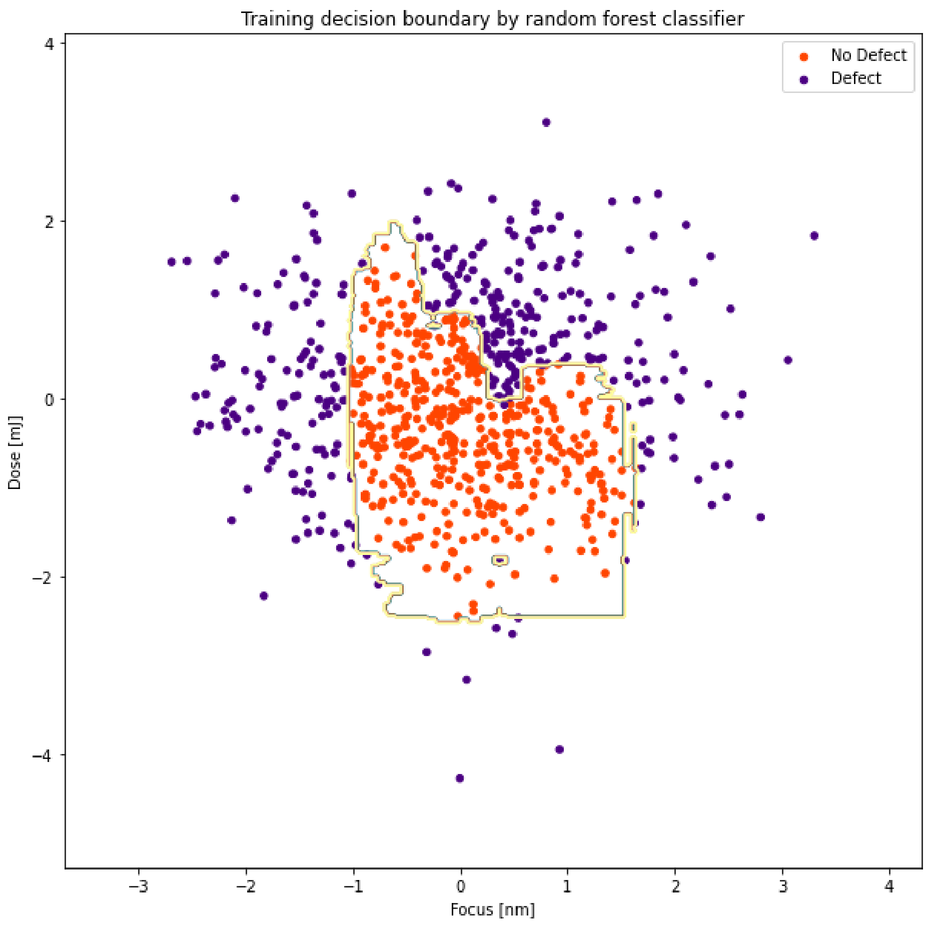
This is the interpretation of confusion matrix. In this classifier, our classifier correctly predicted 108 cases where there is no defect and 83 cases where there is defect. However, it incorrectly predicted that there was no defect in 7 cases (false negatives) and that there was a defect on 2 cases where there actually was no defect (false positives).

We can also generate a classification report with some key metrics. The precision is defined as the ratio of the number of true positives (defects in this case) predicted to the total number of positives (true and false) predicted by the classifier, i.e., TP/TP+FP. The recall is defined as the ratio of the number of true positives predicted to the total number of positives in the data, i.e., TP/TP+FN The F-score is defined as the harmonic mean of the precision and recall i.e. 2*precison*recall/Precision+recall.

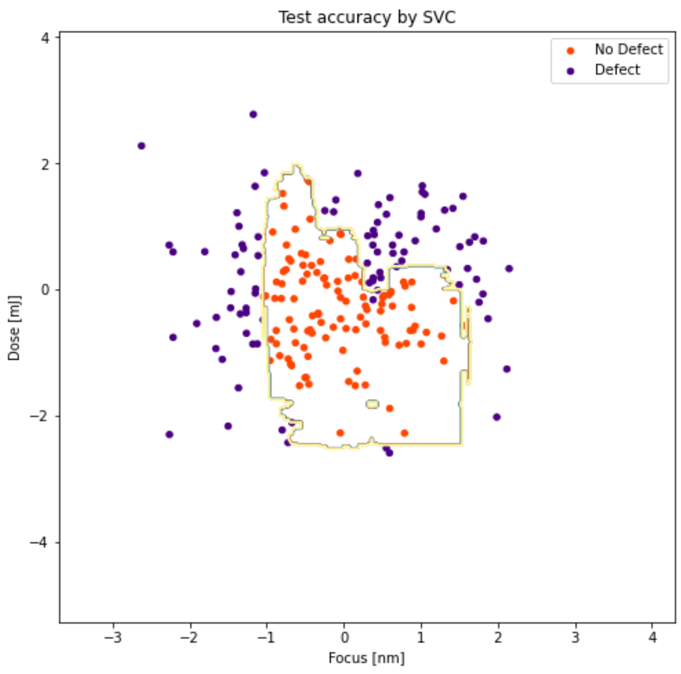
4.4.2 Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. It has a hyper parameter n_estimators which is the number of trees in the forest. I checked the result of the decision boundary on various number of trees. Finally, I tuned this hyper parameter to 230.

As we can see in following figures the accuracy for this classifier is more than that of SVC. We got only five points wrong in a set of 200 test points.



This is the decision boundary by training dataset. The test data is plotted below.



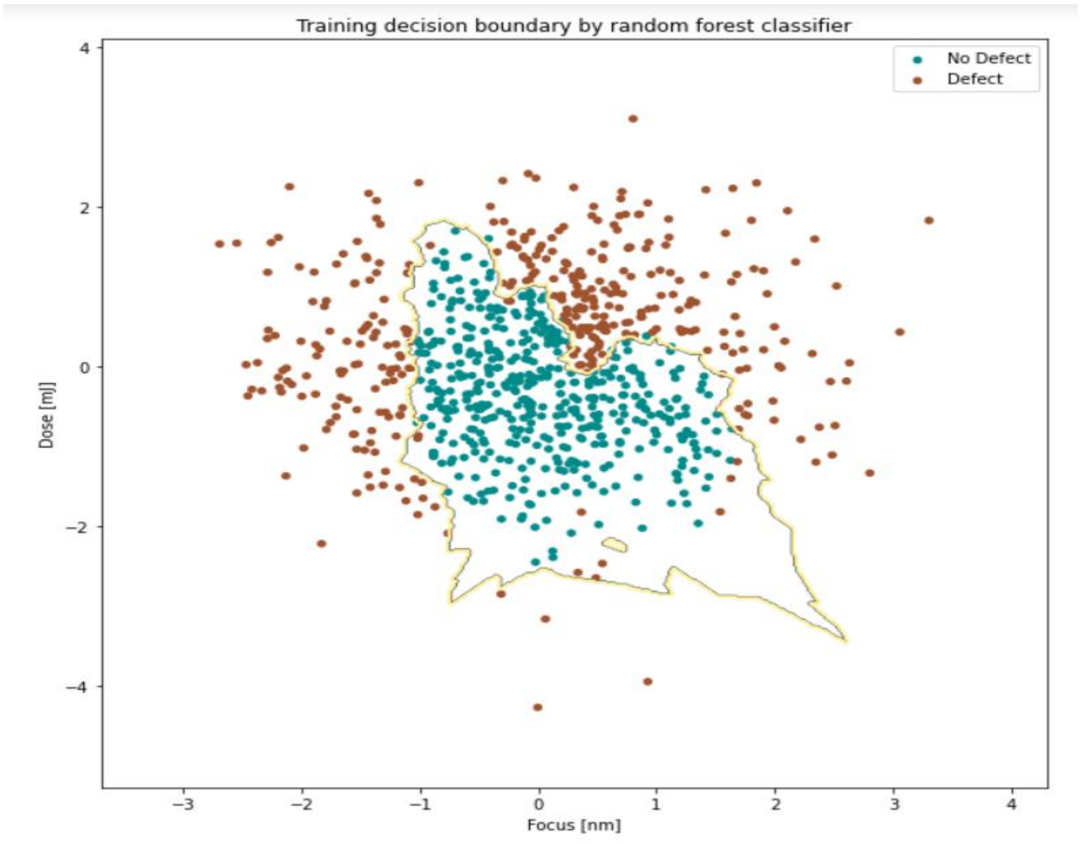
```
In [34]: print(confusion_matrix(Y_test,predictions))

[[108   2]
 [  3  87]]
```

```
In [35]: print(classification_report(Y_test,predictions))

              precision    recall  f1-score   support

           0       0.97      0.98      0.98       110
           1       0.98      0.97      0.97        90

    accuracy                           0.97       200
   macro avg       0.98      0.97      0.97       200
weighted avg       0.98      0.97      0.97       200
```

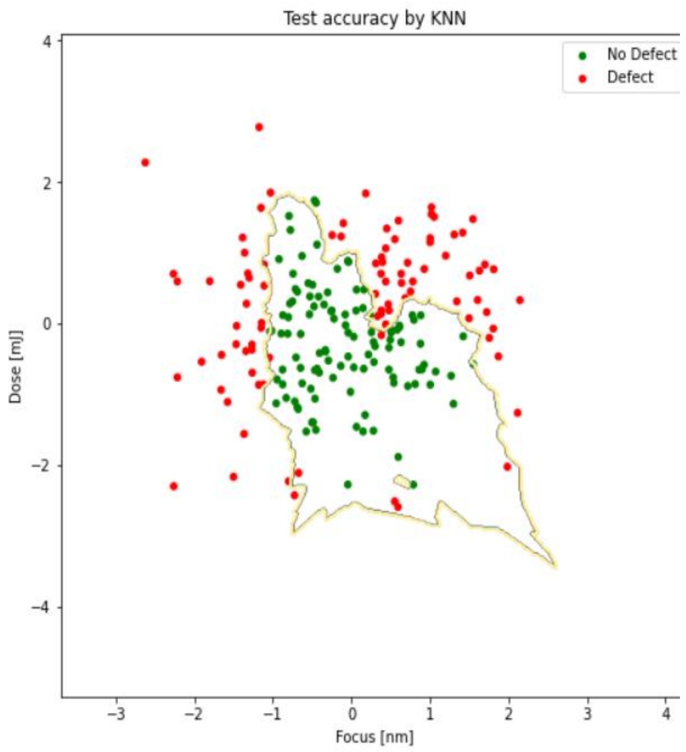
Here we can see that, this model gives better results than the previous one. Here total 5 dataset we got wrong. Accuracy of this model is better than the previous one.

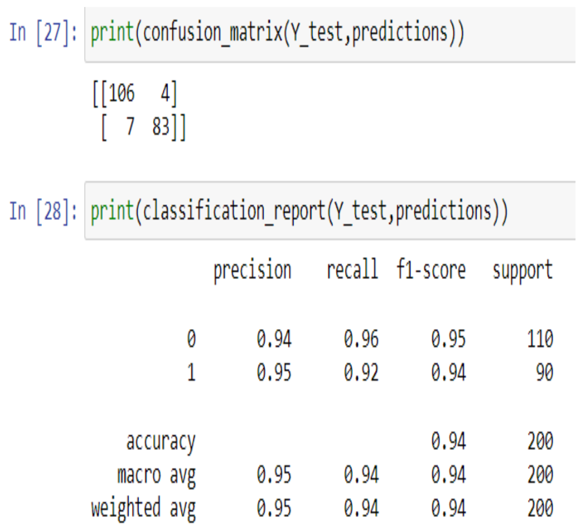
5.4.3 K-nearest neighbours (KNN)

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new data and available cases and put the new case into the category that is most similar to the available categories. The hyper parameter in this model is the number of nearest neighbours that we consider while training the model. I tuned the hyper parameter k to 7, in this case.



The decision boundary by this model is as above, we will plot the test data on this boundary now.





```
In [27]: print(confusion_matrix(Y_test,predictions))

         [[106   4]
          [  7  83]]

In [28]: print(classification_report(Y_test,predictions))

                      precision    recall  f1-score   support

                   0       0.94      0.96      0.95       110
                   1       0.95      0.92      0.94        90

            accuracy                           0.94       200
           macro avg       0.95      0.94      0.94       200
        weighted avg       0.95      0.94      0.94       200
```

So, as we can see here the wrong predictions are 11 out of 200. This is classifier is not that better than previous one.

6 Conclusions

In this example, we followed the OSEMN approach to data science to build a classifier system to predict whether a printability defect was likely two occur based on two observed features (scanner dose and focus). We started with a set of raw data consisting of 1,000 observations of dose and focus along with corresponding labels for each observation indicating whether a defect occurred in those conditions or not.

After 'scrubbing' the data to remove or resolve any invalid entries, we used some visualization tools to explore the data. We observed that the way that the 'good' observations were clustered together might lend itself to a classification approach. After some further work to normalize the data and split it into training and test sets, we were ready to build our classifier.

The classifiers that we implemented, SVC, Random Forest classifier, KNN gave us the decision boundaries. Then we checked the accuracy of those models on test data.

Accuracy of all the three classifiers are given below.

SVC = 0.96

Random Forest Classifier = 0.98

KNN = 0.95

We can conclude here that Random Forest Classifier is the best classifier for this classification.

7 applications

Now that we have a classifier that seems to predict the occurrence of the defect based on the scanner data with reasonable accuracy, we could consider making it part of the process control scheme for the fab. The factory automation software could route scanner data (including focuse and exposure dose) to the classifier and take the appropriate action based on the prediction made. If the classifier predicts a defect, the lot could be flagged for closer inspection or rework. Otherwise, the wafer can proceed through the process flow normally.

Of course, any change to the process (e.g., nominal process conditions, materials, etc.) could impact the accuracy of the classifier. After any such change is made it may be necessary to retrain the model.

8 Link to the dataset

https://drive.google.com/drive/folders/1q7cH_fU8hGCbQwcRRn1K6eEG3O7SJIvB?usp=sharing