Lazy binary FCA classifier

Valeriia Demina

December 9, 2024

1

Link to Github repository with .ipynb file: https://github.com/Valdem49753/lazyFCA/tree/main

2 Description of the selected dataset

Heart Attack Analysis and Prediction Dataset Heart Disease Data Set from Kaggle was selected to test the performance of the algorithms. The dataset is available at Heart Attack Analysis and Prediction Dataset on Kaggle. The dataset presents data on the presence or absence of heart disease in 303 patients based on various parameters patients have. Row_i represent observation for person$_i$.

The goal is to determine based on the given patients features whether a patient$_i$ has: 0 = small chance of heart attack, 1= high chance of heart attack.

Number of objects (initial): 303

Number of attributes (initial): 13:

- age (numeric)
- sex (binary)
- chest pain type (categorical: 4 values)
- resting blood pressure (numeric)
- serum cholestoral in mg/dl (numeric)
- fasting blood sugar $\geq$ 120 mg/dl (binary: 1=true, 0=flase)
- resting electrocardiographic results (categorical: values 0,1,2)
- maximum heart rate achieved (numeric)
- exercise induced angina (binary: 1 = yes; 0 = no)
- oldpeak = ST depression induced by exercise relative to rest (numeric)
- the slope of the peak exercise ST segment (categorical: 0,1,2)

- number of major vessels (0-4) colored by flourosopy (categorical: 0,1,2,3,4)
- thall 1 = normal; 2 = 3ixed defect; 3 = reversable defect defectle defect (categorical: 1,2,3)

The target feature is binary, representing one of 2 classes observations belong to: the presence or absence of heart disease.

The dataset has multivariate features — they take on binary, categorical and numeric values. To perform successful Lazy FCA Classification, since the classifier works only with binary features, we have to transform the selected data set feature columns to the binary form (scaling), i.e. to such form, where features can take only two values — binary.

To all features we have to apply its own way of scaling:

1. Binary (sex, fbs, exng). For such features, the values were replaced with 0 and 1 if they already weren't already represented with 1 and 0.
2. Categorical (cp, restecg, slp, caa, thall). For each such feature, new ones were added, one for each categorical value the corresponding feature has. The value 1 was set if the given attribute had this value. After that, the original categorical features were removed.
3. Numeric (age, trtbps, chol, thalach, oldpeak). For each such feature, intervals of acceptable values of the form $(\geq n_1, +\infty), ..., (\geq n_i, +\infty)$ were calculated, where $n_i \leq n_{i+1}$, $n_1, ...n_i$ are threshold values for new columns and belong to value set of an original attribute. The value 1 was set if the condition "greater than or equal to the threshold value of new feature" of value from the original attribute was met for the object. After that, the original numeric characters were removed.

A binary data set was obtained that the classifier can work with. The new dataset consists of 60 binarised and scaled feature columns and 1 binary class column and 301 rows. After scaling and binarisation features columns and target column were separated into different datasets. These datasets were split into training and testing datasets.

3 Algorithm description

To perform the classification, 3 different lazy-methods were used and their results were compared with each other to find the best algorithm:

- An initial algorithm based on "generators" using support criterion. This method along with the code has been described in the manual for building a binary classifier.
- The initial algorithm based on "generators" exploring various support criteria.

- Proposed algorithm. The principle of its operation:
  1. From the set of all attributes, generate all attribute-subsets (patterns) of size up to a predefined maximum. For example, if the maximum pattern size is 2 and the attributes are {A, B, C}, then the patterns could include (A), (B), (C), (A,B), (A,C), and (B,C).
  2. For each pattern, calculate its discriminative score based on how frequently it appears in the positive and negative subsets of the training data.
  We define:

  $$P(\text{pattern} \mid +) = \frac{\#\{\text{positive samples containing the pattern}\}}{\#\{\text{all positive samples}\}},$$

  $$P(\text{pattern} \mid -) = \frac{\#\{\text{negative samples containing the pattern}\}}{\#\{\text{all negative samples}\}}.$$

  The score of the pattern is:

  $$\text{score(pattern)} = P(\text{pattern} \mid +) - P(\text{pattern} \mid -).$$

  3. Patterns which are strongly indicative of either class (either + or -) are retained: we keep only those patterns whose absolute score exceeds a given threshold. These selected patterns, each with an associated score, form the "dictionary" of discriminative patterns to be used later during classification.
  4. Classification: For a new object, determine which of the retained patterns are present. A pattern is considered present if all its attributes are active (equal to 1) in the object attribute set. Then sum the scores of all the patterns that are present in the test object.
  5. If the total summed score is greater than zero, classify as positive. If it is less than zero, classify as negative. In case the sum is exactly zero, apply a tie-breaking rule (in out case we default to the majority class).

4 The results of testing the classifiers

Data set is randomly divided into a training and test sample usng reproducible random seed , the test sample is classified, compared with the true values, and quality metrics are calculated.

- For the second classifier, the following parameters were chosen:
  - For positive class objects:
    * `min_positive_support` — the minimum number of "+" objects' descriptions, in which an intersection is present, for this intersection to increment (+1) positive discriminator of a given test sample - for an intersection of this test sample with a positive training sample,

    - `max_negative_allowed_for_positive` — the max number of "-" objects' descriptions,in which an intersection is present, for this intersection to increment (+1) positive discriminator of a given test sample
  - For negative class objects:
    - `min_negative_support` — the minimum number of "-" objects' descriptions,in which an intersection is present, for this intersection to increment (+1) negative discriminator of a given test sample - for an intersection of this test sample with a negative training sample,
    - `max_positive_allowed_for_negative` — the max number of "+" objects' descriptions,in which an intersection is present, for this intersection to increment (+1) negative discriminator of a given test sample
  - `min_cardinality` — min value of intersection size for a given intersection to contribute to + or - discriminators

- For the 3 classifier, the following parameters were chosen:
  - `max_pattern_size` – maximum size of attribute subsets to consider as patterns,
  - `score_threshold` – minimum absolute score required to keep a pattern

All the metrics proposed in the manual were used as quality metrics.

Metrics	**Initial**	**2 with (0,10,2,0,10)**	**2 with (0,0,2,0,0)**	**2 with (0,4,2,0,4)**
True Positive	29.00	27.00	28.00	28.00
True Negative	21.00	19.00	22.00	21.00
False Positive	7.00	9.00	6.00	7.00
False Negative	4.00	6.00	5.00	5.00
True Negative Rate (Specificity)	0.75	0.68	0.79	0.75
Negative Predictive Value	0.84	0.76	0.81	0.81
False Positive Rate	0.25	0.32	0.21	0.25
False Discovery Rate	0.19	0.25	0.18	0.20
Accuracy	0.82	0.75	0.82	0.80
Precision	0.81	0.75	0.82	0.80
Recall (True Positive Rate)	0.88	0.82	0.85	0.85
F1 Score	0.84	0.78	0.84	0.82

Tab. 1. Metrics for Lazy classifier with "Generators" method:
1 - initially suggested algorithm,
2 - initial algorithm with modified parameters (Param values:
max_neg_for_pos=0 , min_pos_sup = 10 , min_card = 2 , max_pos_for_neg = 0 , min_neg_sup= 10)
3 - initial algorithm with modified parameters

(Param values:
max_neg_for_pos=0 , min_pos_sup = 0, min_card =2 , max_pos_for_neg = 0, min_neg_sup= 0)
4 - initial algorithm with modified parameters (Params: 0,4,2,0,4)

Metrics	**Model 1 (Initial)**	**Model 3_0 (3 with (2,0))**	**Model 3_1 (3 with (3,0))**
True Positive	29.00	27.00	27.00
True Negative	21.00	24.00	24.00
False Positive	7.00	4.00	4.00
False Negative	4.00	6.00	6.00
True Negative Rate (Spec.)	0.75	0.86	0.86
Negative Predictive Value	0.84	0.80	0.80
False Positive Rate	0.25	0.14	0.14
False Discovery Rate	0.19	0.13	0.13
Accuracy	0.82	0.84	0.84
Precision	0.81	0.87	0.87
Recall (True Positive Rate)	0.88	0.82	0.82
F1 Score	0.84	0.84	0.84

Tab. 1. Metrics for Lazy classifier with a new proposed method:
1 - initially suggested algorithm,
2 - proposed algorithm with modified parameters (Param values:
max_pattern_size = 2, score_threshold= 0)
3 - proposed algorithm with modified parameters (Param values:
max_pattern_size = 3, score_threshold= 0)

Metrics	**Model 1 (Initial)**	**Naive Bayes (Bernoulli)**	**Random Forest**	**XGBoost**
True Positive	289.00	28.00	29.00	29.00
True Negative	21.00	23.00	19.00	21.00
False Positive	7.00	5.00	9.00	7.00
False Negative	4.00	5.00	4.00	4.00
True Negative Rate (Spec.)	0.75	0.82	0.68	0.75
Negative Predictive Value	0.84	0.82	0.83	0.84
False Positive Rate	0.25	0.18	0.32	0.25
False Discovery Rate	0.19	0.15	0.24	0.19
Accuracy	0.82	0.84	0.79	0.82
Precision	0.81	0.85	0.76	0.81
Recall (True Positive Rate)	0.88	0.85	0.88	0.88
F1 Score	0.84	0.85	0.82	0.84

5 Conclusion

Overall all classifiers, including standard classification algorithms, show approximately the same performance.
The best result from the suggested methods based on F1 score is achieved with proposed method with parameters: max_pattern_size = 3, score_threshold= 0.
Increasing score_threshold decreases F1 score, increasing max pattern size time

grows considerably.
As for the 'generator' classifier method:
Decreasing min_cardinality considerably decreases model performance.
Decreasing minimum_positive_support and minimum_negative_support increases model performance, increasing them does the opposite.
Increasing max_negative_for_positive and max_positive_for_negative from 0 decreases model performance.
The best parameters configuration for generator method is thus:
max_neg_for_pos=0 , min_pos_sup = 0, min_card =2 , max_pos_for_neg = 0, min_neg_sup= 0