Classifiers for Predicting the Success of the Generalization of Fulton's Intersection Multiplicity Algorithm on Polynomial Systems

CS 9637

Taabish Jeshani, Wilson Poulter, Ryan Sandford

November 2021

Table of Contents

What Are Intersection Multiplicities

- Intersection multiplicities generalize the notion of multiplicity of a root for more than one polynomial.
- When an algebraic set has a singular point, local approximation at that point by a linear space is not possible.
- When this occurs, tools like the tangent cone and intersection multiplicity allow us to understand the behaviour of the algebraic set at that singularity.
- In a sense, intersection multiplicities tell us how complex a singularity is.

Example

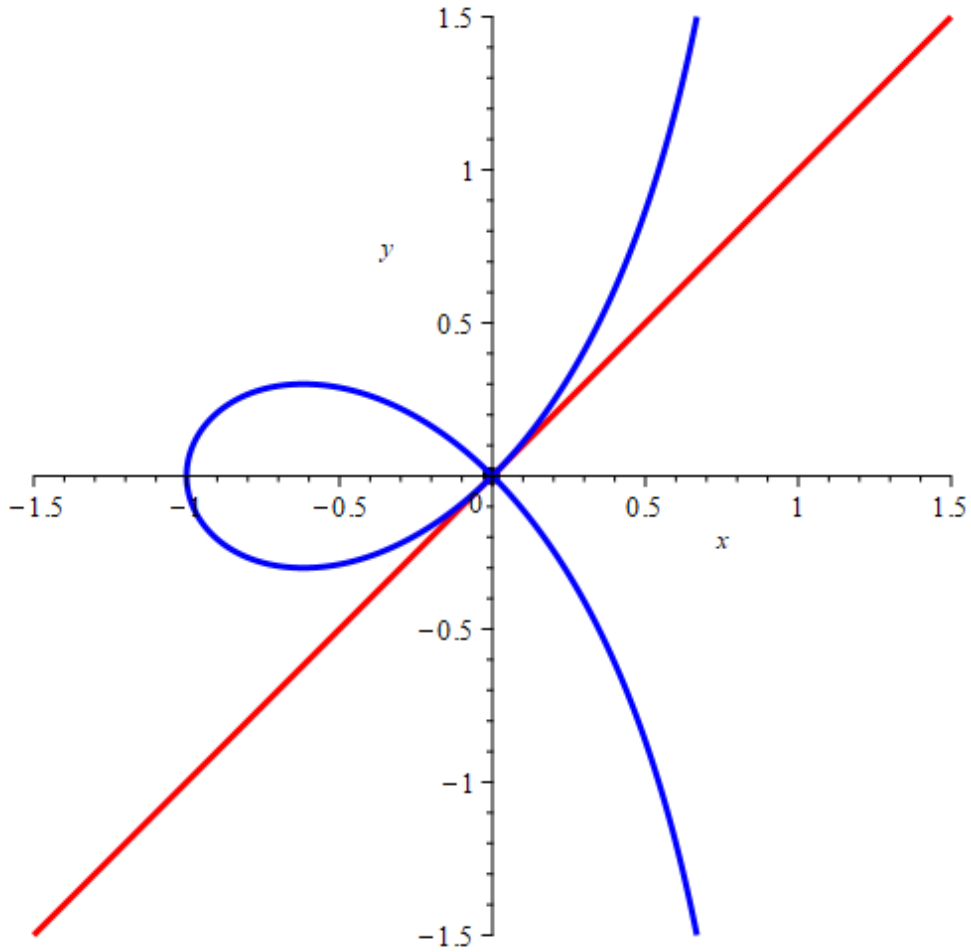


Figure: $\mathrm{Im}\left((0,0);\, y-x, x^3+xy^2+x^2-y^2\right)=3$

Standard Basis Free Algorithms for Computing Intersection Multiplicities

- Recently, in CASC 2021 [1], a new standard basis free approach was investigated, where the authors present a partial algorithm extending Fulton's bivariate intersection multiplicity algorithm to the n-variate setting.
- Since this algorithm is a partial algorithm a natural question to ask, is "when does it succeed?".
- Without running the procedure, it is difficult to tell whether a given input system will cause the algorithm to fail,
- This is because it is difficult to predict how a system will be rewritten without running the procedure itself.

Our Contribution

- We investigated several properties of polynomial systems which we believe may correlate with the success/failure of the generalization of Fulton's algorithm.
- We then randomly generated systems of polynomial equations, extracted these properties and trained several classifiers on them.
- The results suggest that some of these properties can indeed provide an indication of whether the generalization of Fulton's algorithm will succeed on a given system of polynomial equations.

Table of Contents

Generating Systems

- Using Maple's `randpoly` command we generated over 20,000 systems.
- The algorithm also requires the input systems form a regular sequence.
- This constraint is highly technical and is discussed in more detail in the appendix of our corresponding paper.
- Using the `is_regs` command in Singular, we removed all systems which were not regular sequences.
- This left 2271 valid systems to run the algorithm on.
- The systems are class imbalanced, the algorithms succeeds on 792 of the sample systems and fails on 1479 sample systems.

Table of Contents

Overview

- Data Preparation:
  - Removed duplicates,
  - Transformed raw data into feature sets,
  - Standardized features for regularization,
  - Stratified data by success/fail due to class imbalance, and
  - 7:3 split of training and test data.
- Model Selection:
  - Selected four supervised-learning methods + hyper-parameters,
  - Conducted a 2-fold cross-validation using a grid-search on features, hyper-parameters, and voting methods using the training data, scored according to average ROC AUC score from the validation sets
  - Best models of each supervised-learning method were trained on all the seen data and then tested with the unseen data, and a number of classification metrics were produced for model comparison

Feature Sets

- Polynomial Coefficients (Naïve Approach)
  - the most obvious feature set, completely determines the algorithm
  - has the drawback of having a very high dimension while not having any reasonable reduction
- Number of Vanishing Terms & Modular Degree
  - the number of vanishing terms describe the number of "bad" terms in a polynomial and the modular degree describes how difficult it is for the algorithm to remove these "bad" terms.
  - due to this, it is quite likely that most of the information needed in this problem is contained in these features, making it the better feature set

Symmetries and Voting on Permutations

- The implementation of the generalized Fulton's algorithm is symmetric:
- Let σ be a permutation of the set $\{1, \ldots, n\}$, then

$$\mathrm{Im}(p;\, f_1, \ldots, f_n) = \mathrm{Im}\left(p;\, f_{\sigma(1)}, \ldots, f_{\sigma(n)}\right)$$

- This is a property that our model should have as well.

Symmetries and Voting on Permutations (cont.)

- To achieve this, consider a voting method called *vote* that produces a single value from a set of values
- Examples include:
  - maximum value,
  - minimum value, and
  - average value
- For a trained classifier C, we define a new classifier C_{vote} that given by:

$$C_{vote}(f_1 \ldots, f_n) = vote(\{C(f_{\sigma(1)}, \ldots, f_{\sigma(n)}) : \sigma \in \mathrm{Sym}(n)\})$$

- Note that C_{vote} is indeed symmetric.

Model #1: Logistic Regression

$$\frac{1}{1 + e^{-(\vec{b} \cdot \vec{x})}}$$

Classifier:

- `sklearn.linear_model.LogisticRegression`

Fixed Parameters:

- Solver: Saga (for compatibility with L_1 penalty)

Hyper-parameters:

- Penalty: $\{L_1, L_2\}$
- Regularization: $\{1, 0.1, 0.01\}$
- Intercept: {Include, Exclude}

Model #2: Support Vector Machine

$$\vec{w} \cdot \varphi(\vec{x}) - b$$

Classifier:

- `sklearn.svm.SVC`

Fixed Parameters:

- N/A

Hyper-parameters:

- Kernel: {Polynomial, Radial Basis Function, Sigmoid}
- Regularization: {1,0.1,0.01}
- Gamma:{0.1, 0.01, 0.001}

Model #3: AdaBoost (with Tree)

$$\sum_{i=1}^{N} \alpha t_i(\vec{x})$$

Classifier:

- `sklearn.ensemble.AdaBoostClassifier`

Fixed Parameters:

- `sklearn.tree.DecisionTreeClassifier`

Hyper-parameters:

- Total Estimators: $\{10, 50, 100, 200\}$
- Learning Rate: $\{1, 0.1, 0.01\}$
- Tree Depth: $\{1, 5, 10\}$

Model #4: Gradient Boosting (with Tree)

$$\sum_{i=1}^{N} \alpha t_i(\vec{x})$$

Classifier:

- `sklearn.ensemble.GradientBoostingClassifier`

Fixed Parameters:

- Loss Function: Exponential

Hyper-parameters:

- Total Estimators: $\{10, 50, 100, 200\}$
- Learning Rate: $\{1, 0.1, 0.01\}$
- Tree Depth: $\{5, 6, 7, 8, 9\}$

Summary of Model Selection

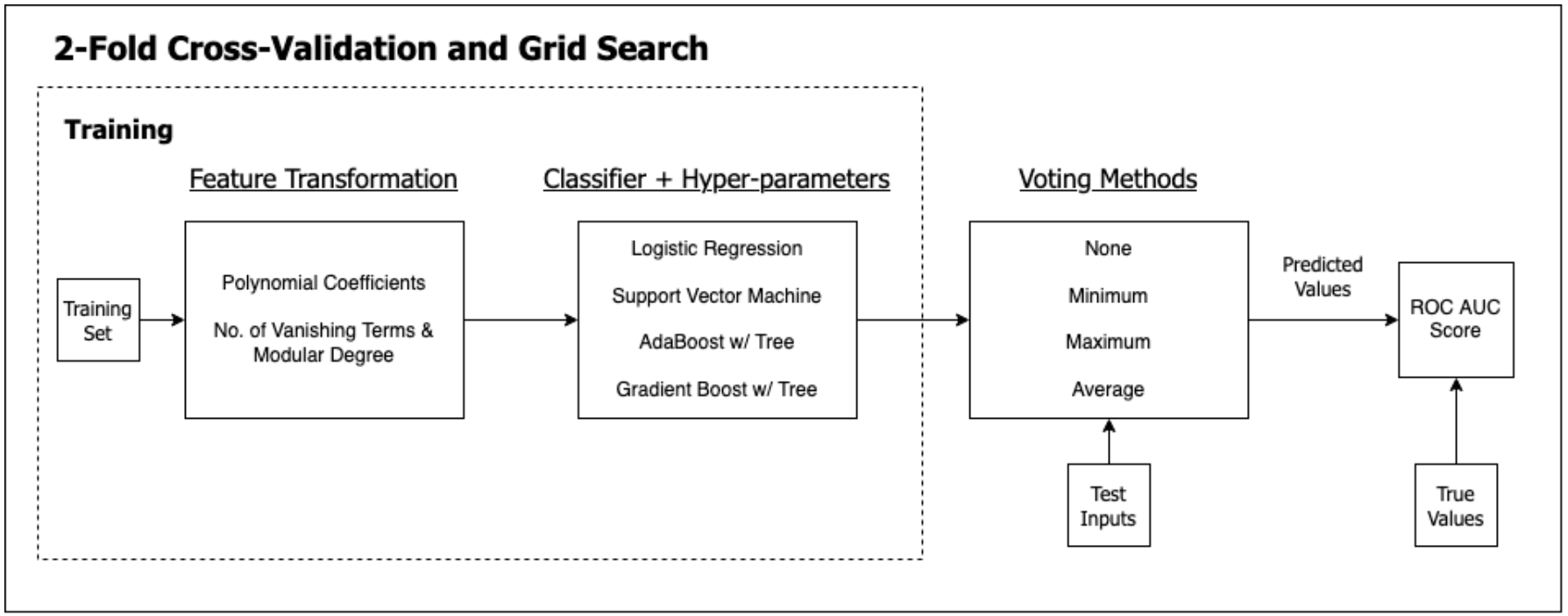


Table of Contents

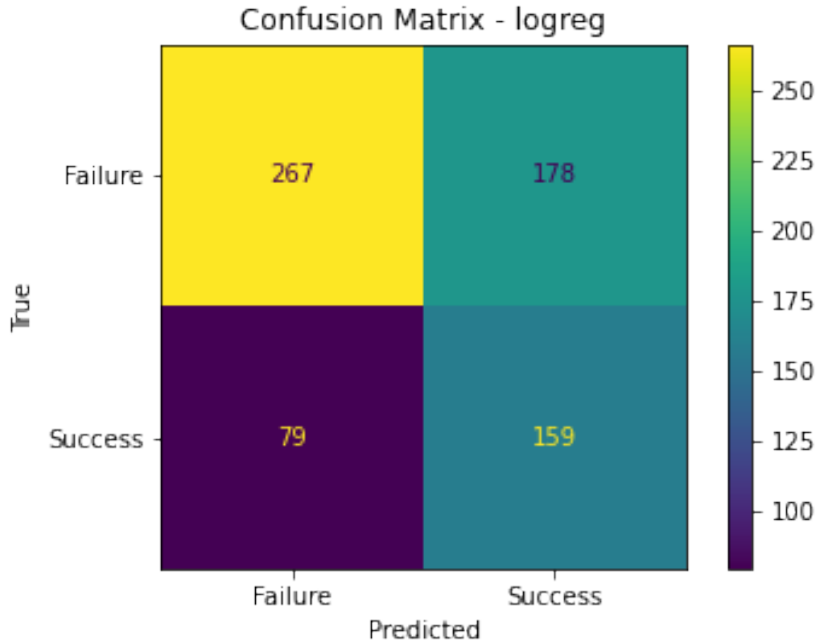
Model #1: Logistic Regression

Cross Validation Results:

Penalty	Regularization	Intercept	Feature Set	Voter	CV Score
L_2	0.01	Excluded	No. Terms + Mod. Degree	Average	0.66917888

Test Results:

- Accuracy: 0.623719
- Precision: 0.471810
- Recall: 0.668067
- F-Measure: 0.553043
- ROC AUC: 0.634034



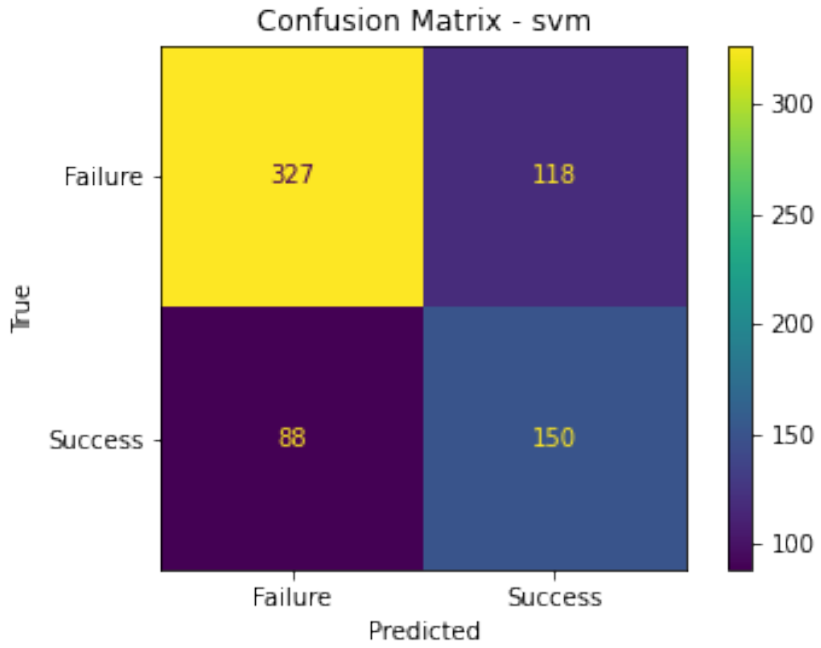
Model #2: Support Vector Machine

Cross Validation Results:

Kernel	Regularization	Gamma	Feature Set	Voter	CV Score
Radial Basis Function	1	0.1	No. Terms + Mod. Degree	Maximum	0.681360

Test Results:

- Accuracy: 0.698389
- Precision: 0.559701
- Recall: 0.630252
- F-Measure: 0.592885
- ROC AUC: 0.682542



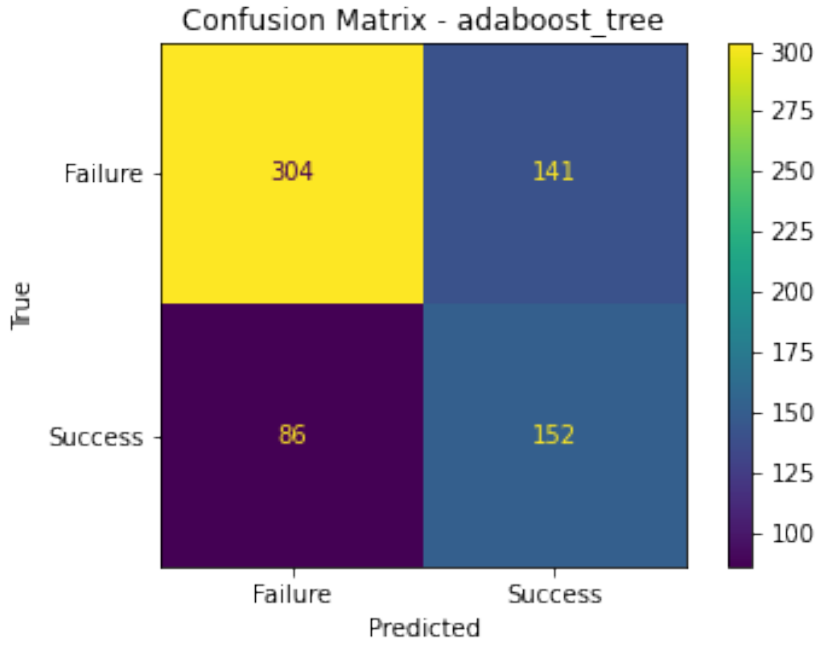
Model #3: AdaBoost w/ Tree

Cross Validation Results:

Total Estimators	Learning Rate	Tree Depth	Feature Set	Voter	CV Score
100	0.01	5	No. Terms + Mod. Degree	Maximum	0.691113

Test Results:

- Accuracy: 0.667643
- Precision: 0.518771
- Recall: 0.638655
- F-Measure: 0.572505
- ROC AUC: 0.660901



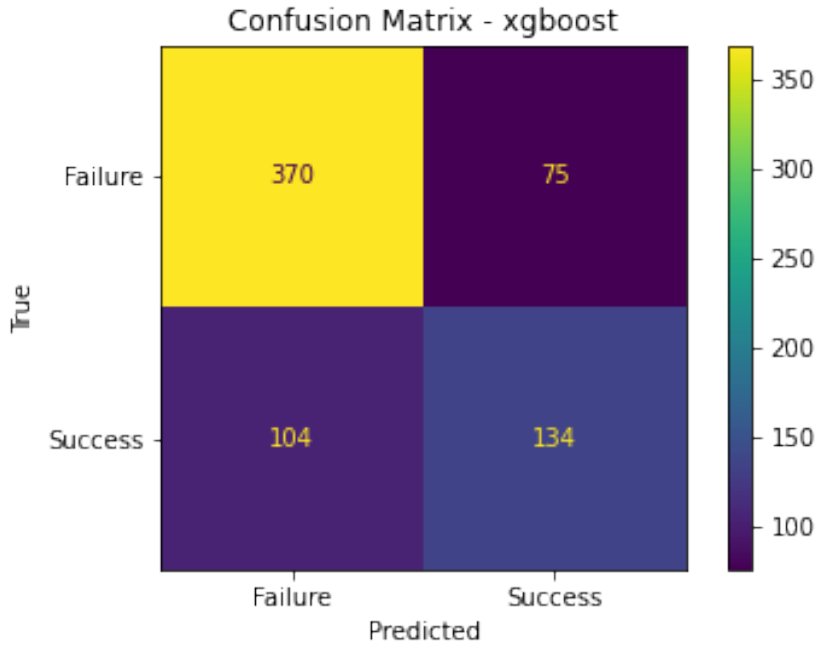
Model #4: Gradient Boosting w/ Tree

Cross Validation Results:

Total Estimators	Learning Rate	Tree Depth	Feature Set	Voter	CV Score
200	0.01	5	No. Terms + Mod. Degree	Maximum	0.705638

Test Results:

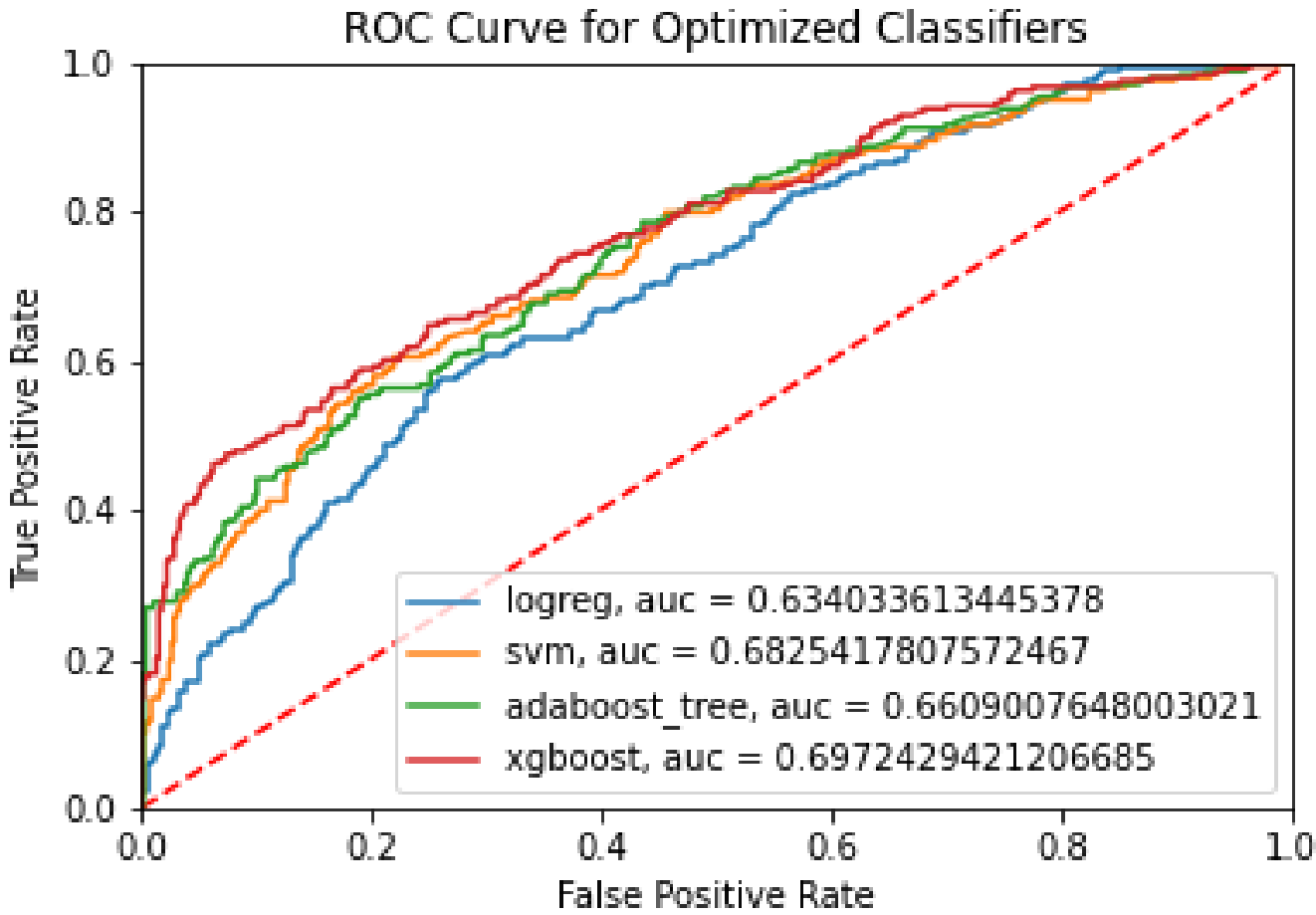
- Accuracy: 0.737921
- Precision: 0.641148
- Recall: 0.563025
- F-Measure: 0.599553
- ROC AUC: 0.697243



Summary of Results

Classifier	Accuracy	Precision	Recall	F-Measure	ROC AUC
logreg	0.623719	0.471810	0.668067	0.553043	0.634034
svm	0.559701	0.592885	0.630252	0.592885	0.682542
adaboost_tree	0.518771	0.572505	0.638655	0.572505	0.660901
xgboost	0.737921	0.641148	0.563025	0.599553	0.697243

Summary of Results (cont.)



Summary of Results (cont.)

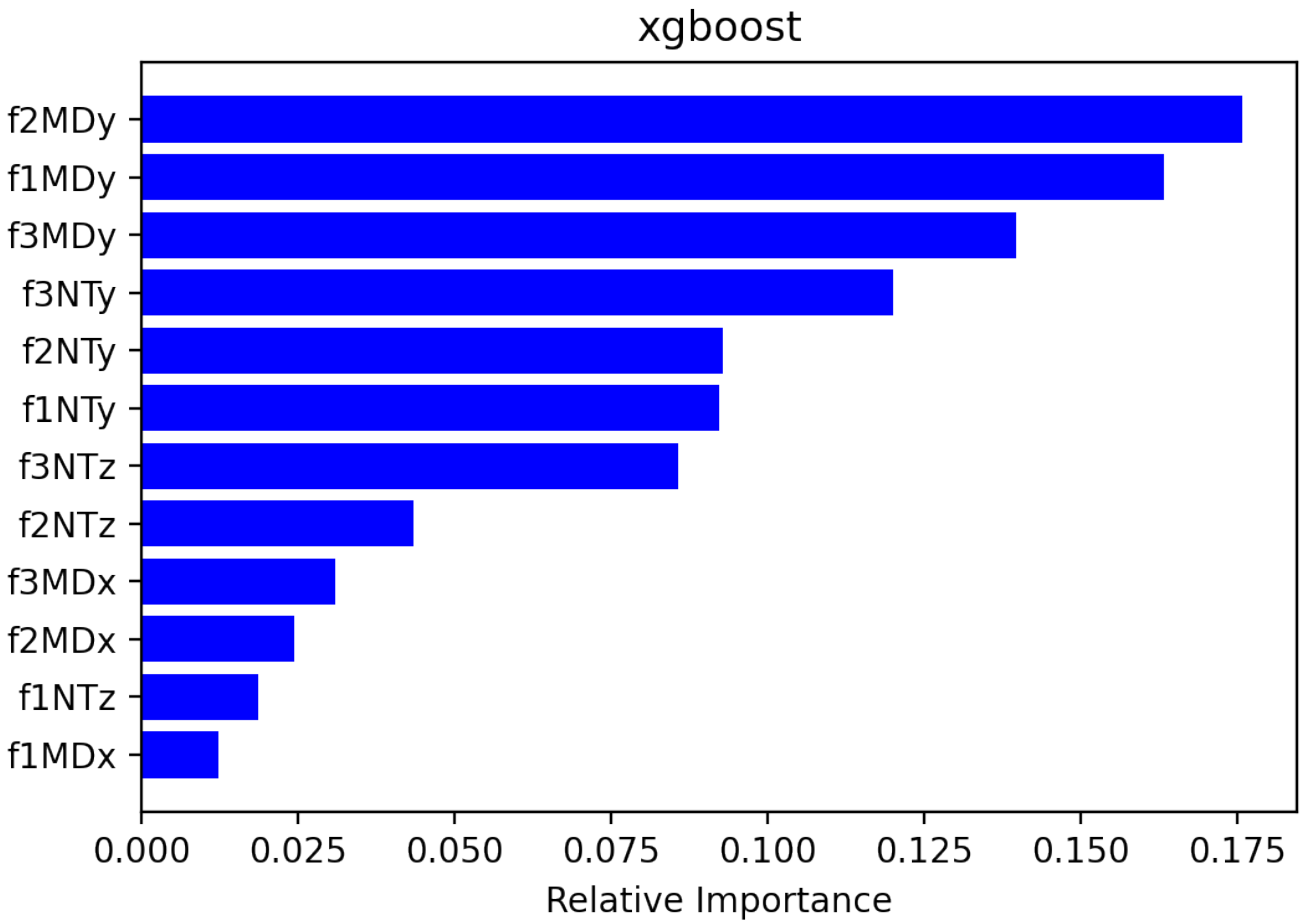


Table of Contents

References

Marc Moreno Maza and Ryan Sandford.
Towards extending fulton's algorithm for computing intersection multiplicities beyond the bivariate case.
In François Boulier, Matthew England, Timur M. Sadykov, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 23rd International Workshop, CASC 2021, Sochi, Russia, September 13-17, 2021, Proceedings*, volume 12865 of *Lecture Notes in Computer Science*, pages 232–251. Springer, 2021.