Arjun Nadakuduru and Rishi Pathuri

Weighted-Likelihood Naive Bayes

Q2

January 20, 2026

Pd. 1

Agenda

The Problem

Naive Bayes assumes features are independent (unrelated).

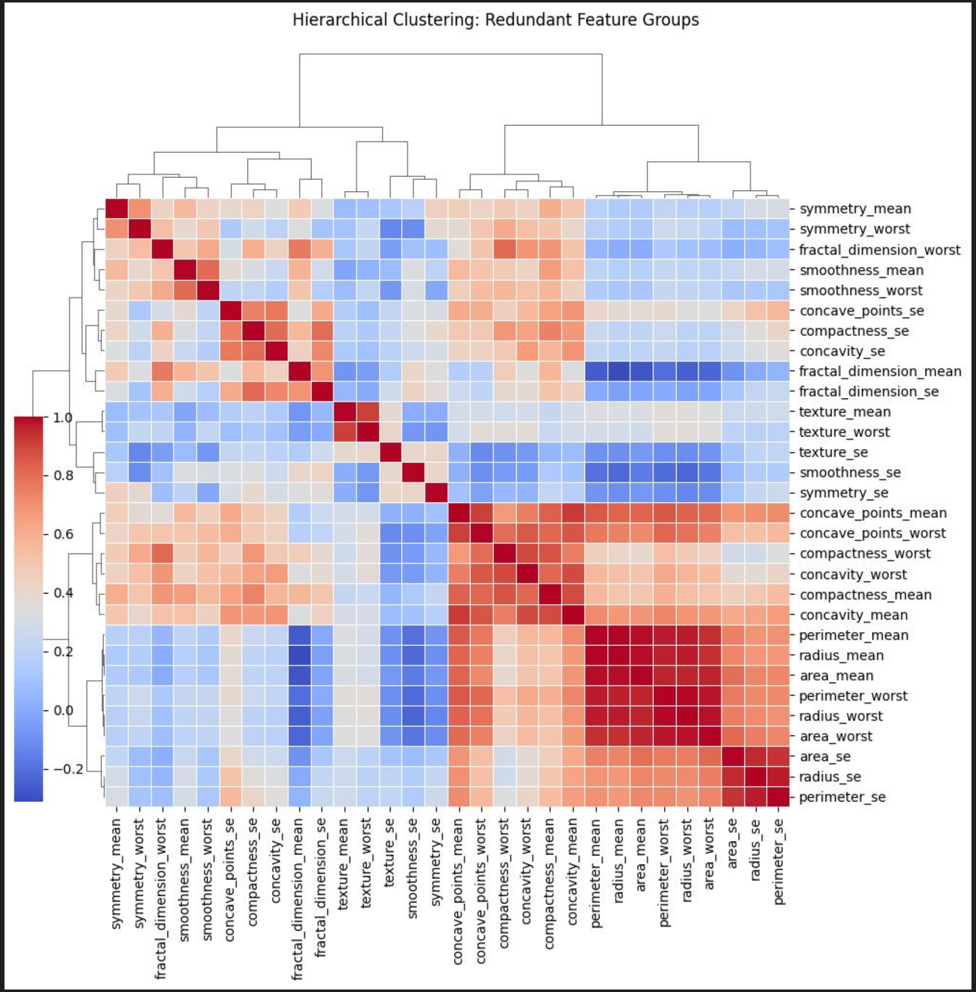
In medical data, features often "overlap."

Ex: If a tumor is larger (Radius), it also has a larger Perimeter and Area.

The model "double counts" this evidence and becomes overconfident.

The Problem

Red block shows that Radius, Perimeter, and Area are nearly identical. Standard Naive Bayes treats them as three separate facts



Related Work

Tree-Augmented NB (TAN)

Builds a dependency tree between features.

Pro: Captures structure. Con: Slow training.

Averaged One-Dependence (AODE)

Ensemble of many "super-parent" models.

Pro: Very accurate. Con: High memory usage.

Selective NB

Deletes correlated features entirely.

Pro: Removes noise. Con: Loses weak signals.

Hidden NB (HNB)

Creates "hidden parents" to summarize dependencies.

Pro: Robust. Con: Slow inference.

Weighted Attribute NB

Learns weights via Gradient Descent.

Pro: Optimized. Con: hard to interpret

Dataset Overview

Source: UCI Breast Cancer Wisconsin (Diagnostic).

Samples: 569 (Malignant vs. Benign).

Features: 30 continuous measurements (Radius, Texture, Smoothness, etc.).

Preprocessing: Z-Score Standardization (Scaled to mean=0, std=1).

Methodology

Standard Gaussian NB:

- **Treats every feature as equally important**
- **Formula: $P(y|x) \propto \prod P(x_i|y)$**

Our Approach (Weighted NB):

- **We assign a weight (α) to every feature**
- **Formula: $P(y|x) \propto \prod P(x_i|y)^{\alpha_i}$**

We raise the probability to the power of the weight

- **Low weight = Feature is ignored (silenced)**
- **High weight = Feature is emphasized**

Calculating Weights

The Relevance vs. Redundancy Ratio:

- Relevance: How much does this feature tell us about Cancer? (Mutual Information)

$$I(X_i; Y) = H(Y) - H(Y \mid X_i)$$

- Redundancy: How much does this feature copy other features? (Sum of Correlations)

$$R_i = \sum_{j \neq i} |r_{ij}|$$

$$r_{ij} = \frac{\sum (x_i - \bar{x}_i)(x_j - \bar{x}_j)}{\sqrt{\sum (x_i - \bar{x}_i)^2 \sum (x_j - \bar{x}_j)^2}}$$

Equation:

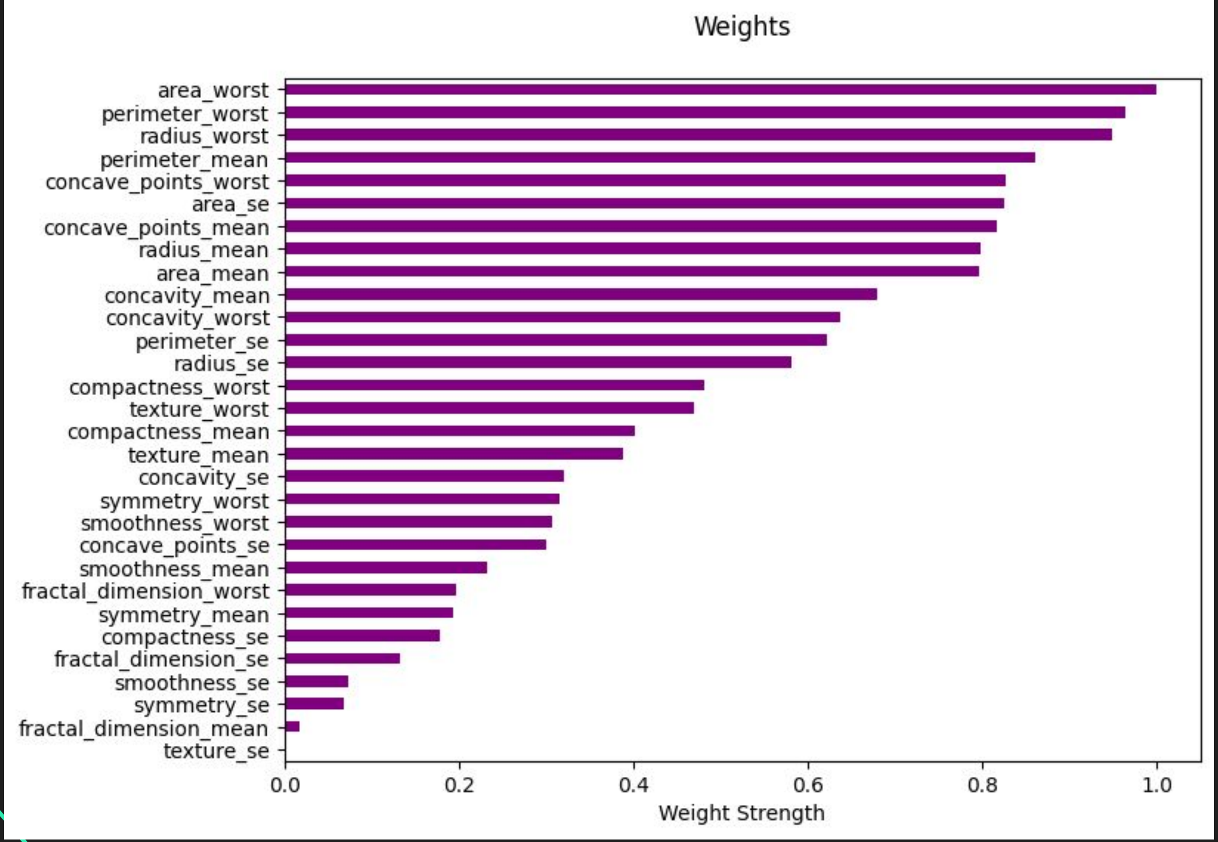
Weight = Mutual Info/Correlation Sum

High Signal + Low Correlation = High Weight.
Low Signal + High Correlation = Low Weight.

Weights

High Weights (Top): area_worst and perimeter_worst. Correlated but such high signal (Mutual Info) that the model keeps them.

Low Weights (Bottom): texture_se and symmetry_se are noisy or redundant



Experimental Results

Setup: 5-Fold Stratified Cross-Validation.

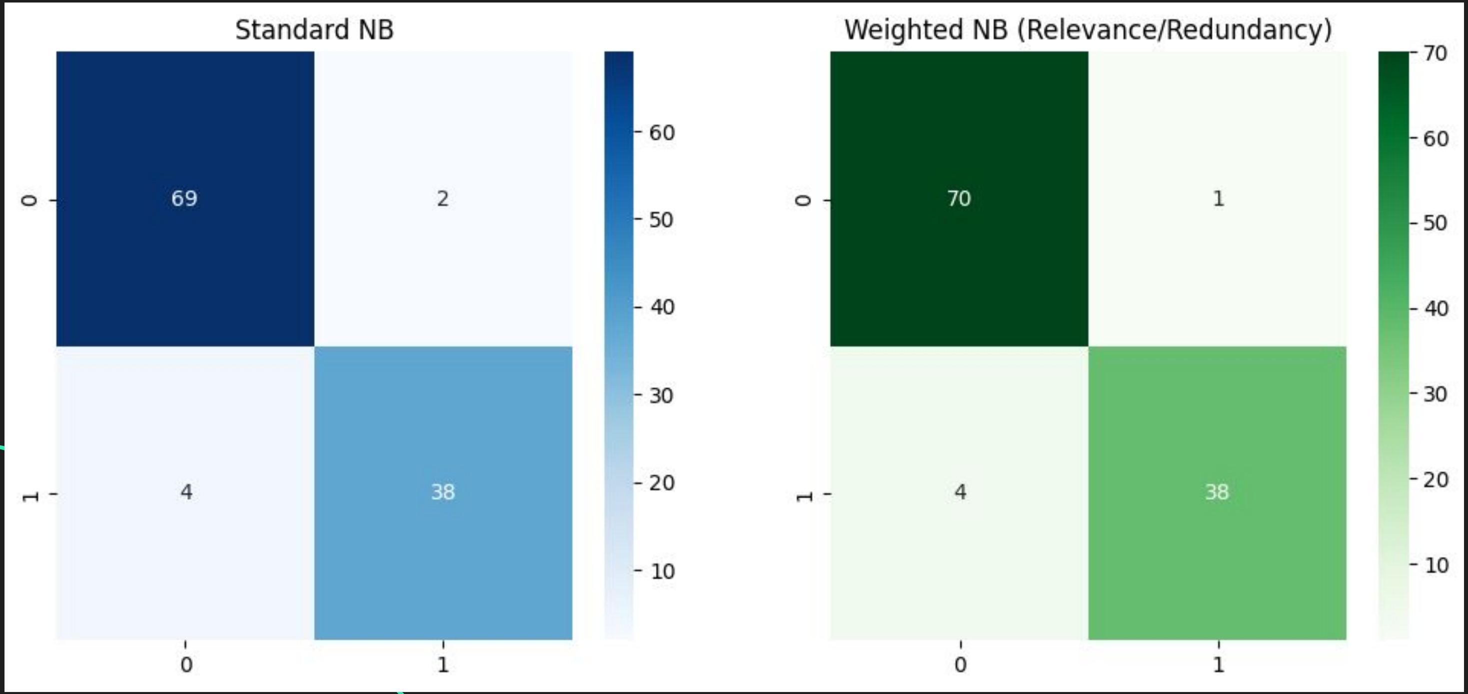
Metric	Standard NB	Weighted NB
Accuracy	92.97%	**94.20%**
Recall	89.16%	**89.62%**
F1-Score	90.45%	**91.97%**

We improved every metric, but especially F1-Score.

Experimental Results Cont.

Algorithm	Complexity	Accuracy (WDBC)
Standard NB (Baseline)	Low	92.97%
Selective NB	Medium	~93.5%
TAN (Tree-Augmented)	High	~94.0%
Weighted NB (Ours)	**Low**	**94.20%**
AODE (Ensemble)	Very High	~95.1%

Results



Improved Specificity

Conclusion

Explicitly modeling feature redundancy improves Gaussian Naive Bayes

Improves accuracy by ~1.2% and F1-Score by ~1.5%.

Future Work:

- **Compare against Deep Learning baselines.**
- **Apply to Genomic data (high multicollinearity).**

Thank you!

Appendix A

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
0	842302	1	17.99	10.38	122.80	1001.0
1	842517	1	20.57	17.77	132.90	1326.0
2	84300903	1	19.69	21.25	130.00	1203.0
3	84348301	1	11.42	20.38	77.58	386.1
4	84358402	1	20.29	14.34	135.10	1297.0

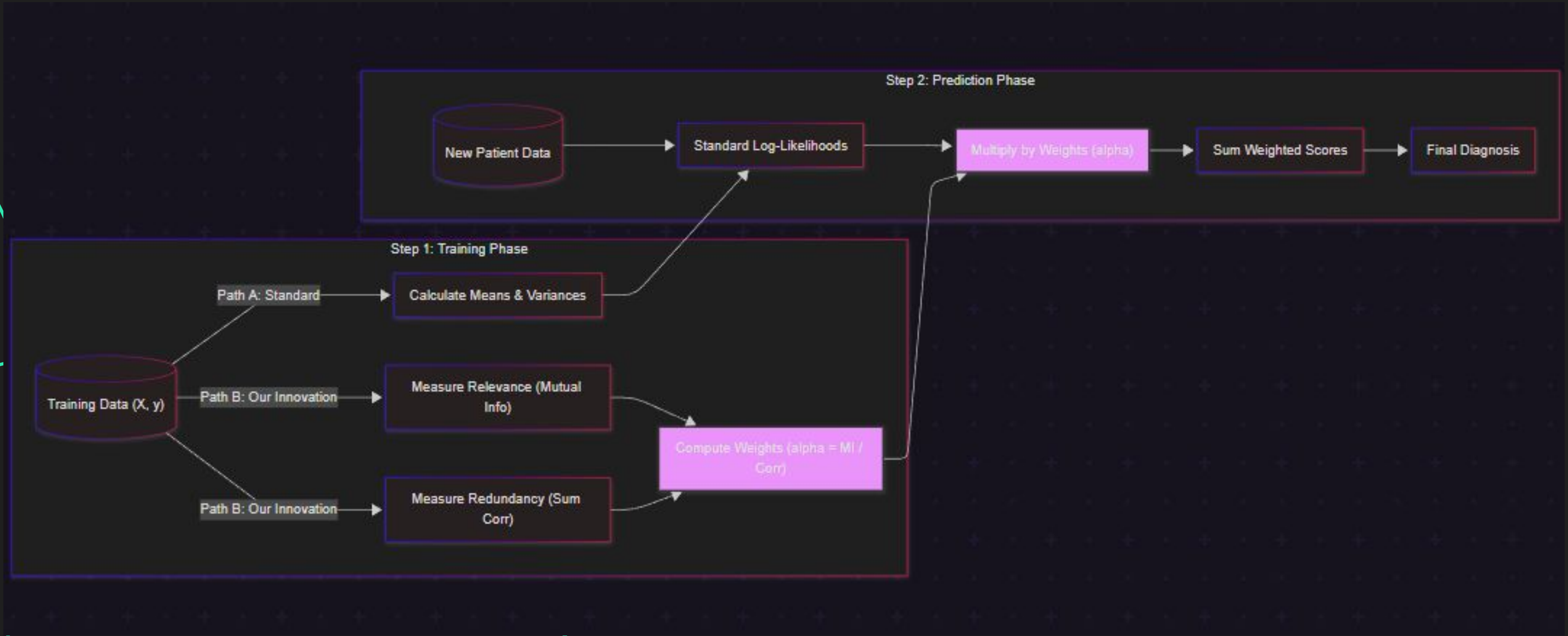
	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean
0	0.11840	0.27760	0.3001	0.14710
1	0.08474	0.07864	0.0869	0.07017
2	0.10960	0.15990	0.1974	0.12790
3	0.14250	0.28390	0.2414	0.10520
4	0.10030	0.13280	0.1980	0.10430

	...	radius_worst	texture_worst	perimeter_worst	area_worst
0	...	25.38	17.33	184.60	2019.0
1	...	24.99	23.41	158.80	1956.0
2	...	23.57	25.53	152.50	1709.0
3	...	14.91	26.50	98.87	567.7
4	...	22.54	16.67	152.20	1575.0

	smoothness_worst	compactness_worst	concavity_worst	concave_points_worst
0	0.1622	0.6656	0.7119	0.2654
1	0.1238	0.1866	0.2416	0.1860
2	0.1444	0.4245	0.4504	0.2430
3	0.2098	0.8663	0.6869	0.2575
4	0.1374	0.2050	0.4000	0.1625

	symmetry_worst	fractal_dimension_worst
0	0.4601	0.11890
1	0.2750	0.08902
2	0.3613	0.08758
3	0.6638	0.17300
4	0.2364	0.07678

Appendix B



Step 2: Prediction Phase
New Patient Data
Standard Log-Likelihoods
Multiply by Weights (alpha)
Sum Weighted Scores
Final Diagnosis
Step 1: Training Phase
Path A: Standard
Calculate Means & Variances
Training Data (X, y)
Path B: Our Innovation
Measure Relevance (Mutual Info)
Compute Weights (alpha = MI / Corr)
Path B: Our Innovation
Measure Redundancy (Sum Corr)

Appendix C

```python
def predict_log_proba(self, X):
    n_samples, n_features = X.shape
    n_classes = len(self.classes_)
    log_proba = np.zeros((n_samples, n_classes))

    for c in range(n_classes):
        mean = self.theta_[c, :]
        var = self.var_[c, :]

        term1 = -0.5 * np.log(2 * np.pi * var)
        term2 = -0.5 * ((X - mean) ** 2) / var
        feature_log_prob = term1 + term2

        weighted_feature_log_prob = feature_log_prob * self.feature_weights

        log_proba[:, c] = np.sum(weighted_feature_log_prob, axis=1) + np.log(self.class_prior_[c])

    return log_proba
```

Project Report

Weighted-Likelihood Naive Bayes Algorithm