

MANUAL FOR DEFECT PREDICTION WITH PREST

1. Things To Know For Defect Prediction:

Defect prediction in the most basic sense is classifying your functions or classes into two classes: defective or non-defective. While separating your data into two classes or defective or non-defective the first thing we need to decide is the granularity level of our prediction. In other words, whether we will do defect prediction for functions, classes, files or packages. For each granularity level, Prest (defect prediction tool of SoftLab) can extract metrics, but we will come to metrics and so on a little later. For the ease of use, we will assume our granularity level as functions, i.e. we will assume that we are trying to decide whether the functions in a project are defective or not. However, we can generalize everything that is told on functions to classes, files or packages. Below are some concepts we will need to know for defect prediction:

1.1 Dataset: The dataset we use for defect prediction is basically a 2D array, whose rows correspond to functions or classes (depending on our granularity level of matching bugs) and whose columns correspond to static code metrics that are derived by Prest. Only the last column of the dataset is the class label that can have two values: TRUE, meaning the row is defective or FALSE, meaning the row is non-defective. Below is a screenshot of our dataset (I could not show all the 28 metrics, but it gives an idea of our dataset), where green colored ones are non-defective and red colored ones are defective.

Table 1. Dataset on which defect prediction will be performed

Method Name	Method Id	Call Pair Length	Defected?(false/true)
trial1Func1	18	1	FALSE
trial1Func2	19	0	FALSE
main	20	2	FALSE
trial2Func1	23	0	TRUE
trial2Func2	24	1	FALSE

1.2 Train Data: The data for which we already matched the actual bugs to functions and/or classes. In fact the dataset which is given in Table 1 is an example of a train data.

1.3 Test Data: When Prest parses a project and generates the datasets as in Table 1, the last column only has the value of FALSE, indicating that no actual bugs are matched yet. This will be our test data and depending on train data (data related to a past release for which we have already matched the bugs to functions), we will be able to predict which functions in the test data are going to be defective.

2. Getting Hands Dirty: How to Perform Defect Prediction with Prest?

Since now we know some of the necessary jargon related to defect prediction, we can proceed with performing defect prediction by using Prest. When you open Prest, you will see the GUI of metric extraction, which is given below in Figure 1. However, the GUI we will use will be the “Prediction” view. To switch to prediction view, press the “Switch View” button.

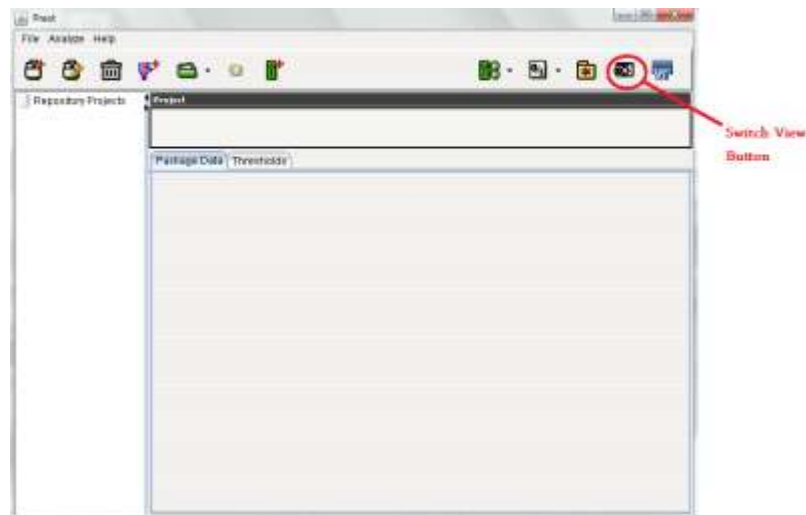


Figure 1. Opening view of Prest

In Figure 2, defect prediction view and the related buttons together with their brief explanations are given.



Figure 2. Defect prediction view of Prest and related buttons

2.1 Step by Step Defect Prediction: After switching to defect prediction view, we can perform the defect prediction step by step as follows:

2.1.1 Load Training Data: Press the drop-down arrow of dataset selection button and select the train data. One caution here is that, the only data format that is accepted by the defect prediction of Prest is *.arff file format.



Figure 3. Press Load Training Set

In our scenario, let us assume that we have the actual bugs matched with functions for release 4 and we are trying to predict the bugs for release 5. In that case, as the training dataset we need to select the data related to release 4 (as given in Figure 4).

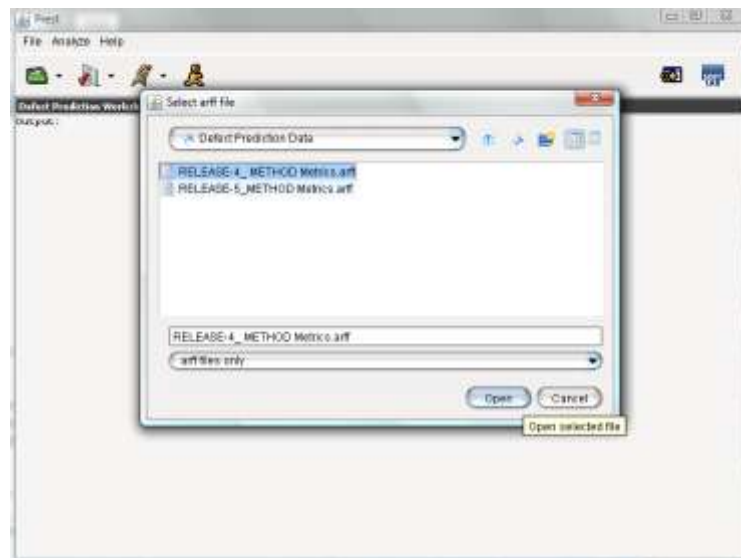


Figure 4. Select train dataset

After selecting the train dataset, we will return to the defect prediction window and we will select the test dataset just like the way we selected the train dataset. If the loading of the file is successful, we will see a message indicating that the file was loaded successfully.

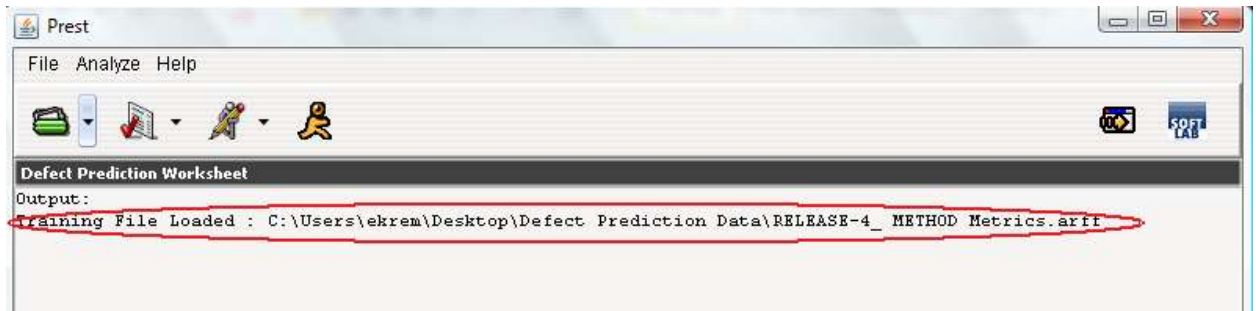


Figure 5. Message indicating that train file was loaded successfully

- 2.1.2 **Load Test Data:** Loading the test data is the same as loading the train data. We press the drop-down arrow of dataset selection button and select the test data. Since in our scenario, we are aiming the predict the defects for release 5, we will select the dataset of release 5. Remember that for release 5, the actual bugs are not matched and we want the program to predict the likely error-prone functions for us before testing.

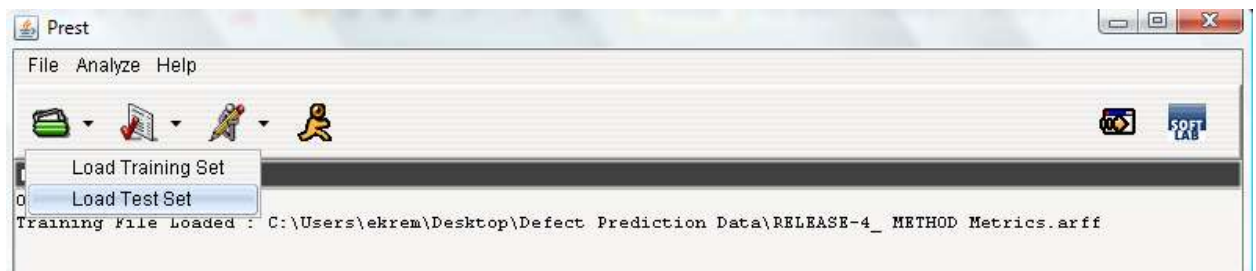


Figure 6. Press Load Test Data

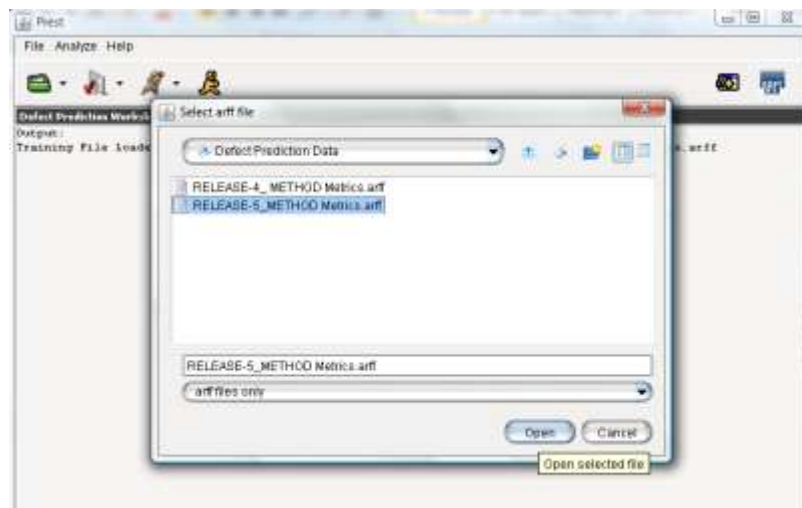


Figure 7. Select test dataset

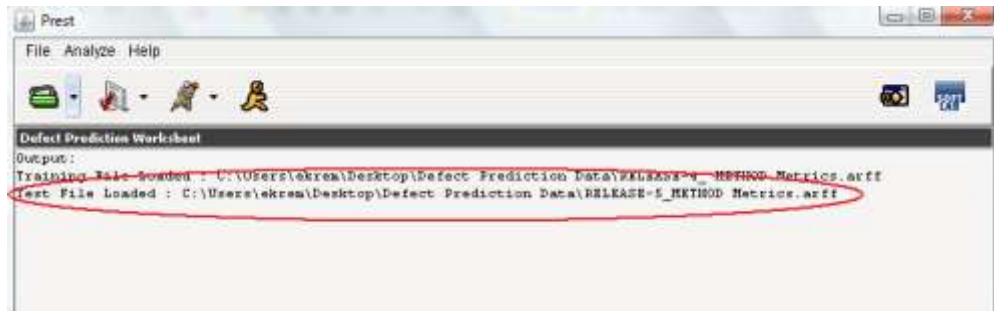


Figure 8. Message indicating that test file was loaded successfully

2.1.3 **Select Options (optional):** Since now we have successfully loaded the test and train data, we can start with prediction. However, before pressing the run button, we can change a couple of things via the options button. When we click the drop-down arrow of the options button, we see two options: 1) 10 Fold Cross Validate and 2) Normalize Data. We can proceed with or without selecting them. However, if you want to try them out their functionalities are as follows:

- **10 Fold Cross Validate:** Divides the test data into ten equal pieces and for each one of the ten pieces, it makes prediction for 10 times and the actual prediction is the mean of those ten predictions.
- **Normalize Data:** When the deviation of the values within the dataset is too big, it is better to normalize the data so that the deviation becomes limited.

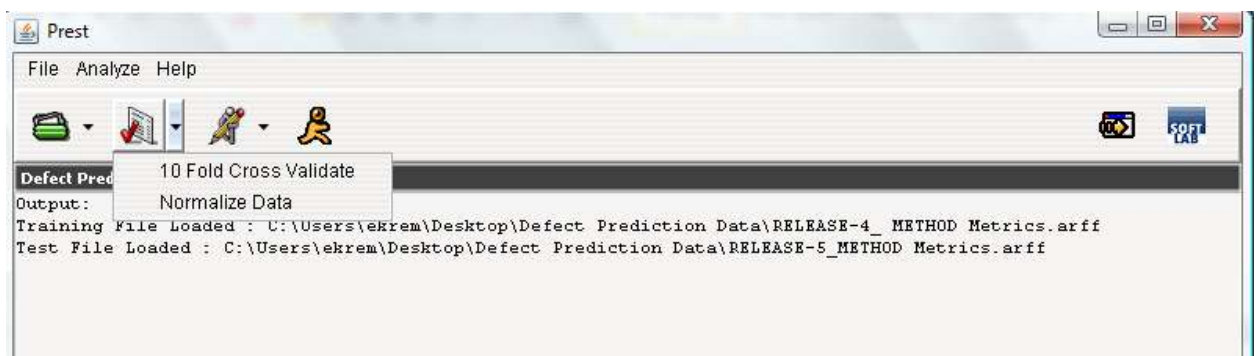


Figure 9. Options that are available via Options Button

2.1.4 **Select Algorithm (optional):** Currently two algorithms are provided in Prest: 1) Naïve Bayes 2) J48 (Decision Tree). By default Naïve Bayes algorithm is selected. However, if you wish, you can change the algorithm that is going to be used in the defect prediction.

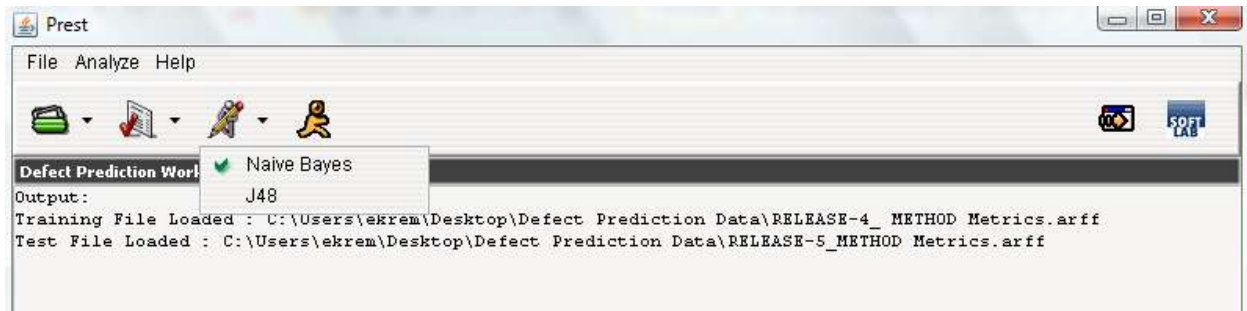


Figure 10. Selecting the algorithm that will be used in defect prediction

2.1.5 **Start Defect Prediction:** Now all that is left to do is to start the defect prediction and get the output. For that, just press the start button and the output will be displayed in the output section.

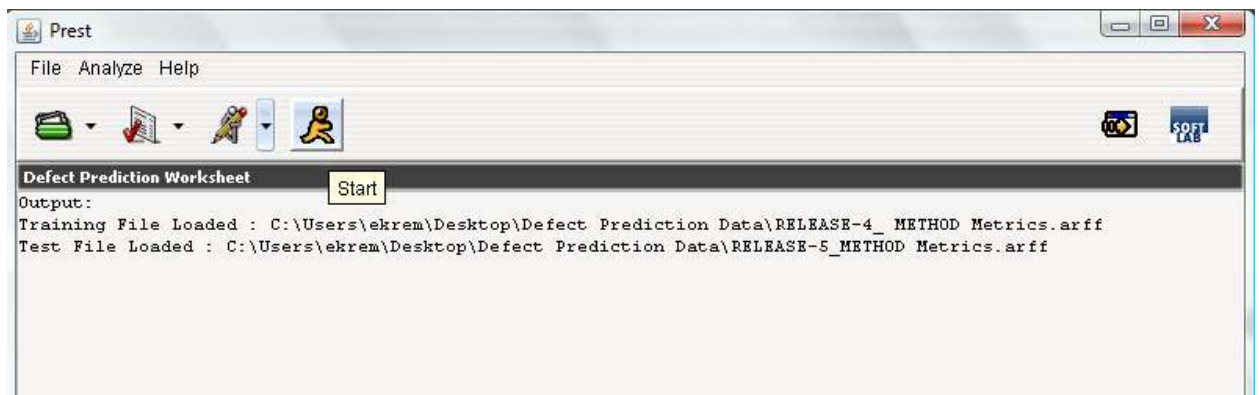
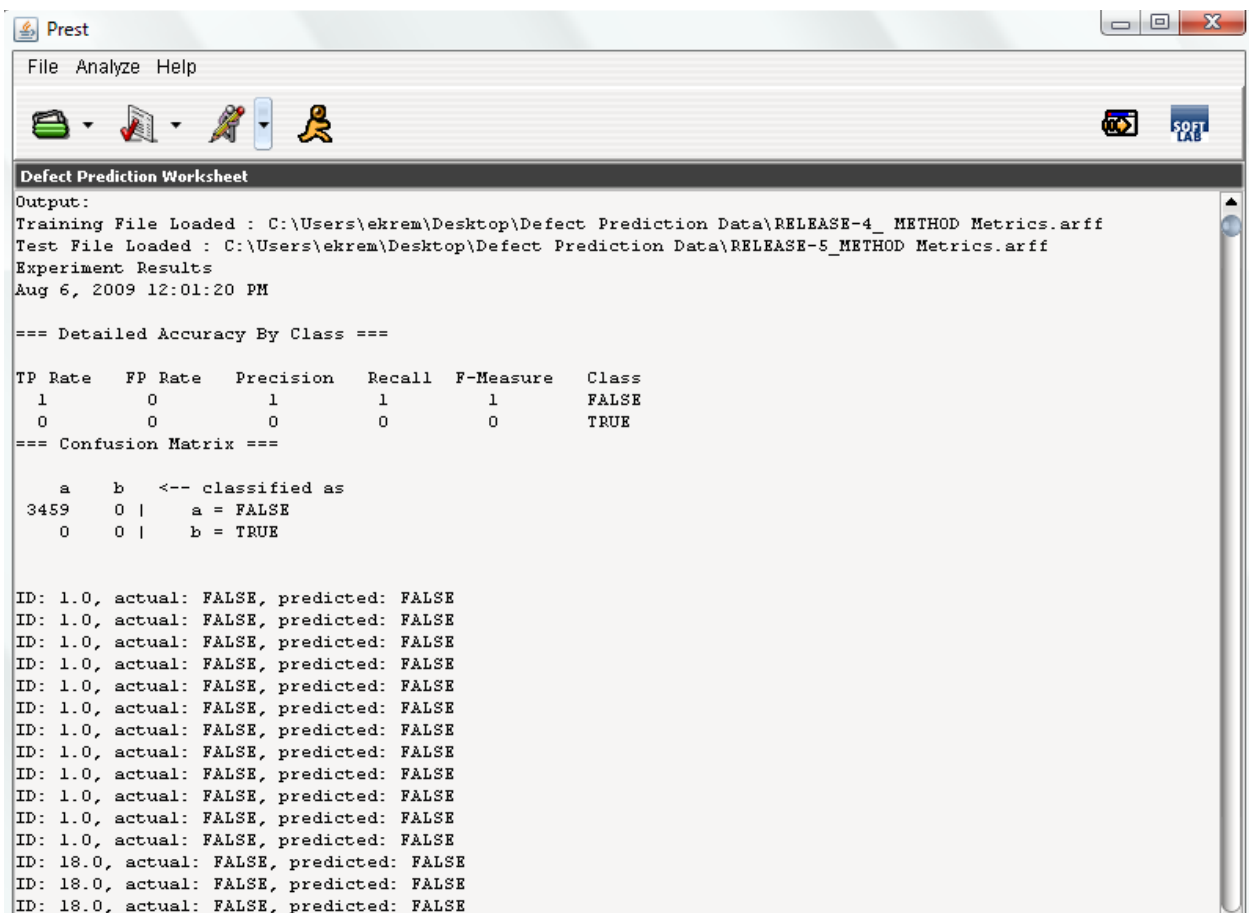


Figure 11. Press start button for Prest to start defect prediction

2.1.6 **Analyse the output:** Below is the output of Prest. In that output, the actual and the predicted values for release 5 are provided. By the way, it is worth mentioning that depending on the size of train and test data, the prediction process may take some time.

On the output screen, there is a long list of ID's, actual and predicted values. Where we need to focus is the predicted values. If the predicted value of a function is TRUE, then this function is predicted to be defective and depending on the bugs of release 4, it is highly likely that this function will be defective and is worth taking a look at. On Figure 12, the screenshot of the output is provided.



The screenshot shows the Prest application window with a menu bar (File, Analyze, Help) and a toolbar. The main area displays the 'Defect Prediction Worksheet' with the following content:

```

Output:
Training File Loaded : C:\Users\ekrem\Desktop\Defect Prediction Data\RELEASE-4_METHOD Metrics.arff
Test File Loaded : C:\Users\ekrem\Desktop\Defect Prediction Data\RELEASE-5_METHOD Metrics.arff
Experiment Results
Aug 6, 2009 12:01:20 PM

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall   F-Measure   Class
1         0         1           1         1           FALSE
0         0         0           0         0           TRUE

=== Confusion Matrix ===

  a    b  <-- classified as
3459   0 |    a = FALSE
  0     0 |    b = TRUE

ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 1.0, actual: FALSE, predicted: FALSE
ID: 18.0, actual: FALSE, predicted: FALSE
ID: 18.0, actual: FALSE, predicted: FALSE
ID: 18.0, actual: FALSE, predicted: FALSE

```

Figure 12. Output of Prest defect prediction