# Binary Classification based on Formal Concept Analysis

Sheikh Tajamul

13 December 2022

## 1 Introduction

This report is created as part of the course "Ordered sets in data analysis". The project is focused on the implementation of a lazy binary classifier based on Formal Concept Analysis methodologies (FCA). Typically, the classifier's job can be divided into two steps: pattern selection in the training sample (training) and classification. Because only a subset of the information from the training sample is used in the classification, the classification is completed rapidly. The lazy classification approach differs in that the classifier uses the complete training sample without prior training, which takes much longer but can improve classification accuracy. A total of three different methods were used to perform the lazy classification. Later the proposed methods were compared based on the accuracy with different rule based models.

## 2 Description of the selected data sets

Three data sets were selected for the project:

1. Breast Cancer Data Set from the UCI Machine Learning Repository. The data set presents data whether the patient has benign or malignant cancer on 699 patients
2. Heart Disease Data Set from the Kaggle. The data set presents data on the presence or absence of heart disease in 297 patients.
3. Fruits Data Set from the Kaggle. The data set presents data whether fruit is orange or grapefruit.

A data set is a table where the rows correspond to observations and the columns correspond to feature values. Note that the formal context $K$ can be represented as a similar table, where the rows correspond to objects (observations), and the columns — attributes (features). In this case, each $i$-th row and $j$-th column will have a value of 1 if $(g_i, m_j) \in I$ and 0 if $(g_i, m_j) \notin I$. Such a set of data can be called binary, and features that can take only two values —

binary. Since the classifier works only with binary features, it is necessary to transform the selected data sets to the binary form (scaling). Different scaling strategies were chosen for (1) and (2).

Let us consider dataset 1, where each patient has features as 'Sample code number','Clump Thickness','Uniformity of Cell Size','Uniformity of Cell Shape', 'Marginal Adhesion','Single Epithelial Cell Size','Bare Nuclei','Bland Chromatin', 'Normal Nucleoli','Mitoses', depends on the feature values present, patient is classified as either having a benign or malignant cancer. The label for benign in the dataset is given as 2 and for malignant it's 4. A replacement was performed for the target attribute: 0 if "benign" was specified and 1 otherwise. Two different scaling operations were performed on binary data that the classifier can work with.

Consider dataset 2. Different information is provided for each patient, such as gender, age, and blood pressure. As a target feature, a binary one was selected: the presence or absence of heart disease. All these features can be divided into three groups, each of which has been applied its own way of scaling:

1. Binary (for example, gender, presence of disease). For such features, the values were replaced with 0 and 1. The specific value for which we need to specify 0 (or 1) does not play a role, so it was chosen randomly.

2. Categorical (for example, the type of pain). For each such feature, new ones were added, one for each value. The value 1 was set if the given attribute had this value. After that, the original categorical features were removed.

3. Numeric (for example, blood pressure). For each such feature, an interval of acceptable values of the form $[\min(m_i), \max(m_i)]$ was calculated. Further, each interval was proportionally divided into 5 intervals, each of which was added as a new feature. The value 1 was set if the condition "greater than or equal to" value from the original attribute was met for the object. After that, the original numeric characters were removed. Thus, a binary data set was obtained that the classifier can work with.

The fruits dataset contained 6 features names as (name, diameter, weight, red, green, blue). "name" feature is basically a target label. A replacement was performed for the target attribute: 0 if "grapefruit" was specified and 1 otherwise. After the replacement the data was shuffled randomly and first 900 samples were taken. The data sample was evenly distributed which contained 461 grapefruit and 439 oranges. After that the scaling was performed on that datset.

# 3 Models and classification method

Three classification models were tested on the Breast Cancer dataset and later only one model was used for classification of other datasets:

1. The given pipeline was adapted on the breast cancer dataset and its accuracy results were compared with other classification models.

2. In the proposed classification algorithm, the dataset set was divided into positive and negative classes and after that the algorithm would try to determine the objects from the unclassified set to either positive or negative class. The alogrithm would make the intents of the classes (+ and -). All the unclassified objects would then obtain their intents. After that every unclassified object would intersect with the positive and negative objects. However, there is a small problem that we need to consider that is, if unclassified obejct has inteersection only with positive intents, then we define it as positive like wise we do for the negative. if there is both positive and negative then there is a contradiction.

3. Represent the training data set as ordered sets (context) $K = (G, M, I)$, where $G$ — the set of objects, $M$ — the set of attributes, $I$ — the set of relations between $G$ and $M$. We can define $I$ as a set $\{(g, m)|g \in G, m \in M, gIm\}$. One of all attributes is called the target attribute $m_t$. Then the initial set of objects are divided into two sets according to the rule:

$$G_+ = \{g \in G|(g, m_t) \in I\}$$

$$G_- = \{g \in G|(g, m_t) \notin I\}$$

Now consider the set of test objects $G_\tau$, whose relations with all attributes except the target are known. Objective of the classifier — define $(g_\tau, m_t) \in I$ or $(g_\tau, m_t) \notin I$.

In the next sections we will use the "prime" operator $(\cdot)'$ from FCA theory, which define as:

$$A' = \{m \in M|(g, m) \in I, \forall g \in A\}$$

$$B' = \{g \in G|(g, m) \in I, \forall m \in B\}$$

Call the set $g'$ as the description of the object $g$. Then the algorithm can be represented as:

   (a) An arbitrary object $g_\tau$ from the test sample $G_\tau$ is fed to the classifier input.

   (b) We select the $i$-th element $g_i^+$ from the set $G_+$ and calculate the intersection of its description with the description $g_\tau$, that is

$$\Delta_i^+ = (g_i^+)' \cap (g_\tau)'$$

   (c) The proportion of objects $S_i^+$ from the set $G_-$ for which the resulting intersection is included in the descriptions of these objects is calculated. That is

$$S_i^+ = \frac{|\{g^- \in G_-|\Delta_i^+ \subseteq (g^-)'\}|}{|G_-|} \quad (1)$$

(d) If the share of $S_i^+$ does not exceed the pre-selected threshold value $p \in [0, 1]$, then the power of $i$-th intersection $\Delta_i^+$ is taken into account in the support for a positive decision:

$$Support^+ = \frac{\sum_i |\Delta_i^+| \mathbb{1}\{S_i^+ < p\}}{|G_+|} \tag{2}$$

This sums up the support for all elements from the positive set.

(e) Similarly, support for a negative decision is calculated

$$Support^- = \frac{\sum_i |\Delta_i^-| \mathbb{1}\{S_i^- < p\}}{|G_-|} \tag{3}$$

(f) The classifier evaluates the rule

$$Prediction = \begin{cases} g_\tau \in G_+ \text{ if } Support^+ > Support^- \\ g_\tau \in G_- \text{ if } Support^+ < Support^- \end{cases}$$

We should also consider the situation when $Support^+ = Support^-$. In this case, one (pre-selected) of the three strategies is used:

(a) The class is chosen randomly with a probability of each class 0.5

(b) Always choose a positive class

(c) Always choose a negative class

In addition, consider a modification of the method in which the classification is not used for the entire training sample, but only a random fraction of it. Then at all classification steps, instead of $G_+$ and $G_-$ will be used respectively:

$$G_+^{rand} = \{g_i | g_i \in G_+, i \in X_+\}$$
$$G_-^{rand} = \{g_i | g_i \in G_-, i \in X_-\}$$

where $X_+$ — a random set of indexes of elements from $G_+$ and $k < |G_+|$, each of which is equally likely to fall into this set. Similarly, the set $X_-$ for $l < |G_-|$ indexes of elements from $G_-$is defined. The values $k$ and $l$ are selected in advance before the classification begins. This modification allows to significantly reduce the calculation time. Although the *Prediction* estimate loses its accuracy, it remains unbiased by using random equally probable selection of elements.

# 4 Implementation and Time Complexity

Two methods were developed and used on different datasets the main interfaces of the classifier were:

- fit — the method is used to train the model. A training sample is sent to the input, on the basis of which the classification rules are formed. Since no training is required in our case, this method is used to store the entire training sample in the classifier object.
- predict — the method is used for classification. The input is a sample, for each element of which you want to predict the value of the target attribute.
- score — the method is used to calculate the accuracy metric. The input is a sample for prediction, as well as the true values of the target attribute. By comparing the predicted and true values, the accuracy metric is calculated.

Now evaluate the categorization algorithm's algorithmic complexity. Indicate the number of objects and attributes

$$m = |G|$$

$$n = |M|$$

Since $\Theta(|G_+|) = \Theta(|G_-|) = \Theta(|G|)$ the number of objects in a positive and negative sample cannot differ from the total number of elements by more than a certain constant, so we will only consider the complexity of the calculation of $support^+$, keeping in mind that the calculation of $support^-$ has the same asymptotic complexity.

Think about first finding the intersection of $\Delta_i^+$. In the worst scenario, we obtain the entire collection of characteristics when crossing the descriptions of two identical objects, which is the complexity of such an operation —- $Theta(n)$.

Next, to calculate the parameter $S_i^+$, we need to check whether the intersection is included in the description of each object of the negative selection, that is, the complexity of the operation taking into account the previous one — $\Theta(n \cdot n \cdot m)$.

The values of $S_i^+$ and $\Delta_i^+$ for each item in $G_+$ must then be calculated in order to determine support for $Support+$. This operation has no impact on the total complexity because it may be done directly during the calculation of this intersection for each $\Delta_i^+$. That is, the operation's complexity when the prior —- — $\Theta(m \cdot n \cdot n \cdot m)$ is taken into account.

Lastly, the algorithm's overall complexity (in time):

$$Complexity = \Theta(m^2 \cdot n^2)$$

However, given the fact that the number of attributes is usually significantly less than the number of objects, we can perform the following optimization. Before starting the classification for each attribute $m_i$ we get "index" — a set of objects for which $(g, m_i) \in I$, that is

$$Index_i = \{g | (g, m_i) \in I\}$$

This operation must be performed for a positive and negative class and thus for each attribute $m_i$ we get a pair of indexes $Index_i^+$ and $Index_i^-$.

Now we can rewrite the formula (1) as:

$$S_i^+ = \frac{\left|\left\{g | g \in \left(\bigcap_{j \in \Delta_i^+} Index_j^-\right)\right\}\right|}{|G_-|}$$

Thus, the overall complexity of the algorithm after optimization:

$$Complexity = \Theta(m \cdot n^3)$$

# 5 Results and comparison with other classifiers

Dataset was scaled and then randomly divided 70:30 and some models were tested on 40:60 dataset too, the test sample is then classified, compared with the true values, and quality metrics are calculated.

The model parameters were selected as:

- **Bias** — the decision to make if $Support^+ = Support^-$. There are three options: Positive (always set a positive class), Negative (always set a negative class), and Random (set a random class).
- **Threshold** — threshold value for the parameter $p \in [0, 1]$, which is used in the formula (2) and (3).
- **Random** — True to enable a mode that uses only a randomly selected portion of the training sample, False — to disable the mode.
- **Sample share** — if **Random** mode is used, this parameter sets the percentage of entries from the positive and negative set. Valid values in the range $[0, 1]$.

Widely known metrics were used to evaluate the quality of the model:

$$Recall = \frac{tp}{tp + fn}$$

$$Precision = \frac{tp}{tp + fp}$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

The test results for different datsets are given below. The "#" column contains the number of the cross-validation partition.

## 5.1 Breast-Cancer

| Classifier | Accuracy | Presicion | Recall |
|---|---|---|---|
| FCA | 0.859915 | 0.952199 | 0.623888 |
| DecisionTreeClassifier | 0.908943 | 0.952199 | 0.900452 |
| RandomForestClassifier | 0.956098 | 0.952199 | 0.936652 |
| AdaBoostClassifier | 0.951220 | 0.952199 | 0.909502 |
| GradientBoostingClassifier | 0.917073 | 0.952199 | 0.886878 |

Tab. 1. Breast cancer dataset on given pipeline

| Classifier | Accuracy | Presicion | Recall |
|---|---|---|---|
| FCA | 0.92 | 0.10 | 0.78 |
| DecisionTreeClassifier | 0.93 | 0.91 | 0.91 |
| RandomForestClassifier | 0.97 | 0.96 | 0.97 |
| AdaBoostClassifier | 0.94 | 0.93 | 0.93 |
| GradientBoostingClassifier | 0.97 | 0.96 | 0.96 |
| XGBoostClassifier | 0.96 | 0.96 | 0.96 |
| CatBoostClassifier | 0.97 | 0.98 | 0.97 |
| EntireTreeclassifier | 0.98 | 0.97 | 0.97 |

Tab. 2. Breast cancer dataset using the first proposed algorithm without using the concept of threshold

| Classifier | Accuracy | Presicion | Recall |
|---|---|---|---|
| FCA | 0.892683 | 1.000000 | 0.710526 |
| DecisionTreeClassifier | 0.951220 | 0.971429 | 0.894737 |
| RandomForestClassifier | 0.975610 | 0.986301 | 0.947368 |
| AdaBoostClassifier | 0.951220 | 0.971429 | 0.894737 |
| GradientBoostingClassifier | 0.970732 | 0.986111 | 0.934211 |

Tab. 3. Breast cancer dataset using the first proposed algorithm using the concept of threshold.

## 5.2 Heart Disease

| Classifier | Accuracy | Presicion | Recall |
|---|---|---|---|
| FCA | 0.815642 | 0.849315 | 0.738095 |
| DecisionTreeClassifier | 0.748603 | 0.746835 | 0.702381 |
| RandomForestClassifier | 0.804469 | 0.826667 | 0.738095 |
| AdaBoostClassifier | 0.787709 | 0.802632 | 0.726190 |
| GradientBoostingClassifier | 0.765363 | 0.776316 | 0.702381 |

Tab. 4. Heart disease dataset using the proposed algorithm with the concept of threshold.

### 5.3 Fruits Classification

| Classifier | Accuracy | Presicion | Recall |
|---|---|---|---|
| FCA | 0.677778 | 0.707692 | 0.652482 |
| DecisionTreeClassifier | 0.903704 | 0.913669 | 0.900709 |
| RandomForestClassifier | 0.918519 | 0.934307 * 0.907801 | |
| AdaBoostClassifier | 0.922222 | 0.947761 | 0.900709 |
| GradientBoostingClassifier | 0.911111 | 0.927007 | 0.900709 |

Tab. 5. Fruits Classification dataset using the proposed algorithm with the concept of threshold.

## 6 Conclusion

From the results obtained, it can be seen that on the Heart diseasedata set, the lazy classifier managed to achieve 81 percent accuracy with the parameters Threshold = 0.001, Random = True, Bias = R. The prediction time for each set averaged 60 seconds. If we use Random mode, the time is significantly reduced. The random forest also achieves a high accuracy of 0.80, but the classification is almost instantaneous — on average 0 seconds, which is much better than the lazy classification in any mode. However we can say our model outperformed other while classifying heart diseases.

For the Breast Cancer data set, the proposed lazy classifier achieves the accuracy (0.92) which is better than the pipeline that was given to us.

We can conclude that the methods of lazy classification are guaranteed to work longer than the usual classification models — decision tree and so on. Although the calculation time can be reduced using Random mode, it still remains quite large. Perhaps the speed and accuracy of the lazy classification model can be improved by using probabilistic approaches, such as boosting. It is worth noting that the heart disease data set was able to great accuracy, which was not possible for the decision tree, but there is no certainty that a similar result will be reproducible on other data sets.

### 6.1 Working Code

The working code for the project can be found at: https://github.com/sheikhtajamul38/LazyFca . Repository contains the python notebooks as:

1. Lazyfca.ipynb contains the given code without any optimizations
2. lazy_optimized_threshold.ipynb contains the code with implementation of threshold and other things which were discussed earlier
3. lazy_optimized_without_threshold.ipynb is basically the first proposed model. see other details in the models section.
4. lazy_test_heart_disease.ipynb contains the code on heart disease dataset.
5. lazy_test_fruit.ipynb contains the code on fruits dataset.