GAUSSIAN NAÏVE BAYES

Leah Hoogstra, Presley Kimball, Malindi Whyte

the calculated collective

Outline

1. Representation
2. Loss and Optimizer
3. Algorithm
4. Results of Testing
5. Summary

Representation

Classification of Data with Continuous features

$$\mathcal{X} \in \mathbb{R}^d \quad \Longrightarrow \quad Y = \{y_1, ..., y_n\}, y_i \in \mathbb{Z}$$

Building our classifier

$$h_{\mathrm{Bayes}}(x) = \arg\max_y P[Y = y | \mathbf{X} = \mathbf{x}]$$

$$= \arg\max_y \frac{P[Y = y]\boxed{P[\mathbf{X} = \mathbf{x} | Y = y]}}{P[\mathbf{X} = \mathbf{x}]}$$

Naïve Assumption

The features conditioned on the label are independent

$$P[\mathbf{X} = \mathbf{x} | Y = y] = \prod_{j=1}^{d} P[X_j = x_j | Y = y]$$

Assumption on our Feature space

$$X_j | Y = y \sim N(\mu_{j,y}, \sigma^2_{j,y})$$

Mean of feature j of samples of class y

Variance of feature j of samples of class y

Independence assumption + PDF of Gaussian + Algebra

$$P[\mathbf{X} = \mathbf{x} | Y = y] = \frac{1}{(2\pi)^{\frac{d}{2}} \left(\det \Sigma_y\right)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(x - \mu_y\right)^T \Sigma_y^{-1} \left(x - \mu_y\right)\right)$$

$$\Sigma_y = \begin{pmatrix} \sigma_{1,y}^2 & 0 & \cdots & \cdots & 0 \\ 0 & \sigma_{2,y}^2 & 0 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & \sigma_{d-1,y}^2 & 0 \\ 0 & \cdots & \cdots & 0 & \sigma_{d,y}^2 \end{pmatrix}$$

Loss and Optimizer

Motivation

Recall

$$X_j|Y = y \sim N(\mu_{j,y}, \sigma_{j,y}^2)$$

How do we choose $\mu_{j,y}$, ${\sigma_{j,y}}^2$ so that our model has the most optimal representation?

Maximum Likelihood Estimation!!

Notation

We let S denote the training data and $S_y \subseteq S$ be the training data of some class y.

- Let m be the number of examples in S_y such that $S_y = (x_1, x_2, \ldots, x_m)$.
- We assume each example (or the set of $\{x_i\}_{i=1:m}$) is i.i.d.
- We assume each example was sampled from P_{θ_y}

We are interested in estimating P_{θ_y} such that we maximize the likelihood of observing the given sample.

Likelihood

The likelihood of our sample is

$$l(S_y; \theta_y) = \prod_{i=1}^{m} P_{\theta_y}(x_i).$$

The log-likelihood of our sample is

$$L(S_y; \theta_y) = \sum_{i=1}^{m} \log(P_{\theta_y}(x_i)).$$

Likelihood (continued)

By independence we have that

$$P_{\theta_y}(x_i) = \prod_{j=1}^{d} P_{\theta_{j,y}}(x_i)$$

where $P_{\theta_{j,y}}(x_i)$ is the probability of observing the jth feature in x_i conditioned on class y. Then

$$L(S_y; \theta_y) = \sum_{j=1}^{d} \sum_{i=1}^{m} \log(P_{\theta_{j,y}}(x_i)) = \sum_{j=1}^{d} L(S_y; \theta_{j,y})$$

Likelihood of class y sample w.r.t. all feature parameters

Likelihood of class y sample w.r.t. the jth feature's parameters

We now want to estimate maximize $L(S_y; \theta_{j,y})$ for each feature.

Maximum Likelihood Estimation (notation)

By assumption of the model, for the jth entry of x_i

$$X_{i,j} \sim N(\mu_{j,y}, {\sigma_{j,y}}^2)$$

For some $\mu_{j,y}$, $\sigma_{j,y}$. In other words, we assume $P_{\theta_{j,y}} = N(\mu_{j,y}, \sigma_{j,y})$ which implies $\theta_y = (\mu_y, \sigma_y)$ where

$$\mu_y = (\mu_{1,y}, \mu_{2,y}, \dots, \mu_{d,y})$$

$$\sigma_y = (\sigma_{1,y}, \sigma_{2,y}, \dots, \sigma_{d,y}).$$

Equivalently,

$$P_{\theta_{j,y}}(x_{i,j}) = \frac{1}{\sigma_{j,y}\sqrt{2\pi}} exp\left\{\frac{-(x_{i,j}-\mu_{j,y})^2}{2{\sigma_{j,y}}^2}\right\}.$$

Maximum Likelihood Estimation

We can then write the likelihood as

$$
\begin{aligned}
L\left(S_y; \theta_{j,y}\right) &= \sum_{i=1}^{m} \log(P_{\theta_{j,y}}(x_i)) \\
&= \sum_{i=1}^{m} \log\left(\frac{1}{\sigma_{j,y}\sqrt{2\pi}} exp\left\{\frac{-(x_{i,j} - \mu_{j,y})^2}{2{\sigma_{j,y}}^2}\right\}\right) \\
&= -m \log\left(\sigma_{j,y}\sqrt{2\pi}\right) - \frac{1}{2{\sigma_{j,y}}^2} \sum_{i=1}^{m} (x_{i,j} - \mu_{j,y})^2
\end{aligned}
$$

Maximum Likelihood Estimation

Taking derivatives w.r.t. $\mu_{j,y}$, $\sigma_{j,y}$,

$$\frac{d}{d\mu_{j,y}} L(S_y; \theta_{j,y}) = \frac{1}{{\sigma_{j,y}}^2} \sum_{i=1}^{m} (x_{i,j} - \mu_{j,y})$$

$$\frac{d}{d\sigma_{j,y}} L(S_y; \theta_{j,y}) = -\frac{m}{\sigma_{j,y}} + \frac{1}{{\sigma_{j,y}}^3} \sum_{i=1}^{m} (x_{i,j} - \mu_{j,y})^2$$

Setting these to zero and solving for each parameter, we have

$$\boxed{\begin{aligned} \mu_{j,y} &= \frac{1}{m} \sum_{i=1}^{m} x_{i,j} \\ \sigma_{j,y} &= \sqrt{\frac{1}{m} \sum_{i=1}^{m} (x_{i,j} - \mu_{j,y})^2} \end{aligned}}$$

Maximum Likelihood Estimators!!

Maximum Likelihood Estimation (result)

In conclusion, to maximize $L\left(S_y;\theta_y\right)$ we maximize $L\left(S_y;\theta_{j,y}\right)$ for each feature.

We found the parameters $\theta_{j,y}=(\mu_{j,y},\sigma_{j,y})$ which maximize $L\left(S_y;\theta_{j,y}\right)$ are the empirical mean and variance:

$$\mu_{j,y}=\frac{1}{m}\sum_{i=1}^{m}x_{i,j}$$

$$\sigma_{j,y}=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_{i,j}-\mu_{j,y})^2}$$

Loss

We use loss as the negative of the log-likelihood

$$\ell(\theta, x) = -\log(P_\theta(x))$$

for a given example x.

Thus, we want to minimize the loss on ***<u>all</u>*** m examples in our training data

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{m} \ell(\theta, x_i) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log(P_\theta(x_i))$$

Optimizing Loss

Then

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{m} \ell(\theta, x_i) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log(P_\theta(x_i)) = \sum_{y \in Y} \underset{\theta_y}{\operatorname{argmax}} \sum_{i=1}^{m} 1_{y_i = y} \log(P_{\theta_y}(x_i))$$

$$= \sum_{y \in Y} \underset{\theta_y}{\operatorname{argmax}} L(S_y; \theta_y)$$

(due to definition of $L(S_y; \theta_y)$ and S_y).

We have shown $L(S_y; \theta_y)$ is maximized using $\theta_y = (\mu_y, \sigma_y)$. Thus, we minimize the loss for the same choice of θ.

The Algorithms

Train algorithm

INPUT: X_train, Y_train

For $y \in Y$: *(using get_params)*

X_y = X_train[Y_train=y] *(subset data for class y)*

For $j = 1{:}\,d$:

$\mu_{j,y}$ = mean(X_y[j,:])

$\sigma^2_{j,y}$ = variance(X_y[j,:])

Prior[y] = $\frac{\text{|Y\_train==y|}}{\text{|Y\_train|}}$

RETURN (stored): μ, σ^2, Prior

Predict Algorithm

INPUT: X_test

For x ∈ X_test:

For $y \in Y$:

$$P[X = x|Y = y] = \frac{1}{(2\pi)^{\frac{d}{2}} \det(\Sigma_y)^{\frac{1}{2}}} exp\left\{\frac{-1}{2}\left(x - \mu_y\right)^T \Sigma^{-1}(x - \mu)\right\}$$

$P[Y = y|X = x]$_proxy = Prior[y] $P[X = x|Y = y]$

Prediction[x] = $\underset{y \epsilon Y}{\text{argmax}}\, P[Y = y|X = x]$_proxy

RETURN (stored): Prediction

Results of Testing

Data

UC Irvine Machine Learning Repository

Datasets Contribute Dataset About Us

Wine

Donated on 6/30/1991

Using chemical analysis to determine the origin of wines

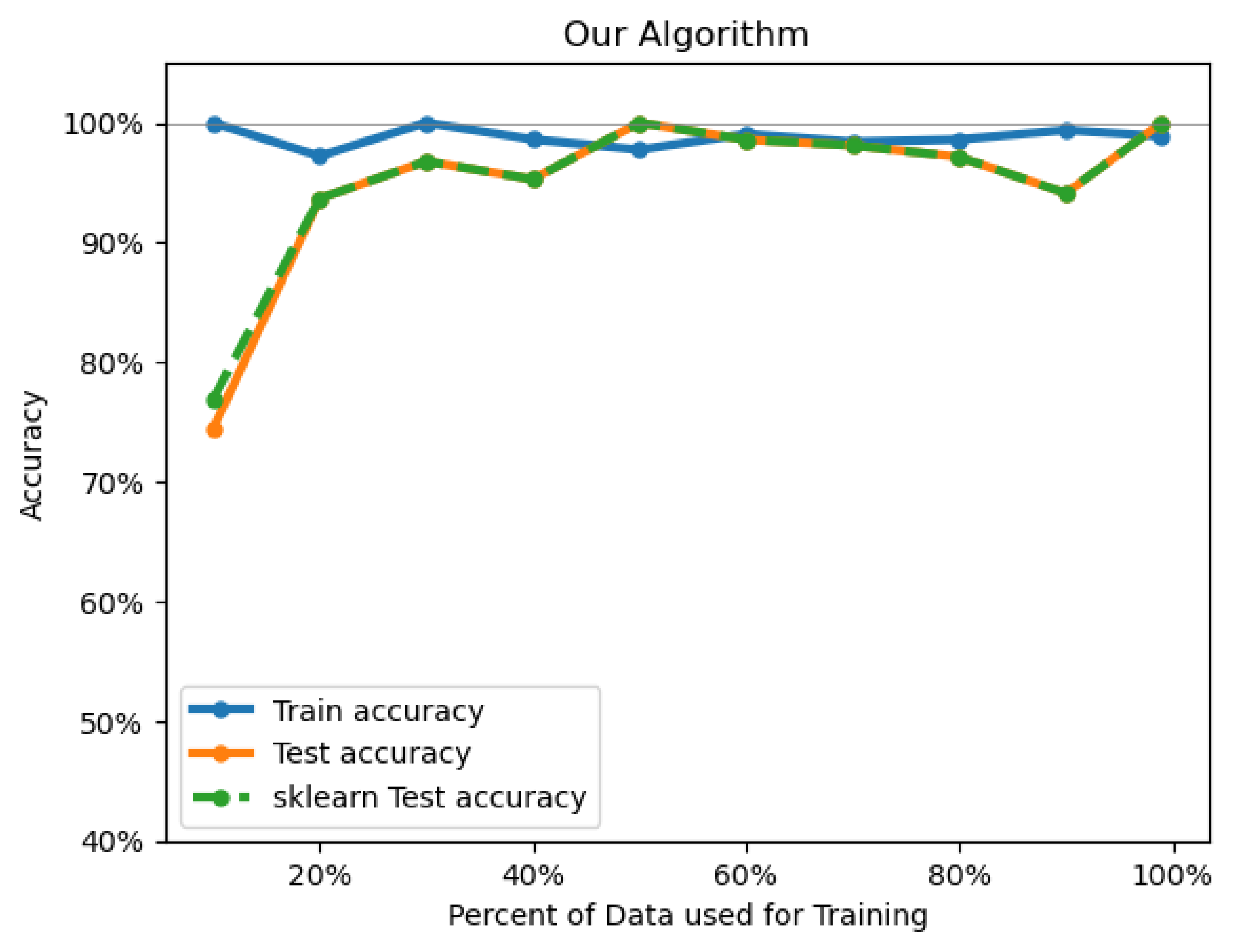
Dataset Characteristics	Subject Area	Associated Tasks
Tabular	Physics and Chemistry	Classification
Feature Type	**# Instances**	**# Features**
Integer, Real	178	13

Reproduce scikitlearn’s GuassianNB

The Calculated Collective

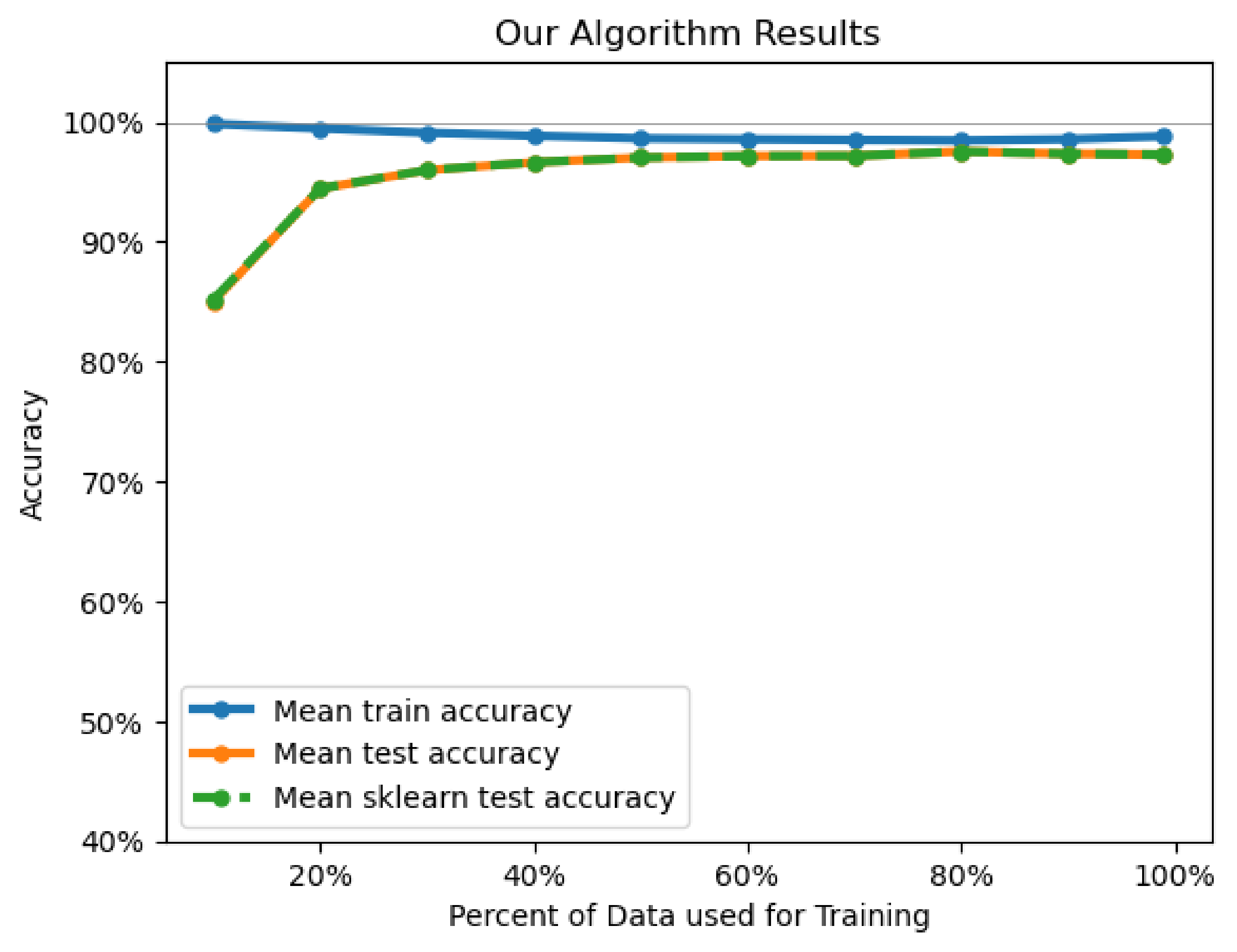
vs.

Accuracy: a first look



- On our initial test with an 80/20 train-test split, we not only matched scikitlearn’s test accuracy, we matched **every single case** of scikitlearn’s output
- When we varied the split ratio, our output still matched exactly
  - Except when using only 10% of data for training, where we mismatched 1 case

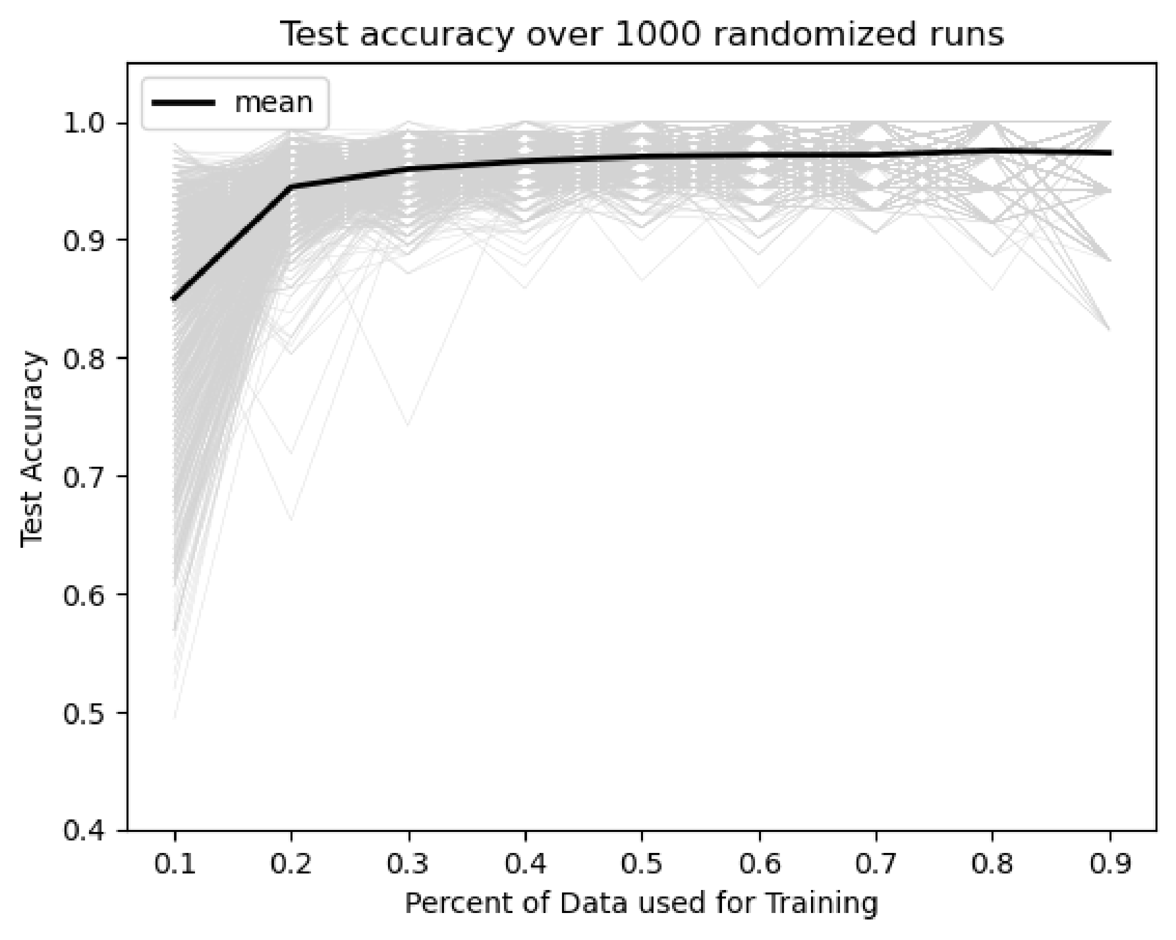
Accuracy statistics



- We reran our tests 1000 times with different randomized training and test data.
- Our results matched scikitlearn's algorithm exactly in **96.7%** of runs

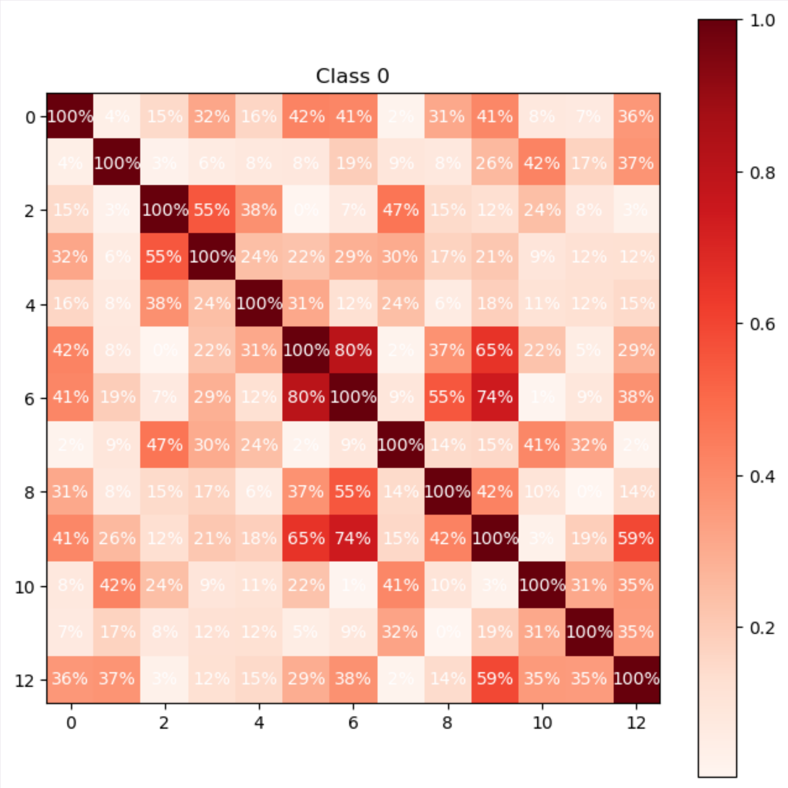
# cases mismatched with sklearn per run (1000 runs with 9 splits each)	
Mean	0.05
Median	0
Max	8

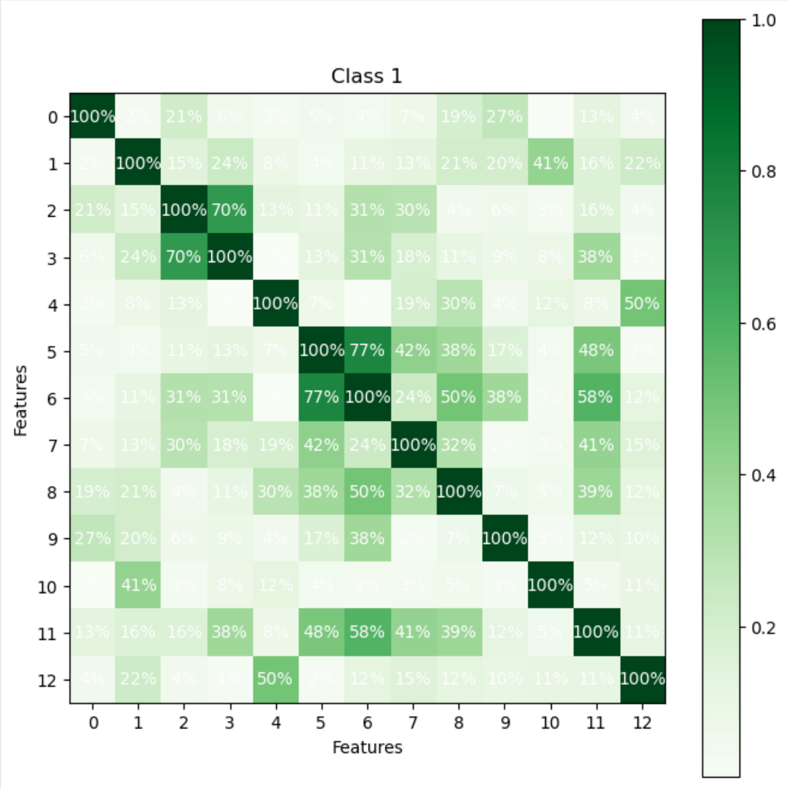
Accuracy statistics

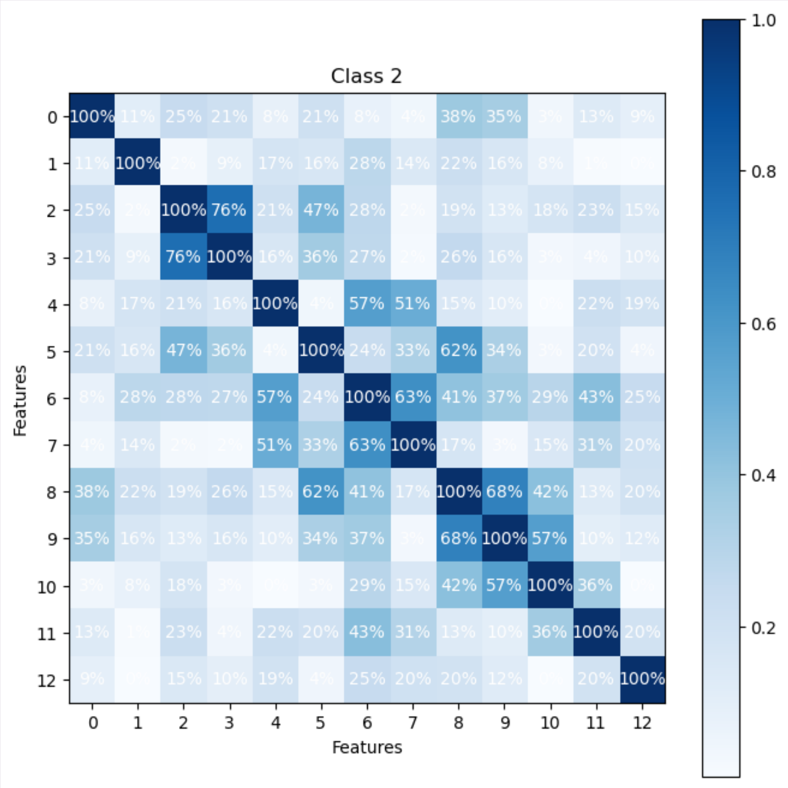


- The percent of training data used did impact the overall accuracy, but we found a good fit using anywhere from 40% to 80% of the training data.
- There was risk of overfitting when using 90% or more of the training data.

Validity of Naïve Assumption



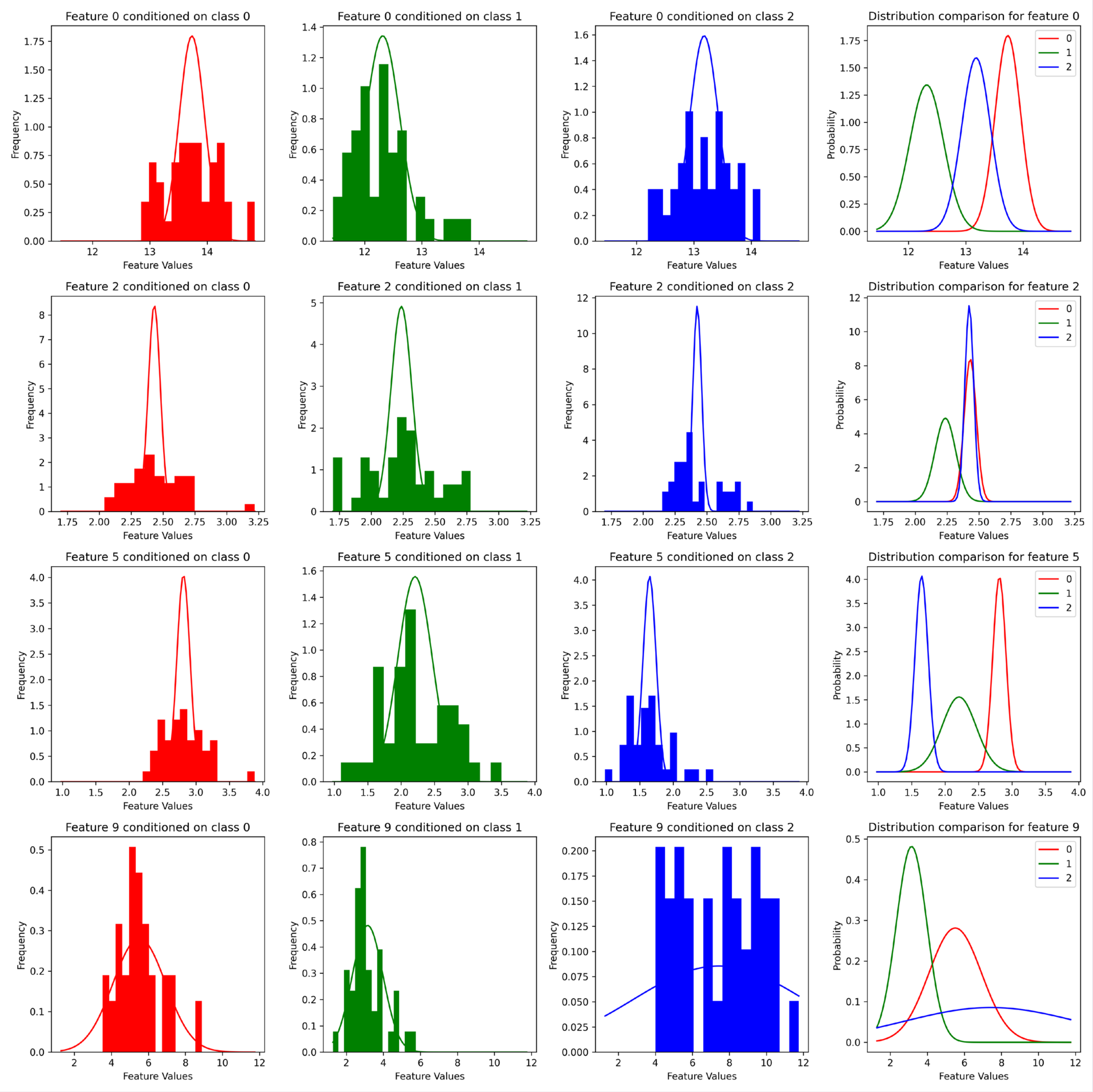




Correlation coefficients between features for each class

Validity of Normal Assumption

- Not all features are actually normally distributed
- We do see different distributions across classes, so our model still makes sense even if this assumption isn't always accurate



Summary

Who won?

The Calculated Collective

vs.

(pro since 2010)

AVG TEST ACCURACY

CC	SKL
95.56%	95.58%

Closing Notes

- Challenges:
  - Textbook calculations are given for a single class, we had to derive the loss function for all classes and make it work for an arbitrary number of classes
  - Coming up with unit tests to capture several edge cases
- Interesting:
  - Our algorithm was quite robust to the train/test split percentage chosen
  - Exciting to be able to match scikitlearn line for line 97% of the time

Sources

- Aeberhard, S., Coomans, D., and de Vel, O. (1994). 'The performance of statistical pattern recognition methods in high dimensional settings,' **Proc. IEEE Signal Process. Workshop Higher Order Statist** (pp. 14-16).
- Shalev-Shwartz, S., and Ben-David, S. (2014). **Understanding machine learning: From theory to algorithms.** 1st edn. New York: Cambridge university press.
- scikit-learn developers. (2024) **load_wine** [Online]. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.load_wine.html (Accessed: 4 December 2024)
- scikit-learn developers. (2024) **GaussianNB** [Online]. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html (Accessed: 4 December 2024)