BBC2, IBC, PBC, WPBC2 & WPIBC

(Code, Visualisation and Analysis)

Data Sets and Training

The Dataset is based on CK metrics of Object Oriented System. There are 27 different datasets and the following Results will be based on these datasets.

The size of each dataset varies from 125 to 2000. The No. of attributes are 7 and the target variable is isBug.

The given data illustrates The structure of other datasets

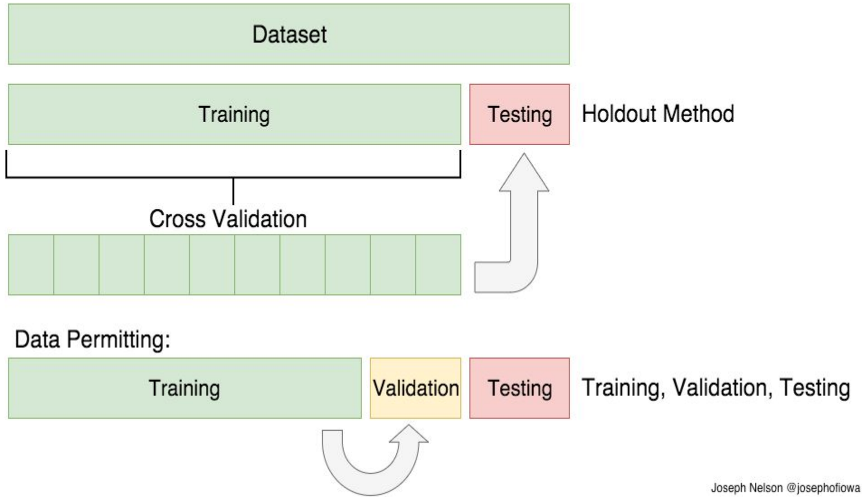
	wmc	dit	rfc	noc	cbo	lcom	loc	isBug
0	34	2	74	0	14	0	2890	NO
1	46	1	61	0	13	833	579	NO
2	4	1	4	0	0	6	4	NO
3	4	1	4	0	0	6	4	NO
4	41	2	75	0	11	280	1225	NO
..	...	...	...	...	...	...	...	...
448	3	1	3	0	0	3	3	NO
449	11	1	18	1	1	15	102	NO
450	8	1	14	0	2	0	144	NO
451	3	3	4	0	0	1	481	YES
452	2	3	4	0	0	1	9	NO

[453 rows x 8 columns]

- PROMIS
  - CK
    - ant-1.3--CK.csv
    - ant-1.4--CK.csv
    - ant-1.5--CK.csv
    - ant-1.6--CK.csv
    - camel-1.0--CK.csv
    - camel-1.2--CK.csv
    - camel-1.4--CK.csv
    - camel-1.6--CK.csv
    - Eclipse_JDT_Core--CK.csv
    - Eclipse_PDE_UI--CK.csv
    - Equinox_Framework--CK.csv
    - ivy-2.0--CK.csv
    - jedit-3.2--CK.csv
    - jedit-4.0--CK.csv
    - jedit-4.1--CK.csv
    - jedit-4.2--CK.csv
    - jedit-4.3--CK.csv
    - log4j-1.0--CK.csv
    - Lucene--CK.csv
    - Mylyn--CK.csv
    - poi-2.0--CK.csv
    - synapse-1.0--CK.csv
    - synapse-1.1--CK.csv
    - synapse-1.2--CK.csv
    - velocity-1.6--CK.csv

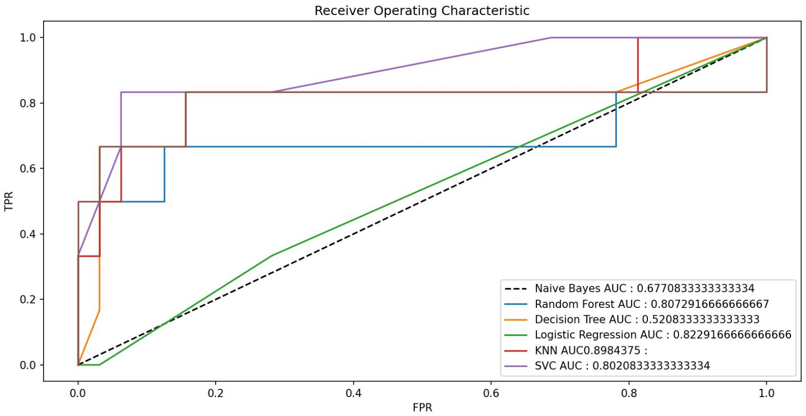
1. Firstly, We train 6 Models on these 27 datasets. They are Naive Bayes, RF, DT, SVM, LR, KNN.
2. For every training we split the dataset to 70 % train and 30 % test. Thereafter, perform a 5 fold cross validation using gridsearchCV to tune and obtain best parameters.
3. To compute the soft scores we predict the probabilities on the test data for every models. Hence Every dataset obtain 6 soft detectors with their length of Y_test.
4. Hence the size of the soft detectors respective Dataset is [6 x len(Y_test)]

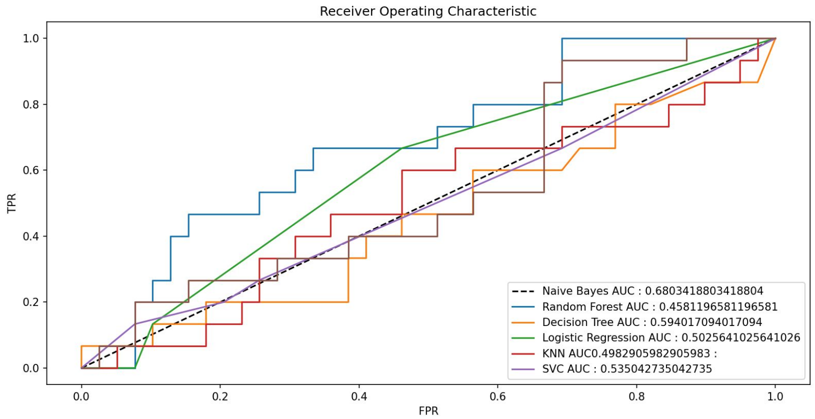
[6 x 38] [6 x 54] [6 x 88] [6 x 106] [6 x 102] [6 x 183] [6 x 262] [6 x 290]
[6 x 300] [6 x 450] [6 x 98] [6 x 106] [6 x 82]
[6 x 92] [6 x 94] [6 x 111] [6 x 148] [6 x 41] [6 x 208]
[6 x 559] [6 x 95] [6 x 48] [6 x 67] [6 x 77] [6 x 69] [6 x 132] [6 x 136]

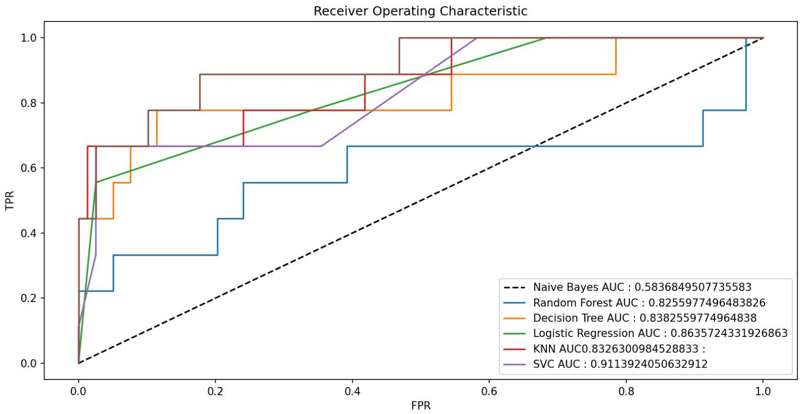


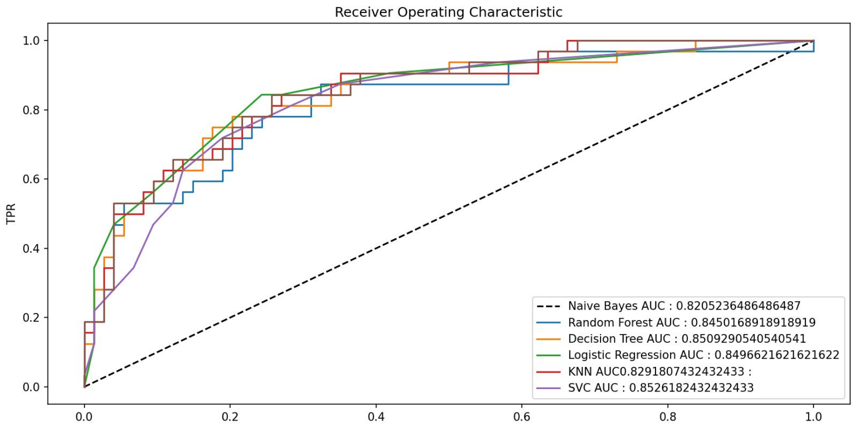
ROC and AUC results of Models

1. AUC_Table
2.









BBC2

The Pair-wise Brute-force Boolean Combination (BBC2) fuses all possible pairs of crisp classifiers generated from all the available soft classifiers using all Boolean functions. For Example, Consider 3 soft detector scores with 4 thresholds each, That would make 12 crisp detectors, Hence 1440 combinations (N*N*10) .

<u>Pros and Cons of BBC2:</u>

1. Exploits all Boolean functions using an exhaustive brute-force search to determine optimum points leads to an exponential number of combinations.
2. High Computation Complexity. O(N*N)

BBC2 Pseudo Code & Visualisation

$n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_k).

let $n = \sum_{k=1}^{K} n_k$

$BooleanFunctions \leftarrow$

$\{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$

allocate $\boldsymbol{C}$ an array of size: $[|\mathcal{V}|, n]$ // storage of all crisp detectors' decisons.

convert soft detectors to crisp detectors

for $i \leftarrow 1$ **to** K **do**

  for $j \leftarrow 1$ **to** n_i **do**

    $\boldsymbol{R} \leftarrow (D_i, t_j)$ // responses of D_i at decision threshold t_j using $\mathcal{V}$.

    push $\boldsymbol{R}$ onto $\boldsymbol{C}$

allocate $\boldsymbol{F}$ an array of size: $[2, U^2 \times size(BooleanFunctions)]$

// temporary storage of combination results.

foreach $bf \in BooleanFunctions$ **do**

  for $i \leftarrow 1$ **to** U **do**

    $\boldsymbol{R_1} \leftarrow \boldsymbol{C}_{\text{selected}}[i]$ // Retrieve Decision Vector

    for $j \leftarrow 1$ **to** U **do**

      $\boldsymbol{R_2} \leftarrow \boldsymbol{C}_{\text{selected}}[j]$

      $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_1}, \boldsymbol{R_2})$ // combine responses using current Boolean func.

      compute (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combination to ROC plane

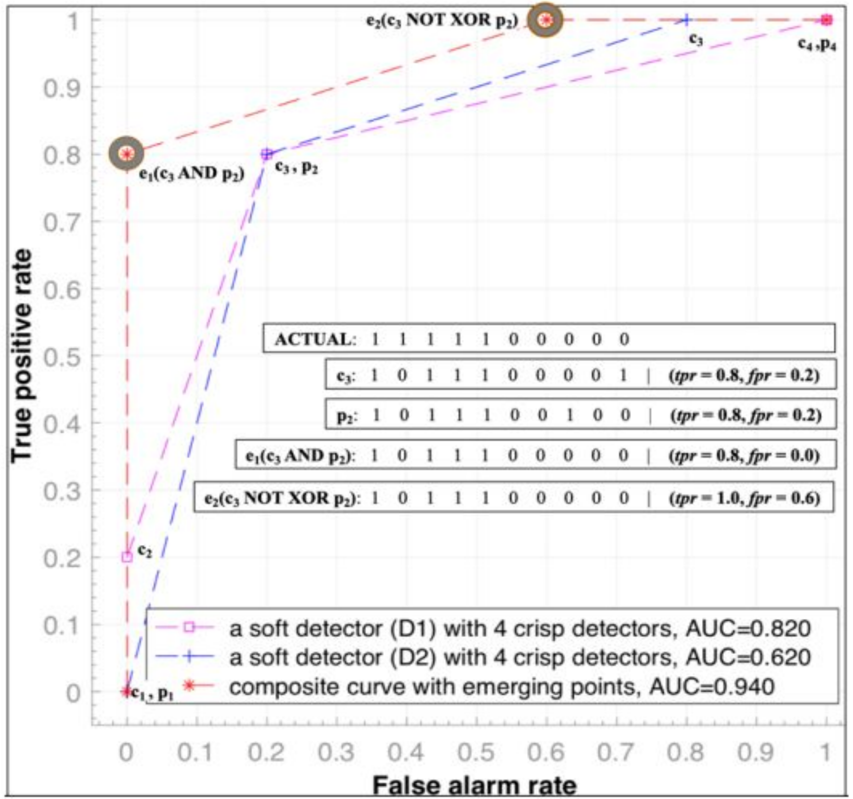
      push (tpr, fpr) onto $\boldsymbol{F}$

compute *ROCCH of all ROC points in* $\boldsymbol{F}$

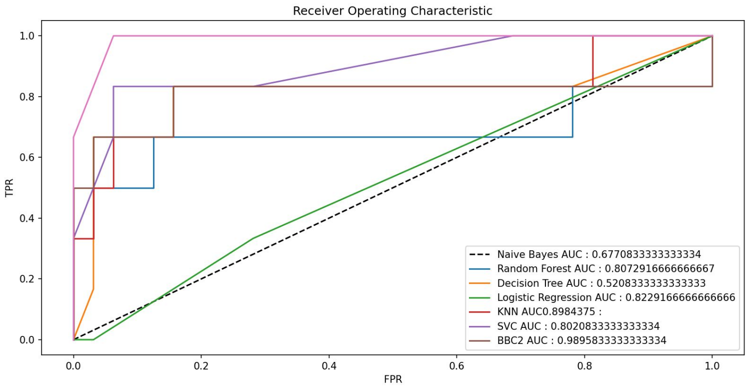
$n_{ev} \leftarrow$ number of emerging vertices

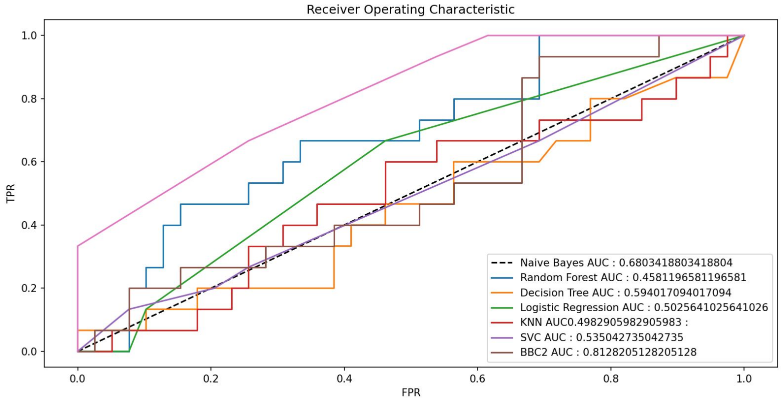
$\mathcal{S} \leftarrow \{(D_1, t_i), (D_2, t_j), \ldots, (D_k, t_k), bf\}$ // set of selected decision thresholds from each detector and Boolean functions for emerging vertices.

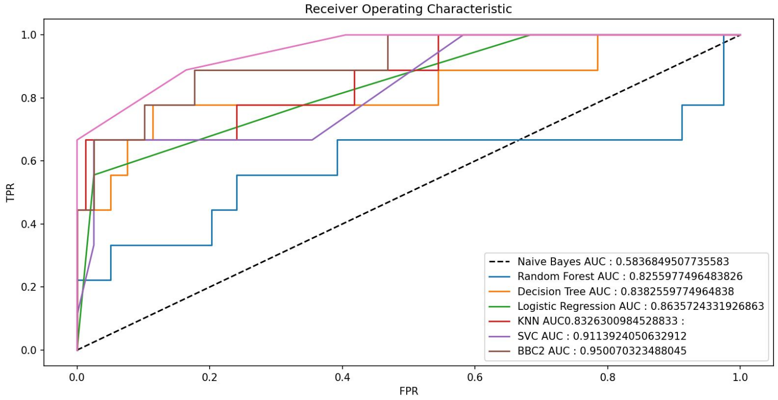
store $\mathcal{S}$; **return** *ROCCH*

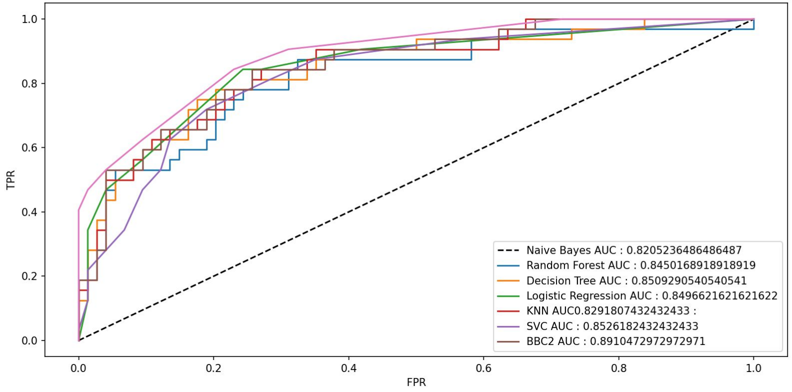


BBC2 AUC and ROC Results:









IBC

IBC avoids the impractical exponential explosion associated with the BBC2 by combining the emerging responses on a composite ROCCH sequentially. It first combines the first two ROC curves of the first two soft classifiers. Then, the combined ROCCH, particularly, the emerging points are combined with the next ROC curve, and so on until the K th ROC curve is combined. IBC repeats these sequential combinations iteratively until there are no further improvements or it reaches to a predefined maximum number of iterations.

IBC Pseudo Code

Assumptions : Max_Iter = 20, Tol = 0.001, nb_thresh = 12, All Boolean Func Included

Algorithm 1: IBC($D_1, D_2, \ldots, D_K, \mathcal{V}$): Iterative Boolean Combination

input : K soft detectors $(D_1, D_2, \ldots, D_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$

output: ROCCH of combined detectors.
- Each vertex is the result of 2 to K combination of crisp detectors.
- Each combination selects the best decision thresholds from different detectors (D_i, t_j) and Boolean function (stored in the set $\mathcal{S}$)

$n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_k).

$BooleanFunctions \leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$

compute ROCCH$_1$ of the first two detectors (D_1 and D_2)

allocate $\boldsymbol{F}$ an array of size: $[2, n_1 \times n_2]$ // temporary storage of combination results.

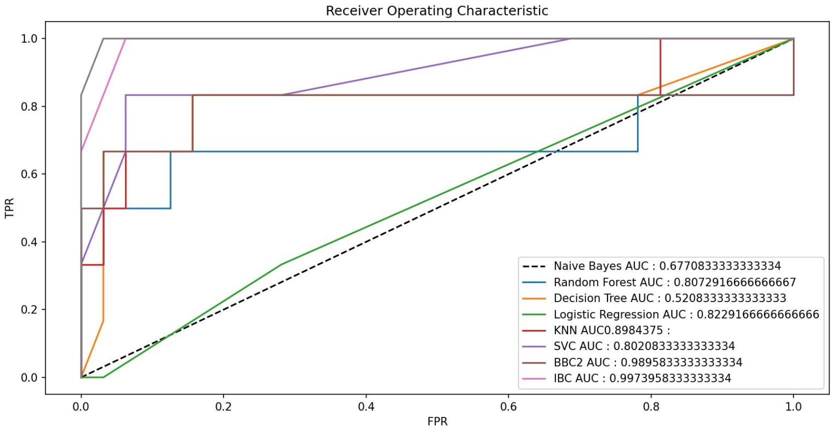
foreach $bf \in BooleanFunctions$ **do**
- **for** $i \leftarrow 1$ **to** n_1 **do**
  - $\boldsymbol{R_1} \leftarrow (D_1, t_i)$ // responses of D_1 at decision threshold t_i using $\mathcal{V}$.
  - **for** $j \leftarrow 1$ **to** n_2 **do**
    - $R_2 \leftarrow (D_2, t_j)$ // responses of D_2 at decision threshold t_j using $\mathcal{V}$.
    - $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_1}, \boldsymbol{R_2})$ // combine responses using current Boolean func.
    - **compute** (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combination to ROC plane
    - **push** (tpr, fpr) onto $\boldsymbol{F}$
- **compute** *ROCCH$_2$ of all ROC points in* $\boldsymbol{F}$
- $n_{ev} \leftarrow$ number of emerging vertices
- $\mathcal{S}_2 \leftarrow \{(D_1, t_i), (D_2, t_j), bf\}$ // set of selected decision thresholds from each detector and Boolean functions for emerging vertices.

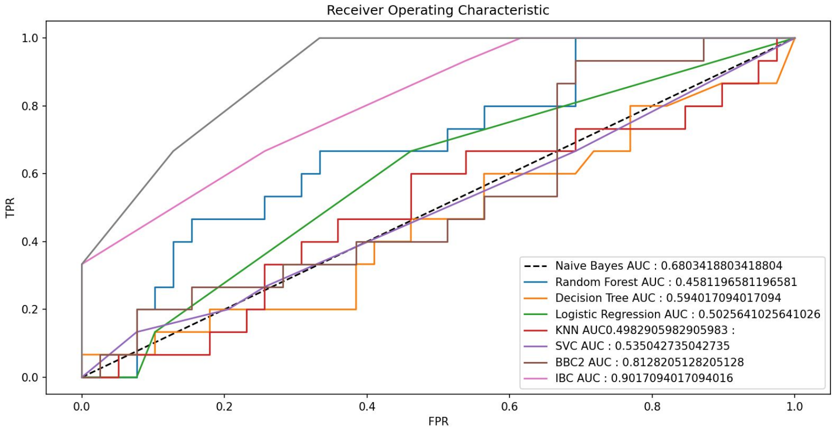
for $k \leftarrow 3$ **to** K **do**
- **allocate** $\boldsymbol{F}$ of size: $[2, n_k \times n_{ev}]$
- **foreach** $bf \in BooleanFunctions$ **do**
  - **for** $i \leftarrow 1$ **to** n_{ev} **do**
    - $\boldsymbol{R_i} \leftarrow \mathcal{S}_{k-1}(i)$ // responses from previous combinations.
    - **for** $j \leftarrow 1$ **to** n_k **do**
      - $\boldsymbol{R_k} \leftarrow (D_k, t_j)$
      - $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_i}, \boldsymbol{R_k})$
      - **compute** (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$
      - **push** (tpr, fpr) onto $\boldsymbol{F}$
- **compute** *ROCCH$_k$ of all ROC points in* $\boldsymbol{F}$
- $n_{ev} \leftarrow$ number of emerging vertices
- $\mathcal{S}_k \leftarrow \{\mathcal{S}_{k-1}(i), (D_k, t_j), bf\}$
  // $\mathcal{S}_k$ is the set of the selected subsets from the previous combinations; the decision thresholds from the newly-combined detector; and the Boolean functions that yields to the emerging vertices on the ROCCH.

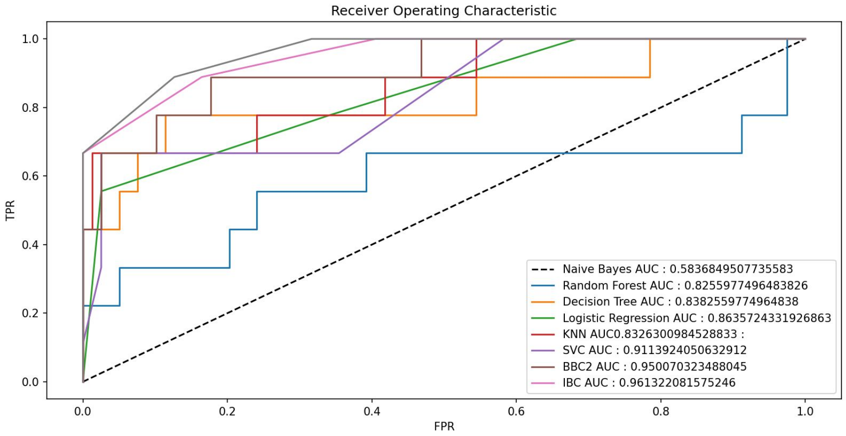
store $\mathcal{S}_k : 2 \leq k \leq K$

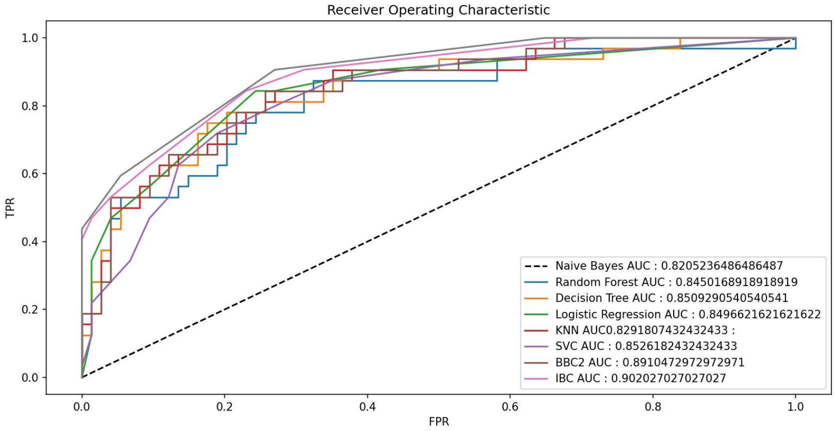
return *ROCCH$_K$*

IBC AUC and ROC Results









Doubts and Further Engagements

1. MinMaxKappa & ROCCH-Kappa Pruning
2. Pseudo Code / Matlab Code of PBC, WPBC2
3. Matlab Code of WPIBC and Pseudo Code Doubt Clarification.
4. Difference between WPBC2 and PBC.
5. Discuss the scope, deliverables and deadlines of the project.

Algorithm 2: PBC($D_1, D_2, \ldots, D_K, \mathcal{V}$): Pruned Boolean Combination

input : K soft detectors $(D_1, D_2, \ldots, D_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$

output: ROCCH of combined detectors.
- Each vertex is the result of exact 2 combination of crisp detectors.
- Each combination selects the best decision thresholds from different detectors (D_i, t_j) and Boolean function (stored in the set $\mathcal{S}$)

$n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_k).
let $n = \sum_{k=1}^{K} n_k$
$BooleanFunctions \leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$
allocate C an array of size: $[|\mathcal{V}|, n]$ // storage of all crisp detectors' decisons.
convert soft detectors to crisp detectors
for $i \leftarrow 1$ **to** K **do**
  for $j \leftarrow 1$ **to** n_i **do**
    $R \leftarrow (D_i, t_j)$ // responses of D_i at decision threshold t_j using $\mathcal{V}$.
    push R onto C

choose Pruning Technique {MinMax-Kappa, ROCCH-Kappa}
reduce n to U // $U \ll n$: is a user defined max number of detectors
return $C_{\text{selected}} \leftarrow C$ - Pruned Detectors
// Subset of size U detectors selected from all original detectors and returned for combination

allocate F an array of size: $[2, U^2 \times size(BooleanFunctions)]$
// temporary storage of combination results.
foreach $bf \in BooleanFunctions$ **do**
  for $i \leftarrow 1$ **to** U **do**
    $R_1 \leftarrow C_{\text{selected}}[i]$ // Retrieve Decision Vector
    for $j \leftarrow 1$ **to** U **do**
      $R_2 \leftarrow C_{\text{selected}}[j]$
      $R_c \leftarrow bf(R_1, R_2)$ // combine responses using current Boolean func.
      compute (tpr, fpr) of R_c using $\mathcal{V}$ // map combination to ROC plane
      push (tpr, fpr) onto F

compute *ROCCH of all ROC points in* F
$n_{ev} \leftarrow$ number of emerging vertices
$\mathcal{S} \leftarrow \{(D_1, t_i), (D_2, t_j), \ldots, (D_k, t_k), bf\}$ // set of selected decision thresholds from each detector and Boolean functions for emerging vertices.
store $\mathcal{S}$; **return** *ROCCH*