**FIT 3162/ FIT COMPUTER SCIENCE PROJECT**

**Software Test/QA Report**

**Project title: Improving Software Testing using Software Fault Prediction Methods when data is highly imbalanced**

**Team:**

FIT3162\_MA\_4

**Team Members:**

* Tah Wen Zhong
* Jason Toh Zhern Wee
* Ethan Hor Sheng Jian

**Project Supervisor:**

Golnoush Abaei

**Table of contents**

[**1. Introduction**](#_heading=h.2ah822pvvhq3) **4**

[**2. Testing Plan**](#_heading=h.72f5pdr1db2h) **4**

[**3. Test approach used**](#_heading=h.x71xflbmjgzh) **5**

[**4. Unit Testing**](#_heading=h.qxnjgwe7054) **5**

[4.1 Base Models](#_heading=h.aa28n1rlu106) 6

[4.1.1 Complement Naïve Bayes](#_heading=h.oxbxxnnxmq0i) 6

[4.1.2 Decision Tree Model](#_heading=h.2889o4a54h) 7

[4.1.3 Logistic Regression Model](#_heading=h.byu54h28uv53) 8

[4.1.4 Multi-Layer Perceptron Model](#_heading=h.jv5vhqrxv0lo) 9

[4.1.5 Naïve Bayes Model](#_heading=h.ijf6i6xlvwun) 10

[4.2 Ensemble Models](#_heading=h.y78w1pe1f1h0) 11

[4.2.1 Random Forest Model](#_heading=h.vibs1wsewv3u) 11

[4.2.2 Rotation Forest Model](#_heading=h.65qth36anuu9) 12

[4.2.3 Voting](#_heading=h.uded7l6ouxhc) 13

[4.3 Evaluation Metric](#_heading=h.kxbeyz3st3j6) 14

[4.3.1 AUC ROC](#_heading=h.fh6vhwsltwyc) 14

[4.3.2 F1-Score](#_heading=h.5tkvrtyik29s) 15

[4.4 Data Preprocessing](#_heading=h.6o8rpxb7ey0s) 16

[4.4.1 CFS](#_heading=h.5bkoheazx5vu) 16

[4.4.2 RFE](#_heading=h.9leae9t2v5g3) 17

[4.4.3 IHT](#_heading=h.rf7vsvshjygf) 19

[4.4.4 K-fold](#_heading=h.z733pp3ibv0s) 20

[4.4.5 Normalization](#_heading=h.eysmrf2wlew4) 21

[**5. Integration Testing**](#_heading=h.kmf00fbe3dcd) **22**

[5.1 Main Test](#_heading=h.74nrr5awrcow) 23

[5.2 Preprocess](#_heading=h.j6dvundd7v6v) 31

[**6. Graphical User Interface Testing**](#_heading=h.rrwql2zb63jo) **32**

[**7. System Testing**](#_heading=h.b6d3tan4z44j) **39**

[7.1 Algorithms](#_heading=h.j7jv4ev968ma) 39

[7.2 Preprocessing](#_heading=h.7617lr9ms2hm) 40

[7.3 Main program](#_heading=h.hthg02ukjbgs) 40

[7.4 User interface](#_heading=h.khcayjgf9q4q) 41

[**8. Usability Testing**](#_heading=h.80pxyniup7pc) **45**

[**9. Performance, Scalability, Security**](#_heading=h.80auud5bs1s) **46**

[9.1 Performance](#_heading=h.gkbw7ehlvdhe) 46

[9,2 Scalability](#_heading=h.sxlb36u4s81v) 46

[9.3 Security](#_heading=h.ju2a7vivnfoa) 46

[**10. Limitation of the software**](#_heading=h.73mlebgbubaw) **47**

[**11. Recommendation for improvements**](#_heading=h.nvd7dsdr4t0k) **47**

[**12. Test Limitation**](#_heading=h.aobjsjt1r7lf) **48**

[**13. Conclusion**](#_heading=h.oafbgt3rciqp) **49**

# 1. Introduction

Our project utilizes many machine learning algorithms to build models that can predict the fault-proneness of software modules. These algorithms need to be thoroughly tested to ensure correctness as well as producing higher accuracy. Not only that, our program contains a variety of components such as a graphical user interface for better usability, the normalization of datasets through preprocessing and algorithms for feature selection. With that said, further details regarding the testing plan will be explained in the next section.

# 2. Testing Plan

Our test plan follows the testing pyramid methodology which consists of unit testing, integration testing, system testing as well as manual testing. Unit testing would be carried out on the machine learning and feature selection algorithms to ensure it produces a working model when given appropriate parameters. Integration testing is done on the main program and preprocessing component to verify that it is able work as a whole. System testing is black box testing where we will be testing the overall program to ensure it meets the functional and non-functional requirements. Usability testing is conducted where a group of participants will be given a user manual to test out the said functionalities of the program and provide appropriate feedback. Finally, the findings will be compiled to identify any testing limitations of the program.

# 

# 3. Test approach used

The nature of our program is a GUI application where users are able to interact with the interface of our program. Therefore, it will contain both user interface testing and backend testing. Also, each testing will also be tested for robustness to ensure the program does not crash whilst in the middle of running.

For user interface testing, testing will be done in two iterations. The first is manual testing and it will be done by our team and the second will be done by participants. During manual testing, our team will be deliberately providing a wide variety of inputs both valid and invalid and each attempt is logged in a table. We will also deliberately provide invalid inputs in the program to test the behavior of the program and how it will respond to it.

In backend testing, we utilize Python’s Unittest framework as it is not only intuitive but easy to understand and use. We apply different approaches such as random testing, boundary testing and category partitioning testing in all the test cases and test suites.

Each test case and test suite will be logged as an entry in the table. Any failed test cases (i.e., actual output does not match expected output) will be investigated and re-conducted after modifying the code.

# 4. Unit Testing

Our program makes use of several machine learning models that are split into base models and ensemble models. Unit testing is performed on these models to ensure the accuracy score passes a certain threshold level as well as ensuring it rejects invalid parameters. In addition to that, evaluation metrics which are metrics to evaluate the performance of the models are also tested to check the score of the models. Unit testing is conducted as Whitebox Testing, where the source of each function is known and the behavior of each function is to be tested. Unit testing also helps in identifying if the functionality of the program is implemented correctly and producing the correct results for each given input.

## 4.1 Base Models

### 4.1.1 Complement Naïve Bayes

**Tester Name:** Ethan Hor Sheng Jian

**Component:** Complement Naïve Bayes

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** complement\_naive\_bayes.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| CNB.1 | Type test for when the model has been trained with fit() | Create a Complement Naïve Bayes model, and then fit it using dummy data. Then, its type is checked against the base MLP model type | Both the fitted model and the base model are the same type | Both the fitted model and the base model are the same type | Pass |
| CNB.2 | Empty data test for the algorithm that creates the model | Pass in an empty list into the function that is responsible for creating the MLP model | IndexError occurs | IndexError occurs | Pass |
| CNB.3 | Testing the functionality of the fit() function of the prediction model | Create and fit a Complement Naïve Bayes model with dummy data. Then, attempt to use the model to predict data that is of a different size | ValueError occurs | ValueError occurs | Pass |
| CNB.4 | Basic test of the Complement Naïve Bayes Model | Create and fit a Complement Naïve Bayes model with the text file CM1.arff.txt. Then use it to predict the test data from that file. | Accuracy score >= 0.35 | Accuracy score = 0.4545 | Pass |

### 

### 4.1.2 Decision Tree Model

**Tester Name:** Ethan Hor Sheng Jian

**Component:** Decision Tree

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** decision\_tree.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| DT.1 | Type test for when the model has been trained with fit() | Create a Logistic Regression model, and then fit it using dummy data. Then, its type is checked against the base MLP model type | Both the fitted model and the base model are the same type | Both the fitted model and the base model are the same type | Pass |
| DT.2 | Empty data test for the algorithm that creates the model | Pass in an empty list into the function that is responsible for creating the MLP model | IndexError occurs | IndexError occurs | Pass |
| DT.3 | Testing the functionality of the fit() function of the prediction model | Create and fit a Logistic Regression model with dummy data. Then, attempt to use the model to predict data that is of a different size | ValueError occurs | ValueError occurs | Pass |
| DT.4 | Basic test of the Logistic Regression Model | Create and fit a Logistic Regression model with the text file CM1.arff.txt. Then use it to predict the test data from that file. | Accuracy score >= 0.75 | Accuracy score = 0.7879 | Pass |

### 

### 4.1.3 Logistic Regression Model

**Tester Name:** Ethan Hor Sheng Jian

**Component:** Logistic Regression

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** logistic\_regression.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| LR.1 | Type test for when the model has been trained with fit() | Create a Logistic Regression model, and then fit it using dummy data. Then, its type is checked against the base MLP model type | Both the fitted model and the base model are the same type | Both the fitted model and the base model are the same type | Pass |
| LR.2 | Empty data test for the algorithm that creates the model | Pass in an empty list into the function that is responsible for creating the MLP model | IndexError occurs | IndexError occurs | Pass |
| LR.3 | Testing the functionality of the fit() function of the prediction model | Create and fit a Logistic Regression model with dummy data. Then, attempt to use the model to predict data that is of a different size | ValueError occurs | ValueError occurs | Pass |
| LR.4 | Basic test of the Logistic Regression Model | Create and fit a Logistic Regression model with the text file CM1.arff.txt. Then use it to predict the test data from that file. | Accuracy score >= 0.75 | Accuracy score = 0.7879 | Pass |

### 4.1.4 Multi-Layer Perceptron Model

**Tester Name:** Ethan Hor Sheng Jian

**Component:** Multi-Layer Perceptron

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** multi\_layer\_perceptron.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MLP.1 | Type test for when the model has been trained with fit() | Create an MLP model, and then fit it using dummy data. Then, its type is checked against the base MLP model type | Both the fitted model and the base model are the same type | Both the fitted model and the base model are the same type | Pass |
| MLP.2 | Empty data test for the algorithm that creates the model | Pass in an empty list into the function that is responsible for creating the MLP model | IndexError occurs | IndexError occurs | Pass |
| MLP.3 | Testing the functionality of the fit() function of the prediction model | Create and fit an MLP model with dummy data. Then, attempt to use the model to predict data that is of a different size | ValueError occurs | ValueError occurs | Pass |
| MLP.4 | Basic test of the Multi-layer Perceptron Model | Create and fit an MLP model with the text file CM1.arff.txt. Then use it to predict the test data from that file. | Accuracy score >= 0.75 | Accuracy score = 0.8182 | Pass |

### 4.1.5 Naïve Bayes Model

**Tester Name:** Ethan Hor Sheng Jian

**Component:** Multi-Layer Perceptron

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** naïve\_bayes.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| NB.1 | Type test for when the model has been trained with fit() | Create a Naïve Bayes model, and then fit it using dummy data. Then, its type is checked against the base MLP model type | Both the fitted model and the base model are the same type | Both the fitted model and the base model are the same type | Pass |
| NB.2 | Empty data test for the algorithm that creates the model | Pass in an empty list into the function that is responsible for creating the MLP model | IndexError occurs | IndexError occurs | Pass |
| NB.3 | Testing the functionality of the fit() function of the prediction model | Create and fit a Naïve Bayes model with dummy data. Then, attempt to use the model to predict data that is of a different size | ValueError occurs | ValueError occurs | Pass |
| NB.4 | Basic test of the Naïve Bayes Model | Create and fit a Naïve Bayes model with the text file CM1.arff.txt. Then use it to predict the test data from that file. | Accuracy score >= 0.75 | Accuracy score = 0.8788 | Pass |

## 4.2 Ensemble Models

### 4.2.1 Random Forest Model

**Tester Name:** Tah Wen Zhong

**Component:** Random Forest

**Number of Test Suites:** 1

**Number of Test Cases:** 7

**Test file:** random\_forest\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| RA.1 | Check the returned model type | Create a Random Forest model and compare with the returned model  (test1.arff.txt) | Model type are both the same (Random Forest) | Model type are both the same  (Random Forest) | Pass |
| RA.2 | Check fitted model usability with test data | Perform predict on normal test data | Prediction successful | Prediction successful | Pass |
| RA.3 | Check fitted model usability with invalid test data | Perform predict on invalid test data | Prediction fails, error raised | Prediction fails, error raised | Pass |
| RA.4 | Check default number of estimators | Model creation with no additional arguments | Model fit successfully with 108 estimators | Model fit successfully with 108 estimators | Pass |
| RA.5 | Check number of estimators argument correctly passed | Model creation with 64 estimators  (args = [64]) | Model fit successfully with 64 estimators | Model fit successfully with 64 estimators | Pass |
| RA.6 | Check prediction functionality with multiple test data | Model prediction with test data of size 27 | Prediction array contains 27 results | Prediction array contains 27 results | Pass |
| RA.7 | Test with actual dataset | test2.arff.txt  (KC dataset) | Model fits and predicts successfully | Model fits and predicts successfully | Pass |

### 4.2.2 Rotation Forest Model

**Tester Name:** Tah Wen Zhong

**Component:** Rotation Forest

**Number of Test Suites:** 1

**Number of Test Cases:** 5

**Test file:** rotation\_forest\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| RO.1 | Check the returned model type | Create a Random Forest model and compare with the returned model  (test1.arff.txt) | Model type are both the same (Random Forest) | Model type are both the same  (Random Forest) | Pass |
| RO.2 | Check fitted model usability with test data | Perform predict on normal test data | Prediction successful | Prediction successful | Pass |
| RO.3 | Check fitted model usability with invalid test data | Perform predict on invalid test data | Prediction fails, error raised | Prediction fails, error raised | Pass |
| RO.4 | Check prediction functionality with multiple test data | Model prediction with test data of size 27 | Prediction array contains 27 results | Prediction array contains 27 results | Pass |
| RO.5 | Test with actual dataset | test2.arff.txt  (KC dataset) | Model fits and predicts successfully | Failed model construction | Fail |

### 4.2.3 Voting

**Tester Name:** Jason Toh Zhern Wee

**Component:** Voting

**Number of Test Suites:** 1

**Number of Test Cases:** 5

**Test file:** voting\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| VT.1 | To test if the model works on a test set of appropriate size | Using the model to predict the test set | Prediction successful | Prediction Successful | Pass |
| VT.2 | To test if the model works on a test set of inappropriate size | Using the model to predict the test set | An exception occurred  (Insufficient number of features per sample) | An exception occurred  (Insufficient number of features per sample) | Pass |
| VT.3 | To test if the model works on a test size of different dimension size | Using the model to predict the test set | An exception occurred  (Expected 2D array, got 1D instead) | An exception occurred  (Expected 2D array, got 1D instead) | Pass |
| VT.4 | To test if the model works on an empty test set | Using the model to predict the test set | An exception occurred  (Minimum of 1 feature needs to be present) | An exception occurred  (Minimum of 1 feature needs to be present) | Pass |
| VT.5 | To test if the model works on an actual dataset | Using the model to predict the test set | Prediction Successful | Prediction Successful | Pass |

## 4.3 Evaluation Metric

### 4.3.1 AUC ROC

**Tester Name:** Jason Toh Zhern Wee

**Component:** AUC\_ROC

**Number of Test Suites:** 1

**Number of Test Cases:** 5

**Test file:** auc\_roc\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| AUC.1 | Check if the evaluation metric returns a floating value | Perform AUC on a model with appropriate test data | Returns a floating value | Returns a floating value | Pass |
| AUC.2 | Check if the evaluation metric works with test set of different dimensions | Perform AUC on a model and with test set of different dimensions | An exception occurred  (Passing 1D array instead of 2D array) | An exception occurred  (Passing 1D array instead of 2D array) | Pass |
| AUC.3 | Check if the evaluation metric works with insufficient number of samples in test set | Perform AUC on a model with test set | An exception occurred  (Inconsistent number of samples) | An exception occurred  (Inconsistent number of samples) | Pass |
| AUC.4 | Check if the evaluation metric works with one class of sample in test set | Perform AUC on a model with test set | An exception occurred  (One class of sample is present) | An exception occurred  (One class of sample is present) | Pass |
| AUC.5 | Check if the evaluation metric works with empty test set | Perform AUC on a model with empty test set | An exception occurred  (Empty test set) | An exception occurred  (Empty test set) | Pass |

### 4.3.2 F1-Score

**Tester Name:** Jason Toh Zhern Wee

**Component:** F1\_Score

**Number of Test Suites:** 1

**Number of Test Cases:** 6

**Test file:** f1\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| F.1 | Check if the evaluation metric returns a floating value | Perform F1 on a model with appropriate test data | Returns a floating value | Returns a floating value | Pass |
| F.2 | Check if the evaluation metric works with test set of different dimensions | Perform F1 on a model and with test set of different dimensions | An exception occurred  (Passing 1D array instead of 2D array) | An exception occurred  (Passing 1D array instead of 2D array) | Pass |
| F.3 | Check if the evaluation metric works with insufficient number of samples in test set | Perform F1 on a model with test set | An exception occurred  (Inconsistent number of samples) | An exception occurred  (Inconsistent number of samples) | Pass |
| F.4 | Check if the evaluation metric works with one class of sample in test set | Perform F1 on a model with test set | An exception occurred  (One class of sample is present) | An exception occurred  (One class of sample is present) | Pass |
| F.5 | Check if the evaluation metric works with empty test set | Perform F1 on a model with empty test set | An exception occurred  (Empty test set) | An exception occurred  (Empty test set) | Pass |
| F.6 | Check for correctness with the library version of F1 and manual calculation of F1 | Perform F1 from library with manual calculated F1 | Both return the same result | Both return the same result | Pass |

## 4.4 Data Preprocessing

### 4.4.1 CFS

**Tester Name:** Jason Toh Zhern Wee

**Component:** CFS

**Number of Test Suites:** 1

**Number of Test Cases:** 6

**Test file:** cfs\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| CFS1 | Selecting 6 features out of training size of 6 | Using the algorithm to select the features | All 6 features are selected | All 6 features are selected | Pass |
| CFS2 | Selecting 4 features out of training size of 6 | Using the algorithm to select the features | 4 Features are selected | 4 Features are selected | Pass |
| CFS3 | Selecting 0 features out of training size of 6 | Using the algorithm to select the features | No features are selected | No features are selected | Pass |
| CFS4 | Check to see if the algorithm works with negative training size | Using the algorithm to select the features | An exception occurred  (Invalid training size) | An exception occurred  (Invalid training size) | Pass |
| CFS5 | Check to see if the number of features selected is more than the number of features in the sample | Using the algorithm to select the features | An exception occurred  (Number of selections is more than the number of features in the sample) | An exception occurred  (Number of selections is more than the number of features in the sample) | Pass |

### 

### 4.4.2 RFE

**Tester Name:** Jason Toh Zhern Wee

**Component:** RFE

**Number of Test Suites:** 2

**Number of Test Cases:** 5

**Test file:** rfe\_test.py

Test Suite 1

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| RFE1 | Selecting 6 features out of training size of 6 | Using the algorithm to select the features | All 6 features are selected | All 6 features are selected | Pass |
| RFE2 | Selecting 4 features out of training size of 6 | Using the algorithm to select the features | 4 Features are selected | 4 Features are selected | Pass |
| RFE3 | Selecting 0 features out of training size of 6 | Using the algorithm to select the features | No features are selected | An exception occurred  (Minimum of 1 feature needs to be presented) | Fail |
| RFE4 | Check to see if the algorithm works with negative training size | Using the algorithm to select the features | An exception occurred  (Invalid training size) | An exception occurred  (Invalid training size) | Pass |
| RFE5 | Check to see if the number if features to be selected is more than the number of features in the sample | Using the algorithm to select the features | An exception occurred  (Number of selections is more than the number of features in the sample) | Returns an array with all features selected | Fail |

Test Suite 2

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| RFE.3 | Selecting 0 features out of training size of 6 | Using the algorithm to select the features | An exception occurred  (Minimum of 1 feature needs to be presented) | An exception occurred  (Minimum of 1 feature needs to be presented) | Pass |
| RFE.5 | Check to see if the number of features selected is more than the training size | Using the algorithm to select the features | Returns an array with all features selected | Returns an array with all features selected | Pass |

### 

### 4.4.3 IHT

**Tester Name:** Tah Wen Zhong

**Component:** IHT

**Number of Test Suites:** 1

**Number of Test Cases:** 5

**Test file:** iht\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| IHT.1 | Test base functionality of IHT method | Perform IHT with mock data | Result returns successfully, with no errors | Result returns successfully, with no errors | Pass |
| IHT.2 | Check if results output in the correct format | Perform IHT with mock data | Result should contain an array which contains the under-sampled metrics, followed by its labels | Result should contain an array which contains the under-sampled metrics, followed by its labels | Pass |
| IHT.3 | Check if returned metric array contains the same number of columns | Perform IHT with mock data | The matrix array within the result should have an array containing rows with the same size as the initial input metrics array | The matrix array within the result should have an array containing rows with the same size as the initial input metrics array | Pass |
| IHT.4 | Check if invalid fold argument is handled appropriately  (Should not cause an error) | Perform IHT with mock data for  k\_fold = 0 | No error occurs, with the input data being used as the output data | No error occurs, with the input data being used as the output data | Pass |
| IHT.5 | Check if fold argument working correctly | Perform IHT with mock data for  k\_fold = 0 and 2 | Results of IHT for fold 0 and 2 differs | Results of IHT for fold 0 and 2 differs | Pass |

### 4.4.4 K-fold

**Tester Name:** Tah Wen Zhong

**Component:** K-fold

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** k-fold\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| K.1 | Test k-fold working with the default parameters | Perform k-fold with a dataset with 20 data | Returns 10 train-test splits | Returns 10 train-test splits | Pass |
| K.2 | Fold argument (f) correctly adjusts the number of splits | Perform k-fold on a dataset with 21 data, for f= 3,6,9 | * f=3 returns 3 train-test splits * f=6 returns 6 train-test splits   f=9 returns 9 train-test splits | * f=3 returns 3 train-test splits * f=6 returns 6 train-test splits   f=9 returns 9 train-test splits | Pass |
| K.3 | The k-fold should fail when insufficient f is given | Perform k-fold on a dataset with 1 data and f=2 | Error message shown | Error message shown | Pass |
| K.4 | The k-fold can read dataset | Perform k-fold using a dataset file (KC.arff) | Returns 10 train-test splits with appropriate outputs | Returns 10 train-test splits with appropriate outputs | Pass |

### 4.4.5 Normalization

**Tester Name:** Tah Wen Zhong

**Component:** Normalization

**Number of Test Suites:** 1

**Number of Test Cases:** 4

**Test file:** normal\_test.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| N.1 | Test base functionality of normalization method | Perform normalization with mock data | Result returns successfully, with no errors | Result returns successfully, with no errors | Pass |
| N.2 | Check if function removes empty data appropriately | Perform normalization with mock data containing missing values | Function removes all missing values | Function removes all missing values | Pass |
| N.3 | Check if function can distinguish empty data and string values | Perform normalization with mock data containing both string and missing values | Function only removes missing values | Function only removes missing values | Pass |
| N.4 | Check if function run successfully using actual dataset | Perform normalization on test2.arff.txt  (KC dataset) | Result returns successfully, with no errors | Result returns successfully, with no errors | Pass |

# 5. Integration Testing

Now that all the models and data preprocessing functions have been created. The next step is to combine the individual modules and test it as a whole. Integration testing is conducted in Whitebox format where the source code is systematically covered by the test cases. The focus in this integration testing is to cover all possible behaviors exhibited by the main program as well as verifying the output of it.

The main test consists of multiple sections. These sections are as follows

1. Read and process the raw data
2. Perform feature selection using CFS or RFE
3. Preprocess the data using features from CFS or RFE
4. Build the models
5. Perform an evaluation on the model
6. Compile the results and write it to a csv file

## 5.1 Main Test

**Tester Name:** Tah Wen Zhong

**Component:** Main Program

**Number of Test Suites:** 5

**Number of Test Cases:** 32

**Test file:** main\_program\_test.py

Test Suite 1: Read and Process Data (First run: Fail)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.1.1 | Check if the data is extracted correctly from read function | test1.arff.txt | Data is extracted correctly  (An array containing values for each row) | Data is extracted correctly  (An array containing values for each row) | Pass |
| MP.1.2 | Test process which separates the labels and metrics | test1.arff.txt | Correctly identify the metrics and labels | Failed to extract the labels | Fail |
| MP.1.3 | Check correctness of label conversion function | test1.arff.txt | Labels converted from Boolean/string to integer (0,1) | - | - |
| MP.1.4 | Test run with actual dataset | test2.arff.txt  (KC dataset) | Extracted data with labels processed correctly | - | - |
| Bug description | | | The data extraction function extracts a data column with “Defective” as the column name. | | |
| Bug fix | | | Rework data extraction function to take the last column as label | | |

Test Suite 1: Read and Process Data (Second Run: Pass)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.1.2 | Test process which separates the labels and metrics | test1.arff.txt | Correctly identify the metrics and labels | Correctly identify the metrics and labels | Pass |
| MP.1.3 | Check correctness of label conversion function | test1.arff.txt | Labels converted from Boolean/string to integer (0,1) | Labels converted from Boolean/string to integer (0,1) | Pass |
| MP.1.4 | Test run with actual dataset | test2.arff.txt  (KC dataset) | Extracted data with labels processed correctly | Extracted data with labels processed correctly | Pass |

Test Suite 2: Feature Selection + Pre-processing (First Run: Fail)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.2.1 | Test the base functionality of the function | test3.arff.txt  (After processing) | Output successful, with no errors | Failed to output, error raised | Fail |
| MP.2.2 | Check if a single feature selection is correctly identified | Selection set to only CFS | Output contains result using CFS algorithm | - | - |
| MP.2.3 | Check if multiple feature selections are correctly identified | Selection set to CFS and RFE | Output contains result using CFS and RFE algorithm | - | - |
| MP.2.4 | Check k-fold argument correctly passed to pre-processing algorithm | test3.arff.txt  (k\_fold = 4) | Returns 4 train-test splits for all output (All, CFS, RFE) | - | - |
| MP.2.5 | Check train size argument correctly passed to feature selection algorithm | test3.arff.txt  (train\_size = 3) | Feature selection (CFS, RFE) outputs contain 3 columns (3 Software metrics) | - | - |
| MP.2.6 | Check if function handles errors appropriately  (k\_fold) | test3.arff.txt  (k\_fold = 10000) | No error occurs and return a tuple containing (False, 1) | - | Fail |
| MP.2.7 | Check if function handles errors appropriately  (train\_size) | test3.arff.txt  (train\_size = 10000) | No error occurs and return a tuple containing (False, 2) | - | Fail |
| MP.2.8 | Test run, including previous steps, with actual dataset | test2.arff.txt  (KC dataset) | Output successful, with no errors | Output successful, with no errors | Pass |
| Bug description | | | * The bug was found within the argument passed to the pre-processing algorithm * The failed runs are because the fold written in the argument did not overwrite the default folds * Test case 2.6 was successful because it uses actual dataset, so the data size is larger than the default folds | | |
| Bug fix | | | The function was fixed so the folds argument was correctly passed to the pre-processing algorithm | | |

Test Suite 2: Feature Selection + Pre-processing (Second run: Pass)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.2.1 | Test the base functionality of the function | test3.arff.txt  (After processing) | Output successful, with no errors | Failed to output, error raised | Fail |
| MP.2.2 | Check if a single feature selection is correctly identified | Selection set to only CFS | Output contains result using CFS algorithm | - | - |
| MP.2.3 | Check if multiple feature selections are correctly identified | Selection set to CFS and RFE | Output contains result using CFS and RFE algorithm | - | - |
| MP.2.4 | Check k-fold argument correctly passed to pre-processing algorithm | test3.arff.txt  (k\_fold = 4) | Returns 4 train-test splits for all output (All, CFS, RFE) | - | - |
| MP.2.5 | Check train size argument correctly passed to feature selection algorithm | test3.arff.txt  (train\_size = 3) | Feature selection (CFS, RFE) outputs contain 3 columns (3 Software metrics) | - | - |
| MP.2.6 | Check if function handles errors appropriately  (k\_fold) | test3.arff.txt  (k\_fold = 10000) | No error occurs and return a tuple containing (False, 1) | No error occurs and return a tuple containing (False, 1) | Pass |
| MP.2.7 | Check if function handles errors appropriately  (train\_size) | test3.arff.txt  (train\_size = 10000) | No error occurs and return a tuple containing (False, 2) | No error occurs and return a tuple containing (False, 2) | Pass |
| MP.2.8 | Test run, including previous steps, with actual dataset | test2.arff.txt  (KC dataset) | Output successful, with no errors | Output successful, with no errors | Pass |

Test Suite 3: Model Creation

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.3.1 | Test whether all models can fit processed data (All) | test3.arff.txt  (After processing) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.3.2 | Test whether all models can fit processed data after CFS selection | test3.arff.txt  (After processing and CFS) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.3.3 | Test whether all models can fit processed data after RFE selection | test3.arff.txt  (After processing and RFE) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.3.4 | Test base models selection argument | test3.arff.txt  (base\_pred = [0,3]) | Result contains complement naïve bayes and MLP models | Result contains complement naïve bayes and MLP models | Pass |
| MP.3.5 | Test ensemble models selection argument | test3.arff.txt  (ensemble\_preds = [0]) | Result contains random forest model | Result contains random forest model | Pass |
| MP.3.6 | Test whether all models contain the predict() functionality | test3.arff.txt  (All models selected) | All models contain the predict() function and can perform predictions | All models contain the predict() function and can perform predictions | Pass |

Test Suite 4: Evaluation

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.4.1 | Test whether model built can be evaluated | test3.arff.txt  (Model used: Decision tree) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.4.2 | Test whether evaluation result contains a list of 4 items | test3.arff.txt | Output array contains 4 items | Output array contains 4 items | Pass |
| MP.4.3 | Check if the results are all appropriate values | test3.arff.txt | Output array contains values within the range 0 to 1 | Output array contains values within the range 0 to 1 | Pass |
| MP.4.4 | Check whether the evaluation result follows the correct order | test3.arff.txt | Output array follows this order:  AUC, F1-score, FPR, FNR | Output array follows this order:  AUC, F1-score, FPR, FNR | Pass |

Test Suite 5: CSV Writer + Main Function (First run: Fail)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.5.1 | Test the main function with all arguments set to max | test3.arff.txt  (All models selected) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.5.2 | Test the main function with all arguments set to minimum | test3.arff.txt  k-fold = 2  train = 1  (1 model selected) | Output successful, with no errors | - | Fail |
| MP.5.3 | Check if the base model selection argument functioning correctly | test3.arff.txt  (Selected CNB model) | Output only contains evaluation results for the Complement Naïve Bayes model | - | - |
| MP.5.4 | Check if the ensemble model selection argument functioning correctly | test3.arff.txt  (Selected random forest model) | Output only contains evaluation results for the Random Forest model | - | - |
| MP.5.5 | Check if the main function can handle string input instead of integers | test3.arff.txt  (k-fold and train argument values are String type) | Output successful, with no errors | - | - |
| MP.5.6 | Check if the main program passes the feature selection argument correctly | test3.arff.txt  (All and RFE selected) | Evaluation result contains both data that contains all metrics and metrics after reduction from RFE | - | - |
| MP.5.7 | Check if the main program handles invalid train size input appropriately | test3.arff.txt  (train\_size = 1000) | The function halts with no error and returns a tuple containing (False, 2) | - | - |
| MP.5.8 | Check if the main program handles invalid k-fold input appropriately | test3.arff.txt  (k\_fold = 1000) | The function halts with no error and returns a tuple containing (False, 1) | - | - |
| MP.5.9 | Check if the main program handles invalid datasets appropriately | fail1.arff.txt  (An invalid dataset) | The function halts with no error and returns a tuple containing (False, 0) | - | - |
| MP.5.10 | Check if the main program correctly produces the CSV file | test3.arff.txt  k-fold = 2  train = 1  (3 model selected) | CSV file containing the correct values and format | - | - |
| Bug description | | | * The code contains two variables that represent the k-fold value, one of which is not updated based on the argument | | |
| Bug fix | | | Reduce the number of variables representing the k-fold value to one | | |

Test Suite 5: CSV Writer + Main Function (Second run: Fail)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| MP.5.1 | Test the main function with all arguments set to max | test3.arff.txt  (All models selected) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.5.2 | Test the main function with all arguments set to minimum | test3.arff.txt  k-fold = 2  train = 1  (1 model selected) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.5.3 | Check if the base model selection argument functioning correctly | test3.arff.txt  (Selected CNB model) | Output only contains evaluation results for the Complement Naïve Bayes model | Output only contains evaluation results for the Complement Naïve Bayes model | Pass |
| MP.5.4 | Check if the ensemble model selection argument functioning correctly | test3.arff.txt  (Selected random forest model) | Output only contains evaluation results for the Random Forest model | Output only contains evaluation results for the Random Forest model | Pass |
| MP.5.5 | Check if the main function can handle string input instead of integers | test3.arff.txt  (k-fold and train argument values are String type) | Output successful, with no errors | Output successful, with no errors | Pass |
| MP.5.6 | Check if the main program passes the feature selection argument correctly | test3.arff.txt  (All and RFE selected) | Evaluation result contains both data that contains all metrics and metrics after reduction from RFE | Evaluation result contains both data that contains all metrics and metrics after reduction from RFE | Pass |
| MP.5.7 | Check if the main program handles invalid train size input appropriately | test3.arff.txt  (train\_size = 1000) | The function halts with no error and returns a tuple containing (False, 2) | The function halts with no error and returns a tuple containing (False, 2) | Pass |
| MP.5.8 | Check if the main program handles invalid k-fold input appropriately | test3.arff.txt  (k\_fold = 1000) | The function halts with no error and returns a tuple containing (False, 1) | The function halts with no error and returns a tuple containing (False, 1) | Pass |
| MP.5.9 | Check if the main program handles invalid datasets appropriately | fail1.arff.txt  (An invalid dataset) | The function halts with no error and returns a tuple containing (False, 0) | The function halts with no error and returns a tuple containing (False, 0) | Pass |
| MP.5.10 | Check if the main program correctly produces the CSV file | test3.arff.txt  k-fold = 2  train = 1  (3 model selected) | CSV file containing the correct values and format | CSV file containing the correct values and format  (Screenshot 3) | Pass |

## 5.2 Preprocess

Since the preprocessing consists of individual smaller data preprocessing functions, it will be tested under integration testing where all combined functions will operate as one and tested. The following steps are as follows.

1. Normalize the data
2. Perform Stratified K fold to get the training and testing data
3. Perform IHT under sampling on the training and testing data

**Tester Name:** Jason Toh Zhern Wee

**Component:** Preprocessing

**Number of Test Suites:** 1

**Number of Test Cases:** 5

**Test file:** preprocess.py

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| P.1 | Perform preprocessing with different value of folds | Using f = 3,4,5 as an argument to test preprocessing | * f=3 returns 3 train-test splits * f=4 returns 4 train-test splits   f=5 returns 5 train-test splits | * f=3 returns 3 train-test splits * f=4 returns 4 train-test splits   f=5 returns 5 train-test splits | Pass |
| P.2 | Perform preprocessing with CFS Feature Selection | Pass in CFS as an argument to preprocessing | Returns the preprocessed data sets using features from CFS | Returns the preprocessed data sets using features from CFS | Pass |
| P.3 | Perform preprocessing with RFE Feature Selection | Pass in RFE as an argument to preprocessing | Returns the preprocessed data sets using features from RFE | Returns the preprocessed data sets using features from RFE | Pass |
| P.4 | Check to see if preprocessing works when the number of folds is larger than the number of members in each class | Using a large k fold such as 20 | An exception occurred | An exception occurred | Pass |
| P.5 | Checks to see if preprocessing works when the number of folds is larger than the number of features present | Using a large k fold such as 20 along with CFS | An exception occurred | An exception occurred | Pass |

# 6. Graphical User Interface Testing

Our program uses a GUI to allow users to interact with the system in a user-friendly way. It is implemented in tkinter which is Python’s GUI library. All the tests here are conducted in black-box format where users are only able to test the functionality without knowing much of the internal structure/design of the item that is being tested on. All attempts and input values are logged in the table and are compared with the output values for verification.

**Tester Name:** Ethan Hor Sheng Jian

**Component:** User Interface

**Number of Test Suites:** 1

**Number of Test Cases:** 20

**Test file:** main.py

Test Suite 1 (First run: Fail)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| UI.1 | Logging in without entering information | Pressing the Login button without entering user credentials | Error message appears | Error message appears | Pass |
| UI.2 | Logging in by simply entering information | Pressing the Login button after entering random credentials for both username and password | Error message appears | Error message appears | Pass |
| UI.3 | Logging in by entering the correct credentials | Pressing the Login button after entering the correct username and password | Login is successful. UI transitions to Home screen. | Login is successful. UI transitions to Home screen. | Pass |
| UI.4 | Exit button | Pressing the exit button wherever it is present | Program terminates itself  after exit button is pressed | Program terminates itself  after exit button is pressed | Pass |
| UI.5 | Logout button | Pressing the logout button | UI transitions back to the Login screen | UI transitions back to the Login screen | Pass |
| UI.6 | Starting the main algorithm with appropriate inputs | Pressing the start button after selecting at least one base/ensemble predictor and uploading one file | UI transitions to the feature selection screen | UI transitions to the feature selection screen | Pass |
| UI.7 | Starting the main algorithm without any inputs | Pressing the start button without selecting anything | Error message appears | UI transitions to the feature selection screen | Fail |
| UI.8 | Starting the main algorithm with appropriate inputs | Pressing the start button after selecting at least one base/ensemble predictor and uploading one file | UI transitions to the feature selection screen | UI transitions to the feature selection screen | Pass |
| UI.9 | Reports Tab | Pressing the reports tab button | UI transitions to the Reports Tab | UI transitions to the Reports Tab | Pass |
| UI.10 | Main Algorithm Tab | Pressing the Main Algorithm tab button | UI transitions to the Main Algorithm Tab | UI transitions to the Main Algorithm Tab | Pass |
| UI.11 | About Us Tab | Pressing the About Us tab button | UI transitions to the About Us Tab | UI transitions to the About Us Tab | Pass |

Test Suite 1 (Second run: Fail)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| UI.1 | Logging in without entering information | Pressing the Login button without entering user credentials | Error message appears | Error message appears | Pass |
| UI.2 | Logging in by simply entering information | Pressing the Login button after entering random credentials for both username and password | Error message appears | Error message appears | Pass |
| UI.3 | Logging in by entering the correct credentials | Pressing the Login button after entering the correct username and password | Login is successful. UI transitions to Home screen. | Login is successful. UI transitions to Home screen. | Pass |
| UI.4 | Exit button | Pressing the exit button wherever it is present | Program terminates itself  after exit button is pressed | Program terminates itself  after exit button is pressed | Pass |
| UI.5 | Logout button | Pressing the logout button | UI transitions back to the Login screen | UI transitions back to the Login screen | Pass |
| UI.6 | Upload button | Pressing the upload button | A window pops up, letting the user upload the files they want | A window pops up, letting the user upload the files they want | Pass |
| UI.7 | Remove Selected Files button | Pressing the Remove Selected Files button | Selected uploaded files are cleared from the screen | Nothing happens | Fail |
| UI.8 | Starting the main algorithm with appropriate inputs | Pressing the start button after selecting at least one base/ensemble predictor and uploading one file | UI transitions to the feature selection screen | UI transitions to the feature selection screen | Pass |
| UI.9 | Starting the main algorithm without any inputs | Pressing the start button without selecting anything | Error message appears | Error message appears | Pass |
| UI.10 | Starting the main algorithm with all inputs excluding the selection of predictors | Upload datasets and input values for K-fold and feature selection reduction, input a prediction file name, then press the start button | Error message appears, telling user to select predictors | Error message appears, telling user to select predictors | Pass |
| UI.11 | Starting the main algorithm with all inputs excluding the input of feature selection reduction | Upload datasets, input a prediction file name, input values for K-fold and select predictors, then press the start button | Error message appears, telling user to input a value for the feature selection reduction field | Error message appears, telling user to input a value for the feature selection reduction field | Pass |
| UI.12 | Starting the main algorithm with all inputs excluding the input of k-fold | Upload datasets, input values for feature selection reduction and select predictors, input a prediction file name, then press the start button | Error message appears, telling user to input a value for the k-fold field | Error message appears, telling user to input a value for the k-fold field | Pass |
| UI.13 | Starting the main algorithm with all inputs, but without uploading any datasets | Input values for k-fold and feature selection reduction, input a prediction file name, select predictors, then press the start button | Error message appears, telling user to upload at least one dataset | Error message appears, telling user to upload at least one dataset | Pass |
| UI.14 | Starting the main algorithm with all inputs, but upload a file with an incompatible file format | Input values for k-fold and feature selection reduction, input a prediction file name, select predictors, upload a .pdf file, then press the start button | Error message appears, telling user to upload a valid file | UI transitions to the feature selection screen | Fail |
| UI.15 | Starting the main algorithm with all inputs, but leave the field for the CSV file name empty | Input values for k-fold and feature selection reduction, select predictors, upload valid csv files, then press the start button | Error message appears, telling user to name the CSV file | UI transitions to the feature selection screen | Fail |
| UI.16 | Running the algorithm without any feature selection algorithm selected | After inputting everything valid and reaching the Feature selection menu screen, press the run button without selecting anything | Error message appears, informing the user to select at least one feature selection algorithm | Error message appears, informing the user to select at least one feature selection algorithm | Pass |
| UI.17 | Reports Tab | Pressing the reports tab button | UI transitions to the Reports Tab | UI transitions to the Reports Tab | Pass |
| UI.18 | Main Algorithm Tab | Pressing the Main Algorithm tab button | UI transitions to the Main Algorithm Tab | UI transitions to the Main Algorithm Tab | Pass |
| UI.19 | About Us Tab | Pressing the About Us tab button | UI transitions to the About Us Tab | UI transitions to the About Us Tab | Pass |

Test Suite 1 (Third run: Pass)

| Test case ID | Test case description | Test data/setup | Expected Result | Actual Result | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| UI.1 | Logging in without entering information | Pressing the Login button without entering user credentials | Error message appears | Error message appears | Pass |
| UI.2 | Logging in by simply entering information | Pressing the Login button after entering random credentials for both username and password | Error message appears | Error message appears | Pass |
| UI.3 | Logging in by entering the correct credentials | Pressing the Login button after entering the correct username and password | Login is successful. UI transitions to the Home screen. | Login is successful. UI transitions to the Home screen. | Pass |
| UI.4 | Exit button | Pressing the exit button wherever it is present | Program terminates itself  after exit button is pressed | Program terminates itself  after exit button is pressed | Pass |
| UI.5 | Logout button | Pressing the logout button | UI transitions back to the Login screen | UI transitions back to the Login screen | Pass |
| UI.6 | Upload button | Pressing the upload button | A window pops up, letting the user upload the files they want | A window pops up, letting the user upload the files they want | Pass |
| UI.7 | Remove Selected Files button | Pressing the Remove Selected Files button | Selected uploaded files are cleared from the screen | Selected uploaded files are cleared from the screen | Pass |
| UI.8 | Starting the main algorithm with appropriate inputs | Pressing the start button after selecting at least one base/ensemble predictor and uploading one file | UI transitions to the feature selection screen | UI transitions to the feature selection screen | Pass |
| UI.9 | Starting the main algorithm without any inputs | Pressing the start button without selecting anything | Error message appears | Error message appears | Pass |
| UI.10 | Starting the main algorithm with all inputs excluding the selection of predictors | Upload datasets and input values for K-fold and feature selection reduction, then press the start button | Error message appears, telling user to select predictors | Error message appears, telling user to select predictors | Pass |
| UI.11 | Starting the main algorithm with all inputs excluding the input of feature selection reduction | Upload datasets, input values for K-fold and select predictors, then press the start button | Error message appears, telling user to input a value for the feature selection reduction field | Error message appears, telling user to input a value for the feature selection reduction field | Pass |
| UI.12 | Starting the main algorithm with all inputs excluding the input of k-fold | Upload datasets, input values for feature selection reduction and select predictors, then press the start button | Error message appears, telling user to input a value for the k-fold field | Error message appears, telling user to input a value for the k-fold field | Pass |
| UI.13 | Starting the main algorithm with all inputs, but without uploading any datasets | Input values for k-fold and feature selection reduction, select predictors, then press the start button | Error message appears, telling user to upload at least one dataset | Error message appears, telling user to upload at least one dataset | Pass |
| UI.14 | Starting the main algorithm with all inputs, but upload a file with an incompatible file format | Input values for k-fold and feature selection reduction, select predictors, upload a .pdf file, then press the start button | Error message appears, telling user to upload a valid file | After reaching the feature selection UI, running the program will cause an error message to appear, telling user to upload a valid file | Pass |
| UI.15 | Starting the main algorithm with all inputs, but leave the field for the CSV file name empty | Input values for k-fold and feature selection reduction, select predictors, upload valid csv files, then press the start button | Error message appears, telling user to name the CSV file | UI transitions to the feature selection screen | Pass |
| UI.16 | Running the algorithm without any feature selection algorithm selected | After inputting everything valid and reaching the Feature selection menu screen, press the run button without selecting anything | Error message appears, informing the user to select at least one feature selection algorithm | Error message appears, informing the user to select at least one feature selection algorithm | Pass |
| UI.17 | Reports Tab | Pressing the reports tab button | UI transitions to the Reports Tab | UI transitions to the Reports Tab | Pass |
| UI.18 | Main Algorithm Tab | Pressing the Main Algorithm tab button | UI transitions to the Main Algorithm Tab | UI transitions to the Main Algorithm Tab | Pass |
| UI.19 | About Us Tab | Pressing the About Us tab button | UI transitions to the About Us Tab | UI transitions to the About Us Tab | Pass |
| UI.20 | Instructions Tab | Pressing the Instructions tab button | UI transitions to the Instructions Tab | UI transitions to the Instructions Tab | Pass |

# 7. System Testing

In system testing, we are validating whether the program we built meets the documented requirements or not. This will be performed as Blackbox Testing where we will provide a set of test inputs for the program and the results it generates will crosscheck with the requirements. The following section will be divided into the following subsections:

* Algorithms
* Preprocessing
* Main program
* User interface
* Output

## 7.1 Algorithms

For base and ensemble predictors, all models were able to work on different datasets with different software metrics and labels. These models were also able to display the accuracy score for each dataset. Our program consists of 2 evaluation metrics which are AUC ROC and F1 score. The scores from these evaluation metrics were obtained using imported libraries functions and were checked using manual calculation for verification.

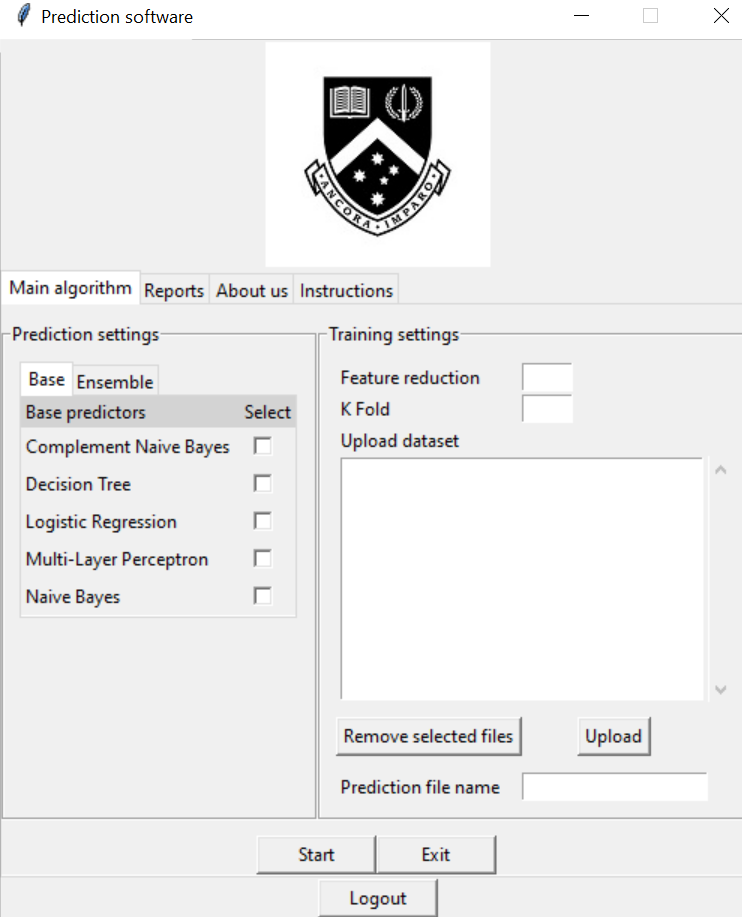
## 7.2 Preprocessing

The preprocessing phase will consist of 3 sections which will turn raw datasets into more useful data to be fitted into the models. The first step is Normalization which is the process of removing missing values and outliers that are present from the dataset. As we are dealing with an imbalanced dataset, Stratified K Fold is used as it is designed specifically for maintaining the same class ratio as the original dataset. The K fold will split the set into 70% training set and 30% testing set. Furthermore, the test sets will be under sample further using IHT to ensure the class samples are of the right proportions.

## 7.3 Main program

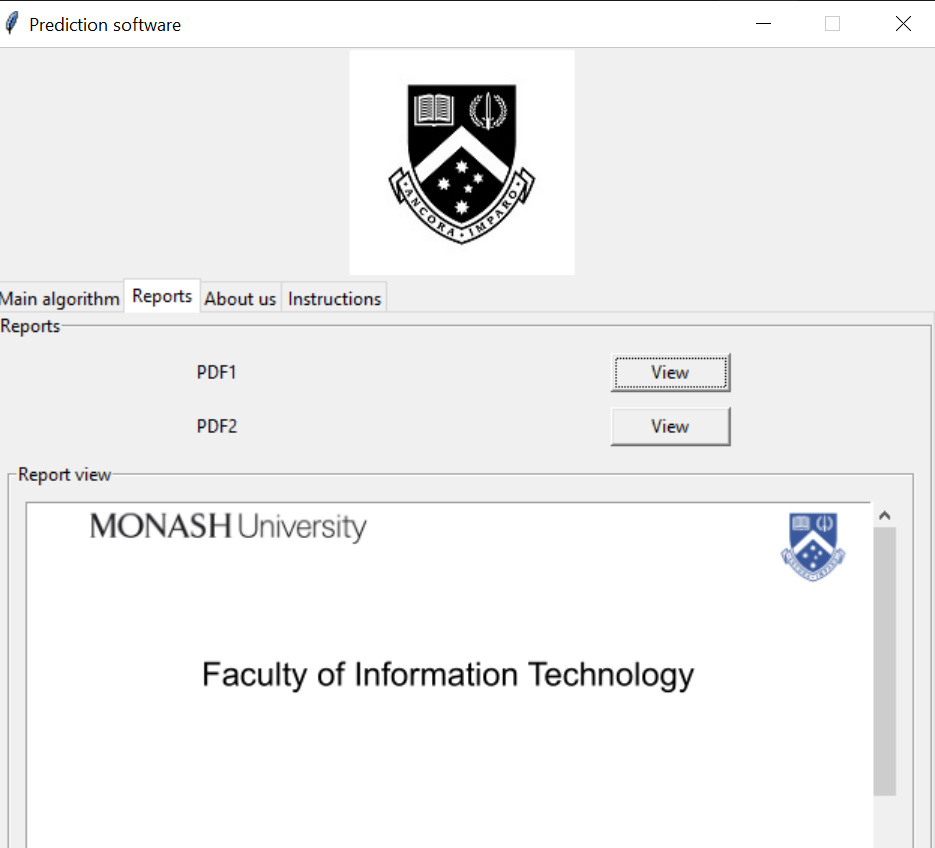
The main program is considered as the backbone of the entire program. Its primary role is connecting all the core components of the program. Using passed-in arguments, the main program is able to use those values to build the models, perform preprocessing and feature selection as well as evaluating the scores of each model. Finally, the scores will be saved as results and written into a csv file.

## 7.4 User interface



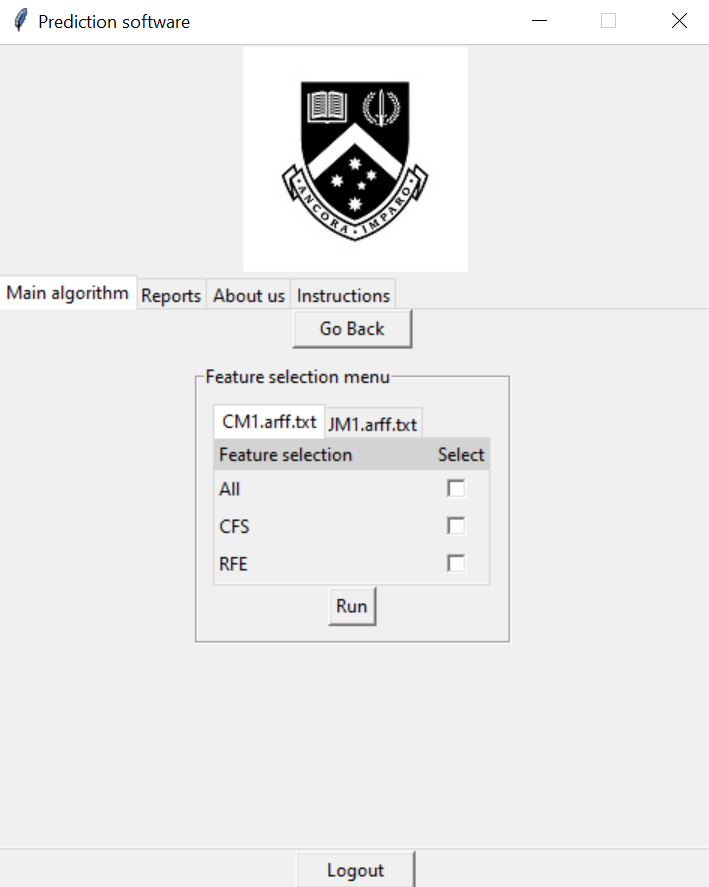
*Figure 1: Home Screen Page*

In the user interface screen, we have a multi-select screen for the base and ensemble predictors and these models will be built in the main algorithm. The feature reduction and K fold are input text fields and allows users to enter a number within a fixed range and will produce an error message if otherwise. Users are able to upload multiple datasets as well as removing any selected datasets. Furthermore, users are able to specify the file name in the prediction file name section.



*Figure 2: Report Section Page*

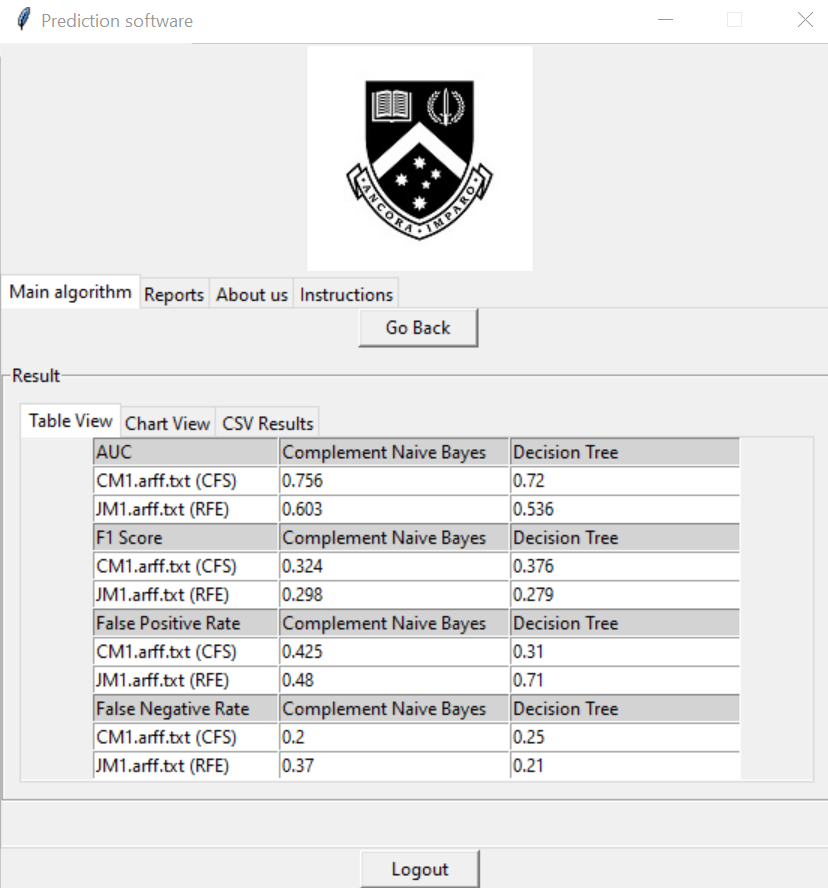
Clicking on the report view button will generate a viewable version of the PDF without having the need to download it.



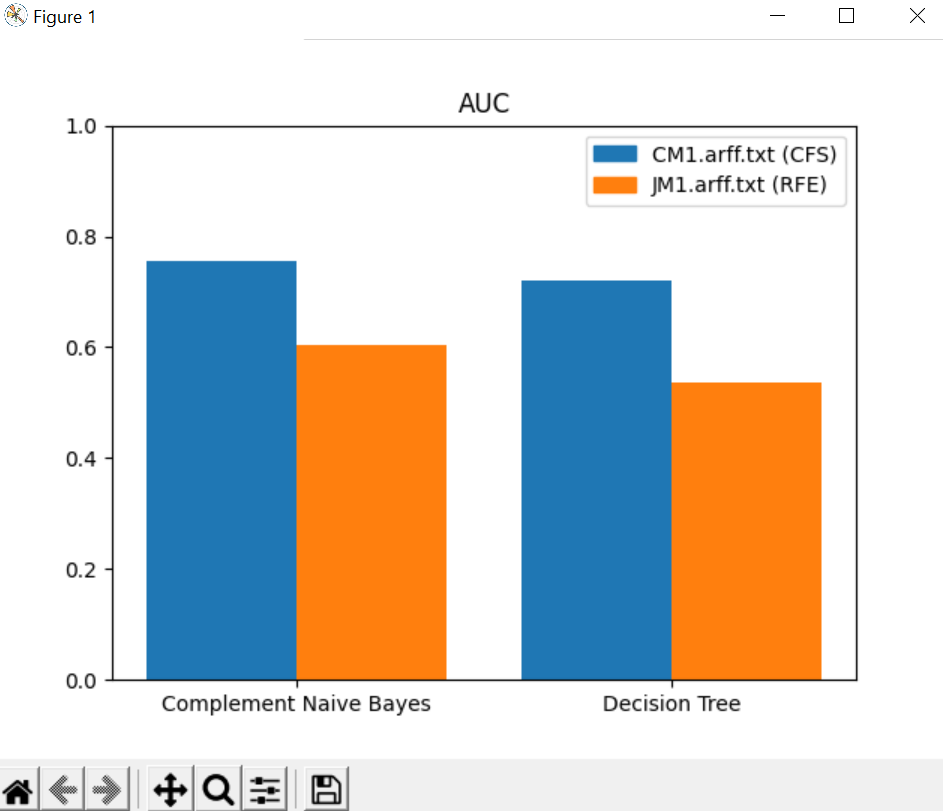
*Figure 3: Feature Selection Menu*

In the feature selection menu, users can choose to select their preferred feature selection algorithm for each dataset uploaded. Multi-select is also implemented to allow users to choose more than one option. However, at least one option needs to be ticked for each dataset in order to run the main program.

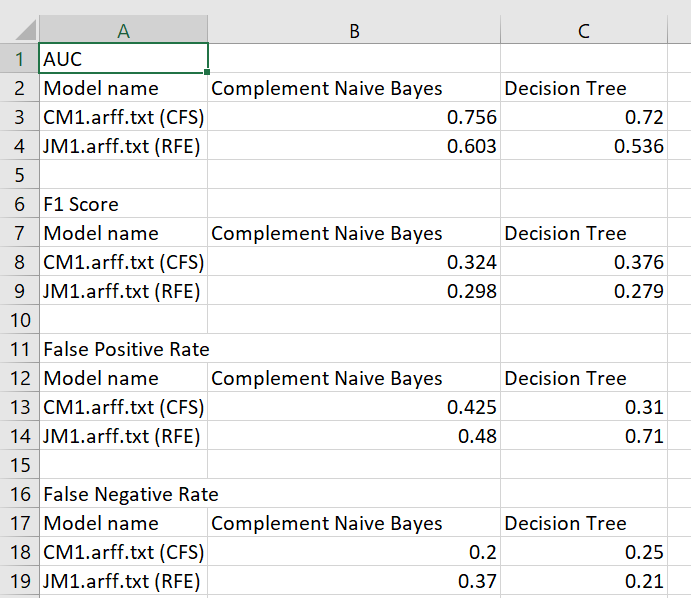
7.5 Output



*Figure 4: Table View*



*Figure 5: Chart View*



*Figure 6: CSV View*

After computation, the program stores the result into a table and displays it in 3 forms. The first being a tabulated view as shown in Figure 4. Tabulated view allows users to see the precise score of each model. Figure 5 is the Chart View where visualizations are created to allow users to see the pattern and make comparison much more easily. Lastly, all results are stored inside the csv file where the file name was specified by the user in the home screen page. The purpose was to save computed results of previous attempts so that it can be used for future reference.

# 

# 8. Usability Testing

For usability testing, we have gathered participants to try out our application and provide any feedback about the program. To provide an overview, our program is a GUI application so our tests mainly revolve around users providing feedback regarding interface design, user friendlies of the application as well as any sections that require improvements.

Below is a summary of the feedback received from the users

1. 100% of the participants said our application is **user-friendly**
2. 100% of the participants said that our application is **easy to use**
3. 57.1% of the participants felt that the application **does not guide the users step by step** while 42.9% think otherwise.
4. 71.4% of the participants agreed that the **charts help in visualizing the charts** while 28.6% did not agree.
5. 100% of the participants said that the **error messages were informative**

The last question was more open-ended as it was about suggestions for improvements.

Here are a few examples of the responses

1. Chart sections should have different colours to visualise the different features easier
2. A loading screen should be added when the application is running
3. Create a dropdown selection for both k-fold and feature reduction
4. Enlarge font size in user interface
5. You can include some setting for basic algorithms such as tree depth

Based on the feedback received from the user, we are surprised to find out all the users found our application user friendly. There was one particular question that caught our attention was that the application did not guide the users step by step. We understand the home screen contains a lot of information which is overwhelming and therefore we have prepared a comprehensive user guide that walkthroughs the program one step at a time.

Besides that, some users reported that loading screens and settings for tree depth to be specified for basic algorithms should be included. Further details regarding these issues will be explained in the software limitations section.

# 9. Performance, Scalability, Security

## 9.1 Performance

To improve the efficiency in our program, we managed to optimize the program by only building the models defined by the user instead of building all the models available which saves computation time. Another time saving trick was creating a lookup index by remembering the position of the model stored in the table. Without creating the lookup index, we would have to search through the table to find the position of the model which wastes computation time.

## 9.2 Scalability

Our program does not scale well with larger datasets due to the number of lines in the file it contains. Given that every line needs to be read, computation time will be longer as the files get larger. However, our program is able to integrate with larger-scale projects as the program is modularized. Therefore, the functions can be easily extensible as well as reuse in other parts of the program.

## 9.3 Security

Our program does not store any information regarding the user so any leakage of confidential information is not possible. Also, any user inputs are sanitized and enforced with strict type checking to prevent users from typing malicious script commands into it. When uploading datasets, only two file formats can be accepted and those are csv file and arff file. This ensures no scripting files which may contain malicious code can harm our program in any way.

# 

# 10. Limitation of the software

One limitation of our program is that it does not handle well when the size of the file is extremely large. Because our program needs to read every line of the file, we could not optimize the performance of the program when the size of the file gets larger. Hence, the program tends to slow down drastically if the size of the dataset were to increase.

Another limitation we have is that we could not implement a loading bar screen when running the main program to view progress. Since we are using imported libraries functions, we are not able to discern the underlying workings on how the model is built. Not only that, we also have to gather the progress from various functions and summarize it into one screen. Hence, we deemed it was infeasible to implement as we lack certain knowledge and skills in that area.

Lastly, our voting model is an ensemble model that uses the results of 3 models to make predictions. These models are Logistic Regression, Random Forest and Naïve Bayes. Currently, there is no option to specify other types of models to be used other than the 3 specified previously.

# 11. Recommendation for improvements

Our program is not perfect and certainly there are improvements to be made to not only make the user experience better overall application itself.

Firstly, our application is a software application that needs to be downloaded on a local computer in order to operate. It is considered inconvenient for public users to use due to the need to download multiple external Python packages listed in the Technical Guide beside the main application. A suitable replacement would either be a web or mobile application as it is more accessible. A web application is intuitive as layout of the page is much more customizable compared to a standard software application and this attracts user attention. Furthermore, the need to download is completely eliminated as the user just needs to click on the URL link to access the web page.

Besides that, our program currently has no option to configure hyperparameters of models (for example, the depth of the decision tree) or to include or exclude IHT under sampling. This limits the potential discovery of a better fault prediction model as possibilities to experiment with different configurations are limited given the current capabilities of our software.

# 12. Test Limitation

When carrying out Blackbox and Whitebox testing, we found numerous issues when testing our program. As our software application relies on the machine learning libraries to create and build the predictive models. Our team is unable to discern the underlying code which builds the model and this limits our capability to test the functionality as well as the correctness of the model. Although we are unable to test the inner workings of the model, we are however able to verify correctness by performing manual calculation and comparing our results with the model’s results.

Another limitation we found through testing is the uncertainty of whether a package may be discontinued or incompatible with our program in the future. The package may receive new updates that might not do what it used to any longer. As our program relies on extensive use of Python’s packages, a change in their syntax may cause numerous errors or unexpected logic errors in the program.

When performing usability testing, we were only able to gather a limited number of participants for our program due to time constraints as well as having limited connections and friends. Another reason could be due to the inconvenience of porting over our application to their local computers as Python and some external packages need to be downloaded before it could be run. Also, participants of our program generally need to be briefed about the overview of our program as it is meant for a niche set of individuals who have keen interest in software testing. However, we were able to gather useful feedback and incorporate those changes into our program.

# 

# 13. Conclusion

In conclusion, we are satisfied with the overall test results as the program was able to meet all the core functionalities and requirements as well as passing all the test suites and test cases. Through testing, we have identified numerous unforeseen bugs in the program that could occur anytime during the later stages of the project. Blackbox testing and Whitebox testing have proven to be an effective testing strategy as it helps to test each part of the program extensively and in detail and ensure the requirements are satisfied. User acceptance testing was a big contributor to the project as the users provided constructive criticism which allows our team to improve on parts that focus on better usability.

All in all, testing is an integral part in any software development project. Despite the limitation of our software and testing process stated in the earlier sections, we still managed to carry out comprehensive test cases and to ensure our program is robust in all ways. Nevertheless, testing in a large-scale project has provided our team many learning opportunities that will be applicable in our future career.