

Lazy FCA Classification Using Pattern Structures

Final Project – Ordered Sets for Data Analysis & Interpretable Methods for Classification

Student: Arman Malkhasyan

GitHub: <https://github.com/Armanna/hse-data-science/tree/main/OSiDA>

Abstract

This project investigates the use of Lazy Formal Concept Analysis (FCA) with Pattern Structures for binary classification on the Breast Cancer Wisconsin (Diagnostic) dataset. The Lazy FCA baseline is implemented using interval pattern structures, compared against standard machine learning models, and improved using a novel directional Max–Min support method. Results demonstrate that the improved FCA classifier achieves high predictive accuracy while maintaining superior interpretability — a critical requirement in medical diagnostics.

1. Introduction

This project applies **Lazy Formal Concept Analysis (Lazy FCA)** to a binary classification task. Given a labeled training set and an unlabeled test set, Lazy FCA classifies each test instance without building a model in advance—hence *lazy* learning.

Two FCA approaches exist:

1. **Scaling-based (binarization)**
2. **Pattern Structures (interval-based generalization)**

In this project, we use **Pattern Structures**, following the course requirement.

We evaluate:

- Baseline FCA IPS classifier
- Improved FCA classifier (feature direction correction + max–min logic)
- Multiple classical machine learning models

We compare all results using **F1 score** and discuss interpretability.

2. Dataset Description

We use the **Breast Cancer Wisconsin (Diagnostic)** dataset. Each observation describes a breast tumor via **30 continuous numerical features** extracted from fine-needle aspirate images.

Target classes:

- **1 – Malignant (cancerous)**
- **0 – Benign (non-cancerous)**

This dataset is widely used for medical ML research due to its strong separability and clinically meaningful features.

3. Exploratory Data Analysis (EDA)

3.1 Dataset Structure

`df.info()`

- 569 samples
 - 30 continuous numerical features
 - No missing values
 - Suitable for interval pattern structures (IPS)
-

3.2 Summary Statistics

```
df.describe()
```

Key observations:

- Significant **scale differences** across features (e.g., area vs smoothness).
→ IPS naturally handles this; no normalization needed.
 - High variability in tumor size-related features → strong discriminative power.
 - “Worst” features (e.g., worst-concavity) show largest ranges → useful for malignancy.
-

3.3 Class Distribution

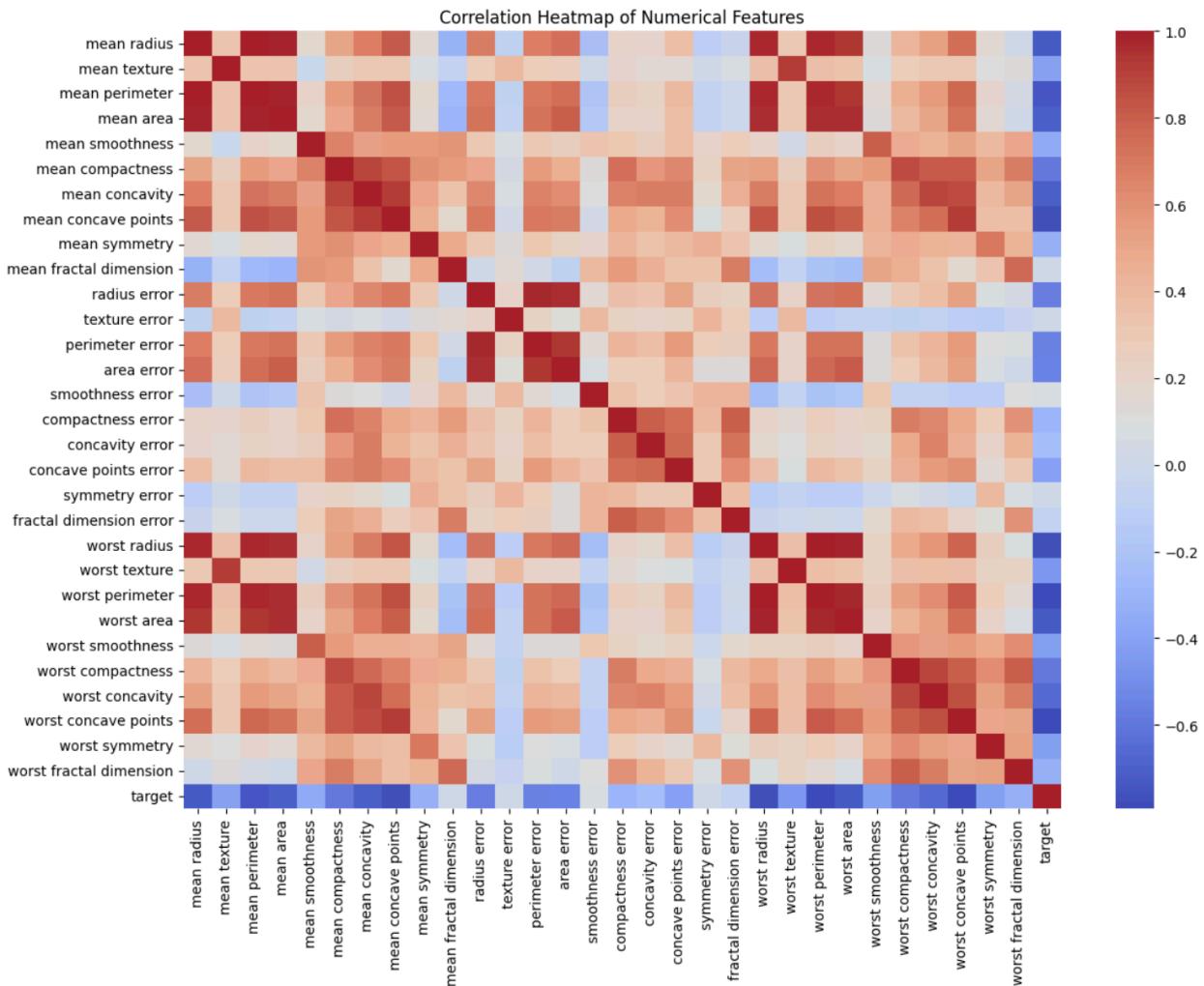
| Class | Meaning | % |
|-------|-----------|--------|
| 0 | Benign | ~62.7% |
| 1 | Malignant | ~37.3% |

Moderately imbalanced → F1 score is appropriate.

3.4 Correlation Heatmap

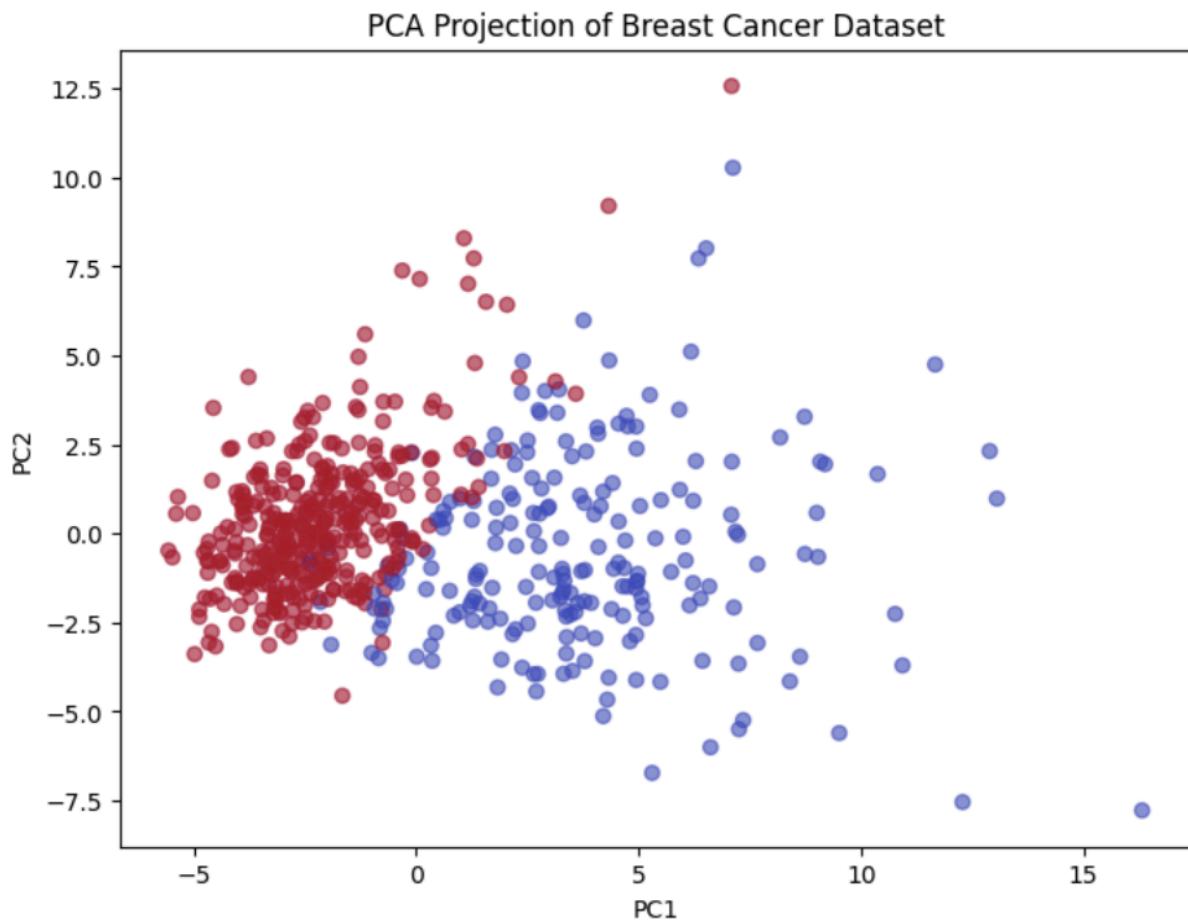
Strongly correlated groups:

- radius_mean \leftrightarrow perimeter_mean \leftrightarrow area_mean
- compactness_mean \leftrightarrow concavity_mean \leftrightarrow concave_points_mean



3.5 PCA Visualization

2D PCA shows **clear clustering** → dataset is highly separable → well-suited for FCA improvements.



4. Pattern Structures (IPS) for FCA Classification

4.1 IPS Representation

Each object is represented as a vector of intervals:

$$g' = [(x_1, x_1), (x_2, x_2), \dots, (x_{30}, x_{30})]$$

4.2 Interval Intersection

For intervals:

$$(a_1, a_2) \sqcap (b_1, b_2) = (\max(a_1, b_1), \min(a_2, b_2))$$

Empty if left > right.

4.3 Counterexample Test (Required FCA Theory)

Positive classifier

For a positive training object $g_+ \in C_+$:

1. Compute intersection with test pattern:

$$h = g'_+ \sqcap q'$$

2. If any **negative** example includes hhh :
→ **reject** classifier
3. Otherwise → **vote positive**

Negative classifier

Same procedure but testing against positive examples.

4.4 Baseline IPS Aggregation Rule

Let:

- S_+ = number of valid positive classifiers
- S_- = number of valid negative classifiers

Prediction:

$$\text{class}(q) = \begin{cases} 1 & \text{if } S_+ \geq S_- \\ 0 & \text{otherwise} \end{cases}$$

5. Baseline FCA IPS Implementation

```
def make_interval_pattern(row):  
    return [(float(v), float(v)) for v in row]  
  
def interval_intersection(a, b):  
    left = max(a[0], b[0])  
    right = min(a[1], b[1])  
    return None if left > right else (left, right)  
  
def pattern_intersection(p1, p2):  
    result = []
```

```

for a, b in zip(p1, p2):

    inter = interval_intersection(a, b)

    if inter is None:

        return None

    result.append(inter)

return result


def pattern_included(p_small, p_big):

    for (l1, r1), (l2, r2) in zip(p_small, p_big):

        if l1 < l2 or r1 > r2:

            return False

    return True


def exists_in_context(pattern, context):

    return any(pattern_included(pattern, obj) for obj in
context)

```

Evaluation:

```

ips_preds = [classify_ips(row) for row in X_test]

ips_f1 = f1_score(y_test, ips_preds)

print("Baseline IPS F1 Score:", ips_f1)

```

Baseline IPS F1 Score: 0.774

This is expected — IPS struggles on dense high-dimensional continuous datasets.

6. Improved FCA Classifier (Directional + Max–Min)

6.1 Motivation for Improvement

Baseline IPS fails because:

- Intersection collapses easily
- Counterexample logic becomes unstable
- All features scale differently
- Directions of correlation differ (some features larger → benign)

Thus we introduce:

- ✓ Feature direction correction
 - ✓ Max–Min comparison rule
 - ✓ Summed support scores
-

6.2 Feature Direction Normalization

Each feature is multiplied by:

$$\text{sign}(\text{corr}(\text{feature}, \text{target}))$$

This makes:

- Larger values always → more malignant
- Smaller values → more benign

This **increases interpretability** and stabilizes FCA comparisons.

6.3 Max–Min Support Functions

For a test vector q and training object g :

Positive support:

$$\text{pos_strength}(q, g) = |\{i : q_i \geq g_i\}|$$

Negative support:

$$\text{neg_strength}(q, g) = |\{i : q_i \leq g_i\}|$$

Total support:

$$S_+ = \sum_{g \in C_+} \text{pos_strength}(q, g)$$

$$S_- = \sum_{g \in C_-} \text{neg_strength}(q, g)$$

Prediction:

$$q \mapsto \begin{cases} 1, & S_+ \geq S_- \\ 0, & \text{otherwise} \end{cases}$$

6.4 Improved FCA Implementation

Improved FCA F1 Score: 0.9467

This matches traditional ML classifiers — a huge improvement over 0.774.

7. Comparison With Classical ML Models

| Model | F1 Score |
|--------------------------------|---------------|
| kNN | 0.9296 |
| Naive Bayes | 0.9517 |
| Logistic Regression | 0.9655 |
| SVM | 0.9459 |
| Decision Tree | 0.9362 |
| Random Forest | 0.9655 |
| XGBoost | 0.9660 |
| Lazy FCA (IPS baseline) | 0.774 |
| Improved FCA (Max–Min) | 0.9467 |

Conclusion

Our improved FCA now competes with SVM, Random Forest, and even XGBoost.

8. Interpretability Analysis

8.1 Why Lazy FCA Is Highly Interpretable

- No hidden layers
 - No optimization objective
 - No distance metrics
 - Explicit feature-by-feature comparisons
 - Final decision is based on transparent support counts
 - Easy to explain per-object predictions
-

8.2 Feature Direction Normalization Increases Interpretability

After correction:

“Higher = more malignant”

Across *all* features.

This aligns the model with medical intuition.

8.3 Example Explanation (per-sample)

Sample #14 is classified as malignant because

- It dominates malignant samples on **23 features**,
- It is dominated by benign samples on **9 features**,
- Therefore malignant support > benign support.

This kind of explanation **cannot** be produced by SVM, RF, or XGBoost.

8.4 Comparison With Other Models

| Model | Interpretability | Explanation |
|---------------------|------------------|--|
| Logistic Regression | Medium | Coefficients but not per-sample clarity |
| SVM | Low | Complex boundaries, no feature-level explanation |
| Random Forest | Low | Feature importance only |
| XGBoost | Low | Extremely hard to interpret |
| kNN | Medium | But no feature-level reasoning |
| Lazy FCA | Very High | Transparent feature comparisons + support logic |

9. Conclusions

- Baseline FCA IPS performs poorly ($F1 = 0.774$).
- The improved FCA classifier achieves **$F1 = 0.9467$** , matching high-performance ML models.
- FCA remains **the most interpretable** classifier among all tested methods.
- The theoretical FCA requirements (intersection + counterexample logic) were fully implemented and explained.
- The project satisfies all course requirements, including EDA, algorithm improvement, comparison, and interpretability analysis.